**CSE-4020**

**MACHINE LEARNING**

**J COMPONENT REPORT**

**FACULTY NAME: Dr R Rajalakshmi**

**Name : S Aswin**

**Registration Number :17BCE1201**



**CREDIT CARD FRAUD DETECTION**

**Context**

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

**Problem Statement:**

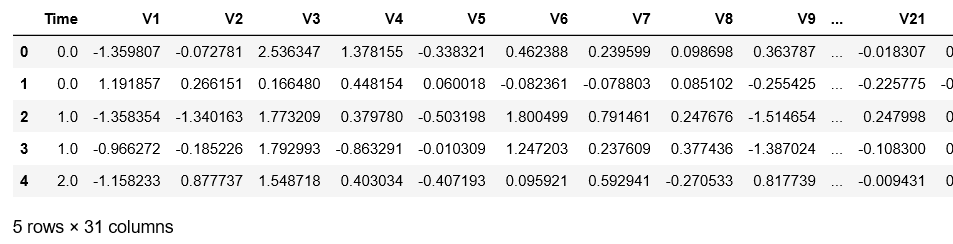
The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

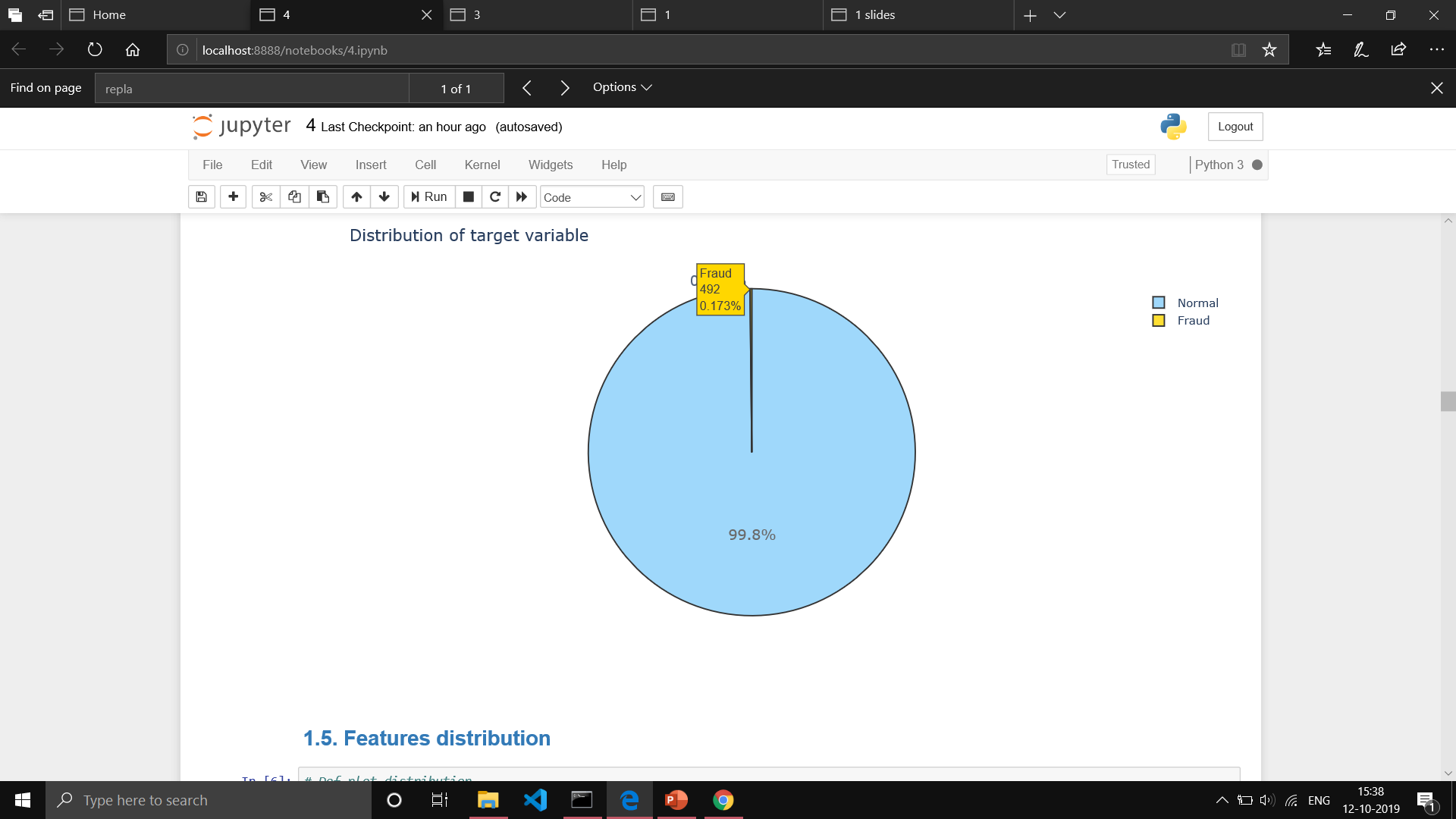
**Database:**

The datasets 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. 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.

