Exploratory Data Analysis

We are given a dataset of shape 2572x55. Each column belongs to one of the following type groups: ID, Numeric, Categorical, Date, Ordinal, Binary. We now briefly explore these types.

The first one is ID. Among the 4 features belonging to this group, we find that country_id and product_id are constant. Thus, we discard them in preprocessing phase. The remaining two features, application_id and customer_id, contain 2571 unique value and we can consider them as unique key for the rows.

The second one is Numeric. The correlation matrix in notebook 1 shows high correlation for the couples of features: Variable_26 and Variable_17, Variable_27 and Variable_16, Variable_35 and Variable_23, Variable_37 and Variable_20, Variable_38 and Variable_21, Variable_39 and Variable_22.

The third one is Categorical. We encode these features in the preprocessing phase to be fed to the model. We store the dictionary storing the encoding at *"data/categorical_dictionary.pkl"*. The feature Variable_5 is constant and we drop it in the preprocessing phase. Variable_45 describe the sex. Seven entries in the dataset have the variable sex set to "?". As we ignore the source of data, we don't know whether "?" is a valid value (e.g. it comes from a form filled by users who don't want to declare their sex). However, as the frequency of this value is very low, we consider it as an outlier, and we discard it in the preprocessing phase.

The fourth one is Date. In the notebook called 1_exploratory_data_analysis, we show the plots of the distribution of the date features. The features due_date, first_status_day_date, paid_date, arrived_date start to increase in June 2015. Their frequency augment until May 2016 and then they drop. As the feature first_status_time_of_day includes only one day, the information is contained in hour, minutes and seconds.

The fifth one is Ordinal. There are only two ordinal features. Variable_13 and Variable_14. The values of Variable_14 are the same as Variable_13 with the adding of 'RATINGSTUFE'. Thus, we discard Variable_14. We encode Variablle_13 adding manually the missing value 'J' and we store the dictionary with the encoded feature at *"data/ordinal_dictionary.pkl"*.

Finally, the only binary variable is the Target. The last 515 rows have NaN. We will predict on this part of the dataset and send the result to Ferratum. In the remaining 2057 rows, the positive ratio is 69.22%.

Preprocessing Phase

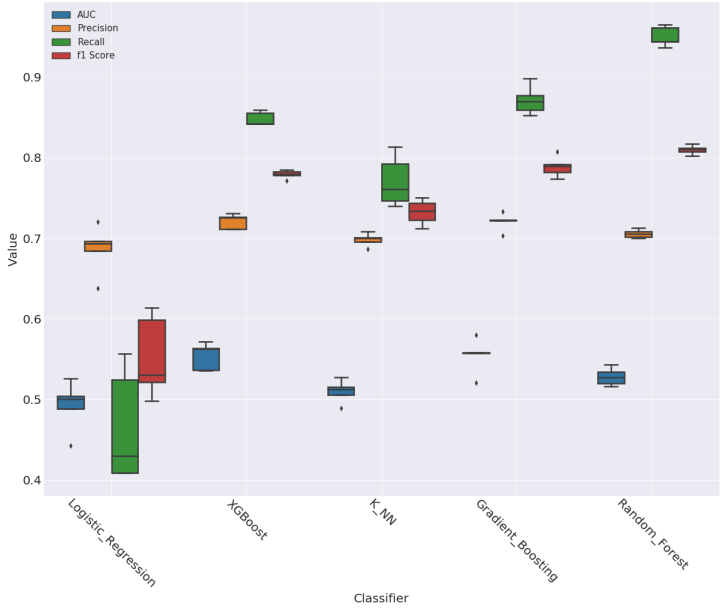
To prepare the dataset to be trained we:

- Transform Categorical and Ordinal features as said before
- Transform dates into integer using the values: year, month, day, hour and minute
- Drop too empty columns
- Impute to the NaN values the mean of the column of which they belong

