Report on the Prediction of Bank Churners

Data Science: Capstone - Choose Your Own Project

by Edin Ceman

15th 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. 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 achieve this goal. The data set is originally from a website with the URL as 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"), 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.
- **Marital_Status**, (Factor w/ 4 levels "Divorced","Married"), marital status.

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

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

- 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. 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 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 missing values, a correlation analysis and principal component analysis is performed.

- 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 method are considered in the model design to find the optimal 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 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, building 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*.

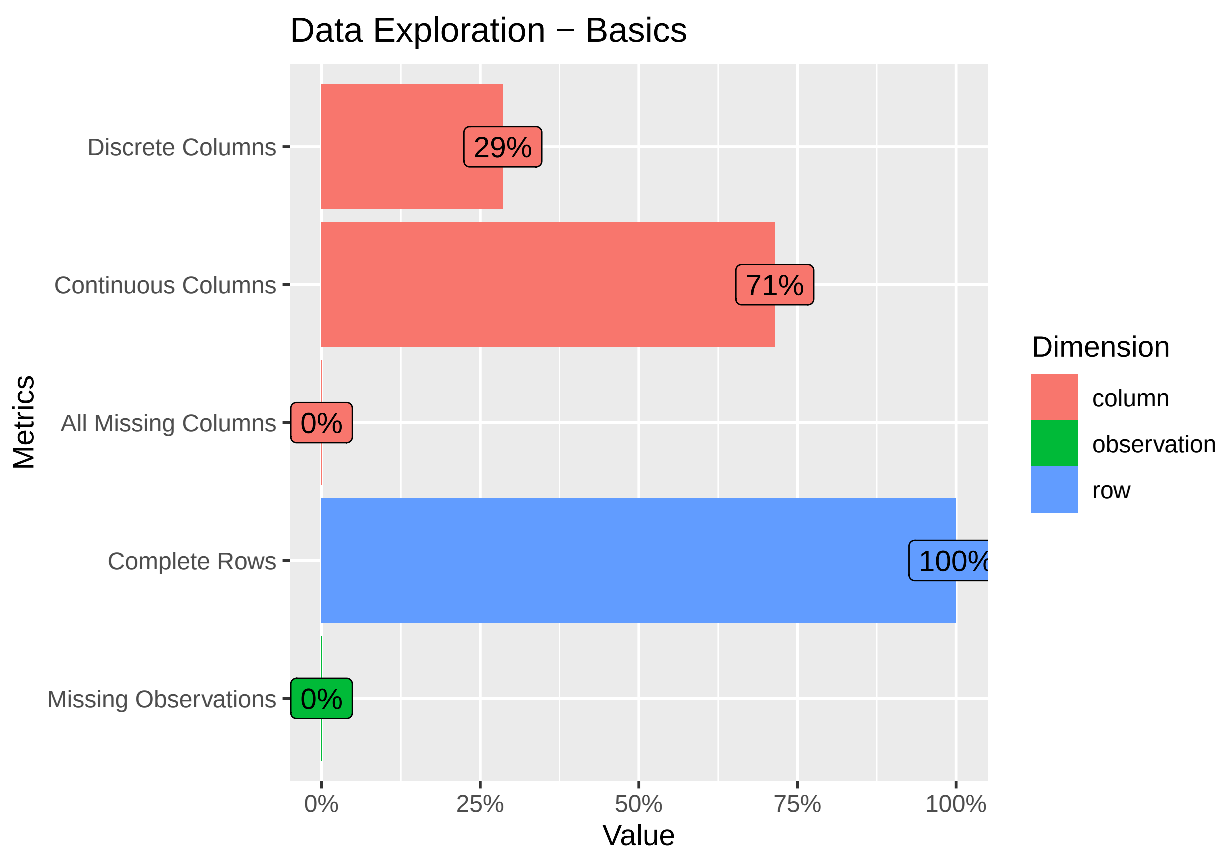
```r
data <- read.csv(curl("https://raw.githubusercontent.com/edce1987/edx_edcem_,
CYO/main/BankChurners.csv"), header = TRUE, stringsAsFactors =  TRUE)

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

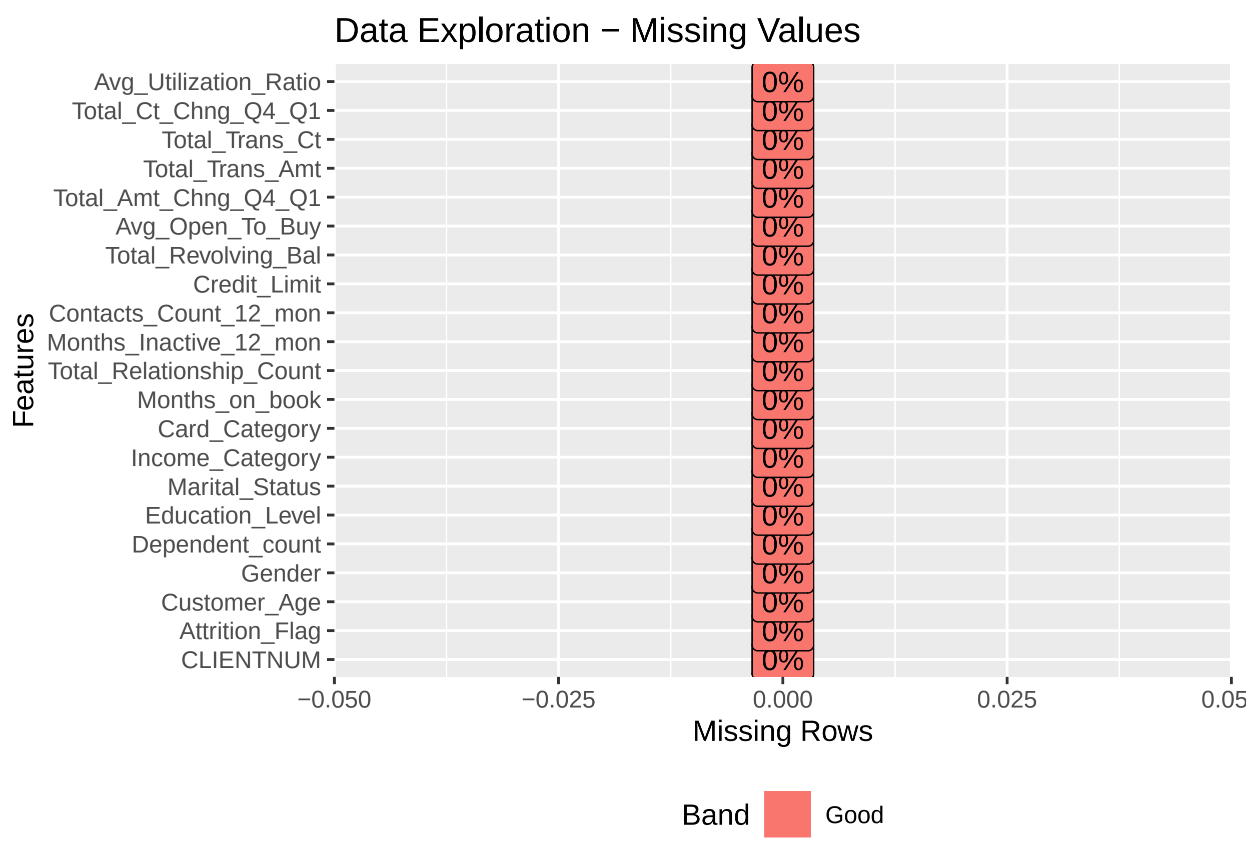
In this section, the *Bank Churners* data set is explored in more detail. Personally, I prefer to use the package “DataExplorer” for a detailed analysis. It is a very powerful package that quickly generates a detailed (html) report for a given data set. The report includes some important metrics such 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.

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



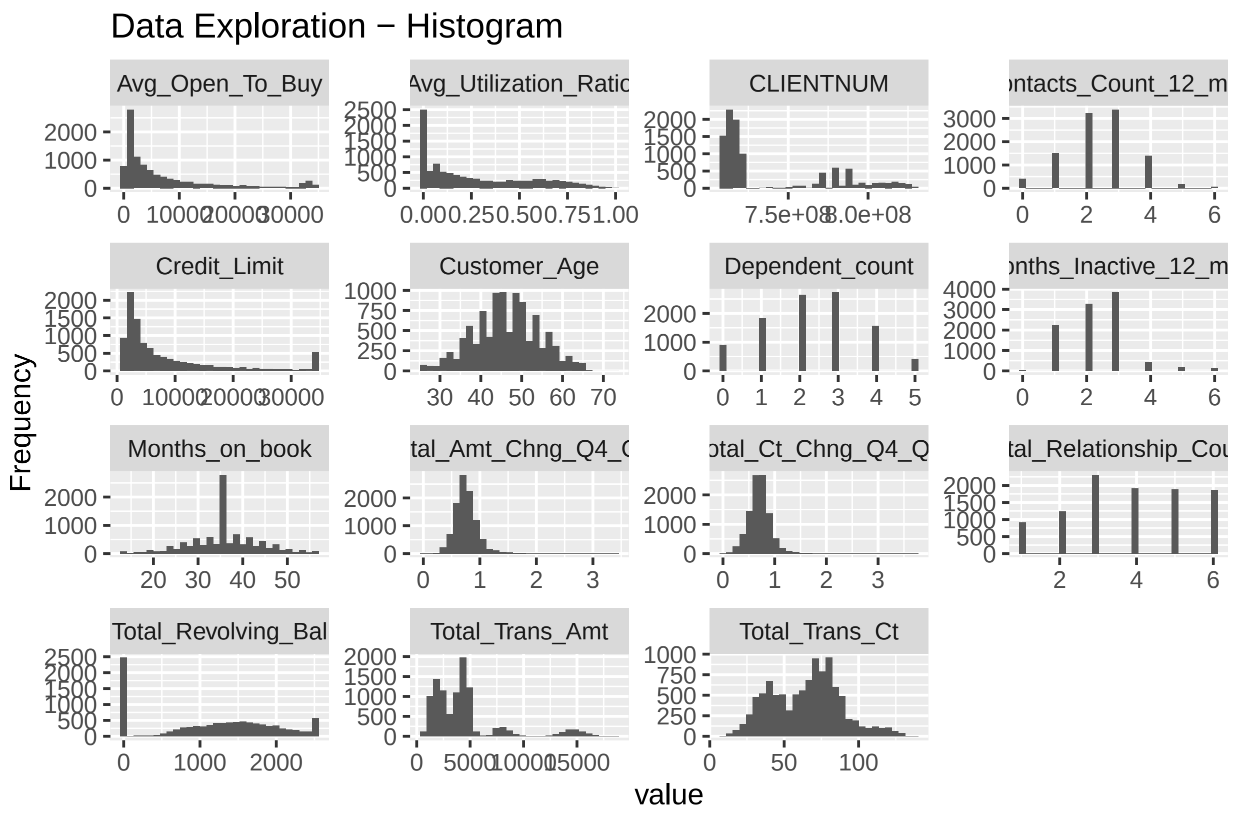