Overview of Database:







Agenda:

* Understand the little distribution of the ‘little’ data that is provided.
* Create a 50/50 sub- Data frame ratio of “Fraud” and “Non fraud” transactions.
* Determine the Classifiers we are going to use and decide which one has a higher accuracy.
* Create a neural network and compare the accuracy to our best classifier.
* Understand common mistakes made with imbalanced datasets

Approach:

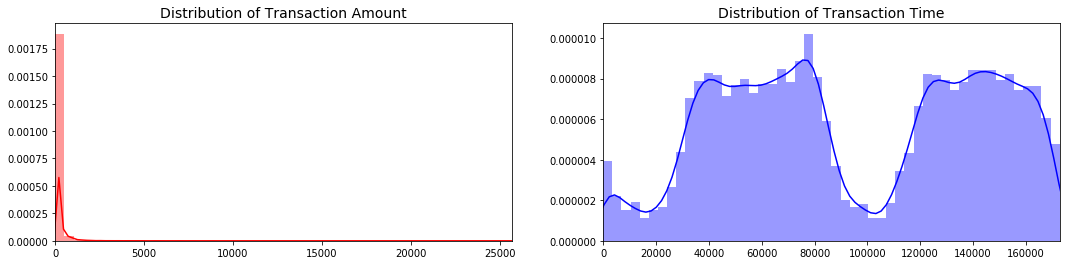
* Changing the performance metric:
  + Use the confusion matrix to calculate Precision, Recall
  + F1score (weighted average of precision recall)
  + Use Kappa - which is a classification accuracy normalized by the imbalance of the classes in the data
  + ROC curves - calculates sensitivity/specificity ratio.
* Resampling the dataset
  + Essentially this is a method that will process the data to have an approximate 50-50 ratio.
  + One way to achieve this is by OVER-sampling, which is adding copies of the under-represented class (better when you have little data)
  + Another is UNDER-sampling, which deletes instances from the over-represented class (better when he have lot's of data)

Feature engineering is not to be done as the data set has already In it .

The dataset has been downgraded in order to contain 30 features (28 anonamised + time + amount).

**Distributions:**

By seeing the distributions we can have an idea how skewed are these features, we can also see further distributions of the other features. There are techniques that can help the distributions be less skewed which will be implemented in this notebook in the future.



### **What is a sub-Sample?**

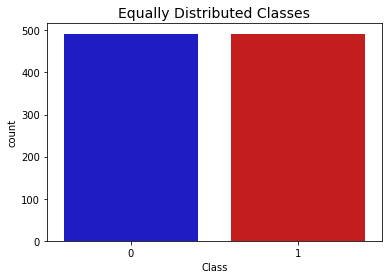
In this scenario, our subsample will be a data frame with a 50/50 ratio of fraud and non-fraud transactions. Meaning our sub-sample will have the same amount of fraud and non- fraud transactions.

### **Why do we create a sub-Sample?**

In the beginning of this notebook we saw that the original data frame was heavily imbalanced! Using the original data frame will cause the following issues:

* **Overfitting:**Our classification models will assume that in most cases there are no frauds! What we want for our model is to be certain when a fraud occurs.
* **Wrong Correlations:** Although we don't know what the "V" features stand for, it will be useful to understand how each of this features influence the result (Fraud or No Fraud) by having an imbalance data frame we are not able to see the true correlations between the class and features.

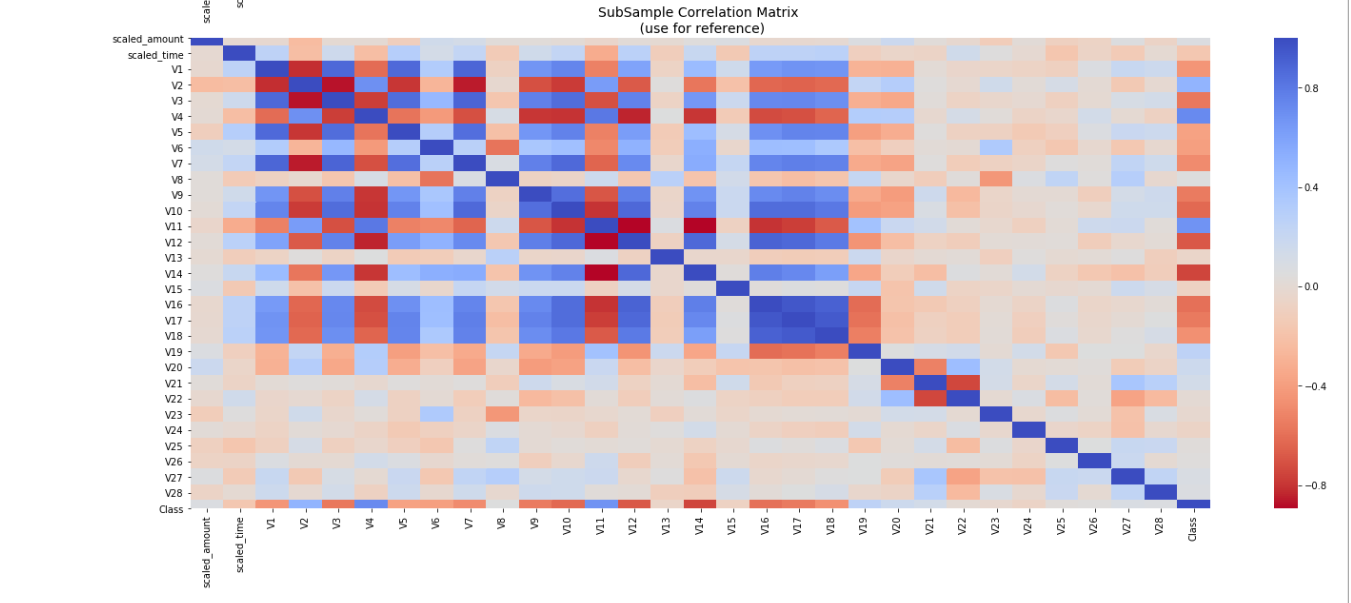
Sub Sampling and equivalent data for Fraud and Non-Fraud Transactions:



