Churning Bank Customers Prediction Report

Data Science: Capstone - Choose Your Own Project

Wei Guan

9th March 2023

1. Introduction

This report provides an overview of my *Data Science: Capstone Choose Your Own Project*. The aim of this project is to create predictions of churning bank customers using a data set from Kaggle. This report analyzes different models/algorithms for the prediction of churning bank customers and it is divided into the following parts:

- Section Introduction describes the used data set from Kaggle about churning bank customers, the goal of the project and the key steps that were performed.
- Section Methods & Analysis explains the process and techniques used, including data cleaning, data exploration and visualization, followed by the gained insights and analyses of different algorithms.
- Section Results presents and discusses the final results of the prediction models.
- Section Conclusion gives a brief summary of the outcomes and discusses its limitations and potential future work.
- Section References provides a list of references that lists sources for the data set and/or other resources used.

Note: All the source code including plots and comments can be found in the corresponding R script in details.

1.1 Backgroud of Churning Bank Customers

Bank managers are concerned with more and more customers leaving their credit card service, these are so-called *Churning Bank Customers*. Therefore, bank managers would really appreciate if there is a way for them to predict which customer is going to churn next based on the data that the bank has collected, so that they can proactively go to the customer to provide them better services or other offers and turn the decisions of the customers in the opposite direction and keep them stay in their banks. It is quite useful for bank managers to have a prediction of churning bank customers with the help of an appropriate model/algorithm. The data set came originally from a website with the URL as https://leaps.analyttica.com/home, which explains how to solve a particular business problem. I got the data set from the website Kaggle and used the background information *[1]*. Furthermore, I have added more information and summarized it for my following work on the project.

1.2 Dataset

The data set of *Churning Bank Customers* contains 10,127 bank customers with various information such as *age*, *salary*, *marital status*, *credit card limit*, *credit card category*, etc. There are 23 columns in the data set totally. The following features/columns can be found in this data set:

- CLIENTNUM: a unique identifier of each customer with the data type *integer*.
- Attrition_Flag: flag if a customer is churned with the data type *Factor* ("Attrited Customer", "Existing Customer").
- Customer_Age: age of a customer with the data type *integer*.
- Gender: sex of a customer with the data type *Factor* (levels "F","M").
- Dependent_count: number of dependents with the data type *integer*.
- Education_Level: education level of a customer with the data type *Factor* ("Graduate","Doctorate", "High School", etc.).
- Marital_Status: marital status with the data type *Factor* ("Single","Married", etc.).
- Income_Category: income interval of a customer with the data type *Factor* ("Less than $40K", "$60K - $80K", etc.)
- Card_Category: category of credit card with the data type *Factor* ("Blue","Gold", etc.).
- Months_on_book: duration in the bank of a customer with the data type *integer*.
- Total_Relationship_Count: total number of products that a customer have with the data type *integer*.
- Months_Inactive_12_mon: number of inactive months in the last 12 months with the data type *integer*.
- Contacts_Count_12_mon: number of contacts in the last 12 months with the data type *integer*.
- Credit_Limit: credit limit on the credit card with the data type *numeric*.
- Total_Revolving_Bal: total revolving balance with the data type *integer*.
- Avg_Open_To_Buy: open to buy credit line (average of last 12 months) with the data type *numeric*.
- Total_Amt_Chng_Q4_Q1: total change in transaction amount from Q4 to Q1 with the data type *numeric*.
- Total_Trans_Amt: total transaction amount (last 12 months) with the data type *integer*.
- Total_Trans_Ct: total transaction count (last 12 months) with the data type *integer*.
- Total_Ct_Chng_Q4_Q1: total change in transaction count from Q4 to Q1 with the data type *numeric*.
- Avg_Utilization_Ratio: average utilization ratio with the data type *numeric*.
- Naive_Bayes_Classifier…12_mon_1: data type *numeric* (full name can be found in the data set).
- Naive_Bayes_Classifier…12_mon_2: data type *numeric* (full name can be found in the data set).

1.3 Goal

The goal of this project is to build a prediction model by analyzing different algorithms/models based on the provided data set of *Churning Bank Customers*. With this prediction model, churning bank customers could be predicted and they could be flagged with the variable *Attrition_Flag*, which is very valuable for bank managers, because they can use this insight to proactively provide better service or other measures in order to prevent customers from churning. In order to achieve this goal, the data set *Churning Bank Customers* should be analysed. Furthermore, important features will be identified, so that a prediction model can be built in order to reveal potential churning bank customers.

