Fraud Detection

HarvardX PH125.9X - Data Science Capston pt. II

Agustin Gallo Fernandez

4/7/2021

Contents

1 Introduction

For obvious reasons is very important being able to recognize a fraud transactions from a legitimate one, these reasons varies from customer experience to billions of dollars in losses caused by fraudulent transactions therefore is of vital importance the development of algorithms which allows to detect and prevent this losses. These algorithms are challenging mainly because of their highly unbalanced data, as we have very few fraud identified cases against the no fraud transactions. At the same time is important to maintain anonymity as a dataset with for this purpose will deal with sensitive data.

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

2 Data Exploratory Analysis

As explained in the introduction, the dataset contains several features already selected from PCA analysis. Now we will review these features to look up for NA values, big values and correlated data.

To get a glance of the information we may see a few lines of the dataset, the dataset contains 284,807 observations, with 30 features, 28 features consist in PCA features, plus 2 more, time and amount. On the table below we just put the first and last features for visual purposes. It is important to note that the PCA features are possible the merge of two or more previous features which may have similar behavior and now are condensed in one unique feature.

Table 1: First six rows of the fraud dataset

Time	V1	V2	V27	V28	Amount	Class
0	-1.3598071	-0.0727812	0.1335584	-0.0210531	149.62	0
0	1.1918571	0.2661507	-0.0089831	0.0147242	2.69	0
1	-1.3583541	-1.3401631	-0.0553528	-0.0597518	378.66	0
1	-0.9662717	-0.1852260	0.0627228	0.0614576	123.50	0
2	-1.1582331	0.8777368	0.2194222	0.2151531	69.99	0
2	-0.4259659	0.9605230	0.2538442	0.0810803	3.67	0

2.1 NA Values and sparse data

Now we will display the number of observations and how many of them are Fraud (class 1) and no Fraud

Table 2: No of Observations

Count
284807

Table 3: Unbalanced data

Class	Number
0	284315
1	492

Also we can see if the number of values which are NA and if any of the amount values are negative (which make no sense in our context)

Table 4: NA/Negative Values

Concept	Count
Number of NA values	0
Amount negative values	0

With respect to the PCA features we will explore they distribution and we can how they are not very sparse having values between -20 and 20.

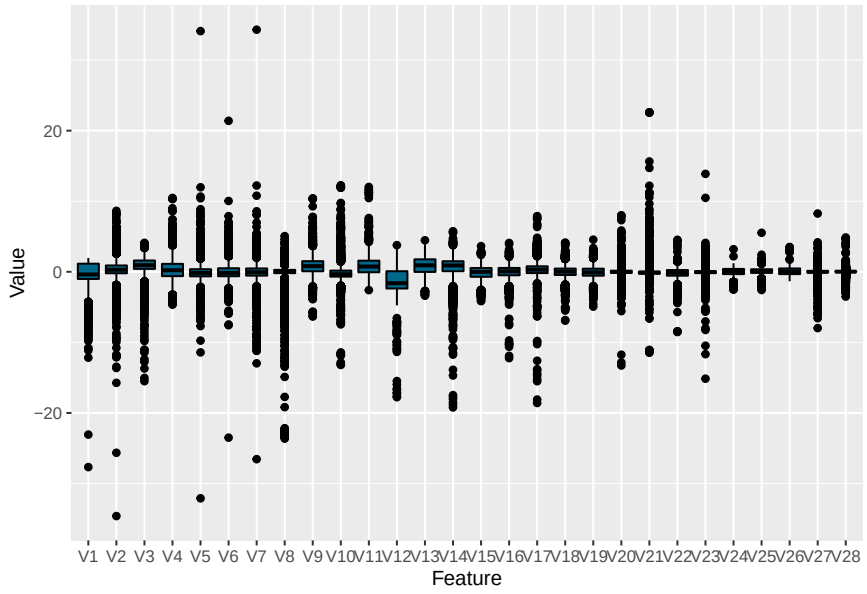

Figure 1: Features Distribution

2.2 Time Values

Times values are and special case on this dataset as only represents the time between the first transaction and the current observation, therefore only by this description we can infer there is no use to use it, but just in case we explore the data.