Co-relation Matrix:

**Correlation** is a statistical measure that indicates the extent to which two or more variables fluctuate together. A positive **correlation** indicates the extent to which those variables increase or decrease in parallel; a negative **correlation** indicates the extent to which one variable increases as the other decreases.

We plot correlation matrix to find variables which influence the classes



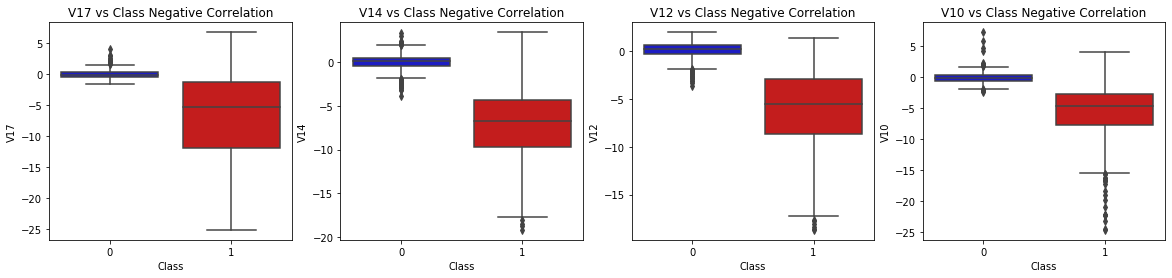
From the above figure we can infer that

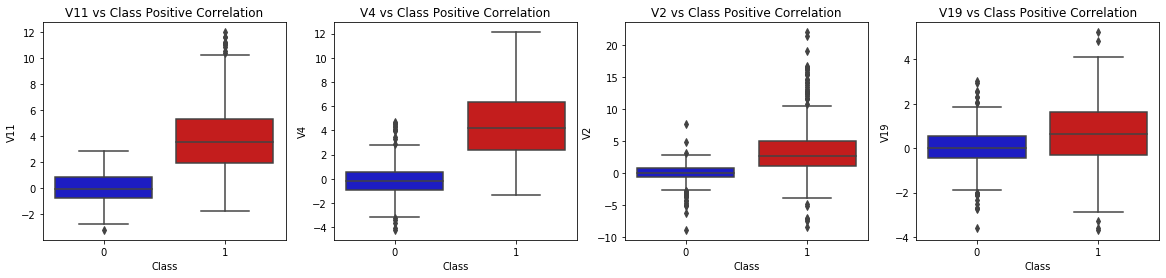
V2, V4 , V11,V19 are positively corelated

V10,V12,V14,V17 are negatively correlated.

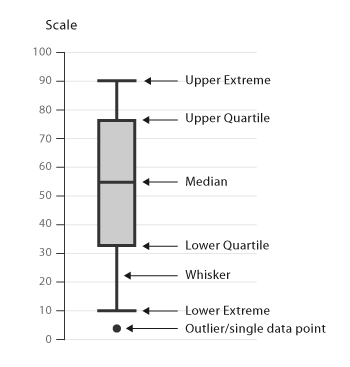
Hence we choose the above variables to determine whether it’s a fraud transaction or non-fraud.

We Plot boxplots to find the range of the above Variables





Anomaly Detection:



Our main aim in this section is to remove "extreme outliers" from features that have a high correlation with our classes. This will have a positive impact on the accuracy of our models.

