Report on the Prediction of Bank Churners

Data Science: Capstone - Choose Your Own Project

by Edin Ceman

18th February 2021

1. Introduction

The following report will give you an overview of my *Data Science: Capstone Choose Your Own* project. In the **Introduction** section, background information on the *Bank Churners* use case is provided. This is followed by a description of the used data set, the goal of the project and the key steps that were performed. In the **Methods & Analysis** section, the modeling approach, process and methods will be explained. The **Results** section will then present and discuss the final results of the prediction model. Finally, the **Conclusion** section will summarize the outcomes and discuss it's limitations and potential future work. Please note that for the report at hand and the depicted R code, all comments have been removed for the purpose of clarity and formatting. The commented code and further details can be found in the corresponding R script.

1.1 The Bank Churners Use Case

A manager at a bank is concerned with more and more customers leaving their credit card services (i.e. *churning*). The bank manager would like to predict which customer is going to churn next based on the underlying data that the bank has collected. The bank manager's goal is to identify likely-to-churn customers such that the bank can proactively contact the customer to provide a customized service or a special offer to prevent him from churning. The bank manager asks for your support in the creation of an appropriate model that can accomplish this task. The data set is originally from a website with the URL https://leaps.analyttica.com/home. The following context and background information is taken from the Kaggle description[1]. I have added further information and summarized it such that it conveys the key information.

1.2 Dataset

The *Bank Churners* data set contains **10,127 bank customers** with information about e.g. their *age*, *salary*, *marital_status*, *credit card limit*, *credit card category* and many more. In total there are **23 variables** in the data set. The following list gives you an overview of the variables, their data types[2] and a brief description.

- **CLIENTNUM**, (int), unique customer ID.
- **Attrition_Flag**, (Factor w/ 2 levels "Attrited Customer", "Existing Customer"), flag if customer churned.
- **Customer_Age**, (int), age of the customer.
- **Gender**, (Factor w/ 2 levels "F","M"), sex of customer.
- **Dependent_count**, (int), number of dependents.
- **Education_Level**, (Factor w/ 7 levels "College","Doctorate"), education level of customer.

[1] Source: https://www.kaggle.com/sakshigoyal7/credit-card-customers.

[2] Please note that int = integer, num = numerical and factor = discrete type or factor.

- Marital_Status, (Factor w/ 4 levels “Divorced”,“Married”), marital status.
- Income_Category, (Factor w/ 6 levels ”), discrete income level e.g. $60K - $80K.
- Card_Category, (Factor w/ 4 levels “Blue”,“Gold”), tier of credit card product.
- Months_on_book, (int), period / length of relationship with bank.
- Total_Relationship_Count, (int), total no. of products held by the customer.
- Months_Inactive_12_mon, (int), no. of months inactive in the last 12 months.
- Contacts_Count_12_mon, (int), no. of contacts in the last 12 months.
- Credit_Limit, (num), credit limit on the credit card.
- Total_Revolving_Bal, (int), total revolving balance on the credit card.
- Avg_Open_To_Buy, (num), open to buy credit line (average of last 12 months).
- Total_Amt_Chng_Q4_Q1, (num), change in transaction amount (Q4 over Q1).
- Total_Trans_Amt, (int), total transaction amount (last 12 months).
- Total_Trans_Ct, (int), total transaction count (last 12 months).
- Total_Ct_Chng_Q4_Q1, (num), change in transaction count (Q4 over Q1).
- Avg_Utilization_Ratio, (num), average card utilization ratio.

1.3 Goal

The goal is to build a prediction model based on the provided *Bank Churners* data set. Following that, the aim of the prediction model is to predict potentially churning bank customers, i.e. precisely the variable *Attrition_Flag*. For the use case at hand, this is a valuable information to the bank since it can use this insight to proactively initiate countermeasures in order to prevent the customer from churning. To achieve this goal, we will build a model that learns on the basis of the existing *Bank Churners* data, and thus finds patterns to identify the parameters that indicate a potential churn.

1.4 Key Steps

To build and evaluate the prediction model, the following key steps are performed:

- Data Initiation: First, the *Bank Churners* data set is downloaded, and the relevant information is extracted, transformed and loaded into R.
- Data Exploration: Subsequently, the data set and it’s data structure are explored with statistics and visual representations of important metrics. Furthermore, the data set is checked for completeness and special characteristics.
- Data Preparation: The data set is then cleansed, and split into a *train set* (80%) for the purpose of development and training, and a hold-out *test set* (20%) for the evaluation of the prediction model.
- Model Design: In this step, the prediction model is built using the *train set*, and evaluated with the *test set*. Multiple machine learning techniques as well as an ensemble model are considered in the model design to find the best approach for the prediction of bank churners.
- Model Evaluation: Next, predictions are made on the *test set* by the each of the different techniques and approaches. Those are evaluated using a confusion matrix. From the confusion matrix selected parameters such as the *Accuracy* and the *F-measure* or *F-score* are considered for the evaluation of the models’ performances.
- Results: The final model results as well as further insights are presented and discussed.
- Conclusion: The report results are summarized, and limitations as well as potential future work is discussed.

