**Customer Churn Prediction in Banking Industry**

**Martin Pažický, Roman Berešík**

# Introduction

Customer churn is defined as the inclination of customers to stop using or purchasing a company’s product in a given time period. It is a common measure of lost customers. Companies have started to realize that they should make an effort not only to seek new customers, but also to retain the existing ones.

Companies, especially banks, often collect lots of data about their clients. One of the options banks have, to fight against customer churn, is to use the data and train a machine learning model, which might predict when a customer is about to churn their services. Banks could then approach the client and try to resolve the reasons that led him to this decision.

In this project we are going analyze the data of a bank and try to predict customer churn using several machine learning models – random forest, logistic regression, and TODO: (neural network, mozno rozsirit introduction trochu). We are going compare the methods and choose the best performing one.

# Related Work

This topic got very popular due to the huge amounts of data banks have collected and many papers trying to predict customer churn have already been published. For example, in [1] authors use data mining tools such as Naïve Bayes or support vector machines to predict the churn behavior among Indian bank customers. Another paper [2] uses several classifiers to predict who is going to churn the bank and a comparison between the classifiers is conducted, of which Random Forest model comes out as the best. Todo: moznopridat nejaku pracu

# Exploratory Data Analysis

In this project we are going to use a dataset which describes customers of a bank based on their credit card activity and their demographic data (<https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers>). The data consists of 10127 records described by 21 attributes, of which 15 are numerical and the other 6 are categorical. The attributes of the data are described in table 1. Our goal will be to classify customers into 2 groups based on the *Attrition\_Flag* attribute – the ones who left the bank and the ones that stayed.

|  |  |  |
| --- | --- | --- |
| Názov | Typ | Popis |
| CLIENTNUM | Numerical | Identificator |
| Attrition\_Flag | Categorical | Customer churn (target var) |
| Customer\_Age | Numerical | Age |
| Gender | Categorical | Gender |
| Dependant\_count | Numerical | Number of dependents |
| Education\_Level | Categorical | Education |
| Marital\_Status | Categorical | Marital status (single/married/divorced) |
| Income\_Category | Categorical | Annual income |
| Card\_Category | Categorical | Card type |
| Months\_on\_book | Numerical | Length of relationship with the bank (number of months) |
| Total\_Relationship\_Count | Numerical | Number of products held |
| Months\_Inactive\_12\_mon | Numerical | Number of inactive months in the last year |
| Contacts\_Count\_12\_mon | Numerical | Number of contacts in the last year |
| Credit\_Limit | Numerical | Credit limit on the card (maximum debt) |
| Total\_Revolving\_Bal | Numerical | Revolving balance (carried debt) |
| Avg\_Open\_To\_Buy | Numerical | Available credit (average over the last year) |
| Total\_Amt\_Chng\_Q4\_Q1 | Numerical | Change in the amount of transactions |
| Total\_Trans\_Amt | Numerical | Total amount of transactions (last year) |
| Total\_Trans\_Ct | Numerical | Total number of transactions (last year) |
| Total\_Ct\_Chng\_Q4\_Q1 | Numerical | Change in the number of transactions |
| Avg\_Utilization\_Ratio | Numerical | Credit card debt to credit limit ratio |

Table 1: Data description

During the exploration of the data, we noticed that the distribution of our target variable is imbalanced, and the positive outcomes (customers that stay in the bank) considerably outnumber the negative outcomes. We will have to apply proper methods in order to deal with this situation while training models.

A picture containing text, screenshot, font, diagram

Description automatically generated

Figure 1: Attrition flag distribution

We have also examined the bivariate relationships between different columns of the data. We visualized the distributions of each numerical column in relation to the target variable using boxplots. In some variables a significant difference between the distributions with respect to the target variable could be spotted. We can for example see that the churners tend to have a smaller count of transactions than the non-churners (*figure 2*).

A picture containing text, screenshot, diagram, line

Description automatically generated

Figure 2: Transaction count distribution with respect to attrition flag

We have also looked into the relationships of categorical columns with the target variable. On *figure 3* we can see graphs with the percentage of attrited customers per category of a given variable. Each category has an annotation carrying the information about its support in the data (what percentage of records contains given value in the column) to emphasize the importance of the information. The red line in each graph depicts the percentage of attritied customers in whole dataset in order to help to portray the difference between attrition percentage of a given category and the average attrition percentage. We can for example see that the churn rate of customers holding a platinum card is remarkably high, however not many customers hold a platinum card. We can also spot that customers with income between 60 and 80 thousand dollars tend to stay in the bank.