### Interquartile Range Method:[¶](http://localhost:8888/notebooks/1.ipynb#Interquartile-Range-Method:)

* **Interquartile Range (IQR):** We calculate this by the difference between the 75th percentile and 25th percentile. Our aim is to create a threshold beyond the 75th and 25th percentile that in case some instance pass this threshold the instance will be deleted.
* **Boxplots:** Besides easily seeing the 25th and 75th percentiles (both end of the squares) it is also easy to see extreme outliers (points beyond the lower and higher extreme).

### Outlier Removal Tradeoff:[¶](http://localhost:8888/notebooks/1.ipynb#Outlier-Removal-Tradeoff:)

We have to be careful as to how far do we want the threshold for removing outliers. We determine the threshold by multiplying a number (ex: 1.5) by the (Interquartile Range). The higher this threshold is, the less outliers will detect (multiplying by a higher number ex: 3), and the lower this threshold is the more outliers it will detect.

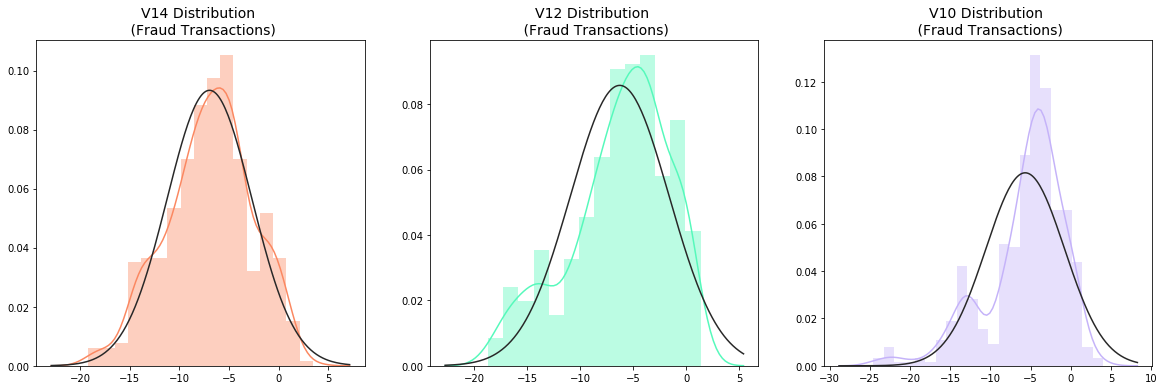
\*The Tradeoff: \* The lower the threshold the more outliers it will remove however, we want to focus more on "extreme outliers" rather than just outliers. Why? because we might run the risk of information loss which will cause our models to have a lower accuracy. You can play with this threshold and see how it affects the accuracy of our classification models.

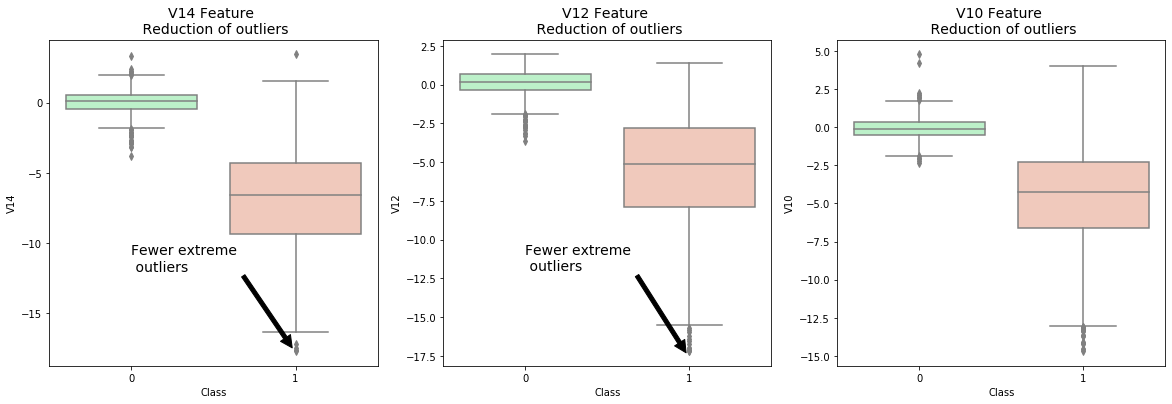
### Summary:[¶](http://localhost:8888/notebooks/1.ipynb#Summary:)

* **Visualize Distributions:** We first start by visualizing the distribution of the feature we are going to use to eliminate some of the outliers. V14 is the only feature that has a Gaussian distribution compared to features V12 and V10.
* **Determining the threshold:** After we decide which number we will use to multiply with the iqr (the lower more outliers removed), we will proceed in determining the upper and lower thresholds by substrating q25 - threshold (lower extreme threshold) and adding q75 + threshold (upper extreme threshold).
* **Conditional Dropping:** Lastly, we create a conditional dropping stating that if the "threshold" is exceeded in both extremes, the instances will be removed.
* **Boxplot Representation:** Visualize through the boxplot that the number of "extreme outliers" have been reduced to a considerable amount.

**Note:** After implementing outlier reduction our accuracy has been improved by over 3%! Some outliers can distort the accuracy of our models but remember, we have to avoid an extreme amount of information loss or else our model runs the risk of underfitting.