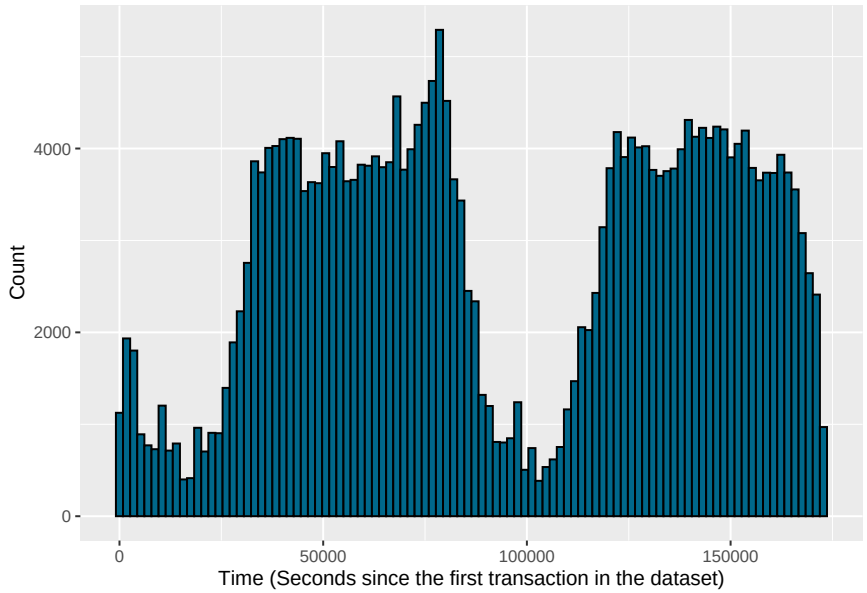

Figure 2: Time Distribution

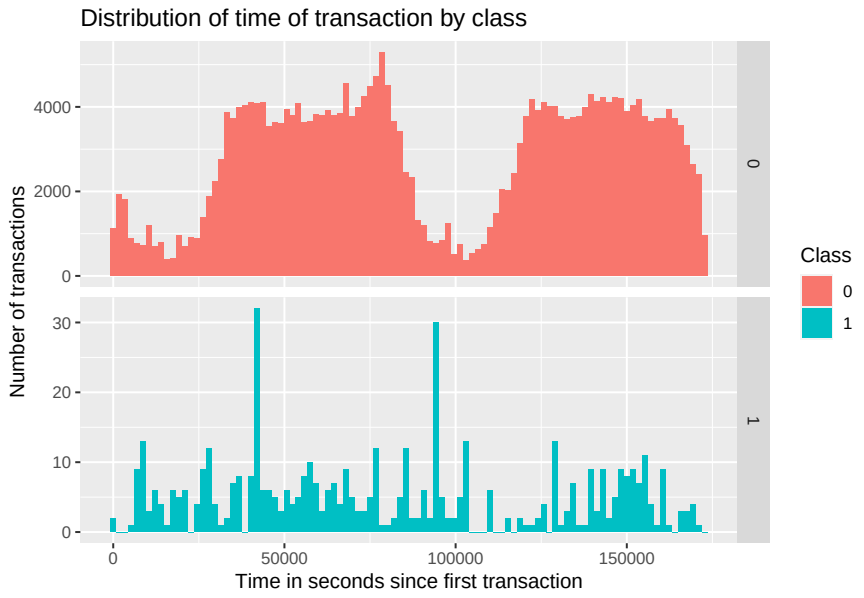

Figure 3: Time/Class Distribution

We can see here how the normal trasactions are made wihtin a given seasonality, while the fraud transactions seems to be more regular. Therefore we will keep the feature to use this seasonlaity.

2.3 Amount Values

Similar to the time values we can see transaction amount distribution in general and grouped by class

