Context-Aware Credit Card Fraud Detection

Dissertation in "Cotutelle"
mit
Institut National des Sciences Appliquées de Lyon

eingereicht an der

Fakultät für Informatik und Mathematik
Universität Passau

Johannes Jurgovsky

September, 2019

To my nephew Leonhard.

Abstract

Credit card fraud has emerged as major problem in the electronic payment sector. In this thesis, we study data-driven fraud detection and address several of its intricate challenges by means of machine learning methods with the goal to identify fraudulent transactions that have been issued illegitimately on behalf of the rightful card owner. In particular, we explore several means to leverage contextual information beyond a transaction's basic attributes on the transaction level, sequence level and user level.

On the transaction level, we aim to identify fraudulent transactions which, in terms of their attribute values, are globally distinguishable from genuine transactions. We provide an empirical study of the influence of class imbalance and forecasting horizons on the classification performance of a random forest classifier. We augment transactions with additional features extracted from external knowledge sources and show that external information about countries and calendar events improves classification performance most noticeably on card-not-present transaction.

On the sequence level, we aim to detect frauds that are inconspicuous in the background of all transactions but peculiar with respect to the short-term sequence they appear in. We use a Long Short-term Memory network (LSTM) for modeling the sequential succession of transactions. Our results suggest that LSTM-based modeling is a promising strategy for characterizing sequences of card-present transactions but it is not adequate for card-not-present transactions.

On the user level, we elaborate on feature aggregations and propose a flexible concept allowing us to define numerous features by means of a simple syntax. We provide a CUDA-based implementation for the computationally expensive extraction with a speed-up of two orders of magnitude compared to a single core CPU-based implementation. Our feature selection study reveals that aggregates extracted from users' transaction sequences are more useful than those extracted from merchant sequences. Moreover, we discover multiple sets of candidate features with equivalent performance as manually engineered aggregates while being vastly different in terms of their structure.

Regarding future work, we motivate the usage of simple and transparent machine learning methods for credit card fraud detection and we sketch a simple user-focused modeling approach.

Résumé

La fraude par carte de crédit est devenue un problème majeur dans le secteur des paiements électroniques. Dans cette thèse, nous étudions la détection de fraude basée sur les données transactionnelles et abordons plusieurs de ces défis complexes en utilisant des méthodes d'apprentissage automatique visant à identifier les transactions frauduleuses qui ont été émises illégitimement au nom du titulaire légitime de la carte. En particulier, nous explorons plusieurs moyens d'exploiter les informations contextuelles au-delà des attributs de base d'une transaction, notamment au niveau de la transaction, au niveau de la séquence et au niveau de l'utilisateur.

Au niveau des transactions, nous cherchons à identifier les transactions frauduleuses qui présentent des caractéristiques distinctes des transactions authentiques. Nous avons mené une étude empirique de l'influence du déséquilibre des classes et des horizons de prévision sur la performance d d'un classifieur de type random forest. Nous augmentons les transactions avec des attributs supplémentaires extraits de sources de connaissances externes et montrons que des informations sur les pays et les événements du calendrier améliorent les performances de classification, particulièrement pour les transactions ayant lieu sur le Web.

Au niveau de la séquence, nous cherchons à détecter les fraudes qui sont difficiles à identifier en elles-mêmes, mais particulières en ce qui concerne la séquence à court terme dans laquelle elles apparaissent. Nous utilisons un réseau de neurone récurrent (LSTM) pour modéliser la séquence de transactions. Nos résultats suggèrent que la modélisation basée sur des LSTM est une stratégie prometteuse pour caractériser des séquences de transactions ayant lieu en face à face, mais elle n'est pas adéquate pour les transactions ayant lieu sur le Web.

Au niveau de l'utilisateur, nous travaillons sur une stratégie existante d'agrégation d'attributs et proposons un concept flexible nous permettant de calculer de nombreux attributs au moyen d'une syntaxe simple. Nous fournissons une implémentation basée sur CUDA pour pour accélerer le temps de calcul de deux ordres de grandeur. Notre étude de sélection des attributs révèle que les agrégats extraits de séquences de transactions des utilisateurs sont plus utiles que ceux extraits des séquences de marchands. De plus, nous découvrons plusieurs ensembles d'attributs candidats avec des performances équivalentes à celles des agrégats fabriqués manuellement tout en étant très différents en termes de structure.

En ce qui concerne les travaux futurs, nous évoquons des méthodes d'apprentissage artificiel simples et transparentes pour la détection des fraudes par carte de crédit et nous esquissons une modélisation simple axée sur l'utilisateur.

Scientific Environment and Notation

The author has carried out the research reported in this dissertation at the Chair of Data Science, Department of Computer Science and Mathematics, University of Passau, Germany, and at the research group for Distributed Systems, Information Retrieval and Mobility (DRIM), Laboratory of Computer Science for Image and Information Systems (LIRIS), INSA Lyon, France.

This research has received support from *Worldline* within the scope of an industrial research project among Worldline, the University of Passau, INSA Lyon and the University of Milano. Additionally, this research has received support from the *Internetkompetenzzentrum Ostbayern* within the research project Big Open Data Analytics (BODA).

Notation

x	Scalar
$\mathbf{x}$	Vector
X	Matrix
$\mathcal{X}$	Set

$p(\cdot)$ We use $p(\cdot)$ to loosely refer to probabilities. In case of discrete random variables, $p(\cdot)$ denotes a probability mass function and in case of continuous variables, $p(\cdot)$ denotes a probability density function.

Terminology

- **Transaction types**: We denote card-present transactions by *face-to-face* or *offline* interchangeably and card-not-present transactions by *e-commerce* or *online*.
- **Data record**: We denote a single data record, e.g. one transaction, generically as *record*, *observation*, *instance* or *example*.
- **Classifier & Model**: If not stated otherwise, a *classifier* refers to the set of algorithms used to estimate parameters from data, perform inference on variables and convert the inferred values into classifications about data records. In machine learning literature the term *model* is considerably overloaded. We use the term *model* to refer to the abstract representation of the data which we obtain after estimating parameters with a classifier under a specific hyper-parameter configuration.

Contents

List of Figures

Chapter 1

Introduction

Which illustration is printed on the back of a 20€ note? - A doorway, a bridge or a gothic arc?[1] After searching our mental image archive for several seconds we are probably able to come up with a reasonable guess. Yet, the level of uncertainty about one's guess might be unexpectedly high. Given how often and regularly these slips of paper pass through our hands whenever we withdraw money from an ATM, pay for the delicious meal in our favorite restaurant or finally purchase that new pair of shoes, we are surprisingly unaware of details of the legal tender itself. On second thoughts, however, it should not surprise us too much that we do not pay close attention to these details. If it would be common to encounter counterfeit money in our daily lives, we would probably double-check the change very carefully every time we purchase something to protect ourselves from getting scammed. Therefore, our lack of knowledge about the 20€ note is not so much a loose oddity as it is a consequence of the trust we have in our society and the correct functioning of governmental authorities.

With cash we are in direct contact with the legal tender and the entity we exchange money with. An individual can, in principle, spot any irregularity that might occur during the act of payment immediately and personally, allowing him to take appropriate measures to prevent financial loss. Cash is not the only legal tender we have at our disposal. Traditional payment systems also offered negotiable instruments such as cheques or promissory notes as means to settle financial transactions through the transfer of monetary value. A payment system supersedes the direct exchange of cash in order to facilitate and secure international transactions. As parts of any payment system, banks and financial institutions manage accounts of individuals or legal entities and offer transactions between these accounts as their major service. In the second half of the 20th century, with the advent of electronic communication several different payment systems, each with its own standards and protocols, have emerged over the years. A payment from one bank account to another is realized via an electronic transmission of transaction information between the two banks without direct intervention of bank employees. Examples of such modern payment systems are: credit card banking, debit card banking or internet banking. The legal tender in these systems is a virtual object - the digital representation of transaction information. The plastic cards in our wallets are

[1] The 20€ note displays a gothic bridge. Due to political reasons, it is not an illustration of a real monument but a hypothetical example of the gothic architectural style in Europe between the 13th and 14th century.

not legal tenders per se but technical tools with the purpose to simplify authentication against the issuing bank. Unless there are obstructive reasons such as e.g. insufficient account balance or insufficient credit[2], the issuing bank authorizes the payment and exchanges information with the acquiring bank via an electronic transaction to finally settle its debt.

In traditional payment systems, the integrity of the legal tender was guaranteed through governmental authorities endowing the medium with apparent security features, and the authentication of the debtor was conducted through personal contact with either bank employees or the recipient himself. Now in electronic payment systems, the legal tender is an abstract object: the exchange of information between banks through secure protocols. And the personal authentication against the issuing bank has been replaced with an exchange of information between the client, or rather the identifying data in his exclusive possession, and his bank through secure protocols. The technological development in the banking economy has effectively cleared the transaction process from human intervention enabling an acceleration of funds exchange for billions of people on a global scale and yet being easy to use in our daily lives. However, the individual now has very little control and even less insight into the transaction process. Instead we are left to trust in the integrity of banks and financial institutes, trust in the reliability of IT infrastructure and trust in the strength of cryptographic techniques.

As businesses continue to provide parts or the entirety of their services via the internet, money is increasingly transacted electronically in an ever-growing cashless banking economy. Even in Germany, a country whose citizens are known to favor cash over electronic payments, 2018 was the first year the volume of cashless payments exceeded cash based payments as found by the EHI Retail Institute[3]. In this environment, credit card fraud has become a key concern for modern banking systems. The financial strain from credit card fraud is turning into a substantial challenge for financial institutions and service providers, forcing them to continuously adapt and improve their fraud prevention and detection systems. Their efforts are not limited to mitigating direct losses incurred by fraudulent transactions, but also to ensure that legitimate customers are not adversely impacted by automated and manual reviews. In the payment industry, fraud occurs when someone steals information from your card to make purchases without your permission and the detection of these fraudulent transactions has become a crucial activity for payment processors.

A fraud detection system detects and recognizes fraudulent activities as they enter the system and reports them to a system administrator. With the widespread usage of electronic payment systems, businesses face great challenges to seek out the rare but costly malicious behavior from the vast stream of transactions that is consistently generated by millions of customers from all over the world. Fraudulent acts can be committed by various groups or individuals such as organized crime, hackers or information buyers to achieve financial gain on false ground by illegal means. Fraudsters may employ a wide variety of strategies and technical tools to gather personal information about card

[2]The attentive reader might already guess a third obstructive reason.

[3]`https://www.welt.de/wirtschaft/article193063435/Zahlungsmittel-Karte-schlaegt-in-Deutschland-erstmals-Bargeld.html`, Last access: 09.05.2019

holders and their credit card details: Key-loggers, sniffers, site-cloning, false merchant sites, physical card theft or even generating artificial card information [Akh13].

A typical fraud detection systems is composed of an *automatic tool* and a *manual process*. The *automatic tool* is based on fraud detection rules. It analyzes all the new incoming transactions and assigns them fraud scores. The *manual process* comprises the expertise and effort of fraud investigators. They focus on transactions with high fraudulent scores and provide feedback on analyzed transactions to update and improve the automatic tool. Fraud detection systems can be based on *expert* driven rules, *data* driven rules or a combination of both. Expert driven rules aim to identify specific scenarios of fraud which have been discovered previously by fraud investigators. An example of a fraud scenario could be: "*A cardholder issues a transaction in country X and, within 2 hours, (s)he issues a second transaction with the exact same amount in country Y.*" If such scenario is detected in the stream of transactions, then the fraud detection system will produce an alert. Data driven rules are based on machine learning algorithms. They mine fraudulent patterns from historic data and aim to detect these patterns in the data stream of new incoming transactions.

1.1 Credit-card fraud detection at Worldline

The following discussion is based on an interview the author conducted with a fraud detection expert from *Worldline*[4]. It should be regarded as a first informal illustration of the various aspects involved in credit card fraud detection from the perspective of a domain expert who provided us with first-hand information. The literature review in section 2 will then examine the identified issues and challenges from the perspective of the scientific community.

Worldline offers merchant services, financial services and mobility services for the economic and the financial sector on a global scale. In particular they are the leading company for financial services in many European countries, processing volumes of payment data from companies such as Commerzbank, ABN AMRO, Postbank and many more. A key asset within their portfolio is fraud risk management which includes services such as data analytics, customer services and rule management. A pan-european team consisting of several dozens of experts work specifically on financial fraud prevention, fraud detection and fraud containment to mitigate losses for Worldline's customers and augment the quality of delivered services. Preventive measures aim to anticipate hazardous activity based on indications obtained from either the stream of financial transactions itself or from external sources such as the media, darknet or companies, in order to block vulnerable cards before any fraud can occur. Detection measures continuously monitor the stream of payments with the objective to raise alerts for individual accounts in cases when suspicious activity has been discovered. The detection happens in two stages: A real-time monitoring system which can potentially fully reject a transaction at the time a card holder uses her card at some terminal. And a near real-time system which monitors already accepted transactions and raises alerts to be further examined by expert investigators who can possibly contact the respective card holders.

[4] `https://de.worldline.com/`, Last access: 09.05.2019

A distinction between these two systems is mostly due to different resource and accuracy requirements. Even though a real-time refusal of a transaction would successfully prohibit any direct monetary loss through fraud, it would however be utterly inconvenient for legitimate card holders to get their transactions falsely refused. The third counter-measure against fraud is containment. It aims at limiting the impact of frauds by blocking compromised cards as early as possible or even blocking entire groups of cards which became compromised under the same data breach.

1.1.1 Fraud types

Credit card fraud originates either from the obtainment of a physical card or from the compromise of sensitive information associated with the bank account of a genuine card holder such as his credit card number, the expiration date or even the card verification code (CVC). There are several common routes by which a fraudster can obtain a working physical card or obtain sensitive information. In case of a stolen card the card holder is timely aware and he can quickly report the incidence to the issuer. In contrast, compromised sensitive information can easily go unnoticed by the card holder for weeks until the fraudster finally uses the credentials to issue a fraudulent transaction. Only then can the card holder, the issuer or the merchant possibly spot the fraudulent act and take measures. The target of any such fraudulent act is always the account of a legitimate card holder.

Fraudulent behavior in electronic payment systems manifests itself in many forms. Existing literature contains a multitude of taxonomies, targeted towards characterizing the different types of frauds. The employed categorizations differ in the viewpoint taken in order to highlight certain distinctive aspects of fraud. These viewpoints can be based on the environment from where the fraud originates, the fraudster's behavior or a mixture of both.

Gosh et al. [GR94] propose six fraud categories which include the types: frauds from lost or stolen cards, frauds from counterfeit cards, online frauds, bankruptcy frauds, merchant frauds or frauds from cards that got stolen at the time they were issued to the cardholder. A more recent taxonomy, introduced by Delamaire et al. [DAP09], is based on the fraudster's strategy on committing fraud. The authors distinguish application frauds from behavioral frauds. Application fraud occurs when fraudsters apply for a credit card and provide false personal information with the intention of never repaying the purchases. Behavioral fraud occurs when fraudsters obtain a cardholder's details and use them for criminal purposes. Another taxonomy, introduced by Patidar et al. [PSO16], classifies frauds into three categories, which are, traditional card related frauds, merchant related frauds and Internet frauds. A more generic categorization was introduced by Laleh et al. [LA09], who propose the types *Offline Fraud* and *Online Fraud*. Offline fraud refers to frauds committed with an actual credit card, also denoted as *card-present fraud*. Online fraud refers to frauds committed with only a credit card's information, also known as *card-not-present fraud*. Even though each of these taxonomies might be valid in its own right for distinctions in particular scenarios, oftentimes the defined categories are overlapping, potentially weakening the benefits of structuring fraud types according to a taxonomy.

Therefore, we deviate slightly from the categorization of Laleh et al. [LA09] and Delamaire et al. [DAP09], and present the different types of frauds in terms of the attack vectors along which a malicious agent could have gotten access to either a physical card, allowing him to commit *card-present fraud* (CP), or the sensitive data he requires to commit *card-not-present fraud* (CNP). Paths by which someone gains access to a physical credit card are straightforward. Merchants, issuers and card holders fought jointly against these types of frauds over decades which led to ever increasing security standards to be implemented in cards and terminals, higher awareness by customers and a strengthened data interchange between issuing banks and credit bureaus. As the European central bank states, "card-present fraud decreased substantially between 2012 and 2016, falling by 9.5%." [Ban18]. We know also from Worldline that only 8 % of the observed credit card frauds are card-present frauds. All remaining 92 % frauds occurred in card-not-present transactions, which are predominantly e-commerce transactions over the internet. Attack vectors for card-present frauds are:

Lost & stolen Fraud arises from the lost or stolen route, when someone steals the physical credit card or acquires it by other means. Either the agent himself or a third party (organized crime) then uses the card to illegitimately purchase goods or services on behalf of the rightful card owner. Before stealing a card, the perpetrator might try to *shoulder surf* his victim to acquire the PIN number along with the victim's card. Another variant within this category was introduced by malicious personnel who intercepted the outgoing mail after card production. Since banks now require additional authentication from the recipient before activating the card, this variant is no longer of major concern. Only 1% of the known frauds encountered at Worldline can be traced back to the lost & stolen scenario and, as [Ban18] reports, they are primarily coming from lost cards.

Counterfeit Fraud arises from counterfeiting, when someone clones sensitive information and security features from an existing card to reproduce them on duplicate fake cards, e.g. the magnetic stripe on a card (see figure 1.1). These cards are particularly used to spend money in countries where authentication via chip, as defined by the EMV standard[5], has not yet been implemented. Since 2011, the implementation of the EMV standard is mandatory for card payments in the single euro payment area (SEPA). This scenario amounts to 7% of known frauds encountered at Worldline. As [Ban18] concludes from responses of its surveyed entities, counterfeit fraud has become an issue of low priority for European transactions but remains an issue outside SEPA.

Application Fraud emerges from the application at an issuing bank, when someone applies for a credit card with false identity information. This can be a partly or entirely synthetic identity, called *identity fraud*, or someone else's stolen identity, called *identity*

[5] `https://www.emvco.com`, Last access: 15.04.2019.

[6] (left) `http://freecreditsoftwarez.blogspot.com/2012/01/credit-card-hack-with-valid-cvv.html`, Last access: 17.04.2019.; (middle) `https://www.focus.de/digital/computer/chip-exklusiv/tid-20321/skimming-anti-skimming-modul-und-tricks-gegen-betrueger_aid_568511.html`, Last access: 17.04.2019.; (right) Provided during a presentation of Worldline on 19.03.2019, Lyon.

Figure 1.1: (Left) Credit-card number generation tool: Credit Wizard, (Middle) Skimming devices integrated into a fake front panel of an ATM, (Right) Fake PIN-pad on a POS terminal. [6]

theft. Application fraud is a particular instance of *identity crime*. The key element of application fraud is the address. It is where the credit card will be sent to and picked up by the fraudster. Typical counter-measures include cross-matching of application details to discover duplicates [PGLSM05], maintaining and sharing blacklists with other banks and the consultation of credit bureaus, such as for instance SCHUFA in Germany.

The sources of card-not-present fraud are much more diverse and therefore less clear. They range from friendly frauds committed by individuals who falsely claim a genuine purchase as fraud trying to request chargeback, all the way to black markets controlled by organized crime [Eur12] [7][8]. Attack vectors of CNP frauds are virtual twins of attack vectors of CP frauds with the unique property that CNP transactions do not require a physical card but only the sensitive data associated with an account. The CNP attack vectors can be distinguished in:

Friendly fraud Friendly fraud occurs when the payer, after having issued a genuine transaction, contacts his card issuer to claim the transaction as fraud and request a chargeback. According to [Ban18] this type of fraud has reportedly been growing in recent years. Which is unsurprising at least for some countries. For instance, in the USA, according to United States federal law the card holder is held liable only for up to 50$ in case of CP fraud [Com86][9]. Some card issuers even adopted a zero liability policy, effectively clearing their customers from any responsibilities for fraudulent charges[10]. After discussions with Worldline, we can confirm that fraud investigation processes are conducted under the policy to trust the customers' statements unless there is clear evidence against them.

[7] `https://www.europol.europa.eu/newsroom/news/95-e-commerce-fraudsters-arrested-in-\international-operation`, Last access: 16.04.2019.

[8] `https://www.europol.europa.eu/newsroom/news/online-scammers-captured-after-causing-\eur-18-million-of-damage-in-more-35-000-cases`, Last access: 16.04.2019.

[9] `https://www.consumer.ftc.gov/articles/0219-disputing-credit-card-charges`, Last access: 17.04.2019.

[10] `https://www.visa.com/chip/personal/security/zero-liability.jsp`, Last access: 17.04.2019.

Identity theft Identity theft covers a broad spectrum of possible attempts to assume a person's identity in order to act on his behalf. A *clean fraud* refers to an instance of identity theft where someone obtains the complete card details including access to the registered 3D-Secure device, the CVC and the registered address of the card holder. Such frauds are virtually unpreventable by a merchant. Another increasingly popular and effective instance of identity theft is phishing [Ban18]; a scam that tricks card holders into revealing their personal information to malicious third parties. Such scams can happen over the phone, email or text messages by luring customers into confirming and/or completing partly provided account information. Phishing can lead to a complete takeover of the victim's bank account, with disastrous consequences for the victim, as this technique is resilient to blocked or reissued cards[11]. Another common source of stolen identities are database hacks at online merchants or financial institutes who maintain millions of records of personal information. There are countless examples of identity thefts of this kind. For instance, the recent data breach at the credit rating agency Equifax 2017[12] or the hospitality company Marriott[13].

Checker Starting from stolen incomplete card details or even from artificially generated ones (see figure 1.1), the perpetrator repeatedly tests combinations of credit card number, expiration dates and CVCs in transactions of small amount at an online merchant with low security measures. If the transaction is processed successfully, the perpetrator has verified the validity of the card information. To avoid attracting the card holder's attention and to avoid draining the credit limit, the purchased item is immaterial and of low monetary amount. It is up to the merchant and his acquiring bank to implement an upper limit on unsuccessful reentries. The validated card details can then be used directly or sold on underground "dump" shops[14] with prices ranging from 1-5$ up to 40-50$ for "fresh" cards with high validity rates[15]. A "dump" is a set of card records stolen during a particular hack or skimming attack. Prices depend on the recency of the dump, the location where the cards have been skimmed, the ratio of successfully validated cards within the dump and the first six digits of the card numbers. The first six digits correspond to the bank identification number (BIN) which allows perpetrators to guess the types of transactions that go undetected and the ones that will lead to blocking.

1.1.2 Fraud prevention and detection

In Worldline's fraud risk management unit, the real-time system can be considered as a fraud prevention system that analyzes transactions at the time they are issued from a

[11] `https://blog.credit.com/2015/09/what-hackers-want-more-than-your-credit-card-number-124442/`, Last access: 17.04.2019.

[12] `https://www.ftc.gov/equifax-data-breach`, Last access: 17.04.2019.

[13] `https://www.businessinsider.de/marriott-data-breach-500-million-guests-affected-2018-11?r=US&IR=T`, Last access: 17.04.2019.

[14] `https://krebsonsecurity.com/2014/06/peek-inside-a-professional-carding-shop/`, Last access: 17.04.2019.

[15] `https://krebsonsecurity.com/2014/02/fire-sale-on-cards-stolen-in-target-breach/`, Last access: 17.04.2019

terminal. Its objective is to either authorize or decline a transaction without introducing a noticeable delay in the payment process. Therefore, the available time budget during real-time decision making is limited to approximately 100 ms. Consequently, the active detection mechanisms are predominantly expert-driven matching rules that act on a reduced version of the card holder's payment history. The design of the rule set follows a conservative blocking strategy to avoid, as much as possible, falsely declined transactions and instead decline only the "obvious" fraudulent cases. A fictitious example of such rule is:

```
IF amount > 300 AND country == "China" AND merchant == "noea" AND uid != 5099
THEN decline()
```

In case of a declined transaction, the terminal (an ATM, a POS terminal or an online shop) refuses the payment and displays an error message. The real-time system stores transactions and rule responses for later in-depth analysis and performance evaluation.

The near real-time system processes already authorized transactions with the objective to raise alerts on suspicious accounts. The near real-time system can be considered the actual fraud detection system which has access to the full payment history and features strong analytic capability due to small resource constraints. It relies on a combination of complex expert-driven rules and statistical data-driven rules. Complex expert-driven rules are similar to the rules used in real-time but they contain even more specific conditions and they rely on information from longer card holder or merchant payment histories. Data-driven rules can be the outputs of classification algorithms, cluster analysis or behavioral profiling methods. Once the near real-time system raises an alert, the internal fraud investigation team can take one or several actions: Block the card temporarily, contact the card holder via phone, SMS, etc. or adapt the real-time system. The fraud investigator then reviews the recent transactions together with the card holder, identifies fraudulent cases, labels the transactions and organizes chargebacks. Depending on the review, the investigator either permanently blocks the card and reissues a new one or unblocks the card. Since fraud investigators are in direct contact with customers, they are the main source of knowledge when it comes to identifying new fraud patterns, updating rules and changing related systems. Based on their observations, they develop fraud scenarios and propose rules to the rule management unit which then creates simulations for the new fraud scenarios and files evaluation reports. Both expert-driven and data-driven rules are embedded in a rule management life cycle which continuously assesses the detection performance of individual and combinations of rules to maintain a highly effective set of rules at all times.

At the moment of writing, Worldline manages several hundreds of real-time rules and several thousands of near-real time rules. This pool of active rules changes drastically from one month to the next, making the rule life cycle management a key component in their system.

1.2 Issues and challenges

The problem of fraud detection is inherently coupled to several peculiar characteristics that hinder a straight-forward application of some well-known prediction methods and thus deserve closer attention. These characteristics are not necessarily specific to credit card fraud detection but their effects are particularly observable in businesses that encounter a high transaction volume issued at high frequencies by many different entities and the goal is to spot costly rare events in a vast realm of legitimate activity.

Imbalanced classes The predominant portion of incoming transactions is legitimate and only a very small fraction has been issued by fraudsters. It's not unusual that this fraction amounts to only 0.5 % of all transactions. For learning algorithms this *imbalance* between the majority class and the minority class can prevent the algorithm from learning a proper model of the data, in the sense that the algorithm is not able to discover patterns within the minority class or even that it completely ignores the minority class.

Feature engineering As in any modeling task, we are looking for a set of independent variables that best explain the dependent variable by using the labeled data set as proxy for the real phenomenon under study. Although transactions are characterized along several variables, there is no prototypical concept of "a fraudulent" or "a legitimate" transaction. Neither can we as humans tell a-priori what constitutes a credit card fraud nor can we be certain that any fraudulent (legitimate) act can be uniquely attributed to the values associated with the transaction. Furthermore, the class variable may depend on one or several independent, or even unknown latent, variables in a complex way and it is up to a thorough analysis to create, select and encode features that unravel this complex dependence.

Sequence modeling In contrast to many static prediction problems, a transaction is just a single event within a sequence of consecutive events and all these events are localized in time. It might very well be possible that there exist certain unknown dependencies between transactions in the sequence which render the occurrence of one event more or less likely whenever some other events have happened before. Likewise might the legitimacy of a transaction be influenced by the sequential context it appeared in. Since transactions represent interactions between customers and merchants it is also not clear at which granularity level such sequence would provide sufficient context for a transaction. Is it the globally ordered stream of transactions, the recently issued transactions of a customer or the recently received transactions at the merchant's end?

Concept drift and online learning Fraudsters constantly change their tactics and some fraud strategies go out of date, because the detection risk increased or because the efficiency of the method dropped due to security changes in the payment process. Other fraud strategies may emerge from technological advancements or from banal changes of customers' habits. These changes of fraudulent behavior are implicitly reflected in

the sequences of fraudulent transactions. Just as changes in fraudulent behavior affect fraud patterns, changes in the lifestyle of customers affect their spending patterns. The lifestyle itself may exhibit various notions of periodicity and vary in intensity over time, influenced by both personal habits and general trends (inflation, political stability, etc.). Detection systems require the capability to update their models regularly and efficiently as new observations arrive.

Performance measures and misclassification cost Measuring detection performance reliably is a vital necessity not only for sustaining scientific and technological progress but also for successfully transferring results from development to production. Most importantly, we have to define which aspect of a detection system we actually care about such that the aspect's quantification through some metric accurately reflects the system's performance. Moreover, different types of errors may induce different costs which require the definition of a realistic cost structure and a reasonable fusion between the quantified aspect and the associated error costs.

In the following chapter, we review related work from literature which specifically addresses these challenges. Later throughout this thesis, we revisit these problems and develop some existing solutions further in the hope to provide useful contributions to the domain of credit card fraud detection.

1.3 Contributions

The goal of this thesis is to analyze and develop machine learning based methods that can be used for raising alerts on suspicious transactions and potentially integrated into a data-driven fraud detection system. In particular, we make the following contributions:

- To mitigate the consequences of working with a proprietary data set, first we provide a statistical analysis of the credit card transaction data set we have at our disposal. As far as confidentiality concerns allow, we expose details about the volume, attributes and their distributions. Due to their different properties, all further analyses are carried out separately for card-present transactions and card-not-present transactions. We report the classification performance of a random forest classifier on the data set as a baseline such that we can compare it against other approaches developed in this thesis. We show the influence of varying majority class undersampling ratios on the prediction performance and illustrate empirically that prediction on future transactions (forecasting) is more difficult than prediction on the same time period.

- We augment transactions with additional features derived from domain knowledge and external knowledge. We integrate domain knowledge by making use of well-established feature aggregation strategies and we exploit external knowledge by extracting country and time related information from public data sources. We add simple demographic and holiday features as well as complex semantic features which we extract from a text corpus and a semantic knowledge graph. The results

show that external features improve the detection performance most noticeably on card-not-present transactions when used in conjunction with feature aggregates (see publications [JGS16] and [ZCG+17]).

- We model a user's transaction sequence with a Long Short-term Memory (LSTM) network. The results show that integrating previous transactions by means of an LSTM improves the prediction performance on card-present transactions while there is no improvement on card-not-present transactions. Moreover, frauds detected with an LSTM are consistently different from the frauds detected with a sequence-agnostic random forest classifier (see publication [JGZ+18]).

- We build on existing definitions of feature aggregates and provide a simple and yet flexible definition of these features to alleviate fast feature extraction through massive data parallelism. A feature selection study reveals that, aside from the aggregates defined in literature, many user-centric feature aggregates yield similar performance while differing from the known ones. Merchant-centric aggregates do not provide a similar performance boost.

1.4 Structure

We have introduced the problem of credit card fraud detection and its inherent challenges within this Chapter 1. Chapter 2 revisits the highlighted challenges from the perspective of related work by elaborating on how researchers in the domain address these challenges. In chapter 3, we introduce the data set, discuss appropriate evaluation metrics and cross validation principles and present results of a random forest classifier which will serve as performance reference throughout this thesis. In chapter 4, we examine different techniques for linking publicly available external data to the labeled transaction data set and analyze empirically the usefulness of such external knowledge for fraud detection. In chapter 5, we exploit the sequential nature of transactions and compare two methods that allow us to incorporate historic transactions in the decision about a transaction. In chapter 6, we exploit these findings and search the space of feature aggregates from a user and merchant point of view in a greedy manner with the goal to select discriminative feature subsets. Finally, in chapter 7, we motivate future work by outlining the potential of very simple probabilistic models that scale to millions of users while, at the same time, offering secondary advantages over raw detection accuracy. A final summary concludes the thesis.

Chapter 2

Related Work

Besides the interest of financial institutions in mitigating financial losses, credit card fraud detection has become an attractive test-bed for data mining researchers to study a broad range of intertwined challenges that rarely arise altogether in a single application domain. These challenges include imbalanced classes, feature engineering, sequence modeling, concept drift, online learning and problem-specific misclassification cost structures. Likewise, a plethora of methods from supervised learning, unsupervised learning (anomaly detection) and ensemble learning lend themselves to address individual or several of these properties.

In their fraud detection survey, Phua et al. [PLSG10] reviewed 51 articles from 1995 to 2004, covering home insurance fraud, automobile insurance fraud, credit card fraud and telecommunication fraud. The authors found that the studied data sets used in credit card fraud research typically comprise between 10 to 60 attributes (6 out of 11 articles), their volume rarely exceeds 5 million transactions (2 out of 11) and the fraud ratio is rarely below 1% percent (1 out of 5).

Recently, Abdallah et al. [AMZ16] published a survey about credit card fraud, telecommunication fraud, health care insurance fraud and automobile insurance fraud. They identify concept drift, skewed class distributions, the large data volume and real-time detection as central issues in fraud detection. According to their analysis, among 38 articles published in the domain of credit card fraud detection between 1994 and 2014, 13 of them addressed concept drift, 13 addressed imbalanced classes, 6 addressed the large data volume and 6 articles focused on issues in online learning. They conclude that most fraud detection systems across the reviewed areas use supervised approaches with decision trees and neural networks ranking among the most popular techniques.

In a meta-study from 2011, [NHW+11] reviewed 49 journal articles published on subjects around financial fraud detection between 1997 and 2008. Their comparisons revealed that classification methods are the most frequently used applications of data mining techniques[1] to fraud detection, accounting for 61% of all articles. The authors also note: "Given that outlier detection is a significant method of fraud detection, which has characteristics that confer comparative advantages over other techniques, more attention should be paid to it in future research." [NHW+11][p.563]. In our own literature research we could also not observe a great surge of interest in rigorously modeling genuine or

[1]Data Mining application classes were defined as: Classification (61%), Clustering (8%), Prediction (6%), Outlier-detection (2%), Regression (20%) and Visualization (2%).

fraudulent patterns with generative models (HMM, Boltzmann Machines, etc.) and in using them as proxies for anomaly detection.

Whereas fraud prevention systems take countermeasures a-priori to prevent frauds from being committed, a fraud detection system implements methods to identify frauds as they enter the company's area of responsibility. Throughout the years, researchers have proposed a large variety of approaches for automating fraud detection using data mining techniques. These approaches make use of a set of (labeled) observations from the past, apply some data mining technique to identify discriminative or class-specific patterns in the data and finally use the extracted patterns to decide upon the legitimacy of future observations. Some systems use only the transactional information while others consider sequences, entire accounts of card holders or merchants, hand-crafted features and even information extracted from external sources. In order to provide a useful overview over these problem-tailored systems, we follow the categorization of David Hand and Martin Crowder [HC12] and align the proposed systems along the data mining paradigm they follow: Rule-based, supervised learning and unsupervised learning (including outlier detection and anomaly detection).

Rule-based

Systems in this category are rather traditional and closest to manual decision making. They rely on expert knowledge about known fraudulent behavior in order to define characteristic signatures of frauds. Any other behavior is considered as legitimate. They utilize pre-defined rules, statistics or adequate heuristics to indicate the occurrence of a suspicious activity. Examples include the work of Deshmukh and Talluru [DT98] who developed a rule-based fuzzy reasoning system for assessing the risk of management fraud or Rosset et al. [RMN+99] who discover sets of decision rules by means of a modified C4.5 decision tree algorithm from both customer-level attributes and behavioral attributes to improve churn prediction on telecommunication records.

Supervised learning

Supervised learning methods explicitly model the differences between fraudulent and legitimate transactions. They have access to a wide variety of supervised classification algorithms, e.g. logistic regression, random forest, support vector machines, neural networks, etc. Supervised learning methods treat fraud detection as a binary classification problem and they aim to model the conditional distribution over the two classes *fraud* and *genuine* given training data. In a systematic review of 49 journal articles [NHW+11] showed that decision trees, neural networks, logistic regression and support vector machines have been the methods of choice among the many available methods.

Bhattacharyya et al. [BJTW11] compared the predictive accuracy of logistic regression, random forests and support vector machines on a real-world credit-card fraud data set under varying proportions of fraud in the training set, finding that random forests demonstrated an overall higher precision at low recall levels.

Carneiro et al. [CFC17] developed a general fraud detection system and deployed it at the site of an online retail merchant. They put particular focus on combining an

automatic data-mining stage with a manual revision stage. They cast the problem as a binary classification task and compare the performance of logistic regression, support vector machines and random forests on around 500.000 observations. They employ target encoding on categorical variables, min-max standardization on continuous variables, use temporal cross-validation with fixed train-validation-test splits and report results in terms of the receiver-operator characteristic curve (ROC) and the precision-recall curve (AUCPR). They achieved the best performance both in terms of ROC and AUCPR with the random forest classifier. The deployed system makes use of the entire set of historic orders, updates its model once every week and integrates a manual revision to determine a fraud score threshold.

Lucas et al. [LPL+19a] showed, that probabilistic models can serve as proxy for deriving anomaly features for credit card fraud detection. They modeled genuine and fraudulent expenditure sequences with a hidden markov model and then quantified the *normality* of test sequences with the genuine and fraudulent likelihood function to create sequence based features. These features are then used in conjunction with the raw transactional data in a random forest classifier.

Overall, there is an abundance of studies applying and comparing different off-the-shelf binary classifiers on fraud detection tasks. Starting from neural networks [GR94, SYY02, MTVM02, YhL09] over bayesian networks [MTVM02, EN96, YhL09], support vector machines [ZS15, DCL+14] and decision trees [MD15, YhL09] or random forests [BJTW11, DCL+14]. Researchers have also explored meta learning strategies such as boosting [VDD04], bagging [SFL+97] and stacking [CFPS99] to further improve the detection accuracy.

Unsupervised and semi-supervised learning

As we have seen, supervised learning techniques make use of the assumption that genuine and fraudulent patterns learned from past data carry over to the future. However, when the customers' or fraudsters' behaviors change, the learned patterns and their relations to each other might no longer admit accurate decisions. In this context, unsupervised learning techniques devise behavioral profiling methods, which are used to create models of the involved entities' spending activities and they monitor these models for departures from the norm. These methods typically incorporate expertise from research fields such as clustering, anomaly detection - also referred to as outlier detection or novelty detection - or peer-group analysis. Anomaly detection methods typically learn a model of normal instances, i.e. genuine transactions, and then identify those transactions as anomalous which do not conform to the model.

These methods require the definition of a suitable anomaly score to quantify an instance's degree of abnormality. Depending on the method, researchers have derived anomaly scores in different ways: In the Isolation Forest algorithm [LTZ08], the average path length from root nodes to leaf nodes is used as outlier score. Breunig et al. [BKSN00] proposed Local Outlier Factor, a method that estimates the local density around an instance and derives an outlier score from the ratio of the local density around one instance and the densities around neighbouring instances. Hemalatha et al. [HVL15] use low-support frequent item sets obtained via Association Rule Mining to

flag instances with infrequent attribute combinations as outliers. Conceptually similar, Zhang et al. [ZZQ17] proposed the Local Outlier Mining algorithm, that uses particle swarm optimization to find subspaces of low-density and then flags instances in these regions as outliers. In general, density based clustering algorithms lend themselves to outlier detection either because the concept of an outlier is built-in such as in the DBSCAN [EKS+96] algorithm (i.e. noise point) or because they are based on probabilistic models that provide direct access to an estimated joint distribution, e.g. the family of EM-clustering algorithms.

In the approach proposed by Juszczak et al. [JAH+08], each user has its own behavioral model together with an associated decision threshold above which any transaction is deemed fraudulent. Their results show that the one-class (anomaly) approach performs slightly worse than a regular two-class classification approach, but with an increasingly large gap between the training period and the test period, the performance of the classification approach deteriorates more quickly than the one-class anomaly approach.

Bolton et al. [BH01] and later Weston et al. [WHA+08] explore peer group analysis. They aim to detect individual accounts that begin to behave in a way distinct from accounts to which they had previously been similar. Each account is selected as a target account and is compared with all other accounts in the database in order to identify the target account's active peer group. Then, an account is flagged anomalous if it deviates by more than some defined threshold from the mean of its active peer group.

Carcillo et al. [CLBCB18] approach credit card fraud detection under the active learning paradigm. They address the real-world situation in which a manual investigation process is constrained by a fixed budget that does not allow more than a few cards to be investigated per day. The authors use active learning to repeatedly select and propose subsets of unlabeled transactions to the investigation process such that a subsequent classification algorithm can build its model from a more informative labeled subset than the overall imbalanced and noisy population of all transactions. The authors compare several unsupervised and semi-supervised (exploratory) active learning strategies for selecting transactions and querying their labels to maintain an up-to-date subset of highly informative labeled training instances. In addition, they study several combination functions for devising card risk scores from transaction risk scores.

In the following sections, we provide an overview of solutions proposed by researchers to specifically address the issues raised in Section 1.2.

2.1 Imbalanced classes

As Chawla states, "a data set is imbalanced if the classification categories are not approximately equally represented" [Cha09]. With increasing interest in applying machine learning algorithms to real-world problems, the effects of imbalanced data have become apparent. In particular, since discriminative methods do not incorporate a notion of class prior explicitly, the decision boundary they represent is solely determined by minimizing an error measure between the true class and the predicted class. With few minority-class instances, minimization of the error is governed predominantly by minimizing misclassifications on majority-class instances which can lead to an insufficient fit of the boundary

in regions where minority-class instances appear side-by-side to majority-class instances. Moreover, imbalanced classes exhibit different (and unknown) extends of sample bias making it difficult to correct for by means of adjusted misclassification costs. Obviously, evaluating the performance of a classifier on imbalanced data in terms of accuracy is not appropriate but instead other performance measures are required (see Section 2.5).

Researchers have developed solutions to mitigate these issues by either adopting sampling techniques for balancing the data before any algorithm is applied or by adapting the learning algorithms themselves.

The sampling strategies aim to determine the correct distribution for a learning algorithm. Empirical evidence suggests that the natural distribution might not always be the best for learning a classifier [WP01, Cha03]. Researchers reported a negative impact of imbalanced data on learning decision trees [JS02, VR05], neural networks [VR05] and support vector machines [YLJH03]. The most common strategies for artificially balancing data are: majority class undersampling, minority class oversampling or a mix of these two extremes [JS02].

Majority class undersampling keeps all minority class examples but discards as many majority class examples at random or according to an heuristic until some desired class distribution is met. The obvious shortcoming of undersampling is the removal of potentially important examples. Pozzolo et al. [DPCB15] analyzed the interaction between the undersampling ratio, the induced warping of the posterior distribution and the increased variance of the classifier and how these factors relate to the ranking error of the posterior probability. They find that a beneficial impact of undersampling strongly depends on the original degree of imbalance, the class separability and the variance of the classifier and, as a consequence, is therefore very much task dependent. Kubat and Martin [KM97] and Zhang et al. [ZM03] propose several nearest neighbour based heuristics for selecting majority class examples close to the decision boundary and discarding those examples that are far away from the boundary.

On the other hand, minority class oversampling leaves the majority class examples untouched, but repeatedly draws minority class examples at random with replacement. This strategy duplicates minority class examples which adds no additional information but increases the misclassification cost of the minority class examples and, finally, may lead to overfitting on the few artificially duplicated examples. Oversampling lends itself naturally to synthetic data augmentation techniques that try to create similar but yet distinct minority class examples artificially.

Chawla et al. [CBHK02] propose *SMOTE*, an oversampling technique that creates new artificial examples of the minority class by interpolating between nearby minority class examples, to effectively populate the feature space in regions where minority examples are located. The authors evaluate the technique for a C4.5 decision tree where it outperforms naive random oversampling. By interpolating the minority class examples with synthetic examples, they effectively reduce the number of disjuncts in the induced tree which leads to a better generalization as opposed to the specialization effect emerging by randomly duplicating minority class examples. Since then, the method has inspired further research and extensions [CLHB03, HWM05, HYGS08] and has also been used to balance classes in credit card fraud detection [Dal13].

Just as sampling methods address class imbalance on a data level, meta learning methods address imbalance on an algorithmic level. In an attempt to overcome the disadvantages of over-fitting on minority class examples or missing out on important majority class examples, Liu et al. [LWZ09] proposed *EasyEnsemble* and *Balance Cascade*, Wang et al. [WTY09] proposed *UnderBagging* and Chawla et al. [CLHB03] extended SMOTE to *SMOTEBoost*. EasyEnsemble creates a set of balanced but independently sampled subsets via undersampling and fits them with an ensemble of weak learners with the *AdaBoost* algorithm. As a variant, Balance Cascade trains the weak learners iteratively and uses the weak learners' predictive accuracy to withhold already correctly classified majority examples from entering the next subset. UnderBagging uses bagging to combine the predictions of classifiers trained on independent balanced subsets via majority voting. SMOTEBoost uses the AdaBoost algorithm but runs SMOTE as an inner loop to create additional synthetic minority class examples in each round of boosting, thus increasing the overall weight of the minority class.

Later in Section 2.5, when we review common performance measures, we discuss cost-sensitive learning which introduces instance-specific costs to penalize misclassification errors unevenly according to some cost-matrix.

2.2 Feature engineering

In credit-card fraud detection, a fraud is supposedly not exclusively a property of the transaction itself but rather a property of both the transaction and the context it appeared in. While a transaction might arouse suspicion if, for instance, the spent monetary amount is very high and it takes place at a particular type of merchant at a certain time of day, such fraud detection strategy considers transactions in isolation from each other. Neither the historic purchases of the customer nor other transactions from the merchant he interacted with are taken into consideration. It might very well be possible that a certain mutual occurrence of transactions is a better indicator of fraud than any transaction on its own, e.g. the co-occurrence of video games and shoe store merchants in the same account). Two transactions occurring in geographically distant locations at almost the same time are often considered as suspicious patterns ("collisions"). Also, it may be that the volume of transactions per time (in an account) is more indicative of fraud than the absolute attributes of any transaction [WHJ+09]. In order to go beyond transaction-level detection, it is crucial to choose a suitable transaction context and extract relevant features from it.

Most feature engineering work in credit card fraud detection follows transaction aggregation strategies such as the ones described by Whitrow et al. [WHJ+09]. The idea is to characterize the spending pattern in an account, the so called *activity record*, by accumulating transaction information over time. Through exploratory data analysis, the authors identified several salient aspects of transactions as being relevant to fraud detection, such as the number of recently issued transactions and the total monetary value of these transactions. Their activity record incorporates the number and the total value of various transaction types aggregated over fixed time windows, spanning 1, 3 and 7 days. The authors point out that the possibilities for creating other statistics,

such as the variance of amounts or maximum difference between successive amounts, are endless. Any combination of attribute values can be considered a transaction type, e.g. the occurrence of shoe store merchant and authentication via signature or a cash withdrawal in England during night hours. It is important to note that aggregates do not require any label information and, therefore, are entirely unsupervised. While aggregates function as account-level features, they can be turned into transaction-level features by updating the activity record with each new transaction. Most commonly, the statistics from the activity record are simply added as supplementary features to the basic feature set when training a fraud classifier. Their experiments show that, across many different classifiers, the prediction accuracy significantly increased as the aggregation spanned more than a single transaction – with random forests clearly performing best. This aggregation strategy was found to work well in commercial applications and it is also an integral part of research in the academic field with broad usage in a number of studies [BJTW11, JGC12, BSAO13, SBD13, DCL+14].

Since then, the proposed aggregation strategy served as generic template for developing new features. For instance, Jha et al. [JGW12] and Bahnsen et al. [BASO16] do not aggregate over specific transaction types but instead on those transactions whose attribute values are identical to the current transaction. This difference is subtle but it has severe implications for the semantics of the aggregated value. For example, Whitrow et al. [WHJ+09] calculate the number of transactions issued in the last 24 hours at a specific merchant (e.g. "MyFancyShoeStore") for all transactions in the dataset. The value of this aggregate is probably zero for most transactions of most users but its semantic is consistent across all transactions. The aggregates defined in [JGW12, BASO16] calculate the number of transactions issued in the last 24 hours at the *same* merchant as the current transaction. If the current transaction was issued at "MyFancyShoeStore", the semantic of the aggregate is identical to the one above. Otherwise, the aggregate represents the user's recent activity at whatever merchant he just interacted with.

In general, most studies seem to agree that the Recency - Frequency - Monetary Value (RFM) framework provides the necessary information for discovering frauds. Recency refers to the time since the last transaction, frequency captures the volume of transactions per time interval and the monetary value corresponds to the total money spent within the interval. Some studies consider only frequency and monetary value [Kri10], whereas others also integrate the time difference [VBC+15, BASO16]. From this perspective, aggregations are convenient because they allow to condition the RFM statistics on transaction types. It should be noted that there is no consensus on how to choose the most suitable time intervals. Usually, researchers apply reasonable rules of thumb and compute aggregates over days, weeks or months.

Fu et al. [FCTZ16] propose *trading entropy*, a feature that is supposed to quantify the amount of information added to a user's purchase history when he issues a new transaction. As an example, they define the feature based on the amount spent and the merchant. In a user's history, they consider all but the most recent transaction and calculate how the total amount is distributed over the merchants. They quantify the uncertainty in this distribution by means of the Shannon entropy. The trading entropy is then the difference between the entropy of this distribution and the distribution they

obtain after adding the most recent transaction to his history.

Another interesting approach for extracting expressive features is discussed in [VBC+15]. Apart from the detection system the authors maintain a bipartite graph in which nodes correspond to merchants or card holders and edges correspond to transactions between these entities. The edge weight is determined by the volume of transactions exchanged between a card holder and a merchant, and it decays exponentially with time. With a page-rank style algorithm the authors propagate the fraud label of a transaction to directly connected or transitively reachable card holder and merchant nodes. The authors extract network features from the graph which measure the current exposure of each node to fraud. These features include a score for the card holder, the merchant and the transaction, aggregated over short, medium and long time periods.

2.3 Sequence modeling

Anomaly detection in sequential data is an active field of research but it is more popular in other domains such as intrusion detection, DNA sequence analysis or navigational path analysis on websites [CBK12]. The approaches in this domain can be categorized in similarity-based, counting-based and model-based techniques.

Similarity-based techniques define a similarity measure on sequences and then threshold the similarity to derive anomaly scores for test sequences. For instance, Bolton et al. [BH01] measure the similarity of two sequences with the euclidean distance between embedded versions of the sequences whereas Chandola et al. [CMK08] measure similarity in terms of the normalized longest common sub-sequence. Counting-based techniques maintain a dictionary of fixed-length normal sub-sequences and then derive an anomaly score of the entire sequence by counting the proportion of known normal sub-sequences. The *t-STIDE* method [WFP99] and its variants have been applied successfully to intrusion detection. Model-based approaches assume explicitly a sequential dependence between consecutive data points. In its simplest form such model could be a Markov chain defined on the data points. However, in many applications of practical interest the sequential dependency is presumably more evident or useful as a sequence of latent, so called *hidden*, states that control the sequence of observed data points. Since these states act as a kind of memory of the past, a decision about a sequence of data points can be reduced to a decision based on the states.

