Orange

# Customer churn prediction report

AUTHOR: PAWEL CISLO <PAWEL.CISLOO@GMAIL.COM>

DATE: 26/03/2020

# Table of Contents

# Table of Figures

# 1. Introduction

Orange wants to estimate the risk of customers churning (i.e. leaving Orange). For this case, a prediction has been created to find out the customers who are most likely to churn out in the next two months with their indicating characteristics.

This document continues with a detailed dataset description explaining the number of available data types, followed by visualisation graphs presenting the distribution of categorical and numerical values in the set.

Afterwards, the 3rd chapter explains choices and possible different routes which could have been taken to tackle this scenario.

The 4th section describes the technologies used to create the prediction, as well as outlines the steps needed to recreate the analysis.

Finally, the last two chapters present the results with an elaboration on the major conclusions in terms of the implemented technical approach and the returned business value coming out from this research.

# 2. Dataset description

The attempt to solve the introduced customer churn prediction problem is performed on the two datasets of Orange customers:

- *orange_train.csv* – dataset used to make analysis and machine learning model
- *orange_score.csv* – dataset used for prediction.

Both datasets contain the same type of information with two major differences. The number of records in *orange_score.csv* is smaller (10% of the overall *orange_train.csv* records), and the *churned* target feature is missing and needs to be predicted.

Furthermore, both datasets contain unique information on different customers and contain no missing values.

The more extensive description of the files is outlined in the tables below:

| Type of information | Training data | Testing data |
|---|---|---|
| Number of observations | 101000 (out of which, 1000 are duplicated)<br><br>After removing duplicates: 100000<br><br>After removing negative values: 99958 | 10000 |
| Number of variables | 18 | 17 |
| Total size in memory | 14.5 MiB | 1.4 MiB |
| Values missing | 0% | 0% |
| Variable types | • Numeric: 12<br>• Categorical: 3<br>• Boolean: 2<br>• Index: 1 | • Numeric: 12<br>• Categorical: 3<br>• Boolean: 1<br>• Index: 1 |

*Table 1 - Summary of information in training and testing sets*

The following table further specifies types of variables from the training data:

| Key | Data type | Range | Mean |
|---|---|---|---|
| primary_key | index | 0 – 100000 | - |
| r_age_val | numeric | 0.00003 – 0.9999 | 0.50 |
| cust_gender_cd | categorical | F, M, Unknown | - |
| cust_language_cd | categorical | DE, EN, FR, NL | - |
| cust_mkt_segm_desc | categorical | MASS, SOHO | - |
| trf_mdl_phonedeal_cd | boolean | 0, 1 | - |
| count_orange | numeric | 0 – 6 | 0.19 |
| cust_total_mobile_qty | numeric | 1 – 6 | 1.53 |
| voice_oob_mean | numeric | 0 – 375.4 | 1.76 |
| voice_oob_sum | numeric | 0 – 1126 | 5.27 |
| voice_oob_nat_mean | numeric | 0 – 275.47 | 1.09 |
| voice_oob_nat_sum | numeric | 0 – 826 | 3.25 |
| mean_bill_rev_vs_trf_plan | numeric | 0 – 39.507 | 1.10 |
| tenure_days | numeric | 88 – 7461 | 2620 |
| days_since_moving | numeric | 4 – 365 | 356 |
| nb_cont | numeric | 0 – 1847 | 46 |
| avg_tp_churn | numeric | 0.004 – 0.053 | 0.029 |
| churned | boolean | 0, 1 | - |

*Table 2 – "orange_train" dataset description*

The rest of the graphs present the distribution of variables.

## 2.1. Categorical distributions

The difference between customer gender is close to equal:

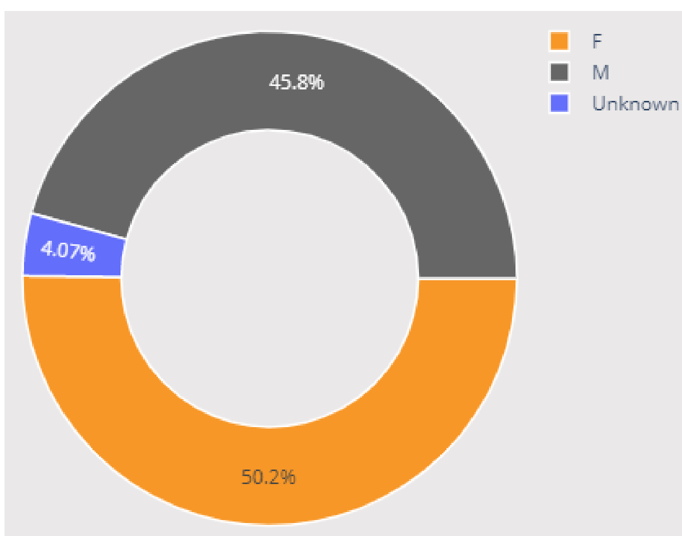



*Figure 1 - "cust_gender_cd" distribution*

99% of customers use either French or Dutch language:

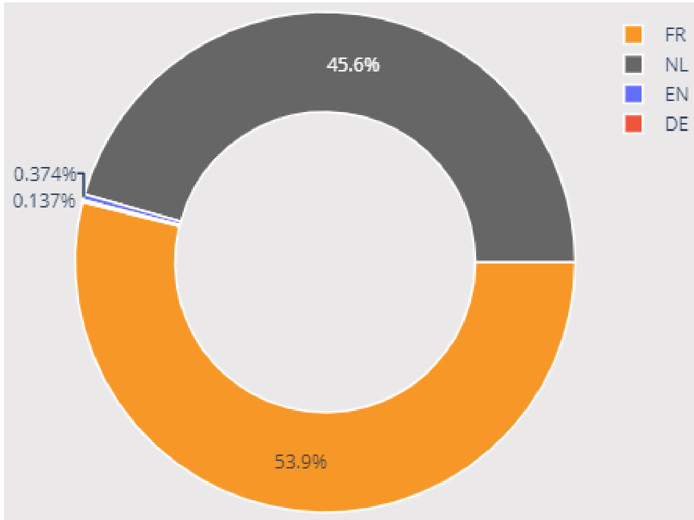



*Figure 2 - "cust_language_cd" distribution*

Only 7% of customers come from the SOHO market:

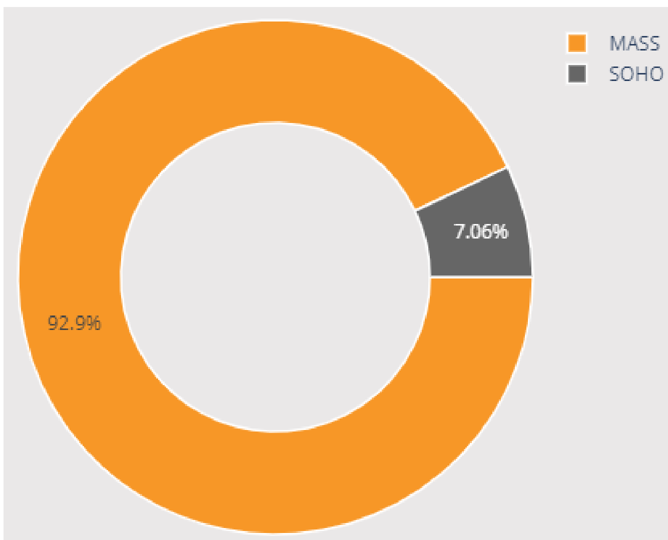



*Figure 3 - "cust_mkt_segm_desc" distribution*

Two-thirds of customers do not use *trf_mdl_phonedeal_cd*:

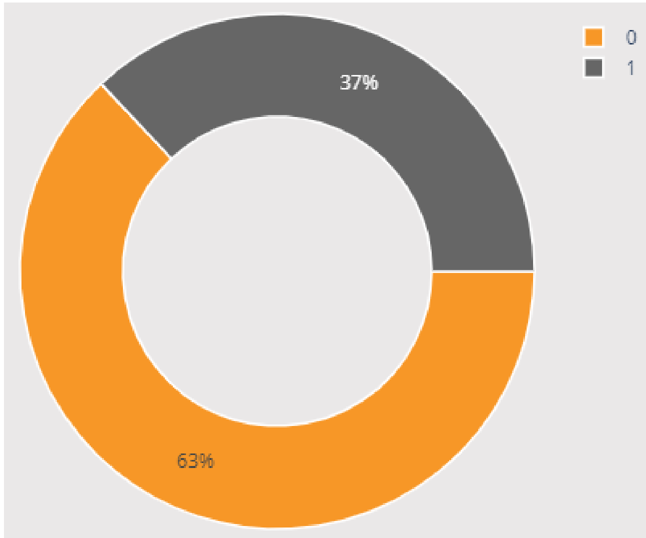



*Figure 4 - "trf_mdl_phonedeal_cd" distribution*

Only 17% of the values are higher than 0 for *count_orange*:

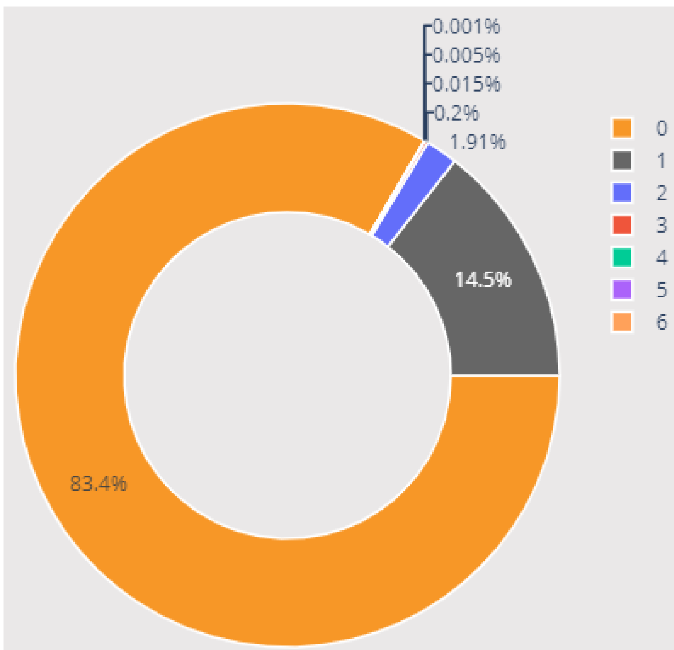



*Figure 5 - "count_orange" distribution*

36% of customers have more than one mobile phone:

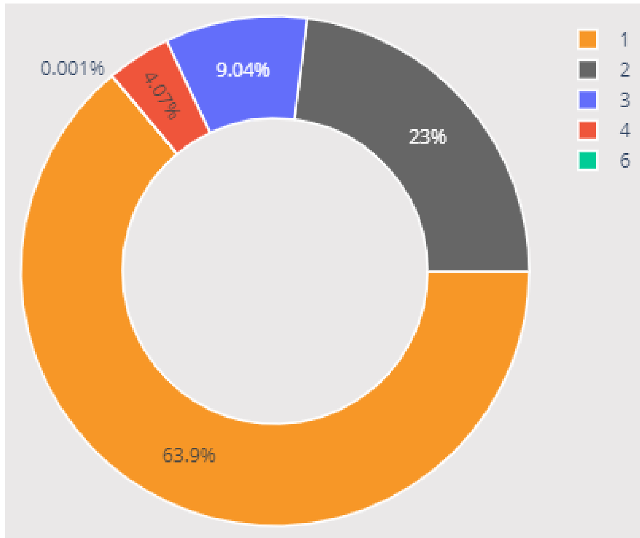



*Figure 6 - "cust_total_mobile_qty" distribution*

Analysis of churn rate distribution has been presented in § 5.1.1.

## 2.2. Numerical distributions

*r_age_val* is distributed equally from 0 to 1:

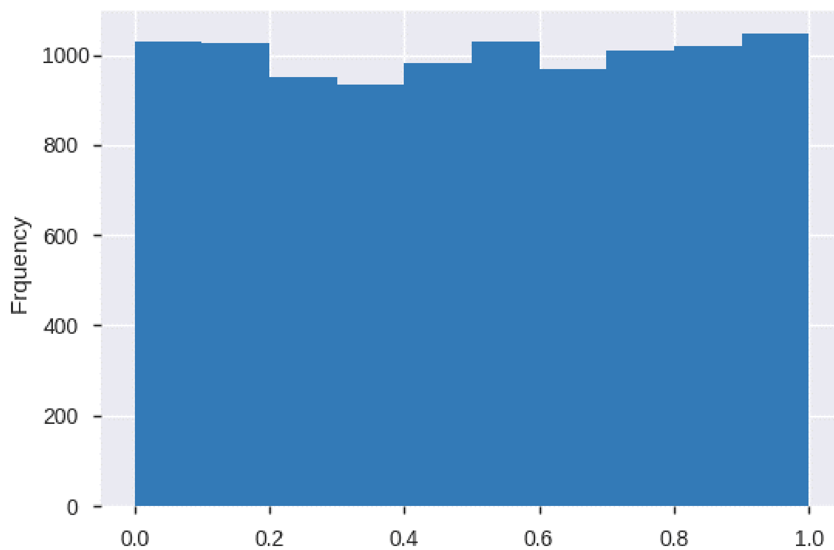
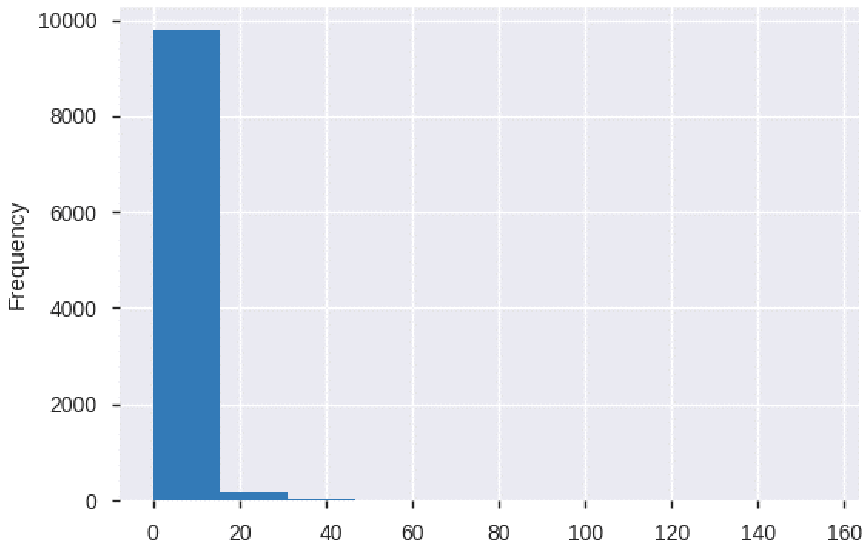



*Figure 7 - "r_age_val" frequency distribution*

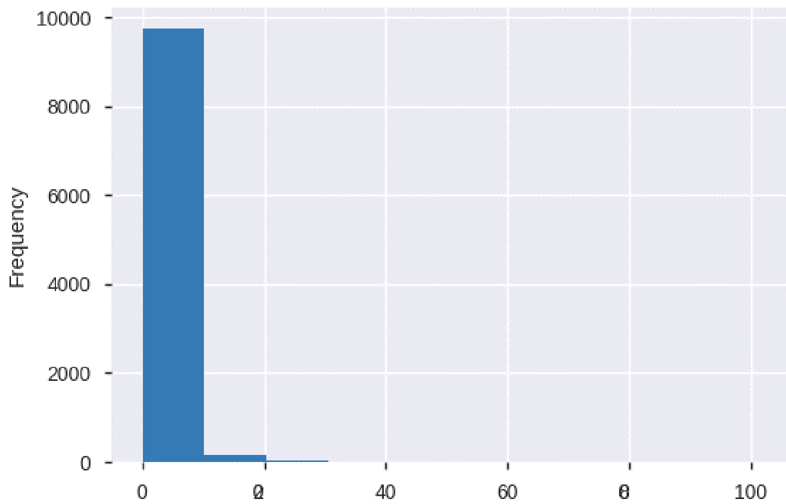
68% of *voice_oob_mean* are equal to 0:




| Value | Count | Frequency (%) |
|---|---|---|
| 0.0 | 6748 | 67.5% |
| 0.2755 | 115 | 1.2% |
| 0.17906667 | 72 | 0.7% |
| 0.551 | 54 | 0.5% |
| 0.8265 | 20 | 0.2% |
| 0.41323333 | 18 | 0.2% |
| 0.35813333 | 18 | 0.2% |
| 0.55096667 | 17 | 0.2% |
| 0.45456667 | 14 | 0.1% |
| 1.102 | 10 | 0.1% |
| Other values (2731) | 2905 | 29.1% |