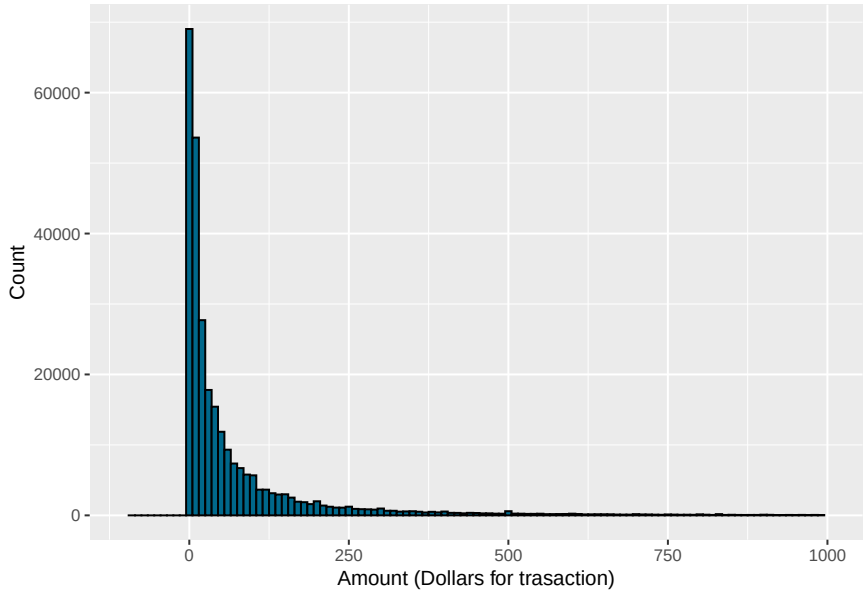

Figure 4: Amount Distribution

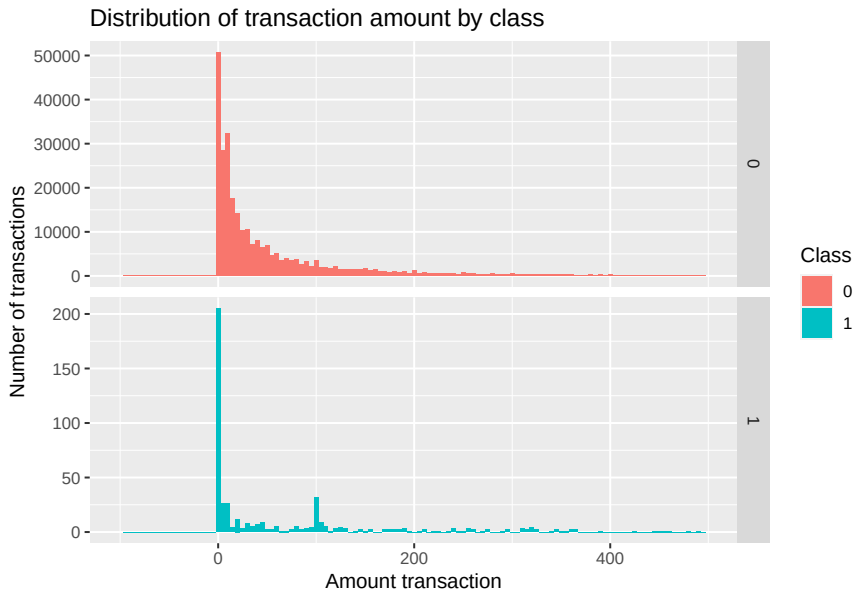

Figure 5: Amount Distribution by Class

We see most of the transactions are of a small value, mainly for the fraud class, with some little higher amounts (around $200) on the no-fraud class.

2.4 Features Correlation

Other interesting feature analysis we can try is to find the correlations between each feature to see if we can omit one of them. Another interesting visualization is to use tSNE (t-Distributed Stochastic Neighbor Embedding) which reduces the features space to 3 or 2 dimension (2D in our case) to see if is feasible to to distinguish one class from another using the parameters given.

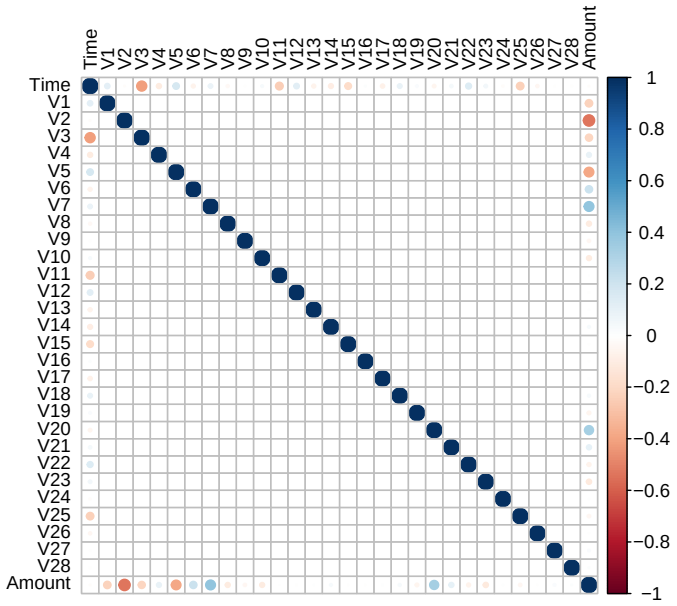

Figure 6: Correlation Plot

So, as expected (As this features are PCA features) we see there is technically no correlation between different variables, just a little one between V3-Time and V3-Amount.

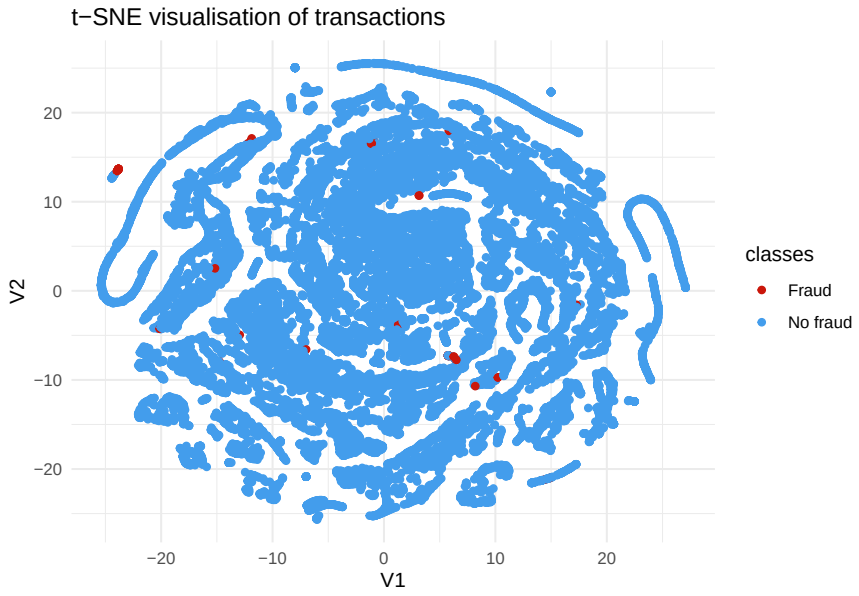

Figure 7: t-SNE Plot

Here we see on the axis V1 and V2, but they are not the original V1 and V2 from our fraud data, these are the parameters resulting from the incorporation of all variables for this analysis. We also can see the red dots corresponding to the fraud transactions, lucky for us, they seem to be on the border of the no fraud transactions, meaning that is less complicated to extract a model to find the fraud transaction.

3 Model Development

As we saw previously, we have a very unbalanced data. We can tackle this from to point of views, one of them will be to reduce the no-fraud data to be similar to the fraud cases. This make no sense for us, as if we do this we will be using less than 1 thousand cases to train our model.

