Credit card Fraud detection system

W.A.D.T.L.Wijayaweera

IT17073974

Abstract

Credit card fraud events take place frequently and then result in huge financial losses. The number of online transactions has grown in large quantities and online credit card transactions hold a huge share of these transactions. Therefore, banks and financial institutions offer credit card fraud detection applications much value and demand. Fraudulent transactions can occur in various ways and can be put into different categories. This paper focuses on four main fraud occasions in real-world transactions. Each fraud is addressed using a series of machine learning models and the best method is selected via an evaluation. This evaluation provides a comprehensive guide to selecting an optimal algorithm concerning the type of frauds and we illustrate the evaluation with an appropriate performance measure. Another major key area that we address in my project is real-time credit card fraud detection.

Table of contents

1. Introduction

This is a system analyses user credit card data for various characteristics and detects whether fraudulent or not using machine learning. Customers transact across multiple channels and cybercriminals utilize bots and synthetic identities to launch ever increasingly sophisticated attacks making it exceedingly difficult and critical for you to be able to discern between trusted customers and fraudsters. Enormous Data is processed every day and the model build must be fast enough to respond to the scam in time, imbalanced Data i.e most of the transactions (99.8%) are not fraudulent which makes it hard for detecting the fraudulent ones, Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported, Data availability as the data is mostly private Adaptive techniques used against the model by the scammers are some of the challenges faced by involved in credit card fraud detection.

Fraud and Identity Management solutions combine transformative physical and digital identity insights with advanced decisioning and authentication technology to deliver a precise view of customer identity risk. Our solutions enable your business to raise the effectiveness and accuracy. When the user inputs the amount and time it predicts whether the transaction is fraudulent or not. This is done by using machine learning and the algorithm I used is a random forest algorithm that comes under shallow learning.

2. Overview

Our main goal in this project is to construct models to predict whether a credit card transaction is fraudulent. We'll attempt a supervised learning approach. We'll also create visualizations to help us understand the structure of the data and unearth any interesting patterns. At a high level, an end-to-end view of our ML projects includes the data collection and pipeline, the model itself, and the inferences, which result in the business value. The data pipeline consists generally of (potentially multiple instances) of several processing steps filter, merge, transform, and data storage. The economic value and risk analysis should include the end-to-end process.

3. How the project works

3.1.Data collection

In this project, we analyze a dataset of credit card transactions made over two days in September 2013 by European cardholders. I used the dataset from Kaggle (https://www.kaggle.com/isaikumar/creditcardfraud) to train the machine learning project. The dataset contains 284,807 transactions, of which 492 (0.17%) is fraudulent.

Each transaction has 30 features, all of which are numerical. The features V1 to v28 are the result of a PCA transformation. To protect the confidentiality, background information on these features is not available. The time feature contains the time elapsed since the first transaction, and the amount feature contains the transaction amount. The response variable, the class is 1 in the case of fraud, and 0 otherwise.

Given the class imbalance ratio, we recommend measuring the accuracy using the Area under the Precision-Recall Curve (AUPRC). Confusion matrix accuracy is not meaningful for unbalanced classification.V1 - V28 are the results of a PCA Dimensionality reduction to protect user identities and sensitive features and also other than PCA variables we have amount, time, and class.

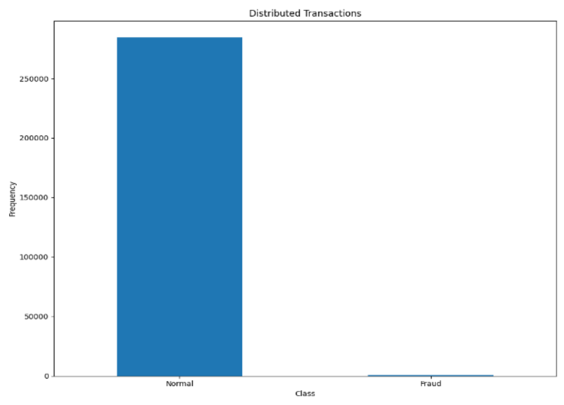
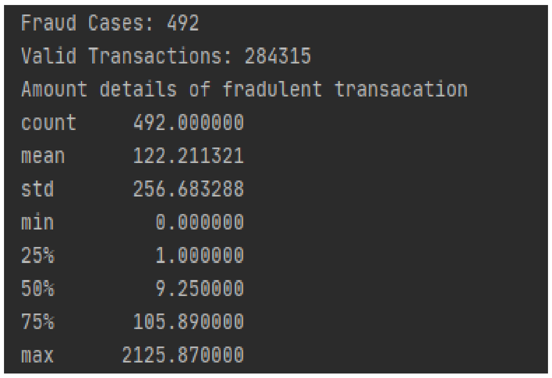


Figure 1:No of fraud cases and non fraud cases



```
Fraud Cases: 492
Valid Transactions: 284315
Amount details of fradulent transacation
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
```

Figure 2:Description of transactions

So from this, I found how many of the data are fraudulent. As you can see there are 284315 normal transactions and 492 fraudulent transactions.

