MSc in Data Science

Leveraging Machine Learning Techniques for Supervised Fraud Detection in Online Bookings

Anastasis D. Boufis

Athens, October 2017
MSc in Data Science

Leveraging Machine Learning Techniques for Supervised Fraud Detection in Online Bookings

Anastasis D. Boufis

<table>
<thead>
<tr>
<th>Academic Supervisor:</th>
<th>Prof. Dimitris Karlis, Department of Statistics, AUEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Supervisor:</td>
<td>Evie Leon, Product Owner, Fraud and Payment Solutions, Tripsta S.A.</td>
</tr>
</tbody>
</table>

Athens, October 2017
Opinions and conclusions included in this document express the writer and must not be interpreted as a representation of Athens University of Economics and Business nor Tripsta S.A. formal views.
1 Abstract

The effectiveness of the online transactions and the convenience which they offer are simultaneously, their sensitive points. Being an online merchant always entails the risk that malicious users try to take advantage of the Card Not Present business setting in order to buy, using stolen cards or false identities.

This thesis discusses the issue of Fraud Detection in massive online services, with the use of Machine Learning algorithms and techniques. The study was accomplished in collaboration with Tripsta S.A., one of the leading Greek online touring operators. The company provided booking data from the past two years, which were already labeled as Frauds or No Frauds. The goal of the thesis is to suggest an effective algorithm/system for labeling new bookings as Frauds or No Frauds, in real time, based on the bookings’ characteristics and features. Statistical analysis was used for analyzing the current counter fraud system the company uses and for selecting variables with statistical significant differences between the two categories. Various Feature Selection algorithms and techniques were tested. For classification, various Supervised Machine Learning algorithms were tried, along with different Validation techniques, in order to obtain the optimal combination. For the Evaluation, various metrics were used.

The thesis is organized into six chapters. The first chapter summarizes some key elements from theory. Second chapter describes the analysis of the current counter fraud system which is used by Tripsta. Third chapter presents a statistical analysis for the features of the bookings and an attempt to find all these features that are distinguished among fraudulent bookings. The fourth chapter is an algorithmic analysis based on a small sample of the whole dataset. The fifth chapter describes the implementation of the optimal algorithms on the whole dataset. The sixth chapter presents the conclusions of the research.

Keywords: Machine Learning, Supervised Learning, Classification, Naive Bayes, AdaBoost, SMOTE, Stratified KFold, Fraud Detection, imbalanced Data
2 Acknowledgements

For the preparation of the current thesis, as well as, for the completion of my postgraduate studies, I feel the need to thank all the people who helped me.

I would like to express my deepest gratitude to Mr. Vasilis Vassalos, Associate Professor at AUEB and Director of the Masters Program in Data Science, for giving me the opportunity to attend the MSc Program and for his guidance during the past year.

I would also like to thank the academic supervisor of this thesis, Mr. Dimitris Karlis, Professor at AUEB for his advices and his assistance, anytime I needed it.

From Tripsta SA, I would like to thank all the people who welcomed me and made me feel comfortable from the beginning up to the last day. More specifically, I would like to express my sincere gratefulness to Evie Leon, Product Owner, Fraud and Payment Solutions and Alexandros Papapostolou, Software Engineer, for their support from day one and for their willingness to answer even my most naive questions. I would also like to thank Vasilis Kalogirou and Kostis Alexandris, developers at Fraud and Payment Solutions department, for their technical assistance and for making me feel as part of the team. Also, a very special thanks goes to Kostas Koukountzis.

I also feel the need to mention and thank, for various reasons, some other people. Aziz Mousas, for the inspiration and for being there every time I needed him during the many years I know him. Panagiotis Karmiris, for his patience and his support. Spyros Mantzouratos, for the accurate advice and for his concern. Finally, I wish to thank my mother, Niki, for her support and faith in me.

To K.
Ad astra per aspera
Table of Contents

1 Abstract 5
2 Acknowledgements 7
3 Introduction 11
4 Elements of Theory 13
  4.1 Feature Selection/Transformation 13
    4.1.1 SelectKBest 13
    4.1.2 PCA 13
    4.1.3 Scaling 14
    4.1.4 SMOTE 14
  4.2 Algorithms 14
    4.2.1 Supervised vs Unsupervised Learning 15
    4.2.2 Naive Bayes 15
    4.2.3 SVM (Support Vector Machines) 16
    4.2.4 Logistic Regression 17
    4.2.5 AdaBoost (Adaptive Boosting) 18
    4.2.6 Stacking 18
  4.3 Validation 18
  4.4 Evaluation 19
    4.4.1 Confusion Matrix 19
    4.4.2 Accuracy 19
    4.4.3 Precision 20
    4.4.4 Recall 20
    4.4.5 F1 Score 20
5 Evaluating the Counter Fraud System in use (FraudBuster) 21
  5.1 Correlation between Score and Category 21
  5.2 Computer Part 22
  5.3 Agents Part 23
  5.4 Modelling the 2nd Part through Machine Learning (Decision Trees) 24
  5.5 Final Conclusions 25
6 Feature Selection and Transformation 27
  6.1 Booking Variables 27
  6.2 Payments 29
  6.3 External Sources 29
  6.4 Rules 30
  6.5 Variable List 30
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Algorithm Analysis</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>Implementing the Algorithm</td>
<td>37</td>
</tr>
<tr>
<td>9</td>
<td>Conclusions - Final Thoughts</td>
<td>41</td>
</tr>
<tr>
<td>10</td>
<td>Bibliography</td>
<td>43</td>
</tr>
</tbody>
</table>
3 Introduction

The main goal of this project is to create a Machine Learning system, able to recognize fraudulent from non-fraudulent bookings, in real time.

As fraudulent, we define all those bookings which are made by the use of stolen credit cards, without the actual owner be aware of the loss. About 0.5% of each month’s bookings are labeled as Frauds by banks (i.e. they are actual frauds). This percentage might seem small, but it is an important problem for tour operators worldwide, with significant economic effects.

As it is already said, banks label each transaction as Fraud or not. This procedure might take up to 6 months. If a transaction is disputed, the tour operator can be sure that it is a Fraud. Until the bank confirmation, only assumptions can be made.

Since the status of a booking is not known beforehand, the most effective way to combat fraudulent behavior, is spotting problematic bookings on time and refusing the service. For this purpose, each major operator has special counter fraud departments, which primary job is to handpick suspect cases and manually check them. The selection is made by obvious (such as: number of failed transactions, the appearance of an account in a previous fraudulent booking etc.) and not so obvious (time to departure, extra services etc.) criteria.

What these departments try to do, is to model fraudulent bookings based on their previous experience. The purpose of this project, is accomplishing the same thing by using Machine Learning algorithms and techniques.

