EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, SVERIGE 2017

A Comparison of Resampling Techniques to Handle the Class Imbalance Problem in Machine Learning

Conversion Prediction of Spotify Users - A Case Study

MICHELLE JAGELID

MARIA MOVIN

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

KTH Royal Institute of Technology

Bachelor Thesis

A Comparison of Resampling Techniques to Handle the Class Imbalance Problem in Machine Learning

Conversion prediction of Spotify Users - A Case Study

Author:
Michelle Jagelid
Maria Movin

Supervisors:
Pawel Herman
Magnus Rööös
Examiner:
Örjan Ekeberg

A thesis submitted in fulfillment of the requirements for the degree of Bachelor in Computer Science

in the

School of Computer Science and Communication

June 2, 2017

KTH Royal Institute of Technology

Abstract

School of Computer Science and Communication

A Comparison of Resampling Techniques to Handle the Class Imbalance Problem in Machine Learning
Conversion prediction of Spotify Users - A Case Study

by Michelle JAGELID
Maria MOVIN

Spotify uses a freemium business model, meaning that it has two main products, one *free limited* and one *premium* for paying customers. In this study we investigated machine learning models' abilities, given user activity data, to predict conversion from free to premium. Predicting which of the users convert from free to premium was a class-imbalanced problem, meaning that the ratio of converters and non-converters was skewed.

Three methods were investigated: logistic regression, decision trees, and gradient boosting trees. We also studied if different resampling methods, which balance the train datasets, can improve classification performance of the models.

We showed that machine learning models are able to find patterns in user data that could be used to predict conversion. Additionally, for all our investigated classification methods, we showed that resampling increased the models' performances. The methods with best performances in our study were logistic regression and gradient boosting tree trained with oversampled data up to equal numbers of converters and non-converters.

Kungliga Tekniska Högskolan, KTH

Sammanfattning

Skolan för Datavetenskap och Kommunikation

Samplingmetoder för att hantera obalanserade klasser i maskininlärning
En fältstudie om prediktion av Spotify-användardes vilja att uppgradera produkten.

av Michelle JAGELID
Maria MOVIN

I den här studien undersökte vi om det går att, givet användardata från Spotify-användare, prediktera vilka användare som konverterar från gratisversionen till premiumversionen. Eftersom det finns fler användare som inte konverterar än som konverterar, var detta ett problem med obalancerade klasser. Obalancerade klasser är ett välkänt problem inom maskininlärning.

Tre maskininlärningsmetoder undersöktes: *Logistic regression*, *Decision trees* och *Gradient Boosting Trees*. Förbehandlingsmetoder som leder till att träningsdata får jämnare fördelning mellan klasserna undersöktes. Detta för att se om sådana förbehandlingar kunde öka modellernas förmåga att klassificera nya användare.

Vi visade att det var möjligt att med maskininlärningsmetoder, givet användardata, hitta mönster i data som kunde användas för att prediktera vilka användare som konverterar. För alla tre maskininlärningsmetoder visade det sig att förbehandling av träningsdata till jämnare fördelning mellan klasserna gav bättre resultat.

Av de undersökta modellerna presterade *Logistic regression* och *Gradient Boosting Tree* bäst då de tränats med förbehandlad data, så att slumpmässiga dubbletter av användare som konverterat lagts till i datasetet upp till helt jämn fördelning.

Acknowledgements

We would first like to thank our supervisor assistant professor Pawel Herman at KTH. Although we know he has a lot to do, being a teacher in several master courses, he always took time to read our report and give meticulous feedback.

We would also like to thank Magnus Röös, our supervisor at Spotify for being the perfect *to-go-to-person* for all our Spotify-specific questions. Thank you for making us feel welcomed.

We would also like to acknowledge Andreas Mattsson, working with machine learning at Spotify, for all his inspiration, his interesting questions and for him to make it super clear why machine learning is the most interesting subject.

We would also like to continue the acknowledgement with thanking Henrik Österdahl Djurfelter and the rest of the Performance Squad at Spotify for letting us be around and for making our time at Spotify very interesting and fun.

Contents

List of Abbreviations

AUC Area under curve
CD Critical distance
DT Decision tree
GB Gradient Boosting Tre
LR Logistic Regression
ROC Receiver operating characteristic
ROS Random Oversampling
RUS Random Undersampling
SMOTE Synthetic Minority Over-sampling Technique

1 Introduction

1.1 Definition of the research space

In recent years there has been an immense increase in the amount of data collected by companies. Valuable information is hidden in the data, and enables companies to improve decision making. One way to extract useful knowledge from data is by using predictive machine learning methods. Predictive machine learning methods have been used in different domains such as predicting prices and trends in the stock market [1], cancer detection in the biomedical sector [2], and credit scoring in the financial sector [3].

Another area in which predictive machine learning methods have been proved useful is by using historical customer data, to predict customer future behavior [4, 5, 6, 7]. Many companies collect data on their user's behavior and one such company is Spotify, developing on-demand services for music streaming. Spotify uses a freemium business model, meaning that it has two main products, one *free* limited and one *premium* for paying customers [8]. A great quantity of the company income comes from paying customers, referred to as premium users, and there is a large interest in keeping the amount of users converting from free to premium high. A model that can predict conversion from free to premium makes it possible to take well-founded design decisions, hopefully leading to higher conversion rate and thus increased income.

Spotify has more free users than premium users [8]. Thus, predicting conversion from free to premium can be expected to be a class-imbalanced problem. Class-imbalanced problem means that the number in one class (the majority class) is considerably greater than the number in the other class (the minority class), i.e. more non-converting users than converting users. When the number of users in the minority class is much lower compared to the majority class, it can be difficult for machine learning models to learn to correctly predict new data [9]. This is a well-known problem, known as the class imbalance problem. There are three main solutions to the problem: resampling, cost-sensitive learning, and ensemble methods [9]. Resampling means that the data used for training the machine learning models are preprocessed by adjusting the ratio between the classes to be more balanced. For some but not all class-imbalanced datasets [9, 6], resampling improves the machine learning models performance. The best resampling ratio depends both on the dataset and the classifier [10]. However, no conclusion is drawn on which resampling method and ratio to use for conversion prediction.