There are different types of hidden state architectures. Within the family of probabilistic models, *Hidden Markov Models* (HMM) are one of the most popular representatives of generative models. They can be used to perform unsupervised learning of normal or abnormal succession patterns in sequential data. When HMMs are used to perform anomaly detection, a typical use-case is to run a new transaction sequence as query against the model and compute the sequence's likelihood of having been generated by the normal or abnormal HMM. The likelihood is then used as a score to flag anomalous sequences [IMJ+11, Dho12, DCP13]. An alternative hidden state architecture is the *Conditional Random Field* (CRF) [PLM01]. Such a model is trained discriminatively with labels in order to directly learn the conditional distribution over classes given a sequence of data points. However, we could not find any application to fraud detection.

Srivastava et al. [IT13] maintain one HMM for each card holder which models his succession of transaction amounts. They use k-means clustering to transform continuous transaction amounts to discrete expenditure symbols. A transaction/sequence is classified as fraudulent if the likelihood of the most recent $m-1$ transactions together with the newly arriving transaction falls below the likelihood of the most recent m transactions by a fixed margin. Robinson et al. [RA18] follow a similar approach, but they model the transaction amounts from the merchants' perspectives using the Kullback-Leibler divergence between the distributions represented by the "old" HMM and its updated version as a measure of anomaly. Lucas et al. [LPL+19a] add another perspective to this general approach by additionally distinguishing between card-present and card-not-present transactions. And instead of relying on some fixed threshold, they use the sequence likelihoods under a genuine and fraudulent HMM as features for a subsequent binary fraud classifier.

Although conceptually similar, a *Recurrent Neural Network* (RNN) is a hidden state architecture from the family of neural networks. The connectivity between hidden states, so called nodes in neural network terminology, also satisfies the Markov assumption. RNNs are trained discriminatively to predict the label of a transaction given the sequence of transactions from the past. Graves et al. [Gra12] give a detailed introduction to RNNs and their variants. More recently, one of the variants, Long Short-Term Memory Networks (LSTM), received much attention due to its capability to learn long term dependencies in sequences. This advantage led to broad success in practical applications such as speech recognition and machine translation [GJ14, SVL14a]. The usage of LSTMs for fraud detection is rather limited. Bontemps et al. [BCMLK16] used them for modeling normal network traffic to subsequently detect denial-of-service attacks. Wiese et al. [WO09] used LSTMs to classify genuine and fraudulent transaction sequences for credit card fraud detection. We will extend this work in Chapter 5.

2.4 Concept drift detection and system adaptation

Fraud detection operates in a dynamic environment. Customers may change their purchase habits on a global or local scale, fraudsters may abandon strategies and instead adopt more effective strategies or it is simply the overall transaction volume that undergoes seasonal changes. As we observe the environment only through records of transactional data, this data is more naturally regarded as a data stream than a finite and static data set. In this stream, groups of patterns - commonly referred to as concepts - are constantly drifting, new concepts may emerge and others may disappear. Researchers refer to this phenomenon as *concept drift* [WK96, ABBL04, GFHY07, MTRAR+12], whose facets we are going to discuss in more detail in Section 3.4. Since data mining techniques are supposed to leverage prevalent patterns in the data to later turn the extracted knowledge into decisions, they need to adapt dynamically to the changing environment. In this context it is important to preserve the previously acquired knowledge as well as to adjust it in the light of new observations. Concepts from the past may re-occur at some later time while others may vanish entirely. A detection system needs to respond to the changing data distribution, while ensuring to retain relevant

past knowledge. Thus, two major questions arise: How to detect concept drift and how to adapt the detection system to new observations?

Following the categorization of Klinkenberg et al. [KRA99], different types of indicators can be monitored to detect concept drift: (i) Performance measures of the current classifier such as accuracy, precision or recall, (ii) properties of the classifier such as the average path length in decision trees or the number of support vectors in SVMs, (iii) properties of the data such as the distribution over dependent and/or independent variables or the rank of discriminative attributes. Drift detection is closely linked to drift adaptation although one can implement adaptation mechanisms to anticipate unfavorable performance degradation without explicitly detecting or quantifying the extent of concept drift. Both concept drift detection and concept drift adaptation mechanisms typically process the stream in batches of data at some fixed temporal resolution (e.g. daily). An adaptation to changing data distributions can be implemented by (i) using a time window of fixed or adaptive size on the training data or (ii) by weighting the training examples or parts of the classifier (e.g. base learners in an ensemble) according to their age or utility for the classification task.

In their *FLORA* framework and its variants, Widmer et al. [WK96] use the accuracy and coverage of a set of boolean rules as indicators to dynamically adjust the window size from which new rules are extracted in the subsequent iteration. Klinkenberg et al. [KRA99] monitor accuracy, precision and recall together with the sample standard errors on a window covering several fixed-size batches. They decrease the window size whenever the most recent indicator values fall below the lower end of their confidence interval and increase the window size as long as the recent indicator values stay within the confidence intervals. Wang et al. [WFYH03] do not aim to detect concept drift but maintain an ensemble of classifiers, where each of them is trained on one of the incoming batches. They propose to weigh the classifiers' predictions inversely proportional to an estimate of the classification error calculated on the most recent batch. Gao et al. [GFHY07] pursue a similar direction and propose to retain all minority class examples while dropping the majority class examples from all but the most recent batch. They train an ensemble of classifiers on bootstrapped samples drawn at a fixed class-ratio from the global set of minority class examples and the most recent set of majority class examples and report the ensemble's mean posterior probabilities as final scores. Pozzolo et al. [DCL+14] compared these two concept drift adaptation mechanism to a static approach on a credit fraud detection data set, finding that both adaptation mechanism are superior to the static scenario over a range of hyper-parameters such as the number of consumed batches, updating frequency and re-sampling method.

Lucas et al. [LPL+19b] introduced a technique to detect and quantify the extent of concept drift between transactions from different days. For each pair of days (a, b), they collect all transactions from these days $\mathcal{D} = \mathcal{D}_a \cup \mathcal{D}_b$ and label the transactions according to the day they came from. They partition the set and train and evaluate a classifier on a hold-out test set from $\mathcal{D}$. Through manual inspection and after clustering, the authors can show that low "day distinguishability performance" clearly corresponds to calendar clusters of weekdays, Sundays and work holidays or Fridays and Saturdays.

2.5 Performance measures and misclassification cost

Three peculiarities have guided the choice of performance measures for evaluating fraud detection systems: The resource limits of an obligatory manual investigation process, the imbalance of classes and the asymmetry between false positive and false negative errors.

Since the fraud investigation team can only review a few hundred suspicious cases each day, we are not so much interested in a strict 0/1 classification as it does not permit any ranking from more suspicious to less suspicious examples. Therefore, we typically consider continuous fraud scores that are based on the posterior probability as assigned by a classifier. We want the positive examples (frauds) to rank at the top of such a sorted prediction vector while all negative examples (genuine transactions) should rank at the very end of the vector. Moreover, when the negative examples largely outnumber the positive examples, a suitable performance measure must factor in this non-uniform prior. And finally, classifying a positive example as negative may result in higher business costs than classifying a negative example as positive[2]. It's even worse than that. The cost of a classification error not only depends on the class but also on the example itself. Missing out on a 1000€ fraud is most likely more expensive than missing out on a 100€ fraud. Falsely rejecting ATM transactions while the customer is on vacations may create more inconvenience than it would during any other time of the year. Even within a particular business context, it might not be clear what the relevant costs are and how they can be determined.

The research community has largely abandoned simple measures such as accuracy, misclassification rate or the true positive rate and instead adopted the area under either the receiver operator characteristic curve (ROC) or the precision recall curve (PR) to summarize the performance of a system over a range of decision thresholds under imbalanced classes [PLSG10]. Beyond the bare classification performance, researchers have explored several criteria to measure different notions of cost [HWA+08]. Whitrow et al. [WHJ+09] provide a detailed example together with the necessary reasoning about possible factors that might influence the design of a suitable cost matrix. A common strategy is to set the misclassification cost proportional to the available credit limit on the card in order to place more weight on the first frauds in a sequence. Sahin et al. [SBD13] and Bahnsen et al. [BAO15] inject an amount-based cost in decision trees via a modified splitting criterion that minimizes the sum of misclassification costs. Mahmoudi et al. [MD15] use Fisher Discriminant Analysis for credit-card fraud detection and they account for the dynamic misclassification cost by adding an expenditure-dependent weight in the computation of the classes' sample means. Beyond the difficulties that revolve around proper definitions of mislcassification costs, Hand & Crowder [HC12] point out that there is a selectivity bias in the fraud detection rates when comparing a new detection system against the system that is currently in operation. This bias tends to favor the existing system and it arises from the fact that the active system gets to trigger account investigation and labeling whereas the proposed system does not.

[2]In fact, the reverse might be true as well, since companies typically have insurance plans to cover their chargebacks while they need to maintain a team of investigators.

Chapter 3

Dataset and Baseline

Payment data contains very sensitive private information about individuals and businesses and access to such data is highly restricted to only the data owners and the companies that manage the data. Which is the reason why there is no publicly available data set that would sufficiently reflect the magnitude and variety of real-world card payments and that could therefore be considered as a basis for studying the many interesting challenges present in this domain. During a research project with Worldline we had access to real-world credit card payment data and corresponding ground truth labels. Most of the work presented throughout this thesis is based on this data set. The only alternative we were able to find was a small sample of 280.000 transactions with PCA-transformed features[1] published alongside a study which was conducted jointly together with the exact same payment service provider [PCJB15]. This scarcity of publicly available real-world data is problematic for the research community, since it compromises transparency, comparability and reproducibility of published findings. Therefore, we decided to include a dedicated chapter about the data and its intrinsic properties with the hope to disclose as much relevant information as possible to the reader without compromising the privacy of individuals.

As listed in Table 3.1, transactions are characterized by several attributes. We can broadly categorize them into attributes characterizing the transaction itself, attributes characterizing the card holder/card that issues a transaction and attributes characterizing the merchant who receives the transaction. Out of the 34 attributes, 15 are transaction related, 16 are card holder related, defining his declarative profile or the card, 2 are merchant related and 1 attribute is the fraud label. In addition, each card holder and each merchant is assigned a unique identifier. In our context, the merchant refers to exactly one specific terminal and not the company. For instance, when you pay the bill in a restaurant by card, the waiter will bring a small card reading device to the table, you plug your card in and issue the transaction. The reading device is called a terminal and it has a globally unique ID. We call such terminal *the merchant* since we do not know the higher-level identifier of the restaurant itself.

The fraud label is binary and it indicates the authenticity of a transaction. The label was assigned by human investigators after consultation with the card holder. The label takes the value `0` in most of the cases. Only when a transaction has been identified as having been issued by a third party and not the rightful card holder, is the label set to

[1] `https://www.kaggle.com/mlg-ulb/creditcardfraud`, Last access: 10.05.2019.

Category	Attribute	Description	Feature(s)
Transaction	Time	Date and time of the transaction in Belgium time.	`weekday`, `hourOfDay`, `timestamp`
	Type	The kind of transaction issued. A real payment (e.g. e-commerce) or a virtual transaction (3D-secure ACS).	`ecom`, `3DSecureACS`
	Authentication	The kind of authentication experienced by the card holder: Chip-based PIN check, signature, 3D-secure.	`emv`, `authentication`, `3DSecure`
	Entry mode	The interaction modality between the card and the terminal: e.g. contactless, magentic stripe, manual entry.	`cardEntryMode`
	Amount	The transacted monetary value in Euro.	`amount`
Card holder	Declarative profile	Several attributes describing the card holder such as his country of residence, age or gender.	`userCountry`, `age`, `gender`
	Card	Several attributes describing the credit card such as the credit limit, card type or expiration date.	`creditLimit`, `cardType`, `expiryDate`
Merchant	Declarative profile	Two attributes describing the terminal: The merchant category code assigned to the terminal (e.g. shoe store, ATM, etc.) and the country the terminal is registered in (e.g. Belgium, Germany, etc.)	`terminalCategory`, `terminalCountry`
Internal	Fraud label	The manually assigned label indicating whether a transaction was a fraud.	`fraud`

Table 3.1: Transactions are described along four groups of attributes: Transaction related, card holder related, merchant related and internal attributes. Underlined attributes are in fact collections of several attributes. Examples of attributes in the collection are listed in the description. Due to confidentiality reasons we can not expose the full list of attributes and their names but we list the most important ones for this thesis with pseudo-names in the last column.

	Fraudulent	Genuine	Total	Fraud ratio (%)
ECOM	72,240	21,702,272	21,774,512	0.333
F2F	11,429	25,926,906	25,938,335	0.044
Total	83,669	47,629,178	47,712,847	0.176

Table 3.2: Number of transactions partitioned into card not present (ECOM) and card present (F2F) transactions.

1, which flags the transaction as fraudulent. Throughout the thesis we will refer to the label as the *fraud label* or the *class* of the transaction.

3.1 Distinction between card-present and card-not-present transactions

The data set comprises transactions from a period of 92 days, starting from 1st of March 2015 to 31st of May 2015. As Table 3.2 shows, the overall proportion of fraudulent transactions amounts to 0.176 % in the entire data set, confirming the severe class imbalance which is characteristic of fraud detection problems. The fraud ratio in face-to-face (card-present) transactions is by almost one order of magnitude smaller than in e-commerce (card-not-present) transactions, which is a first hint that these two scenarios require separate treatment.

In total, the data set comprises transactions issued between 3.5 million card holders and 2.3 million terminals. The number of transactions per card holder roughly follows a power law distribution with 99% (50%) of the users having issued less than 79 (8) transactions. Similarly 99% (50%) of the merchants have received less than 269 (2) transactions. We call the collection of transactions from a user or merchant *a sequence*, because the transactions are localized in time and hence exhibit a natural ordering.

Figure 3.1 displays the sequence length distribution from the perspective of users (left) and merchants (right). As expected, in both the e-commerce and face-to-face scenario there are few ”power” users who issue hundreds of transactions within the three months while the vast majority stays below one hundred (Note: log-scale in the figure).

When we view the number of transactions issued per hour over the course of three months, we observe a clear weekly and daily periodicity in both the e-commerce and face-to-face scenarios (see Fig. 3.2). However, there are some peculiarities that distinguish the two scenarios: (i) The nightly activity in face-to-face is consistently smaller than in e-commerce, (ii) the activity in face-to-face peaks during the afternoon whereas the activity in e-commerce is more spread out across the hours of a day and (iii) the hourly volume of face-to-face transactions increases consistently throughout a week until it reaches its maximum on Saturdays. In contrast, the fraudulent activity does not show any obvious regularity, neither in face-to-face nor in e-commerce. We note, however, that maximums in fraudulent activity tend to appear during the night.

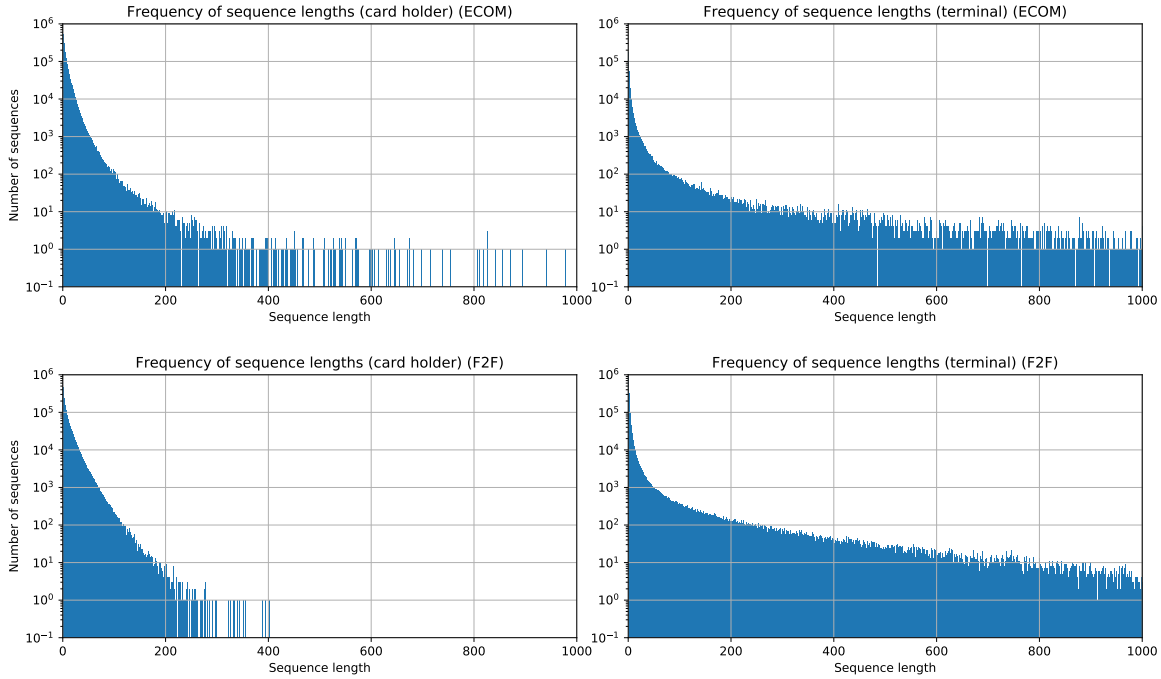


Figure 3.1: Frequency of transaction sequence lengths when transactions are grouped by e-commerce/face-to-face (top/bottom) and card holder/terminal (left/right).

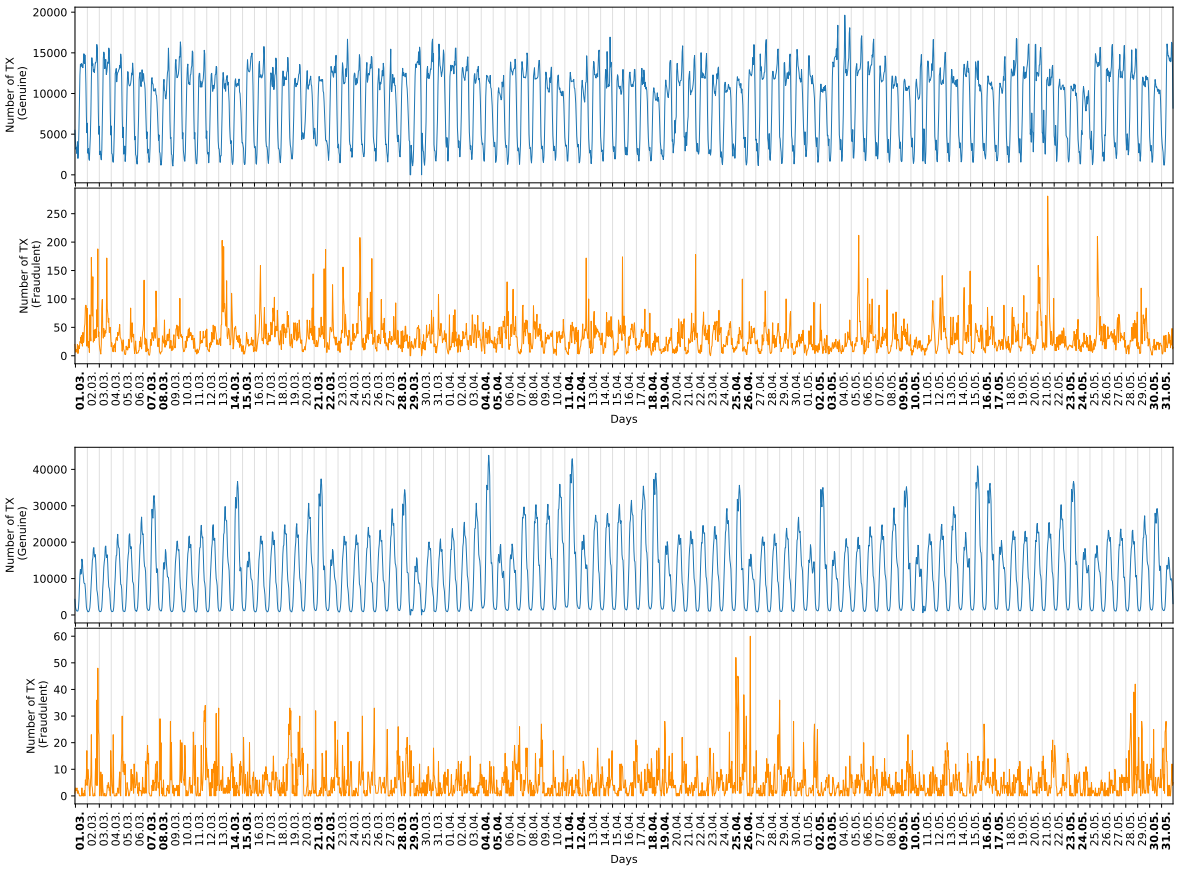


Figure 3.2: Evolution of the number of transactions per hour displayed over three months. Top: e-commerce; bottom: face-to-face. Weekends are highlighted in bold. Tick marks and grey vertical lines correspond to midnight.

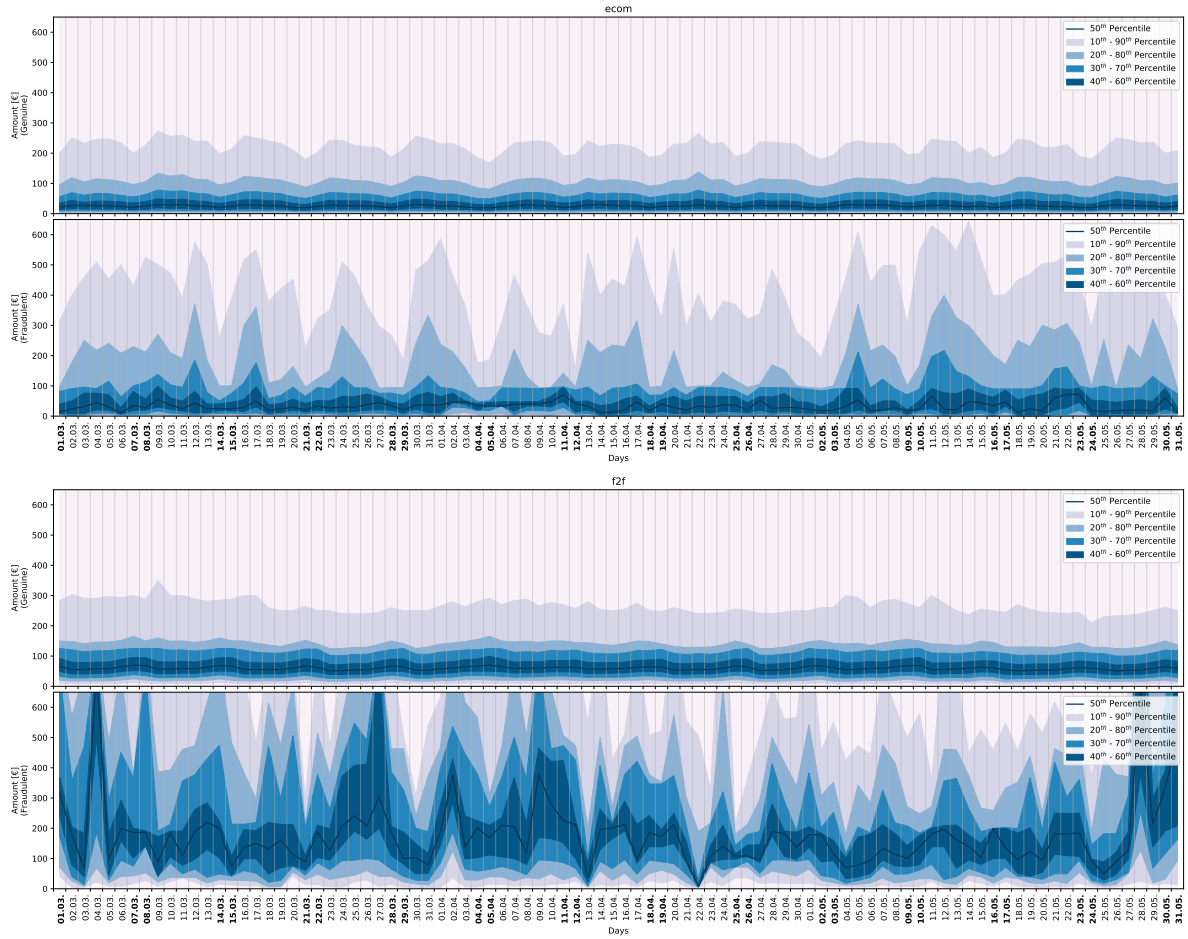


Figure 3.3: Evolution of transaction amount per day displayed over three months. Top: e-commerce; bottom: face-to-face. Weekends are highlighted in bold. Tick marks and grey vertical lines correspond to midnight. Shaded regions enclose different percentile intervals as indicated by the legend.

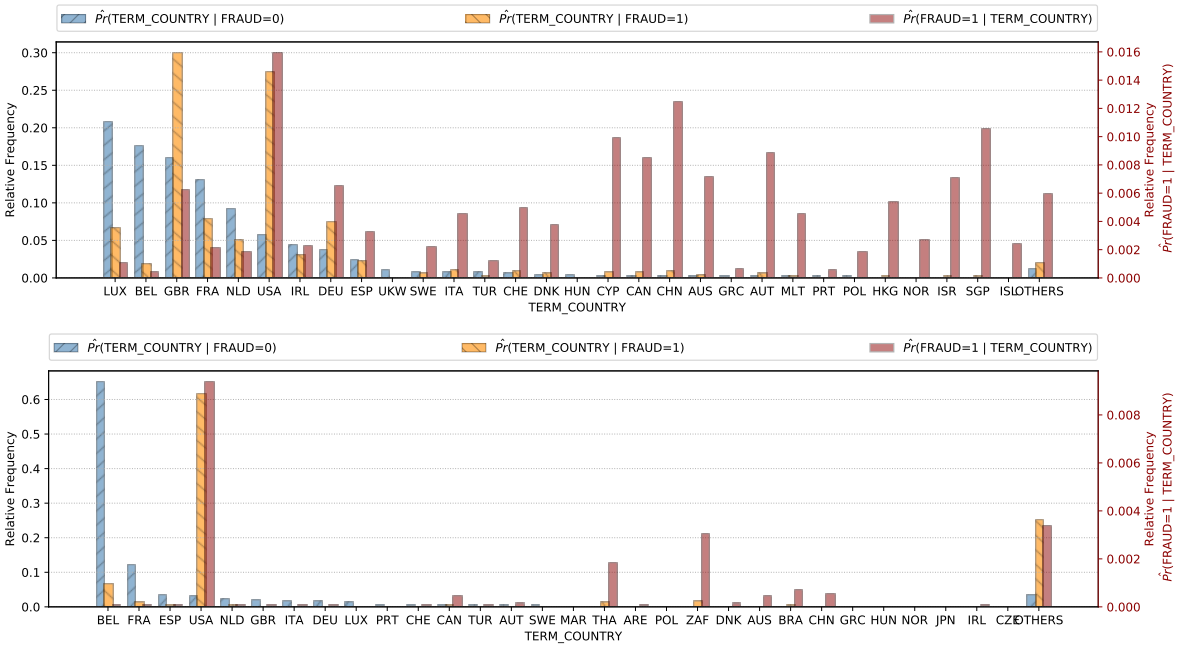


Figure 3.4: Distribution over merchant countries, sorted by frequency in the genuine set. Top: e-commerce; bottom: face-to-face. Values of the class conditional relative frequencies are measured on the left axis, attribute conditional fraud ratio is measured on the right axis.

We denote the monetary value transferred via a transaction as the transaction *amount*. It is recorded in Euro currency. Regarding the distribution of amounts over time, we can not observe such a crisp periodicity as with the sheer transaction numbers. Based on Fig. 3.3, we highlight the following central observations: (i) Throughout the three months, the median of daily transaction amounts is consistently lower in e-commerce ($25.01^{\pm 3.10}$) than in face-to-face ($59.48^{\pm 4.99}$), (ii) a subtle weekly periodicity suggests the occurrence of higher amounts in e-commerce during the beginning of a week but during weekends in face-to-face and (iii) fraudulent amounts exhibit a smaller median with less variation in e-commerce ($32.16^{\pm 14.19}$) than in face-to-face ($173.50^{\pm 107.58}$). As in the transaction numbers, a periodicity on a monthly basis is not noticeable.

In e-commerce, the largest fraction of genuine transactions was issued at merchants with terminals registered in Luxembourg, Belgium, Great Britain or France. In contrast, fraudsters tend to favor terminals in Great Britain and the US with more than 50% of the observed frauds being recorded in these two countries (see Fig. 3.4, top). In face-to-face, 80% of all genuine transactions have been issued at terminals in Belgium or France whereas more than 60% of all frauds have been recorded in the US. Regarding the proportion of frauds within each country (see Fig. 3.4, right axis), it is apparent that some countries exhibit a fraud ratio that is by one order of magnitude higher than the overall base fraud ratio according to table 3.2: These are the US, China, Singapore and Cyprus.

Even though we can not identify the company a merchant belongs to, we have access to a merchant category code (MCC). The MCC is defined in an international standard[2]

[2] `https://www.iso.org/standard/33365.html`, Last access: 13.05.2019.

that classifies businesses by the type of goods and services they provide for the purpose of implementing restrictions, fees, cash back reward programs or tax reporting for payments at particular merchants. The MCC is a 4-digit number and the standard allocates different number ranges to different merchant types. Some number ranges are left unallocated thus leaving space for banks to define their own categories. Figure 3.5 shows that genuine e-commerce transactions occur most often in record shops, direct marketing or travel agencies and hotels. Among the merchant codes that exhibit above average fraud ratios are: Car rental services, generic business services, video game, electronic sales and clothing stores. Within face-to-face transactions, restaurants, grocery stores, ATMs and hotels are among the most common MCC codes. Here, video amusement game supplies, money wire transfers, mobile home dealers, fast food restaurants and drug stores are among those with the highest fraud ratio. This is in line with publicly available information about high-risk merchant categories as identified by merchant consulting agencies[3].

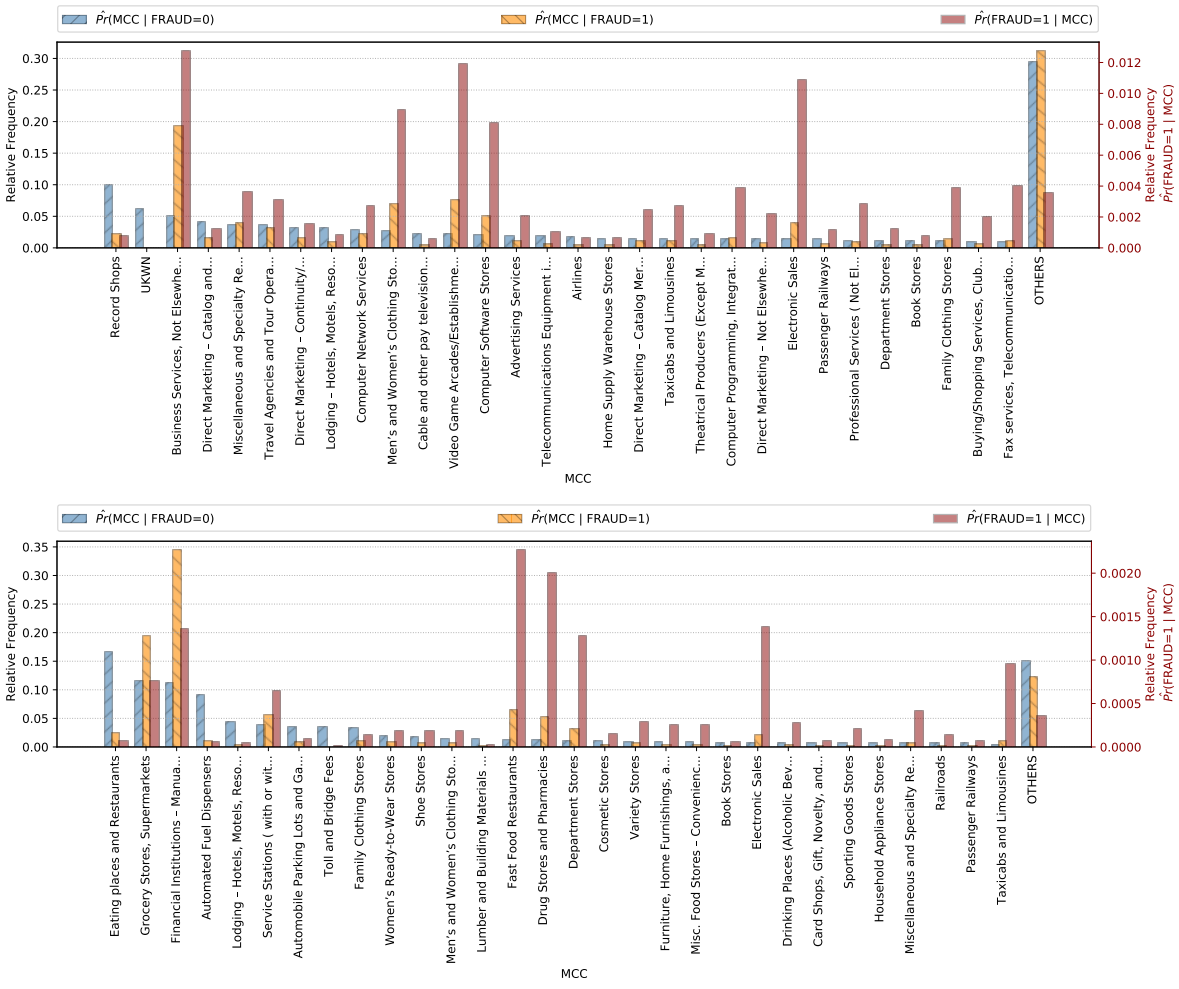


Figure 3.5: Distribution over merchant category codes, sorted by frequency in genuine set. Top: e-commerce; bottom: face-to-face.

In the figures presented here, we used a MCC mapping table[4] which had been ex-

[3] `https://chargebackhelp.com/mcc-code-list/`, Last access: 13.05.2019.
[4] `https://github.com/greggles/mcc-codes`, Last access: 13.05.2019.

tracted from US Internal Revenue Service (IRS) specifications. Since not all MCC codes are globally defined and banks are free to assign their own meaning to certain code ranges, this mapping is not necessarily correct for rare MCC codes. However, with this particular mapping table we found a curiosity in the data set which is worth sharing.

When looking at the MCC codes with the highest fraud ratios in Fig. 3.6, the descriptions associated with e-commerce merchant codes bear a peculiarity. First, they are very specific - rather do they correspond to names of individual businesses but to a broad business category. Second, most of the names seem to represent major car rental businesses. After a little bit of research, many names deviate from the suggested original company name by subtle spelling mistakes:

`TILDEN TENT-A-CAR` vs. `TILDEN RENT-A-CAR`, `ALTRA AUTO RENTAL` vs. `ALTA AUTO RENTAL`, `AVCAR RENT-A-CAR` vs. `ACAR RENT-A-CAR`, `INTERENT RENT-A-CAR` vs. `INTERRENT`.

All these merchant codes in Fig. 3.6 (top) contain between 2 to 21 transactions. The consistency in the types of businesses the names refer to and the high fraud ratios leave us to believe that we are witnessing a very bold fraud strategy, where fraudsters managed to register bogus companies, with names that are very similar to major brands, to cover up their scam. Since we can't know for sure whether our MCC mapping table is correct and since we don't know the real businesses behind these transactions, it is only a guess. However, we thought it might be interesting to share this suspicion to point out that it might very well be possible that some merchants, either in terms of MCC codes or terminal IDs, are entirely compromised or at least subject to surprisingly high fraud rates.

We end our exploration of the dataset's properties with an analysis of the temporal succession patterns of transactions. Each transaction occurs as an event between a card holder and a merchant. The events are localized in time as given by the time stamps of transactions. For a given pair of card holder and merchant, the set of time-localized events forms as sequence and we are interested in the elapsed time between two consecutive events in such sequence - the *inter-arrival times*. Since we observe only very few transactions for each card holder-merchant pair, it is prohibitive to analyze the inter-arrival times on such fine granularity. Instead, we analyze inter-arrival times of sequences centered on the card holder only, ignoring the identities of the merchants, and sequences centered on the merchant only, ignoring the identities of the card holders.

In particular are we interested in the relationship between the two inter-arrival times in triplets of three consecutive transactions. For each transaction in a sequence, we form a triplet of the transaction time stamps denoted as (t_{-2}, t_{-1}, t_0), where t_0 is the time stamp of the transaction of interest, t_{-1} is the time stamp of its immediate predecessor and t_{-2} is the time stamp of the predecessor of the immediate predecessor. We denote the inter-arrival times between these three events as $dt_{-1} = t_{-1} - t_{-2}$ and $dt_0 = t_0 - t_{-1}$. Figure 3.7 shows 2D histograms of pairs dt_{-1}, dt_0, in which the three transactions are either all genuine or all fraudulent.

A general observation is that fraudulent transactions appear predominantly in very short temporal succession with most of the probability mass being concentrated within 24 hours. This is slightly more pronounced in the face-to-face scenario. The merchant centered triplets follow in shorter temporal succession than user centered triplets. When

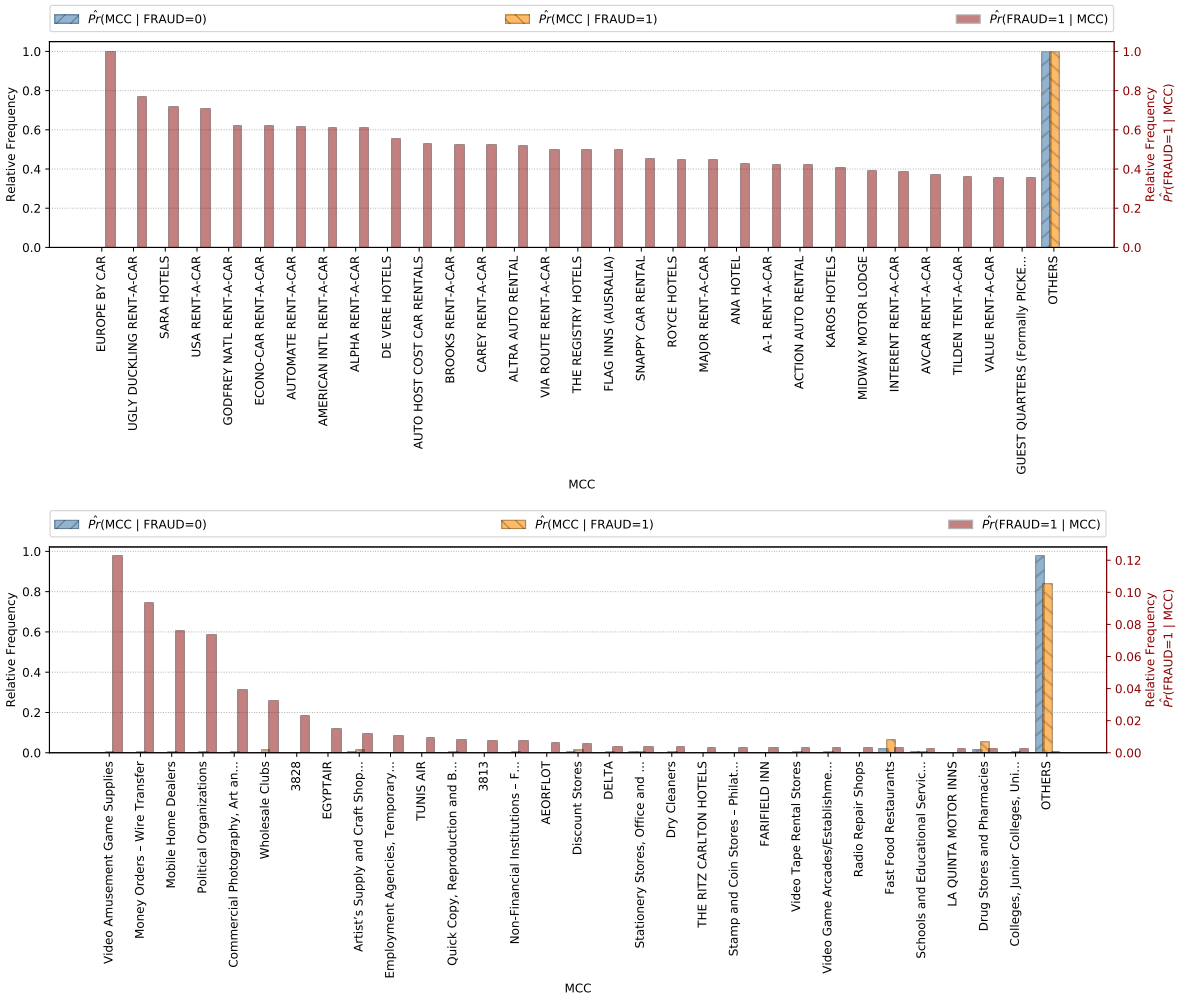
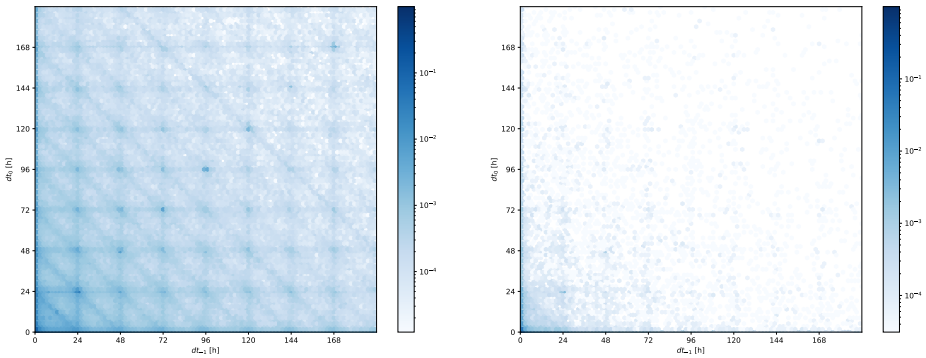
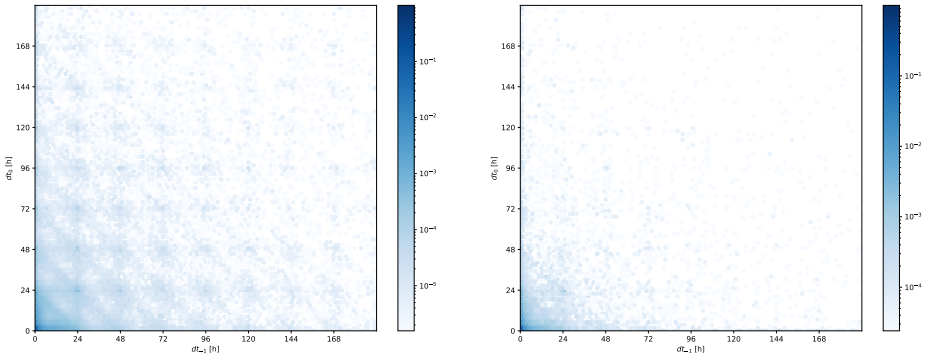


Figure 3.6: Distribution over merchant category codes, sorted by proportion of frauds. Top: e-commerce; bottom: face-to-face.

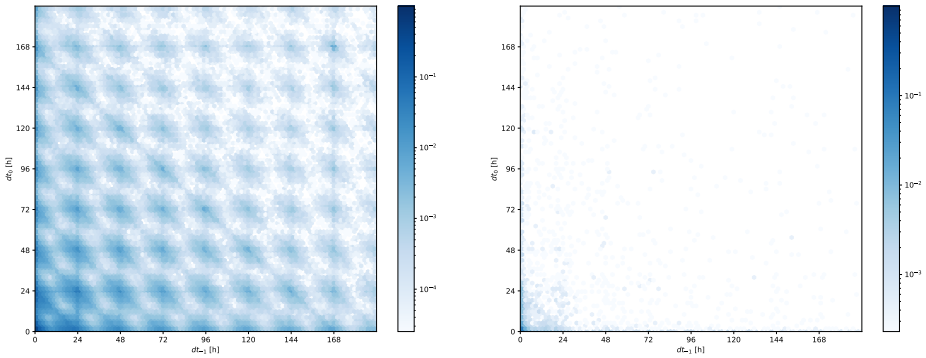
(a) ECOM - User - Genuine (b) ECOM - User - Fraudulent



(c) ECOM - Merchant - Genuine (d) ECOM - Merchant - Fraudulent



(e) F2F - User - Genuine (f) F2F - User - Fraudulent



(g) F2F - Merchant - Genuine (h) F2F - Merchant - Fraudulent

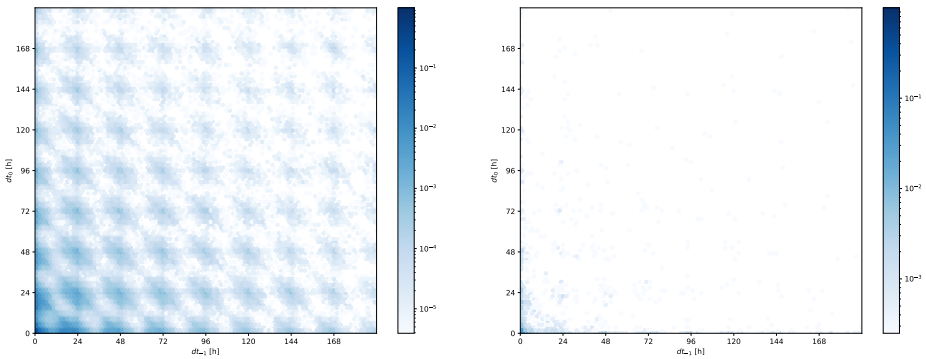


Figure 3.7: 2D histogram of inter-arrival times of user/merchant-consecutive transactions in e-commerce (a, b, c, d) and face-to-face (e, f, g, h). For all triplets of three consecutive transactions with timestamps (t_{-2}, t_{-1}, t_0), the figures show $dt_{-1} = t_{-1} - t_{-2}$ plotted on the x-axis against $dt_0 = t_{-1} - t_0$ on the y-axis. A point at $(24, 48)$ corresponds to all transactions that occurred 24 hours after their predecessors and 48 hours before their follow-up transactions. The color represents the relative frequency at each point.

we compare succession patterns of genuine e-commerce transactions with genuine face-to-face transactions, we notice that the probability mass has maximums on 24 hour cycles. Within e-commerce the mass is more spread out as in face-to-face. Also in face-to-face, the clusters around 24 hour cycles are not exactly circular but skewed in the direction $(dt_{-1}, -dt_0)$, which is expected when we consider that merchants usually have fixed opening hours. If the last transaction was a little longer ago than 24 hours, we would expect the next one to occur in a little less than 24 hours or multiples of it. Or in other words, if we want to predict the time to the next transaction, we need to know the current time.

3.2 Feature encoding

Most learning algorithms can not handle categorical variables with a large number of levels directly. Both supervised [KZP07] and unsupervised methods [LWW15] require preprocessing steps to transform them into numerical features. In this section, we discuss several common solutions for encoding categorical attributes such that they can be used in learning algorithms.

Integer Encoding The most direct encoding of categorical attributes with K different values is to create discrete variables $x \in \{1, 2, \ldots, K\}$. For instance, when an observation of a variable with $K = 4$ states takes on the k-th state, we encode the observation as $x = k$. Depending on the learning algorithm, integer encoding might not be adequate to represent categorical attributes. The distance on integers typically does not reflect the distance on the attribute's values and would cause misleading results in distance-based algorithms.

1-of-K Encoding (also known as One-Hot Encoding) The K values of discrete nominal attributes correspond to distinct categories with no relation to each other, e.g. different states of a system, countries or payment modes. Although there are various ways to encode these variables, the 1-of-K encoding scheme is particularly convenient and commonly used in machine learning literature [Bis06]. A categorical variable is represented by a K-dimensional vector $\mathbf{x}$, in which one element x_k is set to 1 and all remaining elements are set to 0. For instance, if we have a variable with $K = 4$ states and we want to encode one observation where the variable assumes state x_2, we represent the observation by $\mathbf{x} = (0, 1, 0, 0)^T$. This encoding scheme is related to but not to be confused with dummy variables used in statistics, where for a categorical variable with K factors, we define $K - 1$ indicator variables to indicate the presence of any of $K - 1$ states with the K-th state being implicitly determined when all indicators equal zero.

Likelihood Encoding (also known as target or label encoding) The value of a nominal attribute is encoded by its relative frequency of showing up in the positive class in a training set. If the k-th state of a nominal attribute appears n_p times in the positive class and $n - n_p$ times in the negative class, the state gets assigned the value $x_k = \frac{n_p}{n-n_p}$. This type of encoding requires class labels and it assumes that the nominal values'

representation in both classes provides sufficient information to distinguish the values. Therefore, we consider this technique rather as a feature engineering method than an actual encoding technique.

Vector Embeddings The values of categorical attributes may not always be orthogonal as suggested by a 1-of-K coding scheme but they rather exhibit some notion of similarity relevant to the prediction task. For instance, an attribute like the terminal's country can be endowed with different notions of similarity based on geographical locations, economic alliances or cultural backgrounds. It is common practice in machine learning to account for such expert knowledge through purposefully encoded attributes. Even if the main motivation is not an integration of a-priori knowledge, categorical attributes with many different levels partition the set of examples into many small sets for which it might be challenging to obtain reliable parameter estimates.

Vector embeddings are the outcome of unsupervised encoding techniques with the aim to embed the values of a categorical variable in a continuous valued vector space. The role of unsupervised methods is thereby to estimate a map $c : X \to V^d$ from categorical values X to points in a d-dimensional vector space V^d such that distances between points reflect some sought notion of similarity between categorical values. We keep this superficial description of vector embeddings because there is an abundance of techniques by which such maps can be created, each with its own definition and interpretation of c and V. For instance, the rows and columns in a tf-idf - matrix [Jon73] in information retrieval, the projections on the first d principal components in dimensionality reduction, the mixture coefficients in probabilistic models with mixture distributions [BNJ01] or the weights in neural networks [MDK+11]. Similar to the likelihood encoding, the continuity in the embedding space is assumed to permit reasonable interpolation between otherwise distinct categorical values.