*Figure 8 - "voice_oob_mean" frequency plot*

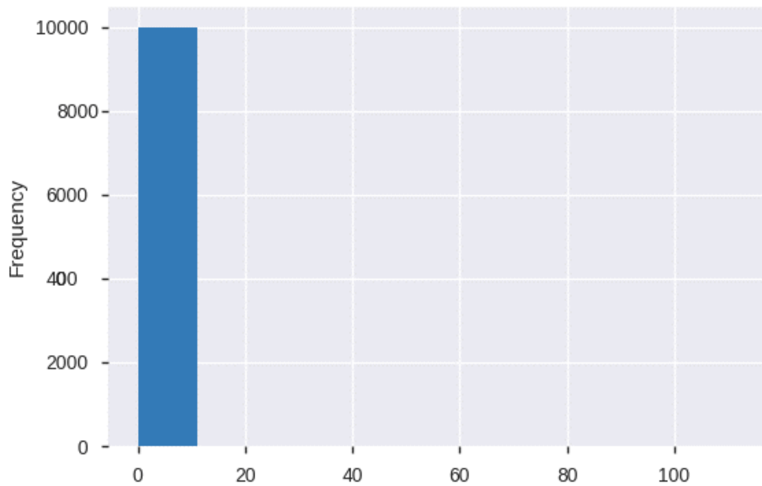
82% of *voice_oob_nat_mean* are equal to 0:




| Value | Count | Frequency (%) |
|---|---|---|
| 0.0 | 8186 | 81.9% |
| 0.06313333 | 3 | 0.0% |
| 0.01836667 | 3 | 0.0% |
| 0.10099999999999999 | 2 | 0.0% |
| 0.15956667 | 2 | 0.0% |
| 0.74386667 | 2 | 0.0% |
| 0.03443333 | 2 | 0.0% |
| 0.32256667 | 2 | 0.0% |
| 0.05623333 | 2 | 0.0% |
| 1.10196667 | 2 | 0.0% |
| Other values (1761) | 1785 | 17.9% |

*Figure 9 - "voice_oob_nat_mean" frequency plot*

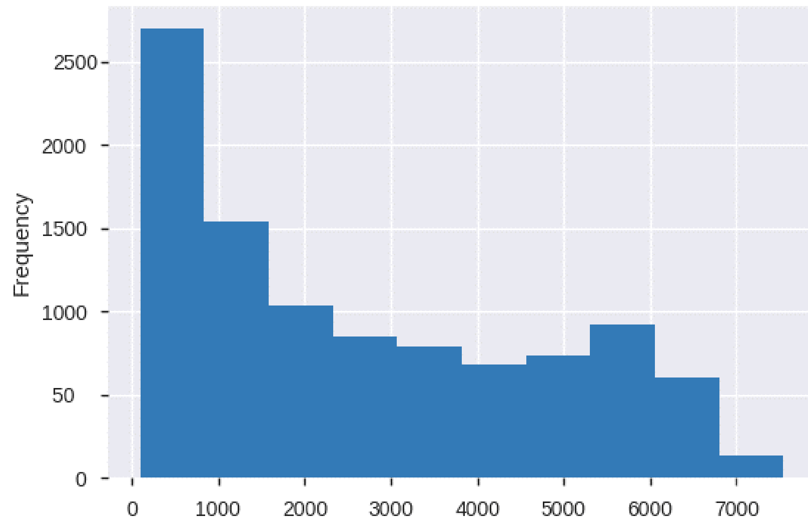
Most of the *mean_bill_rev_vs_trf_plan* values are close to 1:




| Value | Count | Frequency (%) |
|---|---|---|
| 0.826446666666667 | 394 | 3.9% |
| 0.82645 | 278 | 2.8% |
| 0.826441666666667 | 238 | 2.4% |
| 0.8264450000000001 | 184 | 1.8% |
| 0.99174 | 109 | 1.1% |
| 0.8264440000000001 | 92 | 0.9% |
| 0.72727 | 50 | 0.5% |
| 0.826448 | 43 | 0.4% |
| 0.8264457142857142 | 41 | 0.4% |
| 0.8308911113333329 | 39 | 0.4% |
| Other values (7636) | 8523 | 85.3% |

*Figure 10 - "mean_bill_rev_vs_trf_plan" frequency plot*

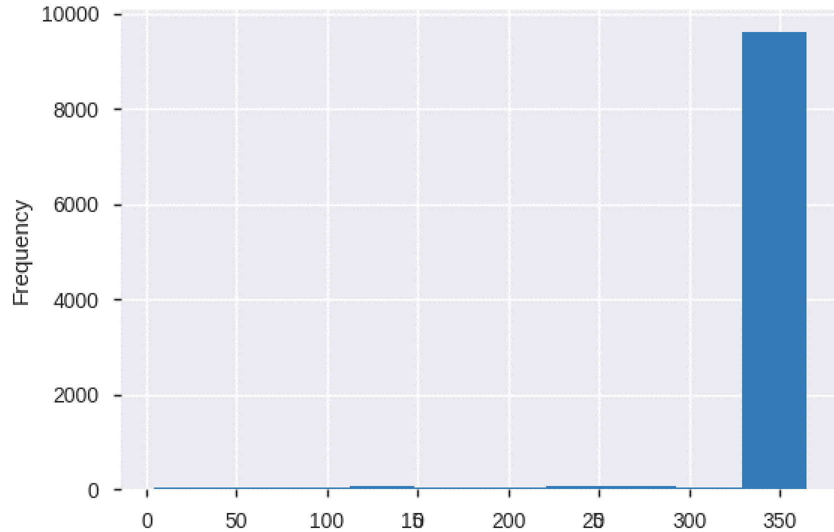
Tenure days plot resembles right-skewed shape:




| Value | Count | Frequency (%) |
|---|---|---|
| 117 | 17 | 0.2% |
| 497 | 14 | 0.1% |
| 174 | 13 | 0.1% |
| 125 | 13 | 0.1% |
| 480 | 12 | 0.1% |
| 5711 | 12 | 0.1% |
| 277 | 12 | 0.1% |
| 511 | 11 | 0.1% |
| 489 | 11 | 0.1% |
| 499 | 11 | 0.1% |
| Other values (4517) | 9865 | 98.7% |