1.2 Aim and research questions

The aim of this study was to investigate machine learning models' abilities to predict conversion, and to examine if different resampling methods could improve performance of the models when predicting conversion.

Specifically, the goal of the study was to answer the following questions:

- *Can machine learning classification models learn from user data to predict conversion with higher performance compared to a classifier assigning all users to the majority class, i.e. non-converting users?*
- *Can preprocessing the data with resampling methods, adjusting the ratio between converters and non-converters to 30/70 or 50/50, improve the performance of the classifiers?*

1.3 Scope

In this study we investigated three different machine learning classifiers: Logistic Regression (LR), Decision Trees (DT) and Gradient Boosting trees (GB). Our focus was not on tuning the parameters of the models to state-of-art performance. Rather the focus was to answer the question if models can be trained with user data, to find patterns valuable for classifying users as converters or non-converters. In this study, the performance of the models was examined based on the area under the receiver operating characteristics curve (AUC).

The resampling methods investigated in this study are the two most common oversampling methods, Random Oversampling (ROS) and Synthetic Minority Oversampling Technique (SMOTE), and the most common undersampling method, Random Undersampling (RUS) [9]. Resampling ratios investigated were 30/70 and 50/50.

Features on user demographic, user activity and product performance were considered. Other features can be of interest when predicting conversion but were not considered in this report. Specifically, data from Spotify users, joining at ten distinct dates (20170131-20170209), were used.

2 Background

2.1 Binary classifiers and supervised learning

Similar to an earlier study on conversion prediction [11], we approached the problem as a binary classification task. Either a user is classified as a non-converting user (class 0) or as a converting user (class 1) based on user's historical data. Supervised machine learning classifiers are classifiers that can be trained using historical data with known outcome. Training of the model means iteratively updating parameters in order to reduce the classification error on training data. The aim of the trained model is thereafter to be able to classify unseen (testing) data correctly [12]. One well known problem for supervised machine learning models is the problem of overfitting. Overfitting occurs when the model is too well trained to classify the training data, but cannot generalize to the testing data. By restricting the model to not have too many parameters the problem of overfitting can be reduced [12].

Several machine learning classifiers exists today e.g. LR, DT and GB. A more detailed description of these methods will follow in the sections below.

2.1.1 Logistic Regression (LR)

LR is a linear classification model, meaning that the goal of the model is to represent a hyperplane that separates the classes in feature space [13]. Each user is represented by a number of features, and thus the users can be placed in the feature space. To model the plane, each input feature is multiplied by a weight and then all products are summed together and a bias term is added. The bias term makes it possible to shift the plane from origo in feature space. This linear sum is thereafter processed with a non-linear logistic function transforming the sum to a probability of belonging to class 1 (range [0,1]). An overview of the model and its parameters are shown in Figure 2.1.

During training, the weights and the bias term are updated. They are updated by iteratively optimizing the loss function, which is to minimize the classification error on the training data [13].

To reduce the problem of overfitting, a penalizing term on the weights' sizes can be added to the loss function. By doing so, the optimization will lead to a solution with low classification error, and in addition, with many weights near to zero. One common way is to use L2-regularization, which adds the sum of the squares of the weights as penalizing term to the loss function [12].

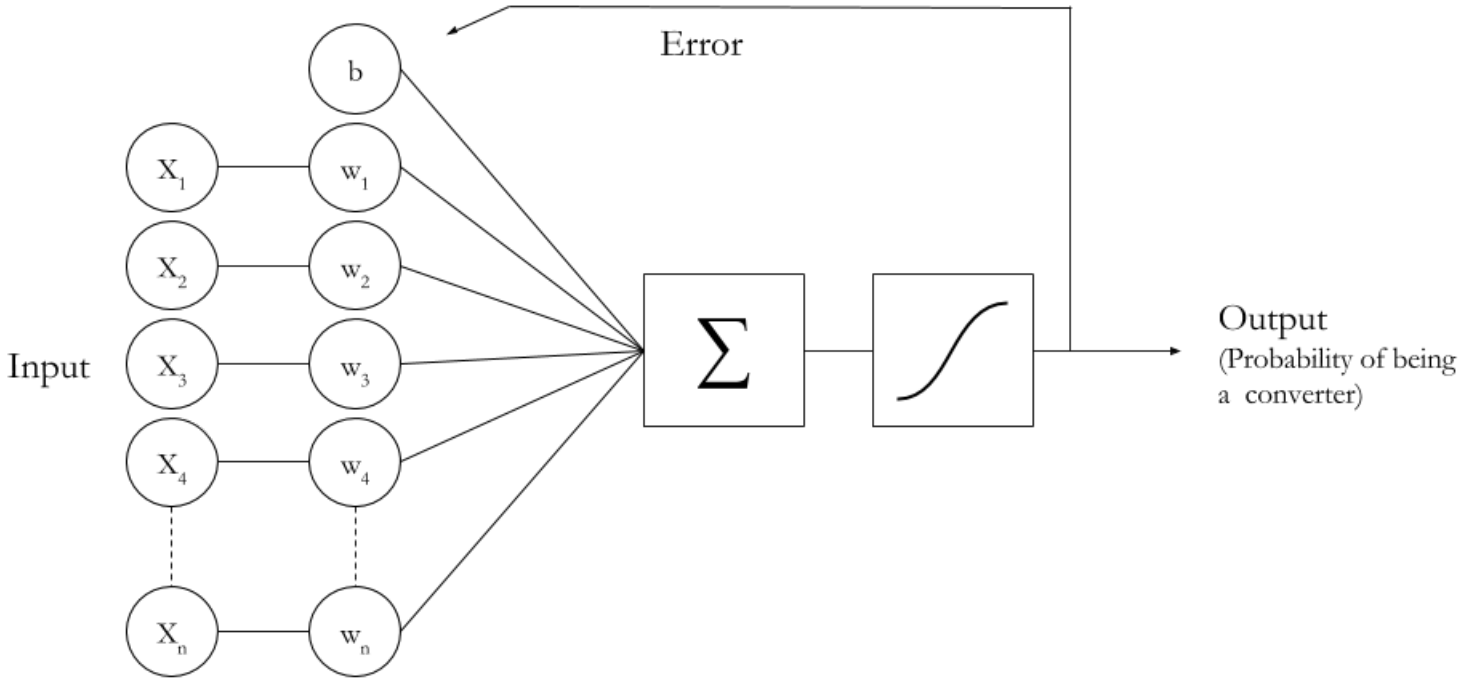

FIGURE 2.1: **Overview of the logistic regression model**
$X_1, ..., X_n$ represent features to be investigated, $w_1, ..., w_n$ are the corresponding weights of the features. b is the bias term.