2. Methods & Analysis

This section explains the approach for initiating and preparing the data set as well as the designing, training and evaluation of the prediction model. The *Data Initiation*, *Data Exploration* and *Data Preparation* sections describe how the *Bank Churners* data is loaded, analyzed and prepared. Based on that, the section *Model Design* describes the approach, rationale and the methods used to build the prediction model. The *Model Evaluation* section describes the evaluation and optimization process and the metrics used to determine the model performance.

2.1 Data Initiation

The code shown below downloads the *Bank Churners* data set from my personal *GitHub Repository*, and extracts and transforms the relevant information into an R data frame called *data*. It should be noted that the data set has a header, and that all *string* data types are automatically transformed in a *factor* data type. This step is necessary to enable us to use classification models later on. Furthermore, from the *Bank Churners* website on Kaggle that is mentioned above, we receive the information from the author that the variables with the name *Naive_Bayes_Classifier...mon_1* and *Naive_Bayes_Classifier...mon_2* should be removed from the data set. This cleansing step is also done as part of the following code. The cleansed data set is then assigned the name *data_clean*.

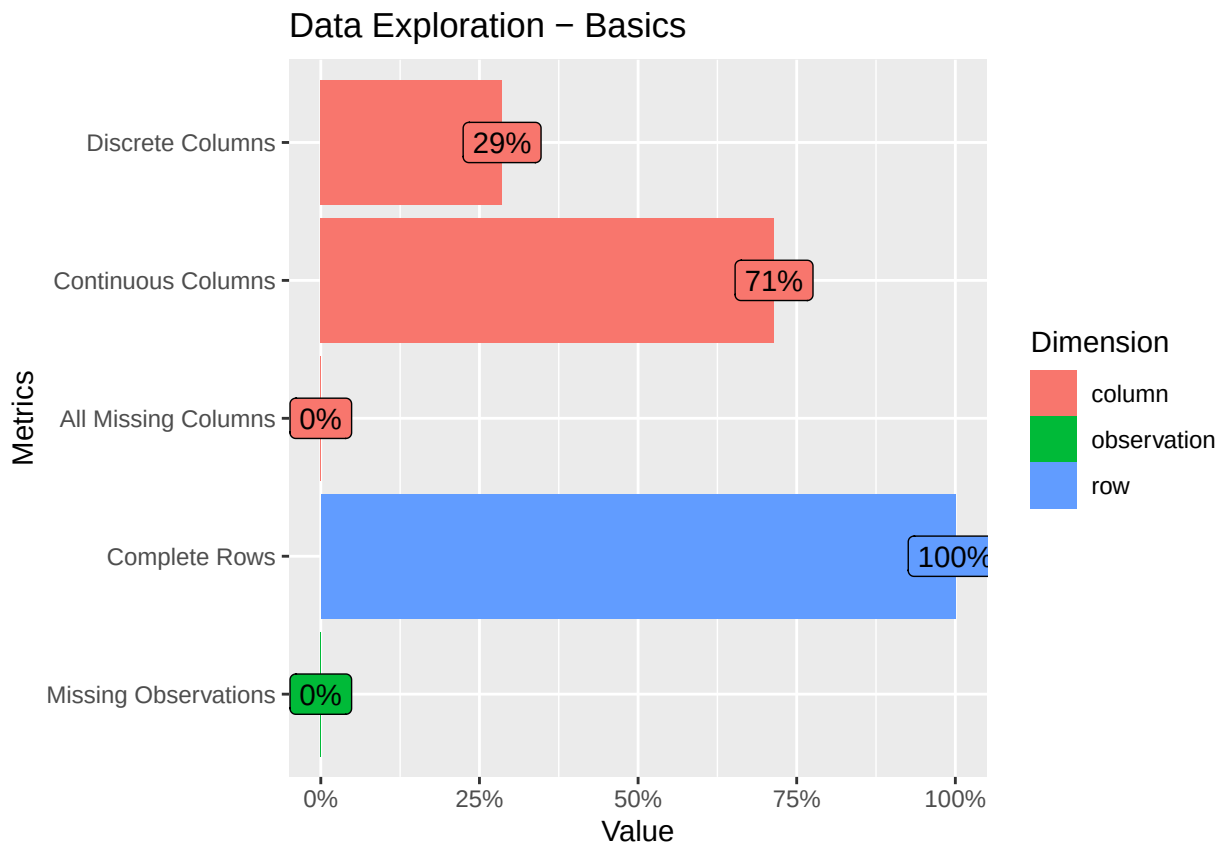
```
data <- read.csv(curl("https://raw.githubusercontent.com/edce1987/edx_edcem_,
CYO/main/BankChurners.csv"), header = TRUE, stringsAsFactors =  TRUE)

data_clean <- data %>% select(-Naive_Bayes_Classifier_Attrition_Flag_Card_Cate,
gory_Contacts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_,
mon_1 & -Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_,
mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2)
```