Is for this that we will use the oposite approach, meaning that we will upsample our fraud data to be similar to the no fraud cases.

3.1 Data Splitting

We will split our data in two parts, the first of them will be our training data, which we will later augment to have a similar number of cases in fraud and in no-fraud. The proportion decided for this is going to be of 80% for training and 20% for testing, note that in testing we will not augment the fraud cases, we will test just as the data came from. This proportion is to have at least some cases to test the detection for the fraud cases in our test data, because if we have less data than that in our test dataset we will have very few cases.

3.2 Measure metrics

On this highly unbalance data we cannot use the typically accuracy metric, as this metric is of the form:

$$acc = \frac{(TP + TN}{Total)}$$

Where TP are True Positives; TN are True Negatives; and Total the total observations

So we can have a lot o True Positives with a lot false Positives and this will not appear in our score, so we will use another metrics which use recall to measure of model.

Other option to use is to use F1 score as it mix True and False Negative rates.

$$F1_score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Another option also is to use the ROC - Area Under Curve which uses the True Positive Rate against the False Positive Rate to obtain its value

Just as a reminder the Recall and Precision formulas:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

We will try all of these metrics on the following steps and find which is the more descriptive way to measure our model.

3.3 Upsample method

As commented previously we will upsample the fraud cases in our training data to improve the performance in our model training. The method that we will use is the SMOTE method (Synthetic Minority Oversampling Technique).

SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

Specifically, a random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

This procedure can be used to create as many synthetic examples for the minority class as are required. In the paper, it suggests first using random undersampling to trim the number of examples in the majority class, then use SMOTE to oversample the minority class to balance the class distribution. But in this case we will go directly to upsample the minority class.

You can read more about this on SMOTE: Synthetic Minority Over-sampling Technique

3.4 Models

We will start with the design of three dummy models to test our model metrics and also to get a starting point, after this we will proceed developing 4 more models using Classification and Regression Trees (CART), Random Forest and Gradient Boosting (XGBoost). So summarizing we will use: * Dummy models. Test metrics and set starting point. * CART models. Single Tree using unbalances and balanced data using SMOTE. * Random Forest Model. Using SMOTE. * XGBoost Model. Two models using XGBoost, one of them using all the parameters the other using top 7 parameters.

3.4.1 Dummy models

We will try 3 dummy models in order to test our metrics, accuracy, F1 score and ROC-AUC, and to use as comparing point for the following models.

Our dummy attempts will consist in 3 types off prediction:

1. **Random**: Choose randomly between 1 and zero to predict our class.
2. **No-fraud**: Choose always 0, do not make any prediction at all, as fraud cases are very small we can say is never fraud and still make a good prediction.
3. **Dummy statistic**: Based on the probability between fraud and no fraud, chose one of them based on their occurrence i.e. i if relationship is 10 to 1 predict 10 to 1 occurrence of the class.

3.4.2 CART

The representation for the CART model is a binary tree.

This is your binary tree from algorithms and data structures, nothing too fancy. Each root node represents a single input variable (x) and a split point on that variable (assuming the variable is numeric).

The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.

For example, in the titanic surivival analysis, the classification tree will be as follows.

“sibsp” is the number of spouses or siblings aboard. The figures under the leaves show the probability of survival and the percentage of observations in the leaf. Summarizing: Your chances of survival were good if you were (i) a female or (ii) a male younger than 9.5 years with strictly less than 3 siblings.

Our case is similar but instead off having 3 parameters (gender, age and sibsp), we have 30. Crazy eh!

3.4.3 Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.