2.1.2 Decision Tree (DT)

DT is a classification method that uses the features in a tree based way to classify users. An example of a DT is shown in Figure 2.2. To classify a new user, a path in the tree is followed, depending on the features of the user, from the top down to a leaf. The leaf belongs to a class, and assigns the new user to that class.

The tree is built up during the training phase. The train dataset is used to decide which features that should be in the nodes to split the dataset. The goal is to split the dataset in a way such that all leaves only have users from one of the classes. The training is done from the top, choosing a feature that best splits the data and use that at the top. Thereafter recursively find the features that best split each sub dataset until each leave only has one class or until the branch has reached a maximum depth. By limit the depth of the tree, overfitting can be reduced. If the depth is limited and a leave has more than one class, the class with majority of users is used for that leave. To decide which split that is best for a given dataset Gini impurity or information gain can be used [12].

2.1.3 Ensemble methods and Gradient Boosting Tree (GB)

Ensemble methods mean that a prediction is made by a number of simpler classifiers. Theoretically any classifier could make up an ensemble, although a very common classifier to use is DT [14]. One such ensemble method constructed of DTs is GB. In GB, narrow DTs are combined iteratively. The impact of the different trees are weighted by minimizing the loss function with gradient decent [14].

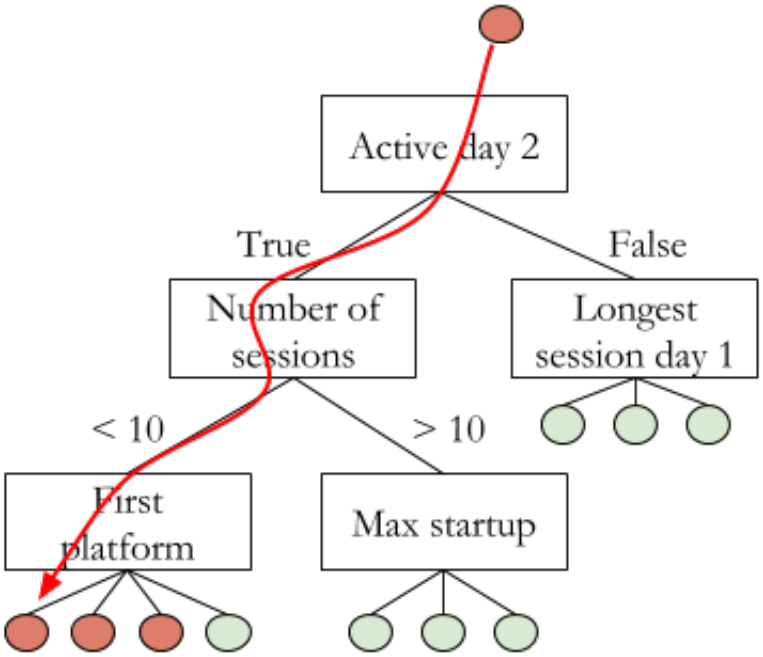

FIGURE 2.2: **Example of a Decision Tree.**
The colored circles at the bottom of the tree represent users from training data. The red circle at the top represent a user from testing data being classified.

2.2 Classification with imbalanced data

An imbalanced dataset in a binary classification problem refers to unequal number of users between the classes. The class imbalance problem is well known [9]. One reason is the tendency of binary classifiers learning only to classify data as if belonging to the majority class. However,in our context, correctly classifying the converters was of interest. In other areas, correctly classifying the minority class might be even more important e.g. when predicting diseases. In the following section examples of solutions is presented.

2.2.1 Solutions to the class-imbalance problem

In order to handle the class-imbalance problem there are three main categories of solutions: resampling, cost-sensitive learning and ensemble methods [9]. Resampling aims to balance the ratio between the classes of data. For binary classification, it can either be performed by duplicating samples of the minority class (oversampling) or by eliminating samples of the majority class (undersampling) [9]. Cost-sensitive learning opposes the imbalance problem by penalizing misclassification of the minority class with a higher cost than with samples from the majority class during training [9]. Ensemble methods consists of multiple classifiers combined, such as GB. Each classifiers' output is weighted depending on how it classifies minority samples [14].

2.2.2 Resampling methods

Three widely used methods for resampling are: ROS, RUS and SMOTE [9]. ROS duplicates randomly selected users from the minority class and add these to the

dataset until a preferred ratio has been reached. RUS does the opposite i.e randomly eliminates users from the majority class until preferred ratio has been reached [9]. SMOTE as presented by Chawla et al. [15] is an oversampling method which creates synthetic users similar but not equal to existing minority users and add those to the dataset. By selecting a random user in the minority class, and investigating an other minority user similar to the randomly chosen user, SMOTE creates a new synthetic user which is a mix of the two users considered.

2.3 Related work

Researchers have not treated conversion prediction in much detail. Sifa et al. [11] compared models for premium conversion in free-to-play games. They concluded that it was possible to valuably predict premium conversion with machine learning models (overall accuracy: 0.997 and recall: 0.439) [11]. Specifically, DT, random forest and support vector machine were investigated. When training data were preprocessed with oversampling, all models performed better regards to correctly classifying converting users [11].