From this, we see that the data set has **29% discrete columns** (i.e. factor), and **71% continuous columns** (i.e. numerical or integer). We can also see that there are no missing values, and that the rows are complete. To have a more detailed overview on potentially missing values, we can perform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

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



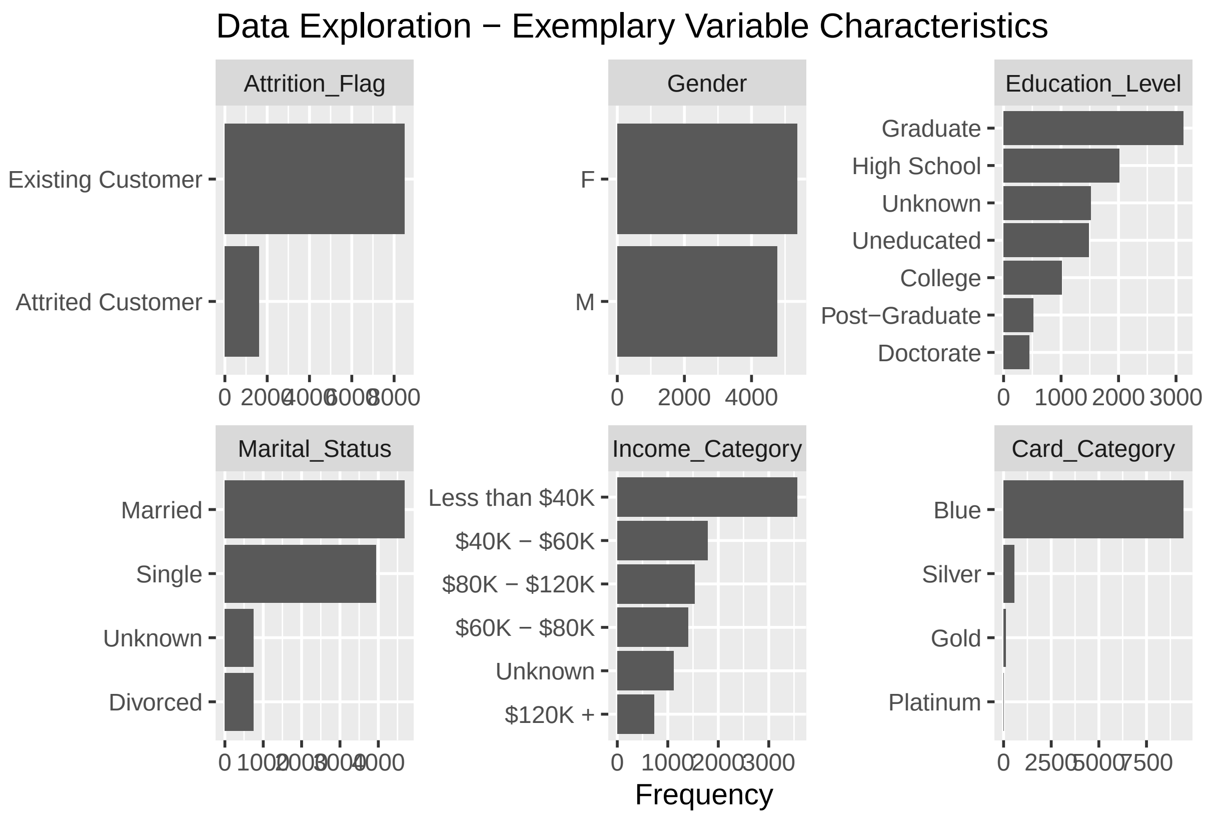
In this plot, we can see a more detailed overview of missing values for each variable. However, since there are no missing values for the variables, we don't need to perform further steps to fill them. Another important information is the distribution of the values for each of the variables. By using a histogram, we can visually inspect the distribution to see if the variables' values are e.g. normally distributed and what kind of special characteristics they show.