3.3 Performance measures

Throughout this thesis we will construct classifiers on a range of different features and with several methods such as decision trees, neural networks and logistic regression. The question naturally arises as to which set of features or method is *best* or better than another for our particular problem. There is no simple answer to this question, because the notion of *best* depends on many factors. Factors that we have to define in the light of peculiarities in the domain. Since we cast credit card fraud detection as a binary classification problem, we have to decide on criteria for evaluating, illustrating and comparing the performances of binary classifiers. As with the methods themselves, researchers have defined and used an abundance of such criteria including misclassification rate, likelihood ratios, the receiver-operator characteristic (ROC) curve, precision and recall and many others [Pep03].

Ideally, we choose the criterion such that it reflects our aims. Our aims are threefold and we can motivate them from different perspectives. First, there is a practical requirement from a business point-of-view: Predictions from data-driven approaches are used to raise alerts for a subset of transactions which got assigned high fraud scores. Since

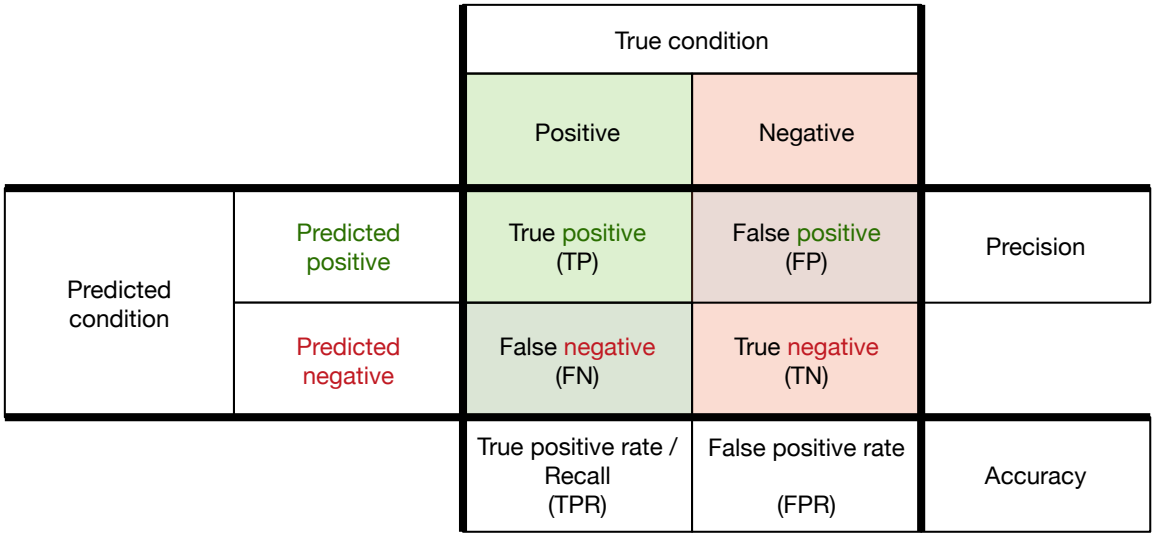


Figure 3.8: A contingency table displaying the four possible outcomes of a binary decision (colored boxes) and an excerpt of measures that quantify different properties of the decision maker in terms of ratios of these outcomes. Type 1 errors (false positives) and type 2 errors (false negatives) are shaded in gray.

the validity of each alert must be checked by human experts before blocking a credit card, the number of alerts to be raised per day is limited and determined by the experts' efficiency. A high confidence in the validity of raised alerts is more important than raising alerts for every possible fraudulent observation. Secondly, there is the requirement of being able to inspect the prediction properties of different classifiers over the whole range of operating points. Many classifiers output scores that can be turned into binary decisions by choosing an operating point - a threshold. The choice of threshold is non-trivial and we seek a criterion that factors in these different choices. And finally, there is necessity to compare the performances of different classifiers by means of a concise single-valued measure. The measure must be objective such that given the predictions there is no additional parameter to be specified.

In consideration of these requirements, we briefly recap some basic measures for assessing the performance of a binary classifier at some fixed decision threshold. The measures are defined on the four elementary outcomes of a binary decision, which we summarized as a contingency table in Fig. 3.8.

Accuracy Accuracy quantifies our intuitive interpretation of the degree of correctness of a classifier as the number of correct decisions over all decisions taken. It has a range of $[0, 1]$, approaches 1 when all decisions are correct and is invariant to the error types. The complementary measure quantifies the ratio of incorrect decisions over all decisions and it goes by the name *misclassification rate*. Accuracy is defined as:

$$\alpha = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision In contrast to accuracy, precision quantifies the degree of correctness within one single type of prediction. It is the ratio of correct decisions in favor of one class over all decisions that have been made in favor of that class. It relates the number of correct decisions for a class with the type 1 error. The range is again $[0, 1]$, it approaches 1 as all decisions in favor of the class are in fact correct. In binary classification, precision is defined on the positive class:

$$\pi = \frac{TP}{TP + FP}$$

True Positive Rate (Recall) While precision measures correctness relative to the number of decisions that have been made in favor of a class, recall measures correctness relative to the number of elements in a class. It relates the number of correct decisions for a class with the type 2 error. The range is $[0, 1]$ and it approaches 1 when the decisions for all elements of a class are correct. Precision and recall together fully determine the proportions of outcomes in the confusion matrix. In binary classification, recall is defined on the positive class as:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate Similar to recall, the false positive rate is measured relative to the number of elements in a class. However, it is the ratio of *in*correct decisions over all elements of a class. The range is $[0, 1]$ and it approaches 1 when the decisions for all elements in a class have been made in favor of the same but incorrect class. In binary classification, the true positive rate and the false positive rate characterize the proportion of positive decisions in the positive class and the negative class, respectively. The false positive rate is defined as:

$$FPR = \frac{FP}{FP + TN}$$

Given our aims, we can already rule out accuracy as a performance measure since it does not distinguish the two kinds of errors. In our setting we want to avoid unnecessary manual effort. Therefore it is acceptable to miss out on several frauds, as long as the ones we flag positive are in fact frauds and not falsely reported genuine transactions. The measures described above are defined on one set of decision outcomes. However, many classifiers output scores which express some form of affinity towards one or the other class. It is up to us to define a threshold in the range of scores to turn the scores into binary decisions. Each choice of threshold gives rise to a different confusion matrix and different values of the measures accordingly. Since it is impossible for us to specify precise values for parameters such as the misclassification costs, we opt for an aggregate measure that combines measures of performance under different thresholds.

One aggregate measure that is commonly used to compare binary classifiers is the area under the receiver operator characteristic curve (AUCROC). The receiver operator characteristic is a pair of TPR and FPR, calculated at a particular threshold. Evaluated at all thresholds, these pairs form a curve which is typically displayed by plotting the TPR values against the FPR values. Since TPR and FPR are invariant to the number of elements in each of the two classes, a random decision oracle with a fair chance of

$p = 0.5$ to predict positive produces a ROC curve that lives on the diagonal. The more the curve is bent away from the diagonal and tends towards the corners $(0, 1)$ or $(1, 0)$, the more can we support the hypothesis that the decisions are in fact not random. This intuitive interpretation is compromised if the classes are severely imbalanced. In our case, the baseline random oracle has a chance of around $p = 0.001$ to predict positive. Such oracle produces a ROC curve which is already heavily bent towards the corner $(0, 1)$. Consequently, the ROC curve of any genuine non-random classifier is visually indiscernible from the curve of the adequate random baseline. It is common to report a single-valued summary of the ROC curve by calculating the area under that curve. However, as Hand [Han09] notes, the area under the ROC curve as a measure to compare different classifiers has a fundamental weakness. He shows that, "[...] instead of regarding the relative severities of different kinds of misclassification costs as the same for different classifiers, the area under the ROC takes these relative severities themselves to depend on the classifier being measures. [...] It is as if one chose to compare the heights of two people using rulers in which the basic units of measurement themselves depended on the heights."

An alternative aggregate measure is the area under the precision recall curve. The precision recall curve displays explicitly the quantities we are interested in: the precision plotted against the recall for every choice of threshold. It allows us to read the expected accuracy of a classifier in the early retrieval range (low recall) directly from the curve (see Fig. 3.9). For any level of recall, the precision varies with class imbalance. This is a desirable property because a system might perform well on a balanced datatset while it performs poorly on an imbalanced dataset. In a ROC curve we can not see such difference, because the proportion of true positives over false positives is not explicitly displayed. Further details about the intricacies in interpreting ROC curves versus precision recall curves have been discussed in [SR15] and [DG06]. Although the interpolation between points requires special care, the area under the precision recall curve (AUCPR) can serve as a single-valued aggregate measure of a classifier's performance [Han09]. In Fig. 3.9 we illustrate several measures that can be derived from a precision recall curve. The most important ones for us are the area under the entire curve (AUCPR) and the area in the early retrieval range (AUCPR@0.2):

Precision at K (P@K) The proportion of true-positives in the set of K transactions with the highest scores. Each choice of K corresponds to a specific level of recall on the x-axis.

Area under the precision recall curve (AUCPR) The integral of precision over all levels of recall over the interval $[0, 1]$. The range of AUCPR is $[0, 1]$, where the maximum is achieved when all fraudulent transactions got assigned higher scores than all genuine transactions.

Area in the early retrieval range (AUCPR@0.2) The integral of precision over recall levels from 0 to 0.2. Instead of reporting precision at several values of K, we use this early retrieval range to reflect the application-specific focus on high precision in the

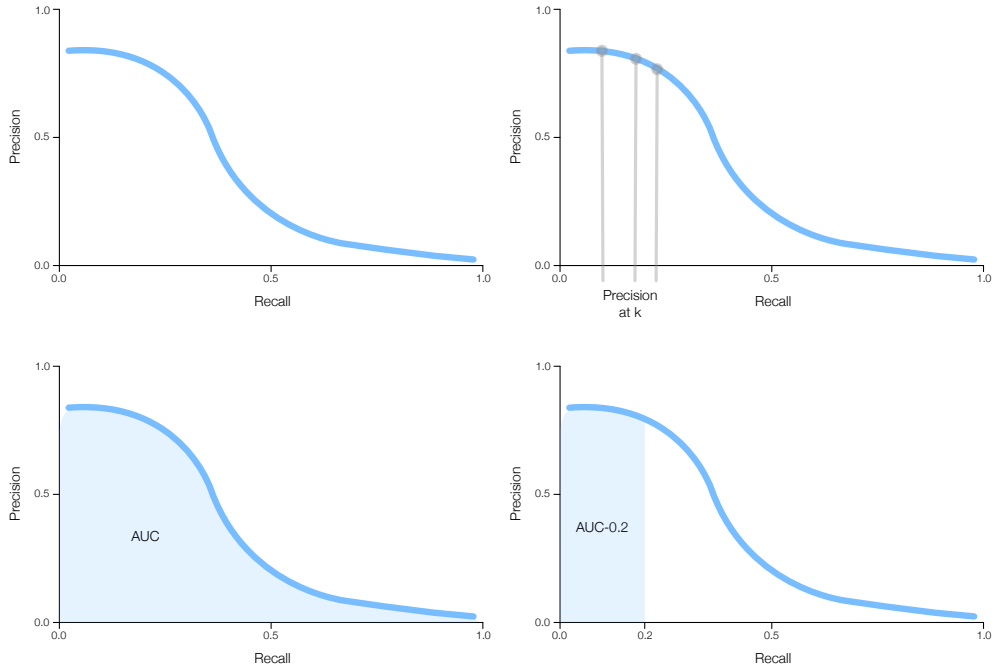


Figure 3.9: Top: Precision recall curve and precision at K, indicated by grey lines. Bottom: Area under the precision recall curve and area in the early retrieval range, calculated only over the interval $[0, 0.2]$.

range of low recall. The range of AUCPR@0.2 is $[0, 0.2]$ and the maximum is achieved when at least 20% of all fraudulent transactions got assigned higher scores than all genuine transactions.

3.4 Forecasting and data set shift

After having defined which aspect of performance we are going to measure, we now turn our attention to how we are going to do that. Many learning algorithms, that have been derived in statistical learning theory and then later applied in the field of machine learning, rely on the assumption of observations being independently and identically distributed. Although these two assumptions rarely ever hold in practice, we want to illustrate briefly why these assumptions are important in terms of the implications they have for prediction and model validation.

We generically denote by $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n} \subset \mathcal{X} \times \mathcal{Y}$, with $\mathbf{x}_i$ being a feature vector of observation i and y_i being the corresponding label, the set of observations (sample) we recorded from some broader phenomenon of interest (population). The assumption is that all pairs $(\mathbf{x}_i, y_i)$ are independent and identically distributed under the same but unknown probability distribution $p(y|\mathbf{x})p(\mathbf{x})$. The assumption of *identically distributed* observations is what makes learning useful. In supervised learning, we aim to learn a model of $p(y|\mathbf{x})$, which we call $h_\theta(y_i|\mathbf{x}_i)$, and we assume such a model exists - there is a single parameter vector θ^* such that the conditional distribution over any y_i given $\mathbf{x}_i$ can be approximated sufficiently with $h_{\theta^*}(y_i|\mathbf{x}_i)$. As a consequence of the *identically distributed* assumption, we can re-use our feature-conditional model h_{θ^*} for prediction,

since y_i depends on i only through $\mathbf{x}_i$, regardless of the identity of i. As long as each $\mathbf{x}_i$ contains all the information we need to have in order to reason about the corresponding y_i under the same model h_{θ^*}, we can also do so for yet unseen observations.

The consequences of the *independence* assumption are more extensive. When modeling y given $\mathbf{x}$ we can go down two major routes. One is called discriminative modeling and the other one is generative modeling. In discriminative modeling we proceed as described above by modeling $p(y|\mathbf{x})$ directly as a (parameterized) function $h_\theta(y|\mathbf{x})$. Such models are typically called classifiers in machine learning with logistic regression, decision trees or neural networks being examples. Here, the independence assumption is only a conditional independence assumption on the y_i's given $\mathbf{x}_i$'s and we generally do not make assumptions about the distributions over $\mathbf{x}_i$'s themselves. In generative modeling, we specify a model for the joint distribution $p(y, \mathbf{x})$, typically in terms of a product of a class-conditional distribution $h_{\theta_1}(\mathbf{x}|y)$ and a marginal distribution $h_{\theta_2}(y)$. Then we use Bayes' formula to derive the classification model $h_{\theta_1,\theta_2}(y|\mathbf{x})$. The naive Bayes classifier is the canonical example for devising a classifier under a generative modeling approach. The assumption made in generative modeling is typically that the pairs $(\mathbf{x}_i, y_i)$ are independent and identically distributed. In the end, whichever modeling approach we choose, the independence assumptions are used to derive learning methods such as, for instance, minimizing the empirical risk or maximizing the likelihood function. They also give rise to scalable learning algorithms because independence allows to decompose the joint distribution over observations as a product of observation-specific independent factors.

One central question in statistical learning is whether the learning method eventually results in a model that is close to the true distribution $p(y|\mathbf{x})$. Absolute answers to this question are given by learning theory, which characterizes consistency, bias and variance of estimators in the limit of infinite sample sizes. It also provides error bounds and confidence intervals for estimates obtained from limited data. Since, in practice, the true $p(y|\mathbf{x})$ is unknown and i.i.d. assumptions might not fully hold, a relative assessment of the quality of a model is often preferred in applied machine learning. In practical applications, it is of great value to obtain this relative assessment of the performance of a model which can then be compared to others. Estimates of assessment scores are computed empirically by repeatedly splitting the data set into training and test data sets. A principled approach to obtain estimates of model performance is cross-validation. Throughout this thesis, all model assessments are computed empirically. If the assumptions are wrong, we can thereby at least create a comparison between several wrong models and then select the one that is still most useful according to the performance measure and the test data set.

When we treat individual transactions in credit card fraud detection as the elementary observations, the i.i.d. assumption is broken in many ways. First, the data is inherently grouped on card holders with multiple transactions observed for each card holder. It is likely that the transactions from a card holder are dependent on the identity of that card holder. The question we should have in mind is whether knowing the ID of the card holder would change our belief about the variety of different transactions that are likely to occur compared to our belief when we would not know his identity. The

answer is likely positive, at least for card holders whose card usage style deviates from the expected general card usage style. This line of reasoning applies in equal measure to merchants. Knowledge of the identity of a merchant is likely to change our belief about the variety of transactions we are likely to see. Merchants sell different types of goods and services, have different ranges of opening hours or simply restrict certain payment modes, all of which render some transactions more likely than others. Throughout the thesis, we absorb the violation of independence within groups to some extent by explicitly adding information about card holders and merchants to the feature vectors. It would not make sense to add the card holders' IDs directly because in the end our goal is to choose a model that generalizes to unseen transactions/card holders and not just remember the compromised card holders from the training set.

Apart from the within-group dependency we might also encounter a dependency of transactions across time. The transactions appear sequentially as events localized in time. Therefore, they are naturally ordered on the time axis which may make all variables to depend on time. This is particularly problematic in our scenario as we aim to use a model which we fitted on data from some time period to predict the class labels of transactions that we encounter in the future. Such "abuse" of model-based prediction for temporally successive data is, more accurately, called *forecasting*.

Even if we had a model that perfectly fits $p(y, \mathbf{x})$ for a certain time period, it might be wrong in the future due to time-dependent changes of the reality $p(y, \mathbf{x})$. In supervised classification, we can, to some extent, account for the time dependence of y by adding time specific features to $\mathbf{x}$ but we can't, without further assumptions, account for $p(y, \mathbf{x})$ itself changing arbitrarily. Even though this problem goes under many different names [WK96, QCSSL09], we follow the terminology used by Moreno & Torres [MTRAR+12] who refer to the problem generically as *dataset shift* and further distinguish different kinds of shift. They propose a taxonomy based on the direction of inference and thereby based on which roles the individual factors take in the factorized joint distribution over covariates and class labels

$$p(y, \mathbf{x}) = p(y|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|y)p(y).$$

Covariate shift and *prior probability shift* refer to discrepancies in the marginal distributions between the distribution of the train set p_{train} and the distribution of the test set p_{test}. If the inference direction is from known feature values to class labels $\mathbf{x} \to y$, they call the train/test discrepancy

$$(p_{train}(y|\mathbf{x}) = p_{test}(y|\mathbf{x})) \wedge (p_{train}(\mathbf{x}) \neq p_{test}(\mathbf{x}))$$

covariate shift. If the inference direction is from class labels to feature vectors $y \to \mathbf{x}$, they denote the train/test discrepancy

$$(p_{train}(\mathbf{x}|y) = p_{test}(\mathbf{x}|y)) \wedge (p_{train}(y) \neq p_{test}(y))$$

as *prior probability shift*. A *concept drift* appears in situations when there is a discrepancy in the conditional distributions between the training and the test set, thus when either

$$p_{train}(y|\mathbf{x}) \neq p_{test}(y|\mathbf{x})$$

or

$$p_{train}(\mathbf{x}|y) \neq p_{test}(\mathbf{x}|y)$$

changes while its corresponding marginal remains unchanged. The consequence of any of these discrepancies is that the decision boundary induced by the classification model is no longer optimal for data that has been generated by the test distribution. It depends on the application whether such degradation of predictive performance is still acceptable.

Customers' payments are likely to exhibit both within-group dependencies and temporal dependencies. And the problem of data set shift is a vast topic in its own right, which, if one aims to address it adequately, involves the study of stochastic processes and their properties. We want to highlight that the methods developed in the area of time series analysis are mostly concerned with the analysis of realizations from discrete-time stochastic processes but our problem naturally lives in continuous time. One example of modeling continuous-time event sequences with stochastic processes for item recommendation can be found in [DWHS15] and related papers from the same research group. In this thesis, we do not address the problem of data set shift. We bluntly assume that all transactions within the period covered by our data set are independent and identically distributed under some fixed but unknown distribution. This is equivalent to assuming the non-existence of data set shift.

Even though we ignore the potential violation of independence assumptions and instead employ standard classification algorithms for fraud detection, we aim to make the models' generalization performances across groups and across time periods transparent. Therefore, we are going to discuss two kinds of model evaluation in the next section.

3.5 Availability of labels and model evaluation

Credit card payments are naturally grouped by card holders and ordered in time. As we have discussed in the previous section, we do not account for these structural traits during modeling but we provide performance estimates of our models along two directions. We determine empirically one estimate of predictive performance when the classifier is trained on data coming from one set of card holders and then tested on a second disjoint set of card holders. The second estimate is determined by training the classifier on a certain time period and then testing it on a subsequent time period. In this evaluation we make use of the complete set of card holders but only shift the time window for training and testing. The two directions of evaluation are necessary to determine the source of error for which we did not explicitly account for during modeling. Is it because the style of payment of new users appearing in the test set is different from the styles seen in the training set or is it because the payment style of card holders changes across time? We determine the group-based and time-based performance estimates with cross-validation over several splits, as illustrated in Fig. 3.10.

There is another application-specific peculiarity we have to be aware of. After consulting the card holders, fraud investigators label a subset of transactions as fraudulent or genuine. But they only label those which have been brought to their attention, either

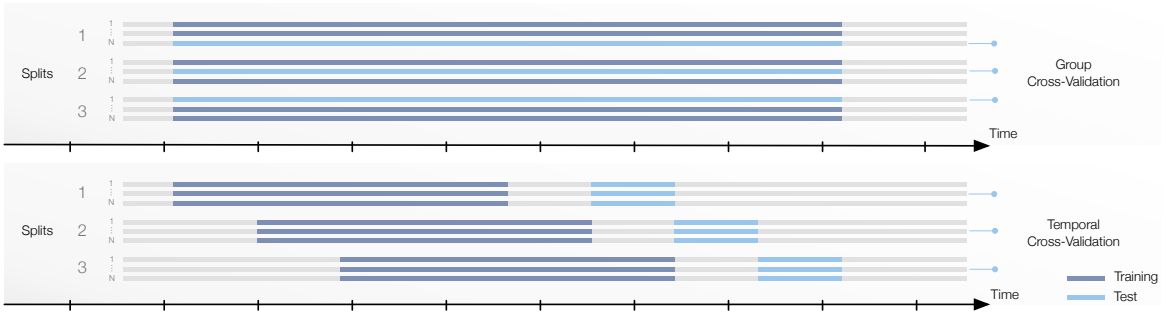


Figure 3.10: Cross validation strategies for model evaluation on temporal sequences. Top: Cross validation across groups of sequences. Each sequence corresponds to one card holder and the elements of that sequence are transactions issued by that card holder. Bottom: Cross validation across time periods. Here, we use transactions from all card holders during training and testing but the time periods differ between the training set and the test set.

by the active detection systems or by the card holders themselves. The vast majority of transactions never undergo a manual inspection but get labeled as genuine by default. This labeling process rarely produces false positives but it might, as we suspect, produce many false negatives. If some label indicates fraud, the corresponding transaction is most likely truly fraudulent. However, if the label indicates legitimacy, we can not have the same confidence in the correctness of that label. Unfortunately, estimating the proportion of false negatives is prohibitive as it requires recovering truly honest and correct feedback from a subset of card holders about all their issued transactions. Under these circumstances we simply have to assume that all fraud labels assigned to transactions in the data set are correct. The data set is considered the ground-truth and a representative sample of both genuine and fraudulent credit card transactions.

When we estimate the forecasting performance, i.e. the predictive performance for transactions that appear after the training period (see Fig. 3.10, bottom), we introduce a gap of several days between the end of the training period and the beginning of the test period. In the real-world detection scenario, the labeling process takes several days and only at the end of it are the ”true” labels available for further analysis. Although there are incidences when the ”true” labels become available much later, Worldline informed us that, in practice, a delay of one week is considered sufficient to expect the majority of incidences to be settled. In order to avoid overly optimistic estimates, we follow this rule of thumb and set the gap between the training period and the test period to one week.

3.6 Random Forest: A baseline for fraud detection

Much work in credit card fraud literature focuses on supervised learning methods and, in particular, on decision trees or random forests [BHPB02]. Pozzolo et al. [DCL+14] compared several frequently used supervised learning algorithms, namely support vector machines, neural networks and random forests, for the task of credit card fraud

detection. They found that random forests generally rank highest in terms of predictive performance measured as average precision and area under the ROC curve across different training set sizes. In order to establish a baseline of the predictive performance of supervised classification methods on our data set, which we can refer to throughout this thesis when we discuss alternative approaches for fraud detection, we use a random forest as baseline classifier. A random forest is a particular instance from the family of meta-learning algorithms, called *Bagging*, which averages the predictions of a set of decision trees.

3.6.1 CART decision trees

Before we introduce random forests, we briefly present the fundamentals of decision trees and, in particular, the algorithm for induction of classification and regression trees (CART).

A decision tree is a non-parametric supervised classification model which consists of a set of cascaded *if-then* rules extracted from a set of labeled observations. The set of rules (i.e. their boolean conjunction/disjunction) can be represented in form of a tree structure with each node corresponding to an *if*-clause and each branch corresponding to a *then*-clause. An *if*-clause within a node compares the value of a feature with a parameter that was assigned to that rule. If the statement evaluates to `true`, we descend the tree along the branch in the *then*-clause to reach a child node. The child node again has an associated rule that checks some feature's value. We proceed traversing the tree recursively in this fashion until we eventually reach a leaf node. Instead of an associated rule, a leaf node contains class counts that have been recorded during training. The class counts in a leaf node represent the number of observations from each class for which the set of rules along the path that lead to the leaf node evaluated to *true*. A majority vote on the class counts then determines the predicted label of a new, yet unseen, observation. Since each path in the tree represents a conjunction of boolean feature value based comparisons, observations that trigger all rules along the path end up in the same leaf node. Ideally we would want to create paths such that only observations from the same class end up in the same leaf node. Then, the majority vote in the leaf node most likely yields the correct label for a new observation.

Therefore, inference for a test observation in a given decision tree means traversing the tree from the root node to a leaf node while following an observation-specific path that is dictated by evaluating the rules along the way. The more interesting question is how we can create such a tree or, equivalently, the set of rules that produce accurate classifications for unseen observations. The creation of a classification or regression tree from a set of observations is called *tree induction*. Research around decision trees has produced a lively collection of tree induction algorithms differing in their assumptions, employed heuristics and pursued objectives. Among the most commonly used methods are CART [BFOS17], ID3 [Qui86] and C4.5 [Qui96, Qui14].

As before, we consider a generic set of training observations

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq n}; \quad \mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})^\top \in \mathbb{R}^d, y_i \in \{1, \ldots, K\}$$

where $\mathbf{x}_i$ is the feature vector of observation i and y_i is the observation's associated

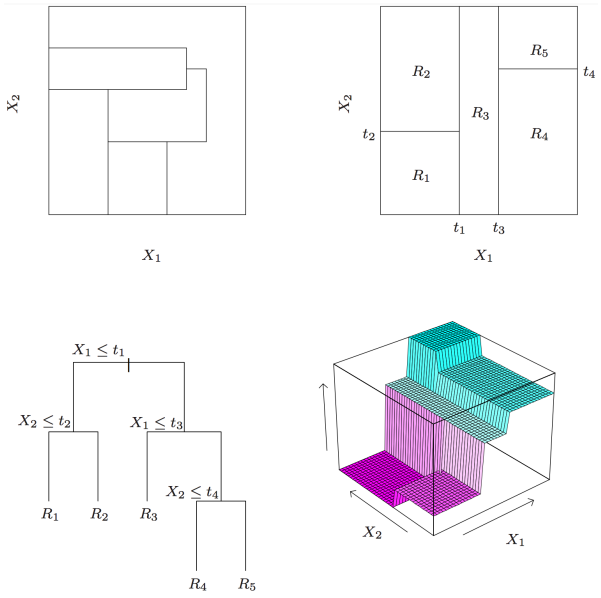


Figure 3.11: Top: Illustration of a 2-dimensional feature space with exemplary partitions. The partitioning in the top left corner can not be represented with a binary decision tree whereas the partitioning in the top right corner can. Bottom left: The binary decision tree corresponding to the partitioning in the top right corner. Bottom right: The function represented by this tree. Each leaf node, or equivalently the partition in the feature space R, has an assigned constant value which is the decision value for all samples falling into that region. The figure is taken from Hastie et al. [HTF09].

discrete label. CART partitions the feature space into M disjoint axis-aligned regions $(R_1, \ldots, R_M)$ and fits a constant model to each region. As an illustration, see Fig. 3.11 for a regression problem with continuous y and a two-dimensional feature space. Tree induction for a classification problem can be illustrated in the same way except that the value assigned to regions in the bottom right figure would have K levels, one for each of the classes.

The tree induction algorithm should both find a partition of the feature space $(R_1, \ldots, R_M)$ and estimate parameters

$$\hat{p}_{m,k} = \frac{1}{n_m} \sum_{\mathbf{x}_i \in R_m} \mathbb{1}_{\{y_i = k\}}$$

for each region m, where we denoted the number of observations per region by $n_m = \sum_{i=1}^{N} \mathbb{1}_{\{\mathbf{x}_i \in R_m\}}$. Parameters $\hat{p}_{m,k}$ are the proportion of observations of class k falling into region m. If the regions and parameters were known, the classification of a generic $\mathbf{x}$ would then be given by

$$\hat{y}(\mathbf{x}) = \underset{k}{\operatorname{argmax}} \sum_{m=1}^{M} \hat{p}_{m,k} \mathbb{1}_{\{\mathbf{x} \in R_m\}} \tag{3.1}$$

However, finding the globally best partitioning in terms of a classification error measure, such as for instance the misclassification rate $\iota(\mathcal{D}) = 1 - \frac{1}{n} \sum_i \mathbb{1}_{\{y_i = \hat{y}_i\}}$, is computationally infeasible. Therefore, the CART algorithm proceeds with a greedy strategy

that, given some region, finds the locally best axis-aligned binary partitioning of that region.

Let $\mathcal{D}_m = \{(\mathbf{x}_i, y_i) \in \mathcal{D} : \mathbf{x}_i \in R_m\}$ be the set of observations falling in region R_m. To each region in the feature space we associate a node in the tree, where the entire feature space corresponds to the root node. For any region R_m, consider the tuple (j, s) of a splitting feature indexed by $j \in \{1, \cdots, d\}$ and a split value $s \in \{\mathbf{x}_{i,j}\}_{1 \leq i \leq n}$. The tuple (j, s) induces a binary partitioning of R_m into the two half-planes

$$R_{m_L} = \{\mathbf{x}_i \in R_m : \mathbf{x}_{i,j} \leq s\} \quad \text{and} \quad R_{m_R} = \{\mathbf{x}_i \in R_m : \mathbf{x}_{i,j} > s\}$$

and correspondingly, a partitioning of the set of observations $\mathcal{D}_m$ into the set of observations associated with the left child node $\mathcal{D}_{m_L}$ and the set of observations associated with the right child node $\mathcal{D}_{m_R}$ of parent node m. Since we are interested in the tuple (j, s) that induces the best partitioning of region R_m in terms of minimization of some classification error, we first need to define what these error measures could be. In the terminology of classification trees, the error measures are commonly referred to as impurity functions $\iota(\mathcal{D}_m)$ due to their purpose of quantifying how well the probability mass of $\hat{p}(y|R_m)$ is concentrated on one single class. Different impurity functions can be considered:

- **Misclassification rate:**
$$\iota(\mathcal{D}_m) = 1 - \max_k \hat{p}_{m,k}$$
- **Gini index:**
$$\iota(\mathcal{D}_m) = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k})$$
- **Entropy:**
$$\iota(\mathcal{D}_m) = -\sum_{k=1}^{K} \hat{p}_{m,k} \ln \hat{p}_{m,k}$$

Since we are seeking a tuple (j, s) that minimizes the impurity of both child nodes, the local objective function to be minimized in node m is given by

$$\underset{j,s}{\operatorname{argmin}}(n_{m_L}\iota(\mathcal{D}_{m_L}^{(j,s)}) + n_{m_R}\iota(\mathcal{D}_{m_R}^{(j,s)}))$$

To emphasize the dependence of the sets $\mathcal{D}_{m.}^{(j,s)}$ on (j, s), we indexed them with the tuple (j, s).

Thus, the two essential mechanics in tree induction are the search for a split that minimizes the impurity measure in each recursion step and the assessment of a split by means of said measure. In practice, the minimization problem is solved by either exhaustively searching or heuristically probing combinations of features j and split values s. One such naive heuristic is to randomly select only a subset of features to consider and then minimize the local objective function w.r.t. the constrained feature set. After having found the minimizer $(j, s)^*$, we can create the two child nodes m_L, m_R, associate them with the corresponding regions R_{m_L}, R_{m_R} and the set of observations $\mathcal{D}_{m_L}^{(j,s)^*}$,$\mathcal{D}_{m_R}^{(j,s)^*}$ and finally apply the algorithm recursively to the nodes m_L and m_R. Thereby, the recursive

partitioning defines the branching structure of the tree and the combination of feature j and threshold s defines the decision rule within each node of the tree. The recursive splitting process is repeated until some stopping criterion is met. Such stopping criteria include purity of a node, the maximum depth of the tree, a maximum number of leaves or a minimum number of observations per leaf node. Any node that runs into one of the stopping criteria becomes a leaf node and its parameters $\hat{p}_{m,k}$ serve as constant model for making classifications for all points within the entire region R_m covered by that leaf node. A classification for one test observation can be obtained by first traversing the tree from the root node to a leaf node along the path dictated by the node-specific rules evaluated on the test observation. Then, the classification for that test observation is derived from the leaf node's parameters according to Eq. (3.1).

The drop in impurity, incurred by splitting the parent node into its child nodes, is called impurity reduction

$$\Delta\iota(\mathcal{D}_m) = \iota(\mathcal{D}_m) - \left(\frac{n_{m_L}}{n_m}\iota(\mathcal{D}_{m_L}) + \frac{n_{m_R}}{n_m}\iota(\mathcal{D}_{m_R})\right) \tag{3.2}$$

where n_{m_L} and n_{m_R} are the number of observations associated with the left and the right child node, respectively. After tree induction, the impurity reductions can be used to quantify the importance of a variable for classification. In case of the Gini index, the importance measure is called the *Gini importance* of a variable j and, for instance, in the machine library `sklearn`[5] it is defined as the weighted sum of the impurity reductions induced by all the nodes $M(j)$ in the final tree, which split on variable j:

$$\widehat{GI}(j) = \sum_{m \in M(j)} |\mathcal{D}_m| \cdot \Delta\iota(\mathcal{D}_m) \tag{3.3}$$

These importance scores are then normalized over all variables such that they sum to one:

$$GI(j) = \frac{\widehat{GI}(j)}{\sum_{k \in \{1,\ldots,d\}} \widehat{GI}(k)} \tag{3.4}$$

Decision trees are non-parametric statistical models and thereby inherently prone to overfitting the observations in the data set. Clearly, the stopping criteria are what turns decision trees into useful models for predictions. If we were to grow the tree to full depth such that each observation has its own dedicated leaf node, the tree collapses to an efficient index structure for the data set it was trained on but no longer exhibits useful potential for generalization to new observations. Therefore, researchers have studied decision tree pruning, a strategy for preventing overly specific sub-trees, to ameliorate this deficiency by integrating pruning heuristics either during or after tree induction. See [BFOS17, ch. 10] or [HTF09, ch. 9-10] for an introduction to pruning in decision trees.

A related characteristic of decision trees is that they are unstable in the sense that they exhibit high variance. Due to the hierarchical structure, a small change in the data can lead to very different trees. The effect of a particular choice of split point at the

[5] `https://scikit-learn.org`, Last access: 31.05.2019.

top of the tree is propagated down to all subsequent splits. Therefore, the set of rules or equivalently the decision boundary represented by the decision tree can vary largely under slightly different sets of training observations. The smaller the tree the smaller its variance but also the weaker it is as accurate predictor. Pruning is one strategy for controlling the variance via regularization. We do not further pursue this direction but instead make use of bagging as an ensemble technique that counters the high variance through averaging the decision boundaries of many different decision trees, which gives rise to the *random forest classifier* developed by Leo Breiman [Bre01].

3.6.2 Random Forest classifier

A commonly used technique in machine learning for creating a robust (low variance) predictor is to build an ensemble of predictors and then average their predictions. Besides *Boosting* and *Stacking*, *Bagging* is an ensemble method for creating a low-variance predictor by averaging the predictions of several *base learners* that have been induced from bootstrapped samples of the available data. If all base learners in the ensemble are of the same kind, the ensemble is called homogeneous. A particular instance of Bagging where all base learners are de-correlated decision trees has been coined a random forest classifier [Bre01].

The predictive performance of an averaged predictor is usually better than that of any single one of the base predictors. To that end we require access to several samples from the population, but since this is typically not the case we use bootstrapping to randomly draw B data sets $\{\mathcal{D}_b\}_{1\leq b\leq B}$ with replacement from the training data $\mathcal{D}$, with each $\mathcal{D}_b$ having the same size as the training data set. Then we learn a predictor $\hat{y}_b(\mathbf{x})$ on each of these B data sets and average their predictions. For classification trees, an average over the predicted classes is not meaningful, therefore a typical choice consists in collecting the votes from all trees and then predict the class the majority of trees have voted for:

$$\hat{y}_{bag}(\mathbf{x}) = \underset{1\leq k\leq K}{\operatorname{argmax}} \left(\frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{\hat{y}_b(\mathbf{x})=k\}} \right) \tag{3.5}$$

The distinguishing characteristic between a "bag of decision trees" and a random forest lies in the additional effort to de-correlate the predictions of the B trees by randomizing their construction. This randomization is introduced in the tree induction process by randomly choosing a set of $m \leq d$ features to consider in each split. Note that this is different from inducing the entire tree from only m features. Here we randomly choose m candidate features at each split and then search for the best splitting feature and split value among the m candidates. The random forest algorithm, as summarized by Hastie et al. [HTF09], is given in Algorithm 3.1.

A convenient by-product of decision trees and thereby also of random forests is the implicit assessment of each feature's importance in terms of the decrease in impurity when using this feature for splitting the data set. At each split in each tree, the improvement in the split criterion induced by the splitting feature counts towards the importance of that feature. These importance scores are accumulated over all trees in

Algorithm 3.1 Random Forest Classifier

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathcal{D}_b$ of size $|\mathcal{D}|$ from the training data $\mathcal{D}$.
   (b) Induce a random forest tree $\hat{y}_b$ from the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until some stopping criterion is reached.
      i. Select m variables at random from the d variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two child nodes.
2. Return the ensemble of trees $\{\hat{y}_b\}_{1 \leq b \leq B}$.

To make a prediction for a new observation $\mathbf{x}$, use Eq. (3.5).

the forest separately for each feature to yield overall feature importance scores over all trees.

Throughout this thesis, whenever we present results of a random forest classifier, we used the implementation in the Python machine learning package `sklearn`.

3.7 Experiments

Since random forests are very robust classifiers with typically strong performance on many tasks [HTF09], including the task we are dealing with in this thesis [DCL+14], while at the same time being easy to train and tune, we present in this section the baseline predictive results obtained with random forests. In particular, we address the following three questions:

- **RQ1**: What is the average difference in predictive performance between predictions on transactions from different users when the time period of train and test is the same versus predictions on transactions from the same users when the time period of train and test is different?
- **RQ2**: How much does the predictive performance vary under training sets with different class imbalance ratios?
- **RQ3**: Is the forecasting performance more strongly influenced by the forecasting horizon or by the identity of any specific day?

To answer these questions, we used the group-wise and temporal cross-validation strategy as described in Section 3.5 and measure the predictive performance by means of the $AUCPR$ and $AUCPR_{0.2}$ as described in Section 3.3. To address the questions related to the forecasting scenario, we used 3-fold temporal cross-validation over the validation period 07.04.2015 - 30.04.2015 for model selection and 17-fold temporal cross-validation over the evaluation period 08.05.2015 - 31.05.2015 for model evaluation. In

each iteration, the training and test window was shifted by 7 days in case of model selection and shifted by 1 day in case of model evaluation.

To address the questions related to predictions over groups of users within a common fixed time period, we considered the entire time period 01.03.2015 - 31.05.2015 for model selection and evaluation but we applied 5-fold *group-wise* cross-validation and the training sets consisted of transactions from 100,000 randomly sampled users. In both scenarios, we used random search to find the best hyper-parameters of a 300-tree random forest classifier within a grid spanned by `max_depth` : $[4, \inf]$, `max_features` : $[1, 3, \sqrt{d}]$, `min_samples_leaf` : $[1, 10, 20]$. These three hyper-parameters serve as stopping criteria during tree induction. `max_features` denotes the maximum number of features to be evaluated during split point selection. Whenever a node runs into one of either `max_depth` or `min_samples_leaf`, the node is not further partitioned. Thereby, `max_depth` denotes the maximum number of nodes on a path from the root node to a leaf node and `min_samples_leaf` denotes the minimum number of observations associated with a node such that it still qualifies as a leaf node. We used $AUCPR_{0.2}$ averaged over the cross-validation splits to score the grid points and we finally selected the grid point with the highest score to train and evaluate a random forest classifier.

In order to address question **RQ2**, we design a simple resampling strategy for creating training data sets with different class imbalance ratios. Each transaction has an ID that identifies the card holder's account. Any account that contains at least one fraudulent transaction is considered *compromised* and all other accounts are considered *genuine*. By fixing some maximum number of accounts G and a desired resampling ratio $r \in [0, 1]$, we sample $r \cdot G$ accounts from the set of compromised accounts and $(1-r) \cdot G$ accounts from the set of genuine accounts. Thereby, we obtain a training set consisting of transactions from G accounts, of which $r \cdot 100\%$ are compromised. In the following figures we display the resampling ratio as account proportions. For instance, a ratio of $r = 0.1$ will be denoted as $10 : 90$. In the experiments we set $G = 1,000,000$.

3.7.1 Results: Account-based resampling

As motivated in section Section 3.1, we report the results on card-present (F2F) and card-not-present (ECOM) transactions separately. Table 3.3 summarizes the results. Looking at Table 3.3, a first observation is that the predictive performance on e-commerce transactions is overall much higher than on face-to-face transactions. This effect is more pronounced in the forecasting scenario (TIME), where we predict on future transactions. We hypothesize that this discrepancy comes mostly from the fact that the fraud ratio is about one order of magnitude larger in the e-commerce data set.

Secondly, in the prediction scenario (GROUP) the scores are generally higher than in the forecasting scenario (TIME) on both e-commerce and face-to-face transactions, despite comparable fraud ratios. This effect is especially pronounced on face-to-face transactions, which suggests that generalization over time is much more difficult for face-to-face transactions. Thirdly, account-based resampling had minor influence on the predictive performance across all scenarios. There is a slight tendency to obtain better predictive performance when we use a moderate resampling ratio.

Based on the results, we can already provide an answer to questions **RQ1**. The aver-

age difference between the mean GROUP scores and mean TIME scores is $\Delta AUCPR = 0.0504$ ($\Delta AUCPR_{0.2} = 0.0064$) in e-commerce and $\Delta AUCPR = 0.1034$ ($\Delta AUCPR_{0.2} = 0.0473$). This amounts to a relative difference of 22% (0.05%) in e-commerce and 64.9% (43.5%) in face-to-face. We conclude that generalization over time is more difficult than over accounts. Regarding question **RQ2**, we state that, given the large standard deviation, we can not conclude that any particular choice of resampling ratio applied to the training set yields clearly better results than any other resampling ratio in terms of predictive performance. Throughout the remainder of this thesis we fix the resampling ratio to a moderate 10:90 proportion.

3.7.2 Results: Forecasting

In this evaluation we study the influence of data set shift on the predictive performance. As discussed in Section 3.4, our prediction problem is slightly more difficult than in other domains, such as image or text classification, where the influence of temporal shift on the target concept can be safely ignored. The predictive performance of a classifier learned on a certain period of time may deteriorate when we use it for predicting the labels of transactions from the future. However, in practice, the sole purpose of an automatic fraud detection system is forecasting, i.e. the prediction on observations from the future and not on observations from the training period. We are particularly interested in the performance gap faced when switching from the regular prediction scenario (same time period during training and testing) to our use-case of forecasting (testing period after the training period). We carry out this analysis by considering several forecasting horizons which we define as the number of days between the end of the training period until the beginning of the test period. If data set shift is a relevant source of error in the short time periods we consider in this thesis, we should be able to identify it by recording the prediction scores for one particular test day under varying forecasting horizons. Or vice versa, we would expect one forecasting horizon to yield a performance that is consistently higher compared to performances of any other forecasting horizon - regardless of the identity of the test day.

For this evaluation we consider an overall evaluation period of 23 days, starting from 8th of May until 30th of May. Repeatedly, we split the data set along the time axis in a 60-days training set and a 7-days test set. Then we fit a random forest classifier on the training data and evaluate its predictive performance on each of the seven test days. Then we shift the entire train-test window by one day and repeat the process. Since each test day participates in up to seven test sets, in the roles of being the 8th up to the 14th day after the training period, we can observe how its prediction score changes as the forecasting horizon increases.

Assuming the data distribution changes gradually over days, we would expect close-by forecasting horizons to be positively correlated and distant forecasting horizons to be negatively correlated in terms of the ranks of their performances. Let us consider the following example: A 8-th day forecast might result in the highest scores on all test days and a 9-th day forecast might result in the second highest scores on all test days, because the data distribution has only changed slightly since the end of the training period. As the forecasting horizons become increasingly disparate (e.g. 8-th day vs. 14-th day), we

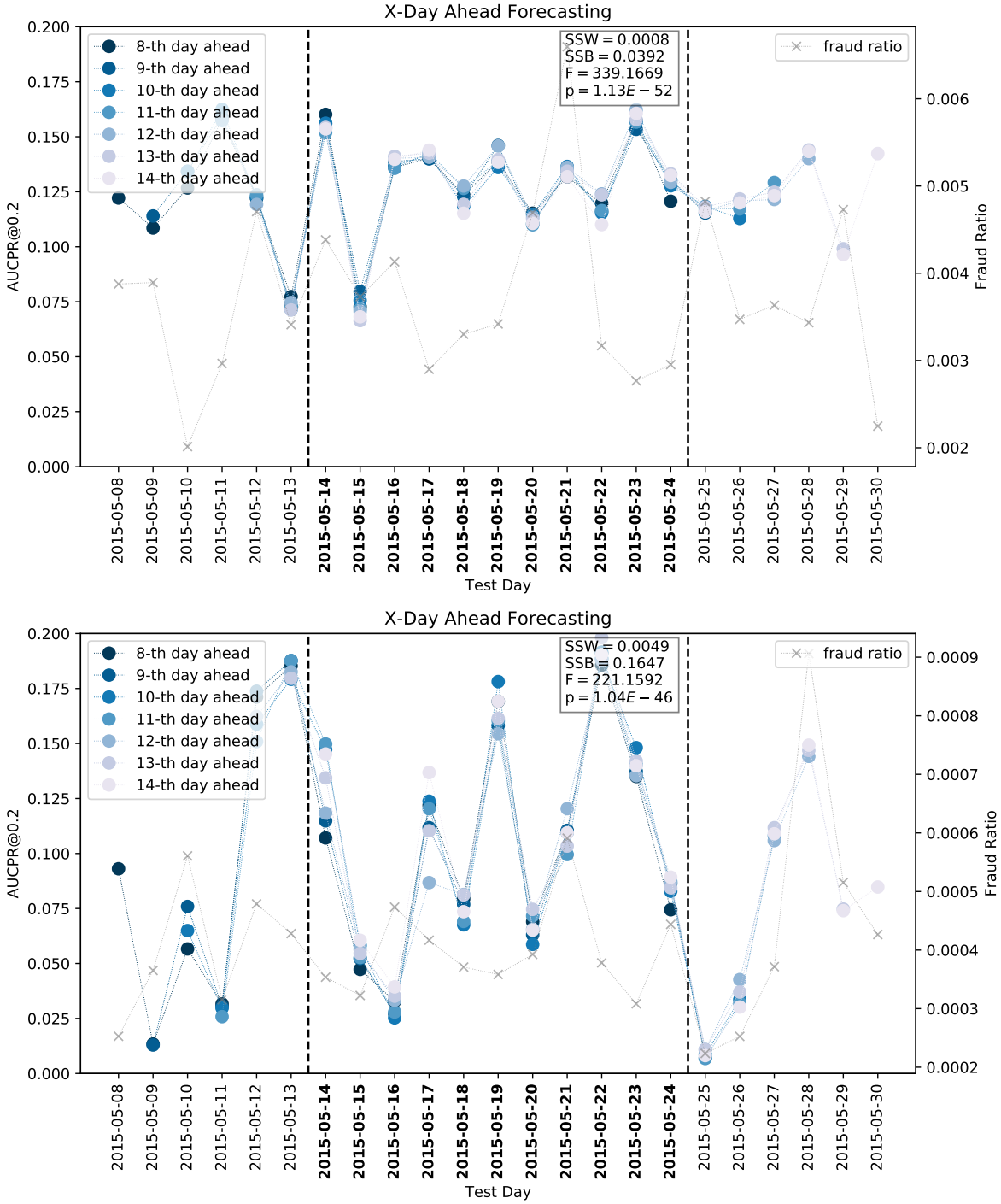


Figure 3.12: Evolution of predictive performance across test days at an undersampling ratio of 10:90. Each point represents the $AUCPR_{0.2}$ obtained on a particular day, when that day is the X-th day after the end of the training period. Top: e-commerce; bottom: face-to-face.

would expect one horizon to exhibit high scores on all test days and the other one to exhibit relatively low scores.

However, as Fig. 3.12 illustrates, there is no consistent ranking of the forecasting horizons over the test days. On some days, the shortest 8-th day horizon yields the highest $AUCPR_{0.2}$, on other days it yields the lowest. The variation induced by the identity of a test day is much larger (ECOM $SSB = 0.0392$, F2F $SSB = 0.1647$) than the variation induced by different forecasting horizons (ECOM $SSW = 0.0008$, F2F $SSB = 0.0049$). Therefore, we answer question **RQ3** by concluding: The predictive performance on a test day is more strongly influenced by the day's identity than by the forecasting horizon. This is by no means a general statement about the presence of data set shift in credit card transactions and fraud, but an indication that gradual data set shift on a daily basis should not be considered the primary source of error in short-term forecasting.

