Brandenburgische Technische Universität Cottbus-Senftenberg

Lehrstuhl Rechnernetze und Kommunikationssysteme

Bachelor Thesis

A Machine Learning Approach For Churn Prediction Within The Social Network Jodel

Max Bergmann

max.bergmann@b-tu.de

supervised by
Prof. Dr. rer. nat. Oliver Hohlfeld
M. Sc. Jens Helge Reelfs

March 2, 2021

10.26127/BTUOpen-6087

Abstract

Within the last decades, the number of social networks is growing fast. The competition of retaining the customers to grow their platform and increase their profitability is rising. That is why companies need to detect possible churners to retain these. The problem of predicting the users' lifetime, churning users, and the reasons for churning can be tackled by using machine learning.

The goal of this bachelor thesis is to build machine learning models to predict user churn and the user lifetime within the social network Jodel, a location-based anonymous messaging application for Android and iOS.

To get the best possible prediction results, we have started with extensive literature research, whose approaches we have tested and added to a machine learning pipeline to build predictive models. With these models, we have investigated the performance after different observation time windows and have finally compared the strongest models to detect similarities and understand the insights to learn their behaviour.

The results of this thesis are machine learning models for a selected representative set of communities varying in size within the Kingdom of Saudi Arabia and a country model leveraging all data. These models are used for a regression task by predicting the lifetime of a user and a multi-label classification of a user into six different churn classes. Additionally, we have also given models for a binary classification, where the model will predict if the user will churn within a given time or not. These models have shown general strong predictive power, which is shrinking when limiting the observation time window. Especially the binary classification yielded high accuracy of over 99%.

The best models have been used for predicting user churn within other communities to detect communities with possible similar behaviour. These similarities then have been determined by features' importance, where the most important features have got fed back into empirics. This has shown statistically significant differences between user groups with a different active time but as of today no clear trends were visible that had led us to define the communities' behaviours.

Since the competition of social networks is still growing, the retaining of users will stay a core marketing strategy, which will need to be tackled by machine learning and artificial intelligence. The created models could be useful for predicting churning users within the platform Jodel to detect these customers that will churn within a given time.

Researches did not focus much on anonymous and location-based messaging. That is why the results of this thesis on the anonymous messaging application Jodel opens a variety of possible tasks for the future in this context.

Contents

List of Abbreviations

AI - Artificial Intelligence
ANN - Artificial Neural Network
AUC - Area Under The Curve
CDF - Cumulative Density Function
CQA - Question Answering Site
CV - Cross-Validation
CLV - Customer Lifetime Value
DT - Decision Tree
FN - False Negative
FP - False Positive
FPR - False Positive Rate
GD - Gradient Descent
KNN - K-Nearest Neighbours
LogR - Logistic Regression
LR - Linear Regression
LTU - Linear Threshold Unit
MAE - Mean Absolute Error
ML - Machine Learning
MLP - Multi-Layer Perceptron
MSE - Mean Squared Error
NB - Naïve Bayes
NN - Neural Network
PCA - Principal Component Analysis
RanOS - Random Oversampler
RanUS - Random Undersampler
RF - Random Forest
RFE - Recursive Feature Elimination
RFECV - Recursive Feature Elimination with Cross-Validation
RFFI - Random Forest Feature Importance
RMSE - Root Mean Squared Error
ROC - Receiver Operating Characteristic
SGB - Stochastic Gradient Boost
SGD - Stochastic Gradient Descent
SMOTE - Synthetic Minority Oversampling Technique
SVM - Support Vector Machine
TN - True Negative
TP - True Positive
TPR - True Positive Rate

1 Introduction

Within the last years, the number of social networks is growing fast and the competition between the companies is getting bigger. Each of these platforms depends on an active user-base to stay alive and be attractive for new users. This user-base is threatened by user churn, which represents the leaving users on the platform. That is why the retaining of existing customers is a core marketing strategy [1], to avoid churning users. Therefore, the management tries to analyse what leads users to churn from their platform to increase their profitability and grow their platform. To achieve this, data and behavioural analysis are used to achieve a positive customer relationship. Each network has loyal users, that probably inadvertently advertise the product freely. Because of this, they tend to be more profitable for a company and are, therefore, more important to retain. To measure the profitability of a user, the customer lifetime value (CLV) is used, which denotes the expected profit over time in marketing. The users with a 'high value' are, therefore, more important to retain. One of the most popular scenarios is users who entering a network and leaving it after a short time for many different reasons. They probably only wanted to explore the platform shortly or had a bad experience with other customers. To counteract against churn, detecting these reasons is one of the main but also complex tasks.

The prediction of user churn is a well-studied data mining task (cf. Sec. 3). This task is not only limited to the detection of these users that will churn but also the likelihood of time until a user might churn. These probabilities allow the companies direct timed steering to avoid churn and improve the retention of users, by, e.g., sending (push) notifications or emails. This also refers to the steering of communities or user-bases to achieve an optimal feature in mind. This could include maintaining of healthy user-base, that should grow and converge into a well-mixed population to achieve a sustainable level.

Within this thesis, we will predict the user churn by using Machine Learning (ML) within the social network Jodel, which is a mobile-only location-based anonymous messaging application. The application uses the GPS location of a user and builds the communities around this. As a result, several distinct communities emerge within the country. The users then communicate within the different communities when changing their location, because they cannot communicate outside their location and therefore, makes the different communities disjoint. Because of this disjunction, the churn prediction enables the comparison between the different geographic communities on a countrywide level. In contrast to many other works, where non-anonymous networks were focused, the prediction on Jodel, which does not include user-profiles and the social credit does not play a role, enables us to study user churn in absence of any form of social ties.

To achieve the best prediction result, we will focus on the Random Forest (RF) in this thesis. With the help of this ML algorithm, we will build strong predictors for different tasks. On the one hand, we will predict the user lifetime on the platform which will give us a specific time (regression), and on the other hand, we will classify the users into different classes that symbolise the time period until a user churns (classification). This will show us which users tend to churn earlier and are, therefore, interesting to improve their retaining. Additional to this multi-label classification we will then break down the problem to a classification of a user, as a customer with a lifetime lower than a given threshold or greater. This makes the prediction easier and more practical by giving the information which users will churn within a given time and are, therefore, focused to avoid their churn.

With the built models, we will then have a look at their insights, to see how models of different communities behave in others and if we can detect similarities between them, by observing the most important features, their correlations, and detecting statistically significant differences between user groups for these features. It will be evaluated if this information can be used to define different community states, as well as learning and understanding the models' behaviour.

1.1 Jodel

In the spotlight of this bachelor thesis stands Jodel. It is a German mobile social media app for Android and iOS. The goal of Jodel is the local and anonymous real-time interaction between users. The providers want to give everyone a voice no matter who they are and aim to have helpful and peaceful communities. Therefore, they established the hashtag ''#GoodVibesOnly'' [2]. Within a dynamic radius, a user can post messages and pictures with short captions into a feed or channels that handle special topics. Users can also interact with other users by commenting and voting their posts up or down to decide what is talked about or what he likes to see [2]. The radius where a user can see and add posts can vary up to 20 km [3]. The users are communicating anonymously, as users do not have a dedicated user profile. This means users do not have a profile picture or other personal information available. When replying to others' posts, each user gets a consecutive ID within a single thread, to which everybody can refer. To represent the good vibes a user is spreading, Jodel established the so-called 'Karma' points which reflect light gamification for users to generate a positive atmosphere. They can be generated by posting and replying to help other users by commenting or voting up or down [4].

We show the main Android interface of Jodel in Fig. 1. It consists of three important parts. First, the header (1) where the user can switch between different channels (A), the current feed of the actual location, the hometown (B), or the personal information displayed by Karma points (C). The main part (2) includes the feed with up to 150 listed posts. And finally, the footer (3) where the user can switch between the real-time feed (D) or a feed with posts of the current day or the past week (E) or come back to the top with the newest posts (F). A typical post is presented in Fig. 2. To represent the diversity of the Jodel community each post gets a random colour [5]. At (I) the user can see the time since it was posted, the channel, e.g., @main, and the distance to the posting spot in different steps, which is shown in Tab. 1. (II) shows the number of comments and at (III) the user can vote the post and see the actual vote-score, which is a cumulated score calculated by *upvotes* − *downvotes*.

Figure 1: The Android interface of the Jodel App including the header at (1) with channels (A), current locations (B) and Karma points (C), the feed (2), and the footer (3) for selecting the time (D) - (F).

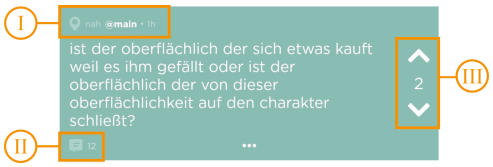

Figure 2: An example Jodel post including the distance and channel at (I), the number of comments at (II), and the panel for voting at (III), which also shows the actual votescore (*upvotes* − *downvotes*).

Tag	Distance
here	$\leq$ 1 km
very close	$\leq$ 2 km
close	$\leq$ 10 km
far	$>$ 10 km
hometown	Post with hometown feature outside the radius

Table 1: The distance tags and their meanings [6] are displayed at (I) in Fig. 2.

A special feature of Jodel is the distributed moderation system. The app integrates the users into moderation by letting them downvote the post or flagging it directly for moderation. To sort the reported posts, Jodel uses a selection of users, which are outstanding within the community and have shown that they can support the community guidelines strictly. These moderators can give each reported post a vote (blocked or allowed) or skip it if they are unsure. If enough moderators voted, an algorithm of Jodel finally decides if the post gets removed. If the post gets removed, the owner loses all earned Karma of this post and gets a penalty of -1000 Karma [7].

1.2 Dataset

In this bachelor thesis, we base our work on a dataset of Jodel from 20.12.2014 to 07.08.2017 within the Kingdom of Saudi Arabia with 95 cities. As described in Tab. 2, it includes three different tables, a user table with general information about each registered user, like the date of registration, the number of posts and replies, happyratio ($\frac{upvotes}{upvotes+downvotes}$), postratio ($\frac{creat.posts}{creat.posts+creat.replies}$), or the generated Karma, the content table with information about all created posts and replies including the date of creation, votes, a reference to the creator, or the number of flags, and an interactions table, that includes information like registrations, the creation of posts, or votes. This includes the date and the related user and content.

In our prediction tasks, we will focus on the *active duration in minutes* of a user from the user table. Because of the fact, that our dataset only includes meta-data and that we do not have data of passive consumption like just scrolling along the feed without posting or voting, this value is a lower bound, which is set at the last active interaction. There is also a limitation on the case of dates of votes, which are post affiliated. That is why the minimal time a user was active on Jodel represents our optimization target. This includes also the users who did not actively use Jodel by creating posts or voting. In the further progress of this thesis, we will refer to this value as '*active minutes*'.

table	content	# columns	# rows
user	user information like registration date, number of created posts or Karma	20	1,214,881
content	information about all created content like date, votes, Karma or the associated user	14	469,323,817
interactions	information about different interactions like registrations, votes or created posts	7	966,865,197

Table 2: The tables of the dataset and their columns.

The used dataset includes interactions of 1,214,881 users within the given time. To determine a period where many users registered at Jodel we looked at the numbers of registrations per month, which are shown in Fig. 3. As we can see, the number of users increased explosively in March 2017.

To determine characteristics of the observed users we looked at the number of interactions of a user, like posts, replies, or votes overall and outside the city of registration. As we can see in Tab. 3, 35% of the users did not post, about 28% did not reply and about 25% did not actively create content. In contrast to that, there are just 19% of the users who did not vote and therefore, less than active content creation. This shows us that there exist more users who are more likely to interact in a passive than in an active way.

In Tab. 4, we can also see that about one-third of the users mainly interacted outside their registration city. That is why we added the virtual capital for a user. This virtual capital is defined as the city, with most interactions of a user within. We have shown the number of users by their registration city, as well as by their virtual capital in Tab. 5.

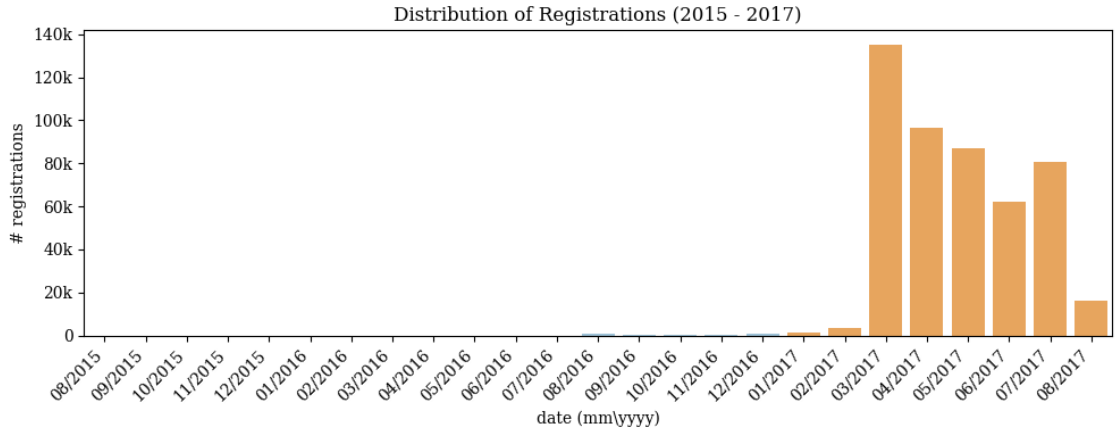

Figure 3: Distribution of registration numbers in the period from August 2015 to August 2017.

interaction	#users	ratio
posts	428,104	35%
replies	335,010	28%
active	299,889	25%
passive (votes)	231,472	19%

Table 3: The ratios of users with zero active content creation interactions (posts, replies, reply & post) and passive interactions (votes) to the number of all users (1,214,881).

	#users	ratio
within reg. city	679,358	56%
outside reg. city	361,490	30%
same in both	174,033	14%

Table 4: The number of users who interacted most within and outside their registration city, as well as users who equally interacted in both regions. We can see that there are about one-third of the users mainly interacting outside their registration city.

City	#user reg. city	#user virtual capital
Saudi Arabia	1,214,881	
Riyadh	314,156	289,963
Jeddah	114,976	103,386
Mecca	50,661	45,885
Al Bahah	12,931	11,458
Al Jafr	213	176

Table 5: The number of users in the dataset within the cities Riyadh, Jeddah, Mecca, Al Bahah, and Al Jafr, divided into the city of registration and the city with their most interactions as *virtual capital*.

As shown in Fig. 4, we also looked at the cumulative density function (CDF) and the quantiles of the numbers of posts, replies, and votes, as well as posts and replies as one. We can see that 25% of the users did not post, replied just once, and voted just twice, which means that there is a huge number of users who interact almost not at all. What we can also see is that in the 50% and 75% quantiles the users are more likely to interact by an active content creation than by votes. A special finding from this is the imbalance of the dataset. We see that there are 75% of the users who created content a maximum of 223 times and just 25% with more interactions and a maximum of about 38k posts or replies. When we have a look at the number of votes, we see that 75% of the users voted a maximum of 97 times and just 25% voted more often with a maximum of about 358k. That is why we identified the user interactions within Jodel as imbalanced. This will be discussed and considered in Section 6.2.

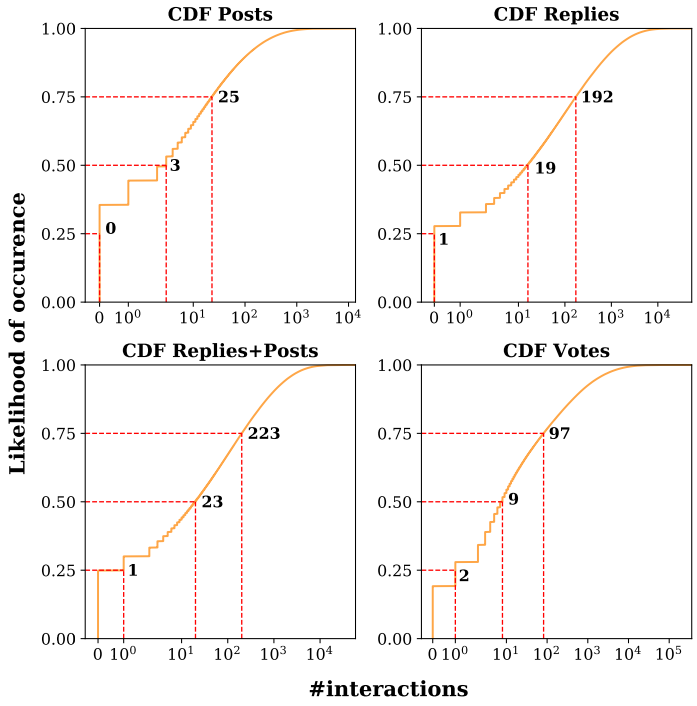

Figure 4: The CDF of posts, replies, posts+replies, and votes with given 25%, 50%, and 75% quantiles, which shows us, that users are more likely to interact by votes than by actively creating content.

1.3 Definition of Churn

What we want to observe and predict within this dataset, is the churn of users within a specific time. A churner in the topic of social networks is a user who joined the network and leaves it forever for some reason after a specific time. Therefore, we will look at different characteristics within the observed users to classify them as churners. As a criterion, we will consider the active time of a user on the platform.

We have therefore observed the CDF of the active minutes. As shown in Fig. 5, 25% of the users did not stay longer than 7,359 minutes or about 5 days. We can also see that 50% of them did not stay longer than about 38 days and thus just a bit longer than one month. We also learned, that just 25% of the users did stay longer than 102 days ($\approx$ 3 months). This leads us to the conclusion, that many users stay for a really short time and churn from the platform.

Because of users that could not have been active for a longer time since registration and the, therefore, bias, we have observed the ratio of time they could stay active to the real active time in Fig. 6. We can see that 50% of the users (206,095) stayed a maximum of 66.4% of the time they could. We can also see that there are just 25% of the users (103,047) who used more than 96.5% of the time. This shows us that this distribution has about 25% of users who used 10% of the time in a maximum of their possible time, 25% of the users who used nearly every minute they could and about 50% of them used it in a high time variety. This shows us that there is a high variety in using the possible time, but also many users who are using just a fracture of it.

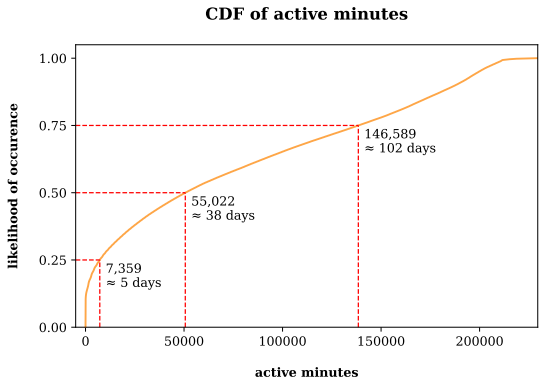

Figure 5: The CDF of the active minutes with the given quantiles, which shows that 25% of the users did not stay longer than about 5 days, 50% not longer than about 38 days, and 75% not longer than 102 days. This also means that many users stay for a very short time and therefore, churning soon after joining the platform.

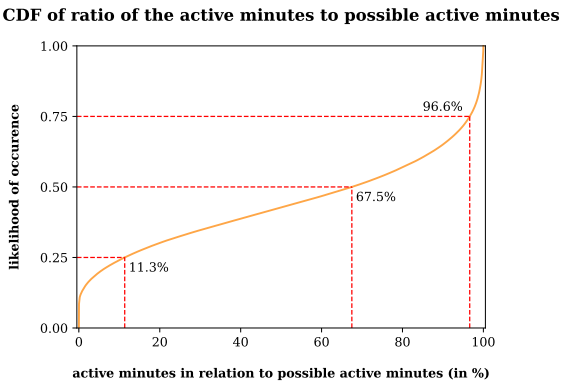

Figure 6: The active minutes in relation to the possible active minutes of the users and the given quantiles. This shows that 25% of the users used Jodel a maximum of 10.8% of the possible time, 50% a maximum of 66%, and 75% of them a maximum of 95%.

To define our churners, the definition was split into different classes to observe later in which class we can achieve the highest prediction accuracy. After the observation we just made, we decided to classify our churners as shown in Fig. 7 and Tab. 6. We decided to take a look at users who churned after one day to see how good the predictive models can detect this subset of just 15.5% of users. Because of the quantiles of the active minutes, we also decided to observe the first week and month, and the first three months. To get a more detailed look we added also the class for users who stayed just two weeks and finally, a class for all other users who stayed longer than three months.

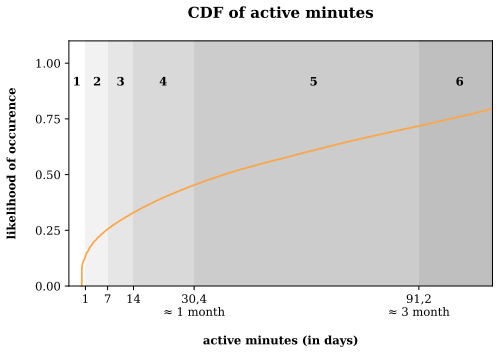

Figure 7: The CDF of the active minutes in days separated into six classes, which will be observed and discussed.

class	active minutes	# users	ratio
1	$\leq$ 1 day	135,158	13.3%
2	1 day $<$ ... $\leq$ 7 days	122,992	12.1%
3	7 days $<$ ... $\leq$ 14 days	75,101	7.4%
4	14 days $<$ ... $\leq$ 1 month	125,449	12.4%
5	1 month $<$ ... $\leq$ 3 month	268,543	26.5%
6	$>$ 3 months	185,376	28.2%

Table 6: The definition of the six shown classes in Fig. 7, where we have given their borders, as well as the number of users within this subset and their ratio to all users.

1.4 Ethical and Social Aspects

The importance and sensitivity of data privacy and protection within a social network was the topic of many discussions since social networks were established, but increased constantly within the last years. Since social networks have been used, users trust the companies to keep their private information safe and away from the public. In many cases, the protection of data is discussed much. At least since 2018, where a huge amount of personal data of Facebook, one of

the biggest social networks, was leaked, data protection is extremely discussed and came into special focus.

The used data in this thesis was provided by Jodel and includes meta-data like registering dates, number of votes for a user, or the number of created posts, that belong to a user ID that cannot be tracked back to personal information. Jodel itself does not store any personal information about a user. Registering needs just some technical information about the user device and no phone number, mail address nor other personal details. Therefore, Jodel cannot trace back a Jodel account [8]. We did not have access to any other data that let us conclude with a specific person.

1.5 Research Question

Within this bachelor thesis the main research question will be:

> *Can we build predictive models for single communities that achieve high prediction accuracy and can we build a single model that can predict user churn of different communities in the Kingdom of Saudi Arabia with high prediction accuracy?*

Based on these models, we will ask furthermore:

> *How do the different models perform on other communities, and can we detect similarities to possibly define different behaviours of models, hence communities?*

1.6 Contributions

The contributions of this thesis will be:

- creating and selecting of different features and feature subsets
- expansion and improvement of an existing ML framework for building a pipeline for data preparation, predictive models, and evaluation
- evaluating the performance of the models based on different time-window feature subsets
- using the best models for predicting user churn in other communities to:
  a) find a model with general strong performance in many communities
  b) find similarities between the different models (using feature importance)
- using the most important features to detect trends in empiricism

1.7 Outline

We have organized this bachelor thesis as follows: In Section 2 we will explain the basics of ML to help understand the presented work. After that, we will have a look at the existing work for predicting churn in different contexts in Section 3. We will then have a look at the used features and their engineering in Section 4, followed by a presentation of our ML pipeline and its implementation in Section 5. As the main part, we will then present our evaluation of the models in Section 6. Before summarizing the results and concluding in Section 9, we will have a look at improvements of the models that could enhance our results, as well as future tasks, that can be built on our results in Section 7 and 8.

2 Background

In the following, we explain the processes that will be used within this thesis. For this, we will start by feature engineering in Section 2.1, followed by the tools that were used for pre-processing the data in Section 2.2 for scaling and balancing the data. After this, we will explain the different used Machine Learning (ML) methods, including, e.g., the Decision Tree (DT), but also some Ensemble methods, like the Random Forest (RF) in Section 2.3. Because of the different parameters for each method, we will then explain the hyperparameter tuning in Section 2.4 and how we avoid overfitting of our models by k-fold cross-validation in Section 2.5. For evaluating and comparing the models, we will end this section by defining the used metrics in Section 2.6, statistical tests in Section 2.7, and correlation coefficients in Section 2.8.

2.1 Feature Engineering

When training a model for a specific task, it needs to be fed with data. This data in ML is represented by features. These features are representing different aspects of a user, from which the models learn and create rules for predicting the correct result. To achieve the best possible model, we need to feed it with ‘‘enough relevant features and not too many irrelevant ones" [9]. The selection of the best number and overall best features is called *feature engineering*. This includes the selection of the most useful features, but also extracting and combining to new more useful, as well as generating new features [9].

For selecting the most useful features, we will see in Section 3, a range of methods that can be used to calculate the importance of features within the prediction process and giving us a set of features that could work optimally.

Within this section, we will present the three methods, that we will use within this thesis. These methods will calculate the importance of the features, that will be used to select the most important features, as well as for detecting similarities of different models, hence communities by using correlation coefficients. Because of their high importance, they could also describe the characteristics of the models, hence communities, which will need to be confirmed by feeding them back into empiricism.

2.1.1 Feature Importance from Decision Trees

The **Decision Tree** can be used when running a DT algorithm (cf. Sec. 2.3.1). It is the visualisation of the created tree. This lets us interpret and understand the decision-making of the created DT. A visualisation of a DT can be seen in Fig. 8. This enables the reading of the

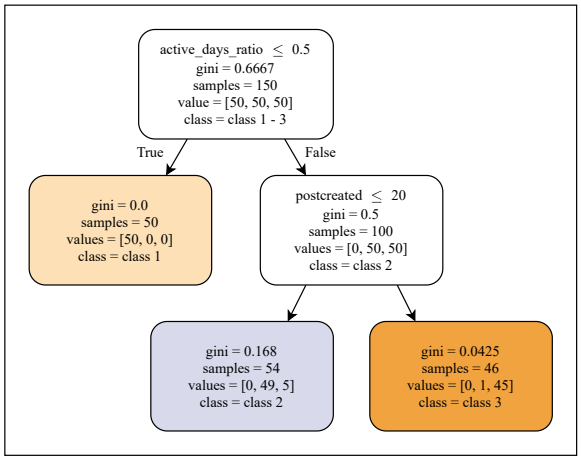

Figure 8: An example of a DT which displays a Boolean function for the decision, including the node attributes: *feature name* and boundary, *gini* for impurity, *values* are the training instances of each class at this node, *class* for the predicted class at this node [9].

used features, as well as the Gini impurity, instances of each class at this node or the predicted class at this node [9].

Within tree-based algorithms like the RF (cf. Sec. 2.3.10.3) or the DT (cf. Sec. 2.3.1), the **Feature Importance** can be used. It is based on the *Gini Importance*, which is a measure that bases on the impurity reduction of splits within a DT. At each node, the *Gini Impurity* is measured. The used feature at a split that has a large decrease of impurity, will be categorized as important. The importance of a feature is then calculated as the sum of all impurity measures overall nodes within the tree or forest [10].

2.1.2 ReliefF

The *ReliefF* algorithm was proposed in 1997 by Kononenko et al. in [11] and is a member of the Relief family which was introduced in 1992 by Kira and Rendell in [12]. The Relief algorithm returns the importance of the features, by observing the difference between the values of two samples that are near to each other. It, therefore, searches first for the so-called *nearest hit*, a sample of the same class, and *nearest miss*, a sample from the other class. It then calculates the difference between the values of both instances and gives them weights, to finally, calculate the total distance by cumulating the differences over all attributes [11].

The **ReliefF** is an extension of the Relief, ''that improves the original algorithm by estimating probabilities more reliably and extends it to deal with incomplete and multi-class data sets'' [11].

2.1.3 RFECV

The RFECV is a combination of the *Recursive Feature Elimination (RFE)* and *cross-validation (CV)*.

The RFE is used to iteratively exclude features and was firstly proposed in [13], where for each iteration the feature importance is calculated and the worst feature excluded until a 'perfect' number of features with maximum importance is reached. When excluding more than one feature at a time it can be sub-optimal and possibly remove features that could have more effect. Therefore, the RFE iteratively removes one feature after another until a given subset size is reached. It trains a given classifier and calculates the ranking criterion and ''removes the feature with the smallest ranking criterion'' [13]. The final subset is then an optimal subset of relevant features, but not all features within this subset must be individually most relevant [13].

In *SciKit Learn* the RFE is combined with cross-validation, which not returns a subset of a given size, but a subset with an optimal number of features by looping until an optimal subset was found [14].

2.1.4 Input Variance

The variance ($\tilde{s}^2$) of a dataset is a value, that represents the scattering of samples around their mean. It is the square of the standard deviation and is, therefore, small when the values are laying close to the mean. If the values laying far away and therefore, strongly scattered, the variance is big [15]. According to [15], it is defined as follows:

$$\tilde{s}^2 = \frac{1}{n}[(x_1 - \bar{x})^2 + ... + (x_n - \bar{x})^2] = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{1}$$

Because of more differences when having a higher variance for a feature, one can assume that the samples are strongly differing for this feature and that it gains more information because of a wider value range, which could help our predictor by classifying the samples. That is why detecting features with the highest variance in ML, could help to improve the prediction result.

2.2 Data Pre-Processing

The task of data pre-processing can be a very important part, to achieve strong results for ML. In the following, we will show the used tools for scaling and balancing the data.

2.2.1 Scaling

When using numerical input features with a very different scale, many ML algorithms do not perform very well. Therefore, a transformation of the data is often used [9]. In the following, we will present the two main used scalers, the Standard Scaler (cf. Sec. 2.2.1.1) for standardisation and the Min-Max-Scaler (cf. Sec. 2.2.1.2) for normalization from SciKit-Learn.

2.2.1.1 Standard Scaler

The Standard Scaler is used to standardise the values of a feature. This means that the standardised values have a zero mean. To achieve this, the Standard Scaler first ''subtracts the mean value [...], and then it divides by the variance so that the resulting distribution has unit variance'' [9].

2.2.1.2 Min-Max-Scaler

The Min-Max-Scaler is used for normalizing the data, to rescale them to range between a given interval [9]. According to [16] this can be achieved by calculating for each value x of the feature X:

$$\text{Min-Max-Scaler}(x) = \frac{x - min(X)}{max(X) - min(X)} \tag{2}$$

The Min-Max-Scaler from SciKit-Learn has the *feature_range* parameter which can be used for defining the range of the feature values [9].

2.2.2 Balancing

Especially in churn prediction, a special characteristic of the data is the class imbalance. This means that we have differences in data quantities between the different classes, which means classes with just a few representatives and classes with a great quantity of representatives. This could follow to ML algorithms that may not get enough information about the smaller class and result in potentially lower accuracy of the prediction. To avoid this, several sampling methods to handle imbalance can be used. Besides the upcoming three presented methods, there exist a variety of different variants. In the following two sections we will explain two used non-heuristic methods for under- and oversampling and the used heuristic method for oversampling.

2.2.2.1 Random Over- and Undersampler

The **Random Undersampler (RanUS)** is a non-heuristic method from *Imbalanced Learn* [17], that randomly eliminates examples of the majority class. This can be done *fast and easy* but also increases the possibility of *discarding useful data*, because of the non-heuristic decision. These could gain important information for the classifier and could therefore decrease the prediction accuracy [18].

The **Random Oversampler (RanOS)** is also a non-heuristic method from *Imbalanced Learn* [19], that randomly replicates examples of the majority class. This can also be done *fast and easy* but increases the possibility of *overfitting* the data, because of the non-heuristic decision. The exact replicates of the minority class can result in constructed rules that cover a replicated example, which makes the prediction accurate [18].

2.2.2.2 Synthetic Minority Oversampling Technique (SMOTE)

The *Synthetic Minority Oversampling Technique (SMOTE)*, is an oversampling approach that was introduced in 2002 by Chawla et al. in [20]. They compensate the oversampling with replacement, like in the Random Oversampler, by creating '*synthetic*' examples by operating in 'feature space'. ''The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors'' [20]. From the k nearest neighbours, randomly the required neighbours, for the amount of over-sampling, are chosen [20].

The synthetic samples are generated by the difference of the feature vector of a sample and its nearest neighbor, which gets multiplied by a random number between zero and one. The result is added to the vector and results in a point between two specific features. This generates ''larger and less specific decision regions [...], rather than smaller and more specific regions'' [20]. This leads to the learning of more general regions and therefore, a better generalisation [20].