*Figure 11 - "tenure_days" frequency plot*

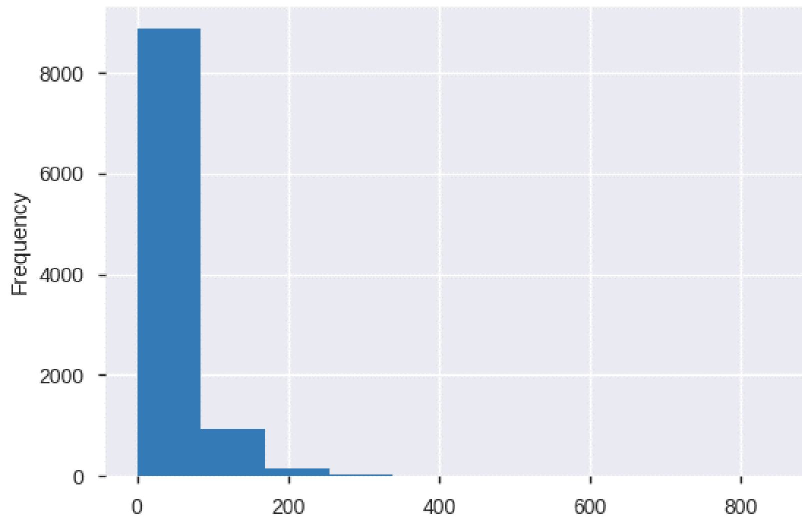
96% of customers are there 1 year since moving:




| Value | Count | Frequency (%) |
|---|---|---|
| 365 | 9562 | 95.7% |
| 349 | 7 | 0.1% |
| 314 | 7 | 0.1% |
| 307 | 6 | 0.1% |
| 95 | 5 | 0.1% |
| 74 | 5 | 0.1% |
| 112 | 5 | 0.1% |
| 4 | 5 | 0.1% |
| 116 | 4 | 0.0% |
| 355 | 4 | 0.0% |
| Other values (226) | 381 | 3.8% |

*Figure 12 - "days_since_moving" frequency plot*

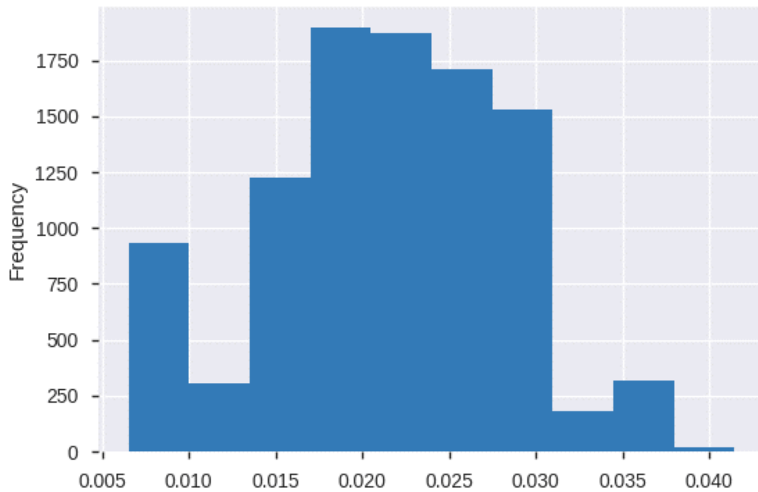
*Nb_cont* values tend to oscillate within values smaller than 200:




| Value | Count | Frequency (%) |
|---|---|---|
| 18 | 194 | 1.9% |
| 21 | 191 | 1.9% |
| 28 | 190 | 1.9% |
| 30 | 181 | 1.8% |
| 23 | 178 | 1.8% |
| 27 | 176 | 1.8% |
| 24 | 173 | 1.7% |
| 17 | 171 | 1.7% |
| 38 | 170 | 1.7% |
| 26 | 169 | 1.7% |
| Other values (270) | 8198 | 82.1% |

*Figure 13 - "nb_cont" frequency plot*

Distribution of *avg_tp_churn* is close to the shape of Gaussian distribution:




| Value | Count | Frequency (%) |
|---|---|---|
| 0.0212201105862536 | 1203 | 12.0% |
| 0.0254639285841984 | 1127 | 11.3% |
| 0.0170355247039608 | 830 | 8.3% |
| 0.0185643833235569 | 617 | 6.2% |
| 0.0288548241659152 | 568 | 5.7% |
| 0.0165509521461601 | 526 | 5.3% |
| 0.00956684325334995 | 409 | 4.1% |
| 0.0216953741472364 | 374 | 3.7% |
| 0.00652020082892476 | 355 | 3.6% |
| 0.0287215542549412 | 346 | 3.5% |
| Other values (32) | 3636 | 36.4% |

*Figure 14 - "avg_tp_churn" frequency plot*

# 3. Implemented approach

The performed analysis uses Python programming language since it is one of the most frequently used programming languages for financial data analysis, with plenty of useful libraries and built-in functionality. Furthermore, the online Google Colab notebook was used as the coding environment for the reason of testable, well-documented code and the ability to see the results immediately from the specific parts of code. Moreover, Google Colab did not require the installation of Python, as well as all the imported libraries.

The customer churn prediction worked mostly on the training dataset, which was also used to train all the machine learning models. Afterwards, the testing dataset was used to output the churn rate prediction for the 10000 customers.

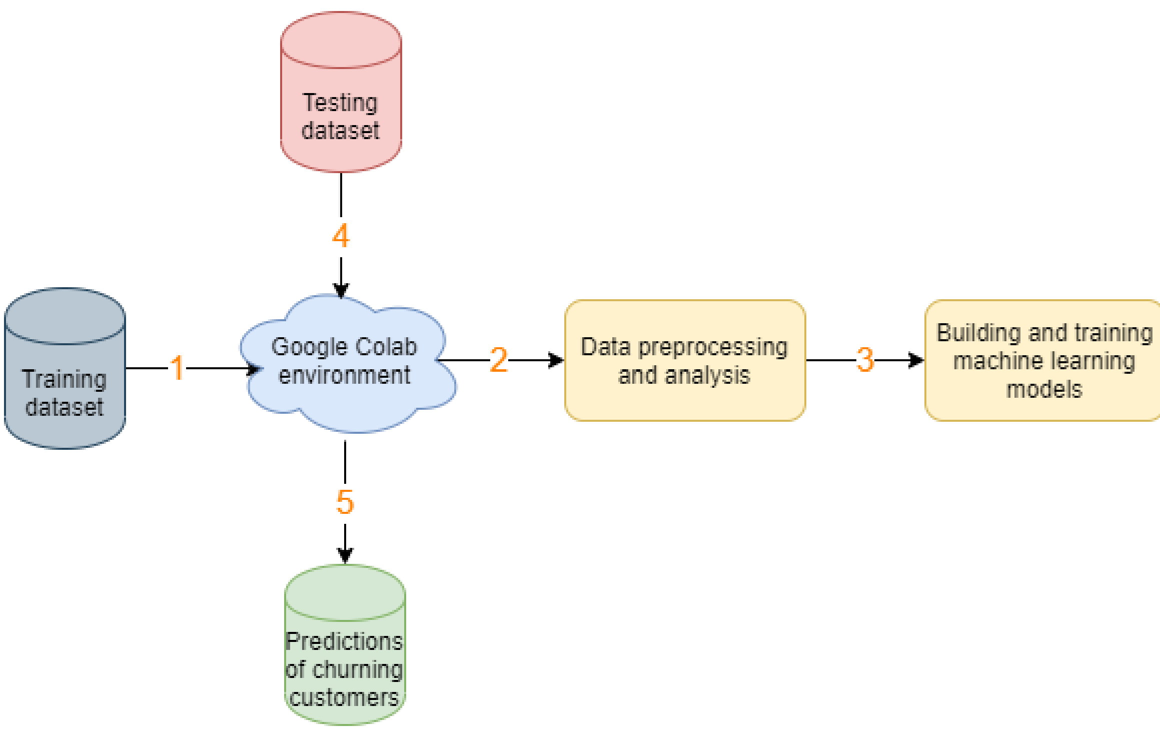



*Figure 15 - Steps performed in the customer churn prediction*

Specific operations performed inside the IPython notebook are described in the following subsection.

## 3.1. Code structure

The code structure inside a .ipynb notebook is supported with the markdown comments that explain the performed operations.