Machine Learning Process

The standard procedure for applying Machine Learning in a problem, can be described in the following graph.

![Machine Learning Process Diagram]

So, the first task is trying to recognize the problem and simplify it. The data have two main characteristics. First, their labels are known. We are aware of a booking status (Fraud or No Fraud), even if this procedure does not happen in real time. This information comes from banks. Secondly, we can observe that the vast majority of bookings are No Frauds. So, a simplified version of our problem is classification of imbalanced data.

After specifying the problem and formulating the questions, the next step is Feature Selection. The dataset that was provided to us, contained almost 150 variables (the flatten version contained more than 300). The simultaneous use of all features would probable make the model extremely complicated, extremely difficult to be trained and possibly,
prone to overfitting. A selection of variables that contain useful information is necessary. This could be accomplished initially, by the statistical analysis of the features, based on the category they belong (Fraud or NoFraud).

After the selection of the variables, that seem to have statistical significant differences between Frauds and NoFrauds (and a meaning i.e. seem possible to affect somehow the category), their correlation is checked. This is made possible through a series of specializing algorithms.

Data also need some kind of transformation, for two reasons. First, in order to be in a right form for being used as input to ML Algorithms. Second, because transformation is another method for data reduction. Again, this stage of Feature Selection, is made possible through the use of certain algorithms (not manually).

The next stage is the application of Machine Learning algorithms and the calibration of their parameters. The data in our disposal are labeled (i.e. we know by the bank if they are actually Fraud or NoFraud), so we will base our estimation/guess on this knowledge. In other words, we will use Supervised Learning algorithms.

Of course, ML algorithms cannot be applied without the use of proper Validation techniques, in order to split our data appropriate into training and testing subsets. In this section, will be taken into consideration the fact that the data are imbalanced and therefore, certain types of Validation are preferred over others.

The final step is the Evaluation of the results. Since the fraudulent cases are pretty rare, one may say that the easiest way to built a Fraud Detection System, is by always answering negatively (NoFraud). The provided data are imbalanced, so the main metric (Accuracy) is somehow useless because the majority (NoFraud) would probably be predicted right, most of the times.

So, other metrics have to be used. These metrics are Precision and Recall. They will be explained more thoroughly, in another part of this thesis, but the main idea behind them is the following. Recall measures the predicting power of the algorithm for certain things (for example, how good is the algorithm in predicting Frauds), while Precision measures the certainty over the results. Also, those metrics would provide us with signs of possible overfitting (due to the rarity of positive cases, it is possible that the algorithms might memorize certain patterns, instead of being trained).

We must also bear in mind that the whole procedure is, somehow, cyclic. This means that if the metrics are not adequate enough, we might return to calibrating the parameters or, even, going further back, in Feature Selection stage. Features that contain information might somehow, been missed. So, the idea is to repeat this cycle, as many times needed, by using different combinations of features/validation/algorithms/parameters, in order to find the optimal one. That's why Machine Learning application might be described as a form of art. Tuning or Feature Selection needs experience, in order to be performed correctly.
4 Elements of Theory

In this part, we are going to describe briefly the algorithms and the methods that were used.

4.1 Feature Selection/Transformation

4.1.1 SelectKBest

SelectKBest places variables/features of the dataset in a hierarchical order based on some type of score. Score results from various statistical tests (ANOVA, chi-squared, F-scores, family-wise error etc.).

In our case, ANOVA was used (through the parameter f_classif). ANOVA (Analysis of Variance) is a statistical test which can locate differences between the means of various categories and check the statistical significance of these differences. The idea is similar to the t-test. The difference in ANOVA is the application in multiple categories.

We have also tried to use Mutual Information (mutual_info_classif) as a metric, believing it will perform better with categorical variables, in comparison to ANOVA. We got exactly the same results, but the application was also significantly slower.

4.1.2 PCA

PCA (Principal Component Analysis) is a method for transforming the data. PCA changes the axis/reference system with a new one, whose first dimension is the direction where the largest variability in the data can be observed (second dimension is the direction where the second largest variability can be observed etc). Maximum number of new dimensions is the number of the old ones. The following plot is pretty explanatory.

At this point, we must also note, that large variability means that the data hide information which can somehow be exploited (another way to express it is by entropy).
4.1.3 Scaling

Scaling is a method for normalizing the data. What it does, is changing the scale of various features (usually between 0 and 1), so that they become comparable. Otherwise, there is a chance that features with large values might overcome others, equally important, but with smaller values.

Scaling is useful when variables are used simultaneously by the algorithm (for example SVM etc). In the event that the variables are used one after another (for example with Decision Trees) scaling will not make a difference.

The most common function for scaling is MinMaxScaler:
\[
X_{\text{std}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}
\]
\[
X_{\text{scaled}} = X_{\text{std}} \times (\text{max} - \text{min}) + \text{min}
\]

4.1.4 SMOTE

SMOTE (Synthetic Minority Oversampling Technique) is a method for oversampling the sample.

One way to deal with the problem of imbalanced data is to use oversampling. In other words, to use samples where we have artificially created some minority datapoints, in order to achieve some balance.

SMOTE achieves oversampling by the use of kNearestNeighbors algorithm. Based on a probability, some neighbors, which belong to majority, of a minority datapoint are chosen and, again, based on a probability, are copied and assigned the label of minority.

4.2 Algorithms

Let’s imagine a two-dimensional scatterplot, where the data of each category are marked with different color and shape
What a Machine Learning Algorithm tries to do, is to locate the boundary between the two categories.

4.2.1 Supervised vs Unsupervised Learning
The basic difference between Supervised and Unsupervised Learning lies on the knowledge of the categories/labels in which our data belong. In our case, we have two categories (Fraud, NoFraud), so we are going to use Supervised Learning.

4.2.2 Naive Bayes
Naive Bayes algorithms are based on the application of Bayes Theorem, supposing that observations are independent. Supposing that y are classes/categories and x are vectors of features/variables, we get from Bayes Theorem
\[ P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \]

Using the hypothesis that the variables are independent:

\[ P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid y), \]

and then, for all i:

\[ P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \]

But, for certain data, denominator remains stable:

\[ P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \]

\[ \arg \max_y \quad P(y) \prod_{i=1}^{n} P(x_i \mid y), \]

where a decision rule is used in order to get the most probable class/label for a particular datapoint (P(y) is the relative frequency of class y in our data. Product comes from the relative frequencies of various features, given a particular class/label)

Naive Bayes work very well as classifiers, but no so good as estimators. In other words, they can predict/locate the category in which an observation belongs, but they are not so good in estimating the probability for an observation to belong to a certain category.