```r
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.

```r
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. 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*. 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*.

Of course, we could go into even more detail at this point and e.g. perform correlation analyses for the variables and perform a dimensionality reduction using a principal component analysis. However, I have scoped those steps out and decided to simplify the model design process in this regard. Hence, we will apply the 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 to fill NAs. We can move further and prepare the data set for the model design. To conduct that, we split the set *data_clean* 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.

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

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

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

2.3 Model Design

The recommendation model we want to build is called **Regularized Recommendation Model with Movie, Restricted User, Time & Genre Effect**. The goal of the model is to predict a rating for a certain movie, by a certain user at a certain point in time using the *MovieLens* data. The model is designed such that it captures different biases or effects in the training data set to identify the **biases** or **effects** that influence or shape the resulting rating. For the model, we are going to include a *movie*, *user*, *time* and *genre* bias. Those will be explained later in this section. For each of the selected effects, the concept of **regularization** is used to account for **overfitting**. This issue 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.

To apply the concept of regularization, we introduce a penalty parameter **lambda** to each bias. The penalty parameter is applied to penalize the effect of a bias such that it's influence on the predicted rating is adjusted. The penalty parameter can be considered as a tuning parameter. Hence, we will define a range of lambdas, and repeat the model training and evaluation against the test set to find the optimal lambda. From my trials, I narrowed down the optimal lambda range to be somewhere between 4.5 to 5.5 in 0.1 steps resulting in only 10 iterations to avoid excessive computing time. We could also conduct a finer penalty parameter search, e.g. 1 to 10 in 0,001 steps which would result in 9,000 iterations to optimize our algorithm. However this approach would most probably result in overfitting and massively extend computing time. Hence, we will stick to the selected sequence from 4.5 to 5.5 in 0.1 steps.

To build the recommendation model, we will use the *train* set and evaluate it with the *test* set that we have created in the previous section. To find our optimal lambda, we will use the *Root Mean Squared Error* (RMSE) to evaluate the model performance for each iteration. The RMSE is a widely used performance measure and is constructed such that it calculates the standard deviation of the *residuals*, i.e. the difference between the true rating and the predicted rating. The RMSE is always non-negative due to the residuals getting squared, and it therefore also penalizes large deviations disproportionally. Hence, it is sensitive to outliers. In terms of evaluation, the smaller the RMSE is, the better is the performance of our model.

We construct our recommendation model using different biases or effects in the data set. A *bias* or an *effect* can be considered a feature or explanatory variable that captures a certain pattern that influences a user's movie rating. As a starting point, some of the biases have been selected on the basis of the information that was provided in the edx module "Data Science - Machine Learning", and on top of that, the recommendation model has been extended, refined, and developed further based on additional insights.

The recommendation model at hand considers different biases or effects in the data, i.e. more specifically the training data. Each bias or effect is constructed such that it captures the difference or "distortion" of an effect in relation to the overall average rating (called *avg*). This difference (= bias) is averaged and penalized with lambda to avoid overfitting. For example, the *Movie Effect* is capturing the averaged and penalized difference of ratings belonging to a certain movie rating in relation to the overall average *avg*. Every additional effect is building on top of that, i.e. for example the subsequent *User Effect* is capturing the averaged and penalized difference of a certain user and his ratings in relation to the overall average *avg* and the *Movie Effect*. This leads to the insight that the order of effects that we use to construct our recommendation model has an impact on the model outcome and performance. The earlier an effect is in the sequence, the more weighty it is by design. I have chosen this order since it produced the best results. Hence, the model has the following effects:

- **Movie Effect:** This parameter captures the effect that a certain movie has on the rating, while adjusting for the overall average rating and regularizing it. Assumption: The movie itself has an effect on the rating.