1.4 Key Steps

In this project the following key steps are performed in order to build and evaluate the prediction model:

- Data Initiation: The *Churning Bank Customers* data set is downloaded and the relevant information is extracted, transformed and loaded into R.
- Data Cleaning, Exploration and Visualization: The data set is explored and several data visualizations are made in order to get the insights from the data. Furthermore, the data set is cleansed and split into a *train set* (80%) for the purpose of development and training of models and a *test set* (20%) for the evaluation of the prediction model.
- Model Design: According to the data exploration, the prediction model is built by using the train set and then it is evaluated with the test set. Several machine learning algorithms and an ensemble model are used in the model design to find the best approach for the prediction of churning bank customers.
- Model Evaluation: The predictions are performed on the *test set* by the each of the different approaches. They are evaluated with the help of the confusion matrix, in which selected parameters such as the *Accuracy* and the *F-measure* are computed for the evaluation of the performance of the models.

- **Results:** The final model results as well as further insights are presented and discussed.
- **Conclusion:** Last but not least, there is a brief summary of the report, limitations as well as potential future work.

2. Methods & Analysis

In this section the approach will be explained for initiating and analyzing the data set and also the process of designing, building and evaluation of the prediction models. The first step is to load the necessary libraries and to load the data from the data set *Churning Bank Customers*. The data will be explored and several data visualizations will be performed in order to get the insight more easily. Then, the data will be cleansed and split into two data sets: **train Set** and **Test Set**. In the Model Design section the approach used to build the prediction model will be described. At the end the evaluation and optimization process will be presented and the performance of the prediction model will be discussed.

2.1 Data Initiation

The data set *Churning Bank Customers* has been stored in my personal GitHub Repository. From there the data is downloaded, extracted and transformed for the relevant information into a data frame called **customer**. Please note that the data set has a header and all of the string data types are automatically transformed in a *factor* data type, which is necessary for us to use classification models in the later modeling. Furthermore, on the website of Kaggle for the data set *Churning Bank Customers* mentioned above, it is mentioned that the variables *Naive_Bayes_Classifier…12_mon_1* and *Naive_Bayes_Classifier…12_mon_2* should be removed from the data set. The following code shows that necessary packages, the download of the data set and data cleansing. The cleansed data set is called **customer_clean**.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(curl)) install.packages("curl", repos = "http://cran.us.r-project.org")
if(!require(Rborist)) install.packages("Rborist", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(DataExplorer)) install.packages("DataExplorer", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(ggplot2)
library(caret)
library(curl)
library(Rborist)
library(rpart)
library(DataExplorer)
library(data.table)
```

```
Customer <- read.csv(curl("https://raw.githubusercontent.com/WeiGuan-DE/edX_WGU_ChooseYour
OwnProject/main/ChurningBankCustomers.csv"), header = TRUE, stringsAsFactors =  TRUE)

Customer_clean <- Customer %>% select(-Naive_Bayes_Classifier_Attrition_Flag_Card_Category
      _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)
```

Now, the feature *CLIENTNUM* is also going to be removed from the data set, because it is just a unique identifier for each customer and it is a randomly assigned to each customer. Since there is no join with other tables and in order to avoid potential "noise" for the models. Therefore, *CLIENTNUM* is removed by the following code:

```
Customer_clean <- Customer_clean %>% select(-CLIENTNUM)
rm(Customer)
```

2.2 Data Cleaning, Exploration and Visualization

Now let's do some research and have a close view of the data.

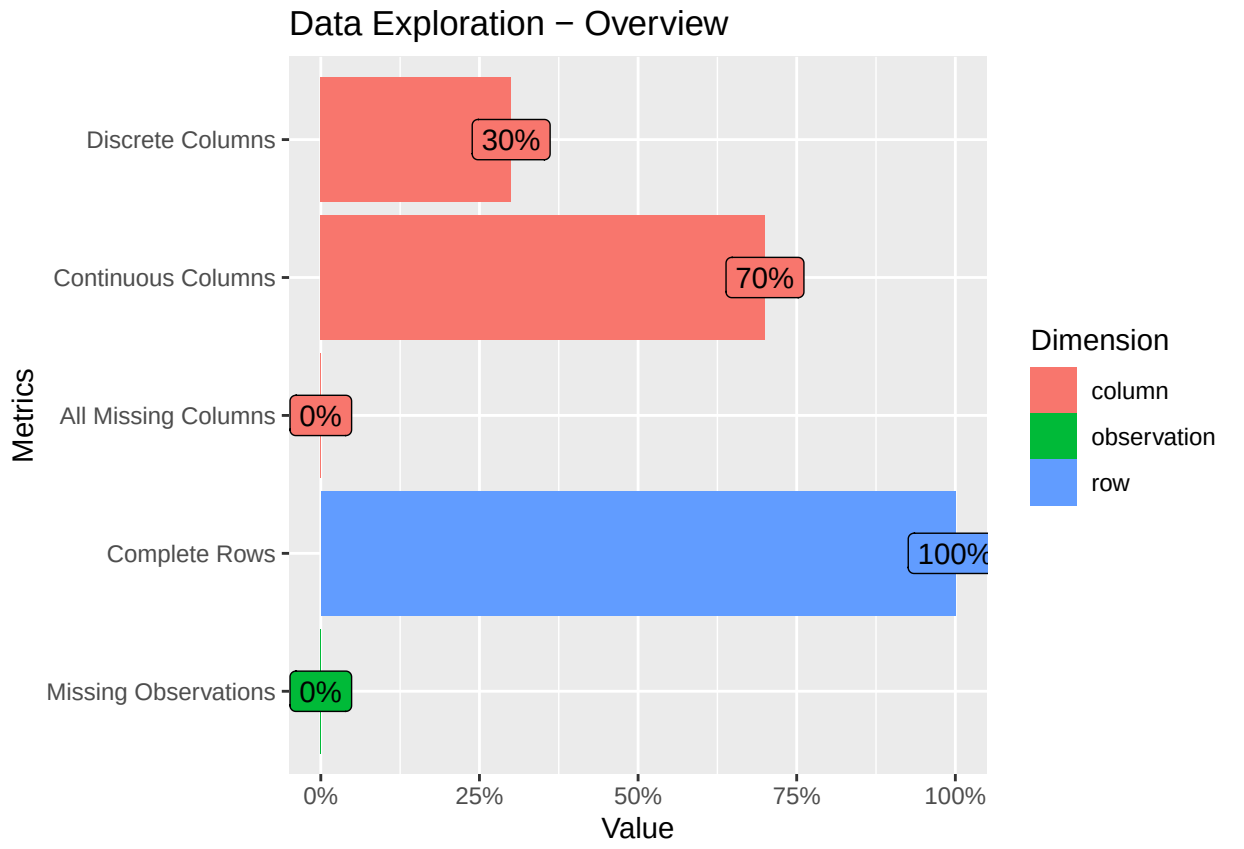
There are 10,127 rows and 20 columns in the data set *Customer_clean*.