3.8 Summary

In this chapter we introduced the credit card transaction data set as provided by Worldline. We motivated the importance of separating the detection of e-commerce frauds from the detection of face-to-face frauds. We discussed suitable measures for assessing the predictive performance of classification models on our task and introduced group-wise and temporal cross-validation techniques to derive estimates of the predictive performance from hold-out test data. We evaluated the predictive performance of a random forest classifier, which will serve us as a baseline throughout this thesis. The experiments revealed that forecasting is much more difficult than prediction, account-based resampling of the training set has minor effect on the performance and the size of the forecasting horizon is less important than the identity of each individual test day.

Since our primary interest is the prediction of frauds on future, yet unseen, transactions, from now on we will focus on the forecasting scenario. Therein, we fix the training set resampling ratio to a moderate 10:90 proportion at a total of $G = 1,000,000$ user accounts.

Dataset	Evaluation	Sampling	AUCPR $(\hat{\mu}, \hat{s})$	AUCPR0.2 $(\hat{\mu}, \hat{s})$
ECOM	TIME	original	0.193 $\pm$ 0.020	0.118 $\pm$ 0.009
	$n_+ =$ 5187.35 ± 363.77	10:90	0.225 $\pm$ 0.016	0.127 $\pm$ 0.004
	$r_+ =$ $0.00383 \pm 2.1 \cdot 10^{-4}$	20:80	0.216 $\pm$ 0.017	0.125 $\pm$ 0.004
		30:70	**0.263** $\pm$ 0.009	**0.128** $\pm$ 0.005
		40:60	0.213 $\pm$ 0.018	0.122 $\pm$ 0.005
		50:50	0.256 $\pm$ 0.011	0.127 $\pm$ 0.006
	GROUP	original	0.250 $\pm$ 0.027	0.130 $\pm$ 0.016
	$n_+ =$ 5902.40 ± 168.90	10:90	**0.290** $\pm$ 0.029	**0.133** $\pm$ 0.017
	$r_+ =$ $0.00448 \pm 1.3 \cdot 10^{-4}$	20:80	0.287 $\pm$ 0.029	0.132 $\pm$ 0.017
		30:70	0.288 $\pm$ 0.027	0.132 $\pm$ 0.016
		40:60	0.282 $\pm$ 0.031	0.130 $\pm$ 0.018
		50:50	0.271 $\pm$ 0.029	0.128 $\pm$ 0.016
F2F	TIME	original	0.149 $\pm$ 0.019	0.101 $\pm$ 0.013
	$n_+ =$ 709.24 ± 39.58	10:90	**0.167** $\pm$ 0.021	**0.113** $\pm$ 0.014
	$r_+ =$ $0.00040 \pm 2.8 \cdot 10^{-5}$	20:80	0.165 $\pm$ 0.021	0.112 $\pm$ 0.014
		30:70	0.158 $\pm$ 0.019	0.109 $\pm$ 0.012
		40:60	0.155 $\pm$ 0.020	0.108 $\pm$ 0.013
		50:50	0.162 $\pm$ 0.019	0.110 $\pm$ 0.012
	GROUP	original	**0.273** $\pm$ 0.030	**0.160** $\pm$ 0.013
	$n_+ =$ 924.80 ± 74.88	10:90	0.272 $\pm$ 0.019	0.159 $\pm$ 0.008
	$r_+ =$ $0.00052 \pm 4.2 \cdot 10^{-5}$	20:80	0.261 $\pm$ 0.015	0.155 $\pm$ 0.009
		30:70	0.264 $\pm$ 0.014	0.156 $\pm$ 0.008
		40:60	0.258 $\pm$ 0.015	0.156 $\pm$ 0.009
		50:50	0.248 $\pm$ 0.015	0.151 $\pm$ 0.009

Table 3.3: Classification performance of a random forest classifier evaluated across time, groups and varying sampling proportions of the training set. Scores are averages over 17 temporal splits (TIME) or 5-fold cross validation splits (GROUP). n_+ denotes the average number of frauds and r_+ the average ratio of frauds in the test sets. The maximum mean score within each scenario is marked in bold; standard deviation is given in $\pm$ notation.

Chapter 4

Data Augmentation

Parts of the work presented in this chapter have been published and presented at an International and an European conference.
The work on injecting semantic background knowledge in fraud detection classifiers (Section 4.2) was published in the proceedings of the IEEE International Conference on Enabling Technologies [ZCG+17].
The study of robustness of word embeddings against post-processing methods (Section 4.3) was published in the proceedings of the European Conference on Information Retrieval [JGS16].

The goal of data augmentation is to increase the value of existing data by adding information derived from internal or external sources. Augmentation can be performed on a record level or on a feature level. In data-driven applications, the quality of the data determines the usefulness and success of the applications that are built upon it. Being the most valuable asset, data is usually also a scarce resource. Record level augmentation aims at enriching the base data with records obtained under the additional effort of collecting more first-hand data or by consulting third-party sources. For instance, in image classification, augmentation often refers to the task of artificially duplicating images by applying transformations such as rotation, translation or cropping to the original images - or by adding different levels of random noise to the pixels. Such artificial duplication can be performed easily on image data without having to worry too much about whether the transformations will eventually compromise the semantics of the scenery and the objects displayed in the image. In our domain it is virtually impossible to create artificial but yet valid genuine or fraudulent transactions, as we have no simple means to tell which transformation would create valid records while at the same time preserving the transaction's semantics in terms of its fraudulent character.

Therefore, feature level augmentation appears to be a more promising strategy for credit card fraud detection as it enables us to probe the fraudulent or genuine character of transactions directly by including either manually crafted or automatically derived features in the classification pipeline. Additional information can help provide more in-depth insight for humans but likewise enhance accurate classification if the added information permits discrimination between the classes. Features, derived in an unsupervised fashion, can reduce the manual intervention required to develop meaningful indicators. Moreover, if we can extract features that are useful for fraud detection from task-independent external public sources, we are able to provide additional insight into the business data and we effectively transfer knowledge between domains.

In the first section of this chapter we introduce feature engineering strategies as they have been proposed in related work for the domain of credit card fraud detection over recent years. After manual inspection of the dataset and after consulting Worldline we can confirm that such hand-crafted features cover well the known fraudulent activity as observed by fraud investigators. The design of these features is driven by the motivation of maintaining an activity record for a card holder's account which aggregates his recent purchases over some fixed time interval. As we shall see shortly, features which encode some notion of temporal co-occurrence are reliable indicators of fraudulent activity. In the second part of this chapter, we investigate another set of feature candidates. Instead of manually designing features for our task, we aim to leverage publicly available knowledge sources by linking them to our transactional data set through features. In our study, we review sources containing statistics about the social and economic status of countries as a direct means for augmenting our dataset. In addition, we consider a text corpus and a semantic knowledge graph as sources that contain highly unstructured and structured complex knowledge, respectively. The integration of this *external knowledge*, requires the identification of possible connecting attributes and the representation of the external knowledge in form of features. To that end, we will discuss the representation of complex external knowledge, such as text and semantic concepts represented in the knowledge graph, by making use of vector embeddings, a well-established vehicle in natural language processing for transferring knowledge about semantic concepts from one domain to another. In this chapter we address the following questions:

- **RQ1**: Regarding the feature engineering strategies proposed in literature, what is the performance increase we can expect from these features on our dataset?
- **RQ2**: Which attributes are suitable candidates to link external data to and which external sources can we exploit?
- **RQ3**: How can we represent implicit knowledge contained in complex external sources (text, semantic knowledge graph) in form of features?
- **RQ4**: Can features extracted from external sources improve the fraud detection performance?

4.1 Domain knowledge

Due to international reporting standards in the financial sector, the set of raw transaction features is similar across multiple studies and several studies use only these raw features for conducting their analyses [BLH99, MN16b, SVCS09]. However, as we have seen in related work (see Section 2.2), a characterization of the interplay between a transaction and its temporal neighborhood seems to be a promising strategy. Especially for the detection of behavioral frauds, which are unsuspicious in terms of a transaction's attributes but become suspicious when contrasted with the card holder's recent activity, the card holder's transaction history seems to be the appropriate *context*. In this regard, we define the following card holder level features:

Time Delta Based on our discussions with experts from the domain and the temporal succession patterns presented in Fig. 3.7, we aim to capture the closeness of two consecutive transactions in time. Since frauds tend to appear in bursts of several transactions within a short time window, the time difference between consecutive transactions seems to be a promising indicator of fraud. Let $(\mathbf{x}_t)_{t\geq 1}$ be the temporally ordered sequence of transactions from a single account (card-holder) indexed by t. We denote the value of a particular variable in a transaction by superscript, e.g. $\mathbf{x}_t^{(\text{Time})}$ is the time stamp associated with the t-th transaction $\mathbf{x}_t$. Then, for any two consecutive transactions $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ with their time stamps denoted by $\mathbf{x}_{t-1}^{(\text{Time})}$ and $\mathbf{x}_t^{(\text{Time})}$ we define the `tdelta`-feature of the t-th transaction as:

$$\text{tdelta}_t = \mathbf{x}_t^{(\text{Time})} - \mathbf{x}_{t-1}^{(\text{Time})} \tag{4.1}$$

Feature Aggregates An elegant way to extract information from a card holder's history is to aggregate the values of a variable along the transaction sequence. The value of the new feature is computed with an aggregation function applied to a subset of the most recent transactions. For constructing such feature aggregates, we follow the procedure that was recently proposed by Bahnsen et al. [BASO16]. The calculation of one feature aggregate consists in grouping the transactions made during the last given number of hours, first by account, then by transaction type, merchant group, country or any other categorical variable, followed by counting the number of transactions or the total amount spent in these transactions. Since the feature aggregate is defined relative to a single pivot transaction, the value of the aggregate is specific to that pivot transaction. This simple and yet powerful procedure can be considered the state of the art of feature engineering in credit card fraud detection. To each transaction, we add several such feature aggregates differing in the time window over which we aggregate and the choice of categorical variables. Several feature aggregates together form an activity record of an account where the activity record gets updated with each new transaction from that account. In most of the studies, the activity record does not accumulate information from the account's entire history but only from the most recent past, such as the past 24 hours. The hope is that by having both an up-to-date activity record and the raw features, it is easier to determine whether the current transaction is abnormal

with respect to the recent activity within the account.

However, as Bahnsen et al. note, there is a downside to these features: "When implementing this framework on a production fraud detection system, questions regarding response and calculation time of the different [aggregation] features should be addressed. In particular, since there is no limit on the number of features that can be calculated, a system may take too long to make a decision based on the time spent recalculating the features with each new transaction."[BASO16, p. 8]. We are going to address this problem in Chapter 6.

Based on a single pivot transaction $\mathbf{x}_k$, we select a subset of transactions from the past t_h hours according to the sets of categorical variables $\{\}$, $\{A\}$ or $\{A, B\}$:

$$\begin{aligned} \mathcal{S}_k^{\{\}} &= \{\mathbf{x}_i | hours(\mathbf{x}_i^{(Time)}, \mathbf{x}_k^{(Time)}) < t_h\} \\ \mathcal{S}_k^{\{A\}} &= \{\mathbf{x}_i | (hours(\mathbf{x}_i^{(Time)}, \mathbf{x}_k^{(Time)}) < t_h) \wedge (\mathbf{x}_i^{(A)} = \mathbf{x}_k^{(A)})\} \\ \mathcal{S}_k^{\{A,B\}} &= \{\mathbf{x}_i | (hours(\mathbf{x}_i^{(Time)}, \mathbf{x}_k^{(Time)}) < t_h) \wedge (\mathbf{x}_i^{(A)} = \mathbf{x}_k^{(A)}) \wedge (\mathbf{x}_i^{(B)} = \mathbf{x}_k^{(B)})\} \end{aligned} \tag{4.2}$$

The function $hours(\cdot, \cdot)$ takes two time stamps as arguments and returns the time difference between them in hours. Each set $\mathcal{S}_k$ contains all transactions from the previous t_h hours before x_k, where all categorical variables assumed the same value as in $\mathbf{x}_k$. The categorical variables and the time horizon t_h can be seen as constraints imposed on the subset. For instance, if we define $A \coloneqq Country$, $B \coloneqq Merchant$ and $t_h = 24$, the subset $\mathcal{S}_k^{\{A,B\}}$ contains all transactions from the past 24 hours which were issued in the same country and at the same merchant as transaction $\mathbf{x}_k$. Likewise, we can employ other categorical variables by adding further identity constraints to the boolean conjunction.

Now we can calculate an aggregate on $\mathcal{S}_k$. There are many aggregation functions that we could apply to $\mathcal{S}_k$. And we can think of many choices for the *constraint* variables A, B - or consider even more than only two constraint variables and also all their combinations. All these choices might be equally valid, however, for now we restrict ourselves to the setup proposed in the cited work [BASO16], which uses up to two constraints and the number of transactions as well as the total amount spent as aggregation functions:

$$\text{sum}_{\mathcal{S}_k} = \sum_{x \in \mathcal{S}_k} x^{(Amt)} \tag{4.3}$$

$$\text{count}_{\mathcal{S}_k} = |\mathcal{S}_k| \tag{4.4}$$

Each choice of a set of categorical variables, a time window and an aggregation function yields a feature aggregate. We calculate the count-feature aggregate and the sum-feature aggregate for each element in power set of {*country*, *merchant code*, *card entry mode*}, except for the case where all three variables form a constraint. As in the original paper, we fix the time horizon t_h to 24 hours. Finally, we append the resulting 14 feature aggregates to the feature vector of transaction $\mathbf{x}_k$.

By design, these features are inherently account-specific and up-to-date in terms of the user's most recent activity. Due to the identity constraints (see Eq. (4.2)) they are also transaction-specific in the sense that the aggregation applies only to those transactions in the past which are identical to the current one in terms of the specified

categorical variables. This selection of past "matching" transactions is related to a nearest neighbor search around the pivot transaction with the difference that we don't select a fixed number k of neighbors under a generic distance measure but instead we select an arbitrarily large set of neighbors whose boolean conjunctions in Eq. (4.2) evaluate to true. Thereby, the definition of one feature aggregate determines both the terms in the boolean conjunction responsible for selecting a neighborhood and the aggregation function applied to transactions in the neighborhood.

Results In order to evaluate the influence of domain knowledge features on the fraud detection performance, we define the following feature sets:

- `BASE`: The set of raw features
- `TDELTA`: The set of raw features plus the `tdelta` feature
- `AGG`: The set of raw features plus the feature aggregates
- `AGG` + `TDELTA`: The `AGG` feature set plus the `tdelta` feature

As experimental setup we used the random forest classifier and a training set resampling ratio of 10:90 as discussed in the previous chapter. The evaluation scores are cross-validated averages over five successive temporal splits of daily $AUCPR$ and $AUCPR_{0.2}$ scores. The results are summarized in Table 4.1.

We can notice a small improvement of the predictive performance after adding the time delta feature to the raw features in both the e-commerce and the face-to-face scenario. As expected, feature aggregates improve the predictions by a large margin in e-commerce. In face-to-face we can not observe any difference compared to only adding the time delta feature alone.

Dataset	Feature Set	AUCPR $(\hat{\mu}, \hat{s})$	AUCPR0.2 $(\hat{\mu}, \hat{s})$
ECOM	BASE	0.2277 $\pm$ 0.0158	0.1294 $\pm$ 0.0031
	TDELTA	0.2616 $\pm$ 0.0161	0.1355 $\pm$ 0.0025
$n_+ =$ 5098.00 $\pm$ 470.24	AGG	0.3977 $\pm$ 0.0183	0.1585 $\pm$ 0.0025
$r_+ =$ 0.00376 $\pm 2.7 \cdot 10^{-4}$	TDELTA + AGG	**0.4068** $\pm$ 0.0180	**0.1602** $\pm$ 0.0033
F2F	BASE	0.1639 $\pm$ 0.0169	0.1095 $\pm$ 0.0115
	TDELTA	0.2090 $\pm$ 0.0158	**0.1299** $\pm$ 0.0116
$n_+ =$ 706.33 $\pm$ 48.03	AGG	0.2030 $\pm$ 0.0062	0.1162 $\pm$ 0.0070
$r_+ =$ 0.00040 $\pm 1.0 \cdot 10^{-5}$	TDELTA + AGG	**0.2095** $\pm$ 0.0079	0.1155 $\pm$ 0.0095

Table 4.1: Classification performance of random forest classifier evaluated across time and over different choices of feature sets. Scores are daily $AUCPR/AUCPR_{0.2}$ averaged over five temporal splits along time. n_+ denotes the average number of frauds and r_+ the average ratio of frauds in the test sets. The train set resampling ratio was set to 10:90. The maximum score within each scenario is highlighted in bold.

Given this magnitude of improvement, the explicit summary of a transaction's immediate past in form of feature aggregates seems to contain valuable information for

discriminating fraudulent e-commerce transactions from genuine ones. The summary of a transaction's immediate past is order-agnostic, meaning that exchanging any two transactions in the neighborhood does not change the value of the feature. In Chapter 5, we will discuss an alternative approach for deriving an activity record by means of latent states of a recurrent neural network.

4.2 External knowledge

Most of the attributes of a transaction have clear and seemingly rich semantics, such as the gender of a card holder, his country of origin or the type of merchant the card holder interacts with. Since most attributes are also of categorical type, their values have to be treated as if they would be entirely independent and not exhibit any relation to one another. However, this is definitely not the whole truth. As humans, we can easily equip genders or country names with associations to other semantic concepts based on our real life experience. Different genders might have different preferences for purchasing certain goods and services and different countries might very well be geographic neighbors that share the same currency, legal standards or cultural traditions. Any classification algorithm would implicitly group such seemingly different values if the response variable, i.e. the indicator whether a transaction is fraudulent, is constant over all elements in these virtual groups and thereby encode some notion of similarity between categorical attribute values.

By extracting features from external knowledge bases, we aim to provide a more reasonable encoding of similar categorical values a-priori with the hope to improve the fraud detection performance. For that purpose, we first identify possible entry points for external data and then discuss the created features in subsequent sections. The attributes of a transaction can broadly be categorized into the following three categories and we select one entry point in each category:

Card holder location: The account attributes can be distinguished into descriptive attributes of the card holder and the card itself. Card-specific attributes such as the credit network, e.g. VISA or MasterCard, the credit limit or the issuing bank are less interesting for augmentation simply because it is difficult to obtain public data that could be linked to these attributes. Hence, we consider attributes that describe the card holder as a more promising route on which we might be able to find external data in public sources. Card holder attributes constitute a coarse-grained demographic profile of the card holder including his age, sex, address and preferred language. The address is further subdivided into country, city and zip code but both city and zip code are rather noisy and would require manual alignment of their values. Due to the clear semantics as given by the ISO country code tables, we consider the *card holder's country* as promising entry point, such that we can augment transactions with external data.

Merchant location: Similar to the account attributes, we also have access to a few attributes that describe the merchant. Unfortunately, apart from the ID of the terminal, we only have access to the merchant category code and the country where the merchant

is registered in. As shown in Section 3.1, there is no global complete definition of all MCC codes but rather each bank enjoys a certain level of freedom in assigning merchants to their own proprietary MCC code tables. Therefore, we follow the same reasoning as with the card-holder's country and select the *merchant's country* as entry point for augmentation with external data.

Time: Each transaction has an associated time stamp and there are no missing values in this attribute. The time stamp is the time in Belgium when the transaction enters Worldline's processing pipeline. Time is a convenient attribute to link data to because of its unique semantics and the abundance of temporally indexed publicly available data. Basically, many kinds of calendar events can serve as indicators of abnormal purchase behaviour and may therefore be useful predictors of fraud. For instance, seasonal sales, release dates of popular products or public holidays are likely to bear valuable information about the spectrum of purchases we are going to see. In our analysis, we consider the *date* as entry point for time-based external data.

So far, we identified the card holder's country, the merchant's country and the date as potential entry points for information from external sources. By external sources or external knowledge bases we refer to any publicly available data that does not have any direct connection to credit card transactions or fraudulent behavior.

We aim to exploit basic demographic statistics about countries themselves and measures of how these countries are related in terms of the usage of the country's name in a corpus. We call the basic demographic statistics *explicit knowledge* because they can be used directly in form of additional features and we call the measures of relatedness *implicit knowledge* because we need to mine the relations by means of a representation learning method before we can then use the representations as additional features. To this end, we encode countries as points in a continuous feature space such that the distances between points reflect semantic similarities in terms of the word co-occurrences.

A central question is: Why would basic demographic statistics about countries or representations of countries be useful features for classifying transactions? The country is a categorical variable with more than a hundred levels. The levels of the variable are encoded as integers, however the assignment of levels to integers is arbitrary. The classifier then needs to figure out how one country is related to all other variables and finally how the combination of all these variables can best predict the classes. The choice of classification algorithm determines the functional form of this combination - a linear combination, a composition of non-linear transformations in neural networks or piecewise constant functions in decision trees. The family of functions of a particular form is indexed by a parameter vector. And learning means estimating a particular choice of parameters such that the corresponding function minimizes some error criterion between the data and the function evaluated at the data points. Since the variable is categorical, there are disjoint subsets of parameters assigned to the levels of the variable. The parameters of each of these levels can only be estimated from those data points where the variable assumes that level. If some levels occur rarely in the data, the estimates of their parameters can be poor.

When we now "encode" the levels of the country variable with demographic statistics such as the size of its population, the gross domestic product, etc., we effectively create new variables that are at least ordinal and thus permit reasonable groupings of countries with similar demographic statistics. The parameters of countries within the same group can then be estimated from data points that exhibit any one of the countries from the group, which leads to better estimates of the parameters and hopefully to a better generalization on a hold out test set. We do not group or discretize the new demographic variables explicitly but leave it up to the classification algorithm to figure out some sort of implicit grouping that is suitable for the task. Certainly, the additional features can only then be beneficial for fraud detection if groups of countries with similar demographic statistics result in groups of transactions with higher similarity regarding other variables. Since a reasonable ordering of countries is more important for us than having highly accurate absolute demographic statistics, we collected different types of data from several online sources.

4.2.1 Demographic statistics

We extract the following country-based features from databases provided by World Bank (`worldbank.org`), Transparency International (`transparency.org`) and the crowd-sourcing platform Numbeo for wordwide living conditions (`numbeo.org`). The features can be linked to both location entry points: the card-holder's country and the merchant's country.

Population: The total population in a country measured in millions of people. It is based on the de facto definition of population, which counts all residents regardless of legal status or citizenship[1].

Gross Domestic Product: The gross domestic product measured in billions US dollar. The GDP at purchaser's prices is the sum of gross value added by all resident producers in the economy plus any product taxes and minus any subsidies not included in the value of the products[2]. The dollar figures are calculated using single year official exchange rates.

Crime Index: The Crime Index is an estimation of the overall perceived level of crime in a given country measured on a scale from $[0, 100]$, where higher numbers represent higher perceived levels of crime. The data comes from visitors of the platform `numbeo.org` and it is collected in form of surveys in which participants are asked to express their agreement towards statements such as the safety at night, worries about robberies, drug abuse, etc on a 5-point scale from $[-2, 2]$. The exact details regarding the design of the survey and the calculation of the index are available on the website[3]. Due to crowd-sourcing and the lack of a review panel, the data might exhibit a strong

[1] `https://data.worldbank.org/indicator/SP.POP.TOTL`, Last access: 30.06.2019.
[2] `https://data.worldbank.org/indicator/NY.GDP.MKTP.CD`, Last access: 01.07.2019.
[3] `https://www.numbeo.com/crime/rankings_current.jsp`, Last access: 01.07.2019.

unknown bias. We use the data anyway because we are more interested in a global ranking of countries and less in perfectly accurate absolute numbers.

Cost of Living Index: The Cost of Living Index is a relative indicator of consumer goods price, including groceries, restaurants, transportation, utilities and rent in the city compared to New York City[4]. Similar to the crime index, the data comes from surveys answered by visitors of the site and should be treated with the same care.

Quality of Life Index: The Quality of Life Index is an estimation of the overall quality of life including estimations of purchasing power, pollution, house price to income ratio, cost of living, safety, health care, traffic commute time and climate[5], measured on a scale $[0, 100]$. Similar to the crime index, the data comes from surveys answered by visitors of the site and should be treated with the same care.

Corruption Perceptions Index: The Corruption Perception Index is a country's or territory's score that indicates the perceived level of public sector corruption on a scale from 0 (highly corrupt) to 100 (very clean)[6]. The score draws on data sources from independent institutions specializing in governance and business climate analysis such as the World Economic Forum[7], Bertelsmann Foundation[8] and IHS Global Insight[9]. The full list of data sources is available from Transparency International's website[10].

All these features are numerical and we standardize them to zero mean and unit variance.

Apart from the country-based features we also add a time-based feature. More specifically, we add a public holiday feature that links the country of a transaction to the date of the transaction. Public holidays are interesting in the context of credit card fraud detection, because the card holder's behavior is expected to change on public holidays. Therefore, knowing if a transaction takes place on a public holiday or not could be an informative feature for our classification task. In this context, we assess the performance gain induced by the new feature:

Public Holiday in a Country: A boolean attribute indicating whether the transaction date is a public holiday in a given country. We extract the respective information from 83 publicly available calendars provided by the Thunderbird and Mozilla Foundation[11]. Due to the coarse resolution of locations based on countries, the holiday feature is not very precise but has high recall: The holiday feature is set to `true` whenever there is any holiday at that date in the entire country. Different regions within a country can

[4] `https://www.numbeo.com/cost-of-living/`, Last access: 01.07.2019.

[5] `https://www.numbeo.com/quality-of-life/rankings.jsp?title=2015`, Last access: 01.07.2019.

[6] `https://www.transparency.org/cpi2015/#results-table`

[7] `https://www.weforum.org/reports`, Last access: 01.07.2019.

[8] `https://www.bertelsmann-stiftung.de/de/unsere-projekte/sustainable-governance-indicators-sgi/`, Last access: 01.07.2019.

[9] `https://ihsmarkit.com/industry/economics-country-risk.html`, Last access: 01.07.2019.

[10] `https://www.transparency.org/cpi2015#downloads`, Last access: 01.07.2019.

[11] `https://www.thunderbird.net/en-US/calendar/holidays/`, Last access: 01.07.2019.

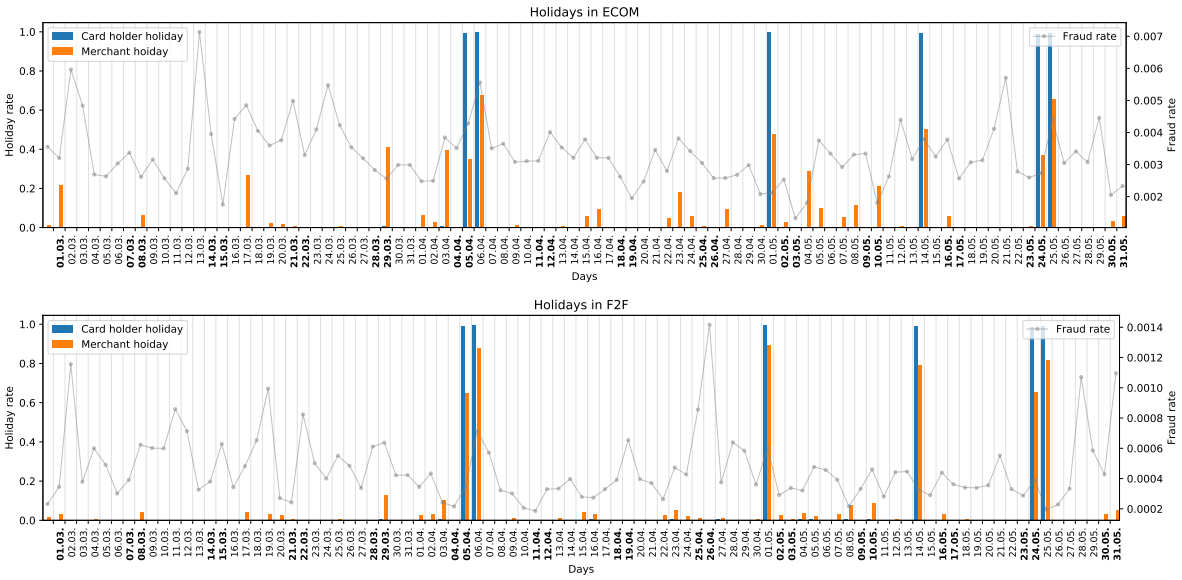


Figure 4.1: Occurrence of public holidays over all days in the data set. Holiday rate is the fraction of transactions that occur in a country where the day is a public holiday. For instance, a card holder holiday rate of 0.9 at a particular day means that 90% of all transactions of that day were issued by card holders who's country happens to be on holidays on that day. The fraud rate is the fraction of fraudulent transactions over all transactions of a day.

have different holidays. We did not account for these difference but instead collected all holidays and treated them as if they were country-wide holidays.

All the above-mentioned features were linked to both the card holder's country - the registered address of the card holder, and the merchant's country - the registered address of the merchant, leading to a total of $2 \cdot 7 = 14$ explicit external features. As mentioned in Chapter 3, the merchant's country is actually the country where the POS terminal is registered. Unfortunately, this is not necessarily the same. Both in face-to-face and in e-commerce, the company can be registered in one country whereas their terminals can be registered in various other countries due to the company's internal business units' structure. We bluntly ignore this discrepancy in our analysis and hope that most of the terminals are registered in the same country as the operating merchant.

4.2.2 Word embeddings derived from knowledge bases

In the following sections, we explore possibilities to encode the levels of a categorical variable with features derived from knowledge bases. Our goal is to find compact representations of countries which encode the semantic concept of a country. Since the country's name is only a distinct identifier that does not incorporate any notion of relatedness to other countries, we aim to create a more meaningful set of features by encoding not the name of the country but the country's semantic concept. For instance, the country name "Germany" is only one of many properties of the semantic concept `<Germany>`. Since the semantic concept covers the entire spectrum of agreed meaning,

it has various relations to other semantic concepts. We could think of relations to semantic concepts like `<Europe>`, `<Currency>`, `<BMW>`, `<Politician>`, `<SecondWorldWar>`. Semantic concepts and their relations are typically structured in form of ontologies with strictly typed classes. However, since we are not interested in inference over concepts but only in the encoding of concepts, knowledge sources with less structure might also serve as sufficiently accurate proxy for learning about semantic concepts. The usage of words in natural language and, in particular, the co-occurrence with other words frames the agreed meaning of words - their semantic concepts. In information retrieval or linguistics such co-occurrence statistics have been exploited ever since to transition from surface forms of words to their semantic concept. In a more general sense, this discrepancy is also known as the *semantic gap*. We can imagine that a compact co-occurrence based encoding of the semantic concept provides a much richer description of a country than an integer encoding of its name. We shall see in this and the following sections an unsupervised method that allows us to encode the semantic concept behind a word by mapping it to a point in a real-valued "semantic" vector space. The method embeds all words from a corpus in the semantic space in such a way that distances in the space reflect similarities of semantic concepts. By encoding a country with the coefficients of its word embedding vector, we hope to observe similar effects on the predictive performance in fraud detection as by adding explicit demographic statistics. In computational linguistics, generating count-based language models has been an active research area since decades. The most common approach involves three parts: Collecting co-occurrence statistics of words from large text corpora, transforming (e.g. tf-idf, Pointwise Mutual Information (PMI)) the counts to derive word association scores and finally applying a dimensionality reduction method (e.g. PCA, SVD). Dimensionality reduction is used for both smoothing sparsity and reducing the overall amount of parameters in order to obtain a low-dimensional and dense embedding matrix [TP10]. Advances in recent years gave rise to new techniques [BDVJ03, CWB+11a, MH08, HSMN12], that implicitly model word co-occurrences by predicting context words from observed input words. Instead of first collecting co-occurrences of context words and then re-weighting these values, predictive approaches treat the word vectors as parameters and estimate them directly to optimally predict the contexts in which the corresponding words tend to appear. In an extensive evaluation, Baroni et al. [BDK14] conclude that embeddings from predictive models are superior to their count-based counterparts on word similarity tasks.

Embedding is a collective name for a set of models and feature learning methods where any object is mapped to a vector of real numbers in a low-dimensional space. Methods for embedding words have been studied extensively in literature [BDVJ03, CWB+11a, MSC+13, PSM14]. In the following sections we focus on the Skip-gram method which is part of the Word2Vec toolkit [MCCD13, MSC+13, MYZ13][12], a group of algorithms for creating embeddings of words from large corpora.

Embeddings computed with the Skip-gram method have the convenient characteristic to exhibit linear structure which makes it possible to conduct analogical reasoning on words via simple vector arithmetic. In their evaluations, the authors could show

[12] `https://code.google.com/archive/p/word2vec/`, Last access: 01.07.2019.

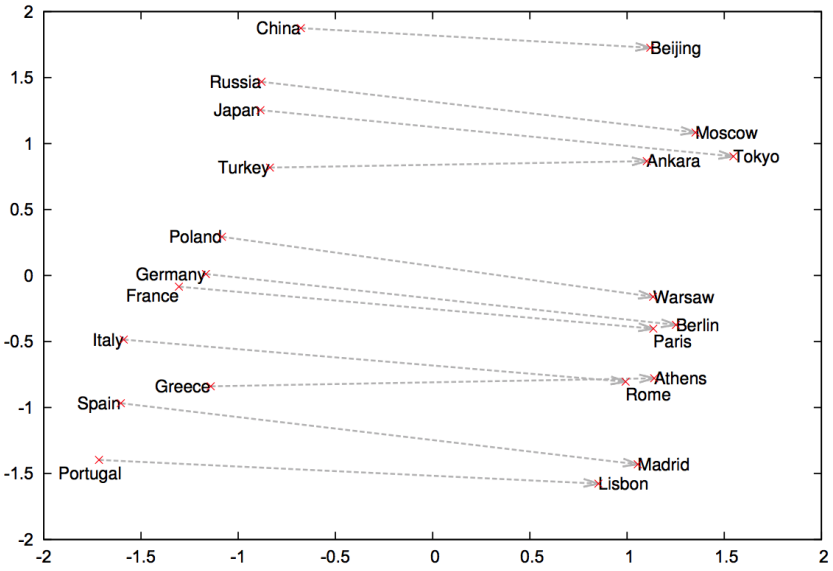


Figure 4.2: A two-dimensional PCA projection of 1000-dimensional Skip-gram word vectors of countries and their capitals calculated on the Google News corpus [MSC+13].

that additions and differences on word vectors accurately reflect semantic and syntactic operations on words [MSC+13]. We chose to use these word representations in our experiments, since they encode a variety of language related information. Figure 4.2 shows a low-dimensional projection of Skip-gram vectors and illustrates the model's capability to organize words and their relationships based on textual appearances. Since the method is unsupervised it relies solely on a corpus of sequentially arranged objects, e.g. words or names of semantic concepts. Skip-gram repeatedly draws two kinds of pairs randomly from the corpus: Pairs of objects that co-occur within some neighborhood in the corpus and pairs of object that do not co-occur within some neighborhood in the corpus.

The method does not postulate a language model per-se but rather an objective function whose optimization leads to "good" embeddings. There is no clear definition of what makes "good" word embeddings, but a widely accepted description seems to be that two words that co-occur with the same set of surrounding context words should get assigned similar word vectors [GL14]. This hypothesis originates from work in distributional semantics where John Rupert Firth popularized the expression that "a word is characterized by the company it keeps" [Fir57].

The original Skip-gram paper is rather cryptic in its motivation for the proposed objective and why the optimization leads to good embeddings. Consequently, several researchers tried to unravel the underlying rationale and work out the connections to well-known parameter estimation techniques [Dye14, LG14, GL14]. The basis of Skip-gram is a model $p(c \mid w; \theta)$ of the conditional probabilities of context words c given pivot words w. The goal is to estimate θ so as to maximize the likelihood of a data set of word-context pairs $\mathcal{D} = \{(w, c)_i\}_{1 \leq i \leq N}$ under the model. Here, a *context word* is any word that appears no more than a fixed number of words before or after the pivot word w in the corpus. Two relaxations on the definition of the pivot word's context make the estimation procedure particularly efficient. First, the context words are chosen from a

small (usually between 5-10) window to the left side and the right side of a single pivot word. And secondly, the model ignores the order of context words around the pivot, which is an unreasonably strong restriction for holistic language models but is sufficient for learning word representations. The likelihood function for the skip-gram model is given by:

$$\mathcal{L}_L(\theta; \mathcal{D}) = \prod_{(w,c)\in\mathcal{D}} p(c \mid w; \theta)$$

And the word-context model itself is defined using the soft-max function:

$$p(c \mid w; \theta) = \frac{e^{\mathbf{v}_c^\top \mathbf{v}_w}}{\sum_{c'\in C} e^{\mathbf{v}_{c'}^\top \mathbf{v}_w}} \tag{4.5}$$

where $\mathbf{v}_c, \mathbf{v}_w \in \mathbb{R}^d$ are parameter vectors for words c and w, respectively. The entire collection of all parameters θ includes the coefficients from both all pivot word vectors $\mathbf{v}_w, \forall w \in V$ and context word vectors $\mathbf{v}_c, \forall c \in C$. The word vocabularies $V = \{1, 2, \ldots, N_V\}$ and $C = \{1, 2, \ldots, N_C\}$ are usually identical but the parametrization of their word vectors is different. The log-likelihood is then:

$$\mathcal{L}_{LL}(\theta; \mathcal{D}) = \sum_{(w,c)\in\mathcal{D}} \left\{ \mathbf{v}_c^\top \mathbf{v}_w - \log \underbrace{\sum_{c'\in C} e^{\mathbf{v}_{c'}^\top \mathbf{v}_w}}_{Z_\theta(w)} \right\}$$

The partition function $Z_\theta(w)$ is expensive to compute as we have to marginalize over the entire context word vocabulary for each pivot word w in the corpus. For making the computation of $Z_\theta(w)$ more tractable, Mikolov et al. introduce the *negative sampling* objective:

$$\mathcal{L}_{NegSam}(\theta; \mathcal{D}) = \sum_{(w,c)\in\mathcal{D}} \left\{ \log \sigma(\mathbf{v}_c^\top \mathbf{v}_w) + \sum_{i=1, n\sim q(w)}^{k} \log \sigma(-\mathbf{v}_{c'}^\top \mathbf{v}_w) \right\} \tag{4.6}$$

where k is some fixed number of negative samples drawn from a re-scaled version of the empirical unigram distribution $q(w)$[13].

As Levy & Goldberg showed [LG14], the parameters found by maximizing the negative-sampling objective no longer correspond to a model of $p(c \mid w)$ but rather some quantity related to $p(c, w)$. Negative sampling was presented as a variation of Noise Contrastive Estimation (NCE) [GH10]. As Chris Dyer showed [Dye14], negative sampling is equivalent to NCE when $k = |C|$ and $q(w)$ is uniform, which is typically not the case. Finally, all we can say is the following: The negative sampling objective is related to the NCE objective but it is not the same. The maximizer of the negative sampling objective is not the maximum likelihood estimate of the language model described in Eq. (4.5). Nonetheless, the objective in Eq. (4.6) tries to increase the quantity $\mathbf{v}_c^\top \mathbf{v}_w$ for word-context pairs that actually appear in the corpus and decrease it for word-context pairs that do not appear in the corpus. Intuitively, this means that words that share a similar set of context words get assigned similar word vectors. Even though the

[13]The authors note in their article that raising the unigram distribution to the $\frac{3}{4}$-th power and re-normalizing it, resulted in the best performance.

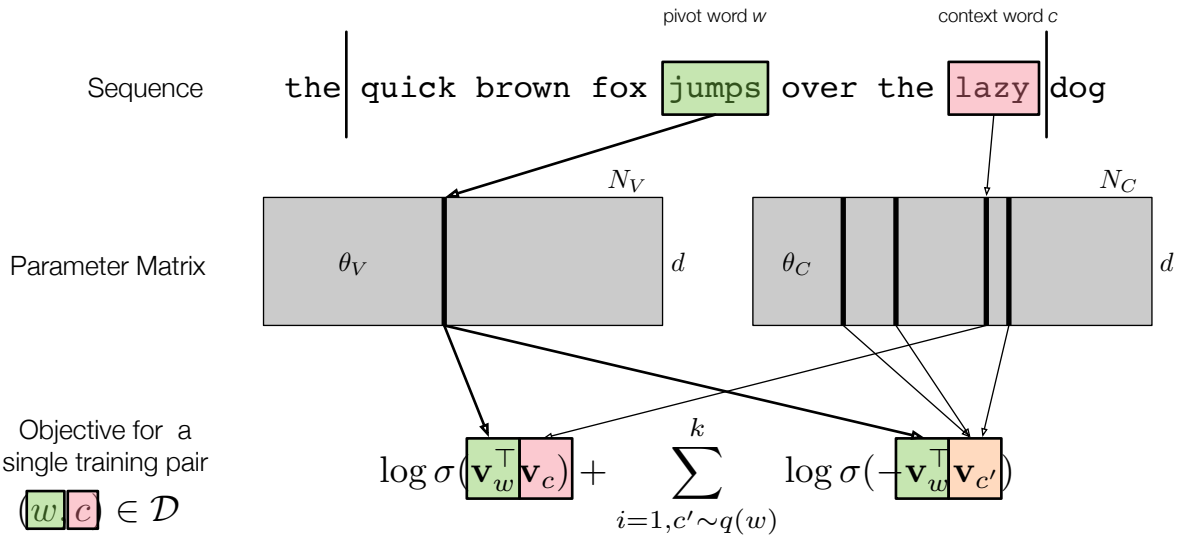


Figure 4.3: Illustration of the Skip-gram algorithm for the sentence "the quick brown fox jumps over the lazy dog" with a symmetric context window of size 3. The calculations of the algorithm are illustrated for the single training pair (`jumps`, `lazy`) and three negative samples.

connection between the language model and the negative sampling objective is not completely clear, the word representations obtained from the negative sampling objective capture a broad spectrum of linguistic properties of words and they have been used with great success as features in downstream classification tasks [CWB+11a, BDK14]. In our experiments we use the highly optimized Skip-gram implementation available from the python package `gensim` (see Appendix B).

Even though Word2Vec and the Skip-gram model were initially developed for computing textual word representations, since then researchers used the method for creating embeddings of other types of objects. In particular, the method was used extensively to create embeddings of nodes in graph structures, for instance in DeepWalk [PARS14], Node2Vec [GL16], Rdf2Vec [RP16] and the Doser framework for entity linking proposed by Zwicklbauer et al. [ZSG16]. At their core, these works rely on the Skip-gram method but they motivate different sampling strategies for extracting paths from the graph. The nodes on the path are then treated as if they were words in sentences and fed to the Skip-gram algorithm. For our experiments, we selected embeddings derived from the following two knowledge bases:

- **Google News articles**: The corpus is a collection of news articles from Google consisting of one billion words. The corpus itself is not publicly available, but the word embeddings were published along with the paper that introduced Skip-gram [14].

- **DBpedia**: DBpedia is a general-domain knowledge base extracted from Wikipedia content and converted into the Resource Description Framework (RDF) data

[14] `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing`, Last access: 04.07.2019.

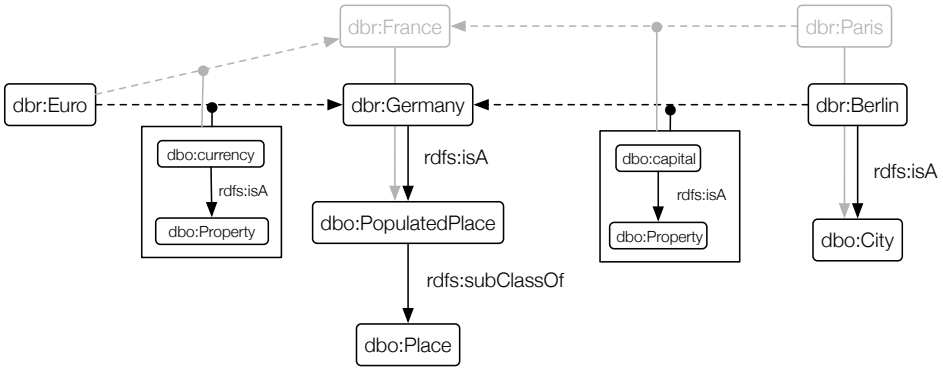


Figure 4.4: Illustration of entities and relationships in the DBpedia graph. The entity `dbr:Germany` has a property `dbo:capital` with the value `dbr:Berlin` and a property `dbo:currency` with value `dbr:Euro`. The entity `dbr:Euro` is shared by both Germany and France, which links the two countries via their common currency.

model. Based on Wikipedia articles and disambiguation pages, DBpedia describes structured information such as info boxes, geo-coordinates, category information, re-directions and external links in form of machine-readable subject-predicate-object statements about resources. DBpedia comes with its own OWL-based ontology which imposes semantic structure on the RDF resources and relationships between resources. Specifically, each Wikipedia page is an entity within DBpedia's ontology and it is connected to various other entities through semantic relationships.

Figure 4.4 shows an excerpt from the ontology for a small example. The collection of entities and relationships represents a labeled and directed graph. Since entities with the same concept inherit the same properties and relationships, there are many paths connecting entities through shared properties. The motivation for creating embeddings of nodes is similar as for words in sentences. Nodes that occur together on the same paths are connected via semantic relations and therefore they should get assigned similar embedding vectors. Embeddings of DBpedia entities were published by Ristoski & Paulheim [RP16][15].

[15] `http://data.dws.informatik.uni-mannheim.de/rdf2vec/models/DBpedia/2015-10/8depth/skipgram/DB2Vec_sg_200_5_5_15_4_500`, Last access: 04.07.2019.

4.3 Robustness of word embeddings

Before using the Skip-gram word embeddings in the fraud detection task, we study their robustness against multiple post-hoc parameter reduction methods in terms of the loss of accuracy on linguistic analogy tasks. The work was published in the proceedings of the European conference on Information Retrieval [JGS16].

The natural language processing (NLP) community has been successfully exploiting word embeddings over the last years, see for instance [CWB+11a, BDK14]. However, the gain in task-accuracy brings the downside that high-dimensional continuous valued word vectors require a large amount of memory. Moreover, the dimensions of embeddings and likewise the total number of parameters are fixed a-priori. Further, there is no natural transition to more memory efficient embeddings, by which one could trade accuracy for memory. This is particularly limiting in NLP-applications on resource limited devices where memory is still a scarce resource. An embedding matrix with 150,000 vocabulary words can easily require 60-180 Megabytes of memory, which is rather inconvenient to be transferred to and stored in a browser or mobile application. This restriction gives rise to contemplate different types of post-processing methods in order to derive robust and memory-efficient word vectors from a trained embedding matrix.

The word embeddings we use in our experiments were obtained from Mikolov's Skip-Gram algorithm [MCCD13]. As was shown shortly after the method was published, the algorithm factorizes an implicit word-context matrix, whose entries are the point-wise mutual information of word-context pairs shifted by a constant offset [LG14]. This PMI-matrix $M \in \mathbb{R}^{|V|\times|V|}$ is factorized into a word embedding matrix $W \in \mathbb{R}^{|V|\times d}$ and a context matrix $C \in \mathbb{R}^{d\times|V|}$, where $|V|$ is the number of words in the vocabulary and d is the number of dimensions of each word vector. The context matrix is only required during training and usually discarded afterwards. The result of optimizing the Skip-Gram's objective is that word vectors (rows in W) have high similarity with respect to their cosine-similarity in case the words are syntactically or semantically similar. Besides that, the word vectors are dense and have significantly fewer dimensions than there are context words - columns in M. With sufficiently large d, the PMI-matrix could be perfectly reconstructed from its factors W and C, and thus provide the most accurate information about word co-occurrences in a corpus [BDK14]. However, increasing the dimensionality d of word vectors also increases the amount of memory required to store the embedding matrix W. When using word embeddings in an application, we do not aim for a perfect reconstruction of the PMI-matrix but for reasonably accurate word vectors that reflect word similarities and word relations of language. Therefore, a more memory-efficient, yet accurate version of W would be desirable.

4.3.1 Memory reduction with post-processing

More formally, we want to have a mapping τ from the full embedding matrix W to $\hat{W} = \tau(W)$, where $\hat{W}$ can be stored more efficiently while at the same time its word vectors are similarly accurate as the original vectors in W. For the vectors in $\hat{W}$ to have an accuracy loss as low as possible, word vectors in W must be robust against the mapping function τ. We consider W *robust* against the transformation τ, if the

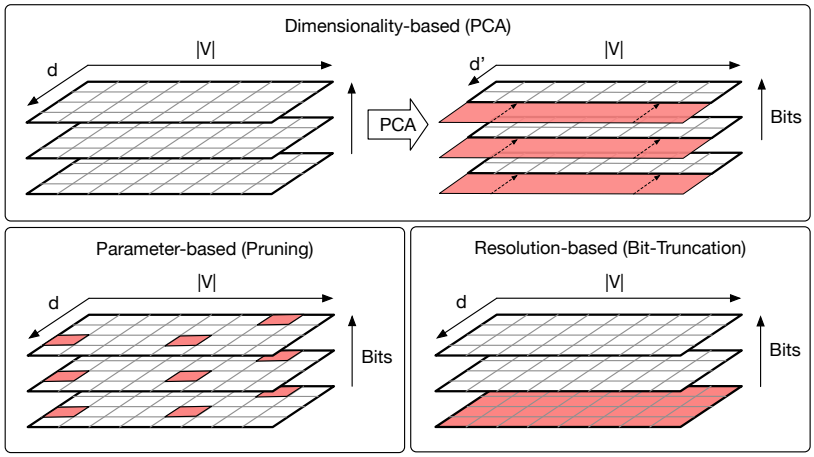


Figure 4.5: Three methods for post-processing a word embedding matrix: PCA-Reduction (*top*), Pruning across all Bit-planes (*left*) and Truncation of the least significant Bits (*right*).

loss of $\tau(W)$ is small compared to W across very different evaluation tasks. A memory reduction through τ can be induced by reducing the number of dimensions, the amount of effective parameters or the parameters' Bit-resolution. Accordingly, we employed three orthogonal post-processing methods that can be categorized into *dimensionality-based*, *parameter-based* and *resolution-based* approaches:

Linear Transformation via Principal Components Dimensionality-based approaches assume that points are not uniformly scattered across the embedding space but exhibit certain directions of dominant variations. If there is some kind of structure in the data, it should be possible to exploit it by means of representing the same data with fewer dimensions. If the discarded dimensions only accounted for redundant information, we would obtain basis vectors that describe the word embeddings equally well but with fewer parameters. Since our evaluation tasks rely on vector arithmetic and cosine similarities, we do not use nonlinear dimensionality reduction methods as these operations would be meaningless on the transformed embeddings $\hat{L}$ produced by a nonlinear mapping. Therefore, we used the PCA-solution as a linear transformation to obtain lower dimensional embeddings.