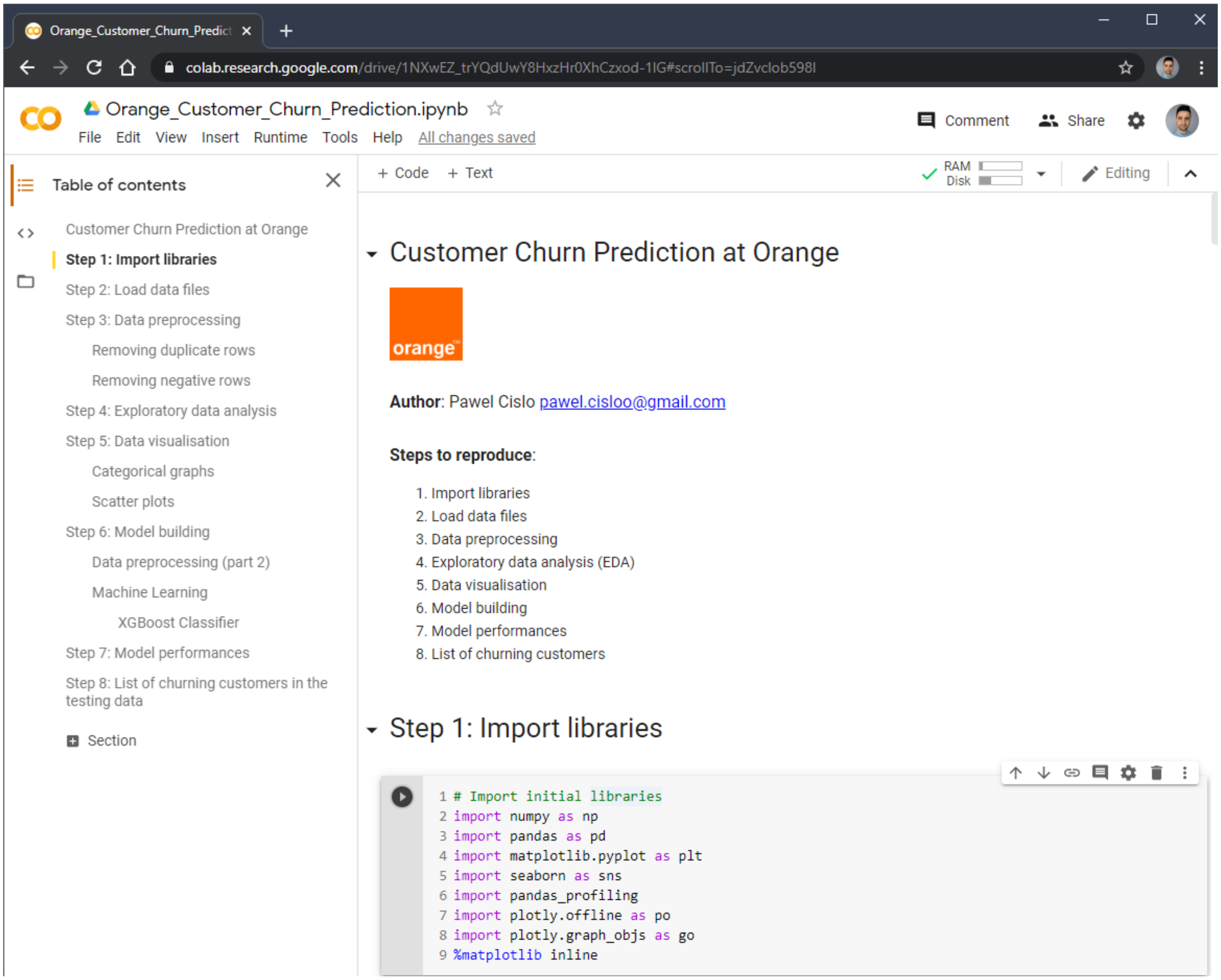



*Figure 16 - Window of a Google Colab experiment window used for the prediction*

As presented in the figure above, the steps taken to analyse the code are the following:

1. Import of libraries
2. Load of data files
3. Data preprocessing
4. Exploratory data analysis
5. Data visualisation
6. Model building
7. Comparing model performances
8. Generating a list of churning customers.

**Step 1 – Import of libraries**:

The initial Python libraries are imported:

- Numpy (support for large, multi-dimensional arrays and matrices)
- Pandas (data structures and operations for manipulating numerical tables and time series)

- Pandas Profiling (generation of profile reports)
- Matplotlib (basic plotting library)
- Seaborn (extended plotting library)
- Plotly (interactive plotting library)

**Step 2 – Load of data files**:

Both of the files are loaded with using Pandas read_csv() function.

**Step 3 – Data preprocessing**:

- Datasets are checked for duplicate rows, where one thousand duplicated records have been removed prior to the analysis from the *orange_train.csv*
- Negative outliers are removed from the *orange_train.csv*.

**Step 4 – Exploratory data analysis**:

- Data is checked for the null values, and none is found
- Profile report is generated using Pandas Profiling library
- Correlation heatmap between features is generated using the Matplotlib library.

After this step, it is known that both datasets are clean and quite normalised; therefore, they should not require any manipulation. The only concern may be raised by the *Unknown* value of the *cust_gender_cd* field, which is only 4% of all the values.

**Step 5 – Data visualisation**:

Plotly is used to generate categorical graphs (for the categorical columns) and scatter plots (for the numerical columns) between all the variables and the *churned* field. This step gives an impact of what features might affect the churning of customers.

**Step 6 – Model building**:

At first, another data preprocessing step is performed:

- Model building is preceded with One Hot Encoding on categorical & boolean values using get_dummies method. This helps the machine learning algorithms to use the non-numerical values in the training step.
- Scikit-learn is used to perform feature scaling since most of the values do not line on the same scale. Without such optimisation, one could not get optimised predictions.

The analysis continues with building machine learning models:

- Feature variable X and target variable y is created
- Training data is split in the ratio 70:30 (70% for training and 30% for testing) so that the model will be able to test itself on data it did not see before.
- 5 typical classification models are imported from the Scikit-learn library (logistic regression, k-nearest neighbours, support vector machines, decision tree and random forest) together with the XGBoost (distributed gradient boosting library). The use of libraries was random since according

to the no free lunch theorem, no model works well for every problem. Parameters of the models were set according to the gut instinct.

- The models are fit, trained and tested on the training data.

XGBoost is further used for extended data analysis:

- The plot_tree function is used to create the boosting decision tree for determination of characteristics impacting the target variable.
- Plot_importance function generates the feature importance graph to visualise the most impactful features for the algorithm prediction.

**Step 7 – Comparing model performances**:

- Used models are compared in a single Pandas DataFrame
- The confusion matrix is generated to check the number of correct and incorrect predictions

**Step 8 – Generating a list of churning customers**:

logmodel.predict_proba() function is used to assign the churn probability rate to the customers from the testing set. Later to_csv() function saves the results in a CSV file.

# 4. Experimental setup

This section explains the technologies used to create the prediction, as well as provides a step by step usage tutorial.

## 4.1. Prerequisites

The experiment has been run entirely on a cloud service: Google Colab with the use of provided GPU hardware acceleration; therefore, the code can be compiled on any desktop machine with access to a web browser and a registered Google account.

In case you prefer to run the code locally, please follow the steps indicated in the next subsection.

## 4.2. Usage tutorial

**Running the experiment in the cloud (recommended)**:

1. Open colab.research.google.com in your browser.
2. Log in with your Google account.
3. Select from the menu: "*File*" > "*Upload notebook*" > "*Choose file*" and select the *Orange_Customer_Churn_Prediction.ipynb* provided with this document.
4. Using the left-hand-side "*Files*" icon upload files in the following schema to the root directory:
   4.1. DATASETS
      4.1.1. Orange_data
         4.1.1.1. orange_score.csv
         4.1.1.2. orange_train.csv
   Alternatively, you can modify the paths used in the following cell:

   Step 2: Load data files

```
# Load customer data files
training_data = pd.read_csv('/DATASETS/Orange_data/orange_train.csv')
testing_data = pd.read_csv('/DATASETS/Orange_data/orange_score.csv')
```

*Figure 17 - Data loading function*

5. Continue by running the cells in sequential order, as otherwise there will be errors of unassigned variables or missing libraries.