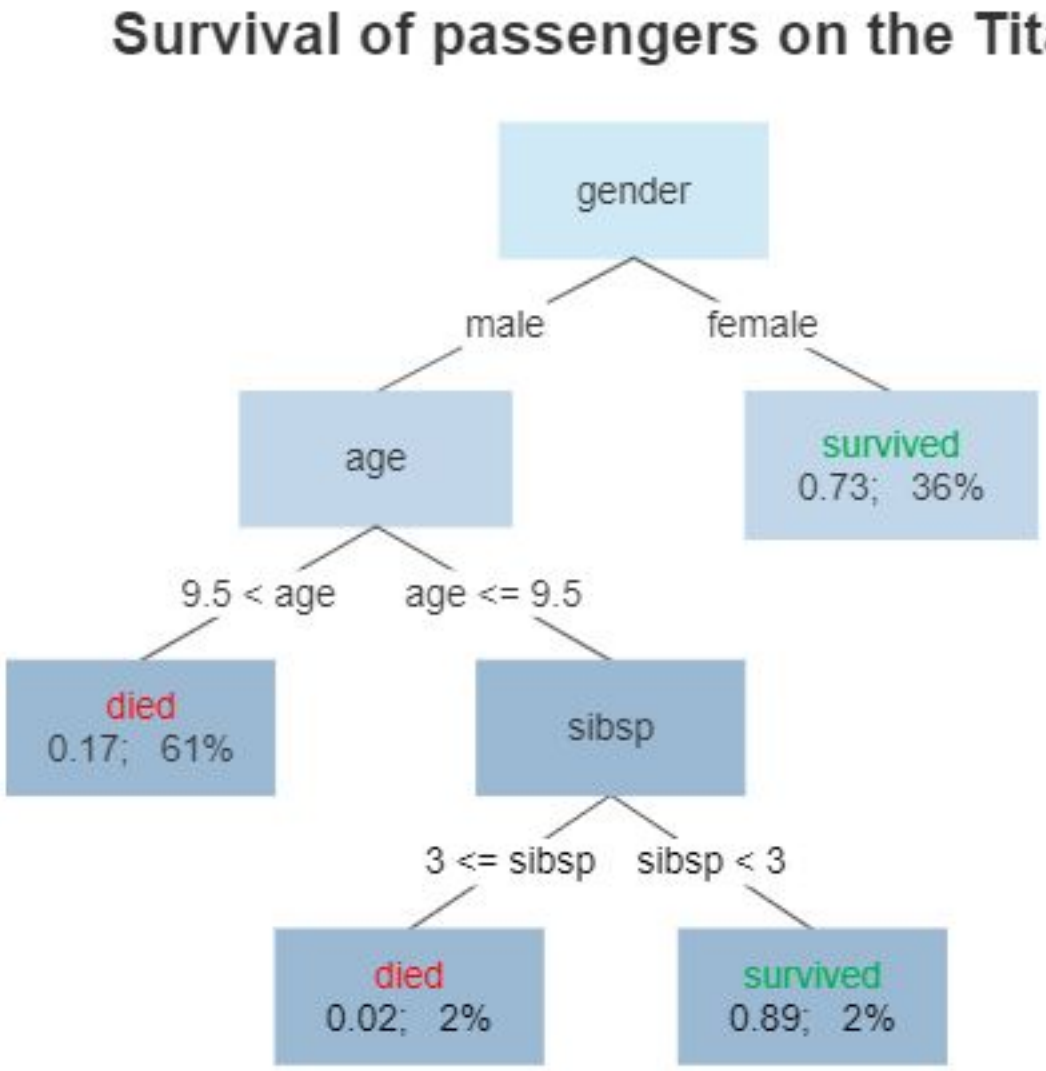

Figure 8: Titanic Binary Decision Tree

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

3.4.4 XGBoost

XGBoost stands for "Extreme Gradient Boosting", where the term "Gradient Boosting" originates from the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman.

XGBoost is used for supervised learning problems, where we use the training data (with multiple features) to predict a target variable.

XGBoost minimizes a regularized objective function (L1 and L2) that combines a convex loss function (based on the difference between the expected and target outputs) and a penalty term for the complexity of the model (in other words, the functions of the tree of regression). Training continues iteratively, adding new trees that predict the residuals or errors from previous trees that are then combined with previous trees to make the final prediction. It is called gradient augmentation because it uses a gradient descent algorithm to minimize loss when adding new models.

4 Results

Following we will present the results obtained from the procedures described above and table sumarizing the results for each one of the models used.

4.1 SMOTE Upsampling

As a recap, we saw on table 3 that for the Fraud class we only had 492 cases in all our data set. We perform the data split and get the next results.

Table 5: Train Unbalanced Data

Class	Count
0	227452
1	394

Table 6: Train Balanced Data

Class	Count
0	227452
1	227338

With this, we can see that our upsampling technique seem to work, now it is time to check our metrics and if this upsample represents a benefit for the model training.

4.2 Metrics Evaluation and Dummy Attempts

Now we will test our dummies attempts. We will show the ROC curves for each one of the models and a final table comparing each of the metrics.