### 4.2.3 SVM (Support Vector Machines)

SVM is a family of algorithms which can be applied on linear separable data. The main idea in this algorithm is maximizing the margin between the boundary and some of the data points, which are called Support Vectors.
SVM are particularly effective with multi-dimensional spaces. Also, due to the fact that only some of the data points are used in the process of estimating the boundary, they are particularly memory effective.

SVMs are used with linear separable data. If we want to use them with other cases, we must change the kernel function and, most important, we must be tolerant towards some cases of misclassification by the algorithm.

### 4.2.4 Logistic Regression

The way Logistic Regression works, is the following: Suppose $y$ is class/category and $x$ are vectors of variables/features. We want to create a relationship which connects linearly category with variables, through some weights

$$y(x, w) = w_0 + w_1 x, \quad w = (w_0, w_1)$$

Then, we pass the function above, through a new function, which is called sigmoid and gives as result 1 or 0.

$$p(t = 1|x, w) = \frac{1}{1 + e^{-y(x, w)}} = \frac{1}{1 + e^{-w_0 - w_1 x}}$$

In practice, a datapoint would belong to class 1 or to class 0 if the following apply:

$$p(t = 1|x, w) \geq 0.5 \quad \Rightarrow \quad t^* = 1$$

$$p(t = 1|x, w) < 0.5 \quad \Rightarrow \quad t^* = 0$$

or, equally:
Likelihood is given by the following formula:

\[ p(t|X, w) = \prod_{n=1}^{N} p(t_n|x_n, w) \]

\[ = \prod_{i=1}^{N} \sigma(w^T x_n)^{t_n} (1 - \sigma(w^T x_n))^{1-t_n} \]

(note the sigmoid function, with variables and weights).

In order to find the relationship, which connects categories with data, we want to calculate the weights. This will happen by maximizing the Likelihood.

The main problem is that Likelihood cannot be calculated analytically. We must use numerical methods.

We already know that log-Likelihood converges. So, instead of trying to find the maximum of log-Likelihood, we can find the points where its partial derivatives become 0. By finding these points, we can update the weights and continue the circle until partial derivatives become 0. This method is known as gradient ascent (or gradient descent).

### 4.2.5 AdaBoost (Adaptive Boosting)

AdaBoost belongs to a category of classifiers, which are called meta-classifiers or ensembles (they use a combination of other classifiers). To be more specific, the main idea is to apply a series of weak classifiers (classifiers whose predictions are slightly better than random guesses) on repeated, modified versions of the data. Modifications lie on changing the weights (initially, the weights are the same). After each iteration, weights of examples, which were predicted correctly, are reduced, while the weights of the examples which were predicted wrong are increased (in order to draw more attention). So, every next algorithm is obliged to focus on harder cases/missed examples.

### 4.2.6 Stacking

Stacking is a technique for combining different ML algorithms. The idea of stacking is to create different models (usually of different types - these models are called primary), find the results and then create a supervisor model, which learn how to combine the results of the primary models in the best, possible way.

### 4.3 Validation

Every dataset which is going to be used, will be split into 3 subsets:

- Training
- Validation
● Evaluation

Training dataset includes the majority of the data points and it is going to be used for training. Validation dataset is going to be used for calibrating and validating the algorithms.

Because validation subset is also used for calibrating the parameters of the algorithms (i.e. it affects the actual form of the algorithm) and not for validation reasons only, we want to keep a slice of data, which won’t be used during training at all, for testing the final result.

So, the method is the following. Initially, separating a small part of our data for the final test. Then, splitting the remaining data into training and validating (or, training and testing), in order to train and evaluate our algorithms.

**Stratified KFold**

There is always the possibility, data been placed in some type of hierarchical order. This fact might affect the way the data are splitted into training and validating subsets (might have more data from a particular category in one subset than in the other). To avoid something like this, it is recommended to split the data into k partitions, to run the experiment k times and to average the results of the experiments. This technique is known as KFold.

But there is also another problem with our data. They are imbalanced (the number of NoFrauds is much higher than the number of Frauds). One way to deal with this problem is to keep the percentage of each class steady in each partition. This technique is known as Stratified KFold.

### 4.4 Evaluation

#### 4.4.1 Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>Actual Positives</th>
<th>Actual Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Estimated Positives</strong></td>
<td>True Positives</td>
<td>False Positives</td>
</tr>
<tr>
<td><strong>Estimated Negatives</strong></td>
<td>False Negatives</td>
<td>True Negatives</td>
</tr>
</tbody>
</table>

#### 4.4.2 Accuracy

The first metric that is used, when evaluating an algorithm, is Accuracy. It expresses the general predictive power of the algorithm and it is calculated by the following relationship:

\[
\text{Accuracy} = \frac{(\text{true positives} + \text{true negatives})}{(\text{true positives} + \text{false positives} + \text{true negatives} + \text{false negatives})}
\]

The use of Accuracy with imbalanced data might not be particularly helpful. The existence of many data points from one class, increases the number of correct predictions.
It is a little bit tricky and counter-intuitive, since we (as humans) understand the relationship that occurs between Frauds and NoFrauds (one derives from each other - if a booking is Fraud it cannot be NoFraud and vice versa). The algorithm works a little bit different. It cannot detect or understand this relationship. It just needs to know what values must be identified as "good" (true positives) and then it will try to identify this values

**4.4.3 Precision**

As Precision, we define the following metric:

\[
\text{Precision} = \frac{(\text{true positives})}{(\text{true positives} + \text{false positives})}
\]

We may notice that Precision is connected inversely with the number of false positives. The greater the number of false positives, the smaller the score becomes. Precision expresses the certainty of an algorithm i.e. how certain we are that a result which was predicted as positive, is actual positive (in our case, if we predict a Fraud, this is actually a Fraud)

**4.4.4 Recall**

Recall is calculated by the following formula:

\[
\text{Recall} = \frac{(\text{true positives})}{(\text{true positives} + \text{false negatives})}
\]

Recall is reversely connected to the number of false negatives. It expresses the ability of the algorithm to identify the positive cases (in our case, to identify Frauds). As the score increases, we become more confident, that if a Fraud is occurred, our algorithm will track it.

**4.4.5 F1 Score**

F1 Score is given by the formula:

\[
\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})}
\]

It is a relationship between Precision and Recall. The higher this metric is, the better our algorithm is, in general.
5 Evaluating the Counter Fraud System in use (FraudBuster)

The current system, which is used by Tripsta, is comprised of two parts. The first part, which is conducted by computers, assigns a score to each booking. This score is based on certain parameters and rules (the cost of the booking, if it contains a high risk route etc.). In theory, the height of a score corresponds to the probability that the booking is fraudulent. Second part of the system is about bookings with high score (high risk), which are forwarded to people (agents) and manually checked for extrapolation and for drawing final conclusions.