```
dim(Customer_clean)
```

```
## [1] 10127    20
```

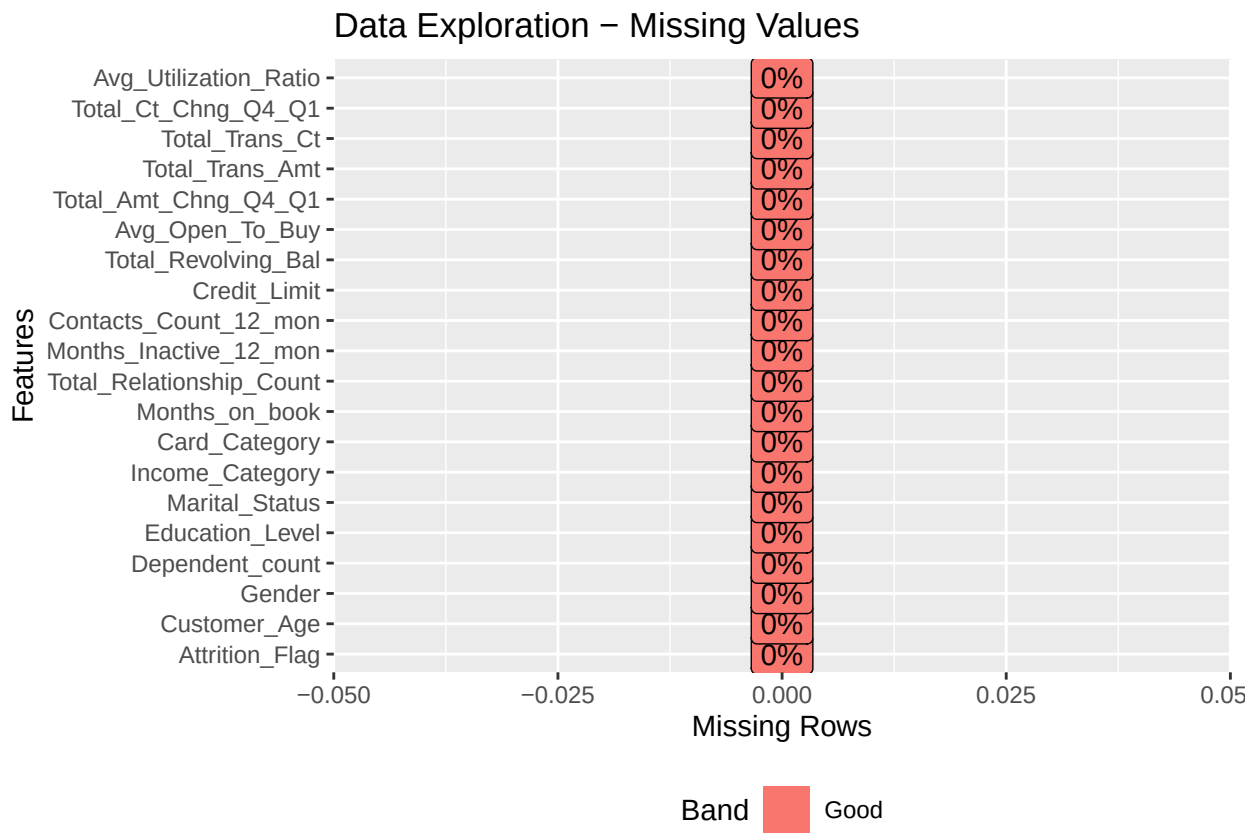
In the following part, the package **DataExplorer** is mainly used to create the data visualization, so that the insights of the data set *Churning Bank Customers* can be easily explored.