2.2 Data Exploration

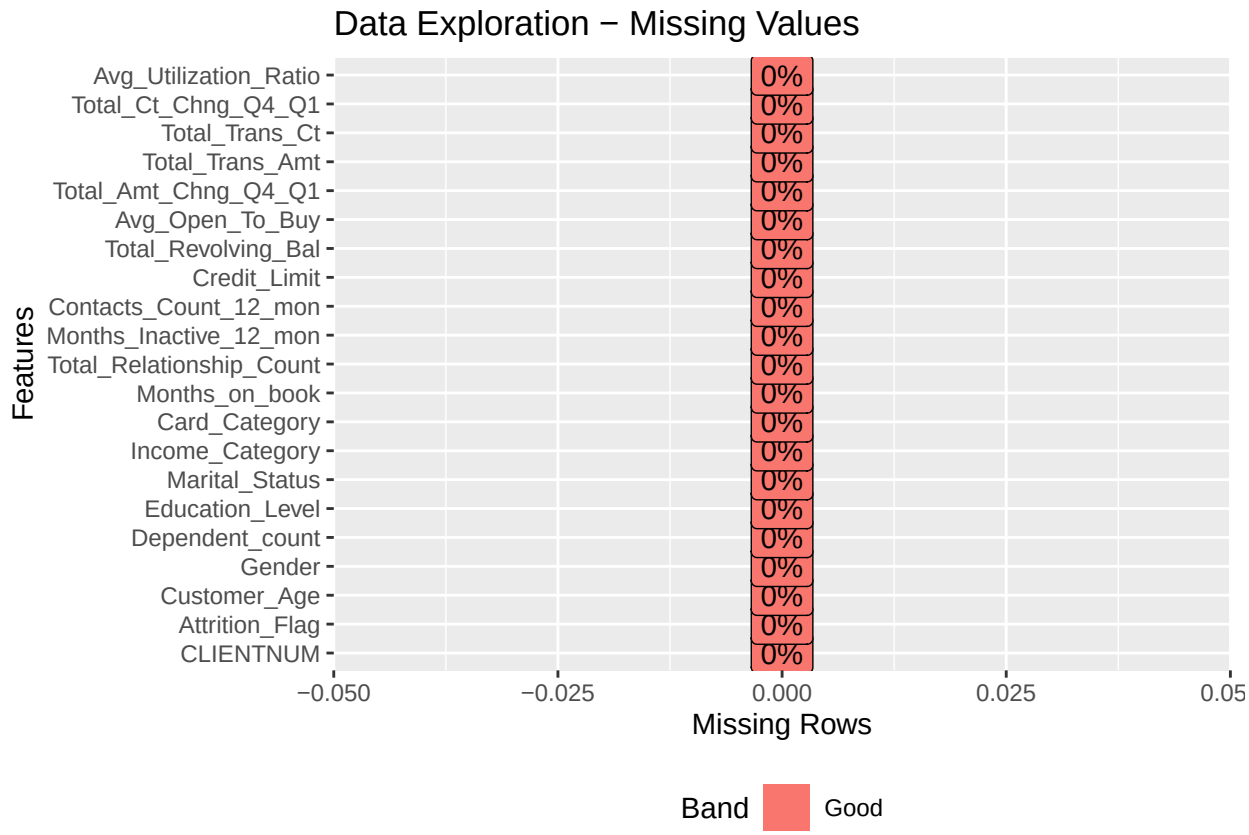
In this section, the *Bank Churners* data set is explored in more detail. Personally, I prefer to use the package DataExplorer for a detailed analysis. It is a very powerful package that quickly generates a detailed report for a given data set. The report includes some important information such as histograms for the variables, QQ plots, information on missing values, variable correlations and a principal component analysis. For the report at hand, we will perform some of the exploration steps manually.

```
plot_intro(data_clean, title = "Data Exploration - Basics")
```



From this illustration, we see that the data set has **29% discrete columns** (i.e. factor types), and **71% continuous columns** (i.e. numerical or integer). We can also see that there are no missing values, and that the rows are complete. To have a more detailed overview on missing values, we can perform a deeper analysis using the following code:

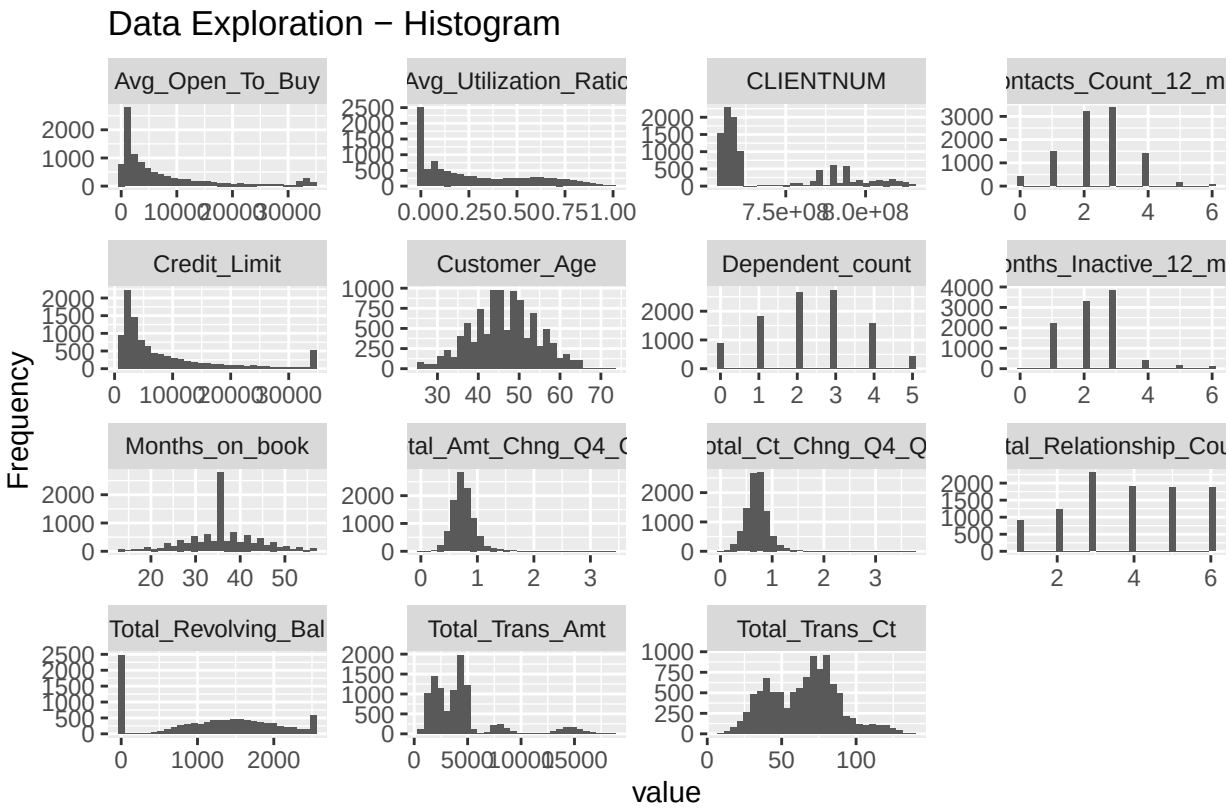
```
plot_missing(data_clean, title = "Data Exploration - Missing Values")
```



In this plot we can see a more detailed overview of missing values for each variable. However, since there are no missing values for the variables, we don't need to perform further steps to e.g. fill or replace them.

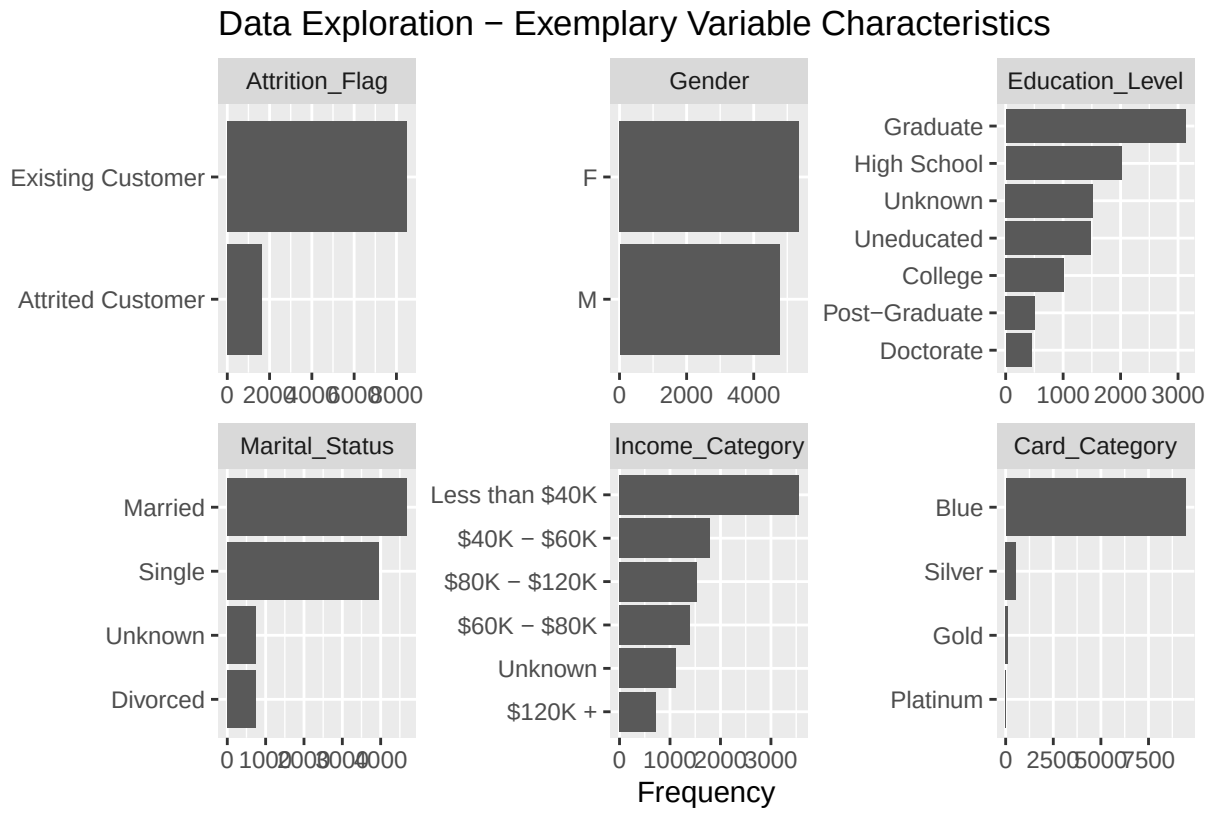
Another important information is the distribution of the values for each of the variables. By using a **histogram**, we can visually inspect the distributions to see if the variables' values are e.g. normally distributed and what kind of special characteristics they show.

```
plot_histogram(data_clean, title = "Data Exploration - Histogram")
```