2.3 Learning Methods

Within the following section, we will explain the different ML methods that were used within this thesis. We will explain common methods, as well as some combinations of them, called ensemble methods.

2.3.1 Decision Trees

Decision Trees (DT's) are ML algorithms, introduced by Leo Breiman in [21], that can be used for classification and regression tasks.

DT's are building a tree that represents a decision function. The algorithm traverses the tree from root to leaf, by using a simple decision rule, as shown in Fig. 8. This is repeated until a leaf is reached and the algorithm can specify the result [9].

DT's acting like a *white box*, because of the simple visualisation and are, therefore, simple to understand and interpret. Any given result can be observed and can easily be explained by boolean logic.

Creating a DT can also result in over-complex trees, which leads to *overfitting*, e.g., because of the not well generalised data. To avoid this, we can use mechanisms like setting the maximum depth of the tree, pruning, or setting the minimum number of samples at a leaf node. Another problem can be the dominance of classes, which leads to a biased tree. That can be avoided by balancing the dataset [22].

2.3.2 K-Nearest-Neighbors

The *K-Nearest Neighbours (KNN)* is an ML algorithm for regression and classification tasks. It classifies a sample based on the already classified samples. For this, it uses the k nearest neighbours by a distance metric and labels the new sample from these, as shown in Fig. 9. The distance can be any metric measure. These k neighbours can be a defined constant or ''vary based on the local density of points'' [23].

For classification, it uses uniform weights, resulting in a simple majority vote of the nearest neighbours of a single point. This can also be changed to distance weights, which assigns weights proportional to the inverse of the distance from the query point [23].

For the regression, the algorithm labels a point by computing the mean of the labels of the nearest neighbours. The weights of each point are uniform and therefore, equal for each point. This can also be changed to distance weights like described for classification [23].

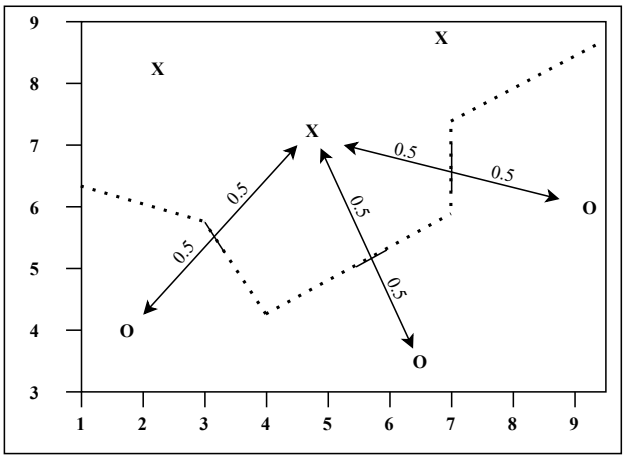

Figure 9: 2D example of the result of classifying samples with class border (dotted line) as Voronoi diagram [24].

2.3.3 Support Vector Machine

The *Support Vector Machine (SVM)*, is a ML method that can perform binary and multi-class classifications, as well as regression. They were proposed in 1992 by Boser et al. [25]. The SVM basically is a binary classifier, that is why we will explain first the basic functionality. We will then explain how to extend to handle multi-class classification and regression.

The SVM "constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space" [26]. The best hyper-plane can be achieved by choosing the largest distance to the data points of classes (*one-versus-one* approach). "The determination of the model parameters corresponds to a convex optimization problem" [27], where the target is to detect the maximum of a margin between decision boundary and data points. "The location of this boundary is determined by a subset of the data points, known as support vectors" [27], as red dotted bordered points, shown in Fig. 10. The larger the distance is, the lower is the generalisation error of the classifier.

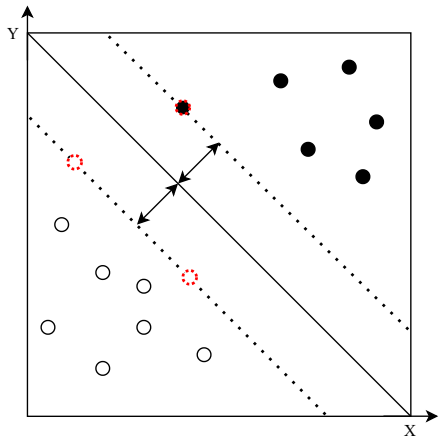

Figure 10: Classification of a SVM. Displays the decision boundary and data points and especially the support vectors (data points with red dotted border) [28].

To handle the multi-class classification has been proposed a *one-versus-the-rest* approach by a combination of multiple binary SVM's. For k classes, we construct k separate SVM's, where the k^{th} model "is trained using the data from class C_k as the positive examples and the data from the remaining $k-1$ classes as negative examples" [27].

The Support Vector Regression is an extended Support Vector Classification and was proposed in 1997 by [29]. It depends on a subset of the training data and constructs a function, to get the training points inside a '*tube*' of a given radius [29].

2.3.4 Gradient Descent

The *Gradient Descent (GD)* is an ML method, that iteratively optimizes the parameters, to minimize a cost function. To achieve this, it ‘‘measures the local gradient of the error function [...], and it goes in the direction of [the] descending gradient’’ [9]. When reaching a gradient of zero a local minimum is found.

The algorithm starts with a so-called *random initialization*, which fills the parameter vector with random values. In each iteration, it will then gradually improve and attempt to decrease the cost function until a minimum is reached [9], as shown in Fig. 11.

The size of the steps, also known as the *learning rate* hyperparameter, is very important. ‘‘If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time’’ [9]. If the learning rate is too big, the algorithm could jump over the minimum and diverge, with larger and larger values and would finally fail [9].

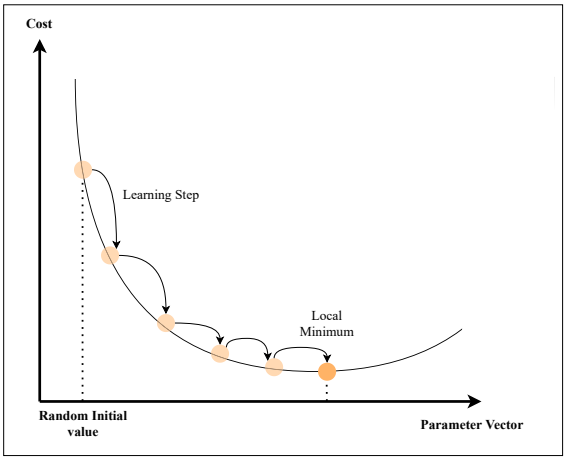

Figure 11: The proceeding of a GD, starting with random values and decreasing the cost function until it reaches a minimum [9].

2.3.5 Stochastic Gradient Descent

The *Stochastic Gradient Descent (SGD)* uses a random subset of the training data at every step and calculates the gradient as seen for the GD before, but based on this subset. This makes the algorithm much faster and makes it possible to run on huge training sets in comparison to the GD. The cost function is decreasing on average but may jump up and down until a local minimum is reached. A disadvantage is that the SGD could continue jumping around the minimum and will end with good final parameters, but not optimal [9], as shown in Fig. 12.

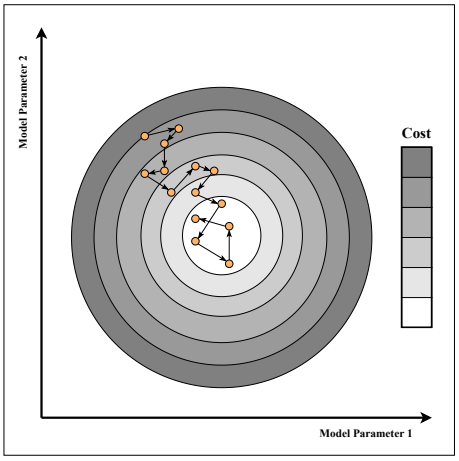

Figure 12: The convergence of the SGD to the cost minimum and jumping around [9].

2.3.6 Linear Regression

The *Linear Regression (LR)* is an ML method for classification and regression tasks. It is a simple algorithm that predicts by optimizing a weighted sum of input features and a constant (bias term), as shown in Equation 3 according to [9], which directly represents the optimization target.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n \tag{3}$$

This includes the predicted value y, the bias term β_0 and n features, displayed by x_i as the i^{th} feature value and β_j the j^{th} model parameter or rather feature weight [9].

2.3.7 Logistic Regression

"Logistic Regression is a statistical method of predictive analysis" [30] and is primary used for binary classification and bases on the sigmoid or logit function (also called *Logit Regression*), with values between 0 and 1 [31]. The *Logistic Regression (LogR)* is often used for estimating the probability, that a data point belongs to a specific class. If it reaches a probability of greater than 50% it predicts a certain class [9].

The LogR calculates like the LR, but returns the logistic of the result. This is also called logit, a sigmoid function σ, that returns a value between 0 and 1 [9], as shown in Equation 4 according to [9].

$$p = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n) \tag{4}$$

2.3.8 Naïve Bayes

The *Naïve Bayes (NB)* is a classification method basing on the Bayes' theorem. By using the associated variables of an event, the algorithm calculates the probability that it will happen [32]. "It adopts the idea of complete variables independence, as the presence/absence of one feature is unrelated to the presence/absence of any other feature" [32]. As a result, the NB returns a probability score and the predicted class [32].

2.3.9 Multi-Layer Perceptron

The *Multi-Layer Perceptron (MLP)* is an Artificial Neural Network (ANN). The ANN's are inspired by the brain's architecture and were introduced by McCulloch in 1943 in [33]. It consists of interconnected artificial neurons. They get two or more binary inputs and one binary output as shown in Fig. 13. Because of the mappings of the brain in researches, the neurons are often separated into consecutive *layers*, as shown in Fig. 14.

An extended, but simple version of ANN's, the *Perceptron* was proposed by Rosenblatt in 1958 in [34]. It bases on the so-called *linear threshold unit (LTU)*. The input and output became numbers and the input is connected with weights. "The LTU computes a weighted sum [Σ] of its inputs [...], then applies a step function [Z] to that sum and outputs the result" [9], as shown

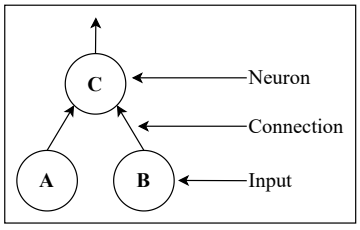

Figure 13: An example of an ANN performing a simple logic [9].

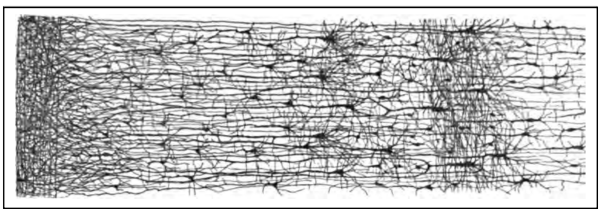

Figure 14: "Multiple layers in a biological neural network (human cortex)" [9].

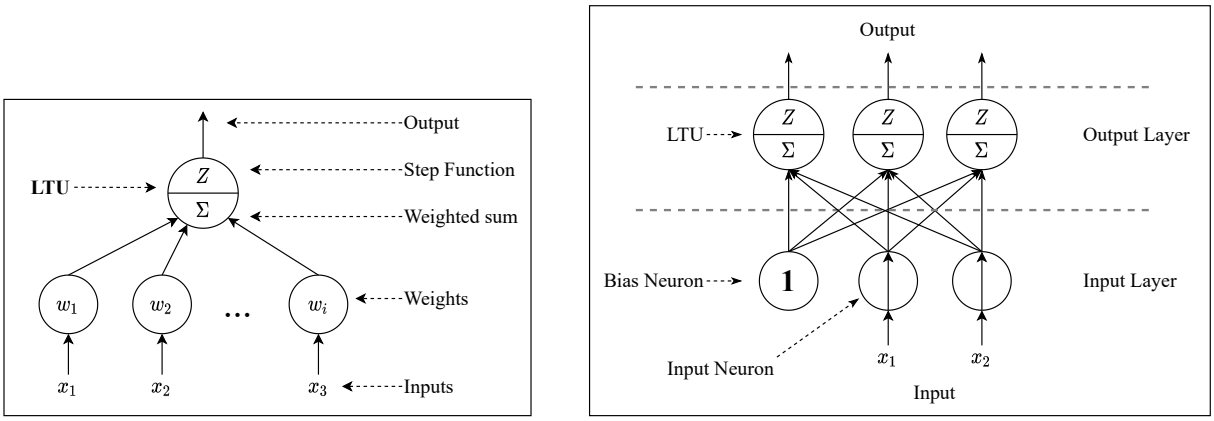

Figure 15: The concept of a Perceptron with LTU (left) and the Perceptron with multiple LTU's and input layer with bias neuron (right) [9].

on the left in Fig. 15. A Perceptron consists of a single layer of LTU's, where each neuron is connected with all inputs. These input connections are often represented by a so-called *input layer*, that includes a bias neuron, which always outputs 1, and neurons that simply output their inputs [9], as shown on the right in Fig. 15.

When training the perceptron, the weights between neurons get increased when they have the same output. This was derived from the brain, where connections between neurons get stronger when they get triggered [9]. But simple perceptrons are limited in, e.g., learning complex patterns, because of the linear output of a neuron or ''the fact that they are incapable of solving some trivial problems'' [9]. To eliminate these limitations the *MLP* was presented, which stacks multiple perceptrons [9], as shown in Fig. 16. It includes the already explained *input layer*, one or more *hidden layers*, which include layers of LTU's and finally an *output layer* with LTU's. Except for the output layer, each layer has a bias neuron and gets ''fully connected to the next layer'' [9], as shown in Fig. 16. The MLP is called a *deep neural network* when using two or more hidden layers [9].

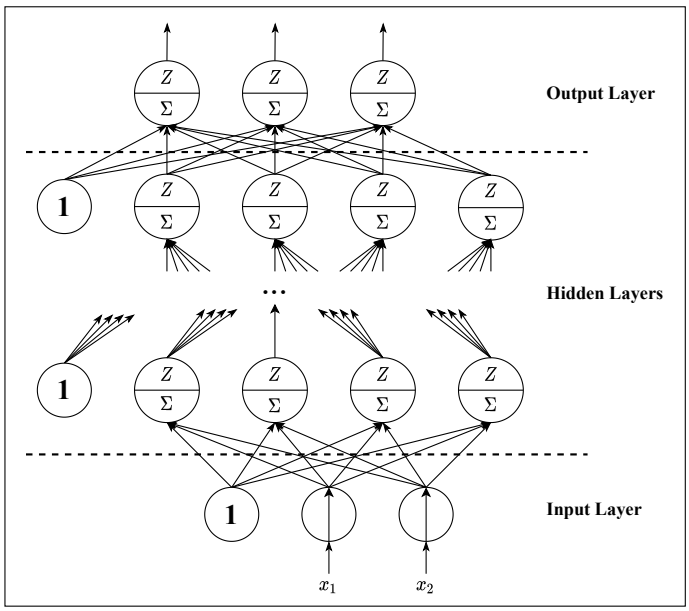

Figure 16: The MLP with an input layer, including two input variables and a bias neuron, multiple hidden layers, including a bias neuron and four LTU's, and the output layer, including three LTU's [9].

The MLP gets trained using *backpropagation*, which was proposed in 1986 by D.E. Rumelhart et al. [35]. This means that each neuron gets trained in each consecutive layer and the network's output error gets measured. ''It computes how much each neuron in the last hidden layer contributed to each output neuron's error. It then proceeds to measure how much of these error contributions came from each neuron in the previous hidden layer -- and so on until the algorithm reaches the input layer'' [9].

2.3.10 Ensemble Methods

Ensemble Methods implement the Idea of *the wisdom of the crowd*, which means the aggregated answer of thousands of random people is better than the answer of one expert [36]. In our case, we aggregate the predictions of a group of predictors to get a better prediction as the best single predictor. We will call this group of predictors an Ensemble Method. In the following sections, we will describe a selection of well-known methods to demonstrate a range of ensemble techniques like bagging or boosting [9].

2.3.10.1 Voting Classifier

A *Voting Classifier* uses multiple classifiers and their prediction result, as shown in Fig. 17. The Voting Classifier can be differentiated into a *hard* and *soft voting* classifier.

A *hard voting* classifier aggregates ''the predictions of each classifier and predicts the class that gets the most votes'' and can, therefore, also be called a *majority vote classifier* [9].

The *soft voting* classifier is an ensemble of classifiers that can predict class probabilities. Each class gets an average probability over all the individual classifiers and predicts the class with the highest probability [9].

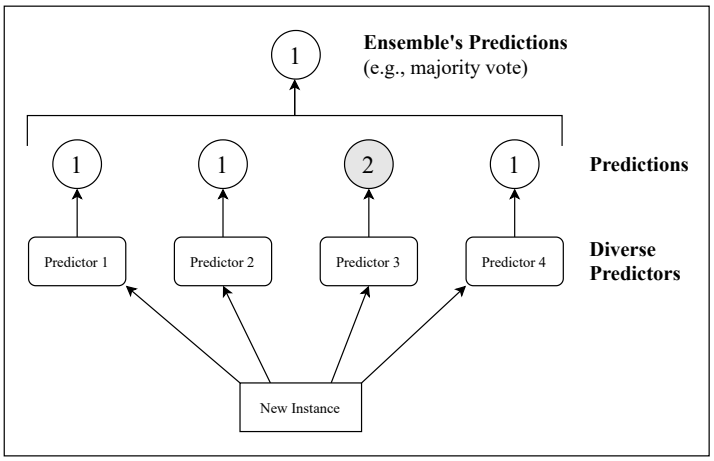

Figure 17: The Voting Classifier uses multiple predictors to compute an ensemble prediction from the single predictions by using, e.g., majority vote [9].

2.3.10.2 Bagging Classifier

A *Bagging Classifier* generates multiple versions of a given predictor. These predictors get trained on multiple random subsets with bootstrap replicates and aggregate the prediction result. For numerical outcomes, the prediction results get aggregated by the average for numerical data or for the classification tasks by a plurality vote, as shown in Fig. 18. Breimann called this procedure "bootstrap aggregating" and gave it the acronym bagging [37].

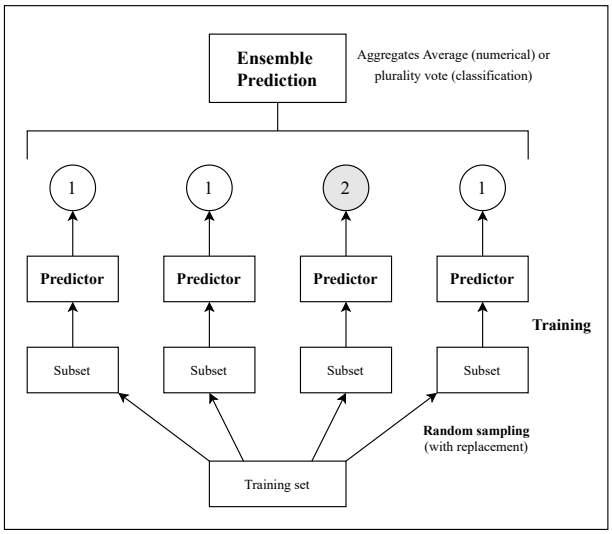

Figure 18: The Bagging Classifier trains a predictor on multiple subsets (with replacement) and aggregates the prediction results for the ensemble prediction. This can be done, e.g., by aggregating the average for numerical data or the plurality vote for classification [9].

2.3.10.3 Random Forest

A *Random Forest (RF)* is a so-called bagging classifier, introduced by Breiman in 2001 [38], that can be used for regression and classification tasks. It combines a group of DT's, a *Decision Tree Ensemble*, and includes all hyperparameters of a DT as well as all of a Bagging Classifier, to control the ensemble [9].

The trees of the RF "are built to explain the different features in data" [28]. When growing the trees, the RF "searches for the best feature among a random subset of features" [9], in contrast to the DT. "This results in a greater tree diversity, which [...] trades a higher bias for a lower variance, [and is] generally yielding an overall better model" [9].

The RF can handle "large data and thousands of input variables" [28], as well as missing and imbalanced data [28]. "Random forest's weaknesses are that when used for regression they cannot predict beyond the range in the training data and that they may over-fit data sets that are particularly noisy" [28].

2.3.10.4 Boosting

Boosting is used by combining multiple weak learners into a single strong learner. The predictors get trained sequentially. Each predictor tries to correct its predecessor and results in a rule-of-thumb for the prediction. Finally, the predictors get combined for a final prediction rule [39].

2.3.10.5 AdaBoost

The *Adaptive Boosting* or AdaBoost is a boosting method. It proceeds iteratively and combines many classifiers. At first, all training samples initialised with the same weight and get classified for the first time. The weights of the misclassified data points get then increased or **boosted**. The next classifiers repeat this procedure by using the weighted samples, to focus more on the difficult cases [39], as shown in Fig. 19. When each predictor got trained, the predictions of all predictors are aggregated and each predictor gets weighted. The class with the majority of weighted votes will then be defined as the final prediction [9].

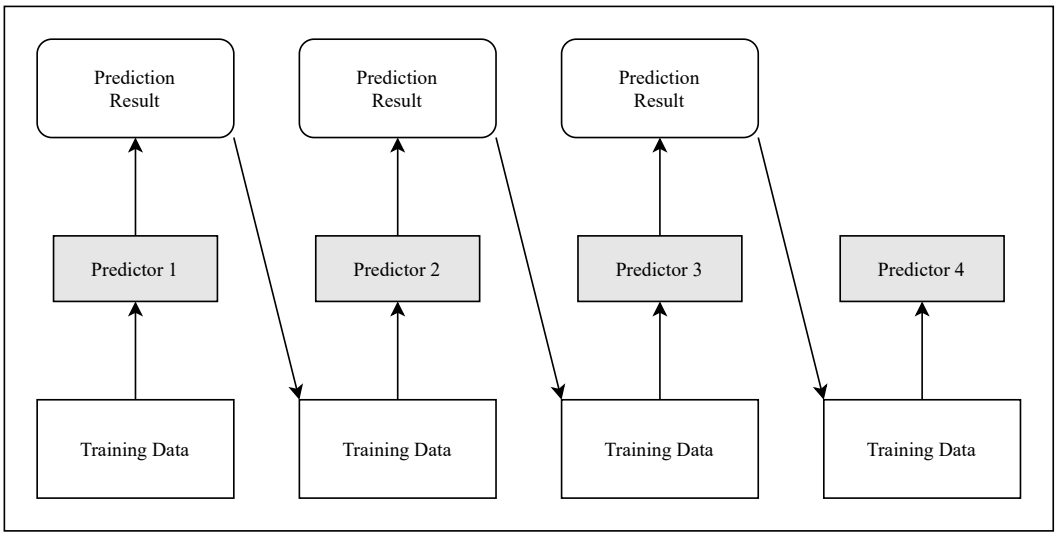

Figure 19: The AdaBoost using multiple classifiers and increases the weights of misclassified samples to focus more on difficult cases within the next classifier [9].

2.4 Hyperparametertuning

When selecting one or more methods to use for the problem, various hyperparameters can be used, which can be adjusted to optimize the prediction result. Because of a wide range of possible parameters, it is needed that these parameters get tested to find an optimal combination, which is called **Hyperparametertuning**. Because of the enormous scope, it would be very tedious work to test each parameter combination manually. That is why we are using a so-called *Grid Search*, which receives the parameters and values as input to test and evaluates the possible combinations. The used Grid Search in this thesis is combined with k-fold cross-validation, from SciKit-Learn, called *GridSearchCV* [40], to avoid overfitting of our models [9].

2.5 k-fold Cross-Validation

For an evaluation of a model, we would split our model into a training and test set. We will then train the model and predict the samples of the test set. This could work very well but could skew the results because of the missing information of samples from the unseen test set and could, therefore, lead to **overfitting**. This means, that the model performs well on the training set, but does not generalize enough to perform well on new data. To avoid this we can use **k-fold cross-validation (CV)**. This splits the data into k distinct subsets, the so-called *folds*. It will then train the model k times, by evaluating each time on a different fold and training on the other $k-1$ folds. This results in k evaluation scores. These can then be evaluated by aggregation, e.g., calculating the standard deviation to measure how precise the model is [9].

This can avoid *overfitting*, because of the training on all samples in subsets and testing on the whole data in subsets. This creates a more generalisation and therefore, a more diverse observation and combination of different samples.

2.6 Metrics

Within the following section, we will first explain the used metrics for regression tasks in Section 2.6.1 and classification tasks in Section 2.6.2. These metrics are used to compare the different algorithms and models and get a deeper look at their performance.

2.6.1 Regression Metrics

For the regression task, we will use three different metrics, for evaluating and comparing the models. Within this thesis, we will use the Mean Squared Error (MSE), Mean Absolute Error (MAE), and the R^2 Score.

2.6.1.1 Mean Squared Error

The most common performance metric for regression tasks is the *Root Mean Squared Error (RMSE)*. The task is to minimize this value. ''Because the value that minimizes a function also minimizes its square root'' [9], and it is simpler to minimizes it and we achieve the same result, in practice the *Mean Squared Error (MSE)* is used [9].

To reflect the consistency of a sample, both the behaviour of the expected value and the variance of the estimation are included. This is done by using the MSE, which indicates the deviation between the estimation (θ) and the true value (T), which is expected for the estimator and is defined in [15] as follows:

$$MSE = \frac{\sum_{i=1}^{n}(T_i - \theta_i)^2}{n} \tag{5}$$

2.6.1.2 Mean Absolute Error

The *Mean Absolute Error (MAE)* calculates the mean of the prediction errors of the samples within the test set, which is calculated by ''the difference between the true value (T) and the predicted value (θ) for the instance'' [41]. According to [41], it is defined as follows:

$$MAE = \frac{\sum_{i=1}^{n}|T_i - \theta_i|}{n} \tag{6}$$

2.6.1.3 R^2 Score

To determine the quality of model adaption, we are using the *coefficient of determination*, also called the R^2 *Score*. It bases on the decomposition of variance and just indicates the proportion of the total dispersion of the samples of set X that is explained by the regression from one set X depending on another Y [15]. It, therefore, measures the correlation between X (the true values) and Y (the predicted values). The R^2 Score can be equal to or less than one, where one symbolises the perfect adaption of the model [42]. According to [15] it is defined as:

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{7}$$

where $\bar{y}$ is the regression line, $\hat{y}$ the predicted values and y the true values.

2.6.2 Classification Metrics

To evaluate the performance of a classifier, we have used the confusion matrix, from which we can derive the precision, recall, F1 Score, and accuracy. Finally, we will explain the receiver operating characteristic (ROC) and area under the curve (AUC), which also is connected to the other metrics.

2.6.2.1 Confusion Matrix

The *confusion matrix* is an $i \times i$ matrix, where i is the number of classes. It includes information on how often a classifier predicted a sample of class A as class B. The confusion matrix can be visualized as shown in Fig. 20 and Fig. 21. The rows of the matrix represent the actual class and the columns the predicted class. With the matrix we get 4 different classification cases:

True Positive (TP): samples that class is positive and got predicted correct

True Negative (TN): samples that class is negative and got predicted correct

False Positive (FP): samples that class is positive and got predicted negative

False Negative (FN): samples that class is negative and got predicted positive

The perfect classifier would achieve only non-zero values along the diagonal and would, therefore, only have true positives and true negatives [9].

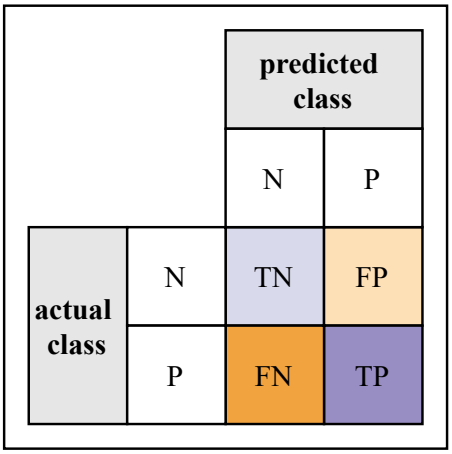

Figure 20: The confusion matrix of a binary classification including the number of true and false predicted samples for each class. From this, the precision and recall can be derived [43].

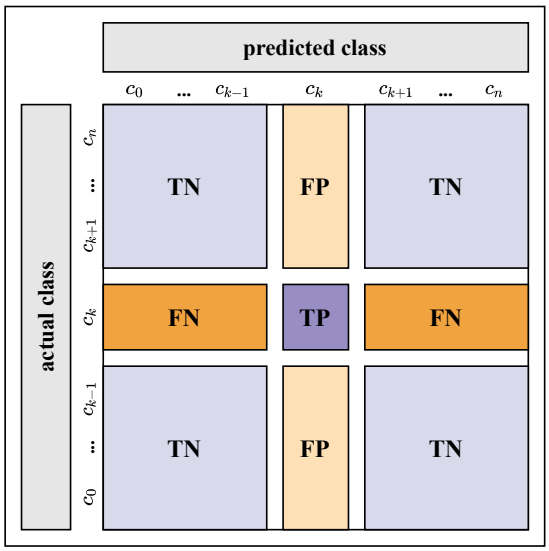

Figure 21: The confusion matrix of a multi-label classification including the number of true and false predicted samples for each class. From this, we can derive the following measures [44].

2.6.2.2 Precision, Recall, F1 Score and Accuracy

The following metrics can be derived from the confusion matrix. The ***precision*** represents the accuracy of positive predictions [9]. In the case of a binary classification a prediction if a user will churn or not. The precision could be explained as the ratio of users that were correctly predicted as churner to the number of all users that were predicted as churner, or how many predicted churners were predicted correctly? According to [24] the precision is defined as:

$$precision = \frac{TP}{TP + FP} \tag{8}$$

The ***recall*** represents the *True Positive Rate (TPR)*, also called *sensitivity*, which means the ratio of instances that got correctly predicted by the classifier to the number of instances that should be correctly predicted [9]. As for the precision, the recall for the binary classification could be explained as the ratio of correctly predicted churner to the number of all churners, or how many churners were predicted as such? According to [24], the recall is defined as:

$$recall = \frac{TP}{TP + FN} \tag{9}$$

When naming sensitivity, an often-used metric is *specificity*, which is also called the *true negative rate* and represents the probability, that non-churners were predicted as such [24]. According to [24], it is defined as follows:

$$specificity = \frac{TN}{FP + TN} \tag{10}$$

Often precision and recall are combined to get the ***F1 score*** as a single metric. It is the harmonic mean of both. This is more weighted on lower values, which results in a high F1 score only if recall and prediction are high [9]. According to [24], it is defined as follows:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{11}$$

The ***accuracy*** represents the ratio of correctly predicted samples and is also called the *true classification rate*. According to [24], it is defined as follows:

$$accuracy = \frac{true\ classifications}{number\ of\ classifications} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

2.6.2.3 ROC/AUC

The *receiver operating characteristic (ROC)* can be used for a visual evaluation in binary classification. When plotting the ROC curve, we are plotting the TPR against the False Positive Rate (FPR). We could also say plotting of *sensitivity* against $1 - specificity$ [9].

When plotting the curve of a purely random classifier the ROC curve would result in a diagonal, as shown by the dotted line in Fig. 22. The farther away the line of the classifier is (towards the top-left corner), the better. In the best case, the curve builds a right angle at the top left corner [9].

To compare different classifiers by a measure, we can calculate the *area under the curve (AUC)* of the ROC curve, so the area under the orange curve in Fig. 22.

The ROC/AUC would be equal to 0.5 when having a purely random classifier and equal to 1 for a perfect classifier [9].

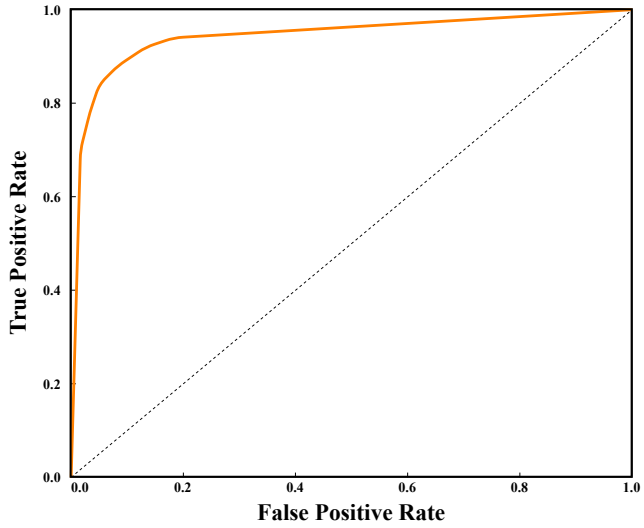

Figure 22: The ROC curve with the result of a purely random classifier (dotted line) and a trained classifier (orange line) [9].