5.1 Correlation between Score and Category

A sample of 10000 bookings from April of 2016 is going to be used. This sample contains 9947 “NoFraud”, 50 “Fraud” and 3 “Friendly Fraud” bookings. The first thing, is trying to observe the distribution of score.

The greatest peak can be observed at 0 (most bookings have 0 score). We can also observe a high concentration of bookings around 0. A second, smaller peak may be observed at -6. Finally, it is worth mentioning, that the majority of bookings, have a score below 5.

The next step, is the analysis of score among each category (Fraud or NoFraud).

For the “Fraud” category, the following can be observed:
<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.50</td>
<td>13.81</td>
<td>20.38</td>
<td>19.49</td>
<td>23.50</td>
<td>36.75</td>
</tr>
</tbody>
</table>

For “NoFraud”:

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
<th>NA's</th>
</tr>
</thead>
<tbody>
<tr>
<td>-15.5000</td>
<td>-2.0000</td>
<td>0.0000</td>
<td>-0.3635</td>
<td>2.0000</td>
<td>22.2500</td>
<td>25</td>
</tr>
</tbody>
</table>

and the corresponding boxplot:

It might be noticed that the majority of “NoFraud” have relatively low scores. A high number of outliers might also be noticed, which is pretty logical, if we take into consideration the big sample (10000 observations/bookings).

The plot of the fraudulent bookings is more concentrated (we have only 50 fraudulent bookings in our sample). Also, the plot of the fraudulent bookings is located higher on the plot, in comparison to the one of the no fraudulent ones. In other words, we may notice a significant difference between the score of each category (*the significance is further proved by statistical tests).

5.2 Computer Part

The evaluation of each booking is based on certain rules. The data for these rules might come from the booking itself (for example if the booking name is shown in the passenger section, if more than one credit cards had been used etc.) or by third party information.
(email evaluation, credit card evaluation etc.). The score for each booking is produced by the number of the rules, which had be broken. If score is high (above 3.5), then, the booking is forwarded to human agents for further investigation. After a period of time (6 months at most), bank confirms whether a booking is fraudulent or not. The system has also by itself, the ability to block bookings which contain parameters connected to or characterized as Hot List factors (emails of credit cards used in fraudulent bookings).

The data at my disposal came from April 2016. In total, there were 152437 bookings. 20473 of them were marked as High Risk. From the High Risk bookings, we could identify 533 Frauds. We were also able to locate 10 bookings which were Frauds without been marked as High Risk.

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive</th>
<th>Actual Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Estimated Positive</strong></td>
<td>533</td>
<td>19940</td>
</tr>
<tr>
<td><strong>Estimated Negative</strong></td>
<td>10</td>
<td>131954</td>
</tr>
</tbody>
</table>

Based on the above, we were able to extract some remarks for the accuracy of the system. It must be noted here, that the data were split unevenly between the two categories (imbalanced data). In such cases, using just the accuracy as a metric, might be pretty meaningless (since there are many data in one category and very few in the other, accuracy would be very high). So, we are going to use to use two other metrics: Precision and Recall.

**Accuracy** = \( \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} = \frac{533 + 131954}{533 + 131954 + 19940 + 100} = 0,869 \approx 87\% \)

**Precision** = \( \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{533}{533 + 19940} = 0.026 \approx 2.6\% \)

**Recall** = \( \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \frac{533}{533 + 10} = 0.982 \approx 98\% \)

It can be observed that the predictive power of the algorithm in general (Accuracy) is pretty high (although there is always room for improvement). It’s ability to detect Frauds is very high (Recall 98%), something expected due to the large number of cases which marked as possible Frauds. On the other hand, the certainty that each case which is marked as Fraud is actually a Fraud is pretty low (Precision 2.6%), again, due to the large number of cases which are labeled as possible Frauds.

**5.3 Agents Part**

High Risk bookings are forwarded to human controllers/agents for the final decision, to keep or to reject the booking. There were 20473 High Risk bookings in April 2016. Of these, 17602 were kept, the ones which were checked by the current agents.
All the bookings that were rejected, were marked with the status cancelledByAgent. Although there might be other reasons for cancellation via agent (e.g. someone made a phone cancellation), we can surely say that the largest proportion of the cancelled bookings, is due to possible Frauds.
Subset High Risk bookings which were checked by certain agents, had 17602 bookings for April 2016. Bookings that were cancelled and proved to be Frauds were 98 (True Positives). Bookings that weren’t cancelled and weren’t Frauds were 16766 (True Negatives). Bookings that were not cancelled but proved Frauds were 0 (False Negatives). Finally, bookings that were cancelled without being Frauds were 740 (False Positives).

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive</th>
<th>Actual Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Positive</td>
<td>98</td>
<td>740</td>
</tr>
<tr>
<td>Estimated Negative</td>
<td>0</td>
<td>16766</td>
</tr>
</tbody>
</table>

So, the metrics become:

**Accuracy** = \( \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}} = \frac{98 + 16766}{98 + 16766 + 740 + 0} = 0.958 \approx 96\% 

**Precision** = \( \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{98}{98 + 740} = 0.117 \approx 11.7\% 

**Recall** = \( \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \frac{98}{98 + 0} = 1 \approx 100\% 

We can make the same remarks as in the previous part. That the system has a great predicting ability in general, it is able to recognize almost every Fraud, but it marks as Frauds cases which are not. The system has a great tolerance towards False Positives. It’s main focus is to detect every possible Fraud, even by marking as Frauds cases which are not.

We can also notice a gap between the False Positives of the first and second part. These are bookings which were cancelled due to Hot Listed Factors (e.g. emails which had been used for fraudulent transactions in the past etc).

### 5.4 Modelling the 2nd Part through Machine Learning (Decision Trees)

An initial Machine Learning approach was used in order to model the second part of the system. Feature “category” was used as label (possible values: “Fraud”, “NoFraud”, “Friendly Fraud”). Due to the fact that the data are labeled, we are able to use Supervised Machine Learning. It is known that the selection in the first part was according to rules and that the results of these rules are stored as ruleResults for every booking, so, we believe that the best choice is a Decision Tree.
Initially, 10000 bookings were chosen from the High Risk subset.

The next part was Feature Selection. We used ruleResults Features (if a rule is applied or not for that particular booking). Initially, we excluded Features with high percentage of missing values (in a second stage we might use the existence of a value as a variable, in other words if a booking has a value for a certain Feature or not. In fact, we did exactly this when we applied Feature Selection on the whole dataset). But in this early stage, we simply reject those Features.

We turned True/False to 1/0 and dropped all the bookings with missing values. 6848 bookings were left. We have also checked for “Friendly Frauds”.

Decision Tree does not need any kind of normalization, so we inserted the data into the algorithm.

We have used two types of validation. Initially we separated our sample by using a simple train/test split. In this case, we had accuracy: 0.983, precision: 0.585, recall: 0.667.