While the body of literature available in conversion prediction is limited [11], work in associated domains, such as churn prediction, have a more established history. Churn prediction studies extend across various of markets, such as the telecommunication industry [4, 5, 6], the energy sector [7] and the gaming industry [16, 17, 18]. Beyond different markets there is a large variety in which methods the studies investigate; LR [4, 5, 6] , DT [4, 5, 6, 17, 16] and GB [6] have been investigated.

Churn prediction and the effect of oversampling to 50/50 ratio were studied by Verbeke et al. [6]. They used several different churn datasets, and investigated several different classifiers. Preprocessing training data with oversampling improved performance for some, but not for all, datasets. No significant general improvement of performance using oversampling was shown [6]. On the other hand, Burez et al. [10] who also studied churn prediction and resampling methods, showed that resampling improved performance on all their churn datasets. Additionally, they showed that undersampling can improve performance.

Interestingly, Zous [19] argued that undersampling can be superior to oversampling because it can lead to same performance but needs less computational power. Regarding resampling ratio, Burez et al. [10] showed that 50/50 ratio is not always the preferable ratio. The best resampling ratio depended both on the dataset and the classifier [10].

3 Method

3.1 Data collection of Spotify data

Data were collected from Spotify users registered to the free version on ten distinct days. The registration dates reach from 31th of January 2017 (Day 1) to 9th of February 2017 (Day 10). For each of the ten dates, a cohort of approximately $50,000$ users, with no missing data, was randomly selected to give a dataset with 2% converting users ($n \approx 1000$). Not all users registered at the inclusion dates were included in the cohort, since real numbers can reveal business critical data.

For each user, the following areas of data were collected: user activity, demographics and the applications performance in the means of startup time. All data collected were from the users first seven days since registration, except for the output. The output, conversion, was defined as being a paying premium user at exactly 14 days after registration. A summary of the input features used is presented in Table 3.1.

TABLE 3.1: **Summary of features.**
Cat: Categorical feature, Cont: Continuous feature

Feature	Type
Registration country	Cat
First platform	Cat
Free or Premium (Day 2 and 7)	Cat
Active (Day 1, 2 and 7)	Cat
Total active days (Day 1, 2 and 7)	Cont
Total consumed hours	Cont
Number of sessions (Day 1 and 7)	Cont
Total session length (Day 1 and 7)	Cont
Shortest session (Day 1 and 7)	Cont
Longest session (Day 1 and 7)	Cont
Length of first session	Cont
Number of sessions < 10 min (Day 7)	Cont
Number of sessions > 10 min and < 60 min (Day 7)	Cont
Number of sessions > 60 min (Day 7)	Cont
Mean startup (slow, fast, superfast, no-startup) (Day 1, 2 and 7)	Cat
Max startup (slow, fast, superfast, no-startup) (Day 1, 2 and 7)	Cat

3.2 Data formatting

The ten datasets (Day 1 - Day 10) were each split into a train dataset (2/3) (following referred to as the baseline dataset) and a test dataset (1/3). Before the datasets were used in the machine learning models the categorical data were encoded as numbers. All categories, for each categorical feature, were given an own feature representation where users belonging to that category got a value 1. All the other users got a value 0 for that feature. Further, normalization is acknowledged to increase accuracy for various machine learning methods [12]. Thus all data were normalized. Since test data cannot be seen before testing, the mean and variance calculated on training data were used to normalize test data.

3.2.1 Resampling

The ten datasets (Day 1 - Day 10) were heavily imbalanced with a initial class-ratio of 2/98 (converters/non-converters). To handle the imbalance there are three solutions: resampling, cost-sensitive learning and ensemble methods, as mentioned in section 2.2.1. Resampling was the main focus of this study due to its simplicity and compatibility with existing machine learning models at Spotify. For resampling the baseline datasets, this research utilized three different techniques: ROS, RUS and SMOTE. ROS and SMOTE were chosen as oversampling methods because of its widely usage [9]. Furthermore, ROS is an intuitive way of balancing data, whereas SMOTE is more complex creating synthetic samples. For undersampling RUS is chosen, since it is considered both simple yet effective [9].

Each resampling method was used with two different ratios: 50/50 and 30/70. Instead of only fully balancing the datasets, i.e. ratio 50/50, a ratio in between 50/50 and the baseline ratio (2/98) was chosen. This was done since Burez et al. [10] showed that 50/50 does not always lead to the best performance. Due to limited scope, not all ratios tested by Burez et al. [10] were tested in this study. Resampling of the training data thus resulted in a total of 70 train datasets (1 baseline, 2 ROS, 2 RUS and 2 SMOTE per day). An overview of the dataset modifications, splitting and resampling, can be seen in Figure 3.1. Importantly no baseline dataset (Day 1 - Day 10) share the same users and thus are independent. However, the datasets made from the same baseline dataset share users, and thus are not independent from each other.

3.3 Training the classification models

All train datasets (70 in total) were used to train learnable classification methods of three different kinds: LR, DT and GB. LR is chosen because of the simplicity of the method. DT and GB were chosen since Sifa et al. [11] showed that tree-based classifiers had great performance on a similar conversion problem. The models' performances depend on hyperparameter settings and the amount of training. The settings of the models are presented below.

3.3.1 Logistic regression (LR)

In order to prevent overfitting a L2-regularization term (a penalizing tern) was added to the loss function. A parameter search with 5-fold cross validation was conducted to set the impact of the regularization term.