2.7 Statistical Tests

Statistical tests are used to determine if an assumption for a parameter or distribution is applicable or not. This needs a null hypothesis (H_0) and an alternative hypothesis (H_1), where H_1 includes the interested research hypothesis and needs to prevail against H_0 [15].

Based on the observed data, we need to calculate the test statistic and the so-called p-value, which will be used for the decision. We decide to reject H_0 if the statistic is greater than a critical value. This critical value is defined by the known distribution of the data. If the distribution of the data is not known, we can use the p-value which needs to be smaller than a predefined level of significance: $p < \alpha$. If one of both variants is true, we can reject the H_0 [15].

2.7.1 Mann-Whitney U-Test

The Mann-Whitney U-Test is a non-parametric or non-distribution test, where not the parameters of the different distributions are focused, but the characteristics like median or quantiles [15]. It is a two-sample test, that uses values from two populations and calculates the test variable U and a p-value [45]. With the help of this test, it can be stated if there is a significant difference between the mean of the two populations. It only requires ordinal scaled data. The test does not base on the measures but the ranks. A greater test variable U requires the rejection of H_0 [46].

2.7.2 Kruskal-Wallis H-Test

The Kruskal-Wallis H-Test is used for k independent populations and their distribution in the case of the median. When rejecting H_0, we can derive that at least two of the k populations have a statistically significant different distribution. For calculation of the test variable H, the measures of the populations are brought into a common (ascending) ranking [46]. According to [46], the test variable H is calculated as follows:

$$H = \frac{12}{n(n+1)} \sum_{j=1}^{k} n_j(\bar{R}_j - \bar{R})^2 = \frac{12}{n(n+1)} \sum_{j=1}^{k} R_j^2/n_j - 3(n+1) \tag{13}$$

$\bar{R}_j$ ... average rank of population j
R_j ... sum of ranks in population j
$\bar{R}$ ... average rank of all k populations
n_j ... sample size of population j
k ... number of populations

n ... sample size over all populations

The greater H gets, the more the average rank of each population ($\bar{R}_j$) is differing from the overall average ($\bar{R}$). A greater test variable H requires the rejection of H_0 [46].

2.8 Correlations

In a further step, we will need to see how much the results of feature importance methods are correlating to detect possible similarities. For this, we will use the Pearson Correlation and the Spearman's Rank Correlation. When getting a high positive or negative correlation between feature values, it signalises that there *could* exist a causal relationship [47].

2.8.1 Pearson Correlation

The Pearson Correlation is a simple linear correlation [28]. "It determines the extent to which two variables are proportional to each other. Proportional means that the two variables have a linear relationship and this relationship can be represented by a line, called the regression line" [28]. The correlation measures the relationship between the two variables and ranges from -1 to 1, with 1 as total positive linear correlation, -1 as total negative linear correlation, and 0 as no linear correlation [28]. According to [46], the Pearson correlation (r) is calculated as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{14}$$

where n is the sample size, $\bar{x}/\bar{y}$ is the mean of each sample, and x_i/y_i the i^{th} point in the sample.

Example Calculation for Pearson Correlation

For a better understanding, we provide an example for the calculation of the Pearson Correlation. For this, we have given in Tab. 7, the values of four example sets (a, b, c, d) and their mean (μ), as well as the difference between each value in the set and the mean. In Tab. 8, we have given additionally the product of differences between set A and the other three sets and the sum.

		A		B		C		D
i	a_i	$a_i - \mu_a$	b_i	$b_i - \mu_b$	c_i	$c_i - \mu_c$	d_i	$d_i - \mu_d$
1	1	-1.5	5	-1.5	8	1.5	3	0.5
2	2	-0.5	6	-0.5	7	0.5	4	1.5
3	3	0.5	7	0.5	6	-0.5	2	-0.5
4	4	1.5	8	1.5	5	-1.5	1	-1.5
$\sum$	10	-	26	-	26	-	10	-
μ	2.5	-	6.5	-	6.5	-	2.5	-

Table 7: The example sets (a, b, c, d), their sum and mean (μ), as well as the difference between a single point of the sample and the mean.

i	$A_i * B_i$	$A_i * C_i$	$A_i * D_i$
1	2.25	-2.25	-0.75
2	0.25	-0.25	-0.75
3	0.25	-0.25	-0.25
4	2.25	-2.25	-2.25
$\sum$	5	-5	-4

Table 8: The products of A with B, C, and D from Tab. 7 and their sum, where, e.g., $A_i = a_i - \mu_a$.

With the help of the squared values of Tab. 9, we can now calculate the Pearson Correlations as follows:

$$r_{ab} = \frac{5}{\sqrt{5*5}} = \frac{5}{5} = 1 \text{ (completly positive linear relation)}$$

$$r_{ac} = \frac{-5}{\sqrt{5*5}} = \frac{-5}{5} = -1 \text{ (completly negative linear relation)}$$

$$r_{ad} = \frac{-4}{\sqrt{5*5}} = \frac{-4}{5} = -0.8$$

This shows us for A a completely positive linear correlation with B, a completely negative linear correlation with C, and a not completely negative linear correlation with D.

i	$(a_i - \mu_a)^2$	$(b_i - \mu_b)^2$	$(c_i - \mu_c)^2$	$(d_i - \mu_d)^2$
1	2.25	2.25	2.25	0.25
2	0.25	0.25	0.25	2.25
3	0.25	0.25	0.25	0.25
4	2.25	2.25	2.25	2.25
$\sum$	5	5	5	5

Table 9: The squares of the differences of a sample point and sample mean and their sum.

2.8.2 Spearman's Rank Correlation

The Spearman's Rank Correlation is a non-linear correlation, that measures the relationship between two variables, based on the ranks. It ranges from -1 to 1, like the Pearson Correlation, where 1 means a similar rank and -1 a fully opposed rank [42]. According to [46], the Spearman's Rank Correlation (r_s) is calculated as follows:

$$r_s = 1 - \frac{6 * \sum_{i=1}^{n}(x_i - y_i)^2}{n^3 - n} \tag{15}$$

where n is the sample size and x_i/y_i the i^{th} point of the sample.

Example Calculation for Spearman's Rank Correlation

For a better understanding, we provide an example for the calculation of the Spearman's Rank Correlation as well. For this, we have given in Tab. 10, the values of four sets (a, b, c, d) and their rank $(rg(x_i))$. Finally, we have given in Tab. 11, the square of the differences of the sample ranks and their sum.

	A		B		C		D	
i	a_i	$rg(a_i)$	b_i	$rg(b_i)$	c_i	$rg(c_i)$	d_i	$rg(d_i)$
1	100	1	700	1	1300	4	300	3
2	200	2	900	2	1100	3	400	4
3	300	3	1100	3	900	2	100	1
4	400	4	1300	4	700	1	200	2

Table 10: The values of the example sets (a, b, c, d) and their rank $(rg(x_i))$.

i	$(A-B)^2$	$(A-C)^2$	$(A-D)^2$
1	0	9	4
2	0	1	4
3	0	1	4
4	0	9	4
$\sum$	0	20	16

Table 11: The squared difference of the ranks of A and the other three samples, where, e.g., $A = rg(a_i)$, and their sum.

With these values we can now calculate the Spearman's Rank Correlation between A and the other samples as follows:

$$r_{s_{AB}} = 1 - \frac{6*0}{64-4} = 1 - \frac{0}{60} = 1 \text{ (completly positive relation)}$$

$$r_{s_{AC}} = 1 - \frac{6*20}{64-4} = 1 - \frac{120}{60} = -1 \text{ (completely negative relation)}$$

$$r_{s_{AD}} = 1 - \frac{6*16}{64-4} = 1 - \frac{96}{60} = -0.6 \text{ (negative relation)}$$

This shows us for A a completely positive correlation with B, a completely negative correlation with C, and a not completely negative correlation with D.

3 Related Work

Within the last decades, there was published much work that has addressed predictive models and especially the usage of them in the field of churn prediction. The approaches and different tools for various prediction tasks within the following presented work are shown in Tab. 12. We have seen approaches for Question Answering Site (CQA) for predicting churn [48, 49], answer quality [50], user satisfaction [51], or vote score [52]. There is also work for predicting churn in online games [53, 54, 55, 56, 57, 58], telecommunication [59, 32, 60] as well as in financial groups [61] and urban migration [62], and the prediction of news popularity in online discussion sites [63].

In the following, we will first have a look at different Machine Learning (ML) methods used for different prediction tasks in Section 3.1 and will then display the approaches for different churn definitions, observation windows, features and feature selection methods, and observations of balancing in Section 3.2.

3.1 Machine Learning Methods

For investigating different methods, we will distinguish between two different prediction tasks. At first, we will display the work for the churn prediction task in Section 3.1.1 and will then have a look at methods for non-churn predictions in Section 3.1.2.

3.1.1 Churn Prediction

We will start with the main part of churn prediction in CQA's, online games, telecommunication as well as in financial groups, and urban migration.

For the CQA's we have seen investigations on StackOverflow in [49] and Yahoo! Answers in [48]. They have shown many methods like the Random Forest (RF), K-Nearest Neighbours (KNN), Naïve Bayes (NB) in [48] or the Decision Tree (DT), Support Vector Machine (SVM), or Logistic Regression (LogR) in [48, 49]. In both approaches, the tree-based methods like the RF in [48] and the DT in [49] yielded the best prediction results.

Approaches for predicting churn in online games were presented in [53, 54, 55, 56, 57]. Within these works, they used the RF, DT, Linear Regression (LR), NB, SVM, AdaBoost, Neural Networks (NN's) as well as a heuristic and a Survival Ensemble approach. Especially the Survival Ensemble, an ensemble-based learning method with a survival tree as the underlying algorithm, yielded a high accuracy of about 96% in [55]. But also the SVM in [54] with an accuracy of about 79% and the RF in [57] with an accuracy of about 87% yielded good results. An approach for using NN's for churn prediction in online games is shown in [58], where the NN's outperformed the DT, SVM, and LogR.

For the churn prediction in telecommunication companies, we have seen two approaches in [59, 32]. They proposed also a wide range of different methods, like the already named ones, but also some special ones like the Stochastic Gradient Boost (SGB) or an ensemble approach of Rotation Forest, RF, and KNN in [59]. The Rotation Forest is an algorithm that splits the features into k subsets and uses a principal component analysis (PCA) on each. Therefore the new features for the classifier are created by k axis rotations and decide by using a DT. Because of these two parts, the name Rotation Forest was chosen. They have shown that the ensemble approach in [59] and the RF and AdaBoost in [32] yielded the best prediction results. In [60], we have seen an approach of using NN's, that also achieved high accuracy of 91% in churn prediction for telecommunication companies.

In [61, 62], the RF was also presented as the best method for predicting churners within financial groups and urban migrants. In [61], the Multi-Layer Perceptron (MLP) can achieve high accuracy ($\approx 94\%$) when sampling the data by SMOTE (cf. Sec. 2.2.2.2) too, in comparison to the RF with an accuracy of about 96%.

3.1.2 Non-Churn Prediction

We also looked at other non-churn prediction tasks, which include predictions of vote scores, quality of answers, or user satisfaction.

We have seen different approaches for the CQA's StackOverflow, Naver, and Yahoo! Answers in [50, 51, 52]. A simple approach by testing the SVM, RF, NB, AdaBoost, and the C4.5 DT is shown in [51], where the C4.5 yielded the best accuracy of about 77% for predicting user satisfaction. In [52], a Multiple Linear Regression approach for predicting the score of questions was presented, which displays a dependent variable, that gets predicted by knowing several independent variables. By using this method they achieved a high predictive power and showed that their independent variables have much effect on the score of questions. An approach of predicting the quality of answers on Naver was shown in [50]. They used a Retrieval Approach, which uses a maximum entropy approach, that generates statistical models and returns a probability and the kernel density estimation, which is a ''nonparametric density estimation technique that overcomes the shortcomings of histograms''[50], to return the probability that a question gets a good answer as result.

3.2 Dimensions

In this section, we will first present different definitions of churn in Section 3.2.1 and observation windows, which means the time within the used method was trained, in Section 3.2.2. We will then show the different approaches for feature engineering in Section 3.2.3 and finally the impact of different balancing methods in Section 3.2.4.

3.2.1 Churn definitions

As already shown, there are many different tasks for churn prediction. Therefore we have seen different definitions of churners but also work with already defined churners within the given dataset [61, 59, 32, 56, 49].

At first, we want to present approaches for defining churners over time, which means that users are classified as churners if they were not active for a specific time.

For churn prediction in online games, in [54] churners were defined as users who did not return within the second observation week, in [55], users who did not play for 10 consecutive days, and we have also seen a distinguished approach of defining churners as users who did not play for 7 days on the one hand and 14 days on the other hand, in [57].

An approach for predicting the churn of migrants in Shanghai, people who migrate to the city and leaving it after a given time, was presented in [62]. They have seen churners as migrants who were not active within three weeks.

Approaches for defining churners over the number of sessions were proposed within the gaming context in [53]. They differed their task into a classification problem for predicting churner or non-churner, called *hard churn*, and a regression problem for predicting the number of sessions or active days called *soft churn*.

3.2.2 Observation windows

Another aspect many works differ in is the time of observation of the users. This can mean the observation of a period [48, 49, 54, 57, 62] or the time until a specific threshold gets reached [49, 54].

Within the CQA's in [48, 49], we have seen an observation time of the first week after a user created his first answer in [48] and a more fine granular analysis of different time windows (first 7, 15, 30 days) in [49], which showed that the accuracy increased by observing more days and therefore increasing the observation window.

In [54, 57] different observation windows for the gaming context were presented. A distinction between observing the first day, and the first three and seven days, is shown in [54]. This resulted

in the best accuracy when observing the first seven days. Additionally, they have also shown, that the accuracy is increasing when increasing the observation window. We have also seen an approach for differentiation into observing for two and four weeks in [57]. They showed, that the accuracy of the DT and RF enhanced by increasing the window, but decreasing when using SMOTE (cf. Sec. 2.2.2.2), and the accuracy of the SVM decreases when increasing the window. And finally, in the context of urban migration, an approach for observing the first three weeks of a migrant in Shanghai was presented in [62].

The observation of the time until a specific threshold got reached, is another strategy on CQA's. In [49] we have seen a differentiation into the observation of the first 5 and the first 16 to 20 posts (questions and answers). They showed, that the accuracy of the DT is decreasing when observing more posts. Within the gaming context, we found an investigation of the first session in [54], which yielded the weakest accuracy in comparison to the time windows.

3.2.3 Feature Engineering

Feature engineering always is a special part of prediction tasks. Within the following, we will first show different feature subsets within and beyond the churn context in Section 3.2.3.1 and will then have a look at different approaches used for selecting the best feature subsets in Section 3.2.3.2.

3.2.3.1 Feature Subsets

We have seen different kinds of features in different contexts. In the following, we will firstly present different features for churn prediction and secondly for non-churn prediction.

Churn Prediction

Within this section, we will present the different kinds of features for churn prediction. This will include user-based, community-based, as well as temporal, purchase, and topic-specific features.

Within [48, 49] we have seen user features for CQA's. In [49], user features were used that represented number and quality of answers, questions consistency, and gratification, as well as the knowledge level of a user, which presents how useful a user is for the community. A differentiation into question, answer and gratification-related features for a user is shown in [48]. This leads to features that display numbers of created questions and answers, as well as gratification of the questions and answers. They have also shown that the number of posts of a user gains the most information within the prediction process and that users with fewer posts are more likely to churn. We have also seen that the gratification-related features are negatively correlated with churning.

For gaming, activity features in [57], that display the number of logins or number of days of a user, and a similar approach is presented in [55]. They used just game-independent features, which displayed the information about time on the platform, active days or days until first purchase, number of sessions, or the player level. In this case, the amount and the days since the last purchase yielded the most information for the prediction.

We have also seen the usage of user features in the context of telecommunication in [60, 32] as well as in financial groups in [61], or the urban migration task in [62]. These user features are very context-related in all three cases, like the number of service calls in [32] or the income per month in [61].

Another context of features is the community, which represents the activities around a given user. In [49] the community features were used for competitiveness to compare the quality of answers of users in comparison to others. In [57], community features that display the sociability of a user in the context of gaming were presented, by, e.g., attending a guild or communicating to friends. In the special context of predicting churning migrants, we have seen in [62], that the

community features are very important, which follows from the sociability of migrants and their need for relationships or a circle of friends.

A very strong kind of feature is the time-relation which yielded the best results in [49] and [53]. These include features like time gaps between activities and time until a user responds in [49]. On their feature subsets, they performed an extensive analysis. They show that the isolated temporal features yielded a competitive accuracy to all features when varying the observation window in posts and days. In [53] we have seen temporal features like the playtime per session or the average time between sessions. The last one in addition to the number of sessions were most frequently used within the decision tree, and can, therefore, be seen as more important.

Another aspect of predicting churn is transactions and purchases, as presented in [57, 60]. In the context of gaming, we have seen economy-based features in [53], like flags for premium users or the average spending per session, and in [57], that users who purchased are less likely to churn. For the telecommunication context, features like total payment or the total bill of a customer were presented in [60].

In the last part, we want to show context-specific features, which differ in many cases. We have seen demographic features in [48, 60], which presented the age or gender of a customer. In [59], an investigation of features in numerical and nominal format was shown. For financial groups, context-specific features like the monthly income or the number of web actions in [61] and the context-specific features in [62], including information about housing or geographical patterns of migrants, which are very special to the seen features from other work, were presented.

Non-Churn Prediction

As we have already seen, there is also work with non-churning prediction tasks. For this, in [52, 51] were presented user features. Especially in [51], a differentiation into questioner and respondent features was done, as well as features for relations between them.

For the prediction of vote scores in [52] and for user satisfaction in [51], we have also seen content-based features, which display the number of words of a question or the number of specific words, as well as features for the category of a question.

At the CQA Naver, 13 non-textual features were presented in [50] to predict the quality of answers. These features are not categorized and include information about the acceptance ratio or recommendations of users as well as the number of clicks or answers activity level and answer length. They have also shown, that excluding the feature for questions drops their prediction accuracy.

3.2.3.2 Feature Importance and Selection

Based on the created features many different methods for calculating the importance of features or selecting the best ones for a specific method were shown. The Feature Importance of the RF [54, 57, 32] and the DT [53, 61], which we have explained in Section 2.1, was used in much work. We have also seen different coefficients and correlations like the Pearson Correlation in [50] (cf. Sec. 2.8.1), where the feature value was set into the correlation of a manually judged quality score, Spearman's Rank Coefficient in [52] (cf. Sec. 2.8.2), where the factors and scores of questions were set into correlation, Integrated Brier Score in [55], or the LogR Coefficient and standard error value in [54]. Also, the Gini Importance in [62], which is often connected with the RF Feature Importance was presented. Finally, we have also seen some special algorithms like the mRMR algorithm in [59], or the Boruta algorithm in [32], which were not used within this thesis.

3.2.4 Impact of Balancing

For churn prediction tasks we have often seen an imbalance between the many users who are churning from a network or company very fast and the few who are staying like described in

Section 1.2. We have therefore seen some approaches for handling this problem. In [49, 59], handling of imbalance by undersampling the data is presented. Another attempt, as shown in [53], was the balancing by discarding data of the majority.

A deeper look into the impact of balancing was shown in [57] for predicting churn in online games. They have observed the impact of using SMOTE (cf. Sec. 2.2.2.2) for the RF, DT, and SVM. They have shown that the accuracy, precision, and recall, and therefore also the F1 Score are slightly increasing when using SMOTE.

In [61], we have seen a very detailed investigation of SMOTE, over- and undersampling, as well as a combination of both at different levels. They have shown, that in general using the sampling methods, the accuracy and specificity are slightly decreasing, but also the sensitivity and ROC/AUC score (cf. Sec. 2.6) are increasing. Especially the usage of SMOTE largely increased the sensitivity and ROC/AUC. As two highlights we have also seen that using oversampling by 100% or 300% increases the accuracy, sensitivity, and ROC/AUC score slightly in a trade-off for decreasing the specificity.

3.3 Conclusion

Within the related work, different definitions of churn were shown. We have seen this over time in minutes, days, weeks, or months, but also for defining churners over the number of sessions or other interactions.

A wide range of different ML methods within the literature using simple algorithms like the DT or LR were presented, but also some more advanced approaches by using different ensemble methods or NN's. These methods were tested and evaluated to compare them and find the best working models for a specific topic. We have seen that the tree-based methods DT and RF often yielded the best results, but also methods like the KNN, SVM, or MLP as well as NN's, can achieve strong results by using for example over and under sampling methods.

The different approaches for the observation windows showed us, that we can specify our observation window by time or different thresholds for, e.g., sessions. In general, we have seen, that the accuracy is increasing when increasing the observation window.

In the case of features, we have especially seen that temporal features, like the time between interaction, community, and user-based features, were classified as most important. These were used for detecting the behaviour of a user and his community around, as well as different using patterns like playtime or number of sessions. To select these, we have also seen a wide range of different feature selection methods, like the Feature Importance of the RF or the DT.

We can finally say, that different sampling methods for handling imbalance and their impact on accuracy, specificity, and sensitivity, which in general decrease the accuracy more or less, but can therefore increase specificity or sensitivity, were shown.

		task	churn definition	observation window	ML-Methods	Features	Feature Selection	Data preparation	Results
CQA's	[48]	**churn**	-	first week after first answer	DT, RF, SVM, KNN, NB, LogR	user, demograph. Gratification, content	-	-	RF F1=0.755
	[49]	**churn**	inactive for 6 months	1, 7, 15, 30 days 5th, 16-20th post	DT, SVM, LogR	Temporal, user, content, community	Perform. comp. of diff. subsets	undersampling	DT 30 days 74% Accur.
	[52]	votescore	-	-	Multiple LR	content, reputation, user	Spearmans Rank Coefficient	-	R^2=0.879
	[50]	answer quality	-	-	Retrieval Approach	non-textual	Pearson Correl.	-	-
	[51]	satisfaction	-	-	SVM, C4.5, RF, NB, AdaBoost	content, category, user (Quest., Answe.)	-	-	DT 77% Accur.
Online Gaming	[53]	**churn**	*hard churn*, *soft churn*	-	DT, LR, NB, NN	temporal, economy	DT	Discarding data of majority	DT F1=0.916
	[54]	**churn**	not active in second week	first session, day, week	RF, SVM, heuristic	Installation, game pattern	RF Feat. Import., LogR Coefficient, Std. error value	-	RF: 78.9% Accur.
	[55]	**churn**	inactive for 10 consecutive days	-	Survival Ensemble, SVM, NB, DT	game-independent	Integrated Brier Score	-	Survival Ens. 96% Accur.
	[56]	**churn**	given in dataset	-	NB, J48, AdaBoost	engagement, community	Perform. comp. of diff. subsets	-	-
	[57]	**churn**	not active for 7 and 14 days	7 | 14 days	RF, DT, SVM	community, purchase, activity	RF Feature Importance	SMOTE [20]	RF: 87% Accur.
	[58]	**churn**	not active for 14 days	-	NN, LogR, DT, SVM	-	Perform. comp. of diff. subsets	-	NN 0.930 AUC
Telecommun.	[60]	**churn**	given in dataset	-	NN, LogR, Multiple Reg.	demograph., purchases, user	-	-	NN: 91% Accur.
	[59]	**churn**	given in dataset	-	RF, KNN, Rotation Forest, Ensemble	numerical, nominal	mRMR [64]	undersampling (RanUS)	Ensemble 0.745 AUC
	[32]	**churn**	given in dataset	-	DT, NB, SVM, KNN, SGB, RF, MLP, LDA LogR, AdaBoost	communication, #service calls, account active time	RF Feature Importance, Boruta [65]	-	RF|AdaBoost 96% Accur.
	[61]	**churn** *financial groups*	given in dataset	-	DT, MLP, RF	user, income, web actions	DT Feature Importance	SMOTE, under- and oversampling, Combination	RF (under-sampled) 96% Accur.
	[62]	**churn** *urban migration*	not active for 3 weeks	-	RF, MLP, SVM, LogR	community, call behaviour, housing, geograph. patterns	Gini Importance (Mean decrease impurity)	-	RF F1=0.2576
	[63]	news popularity *online discussion*	-	-	RF	graph based	-	-	-

Table 12: The related work for prediction tasks in different contexts, and for this thesis interesting facts, like the used ML methods or the used features, as well as the methods for feature selection, scores of the best models, or used balancing methods.

4 Feature Engineering

In the following, we will show our feature engineering process. This will first include an explanation of our features based on an existed set, as well as their partitioning into different subsets.

4.1 Features

Base Features. We based on an already existed set of 33 features. They are grouped into generic, user, and community features. The generic features tell us time information about the month, year, and day of the month, and the week of registration. The user features are calculated values for a specific user. They include values like the number of down- or upvotes, or the number of replies per thread within different periods. The community features finally, display values for the whole community after different periods after registration, e.g., the number of created posts of the community within the first hour after registration, as shown in Tab. 16. This also includes the optimization target of our prediction task within the user features: *active_duration_minutes_lower_bound*. It describes a lower bound of minutes a user is active on Jodel.

Additional Features. In addition to these 33 features, we created 64 features for the user (61) and community (3) group. These features are shown in Tab. 17. With the added community features, we have added the postratio within the first hour and day after registration, and the average replies per post over the whole time.

Temporal Features. For the user features, we have added 61 new ones. These were adopted to some of the feature kinds, we have seen in the related work (cf. Sec. 3). in adoption to [49], we have added temporal features, that, e.g., represent the time between posts or interactions, or the time between registration and first post (*RegPostGap_h*).

Another set of features was adapted to the results in [53], where the *number of sessions and the days since registration*, and the *average time between sessions* were most important within the prediction task. We have adopted these, by adding the feature *registered_days*, which displays the absolute number of days the user was active. Additionally to this, we have also added features that give us information about the active days, in adoption to the session. We defined an active day as a day with at least one interaction, like posting or voting. These active days we have set into relation to the maximum possible number of active days since registration (*active_days_ratio*). When observing the active days, we have also investigated the inactive days, where we added a feature, that sets the longest absence time of a user in days in relation to the maximum possible number of days (*max_following_inactive_days_ratio*).

Interaction Features. Another set of features are the average posts and replies per day, which represent how active a user is on average a day, but also the average up- and downvotes. Additionally, the votescore per post, that represents indirectly the acceptance of the community, when, e.g., having a high average votescore, but also the number of created posts and replies or the overall happyratio.

Time Window Features. To observe the performance of the model when predicting user after a specific time (a day, week, two weeks, month, and three months), we have also expanded some features by time limitations, which resulted in multiple features that are limited by the given time boundaries. These features represent the development of user behaviour over time, e.g., *postcreated_Week* and *postcreated_3_months*.

4.2 Feature Selection and Limitations

After creating our features, we have used the Random Forest Feature Importance (RFFI) and ReliefF algorithm to observe the importance of the features. With the results of these, we got a view on which features are not important and which are significantly more important than others. This has shown, that we had one feature that yielded an importance of 0.79 in comparison to all others with importance lower or equal to 0.03. That is why we have decided to exclude the

feature *registered_days*. Because of the upper bound of active days it represents, the feature could lead the predictor to predict the lifetime of a user lower or equal than the feature value, which could be seen as 'cheating'.

After removing this feature, we have calculated the importance again, which resulted in a relatively 'balanced' importance distribution and we have decided to also use the less important features. Starting from this point, future work could investigate the performance change when using only the most important features, which could improve the model or decrease the performance in a trade-off with the runtime or memory usage.

4.3 Feature Subsets

After we had set our features, we have partitioned them into different subsets for future investigation.

General Subsets. Firstly, we have subdivided the features into the two general subsets *user* and *community*, where each subset includes the presented features in Tab. 16 and 17 and additionally the generic features. This has led to subset sizes as shown in Tab. 13, where we have not counted the optimization target and given the number of features overall. Both subsets do not include generic features. From overall 95 features, we will use 73+5 features as user subset and 17+5 features as community subset.

group	all	user	community	generic
subset size	95	73	17	5

Table 13: The number of features per group. Within our further subset analysis, we will include the generic features into the other three subsets. We have also not counted the optimization target *active_duration_minutes_lower_bound*.

Time-Window Subsets. For an investigation of the performance of models after a given time, we have subdivided our features into time-dependent and independent features. The independent features are not limited to a time boundary and would, therefore, yield too much information when, e.g., predicting for observation of one day. The numbers of features are shown in Tab. 14. We have subdivided the time-dependant features into five time-window subsets. Each subset has an upper border that represents the time of observation. That is why each subset includes the community subset, because of features that are only related to a smaller or greater time than the given windows. Additionally, each subset includes the features that have a lower or equal time border. That is why the subsets are cumulated, which means that all features of the smaller time-window subset are also included in all bigger ones. The sizes of the subsets are shown in Tab. 15. The affiliation of the single features is have shown in Tab. 16 and 17 in the right column, where the smallest subset affiliation is given, which means that this feature also belongs to all bigger subsets, as mentioned before.

	time-indep.	**time-dep.**
user	23	50
community	1	16
generic	5	0
$\sum$	29	66

Table 14: The number of features within the three general groups, which are time-dependent and independent. Where independent means, that the features are not limited to a time boundary.

window	day	week	2 weeks	month	3 months
subset size	33	50	57	64	71

Table 15: The number of features per observation window, where the features of the smaller window subset are also included in all bigger window subsets. Each subset includes the 16 time-dependent community features, as well as the five general features.

	Description	window affiliation
generic features (5)		
month	Month of registration	-
year	Year of registration	-
hourUTC	The hour of registration	-
dayOfWeek	Day of the week of registration	-
dayOfMonth	Day of month of registration	-
user features (14)		
active_duration_minutes_lower_bound	The minimal number of active minutes of a user	-
upvoted_24h	Number of given upvotes within the first day	day
downvoted_24h	Number of given downvotes within the first day	day
participated_threads_24h	#threads a user participated within the first day	day
postcreated_24h	#posts created within the first day	day
postflagged_24h	#posts that got flagged within the first day	day
postreply_positive_karma_24h	#posts that generated positive karma within the first day	day
received_downvotes_24h	#downvotes received for posts and replies within the first day	day
received_upvotes_24h	#upvotes received for posts and replies within the first day	day
replies_per_thread_24h	#replies per thread within the first day	day
replycreated_24h	#replies created within the first day	day
voted_within_threads_24h	#threads the user voted within the first day	-
user_replies_per_thread	#replies per posts the user replied to	-
user_threads	#posts the user replied to	-
community feature (14)		
posts_day	#posts in community within the first day	day
replies_day	#replies in community within the first day	day
picture_posts_day	#posts with pictures in community within the first day	day
downvoted_day	#downvotes in community within the first day	day
upvoted_day	#upvotes in community within the first day	day
happyratio_day	happyratio in community within the first day	day
avg_response_time_minutes_day	Avg. minutes between post creation and the first response in community within the first day	day
posts_hour	#posts in community within the first hour	day
replies_hour	#replies in community within the first hour	day
picture_posts_hour	#posts with pictures in community within the first hour	day
downvoted_hour	#downvotes in community within the first hour	day
upvoted_hour	#upvotes in community within the first hour	day
happyratio_hour	happyratio in community within the first hour	day
avg_response_time_minutes_hour	Avg. minutes between post creation and the first response in community within the first hour	day

Table 16: The basis set of features, grouped into generic, user, and community features, with their explanation and their affiliation to the time window subsets (cf. Tab. 15).