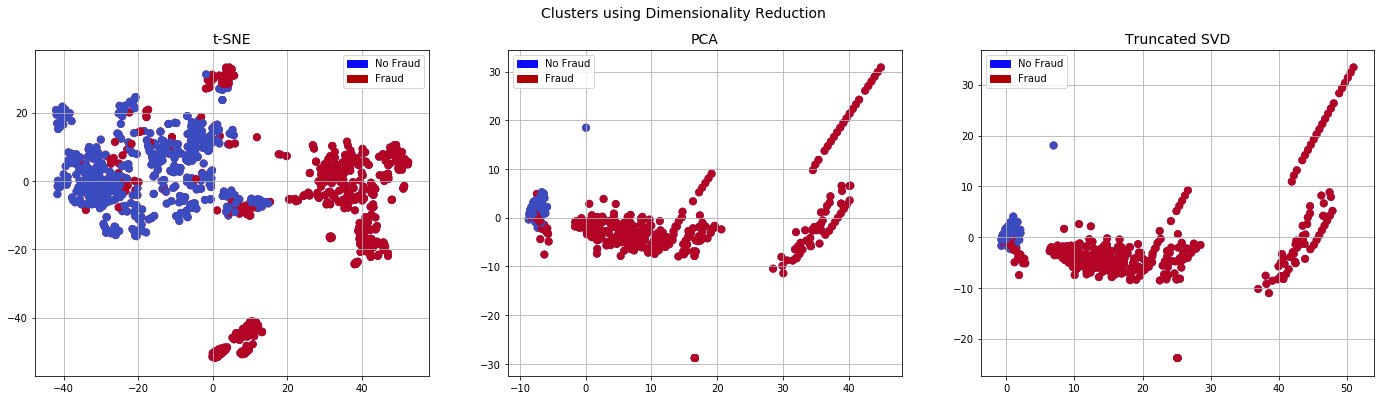
**Reference**: More information on Interquartile Range Method: How to Use Statistics to Identify Outliers in Data by Jason Brownless (Machine Learning Mastery blog)