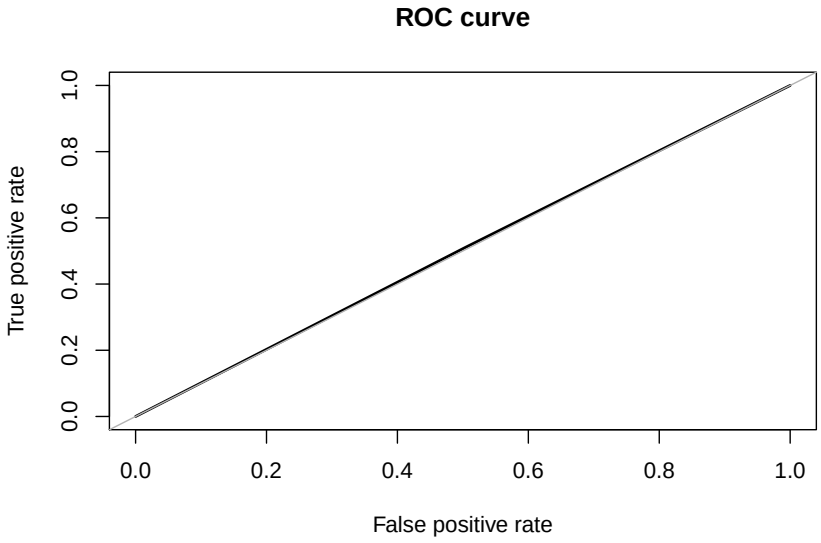

Figure 9: ROC Curve for random attempt

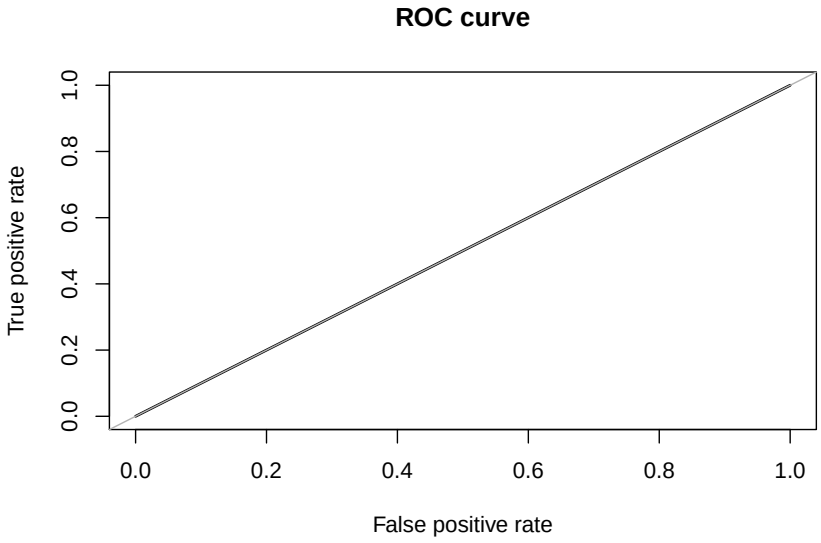

Figure 10: ROC Curve for no-Fraud attempt

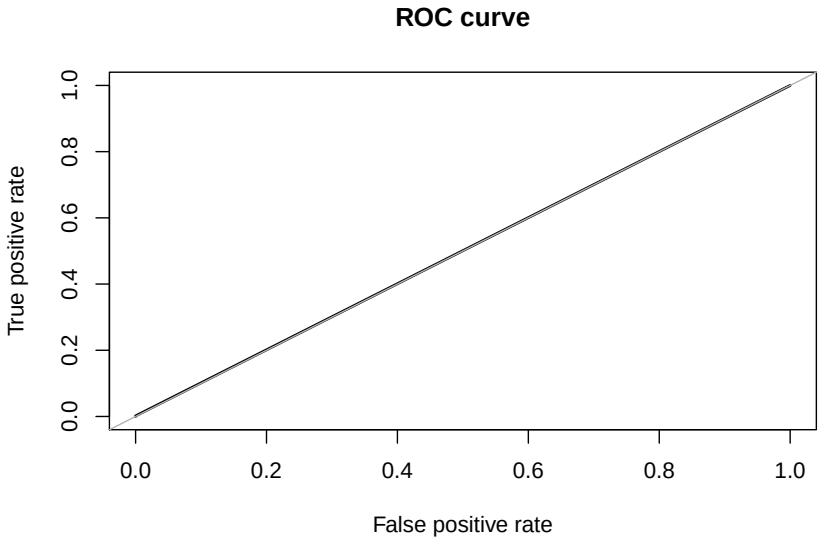

Figure 11: ROC Curve for statistic attempt

Finally we will show the results obtained from the previous attempts.

Table 7: Dummy Attempts Summary

Attempt	Accuracy	F1_Score	AUC_Value
Random Choice	0.4967258	0.6633551	0.5034533
No-Fraud	0.9982795	0.9991302	0.5000000
Statistic	0.9966117	0.9983030	0.5008353

On the above table we can see how AUC gives a better look of how to measure our models as F1 Score and Accuracy do not give a correct metric of how well our model is performing when clearly we are using a bad prediction model.

For this reason we will further use ROC-AUC as our metric for the next models.

4.3 CART results

Now we show the result obtained using a CART models, the first of them will use the unbalanced data and the second model will use the balanced dataset.