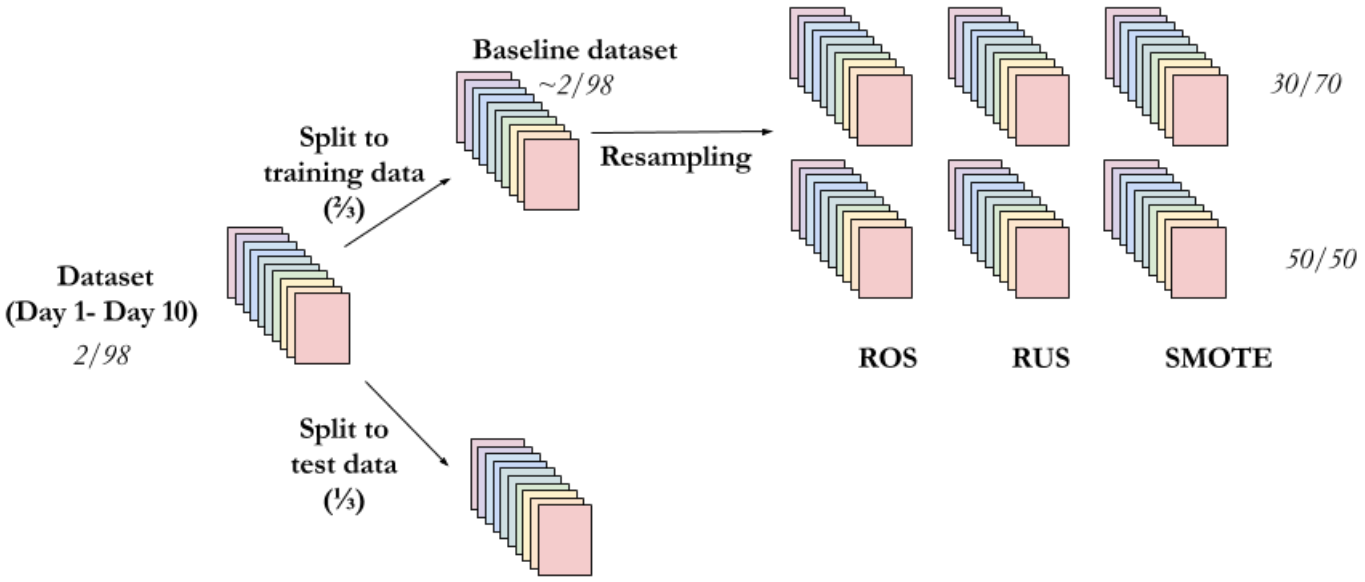

FIGURE 3.1: **Overview of datasets used.**
Numbers in italic are the ratio between converters/non-converters in the dataset.
ROS: Random Oversampling, RUS: Random Undersampling, SMOTE: Synthetic Minority Oversampling Technique.

3.3.2 Decision Tree (DT)

The structure of the tree was set to a maximal depth of 10 and the minimum number of samples required to be at a leaf node was set to 5. As a criterion to value the quality of a split the Gini Impurity function was used [13].

3.3.3 Gradient Boosting Classifier (GB)

The gradient boosting classifiers were built on a tree structure with a maximal depth of 6. In order to prevent overfitting, L2-regularization was used. Moreover, the learning rate was tuned to 0.3.

3.4 Model evaluation

TABLE 3.2: **Confusion matrix**

	Predicted true	**Predicted false**
Actual true	True Positive	False Negative
Actual false	False Positive	True Negative

In our models, two fundamental errors may occur: classifying a non-converter falsely as a converter, and classifying a converter falsely as a non-converter. These errors are more commonly known as false positive and false negative results. Other possible classifications will be correct i.e. true positive and true negative results. The correlation between these are presented in a confusion matrix in Table 3.2.

3.4.1 Performance metric - AUC

Looking at the percentage of correctly classified users (accuracy) is problematic for class-imbalanced data. Classifying only the majority class (non-converters) correctly would result in a high accuracy, although the minority class (converters) is of interest. Thus a more complex and widely used metric for imbalanced data is the AUC [9]. It minimize the negative influence of class-imbalance, by examinating both true positive rate and false positive rate and was therefore adapted in this report. The following paragraph will describe the AUC metric in more detail.

AUC is the area under the receiver operating characteristics curve (ROC). ROC is used to visualize true positive rate in relation to false positive rate. For a single run of a classifier, every user retrieved a probability value. A threshold value was then run through the result, starting from 1 decreasing down to 0. All probability values higher than the threshold were classified as positive (converters) whereas the rest were classified as negative (non-converters). For each threshold, the true positive rate and the false positive rate were calculated which generates a point on the ROC-curve [12]. An example of a ROC curve is presented in Figure 3.2.

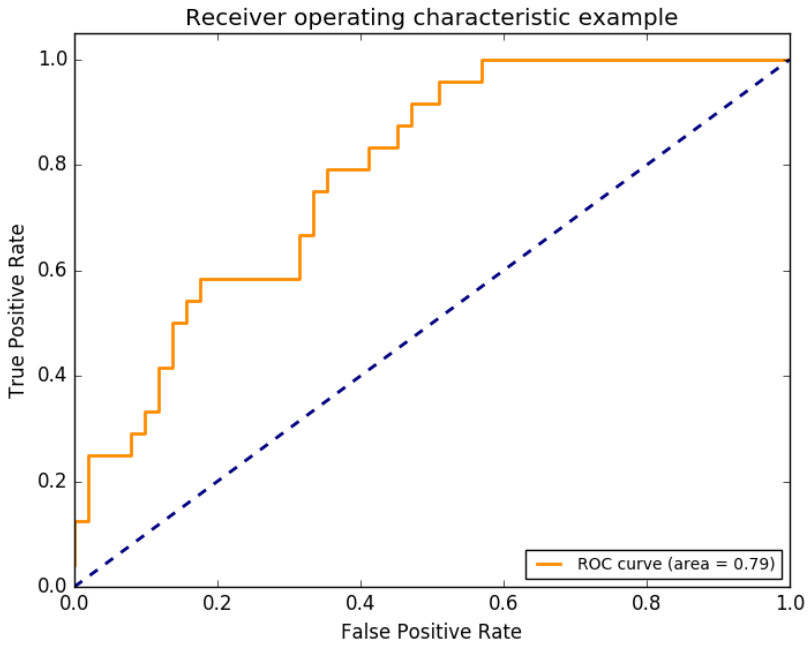

FIGURE 3.2: **Example of a ROC curve.**
AUC is defined as the area under the ROC curve. Figure taken from Scikit Learn [20].

A perfect run would generate a point with 100% true positive and no false positive i.e. a point at (0,1). Thus, an optimal ROC curve is fully extended in the top-left corner leading to AUC 1 [13]. A classifier that assigns all as the majority class, would result in AUC 0.5. Hence, a good performance would be an AUC in between 0.5 and 1 where the higher the AUC score, the better the classifier.