Random Parameter Pruning With *Pruning* we refer to a parameter-based method that discards a subset of the values in the embedding matrix by setting them to zero. With $\lambda \in [0, 1]$ we denote the *Pruning level*. Our naive pruning strategy is agnostic to word vectors since it determines a global threshold value p_λ from the whole matrix in such a way that $\lambda * 100\%$ of the matrix's values are greater than the threshold. All values w_{ij} below that threshold $|w_{ij}| < |p_\lambda|$ are set to zero. As a result of the pruning operation, we obtain a sparser embedding matrix with a degree of sparsity equivalent to $1 - \lambda$. Sparse matrices can be compressed more easily and thus require less memory than dense matrices. The rationale for using Pruning as reduction strategy arose from the observation that on normalized word vectors, pruning gradually projects points onto their closest coordinate axis. As we increase the pruning level, more points have

coordinates that are aligned with the coordinate axes. Due to the normalization, this alignment gradually affects some but not all dimensions of individual word vectors. We hypothesize that up to a certain pruning level, the inaccuracy induced by Pruning has no qualitative effect on word vector arithmetic and word similarity computations.

Bit-Truncation Besides a plain reduction of parameters by means of projection on fewer principle components, we explored a rather memory-focused approach that leaves the embedding dimensions untouched but migrates continuous word embeddings to discrete ones. The motivation is that in distributed embeddings the factors on all dimensions partially contribute to the meaning of a word. Thus, there should exist some degree of contribution which makes the meaning shift from one notion to another whereas for smaller contributions, the meaning is unaffected. We can exploit this relationship between the proximity of the embeddings' values and their similarities in meaning for purposes of memory efficiency by imposing resolution constraints on the value range along each coordinate. The Skip-Gram algorithm is defined on continuous valued word vectors which assumes each dimension to be real-valued. Figuratively, continuous embeddings allow for arbitrary positioning of a word's embedding in embedding space up to the precision of the datatype used. With *Bit-Truncation* we rasterize the embedding space uniformly by subdividing the range of values on each coordinate axis into distinct groups. Thus, all the factors of a distributed embedding still contribute to the meaning but only up to some pre-defined precision.

For the Bit-Truncation method, we adopt the approach described in [CPARS13] with slight adaptions. To reduce the resolution of the real numbers that make up the embedding matrix, first we shift the values to the positive range. Then we re-scale the values to the interval $[0, 1]$ and multiply them by 2^B, where B is the number of Bits we want to retain. Finally we cast the values to a 32-Bit Integer datatype. After casting to Integer, each coordinate axis has a resolution of $r = 2^B$ non-overlapping equally-spaced intervals. Consequently, the number of distinguishable regions in embedding space $R_\top = r^d$ is exponential in the number of dimensions d.

4.3.2 Experimental setup

Evaluations of word embeddings are published whenever new embedding methods are proposed. Besides manually inspecting 2D-projections of word vectors (e.g. t-SNE [VH08], principal component analysis), it is difficult to associate meaning to individual dimensions. In language modeling, authors have traditionally employed perplexity to evaluate their models. In recent years, the common approach shifted towards testing the embeddings on various word similarity or word analogy test data sets [BDK14, MYZ13]. In this domain, the work of Chen et al. [CPARS13] is the closest one to ours. Therein, they include a short section about information reduction capabilities of embeddings with limited experiments on other types of embeddings. We were particularly interested in preserving the linear structure in WORD2VEC-embeddings under limited memory conditions.

In all experiments we used word vectors estimated with the Skip-Gram method of the WORD2VEC-toolkit from a text corpus containing one billion words. The corpus

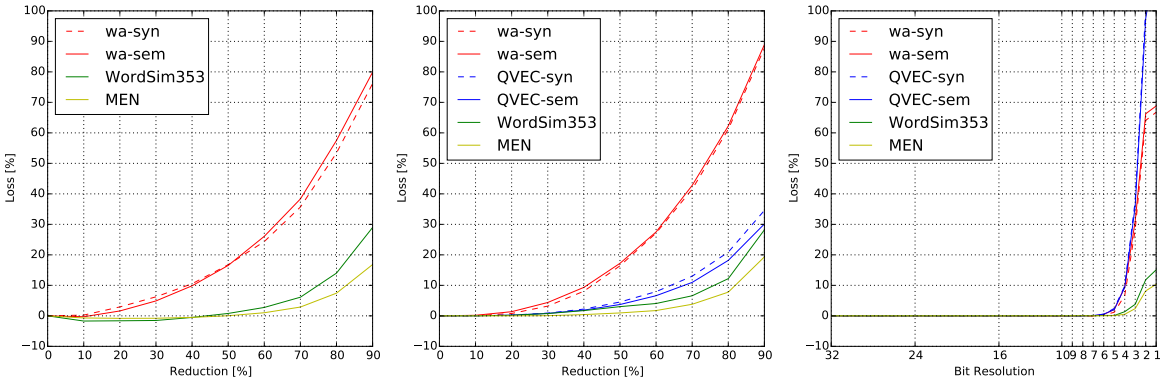
was collected from the latest snapshot of English Wikipedia articles[16]. After removing words that appeared less than 100 times, the vocabulary contained 148,958 words, both uppercase and lowercase. We used a symmetric window covering $k = 9$ context words and chose the negative-sampling approximation to estimate the error from $neg = 20$ noise words. With this setup, we computed word vectors of several sizes in the range $d \in [50, 100, 150, 300, 500]$. After training, all vectors are normalized to unit length. To evaluate the robustness and efficiency of word vectors after applying post-processing, we compare PCA-reduction, Pruning and Bit-Truncation on three types of intrinsic evaluation tasks: word relatedness, word analogy and linguistic properties of words. In each of these tasks, we use two different data sets.

Word Relatedness: The *WordSim353*(WS353) [FGM+02] and *MEN*[BTB14] data sets are used to evaluate pairwise word relatedness. Both consist of pairs of English words, each of which has been assigned a relatedness score by human evaluators. The *WordSim353* data set contains 353 word pairs with scores averaged over judgments of at least 13 subjects. For the *MEN* data set, a single annotator ranked each of the 3000 word-pairs relative to each of 50 randomly sampled word-pairs. The evaluation metric is the correlation (Spearman's ρ) between the human ratings and the cosine-similarities of word vectors.

Word Analogy: The word analogy task is more sensitive to changes of the global structure in embedding space. It is formulated as a list of questions of the form "a is to $\hat{a}$ as b is to $\hat{b}$", where $\hat{b}$ is hidden and has to be guessed from the vocabulary. The data set we use here was proposed by Mikolov et al. [MYZ13] and consists of 19544 questions of this kind. About half of them are morpho-syntactical (wa-syn) (*loud* is to *louder* as *tall* is to *taller*) and the other half semantic (wa-sem) questions (*Cairo* is to *Egypt* as *Stockholm* is to *Sweden*). It is assumed that the answer to a question can be retrieved by exploiting the relationship $a \rightarrow \hat{a}$ and applying it to b. Since WORD2VEC-embeddings exhibit a linear structure in embedding space, word relations are consistently reflected in sums and differences of their vectors. Thus, the answer to an analogy question is given by the target word w_t whose embedding $\vec{w}_t$ is closest to $\vec{w}_q = \vec{\hat{a}} - \vec{a} + \vec{b}$ with respect to the cosine-similarity. The evaluation metric is the percentage of questions that have been answered with the expected word.

Linguistic Properties: Schnabel et al. [SLMJ15] showed that results from intrinsic evaluations are not always consistent with results on extrinsic evaluations. Therefore, we include the recently proposed QVEC-evaluation[17][TFL+15] as additional task. This evaluation uses two dictionaries of words, annotated with linguistic properties: a syntactic (QVEC-syn) dictionary (e.g., `ptb.nns`, `ptb.dt`) and a semantic (QVEC-sem) dictionary (e.g., `verb.motion`, `noun.position`). The proposed evaluation method assigns to each embedding dimension the linguistic property that has highest correlation across all mutual words. The authors showed that the sum over all correlation values can be used as an evaluation measure for word embeddings. Moreover, they showed that this score has high correlation with the accuracy the same embeddings achieve on real-world classification tasks.

[16] `https://dumps.wikimedia.org/enwiki/20150112/`
[17] `https://github.com/ytsvetko/qvec`



(a) PCA (b) Pruning (c) Bit-Truncation

Figure 4.6: Mean relative loss of embeddings after (a) PCA: percentage of removed dimensions, (b) Pruning: percentage of removed parameters and (c) Bit-Truncation: remaining Bits. Scores on the QVEC datasets are not shown for PCA since they are not comparable across different word vector sizes.

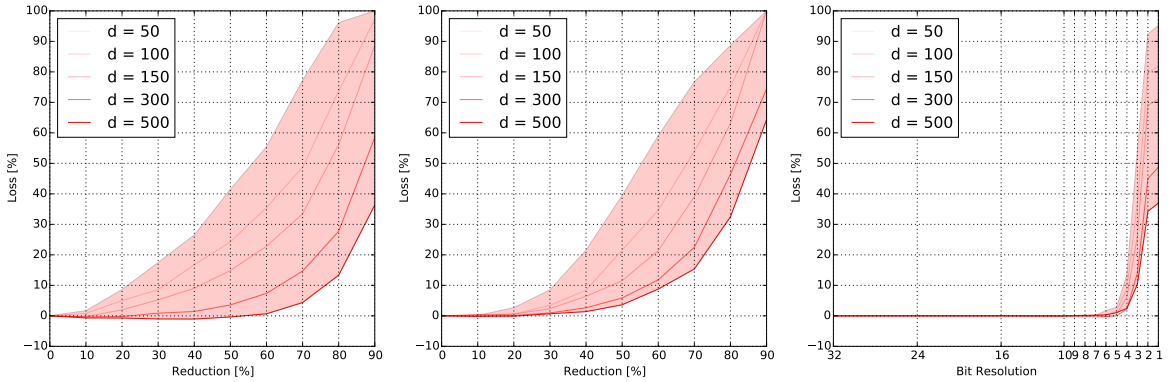
4.3.3 Results

Since we evaluate the robustness of word embeddings against post-processing, we report the *relative loss* induced by applying a post-processing method. The loss is measured as the difference between the score of original embeddings and the score of post-processed embeddings. In case of the word relatedness task, the score is the Spearman correlation. On the word analogy task, the score is given as accuracy. And on the linguistic properties task, the score is the output of the QVEC evaluation method. We divide the loss by the score of the original embeddings to obtain a relative loss that is comparable across tasks.

Robustness of word vectors

In Fig. 4.6 we report the mean relative loss, averaged over the five word vector sizes on all data sets. The percentage of reduction refers to the fraction of principle components with lowest eigenvalues that were discarded after applying PCA and to the fraction of parameters that were set to zero after pruning, respectively.

The word embeddings show a similar trend for all three post-processing methods. A small relative reduction results in a small loss, whereas a large reduction leads to a large loss. For all methods, the loss increases exponentially with the percentage of reduction. On the word analogy data sets, the loss is consistently higher than on the word relatedness and QVEC data sets. In particular, the relative loss on word relatedness data sets is predominantly unaffected (relative loss $< 10\%$) by post-processing up to a reduction threshold of 40%. Compared to the naive Pruning approach, PCA-transformed embeddings suffer lower loss on all tasks. Actually, on WordSim353 and MEN, PCA-reduced embeddings exhibit slight negative loss ($< 3\%$). Bit-Truncation produces no loss on any data set until the Bit-resolution of the parameters is lower than 8-Bit. To summarize, the Skip-Gram word embeddings are most robust against post-processing with resolution-based Bit-Truncation and the dimensionality-based PCA-reduction.



(a) PCA (b) Pruning (c) Truncation

Figure 4.7: Relative loss of embeddings on the syntactic word analogy dataset (wa-syn) after PCA (a), Pruning (b) and Bit-Truncation (c).

Memory efficiency

The percentage of reduction achieved by PCA is directly proportional to memory savings induced by the smaller number of dimensions. There, the sweet spot is task-dependent and the relative reduction can be rather high before word vector quality suffers a loss. In contrast, the number of pruned values is not directly proportional to memory savings, since the coding of sparse matrices requires additional memory. For instance, the row compressed storage method (see [Saa03]) has, without further assumptions about the shape of the matrix, a memory complexity of $\mathcal{O}(3k)$, where k is the number of non-zero elements in the sparse matrix. Thus, the pruning method would only start to pay off in terms of memory consumption above a pruning level of $\frac{2}{3}$, which would result in serious quality-loss. Finally, post-processing the embedding matrix with Bit-Truncation does not cause any loss on any of the evaluated data sets up to 75% reduction (24Bit). For resolutions below $r = 2^8$, all evaluated data sets respond to the low-precision embeddings with abruptly increasing loss.
Figure 4.7 shows that higher-dimensional embeddings ($d = 500$) can be reduced more aggressively than lower-dimensional ones before reaching the same level of relative loss. Since a similar behavior holds on all tasks, the observation is two-fold: First, it suggests that, the higher-dimensional the embedding space is, the more non-zero parameters there are and the higher their resolution is, the more redundant is the information that is captured in the embeddings. Secondly, the consistency across dimensionality-based and parameter-based methods indicates that neither the number of dimensions or parameters nor the continuous values alone but the number of distinguishable regions in embedding space is crucial for accurate word embeddings.

With a sufficiently large Bit-resolution the accuracy of all embedding sizes approximates the same accuracy level as with continuous values. Thus, we can confirm the finding in [CPARS13] also for Skip-Gram embeddings: The same accuracy can be achieved with discretized values at sufficiently large resolutions. Additionally, we state that this observation not only holds for cosine-similarity on word relatedness tasks but also for vector arithmetic on the word analogy task and for QVEC on the linguistic properties

Questions	Expected	3-Bit	4-Bit	5-Bit	6-Bit	7-Bit
Europe euro : Japan _____?	yen	Nagasaki	Taiwan	yen	yen	yen
Europe euro : Korea _____?	won	Kim	PRC	PRC	PRC	dollar
Europe euro : USA _____?	dollar	Dusty	proposal	US	euros	dollar
Europe euro : Brazil _____?	real	Alegre	proposal	dollar	euros	euros
Europe euro : Canada _____?	dollar	Calgary	Calgary	dollar	dollar	dollar

Table 4.2: Answer words for several country-currency analogy questions from word embeddings at different resolutions. Finally, all answer words are currencies.

task.
To summarize, Skip-Gram word embeddings can be stored more efficiently using a post-processing method that reduces the number of distinguishable regions $R_\top = (2^B)^d$ in embedding space. Pruning does so by producing increasingly large zero-valued regions around each coordinate axis ($2^B - const.$). PCA does so by mapping the word vectors into an embedding space with fewer dimensions $\hat{d} < d$. And Bit-Truncation directly lowers the resolution of each coordinate by constraining the Bit-resolution $\hat{B} < B$.

If an application can take a loss in word vector accuracy in favor of memory or transmission times, Skip-Gram embeddings can be reduced with all three evaluated methods. Thereby, pruning is the least efficient method as the overhead introduced by sparse coding could only be compensated by pruning levels above $\frac{2}{3}$. Such an aggressive pruning strategy would result in an average accuracy loss of more than 30%. In contrast, the linear dimensionality reduction technique worked well on our tasks and it allows for a consistent transition from higher to lower dimensional embeddings. The resolution-based approach provides the greatest potential for memory savings. With only 8-Bit precision per value, there is no loss on any of the tasks. A straight-forward implementation can thus fit the entire embedding matrix in only 25% of memory.

Resolution and semantic transition

Another observation is depicted in Table 4.2. On the word analogy data set, the transition from lower to higher-precision values not only yields increasingly better average accuracy but also corresponds to a semantic transition from lower to higher relatedness. Even if the embeddings' values have only 3-Bit precision, the retrieved answer words are not totally wrong but still in some kind related to the expected answer word. It seems that some notions of meaning are encoded on a finer scale in embedding space and that these require more Bits to remain distinguishable.

For coarse resolutions (3-Bit) the regions in embedding space are too large to allow an identification of a country's currency. Because there are many words within the same distance to the target location, the most frequent one is retrieved as answer to the question. As the resolution increases, regions get smaller and thus more nuanced distances between word embeddings emerge, which yields not only increasingly accurate but also progressively more related answers.

4.3.4 Summary

In this section, we explored three methods to post-process Skip-Gram word embeddings in order to identify means to reduce the amount of memory required to store the embedding matrix. Therefore, we evaluated the robustness of embeddings against a dimensionality-based (PCA), parameter-based (Pruning) and a resolution-based (Bit-Truncation) approach. The results indicate, that embeddings are most robust against Bit-Truncation and PCA-reduction and that preserving the number of distinguishable regions in embedding space is key for obtaining memory efficient (75% reduction) and accurate word vectors.

4.4 Fraud detection with augmented data

Initially, we evaluated the impact of injecting word embeddings from DBpedia and the holiday feature alone with a feed-forward neural network and we found a slight improvement of the predictive performance through features. The work was published in the proceedings of the IEEE International Conference on Enabling Technologies [ZCG+17]. Within this thesis, we now provide a more extensive study of external data by considering also other types of demographic statistics (introduced in Section 4.2.1) and a second source from which we derive country embeddings (see Section 4.2.2). For evaluating the impact of these feature, we also resort to a random forest classifier and the evaluation protocol described in Sections 3.5 and 3.6.

Experimental Setup We used transactions from the period 01.03.2015 - 30.04.2015 for model selection and reserved the remaining transactions from 08.05.2015 - 31.05.2015 for model evaluation. Model selection was performed via a random search over 20 points on a hyper-parameter grid, spanned by: the maximum depth of the trees ($[4, \inf]$), the maximum number of features to be searched for the best split ($[1, 3, \sqrt{d}]$) - where d is the total number of features in the feature set -, the minimum number of samples per leaf node ($[1, 10, 20]$) and whether to bootstrap samples for each tree creation ($[true, false]$). The number of decision trees was set to 300. We measured the performance of each grid point by means of the mean $AUCPR_{0.2}$ over 3-fold temporal cross-validation. The cross-validation window was set to 30 days for training and 7 days for validation within the model selection period.
For model evaluation, we selected the best hyper-parameters as obtained from the random search and trained a random forest classifier 10 times with one million accounts drawn randomly from 01.03.2015-30.4.2015 at a re-sampling ratio 10:90 and we finally evaluated the performance of each of the 10 models on the entire test period 08.05.2015-31.05.2015. We encoded categorical variables as integers and we normalized numerical variables to zero mean and unit standard deviation. For replacing country codes with their word embeddings, we created lookup tables that map DBpedia/GoogleNews country names to the 3-letter country codes[18]. We extracted the country word embeddings of the 207 different countries that appear in the data set and projected them on the first 5 principal components via Principal Component Analysis.

4.4.1 Demographic statistics

In the first experiment, we evaluate the impact of adding demographic statistics to the transactions. We link all demographic statistics, including the holiday feature, to both the card-holder's country and the merchant's country. A total of 16 new features were added in this way. For comparability with domain knowledge features, we keep the naming convention as introduced in Section 4.1: The set of raw features (`BASE`), the set of raw features and the time delta between consecutive transactions (`TDELTA`), the set

[18]The country codes in our data set comply with the Alpha-3 encoding as defined in the international standard ISO 3166-1: `https://www.iso.org/obp/ui/#search/code/`, Last access: 9.7.2019.

of raw features and feature aggregates (`AGG`) and the union of all these feature sets (`AGG + TDELTA`). The set of demographic features is denoted by `ext`.

Dataset	Feature Set	Augmentation	$AUCPR$ $(\hat{\mu}, \hat{s})$	$AUCPR_{0.2}$ $(\hat{\mu}, \hat{s})$
ECOM	base	-	**0.2118** $\pm$ 0.0023	**0.1266** $\pm$ 0.0008
		ext	0.1877 $\pm$ 0.0015	0.1218 $\pm$ 0.0010
$n_+ =$ 16960	tdelta	-	0.2500 $\pm$ 0.0011	0.1339 $\pm$ 0.0008
$r_+ =$ 0.00369		ext	**0.2920** $\pm$ 0.0027	**0.1375** $\pm$ 0.0014
	agg	-	0.3662 $\pm$ 0.0020	0.1506 $\pm$ 0.0013
		ext	**0.4163** $\pm$ 0.0018	**0.1617** $\pm$ 0.0012
	agg+tdelta	-	0.3755 $\pm$ 0.0025	0.1509 $\pm$ 0.0017
		ext	**0.4474** $\pm$ 0.0025	**0.1691** $\pm$ 0.0015
F2F	base	-	0.1361 $\pm$ 0.0023	**0.0903** $\pm$ 0.0020
		ext	**0.1415** $\pm$ 0.0033	0.0883 $\pm$ 0.0029
$n_+ =$ 2545	tdelta	-	0.1895 $\pm$ 0.0023	0.1131 $\pm$ 0.0027
$r_+ =$ 0.00043		ext	**0.1968** $\pm$ 0.0039	**0.1138** $\pm$ 0.0025
	agg	-	**0.2010** $\pm$ 0.0035	**0.1122** $\pm$ 0.0029
		ext	0.1939 $\pm$ 0.0056	0.1001 $\pm$ 0.0041
	agg+tdelta	-	0.2140 $\pm$ 0.0037	**0.1134** $\pm$ 0.0027
		ext	**0.2165** $\pm$ 0.0027	0.1105 $\pm$ 0.0022

Table 4.3: Classification performance of random forest classifier evaluated across time and over different choices of feature sets with and without demographic country features (ext). Scores are the mean and standard deviation of AUCPR/AUCPR@0.2 over the entire test period averaged over 10 runs. n_+ denotes the number of frauds and r_+ the ratio of frauds in the test set. The training set undersampling ratio was set to 10:90. The maximum mean score within each scenario is marked in bold; standard deviation is given in $\pm$ notation.

Table 4.3 displays the results. On face-to-face transactions (bottom half of the table) we can not observe any influence of the demographic features. Over all feature sets, the scores obtained when adding demographic features are comparable to the scores when no such features are added. In contrast, on e-commerce transactions we can observe an improvement of the predictive performance under the `TDELTA`, `AGG`, and `AGG+TDELTA` features sets. The largest increase is observed when demographic features are added to the joint set of feature aggregates and time delta, both in terms of $AUCPR$ and $AUCPR_{0.2}$. We hypothesize that the discrepancy between e-commerce and face-to-face has to do with the larger variety of countries we see in a card holder's sequence. The large majority of card holders is registered in Belgium (95%) and the large majority of F2F-transactions is issued against merchants in Belgium (66%) or some neighbouring country like France (12%). For a detailed overview see Fig. 3.4. Therefore, the information added along the card holder's country of origin or along the merchant's country is mostly

the same for an overwhelming portion of transactions. However, in e-commerce we see a larger variety of merchant countries in card holders' accounts. The occurrence of different merchants in a card holder's account first leads to more nuanced country-based aggregates and secondly opens a wider realm for finding useful correlations between the countries' properties and other features. This would at least explain the noticeable improvement when demographic features are used jointly with aggregation features and why, in case of the base feature set, the addition of demographic features deteriorates performance.

4.4.2 Country embeddings

In the second experiment, we evaluate the impact of encoding countries with word vectors derived from public knowledge bases. Both the card-holder's country and the merchant's country are encoded with their corresponding word vectors.

We denote the augmentation with vectors derived from DBpedia as `DBpedia` and the the augmentation with vectors derived from Google News articles as `gnews`. As in the previous experiment, we report the results individually for the different domain knowledge based feature sets. Table 4.4 summarizes the results.

Similar to the previous experiment, on face-to-face transactions there is no clear difference between the integer encoding of countries and the word vector encoding of countries within the different domain knowledge based feature sets. Even though some augmentation strategies yield higher predictive performance together with individual domain knowledge based feature sets, the improvement is not consistent across all feature sets. The performance increase is dominated by the domain knowledge features and not by the external features.

However, on e-commerce transactions we observe a consistent improvement when external features are combined with feature aggregates. Within the `AGG` scenario and the `AGG+TDELTA` scenario, all augmentation strategies improve the performance, both in terms of $AUCPR$ and $AUCPR_{0.2}$. Interestingly, the margin of increase is comparable between all augmentation strategies - whether we encode the countries with word embeddings or explicitly add country-specific demographic features. Without feature aggregates, we can not observe a consistent improvement through data augmentation. Especially together with the raw features alone (`BASE`), the augmentation strategies seem to severely deteriorate performance in one case (`BASE + DBpedia + ext`).

4.4.3 Daily evaluation

As we have seen in Section 3.7.2, the predictive performance of a classifier on future transactions varies strongly from one day to another. This is not necessarily due to the varying proportion of frauds that occur at some day but, as we suppose, more due to the nature of fraudulent and genuine transactions.

The entire model selection and model evaluation process is subject to several sources of randomness: The random sampling of compromised and non-compromised accounts, the random search over hyper-parameters and the bootstrap sampling during random forest creation. We have accounted for the randomness by running each experiment

Dataset	Feature Set	Augmentation	$AUCPR$ $(\hat{\mu}, \hat{s})$	$AUCPR_{0.2}$ $(\hat{\mu}, \hat{s})$
ECOM	base	-	0.2118 ± 0.0023	0.1266 ± 0.0008
		DBpedia	0.2036 ± 0.0019	0.1250 ± 0.0007
$n_+ =$ 16960		gnews	0.2085 ± 0.0021	0.1268 ± 0.0010
$r_+ =$ 0.00369		DBpedia+ext	0.1280 ± 0.0026	0.0995 ± 0.0022
		gnews+ext	**0.2357** ± 0.0012	**0.1291** ± 0.0008
	tdelta	-	0.2500 ± 0.0011	0.1339 ± 0.0008
		DBpedia	0.2349 ± 0.0018	0.1314 ± 0.0010
		gnews	0.2498 ± 0.0013	0.1355 ± 0.0008
		DBpedia+ext	**0.2838** ± 0.0018	**0.1366** ± 0.0009
		gnews+ext	0.2751 ± 0.0023	0.1364 ± 0.0012
	agg	-	0.3662 ± 0.0020	0.1506 ± 0.0013
		DBpedia	0.4195 ± 0.0014	0.1638 ± 0.0011
		gnews	0.4168 ± 0.0023	0.1628 ± 0.0015
		DBpedia+ext	**0.4287** ± 0.0029	**0.1640** ± 0.0026
		gnews+ext	0.4085 ± 0.0024	0.1613 ± 0.0017
	agg+tdelta	-	0.3755 ± 0.0025	0.1509 ± 0.0017
		DBpedia	0.4470 ± 0.0025	0.1689 ± 0.0020
		gnews	**0.4482** ± 0.0019	**0.1695** ± 0.0014
		DBpedia+ext	0.4358 ± 0.0016	0.1683 ± 0.0013
		gnews+ext	0.4364 ± 0.0033	0.1686 ± 0.0024
F2F	base	-	0.1361 ± 0.0023	0.0903 ± 0.0020
		DBpedia	0.1358 ± 0.0043	0.0856 ± 0.0038
$n_+ =$ 2545		gnews	**0.1450** ± 0.0030	**0.0911** ± 0.0032
$r_+ =$ 0.00043		DBpedia+ext	0.1322 ± 0.0027	0.0814 ± 0.0028
		gnews+ext	0.1281 ± 0.0021	0.0813 ± 0.0021
	tdelta	-	0.1895 ± 0.0023	0.1131 ± 0.0027
		DBpedia	0.1820 ± 0.0053	0.1096 ± 0.0043
		gnews	0.1973 ± 0.0024	0.1137 ± 0.0029
		DBpedia+ext	0.1738 ± 0.0054	0.1058 ± 0.0045
		gnews+ext	**0.1993** ± 0.0033	**0.1140** ± 0.0030
	agg	-	**0.2010** ± 0.0035	**0.1122** ± 0.0029
		DBpedia	0.1951 ± 0.0031	0.1034 ± 0.0024
		gnews	0.1885 ± 0.0043	0.1001 ± 0.0045
		DBpedia+ext	0.1913 ± 0.0054	0.0984 ± 0.0043
		gnews+ext	0.1978 ± 0.0043	0.1008 ± 0.0035
	agg+tdelta	-	0.2140 ± 0.0037	**0.1134** ± 0.0027
		DBpedia	0.2090 ± 0.0044	0.1092 ± 0.0030
		gnews	0.2112 ± 0.0035	0.1109 ± 0.0026
		DBpedia+ext	0.2123 ± 0.0037	0.1084 ± 0.0026
		gnews+ext	**0.2165** ± 0.0055	0.1104 ± 0.0043

Table 4.4: Classification performance of random forest classifier evaluated across time and over different choices of feature sets with and with different data augmentation strategies. Scores are the mean and standard deviation of AUCPR/AUCPR@0.2 over the entire test period averaged over 10 runs. n_+ denotes the number of frauds and r_+ the ratio of frauds in the test set. The training set undersampling ratio was set to 10:90. The maximum average score within each scenario is marked in bold; standard deviation is given in ± notation.

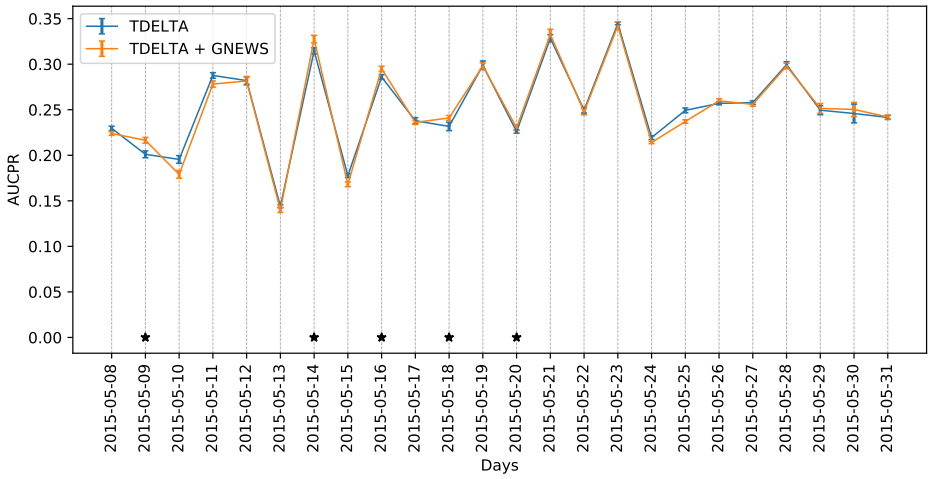


Figure 4.8: Example: Evolution of daily $AUCPR$ for the TDELTA feature set and the TDELTA + GNEWS feature set. Star markers highlight test days on which the orange model yielded a higher mean AUCPR than the blue model and the difference is significant at significance level $\alpha = 0.05$. The orange model is considered as good or worse than the blue model since it improves the prediction score on less than 12 out of 24 test days.

10 times and reporting the mean $AUCPR$ scores averaged over the 10 repetitions. Each $AUCPR$ score was calculated on predictions from the entire test set consisting of transactions from 24 days.

We can imagine that there might be several globally suspicious transactions that get assigned high fraud scores by the classifier. And when we calculate the $AUCPR$ on the entire test set, we effectively ignore that the collection of true-positives and false-positives consists of transactions from completely different test days. Even though this procedure is valid for the purpose of model comparison, in practice, we are interested in the performance of the classifier on each individual test day. Now we can make use of the 10 runs to determine whether the difference in average $AUCPR$ scores of two models on some test day can be regarded as a meaningful difference or just an oddity introduced by the randomness of the process. To that end, we test the significance of the difference of *daily* mean $AUCPR$ scores between two models with the two-sample t-test at a significance level of $\alpha = 0.05$. Then, we count the number of days on which the mean $AUCPR$ of one model is significantly higher than the mean $AUCPR$ of the other model.

For clarity, figure 4.8 shows the evolution of the daily $AUCPR$ over all test days of the `tdelta+gnews` model (orange) and the `tdelta` model (blue). Since the difference between the two curves is very small compared to the variation over days, we would not be able to notice any difference between two models if we simply compared the averages of their daily $AUCPR$ scores. Therefore, we resort to an evaluation where we count the number of days on which one model yields a higher $AUCPR$ than the other. These days are highlighted with a star-marker in the figure.

Figure 4.9 summarizes all pair-wise comparisons between models introduced in this chapter. As we have already observed in Sections 4.4.1 and 4.4.2, data augmentation on

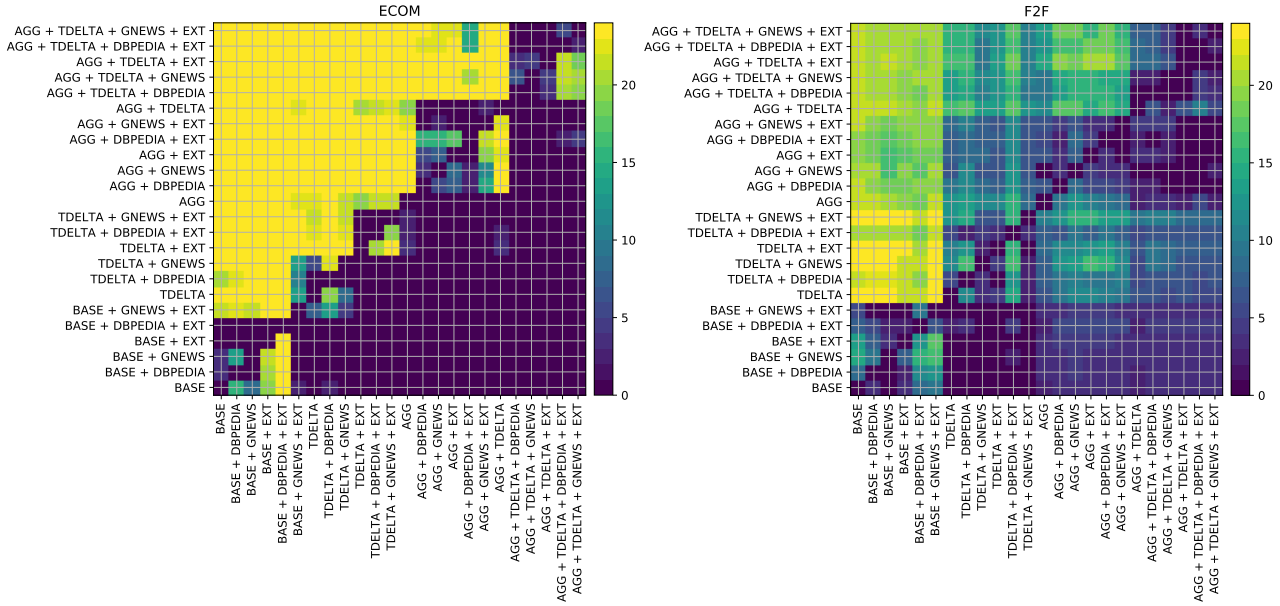


Figure 4.9: Pairwise comparison of feature sets. Color indicates the number of test days, among a total of 24, on which the "row" feature set results in significantly higher $AUCPR$ scores than the "column" feature set.

face-to-face transactions does not improve the predictive performance. Even the addition of domain knowledge only improves the prediction scores on barely half of the test days. The most consistent improvement comes from the time delta feature, which improves the $AUCPR$ on at least 16 out of 24 test days over the base feature sets. On the other hand, the pairwise comparisons on e-commerce (Fig. 4.9, left) reveal three aspects: (i) there is no consistent benefit from data augmentation when using the raw features alone (base). (ii) there is a clear gain from data augmentation when combined with feature aggregates and (iii) there is no consistent difference between augmenting transactions with country embeddings or explicit demographic features.

4.5 Summary

In this chapter, we explored domain knowledge and external knowledge that we extracted from public knowledge bases for the purpose of augmenting transactions with contextual information beyond the basic transaction attributes.

Regarding research question **RQ1**, we can confirm the findings from literature: Classic domain-specific feature engineering approaches produce highly informative features for credit card fraud detection. We note that feature aggregates provide a simple and yet efficient means to improve the predictive performance by a large margin with, on average, 76% increase in terms of $AUCPR$ on e-commerce transactions and a 53% increase on face-to-face transactions. Regarding research question **RQ2**, we identified the country of the card holder, the country of the merchant and the date of the transaction as the most promising links to data from external sources. We collected demographic

statistics about countries from `worldbank.org`, `numbeo.org` and `transparency.org`, public holiday information from calendars provided by the Thunderbird and Mozilla Foundation. Regarding research question **RQ3**, we selected a natural language text corpus and a semantic knowledge graph as external sources because they incorporate implicit knowledge about countries and we employed the unsupervised feature learning method Skip-gram to derive word embeddings for country names. We studied the robustness of Skip-gram word embeddings under multiple parameter reduction methods and found that Bit-truncation can reduce the memory footprint of word embeddings by 75% without incurring loss on synthetic word relatedness and word analogy tasks. Finally, we used word embeddings of country names derived from GoogleNews articles and the semantic knowledge graph DBpedia to encode the country codes in the credit card data, thereby augmenting the data set with features that represent countries in terms of word co-occurrences in text documents or in terms of node neighborhoods in a knowledge graph.

The results reveal a mixed picture as to whether our data augmentation strategies improve fraud classification. On face-to-face transactions, we could not observe any improvement that would be consistent across all domain knowledge feature sets. However, on e-commerce transactions we see some benefit in augmenting the transactions. Therefore, regarding research question **RQ4** we state: When combined with feature aggregates the augmentation strategies (country embeddings or demographic features) yield a small but consistent improvement between, on average, 13% (`AGG`) and 19% (`AGG+TDELTA`) on e-commerce transactions. Interestingly, the improvement induced by either explicit demographic features or by country embeddings from either knowledge base is in a comparable range. This observation not only suggests that these unsupervised word representations contain a broad spectrum of information relevant to fraud detection but it also shows that external data can be leveraged very easily with unsupervised learning methods.

From a practical point of view, it is questionable whether the small performance increase justifies the additional overhead of having to review and select external sources and having to maintain appropriate processes that keep the extracted features up to date with the sources. In particular, a practical application would first need to assess the performance margin throughout an entire year and then take appropriate decisions. While the cost of maintaining manually extracted demographic statistics might be too high, the unsupervised extraction of country embeddings from a text corpus or a knowledge graph seems much more promising. The extraction does not involve much manual labor and it can easily be triggered repeatedly to update the embeddings.

Chapter 5

Sequence Classification for Credit Card Fraud Detection

The work presented in this chapter was published in the journal Expert Systems with Applications [JGZ+18].

In the previous chapter we observed that feature aggregates improve the prediction performance by a large margin. In contrast to the raw transaction-specific features that characterize a transaction in absolute terms and independently from any other transaction, an aggregate characterizes a transaction *relative to its recent transaction history.* To see the implication of this subtle difference, we will disentangle the statement and elaborate on the individual terms:

- *...recent transaction history:* Transactions are real-world events and thus they occur at certain points in time, one after the other. At any point in time we can look back and consider the entire set of all transactions that were ever issued as the transaction history and then derive statistics from the set of historic transactions. Apart from being practically impossible to handle the entire history, it might also not be relevant for a decision at the current point in time. Purchases from years ago are most likely outdated and no longer reflect the general purchase habits of the population of all users. However, once we reduce the time window to only one year, we might already feel tempted to ponder whether there exist recurring patterns in a yearly cycle. Even if we had an oracle that determines some sweet-spot on the size of a time window, the size would heavily depend on the actual prediction task, the user or even the transaction itself. When we fit a classifier on data from some training period, we fixed a global time window and estimated

the parameters of the classifier from the corresponding set of historic transactions. Now, when we calculate the values of one feature aggregate, we still use a fixed time window but we center it on each and every transaction. The value of an aggregate for a single transaction is calculated from the transaction's recent history, no matter when the transaction occurred on the absolute time axis. Therefore, each value of an aggregate is essentially a transaction-specific statistic in its own right and it varies with that transaction's recent history. In fraud detection, the aggregation window is often set between 24 hours and a couple of days. Despite being a rule of thumb, this choice can be motivated by the fact that frauds tend to appear in bursts within short time intervals of up to 24 hours. And whenever the number of transactions within the last few hours exceeds some expected count, we might deem the most recent transaction to be suspicious.

- *Relative to its...:* Regarding the raw features, such as the country, payment mode or the amount, a transaction assumes one value in the range of each of these feature. Thereby, the value is absolute in the sense that it depends only on the transaction itself and nothing else. In contrast, the value of an aggregate at one transaction depends on the feature values of the transaction and the feature values of the transactions in its recent history. Given the definition of aggregates in Section 4.1, we impose constraints on the set of transactions we select from the recent history. We already discussed the temporal constraint of not looking further back in time than a fixed number of hours or days. Another implicit constraint is to only consider transactions from the same card holder. But apart from these, we can also impose further constraints to select only those transactions that have been issued in the same country, under the same payment mode or with a similar amount as the current transaction. All these constraints are defined relative to the values of the current transaction. Consequently, when we aggregate the selected transaction history in terms of counting or summing, we obtain a relative quantity that depends on the transaction and its history. If the exact same transaction would occur at a different point in time, the values of all its feature aggregates might change. However, it is important to note that aggregates are invariant to permutations of transactions within the selected history. They indicate the presence of similar transactions but not the order in which these occurred.

To summarize, the value of one feature aggregate depends on both the transaction and the transaction's recent user history. It can be considered as an explicit description of the state the user's account is in, at the time when that transaction occurs. By using several different feature aggregates, we effectively create a multi-variate state that summarizes a user's current activity and changes with each new transaction. A consequence of the aggregation functions is that the multi-variate state is invariant to the ordering of the transactions it represents.

In this chapter we are going to explore an alternative approach for summarizing a user's current activity in form of states associated with his transactions. But instead of creating the states manually as a set of features, we choose a model-based approach and integrate the recent history at an algorithmic level. For that purpose, we make use of a classifier that assumes the observations to appear as an ordered sequence because

we are particularly interested in the question whether the order of transactions conveys additional information which we could exploit for fraud classification.

We provide an empirical study of the classification accuracy obtained with a Long Short-term Memory network (LSTM) on e-commerce and face-to-face transactions. Also, we discuss typical pitfalls encountered in the application of a LSTM in the context of credit-card fraud detection. In particular, we raise the following questions:

- **RQ1**: What is the difference in predictive performance between a sequence learner (LSTM) and a static learner (random forest) on the e-commerce and the face-to-face dataset?
- **RQ2**: Does the LSTM benefit from feature aggregates comparable in magnitude to the random forest or does it extract similar information such that aggregations become redundant?
- **RQ3**: Does the LSTM detect the same set of fraudulent transactions as the random forest classifier?

5.1 Methodology

For modeling transaction sequences of users, we collect k-element sequences from all accounts and we treat each k-element sequence as input for the LSTM. While iterating through a sequence, the network produces a sequence of latent states. We consider the last state as a feature vector that comprises information about all previous input transactions in the sequence and we train a binary Logistic Regression classifier on these features to make a decision about whether the sequence is likely to end in a fraudulent or a genuine transaction. The parameters of the LSTM and the Logistic Regression output model can be trained jointly by minimizing the cross entropy loss between the predicted label and the true fraud label via backpropagation through time. Within this architecture, the LSTM models a set of completely diverse time series. Each k-element sequence is a time series in its own right and we're using one single LSTM, i.e. set of parameters, to model all these different series. This is a gross over-complication of fraud detection. In the optimal case we would fit one sequence model to each user's account. But such model would have to be very simple or, to put it in other words, it should have very few parameters since we have to estimate them from the few dozens of transactions each individual user has ever issued. There is also no obvious grouping of users which we could exploit to model sets of time series with similar dynamics. In principle, recurrent neural networks can learn from variable length sequences. The reason why we require a fixed number of k elements in the sequences, is simply to enforce comparability of the predictions at the k-th state after having observed $k-1$ transactions. Consequently, we model only $p(y_k|\mathbf{x}_{1:k})$ instead of $p(y_{1:k}|\mathbf{x}_{1:k})$ because we are not interested in the succession of labels in a sequence but only in the label of the last transaction $\mathbf{x}_k$ after having observed $\mathbf{x}_{1:k}$.

One of the few studies that used LSTM networks for modeling sequences of transactions for the purpose of credit card fraud detection is the work of Wiese & Omlin [WO09].

In a comparative study of an LSTM, a support vector machine (SVM) and a feed-forward neural network (FFNN), the authors obtained significantly better results with an LSTM on a small data set consisting of 31,000 transactions. Our work is different in the following regards: (i) We use a different LSTM architecture that is trained with only the label of the last transaction of a sequence, (ii) we report results independently for the e-commerce and the face-to-face scenario and (iii) we compare the LSTM results over multiple domain knowledge based feature sets to a random forest classifier. Since we have access to a much larger data set (47 million), we hope to complement the findings from related work with performance estimates that reflect the real-world use case.

5.1.1 Recurrent neural network for sequence classification

A Long Short-Term Memory Network (LSTM) is a special type of recurrent neural network (RNN). Before we introduce the model formulation of an LSTM, we briefly recap recurrent neural networks in general. Recurrent neural networks have been developed in the 80's [RHW86, Wer88, Elm90] to model time series data. The structure of a RNN is similar to that of a standard multilayer perceptron, with the addition that it allows connections among hidden units associated with discrete time steps. The time steps index the individual elements in a sequence of inputs. Through the connections across time steps the model can retain information about the past inputs, enabling it to discover temporal correlations between events that are possibly far away from each other in the input sequence. This is a crucial property for proper learning of time series where the occurrence of one event in the past provides information about future events.

A vanilla recurrent neural network is initialized with a state vector $\mathbf{s}_0$, usually consisting of only zeros, and then receives a sequence $\mathbf{x}_{1:T}$ of input vectors $\mathbf{x}_t$, indexed by positive integers t. In our experiments we fix the length of input sequences to either $T = 5$ (SHORT) or $T = 10$ (LONG). The RNN then runs through a sequence of state vectors $\mathbf{s}_t$, determined by the following recurrence equation:

$$\mathbf{s}_t = \sigma(W \cdot \mathbf{s}_{t-1} + U \cdot \mathbf{x}_t + \mathbf{b}), \tag{5.1}$$

where the trainable parameters of the model are the recurrent weight matrix W, the input weight matrix U and the biases $\mathbf{b}$. The hyper-parameters of the model are the dimensions of the vectors and matrices, and the non-linear element-wise activation function σ – hyperbolic tanh in our case. The mapping from an input sequence to a state sequence is typically referred to as a *recurrent layer* in neural network terminology. RNNs can have several of such archetypes stacked on top of each other (see experiments in Section 5.2). For the training, we now additionally specify a cost function and a learning algorithm.

A cost function $\mathcal{E}_T$ measures the performance of the network on some given task after having seen T input vectors and transitioning to state $\mathbf{s}_T$. The distribution over classes *fraud* and *non-fraud* given state $\mathbf{s}_T$ is modeled with a logistic regression output model outputting predictions $\hat{y}_T = p(y_T|\mathbf{s}_T)$. We interpret the true label $y_t \in \{0, 1\}$ of a transaction as the probability of $\mathbf{x}_t$ to belong to class 0 or 1 and measure the cost induced by the model's predicted probabilities $\hat{y}_t$ by means of the cross-entropy error,

defined as:

$$\mathcal{E}_T = \mathcal{L}(\mathbf{x}_{1:T}, y_T) = -y_T \log \hat{y}_T - (1 - y_T) \log(1 - \hat{y}_T)$$

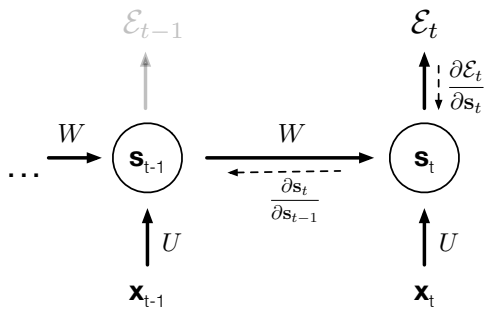


Figure 5.1: Schematic of a recurrent neural network unrolled in time by creating a copy of the model for each time step.

The model parameters $\theta = (W, U, \mathbf{b})$ are learned by minimizing the cost $\mathcal{E}_T$ with a gradient-based optimization method. One approach that can be used to compute the required gradients is Backpropagation Through Time (BPTT). BPTT proceeds by unrolling a recurrent network in time to represent it as a deep multilayer network with as many hidden layers as there are time steps (see Fig. 5.1). Then, the well-known Backpropagation algorithm [RHW86] is applied on the unrolled network.

While in principle, the recurrent network is a simple and powerful model, in practice, it is hard to train properly with gradient descent. Among the many reasons why this model is so unwieldy, there are two major problems that have been coined the *vanishing* and *exploding gradient* problem [YPP14]. These problems refer to the fact that gradients can either decrease or increase exponentially as the length of the input sequence grows. Very small gradients cause the network to learn very slowly or even stop completely. Very large gradients make the parameters diverge which leads to numerical problems and consequently derails learning.