- **User Effect:** This parameter captures the effect that a certain user has on the rating, while adjusting for the overall average rating and the movie effect and regularizing it. Assumption: The user's behavior and taste has an effect on the rating. We have filtered the *edx* data set in the *Data Preparation* section to users with at least

15 ratings. The assumption is that users who rate more often have more experience / more data points and thus enable more reliable insights into their behavior.

- Time Effect: This parameter captures the effect of the date (time) on the rating, i.e. the time when a rating was made, while adjusting for the overall average, the movie effect, the user effect and regularizing it. Assumption: The date when a rating was made has an effect on the rating.
- Genre Effect: This parameter captures the effect the genre of the movie has on the rating, while adjusting for the movie, user, and time effect and regularizing it. Assumption: The genre that a movie is belonging to has an effect on the rating.

The code below builds the model and conducts a search for the optimal penalty parameter lambda based on the train and test set. Predictions are calculated for the test set as the sum of *avg*, *Movie Effect*, *User Effect*, *Time Effect* and *Genre Effect* for each row in the test set. Please note that we generate an additional feature called *movie_avg* that represents the specific movie average in contrast to the overall average *avg*. The movie average is used in the prediction process to fill NAs or missing predictions with a finer measure than the overall average. Due to our applied filtering in the *Data Preparation* section (only users with more than 15 ratings) and selected biases (especially time and genre), some NAs or missing values are present in the predictions since there is no information available for all movie, user, genre, time combinations. Hence, we replace those NAs with either 1. the movie average (more precise) or 2. with the overall average in case the movie average is not available too. Additionally, we observe that computed predictions are numerical and can therefore be out of the "allowed" rating range from 0.5 to 5.0 (see *1.2 Data Set* section). To avoid penalties in the RMSE, we trim these values to the nearest allowed rating.

Finally, each iteration of a lambda is evaluated using the RMSE.

The following figure shows the result of the parameter search:

The optimal lambda or penalty parameter is the value that minimizes the RMSE. In this case it is 4.8.

We can use this insight for the subsequent model evaluation.

2.4 Model Evaluation

From the previous section, we have identified the optimal penalty parameter lambda to be **4.8**. We can now re-run the recommendation model one more time with the optimal parameter to evaluate our model with the training and test data.

We can observe that the recommendation model produces a **training RMSE** of **0.8646391** with the optimal penalty parameter based on the train and test set. Please note that this is not the final RMSE using the validation set. The final evaluation of the recommendation model will be conducted in the next section.

3. Results

This section will perform the final evaluation of the recommendation model. Subsequently, the model results will be presented and discussed. To evaluate the model results, we will use the RMSE as before.

The following code performs the final evaluation of the recommendation model using the final hold-out validation set.

We receive a **final RMSE** of **0.8646854** on the validation set. We can see that the final RMSE has slightly worsened in comparison to the RMSE when using the test data set in the previous section. This was expected since the search for the optimal parameter lambda was conducted on the test set and therefore performed slightly worse on the validation set. This might be a minor overfitting issue. However, the difference is marginal.

4. Conclusion

The report at hand described the key steps and results of my *Capstone Project*. Initially, the *MovieLens Project* and the used data set have been explained and background information was provided. Following that, the goal and the key steps of the *Capstone Project* have been described. Afterwards, the modeling process as well as the approach for building and evaluating the recommendation model have been explained and a reasoning for the chosen design has been given. Finally, the results have been evaluated, presented and discussed. The recommendation model has a final RMSE of **0.8646854** on the **validation set**.

The limitations of this project are that we have used a restricted *MovieLens* data set (10m) instead of the full-scope data set. The full-scope set might have revealed further effects or insights into the data. Another limitation - even though we used a reduced data set - is the limitation that is put 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 modeling techniques in a reasonable time, e.g. using ensemble methods with Random Forests, AdaBoost or GamLoess. Future work can focus on the full-scope *MovieLens* data set, and even join additional publicly available data to the existing data to gather further insights. In addition, the usage of cloud services, e.g. *Microsoft Azure*, *Google Cloud Platform* or *Amazon AWS* could enable the application of more advanced and computationally intensive models in a reasonable time frame.