3.4.2 Experiments and statistical tests

Three main experiments were conducted. Firstly, one comparing the AUC performance of the classifiers on the baseline datasets to a classifier assigning all users as non-converters. Secondly, independently for each classifier, compare the difference in AUC performance between the different sampling settings. Thirdly, compare the classifiers to each other, by comparing the best sampling setting for each classifier to one another.

In all experiments, Friedman test with post hoc Nemenyi tests were used to test if differences in AUC performance were significant, a procedure inspired by Demsar et al [21]. Friedman test is a non-parametric test, using the average rank of the different settings. When comparing different classifiers, the rank is calculated for each classifier; When comparing the sampling settings, the rank is calculated for each sampling setting. Friedman test statistic is defined in Equation 3.1, with R_j as the average rank for the setting $j = 1, ..., k$, and N as the number of datasets.

$$\chi_F^2 = \frac{12N}{k(k+1)}[\sum_j R_j^2 - \frac{k(k+1)^2}{4}] \tag{3.1}$$

Under the null hypothesis that there is no difference in AUC performance between the different settings, χ_F^2 has a χ^2 distribution with $k-1$ degrees of freedom. The null hypothesis is rejected if the p-value, calculated by the means of χ^2 cumulative density function, is less than decided significance level. In all our tests, p-values less than 0.05 were considered significant.

Friedman test answers the question *if* there is a difference between the settings. Thus, if Friedman test reject the null hypothesis, post hoc tests need to be conducted to decide *which* settings differs from each other. Nemenyi post hoc test is one such test [21]. Nemenyi test uses critical distance (CD) calculated as defined in equation 3.2. The q_α is chosen from tables dependent on the number of settings and significance level: k and α [21]. In this report α is chosen to be 0.05. If the average rank for one setting differs more than CD from the average rank for another setting, the difference in AUC is considered to be statistically significant.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{3.2}$$

For the first experiment, when the three classifiers were compared to a classifier that assigns all users as non-converters, the AUC was evaluated for each model trained with the baseline datasets (Day 1 - Day 10). For each test dataset, the classifiers got a rank between 1 and 4. The classifier getting the highest AUC value got rank 1 and the one with the lowest got rank 4. The average rank for each classifier was calculated. Thereafter, Friedmans test statistic was calculated according to equation 3.1 with $k = 4$ and $N = 10$. Pairwise comparisons of the classifiers were thereafter performed with Nemenyi test, calculating the CD according to equation 3.2 with $k = 4$ and $q_{0.05} = 2.569$.

In the second experiment, conducted independently for each classifier, Friedman test was performed to examined if the difference in mean AUC performance for the 7 different sampling settings was due to chance. The ranking was calculated for each test dataset giving each sampling setting a ranking between 1 and 7. Friedman test statistic was calculated according to equation 3.1 with $k = 7$ and $N = 10$. Post hoc Nemenyi test with pairwise comparisons of the sampling settings was performed with $k = 7$ and $q_{0.05} = 2.949$.

In the third experiment, comparing the best sampling setting for each classifier to one another, the Friedman test was performed similarly to the first experiment. The rank was this time calculated for the three best performing classifiers. Since we during this experiment compared three classifiers, as opposed to four in the first experiment (which included the classifier assigning all as non-converters), we used $k = 3$ instead of $k = 4$. For Nemenyi test we thus used $q_{0.05} = 2.343$

3.5 Implementation frameworks

For the purpose of implementing data formatting and model training, Sci-kit learn and NumPy were used [20]. Both Sci-kit learn and NumPy are open source libraries for machine learning in Python. The hypothesis testing was done in another open source library named Scipy [22], designed for mathematics in Python. The gradient boosting classification model was implemented using a python library named XGBoost [23].

4 Results

The result section is divided into four parts. The first part describes the resulting baseline datasets after the data collection. The second part describes the results from experiments focusing on the first research question. The third part describes results from experiments focusing on the second research question. Finally, the fourth part describes results on which classifier of the ones tested that performed best overall.

4.1 Baseline datasets

The data collection resulted in data from 523,838 unique users. Description of the baseline datasets (Day 1 - Day 10) are presented in Table 4.1. Given our method, the dataset before splitting into train and test sets had 2 % converters. As seen in Table 4.1 the random splitting into train datasets resulted in datasets with approximately 2 % converting users each.

TABLE 4.1: **Descriptive statistics of baseline datasets.**

Dataset	#Users	#Converters
Day 1	34604	712
Day 2	35040	692
Day 3	34939	704
Day 4	34973	699
Day 5	37753	761
Day 6	37418	743
Day 7	33700	681
Day 8	33231	683
Day 9	34202	678
Day 10	33365	702

4.2 Ability to predict conversion

Separately for each classifier method, mean value and standard deviation of AUC measurements from models trained on the 10 baseline datasets (Day 1 - Day 10, no sampling) are presented in Table 4.2. Friedman test showed that there was a significant difference in AUC performance between the classifiers and thus Nemenyi test was performed. The results of the Nemenyi test when the classifiers are pairwise compared to a classifier assigning all users as non-converters, show that LR, DT and GB perform significantly better regarding mean AUC performance ($p<0.05$) (4.2). No difference was shown between the other classifiers (LR, DT, GB).

TABLE 4.2: **Mean AUC performance of the 10 models of each classifier that were trained with the baseline datasets.**
DT: Decision Tree, GB: Gradient Boosting, LR: Logistic Regression, SD: Standard deviation.
* Results from Nemenyi test for comparison with a classifier assigning all users as non-converters.

	mean	SD	Nemenyi test*
LR	0.822	0.014	Difference
DT	0.824	0.013	Difference
GB	0.822	0.014	Difference

4.3 Comparing sampling settings

Mean AUC performance of models trained on differently sampled datasets (average over 10 models per sampling setting) are presented in Table 4.3. For all classification methods, the Friedman test showed that the differences in AUC between the sampling settings were not due to chance ($p<0.001$). Therefore, independently for each classifier method, post hoc Nemenyi tests were conducted. Results from the pairwise Nemenyi tests, comparing the sampling settings to baseline, are presented in Table 4.3. Mean AUC values that were significantly different from baseline AUC values are in bold. The highest value for each classifier is underlined. The difference between the best value and the other bold marked values was not significant regarding Nemenyi test for any of the classifiers. For all classifiers, models trained with ROS 50-50 ratio performed best and were significantly better than models trained with the baseline dataset.