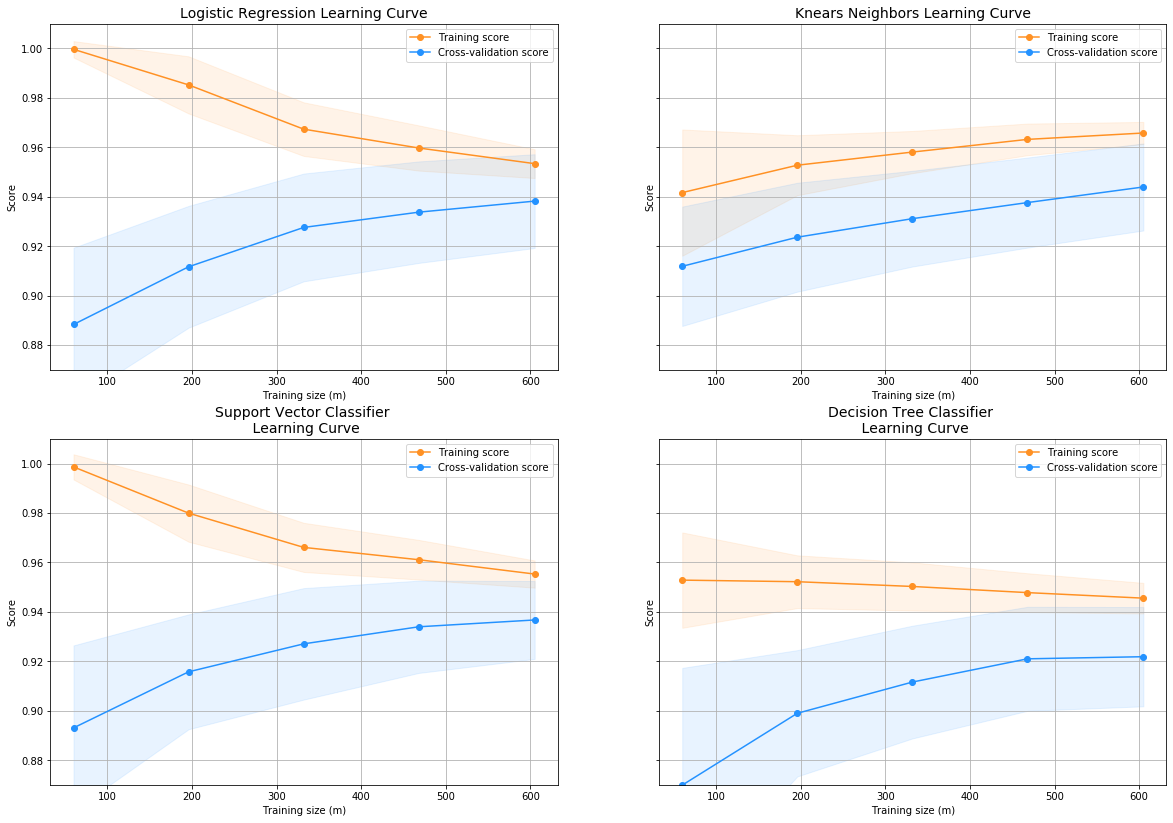
Dimensional Reduction and Clustering

* T-SNE (t-Distributed Stochastic Neighbor Embedding) is a dataset decomposition technique which reduced the dimensions of data and produces only top n components with maximum information.
* Every dot in the following represents a transaction. Non Fraud transactions are represented as Green while Fraud transactions are represented as Red. The two axis are the components extracted by T-SNE.
* T-SNE algorithm can pretty accurately cluster the cases that were fraud and non-fraud in our dataset.
* Although the subsample is pretty small, the T-SNE algorithm is able to detect clusters pretty accurately in every scenario(shuffled dataset).
* This gives us an indication that further predictive models will perform pretty well in separating fraud cases from non-fraud cases.



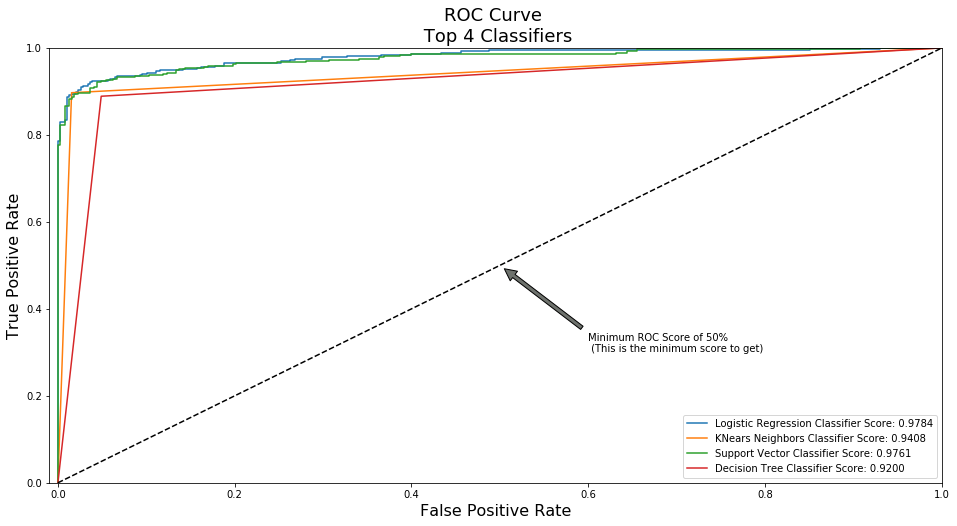
PREDICTION USING CLASSIFIERS:

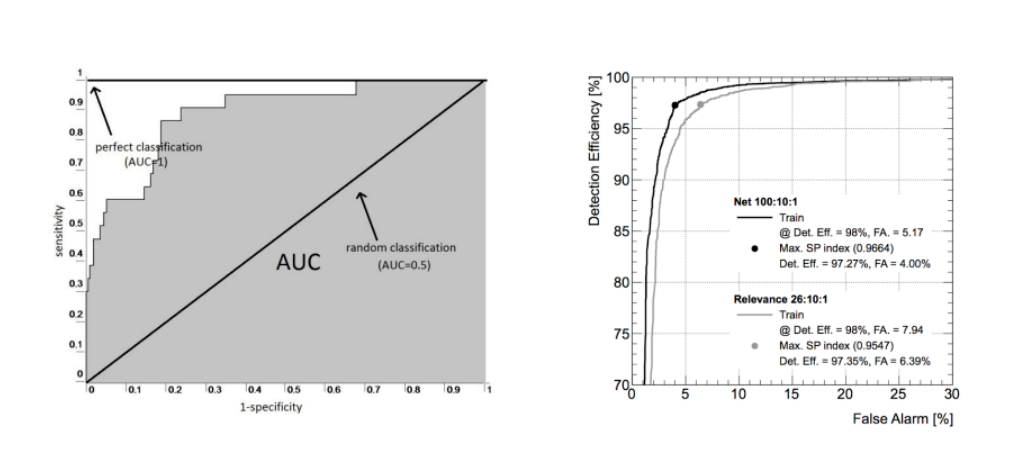
* Classifiers: Logistic Regression Has a training score of 94.0 % accuracy score
* Classifiers: K Neighbors Classifier Has a training score of 93.0 % accuracy score
* Classifiers: SVC Has a training score of 93.0 % accuracy score
* Classifiers: Decision Tree Classifier Has a training score of 90.0 % accuracy score



ROC\_AUC curve score

* **Roc Curve** : The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
* Logistic Regression: 0.9783966138630458
* K Nears Neighbors: 0.9407775263319645
* Support Vector Classifier: 0.9760637554829158
* Decision Tree Classifier: 0.9199599083619702





LGBM MODEL

LIGHT GRADIENT BOOSTING METHOD

* \*\* LightGBM\*\* is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:
* Faster training speed and higher efficiency.
* Lower memory usage.
* Better accuracy.
* Support of parallel and GPU learning.
* Capable of handling large-scale data
* **Light GBM grows tree vertically**while other algorithm grows trees horizontally meaning that Light GBM grows tree **leaf-wise**while other algorithm grows level-wise.
* It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.