Modelling Phase

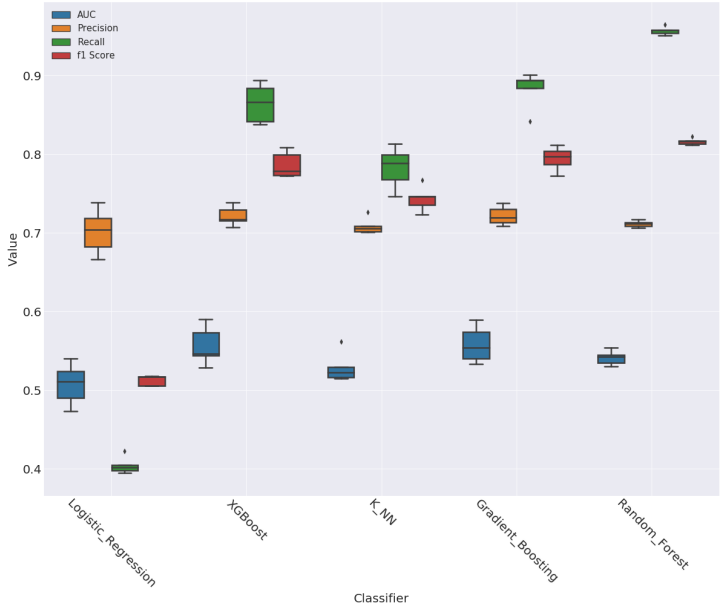
As the target assumes only two values, we are facing a classification problem. We chose which model suits more the problem among the most popular ones: Logistic Regression, K-nn, Random Forest, Gradient Boost and its smarter implementation XGBoost.

For each model, we split in five different ways the original set in training set and test set and then we compute aggregated values for the performance. We check Precision, Recall, F1-Score and AUC. When we test the five models with basic settings, we obtain the following results.



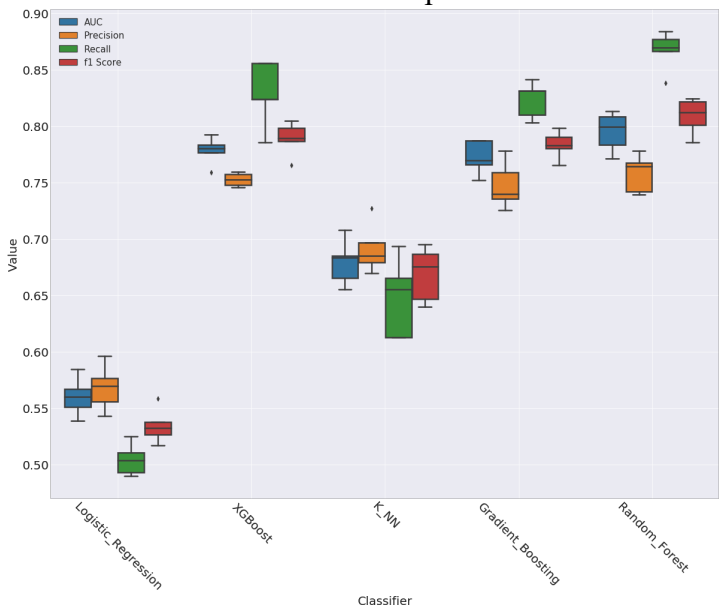
The precision is around 70% for every model but recall varies a lot with models. Looking at the AUC, we realize that the model is outputting almost random prediction. We now focus on AUC and try to improve it.

As we don't know business meaning of the data, it's not possible to perform accurate feature engineering. However, we compute the difference in days between every couple of date features and we add 27 new features. We train again the five models and we obtain the following results:



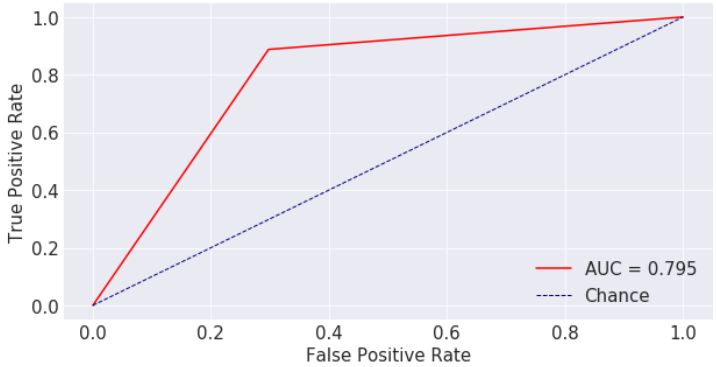
Apart from the K-nn model, we can't observe any improvement in the AUC. In order to improve the performance, we now try to oversample the training set. We use the SMOTE technique and we oversample the class which has less examples, that is 1.

The results of the models trained on the oversampled dataset are in the following graph.



The oversampling improved significantly the AUC of the models. In particular the median of the Random Forest's AUC is around 80%.

As Random Forests appears to be the model that performs better, we perform a grid search on a validation set to find its best parameter. You can see the procedure in the notebook called 5_random_forest. We train the model with the optimized parameters and we obtain the following performance



We perform a 5-fold cross validation that confirms this AUC (see it in notebook 6_final_training).
To conclude we store the result of the prediction at data/probability_class_1.csv.