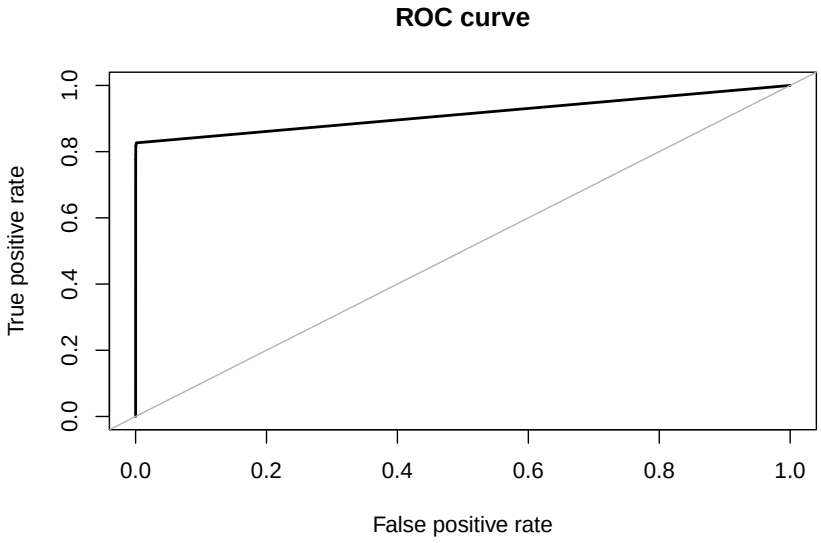

Figure 12: ROC Curve for UNBALANCED data CART

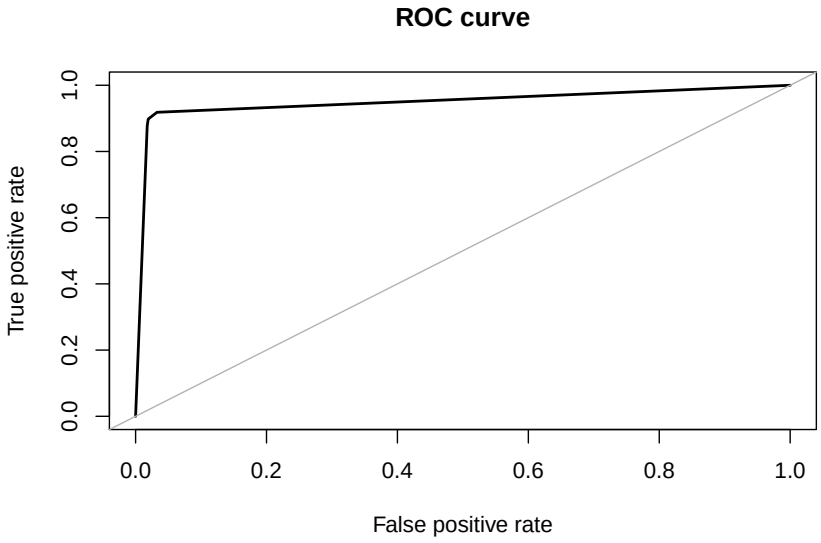

Figure 13: ROC Curve for BALANCED data CART

Now that we have the ROC curves obtained we can now see if our prediction is better with the SMOTE upsample data or not. And it is, as its show in the below table.

4.4 Random Forest

Now take a look using a Random Forest Approach

Table 8: Balanced vs Unbalanced train data

Data	AUC_Value
Unbalanced Fraud cases	0.9131086
Balanced Fraud Cases	0.9491560

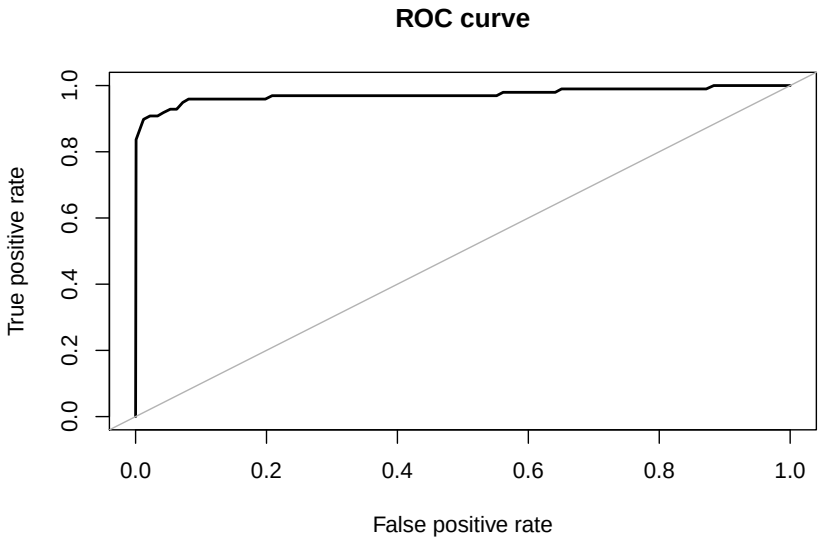

Figure 14: ROC Curve for Random Forest

This looks pretty great, and we have an Area under the Curve of: 0.9727891

4.5 XGBoost results

Now its time to hit our last models, we will see if XGBoost can outperform what random forest did