***Why Light GBM is gaining extreme popularity?***

* The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as ‘Light’ because of its **high speed.**Light GBM can **handle the large size** of data and **takes lower memory to run**. Another reason of why Light GBM is popular is because it **focuses on accuracy of results**. LGBM also **supports GPU learning** and thus data scientists are widely using LGBM for data science application development.

Parameters:

**max\_depth:** It describes the maximum depth of tree. This parameter is used to handle model overfitting.

**min\_data\_in\_leaf:** It is the minimum number of the records a leaf may have.

The default value is 20, optimum value. It is also used to deal over fitting.

**feature\_fraction:** Used when your boosting is random forest. 0.8 feature fraction means LightGBM will select 80%

of parameters randomly in each iteration for building trees.

**bagging\_fraction:** specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid overfitting.

**early\_stopping\_round:** This parameter can help you speed up your analysis. Model will stop training if one metric of one validation data doesn’t improve

in last early\_stopping\_round rounds. This will reduce excessive iterations.

**lambda:**lambda specifies regularization. Typical value ranges from 0 to 1.

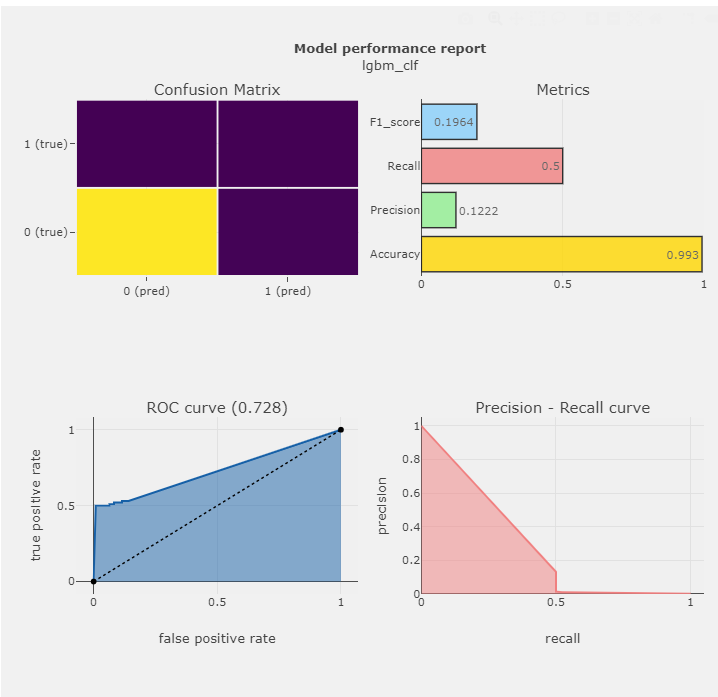
**min\_gain\_to\_split:** This parameter will describe the minimum gain to make a split. It can used to control number of useful splits in tree.

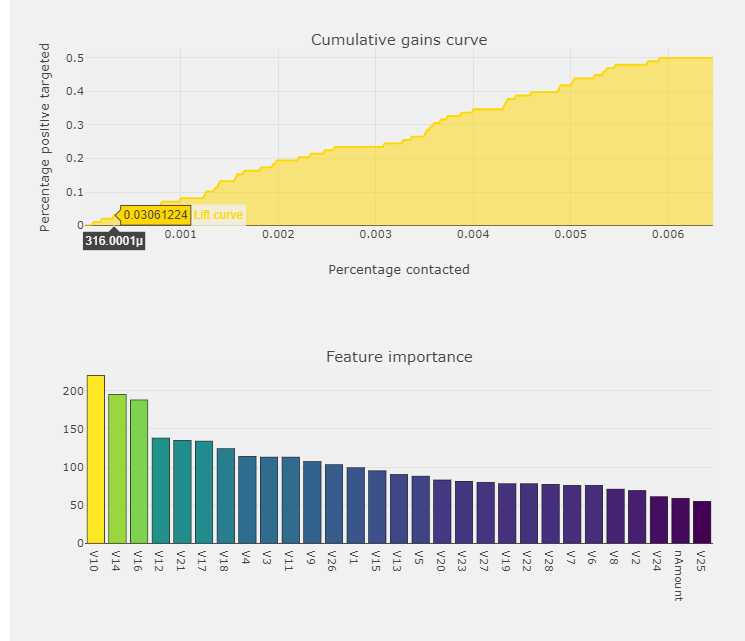
**max\_cat\_group:**When the number of category is large, finding the split point on it is easily over-fitting.