From this plot, we can see that most of the variables in our data set are **not** normally distributed. However, an example of an approximately normal distributed variable is the *Customer_Age*. With this knowledge, we should avoid using models or techniques that strictly require the data to be normally distributed. We can gain further insights by looking deeper into a selected set of variables.

```
plot_bar(data_clean, title = "Data Exploration - Exemplary Variable Characteristics")
```



Here we can gain more insights about the characteristics of a selected set of variables with a factor data type. For example, we see that for our target variable *Attrition_Flag* the majority has the characteristic *Existing Customer*, whereas we are more interested in the outcome *Attrited Customer*. This "prevalence" will be discussed later on and it's impact on our results.

```
mean(data_clean$Attrition_Flag == "Attrited Customer")
```

```
## [1] 0.1606596
```

From the variable *Gender*, we can see that most customers are female. From the variable *Income_Category*, we can see that most customers have an income *Less than $40k*. Most customers are married according to the variable *Marital Status*. Most customers have a *Blue* card category.

Of course, we could go into even more detail at this point and e.g. perform correlation analyses for the variables and try to perform a dimensionality reduction using a principal component analysis (PCA). However, I have scoped those steps out and decided to simplify the model design process in this regard. Hence, we will apply the selected machine learning techniques and approaches on the full-size and unrestricted *Bank Churners* data set, and evaluate how the selected models perform in this case. A deeper analysis can be done as part of future work as described in the section *4. Conclusion*.

2.3 Data Preparation

Since in the previous section we observed that there are no missing values in the data set, and that the rows and columns are complete, we do not have to perform further steps for e.g. filling of NAs. However, it makes sense to remove the variable *CLIENTNUM* which is a unique ID for each customer. Since this is a randomly assigned and unique ID, it is plausible to expect that is has no explanatory power. Since no e.g. *joins* are performed later on, we also do not need it for mapping purposes. Also, we want to avoid causing a potential interference for the models. Hence, we remove the *CLIENTNUM* from the data set and receive the set *data_prepared*.

```
data_prepared <- data_clean %>% select(-CLIENTNUM)
rm(data, data_clean)
```

From here, we can move further and prepare the data set for the subsequent model design. We split the set *data_prepared* it into a *train set* (80%) (or 8,101 observations) and a *test set* (20%) (or 2,026 observations). The *train set* is used for the purpose of model design and training, whereas the *test set* is used for the model evaluation and parameter optimization.

```
set.seed(1, sample.kind="Rounding")

test_index <- createDataPartition(data_prepared$Attrition_Flag,
                                  times = 1, p = 0.2, list = FALSE)
train_set <- data_prepared[-test_index,]
test_set <- data_prepared[test_index,]
```