	Description	window affiliation
community feature (3)		
postratio_day	postratio in community within the first day	day
postratio_hour	postratio in community within the first hour	day
avg_replies_post	The average number of replies per post within the community	-
user features (61)		
active_days_ratio	#days with min. one interaction since registration in ratio to all possible days	-
active_days_firstWeek_ratio	#days with min. one interaction within the first week in ratio to all possible days	week
active_days_first2Weeks_ratio	#days with min. one interaction within the first two weeks in ratio to all possible days	2 weeks
active_days_firstMonth_ratio	#days with min. one interaction within the first month in ratio to all possible days	month
active_days_first3Months_ratio	#days with min. one interaction within the first three months in ratio to all possible days	3 months
avg_minutes_between_interactions	Average minutes between user interactions	-
avg_minutes_between_downvoted	Average minutes between user downvotes	-
avg_minutes_between_upvoted	Average minutes between user upvotes	-
avg_minutes_between_posts	Average minutes between user postings	-
avg_minutes_between_replies	Average minutes between user replies	-
avg_postcreated_day	Average #posts per day of user	-
avg_replycreated_day	Average #replies per day of user	-
avg_upvotes_per_post	Average #upvotes per post for user	-
avg_downvotes_per_post	Average #downvotes per post for user	-
avg_votescore_per_post	Average #votescore per post of user	-
avg_votescore_firstDay	Average #votescore per post within the first day	day
avg_votescore_first2Days	Average #votescore per post within the first two day	week
avg_votescore_first3Days	Average #votescore per post within the first three day	week
avg_votescore_firstWeek	Average #votescore per post within the first week	week
avg_votescore_first2Weeks	Average #votescore per post within the first two weeks	2 weeks
avg_votescore_firstMonth	Average #votescore per post within the first month	month
avg_votescore_first3Months	Average #votescore per post within the first three month	3 months
postcreated	Number of created posts (overall)	-
replycreated	Number of created replies (overall)	-
upvotes	Number of received upvotes (overall)	-
downvotes	Number of received downvotes (overall)	-
votescore	Votescore (overall)	-
postratio	Postratio (overall)	-
happyratio	Happyratio (overall)	-
blocked	Number blocks (overall)	-
happyratio1stWeek	Happyration within the first week	week
happyratio1st2Weeks	Happyration within the first two weeks	2 weeks
happyratio1stMonth	Happyration within the first month	month
happyratio1st3Months	Happyration within the first three months	3 months
postcreated_2_Days	#posts created within first two days	week
postcreated_3_Days	#posts created within first three days	week
postcreated_Week	#posts created within first week	week
postcreated_2_Weeks	#posts created within first two weeks	2 weeks
postcreated_1_month	#posts created within first month	month
postcreated_3_months	#posts created within three months	3 months
replycreated_2_Days	#replies created within first two days	week
replycreated_3_Days	#replies created within first three days	week
replycreated_Week	#replies created within first week	week
replycreated_2_Weeks	#replies created within first two weeks	2 weeks
replycreated_1_Month	#replies created within first month	month
replycreated_3_Months	#replies created within first three months	3 months
received_upvotes_first2Days	Number of received upvotes just for posts within first two days	week
received_upvotes_first3Days	Number of received upvotes just for posts within first three days	week
received_upvotes_firstWeek	Number of received upvotes just for posts within first week	week
received_upvotes_first2Weeks	Number of received upvotes just for posts within first two weeks	2 weeks
received_upvotes_firstMonth	Number of received upvotes just for posts within first month	month
received_upvotes_first3Months	Number of received upvotes just for posts within first three months	3 months
received_downvotes_first2Days	Number of received downvotes just for posts at first two days	week
received_downvotes_first3Days	Number of received downvotes just for posts at first three days	week
received_downvotes_firstWeek	Number of received downvotes just for posts at first week	week
received_downvotes_first2Weeks	Number of received downvotes just for posts at first two weeks	2 weeks
received_downvotes_firstMonth	Number of received downvotes just for posts at first month	month
received_downvotes_first3Months	Number of received downvotes just for posts at first three months	3 months
max_following_inactive_days_ratio	Maximum number of following days without interactions in relation to possible days	-
RegPostGap_h	Hours between registration and first post	-
registered_days	Number of days since registration (**removed**)	-

Table 17: Created additional features for the community and user group separated into 12 subgroups within the user features. We have given their explanation and affiliation to the window subsets (cf. Tab. 15).

5 Pipeline and Implementation

In the following, we will describe and explain our Machine Learning (ML) pipeline, as shown in Fig. 23. We structured this into three main steps. The data preparation, the model building and training, and the final prediction task, which also includes evaluations. In addition to this, we will also explain the framework implementation, which was written in Python. We used the pipeline of the Imbalanced-Learn package, which gives us the possibility to use also methods from SciKit-Learn within our pipeline.

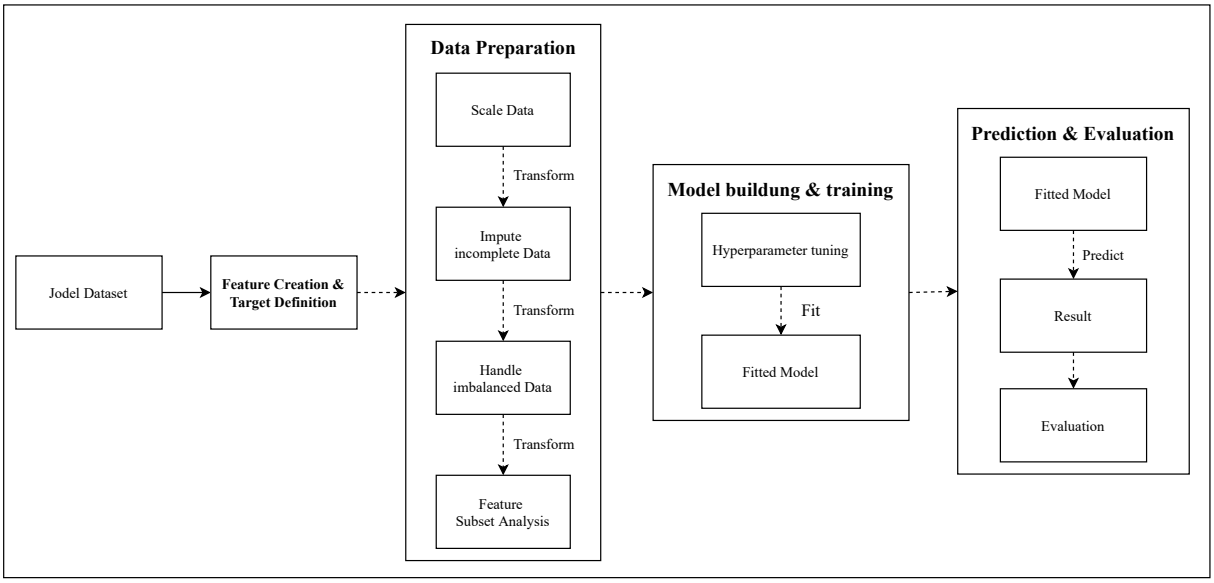

Figure 23: The ML pipeline including feature creation from the dataset, the preparation, which includes scaling, balancing, as well as the feature subset analysis and the handling of incomplete data, the training of the model on the training data, and the prediction of the best model on the test data with a final evaluation.

5.1 Data Pre-Processing

As the first step, we will need to select and prepare our data to handle different issues. We will, therefore, use the tools we described in Section 2.2.

To work with discretization, we are using manual bins or the *KBinsDiscretizer* from SciKit-Learn [66], which will automatically bin the data into generated intervals. This will let us classify our users into the different churn classes we described in Section 1.3.

As the second step, we will then create different features from our dataset and define our optimization target. Before building and testing the models, we will then run a feature subset analysis, which was explained in more detail in Section 4. Therefore, we will use the *ReliefF* from Skrebate [67], which we have explained in Section 2.1.2. In addition to this method, we will also use the *feature importance* and *decision tree* of the tree-based learning methods (cf. Sec. 2.1.1), as well as the RFECV (cf. Sec. 2.1.3). These will calculate the features which weigh most within the decision process. We will then build and test our model on different feature subsets to evaluate later which subsets will yield the best results.

As we have shown in Section 1.2, we will also need to handle the imbalanced data. Therefore, we will use over- and under sampling methods. These will adjust the class distribution.

5.1.1 Imputing

Due to the fact that the data was incomplete and therefore, missing values for some users in case of *active minutes*, because of no *post- and happyratio*, that results from no created posts or replies and respectively no existing feedback of the community. This leads to NULL values within our created features and therefore, an incomplete dataset. To handle this we will exclude

all users without an *active minutes* value, because of the distortion of the results by imputing the optimization target which we want to predict. To handle the incompleteness of all other features, we will use imputation. This means we will use the *Simple Imputer* from SciKit-Learn [68] and replace the missing values by mean.

5.1.2 Scaling

As the next step, we will also scale and normalize our data. We will need this because of the usage of different units within the dataset. We, e.g., use the feature which tells us about active minutes and the number of posts that have different units and can vary greatly. This is one of the things that ML methods can be highly sensitive to. To handle this, we will use from Scikit-Learn on the one hand a *Min-Max-Scaler* (cf. Sec. 2.2.1.2) which will normalize the values, so they range between zero and one. And on the other hand, we will use a *Standard Scaler* (cf. Sec. 2.2.1.1), which will transform the data, so that we get a mean of zero and a standard deviation of one [69].

5.2 Model building and Testing

For the next two steps, we will split our data into a random testing and training subset, by using the *ShuffleSplit* from SciKit-Learn [70]. With the training subset, we will build our different models. We will, therefore, perform a hyperparameter tuning by running a grid-search (cf. Sec. 2.4). This will test different parameter combinations of the model to find the best ones. In our framework, this grid-search will be in combination with a k-fold cross-validation (CV) [40] which we have described in Section 2.5. This will take every parameter combination and will run a k-fold CV and calculate different metrics. These metrics will be the R^2 Score, the Mean Squared Error (MSE), and Mean Absolute Error (MAE) for regression tasks, and the precision, recall, F1 Score, and accuracy for classification tasks (cf. Sec. 2.6). This will allow us to compare different models and select the best ones. We will then validate the optimized models for different cities and different feature subsets.

5.3 Prediction and Evaluation

As the last step, we will use our prepared testing data and the fitted and optimized models for the final prediction task. This prediction will then also calculate different evaluation metrics for better comparison and save them into a JSON file, together with the results of the feature importance and ReliefF.

In addition to this, the framework plots a heatmap in dynamic bins for regression or the defined classes as described in Tab. 6 in Section 1.3. To compare the accuracy of the models within the different classes, we will then also calculate the differences of the heatmaps to figure out which model works best in which section. As another evaluation part, the framework will also calculate and plot the ROC/AUC (cf. Sec. 2.6). To validate the final models we will also run a k-fold CV, which we described in Section 2.5.

5.4 Framework Structure

We have already described the packages we used within our framework. In the following, we will put the different parts of our framework together to an overall diagram to get an overview. We have drawn the parts we added for this thesis in orange ($\propto$) and modified in green ($\bowtie$).

As shown in Fig. 24, we added the possibility of querying the data from the database we want to use (`process_query()`).

We have also modified the plotting of the grid search results and heatmaps (Package: `plot`), to get a better view of our defined classes. That is why now automatically six heatmaps are generated for regression problems, which are zooming into the time windows we have defined, to get a closer look.

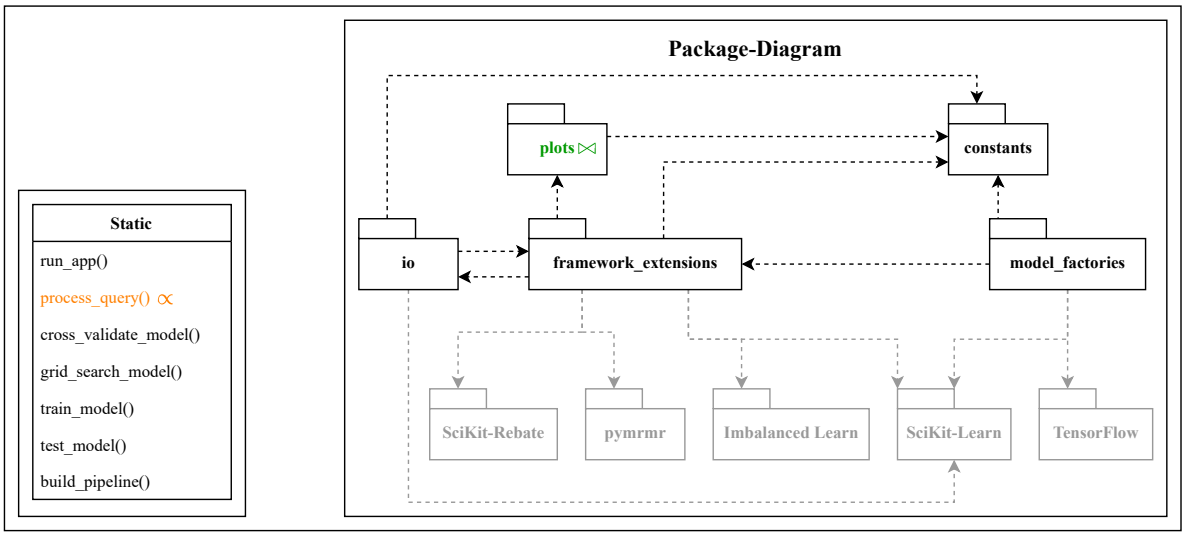

Figure 24: The package diagram of the framework and the static functions, which do not belong to any package.

We have also switched the used pipeline implementation from SciKit-Learn to the pipeline from Imbalanced-Learn, to get the chance for using implementations of both packages within our pipeline, as shown in Fig. 25.

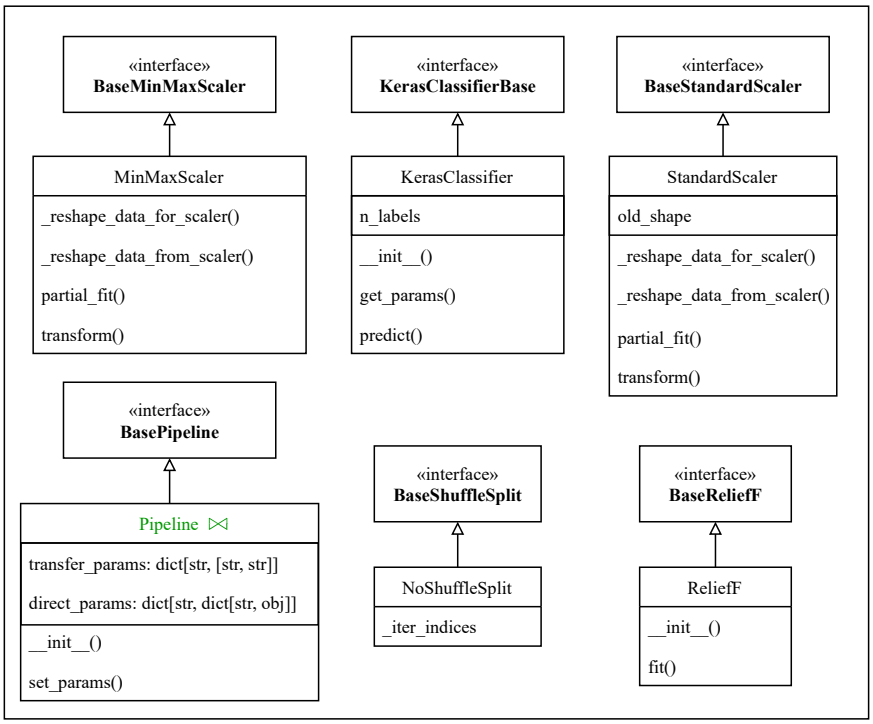

Figure 25: The class diagram of the *framework extensions* package, which includes the changed pipeline of the Imbalanced-Learn Package.

For the data preparation we added the balancing methods (`DataBalanceType`) and the imputing (`ImputingType`), which can handle different strategies, e.g., replacing by mean or median.

As shown in Fig. 26, we also added ML methods, like the Naïve Bayes (NB), AdaBoost, or the Multi-Layer Perceptron (MLP) (`ModelType`). In this context we also added functions to figure out if we are using a classification model and if we want to use a feature selection method (`is_classification()`, `has_feature_importance()`).

To have more variety in evaluating the models, we added the ROC/AUC score for classification tasks, and the explained variance score, and the maximum residual error for regression tasks.

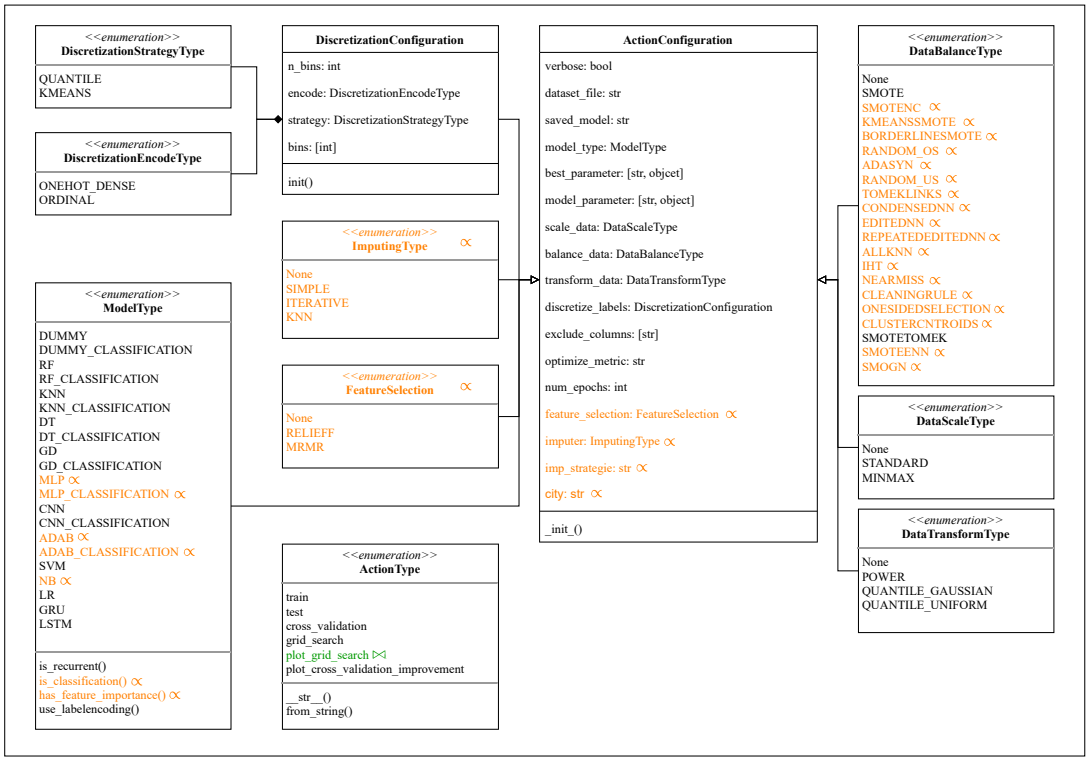

Figure 26: The class diagram of the *constants* package, which includes the added `ImputingType`, the `FeatureSelection`, newly added ML methods, the functions for detecting classification and feature selection, and new methods for the balancing.

5.5 Framework Testing

To ensure that the framework works flawlessly, we used different intermediate results. While creating the features, we have tested the calculation of the features on a range of users, which should be as different as possible. We calculated the features for these users by hand and compared the results to ensure correctness.

Within our framework and pipeline, we ensured correctness by checking the intermediate results. This includes graphics, which showed us the effect of balancing in Fig. 27 and scaling in Fig. 28, as well as the output of numbers of missing values before and after imputing, or the features of a subset, which were selected within the JSON file, which include the results. These intermediate results can then be checked to ensure correctness.

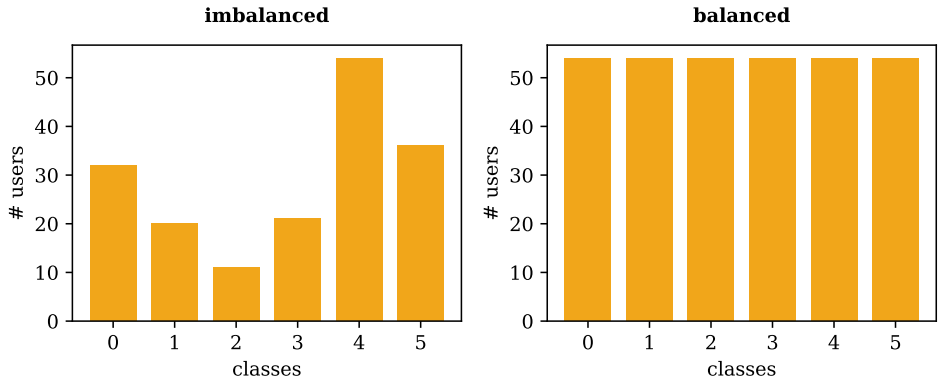

Figure 27: An example of balancing with SMOTE for the defined churning classes in Al Jafr. The number of samples raised from 174 to 324 and led to a regular class size of 54 samples.

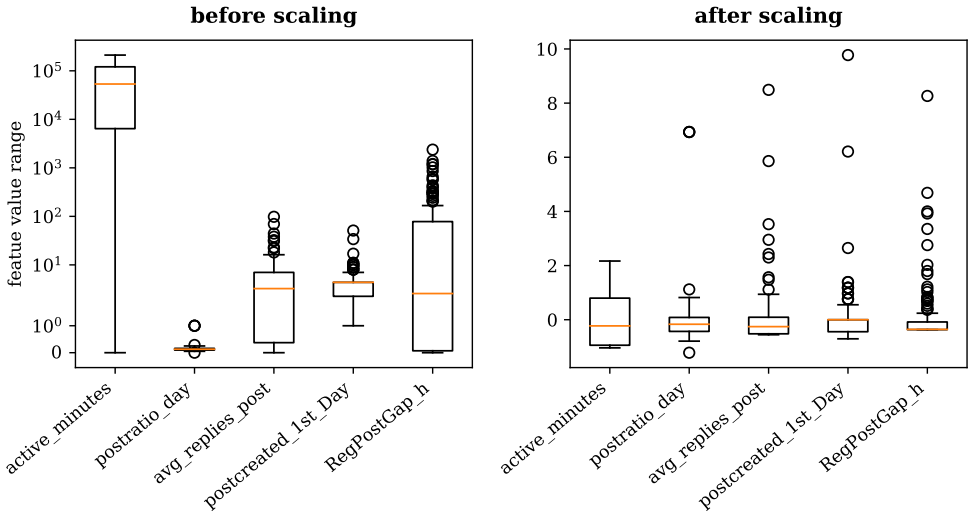

Figure 28: An example of scaling the presented selected features. The value ranges of the features differ greatly before scaling. The, e.g., *active minutes* range from 0 to 211,047 in contrast to a range of 0 to 1 for the *postratio_day* feature.

6 Evaluation

Within the following section, we will present and explain our Machine Learning (ML) approach results. For this, we will first present the theory of our proceedings, which will be followed by the results.

We will start by defining the baselines for the different tasks, which the different models will need to exceed. The different tasks will be the regression, where we will need to predict the exact *active minutes*, and the classification where our model will predict a) one of the six churn classes in Tab. 6 and so a multi-label classification, and b) a binary classification, where our model will predict if a user has an active time lower or equal than a given threshold or a greater one, where the threshold will be one day, week, two weeks, one month or three months.

Our proceeding can be split into two parts. As a first step, we will run a **breadth-first search**, as shown in Fig. 29. Within this step, we will test the presented ML methods, we have presented in Section 2.3, for the communities of Jeddah with more than 100k users, Al Jafr with 174 users, and the whole country with over one million users to get a view on their performance on different data sizes. With these results, we will select the method, that yielded the best R^2 Score for the regression task and accuracy for the classification task.

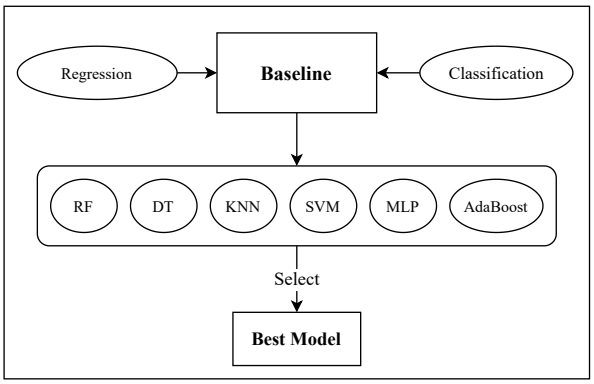

Figure 29: The *breadth-first search*, where we will use the different usable methods, to get an overview of which models work best for the different tasks, which will then be fixed and be investigated in detail. The methods will be compared by different baselines, which they need to exceed.

As the second step, we will run a **depth-first search**, as shown in Fig. 30. For this, we will use the best method, which will be fixed, and will have a more fine granular investigation. This will start with an observation of different data transformation tools, as we have shown in Section 2.2, to detect, e.g., the best imputing strategy. After we have defined which tools strengthen our model, we will fix our setup and will observe the performance change within the different communities, as well as the change when using different feature subsets, as we have shown in Section 4. Besides the communities of different cities, we will also present the performance of a country model, which could be more generalised and therefore, work well on the whole country, but also perform strongly when predicting only users of the single communities. After this step, we will have an overview of the different tasks and the best models for each.

After the evaluation of the different models and detecting the best ones for different tasks, we will then investigate the insights of the models. For this, we will start by training the model on its specific community and will let it predict the active time within the other communities. With these models, we will see, if these also work very well within different communities and therefore, seem to have similarities in behaviour, w.r.t. user churn. We will also see if the country model has an equal or even better performance compared to the specific models within the different communities. To detect the specific behaviours, we will then have a look into the feature importances of the Random Forest (RF), as well as of the ReliefF, which we have described in Section 2.1.2. With the help of correlations between the important features of the different communities, we will then see which communities have also similarities in feature

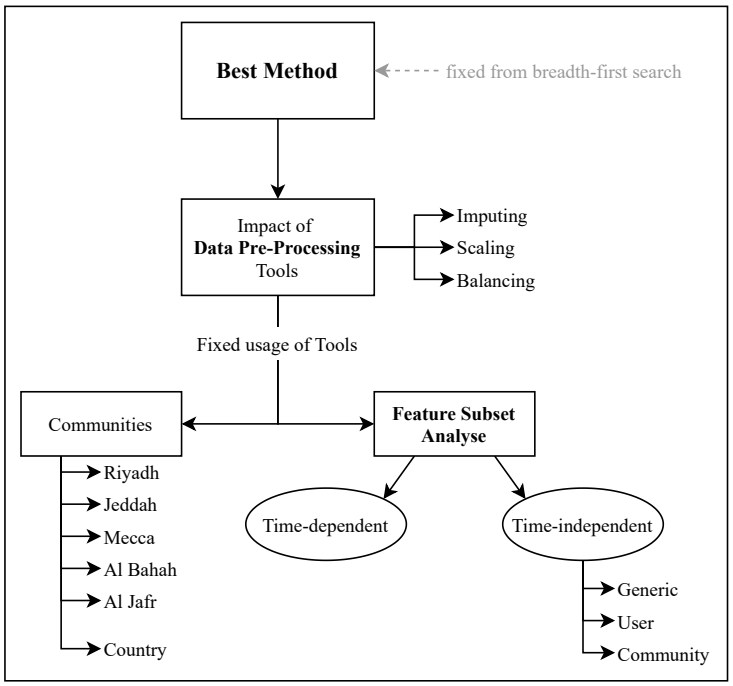

Figure 30: The *depth-first search*, which will use the fixed selected method from the breadth-first search. We will start by testing different data transformation tools, which then will also be fixed and used, followed by a fine granular observation of performance change within the different communities, as well as a detailed feature subset analysis.

importance and thus, similar models.

The features with the highest importance are then presented, as well a the cumulated importance of the subsets to detect these similarities. These features will be seen as features with the most impact on predicting the active time.

As a final step, we will then investigate the empiricism of the best features and will search for feedback from it. This will include the test of which user groups are statistically significantly different distributed for the best features, as well as detecting significant trends for these features. With these trends, we will then probably see connections between the active time of a user to specific features, which will give us explicit feedback on why the features was that important and how the users behaviour is changing within the user groups, which could be used for a better description of the communities.

For our ML approach, we will have a look at the cities Riyadh, Jeddah, Mecca, Al Bahah, and Al Jafr, which differ in user-base size.

Besides these, we will present a model approach for the whole country. We will start our investigation by selecting the best working method in Section 6.1, which will be followed by a precise observation of the best one, in Section 6.2 and Section 6.3.

6.1 Machine Learning Algorithm Selection

In this section, we will have a look at the general performance of the different usable methods. These will include the Decision Tree (DT), K-Nearest Neighbours (KNN), Support Vector Machine (SVM) Multi-Layer Perceptron (MLP), Stochastic Gradient Descent (SGD), Random Forest (RF), and the Adaptive Boosting (*AdaBoost*). We run a grid-search (cf. Sec. 2.4) of some generally accepted parameters for the different methods on the data of the communities of Jeddah and Al Jafr, as well as on the whole country in Tab. 18. This gives a view of the general performance of the methods on different data sizes. This is shown for the regression, where the R^2 Score and its standard deviation were used, and the multi-label classification task, where the accuracy and its standard deviation (std dev) were used. We have also given the *baseline* for the given tasks, where we used the *Dummy Regressor/Classifier* from SciKit-Learn [71, 72]. For the

	score	**DT**	**KNN**	**MLP**	**SGD**	**RF**	**AdaBoost**	**SVM**	*Baseline*
Jeddah	R^2	0.9336	0.6555	0.9215	0.4525	**0.9667**	0.8062	--	-0.00009
	std dev	±0.0025	±0.0032	±0.0266	±0.01057	±0.0016	±0.0030	--	
	accur.	0.8000	0.5668	0.7844	*0.2302*	**0.9413**	0.6856	0.9251	0.2554
	std dev	±0.0065	±0.0033	±0.0262	±0.0828	±0.0006	±0.01668	±0.0010	
Al Jafr	R^2	0.7220	0.4881	0.3887	0.2753	**0.8219**	0.7948	--	-0.02382
	std dev	±0.1168	±0.1321	±0.1953	0.4098	±0.1115	±0.0783	--	
	accur.	0.6551	0.5286	0.5753	*0.2590*	**0.7012**	0.6035	0.5861	0.3104
	std dev	±0.0730	±0.0511	±0.0977	±0.0916	±0.0384	±0.0655	±0.0359	
Country	R^2	0.9580	0.6764	0.9668	0.3422	**0.9819**	0.7654	--	-0.00001
	std dev	±0.0009	±0.0013	±0.0036	±0.0551	±0.0002	±0.0012	--	
	accur.	0.8288	0.5871	0.7271	0.2639	**0.9697**	0.6974	--	0.2818
	std dev	±0.0059	±0.0006	±0.0222	±0.0062	±0.0004	±0.0077	--	

Table 18: The results of the grid-search with 5-fold cross-validation for the regression (R^2 Score) and classification (accuracy score) task and their standard deviations (std dev), for the different ML algorithms over all features. This gives us information on which method works best to use for a detailed observation. As we can see, for all data sets, the RF yielded the strongest score and only the SGD did not exceed the baselines. We have also listed only the results of the SVM for the classification task, because of the non-termination of for the regression task and the classification task on the country data.

regression task, the dummy regressor always predicted the mean, and for the classification the most frequent label. These baselines need to be exceeded by the methods.

As we can see in Tab. 18, for all three datasets, the RF yielded the best score and just the SGD falls below the baselines for the classification task in all three datasets. For the regression task, we can also see that there are also the DT and MLP that resulted in high scores.

Because the RF is the best method in all three communities by R^2 Score and accuracy, which is an aggregation, we looked at the detailed results in case of a heatmap, to see how the prediction of the RF is distributed and scattered and confirm it as the best method, as shown in Fig. 31 and 32 for Jeddah. We only visualised the results of Jeddah because of the small number of samples in Al Jafr which results in too few data points, which do not give a conclusive image.

We have plotted the results of the regression in Fig. 31 and the classification task in Fig. 32. The heatmaps represent the real *active days*/churn class of a user on the x-axis and the predicted *active days*/churn class on the y-axis. It displays the normalised number of users predicted for the different cases.

For the *regression task*, we can see, that we have an overall relatively low scattering around the diagonal line. This diagonal displays the optimal prediction result. Just above around 80 days, we can see an increased scattering, which may be due to a lower number of users that were active for that long time and therefore, less information of that users. This confirms that the RF performs well for this problem.

In Fig. 32, the results of the *classification task* and therefore, the real and predicted classes of the users are shown. We have again normalised the results to get a more generalized view. As we can see, we have a very low scattering around the diagonal and 88% of the users were predicted on the diagonal and therefore, correct. This shows us, that we can also confirm the RF as a classification method with high predictive power.

As we have seen, the RF resulted in the best scores for both tasks, which we could also confirm by looking at the predicted values. Because of the large parameter space and limitations in runtime for tuning and testing, detailed research for all methods is infeasible. That is why we will focus on the RF, as the best method, within the next sections. For this method, we will observe the impact of changing the imputing strategy, as well as the use of scaling and balancing methods and a further investigation of the performance of the model, when using the different feature subsets.

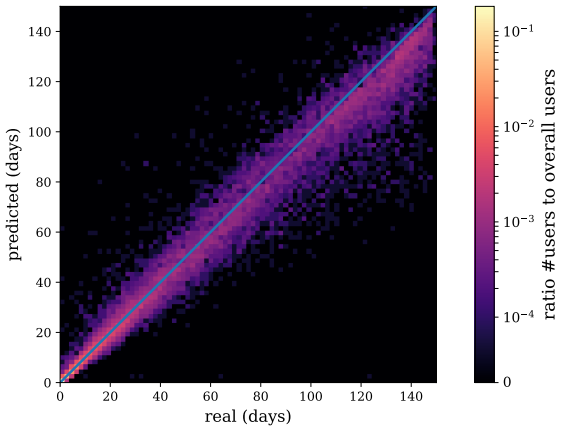

Figure 31: A detailed look at the prediction of the regression task in Jeddah. The x-axis displays the real *active days* of a user and the y-axis the predicted *active days*. This shows us that we have a huge ratio of users that were predicted correctly along the optimal diagonal up to around 80 days and a bit more scattering above. Overall we can see that we have a relatively low scattering around the diagonal.