**Running the experiment locally**:

1. Make sure you have Python installed on your computer.
2. Install all the necessary libraries:
   a. Numpy

   b. Pandas
   c. Matplotlib
   d. Seaborn
   e. Pandas Profiling
   f. Plotly
   g. Sklearn
   h. Xgboost
3. Open the *orange_customer_churn_prediction.py* file in a Python editor, such as in the cross-platform software like Visual Studio Code.
4. Make sure that the functions to load the file paths (as indicated on the figure above) are set up correctly.
5. Compile the code.

# 5. Results

This section presents and interprets the results returned from the customer churn prediction.

## 5.1. Exploratory Data Analysis

The correlation heatmap outlines the connection between the *voice* and *mean_bill* related columns. *r_age_val* feature seems to be also greatly correlating with those values.

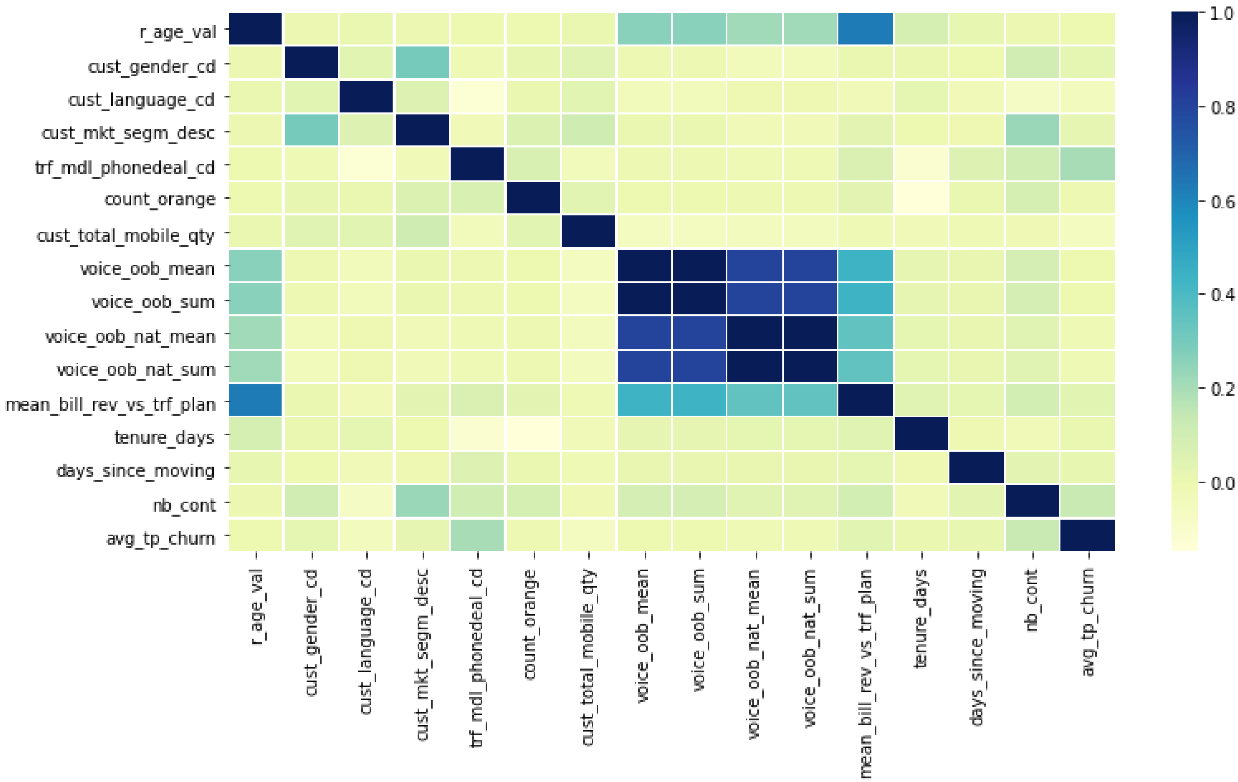



*Figure 18 - Correlation heatmap between dataset features*

### 5.1.1. Churn related categorical graphs

Overall, the dataset contains very little (2.2%) of churned examples, which might not be enough to generate the most accurate characteristics of churning customers:

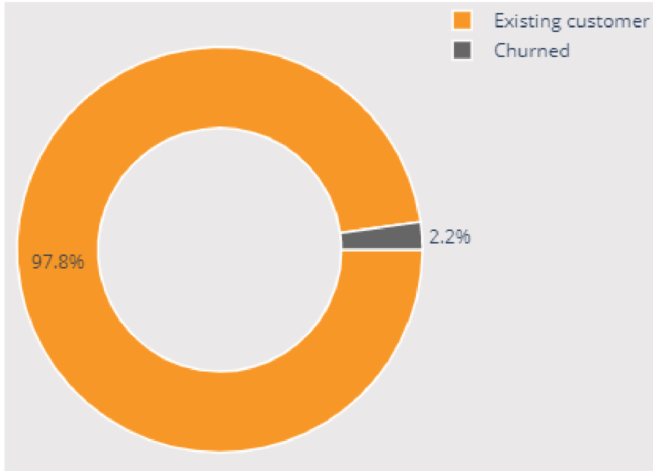



*Figure 19 - 100 000 customer churn comparison*

Gender does not impact the churn rate:

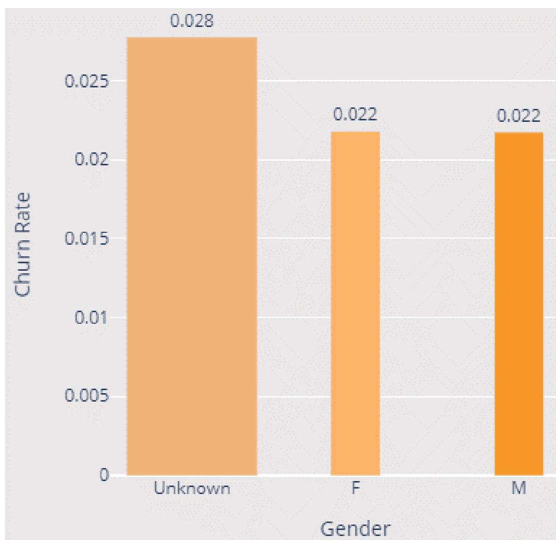



*Figure 20 - Churn rate by gender*

English customers seem to be less likely to churn out, but little data records about them might also cause that:

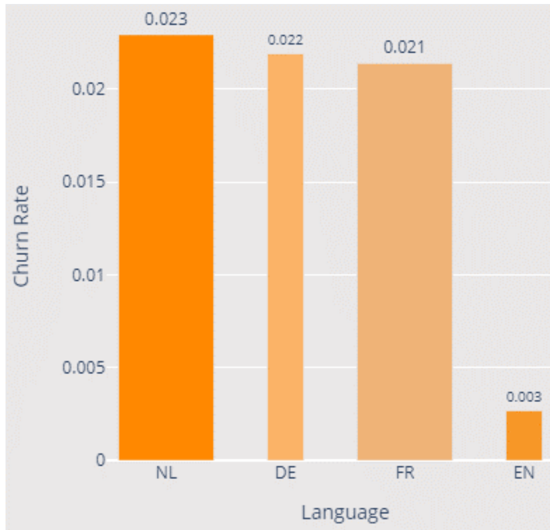



*Figure 21 - Churn rate by customer language*

SOHO market is slightly more keen to churn:


*Figure 22 - Churn rate by market segment*

Lack of *trf_mdl_phonedeal_cd* is more likely to churn:

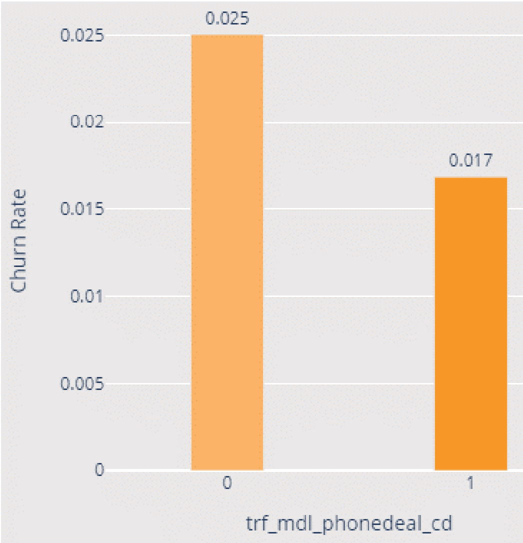



*Figure 23 - Churn rate by trf_mdl_phonedeal_cd*

Count orange higher than 1 is much more likely to churn. The value 6 is possibly an outlier of a single example in the dataset:

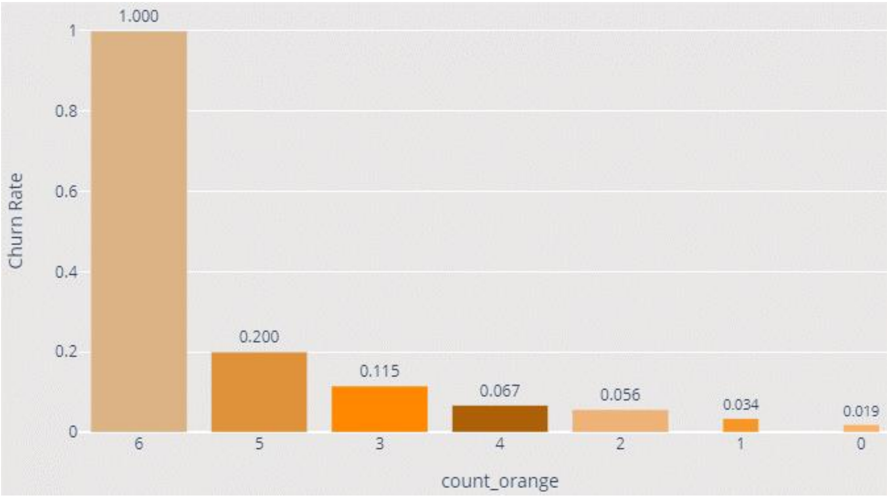



*Figure 24 - Churn rate by count_orange*

### 5.1.2. Churn related scatter plots

Customers with *mean_bill_rev_vs_trf_plan* of value closest to 1 are much more likely to churn:

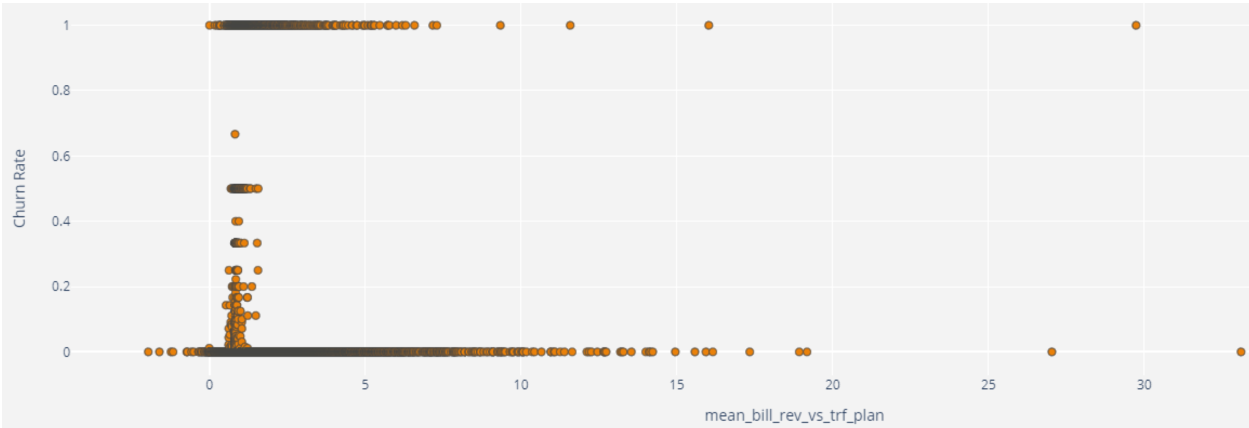



*Figure 25 - Relation between mean_bill_rev_vs_trf_plan & churn rate*

The higher *nb_cont*, the higher the likelihood of customer churn:

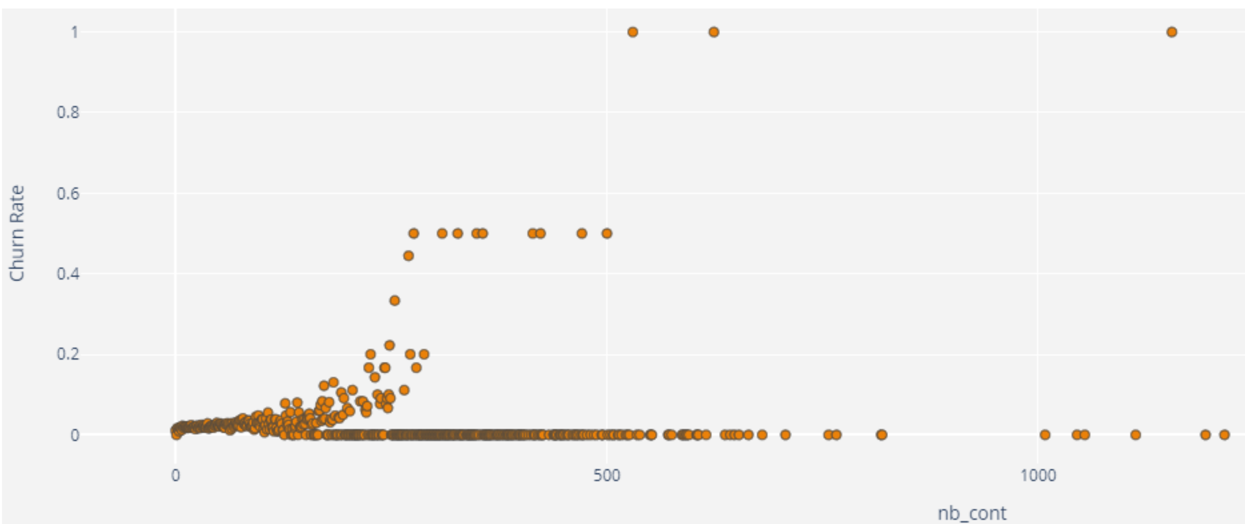



*Figure 26 - Relation between nb_cont & churn rate*

Churn_rate forms the best relationship with *avg_tp_churn*:

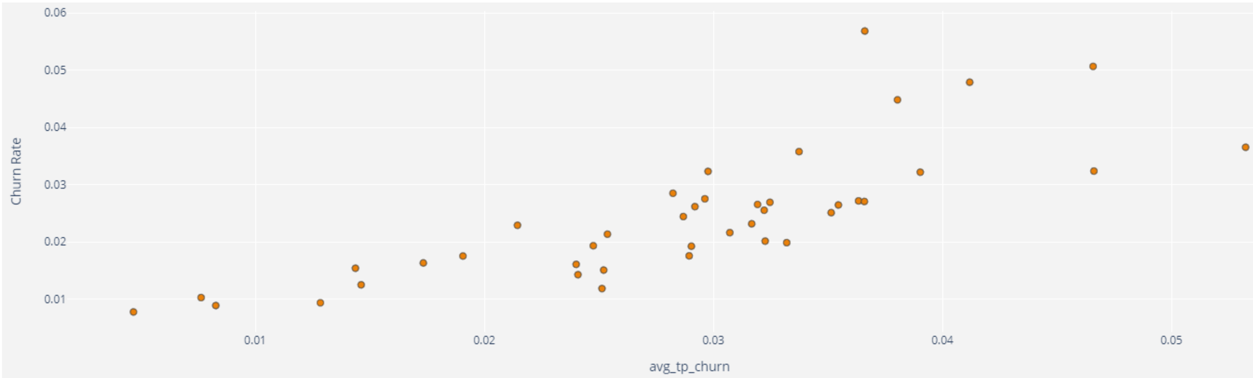



*Figure 27 - Relation between avg_tp_churn & churn rate*

## 5.2. Model predictions

Out of the 6 tested machine learning models, 5 of them score almost identical with the simplest decision tree being slightly less accurate. Generally, the accuracy scores are very high:

| **Model** | **Accuracy** (in %) |
|---|---|
| Support Vector Machine | 97.85 |
| Random Forest | 97.85 |
| Logistic Regression | 97.84 |
| K-Nearest Neighbor | 97.83 |
| XGBoost Classifier | 97.78 |
| Decision Tree | 95.40 |

*Table 3 - Comparison of machine learning model performances*

As an example, the logistic regression model classified 29340 examples correctly and 648 incorrectly, which is visible on the following confusion matrix:

```
array([[29340,     3],
       [  645,     0]])
```

*Figure 28 - Confusion matrix for the logistic regression model*

After testing the models, XGBoost library was used to generate the gradient boosting decision tree. The tree helps with a determination of characteristics impacting the target variable. In order to make meaning out of the leaf scores, one has to use the logistic function to convert the raw score for class 1 (leaf value) to a probability score. The example conversion is visible in the code.