2.4 Model Design

We want to design a model that predicts a churning customer using our prepared *train set*. The target variable we want to predict is *Attrition Flag*. This is a discrete type variable and can have two different outcomes or levels. One outcome is *Attrited Customer*, the other is *Existing Customer*. From our previous data exploration, we know that the outcome *Attrited Customer* has a prevalence of ~16%, i.e. those customers churned according to the data set. Hence, we need a model type that is suitable for **classification** since we have a discrete target variable here that can only have one of the two mentioned outcomes or levels. To find suitable classification models, we can inspect the list of available models from the *caret* package. The following excerpt shows a set of exemplary models.

```
mod <- modelLookup()
mod <- mod %>% filter(forClass == TRUE)
head(mod)
```

```
##      model parameter          label forReg forClass probModel
## 1      ada      iter         #Trees  FALSE     TRUE      TRUE
## 2      ada  maxdepth Max Tree Depth  FALSE     TRUE      TRUE
## 3      ada        nu  Learning Rate  FALSE     TRUE      TRUE
## 4   AdaBag    mfinal         #Trees  FALSE     TRUE      TRUE
## 5   AdaBag  maxdepth Max Tree Depth  FALSE     TRUE      TRUE
## 6 adaboost     nIter         #Trees  FALSE     TRUE      TRUE
```

Since we do not know yet which model will perform best on our *Bank Churners* data set, we will select some popular models and evaluate them in a later step.

We will evaluate the following set of selected models[3]:

- **adaboost**: Adaptive Boosting
- **bayesglm**: Bayes Generalized Linear Model
- **knn**: K-Nearest Neighbors
- **naive_bayes**: Naive Bayes
- **Rborist**: Extensible and parallelizable implementation of Random Forest
- **rf**: Random Forest
- **rpart**: Recursive Partitioning and Regression Trees
- **svmLinear**: Support Vector Machine Linear
- **svmPoly**: Support Vector Machine Polynomial
- **svmRadial**: Support Vector Machine Radial

```
models <- c("adaboost", "bayesglm", "knn", "naive_bayes", "Rborist", "rf",
            "rpart", "svmLinear", "svmPoly", "svmRadial")
```

Each of the selected models will be trained (or fitted) on our prepared *train set*. Each model will be trained using all available variables (or features). To avoid **overfitting**, we will make use of *k-fold cross-validation*. Overfitting arises when a model is too specific, i.e. such that it performs well on a well known data set, but rather poorly on an unknown data set. The concept of *k-fold cross-validation* is a procedure that resamples a given data set **k** times to reduce a potential **selection bias** (or to increase generalization) when deciding which part of your data you use for training and which one for test purposes[4].

For our model design and training, we will use a 10-fold cross-validation.

[3] Explaining each model in more detail would go beyond the scope of this report. Hence, the selected models will only be listed here.
[4] Further information: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f.

```
set.seed(1, sample.kind="Rounding")
control <- trainControl(method = "cv", number = 10, p = .8)
```

Following that, the models are fitted based on the *train set* and predictions for the target variable *Attrition_Flag* are made with each of the models for the *test set*.

```
set.seed(1, sample.kind = "Rounding")
predictions <- sapply(models, function(model) {
  print(model)
  fit <- train(Attrition_Flag ~ ., method = model, trControl = control, data = train_set)
  prediction <- predict(fit, test_set)
  data.frame(model = prediction)
})
predictions <- as.data.frame(predictions)
```

In the next section, we will evaluate the outcome of the training and look at results that were achieved by the selected models.

2.5 Model Evaluation

In this section the selected models will be evaluated by computing a **confusion matrix** for each model. The confusion matrix is a table that summarizes the outcome of a classification model. In our case it shows the number of correctly predicted "Attrited Customer" (true positives), correctly predicted "Existing Customer" (true negatives), as well as incorrectly predicted "Attrited Customer" (false positives) and incorrectly predicted "Existing Customer" (false negatives). From this description you may notice why the table is named "confusion matrix". From the numbers mentioned further metrics can be derived, e.g. *Accuracy*, *Sensitivity*, *Specificity* and *Prevalence* among others. These metrics, among others, can be used for the evaluation of classification models.

We will use multiple metrics to evaluate our models to get a clearer and more diversified picture of the models' performances. We will use the **Accuracy** and the **F-measure** to determine the performance of our models. The *Accuracy* is the number of correct predictions in relation to the total predictions made. It is one of the most common performance measures.

Furthermore, we will use the **F1 score** or **F-measure**. It is a number between 0 and 1 and it is the harmonic mean of **Precision** and **Recall**, i.e. it balances *Precision* and *Recall*. *Precision* (or Pos. Pred. Value) is the share of correct positive predictions (true Positives) in relation to the total positive predictions (true positives + false positives). *Recall* (or Specificity) is the share of correct positive predictions (true positives) in relation to the total positives (true positives + false negatives). The *F-measure* is also a very common measure to evaluate classification models[5].