```
plot_intro(Customer_clean, title = "Data Exploration - Overview")
```



The above figure shows that in the data set there are **30% discrete columns** and **70% continuous columns**. It also shows that there are no missing columns or missing values and all of the rows are complete. In order to see the details of the missing values, the following code is used to have a deeper analysis:

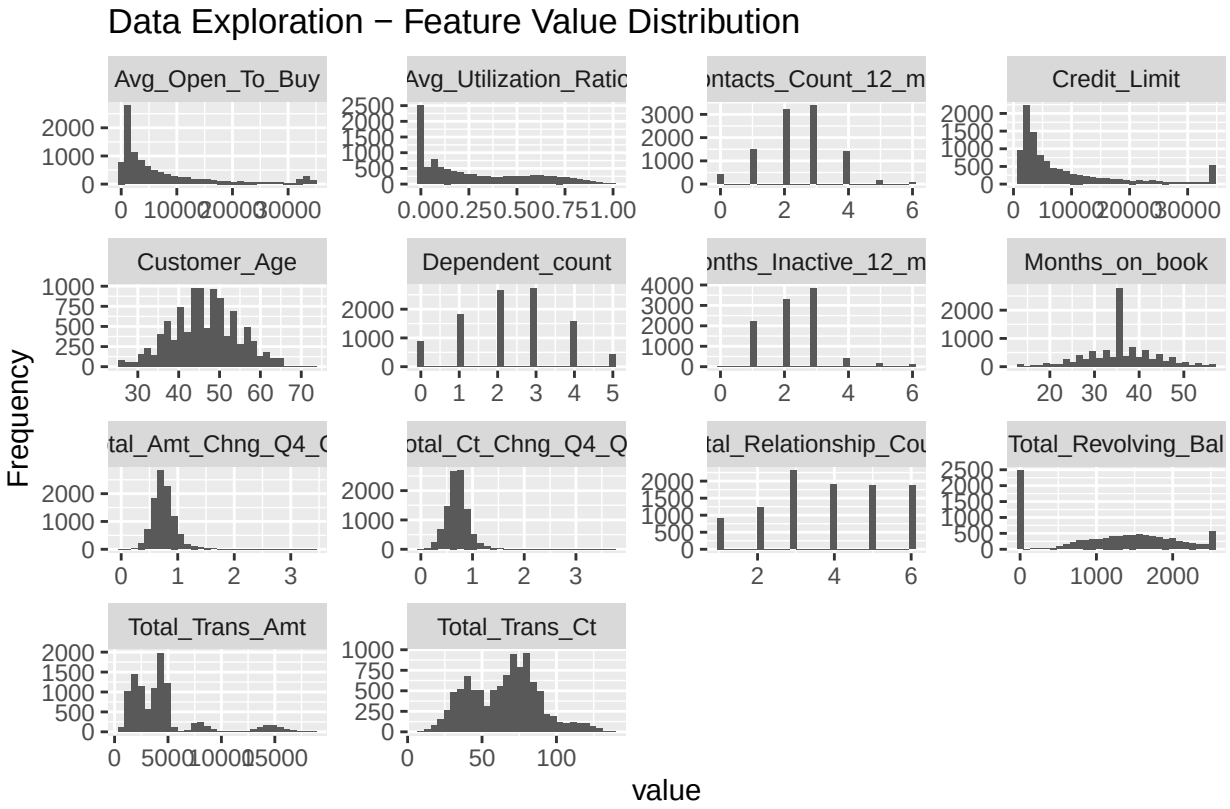
```
plot_missing(Customer_clean, title = "Data Exploration - Missing Values")
```



In the above figure a more detailed overview of missing values for each feature is shown. All the features have no missing values. Therefore, it is not necessary to implement more steps to deal with missing values in the data set.

Next, let's have a look at the distribution of the values for each feature. For this purpose, the histogram is used to visually explore the distribution of each feature.

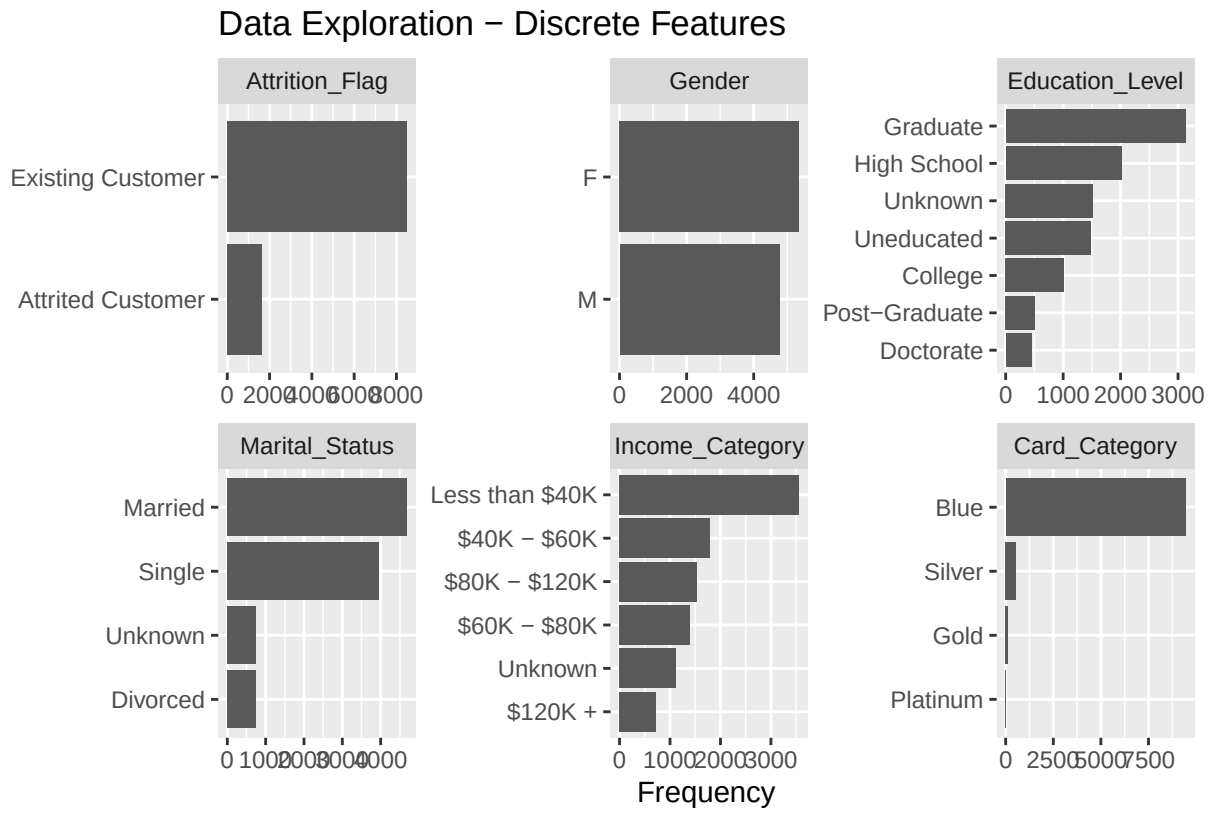
```
plot_histogram(Customer_clean, title = "Data Exploration - Feature Value Distribution")
```



In the figure above, according to the histograms most of the features in the data set *Churning Bank Customers* are not normally distributed. However, the feature *Customer_Age* is an example of an approximately normal distributed feature. Since most of the features are not normally distributed, it is not suitable to use models that strictly require a normal distribution of the data.

Now, let's also have a look at the visualization of frequency distributions for all discrete features to gain more insights.