Then, we used KFold. In this case, we got accuracy: 0.981, precision: 0.651, recall: 0.590 (we were able to notice that the algorithm’s ability to identify all Frauds was reduced (Recall), but the certainty for picking the right ones increased (Precision).

We tried to examine the degree in which each Feature affected the classifier (through feature_importances_). We can notice that the majority of Features affected the classifier into very small scale. An exception to this is the feature that describes if elements of the booking can be found in previous fraudulent cases, which affected the classifier at about 50%. Also, there were 5 features (NoDeviceIdGenerated, CustomerUseDiscountVoucher, ThxGlobalRuleDeviceBlacklistedTriggered, NoTrueIPDetected, HighRiskRoute) that didn’t seem to affect the classifier at all.

For testing, we chose 100 bookings from the High Risk pool. The results were accuracy: 0.986, precision: 0.5, recall: 1.0. The algorithm managed to identify all fraudulent bookings and we were 50% sure that if something is marked as Fraud, it is actually a Fraud.

5.5 Final Conclusions

Current system is pretty capable in identifying Frauds. If a Fraud occurs, it is going to be detected. This becomes possible by marking many bookings as possible Frauds. As a result, many bookings are marked falsely as Frauds, although they are not.

Finally, some proposals for the improvement of the current system:
- Increase the threshold above which a booking is characterized as High Risk.
- Finding the rules which affect score at most and increase their weights
- Abolishing Rule System. Starting to model bookings based on their Features. Unifying the two parts through Machine Learning
6 Feature Selection and Transformation

The dataset, which provided to us, included about 130 features. Many of them where datasets of their own. Feature “Rules” is an example. For every booking, this field includes a whole dataset. This dataset within a dataset has the rules as keys and boolean values (if a rule is applied or not). If we try to flatten the dataset (integrating the features of the individual/smaller datasets to the initial), we get, in total, about 340 features.

For Feature Selection, we checked the relationship of every feature with the feature “category” (“category” is the feature that reveals if a booking is actually a fraud or not and it is used as our label). The initial dataset was splitted into two subsets, based on “category” (Fraud or NoFraud). Then we compare the distribution/formulation of every feature in each subset.

The variability of each feature is going to be checked by using certain feature selection algorithms (kmeans etc), before the application of any Machine Learning algorithm. In this stage, we are trying to select variables which are meaningful and with obvious differences between Frauds and NoFrauds. We are also trying to exclude features which provide the same information or don’t give any type of information at all.

Except from selecting the variables which seem suitable, we are also trying to transform them. For two reasons. First, because we are trying to bring the variables in a form which is accepted by ML algorithms and secondly, because a transformation might help us extract the needed information.

6.1 Booking Variables

“_id” feature is going to be used as identifier. We are also going to keep “category” as labels (any type of classification will be based on this feature).

For feature “partnerName”, it was discovered that the percentage of missing values in Fraud subset is 70%. The percentage of missing values in the NoFraud subset is 20%. We decided to create a new feature (partnerName_exists) which shows if “partnerName” has value or not. At this point we must note that the handling of missing values is a very delicate procedure, because a missing value might be caused by many different reasons. In this case, due to the high frequency in the Fraud subset, we suspect that it might has to do with the way a fraudster tries to use the service.

Feature “wing” is the same as feature “market”. We are going to use only one of them as categorical. Then, we checked the feature “brand”. We discovered that the percentage of brand “tripsta” in Fraud subset is higher than the percentage in NoFraud (other brands represent other services which are offered by the company e.g. airtickets, travelplanet24). So, it seemed helpful to use this variable as categorical.

Feature “source” has one value (Api) and NA (missing values). Significant percentages of missing values (70%) were noticed in Fraud subset. We created a new variable (source_exists) which shows the presence or not, of value. “deviceSource” has many possible values (mobile, mobile-android, mobile-ios). Our initial thought was to use it as
categorical. Then, we observed the higher percentage of missing values in the Fraud subset (73%) than in the NoFraud (65%). We decided to create a new, categorical variable which shows if we have a missing value or not (deviceSource_ext).

From cost/price variables, we choose to use “totalPriceInEuro” (in order to have comparable results). We keep it as number (we might also use it as categorical, by placing prices in bins). We discovered that Frauds span a range from 500 to 1500 Euros. Bookings with prices lower than 500 were, in general, NoFrauds, as well as, bookings with high prices (outliers).

The dataset contains 4 airport variables. Departure and arrival airports for leaving (Outbound departure, outbound arrival) and for returning (Inbound departure, Inbound arrival). During the investigation, we noticed a large percentage of missing values at Inbound airports (about 55%) for the Fraud subset. In other words, 55% of Frauds are one way tickets (in NoFrauds, the percentage is 47%). We created a new, boolean feature (is_oneway) which says if a booking has appointed return or not (we created this variable by checking if Inbound Departure and Inbound Arrival are missing at the same time). After a discussion, we found out that this already variable exists and used, but in rule combinations, which were not available to us. Another thought was to create departure-arrival pairs and investigate for high risk routes. Such features are already part of the rules, so we thought it would be less time-consuming if we simply use them as they are.

“passengersAdult”, “passengersChild”, “passengersInfant” give numbers of passengers in a booking. We found out that most Frauds are bookings which include one or two persons. Bookings with more passengers are, in general, without a problem. We also thought it might be a good idea to check if the name of the person who is charged for the booking is included in the passenger list. This check is also included in the rule list (CreditCardHolderDoesNotMatchAnyPassengerName), so we are going to use as it is (as a boolean variable).

Feature “passengers” is a passenger list. It contains a dictionary for every passenger, with keys: name, date of birth, sex and if the passenger is frequent flyer or not. We were able to observe that most of the frauds are conducted by males. We created two new variables (no_of_males, no_of_females) which give the number of males and females per booking. We have also created a variable with the number of frequent flyers per booking (no_of_frequent_flyers).

“segment” section is about transfers. We thought that fraudsters might choose direct flights, so we created a new variable, which shows the number of segments per booking (no_of_segments).

“Inboundwebfare”, “Outboundwebfare” are features that have to do with low cost flights. Their possible values are True, False and NA. For both cases, we discovered a difference between the percentages of Trues among Frauds and NoFrauds. Two new, boolean variables, which show if the value is True or not, are going to be used (is_InboundWebfare_true, is_OutboundWebfare_true).