So LightGBM merges them into ‘max\_cat\_group’ groups, and finds the split points on the group boundaries, default:64

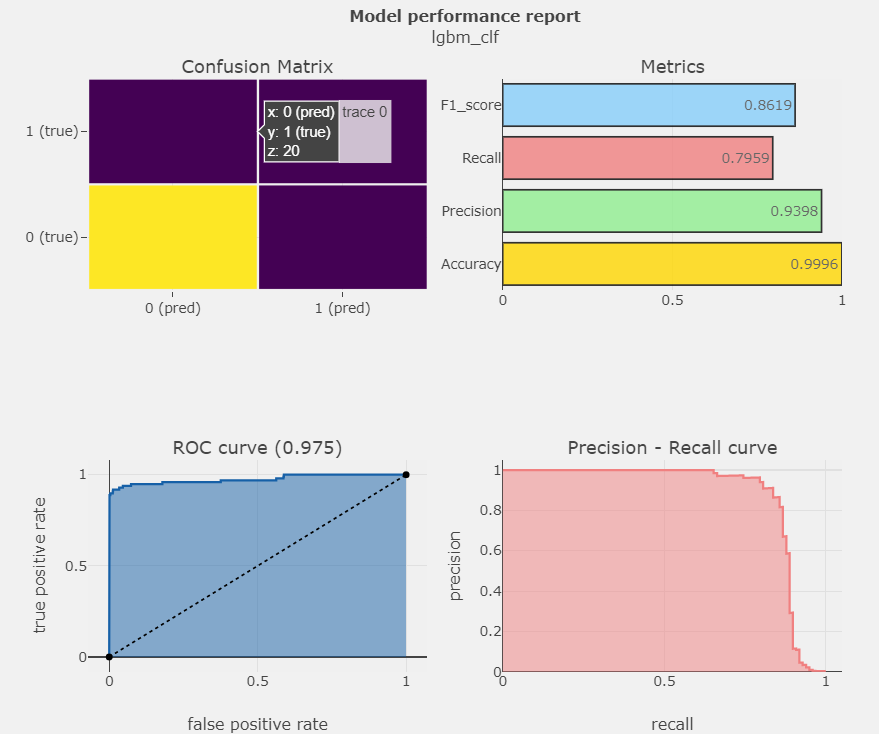
Performance Report:

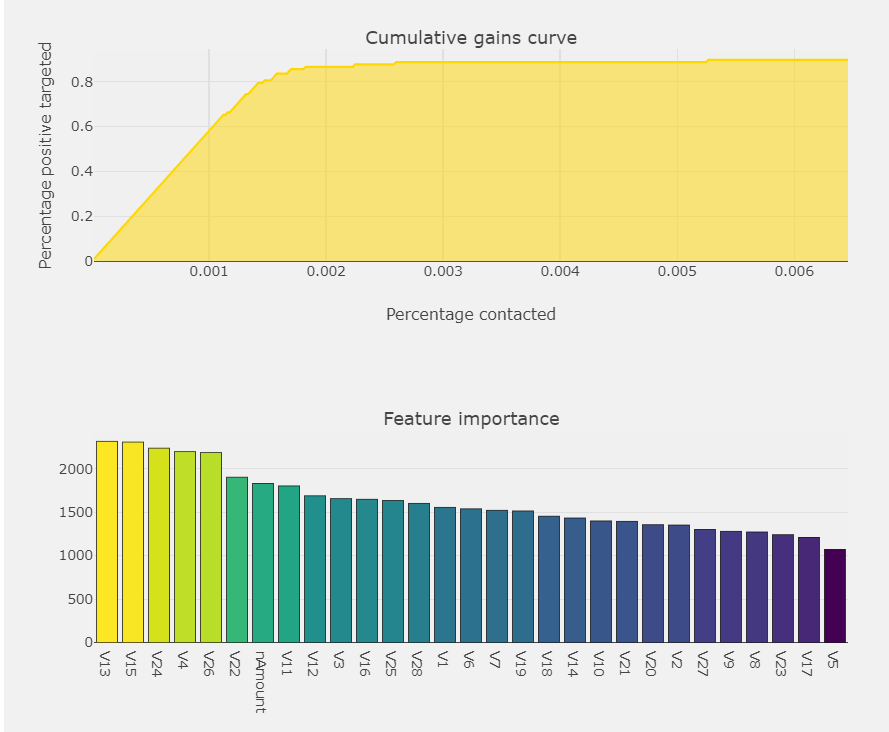
Before Randomized CV:





After Randomized CV:





Conclusion:

Implementing SMOTE on our imbalanced dataset helped us with the imbalance of our labels (more no fraud than fraud transactions).

Removal of outliers was implemented on the under sample dataset and not on the oversampled one.

Also, in our under sample data our model is able to detect for a large number of cases non fraud transactions correctly .

LGBM along with Randomized CV turns out the best Model for the above case.