```
plot_bar(Customer_clean, title = "Data Exploration - Discrete Features")
```



In the above figure there are more insights about the discrete features. For example, the feature *Attrition_Flag* that is to be predicted later has the majority of the value *Existing Customer* and the value *Attrited Customer* is in the minority (16% shown in the following code), which is actually the focus group.

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

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

Furthermore, from this figure the following information could be extracted:

- There are more female customers than male customers.
- More customers are married than single or divorced ones.
- Most of the customers have an income Less than $40k.
- Most of the customers have a Blue Card Category.

For the simplicity Of the model design process more data analysis is out of scope in this report. It is still interesting to see how the selected machine learning approaches/models perform on all of the existing features in the cleansed data set *Churning Bank Customers* followed by the evaluation.

Before moving to the *Model Design*, the data set *Customer_clean* is split into a *train set* with 8,101 observations (80%) and a *test set* with 2,026 observations(20%). The *train set* is used for model design and training and the *test set* is used for the model evaluation and parameter optimization.

```
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
```

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

2.3 Model Design

In order to design a model that predicts churning bank customers, the *train set* is used in this phase at first. As mentioned before, *Attrition Flag* is what has to be predicted in the data set. It has discrete values: *Attrited Customer* and *Existing Customer*. Therefore, classification models are good choices in this case. To find suitable classification models, let's inspect the list of available models from the *caret* package with the following code to show some models for example.

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

```
##      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
```

```
dim(all_models)
```

```
## [1] 431   6
```

There are totally 431 models for classifications in the *caret* package. It is hard to tell which model is the most suitable one for the data set *Churning Bank Customers*. Let's select some popular and famous models with the following code. These models will be then evaluated later. Please note that it is out of scope in this report to explain the models in details.

The following models are selected for the model design and will be evaluated later.

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

Each of these selected models will be trained with the prepared *train set* above and used by all the features. There might be overfitting, which occurs when a model performs well only on a specific data set, but rather poorly on an unknown data set. To avoid overfitting, k-fold cross-validation will be used in order to re-sample a given data set k times to reduce a potential selection bias when deciding which part of the data that is used for training and which one for test purposes *[2]*. Let's take a 10-fold cross-validation for the model design and training.

```
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
trainControl <- trainControl(method = "cv", number = 10, p = 0.9)
```

The selected models are fitted based on the *train set* and predictions for *Attrition_Flag* are made on the *test set* for each of the selected models.