Then, we tried to deal with date variables. More specifically, with “bookingDateCreated” (when the booking was created), “depDateTimeOutbound” (date of departure) “depDateTimeInbound” (date of arrival - if it exists). From “bookingDateCreated”, four new variables were exported: Hour, Day, Month, Day_Of_Week (0 for Monday up to 6 for
Sunday). In the provided dataset, it was not possible to observe any pattern about Months or Days of Month (since the data came from one month only), but we manage to make some observations about Hours. More specifically, most of NoFrauds are booked in the morning (between 7 and 8). Most of Frauds occur after midnight (1 and 4 in the morning). Finally, we created two more numerical variables: time_to_departure_hours, which shows the time between booking and departure in hours and journey_length_hours (the length of the trip - where this doesn’t exist, we placed 0 instead).

Then, we checked variables related to user profile. We discovered that most Frauds (about 80%) are conducted by Males. We keep the feature “profileGender” as categorical. We are also going to keep as categoricals, the features “profileCity” and “CountryCode”.

### 6.2 Payments

The next bunch of variables has to do with payments. “paymentCardVendor” shows the vendor of the credit card which was used. The majority of the credits cards which are involved in Frauds, came from Visa and American Express. This variable is going to be used as categorical. “paymentType” shows the type of payment which was used (credit card, cash etc). Frauds pertain to credit cards. So, it is safe to say that when another mean of payment occurs, most probable we have a noFraud. A new feature is created (payment_with_cc) which tells us if a credit card was used on not. Something similar stands for “paymentNumberofInstallments”. Frauds have no installments (it is a single transaction). If there is not a number of installments, then we don’t have a Fraud. We could probably use this variable as boolean (if a transaction has installments or not), depending on the algorithm we are going to choose.

“paymentCurrency” seems to be a pretty important feature. We noticed that for the vast majority of transactions (97%), Euro is used as currency of choice. In Frauds, this percentage drops to 50%. We created a boolean variable which shows if a transaction uses Euro or not (is_currency_euro).

“creditCardTransactions” contains a list of all transactions (successful and unsuccessful) for the current booking. We will use it in order to extract new features. We are going to count the total number of transactions (no_of_transactions), the number of credit cards which were used and how many times a credit card number was copied (no_of_copied_cnn).

### 6.3 External Sources

Then, we want to deal with features that come from external sources. As regards TMX, the only feature on which we noticed differences between Frauds and NoFrauds, is TMX_policy_score. We will use it as numerical.

TMX also provides information about the actual locations of credit card and ip. An initial thought was cross examining these data with the locations declared by user. This information already exists in our dataset, in the form of rules. We decided, it is preferable to use the already existing data, than to create new.

From EA variables, we found two with differences between Frauds and NoFrauds (EAScore, ReasonID). We intended to use Score as numerical and Reason as categorical. But then, we noticed many missing values. Since there is no way to fill these missing
values (they come from an external service) we won’t be able to use Score. Reason is still going to be used as categorical, with the addition of “NA” category (we noticed different percentages of missing values between Frauds and NoFrauds).

As regards addon variables, we couldn’t identify differences between Frauds and NoFrauds. But we noticed something strange with the feature “addon.Parking”. In 10000 bookings, we observe 9600 False and 400 missing values. In the Fraud subset, we have 50 NAs (all the 50 Frauds have missing value in the “addon.Parking” variable). We are thinking that it might relates with the way someone uses the service. We decided to create a boolean variable which shows if “addon.Parking” has a value or not (res_parking).

6.4 Rules

Then, we targeted the features-rules. The general idea was not to use these variables at all (our goal is to create a system which is modelling the bookings based on their characteristics). We will keep only those rules, which can be extracted by our data, i.e. variables which we could create by ourselves. It seems fit, not only because it is less time consuming, but because it is more efficient in terms of uniformity.

The only exceptions, which use external information are a rule which reveals if any type of suspicious transaction has been detected (which is going to be used as categorical with true, false, NA categories, due to many missing values) and a rule which gives information for any type of problematic elements (i.e. elements that had been seen in previous fraudulent bookings) in the booking (as categorical - we suppose that every booking is checked for hot listed elements before it is entered in the system).

We have also included rules which contain information about crosschecking actual and declared locations (where we have many missing values, instead of boolean, we turn the variable into categorical with values true, false, NA). We tried to include rules which reveal if the user is going to travel in short time, from the time the booking created (we have already created a variable which shows the time between booking and departure, so this variable might not be needed). Finally, we have used some rules which refer to transactions. Note that the information which is offered by the above rules, could be produced by the feature characteristics. Rules have been used for time saving and uniformity.

6.5 Variable List

The final list of variables is the following:

```
_id
partnerName
brand
market
totalPriceInEuro
departureAirportOutbound
arrivalAirportOutbound
passengersAdult
passengersChild
passengersInfant
profileGender
```
profileCity
profileCountryCode
paymentCreditCardVendor
paymentType
TMX_policy_score
category
is_oneway
is_currency_euro
no_of_transactions
no_of_segments
Hour
Day
Day_Of_Week
Month
time_to_departure_hours
journey_length_hours
no_of_copied_cnn
no_of_males
no_of_females
no_of_frequent_fliers
source_exists
payment_with_cc
res_parking
is_InboundWebfare_true
is_OutboundWebfare_true
EAReasonID_category
partnerName_exists
DetectedSuspiciousTransaction_ext
deviceSource_ext
7 Algorithm Analysis

This section is focused on the analysis for finding the optimal algorithm/parameter combination.

The first thing was trying to identify the type of the data and the type of analysis we wish to perform. To put it simpler, trying to simplify the problem and choose the optimal strategy. Our data have two main characteristics: 1) they are labeled, 2) they are imbalanced (on the category we want to perform our analysis). Our target was to cope with each booking as a separate case. In this stage we were not interested in locating any connections/relationships between the bookings. Any piece of information which shows relationship/correlation/connection between a booking and fraudulent factors or previous fraudulent bookings, exists in an already existing variable ("connectedtohotlistedfactors").

If we want to simplify the problem of Fraud Detection, we can say that it is a classification problem with imbalanced, labeled data. So, our thought is to use Supervised Learning classifiers in combination with oversampling techniques.

For our analysis, we used a sample of 10000 bookings from April 2016. Before anything else, we separated a piece of our dataset (about 10%) and kept it for testing. Then, we started to try various algorithm/feature selection/validation compilations.

The first validation technique, suitable for use with imbalanced data, is Stratified KFold. What it does, is separating the dataset into k different subsamples and performing k different experiments. The final result derives from averaging the results of the individual experiments. The key characteristic of the subsamples, is that they contain a stable proportion of elements from each class (the proportion is the same as in the whole sample).

We began the analysis by keeping the type of validation stable and started to test various combinations of feature selection (selectkBest, PCA) and machine learning algorithms - classifiers and ensembles (Naive Bayes, Logistic Regression, AdaBoost, SVM, Random Forest). We tried to locate the best combination, in terms of accuracy, precision and recall. Except from trying to find the optimal combinations, we have also tried to calibrate the algorithms. In other words, we tried many different values for the parameters. Due to the large amount of boolean features, one of the ideas was using some type of Decision Tree.