A screenshot of a graph

Description automatically generated with low confidence

Figure 3: Attrition of customers in different classes of categorical variables

## Handling Missing Values

The data did not contain any null values, however we noticed that some variables had *Unknown* values present. We considered those values missing and applied the imputation methods on these variables.

Missing values were only present in 3 variables, all of which are categorical.

|  |  |
| --- | --- |
| Column | Missing values number |
| Education\_Level | 1519 |
| Marital\_Status | 749 |
| Income\_Category | 1112 |

Table 2: Missing values

To impute the missing values two approaches were used and evaluated. The first approach was to impute the values using decision tree. For each attribute with missing values, the tree was first trained on records with valid values in that column. The attribute that contained the missing values was chosen as the target variable, while the other attributes of the dataset were used as predictor variables. After the training phase, the records that contained missing value in the given attribute could be passed to the model, which decided what value will be used instead of null.

Before training the tree, it was also necessary to temporarily deal with missing values in the predictor variables. Since the attributes with missing values are exclusively categorical, the most frequent value occurring in the given attributes was chosen as a temporary replacement. Decision tree is also not able to work with categorical variables, therefore we transformed categorical variables into numerical using one hot encoding.

Another approach we used was imputation using k-nearest neighbors algorithm (KNN). To be able to use KNN, similarly as in previous approach, some preprocessing needed to be done. The same preprocessing steps as in the decision tree imputation were applied, with one extra step, which was normalization of numerical values using min-max scaling, to prevent one variable from having significantly higher impact on the result, than others. Otherwise, the process was the same as in the first approach (temporary imputation using the most frequent value, target / predictor variables choice), the only change was in the choice of model which was used.

To assess these two approaches, splitting of the dataset to training and test data was used. The training data were used to build the models, then the predictions of model were evaluated on the test data. The evaluations and splits were done for each variable which contained missing data separately, and the scores were then averaged for each method. As a metric to evaluate the results of different models accuracy was used, since in this case there is not a big difference between the cost of false positives and false negatives.

Assessment results can be seen in *table 3*. Each row represents one method and the numbers in the cells represent the test accuracy score of a particular method for corresponding column. We can see that the decision tree has performed better on all the variables. Therefore, we decided to proceed with the dataset imputed by the decision tree method.

A screenshot of a computer

Description automatically generated with low confidence

Table 3: imputation methods evaluation

## Data Transformation

In order to use the data for training the selected machine learning models, various data adjustments had to be made. The models are not able to deal with categorical data directly, therefore we needed to transform them to numerical data first. Applying domain knowledge allowed us to spot certain hierarchical relationships between the values of various features. To preserve the relationships in the transformed features ordinal encoding method was applied. The mentioned hierarchies are depicted in *table 4*.

|  |  |
| --- | --- |
| Column | Order |
| Card\_Category | Blue < Silver < Gold < Platinum |
| Income\_Category | Less than $40K < $40K - $60K < $60K - $80K < $80K - $120K < $120K + |
| Education\_Level | Uneducated < High School < College < Graduate < Post-Graduate < Doctorate |

Table 4: Ordinal relations in categorical variables

To deal with other categorical columns one-hot encoding was used – a popular technique used to represent categorical variables as binary vectors. This technique has one drawback which is increasing the dimensionality of a dataset, especially if there are categorical variables with a large number of unique categories. This, however, is not our case and the increase in the number of variables is minor.

## Outlier Detection

Todo:

## Feature Selection

Feature selection is a technique that can help to reduce the training time of a model. Even though we don’t have too many features in our dataset and the training is quite fast, feature selection might also improve the performance of a model by reducing noise or getting rid of irrelevant data.

We have chosen a feature selection method from *sklearn* library – *SelectKBest.* It is a method used to select the k most informative features from a given dataset. It operates by scoring the features based on their individual relationship with the target variable and selecting the top k features with the highest scores. The scoring of relationships was done using ANOVA (Analysis of Variance) statistical method. ANOVA (Analysis of Variance) is one of the statistical tests that tries to decode the correlation among the various features of data. The main reason why we chose this method is that studies the statistical differences between both numerical and categorical sets of features of the data, and our target variable happens to be categorical, while other features are numerical.

To apply the K best features method, we needed to choose a proper value for K. We first obtained scores for all the features ordered from highest to lowest and then evaluated where the score drop is huge (after which point the features have very low importance scores) and decided to stick with the features before the drop. We also plotted a chart to be able to better detect the drop visually. The process is similar to finding the best k for k-means using the elbow method. With this method we decided to continue with k equal to 9.

A picture containing text, screenshot, plot, line

Description automatically generated

Figure : Feature importances

Todo: evaluacia, modely

Todo: bibliography