TABLE 4.3: **Mean AUC performance for different sampling settings averaged over 10 models of type A: Logistic Regrssion, B: Decision Tree, C: Gradient Boosting tree.**
For each classifier, bold marked results are significantly better than classifying the baseline distribution. Underlined results show the best sampling setting for each classifier. ROS: Random Oversampling, RUS: Random Undersampling, SMOTE: Synthetic Minority Oversampling Technique
* Class ratio: converter/non-converter

Panel A. Logistic Regression							
Sampling	Baseline	ROS		RUS		SMOTE	
	*2/98**	*30/70**	*50/50**	*30/70**	*50/50**	*30/70**	*50/50**
AUC	0.822	**0.866**	**0.869**	**0.865**	**0.867**	0.844	**0.861**
Panel B. Decision Tree							
Sampling	Baseline	ROS		RUS		SMOTE	
	*2/98**	*30/70**	*50/50**	*30/70**	*50/50**	*30/70**	*50/50**
AUC	0.824	**0.850**	**0.854**	**0.849**	0.831	0.828	0.828
Panel C. Gradient Boosting tree							
Sampling	Baseline	ROS		RUS		SMOTE	
	*2/98**	*30/70**	*50/50**	*30/70**	*50/50**	*30/70**	*50/50**
AUC	0.822	**0.865**	**0.873**	**0.867**	**0.870**	0.822	0.825

4.4 Comparing the classifiers

For each classifier, mean AUC performance of the model that performed best in the sampling experiment (Table 4.3) is represented in Table 4.4. For all classifiers, the best sampling setting was ROS with 50-50 sampling ratio. Friedman test showed that the difference between AUC performance for these models was not due to chance ($p<0.001$), and thus Nemenyi test was performed. The results of the Nemenyi test showed that there was no significant difference between the AUC for LR and GB ($p<0.05$, Table 4.4). Conversely, DT had significantly lower AUC performance compared to the other two ($p<0.05$).

TABLE 4.4: **Comparing the classifiers to each other.**
DT: Decision Tree, GB: Gradient Boosting, LR: Logistic Regression, ROS: Random Oversampling, RUS: Random Undersampling, SMOTE: Synthetic Minority Oversampling Technique
* Results from Nemenyi test for comparison with the classifier with highest mean AUC (GB ROS 50-50).

	mean	Nemenyi test *
LR with ROS 50-50	0.869	No difference
DT with ROS 50-50	0.854	Difference
GB with ROS 50-50	0.873	-

5 Discussion

5.1 Discussion of the results

In this report, three machine learning methods' abilities to predict conversion and the effect of resampling on those models were investigated. The results of this study showed that machine learning methods can be trained with user data to predict conversion with higher AUC performance compared to a classifier assigning all users as non-converters. Additionally, the results showed that the performance of the classifiers can be improved by preprocessing the data with resampling.

Our results are in alignment with those obtained by Sifa et al. [11]. As described in section 2.3, they observed that machine learning classifiers were able to correctly classify both the majority class and the minority class with relatively high accuracy [11] and that the performance improved with oversampling. They investigated DT, random forest and support vector machine. Thus, our results are in agreement with Sifa et al. [11] regarding DT and bring new insights for the effect of resampling when predicting conversion with LR and GB.

It is interesting to note that all classifiers got significantly better results when training data were preprocessed with ROS, at both 30/70 and 50/50 sampling ratio. Thus, if there is enough computational power, ROS could be suggested as a preprocessing technique when building conversion prediction models. Additionally, it is worth noting that LR and GB were actually able to perform significantly better than baseline when training data were preprocessed with RUS 50/50. This is interesting since RUS with ratio 50/50 only train on $\sim$4 % of the train dataset and this is computationally very beneficial. These results are in line with Zous' [19] conclusion that RUS are superior to oversampling methods when the minority class have hundreds of observations.

Interestingly, resampling with SMOTE did not lead to improved performance (except in one case). Since SMOTE for each added user creates a user who is a mix of two converters [15], this could indicate that the classes of converters and non-converters overlap in feature space. By mixing two converters, if the classes overlap, the new user might finishes in feature space where it is more alike a non-converter. Thus, in such case, SMOTE resampling might just increase the overlap of the classes and thus not make it easier for the classifier to learn the decision boundary.

In agreement with Burez et al. [10], who examined class imbalance in churn prediction, we see that 50/50 resampling ratio was not always the best ratio. We see that DT performed significantly better with RUS with ratio 30/70 compared to RUS with ratio 50/50. A possible explanation to this might be that RUS with ratio 50/50 discarded too much valuable information for DT to be able to find informative decision features.

The findings of this study showed that although ROS 50/50 gave the highest performance for all classifiers, no significant difference was shown between ROS 50/50 ratio and RUS 30/70 ratio for any of the classifiers. Thus, a simple classifier

such as LR run on preprocessed data with RUS 30/70 could be recommended for this dataset, since it showed no significant difference in performance from the best result and needs less computational power.

5.2 Limitations

The generalizability of our findings can be limited since we only look at data from users joining Spotify in a specific timespan, ten days from the end of January to the beginning of February 2017. It is possible that the user behavior is different at other times of the year, and thus selection bias cannot be ruled out. Likewise, the research was only conducted on Spotify's specific data, therefore we cannot conclude the result to be applicable for conversion prediction in general, thus further limiting the generalizability. It is possible that the study will show different results for datasets collected from other freemium companies.