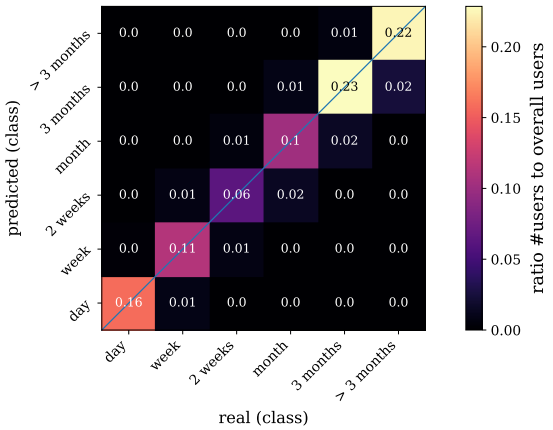

Figure 32: A detailed look at the prediction of the regression task in Jeddah. The x-axis displays the real *churn class* of a user and the y-axis the predicted *churn class*. We can see that there is a very low scattering around the optimal diagonal, and 88% of the users were predicted on this diagonal, which shows us that RF also performs very well for the classification task.

6.2 Impact of Data Pre-Processing

In this section, we will have a look at the impact of different tools for pre-processing the data. We will start by observing the change of the imputing strategy from *mean* to the *median* and *most frequent*. After that, we will have a look at the two presented scaling methods and finally at the three presented balancing methods, Random Oversampler (RanOS), Random Undersampler (RanUS), and SMOTE.

6.2.1 Impact of Imputing

Because of the already named non-treatment of missing data, we imputed our data till now by *mean*. In the following, we will investigate the other imputing strategies *median* and *most frequent* for the community models. We will, therefore, have a look at the observation of *all features* to get a generalised result and to get information on if and how much our model improves. For the regression task, we have shown the results of imputing by the three strategies within the different communities in Tab. 19 and for the classification task in Tab. 20.

As we can see in Tab. 19, the change of the imputing strategy decreases the R^2 Score in all communities, except Riyadh and Mecca, where using the *most frequent* can enhance the score. Overall we can see, that changing the strategy does not enhance or deteriorate the score for the regression task. This can also be confirmed by a Kruskal-Wallis H-Test, where we have tested if one of the strategies is significantly different from the others. The equal distribution of the scores for the three methods was H_0. Because of a statistic of 0.2454 and a p-value of 0.8845, we will not reject H_0, which means for us that the change of the imputing strategy does not significantly change the R^2 Score.

In Tab. 20, we can see that using the *most frequent* strategy enhances the accuracy in nearly all communities, which shows us, that using the *most frequent* strategy will yield better results for the RF classification task. We again used the Kruskal-Wallis H-Test with equally distributed scores of the communities for the different imputing strategies as H_0. With a statistic of 0.18 and a p-value of 0.9139, we can also say that changing the imputing strategy will not trigger a statistically significant difference in the score. Because of the fact, that the score increased in four of five communities, we nonetheless have decided to use the *most frequent* strategy for the following classification tasks.

Regression

comm.	**mean**	**median**	**most frequent**
Riyadh	0.9728	0.9722	**0.9733**
std dev	±0.0013	±0.0011	±0.0008
Jeddah	**0.9667**	0.9658	0.9667
std dev	±0.0016	±0.0021	±0.0017
Mecca	0.9551	0.9539	**0.9571**
std dev	±0.0035	±0.0035	±0.0026
Al Bahah	**0.9457**	0.8873	0.9430
std dev	±0.0032	±0.0022	±0.0018
Al Jafr	**0.8219**	0.8033	0.7866
std dev	±0.1115	±0.1020	±0.1134

Table 19: The mean R^2 Score and its standard deviation (std dev) for the regression task within the different communities when imputing by the three different strategies, shows us that changing the imputing decreases the score in most cases, except for Riyadh and Mecca, and therefore, imputing by *mean* yields the best results. Besides the fact, that the scores are higher, we can also see no significant in- or decreasing, when using the Kruskal-Wallis H-Test.

Classification

comm.	**mean**	**median**	**most frequent**
Riyadh	0.9578	0.9551	**0.9625**
std dev	±0.0006	±0.0011	±0.0007
Jeddah	0.9413	0.9427	**0.9436**
std dev	± 0.0006	±0.0007	±0.0008
Mecca	**0.9240**	0.9218	0.9231
std dev	± 0.0035	± 0.0032	±0.0039
Al Bahah	0.8811	0.8873	**0.8885**
std dev	± 0.0080	±0.0071	±0.0060
Al Jafr	0.7012	0.7183	**0.7240**
std dev	±0.0384	±0.0383	±0.0694

Table 20: The mean accuracy and its standard deviation (std dev) within the different communities when imputing by the three different strategies, which shows us that using the *most frequent* strategy increases the accuracy and can, therefore, yield the best results. Besides the fact that we cannot see a significant increase when using the Kruskal-Wallis H-Test, we nonetheless have decided to use the most frequent strategy for the classification task.

6.2.2 Impact of Scaling

Another aspect to look at when transforming the data is scaling. In the following, we considered using a Standard Scaler for standardisation or the Min-Max-Scaler for normalising from SciKit-Learn, which are explained in Section 2.2.1. In this section, we will observe the performance change when using both tools. The results within the different communities for the regression task are shown in Tab. 21 and for the classification task in Tab. 22. The results were calculated on *all* features.

As we can see in Tab. 21, using both scaling methods does not affect the R^2 Score, except in Al Jafr where the Min-Max-Scaler increased the score of just 0.03, but with a standard

Regression

community	**without**	**Standard**	**MinMax**
Riyadh	0.9728	0.9728	0.9728
std dev	±0.0013	±0.0013	±0.0013
Jeddah	0.9667	0.9667	0.9667
std dev	±0.0016	±0.0016	±0.0016
Mecca	0.9551	0.9551	0.9551
std dev	±0.0035	±0.0035	±0.0036
Al Bahah	0.9457	0.9457	0.9457
std dev	±0.0032	±0.0032	±0.0031
Al Jafr	0.8219	0.8218	**0.8221**
std dev	±0.1115	±0.1113	±0.1115

Table 21: The mean R^2 Score and its standard deviation (std dev) within the different communities for the regression task using all features, when using the Standard Scaler and Min-Max-Scaler from SciKit-Learn in comparison to the score without scaling. This shows us that scaling the data will not change our results and just in Al Jafr the score is increasing by about 0.003 when using the Min-Max-Scaler, which does not signalise a significant difference, especially when having a standard deviation of over 0.1.

Classification

community	**without**	**Standard**	**MinMax**
Riyadh	0.9578	0.9578	*0.9574*
std dev	±0.0006	±0.0006	±0.0006
Jeddah	0.9413	0.9413	**0.9415**
std dev	±0.0006	±0.0006	±0.0004
Mecca	**0.9240**	*0.9238*	*0.9234*
std dev	± 0.0035	±0.0033	±0.0032
Al Bahah	**0.8811**	*0.8806*	*0.8810*
std dev	± 0.0080	± 0.0077	± 0.0076
Al Jafr	0.7012	0.7012	0.7012
std dev	± 0.0383	±0.0384	±0.0384

Table 22: The mean accuracy and its standard deviation (std dev) within the different communities for the classification task using all features, when using the Standard Scaler and Min-Max-Scaler from SciKit-Learn in comparison to the score without scaling. This shows us that scaling the data for the classification task just increases in Jeddah when using the Min-Max-Scaler, but decreases or stands equal in all other communities when using scaling. This can also be seen with the standard deviation, where the possible scorings are crossing. This leads to the same result as for the regression task.

deviation of over 0.1, there exists no significant difference. In Tab. 22, we have shown, that the methods for the classification task are just increasing by using the Min-Max-Scaler in Jeddah. In all other communities, we can detect an equal or lower score which shows us, that using the scaling methods will not improve our prediction results. This can also be seen with the standard deviations, which symbolise that the usage of a scaler does not significantly improve the models. We have seen that using scaling methods will not improve the prediction results of the RF, and if, it improves, it does not have a huge impact and cannot be generalised.

6.2.3 Impact of Balancing

As we have mentioned in Section 1.2, a common problem in churn prediction is the imbalance of the data, which can lead to inaccuracy, due to the lack of information. To solve this we will use different balancing methods, which we have described in Section 2.2.2. We have tested these methods on the different communities by using *all features*, to get a look at the general improvements when using balancing.

As we can see in Tab. 23, the balancing methods for the classification task do not enhance the accuracy within the different communities and especially the RanUS decreases the accuracy strongly, which shows, that balancing methods are not very sensible for the classification task. This also gets confirmed by using the Kruskal-Wallis H-Test with equally distributed scores of the communities for the different balancing methods as H_0. With a statistic of 4.12 and a p-value of 0.2488, we can confirm that the usage of a balancing method has no statistically significant difference between the scores, because of the non-rejection of H_0.

	Classification				
community	**without**	**RanOS**	**RanUS**	**SMOTE**	*Baseline*
Riyadh	**0.9578**	0.9550	0.9358	0.9532	0.1892
std dev	±0.0006	±0.0005	±0.0015	±0.0002	
Jeddah	**0.9413**	0.9367	0.9201	0.9365	0.1761
std dev	±0.0006	±0.0018	±0.0017	±0.0021	
Mecca	**0.9240**	0.9154	0.8974	0.9175	0.1793
std dev	±0.0035	±0.0030	±0.0035	±0.0022	
Al Bahah	**0.8811**	0.8806	0.8595	0.8746	0.1754
std dev	±0.0080	±0.0075	±0.0075	±0.0096	
Al Jafr	**0.7012**	0.6494	0.5462	0.6489	0.2471
std dev	±0.0383	±0.0661	±0.0534	±0.1181	

Table 23: The change of the mean accuracy and its standard deviation (std dev) for the regression task within the different communities when using the different balancing methods. We have printed the highest scores for each community in bold. We can see, that balancing is not enhancing the score within the different communities. Especially when using the RanUS, a decrease in the accuracy can be seen. Overall we cannot detect a significant difference, which also follows by the high standard deviations.

In contrast to the classification task, we can also see in Tab. 24, that using the RanOS increases the mean R^2 Score in all communities, except Riyadh. Especially in Al Jafr, an enhancement of 0.0031 and in all other communities of just 0.0001 can be seen. But this enhancement in Al Jafr also needs to be relativised, because of the high standard deviations. The RanUS did not affect the scores, except of Riyadh, where a small decrease was detected. When having a look at the significant difference of the scores with the Kruskal-Wallis H-Test and the same H_0 as for the classification task, a statistic of 0.4558 and a p-value of 0.7962 shows, that the scores of the communities are not statistically significantly different, because of the non-rejection of H_0.

	Regression			
community	**without**	**RanOS**	**RanUS**	*Baseline*
Riyadh	0.9728	0.9728	0.9709	-0.00032
std dev	±0.0013	±0.0013	±0.0013	
Jeddah	0.9667	**0.9668**	0.9655	-0.00046
std dev	±0.0016	±0.0017	±0.0019	
Mecca	0.9551	**0.9553**	0.9548	-0.00036
std dev	±0.0035	±0.0035	±0.0045	
Al Bahah	0.9457	**0.9458**	0.9448	-0.00056
std dev	±0.0032	±0.0031	±0.0032	
Al Jafr	0.8219	**0.8250**	0.8144	-0.02325
std dev	±0.1115	±0.1157	±0.1091	

Table 24: The change of the mean R^2 Score and its standard deviation (std dev) for the regression task within the different communities when using the different balancing methods. We have printed the highest scores for each community in bold. The RanOS enhanced the score within all communities, except for Riyadh. The highest increase we have seen in Al Jafr, where the smallest number of users are used, but because of the high standard deviations, no significant difference when using a sampling method can be detected. As we have seen in Section 3, using balancing methods can also have side effects. That is why we will have a look at the change of the MSE and MAE when using these methods in the following.

As we have seen in Section 3, using balancing methods can also have side effects on the one hand, which can improve the prediction results in case of alleviating the, e.g., R^2 Score. On the other hand, a higher Mean Squared Error (MSE) in special areas, e.g., in areas where we had too few samples before balancing could be visible. To get a look at the side effects of the balancing methods, we have investigated the change of the MSE and Mean Absolute Error (MAE) for the regression task. The results of this are shown in Tab. 25 for the RanUS and Tab. 26 for the RanOS.

In Tab. 25, we show the deltas of MSE and MAE of the models with and without balancing by RanUS. As we can see, all deltas are negative and therefore, represent an increasing MSE and MAE, which confirms the results from Tab. 24, that using the RanUS deteriorates the results of the regression task and is, therefore, not sensible for the regression task.

For the RanOS, we have calculated these deltas in Tab. 26. Because of the positive deltas, we can see also an improvement for the MAE and MSE, which can especially be seen in the example Al Jafr. This confirms our findings from Tab. 24, that the RanOS improves our model and is, therefore, sensible for the classification task.

	MSE			MAE		
community	**without**	**RanUS**	**Δ**	**without**	**RanUS**	**Δ**
Riyadh	-199,319,455	-213,159,831	+13,840,376	-6,992.9	-7,427.1	+435.1
std dev	±10,173,116	±9,841,031		±45.7	±55.2	
Jeddah	-166,686,375	-172,408,347	+5,721,972	-6,769.6	-6,991.7	+222.1
std dev	±8,650,275	±10,306,674		±113.1	±91.9	
Mecca	-217,252,687	-218,420,066	+1,167,379	-7,903.3	-7,935.9	+32.6
std dev	±17,252,794	±21,702,672		±51.8	±70.2	
Al Bahah	-222,141,567	-225,832,997	+3,691,430	-8,274.2	-8,351.1	+76.9
std dev	±12,997,544	±13,069,353		±180.6	±159.7	
Al Jafr	-759,772,901	-784,380,088	+24,607,187	-17,594.0	-17,882.2	+288.2
std dev	±485,134,990	±470,019,128		±4327.1	±3930.6	

Table 25: The difference between imbalanced and random undersampled model by MSE and MAE and their standard deviation (std dev), which shows, that the RanUS also decreases the MSE and MAE beside the R^2 Score, as seen in Tab. 24. This shows us, that the RanUS does have just negative impacts on the metrics.

	MSE			MAE		
community	**without**	**RanOS**	**Δ**	**without**	**RanOS**	**Δ**
Riyadh	-199,319,455	-199,274,403	-45,052	-6,992.9	-6,991.7	-1.2
std dev	±10,173,116	±10,048,456		±45.7	±46.0	
Jeddah	-166,686,375	-165,789,879	-896,496	-6,769.6	-6,754.5	-15.1
std dev	±8,650,275	±8,931,702		±113.1	±114.1	
Mecca	-217,252,687	-216,271,482	-981,205	-7,903.3	-7,891.8	-11.5
std dev	±17,252,794	±17,168,107		±51.8	±59.0	
Al Bahah	-222,141,567	-221,765,338	-376,229	-8,274.2	-8,260.0	-14.2
std dev	±12,997,544	±12,981,777		±180.6	±184.3	
Al Jafr	-759,772,901	-751,360,947	-8,411,954	-17,594.0	-17,368.5	-225.5
std dev	±485,134,990	±499,346,128		±4327.1	±4673.4	

Table 26: The difference between imbalanced and random oversampled model by MSE and MAE and their standard deviation (std dev), which shows, that the RanOS also improves the MSE and MAE beside the R^2 Score, as seen in Tab. 24. This means, that the RanOS does not have any negative impacts in case of the both regression metrics.

6.3 Random Forest Evaluation

After detecting the best working data pre-processing methods, we will now display the scores of the RF for the regression and classification task, when using all features and the presented feature subsets from Section 4. For this, we will again have a look at the already seen communities, but also at the country model, to see for each step, if we can also build a model that works well for the whole country.

6.3.1 Regression Model

We will start the investigation of the regression task by looking at the general subsets of *all*, *user* and *community* features. The data were imputed by *mean* and balanced with the RanOS. The different models will be evaluated by the mean R^2 Score and its standard deviation (std dev). All models need to exceed the baselines in Tab. 27, where the *Dummy Regressor* from SciKit-Learn, which predicted the mean and was oversampled by RanOS, was used. The results of this, are shown in Tab. 28.

community	**Riyadh**	**Jeddah**	**Mecca**	**Al Bahah**	**Al Jafr**	**Country**
Baseline	-0.00032	-0.00046	-0.00036	-0.00056	-0.02325	-0.0002

Table 27: The baselines of the different communities, calculated by *Dummy Regressor* from SciKit-Learn, always predicted the mean and used the RanOS.

This shows that the RF yields the best score for the community models in Riyadh when using just the *user* features with a score of 0.9740 (±0.001). What can also see, that the score is decreasing when training on a smaller number of users, but still yields a score of 0.8250 (±0.1157) for just 174 users in Al Jafr. The country model outperformed the community models with an R^2 Score of 0.9826 (±0.0002) when using the user features, but also did not exceed the score of Riyadh when using the community features.

Another significant finding is the strong decrease of the score when using just the community features, with a maximum loss of 0,4494 in Jeddah, but also the remaining strong score when using just the user features, in contrast to using all features. This leads to the finding, that the user features yield more and better information for predicting churning users and therefore, in general work better.

As the second step, we looked at the performance change when changing the time-depended feature subsets within the different communities and for the country model. This will give us information after what time we can predict the *active minutes* of a user by which accuracy. For this, we will first have a look at predicting all users in Tab. 29 and after that at the results when predicting on these users, who were active for equal or longer than the lower bound of the time window in Tab. 30.

community	**all**	**user**	**community**	*#users*
Riyadh	0.9728	**0.9740**	0.6005	283,741
std dev	±0.0013	±0.0010	±0.0032	
Jeddah	0.9668	**0.9682**	0.5174	100,654
std dev	±0.0017	±0.0020	±0.0046	
Mecca	0.9553	**0.9573**	0.5217	44,622
std dev	±0.0035	±0.0027	±0.0038	
Al Bahah	0.9458	**0.9479**	0.6127	10,942
std dev	±0.0031	±0.0037	±0.0307	
Al Jafr	**0.8250**	0.8213	0.4056	174
std dev	±0.1157	±0.1303	±0.1441	
Country	0.9819	**0.9826**	0.5763	1,012,619
std dev	±0.0002	±0.0002	±0.0014	

Table 28: The mean R^2 Scores and its standard deviation (std dev) for the regression task within the different communities and the country for the feature subsets *all*, *user* and *community*. As we can see, the highest scores can be achieved when using just the *user* features, except for Al Jafr, with a maximum in Riyadh with R^2 = 0.9740 for the community models. Using only the *community* features will strongly decrease our score. The country model achieved a higher score than the community models with a score of 0.9826 when using the user features and did not exceed the best score of the community models when using the community features. This leads to the finding, that community behaviour is not that significant for predicting the churn of a user.

As we can in Tab. 29, the mean R^2 Scores of the different communities are enhancing, when increasing the feature time window, up to a score of 0.8102 (±0.0018) in Riyadh when observing for 3 months. This can also be seen in Fig. 33, where we have given the time window subset on the x-axis and the mean R^2 Score and its standard deviation on the y-axis. We can also see that the prediction scores after one day of observation are relatively low and a prediction after one day would result in a relatively low R^2 Score. The country model also yielded the highest score for this task in comparison to all others when observing for three months, but also did not exceed the models of Riyadh and Al Bahah when using a lower observation window.

	Regression					
community	**Day**	**Week**	**2 Weeks**	**Month**	**3 Months**	*#users*
Riyadh	0.6678	0.6910	0.7047	0.7317	**0.8102**	283,741
std dev	±0.0039	±0.0039	±0.0033	±0.0026	±0.0018	
Jeddah	0.5919	0.6294	0.6496	0.6880	**0.7918**	100,654
std dev	±0.0031	±0.0014	±0.0021	±0.0037	±0.0048	
Mecca	0.5920	0.6189	0.6398	0.6750	**0.7804**	44,622
std dev	±0.0051	±0.0071	±0.0060	±0.0065	±0.0050	
Al Bahah	0.6817	0.6972	0.7096	0.7355	**0.8022**	10,942
std dev	±0.0205	±0.0167	±0.0176	±0.0105	±0.0097	
Al Jafr	0.5274	0.5371	0.5541	0.5790	**0.6301**	174
std dev	±0.1335	±0.1187	±0.0706	±0.0962	±0.0775	
Country	0.6445	0.6715	0.6893	0.7217	**0.8189**	1,012,619
std dev	±0.0018	±0.0019	±0.0015	±0.0010	±0.0015	

Table 29: The mean R^2 Scores and its standard deviation (std dev) within the different communities and the country for the 5 time window feature subsets. As we can see, the score is enhancing when increasing the time window, which can also be seen in Fig. 33. The highest score can be achieved in Riyadh with an R^2 Score of 0.8102 for the community models, which gets exceeded with a score of 0.8189 for the country model, which, therefore, yielded a lower score for the smaller time windows than 3 months in comparison to Riyadh and Al Bahah. This shows us, that we can predict the users as churners best when observing for three months.

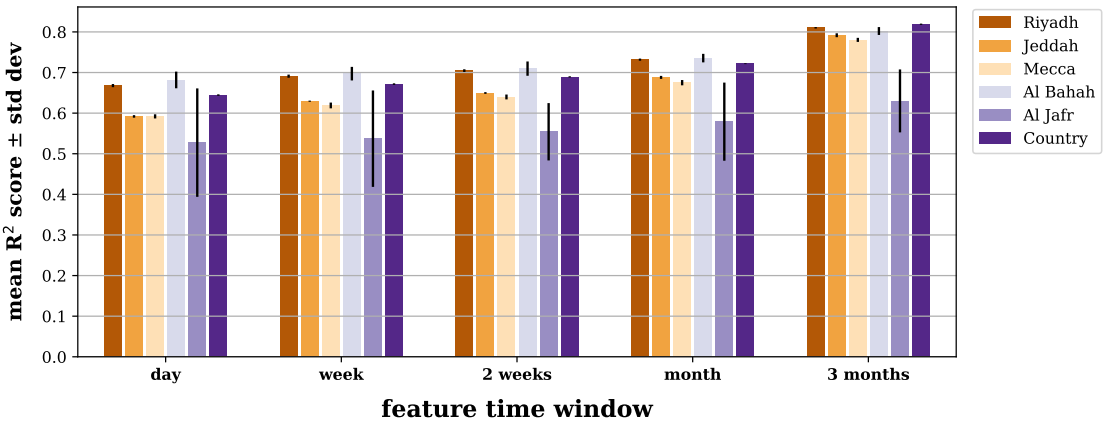

Figure 33: The visualised mean R^2 Scores of the communities for the different time window feature subsets and their standard deviation (std dev). As we can see, the increasing of the time window also increases the R^2 Score. We can also see, that the standard deviation is relatively low for all communities, except for Al Jafr where we have a high standard deviation of up to ±0.1335.

Time Window Features. In Section 4.3, we have described the cumulated time window features, where, e.g., the features of the one-day time window are also part of the three months time window. Because of users that were active for a smaller time than the time window, the features of the greater window would result in the same value as the features of the smaller window, e.g., for the features *postcreated_24h* and *postcreated_3_Months*, where a user with an active time of one day and five created posts would have the same feature value for both. Because of this, these features could gain too much information for longer time windows and could result in a skew, which can be seen by the increasing R^2 Score when increasing the time window in Tab. 29. To observe this impact, we have also calculated the mean R^2 Score and its standard deviation (std dev) for the users who were active for minimally the time of observation for the community models. These results are shown in Tab. 30 and visualised in Fig. 34, where we again have given the time window subsets on the x-axis and the R^2 Score and its standard deviation on the y-axis.

	Regression | time window			
community	**Day**	**Week**	**2 Weeks**	**Month**
Riyadh	0.6380	0.6404	0.6486	**0.6752**
std dev	±0.0038	±0.0039	±0.0046	±0.0040
Jeddah	**0.5504**	0.5395	0.5393	0.5444
std dev	±0.0035	±0.0080	±0.0093	±0.0120
Mecca	**0.5568**	0.5494	0.5447	0.5490
std dev	±0.0064	±0.0074	±0.0072	±0.0061
Al Bahah	0.6730	0.6897	0.6868	**0.7005**
std dev	±0.0176	±0.0154	±0.0171	±0.0146
Al Jafr	**0.5123**	0.4285	0.3521	0.4113
std dev	±0.1201	±0.1808	±0.3214	±0.2752

Table 30: The mean R^2 Scores and its standard deviation (std dev) within the different communities, when predicting users, who were minimally active as long as the observation time. As we can see, the progress of the score when changing the observation window has changed from the observation before in Tab. 29. We cannot see a clear increase and therefore, a more balanced score for all subsets, which can also be seen in Fig. 34.

As we can see, the mean R^2 Scores of the different communities have changed and are more balanced, except for Al Jafr. We cannot see the improvement when increasing the observation window like we have seen before in Fig. 33. This supports the assumption, that the number of users with fewer active minutes than the observation window results in a bias. This results in the highest score in Al Bahah of 0.7005 (±0.0146) when observing for three months and has decreased by 0.1017 in comparison to the result of Al Bahah in the observation before (cf.

Tab. 29).

Another finding of this is the volatile score of Al Jafr, which differs strongly and also has a huge standard deviation, with a maximum of 0.3214 when observing two weeks. This shows us that we have a high scattering of the R^2 Score in Al Jafr, which means that our prediction for Al Jafr does not have a guaranteed high or low prediction power, because of the high standard deviation.

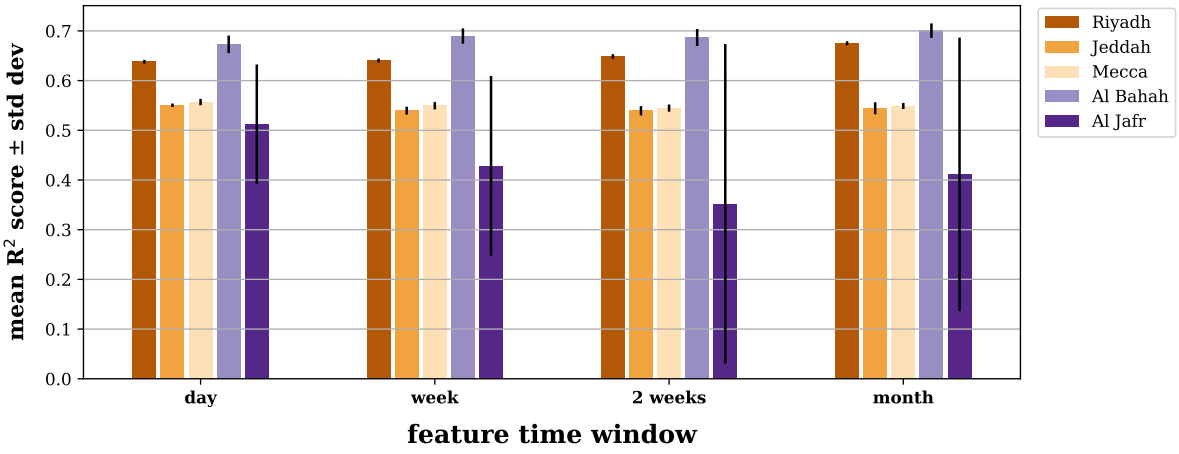

Figure 34: The mean R^2 Score and its standard deviation (std dev) when predicting on users with active minutes greater or equal than the time window, to remove the bias, that results of users who were less active than the given time window and therefore, result in repeating values for the further window features.
As we can see, the scores are much more balanced than before (cf. Fig. 33), where we cannot see the increase of the score when increasing the window. The results of Al Jafr differ from the others in the case of the volatile score and a huge standard deviation, as we can especially see for the observation time of two weeks.

6.3.2 Classification Model

The next step of the RF investigation was the performance of the classification tasks. The method will have the task to predict the churning class of a user, as we have described in Tab. 6. We will distinguish the classification into multi-label and binary classification. The data were imputed by *most frequent* and we will evaluate the models by mean *accuracy* and its standard deviation (std dev).

6.3.2.1 Multi-Label Classification

As a first step, we looked at the multi-label classification, where the model will predict one of the six churn classes for a user. For this we have started with the feature subsets *all*, *user* and *community* within the different communities and for the country model. The models will need to exceed the baselines we have defined in Tab. 31, where we used the *Dummy Classifier* from SciKit-Learn, predicting always the most frequent label. The results for this task are shown in Tab. 32

We can achieve a relatively high mean accuracy (over 0.9) in Riyadh, Jeddah, and Mecca when using *all* and *user* features, and just in Al Bahah and Al Jafr is the accuracy falling back, which could be derived by a smaller number of samples, which gain less information and therefore,

community	**Riyadh**	**Jeddah**	**Mecca**	**Al Bahah**	**Al Jafr**	**Country**
Baseline	0.3335	0.2554	0.2642	0.2736	0.3104	0.2818

Table 31: The baselines of the different communities for the multi-label classification, calculated by *Dummy Classifier* from SciKit-Learn, which always predicted the most frequent label.

lead to a poorer generalisation. As for the regression task, also the user features are the strongest with higher accuracy than using all features, except in Al Jafr. The accuracy is also decreasing strongly when using just the community features. In this case, Al Bahah results in the best accuracy with 0.5667 (±0.0052). This leads us to the same finding as for the regression task: The user features are the strongest and the impact of the community features for predicting churn is relatively low. This can also be seen for the country model, which outperformed the other models when using all and the user features, but also did not yield the highest score when using the community features.

Multi-label Classification

community	**all**	**user**	**community**	*#users*
Riyadh	0.9625	**0.9641**	0.5521	283,741
std dev	±0.0007	±0.0005	±0.0009	
Jeddah	0.9436	**0.9464**	0.5264	100,654
std dev	±0.0008	±0.0007	±0.0035	
Mecca	0.9231	**0.9271**	0.5285	44,622
std dev	±0.0039	±0.0029	±0.0024	
Al Bahah	0.8885	**0.8933**	0.5667	10,942
std dev	±0.0060	±0.0057	±0.0052	
Al Jafr	**0.7240**	0.7239	0.5113	174
std dev	±0.0694	±0.0923	±0.0821	
Country	0.9697	**0.9711**	0.5444	1,012,619
std dev	±0.0004	±0.0005	±0.0008	

Table 32: The mean accuracy and its standard deviation (std dev) of the different communities and the country for the classification task, when using *all*, *user* and *community* features. The data were imputed by *most frequent*. As we can see, the usage of user features achieves the highest accuracy in all communities, except for Al Jafr. Using just the community features decreases the accuracy extremely, which shows us that the multi-label classification task is not sensible for the community features and would, therefore, yield weak results. This leads us to the assumption that the behaviour of the community probably does not have a huge impact on the retention of a user, which will be investigated in more detail in the following sections. Using only the user features results in the best mean accuracy. For the country model, we can see, that it exceeds all other models when using the user features but also did not exceed the models of Riyadh and Al Bahah when using the community features.

As a second step for the classification task, we looked at the time window features and the change of accuracy when increasing the observation time. This will give us a deeper look at what time we can predict the churning class of a user by which accuracy. The results of this are shown in Tab. 33 and are visualised in Fig. 35, where the time window subsets are given on the x-axis and the mean accuracy and its standard deviation on the y-axis.

As we can see in Tab. 33, the accuracy is enhancing when increasing the feature time window, up to an accuracy of 0.7910 (±0.0021) in Riyadh, when observing for 3 months. This can also be seen in Fig. 35, where the accuracy is increasing when increasing the time window. Another finding is the deficit of Al Jafr in all cases of around -0.1, which again could be explained by the small number of samples, which does not gain enough information to create a more generalised model. The country model outperforms all other models for all time windows, which shows that using the country model yields the best results for the classification task with a maximum of 0.8169 when observing for three months, in contrast to the regression task.

For the regression task, we have calculated the accuracy when using only the users who were minimally active for the given time of the window. This observation cannot be done for the classification task, because of the decreasing number of classes when increasing the time window. This limits the observation to one month as maximum, because of the single remaining class within users. This results in a bias, because of this volatile number of predictable classes for different time windows. That is why the calculated accuracies would not be directly comparable and would, therefore, also not result in the desired conclusions. We have, therefore, left this part for the classification task.