We discovered that for Feature Selection, the best techniques are PCA with 14 variables (although, according to score, 10 are useful, we get better results by using 14) and selectKBest, including all variables/features and using ANOVA.

4 combinations emerged, which gave similar results, especially in the testing subset.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Feat Selection</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (c=100, kernel=rbf),</td>
<td>PCA(n_components = 14)</td>
<td>0.997</td>
<td>0.823</td>
<td>0.705</td>
<td>0.738</td>
</tr>
<tr>
<td>AdaBoost (n_estimators = 50,</td>
<td>SelectKBest(f_classif, k=&quot;all&quot;)</td>
<td>0.997</td>
<td>0.859</td>
<td>0.73</td>
<td>0.765</td>
</tr>
<tr>
<td>learning_rate = 1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multinomial Naive Bayes</td>
<td>SelectKBest(f_classif, k=&quot;all&quot;)</td>
<td>0.998</td>
<td>0.94</td>
<td>0.71</td>
<td>0.796</td>
</tr>
<tr>
<td>Logistic Regression(C=10000,</td>
<td>PCA(n_components = 14)</td>
<td>0.997</td>
<td>0.9</td>
<td>0.685</td>
<td>0.757</td>
</tr>
<tr>
<td>solver=’lbfgs’)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Testing dataset contains 3 Frauds. All the above combinations manage to identify 3 possible Frauds. 2 were correct guesses (true positives) and 1 was wrong (false positive). It also did not manage to recognize 1 Fraud (false negative).

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.997</td>
<td>0.667</td>
<td>0.667</td>
</tr>
</tbody>
</table>

In all the above cases, we can observe high Precision and relatively lower Recall. Recall reveals the ability of an algorithm to detect the cases that we wish to be detected (in our case, the ability to detect Frauds). High Recall means that if a Fraud occurs, it is almost certain that it will be detected. Precision expresses the certainty that something which is detected as positive (in our case a Fraud), is actually positive. High Precision means that the algorithm’s predictions are well assured.

The metric analysis of the above cases reveals that the fraud predictive power of the suggested algorithms is not so high (in other words, some fraudulent cases might be missed), but, on the other hand, every booking that is marked as Fraud has a greater chance to be such. The suggested algorithms seem to fix the main problem that we located with our initial analysis: the great number of false positives.

From the algorithms above, the most correct, from a numeric perspective, is Naive Bayes. After that, we would probably choose SVM and Adaboost (we must note here that Adaboost seems to be more balanced). We must also note that, according to Liu et al, Naive Bayes and Adaboost are somehow connected. Their main difference is, that Adaboost can be identified as a soft classifier (it locates the bookings in a hierarchy based on how probable each booking might be fraud), while Naive Bayes is a hard classifier (it simply separates the cases). The final choice between Naive Bayes and Adaboost depends on the needs of the company and their business model. A solution to this, which I
did not had the chance to test in depth, is stacking i.e. the combination of the results from the two models by using a new model (for example Regression) as a supervisor.

We have also tried various validation methods (Stratified Shuffle Split, KFold), but they produced worse results than StratifiedKFold on the same algorithms.

Finally, an oversampling technique (SMOTE) was tested, in combination with KFold. The idea was to oversample the subsamples which were produced by KFold. It might be better if we oversampled our initial sample and then apply KFold, but this would affect not only the training, but the testing/validating data, too.

Here, we found a pretty interesting case. Application of Logistic Regression, just as before, but with the use of oversampled subsamples (apply SMOTE also). The results after validation, are:

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.993</td>
<td>0.428</td>
<td>0.886</td>
<td>0.553</td>
</tr>
</tbody>
</table>

After trying the testing subset, we got the following results:

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.989</td>
<td>0.231</td>
<td>1.0</td>
</tr>
</tbody>
</table>

We discovered that the combination managed to predict all the 3 fraudulent bookings, but it predict as Frauds 10 more cases, which were not (false positives). This model reminds the current counter fraud system, which manages to identify almost all Frauds, but it wrongly predicts many legit bookings as Frauds, too. We kept it because we believe it is a good idea to use it as one of the base models in a stacking algorithm. The results from this model would probably draw the final result towards higher Recall (better predictions), the results from one of the above models (for example Naive Bayes) would draw towards higher Precision (more certainty with the predictions) and the supervisor model would modify the weights and make the final decision.

We must also note that we tried SMOTE combined with Stratified KFold in all the 4 suggested algorithms. The general conclusion is that it increases the number of false positives, without increasing the number of true positives, too.

As a general conclusion, we tried to cope with the following strategy. We kept a validation technique, which was recommended for imbalanced data (StratifiedKFold), as constant and then, we tried to find the optimal feature selection techniques/algorithms/parameters combinations. After picking the combinations, we also tried various validation techniques. We ended with 3 combinations. The first two, Multinomial Naive Bayes and Adaboost, work with the exact opposite way than the current counter fraud system (they try to be sure for their choice instead of marking many cases as possible frauds). The third, Logistic Regression with SMOTE, simulates the way the current system works (identifies all the frauds by marking many bookings as possible Frauds, resulting to many false positives/false frauds).
8 Implementing the Algorithm

After choosing some possible combinations, the next step is trying to check the algorithms’ performance and predicting ability towards the available data. The strategy remains the same. Keep some data for testing and split the remain in training and validating subsets. During validation, the need for some further calibration and optimization might rise. We might even discover or think some new features that are useful. But the core idea is to work with the combinations we picked at the previous steps and implement any possible change to these algorithms.

April 2016 dataset (the available data) contains more than 150000 bookings. The features that we are going to use are mainly, those that we have picked in the initial Feature Selection, with some modifications. First of all, we want to deal with the categorical variables.

Many ML algorithms do not accept categorical variables as inputs. They need numerical values. There is also another problem, that has to do with the structure of categorical variables. If a categorical variable includes many levels, it increases the complexity of the model and encumbers its performance.

We can deal with many levels by using combinations. We can combine different levels by using experience, or common logic. For example, we can combine levels of a variable “zip code” at state or district level. Another example is using bins instead of using ages.

But there is also the problem that ML algorithms need numbers as inputs. There are multiple approaches here. If we have bins, it might be a good idea to use the average. If we have hierarchical categories, we might use numbers to express this hierarchy (for example 1 for the most prominent value, 2 for the second most prominent value etc.).

One of the most efficient ways in dealing with categorical variables is dummy or one-hot encoding. The main idea behind dummies, is to create a new column for each value of the variable. The new columns would obtain boolean values, depending on the value of the initial categorical variable for a specific data point. For example, suppose that we have a variable named sex with possible values male and female. We will create two new, boolean columns: sex_male and sex_female. Every data point will have 1 to one column and 0 to the other, according to its sex. This is pretty useful, because it allow us to express relationships that are not understandable by a computer (if one variable is true the other must be false etc.)