Additionally, our findings may be somewhat limited by only investigating AUC as a performance metric. Although the AUC minimizes the negative influence of the class-imbalance, it cannot distinguish between the contribution of each class to the overall AUC performance [24]. This means that different combinations of true positive rate and true negative rate may produce the same AUC result. Other metrics, such as true positive rate, could have been interesting to analyze to see how many of the converters that actually were correctly classified. However, for readability of the report we focused on one metric and chose AUC since it is the most commonly used for class-imbalanced problems [9].

Since the best ratio may differ between both classifiers and datasets [10], this study is rather limited when investigating only two alternative ratios on one dataset. Nevertheless, Burez et al. [10] showed that although the best ratio may vary between classifier and dataset the results between ratios up to 50/50 does not differ significantly in many cases. Thus the limitation of two ratios in this study does not necessarily have a negative impact on the results.

This study did not handle the technique of feature selection, which might limit the performance of the classifiers by increasing the risk of overfitting. In the study the classifiers were, however, modeled with L2 regularization and maximum tree depth which contribute to counteract the risk of overfitting. Additionally, in a lower dimensional space, which is a result of feature selection, the converters might be closer to each other and thus resampling with SMOTE might work better. It would have been interesting to investigate if feature selection would have led to better results for SMOTE. Due to time limitations this was out of the scope for this study.

5.3 Future work

In addition to resampling, there are two more main ways to handle the class imbalance problem: cost-sensitive learning and ensemble methods [9]. Even though we did look at one ensemble method (GB) further research should be undertaken to investigate ensemble methods in more detail and investigate cost-sensitive learning in the process of predicting conversion.

In addition, it would be interesting to collecting data from users from more than ten days to see if these results are generalizable over the seasons. The freemium business model is common, and it would also be interesting to see if these findings are applicable to more companies. Thus, doing similar studies at more companies

with the freemium business model would indicate if our results are generalizable to other companies.

Except for DT with RUS and LR with SMOTE the results between 30/70 and 50/50 do not differ significantly. Therefore it would be interesting to investigate more ratios in alignment with Burez et al. [10] and see if a ratio of for example 10/90 would generate an improvement in the same magnitude.

6 Conclusion

The result of this study showed that machine learning classifiers are able to predict conversion from free to premium with higher AUC performance compared to a classifier assigning all users as non-converters. Additionally, the performance of the predicting classifiers can be improved by preprocessing the data with different resampling methods.

Bibliography

[1] J. Alam and J. Ljungehed. "A comparative study of hybrid artificial neural network models for one-day stock price prediction". In: (2015).

[2] H. Yu, J. Ni, Y. Dan, and S. Xu. "Mining and integrating reliable decision rules for imbalanced cancer gene expression data sets". In: *Tsinghua Science and Technology* 17.6 (Dec. 2012), pp. 666–673.

[3] P. Yao. "Credit Scoring Using Ensemble Machine Learning". In: *2009 Ninth International Conference on Hybrid Intelligent Systems*. IEEE, 2009, pp. 244–246.

[4] K. Coussement, S. Lessmann, and G. Verstraeten. "A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry". In: *Decision Support Systems* 95 (2016), pp. 27–36.

[5] T. Vafeiadis, K. I. Diamantaras, G. Sarigiannidis, and K. C. Chatzisavvas. "A comparison of machine learning techniques for customer churn prediction". In: *Simulation Modelling Practice and Theory* 55 (2015), pp. 1–9.

[6] W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens. "New insights into churn prediction in the telecommunication sector: A profit driven data mining approach". In: *European Journal of Operational Research* 218 (2012), pp. 211–229.

[7] J. Moeyersoms and D. Martens. "Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector". In: *Decision Support Systems* 72 (2015), pp. 72–81.

[8] Spotify. *About Spotify*. Visited: 2017-05-29. 2017. URL: https://press.spotify.com/co/about/.

[9] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73 (2017), pp. 220–239.

[10] J. Burez and D. Van den Poel. "Handling class imbalance in customer churn prediction". In: *Expert Systems with Applications* 36.3 (Apr. 2009), pp. 4626–4636.

[11] R. Sifa, S. Augustin, F. Hadiji, J. Runge, A. Drachen, K. Kersting, C. Bauckhage, F. Iais, and S. Augustin. "Predicting Purchase Decisions in Mobile Free-to-Play Games". In: *Proceedings, The Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-15)* (2015), pp. 79–85.

[12] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*. Vol. 103. Springer Texts in Statistics. New York, NY: Springer New York, 2013.

[13] S. Marsland. *Machine Learning : an algorithmic perspective*. Second. Boca Raton, FL: CRC Press, 2015.

[14] C. Zhang and Y. Ma, eds. *Ensemble Machine Learning*. Boston, MA: Springer US, 2012.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[16] S.-K. Lee, S.-J. Hong, S.-I. Yang, and H. Lee. "Predicting churn in mobile free-to-play games". In: *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Oct. 2016, pp. 1046–1048.

[17] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. "Predicting player churn in the wild". In: *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, Aug. 2014, pp. 1–8.

[18] R Sifa, S Srikanth, A Drachen, and C Ojeda. "Predicting Retention in Sandbox Games with Tensor Factorization-based Representation Learning". In: *Proc. of IEEE* (2016).

[19] L. Zhou. "Performance of corporate bankruptcy prediction models on imbalanced dataset: The effect of sampling methods". In: *Knowledge-Based Systems* 41 (2013), pp. 16–25.

[20] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. *Scikit-learn: Machine Learning in {P}ython*. 2011.

[21] J. Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *Journal of Machine Learning Research* 7 (2006), pp. 1–30.

[22] E. Jones, T. Oliphant, P. Peterson, and others. *{SciPy}: Open source scientific tools for {Python}*. Visited: 2017-05-29. URL: http://www.scipy.org/.

[23] *XGBoost Documents*. Visited: 2017-05-29. 2016. URL: https://xgboost.readthedocs.io/en/latest/.

[24] V. García, J. S. Sánchez, and R. A. Mollineda. "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance". In: *Knowledge-Based Systems* 25.1 (2012), pp. 13–21.