To put theory into practice, we will compute the *Accuracy* for the selected models first.

```
accuracies <- sapply(predictions, function(x) {
  confusionMatrix(data=x, reference=test_set$Attrition_Flag)$overall["Accuracy"]
})

print(accuracies)
```

```
##    adaboost.model.Accuracy    bayesglm.model.Accuracy
##                  0.9693978                  0.8958539
##        knn.model.Accuracy naive_bayes.model.Accuracy
##                  0.9067127                  0.8706811
##     Rborist.model.Accuracy          rf.model.Accuracy
##                  0.9511352                  0.9664363
##       rpart.model.Accuracy   svmLinear.model.Accuracy
##                  0.9002962                  0.9017769
##     svmPoly.model.Accuracy   svmRadial.model.Accuracy
##                  0.9284304                  0.9244817
```

All the models have relatively high accuracies, the bank manager would be happy to see the results. Nonetheless, when looking at the numbers, we see that the *AdaBoost* model has the highest accuracy with **0.9693978**. We can interpret it such that the model predicts ~97% of the *Attrited Customers* (true positives) or *Existing Customers* (true negatives) correctly.

```
print(accuracies[which.max(accuracies)])
```

```
## adaboost.model.Accuracy
##               0.9693978
```

In the next step, we compute the *F-measures* for the used models to see whether we get a different result or if the *AdaBoost* model still shows the highest performance even after balancing for *Precision* and *Recall*.

[5] Further information: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226

```
f_measures <- sapply(predictions, function(x) {
  F_meas(data=x, reference=test_set$Attrition_Flag)
})

print(f_measures)
```

```
##    adaboost.model    bayesglm.model         knn.model naive_bayes.model
##         0.9012739         0.6355786         0.6758148         0.6030303
##     Rborist.model          rf.model       rpart.model   svmLinear.model
##         0.8369028         0.8927445         0.6677632         0.6514886
##     svmPoly.model   svmRadial.model
##         0.7579299         0.7320490
```

We can confirm the previous result, the *AdaBoost* model again has the highest performance with an *F-measure* of 0.9012739. The *F-measure* is not easily interpreted and straightforward as the *Accuracy*, however we can observe that after balancing *Precision* and *Recall* the model has performed less good.

```
print(f_measures[which.max(f_measures)])
```

```
## adaboost.model
##      0.9012739
```

Up to this point, it appears like the *AdaBoost* model is the best model to choose for the prediction of potentially churning customers. However, we can also check whether an ensemble of the selected models will give us better results than the *AdaBoost* model alone. An ensemble is a combination of multiple models. To build an ensemble model, we observe the predictions made by each of our selected models and determine a majority vote. Following that, the prediction of the ensemble model is the prediction that the majority of the underlying models made. To compute the majority vote, we can run the following code:

```
votes <- rowMeans(predictions == "Attrited Customer")
```

With the *votes* computed, we can determine the predictions of the ensemble model in the next step. If the vote is "> 0.5", it means that more than 50% of the underlying models predicted *Attrited Customer*. Hence, the ensemble model's prediction is *Attrited Customer*.

```
predEnsemble <- as.factor(ifelse(votes > 0.5, "Attrited Customer", "Existing Customer"))
```

Now the ensemble model can be evaluated too with the same metrics as before. We compute the *Accuracy* first, and afterwards the *F-measure* to assess the results.

```
accuracyEnsemble <- confusionMatrix(data=predEnsemble, reference=,
                                    test_set$Attrition_Flag)$overall["Accuracy"]

print(accuracyEnsemble)
```

```
##  Accuracy
## 0.9363277
```

```
fMeasureEnsemble <- F_meas(data=predEnsemble, reference=test_set$Attrition_Flag)

print(fMeasureEnsemble)
```

```
## [1] 0.7716814
```

The ensemble model performs relatively good, however with an *Accuracy* of **0.9363277** and an *F-measure* of **0.7716814** it is unfortunately not better than the *AdaBoost* model alone. Hence, we will choose the *AdaBoost* model as our final model.

We can now take a closer look at the *AdaBoost* model to see if we can optimize it further and what else we can figure out. To optimize the model, we will use a larger **tuning grid**. But we should avoid optimizing it too much since this would most probably result in overfitting the model, and depending on the hardware, would take a very long time. The original optimization (see corresponding R script) was performed with a tuning grid from 0 to 1000 in increments of 50 which took extremely long (~15-20 hours). One could also apply an even "finer" parameter optimization, e.g. from 1 to 1000 in increments of 1. But this would take even much longer and would most probably result in overfitting. Hence, we shall refrain from that. From my personal trials during the model development, I found that the optimal parameters for the *AdaBoost* model at hand are **nIter = 450** with **method = "Adaboost.M1"**. Knowing the optimal parameters, we can massively accelerate the fitting process[^7].

```
fitAdaboost <- train(Attrition_Flag ~ ., method = "adaboost", trControl = control,
                     tuneGrid = data.frame(nIter = 450, method = "Adaboost.M1"),
                     data = train_set)
```