	Multi-label Classification					
community	**Day**	**Week**	**2 Weeks**	**Month**	**3 Months**	*#users*
Riyadh	0.5648	0.6189	0.6515	0.6999	**0.7910**	283,741
std dev	±0.0016	±0.0013	±0.0015	±0.0015	±0.0021	
Jeddah	0.5329	0.6011	0.6386	0.6936	**0.7827**	100,654
std dev	±0.0041	±0.0029	±0.0025	±0.0031	±0.0016	
Mecca	0.5324	0.5962	0.6346	0.6822	**0.7707**	44,622
std dev	±0.0022	±0.0035	±0.0029	±0.0059	±0.0054	
Al Bahah	0.5755	0.6172	0.6547	0.6909	**0.7289**	10,942
std dev	±0.0044	±0.0064	±0.0096	±0.0081	±0.0121	
Al Jafr	0.4770	0.5114	0.5229	0.5750	**0.6151**	174
std dev	±0.0379	±0.0810	±0.0826	±0.0593	±0.0483	
Country	0.6086	0.6620	0.6968	0.7422	**0.8169**	1,012,619
std dev	±0.0007	±0.0008	±0.0011	±0.0009	±0.0011	

Table 33: The change of the mean accuracy and its standard deviation (std dev) within the different communities and the country for the classification task when imputing by *most frequent*. For each time window feature subset, we are predicting for all users within the community/country. As we can see, the score is enhancing, when increasing the time window and all models can exceed the baseline. We achieved the highest score in Riyadh with an accuracy of 0.7910 when observing for 3 months for the community models and a higher score of the country model with 0.8169 when observing for 3 months, which also exceeded the scores of all communities for the other time windows.

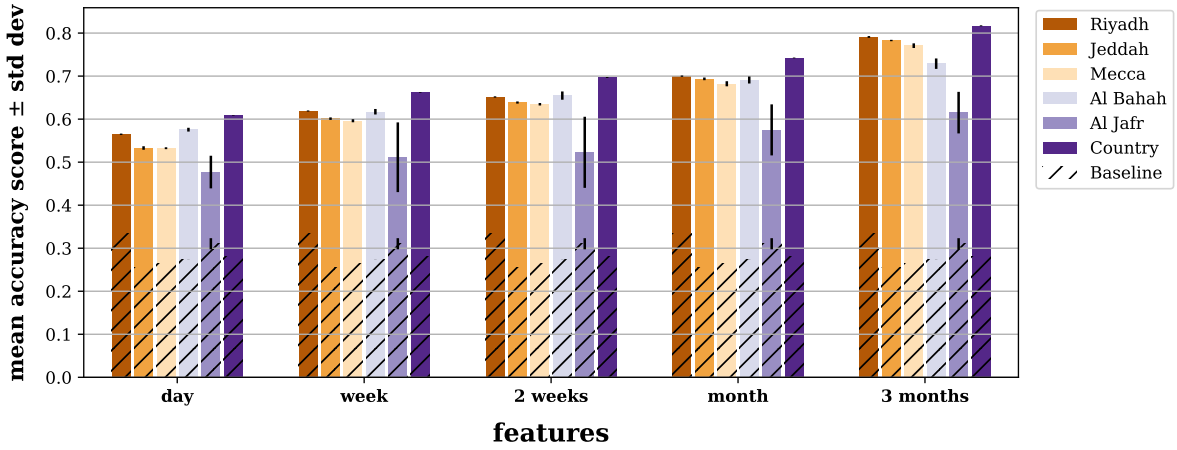

Figure 35: The change of the mean accuracy and its standard deviation (std dev) within the different communities, when changing the feature time window. As we have seen for the regression task (cf. Fig. 33), the increasing time window also increases the accuracy in all communities with a relatively low standard deviation, except for Al Jafr, where the standard deviation is much higher.

6.3.2.2 Binary Classification

After we have looked at the multi-label classification, which did not perform well on the time window subsets, we will now investigate the binary classification. For this, the model will have to predict one class or another. This represents the more practical case, that a company uses the model to predict if a user will leave the platform within a given time. This also makes the prediction task or the predictor easier. Each of the following models needs to exceed the baselines, we have defined in Tab. 34, where we have used the *Dummy Classifier* from SciKit-Learn, that always predicted the most frequent label. In Tab. 35, we have shown the results of these tasks for the five communities and the country model. These were calculated on *all* features and imputed by *most frequent*. Each column displays the prediction of a user to a class of lower or equal to the given time or greater.

Binary Classification Baseline					
community	**one day**	**one week**	**two weeks**	**one month**	**three months**
Riyadh	**0.8754**	0.7670	0.7008	0.5889	0.6665
Jeddah	**0.8415**	0.7121	0.6337	0.5093	0.7461
Mecca	**0.8521**	0.7235	0.6439	0.5136	0.7506
Al Bahah	**0.8574**	0.7098	0.6121	0.5419	0.8155
Al Jafr	**0.8161**	0.7012	0.6380	0.5296	0.7931
Country	**0.8665**	0.7451	0.6709	0.5470	0.7182

Table 34: The imbalanced baselines for the different communities and the country when processing a binary classification with the given threshold. The classification tasks with the highest accuracy per community are printed in bold. We used the *Dummy Classifier* from SciKit-Learn that always predicted the most frequent.

We can see in Tab. 35 and Fig. 36, that the highest mean accuracy can be achieved when using one day as the class border, where we achieve a high score of over 0.97 in all communities, but also a relatively high score for the other tasks. These weaker results could be derived by the smaller number of users for the greater class which leads to less information and therefore, could also decrease the precision and recall. The country model again outperformed the community models for all five binary classification tasks with a maximum accuracy of 0.9970 (±0.0001) for the one-day border. This shows us, that the RF is also very sensible for the binary classification task within the different communities, but also that the country model outperforms the community models for all classification tasks.

	Binary Classification					
community	**Day**	**Week**	**2 Weeks**	**Month**	**3 Months**	*#users*
Riyadh	**0.9966**	0.9953	0.9936	0.9910	0.9830	283,741
std dev	±0.0002	±0.0002	±0.0003	±0.0003	±0.0004	
Jeddah	**0.9944**	0.9924	0.9902	0.9869	0.9779	100,654
std dev	±0.0003	±0.0003	±0.0005	±0.0004	±0.0007	
Mecca	**0.9923**	0.9903	0.9860	0.9803	0.9701	44,622
std dev	±0.0007	±0.0007	±0.0007	±0.0019	±0.0012	
Al Bahah	**0.9881**	0.9808	0.9715	0.9687	0.9720	10,942
std dev	±0.0021	±0.0054	±0.0036	±0.0053	±0.0021	
Al Jafr	**0.9714**	0.9197	0.9313	0.9370	0.9366	174
std dev	±0.0313	±0.0418	±0.0497	±0.0419	±0.0219	
Country	**0.9970**	0.9961	0.9955	0.9939	0.9893	1,012,619
std dev	±0.0001	±0.0002	±0.0001	±0.0001	±0.0003	

Table 35: The mean accuracy and standard deviation (std dev) of the binary classification of the RF on all features and imputed by *most frequent*. The user gets predicted to the class which contains users with an active time lower or equal to the given time or to the class with users with a greater active time.
The highest mean accuracy can be achieved when using one-day as the class border and the accuracy is not strongly decreasing for the other tasks. We can also see that the country model outperforms all other models.

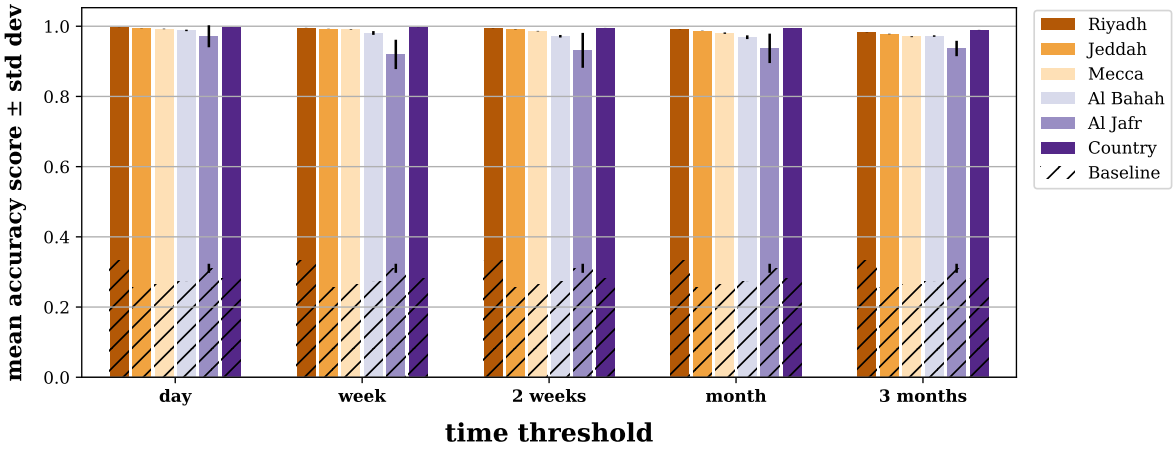

Figure 36: The change of the mean accuracy and its standard deviation (std dev) within the different communities for the binary classification, when changing the time threshold, which represents the time until a user will churn. This enhances the accuracy extremely up to over 99% for the country model and the three big communities Riyadh, Jeddah, and Mecca. This shows that the models are very sensible for the binary classification task.

Conclusion

In both tasks, using only the community features has decreased the score extremely. This leads us to the conclusion, that the community features are insufficient for a good prediction of user churn, but the community probably does have an impact on the user churn. This could follow by a bad representation of the community by the community features and can be more seen when observing the feature importance in Section 6.5.2. There we will specify if there are community features that are important for our predictor and how important they are.

As another main finding, we have seen, that the score is enhancing with increasing observation time. In this context, we have also looked at the skew of the cumulative time window features and the predictions when predicting on users with a minimal active time of greater or equal to the specific time window for the regression task. This had an impact on the resulting R^2 Scores, which were more balanced across all time windows and not increasing when using a bigger time window.

For the classification task, an approach for multi-label classification was shown, which resulted in strong scores for the user features, but low scores for the time window subset analysis. That is why we have also shown an approach for binary classification. We have seen a high accuracy for the different tasks and that the RF also has a possible usage of the models with strong predictive power.

In the proceedings, we have also observed the performance of a country model in comparison to the specific community models. For this we have seen that the country model outperformed the community models for all classification tasks and only for the regression task when using all, the user and the three months feature subsets. This has shown us, that for the regression task and therefore, for predicting the specific active time, the country model seems not to be the strongest. This we will investigate more in Section 6.5.1, where we will have a look at which models work well on which communities and if there is a model that can perform well on the whole country as well as on the specific communities.

6.4 Sweet Spot Detection

When tuning the parameters of the RF, finding a *sweet spot* is important for the daily usage of a 'perfect' model. This needs a trade-off between the accuracy of the model and the fitting time, to predict as strong as possible, but also as fast as possible. As an example, the community of Jeddah is given, which achieved a mean R^2 Score of 0.9668 ($\pm$0.0017) for the regression task, with 2048 estimators and a maximum depth of 32 and a mean fitting time of around 3979 seconds ($\pm$212 sec) $\approx$ 66 minutes. For the classification, it achieved a mean accuracy of 0.9437 ($\pm$0.0007), with 2048 estimators and a maximum depth of 32 and a mean fitting time of around 1982 seconds $\approx$ 33 minutes ($\pm$125 sec).

We first looked at the correlation between the R^2 Score and its standard deviation on the y-axis, and the number of estimators, or rather trees in the forest, on the x-axis for the maximum depth of the trees as different coloured lines, as shown in Fig. 37.

As a second step, the mean fitting time and its standard deviation are given on the y-axis and the two parameters as before on the x-axis and legend, to find a trade-off between score and fitting time, as shown in Fig. 38.

The scorings of the regression are shown in Fig. 37. This shows an increase of the *max. depth* of the trees can increase the score significantly. Especially the maximum depth of 16 and 32 achieved the highest scores and differed of just 0.0027 and could, therefore, be our *max. depth* parameter value for the sweet spot. Another finding is the missing significant improvement when using more than 32 *estimators*, which leads us to a sweet spot of 32 estimators.

To see if we can use the lower depth of 16, we will look at the impact of the maximum depth on the fitting time, as shown in Fig. 38. The fitting time of the RF with a maximum depth of 16 is lower when using less or equal than 1024 estimators and nearly equal when using 16 ($\pm$7 sec) or 32 ($\pm$2 sec) estimators. Because of this small fitting time difference, we can set our sweet spot at 32 estimators with a depth of 32, because of the higher score. This results in a mean R^2

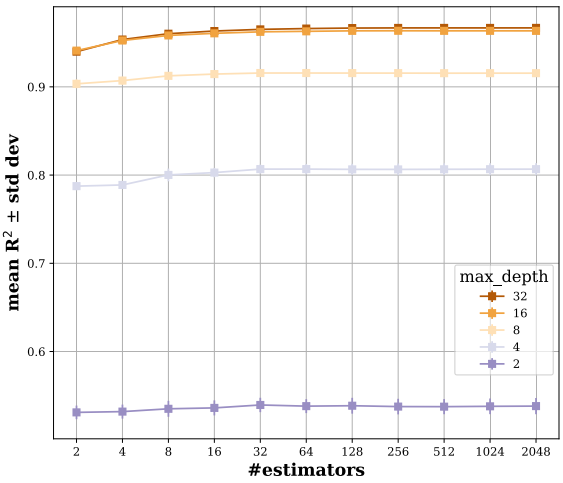

Figure 37: The improvement of the mean R^2 Score and its standard deviation (std dev) when changing the number of estimators and the maximum depth of the trees of the RF for the regression task in Jeddah. This shows us, that we do not achieve a significant score improvement when using more than 32 estimators and a higher score when increasing the maximum depth. The maximum depth of 16 and 32 yielded the best results and differ of just 0.0027 at 32 estimators.

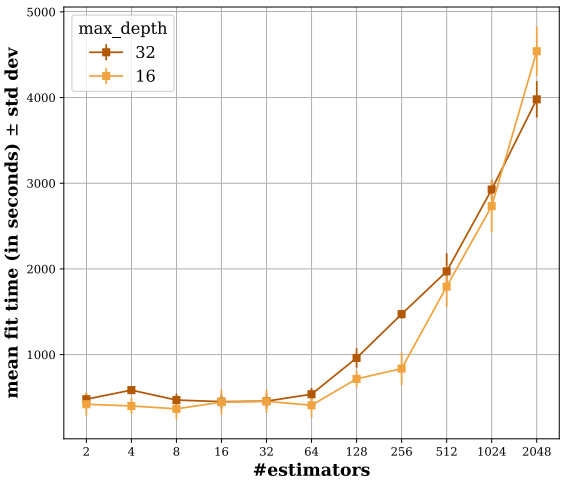

Figure 38: The change of mean fit time in seconds and its standard deviation (std dev) when changing the number of estimators and the maximum depth of the trees of the RF for the regression task in Jeddah. As we can see, the fitting time when using a depth of 16 is lower than for 32 when using lower or equal than 1024 estimators and a nearly equal time when using 16 ($\pm$7 sec) or 32 ($\pm$2 sec) estimators. This results in a sweet spot when using 32 estimators and a maximum depth of 32, with a mean R^2 Score of 0.9651 ($\pm$0.0016) and a mean fitting time improvement of 0.98 hours $\approx$ 59 minutes, and a score decrease of 0.0027.

Score of 0.9651 (±0.0016) of the sweet spot and therefore, a decrease of 0.0017, but also a mean fitting time improvement of 0.98 hours ≈ 59 minutes.

For the classification task, the results are shown in Fig. 39 and 40, where the axes display the same as for the regression task, with the accuracy instead of the R^2 Score. As we can see in Fig. 39, the accuracy of the RF is not significantly increasing, when using 256 or more estimators. Similar to the regression task, increasing the maximum depth also significantly increases the accuracy, and the depth of 16 and 32 again achieve the highest accuracy and differ just 0.0077 when using 256 estimators. As we have done for the regression task, we have also looked at the fitting time of both depths, to make a trade-off, as shown in Fig. 40.

As we can see in Fig. 40, the depth of 16 has a lower fitting time than a depth of 32 when using less or equal than 1024 estimators. When we have a look at the fitting time of both, we have a time advantage of only 25 seconds when using a depth of 16. Because of this small fitting time advantage, but also a small accuracy disadvantage, we can also use a maximum depth of 32 and 256 estimators as a sweet spot. This results in a sweet spot mean accuracy of 0.9333 (±0.0014) with a mean fitting time of 119 seconds (±16 sec), which leads to an accuracy decreasing of just 0.0104 in comparison to the best result and fitting time advantage of approximately 30 minutes.

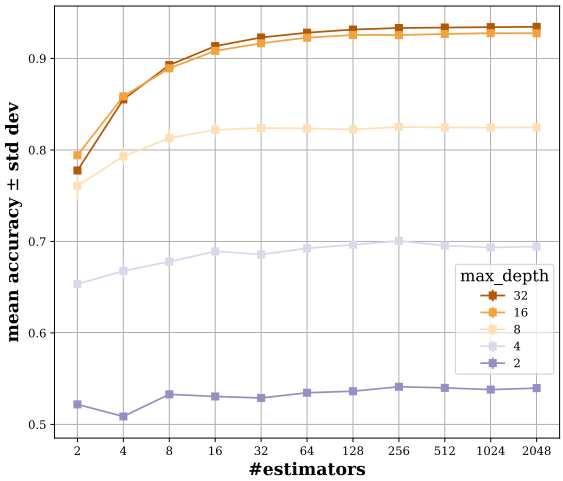

Figure 39: The improvement of the mean accuracy and its standard deviation (std dev) when changing the number of estimators and the maximum depth of the trees of the RF for the classification task in Jeddah. This shows us, that we do not achieve a significant score improvement when using more than 256 estimators. As we have seen for the regression, the accuracy is also significantly increasing when increasing the maximum depth, to a maximum score when using a depth of 16 and 32, which differ by just 0.0077 when using 256 estimators. To make a trade-off, we need to look at the fitting time, which is shown beside.

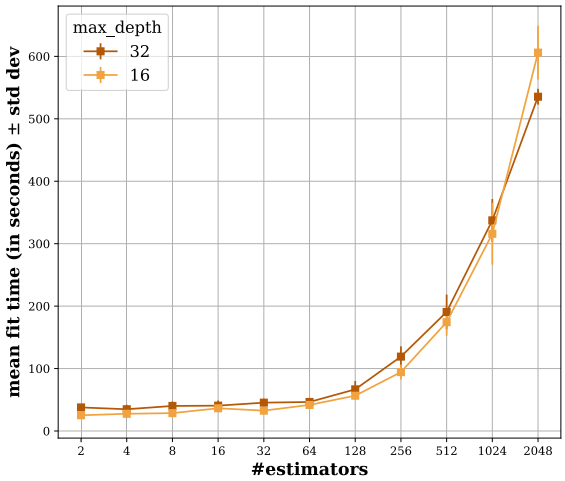

Figure 40: The change of mean fit time in seconds and its standard deviation (std dev) when changing the number of estimators and the maximum depth of the trees of the RF for the classification task in Jeddah. The depth of 16 has a lower fitting time than 32 when using less or equal to 1024 estimators. Both compared depths differ just 25 seconds when using 256 estimators, which leads us to a sweet spot when using 256 estimators and a maximum depth of 32, with an accuracy of 0.9333 (±0.0014) and fitting time of 119 seconds (±16 sec). This decreases the fitting time of about 30 minutes, in comparison to the best result and an accuracy decrease of 0.0104 as a trade-off.

6.5 Detailed View into the Models

After we have detected the best models and got an overview of their performance, we have seen good working models for each community and for the country model.

As a further step, we will now observe the differences and similarities of the different models, starting with cross applications of the models in Section 6.5.1. This will include training of the best models in their community and a prediction of the *active time*/*churn class* of the users from other communities, as well as of the whole country. With this, a generalised view of the models in other communities can be achieved.

We will then run a feature analysis in Section 6.5.2, where we will determine some characteristics of the different regression models. As the last step, we will then use this information and will have a deeper look into the empiricisms of selected features, in Section 6.5.3, for more feedback.

6.5.1 Cross Application

As the first step, we want to have a look at the performance of the best model trained on their community, predicting the *active time* and *user churn class* on users of other communities. This will give us information on which communities could have equal characteristics, as well as show us which model can also work well on different communities.

Regression

We will first have a look at the regression task. These results are shown in Fig. 41, where the best R^2 Score for each community on all features from the grid-search is noted on the diagonal as a baseline for comparison. The model was trained on the community on the x-axis and predicted the *active time* of users from the community on the y-axis. This shows that the group of models of Riyadh, Jeddah, and Mecca achieves a relatively high score within the other communities. The model of Mecca is the only one that achieves a higher score on a community than the specific community model, in Al Jafr. Within both models, we can see that we achieve the smallest decrease of the score in general.

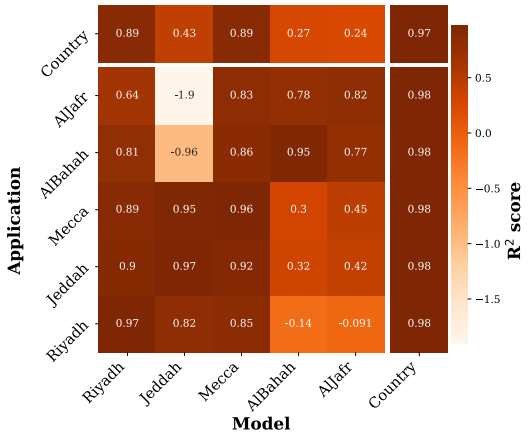

Figure 41: The R^2 Score for the model trained on community A and predicting the *active minutes* of community B. The diagonal shows the best score for the community from the grid-search over all features. The models of Riyadh, Mecca, and the country yielded in all communities a relatively high score and the models of Al Bahah and Al Jafr decreased the scores in Riyadh, Mecca, and Jeddah strongly, with a score below the baseline of Riyadh. Also, the model of Jeddah did not exceed the baselines of AL Jafr and Al Bahah. In general, all models result in a lower, but still strong R^2 Score, except for the model of Mecca for Al Jafr, where we can see a small increase. Especially the country model yielded a high score within all communities, which shows that the country model works better on all communities than the specific model.

The country model outperformed the community models for each task. It achieves the highest R^2 Score within all communities and therefore, also shows us that the country model will work best when predicting the specific *active time* of a user in all communities.

Another special finding is the weak performance of the model Jeddah on the communities of Al Jafr and Al Bahah and the models of Al Bahah and Al Jafr in Riyadh, which resulted in a negative score and also did not exceed the dummy baselines of the communities. The models of Al Bahah and Al Jafr did also achieve a weak score, but still, a positive score when predicting the users in Mecca and Jeddah with a score below 0.5. For all the other models is the R^2 Score decreasing but still relatively high.

Classification

For the classification task, we have shown the results in Fig. 42, where the diagonal again displays the best results for each community from the grid-search. Except for the model of Al Bahah, all models could achieve greater or equal accuracy in Al Jafr than the specific model. We can also see, that the models of Al Bahah und Al Jafr did not work well for the other communities and especially the model of Al Jafr decreases the accuracy strongly, but still exceeds the dummy classifier of 0.3335 in Riyadh. The model of Riyadh yielded the best scores when predicting on the other communities and decreased the accuracy of around 10% in comparison to the actual community model, except for Al Jafr where the model of Riyadh outperformed the model of Al Jafr and enhanced the accuracy of 9%. As we have seen for the regression task, the models of Riyadh, Jeddah, and Mecca worked best when predicting the *user churn* in other communities.

What we have already seen for the regression task is the strong performance of the country model. It again achieved a very high score in all communities, but in contrast to the regression task did not exceed the specific community model, except for Al Jafr, but still yielded stronger results in comparison to all other models. This shows us that the country model did perform best in all other communities and would, therefore, be chosen for predicting many communities over the whole country but not when predicting for a single community.

For both tasks, we can see, that predicting users from another community can still achieve a high prediction score and especially the country model worked best over all communities. This leads us to the finding, that the different communities could have similar behaviour, w.r.t. user

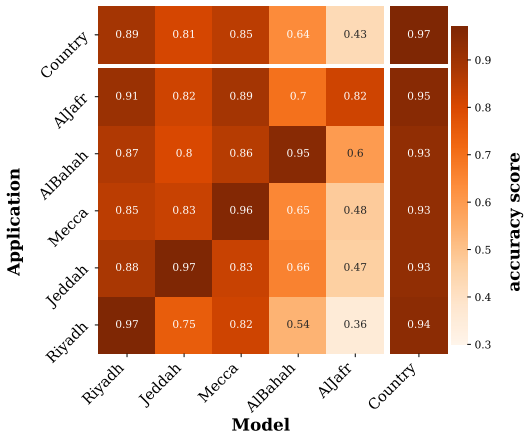

Figure 42: The accuracy for the model trained on community A and predicting the *user churn class* of community B. The diagonal shows the best score for the community from the grid-search over all features. As we can see, the models of Al Bahah and Al Jafr yielded the weakest results, but still exceeded the dummy baselines for the specific community.
Another main finding is the strong performance of the group of models of Riyadh, Jeddah, and Mecca with an accuracy of over 80% within the other communities. The strongest result was achieved by the country model, which performs well in all communities, as we have already seen for the regression task, but does not perform stronger than the specific community models.

churn.

A similar behaviour could be seen when observing the most important features within the models, which then may also be very important for the country model. Within the next section, we will investigate these similarities within the different communities and will try to detect these also within the country model.

6.5.2 Feature Analysis

To detect the similarities and differences between the different models, hence communities, we will now have a look at the features and their importance and rank, calculated by the Random Forest Feature Importance (RFFI) and the ReliefF algorithm (cf. Sec. 2.1), as well as at the selected features of the Recursive Feature Elimination with Cross-Validation (RFECV). For this we will first have a look at the correlations between the importances and most important features of the different models by RFFI and ReliefF, to see if there may also be a similarity. For the models, where we have seen strong correlations, we will then have a look at the specific importance within the communities and detect the most important ones. This will give us a look into the most important features within each community and the similarities between the different models, which could help to derive the different characteristics within the communities and possibly find common features that could also be important for the country model and therefore, could display a common characteristic.

6.5.2.1 Feature Correlations

We will now investigate the Pearson Correlations (cf. Sec. 2.8.1) of the models' feature importance and feature subset importance, as well as at the Spearman's Rank correlations (cf. Sec. 2.8.2) for the feature ranks, to detect which models are strongly correlating and therefore, have models building upon similar features at similar importance and thus, seem to have similar communities with similar behaviour, w.r.t. user churn. Due to the fact, that the correlations of the regression and classification task yielded equal results, the correlations of RFFI and ReliefF of the regression task will be presented in the following.

Random Forest Feature Importance

As the first step, we will have a look at the RFFI. For this, we will observe the correlations between the subsets, the feature ranks, as well as explicit importance.

Subset Correlations. We have first looked at the feature subset correlations, as shown in Fig. 43 by using the Pearson Correlation (cf. Sec. 2.8.1), where each communities' subset importance on the x-axis is set into correlation to each community on the y-axis. For this, we have calculated the cumulated importance of the features for each feature subset in all communities. We have seen, that there is a high correlation within the communities Riyadh, Jeddah, and Mecca, but also with Al Jafr. The correlations of Al Bahah with all communities are standing back and especially with Jeddah we can see a noncorrelation.

For the country model, we can see a high correlation with Jeddah and Al Jafr and a noncorrelation with Al Bahah but also a low correlation with the communities of Riyadh and Mecca. In contrast to the correlations of the single communities, there is less correlation of the country model, which leads us to the finding that the country model differs a little in the case of the subset importance.

Feature Rank Correlation. To get a more detailed look at the specific features, we have also shown the feature rank correlation in Fig. 44, where we have used the Spearman's Rank Correlation (cf. Sec. 2.8.1), where each communities' feature ranks of the x-axis are set into correlation to each on the y-axis. As we can see, Mecca still correlates strongly with Riyadh and Jeddah and also Riyadh and Jeddah have a high correlation. In contrast to the subset correlation, a high correlation between Al Bahah and Jeddah can be seen. This could follow by

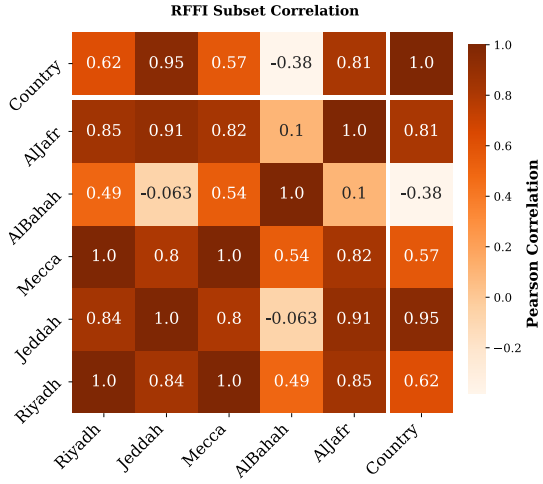

Figure 43: The Pearson Correlation of the cumulated importance of the features for each subset from the RFFI, showing the community and country regression models. The country model has a strong correlation to Jeddah and a bit lower one to Al Jafr. We can also see a very weak correlation to Al Bahah. In contrast to the community models, there is not a high correlation to all other models of the country model, except for Al Bahah. The community models of Riyadh, Jeddah, Mecca, and Al Jafr are highly correlated, but also with the country model, and a noncorrelation between Jeddah and Al Bahah can be seen.

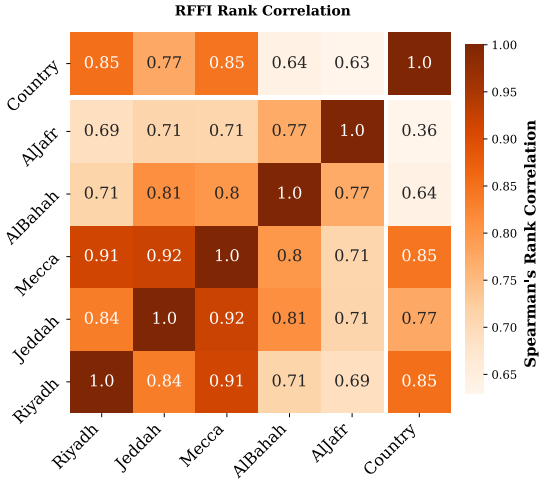

Figure 44: The Spearman's Rank Correlation of the features importance from the RFFI for the community and country regression model. This shows us that there are low correlations between Al Jafr and Al Bahah. We can also see the highest correlation of only 0.8534 in Riyadh with the country model, which shows that the feature ranks of the country model are not very strongly correlated to these of the community models. For the community models, the strong correlations between Riyadh, Jeddah, and Mecca can be confirmed and in contrast to the subsets, a high correlation between Jeddah and Al Bahah can be seen.

a relatively equal ranking of the feature, but a high difference when having a look at the specific importances, which then also have an impact on the cumulated subset importance. There is also a lower correlation of Al Jafr with Riyadh, Jeddah, and Mecca but as we have mentioned for Jeddah, a higher correlation with Al Bahah. This supports our findings that there are some similarities within the different communities and especially in Riyadh, Jeddah, and Mecca. For the country model, we can see that there is also a low correlation, especially in Al Bahah and Al Jafr. The highest correlation exists with 0.8534 in Riyadh. This also shows us, that the feature ranks of the country model are not very strongly correlated and differ strongly, except with these of Riyadh.

Importance Correlation. As a final step, we have observed the correlations of the specific importance of a feature, as shown in Fig. 45, by using the Pearson Correlation, where each communities' feature importance on the x-axis is set into correlation to each on the y-axis. The correlations are again decreasing in comparison to the rank correlations. The highest correlations can be seen again for Mecca with Jeddah and Riyadh. The assumption of high differences between the feature importances but similar feature rankings in Jeddah and Al Bahah, which we have done from the subset and rank correlation, can also be confirmed, because of the low correlation for the detailed importances. This shows us that there are still correlations between the different communities, although these are much lower, which could follow by the higher diversity of values for the features importances than for ranks.

The country model and the model of Riyadh are again strongly correlated and this strong correlation can be seen for all of the three observations. We can also see again, that there is a near noncorrelation to Al Bahah and a weak one to Al Bahah and Mecca. Just the correlation to Jeddah is relatively high in addition to Riyadh.

As an overall finding for the RFFI, we can say, that there are high correlations between the different models which could follow by similarities and equal behaviour within the different

communities. We have also seen that the country model did not correlate that strong to the community models, except for Riyad. That shows us that there are features that seem to be important within the communities, but also for the whole country. To confirm this, we will now observe the correlations of the ReliefF.