```
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
predictions <- sapply(models, function(model) {
  print(model)
  fit <- train(Attrition_Flag ~ .,
               method = model,
               data = train_set,
               trControl = trainControl)
  prediction <- predict(fit, test_set)
  data.frame(model = prediction)
})
```

```
## [1] "adaboost"
## [1] "bayesglm"
## [1] "knn"
## [1] "naive_bayes"
## [1] "Rborist"
## [1] "rf"
## [1] "rpart"
## [1] "svmLinear"
## [1] "svmRadial"
```

```
predictions <- as.data.frame(predictions)
```

2.5 Model Evaluation

In the *Model Design* phase, several algorithms are chosen for predictions. Now let's evaluate them by computing the **confusion matrix** for each model. The confusion matrix summarizes the outcome of a classification model. For the prediction models in this report it shows the following cases:

- the number of correctly predicted Attrited Customers (**True Positives**)
- the number of correctly predicted Existing Customers (**True Negatives**)
- the number of incorrectly predicted Attrited Customers (**False Positives**)
- the number of incorrectly predicted Existing Customers (**False Negatives**)

There are further metrics, e.g. *Accuracy*, *Sensitivity*, *Specificity*, etc., which can be derived from the above numbers and be used to evaluate classification models. In this report **Accuracy** and **F-measure** are used to evaluate the performance of the models. **Accuracy** is the number of correct predictions divided by the total predictions. **F-measure** is a value between 0 and 1 and is the harmonic mean of **Precision** and **Recall**. *Precision* (also called positive predictive value) is the correct positive predictions (True Positives) divided by the total positive predictions (True Positives + False Positives). *Recall* (also Sensitivity) is the correct positive predictions (True Positives) divided by the total correct predictions (True Positives + False Negatives) *[3]*.

The following code calculates the *Accuracy* of the selected classification models.

```
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
##                  0.9348470                  0.9659427
##       rpart.model.Accuracy   svmLinear.model.Accuracy
##                  0.9002962                  0.9017769
##   svmRadial.model.Accuracy
##                  0.9244817
```

In the results above, all the models have high accuracies, especially the *AdaBoost* model with the highest accuracy 0.9693978.

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

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

Now, let's compute the *F-measures* for the selected models.

```
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
##         0.7762712         0.8906498         0.6677632         0.6514886
##   svmRadial.model
##         0.7320490
```

In the result of F-measures, the *AdaBoost* model also has the highest value with an *F-measure* of 0.9012739. Even after balancing for *Precision* and *Recall*, the *AdaBoost* model still shows the best performance. However, the *AdaBoost* model performs a little bit worse with F-measure than with Accuracy.

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

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

There is another way to build a prediction model that is called ensemble model, which is a combination of all the selected models above. In order to build a ensemble model, the predictions made by each of the selected model are going to be observed and a majority vote will be determined, i.e. the prediction of the ensemble model is the prediction that the majority of the selected models made. To compute the majority vote, we can run the following code:

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

The variable *votes* represents the average vote of all the selected model. If the vote is > 0.5, it means that more than 50% of the selected models predict the value Attrited Customer, so that the prediction of the ensemble model is Attrited Customer. Otherwise, the prediction of the ensemble model is Existing Customer.

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

Now, let's evaluate the ensemble model with *Accuracy* and *F-measure*, too.

```
##  Accuracy
## 0.9392892
```

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

```
## [1] 0.7875648
```

The ensemble model also performs well, with an Accuracy of 0.9392892 and an F-measure of 0.7875648. However, it results are not as good as those of the *AdaBoost* model. This is probably because other models do not predict as well as the model *AdaBoost* model itself. Therefore, the model *AdaBoost* is chosen for further optimization.

To optimize the model *AdaBoost*, a larger tuning grid will be used and meanwhile overfitting should be also avoided to make sure that the model is not too specific. There were difficulties in finding the appropriate values for the tuning parameters due to computing power of the hardware. Originally, the optimization was performed with a tuning grid from 0 to 500 with an increment of 20 and it took about 20 hours. This indicates that much more finer parameter optimization (e.g. from 1 to 1000 with an increment of 10) would take even much longer and probably result in overfitting. After many experiments during the optimization, the optimal parameters for the *AdaBoost* model are nIter = 400 with method = "Adaboost.M1".

```
fit_Adaboost <- train(Attrition_Flag ~ .,
                      method = "adaboost",
                      data = train_set,
                      trControl = trainControl,
                      tuneGrid = data.frame(nIter = 400, method = "Adaboost.M1"),
                     )
```

Here is an overview of the inspection of the final model parameters.