The outlier fraction is 0.00173.so it is almost like zero which means there are very few fraudulent transactions in our dataset. All the details about fraudulent cases can be seen as above.

3.2. Data preparation

Next, I found whether there are any missing values of data. So as there are no missing values this dataset is not needed to perform a cleaning. After the data is cleaned we have to visualize the data of the dataset.

Data visualization pulls datasets out of your big data and visualizes a specific data story to monetize your data, machine learning can bring added value by streamlining your data visualization process for more accurate, predictive, and profitable data. We will visualize the amount, time, and PCA variables to shape data sets into a more defined narrative, giving you a better context for the information you are viewing.

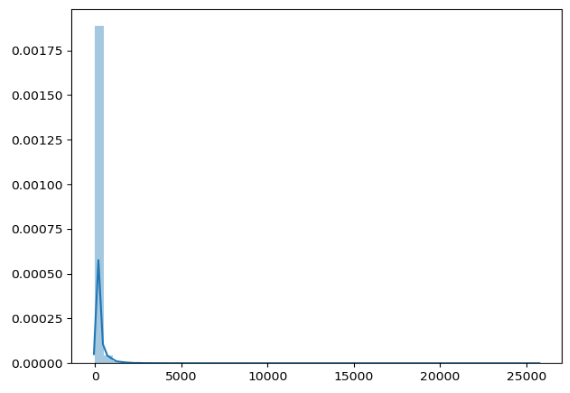


Figure 3:Amount variation

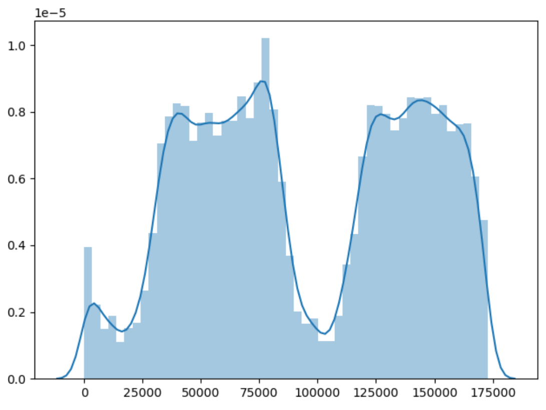


Figure 4:Time variation

The PCA variables or the anomalous features can be visualized as follows. So each variable has a different variation.