With the recurrent connection between latent states, the parameters θ affect the error not only through the last but all previous states. Likewise, the error depends on $\mathbf{W}$ through all states $\mathbf{s}$. This dependency becomes problematic when we compute the gradient of $\mathcal{E}_T$ w.r.t. θ:

$$\frac{\partial \mathcal{E}_T}{\partial \theta} = \sum_{1 \leq k \leq T} \left(\frac{\partial \mathcal{E}_T}{\partial \mathbf{s}_T} \frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_k} \frac{\partial \mathbf{s}_k}{\partial \theta} \right) \tag{5.2}$$

The Jacobian matrix $\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_k}$ contains all component-wise interactions between state $\mathbf{s}_k$ and state $\mathbf{s}_T$. It can be understood as a means to transport back the error from state T to state k. It is given as a product of all pairwise interactions between consecutive states:

$$\frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_k} = \prod_{k < i \leq T} \frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}} \tag{5.3}$$

This product is the reason behind the difficulties to learn long-term dependencies with gradient-based optimization methods. Each factor $\frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}}$ involves both the recurrent weight matrix and the derivative of σ. The number of factors in Eq. (5.3) is $T - k$ and Pascanu et al. [AZLS18] showed that it is sufficient for the largest eigenvalue of the recurrent weight matrix to be smaller than 1 for long term components to decrease exponentially and necessary for it to be larger than 1 for gradients to increase exponentially with $T - k$.

Several solutions exist to address these problems. Using an L1- or L2-penalty on the recurrent weight matrix can ensure that the eigenvalue of largest magnitude never exceeds 1, given an initialization with sufficiently small weights. Another proposal is based on the assumption that if the model exhibits from the beginning the same kind of asymptotic behavior as the one required by the target, then it's less likely for gradients to explode [Doy93]. However, it is not trivial to initialize a model in this specific regime. Gradient-clipping is another rather practical approach that involves clipping element-wise components of the gradient when they exceed a fixed threshold [MDK+11]. Finally, a solution for avoiding the vanishing gradient problem was proposed by Hochreiter et al. [HS97]. They removed the direct dependency of $\frac{\partial \mathbf{s}_i}{\partial \mathbf{s}_{i-1}}$ on the recurrent weight matrix [Bay15] by introducing memory cells. This modified network structure is called a Long Short-Term Memory Network and it has recently become the quasi-standard when using neural networks for real world sequence modeling tasks such as in speech recognition [GJ14], hand writing generation [Gra13] or statistical machine translation [SVL14b].

5.1.2 The Long Short-term Memory architecture

LSTMs are explicitly designed to remember information over long periods of time to eventually use that information when it is required at some later state. All recurrent neural networks have the form of a chain of repeated neural network modules. In the vanilla RNN described above, the recurrence module is a sigmoid layer that receives the previous state vector and the current observation vector as input and outputs the new state vector. The LSTM also exhibits this chain-like structure but instead of one layer, its recurrence module consists of four interacting neural network layers that manipulate the LSTM's so-called memory, depicted by the straight horizontal line running through the top in Fig. 5.2. The layers in the recurrence module can be thought of as three types of logical gates that delete, write and read information to/from the memory.

The first sigmoid layer in the LSTM is called the "forget gate" as it decides what information to remove from the memory. It takes as input the previous state $\mathbf{s}_{t-1}$ together with the input $\mathbf{x}_t$ and outputs a number between 0 and 1 for each component in the memory vector $\mathbf{c}_{t-1}$:

$$\mathbf{f}_t = \sigma(W_f \cdot [\mathbf{s}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

A strict 0 means setting the corresponding component in the memory vector to zero while a strict 1 would leave the memory unchanged. The second gate is called the "input gate" and consists of two neural network layers. The input gate decides what new information we are going to store and where in the memory we are going to store the information. Here, $\tilde{\mathbf{c}}_t$ represents the new memory vector and $\mathbf{i}_t$ determines the proportion by which

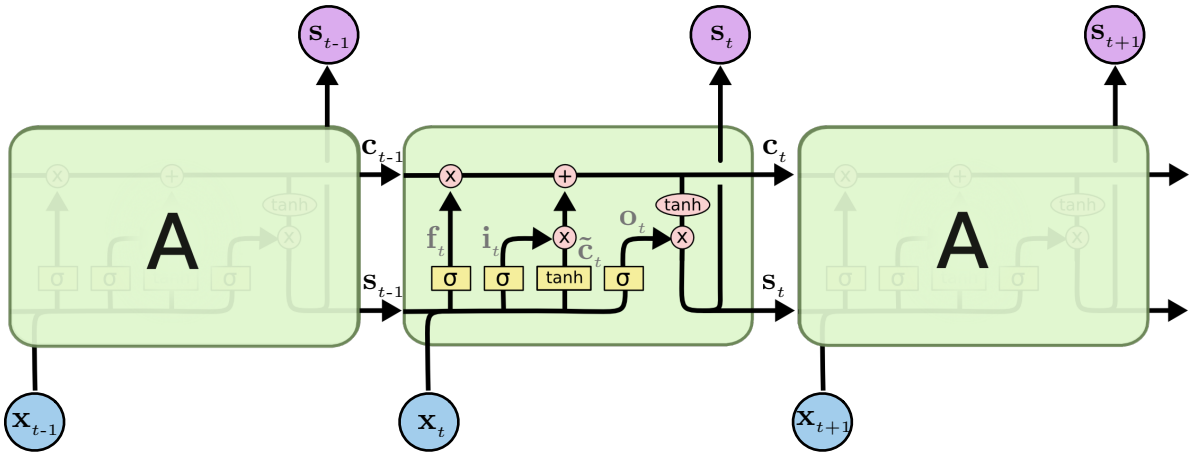


Figure 5.2: Computational graph executed within a LSTM cell. Black arrows depict the data flow, i.e. the transfer of vectors. Red circles depict element-wise operations on vectors and yellow boxes depict neural network layers together with their activation functions, where σ denotes the logistic function and *tanh* the tangens hyperbolicus. The figure was taken and adapted from a blog post on LSTMs[1].

the current data inside the memory will be overwritten by the new information $\tilde{\mathbf{c}}_t$:

$$\mathbf{i}_t = \sigma(W_i \cdot [\mathbf{s}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$\tilde{\mathbf{c}}_t = \sigma(W_c \cdot [\mathbf{s}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c)$$

Once the outputs of the gates are calculated, we can use these outputs to manipulate the current memory:

$$\mathbf{c}_t = \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \cdot \tilde{\mathbf{c}}_t$$

Finally, the LSTM's new state $\mathbf{s}_t$ is a function of the memory and, as in the vanilla RNN, the previous state $\mathbf{s}_{t-1}$ and the input $\mathbf{x}_t$:

$$\mathbf{o}_t = \sigma(W_o \cdot [\mathbf{s}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$\mathbf{s}_t = \mathbf{o}_t \cdot \tanh(\mathbf{c}_t)$$

All bold lower-case variables are vectors with d dimensions, i.e. the forget gate $\mathbf{f} \in \mathbb{R}^d$, the input gate $\mathbf{i} \in \mathbb{R}^d$, the output gate $\mathbf{o} \in \mathbb{R}^d$, the memory and its update $\mathbf{c}, \tilde{\mathbf{c}} \in \mathbb{R}^d$ and the state $\mathbf{s} \in \mathbb{R}^d$. The input variable $\mathbf{x} \in \mathbb{R}^m$ can have a different number of dimensions m. All upper-case W.'s are parameter matrices, i.e. $W_i, W_c, W_o \in \mathbb{R}^{d \times (d+m)}$, which, together with the bias vectors $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^d$, constitute the parameters of the LSTM. The square brackets $[\cdot, \cdot]$ represent the concatenation of two vectors. Although an LSTM contains a more complex recurrence module, its parameters can be learned in the same way as the vanilla RNN, using backpropagation through time with stochastic gradient descent as optimization routine. An exhaustive exposition of the LSTM network is beyond the scope of this work. Therefore, we forward the interested reader to the original article of Sepp Hochreiter [HS97].

[1] `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, Last access: 16.08.2019.

5.2 Experimental setup

Distinction of ECOM from F2F in the sequential setting Our assumption is that the genuine behavior of card holders or, likewise, the malicious behavior of fraudsters, is controlled by some latent, yet consistent, qualities. With its state variables, the LSTM is in principle capable of identifying these qualities from the sequence of observations.

In the real world, societal conventions, official regulations or plain physics impose constraints on the possible variability of observations and thereby on the complexity of the qualities that control them. For instance, opening hours strictly limit when and where customers are able to purchase their goods or services. Geographical distances and traveling modalities restrict the possibilities of consecutive transactions. We can assume that face-to-face transactions respect, to some degree, these real world constraints. In contrast, e-commerce transactions or rather their corresponding online purchases are widely unconstrained, according to both time and place. There is hardly any attribute that could not genuinely change in arbitrary ways from one transaction to the next.

We hypothesize that the presence of real-world constraints in face-to-face transactions gives rise to more obvious behavioral patterns with less variation. In that case, a sequence learner should benefit from the more consistent sequential structure.

Sequence data sets We create the two sequence data sets in the following way: We group all transactions by card holder ID and sort the transactions of each card holder by time. As a result, we obtain a temporally ordered sequence of transactions for each card holder. In the remainder of this chapter, we denote such sequence as the *account* of a card holder and the entire set of all accounts as the *sequence data set*. We further split the sequence data set into two mutually exclusive sets: One sequence data set contains only e-commerce transactions and the other only face-to-face transactions.

As introduced in Section 3.7, an account is considered *compromised* if it contains at least one fraudulent transaction and it is considered *genuine* if it contains only genuine transactions. We employ the same account-based sampling process as in Section 3.7 to build the training set. With probability $p_g = 0.9$ we randomly pick an account from the set of genuine accounts and with probability $1 - p_g$ we pick an account from the set of compromised accounts. This process is repeated 10^6 times to create a training set with one million accounts. The de facto fraud ratio on transaction level is still smaller than 1:10 but we found that this simple approach works well in practice. As in previous experiments, we introduce one week gap between the end of the training period and the start of the test period in order to account for the labeling delay in a production setting. See Table 5.1 for details regarding dataset sizes and time periods.

Dataset alignment Both the random forest classifier and the LSTM are trained to predict the label of individual transactions. However, there is one difference that we need to take into account in the experiments. With an LSTM we can only predict the label of a transaction after having seen several transactions that precede it, whereas with the random forest we do not require any preceding transactions. To improve the compara-

	Training set (01.03.–25.04.)		Validation set (26.04.–30.04.)		Test set (08.05.–31.05.)	
ECOM	$2.9 \cdot 10^6$	1.48%	$0.6 \cdot 10^6$	0.38 %	$3.3 \cdot 10^6$	0.42 %
F2F	$4.3 \cdot 10^6$	0.81%	$0.7 \cdot 10^6$	0.07 %	$4.7 \cdot 10^6$	0.05 %

Table 5.1: Dataset sizes and fraud ratios.

bility of the results, we accounted for this difference by discarding all transactions that were not preceded by at least 9 preceding transactions. The transaction for which we predict the class label is the last transaction in such a sequence. The LSTM iterates over all transactions in the sequence (without taking into account their class labels) and finally makes a prediction on the last transaction. The random forest uses only the last transactions during training and testing. With this protocol we can ensure that the random forest and the LSTM can now be trained, validated and tested on identical sets of labeled transactions.
To study the impact of the length of the input sequence on the LSTM prediction accuracy, we defined two scenarios: In one scenario the LSTM was trained with sequences of length $k = 5$ (`SHORT`), in the other scenario the LSTM was trained with sequences of length $k = 10$ (`LONG`).

Features We used the same feature sets as described in the data augmentation experiments in Section 4.4: The base set of raw features (`BASE`), a set which additionally includes the `tdelta` feature (`TDELTA`) and a feature set that includes both the `tdelta` feature and the feature aggregates (`AGG`). As before, we removed the identifiers of card holders and merchants to encourage generalization to new frauds. The features were encoded as follows:

- **Ratio-scaled variables:** We applied gaussian normalization to ratio-scaled variables such as the transaction amount or the credit limit to center the variable on $\mu = 0$ with a standard deviation $s = 1$. This normalization has no effect on tree induction in the random forest classifier, but it accelerates the convergence of gradient-based optimizers in neural networks.

- **Categorical variables:** In case of the random forest classifier, categorical variables can be used just as-is. We only mapped each value to an integer number.
  In case of neural networks, we wanted to avoid having very high-dimensional one-hot encoded feature vectors. Therefore, we employed a label encoding mechanism which is quite popular in the domain of natural language processing and neural networks [CWB+11b, RP04, TWY+15] and is applicable to arbitrary other categorical variables apart from words [GB16]. For a categorical variable with its set of values C, we assigned each value a random d-dimensional weight vector $\mathbf{v}$, that was drawn from a multivariate uniform distribution $\mathbf{v} \sim \mathcal{U}([-0.05, 0.05]^d)$, with $d = \lceil log_2(|C|) \rceil$. The feature values and their corresponding vectors (vector embeddings of the feature values) are stored in a dictionary. To encode a particular value of the categorical variable, we look up the value of the feature in the

RF		LSTM	
min-sample-leaf	$\{1, 3, 10\}$	learning-rate	$\{10^{-2}, 10^{-3}, 10^{-4}\}$
splitting-criterion	$\{\text{gini}, \text{entropy}\}$	dropout-rate	$\{0.2, 0.5, 0.8\}$
max-features	$\{5, 10\}$	LSTM-nodes (per layer)	$\{20, 100\}$
trees	$\{100, 600\}$	hidden nodes	$\{50, 100\}$

Table 5.2: Hyper-parameters considered during grid search.

dictionary and retrieve its vector. The embedding vectors are part of the model's parameters and can be adjusted jointly during parameter estimation.

- **Time feature:** We considered the time feature as a composition of several categorical variables. For each temporal resolution of the time feature, i.e. year, month, weekday, day, hour, minute and second, we defined a categorical variable in the same way as described above.

Classifiers The Long Short-term Memory network had two recurrent layers, a single hidden layer and a Logistic Regression classifier stacked on top of the hidden layer. The Logistic Regression classifier can be trained jointly with the LSTM state transition model via error back-propagation. We applied dropout [SHK+14] to the LSTM nodes to regularize the parameters and trained the entire model by minimizing the cross-entropy between the predicted and the true class distribution with the ADAM[2] algorithm. Our implementation is based on the deep learning library Keras (see Appendix B). We used the random forest implementation from SciKit-Learn [PVG+12] (see Appendix B). The hyper-parameters of both the LSTM network and the random forest were set via grid-search over a coarse grid spanned by a subset of all hyper-parameters (see Table 5.2). Then, we selected the configuration that maximized $AUCPR_{0.2}$ score on the validation set (see Section 3.3).

Jaccard Index To explore qualitative differences between the two approaches, we used the Jaccard Index to measure the degree to which two classifiers are similar in terms of the frauds they detect. Given two result sets (true positives) A and B, the Jaccard Index is defined as $\mathcal{J}(A, B) = |A \cap B| \,/\, |A \cup B|$. The two sets of true positives were obtained by picking one point on the precision-recall curve - the point that corresponds to a recall of 0.2. By fixing the recall at a particular value, the number of true positives and the number of false negatives is defined implicitly. For instance, the F2F test set (see Table 5.1) contains 2,350 frauds ($0.0005 \cdot 4.7 \cdot 10^6$). A recall, fixed at 0.2, partitions the frauds into 470 true positives and 1,880 false negatives. With this analysis, we aim to quantify the agreement of a pair of classifiers in terms of the identity of their 470 true positives. We quantify this agreement by means of the Jaccard Index.

[2]ADAM is a gradient descent optimization method with an adaptive per-parameter learning rate schedule

		F2F			
		AUCPR(μ)		AUCPR$_{0.2}$(μ)	
	Features	RF	LSTM	RF	LSTM
SHORT	BASE	0.138	0.200	0.086	0.107
	TDELTA	0.170	0.231	0.095	0.118
	AGG	0.241	0.246	0.112	0.113
LONG	BASE	0.135	0.229	0.084	0.106
	TDELTA	0.172	0.217	0.095	0.102
	AGG	0.242	0.236	0.112	0.110

Table 5.3: Classification results in terms of $AUCPR$ and $AUCPR_{0.2}$ on face-to-face sequences.

5.3 Results

We trained a model for each combination of feature set, data set and sequence length and then tested its classification performance on the hold-out test set. In case of random forests, the length of the input sequence has no influence on the model since only the last transaction from the input sequence was used. We evaluated the trained models on each of the 24 test days individually and report their mean performance in terms of the $AUCPR$ and $AUCPR_{0.2}$.

Table 5.3 and Table 5.4 show a summary of the results for the face-to-face and e-commerce data sets. A first observation is that the overall detection accuracy is much higher on e-commerce than on face-to-face, which can be explained by the higher fraud ratio in e-commerce. Secondly, longer input sequences seem to have no effect on the detection accuracy, neither for F2F nor for ECOM. Third, taking into account previous transactions with an LSTM noticeably improves fraud detection on F2F. However, such improvement is not observable on ECOM – rather are the results from the static learning and the sequence learning approach surprisingly similar.

Another observation confirms the finding that feature aggregations improve fraud detection. Their impact is much more evident on ECOM as it is on F2F. The observation that feature aggregations help in cases where the sequence model does not, suggests that these two forms of context representation are not correlated and that the approaches are complementary. Whatever information LSTM-states are tracking in the transaction history, it is not the same as what we manually added through aggregations.

The LSTM improves fraud detection on face-to-face transactions in terms of $AUCPR$ and $AUCPR_{0.2}$. We were curious where this improvement is coming from. Figure 5.3 displays the precision-recall curves of all model variants. In Figure 5.3a we can see, that the PR-curves of the random forest models have a high peak in precision at low recall levels but decay quickly as the recall increases. In contrast, the LSTM models have a

		ECOM			
		AUCPR(μ)		AUCPR$_{0.2}$(μ)	
	Features	RF	LSTM	RF	LSTM
SHORT	BASE	0.179	0.180	0.102	0.099
	TDELTA	0.236	0.192	0.124	0.107
	AGG	0.394	0.380	0.158	0.157
LONG	BASE	0.179	0.178	0.101	0.104
	TDELTA	0.228	0.238	0.118	0.115
	AGG	0.404	0.402	0.158	0.160

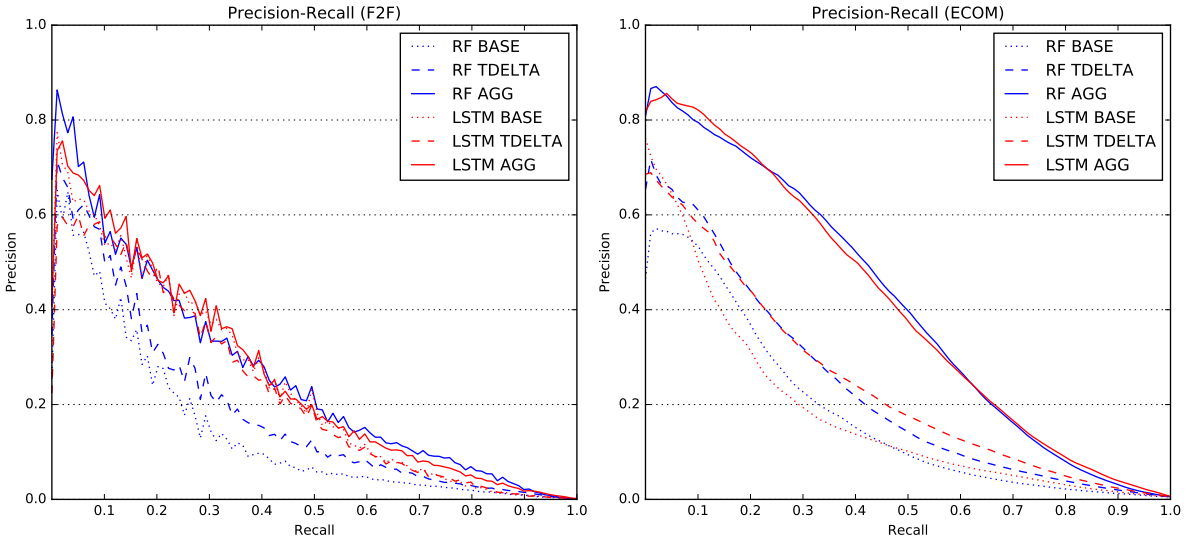
Table 5.4: Classification results in terms of $AUCPR$ and $AUCPR_{0.2}$ on e-commerce sequences.

slightly lower precision for low recall levels but retain a higher precision as the recall increases. However, there is one interesting exception: Once we add aggregated features, the PR-curve of the random forest increases by a noticeable margin up to a performance that is on par with the LSTM models. We can not observe such gain for LSTMs. On e-commerce transactions (see Fig. 5.3b), the PR-curves of the random forests and the LSTMs are almost identical across all feature sets. Random forests and LSTMs benefit from aggregated features by the same margin.

5.3.1 Evolution of prediction accuracy

When plotting the $AUCPR$ scores for individual test days, we can see in Fig. 5.4 that the predictions from both classifiers exhibit large variations across days. However, since the curves seem to follow the same trend, we have reason to assume that on some days the detection problem is more difficult than on others. For instance, on face-to-face transactions both classifiers have their minimum with respect to the $AUCPR$ in the time periods 9.05.–10.05. and 25.05.–26.05. Through manual inspection, we tried to link transactions from these days to calendar or public events, but we could not find an obvious answer that would explain the poor performance.

Nevertheless, the daily variation of the prediction accuracy on e-commerce and face-to-face transactions is not completely unexplainable. Starting from the observation that the $AUCPR$ curves in Fig. 5.4 seem to follow a common trend regardless of the choice of algorithm or feature set, we conclude that the variation is an intrinsic property of the data set. Thus, when we want to find explanatory factors for the variation, we should not look for them in the features or models but in the data set itself. We made a first step towards such inquiry by looking at the fraud density per day (see Fig. 5.5). This simple statistic revealed that the fraud density can vary to a large extent from one day to another. Looking at the density curve for face-to-face, we notice that the densities from



(a) F2F (b) ECOM

Figure 5.3: Precision-recall curves averaged over all days in the test set.

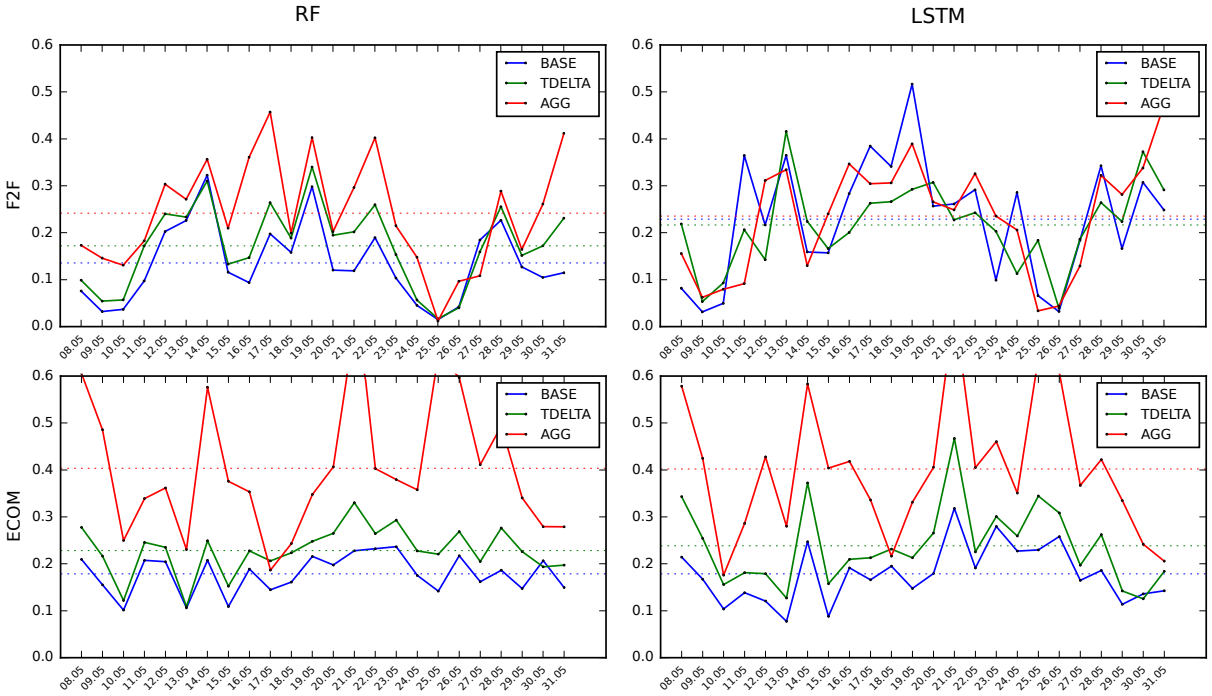


Figure 5.4: Evolution of $AUCPR$ over test days in the sequence classification experiments. Horizontal dashed lines indicate the mean $AUCPR$ for each curve.

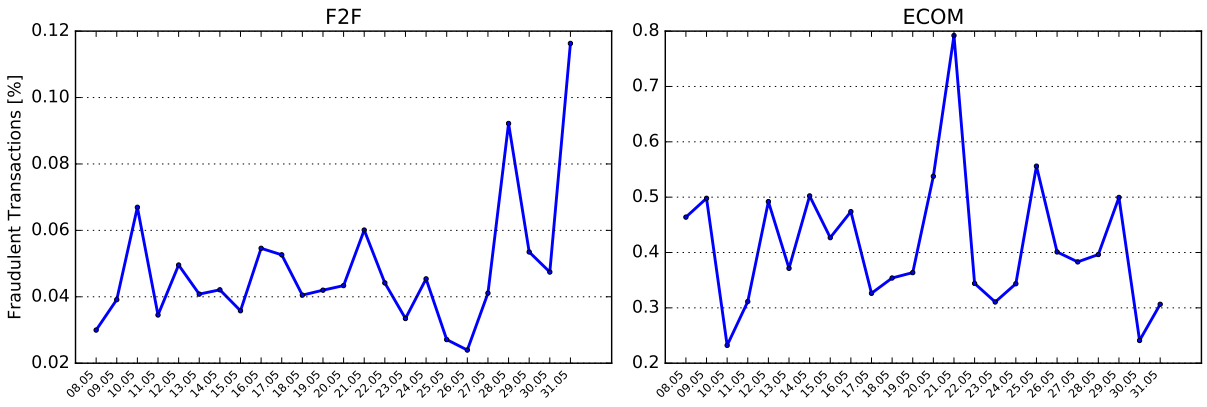


Figure 5.5: Fraud density on each test day. Density is given as the percentage of fraudulent transactions in all transactions of a day.

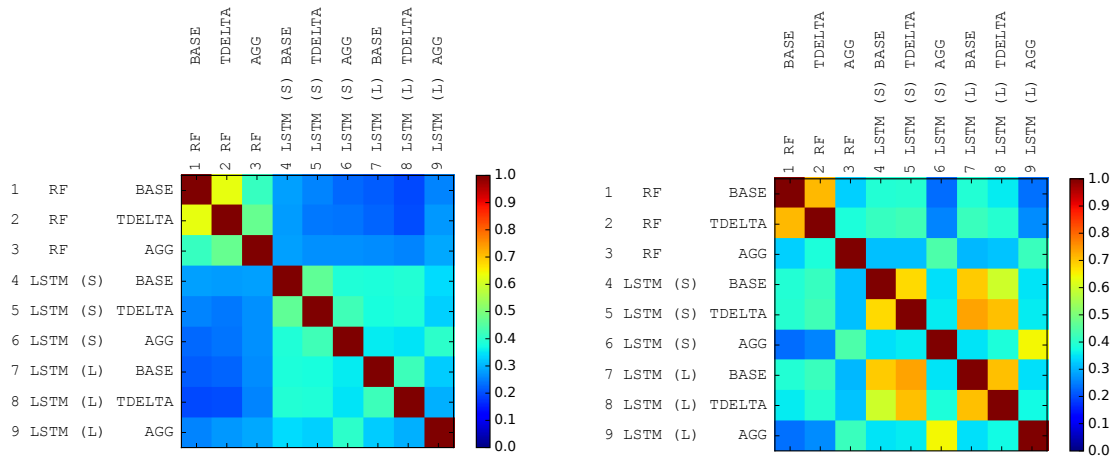
25.5. to 31.5. can be associated with the $AUCPR$ values on the corresponding days. Similarly, the e-commerce fraud densities in the first, middle and last test days seem to be correlated with the $AUCPR$ values on these days (more noticeable in the `AGG` curve). In both scenarios, the nature of the correlation is not linear and we can always find an exception that contradicts a general statement about the direct correlation between fraud densities and prediction accuracy. However, we believe that the fraud density is, among many others, one part of the answer explaining the variation in the quality of predictions across days.

5.3.2 Overlapping predictions

In this analysis we take a closer look at the true positives the two approaches managed to identify. From the set of all trained models, we select a pair of models and compared their prediction vectors. The decision threshold is again chosen such that it corresponds to a recall level of 0.2. All predictions with a score higher than the threshold are considered as positive predictions and all others as negative predictions. By fixing the recall, we ensure to have an equal number of true-positives in the result sets of a pair of models. We are interested in whether the true positives of the random forest are actually identical to the ones of the LSTM. We measure the overlap of the true positive sets of a pair of models with the Jaccard Index. Figure 5.6 displays all pairwise comparisons as a heatmap.

On both heatmaps, we observe four quite distinct areas. The two areas that correspond to intra-model comparisons and the two areas that correspond to inter-model comparisons[3]. The Jaccard Indices suggest that both the random forest models and the LSTM models are consistent in the frauds they detect. This property is slightly more pronounced in the random forest comparisons. Moreover, the random forest models and the LSTM models tend to detect different frauds. On F2F, the random forest models agree on 50.8% of their true positives on average and the LSTM models on 37.8%. Between the two model classes we observe an average agreement of only 25.2%. This is similar on the e-commerce data set with average intra-model agreements of 47.5% (RF) and 50.8% (LSTM) and an average inter-model agreement of only 35.0%.

[3]The Jaccard Index is symmetric and so is the matrix of all pair-wise comparisons. We kept the redundant comparisons to ease interpretation.



(a) F2F

(b) ECOM

Figure 5.6: Pairwise comparison of the sets of true positives of two models measured with the Jaccard Index and color-coded in a heatmap.

There is one exception to this general observation. The models that were trained with aggregated features tend to detect a common unique set of frauds, that were neither detected by random forests nor LSTMs without aggregated features. This property is more pronounced on ECOM than on F2F.

5.4 Discussion

During our experiments we found that the application of Long Short-Term Memory networks to such structured data is not as straight-forward as one might think. Therefore, we would like to share some observations that might be useful for practitioners.

Model regularization: When we are dealing with a temporal process for which we aim to forecast some properties of future events, no collection of historic data points can truly fulfill the requirements one would demand from a representative validation set. The prediction accuracy on days right after the end of the training set is better than on days further into the future, suggesting a time-dependency of the conditional distribution. When we choose the days right after the training period as validation set, the results on this set would suggest to apply small regularization on the model. But this choice has the contrary effect on the performance on days further into the future. An exact and highly-confident model of today's data is probably badly wrong in a few days, whereas a less confident model of today is still valid in a few days. This is less of a concern for ensemble classifiers such as random forests, but it is for neural networks. A neat workaround is to use *Dropout* on the network structure. Dropout samples smaller networks from the complete structure, trains these independently and finally averages the weights of these smaller networks.

Online learning: Stochastic gradient descent and the many variants that have been developed for training neural networks (ADAM, RMSprop, Adagrad) are capable of iteratively updating the model even from imprecise errors that have been estimated from small sets of training examples. This property blends in well with the requirement

of businesses to keep their detection models up-to-date with the incoming stream of transactions.

Remarks on LSTM training: Due to its recurrent structure, the LSTM is prone to overfitting on the training data even when the LSTM-layers have only few nodes. Therefore, it's recommendable to start off with a rather small structure and carefully increase the size as long as there is reason to expect further generalization performance. We noticed, that a ℓ_2-penalty leads to much smoother convergence and consistently better optima than a ℓ_1-penalty. The ADAM optimizer worked much better than stochastic gradient descent in our experiments since it estimates a proper learning rate schedule on the go.

Sequence modeling and feature aggregations: The unfortunate disadvantage of neural networks is the fact that they are not surrounded by a rigorous reasoning framework with clear semantics such as probabilistic models. They are powerful function approximators and we used them to approximate the conditional distribution over classes given a sequence of transactions. However, the internal mappings from a transaction to a latent state and from one state to the next are not equipped with an intrinsic interpretation due to the lack of such framework. Therefore, we can not directly query it for interesting details, i.e. finding the most likely state sequence given some transaction sequence or the most likely next transaction given the current state. Considering the results we obtained when aggregating the transaction history through manually engineered features or a sequence learner, we would need such inference mechanisms to better understand which aspects of the history we are actually aggregating with a sequence learner and why these aspects are so different from the hand-engineered features in some contexts. Quantitatively, when we add aggregated features, the sequence model becomes redundant.

5.5 Summary

In this chapter, we employed Long Short-term Memory networks as a means to aggregate the historic purchase behavior of credit-card holders with the goal to improve fraud detection accuracy on new incoming transactions. We compared the results to a baseline classifier that is agnostic to the past. Our study showed that offline and online transactions exhibit very different qualities with respect to the sequential character of successive transactions.

Regarding research question **RQ1**, we can state that the summary of users' purchase sequences by means of an LSTM's latent states improves the prediction performance on face-to-face transactions in terms of $AUCPR$ by around 50% over a random forest classifier when it is trained on single transactions using basic features. However, we can not observe any improvement on e-commerce transactions.

As we have seen in the previous chapter, feature aggregates strongly improve the prediction performance and we asked ourselves the question whether the LSTM may eliminate efforts of manually defining and extracting such features from the data. Based on our results, we answer research question **RQ2** as follows: Regardless of the data set, the usage of feature aggregates produces the best prediction performances with both the random forest and the LSTM. It is difficult to say whether the LSTM recovers similar

information as feature aggregates since we can only judge by looking at the overall performance. At least the results on e-commerce strongly contradict this hypothesis. Whatever the LSTM learned to condense within latent states, it is not the same as feature aggregates.

In part, this observation already answers our last research question **RQ3**: Our analysis of both classifiers' predictions revealed that modeling the transaction history with an LSTM yields a sets of true positives that differ consistently from the true positives detected with a random forest across all feature sets. This might suggest a combined approach in which an ensemble of several structurally different methods gets to perform fraud detection. However, a naive combination does not necessarily improve the detection because the different approaches do not exhibit the same level of precision. Therefore there is no simple means to determine a common threshold among both approaches such that we would collect the complementary sets of true positives without also collecting the false positives from both approaches.

Our study required sequences of some fixed minimum length. However, from a practical point of view, we desire an algorithm that is able to make predictions on all transactions including the very first ones in an account. As in many other applications, new accounts confront the system with the cold-start problem. We would need to assess actions of a user without having access to the user's context. The cold-start is still an open problem and it might require its own tailored solution.

Chapter 6

Searching Feature Aggregates

As observed in previous chapters, even structurally simple feature aggregates can serve as good predictors for fraud detection. Thereby we employed manually defined aggregates which represent, for instance, the purchase amounts summed over a set of previously issued transactions of a card holder. These aggregates characterize single transactions relative to the purchase context they appear in. The purchase context serves as a proxy for a card holder's previous activity against which the most recent transaction should be compared. The aggregate itself is a statistic calculated on transactions in the purchase context.

The aggregates used so far (see Section 4.1) were defined manually at best-effort while taking into account suggestions about feature engineering from related work and fraud experts. Such features are heavily used in production systems that operate in near real-time and, as we have seen throughout this thesis, for quite a good reason: Without requiring any extensive modeling or parameter estimation process they lift the detection performance to a new level. However, a few questions remain as to why these features work so well in practice. Have we been lucky with our manual selection of the 14 feature aggregates or could we have chosen a completely different set and still obtain decent performance? Wouldn't just two or three features suffice or does every additional one add a constant improvement? For answering these questions, we are going to explore a space of aggregates spanned by their constituent characteristics, with the aim to identify commonalities that result in good performance. To that end, we develop a concept for describing the aggregation space, provide an algorithm for fast feature extraction and finally explore the space by means of feature selection methods. Specifically, we address the following two research questions:

- **RQ1**: Can we leverage feature selection methods for finding a set of aggregates that yields superior predictive performance than the feature aggregates that have been proposed in literature and also been used by practitioners in the domain?
- **RQ2**: Are there commonalities among the best feature aggregates?

6.1 Generalized feature aggregates

The aggregates we consider here are based on previous work from Whitrow et al. [WHJ+09] and Bahnsen et al. [BASO16]. While Whitrow et al. proposed aggregations based on

absolute feature values (e.g. counting all transactions where PIN authentication was used), Bahnsen et al. modified the definition to relative feature values (e.g. counting all transactions where the same authentication method was used as is used now) and they also used combinations (pairs) of features to select identical transactions.

While several absolute aggregates together give raise to an absolute activity record of an account, several relative aggregates together create a relative activity record of an account in the light of a specific transaction. In both cases, the activity record changes its state with each incoming transaction. It is important to note that, in an absolute activity record, each statistic is equipped with well defined semantic and, most likely, it has a simple distribution whereas in an relative activity record the statistic's semantic can change with each transaction and its distribution is a mixture of several distributions. In this sense, the relative activity record covers a broader spectrum of account activity while, at the same time, requiring less a-priori expertise for feature construction. However, the relative record is only useful if combined with the raw transaction features. And a subsequent classifier must be flexible enough to disentangle the connection between a transaction's raw value and the statistic calculated on the basis of that raw value. Since decision trees and multi-layer perceptrons can identify such higher order correlations (i.e. class label, raw feature, aggregate feature) - and as we have seen do they yield pretty much the same performance -, we assume insufficient flexibility to not be a major concern. Therefore, we have used the "relative" aggregation approach throughout this thesis and, in this chapter, we are going to explore its corresponding aggregation space.

Feature aggregates can be easily integrated in any existing classification process and they seem to readily improve the prediction accuracy. However, the existing definition of these aggregates (Section 4.1 from Bahnsen et al. [BASO16]) seemed a little clumsy and unnecessarily limited in terms of the variety of possible aggregates we might want to consider. Moreover, in real-time fraud detection there is a maximal time budget of around 100 milliseconds until a transaction has to be either approved or rejected. Any longer delay is perceivable by the card holder, lowering the quality of service and can therefore not be tolerated. In order to incorporate aggregates in real-time decision making, at least the corresponding extraction must finish in a fraction of the maximal time budget.

Unifying a broad range of possible feature aggregates under a common concept is not only organizationally appealing but it is also a structural prerequisite for an efficient implementation. Therefore, we first conceptualize feature aggregates and then introduce a small C-library with OpenMP[1] and CUDA[2] support, which enables fast extraction through massive data parallelism. This concept assumes that aggregates are calculated on a large number ($> 10^7$) of small ($< 10^3$) and independent groups. A group could be the purchase history of a card holder or a merchant. A median card holder issues about one transaction in eleven days. Thus, a purchase history of length 10^3 represents several years of the card holder's activity. We make two further assumptions about the data set and its variables:

[1] `https://www.openmp.org`, Last access: 18.08.2019.

[2] `https://developer.nvidia.com/cuda-zone`, Last access: 18.08.2019.

- All variables can be converted to numerical types, such that $<, ==$ and $-$ are meaningful operations.[3]
- The instances of a group are ordered by time and some level of ignorance towards instances from the distant past is acceptable.

Our aggregates are best described as *identity* aggregates: For a single transaction we determine the set of transactions that are identical to the current one and then we apply one of several aggregation operations on transactions in the set. In contrast to manually defining specific aggregates as in the experiments in Chapters 4 and 5, we conceptualize the constituents of such aggregates which allows us to easily express a broad spectrum of possible aggregates. In practice, we use JavaScript Object Notation (JSON) to specify the parameters of an aggregate but, for the sake of an uncluttered presentation, we explain the concept along an equivalent pseudo-syntax. The family of aggregates covered by our concept corresponds to basic SQL queries and the syntax for defining aggregates is also very similar. See Figure 6.1 for examples and the next section for a detailed description of the constituents of this syntax.

```
(1) SUM amount FROM [-1000,0] | timestamp WITHIN [-24,0]
(2) COUNT . FROM [-100,0] | country WITHIN [0,0] AND merchant WITHIN [0,0]
(3) RANGE timestamp FROM [-1,0]
```

Figure 6.1: Three examples of aggregates expressed in the pseudo-syntax: (1) The sum of amounts spent within the previous 24 hours, (2) The number of transactions issued in the same country and at the same merchant (3), The elapsed time since the last transaction. The statement before the conditioning bar "`|`" defines the target variable whose values get aggregated and the statements after the conditioning bar are used to select transactions from the history to which the aggregation operator will be applied.

Executing equivalent SQL-queries on a single user's transaction history takes roughly around 0.3 milliseconds per query if issued against an in-memory MySQL database[4]. Since we have to issue one query per transaction, even on a small set of one million transactions it would amount to 300 seconds for calculating only one single feature aggregate. Therefore, exploring a space of tens of thousands of different feature aggregates in reasonable time is only possible with a sufficiently fast feature extraction method.

6.1.1 Concept

The examples in Fig. 6.1 already showcase the elementary constituents of an aggregate. On a high level each aggregate is defined by two statements separated by a vertical bar

[3]Even for a categorical variable, "$-$" can be meaningful to indicate identity: $x_j - x_i = 0 \iff x_j = x_i$

[4]Tested with a single table of 57 transactions in MySQL Community Server 8.0.16 (storage engine = MEMORY) on a MacBook Pro 15" Core i7 2.5Ghz. Certainly, by employing server hardware, proper indexes and caching strategies one can shave off an additional factor from the execution time but the timing will settle somewhere in the range of tenths of a millisecond. We are interested in the order of magnitude and not the exact timing.

|. The first statement defines the *target* and the second statement behind the vertical bar defines the *context*:

- Target: A variable on which we compute a statistic
- Context: A set of constraints imposed on instances

We create a new feature by applying an aggregation to every instance in the data set. To evaluate the aggregate for a particular pivot instance, we need to (1) select the set of instances that comply with all constraints in the context and (2) apply an aggregation operation on the target feature of all instances within the set.

Context The context is a set of constraints. An instance must comply with all constraints in the context to be part of the set of selected instances. Each constraint is a boolean statement and all constraints are combined via a boolean conjunction with the `AND` keyword. A constraint is a 2-tuple and it is defined on the values of a feature:

- A feature name
- A closed interval $[p, q] \subset \mathbb{R}$

The interval $[p, q]$ enables selection of instances by value. Since aggregates are calculated relative to a pivot instance, a constraint's interval sets the bounds for an accepted difference between the pivot instance's feature value and any other instance's feature value. In the pseudo-syntax, constraints have the form `feature_name WITHIN [p, q]`. In the following notation, we use c to index the constraints within a context.

Target A target is a 3-tuple operating on the set of instances selected by a context and it consists of the following components:

- An aggregation operator
- A feature name
- A closed interval $[u, v] \subset \mathbb{N}$

The target feature can be any feature whose values can be reasonably aggregated with one of the following operators: `SUM, MEAN, COUNT, MIN, MAX, RANGE, MEAN-RANGE`. The interval $[u, v]$ enables selection by index, relative to the index of the pivot instance. In the pseudo-syntax, a target is defined by `operator feature_name FROM [u, v]`. We use a to index targets.

Aggregation Let $\mathcal{X} = \{x_j\}_{1 \leq j \leq N}$ be the ordered sequence of N instances in a group, e.g. the sequence of time-ordered transactions of a single card holder, indexed by j. The x_j's are multi-variate objects comprising values of several features and we loosely refer to the value of target feature a or the value of constraint feature c in x_j by x_j^a and x_j^c, respectively. By $x_i \in \mathcal{X}$ we denote the pivot instance for which we wish to evaluate the aggregate.

We denote by $\mathcal{S}_i$ the set of instances which comply with all constraints in the context. The selection of these instances depends on their difference to instance i with respect to their feature values and their indices. To that end, we make use of the value-based interval associated with constraints $[p^c, q^c]$ and the index-based interval associated with a target $[u, v]$. The intervals set the bounds for the allowed difference between two instances. An instance x_j is in $\mathcal{S}_i$ if all (over all constraints c in the context) value-differences $\alpha^c_{ji} = x^c_j - x^c_i$ are in $[p^c, q^c]$ and the index-difference $\beta_{ji} = j - i$ is in $[u, v]$. Therefore, the set $\mathcal{S}_i$ is given by:

$$\mathcal{S}_i = \{x_j \in \mathcal{X} | \forall c : \alpha^c_{ji} \in [p^c, q^c] \wedge \beta_{ji} \in [u, v]\}$$

The selection applies to both continuous and discrete features in the same way, when discrete features are encoded as integers[5]. Once we have $\mathcal{S}_i$, we can apply an aggregation operator on the target feature to compute the value of the aggregate for x_i from all instances in $\mathcal{S}_i$. We define the following aggregation operators on a generic set $\mathcal{S}$ of selected instances:

- SUM: $\sum_{x \in \mathcal{S}} x^a$
- MEAN: $\frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} x^a$
- COUNT: $|\mathcal{S}|$
- MIN: $\min(\{x^a | x \in \mathcal{S}\})$
- MAX: $\max(\{x^a | x \in \mathcal{S}\})$
- RANGE: $\max(\{x^a | x \in \mathcal{S}\}) - \min(\{x^a | x \in \mathcal{S}\})$
- MEAN_RANGE: $\frac{1}{|\mathcal{S}|}(\max(\{x^a | x \in \mathcal{S}\}) - \min(\{x^a | x \in \mathcal{S}\}))$

Calculating aggregates of this kind on a sequence of length N has time-complexity $T(N) \in \mathcal{O}(N^2)$, because for each instance in the sequence, we have to iterate through all previous instances to find the ones that comply with all constraints. In our use-case, transaction sequences of card-holders are either naturally short with $N < 10^3$ or they can be artificially clipped with the `FROM` keyword to trade precision of the statistic for performance. Notice that, regardless of the kind of aggregate, we only need to calculate the quantities α^c_{ji} and β_{ji} and perform comparisons against the interval bounds. All operators can be calculated online with the use of few temporary variables. The computations of different aggregates over millions of instances are entirely independent. Therefore, we abstain from the usage of sophisticated index structures but instead parallelize the computation over aggregates and instances.

[5]However, such selection does not make much sense for nominal variables except with singleton intervals $[0, 0]$ to indicate identity.

6.1.2 Implementation

The above concept is summarized as pseudo-code in Algorithms 6.1 and 6.3. The algorithm is conceptually trivial but computationally heavy due to the cascade of five for-loops. We split the pseudo-code in two procedures to point out the obtrusive potential of data parallelism. The first procedure in Algorithm 6.1 runs over all sequences, features and instances in a sequence and calculates the value of a single aggregate on a single pivot instance by means of Algorithm 6.3. The outer loops in the first algorithm are entirely independent of each other and therefore a parallel implementation does not require any form of message passing, locks or synchronization that would negatively impact the total run-time. In parallel computing lingua, such a problem is called embarrassingly parallel. If the aggregates were absolute, we could exploit dynamic programming to calculate the aggregate's value at one instance from the aggregate's values at preceding instances. However, with relative aggregates, an instance's set of context instances changes with the instance's attribute values, which requires us to recompute the context and the aggregate's value for every instance. Only if several aggregates share the same constraints, could we, theoretically, save some redundant computations. But this minor gain will very likely get depleted by the penalty introduced through concurrent memory access. Therefore, we preferred to keep the algorithm simple and flexible.

Algorithm 6.1 Computation of feature aggregates (Part 1)

Require:

Set of input sequences $\mathcal{X} = \{X_g\}_{1 \leq g \leq G}, X_g \in \mathbb{R}^{N_g \times D}$
Set of output sequences $\mathcal{Z} = \{Z_g\}_{1 \leq g \leq G}, Z_g \in \mathbb{R}^{N_g \times A}$ initialized with zeroes
Specification of aggregates $\Theta = \{\theta_a\}_{1 \leq a \leq A}$

```
1: procedure AGG(𝒳, 𝒵, Θ)
2:     for a ← 1, A do                          ▷ Loop over feature aggregates
3:         for g ← 1, G do                      ▷ Loop over sequences
4:             for i ← 1, N_g do                ▷ Loop over pivot instances
5:                 Z_g[i][a] ← AGGPIVOT(i, X_g, θ_a)
6:             end for
7:         end for
8:     end for
9: end procedure
```

We implemented the feature extraction algorithm[6] in C with OpenMP-support for CPU-based parallelism and CUDA-support for GPU-based parallelism. The OpenMP version parallelizes over different aggregates only, the CUDA version parallelizes over aggregates and instances. Input sequences $\mathcal{X} = \{X_g\}, X_g \in \mathbb{R}^{N_g \times D}$ of lengths N_g and number of input features D and the corresponding output sequences of extracted aggregates $\mathcal{Z} = \{Z_g\}, Z_g \in \mathbb{R}^{N_g \times A}$ are stored in contiguous blocks of memory. The specification of the a-th feature aggregate, θ_a, contains all the information about the target and

[6] `https://gitlab.padim.fim.uni-passau.de/jjurgovsky/feature-aggregations`, Last access: 19.08.2019.

the set of constraints as described in the concept above (Section 6.1.1), where feature names have been replaced by column indexes (a_x, c_x) of the input data matrix X_g and the aggregation operation is indexed by a_f. In practice, the subroutines are inlined to avoid unnecessary call overhead and the memory layout of specifications θ_a is adapted to the respective parallelization scheme to avoid unnecessary memory fetches. The subroutines GetTargetParams and GetConstraintParams extract the definition of the target and the definition of constraint c from θ_a, respectively. EvalAggOperator evaluates the aggregation operator on the selected instances (as described in Section 6.1.1). For evaluating any of the listed aggregation operators, we require four temporary values: The minimum, maximum and the sum over target values of instances that matched the pivot instance and the total number of matches. The subroutine UpdateTemps updates these temporary values whenever an instance matches the pivot instance. Certainly, it is redundant to count the number of matches if we specified the `sum`-operator. However, it is more efficient to update these values than to force different threads to follow different execution paths.