```
fit_Adaboost
```

```
## 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, 7290, 7291, 7291, 7291, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9688925  0.8802951
```

```
##
## Tuning parameter 'nIter' was held constant at a value of 400
## Tuning
##  parameter 'method' was held constant at a value of Adaboost.M1
```

After training the optimized *AdaBoost* model, let's check its performance with *Accuracy* and *F-measure*.

```
predictionAdaboost <- predict(fit_Adaboost, test_set)

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

```
## Accuracy
## 0.970385
```

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

```
## [1] 0.903537
```

From the results above, it is shown that the performance of the optimized *AdaBoost* model has been slightly improved. The optimized *AdaBoost* model has an **Accuracy** of **0.970385** instead of 0.9693978 and an **F-measure** of **0.903537** instead of 0.9012739 before optimization.

3. Results

In the *Model Design* section, 9 classification models/algorithms were chosen and the predictions were made based on the data set *Churning Bank Customers*. The *AdaBoost* model performed best with an **Accuracy** of **0.9693978** and an **F-measure** of **0.9012739**. Furthermore, an *ensemble* model was created with the 9 classification models and its **Accuracy** is **0.9392892** and its **F-measure** is **0.7875648**. Last but not least, The *AdaBoost* model has been optimized and the optimized *AdaBoost* model has even an **Accuracy** of **0.970385** and an **F-measure** of **0.903537**. Therefore, the optimized *AdaBoost* model has the best performance.

In Addition, it is interesting to know which features are more important than others in order to predict the *Attrition Flag* more precisely, so that bank managers can use them as indicators to identify potential churning bank customers earlier and provide them better service or special offers in order to keep those customers in their banks. **Variable Importance** can be used to identify such features.

```
varImp(fit_Adaboost)
```

```
## 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
```

In the result above, it is easy to find out which features have bigger impact than other features to predict churning bank customers according to the data set *Churning Bank Customers*, such as Total_Trans_Ct (total transaction count in the last 12 months), Total_Ct_Chng_Q4_Q1 (total change in transaction count from Q4 to Q1), etc. However, on the other hand, some features seem to have almost no impact on the prediction of churning bank customers, such as Card_Category, Education_Level, etc.

4. Conclusion

In the context of the Choose Your Own Project, I explained the goal and key steps of this project. Furthermore, I introduced the background and the use case of the data set Churning Bank Customers, which I also analyzed and explored at first to get the insights of the data. Data Visualization is quite a useful method. In the phase of Model Design, the approach of building and evaluating the prediction models has been explained and the metrics chosen for evaluating the models have been defined and elaborated according to the data insights. In the end, the optimization has been performed and the importance of the features has been discussed. The results have been presented and discussed.

9 classification models/algorithms were chosen to build different prediction models. The *Ensemble* model was also implemented and its performance was evaluated. Last but not least, the *AdaBoost* model has been optimized. As a conclusion, the optimized *AdaBoost* model has the best performance (Accuracy 0.970385, F-measure 0.903537). These kinds of prediction models provide useful insights for bank managers to predict potential churning bank customers and help bank managers prevent those customers from leaving their banks by offering better services as early as possible.

There are also some limitations of this project. First, the data set Churning Bank Customers is not so big and *Attrited Customers* are even fewer (about 16%). Therefore, a much bigger data set could be used and more data insights could be gained during the data exploration, so that the prediction could be more reliable with machine learning algorithms. A second limitation is the lack of the computing power of a normal notebook, despite of the small data set. Some of the algorithms are quite time-consuming. During my project, R was aborted due to intensive computation, so that I had to restart R from time to time and had to give up more advanced and computationally intensive tuning grids or optimization methods in a given time.

As potential future work, a much bigger data set of Churning Bank Customers could be used to gain more insights of the data. The project could be combined by using cloud technology such as Amazon Web Services and Microsoft Azure in order to finish much more intensive computation with other algorithms/models for the prediction. Furthermore, some features with higher importance could be selected and taken into consideration for the prediction models in order to get better predictions, e.g. total transaction count in the last 12 months, total change in transaction count from Q4 to Q1, etc.

5. References

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

- *[2]*: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f
- *[3]*: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226