Once the optimization is completed, we can inspect the final model parameters. From this, we can see that the *AdaBoost* model uses **450 trees** with the method **AdaBoost.M1** to decide whether a customer is likely to churn or not.

```
fitAdaboost
```

```
## AdaBoost Classification Trees
##
## 8101 samples
##   19 predictor
##    2 classes: 'Attrited Customer', 'Existing Customer'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7291, 7291, 7291, 7290, 7291, 7291, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9696331  0.8832098
##
## Tuning parameter 'nIter' was held constant at a value of 450
## Tuning
##  parameter 'method' was held constant at a value of Adaboost.M1
```

Following the same procedure as before, we can quickly check how the optimized *AdaBoost* model performs in terms of *Accuracy* and *F-measure*.

```
predAdaboost <- predict(fitAdaboost, test_set)

print(confusionMatrix(data=predAdaboost,
                      reference=test_set$Attrition_Flag)$overall["Accuracy"])
```

```
##  Accuracy
## 0.9698914
```

```
print(F_meas(data=predAdaboost, reference=test_set$Attrition_Flag))
```

```
## [1] 0.9020867
```

Indeed, the optimized *AdaBoost* model has improved slightly. The optimized *AdaBoost* model has an *Accuracy* of **0.9698914** vs. 0.9693978 (w/o optimization). The *F-measure* after optimization is **0.9020867** vs. 0.9012739 (w/o optimization). It has also improved slightly.

3. Results

From the previous section, we have seen that the *AdaBoost* model performed best on the *Bank Churners* data set. In this section we will explore and discuss further insights by looking into more detail of the model's outcome.

An important insight can be gained by looking at the **variable importance**, and with that, to see which variables (or features) in the data set have the highest importance or explanatory power for the correct prediction of the *Attrition Flag*. This information can be very valuable and useful, especially for the use case at hand. With this knowledge, the bank manager can consider those variables as **early warning indicators**. Those indicators can then be monitored to e.g. identify potentially churning customers at an early stage and to initiate early countermeasures to prevent customers from churning.

```
varImp(fitAdaboost)
```

```
## ROC curve variable importance
##
##                           Importance
## Total_Trans_Ct             100.00000
## Total_Ct_Chng_Q4_Q1         84.02879
## Total_Revolving_Bal         60.82530
## Avg_Utilization_Ratio       60.82437
## Total_Trans_Amt             59.67286
## Contacts_Count_12_mon       45.47717
## Months_Inactive_12_mon      40.91322
## Total_Relationship_Count    38.51607
## Total_Amt_Chng_Q4_Q1        25.65157
## Credit_Limit                13.23408
## Gender                       7.18629
## Avg_Open_To_Buy              5.70161
## Marital_Status               4.63772
## Income_Category              4.61939
## Dependent_count              4.26002
## Customer_Age                 3.60360
## Months_on_book               2.25346
## Education_Level              0.04365
## Card_Category                0.00000
```

From this, we see the most important variables in the data set to predict potentially churning customers. For example the number of transactions in the last 12 months **(Total_Trans_Ct)**, the change in the number of transactions from Q4 to Q1 **(Total_Ct_Chng_Q4_Q1)**, or the total revolving balance on the credit card **(Total_Revolving_Bal)**. These variables, among others, should be of special importance and serve as early warning indicators to the bank manager. On the other side, we can observe that some variables have a relatively low importance, e.g. the education level **(Educaction_Level)** or the card category **(Card_Category)**.

4. Conclusion

The report at hand described the key steps and results of my *Choose Your Own Capstone Project*. Initially, the *Bank Churners* use case and the used data set have been explained and background information was provided. Following that, the goal and the key steps of the the project have been described. Afterwards, the modeling design process as well as the evaluation of the *Bank Churners* prediction model have been explained and a reasoning for the chosen design has been given. Finally, the results as well as further insights have been presented and discussed.

From our evaluation results, we have seen that the *Adaboost* model has the highest *Accuracy* for the *Bank Churners* use case. Even after using the *F-measure*, which is a more balanced metric, we saw that the *AdaBoost* model was still superior to the other selected models for the data set at hand. For the Bank Churners use case, our insights can be very useful for the bank manager. He could use the model to predict which customer is likely to churn, and hence proactively contact the customer to offer an extended service or special credit card conditions.

A limitation is that, since the underlying data set only has a relatively small prevalence of ~16% of *Attrited Customers*, the final model prediction might not always be as reliable and robust in the prediction of churning customers as our very promising evaluation metrics suggest. Hence, one way to improve or to stabilize the model would be to gather more data at the bank itself, or to buy additional data from external data providers to enrich the existing data set. Another limitation is set by the limited computing power of a regular notebook or desktop PC. In my case, e.g. it did not allow for the application of more advanced and computationally intensive tuning grids or optimization techniques in a reasonable time frame. A potential future work could make use of cloud services to get the necessary computing power, e.g. *Microsoft Azure*, *Google Cloud Platform* or *Amazon AWS*.