Algorithm 6.3 Computation of the value of an aggregate for a single instance (Part 2)

```
10: procedure AggPivot(i, X_g, θ_a)
11:     a_f, a_x, u, v ← getTargetParams(θ_a)
12:     sum, cnt, min, max ← 0, 0, 0, 0
13:     j_s, j_e ← getSequenceLimits(i, N_g, u, v)
14:     for j ← j_s, j_e do                ▷ Loop over previous instances j ∈ [i + u, i + v]
15:         match ← True
16:         for c ← 1, C_a do              ▷ Loop over constraints of aggregate a
17:             c_x, p, q ← getConstraintParams(c, θ_a)
18:             α ← X_g[j][c_x] − X_g[i][c_x]
19:             match ← match and (p <= α) and (α <= q)
20:         end for
21:         if match then
22:             sum, cnt, min, max ← updateTemps(X_g[j][a_x], sum, cnt, min, max)
23:         end if
24:     end for
25:     return evalAggOperator(a_f, sum, cnt, min, max)
26: end procedure
```

For ease of use, we compiled the extraction code as shared library and provide a Python wrapper to call into the shared library. The set of aggregates, that shall be extracted, is specified in form of a JSON file with equivalent syntax as introduced above. As an example, the actual JSON specification of the aggregates from Fig. 6.1 is given in the appendix in Fig. A.1.

6.1.3 Run-time evaluation

In order to get an idea of the run-time of the algorithm, we set up a small evaluation. From the entire data set consisting of around 48 million transactions, we extract the same

	Python	Cython	C		CUDA
#cores	1	1	1	14	GPU
total [s]	6100.514	316.663	147.634	13.804	2.559 (1.159)
sec / aggregate	435.751	22.619	10.545	0.986	0.183 (0.083)

Table 6.1: Wall clock run-time for extracting 14 aggregates with a parallel CPU implementation and GPU implementation averaged over five runs. Timing is reported in seconds. Run-times of Python and Cython implementations are added for matters of comparison. The timings in brackets are pure run-times of the GPU kernel excluding the time it takes to copy data from RAM to device memory.

14 aggregation features that we used in Chapters 4 and 5, and we report the average total wall clock time over five repeated runs. We are aware that the wall clock time is a very naive way of measuring the run-time of an algorithm and that it is subject to variations introduced by the hardware, operating system and any concurrently executed tasks. An in-depth evaluation is out of scope of this thesis and since we are only interested in the algorithm as a tool to render feature selection on aggregates feasible, we hope the reader bears with us and reads the results as indications of orders of magnitude rather than exact timings.

The entire data set consists of 3.5 million card holder sequences with an average length of 13.6. Half of the sequences are shorter than 8 transactions. The 14 extracted aggregates cover sums and counts as operators and between one to four constraints (one interval constraint on the timestamp and identity constraints on the terminal's country, the terminal's MCC code and the card entry mode). The index interval was set to $[-1000, 0]$ for all aggregates. We ran the extraction algorithm on a single CPU core, 14 CPU cores and the GPU[7]. The timings in seconds are summarized in Table 6.1.

First, it is interesting to see that the transition from the loosely typed and interpreted language Python to the strictly typed and compiled language C results in a 43x speed-up. Since the extraction algorithm mainly consists of nested for-loops, we suspect that the overhead from boxing/un-boxing `PyObject`s in each iteration accumulates and finally results in this drastic performance difference. As expected, the parallel implementations scale well over several cores. The CUDA timings in brackets are the run-times of the GPU kernel alone - without memory copies. About half the time is spent for data transfer between RAM and device memory. As an inner loop of a feature selection process, the feature extraction with the CUDA implementation is sufficiently fast because we can keep the data set in device memory between consecutive kernel calls. In the beginning of this section, we mentioned that an equivalent SQL query, issued against an in-memory database, takes about 0.3 milliseconds to calculate the value of one aggregate for a single transaction. With a tailor-made implementation, the same type of "query" takes, on average, 0.2 microseconds (C-1).

We consider the set of all aggregates, that can be expressed with our syntax, as the aggregation space. We can now query the space at specific points (i.e. the param-

[7]CPU: Intel Xeon Gold 5115 CPU x20@ 2.40GHz - GPU: NVIDIA Tesla P100 12GB x3584@1.33Ghz toolkit version 9.1. - GCC 5.4.0. compiler with optimization level 3.

eterizations of different feature aggregates), extract the aggregates and compare their performances in terms of credit card fraud classification to find better features than the ones we used so far.

6.2 Feature selection

Feature selection in supervised learning starts with a set of features and aims to select a significantly smaller subset of maximally relevant and least redundant features for a particular prediction task. Motivations for feature selection include a reduction of computation time both during learning and inference, a better understanding of the data and, in general, an improvement of the generalization performance of a classifier due to fewer dimensions of the training instances. Feature selection provides solutions to the problem of scoring, ranking and the selection of either individual or subsets of features based on their relevance to the target concept (i.e. the true, unknown function that maps instances to the class variable). Blum & Langley [BL02] compiled a list of five definitions of feature relevance which researchers have either explicitly introduced or implicitly assumed. We follow their definition of *incremental usefulness* as a means to characterize relevance:

Incremental Usefulness: Given a sample of data S, a learning algorithm L, and a feature set $\mathcal{A}$, feature x_i is *incrementally useful to L with respect to* $\mathcal{A}$ if the accuracy of the hypothesis that L produces using the feature set $\{x_i\} \cup \mathcal{A}$ is better than the accuracy achieved using just the feature set $\mathcal{A}$.

Other definitions do not depend on a learning algorithm and its predictive accuracy but instead they rely on some kind of scoring function that measures the statistical dependence between a feature and the class variable. However, the definition above describes a general end-to-end notion of relevance and it is especially natural for feature selection methods that search the space of feature subsets by incrementally adding or removing features to their current set. Exhaustively evaluating all feature subsets is usually infeasible due to the combinatorial explosion, which is the reason why researchers often resort to heuristic search methods. In the following sections, we provide a brief overview of feature selection methods with particular focus on how they quantify feature relevance and we review several common heuristics used to search the space of feature subsets.

6.2.1 Feature selection methods

Blum & Langley [BL02] group feature selection methods into three classes: *Embedded* methods incorporate feature selection as part of the training process and they are specific to the type of learning algorithm. *Filters* consider feature selection as a pre-processing step and they score and rank features independently of the chosen algorithm. And *Wrappers* utilize the classifier as black box and repeatedly probe its predictive performance under varying features on a validation set.

Embedded Methods in this class were not necessarily developed for performing explicit feature selection but they have been associated with this domain retrospectively due to their side effect being feature selection. These methods refer to technicalities within the learning algorithm and as such are classifier-specific. They consider the entire set of features as input and derive weights for features based on the influence they have on the value of the objective function. Therefore, the objective function is usually composed of (at least) two competing terms: the goodness-of-fit to be maximized and the number of variables to be minimized. The latter term can be realized with a ℓ_p-norm regularization term on the weights associated with features. In the limit as $p \to 0$, the ℓ_p-norm is just the number of weights or equivalently the number of features involved. The regularization term can be understood as a penalty for using many features when maximizing the goodness-of-fit. Regularization (Ridge, LASSO) as a means for feature selection has been exploited in a variety of forms within classic machine learning methods, such as SVMs [GWBV02, WEST03, NSS05] and neural networks [Hin86, RH97, GL98], where the technique is also known as weight-decay.

In contrast to these regularization based selection schemes, decision tree algorithms (e.g. ID3, Q4.5, CART) select features explicitly during tree induction. At each stage during recursive partitioning, they evaluate all features in terms of their ability to discriminate among classes and select the one that maximizes the splitting criterion (e.g. Information gain, Gini Index). Since the induction algorithm can select the same feature at several stages, the number of times the feature got selected can, together with the average information gain incurred by that feature, serve as an indication of feature importance for a post-hoc analysis [Bre01, SBZH06, GPTM10].

Filter methods Filter methods score and rank features in a pre-processing step before fitting a classifier on the selected set of relevant features. These methods use general characteristics of a training set to select some features and exclude others. They can be used independently of the choice of classifier but they rely on a suitable scoring function to rank the variables and a threshold to remove variables below a certain threshold. Most filtering methods use scoring functions that asses the relevance of each feature in isolation, in terms of its dependency with the target variable by using measures of relatedness such as the Pearson correlation coefficient, coefficient of determination, Fisher's criterion or mutual information.

The Pearson correlation coefficient assumes a continuous target variable and it measures the linear dependency between a variable and a target. Similarly, the coefficient of determination - the square of the covariance - can be used as variable ranking criterion to measure the goodness of linear fit of individual variables. These measures are also used in classification settings with a discrete binary target mapped to $\{-1, 1\}$ and they are related to Fisher's criterion that measures the ratio of the between-class variance to the within-class variance [GE03]. Correlation criteria can only reveal a feature's linear relation with the target.

Several feature selection approaches make use of the mutual information (MI) of a variable and the target [KC02, BEWT03, DMK03, MSB08]. The MI is an information theoretic ranking criterion that quantifies the dependence of two random variables in

terms of the Kullback Leibler divergence between the joint distribution over two variables and the product of their marginal distributions. It measures the loss of information incurred by modeling the feature-target pair as independent random variables when, in fact, they are not independent. Calculation of the MI requires access to the joint and marginal distributions which are unknown and especially difficult to estimate from data when one or both variables are continuous. However, non-parametric density estimation techniques such as the Parzen window method can be used [Par07]. Another filtering scheme is to select features according to their individual predictive power under a classifier that was built with one feature. In the most extreme case, a feature itself serves as discriminant function and it can be turned into a classifier by setting a threshold on the feature's range. Classical performance metrics such as accuracy, F1 or ROC can be used to score and rank the features, as reviewed by Forman et al. [For03], for selecting words in text classification. An extensive survey of different scoring functions and how they are used for ranking features within the filtering paradigm is given by Lazar et al. [SCB+12].

In order to lift the restriction of assessing a feature's relevance in isolation, several authors have developed holistic filter-based feature selection algorithms, with built-in feature scoring functions and search strategies. Kira et al. [KR92] proposed the RELIEF algorithm, that determines the relevance of a feature based on feature value differences between nearest neighbor instance pairs. A large feature value difference between two nearby instances from the same class decreases the feature's relevance score, and a large difference between nearby instances from different classes increases the score. RELIEF and the many variants that have been proposed since then rely on a distance measure for determining the closeness of instances in feature space. Almuallin et al. [AD90] proposed FOCUS, an algorithm that looks for minimal combinations of features that perfectly separate instances of different classes. It begins by scoring one feature in isolation, then evaluates pairs, triplets and so on, until it finds a combination that induces a pure partition of the training set. Although it does not consider combinations of features for assessing relevance, Fast Correlation Based Filter (FCBF) [YL03] uses symmetrical uncertainty, a information theoretic measure that draws from the concepts of Shannon entropy and information gain, to select a set of relevant features while simultaneously minimizing the number of redundant features in the set. A similar line of reasoning applies to the method of minimum redundancy maximum relevance (mRMR) [DP03], where the authors measure a feature's marginal contribution to a set of features by means of the ratio of MI between a feature and the target and the average MI of all feature pairs in the set. A survey about filter methods in general is given in [MN16a] and about mutual information in particular in [VE14].

Wrapper methods A third family of feature selection methods was coined *wrapper* methods [JKP14]. They evaluate a feature subset by running a learning algorithm on the training data and using the accuracy of the resulting classifier as their metric. Wrapped inside a feature subset search, the learning algorithm gets called as a subroutine on each feature subset. The general argument for wrapper methods is that the learning algorithm that will later use the selected features should provide a better estimate of

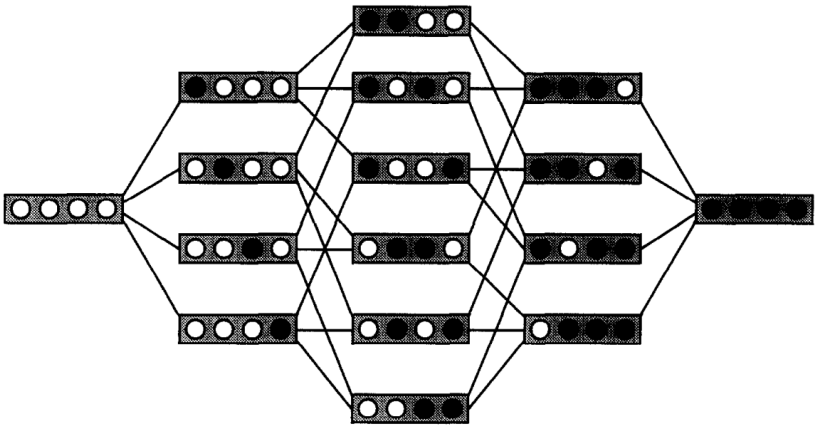


Figure 6.2: The elements of a space of feature subsets organized as states in a lattice. Each state specifies the selected features (black circles) from the four possible features. Connected states differ in exactly one selected feature. The figure was taken from [BL02].

the set's usefulness for the classification task than a separate measure with potentially different characteristics. Wrapper methods have been around in pattern recognition for quite some time but their major disadvantage is always the computational cost of running the learning algorithm as an inner loop. Therefore, researchers have developed algorithm-specific techniques to speed up the evaluation by using caching strategies, e.g. for decision trees [CF14], or by simply reducing the training set [ML94]. Similar to filter methods, wrapper methods are sometimes tightly coupled to the actual search strategy such as in the nearest neighbor-based OBLIVION algorithm [LS94].

The flexibility of wrapper approaches being used in conjunction with any learning algorithm, without requiring adjustment of the algorithm itself, is appealing as long as the computational overhead is acceptable. And as Stracuzzi et al. [SU04] state: "Wrapper methods do tend to outperform filter methods. This is not surprising given that wrappers evaluate variables in the context of the learning problem, rather than independently." A survey of different wrapper methods is given in [JKP14, PS15].

6.2.2 Feature selection as search

For a set of features X, there are $2^{|X|}$ different feature subsets we would need to evaluate to find the globally optimal subset of most relevant features. In many practical applications (e.g. classification of text documents or DNA microarray gene expression data), having to deal with tens of thousands of features is not uncommon although one can expect that only very few of the features are actually relevant to the task at hand. In order to be able to at least search parts of the space, researchers have developed several heuristics for navigating and probing the space of feature subsets. On the basis of Blum [BL02], any feature selection method can be characterized in terms of its stance on four basic issues that determine the nature of the heuristic search process:

- **Start** One must choose a starting point (i.e. an initial state) in the search space. The starting point influences the direction of the search and the operator used to generate successor states. As Fig. 6.2 illustrates, there is a natural partial ordering

on the space of feature subsets. Starting from left to right (right to left) each child state has exactly one more (less) feature than its parent. This observation already suggests an incremental procedure for traversing the state space. Either we start with an empty set of features and incrementally add one feature or we start with the entire set and remove one feature in each step. The former traversal strategy is called forward selection and the latter one is called backward elimination.

- **Step** The second issue deals with the organization of the search. Since an exhaustive traversal of all states is infeasible, one requires an operator that generates neighboring states (i.e. induces a certain connectivity on the lattice). A popular strategy is to search greedily over successive states: In each iteration, one considers the local neighbors of the current state, selects one of them and then iterates. Variants of greedy strategies differ in the way how they select a state from the neighborhood. The hill-climbing strategy, known as sequential selection, selects the best state among neighbors with one additional and one fewer feature, which enables backtracking without having to explicitly keep track of the search path. But one can also simply choose the first state from the neighborhood that improves accuracy over the current state. Apart from the greedy schemes, one can also draw on more sophisticated search alternatives, such as best-first search, to incorporate the expected gain in following a path beyond the direct neighbor. The big disadvantage of local search methods is the so called "nesting-effect" [Yus09], which refers to the fact that these methods usually can not backtrack to arbitrary previous states and therefore will return only a locally optimal solution. In an attempt to avoid bad choices of locally selected feature subsets, researchers have also explored distributed operators such as genetic algorithms [YH98] or particle swarm optimization [XZB13].

- **Score** The third element of feature subset selection is the choice of a suitable metric to evaluate alternative subsets of features. As discussed in the previous section, these metrics assess a feature's relevance with respect to a training or validation set. Many scoring functions are based on statistical correlation or principles of information theory but others directly measure a classifier's accuracy on a validation set after training it with the selected feature subset.

- **Stop** The last building block of a feature selection process consists in the choice of a criterion for halting the search. For instance, one might stop when none of the neighboring states improves the estimate of the classification accuracy; or continue as long as the accuracy does not degrade below a certain threshold. Or one can simply stop when none of the alternatives improves the estimate of the classification accuracy by more than a pre-define system parameter.

6.3 Selecting feature aggregates

In this section, we turn towards selecting a set of feature aggregates that is most useful for fraud detection. We make use of the feature selection principles and methods proposed

in literature but we also have to take into consideration the particularities of our task and thus apply reasonable care during the assembly of the feature selection process.

The first critical property of aggregates is that they exhibit minor relevance themselves but predominantly in combination with the constraint attributes they are defined on. Knowing that someone issued three transactions at the same merchant type within a day is only informative for predicting the class if we additionally know which merchant type that is. For restaurants or bars, such activity seems to be within a plausible range but for an ATM it might raise suspicion. Also the class label might depend on the aggregate and several other attributes in a complex way. In order to assess the usefulness of such a feature aggregate, we can not consider it in isolation but we have to quantify the feature's marginal contribution within a coalition that also contains the aggregate's constraint attributes. This problem severely limits our possibilities of using simple filter methods for scoring the features. More advanced techniques such as FOCUS or RELIEF rely on some kind of distance measure between instances, but defining a reasonable distance on the many categorical attributes is an open question in its own right.

The second critical point is the fact that, in principle, we could instantiate an infinite number of different feature aggregates. Although we have to constrain it in practice, even with only a few targets and constraints the space quickly becomes massive. And this is only the set of n different aggregates. In the end, we want to select the best k-element subset $A*$ among all $\binom{n}{k}$ possible k-element subsets $\mathcal{A}$. And since it is impractical to instantiate the entire set, we can not benefit from embedded methods directly. Due to its computational cost, the wrapper approach is not an option. Fitting and evaluating tens of thousands of random forests on millions of instances surmounts the computational resources we have available.

Let X denote the set of raw features, e.g. payment mode, amount, merchant code, and so forth, then, based on the above considerations, we define the following two exploration strategies:

- **Random:** Repeatedly, we draw a set A from $\mathcal{A}$ randomly and use the test score of a random forest classifier trained on the feature set $X \cup A$ as the score of the subset A. This naive strategy shall serve as baseline. It falls into the class of wrapper approaches with the search heuristic consisting in randomly sampling from all possible k-element sets.

- **Greedy sequential forward selection using Gini importance as relevance score:** Starting with an empty feature set $F = \{\}$, we incrementally add one additional aggregate a^* in each iteration until we reach the maximum number k of aggregates. Which aggregate we add, is determined by a relevance score. In each iteration, we instantiate a set of randomly chosen candidate aggregates $\{a_c\}_{1 \leq c \leq C}$, assign a relevance score to each candidate and add the aggregate that yielded the maximum relevance score, a^*, to the current set, i.e. $F = F \cup \{a^*\}$. After k iterations, we report F as the selected subset.

  As discussed above, obtaining such a relevance score for an aggregate is tricky as the aggregate is conditioned on the values of the constraint attributes. There-

fore, we create a CART decision tree on each feature set $X \cup \{a_c\}$ and quantify a_c's relevance in the presence of all other features X by the Gini importance, which is a measure of a variable's importance derived during tree induction (see Eq. (3.4)). This statistic is often used to inspect and compare the influence of features on the response variable (e.g. association of genotype expressions with diseases [BDF+05]). We prefer to use the Gini importance because it allows us to assess the relevance of a variable when used in conjunction with others and it is comparably cheap to compute during tree induction as it does not require an additional evaluation cycle on some validation set. However, as Strobl [Str08] states, the Gini importance as a criterion for feature selection can be misleading because of its bias towards variables with many different values, i.e. continuous variables with few ties or large categorical variables. Sub-optimal but high-cardinality variables may be artificially preferred as split variables during tree induction, just because, with more potential cutpoints, the chance that at least one of them maximizes the split criterion increases. Since the Gini importance of a variable is calculated as the mean improvement in Gini Index (the Gini gain) produced by that variable, the selection bias in the CART method is reflected as bias in the Gini importance.

After extraction, all aggregates are continuous variables but their scale of measurement may vary, depending on the type of aggregate and the data. Therefore, the relevance score of an aggregate may dominate others simply because of a higher cardinality-dependent bias and not because of the aggregate being a more informative feature. To compensate this bias, we borrow the idea of *permutation importance* [CCS12] and adapt it to our needs. We calculate the permutation importance of a variable by first recording the regular Gini importance derived from a decision tree. Next, we randomly permute the values of the variable, induce a new decision tree and record again the Gini importance of the variable under random permutation of its values. The difference between the Gini importance of the correctly ordered variable and the Gini importance of its randomly permuted version is used as a measure of feature relevance.

6.3.1 Experimental setup

In a small experiment, we create subsets consisting of 1 up to 15 aggregates with both the random selection and the greedy forward selection approach. The subsets of aggregates are sampled from one of three aggregation spaces: Aggregates calculated on sequences grouped by the card holders IDs (user-centric), aggregates calculated on sequences grouped by the terminal IDs (merchant-centric) and aggregates from either of the two aggregation spaces (user-merchant-centric).

In accordance with the concept defined in Section 6.1.1, an aggregation space is spanned by a set of targets $\mathcal{T}$ and a set of contexts, i.e. the power set $\mathcal{P}(\mathcal{C})$ of constraints $\mathcal{C}$. We create one aggregate by drawing one element randomly from both $\mathcal{T}$ and $\mathcal{P}(\mathcal{C})$. The user-centric and the merchant-centric aggregation spaces are defined on the same set of targets $\mathcal{T}$, as listed in Table 6.2. The set of user-centric constraints $\mathcal{C}_u$ and merchant-centric constraints $\mathcal{C}_m$ are listed in Table 6.3. The choice of targets and constraints follows

$\mathcal{T}$ ($\lvert\mathcal{T}\rvert = 12$)
{
(sum, `amount`, [-1000, 0]),
(mean, `amount`, [-1000, 0]),
(count, `amount`, [-1000, 0]),
(min, `amount`, [-1000, 0]),
(max, `amount`, [-1000, 0]),
(range, `amount`, [-1000, 0]),
(mean_range, `amount`, [-1000, 0]),
(count, `timestamp`, [-1000, 0]),
(range, `timestamp`, [-1000, 0]),
(mean_range, `timestamp`, [-1000, 0])
(count, `timestamp`, [-1, 0]),
(range, `timestamp`, [-1, 0]),
}

Table 6.2: The set of targets considered during feature selection.

common sense but, apart from that, it is arbitrary. Of course, we could have chosen more, less or different constraints. We proceed with these sets because they already cover a large spectrum of aggregates, contain the manually engineered aggregates we had used in Section 4.1 and they lead to aggregation spaces that can still be explored within reasonable time.

In case of the greedy forward selection approach, we generate 1000 candidate aggregates in each round, extract their values, evaluate the relevance of each candidate by means of the Gini importance, add the best candidate to the set of selected aggregates and repeat until the set of selected features contains 15 selected aggregates. In each round, we evaluate the predictive performance of the aggregates we have selected so far with a random forest classifier. The data for feature selection with a decision tree and for training the random forest classifier comes from the time period 01.03.2015 - 30.04.2015 and the score of the classifier is reported on data from 08.05.2015-31.05.2019. Due to the high computational cost, we fix the hyper-parameters of both the decision trees created during feature selection and the random forest classifier used for evaluating the selected features (see Appendix A.2 for the list of hyper-parameters). As a consequence, the performance estimates will necessarily be sub-optimal because neither the decision tree nor the random forest model is adequately tuned to the specific set of features it is learned on.

In case of the random selection approach, we simply generated increasingly large sets of aggregates randomly without employing any sophisticated selection strategy. We evaluate the predictive performance of each set in the same way as described above.

We select subsets of sizes ranging from 1 to 15 aggregates with each of the two methods. The greedy selection is repeated 5 times (75 selection/evaluation cycles) and the random selection is repeated 15 times for each subset size (225 selection/evaluation cycles). The set of raw features X is the `BASE` feature set known from previous experi-

User-centric constraints $\mathcal{C}_u$		Merchant-centric constraints $\mathcal{C}_m$	
variable	value ranges	variable	value ranges
`timestamp`	[-1, 0], [-24, 0], [-168, 0], [-672, 0]	`timestamp`	[-1, 0], [-24, 0], [-168, 0], [-672, 0]
`hourOfDay`	[0, 0], [-1, 1]	`hourOfDay`	[0, 0], [-1, 1]
`weekday`	[0, 0]	`weekday`	[0, 0]
`amount`	[0, 0], [-1, 1], [-10, 10], [-100, 100]	`amount`	[0, 0], [-1, 1], [-10, 10], [-100, 100]
`cardEntryMode`	[0, 0]	`cardEntryMode`	[0, 0]
`3DSecure`	[0, 0]	`3DSecure`	[0, 0]
`emv`	[0, 0]	`emv`	[0, 0]
`authentication`	[0, 0]	`authentication`	[0, 0]
`terminalID`	[0, 0]	-	-
`terminalCategory`	[0, 0]	-	-
`terminalCountry`	[0, 0]	-	-
-	-	`userID`	[0, 0]
-	-	`userCountry`	[0, 0]
-	-	`gender`	[0, 0]
-	-	`expiryDate`	[0, 0]
-	-	`cardType`	[0, 0]
-	-	`age`	[0, 0], [-2, 2], [-10, 10]
-	-	`creditLimit`	[0, 0], [-100, 100], [-500, 500]

Table 6.3: The set of user-centric constraints $\mathcal{C}_u$ and merchant-centric constraints $\mathcal{C}_m$. The variable `timestamp` is measured in hours with corresponding value ranges covering one hour, one day, one week and four weeks. The only possible value range for categorical variables is [0, 0], to check for identity.

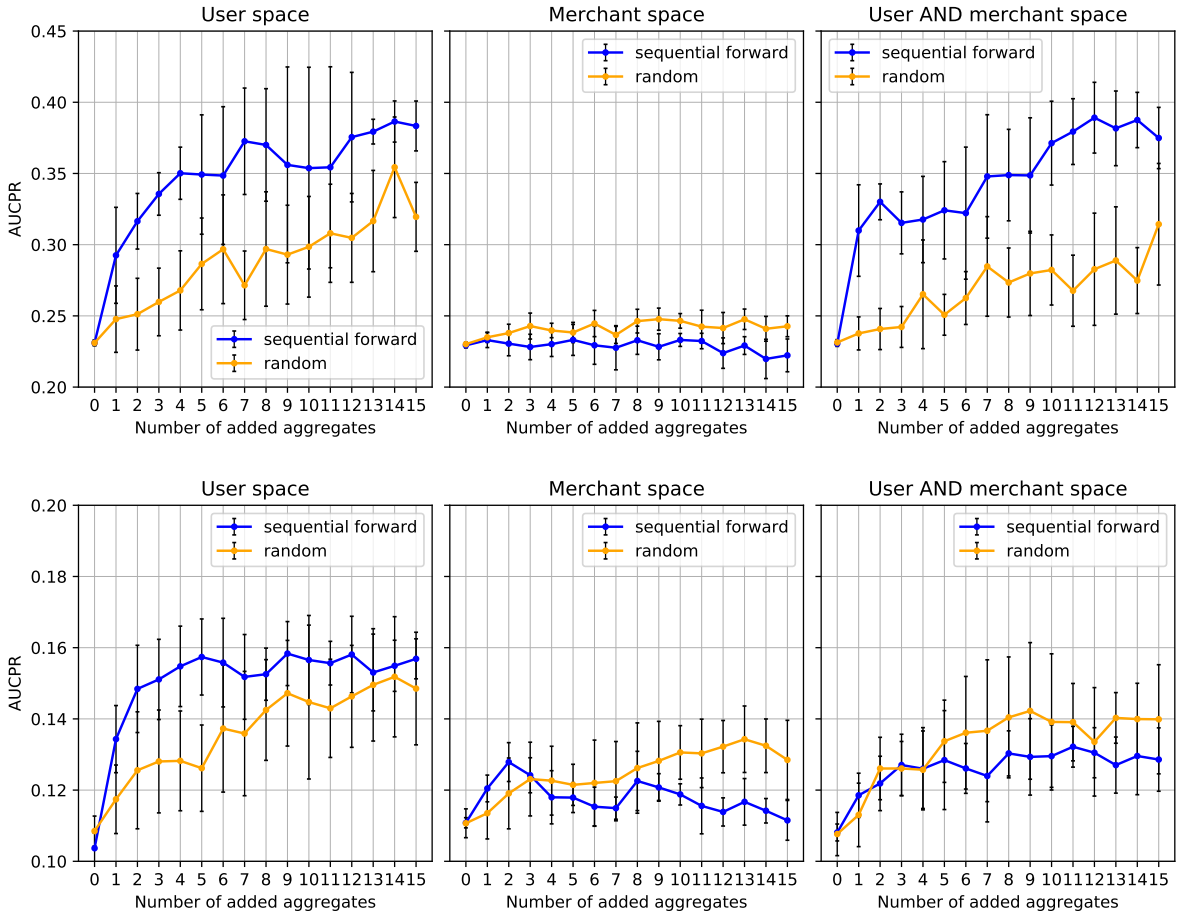


Figure 6.3: Predictive performance of subsets of aggregates for e-commerce (top) and face-to-face (bottom). Subsets were created either with the Gini importance-based greedy forward selection method (blue) or the random selection method (orange). The three columns show the mean AUCPR (± one standard deviation) on the test set when the selection method is allowed to choose aggregates from the user space, the merchant space or from both the user space and the merchant space.

ments. We carry out the greedy selection and the evaluation of the selected features on the union of X and the aggregates.

6.3.2 Results

As usual, we report performance individually for e-commerce transactions and face-to-face transactions. As a reminder, the baseline prediction performance in terms of AUCPR when using the 14 manually defined user-space aggregates from Section 4.1 was 0.398 on e-commerce and 0.203 on face-to-face. The plots in Fig. 6.3 reveal several findings along two dimensions: The aggregation spaces and the subset selection methods.

We can observe that there is hardly any performance gain from merchant-space aggregates in both the e-commerce and the face-to-face scenario. Even with the addition of several aggregates, the AUCPR stays in a comparable regime as it is when using no aggregates at all (indicated by the measurement at location 0) and the ran-

dom selection method appears to come up with slightly better aggregates on average than the greedy selection. In contrast, the user-space aggregates exhibit much greater discriminative power in both scenarios. The predictive performance clearly increases as we add more user-space aggregates. Thereby, the greedy selection method reaches higher AUCPR scores with fewer aggregates than the random selection method, until the curves finally flatten out at an AUCPR between 0.35 to 0.40 (e-commerce) and 0.15 to 0.16 (face-to-face). Given that we did not perform any hyper-parameter tuning, these performances are surprisingly close to the performance we obtained with the manually defined aggregates from literature.

When we allow the methods to pick aggregates from both the user-space and the merchant-space, the results are fundamentally different between e-commerce and face-to-face. In the e-commerce scenario (top, right), the performance curves are comparable to the user-space performance curves (top, left) both in terms of the general trend and the relative difference between the greedy selection method and the random selection method. This behavior is expected considering the poor discriminative power of merchant-space aggregates. After inspection, we found that the subsets selected with the greedy method consist almost entirely of user-space aggregates (98% on average) whereas the random subsets contain the expected ratio of 51% of user-space aggregates. However, in the face-to-face scenario, the greedy selection performance (bottom, right) is unexpectedly below the user-space performance (bottom, left). The subsets selected with the greedy method contain the same ratio of 51% user-space aggregates as does the random selection method. Therefore, we suppose that the much higher class imbalance present in face-to-face not only renders classification more difficult but also complicates a reliable assessment of a feature's relevance.

In the e-commerce scenario which exhibits moderate class imbalance, our custom greedy forward selection method based on the Gini importance seems to work well for choosing aggregates. The aggregate added in the first iteration yields the highest improvement compared to all subsequently added aggregates. We wondered whether these highly discriminative aggregates are somewhat similar to the ones we have used in Section 4.1 and whether they are structurally consistent across several runs regarding their aggregation operators, target variables or their set of constraints. Tables 6.4 and A.1 show the aggregates that have been added in the first iteration of greedy forward selection for all five runs in the e-commerce and the face-to-face scenario, respectively. The aggregates found on face-to-face seem to be arbitrary and we can therefore not identify any consistent pattern among them (Table A.1 can be found in the appendix).

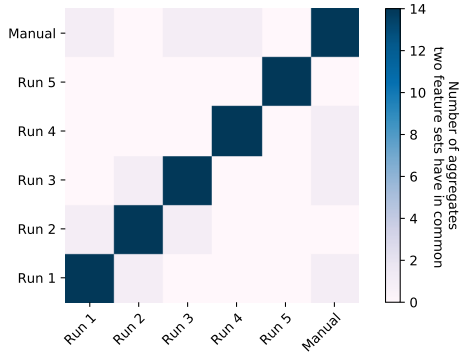
However, on the more meaningful e-commerce dataset (Table 6.4), we notice two consistently recurring patterns over the five runs: (i) Time is always part of the context. This points out the importance of capturing some notion of recency and/or ensuring normalization of the aggregate's extracted values. If there would not be any time constraint, the values of the aggregate would not be comparable over different transactions because the admissible index range by far exceeds the number of transactions in a user's or merchant's history. (ii) The `count` operator is used in the majority of aggregates. When using the count, the values of the target variable are ignored entirely[8], since the sole

[8]The presence of a target variable is a mere artifact of the implementation and it should be ignored.

Exp.	Space	Run	Target Variable	Operator	Index range	Context Variable	Value range
U	U	1	~~amount~~	count	$[-1000, 0]$	timestamp	$[-24, 0]$
						terminalCategory	$[0, 0]$
	U	2	~~amount~~	count	$[-1000, 0]$	timestamp	$[-168, 0]$
						emv	$[0, 0]$
						weekday	$[0, 0]$
						hourOfDay	$[-1, 1]$
	U	3	~~amount~~	count	$[-1000, 0]$	timestamp	$[-24, 0]$
						terminalCountry	$[0, 0]$
						cardEntryMode	$[0, 0]$
	U	4	~~amount~~	count	$[-1000, 0]$	timestamp	$[-24, 0]$
						emv	$[0, 0]$
						authentication	$[0, 0]$
	U	5	~~amount~~	count	$[-1000, 0]$	timestamp	$[-1, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						emv	$[0, 0]$
U/M	U	1	timestamp	span	$[-1000, 0]$	timestamp	$[-24, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						emv	$[0, 0]$
						authentication	$[0, 0]$
						weekday	$[0, 0]$
						hourOfDay	$[-1, 1]$
	U	2	~~amount~~	count	$[-1000, 0]$	timestamp	$[-1, 0]$
						cardEntryMode	$[0, 0]$
						emv	$[0, 0]$
	U	3	timestamp	mean_span	$[-1000, 0]$	timestamp	$[-1, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						weekday	$[0, 0]$
	U	4	amount	span	$[-1000, 0]$	timestamp	$[-1, 0]$
	U	5	timestamp	mean_span	$[-1000, 0]$	timestamp	$[-1, 0]$
						hourOfDay	$[-1, 1]$

Table 6.4: E-commerce: Aggregate selected in the first iteration of the greedy forward selection method from the user-space (U) and the joint user/merchant-space (U/M).

(a) ECOM - user-space



(b) ECOM - user/merchant-space

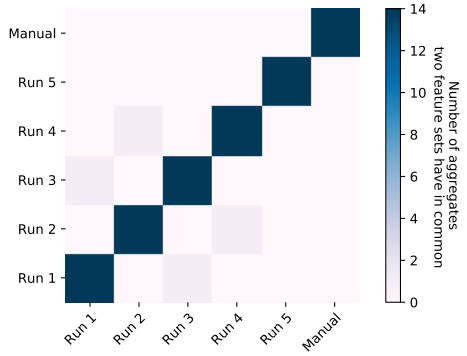


Figure 6.4: Comparison between the set of 14 manually defined aggregates from Section 4.1 and sets of 14 automatically selected aggregates, in terms of the number of features they have in common. The set of manually defined aggregates is denoted by "Manual" and the five repeated runs of greedy forward selection are denoted by "Run 1" to "Run 5". While the predictive performances are comparable, any pair of feature sets shares at most one identical aggregate - all other aggregates are different.

number of matching transactions serves as the aggregate's value. If not the count, either the `span` or `mean_span` operator is used. These operators calculate the delta between the maximum and the minimum value of the target variable over the matching historic transactions. And if applied to the timestamp, they capture different notions of time delta - similar to the time delta feature that we have defined manually in Section 4.1.

While aggregates might differ in their constraints, manual inspection revealed that the presence of time as constraint and the usage of count as operator appears to be consistent across most of the selected aggregates. Due to the broad variety of different aggregates, it is tedious to list or analyze all selected features, however we provide one further comparison between the 14 manually defined aggregates from Section 4.1 and the same-sized sets of automatically selected aggregates. Again, we focus on e-commerce here but the same analysis for face-to-face can be found in the appendix (Fig. A.2). While the predictive performance of the automatically selected sets of 14 aggregates is comparable to the performance of the 14 manually defined ones (around 0.40 AUCPR), the feature sets are mostly disjoint, as illustrated in Fig. 6.4. Not only do the manual aggregates differ from the selected ones but also the selected ones vary over repeated runs.

6.4 Summary

In this chapter, we studied feature aggregates as they have been proposed in literature. We provide a canonical concept around their definition to leverage feature selection from the aggregation space by means of a fast feature extraction algorithm and the Gini importance as relevance measure. The results showed that greedy sequential forward selection with Gini importance is able to select highly discriminative aggregates from the user-space, as compared to a random selection baseline. According to the results,

the merchant-space aggregates are less discriminative than the user-space aggregates.

Regarding our first research question **RQ1**, we could not find a set of aggregates that would clearly improve over the manually engineered ones in terms of AUCPR on the test set. On face-to-face transactions, the found automatically selected aggregates yield consistently lower performance than the manual defined ones. On the e-commerce dataset, we could discover several feature sets which are comparable in performance to the manually defined set (around 0.40 AUCPR). We suppose there are at least two reasons for these results. Firstly, due to the high computational cost we could not perform proper model selection for each choice of feature set but instead we resorted to one configuration of hyper-parameters that was fixed across both the e-commerce and the face-to-face scenario. And secondly, it is possible that we are reaching an upper limit on the distinguishability between fraudulent and non-fraudulent transactions these aggregates, as we have defined them, are able to provide.

Our answer to research question **RQ2** points in a similar direction but from a different angle. We could discover several sets of aggregates with good performance and yet are these sets predominantly disjoint. We conclude that the manually engineered features are only one specific example from a large family of comparably good aggregates. Manual inspection revealed that it is less the exact choice of aggregation operator or the set of constraints but rather any notion of a count per time interval on a user level which turns aggregates into valuable features for credit card fraud detection. We call this relation the *user's event density in time*, in which the *event* can be defined as any combination of raw attribute values.

Chapter 7

Outlook

Before we conclude this thesis, we would like to raise awareness about some concerns that should accompany the development of any data-driven product, especially in domains where these products are allowed to take far-reaching decisions that may, in one way or the other, impact the everyday life of humans. With the broad availability of machine learning libraries and frameworks which mask most technicalities of learning algorithms behind easy-to-use APIs, a tremendous number of people have stepped into this field in recent years and they are eager to use and deploy such algorithms in reality. The past hype around deep learning and the current interest in automated machine learning are results of a lively belief in the possibility to automa*g*ically discover the true mechanics of an underlying process from the data it produces. Of course, given infinite data and a set of functions large enough to contain the true mechanics as one of its elements, it is possible to discover this element. However, if the data is finite and the set of functions is not equipped with an interpretation in terms of real-world quantities, there is a risk of discovering a seemingly correct mechanic which turns out to be wildly wrong as more data becomes available. And as the set of functions did not arise from human-interpretable or carefully chosen modeling assumptions, there is no principled strategy to revise it accordingly. Instead, we resort to hyper-parameter search for revising the set of functions in the hope to find a better solution in the next evaluation cycle. While the algorithm may then eventually learn something useful, the data engineers / data scientists / machine learning engineers do not. By mimicking hypothesis creation and falsification with progressively resource-demanding function search algorithms, we deprive ourselves from the possibility to learn about the problem and risk to deploy a solution which we can neither explain nor provide guarantees for.

Seeing black-box solutions being proposed for recruiting, social credit scoring, fraud detection, autonomous driving, automated grading or medical diagnosis is unsettling. At the time of writing, most of these applications are experimental and the decisions made by these systems are reviewed by human experts before they take effect. And we should not attempt to change this status quo until we have reliable means for assessing not the capabilities but the limitations of such systems.

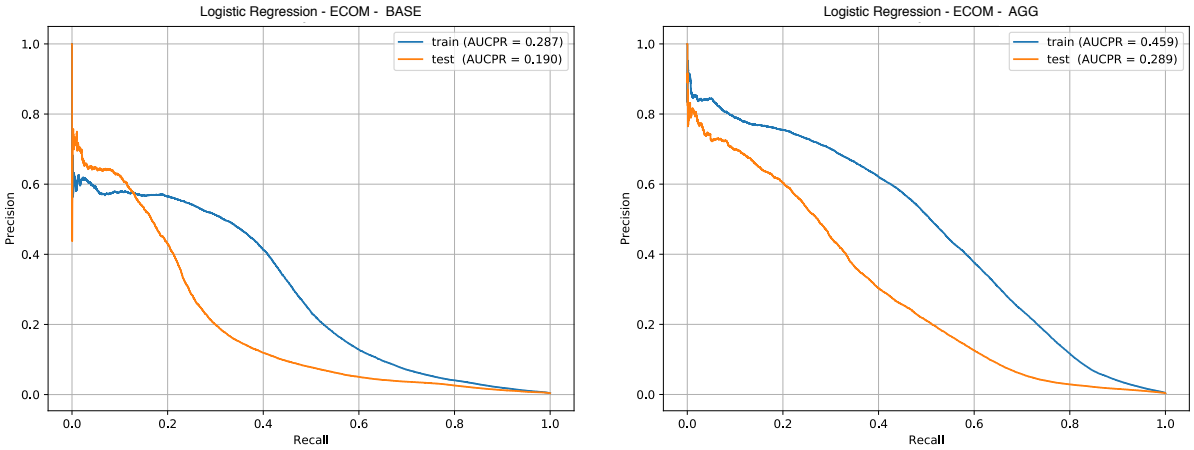


Figure 7.1: Precision recall curves of a Logistic Regression classifier using the `BASE` feature set (left) and the `AGG` feature set (right). Training period March-April and testing period 8.05.-31.05. on the e-commerce data set.

7.1 Future work

For future work, we can imagine simpler, more transparent and more efficient methods that provide secondary benefits for practical applications beyond the raw prediction accuracy. Or, to quote Phua et al. [PLSG10] from their survey: "There is too much emphasis by research on complex, nonlinear supervised algorithms such as neural networks and support vector machines. In the long term, less complex and faster algorithms such as naive Bayes [...] and logistic regression [...] will produce equal, if not better results [...], on population-drifting, concept-drifting, adversarial-ridden data. If the incoming data stream has to be processed immediately in an event-driven system or labels are not readily available, then semi-supervised and unsupervised approaches are the only data mining options." In fact, the performance we obtained with a simple Logistic Regression classifier fitted to likelihood-ratio encoded variables is surprisingly close to the performance of a random forest that averages over 300 decision trees (see Fig. 7.1). Considering that learning requires only one single pass through the data set to compute the likelihood-ratios and a subsequent estimation of one parameter per variable (plus one intercept term), an AUCPR of 0.190 is not too bad compared to the 23.2 we obtained with a random forest classifier. Inference is even more efficient: To get the fraud score of a transaction, we have to sum the likelihood-ratios of all variables weighted by the coefficients of the Logistic Regression classifier - a sum over 19 numbers in our example. When using parametric distributions to model the variables, the parameters possess clear semantics allowing us to inspect the model (see Fig. 7.2 for examples). Of course, if we wanted to integrate interactions of variables, we would have to do so explicitly by modeling joint distributions over subsets of variables.

Obviously, the Logistic Regression classifier can not benefit as much from user-specific features (see AUCPR in Fig. 7.1 (right) compared to an AUCPR of 0.40 of a random forest classifier) as a complex classifier that can exploit feature combinations. When fitting a single normal distribution to the transaction amounts across all users, we essentially average out all peculiarities of different users. This is a gross over-

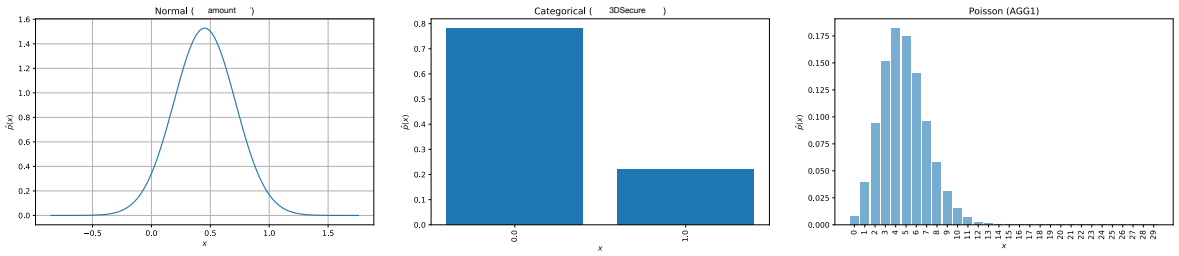


Figure 7.2: Fits of different parametric distributions: Normal(μ, σ^2) for the purchase amount (left), Categorical$(\pi_1, \ldots, \pi_k)$ for the binary variable 3D-Secure (middle) and Poisson(λ) for the number of transactions issued by the card holder in the last 24 hours (right). Hence, modeling these three variables requires 4 parameters.

generalization of the diverse behavioral traits we observe across millions of different users. Likewise, Juszczak et al. [JAH+08] point out that, regarding the distributions over variables, there is a large extent of heterogeneity within and between accounts. By using complex non-linear classifiers, we did not need to account for this heterogeneity because the tree induction algorithm would eventually discover rules that associate amounts, merchants and quasi-identifiers of users. However, as we strive for simplicity and transparency, our goal should be to learn a small model for every single user while still retaining the possibility to "inherit" knowledge from similar users. Very recently, Carcillo et al. [CBC+19] explored several outlier detection methods operating on a global, account and cluster level for credit card fraud detection. Even though they monitored only the transaction amount, their conclusion is that outlier detection on some intermediate level of granularity somewhere between the account level and the global level seems most promising.

We would like to augment this perspective with our own ideas and considerations to maybe inspire interesting future research.

The greatest challenge in modeling a single user is the small number of transactions we have seen from him. In our three months of data, 50% of users have issued less than 8 transactions. Of course, in practice we have access to the entire history of all users but even if we consider several years, a single user's history probably amounts to no more than a few hundred transactions. Regarding the variables we aim to model, we have to consider that some of them exhibit a large number of levels. There are around 1000 different merchant category codes and around 200 different merchant countries. If we were to model these variables with a categorical distribution we had to estimate several hundreds of parameters from only a few hundred observations. And this is only for a single variable. If we wanted to model joint distributions over several variables, we have even less observations per parameter. This estimation problem is completely ill-posed because we will only get meaningful estimates for merchant codes that showed up several dozens of times - all others will effectively end up having zero probability mass. Of course, we can think of a prior for the merchant code variable, but that is tricky. How should we determine a prior for every single user? All we know about users is what we see in the data, but exploiting this knowledge is not exactly an argument for a particular choice of prior. Still, we can always assume a non-informative prior over

the merchant category codes and hope that modeling errors in rare merchant codes are not going to degrade the overall performance too much. However, it is risky to think that rare codes won't matter since, especially when we use the user model for anomaly detection, it is exactly the rare events that inform us about anomalous activities. And just because a user did not yet perform a certain action, it does not mean that he could not legitimately do that. It would be much more plausible to "derive" a user's distribution over such high-cardinality variables from the entirety of actions of all users or, even better, from similar user groups.

One possible approach could consist in embracing probabilistic graphical models and, in particular, techniques such as probabilistic latent semantic analysis (PLSA) [Hof01] or latent Dirichlet allocation (LDA) [BNJ01]. These techniques are often used in topic modeling and they make the assumption that any word in a text document is generated by one of K latent topics. Each topic is represented as a distribution over all words in the vocabulary. And each document is represented as a distribution over the K topics. PLSA and LDA estimate simultaneously the distribution over words and a document's assignment to topics by means of an expectation-maximization algorithm. In our case, documents correspond to users u and words correspond to, say, the merchant IDs x. Just as in PLSA, we can represent a user's distribution over merchant IDs with a K-component mixture distribution:

$$p(x|u) = \sum_{k=1}^{K} z_k(u) p(x|\theta_k); \qquad z_k(u) \in [0,1], \sum_k z_k(u) = 1 \tag{7.1}$$

The neat property is that now we have "global" user-independent component distributions $p(x|\theta_k)$ and "local" user-dependent mixture coefficients $z_k(u)$ that determine how to mix the component distributions. Instead of having to estimate a ridiculous number of $|range(u)| \cdot |\theta_\cdot|$ parameters, we only have $K \cdot |range(u)| + K \cdot |\theta_\cdot|$ parameters for the categorical variable x. Since the number of mixture components K is typically much smaller than $|\theta_\cdot|$, this makes quite a difference and could lead to much more meaningful fits even though any single user has only very few observations.

From a practitioners perspective, we could simply apply EM-Clustering on the account-level average of one-hot encoded merchant IDs, treat the K cluster centroids as θ_k's and average over the cluster assignments of a user's observations to obtain $z_k(u)$. This procedure can be extended to other types of observed variables while still yielding interpretable profiles of all users' activities. Figs. 7.3 and 7.4 illustrate the mixture coefficients $z_k(u)$ of 100'000 card holders and, as an example, the associated distribution $p(x|u)$ when x is the amount or the hour of day, respectively. For instance, the variable hour-of-day has 24 different values and if we wanted to model this variable for each user independently with a categorical distribution, we would need to estimate and store $100'000 \cdot 23$ parameters. With the decomposition in Eq. (7.1) of $p(x|u)$ into 3 mixture components, we only have $100'000 \cdot 3 + 3 * 23$ parameters to estimate and store.