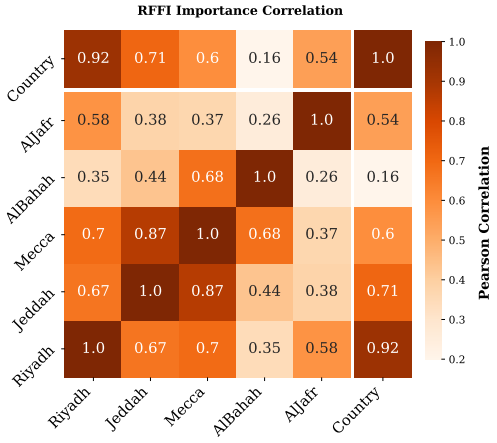

Figure 45: The Pearson Correlations of the specific feature importances from the RFFI for the community and country regression model. This supports the correlation of the country model with the model of Riyadh, because of a high correlation for this task. We can also see that the noncorrelation with Al Bahah can be confirmed. For the community models, the correlations are shrinking but still most strongly in Riyadh, Jeddah, and Mecca, and we can also confirm the assumption of high differences in importance and similarities in ranking between Jeddah and Al Bahah.

ReliefF

After we have observed the feature importance of the RF, we now want to observe these of the ReliefF (cf. Sec. 2.1.2), to see if this leads to equal results and therefore, supports the findings of the RFFI.

Subset Correlations. As we have done for the RFFI, we have first looked at the feature subsets within the different communities. As we can see in Fig. 46, where each communities' subset importance on the x-axis is set into correlation to each on the y-axis, all models have a high correlation, except for Al Jafr with Al Bahah. In contrast to the community models, a weak or rather noncorrelation of the country model to all other models, except for Al Jafr, can be seen, which shows us that for the ReliefF there could be many differences for the feature importance.

Feature Rank Correlation. To have a closer look into the detailed feature importances, we have also looked at their ranking in Fig. 47, where each communities' feature ranks on the x-axis is set into correlation to each on the y-axis. This showed us, that also the ranks of the features are strongly correlating within all models, except Al Jafr. This leads us to the finding, that there could be a range of features that are important within all communities and could, therefore, also gain much information for the country model.

When investigating the country model, the correlations are a big contrast to these of the subset observation. We achieve a high correlation between the country model and the models of Riyadh and Mecca, but also for Jeddah and Al Jafr. Only the model of Al Bahah does not correlate strongly with the country model. This leads us to the finding, that the ranking of the features within the different models seems to be very similar, but differ in their importance. To confirm this we have, therefore, also observed the correlations of the specific importance.

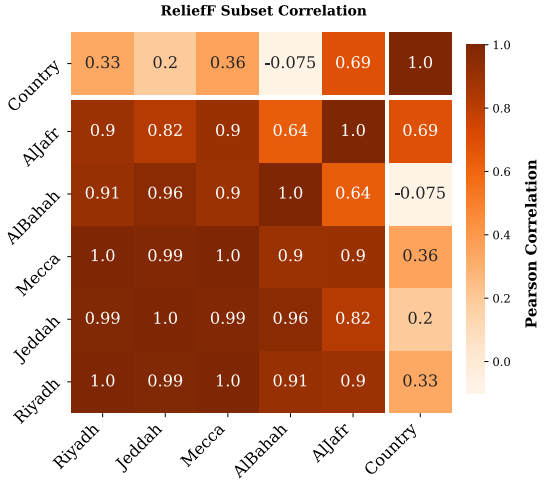

Figure 46: The Pearson Correlations of the feature subset importances of the *ReliefF* for the regression models. We can see a high correlation of all models, except for Al Jafr with Al Bahah. Especially the communities Riyadh, Jeddah, and Mecca a strongly correlating, which leads to the finding, that there are similarities within the different communities.

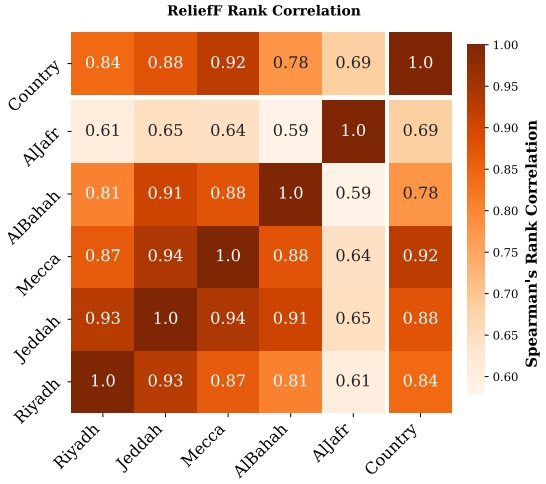

Figure 47: The Spearman's Rank correlations for the *ReliefF* feature ranks for the regression models. There are fewer correlations as for the subsets, but again high correlations between Riyadh, Jeddah, Mecca, and Al Bahah. Al Jafr stands back and has a weak correlation to the other communities. This shows us that the presumed similarities can also be seen for the rank correlation.

Importance Correlation. For the explicit importance, we can see in Fig. 48, where each communities' feature importance on the x-axis is set into correlation to each on the y-axis, that there is just a high correlation within the cities Riyadh, Jeddah, and Mecca, but also a strong correlation between Al Jafr and these three cities, as well as between Al Bahah and Jeddah. The country model has just with Mecca a correlation of greater than 0.9, but still high correlations in Jeddah and Riyadh. This shows us that there are also similarities for the specific importances like we have seen between the community models.

As a final conclusion for the ReliefF, we can say that there are again strong correlations between the different communities, and especially in Riyadh, Jeddah, and Mecca. That leads us to the finding, that there are equally communities with relatively similar behaviour, which could also be seen within the country model, which also had a strong correlation in rank and

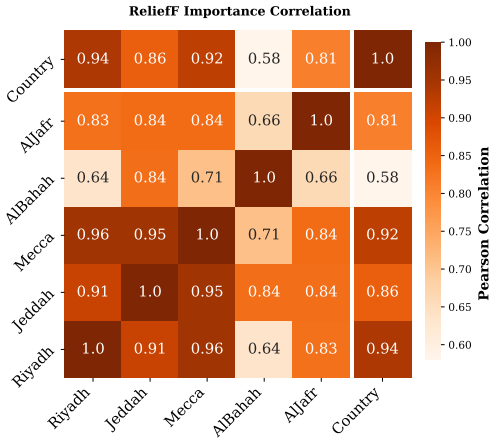

Figure 48: The Pearson Correlations of the feature importances of the *ReliefF*. There are again fewer correlations than for the ranks, but still high correlations within Riyadh, Jeddah, and Mecca. In Al Jafr we can also see a strong correlation to these three models, as well as in Al Bahah with Jeddah. This shows us again, that there could be many similarities within the different communities, which support the presumption of a general model for the whole country.

importance to the cities Riyadh, Jeddah, and Mecca. To specify this behaviour and similarities, we will observe the best and selected features of the different models in detail and will try to detect common features.

Feature Inputvariance

As a last step for the correlations, we have looked at the Spearman's Rank Correlation of the feature rankings of the input variances, as well as on the Pearson Correlation for the specific input variances, as we have shown in Fig. 49. Each communities' input variance (right) and ranks (left) on the x-axis are set into correlation to each community on the y-axis. For this, we have normalised the feature by a Min-Max-Scaler between zero and one.

On the left side, there are strong correlations between the communities of Riyadh, Jeddah, Mecca, and Al Bahah with a value greater than 0.98 for the rank correlation, but also for Al Jafr with values greater than 0.87 with all other communities.

On the right side, we can see for the correlations of the specific input variances, that we have an overall high correlation of the communities with a value greater than 0.9, except for Al Jafr and Al Bahah, where the correlations fall back slightly in Al Bahah. Just in Al Jafr, a noncorrelation to all other communities, except for Al Bahah, can be seen, where just a low correlation can be detected. This shows us that we have high correlations in input variance within the big communities, which supports the findings of the correlations of the RFFI and ReliefF.

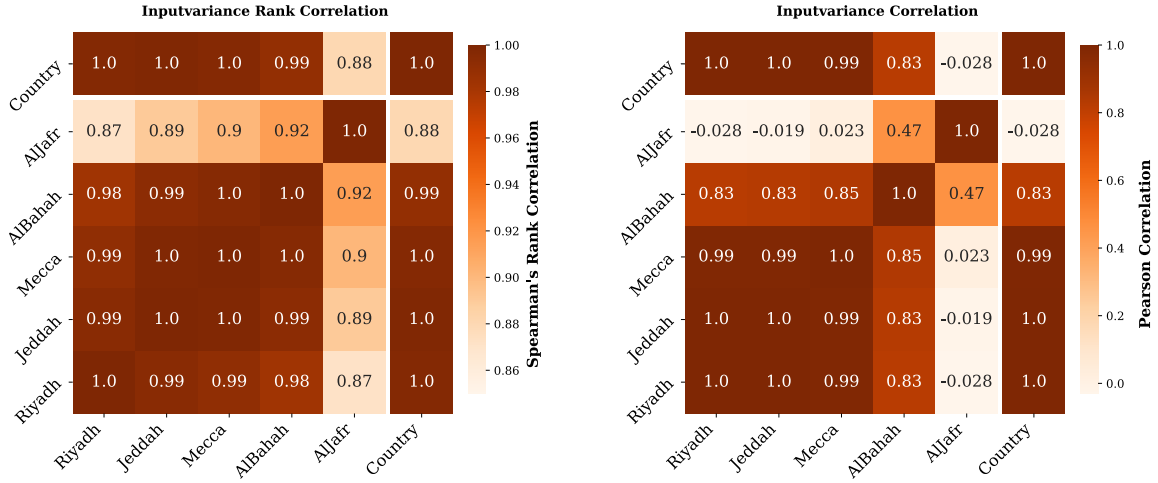

Figure 49: The Spearman's Rank Correlation for the ranking of the feature input variance and the Pearson Correlation for the specific variances for the different communities. On the left, we see an overall strong correlation between the ranks of the different communities, except for Al Jafr, where the correlations to Riyadh and Jeddah are falling below 0.9. On the right side, we can see a difference, where only the communities of Riyadh, Jeddah, and Mecca are strongly correlated with a value greater than 0.98, but also strong correlations of Al Bahah to all other communities with a value greater than 0.8. Just for Al Jafr, we see noncorrelations and a low correlation with Al Bahah.

6.5.2.2 Feature Subset Importances

As we have seen in the previous section, there are strong correlations between the communities for the feature subset importance. These subset importances are shown in Tab. 36, where we have listed the cumulated importance of the features for each subset. For the time window features, we have also given the enhancement for the ReliefF, because of the fact, that the bigger subsets including the features of the smaller ones and are, therefore, equal or more important than the smaller subset.

Riyadh	**generic**	**user**	**community**	**day**	**week**	**2 weeks**	**month**	**3 months**
RFFI	0.1500	**0.6500**	0.1500	0.1500	0.1500	0.1500	0.1500	0.1500
ReliefF	0.3839	**1.5236**	0.7026	0.7303	0.8529	0.9117	1.0498	**1.2979**
enhancement	-	-	-	-	+0.1226	+0.0588	+0.1381	+0.2481

Jeddah	**generic**	**user**	**community**	**day**	**week**	**2 weeks**	**month**	**3 months**
RFFI	0.2400	**0.6500**	0.0600	0.0600	0.0600	0.0600	0.0600	0.0600
ReliefF	0.3760	**1.6271**	0.8748	0.9149	1.0797	1.1476	1.3134	**1.4950**
enhancement	-	-	-	-	+0.1648	+0.0679	+0.1658	+0.1816

Mecca	**generic**	**user**	**community**	**day**	**week**	**2 weeks**	**month**	**3 months**
RFFI	0.1400	**0.6500**	0.1600	0.1600	0.1600	0.1600	0.1600	0.1600
ReliefF	0.3750	**1.7912**	0.7463	0.8151	1.0008	1.0854	1.2717	**1.4726**
enhancement	-	-	-	-	+0.1857	+0.0846	+0.1863	+0.2009

Al Bahah	**generic**	**user**	**community**	**day**	**week**	**2 weeks**	**month**	**3 months**
RFFI	0.0000	0.4200	**0.5000**	0.5000	0.5000	0.5000	0.5000	0.5000
ReliefF	0.8120	2.2679	**2.3937**	2.4246	2.7824	3.0018	3.3834	**3.5782**
enhancement	-	-	-	-	+0.3578	+0.2194	+0.3816	+0.1948

Al Jafr	**generic**	**user**	**community**	**day**	**week**	**2 weeks**	**month**	**3 months**
RFFI	0.0500	**0.7500**	0.1200	0.1200	0.1200	0.1200	0.13	0.3500
ReliefF	0.3387	**2.7870**	0.7993	0.9843	1.1751	1.3065	1.5783	**2.0350**
enhancement	-	-	-	-	+0.1908	+0.1314	+0.2718	+0.4567

Country	**generic**	**user**	**community**	**day**	**week**	**2 weeks**	**month**	**3 months**
RFFI	0.27	**0.71**	0.0000	0.0000	0.0000	0.0000	0.0000	0.01
ReliefF	0.4051	**1.7714**	0.3128	0.3821	0.5781	0.6547	0.8231	**1.0700**
enhancement	-	-	-	-	+0.1960	+0.0766	+0.1684	+0.2469

Table 36: The cumulated feature subset importance for the RFFI and ReliefF within the different communities and in the country and the enhancement for the time-window features for the ReliefF. The user features yielded the highest importance in all models, except for Al Bahah, where the community features outperformed the user features, which may explain the noncorrelation to Jeddah, as well as the low correlation to all other models. We can also see the enhancement for the time-window feature subsets, which follows by their cumulated features, where the smaller window is a subset of the bigger one, as we have described in Section 4.3. That is why we have also given the *enhancement* for the ReliefF importances when increasing the window. Except for Al Bahah, all models yielded the highest improvement for the three months window, which could also be a reason for the correlations of Al Bahah. The stagnating importances for the RFFI can be explained by the calculation to two decimal places and, therefore, a resulting zero importance for many features. The country model shows us, that the community features did not have any importance for the prediction process.

The time-independent feature subsets generic, user and community show us that the user features achieved in all communities the highest importance for both methods, except for Al Bahah where the community features yielded the higher importance for both methods and also the generic features yielded a much lower score. This could be an explanation of the low correlation of the subsets, as we have seen in Fig. 43 and 44, with Riyadh, Jeddah, and Al Jafr.

When we have a look at the time-window feature subsets, we can see stagnating importance for the RFFI in all cities, except for Al Jafr. This follows by a calculation of the importance to two decimal places and therefore, a resulting zero importance for many features. For the ReliefF importances, we can see, an increase of the importance when increasing the window, which follows by the cumulated features, where the smaller windows are subsets of the bigger ones, as we have described in Section 4.3. That is why we have given the enhancement below, which symbolises the importance of each window. This shows us, that the highest enhancement can be seen for three months, except for Al Bahah, which again could be a reason for the noncorrelation.

This has shown us that there is a relatively equal subset importance between the communities. This has only shown us that our assumptions from the previous section can be supported. As a next step, we now want to look at the specific importance and features. This will show us

which characteristics were most important within the different communities by which importance. With this, also similarities and common features can be detected.

6.5.2.3 Most important Features

After we have seen that there are correlations between the feature importances of the models and also how the importance is distributed along with the feature subsets, we now want to have a look at the best features and the number of features of the RFFI, ReliefF, and RFECV.

As a first step, we have observed the number of features that were selected by each method within the different communities, as well as by each method in all communities and the country. This is shown in Tab. 37. For the RFFI, we have counted the features with importance greater than zero, and for the ReliefF, these features with importance greater than 0.056. The importance threshold for the ReliefF is based on the 80% quantile of the feature importances.

There are 15 to 22 features selected by the RFFI, 14 to 26 by RFECV, and 14 to 29 by the ReliefF. This shows us, that we have a relatively equal range of selected features. We can also see, that ten features were selected in all communities by the RFFI, and eight by RFECV and ReliefF. From these features, two were selected by all three methods. In Tab. 38, we can also see in the upper table, that we have three to four features that get selected by both of these methods in all communities.

For the country model we can see in the lower table of Tab. 38, the number of features that were selected by the three methods within the country, as well as by each and two of them. There are 10 to 15 features that were selected by each method and nine features for the RFFI, seven for the RFECV, and six for the ReliefF, that were selected in all communities and for the country as *intersection B* in Tab. 37. From these only one feature was selected by all three methods. In the lower table of Tab. 38, we can also see that there are 4 to 12 features that were selected by two of the methods and four by all three for the country model.

After we have now seen that there are non-empty intersections, we will now look at these features and their specific importance and rank, to see which characteristics we could derive. At first, we want to observe the features that were selected by RFECV in Tab. 39. Within this table, we have shown the features that were selected in all communities, for the country model and in both. Five features were selected for all communities and the country. This shows us

	# selected features		
	RFFI	**RFECV**	**ReliefF**
Riyad	16	14	14
Jeddah	15	24	14
Mecca	15	22	16
Al Bahah	15	26	29
Al Jafr	22	19	23
intersection A	10	8	8
intersect overall		2	
Country	15	19	10
intersection B	9	7	6
intersect overall		1	

Table 37: The number of selected features of the three methods for the communities and the country. *Intersection A* means the number of features that were selected in all communities by the method and *intersection overall* the number of features within all communities and by all three methods. *Intersection B* means the number of features that were selected in all communities as well as for the country by the specific method.

	# selected features communities			
	ReliefF	**RFECV**	**RFFI**	
ReliefF	**8**	4	3	
RFECV	4	**8**	4	
RFFI	3	4	**10**	
all				**2**

	# selected features country			
	ReliefF	**RFECV**	**RFFI**	
ReliefF	**10**	7	4	
RFECV	7	**19**	12	
RFFI	4	12	**15**	
all				**4**

Table 38: The number of selected features that were selected by two of the methods in all communities (top) and by two methods in the country, as well as by all three in the lower right corner.

all communities	country	all communities and country
month	act_days_first2Weeks_ratio	active_days_first3Months_ratio
	act_days_firstMonth_ratio	max_following_inactive_days_ratio
	avg_replycreated_day	avg_min_bet_posts
	avg_postcreated_day	active_days_ratio
	avg_min_bet_replies	postcreated
	replycreated	
	dayOfMonth	
	dayOfWeek	
	downvotes	
	year	

Table 39: The features, that were selected by RFECV in all communities, in the country, and in both.

that these features could symbolise the general behaviour of the different communities, as well as of the whole country. That is why we will observe these features in more detail.

For the RFFI and ReliefF, we have also shown the selected features as well as their average importance over all communities, as well as their median rank. These are shown in Tab. 40, where we have printed these features in bold that were selected by both methods and underlined these, that were selected by RFECV. The feature *max_foll_inact_d_ratio*, which was selected in all communities by RFECV was also ranked at one in all cities by ReliefF and yields high average importance. We can also see that the features *postcreated*, *avg_min_bet_posts, RegPostGap_h*, and *month* were selected by two methods. These features seem to have a high impact on the models' decisions and will, therefore, also be observed in more detail.

RFFI	avg. imp.	median rank	ReliefF	avg. imp.	median rank
postcreated	0.136	4	max_following_inactive_days_ratio	0.3464	1
replycreated	0.128	3	month	0.2815	2
downvotes	0.040	9	posts_day	0.1509	5
avg_min_btw_posts	0.036	8	active_days_firstMonth_ratio	0.0902	10
RegPostGap_h	**0.028**	**10**	**RegPostGap_h**	**0.0819**	**7**
avg_min_bet_replies	0.024	12	avg_min_bet_downvoted	0.0728	10
upvotes	0.016	13			
user_threads	0.014	13			

Table 40: The selected features of the RFFI (importance greater zero) and ReliefF (importance greater 0.056) in all communities, that were not selected by all methods, with their average importance and median rank. We have underlined the features that were also selected by RFECV in all communities in Tab. 39 and printed these features in bold, which were just selected by RFFI and ReliefF in all communities.

For the country model, we have listed the selected Features of RFFI and ReliefF in Tab. 41. We have underlined these features that were also selected by RFECV and printed these bold that were also selected in all communities. Seven features were also important in all communities for the RFFI and also two for the ReliefF. For the RFFI we can also see, that the feature *postcreated* was defined as most important, which we have also seen for the community models, where it was on average also the most important feature. This can also be seen for the feature *max_foll_inact_d_ratio* for the ReliefF. This shows us, that the country model also selected features as important that we have seen for the community models too.

A special interest was in the features that were selected by all three methods in all communities, as well as in the country. These features are shown in Tab. 42, where we have given their average importance and median rank of RFFI and ReliefF over all communities, as well as over all communities and the country for the *avg_min_bet_interactions*, which is the only feature that was selected in all communities and the country by all methods, which, therefore, leads us to the finding that this feature seems to be important for the characteristics of the country and the different communities. The *replies_day* were selected in all communities and therefore,

could be used for a general characteristic of the communities. In comparison to the features that were not selected by all three methods in Tab. 40, we can also see that there are features that achieved higher average importance than these two features. Overall we have seen that there are features that were selected by two or more of these three methods in all communities, which we will observe in more detail in the next section.

RFFI	**imp.**	**rank**	**ReliefF**	**imp.**	**rank**
postcreated	0.26	1	**max_foll_inact_d_ratio**	0.29	2
avg_min_bet_posts	0.11	3	act_d_first3Months_ratio	0.09	4
year	0.08	4	active_days_firstWeek_ratio	0.08	6
avg_replycreated_day	0.05	7	happyratio1st3Months	0.08	7
replycreated	0.05	8	happyratio1stMonth	0.06	9
downvotes	0.03	9	**act_d_firstMonth_ratio**	0.06	10
avg_min_bet_replies	0.02	11			
avg_postcreated_day	0.01	12			
replycreated_3_months	0.01	13			
upvotes	0.01	14			
user_threads	0.01	15			

Table 41: The features of the country model, that were only selected by RFFI or ReliefF with their importance and rank. We have underlined these features that were also selected by RFECV for the country and printed these features in bold, which were also selected by the specific method in all communities, as we have seen in Tab. 40.

	avg. imp.		median rank	
Intersection of all three	**RFFI**	**ReliefF**	**RFFI**	**ReliefF**
replies_day	0.106	0.1778	4	4
avg_min_bet_interactions	0.092	0.0803	4	6
avg_min_bet_interactions	0.867	0.0829	4	5.5

Table 42: The features that were selected in all communities by all three methods and their average importance and median rank for RFFI and ReliefF, as well as the median rank and average importance for the *avg_min_bet_interactions* of all communities and the country as the features, that was selected in all models by all three methods.

6.5.3 Feedback into Empirical Analysis

As the next step, we want to have a look at the empiricism of the best features. We will, therefore, have a look at the mean value and its standard deviation (std dev) for different user groups. The groups will be the defined classes for the active time, see Tab. 6 in Section 1.3. With these user groups, we try to detect significant differences for the different features. With the help of this analysis, we probably will detect statistically significant different user groups for specific features. For these features, there could then be detected a trend, like an increasing happyratio for a longer active time, that let us learn how the model behaves and let us define different community characteristics.

As the first step, we have looked at the distributions of means of the most important features and their standard deviation within the different user groups. For this, we have seen in [15], that for a normal distribution, around 68% of the samples lay inside the standard deviation of the mean, 95% inside the double of the standard deviation of the mean, and 99% inside the triple of the standard deviation, which we can use as a rule of thumb to select specific user groups for specific features that have no overlapping standard deviation and therefore, could have a significantly different distribution.

Because of user groups with no overlapping standard deviation, as we can see in Tab. 43, where the x-axis shows the user groups and the y-axis the feature value normalised between zero and one by Min-Max-Scaler, we will first have a look at the user groups, as we have defined them in Tab. 6, for the features *max_following_inactive_days_ratio, month, posts_day*, and *replies_day*. We assume that these four features could have statistically significant different user groups. To confirm this we have used statistical significance tests (cf. Sec. 2.7).

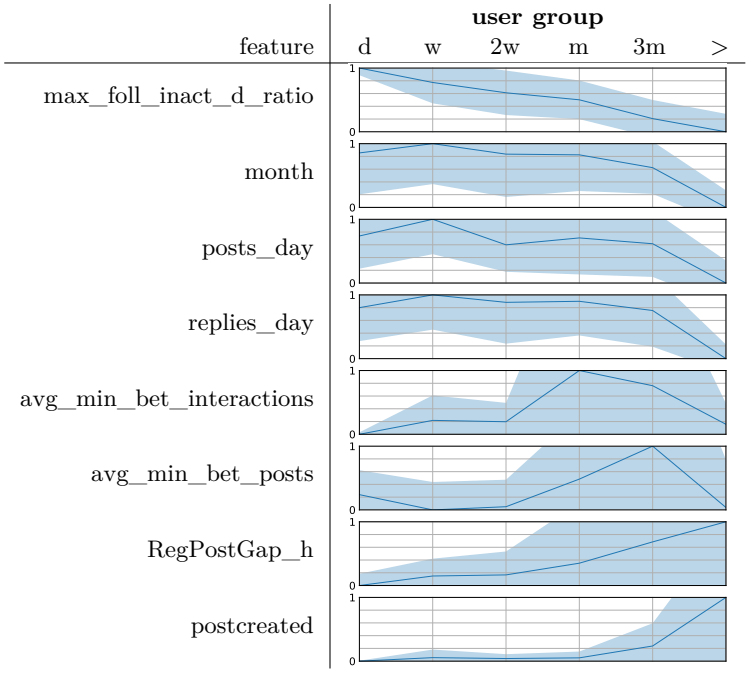

Table 43: The distribution of the mean value of the given feature within the given time-window user groups and their standard deviation, normalised between 0 and 1. We use the rule of thumb that these user groups are statistically significantly distributed when they do not have an overlapping standard deviation. This can be seen for the first four features.

To confirm that these features have significant differences, we have used the **Kruskal-Wallis H-Test** (cf. Sec. 2.7.2) for all models, with $\alpha = 0.05$ and H_0, that all user groups have an equal distribution of the specific feature. The results of the features *max_following_inactive_days_ratio, month, posts_day, month, posts_day*, and *replies_day* are shown in Tab. 44 for all communities, where we have rounded the p-value for four decimal places. Because of the huge number of samples, we can see that we have a high statistic value and a p-value of zero within all cities for all features, which leads to the rejection of H_0. This means, that for each feature at least one group dominates the others and therefore, is statistically significantly different distributed.

	max_foll_inact_d_ratio		**month**		**posts_day**		**replies_day**	
community	stat	p-value	stat	p-value	stat	p-value	stat	p-value
Riyadh	156,742.68	0.0	70,127.05	0.0	71,246.44	0.0	90,127.88	0.0
Jeddah	56,969.24	0.0	30,290.73	0.0	217,45.40	0.0	29,074.37	0.0
Mecca	24,849.06	0.0	14,189.39	0.0	10,658.59	0.0	13,891.33	0.0
Al Bahah	4,548.52	0.0	4,920.32	0.0	4,900.94	0.0	4,854.76	0.0
Al Jafr	110.59	0.0	54.96	0.0	52.17	0.0	55.18	0.0
Country	541,526.02	0.0	314,902.98	0.0	61,906.83	0.0	97,394.93	0.0

Table 44: The results of the Kruskal-Wallis H-test for each community for the four most important features, that had no overlapping standard deviation, as we have seen in Tab. 43. For each feature, the statistic and p-value are given, which was rounded for four decimal places. H_0 was, that all user groups have an equal distribution of the specific feature. As we can see, the p-value for all features in all cities equals zero, which confirms our assumption that we have significant differences between user groups for these features.

To determine these significant user groups, we have used the **Mann-Whitney U-Test**, see Section 2.7.1, to compare each user group to the others for each community. We have used again $\alpha = 0.05$, and the changed H_0, that both user groups have the same distribution for the specific feature. This showed us many statistically significant differences of user groups, but also nonsignificant ones.

We have seen just significant distributions over all user group combinations for the features *max_following_inactive_days_ratio* and *replies_day* in all cities and the country, except for Al

Jafr. The results for the four features of Al Jafr are shown in Fig. 50, where the user group on the x-axis is statistically tested to the user group on the y-axis, and a dark orange symbolises a p-value lower than the α of 0.05 or the darkest orange lower than 0.01, which means that we reject H_0, which means that the specific user groups have a statistically significant different distribution. This shows us, that for the *max_following_inactive_days_ratio* only the user groups month and two weeks do not have a significant difference in distribution. For the other three features, we can see that users with more than three months of active time are significantly different distributed than all other groups, and also the three months user from users from one day and week for the feature *month*. For the *posts_day*, a significantly different distribution between users from one week and users from two weeks, month, and three months can also be detected. Because of the small number of samples/users in Al Jafr (176), we can assume that these findings are a good representation of the city itself, but not for the whole population.

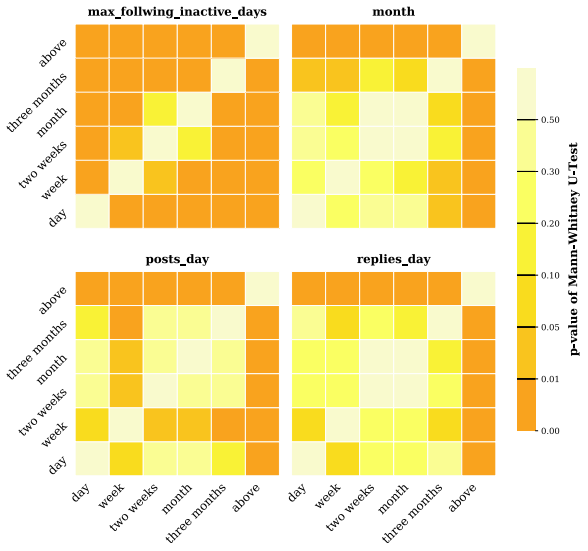

Figure 50: The p-value of the Mann-Whitney U-Test for the four presented features and the defined classes of users in Al Jafr. For the *max_following_inactive_days_ratio* just the month and two weeks user are not significantly different distributed. For the other features, we can see fewer significant differences. Just for the users with more than three months active time a significant difference to all others, and for the three-month user with the day and week users for the feature *month* can be seen. For the *posts_day*, we see that also the week users are significantly different from these of two weeks, month, and three months. Because of the small number of samples/users in Al Jafr (176), we can assume that these findings representing Al Jafr in a good way, but not represent the whole population in a good way.

For the other communities, we have seen in Fig. 51, where the user group on the x-axis is statistically tested to the user group on the y-axis, that we have just a few user groups that are not significantly different for the features *month* and *replies_day*. For the *month* feature we can see, that, except for Al Bahah, the user groups month and day are not significantly different. In Al Bahah we can see that only the groups day and week are not statistically significantly different. For the *replies_day* feature we can see that we have no difference for the groups week and two weeks for all cities, except for Mecca, and additionally the day and month group for Riyadh and Mecca.

This shows us, that nearly all user groups are statistically significantly different for the given features, except for Al Jafr, which has a much smaller number of samples, which does not lead to a good representation. Also, the huge amount of samples in the other communities supports the statistical significance in the case of huge statistic values. This shows for us practically that we have features with statistically significant differences between the user groups, which help our model for a better prediction. For this, we will now also examine the qualitative distribution

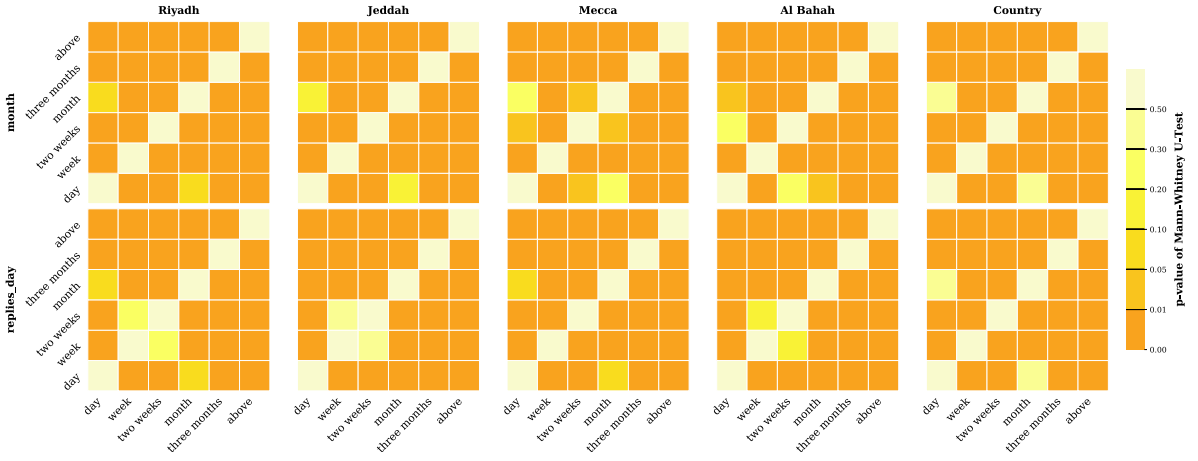

Figure 51: The p-value of the Mann-Whitney U-Test for the two features *month* and *replies_day* and the defined classes of users in Riyadh, Jeddah, Mecca, and Al Bahah, as well as for the country. There are only a few user groups that are not significantly different, like the month and day group for the month feature in Riyadh, Jeddah, and Mecca, as well as in Riyadh and Mecca for the replies_day. Also, the two weeks and week groups are not significantly different for the feature replies_day in Riyadh, Jeddah, and Al Bahah, as well as the groups two weeks and day in Al Bahah for the month feature. This shows us, that there are just a few groups of users, that cannot be marked as significantly different, which supports their strength from the calculated importance and symbolises a better help for the differentiation of the predictor. Because of the higher number of samples/users, e.g., in Riyadh (289,963) we can also say that these results can be used for a better representation of the whole population in contrast to the results of Al Jafr before.

of these features in order to identify any trends.