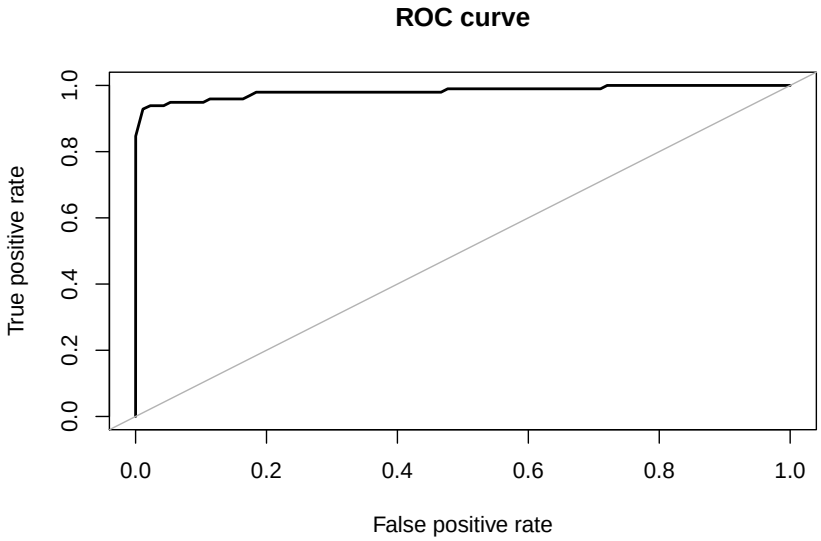

Figure 15: ROC Curve for XGBoost

With a AUC value of: 0.9820185

Now take a look to the top 10 feature for this classifier

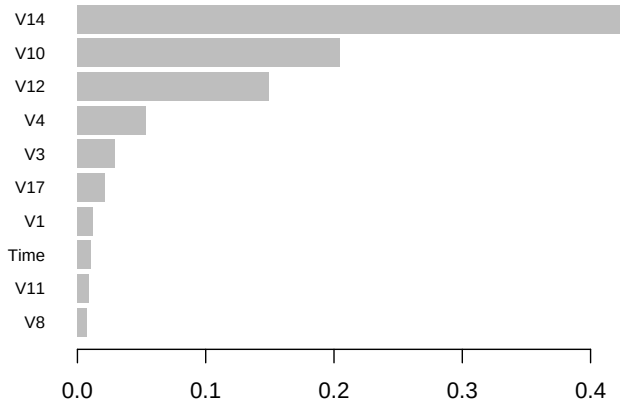

Figure 16: XGBoost, most important features

Now take a look to what happen if we only chose the top 7 features

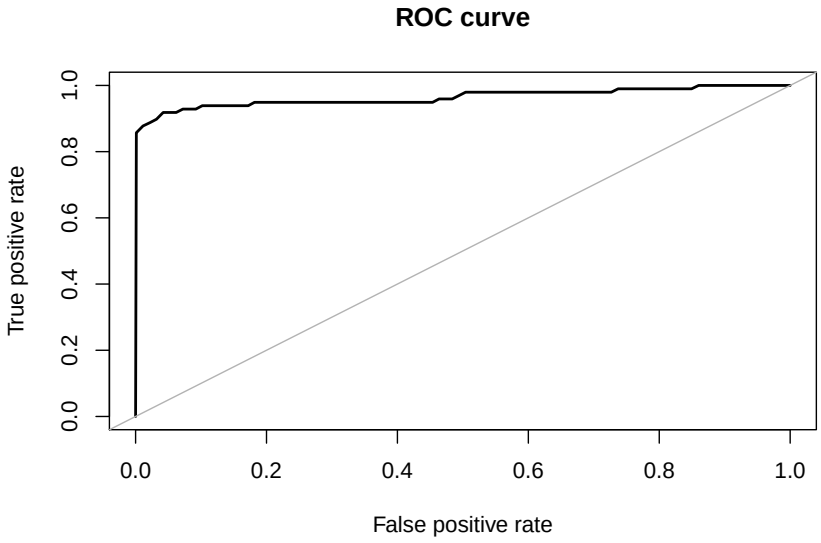

Figure 17: ROC Curve, for XGBoost with top 7 features

With a final AUC: 0.9637488. Which is not as good as using all the features, but still a good try.

4.6 Sumarize

Just to add up everything we made so far, lets summarize it on a final table.

Table 9: Model Comparison

Model	AUC Value
Random	0.5034533
noFraud	0.5000000
Dummy Statistics	0.5008353
CART no Balance Data	0.9131086
CART Balanced Data	0.9491560
Random Forest	0.9727891
XGBoost	0.9820185
XGBoost top 7	0.9637488

5 Conclussions

With this we finalize this project and we can see how XGBoost using all the variables outperform all other methods, but the top 7 features not laying so far from there, even when random forest did a great job.

Hopefully this project can give some insight about how to make the exploratory analysis, deal with unbalanced data, the metrics used for this, how to upsample the minority class to improve your training and also, we saw several algorithms to make a classification of the given data, this will help us in the future how to chose one of them considering the trade off between performance, time and the features the data has.

6 References

Thanks to all the people who made this possible, writing articles or papers over this topic or making the dataset available:

- Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015
- Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective, Expert systems with applications,41,10,4915-4928,2014, Pergamon
- Dal Pozzolo, Andrea; Boracchi, Giacomo; Caelen, Olivier; Alippi, Cesare; Bontempi, Gianluca. Credit card fraud detection: a realistic modeling and a novel learning strategy, IEEE transactions on neural networks and learning systems,29,8,3784-3797,2018,IEEE
- Dal Pozzolo, Andrea Adaptive Machine learning for credit card fraud detection ULB MLG PhD thesis (supervised by G. Bontempi)
- Carcillo, Fabrizio; Dal Pozzolo, Andrea; Le Borgne, Yann-Aël; Caelen, Olivier; Mazzer, Yannis; Bontempi, Gianluca. Scarff: a scalable framework for streaming credit card fraud detection with Spark, Information fusion,41, 182-194,2018,Elsevier
- Carcillo, Fabrizio; Le Borgne, Yann-Aël; Caelen, Olivier; Bontempi, Gianluca. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization, International Journal of Data Science and Analytics, 5,4,285-300,2018,Springer International Publishing
- Bertrand Lebichot, Yann-Aël Le Borgne, Liyun He, Frederic Oblé, Gianluca Bontempi Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection, INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp 78-88, 2019
- Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Frederic Oblé, Gianluca Bontempi Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection Information Sciences, 2019
- Yann-Aël Le Borgne, Gianluca Bontempi Machine Learning for Credit Card Fraud Detection - Practical Handbook
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.SMOTE: Synthetic Minority Over-sampling Technique
- Atharva Ingle, Credit Card Fraud Detection with R + (sampling)
- Jason Brownlee SMOTE for Imbalanced Classification with Python
- Jason Brownlee, Classification And Regression Trees for Machine Learning
- Tony Yiu, Understanding Random Forest
- Gentle Introduction of XGBoost Library