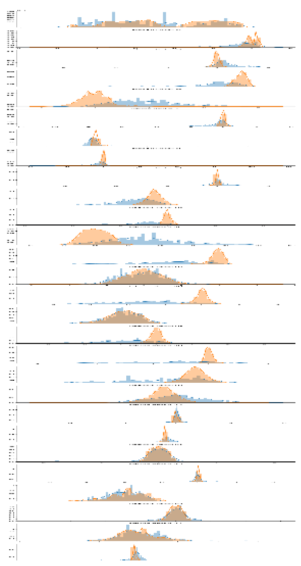


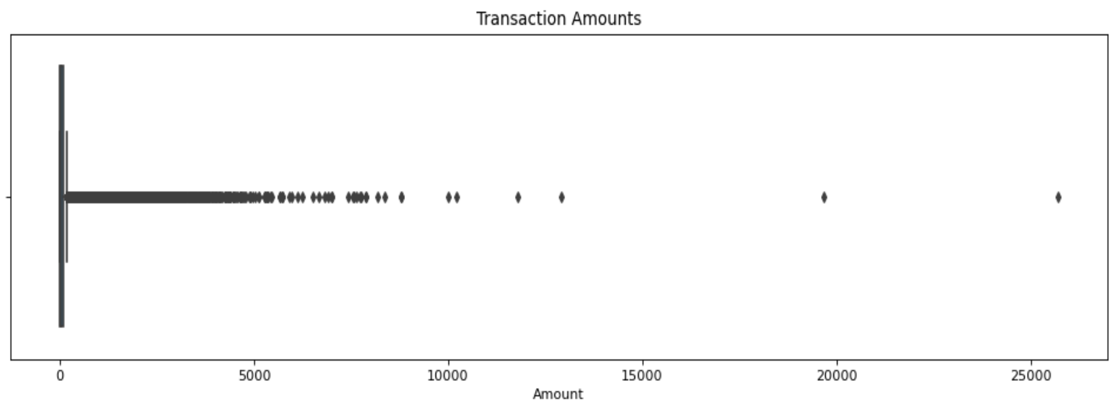
Figure 5:PCA variation

Before we begin preprocessing, we split off a test data set. First split the data into features and response variables. We'll use a test size of 20% and a train set of 80%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions. Using Skicit-learn to split data into training and testing sets. So we have train data set of 227845 and a test data set of 56962.

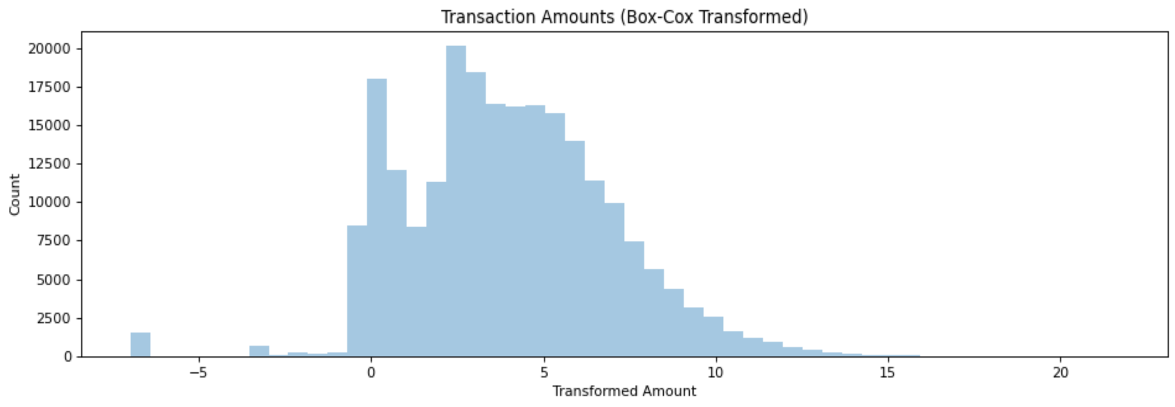
Exploratory data analysis

We conduct exploratory data analysis only on the training set and leave test set unknown. So next we will find descriptive statistics of the variables.

So transactions indeed occur over two days after we convert to days from seconds. Next let 's plot a histogram of transaction times, with one bin per hour.
Comparing the different quantiles, the amounts are very right-skewed. To verify this, plot a histogram of the transaction amounts.

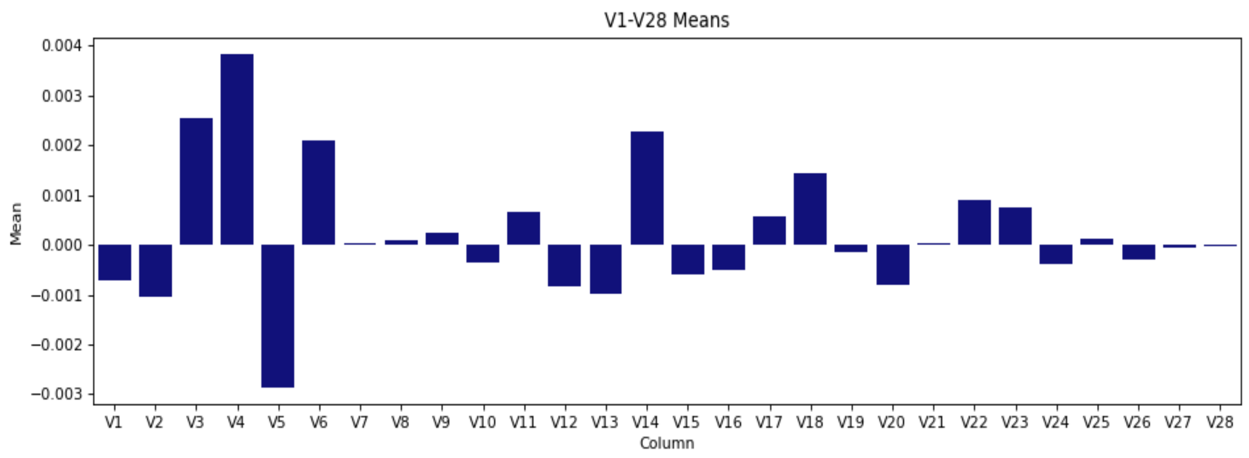


The histogram is hard to read due to some outliers we can't see. A boxplot will show the ou tliers.