For the examination of the qualitative distribution, we have calculated the quantiles of the sets (10% to 90%) and plotted these for the time subsets in Tab. 45, where the x-axis represents the user groups and the y-axis the feature values, which are normalised between zero and one. The line represents the median (50%) and the outermost brightest area of the population between

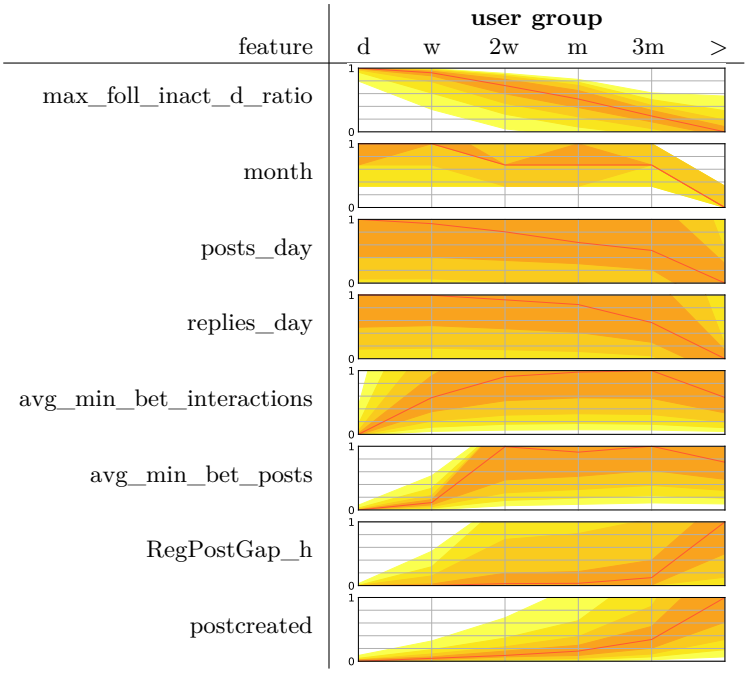

Table 45: The user churn groups for the most important features and their qualitative population. The red line displays the median and the areas around the are between the quantiles of 10%-90%, 20%-80%, etc. The features were normalised by Min-Max-Scaler between zero and one. This shows trends of the different features over the churn time of the users.

the 10% and 90% quantile.

Because of the fact, that we can see for some metrics a clear trend and for others not, and because of the fact that some features are time-invariant, we cannot draw a clear conclusion of this part. This could be part of future work, because of the usage of features with a clear trend that could be useful, e.g., learning from model behaviour.

6.6 A sample Decision Tree

As a small excurse, we now want to have a look at a sample DT. As we have seen in Section 6.1, the DT also yielded strong results for the classification task in Jeddah with an accuracy of 0.8000 ($\pm$0.0065) and for the regression task with a R^2 Score of 0.9336 ($\pm$0.0025). Because of this, we now want to show a sample DT for the prediction of the active time within Jeddah. This we have shown in Fig. 52. The model of Jeddah had a depth of 16. Because of the space limitations, we have shown a depth of four. At each node, we have given the feature boundary, the samples at this node, and the class that would be predicted for a sample at this node. For the root node, we have also given the value, which displays how many samples are ordered to which class, and have printed this class bold that had the most samples and would, therefore, be predicted. At the leaves, we have just given the number of samples, as well as the final prediction class. For better visualisation, the features are given as numbers. At each node, the predictor would predict this class with the highest number of samples.

We have printed six nodes in colours, because of their multiple occurrences and therefore, could be interpreted as more important than other features. For these six nodes, we have given their label in the upper left corner. For this task, we can see that the features *downvotes, received_upvotes_firstWeek* and the *replies_day*, would be seen as possibly more important than the others. With the help of this, we could also create a Boolean function that is represented by this tree. As an example, we have chosen the function for class 3, whose path we have printed as red dotted border. The Boolean function for this would be:

$$\begin{aligned} class3 = &(downvotes > -0.18) \\ &\wedge (replies\_day > 0.33) \\ &\wedge (avg\_response\_time\_minutes\_day \leq -0.43) \\ &\wedge (replies\_day > 1.09) \end{aligned} \tag{16}$$

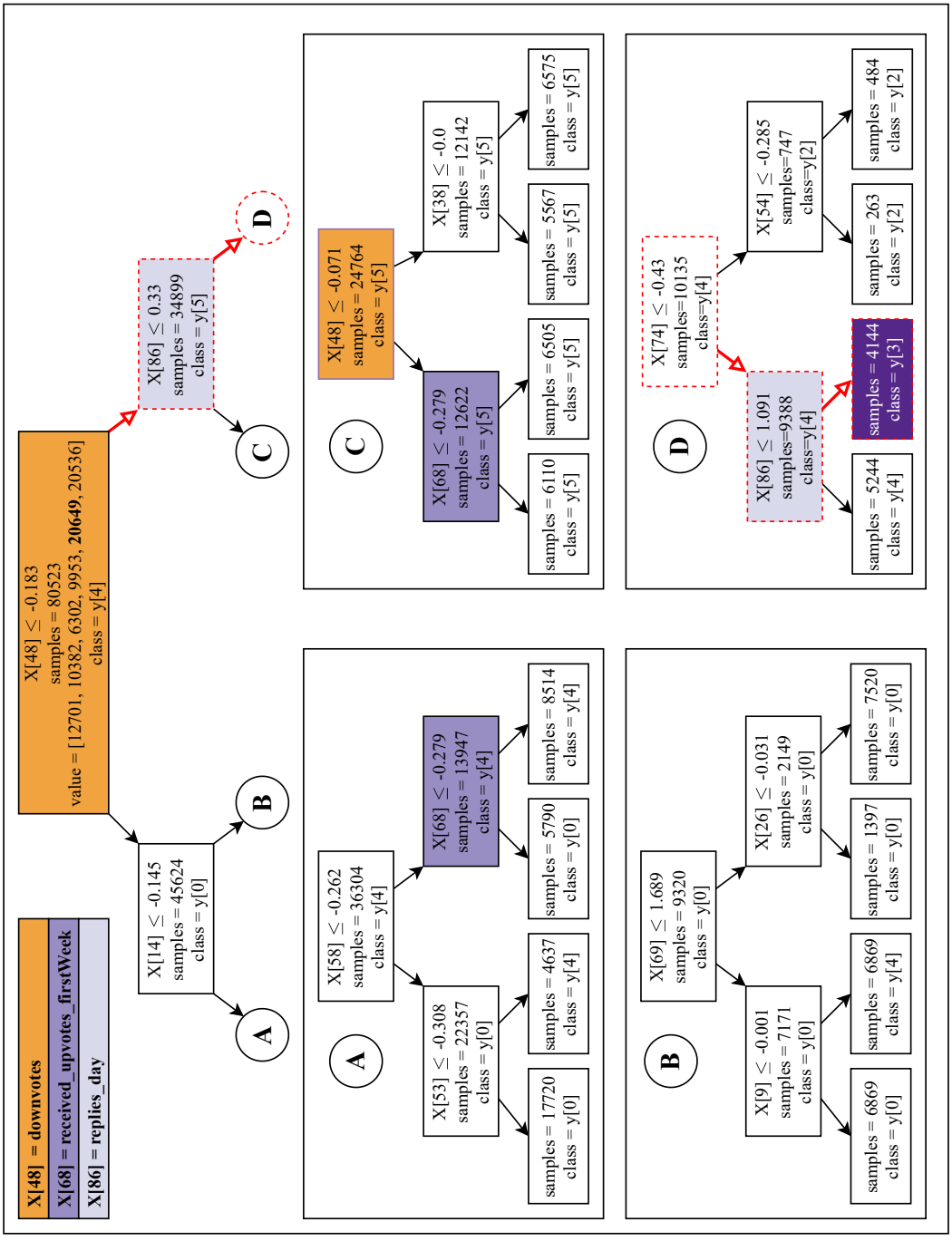

Figure 52: An example of a DT for the community of Jeddah with a depth of four. We have coloured these nodes that used the same feature and have given their feature name in the upper left corner. We have also given an example path for predicting class 3, which is shown as red dotted elements. For the root node, we have given the additional value, which displays the number of samples per class at this node, and printed the highest value bold, which would be predicted. This also exists for all other nodes but is not shown, because of space limitations.

7 Lessons Learned

Within the proceedings of this bachelor thesis, we have seen a range of methods and tools that we have not used, because of time limitations, but which could improve our models. We will now have a look at the different steps, to explore the possible model/evaluation space.

Feature Engineering. Within the Related Work (cf. Sec. 3), a wide range of different features was presented. We have adapted our set of features to these, but also not to all. For this, it could be useful to test a wider range of features, like content-based features or word embeddings. What we have also seen in Section 4, that features like the *registered_days* got calculated importance that clearly outscored all others and therefore, probably gained too much information, which needs to be more avoided in future work. Also features with a strong correlation to the active minutes are possible too predictive because of their time-invariance, and could, therefore, be more normalised to avoid too much impact of the model prediction, as we have seen, e.g., for the *postcreated* feature, which tends to raise with increasing active time. This will then be also more interpretable within the empirical feedback. As we have mentioned in Section 4.2, also an investigation of the performance when using only the most important features could yield some interesting information and could lead to a trade-off between performance, runtime, and memory usage.

Pre-Processing. When pre-processing the data, a range of other imputing methods, e.g., a KNNImputer from *SciKit-Learn* [73] that completes the missing values with the K-Nearest Neighbours algorithm (cf. Sec. 2.3.2), may impute the missing values by a more realistic way and, therefore, could improve the models' performance, could be used. For the scaling of the data, other methods from SciKit-Learn than the Min-Max-Scaler and StandardScaler, like the *RobustScaler*, that uses statistics for scaling the data and is *robust* to outliers [74], could be tested for improvements. We have also seen many more balancing methods from *Imbalanced-Learn* [75], that we have not used, e.g., a version of SMOTE, like the BorderlineSMOTE [76]. These methods should be investigated in the future for an improvement of the predictive model. All these different methods could be useful for improving the models, as well as for different other use cases we did not observe for our models. The possibility of improvement can not really be assessed, because of the fact, that the tested methods in this thesis did not significantly improve our models, also because of the already high scores.

ML Methods. When having a look at the used ML methods, a wider range of methods could be tested and used, e.g., the Naïve Bayes or a self-defined ensemble method, that uses a range of weak methods, as we have seen in [59]. A ML approach we have only touched is the usage of Neural Networks. We have only used the Multi-Layer Perceptron, which also yielded strong results for the regression task. That is why we argue, that an investigation of, e.g., Convolutional Neural Networks, could achieve high predictive power, as we have, e.g., seen in [60, 58].

8 Future Work

Within this bachelor thesis, we have built strong models for predicting the minimal active time of users from different communities. With knowledge about features that increase the retention of a user and therefore, also can be used for avoiding user churn, there are multiple tasks for future work.

Textual Features. Within our research, we have only used meta-data features. That is why a textual approach on the users could gain also information that could help to understand the user behaviour, which could then again be used for a connection to behavioural results of this research to detect trends of textual features in comparison to, e.g., the happyratio to detect word or word groups that in- or decrease the happyratio of a user and therefore, could influence the user churn indirectly.

Optimization Target. A possible task for the future could be the change of our optimization target, *active time*. For this, we could use features, that were most important for different tasks. I.e., the *happyratio* of a user could be a possible target, which means, their ratio of upvotes and downvotes, which indirectly symbolises the acceptance of a user's post within the communities and could, therefore, have an impact on the users active time, e.g., a lower one when a user achieved a low ratio. But also other targets could result in strong models that could each gain information from a certain viewpoint about user behaviour and community characteristics.

User-Community Relationship. Within this thesis, we have observed the models for different characteristics and, therefore, also the communities. That indirectly symbolised the impact of the community on the churn of a single user. A switch of the research from *user churn* to *community churn* could give another point of view. For this, an investigation of the impact of the user on the community could give information about the shrinking or growth of communities concerning user behaviour. This would then show that not only the community can strengthen user churn, as we have shown in this thesis, but also the user on the community churn, by creating unfriendly content that could lead to a shrinking community. This could then be combined to define the user-community relationship and its impact on churn.

Time Observation. A part of our research, was the observation of the performance of the models, when using the different time-window features and therefore, the observation of how the model improves when scaling up the observation time of a user before predicting his active time. Within this thesis, we have only observed for a day, week, two weeks, month, and three months. For the future, a long time observation of a year or multiple years could yield interesting results, as we have seen in [77], where also an optimization of the models for the observation time, and therefore, finding a trade-off between observation time and prediction accuracy was shown. This could then be used for determining long-term user behaviour.

Location. Because of the model approach on the community of Saudi Arabia for the social media app Jodel, there also is a wide range of use cases within other countries. It could be interesting to observe the performance of the model when, e.g., predicting users from Germany, which probably have a different behaviour because of a different culture and therefore, different behaviour in posting, like a higher amount of posts, more replying to posts or another post content. This again could be investigated to build more general models that not only work for specific communities or a single country but rather perform well in multiple countries. With the help of cross applications between different countries, as we have done with the different communities, we could then detect possible similarities between these to find a generalisation, which then could be determined by feature analysis, which could be fed back into the empirics of different countries.

Empiricism. For observing the characteristics of a community, we have observed the importance of the features and therefore, also the most important features of the different models and also within the different communities. These features symbolise an important usage within the models and were, therefore, fed back into an empirical analysis. This helped us for a better understanding of the models' behaviour and insights. This empirical analysis could be expanded to detect community states, that include the most important features that can represent the

communities and their behaviour. This could then be used for a more complex comparison of the models, hence communities. Another step, that could be done is the observation of clear trends within these empiricisms. These trends could return a picture of the characteristics and could, therefore, help to understand and learning the models' behaviour, which nowadays is more known as *explainable AI*.

9 Conclusion

The retaining of users and the associated avoiding of user churn is a core part of marketing to increase profitability and growth of a companies platform. That is why detecting users with a high customer lifetime value and retaining them to a platform is important but also a complex task. Because of this, predicting churners by ML has got much attention in the past. Within this thesis, an approach was shown for predicting churners or rather their lifetime on the anonymous social media app Jodel.

To get closer to the topic, we have looked at the empiricism of the user-base of Jodel within the Kingdom of Saudi Arabia, which we have used to define user churn on Jodel. Based on literature research we have seen many approaches for predicting churn in different contexts, where different ML methods performed very well, and also the feature approach and churn definition differed widely. Also textual and meta-data-based approaches, as well as time-related features that were compared in different ways, were shown. Using ML does not always work with the same success and there is a wide range of methods and proceedings that can be used to build optimal models for a prediction task.

Based on an existed feature set and a range of newly created ones, we have used a ML framework, that we have expanded, to build models for different communities and the whole country to predict the users' lifetime/churn class.

The underlying research question of this thesis "*Can we build predictive models for single communities that achieve high prediction accuracy and can we build a single model that can predict user churn of different communities in the Kingdom of Saudi Arabia with high prediction accuracy?*" was answered by ML models for predicting the exact lifetime of a user (regression problem) and classifying the user into one of six churn classes (multi-label classification) and additionally if a user will churn within a given time (binary classification).

As main findings, we have seen that the Random Forest was most sensible for the regression and classification task. With this method, we have built models for the communities and the country. With these, we proceeded with a feature subsets analysis, which has shown that the models worked very well when using *all* features and the *user* features in general. When having a look at the time window features, we have seen that the models did not work very strong for both tasks, where the strongest score was achieved after an observation time of three months, which has shown us that the prediction accuracy probably increases when observing for a longer time. Due to undesirably bad scores, we decided to ease up the classification problem from a multi-label to a binary classification, which has increased the scores extremely and has shown that the models can perform very strong for this task when predicting if a user will churn until different time thresholds with maximum accuracies over 99%.

Based on these models, we additionally, wanted to answer the question: "*How do the different models perform on other communities, and can we detect similarities to possibly define different behaviours of models, hence communities?*"

For this, we have researched the model performance on other communities to see how applicable they are in these. This has shown that the country model performed very strong for the regression task and outperformed all other models. For the classification task, the country model again yielded the strongest scores over all communities even if it did not outperform the community-specific models. Besides the country model, also the models of Riyadh, Jeddah, and Mecca yielded very strong results even if they did not exceed the country model. By using the models' feature importance, we have detected strong correlations between the models Riyadh, Jeddah, and Mecca but also with the country model. These correlations, hence similarities, were determined by the most important features of the three methods Random Forest Feature Importance, ReliefF, and RFECV. These were fed back into empirics, which has shown significant differences between user groups that were defined by the borders of the six churning classes. These have also shown relations of features to the active time of users and, therefore, trends but no conclusive ones that led us to define community states that represent the communities' behaviours.

Since we could not detect clear trends in empiricism and could not define characteristics of models, hence communities, there are many tasks for feature analysis and empirical researches, that could be used for defining community states for more learning and understanding of the model behaviour. Due to the small amount of literature for anonymous social networks, the research on this topic is very versatile and open, and because of the continuous growth of social networks, the competition will grow too and, therefore, firstly the retaining of users and secondly the knowledge about reasons for churn will be very important for the platforms. With the help of ML and explainable AI, the retaining of users could be improved.

References

[1] Philip Kotler. *A framework for marketing management*. Pearson Education, 2016.

[2] Capucine. *What is Jodel*. Sept. 19, 2020. URL: https://jodel.zendesk.com/hc/en-us/articles/360009688653-What-is-Jodel- (visited on 09/19/2020).

[3] Helge Reelfs; Timon Mohaupt; Oliver Hohlfeld; Niklas Henckell. ''Hashtag Usage in a Geographically-Local Microblogging App''. In: *Companion Proceedings of The 2019 World Wide Web Conference*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 919–927. ISBN: 9781450366755.

[4] Capucine. *Karma*. Sept. 19, 2020. URL: https://jodel.zendesk.com/hc/en-us/articles/360001017113-Karma (visited on 09/19/2020).

[5] Capucine. *Jodel Values*. Sept. 15, 2020. URL: https://jodel.zendesk.com/hc/en-us/articles/360009286874-Jodel-Values (visited on 09/19/2020).

[6] Capucine. *Here/Very close/Close/Far/Hometown*. Sept. 19, 2020. URL: https://jodel.zendesk.com/hc/en-us/articles/360001040974-Here-Very-close-Close-Far-Hometown (visited on 09/19/2020).

[7] Capucine. *Our moderation system*. Sept. 19, 2020. URL: https://jodel.zendesk.com/hc/en-us/articles/360001048074-Our-moderation-system- (visited on 09/19/2020).

[8] Capucine. *Why do you need access to my phone ID?* Sept. 18, 2020. URL: https://jodel.zendesk.com/hc/en-us/articles/360021364313-Why-do-you-need-access-to-my-phone-ID- (visited on 09/19/2020).

[9] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., 2017.

[10] Stefano Nembrini; Inke R. König; Marvin N. Wright. ''The revival of the Gini importance?'' In: *Bioinformatics* 34.21 (May 2018), pp. 3711–3718. ISSN: 1367-4803.

[11] Igor Kononenko; Edvard Šimec; Marko Robnik-Šikonja. ''Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF''. In: *Applied Intelligence* 7 (1997), pp. 39–55.

[12] Kenji Kira; Larry A. Rendell. ''The feature selection problem: Traditional methods and a new algorithm''. In: *Aaai*. Vol. 2. 1992, pp. 129–134.

[13] Isabelle Guyon; Jason Weston; Stephen Barnhill; Vladimir Vapnik. ''Gene selection for cancer classification using support vector machines''. In: *Machine learning* 46.1-3 (2002), pp. 389–422.

[14] SciKit-Learn. *Feature selection*. 2020. URL: https://scikit-learn.org/stable/modules/feature_selection.html#rfe (visited on 12/11/2020).

[15] Ludwig Fahrmeir; Christian Heumann; Rita Künstler; Iris Pigeot; Gerhard Tutz. *Statistik: Der Weg zur Datenanalyse*. Springer Spektrum, Berlin, Heidelberg, 2016.

[16] SciKit-Learn. *sklearn.preprocessing.MinMaxScaler*. 2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (visited on 03/02/2021).

[17] Imbalanced-Learn. *RandomUnderSampler*. 2021. URL: https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html#imblearn.under_sampling.RandomUnderSampler (visited on 02/25/2021).

[18] Alberto Fernández; Salvador García; Mikel Galar; Ronaldo C. Prati; Bartosz Krawczyk; Francisco Herrera. *Learning from Imbalanced Data Sets*. Springer, Cham, 2018. ISBN: 978-3-319-98073-7.

[19] Imbalanced-Learn. *RandomOverSampler*. 2021. URL: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html#imblearn.over_sampling.RandomOverSampler (visited on 02/25/2021).

[20] Nitesh V. Chawla; Kevin W. Bowyer; Lawrence O. Hall; W. Philip Kegelmeyer. ''SMOTE: synthetic minority over-sampling technique''. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[21] Leo Breiman; Jerome Friedman; Charles J Stone; Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[22] SciKit-Learn. *Decision Trees*. 2020. URL: https://scikit-learn.org/stable/modules/tree.html (visited on 12/03/2020).

[23] SciKit-Learn. *Nearest Neighbors*. 2020. URL: https://scikit-learn.org/stable/modules/neighbors.html#neighbors (visited on 03/12/2020).

[24] Thomas A. Runkler. *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. Springer Fachmedien Wiesbaden GmbH, 2020.

[25] Bernhard E. Boser; Isabelle M. Guyon; Vladimir N. Vapnik. ''A Training Algorithm for Optimal Margin Classifiers''. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152. ISBN: 089791497X.

[26] SciKit-Learn. *Support Vector Machines*. 2020. URL: https://scikit-learn.org/stable/modules/svm.html#support-vector-machines (visited on 12/12/2020).

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science + Business Media, LLC, 2006.

[28] Peter Ghavami. *Big Data Analytics Methods: Analytics Techniques in Data Mining, Deep Learning and Natural Language Processing*. Walter de Gruyter GmbH & Co KG, 2019.

[29] Harris Drucker; Christopher J. C. Burges; Linda Kaufman; AlexSmola; Vladimir Vapnik. ''Support Vector Regression Machines''. In: *Advances in Neural Information Processing Systems*. Ed. by M. C. Mozer, M. Jordan, and T. Petsche. Vol. 9. MIT Press, 1997, pp. 155–161.

[30] Praphula Kumar Jain; Rajendra Pamula. ''Content-Based Airline Recommendation Prediction Using Machine LearningTechniques''. In: *Machine Learning Algorithms for Industrial Applications*. 2021.

[31] Srikanta Patnaik; Xin-She Yang; Ishwar K. Sethi. ''Churn Prediction and Retention in Banking, Telecom and IT Sectors Using Machine Learning Techniques''. In: *Advances in Machine Learning and Computational Intelligence*. 2019.

[32] Sahar F. Sabbeh. ''Machine-Learning Techniques for CustomerRetention: A Comparative Study''. In: *International Journal of Advanced Computer Science and Applications* (2018).

[33] Warren S. McCulloch; Walter Pitts. ''A logical calculus of the ideas immanent in nervous activity''. In: *The bulletin of methematical biophysics 5, 155-133* (1943).

[34] Frank Rosenblatt. ''The perceptron: A probabilistic model for information storage and organization in the brain.'' In: *Psychological Review, 65(6), 386–408* (1958).

[35] David E. Rumelhart; Geoffrey. E. Hinton; Ronald. J. Williams. ''Learning Internal Representations by Error Propagation''. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[36] James Suriowiecki. *The wisdom of crowds: Why the many are smarter than the few and how collective wisdom shapes business, economies, societies and nations*. London: Little, Brown, 2004.

[37] Leo Breiman. ''Bagging predictors''. In: *Machine learning* 24.2 (1996), pp. 123–140.

[38] Leo Breiman. ''Random Forests''. In: *Machine Learning* 45 (2001), pp. 5–32.

[39] Trevor Hastie; Saharon Rosset; Ji Zhu; Hui Zou. ''Multi-class adaboost''. In: *Statistics and its Interface* 2.3 (2009), pp. 349–360.

[40] SciKit-Learn. *sklearn.model_selection.GridSearchCV*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=gridsearchcv#sklearn.model_selection.GridSearchCV` (visited on 10/22/2020).

[41] Claude Sammut; Geoffrey I. Webb. ''Mean Absolute Error''. In: *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, pp. 652–652. ISBN: 978-0-387-30164-8.

[42] Werner A. Stahel. *Statistische Datenanalyse*. 4., verbesserte Auflage. Vieweg+Teubner Verlag, Wiesbaden, 2002. ISBN: 978-3-528-36653-7.

[43] Kai Ming Ting. ''Precision and Recall''. In: *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, pp. 781–781. ISBN: 978-0-387-30164-8.

[44] Frank Krüger. ''Activity, Context, and Plan Recognition with Computational Causal Behaviour Models''. PhD thesis. University Rostock, Dec. 2016.

[45] Hans-Joachim Mittag. *Statistik: Eine Einführung mit Interaktiven Elementen*. Springer Spektrum, Berlin, Heidelberg, 2014. ISBN: 978-3-642-54387-6.

[46] Jürgen Janssen; Wilfried Laatz. *Statistische Datenanalyse mit SPSS*. Springer Gabler, Berlin, Heidelberg, 2017.

[47] Thomas Schuster; Arndt Liesen. *Statistik für Wirtschaftswissenschaftler*. Springer Gabler, Berlin, Heidelberg, 2017.

[48] Gideon Dror; Dan Pelleg; Oleg Rokhlenko; Idan Szpektor. ''Churn prediction in new users of Yahoo! answers''. In: *Proceedings of the 21st International Conference on World Wide Web*. 2012, pp. 829–834.

[49] Jagat Sastry Pudipeddi; Leman Akoglu; Hanghang Tong. ''User churn in focused question answering sites: characterizations and prediction''. In: *Proceedings of the 23rd International Conference on World Wide Web*. 2014, pp. 469–474.

[50] Jiwoon Jeon; W. Bruce Croft; Joon Ho Lee; Soyeon Park. ''A framework to predict the quality of answers with non-textual features''. In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 2006, pp. 228–235.

[51] Yandong Liu; Jiang Bian; Eugene Agichtein. ''Predicting information seeker satisfaction in community question answering''. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 2008, pp. 483–490.

[52] Haifa Alharthi; Djedjiga Outioua; Olga Baysal. ''Predicting questions' scores on stack overflow''. In: *2016 IEEE/ACM 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*. IEEE. 2016, pp. 1–7.

[53] Fabian Hadiji; Rafet Sifa; Anders Drachen; Christian Thurau; Kristian Kersting ; Christian Bauckhage. ''Predicting player churn in the wild''. In: *IEEE Conference on Computational Intelligence and Games 2014* (2014).

[54] Anders Drachen; Eric Thurston Lundquist; Yungjen Kung; Pranav Rao; Rafet Sifa; Julian Runge; Diego Klabjan. ''Rapid prediction of player retention in free-to-play mobile games''. In: *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*. 2016.

[55] África Periáñez; Alain Saas; Anna Guitart; Colin Magne. ''Churn Prediction in Mobile Social Games: Towards a Complete Assessment Using Survival Ensembles''. In: *IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (2016).

[56] Jaya Kawale; Aditya Pal; Jaideep Srivastava. ''Churn prediction in MMORPGs: A social influence based approach''. In: *2009 International Conference on Computational Science and Engineering*. Vol. 4. IEEE. 2009, pp. 423–428.

[57] Sang-Kwang Lee; Seung-Jin Hong; Seong-Il Yang; Hunjoo Lee. ''Predicting churn in mobile free-to-play games''. In: *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2016, pp. 1046–1048.

[58] Julian Runge; Peng Gao; Florent Garcin; Boi Faltings. ''Churn prediction for high-value players in casual social games''. In: *2014 IEEE conference on Computational Intelligence and Games*. IEEE. 2014, pp. 1–8.

[59] Adnan Idris; Asifullah Khan. ''Ensemble based Efficient Churn Prediction Model forTelecom''. In: *12th International Conference on Frontiers of Information Technology* (2014).

[60] Mohammad Ridwan Ismail; Mohd Khalid Awang; M Nordin A Rahman; Mokhairi Makhtar. ''A multi-layer perceptron approach for customer churn prediction''. In: *International Journal of Multimedia and Ubiquitous Engineering* 10.7 (2015), pp. 213–222.

[61] Dudyala Anil Kumar; Vadlamani Ravi. ''Predicting credit card customer churn in banks using data mining''. In: *International Journal of Data Analysis Techniques and Strategies* 1.1 (2008), pp. 4–28.

[62] Yang Yang; Zongtao Liu; Chenhao Tan; Fei Wu; Yueting Zhuang; Yafeng Li. ''To stay or to leave: Churn prediction for urban migrants in the initial period''. In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 967–976.

[63] Georgios Rizos; Symeon Papadopoulos; Yiannis Kompatsiaris. ''Predicting News Popularity by Mining Online Discussions''. In: *Proceedings of the 2016 World Wide Web Conference* (2016).

[64] Hanchuan Peng; Fuhui Long; C. Ding. ''Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy''. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (Aug. 2005), pp. 1226–1238. ISSN: 1939-3539.

[65] Miron B. Kursa; Aleksander Jankowski; Witold R. Rudnicki. ''Boruta--a system for feature selection''. In: *Fundamenta Informaticae* 101.4 (2010), pp. 271–285.

[66] SciKit-Learn. *sklearn.preprocessing.KBinsDiscretizer*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.KBinsDiscretizer.html` (visited on 03/02/2021).

[67] Scikit-Rebate. *ReliefF*. 2020. URL: `https://epistasislab.github.io/scikit-rebate/using/` (visited on 03/02/2021).

[68] SciKit-Learn. *sklearn.impute.SimpleImputer*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html?highlight=simpleimputer#sklearn.impute.SimpleImputer` (visited on 02/25/2021).

[69] SciKit-Learn. *sklearn.preprocessing.StandardScaler*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html` (visited on 10/22/2020).

[70] SciKit-Learn. *sklearn.model_selection.ShuffleSplit*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ShuffleSplit.html?highlight=shufflesplit#sklearn.model_selection.ShuffleSplit` (visited on 02/25/2021).

[71] SciKit-Learn. *sklearn.dummy.DummyClassifier*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html?highlight=dummy%20classifier#sklearn.dummy.DummyClassifier` (visited on 11/12/2020).

[72] SciKit-Learn. *sklearn.dummy.DummyRegressor*. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyRegressor.html` (visited on 11/12/2020).

[73] SciKit-Learn. *scikit-learn: Machine Mearning in Python*. 2021. URL: https://scikit-learn.org/stable/ (visited on 06/02/2021).

[74] SciKit-Learn. *sklearn.preprocessing.RobustScaler*. 2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html?highlight=robust#sklearn.preprocessing.RobustScaler (visited on 11/12/2020).

[75] Imbalanced-Learn. *Imbalanced-Learn*. 2021. URL: https://imbalanced-learn.org/stable/index.html (visited on 06/02/2021).

[76] Imbalanced-Learn. *imblearn.over_sampling.BorderlineSMOTE*. 2021. URL: https://imbalanced-learn.org/stable/generated/imblearn.over_sampling.BorderlineSMOTE.html?highlight=borderline#imblearn.over_sampling.BorderlineSMOTE (visited on 06/02/2021).

[77] Michel Ballings; Dirk Van den Poel. "Customer event history for churn prediction: How long is long enough?" In: *Expert Systems with Applications* 39.18 (2012), pp. 13517–13522.

Based on this bachelor-thesis a paper was published, which uses and extends the included results: https://arxiv.org/pdf/2103.01300.pdf