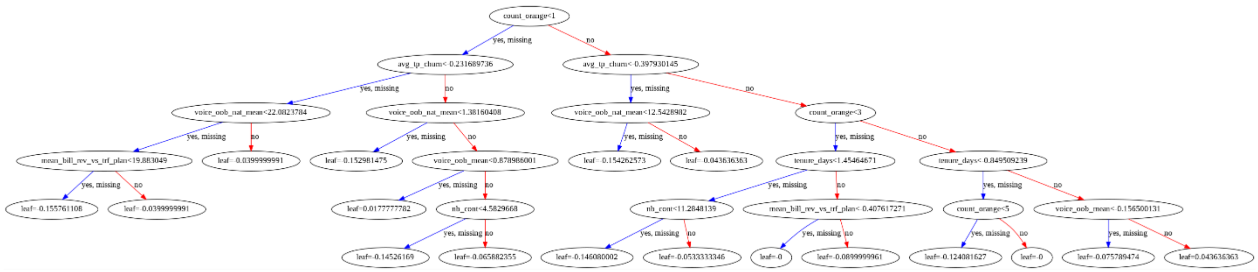



*Figure 29 - Gradient Boosting Decision Tree*

Furthermore, XGBoost was used to generate the feature importance comparison to determine that tenure_days is by far the most important feature used to determine the customer churn:

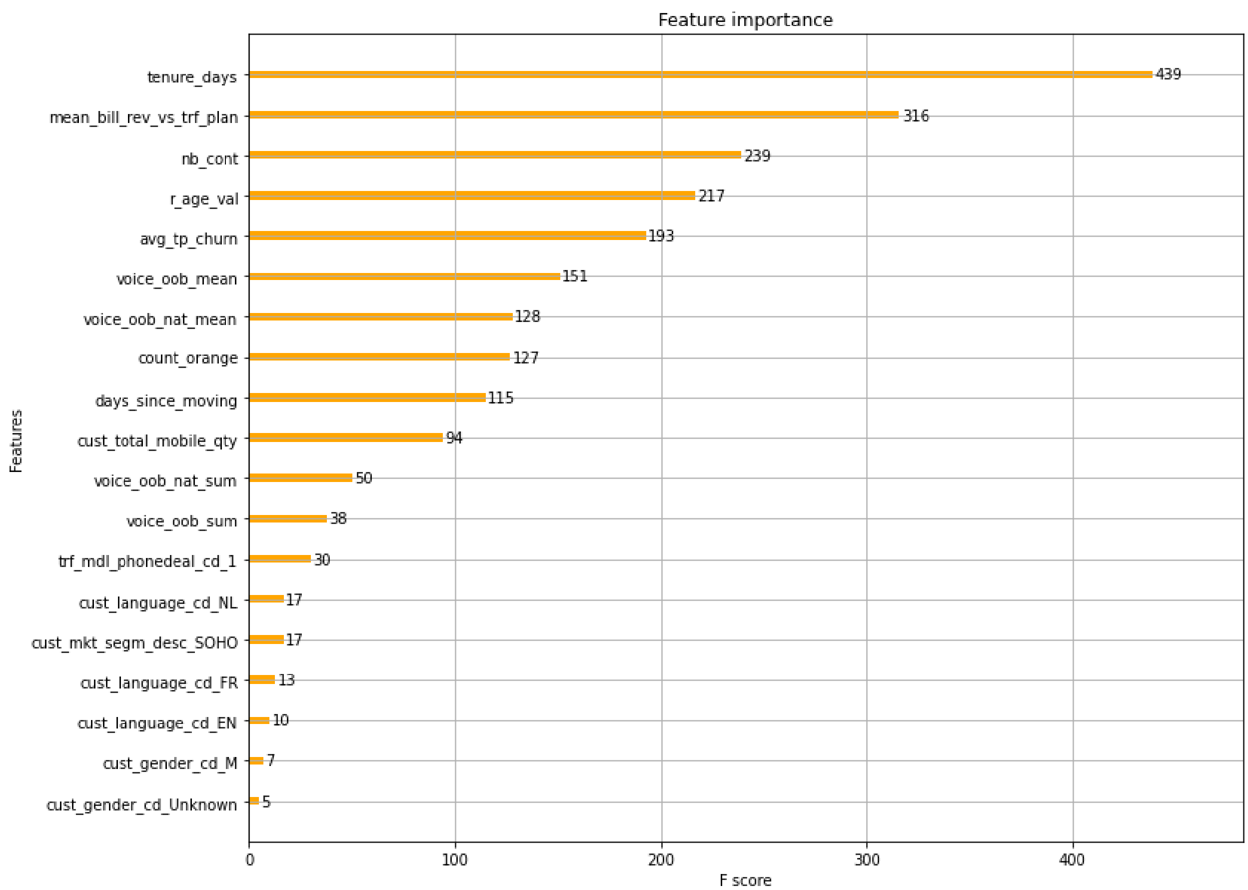



*Figure 30 - Feature importance*

Lastly, all the predictions were used to assign churn probabilities to 10 000 customers in the testing set:

| primary_key | churn_prediction |
|---|---|
| 107360 | 92% |
| 108035 | 56% |
| 108837 | 22% |
| 102438 | 20% |
| 101076 | 20% |
| 104897 | 20% |
| 105364 | 15% |
| 109419 | 14% |
| 101297 | 14% |
| 102882 | 14% |
| 102109 | 14% |
| 104575 | 14% |
| 101941 | 13% |
| 102614 | 13% |
| 104742 | 13% |
| 106070 | 13% |
| 104849 | 12% |
| 106736 | 12% |
| 105177 | 12% |
| 107521 | 12% |
| 107685 | 12% |
| 100340 | 11% |
| 101493 | 11% |
| 107112 | 11% |
| 100720 | 11% |

*Table 4 - 25 customers most likely to churn out of 10 000 predicted*

# 6. Discussions and conclusions

## 6.1. Analytics approach

The selected analytics approach:

- Indicated no performance issues with the technological stack
- Should not drive better results with other machine learning models since there is a lack of clear churn indicators
- Could have dropped eventual outliers from the datasets
- Could have implemented eventual feature engineering
- Could have extended the dataset with data from other organisations
- Could have augmented part of data to generate more churned examples.

Alternatively, one could try to implement the scenario in R or MATLAB, but it should not affect the results.

## 6.2. Business value

The analysis was able to deliver the following conclusions:

- Overall, the churn rate is not dramatic (only 2% out of 100 000 customers)
- Out of 10 000 customers to score, only 35 (0.3%) were predicted with a churn likelihood > 10%, out of which 3 had higher than 20%. Therefore; there is high importance to increase focus on these 35 customers
- Tenure days shall be further concerned as the best predictor of a customer churn
- Orange shall increase focus on customers with:
  - *count_orange* higher than 1
  - *mean_bill_rev_vs_trf_plan* close to 1
  - higher *nb_cont*
- To come up with more conclusions, the dataset needs to be extended by other features or more examples of *churned* customers.