The problem with dummy encoding is that it creates sparse matrices (matrices with many 0s), which burden the calculations and the whole performance, in general. We managed to solve this problem by statistical analysis. We have already found the most predominant values for each categorical value, for Frauds and for NoFrauds. For example, we know that the majority of Frauds are conducted by Males. So, our idea was to keep only those dummy columns with the values that are likely to arise in Frauds.

The list of the dummy variables that we decided to keep is the following:
' EARReasonID_14'
'brand_tripsta'
'category_Fraud'
'deviceSource_ext_NA'
'paymentCreditCardVendor_VI'
'profileGender_Male'
'TrueIpGeoMissmatch_TRUE'
'DetectedSuspiciousTransaction_ext_attributeNotSet'

Note that, when we use dummies, we drop the initial categorical variables. We have also decided not to use time and location variables. Location variables must have been placed in groups, or, at least in pairs, in order to discover high risk routes or other patterns. This analysis would have taken much time, which wasn't available.

On the other hand, time variables were dropped, because during the analysis, they produced sparse matrices, which were not easy to be processed. So, we decided to keep them off our initial application and implement them at a later time.

Finally, we decided to add a new variable, cost_per_passenger, which is the total cost of the booking, divided by the number of passengers. It seemed as a pretty relevant and logical variable.

After this changes and transformations, we apply the data (Apr_16) to the combinations we have already picked. We get the following results.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Feature Selection</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (c=100, kernel=rbf),</td>
<td>PCA(n_components = 14)</td>
<td>0.998</td>
<td>0.716</td>
<td>0.374</td>
<td>0.487</td>
</tr>
<tr>
<td>AdaBoost (n_estimators=50, learning_rate=1)</td>
<td>SelectKBest(f_classif, k=&quot;all&quot;)</td>
<td>0.998</td>
<td>0.776</td>
<td>0.590</td>
<td>0.669</td>
</tr>
<tr>
<td>Multinomial Naive Bayes</td>
<td>SelectKBest(f_classif, k=&quot;all&quot;)</td>
<td>0.998</td>
<td>0.789</td>
<td>0.684</td>
<td>0.732</td>
</tr>
<tr>
<td>Logistic Regression(C=10000, solver='lbfgs')</td>
<td>PCA(n_components = 14)</td>
<td>0.997</td>
<td>0.661</td>
<td>0.351</td>
<td>0.457</td>
</tr>
</tbody>
</table>

and Logistic Regression with SMOTE
Logistic Regression (C=10000, solver='lbfgs') with SMOTE

| LogisticRegression(C=10000, solver='lbfgs') with SMOTE | PCA(n_components = 14) | 0.977 | 0.134 | 0.897 | 0.233 |

The results for the testing dataset become:

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.997</td>
<td>0.716</td>
<td>0.487</td>
</tr>
<tr>
<td>AdaBoost</td>
<td>0.998</td>
<td>0.784</td>
<td>0.667</td>
</tr>
<tr>
<td>NB</td>
<td>0.998</td>
<td>0.782</td>
<td>0.717</td>
</tr>
<tr>
<td>LR</td>
<td>0.996</td>
<td>0.595</td>
<td>0.367</td>
</tr>
<tr>
<td>LR with SMOTE</td>
<td>0.963</td>
<td>0.103</td>
<td>0.933</td>
</tr>
</tbody>
</table>

The results confirm our hypothesis. The best algorithm to use is Naive Bayes. Next comes AdaBoost. Again, our final choice depends on the business model of the company (Naive Bayes is hard classifier, while AdaBoost is a soft one).

We can also keep the results for Logistic Regression with SMOTE. The combination manages to simulate the already existing fraud prevention system.

A final thought is to use two, or more models together, with a technique known as stacking. Here, as primary models, we would probably choose Naive Bayes, AdaBoost (they draw the results towards the direction of predicting certainty) and Linear Regression with SMOTE (it draws the result towards the direction of predicting power). Then, we would use a supervisor model, which will get the results, and combine them optimally, in order to get the balance between certainty and predictive power. We did not have the time to implement this approach thoroughly, but this would probably be the direction that we would follow from now on.
9 Conclusions - Final Thoughts

Through this project, it is proven that it is possible to create an effective classifier for imbalanced data (by the term “effective classifier” we mean a classifier which is able to predict good enough, in comparison to random guess).

We have discovered that the best algorithm for Fraud Detection is Naive Bayes. However, the most balanced algorithm is AdaBoost, due to the fact that it is characterized as soft classifier.

Another discovery is that the optimal validation technique in such occasions, is Stratified KFold. Oversampling techniques (such as SMOTE) produced results with increased numbers of false positives, without increasing the number of true positives.

We have also discovered some interesting insights about the data. For example, we have found out that missing values for variable addon.Parking are disproportionately higher for the Frauds than for NoFrauds. This might be, somehow revealing about the way fraudsters tend to use the service. There might also be a simpler explanation, such as a computer crash or a bug. Anyhow, such cases are worthy of further investigation.

At this point, we believe it is important to emphasize the need and the importance of statistical analysis, before the application of Machine Learning. Computers are capable of finding correlations between variables. But the statistical analysis would show if these correlations are meaningful in the real world. Besides that, analysis would reveal variables with statistical significant differences between the two subgroups (Frauds and NoFrauds).

So, the steps to follow are:

- Statistical Analysis, in order to find features with differences between Frauds and NoFrauds and to identify if these differences are meaningful.
- Feature Selection algorithms, in order to detect correlation between the chosen features.
- Machine Learning algorithms application.

Now, a few words about the next steps. We would transform the data for every month, the same way as we transformed data from April 2016. Then we will train the algorithms with data from every month. This point might prove to be tricky. We might discover other variables, with greater influence than the existing ones. We might also discover that further tuning is needed for the algorithms’ parameters.

Another idea is to use stacking. We discovered that Naive Bayes as well as AdaBoost produce results with better confidence but with smaller predicting ability (i.e. they miss some true positives although they do not produce false positives). On the other hand, we have algorithms such as Logistic Regression with SMOTE, which produce results that they manage to predict all Frauds, but not with great certainty (big number of false positives). An idea is to combine the results from the above, by using a supervisor model. This technique is known as stacking.
Another idea which looks promising is the following. Separate the whole dataset to \( n \) categories, by using Unsupervised Learning techniques (for example, with kmeans). Then, each new booking - datapoint is going to be assigned to one of these newly founded categories. A classifier is going to be trained for each new booking separately, with the majority of the data coming from the unsupervised category the booking belongs to.
10 Bibliography