Modeling $p(x|u)$ is only half of the solution. Finally, we wish to flag suspicious transactions and, at this point, we have two options. Either we do not distinguish compromised user accounts from genuine user accounts and proceed in an unsupervised fashion, flagging any observation as fraudulent whose likelihood under $p(x|u)$ falls below some

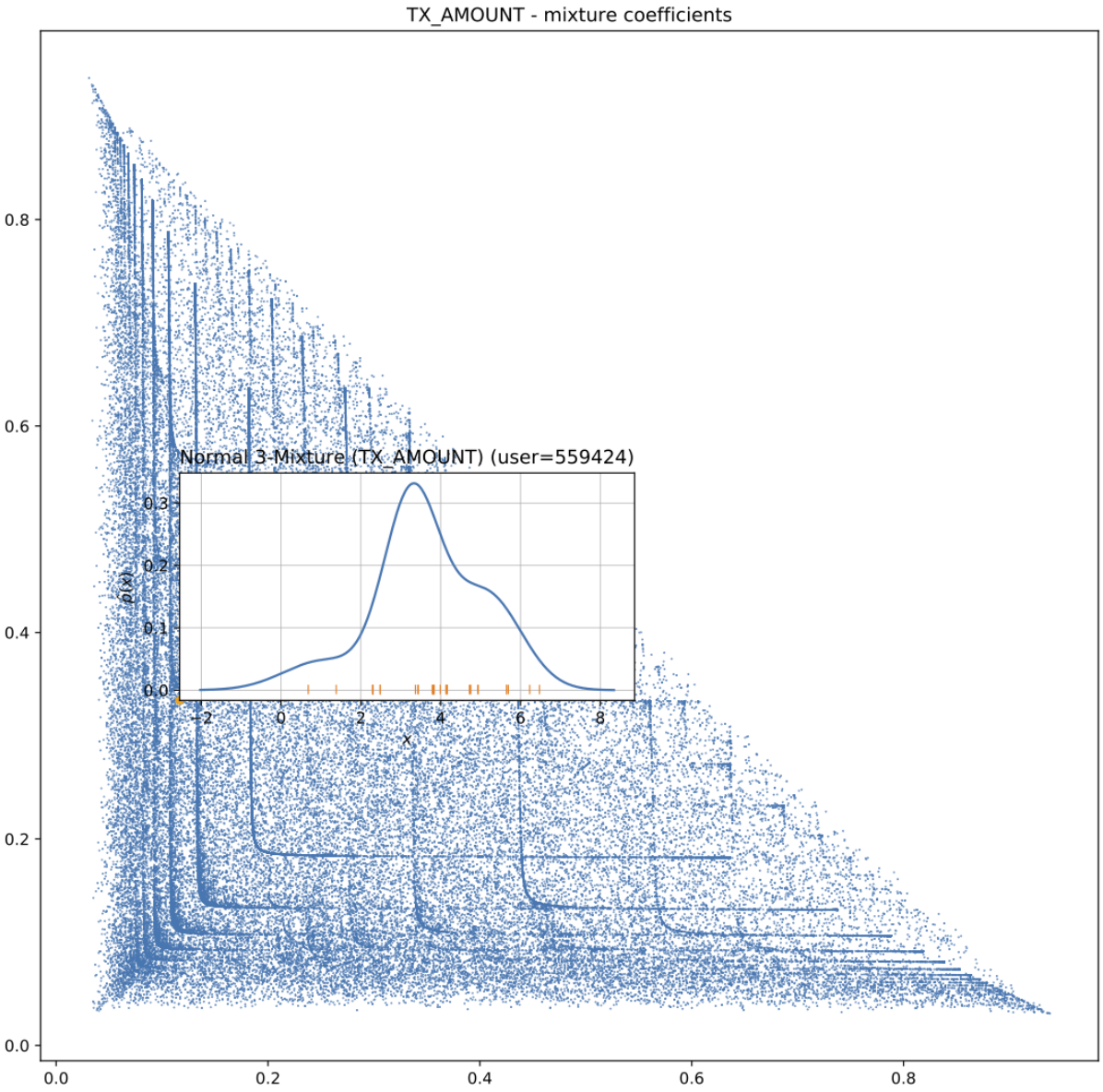


Figure 7.3: Mixture of three normal distributions where the mixture coefficients $z_1(u), z_2(u), z_3(u)$ are user dependent. The plot shows mixture coefficients $z_1(u)$ plotted against $z_2(u)$ for 100,000 users (each blue dot corresponds to one user). Each user's mixture coefficients parameterize a normal mixture distribution over the variable "amount" (location of lower left corner of inset axis parameterizes the displayed density). The little clusters correspond to users who end up having a similar density.

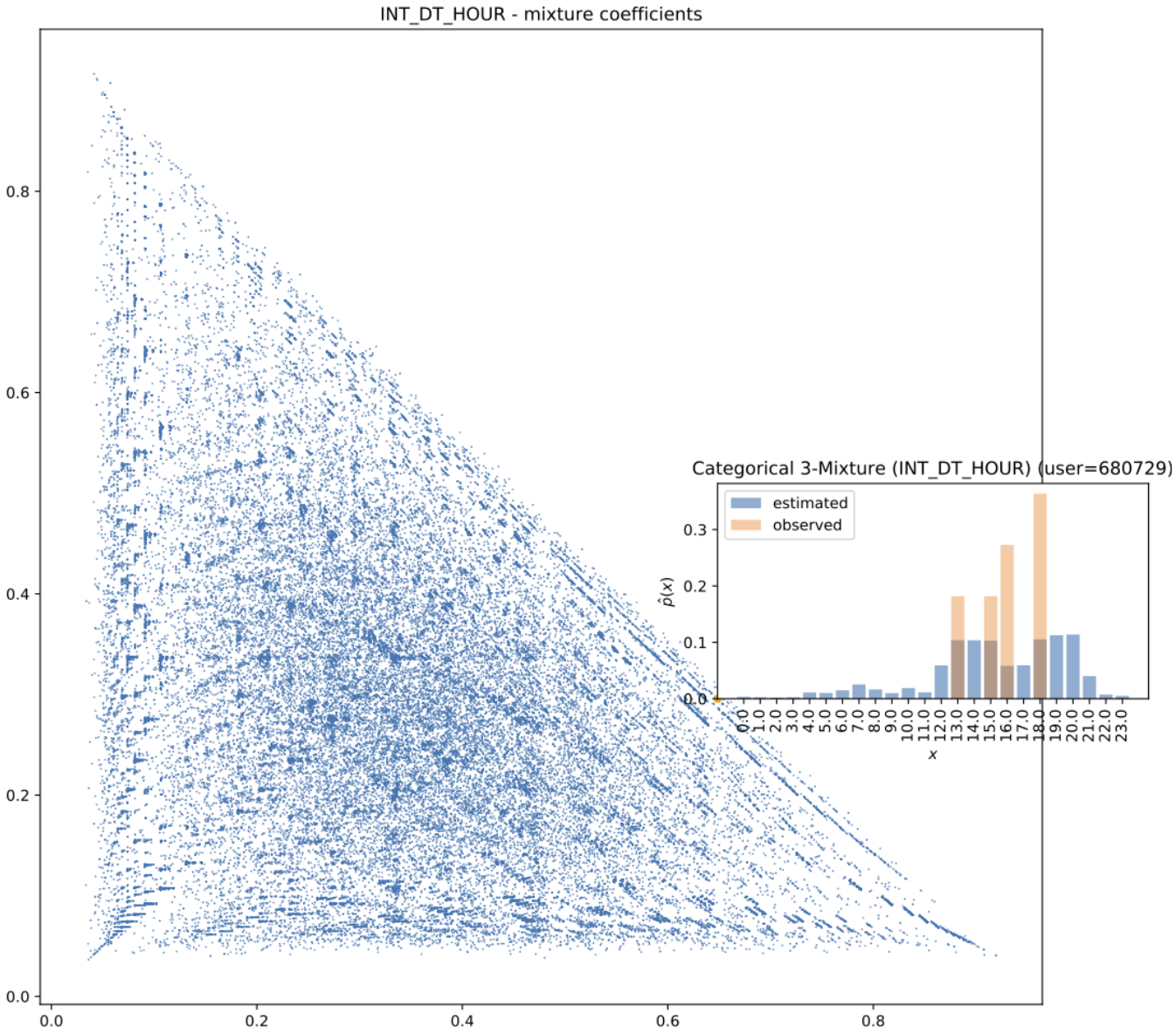


Figure 7.4: Mixture of three categorical distributions where the mixture coefficients $z_1(u), z_2(u), z_3(u)$ are user dependent. The plot shows mixture coefficients $z_1(u)$ plotted against $z_2(u)$ for 100,000 users (each blue dot corresponds to one user). Each user's mixture coefficients parameterize a categorical distribution over the variable "hour of the day" (location of lower left corner of inset axis parameterizes the displayed probability mass function).

user-specific threshold. Or we estimate $p(x|u_f)$ and $p(x|u_g)$ independently from compromised accounts and genuine accounts, respectively. However, it is not clear yet, how to contrast these two quantities in a meaningful way to derive a fraud score for a new observation. The user IDs covered by u_g refer to actual accounts and they are associated with genuine user behavior. On the other hand, the IDs covered by u_f are more of a virtual nature because they do not refer to actual accounts but are only associated with malicious behavior. Therefore, once we observe a new transaction with value $x = \mathtt{x}$ and user ID $u = \mathtt{u}$, the question we should really ask is, whether it's more likely that the transaction was generated by $\mathtt{u}$'s behavior $p(x = \mathtt{x}|u_g = \mathtt{u})$ than by any of the known malicious behaviors $\sum_{\mathtt{u_f}} p(x = \mathtt{x}|u_f = \mathtt{u_f})p(\mathtt{u_f})$. We also have to consider that, depending on the number of observations per user, the estimate of the distribution parameters contains different extents of error. For some users, the linear combination of global component distributions might come reasonably close to the user's true distribution whereas for others it might be quite off. Therefore, a user-specific decision threshold must reflect the level of uncertainty we have in the user's model, incorporating both the number of observations and the goodness of fit.

7.2 Conclusion

In this thesis, we studied several means of exploiting contextual information of transactions to improve data-driven credit card fraud detection. Our analysis is based on a large real-world credit card transaction data set provided by a leading company in this area. We introduced the problem of fraud detection and the peculiar challenges inherent to it: Class imbalance, feature engineering, sequence modeling, concept drift and performance evaluation. In regard of these challenges, we reviewed related work and provided an overview of the spectrum of solutions that have been proposed to address them.

To mitigate the limitations of working with a proprietary data set, first we exposed details about the volume, the available attributes and their distributions. We pointed out the different qualities of card-present and card-not-present transactions and established a fraud detection baseline by means of a random forest classifier. We analyzed the influence of varying class re-sampling ratios on the predictive performance and showed that the real-world task of forecasting is considerably more difficult than prediction.

Our first angle on the transaction context is based on external data. We augment transactions with country related and time related demographic statistics, covering economical and social indicators as well as public holiday indicators. Additionally, we explore word embedding techniques to extract country related features from text corpora and semantic knowledge graphs automatically. Our evaluations show that features from external sources improve the fraud detection performance most noticeably on card-not-present transactions when the features are combined with feature aggregates. In a second approach, we integrate a transaction's immediate past (account centered) on an algorithmic level with a Long Short-term Memory network. The results show that the LSTM leads to improved fraud detection on card-present transaction sequences whereas there is no benefit on card-not-present sequences. Our third approach to context integration focuses on account-level and merchant-level feature aggregates. We wrap existing def-

initions of feature aggregations in a uniform concept and present a feature selection study for exploring and evaluating thousands of features aggregates. The results suggest that account-level aggregates are more informative than merchant-level aggregates and that the aggregation spaces contain many aggregates with similarly good performance as long as the aggregates capture transaction volumes per time-intervals.

The key message to take away from this thesis is: Automated credit card fraud detection is challenging in many regards and, most likely, there is no single method that can address and solve all of the issues in one go. The abnormality of a transaction depends on a number of factors, among which the card holder's purchase history emerges as the most important one. However, the demands for a fraud detection system to be useful in practice go beyond raw detection performance. Companies require means to inspect and trace back the decisions of such systems for being able to convince both customers and stakeholders about the reliability of their systems. In order to emphasize the importance of this aspect, we sketched a more transparent and simpler approach to fraud detection as possible direction for future research.

Appendix A

Searching Feature Aggregates

A.1 Example of JSON specification

As an example, the actual JSON specification of aggregations from Fig. 6.1 is depicted in Fig. A.1.

```
{
    "name": "example-set-of-aggregates",
    "aggregations": [
        {
            "target": {"name": "TX_AMOUNT", "function": "sum", "lower_idx": -1000, "upper_idx":0},
            "context": [
                {"name": "INT_DT_TIMESTAMP", "lower_val": -86400.0, "upper_val": 0.0}
            ]
        },
        {
            "target": {"name": "None", "function": "count", "lower_idx": -100, "upper_idx":0},
            "context": [
                {"name": "TERM_COUNTRY", "lower_val": 0.0, "upper_val": 0.0},
                {"name": "TERM_MCC", "lower_val": 0.0, "upper_val": 0.0}
            ]
        },
        {
            "target": {"name": "INT_DT_TIMESTAMP", "function": "span", "lower_idx": -1, "upper_idx"↩
                :0},
            "context": [ ]
        }
    ]
}
```

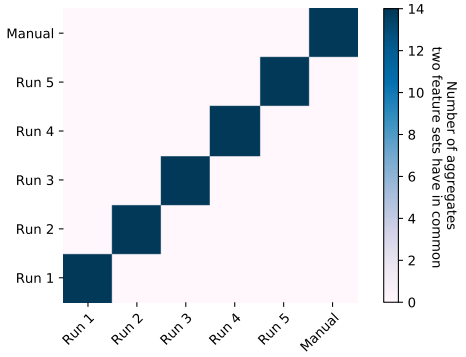
Figure A.1: The specification of aggregations from Fig. 6.1 in `JSON`-Format. *lower_idx* and *upper_idx* correspond to $[u, v]$, *lower_val* and *upper_val* correspond to $[p, q]$.

A.2 Hyper-parameters used in feature selection

- Decision tree: Implementation from `scikit-learn` with `criterion='gini'`, `max_depth=15`, `splitter='random'`, `max_features='auto'`. We kept all other parameters at their default values.

- Random forest: Implementation from `scikit-learn` with `n_estimators=300`, `min_samples_leaf=3`. We kept all other parameters at their default values. The training set consists of transactions from 100000 randomly drawn genuine card

(a) F2F - user-space



(b) F2F - user/merchant-space

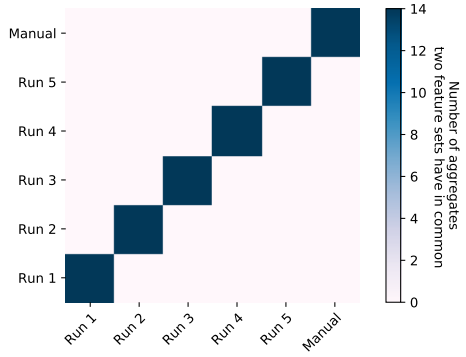


Figure A.2: Comparison between the 14 manually defined aggregates from Section 4.1 and 14 automatically selected aggregates, in terms of the number of features they have in common. The set of manually defined aggregates is denoted by ”Manual” and the five repeated runs of greedy forward selection are denoted by ”Run 1” to ”Run 5”. While the predictive performances are comparable, any pair of feature sets shares at most one identical aggregate - all other aggregates are different.

holders and all compromised card holders from the training period 01.03.2015-30.04.2015. The test set consists of all transactions in the test period 08.05.2015-31.05.2015.

A.3 Feature aggregates selected in the face-to-face scenario

See Table A.1 for a list of aggregates selected in the first iteration. The pairwise comparison of sets of 14 selected aggregates in shown in Fig. A.2.

Exp.	Space	Run	Target Variable	Operator	Index range	Context Variable	Value range
U	U	1	amount	sum	$[-1000, 0]$	timestamp	$[-1, 0]$
						terminalCategory	$[0, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						emv	$[0, 0]$
	U	2	amount	max	$[-1000, 0]$	timestamp	$[-24, 0]$
						terminalCategory	$[0, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						authentication	$[0, 0]$
						weekday	$[0, 0]$
	U	3	amount	max	$[-1000, 0]$	timestamp	$[-672, 0]$
						terminalID	$[0, 0]$
						terminalCategory	$[0, 0]$
						3DSecure	$[0, 0]$
						authentication	$[0, 0]$
						hourOfDay	$[0, 0]$
	U	4	amount	sum	$[-1000, 0]$	cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						emv	$[0, 0]$
						authentication	$[0, 0]$
						weekday	$[0, 0]$
						hourOfDay	$[0, 0]$
	U	5	amount	count	$[-1000, 0]$	terminalID	$[0, 0]$
						terminalCountry	$[0, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						emv	$[0, 0]$
						weekday	$[0, 0]$
						hourOfDay	$[0, 0]$
U/M	M	1	amount	min	$[-1000, 0]$	userID	$[0, 0]$
						age	$[0, 0]$
						gender	$[0, 0]$
						expiryDate	$[0, 0]$
						authentication	$[0, 0]$
						amount	$[-1, 1]$
	M	2	amount	max	$[-1000, 0]$	timestamp	$[-672, 0]$
						age	$[0, 0]$
						expiryDate	$[0, 0]$
						cardType	$[0, 0]$
						authentication	$[0, 0]$
						creditLimit	$[-100, 100]$
						userCountry	$[0, 0]$
						cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						weekday	$[0, 0]$
						hourOfDay	$[-1, 1]$
	M	3	amount	sum	$[-1000, 0]$	gender	$[0, 0]$
						expiryDate	$[0, 0]$
						userCountry	$[0, 0]$
						cardEntryMode	$[0, 0]$
						emv	$[0, 0]$
						weekday	$[0, 0]$
						hourOfDay	$[-1, 1]$
						amount	$[-100, 100]$
	U	4	amount	sum	$[-1000, 0]$	cardEntryMode	$[0, 0]$
						3DSecure	$[0, 0]$
						emv	$[0, 0]$
						amount	$[0, 0]$
	U	5	amount	sum	$[-1000, 0]$	timestamp	$[-168, 0]$
						terminalID	$[0, 0]$
						terminalCategory	$[0, 0]$
						terminalCountry	$[0, 0]$
						cardEntryMode	$[0, 0]$
						emv	$[0, 0]$
						hourOfDay	$[0, 0]$

Table A.1: Face-to-Face: Aggregate selected in the first iteration of the greedy forward selection method from the user-space (U) and the joint user/merchant-space (U/M).

Appendix B

Implementation

The entire code base is written for Python 3.6.4. For building the C/CUDA library, execute the makefile in the respective directory. The code is available from gitlab, hosted at the chair of data science at the university of Passau. To get access, please contact the author of this thesis or Professor Michael Granitzer.

- Credit card fraud detection experimentation framework:
  `https://gitlab.padim.fim.uni-passau.de/jjurgovsky/ADITS/tree/DEV-johannes`
- Searching feature aggregates:
  `https://gitlab.padim.fim.uni-passau.de/jjurgovsky/feature-aggregations`

Used packages:

- numpy, `https://numpy.org/`, version 1.16.1.
- pandas, `https://pandas.pydata.org/`, version 0.24.1.
- matplotlib, `https://matplotlib.org/`, version 3.0.3.
- scikit-learn, `https://scikit-learn.org/stable/`, version 0.20.2.
- keras, `https://keras.io/`, version 2.0.7.
- gensim, `https://radimrehurek.com/gensim/`, version 2.3.0.

Bibliography

[ABBL04] Hessein A. Abbass, Jaume Bacardit, Martin V. Butz, and Xavier Llora. Online adaptation in learning classifier systems: stream data mining. 51(217):61801, 2004. (cited on p. 21.)

[AD90] Hussein Almuallim and Thomas G. Dietterich. Learning with many irrelevant features. In *Proceedings of the 9th National Conference on Artificial Intelligence*, volume 91, pages 547–552. Citeseer, 1990. (cited on p. 117.)

[Akh13] John Akhilomen. Data Mining Application for Cyber Credit-Card Fraud Detection System. In Petra Perner, editor, *Advances in Data Mining. Applications and Theoretical Aspects. (ICDM).*, pages 218–228. Springer, 2013. (cited on p. 3.)

[AMZ16] Aisha Abdallah, Mohd Aizaini Maarof, and Anazida Zainal. Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68:90–113, 2016. (cited on p. 13.)

[AZLS18] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the Convergence Rate of Training Recurrent Neural Networks. *arXiv preprint arXiv:1810.12065*, 2018. (cited on p. 94.)

[Ban18] European Central Bank. Fifth report on card fraud, September 2018. 2018. (cited on p. 5, 6, and 7.)

[BAO15] Alejandro Correa Bahnsen, Djamila Aouada, and Björn Ottersten. Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19):6609–6619, 2015. (cited on p. 23.)

[BASO16] Correa Alejandro Bahnsen, Djamila Aouada, Aleksander Stojanovic, and Bjorn Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134 – 142, 2016. (cited on p. 19, 59, 60, 107, and 108.)

[Bay15] Justin Bayer. *Learning Sequence Representation.* PhD thesis, München, Technische Universität München, 2015. (cited on p. 94.)

[BCMLK16] Loic Bontemps, Van Loi Cao, James Mcdermott, and Nhien-An Le-Khac. Collective anomaly detection based on long short term memory recurrent neural network. In *Proceedings of the 3rd Internation Conference on*

Future Data and Security Engineering. Springer International Publishing, 03 2016. (cited on p. 21.)

[BDF+05] Alexandre Bureau, Josée Dupuis, Kathleen Falls, Kathryn L. Lunetta, Brooke Hayward, Tim P. Keith, and Paul Van Eerdewegh. Identifying SNPs predictive of phenotype using random forests. *Genetic Epidemiology*, 28(2):171–182, 2005. (cited on p. 121.)

[BDK14] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, volume 1, pages 238–247, 2014. (cited on p. 67, 70, 73, and 75.)

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. (cited on p. 67.)

[BEWT03] Ron Bekkerman, Ran El-Yaniv, Yoad Winter, and Naftali Tishby. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 3(Mar):1183–1208, 2003. (cited on p. 116.)

[BFOS17] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Routledge, 2017. (cited on p. 45 and 48.)

[BH01] Richard J. Bolton and David J. Hand. Unsupervised Profiling Methods for Fraud Detection. In *Proceedings of the 7th International Conference on Credit Scoring and Credit Control*, pages 5–7, 2001. (cited on p. 16 and 20.)

[BHPB02] Richard J. Bolton, David J. Hand, Foster Provost, and Leo Breiman. Statistical Fraud Detection: A Review. *Statistical Science*, 17(3):235–255, 2002. (cited on p. 44.)

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 1 edition, 2006. (cited on p. 35.)

[BJTW11] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J. Christopher Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602–613, 2011. (cited on p. 14, 15, and 19.)

[BKSN00] Markus M. Breuniq, Hans-Peter Kriegel, Jörg Sander, and Raymond T. Ng. LOF: Identifying Density-based Local Outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 29 of *SIGMOD '00*, pages 93–104. ACM, 2000. (cited on p. 15.)

[BL02] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 2002. (cited on p. 115 and 118.)

[BLH99] Rüdiger Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. In *In Proceedings of the 11th International Conference on Tools with Artificial Intelligence*, pages 103–106, nov 1999. (cited on p. 59.)

[BNJ01] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, 3(Jan):601–608, 2001. (cited on p. 36 and 132.)

[Bre01] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. (cited on p. 49 and 116.)

[BSAO13] Alejandro Correa Bahnsen, Aleksandar Stojanovic, Djamila Aouada, and Björn Ottersten. Cost sensitive credit card fraud detection using bayes minimum risk. In *Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA)*, volume 1, pages 333–338, 2013. (cited on p. 19.)

[BTB14] Elia Bruni, Nam Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47, 2014. (cited on p. 76.)

[CBC+19] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Yacine Kessaci, Frédéric Oblé, and Gianluca Bontempi. Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, 2019. (cited on p. 131.)

[CBHK02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(Sept. 28):321–357, 2002. (cited on p. 17.)

[CBK12] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012. (cited on p. 20.)

[CCS12] Adele Cutler, D. Richard Cutler, and John R. Stevens. *Random forests*, volume 45, pages 157–175. 2012. (cited on p. 121.)

[CF14] Rich Caruana and Dayne Freitag. Greedy Attribute Selection. In *Machine Learning Proceedings 1994*, pages 28–36. Elsevier, 2014. (cited on p. 118.)

[CFC17] Nuno Carneiro, Gonçalo Figueira, and Miguel Costa. A data mining based system for credit-card fraud detection in e-tail. *Decision Support Systems*, 95:91–101, 2017. (cited on p. 14.)

[CFPS99] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and their Applications*, 14(6):67–74, Nov 1999. (cited on p. 15.)

[Cha03] Nitesh V. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *In Proceedings of the ICML'03 Workshop on Class Imbalances*, 2003. (cited on p. 17.)

[Cha09] Nitesh V. Chawla. Data Mining for Imbalanced Datasets: An Overview. In *Data Mining and Knowledge Discovery Handbook*, pages 875–886. Springer, 2009. (cited on p. 16.)

[CLBCB18] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, and Gianluca Bontempi. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. *International Journal of Data Science and Analytics*, 5(4):285–300, Jun 2018. (cited on p. 16.)

[CLHB03] Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer, 2003. (cited on p. 17 and 18.)

[CMK08] Varun Chandola, Varun Mithal, and Vipin Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *2008 Eighth IEEE International Conference on Data Mining*, pages 743–748, Dec 2008. (cited on p. 20.)

[Com86] Federal Trade Commission. Fair Credit Billing Act. *Truth in Lending Act, Amendments*, 15 USC 160, 1986. (cited on p. 6.)

[CPARS13] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The Expressive Power of Word Embeddings. *arXiv preprint arXiv:1301.3226*, 2013. (cited on p. 75 and 78.)

[CWB+11a] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (almost) from Scratch. *Machine Learning Research*, 12:2493–2537, nov 2011. (cited on p. 67, 70, and 73.)

[CWB+11b] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011. (cited on p. 97.)

[Dal13] Andrea Dal Pozzolo. *Adaptive real-time machine learning for credit card fraud detection*. PhD thesis, Université libre de Bruxelles, 2013. (cited on p. 17.)

[DAP09] Linda Delamaire, Hussein Abdou, and John Pointon. Credit card fraud and detection techniques: A review. *Banks and Bank Systems*, 4(2):57–68, 2009. (cited on p. 4 and 5.)

[DCL+14] Andrea Dal Pozzolo, Olivier Caelen, Yann Aël Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41(10):4915–4928, 2014. (cited on p. 15, 19, 22, 44, and 50.)

[DCP13] E. Dorj, C. Chen, and M. Pecht. A bayesian hidden markov model-based approach for anomaly detection in electronic systems. In *2013 IEEE Aerospace Conference*, pages 1–10, March 2013. (cited on p. 20.)

[DG06] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM. (cited on p. 39.)

[Dho12] Shailesh S Dhok. Credit card fraud detection using hidden markov model. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(1):231–237, 2012. (cited on p. 20.)

[DMK03] Inderjit S Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of machine learning research*, 3(Mar):1265–1287, 2003. (cited on p. 116.)

[Doy93] Kenji Doya. Bifurcations of Recurrent Neural Networks in Gradient Descent Learning. *IEEE Transactions on neural networks*, 1(75):75–80, 1993. (cited on p. 94.)

[DP03] Chris Ding and Henchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *In Proceedings of the IEEE Bioinformatics Conference*, pages 523–528, 2003. (cited on p. 117.)

[DPCB15] Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. When is undersampling effective in unbalanced classification tasks? In Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, Carlos Soares, João Gama, and Alípio Jorge, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 200–215, Cham, 2015. Springer International Publishing. (cited on p. 17.)

[DT98] Ashutosh Deshmukh and Lakshminarayana Talluru. A rule-based fuzzy reasoning system for assessing the risk of management fraud. *Intelligent*

Systems in Accounting, Finance and Management, 7(4):223–241, 1998. (cited on p. 14.)

[DWHS15] Nan Du, Yichen Wang, Niao He, and Le Song. Time-Sensitive Recommendation From Recurrent User Activities. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28 (NIPS)*, volume 1, pages 1–11. Curran Associates, Inc., 2015. (cited on p. 43.)

[Dye14] Chris Dyer. Notes on Noise Contrastive Estimation and Negative Sampling. *arXiv preprint arXiv:1410.8251*, 2014. (cited on p. 68 and 69.)

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. (cited on p. 16.)

[Elm90] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. (cited on p. 92.)

[EN96] K. J. Ezawa and S. W. Norton. Constructing bayesian networks to predict uncollectible telecommunications accounts. *IEEE Expert*, 11(5):45–51, Oct 1996. (cited on p. 15.)

[Eur12] Europol. Payment Card Fraud in the European Union - Situation Report. 2012. (cited on p. 6.)

[FCTZ16] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In Akira Hirose, Seiichi Ozawa, Kenji Doya, Kazushi Ikeda, Minho Lee, and Derong Liu, editors, *Neural Information Processing*, pages 483–490, Cham, 2016. Springer International Publishing. (cited on p. 19.)

[FGM+02] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, 2002. (cited on p. 76.)

[Fir57] J R Firth. A synopsis of linguistic theory studies in linguistic analysis. In *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957. (cited on p. 68.)

[For03] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3(Mar):1289–1305, 2003. (cited on p. 117.)

[GB16] Cheng Guo and Felix Berkhahn. Entity Embeddings of Categorical Variables. *arXiv preprint arXiv:1604.06737*, 2016. (cited on p. 97.)

[GE03] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3(Mar):1157—-1182, 2003. (cited on p. 116.)

[GFHY07] Jing Gao, Wei Fan, Jiawei Han, and Philip S. Yu. A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 3–14. SIAM, 2007. (cited on p. 21 and 22.)

[GH10] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9, pages 297–304. PMLR, 2010. (cited on p. 69.)

[GJ14] Alex Graves and Navdeep Jaitly. Towards End-To-End Speech Recognition with Recurrent Neural Networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32, pages 1764–1772, 2014. (cited on p. 21 and 94.)

[GL98] Amit Gupta and Siuwa M. Lam. Weight decay backpropagation for noisy data. *Neural Networks*, 11(6):1127–1138, 1998. (cited on p. 116.)

[GL14] Yoav Goldberg and Omer Levy. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014. (cited on p. 68.)

[GL16] Aditya Grover and Jure Leskovec. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA, 2016. ACM. (cited on p. 70.)

[GPTM10] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010. (cited on p. 116.)

[GR94] Ghosh and Reilly. Credit card fraud detection with a neural-network. In *Proceedings of the 27th Hawaii International Conference on System Sciences. (HICSS).*, volume 3, pages 621–630. IEEE, 1994. (cited on p. 4 and 15.)

[Gra12] Alex Graves. *Supervised sequence labelling with recurrent neural networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012. (cited on p. 21.)

[Gra13] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. (cited on p. 94.)

[GWBV02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002. (cited on p. 116.)

[Han09] David J. Hand. Measuring classifier performance: A coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123, 2009. (cited on p. 39.)

[HC12] David J. Hand and Martin J. Crowder. Overcoming selectivity bias in evaluating new fraud detection systems for revolving credit operations. *International Journal of Forecasting*, 28(1):216–223, 2012. (cited on p. 14 and 23.)

[Hin86] Geoffrey E. Hinton. Learning distributed representations of concepts. In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, volume 1, pages 1–12. Amherst, MA, 1986. (cited on p. 116.)

[Hof01] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196, Jan 2001. (cited on p. 132.)

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. (cited on p. 94 and 95.)

[HSMN12] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. (cited on p. 67.)

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: Data Mining, Inference and Prediction*, volume 2. Springer New York, 2009. (cited on p. 46, 48, 49, and 50.)

[HVL15] C. Sweetlin Hemalatha, V. Vaidehi, and R. Lakshmi. Minimal infrequent pattern based approach for mining outliers in data streams. *Expert Systems with Applications*, 42(4):1998 – 2012, 2015. (cited on p. 15.)

[HWA+08] D. J. Hand, C. Whitrow, N. M. Adams, P. Juszczak, and D. Weston. Performance criteria for plastic card fraud detection tools. *Journal of the Operational Research Society*, 59(7):956–962, 2008. (cited on p. 23.)

[HWM05] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005. (cited on p. 17.)

[HYGS08] He Haibo, Bai Yang, E. A. Garcia, and Li Shutao. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) (IJCNN)*, number 3, pages 1322– 1328. IEEE, 2008. (cited on p. 17.)

[IMJ+11] Divya Iyer, Arti Mohanpurkar, Sneha Janardhan, Dhanashree Rathod, and Amruta Sardeshmukh. Credit card fraud detection using hidden Markov model. *Proceedings of the World Congress on Information and Communication Technologies (WICT)*, 5(1):1062–1066, jan 2011. (cited on p. 20.)

[IT13] Avinash Ingole and R C Thool. Credit Card Fraud Detection Using Hidden Markov Model and Its Performance. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(6):626–632, 2013. (cited on p. 21.)

[JAH+08] Piotr Juszczak, Niall M. Adams, David J. Hand, Christopher Whitrow, and David J. Weston. Off-the-peg and bespoke classifiers for fraud detection. *Computational Statistics & Data Analysis*, 52(9):4521 – 4532, 2008. (cited on p. 16 and 131.)

[JGC12] Sanjeev Jha, Montserrat Guillen, and J. Christopher Westland. Employing transaction aggregation strategy to detect credit card fraud. *Expert Systems with Applications*, 39(16):12650–12657, 2012. (cited on p. 19.)

[JGS16] Johannes Jurgovsky, Michael Granitzer, and Christin Seifert. Evaluating memory efficiency and robustness of word embeddings. In *European Conference on Information Retrieval (ECIR)*, volume 9626, pages 200–211. Springer, 2016. (cited on p. 11, 57, and 73.)

[JGW12] Sanjeev Jha, Montserrat Guillen, and Christopher J. Westland. Employing transaction aggregation strategy to detect credit card fraud. *Expert Systems with Applications*, 39(16):12650–12657, 2012. (cited on p. 19.)

[JGZ+18] Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre Edouard Portier, Liyun He-Guelton, and Olivier Caelen. Sequence Classification for Credit-card Fraud Detection. *Expert Systems with Applications*, 100:234–245, 2018. (cited on p. 11 and 89.)

[JKP14] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant Features and the Subset Selection Problem. In *Machine Learning Proceedings 1994*, pages 121–129. Elsevier, 2014. (cited on p. 117 and 118.)

[Jon73] Karen Sparck Jones. Index term weighting. *Information Storage and Retrieval*, 9(11):619–633, 1973. (cited on p. 36.)

[JS02] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002. (cited on p. 17.)

[KC02] Nojun Kwak and Chong Ho Choi. Input feature selection for classification problems. *IEEE Transactions on Neural Networks*, 13(1):143–159, 2002. (cited on p. 116.)

[KM97] Miroslav Kubat and Stan Matwin. Addressing the Curse of Imbalanced Training Sets: One Sided Selection. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 179–186. Nashville, USA, 1997. (cited on p. 17.)

[KR92] K Kira and L Rendell. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *In Proceedings of the 10th National Conference on Artificial Intelligence*, volume 2, pages 129–134, 1992. (cited on p. 117.)

[KRA99] Ralf Klinkenberg, Ingrid Renz, and Daimler-benz Ag. Adaptive information filtering: Learning in the presence of concept drifts. 03 1999. (cited on p. 22.)

[Kri10] M. Krivko. A hybrid model for plastic card fraud detection systems. *Expert Systems with Applications*, 37(8):6070–6076, 2010. (cited on p. 19.)

[KZP07] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007. (cited on p. 35.)

[LA09] Naeimeh Laleh and Mohammad Abdollahi Azgomi. A taxonomy of frauds and fraud detection techniques. In *Communications in Computer and Information Science*, volume 31, pages 256–267. Springer, 2009. (cited on p. 4 and 5.)

[LG14] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In Z Ghahramani, M Welling, C Cortes, N D Lawrence, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems. (NIPS).*, pages 2177–2185. Curran Associates, Inc., 2014. (cited on p. 68, 69, and 73.)

[LPL+19a] Yvan Lucas, Pierre-Edouard Portier, Léa Laporte, Sylvie Calabretto, Olivier Caelen, Liyun He-Guelton, and Michael Granitzer. Multiple perspectives hmm-based feature engineering for credit card fraud detection. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, pages 1359–1361, New York, NY, USA, 2019. ACM. (cited on p. 15 and 21.)

[LPL+19b] Yvan Lucas, Pierre-Edouard Portier, Léa Laporte, Sylvie Calabretto, Liyun He-Guelton, Frederic Oblé, and Michael Granitzer. Dataset shift quantification for credit card fraud detection. *Artificial intelligence and knowledge engineering (AIKE2019)*, 2019. (cited on p. 22.)

[LS94] P Langley and S Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAIWorkshop on Case-Based Reasoning*, pages 113–117. 1994. (cited on p. 118.)

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi Hua Zhou. Isolation forest. In *Eighth IEEE International Conference on Data Mining. (ICDM).*, pages 413–422. IEEE, 2008. (cited on p. 15.)

[LWW15] Dao Lam, Mingzhen Wei, and Donald Wunsch. Clustering Data of Mixed Categorical and Numerical Type With Unsupervised Feature Learning. *IEEE Access*, 3:1605–1613, 2015. (cited on p. 35.)

[LWZ09] X. Liu, J. Wu, and Z. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, April 2009. (cited on p. 18.)

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013. (cited on p. 67 and 73.)

[MD15] Nader Mahmoudi and Ekrem Duman. Detecting credit card fraud by Modified Fisher Discriminant Analysis. *Expert Systems with Applications*, 42(5):2510–2516, 2015. (cited on p. 15 and 23.)

[MDK+11] Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černock\‘y. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *12th Annual Conference of the International Speech Communication Association*, pages 606–608, 2011. (cited on p. 36 and 94.)

[MH08] Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA, 2008. ACM. (cited on p. 67.)

[ML94] Andrew W. Moore and Mary S. Lee. Efficient Algorithms for Minimizing Cross Validation Error. In *Machine Learning Proceedings 1994*, pages 190–198. Elsevier, 1994. (cited on p. 118.)

[MN16a] Jianyu Miao and Lingfeng Niu. A Survey on Feature Selection. *Procedia Computer Science*, 91:919–926, 2016. (cited on p. 117.)

[MN16b] Tatsuya Minegishi and Ayahiko Niimi. Proposal of Credit Card Fraudulent Use Detection by Online-type Decision Tree Construction and Verification of Generality. *International Journal for Information Security Research*, 3(1):229–235, 2016. (cited on p. 59.)

[MSB08] P. E. Meyer, C. Schretter, and G. Bontempi. Information-theoretic feature selection in microarray data using variable complementarity. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):261–274, June 2008. (cited on p. 116.)

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. (cited on p. 67 and 68.)

[MTRAR+12] Jose G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012. (cited on p. 21 and 42.)

[MTVM02] Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. Credit card fraud detection using bayesian and neural networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, pages 261–270, 2002. (cited on p. 15.)

[MYZ13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751, 2013. (cited on p. 67, 75, and 76.)

[NHW+11] E W T Ngai, Yong Hu, Y H Wong, Yijun Chen, and Xin Sun. The application of data mining techniques in financial fraud detection. *Decision Support Systems*, 50(3):559–569, 2011. (cited on p. 13 and 14.)

[NSS05] Julia Neumann, Christoph Schnörr, and Gabriele Steidl. Combined SVM-based feature selection and classification. *Machine Learning*, 61(1-3):129–150, 2005. (cited on p. 116.)

[Par07] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 2007. (cited on p. 117.)

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM. (cited on p. 70.)

[PCJB15] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *In Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 159–166. IEEE, 2015. (cited on p. 25.)

[Pep03] Margaret Sullivan Pepe. *The Statistical Evaluation of Medical Tests for Classification and Prediction*, volume 100 of *Oxford statistical science series*. Oxford University Press, 2003. (cited on p. 36.)

[PGLSM05] Clifton Phua, Ross W. Gayler, Vincent C.S. Lee, and Kate Smith-Miles. On the Approximate Communal Fraud Scoring of Credit Applications. *Proceedings of Credit Scoring and Credit Control*, pages 1–10, 2005. (cited on p. 6.)

[PLM01] Fernando Pereira, John D Lafferty, and Andrew McCallum. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *In Proceedings of 18th International Conference on Machine Learning (ICML)*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., 2001. (cited on p. 20.)

[PLSG10] Clifton Phua, Vincent C. S. Lee, Kate Smith-Miles, and Ross W. Gayler. A comprehensive survey of data mining-based fraud detection research. *CoRR*, abs/1009.6119, 2010. (cited on p. 13, 23, and 130.)

[PS15] Hemlata Pant and Reena Srivastava. a Survey on Feature Selection Methods for. *International Journal of Computer Engineering and Applications*, IX(Ii):197–204, 2015. (cited on p. 118.)

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. (cited on p. 67.)

[PSO16] Raghavendra Patidar, Lokesh Sharma, and Others. Credit Card Fraud Detection Using Fuzzy Logic and Neural Network. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(32-38), 2016. (cited on p. 4.)

[PVG+12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2012. (cited on p. 98.)

[QCSSL09] J Quinonero-Candela, M Sugiyama, A Schwaighofer, and N D Lawrence. *When Training and Test Sets Are Different: Characterizing Learning Transfer*. MIT Press, 2009. (cited on p. 42.)

[Qui86] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986. (cited on p. 45.)

[Qui96] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4(1):77–90, 1996. (cited on p. 45.)

[Qui14] J. R. Quinlan. *C4. 5 programs for machine learning. Elsevier*. Morgan Kaufmann, 2014. (cited on p. 45.)

[RA18] William N. Robinson and Andrea Aria. Sequential fraud detection for prepaid cards using hidden markov model divergence. *Expert Systems with Applications*, 91:235 – 251, 2018. (cited on p. 21.)

[RH97] Rudy Setiono and Huan Liu. Neural-network feature selector. *IEEE Transactions on Neural Networks*, 8(3):654–662, 1997. (cited on p. 116.)

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by error propagation. *Nature*, 323(6088):533–536, 1986. (cited on p. 92 and 93.)

[RMN+99] Saharon Rosset, Uzi Murad, Einat Neumann, Yizhak Idan, and Gadi Pinkas. Discovery of fraud rules for telecommunications—challenges and solutions. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 409–413, New York, NY, USA, 1999. ACM. (cited on p. 14.)

[RP04] Andrew Y. Ng Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, number October, pages 1631–1642, 2004. (cited on p. 97.)

[RP16] Petar Ristoski and Heiko Paulheim. RDF2Vec: RDF Graph Embeddings for Data Mining. In Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web (ISWC)*, pages 498–514, Cham, 2016. Springer International Publishing. (cited on p. 70 and 71.)

[Saa03] Youssef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. (cited on p. 78.)

[SBD13] Yusuf Sahin, Serol Bulkan, and Ekrem Duman. A cost-sensitive decision tree approach for fraud detection. *Expert Systems with Applications*, 40(15):5916–5923, 2013. (cited on p. 19 and 23.)

[SBZH06] Carolin Strobl, Anne-laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in Random Forest Variable Importance Measures Variable selection bias. *Statistical Modelling*, 8(1):25, 2006. (cited on p. 116.)

[SCB+12] David Steenhoff, Alain Coletta, Hugues Bersini, Cosmin Lazar, Stijn Meganck, Ann Nowe, Virginie de Schaetzen, Colin Molter, Jonatan Taminau, and Robin Duque. A Survey on Filter Techniques for Feature Selection in Gene Expression Microarray Analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1106–1119, 2012. (cited on p. 117.)

[SFL+97] Salvatore J. Stolfo, David W. Fan, Wenke Lee, Andreas L. Prodromidis, and Philip K. Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. In *In Proceedings of the AAAI'97 Workshop on Fraud Detection and Risk Management*, pages 83–90, 1997. (cited on p. 15.)

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. (cited on p. 98.)

[SLMJ15] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, 2015. (cited on p. 76.)

[SR15] Takaya Saito and Marc Rehmsmeier. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10(3):1–21, 2015. (cited on p. 39.)

[Str08] Carolin Strobl. *Statistical Issues in Machine Learning – Towards Reliable Split Selection and Variable Importance Measures*. Cuvillier Verlag, 2008. (cited on p. 121.)

[SU04] David J. Stracuzzi and Paul E. Utgoff. Randomized variable elimination. *Journal of Machine Learning Research*, 5(Oct):1331–1362, 2004. (cited on p. 118.)

[SVCS09] D. Sánchez, M. A. Vila, L. Cerda, and J. M. Serrano. Association rules applied to credit card fraud detection. *Expert Systems with Applications*, 36(2 PART 2):3630–3640, 2009. (cited on p. 59.)

[SVL14a] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems (NIPS)*, pages 3104–3112, 2014. (cited on p. 21.)

[SVL14b] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014. (cited on p. 94.)

[SYY02] M. Syeda, Yan-Qing Zhang, and Yi Pan. Parallel granular neural networks for fast credit card fraud detection. In *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No.02CH37291)*, volume 1, pages 572–577 vol.1, May 2002. (cited on p. 15.)

[TFL+15] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, 2015. (cited on p. 76.)

[TP10] Peter Turney and Patrick Pantel. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010. (cited on p. 67.)

[TWY+15] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565, 2015. (cited on p. 97.)

[VBC+15] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015. (cited on p. 19 and 20.)

[VDD04] Stijn Viaene, Richard A. Derrig, and Guido Dedene. A case study of applying boosting naive bayes to claim fraud diagnosis. *IEEE Trans. Knowl. Data Eng.*, 16(5):612–620, 2004. (cited on p. 15.)

[VE14] Jorge R. Vergara and Pablo A. Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24(1):175–186, jan 2014. (cited on p. 117.)

[VH08] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2625, 2008. (cited on p. 75.)

[VR05] Sofia Visa and A. Ralescu. Issues in Mining Imbalanced Data Sets - A Review Paper. In *Proceedings of the 16th Midwest Artificial Intelligence and Cognitive Science Conference*, volume 2005, pages 67–73. sn, 2005. (cited on p. 17.)

[Wer88] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988. (cited on p. 92.)

[WEST03] Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3(Mar):1439–1461, 2003. (cited on p. 116.)

[WFP99] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: alternative data models. In *Proceedings of the 1999 IEEE*

Symposium on Security and Privacy (Cat. No.99CB36344), pages 133–145, May 1999. (cited on p. 20.)

[WFYH03] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining Concept-drifting Data Streams Using Ensemble Classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 226—-235. ACM, 2003. (cited on p. 22.)

[WHA+08] David J. Weston, David J. Hand, Niall M. Adams, Christopher Whitrow, and Piotr Juszczak. Plastic card fraud detection using peer group analysis. *Advances in Data Analysis and Classification*, 2(1):45–62, Apr 2008. (cited on p. 16.)

[WHJ+09] C. Whitrow, D. J. Hand, P. Juszczak, D. Weston, and N. M. Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55, 2009. (cited on p. 18, 19, 23, and 107.)

[WK96] Gerhard Widmer and Miroslav Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23(1):69–101, apr 1996. (cited on p. 21, 22, and 42.)

[WO09] Bénard Wiese and Christian Omlin. Credit card transactions, fraud detection, and machine learning: Modelling time with LSTM recurrent neural networks. In *Studies in Computational Intelligence*, volume 247, pages 231–268. Springer, 2009. (cited on p. 21 and 91.)

[WP01] Gary M Weiss and Foster Provost. The Effect of Class Distribution on Classifier Learning. *Technical report*, 2001. (cited on p. 17.)

[WTY09] Shuo Wang, Ke Tang, and Xin Yao. Diversity exploration and negative correlation learning on imbalanced data sets. In *2009 International joint conference on neural networks*, pages 3259–3266. Citeseer, 2009. (cited on p. 18.)

[XZB13] Bing Xue, Mengjie Zhang, and Will N. Browne. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics*, 43(6):1656–1671, 2013. (cited on p. 119.)

[YH98] Jihoon Yang and Vasant Honavar. *Feature Subset Selection Using a Genetic Algorithm*, pages 117–136. Springer US, Boston, MA, 1998. (cited on p. 119.)

[YhL09] I. Cheng Yeh and Che hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2 PART 1):2473–2480, 2009. (cited on p. 15.)

[YL03] Lei Yu and Huan Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 856–863, 2003. (cited on p. 117.)

[YLJH03] Rong Yan, Yan Liu, Rong Jin, and Alex Hauptmann. On predicting rare classes with SVM ensembles in scene classification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing.(ICASSP).*, volume 3, pages III—-21. IEEE, 2003. (cited on p. 17.)

[YPP14] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Network*, 5(2):157, 2014. (cited on p. 93.)

[Yus09] Silvia Casado Yusta. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, 30(5):525–534, 2009. (cited on p. 119.)

[ZCG+17] K Ziegler, O Caelen, M Garchery, M Granitzer, L He-Guelton, J Jurgovsky, P Portier, and S Zwicklbauer. Injecting Semantic Background Knowledge into Neural Networks using Graph Embeddings. In *26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises. (WETICE).*, pages 200–205, 2017. (cited on p. 11, 57, and 82.)

[ZM03] Jianping Zhang and Inderjeet Mani. kNN approach to unbalanced data distributions. In *In Proceedings of the ICML'03 Workshop on Learning from Imbalanced Datasets*, 2003. (cited on p. 17.)

[ZS15] Masoumeh Zareapoor and Pourya Shamsolmoali. Application of credit card fraud detection: Based on bagging ensemble classifier. *Procedia Computer Science*, 48:679 – 685, 2015. International Conference on Computer, Communication and Convergence (ICCC 2015). (cited on p. 15.)

[ZSG16] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Robust and Collective Entity Disambiguation Through Semantic Embeddings. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 425–434, New York, NY, USA, 2016. ACM. (cited on p. 70.)

[ZZQ17] Xujun Zhao, Jifu Zhang, and Xiao Qin. LOMA: A local outlier mining algorithm based on attribute relevance analysis. *Expert Systems with Applications*, 84:272–280, 2017. (cited on p. 16.)