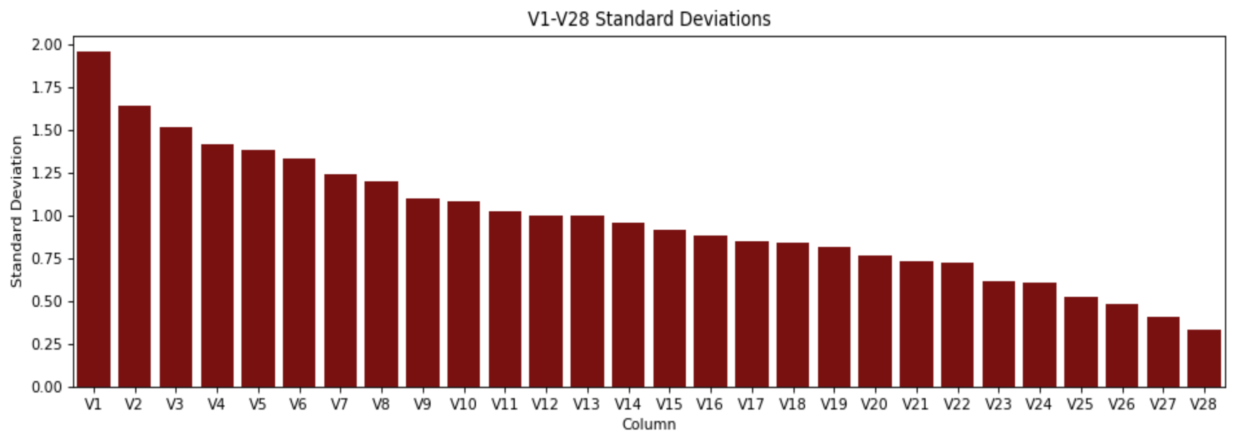
Let's use a power transformer to bring the transaction amounts closer to a normal distribution We'll use the Box-Cox transform in SciPy, but some of the amounts are zero (m in = 0 or above), so we need to shift the amounts first to make them positive. We'll shift by a very small amount, just 10−9. So our power transform removed most of the skewness in t he Amount variable. Now we need to compute the Box-Cox transform on the test data amo unts as well, using the λ value estimated on the training data.

Let's compare the descriptive stats of the PCA variables V1-V28.So let’s visualize the means



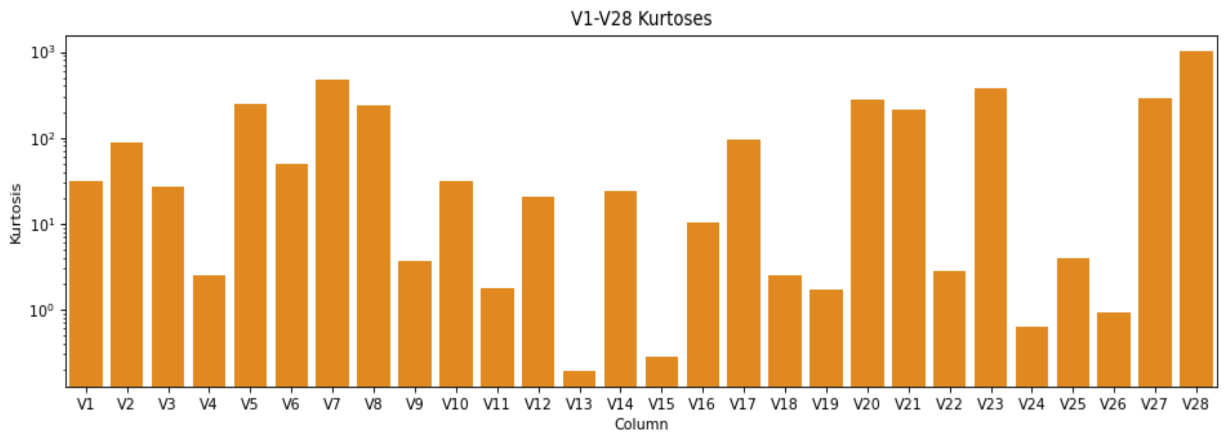
All of V1-V28 have approximately zero means.

Standard deviations among PCA variables can be visualized as follows.

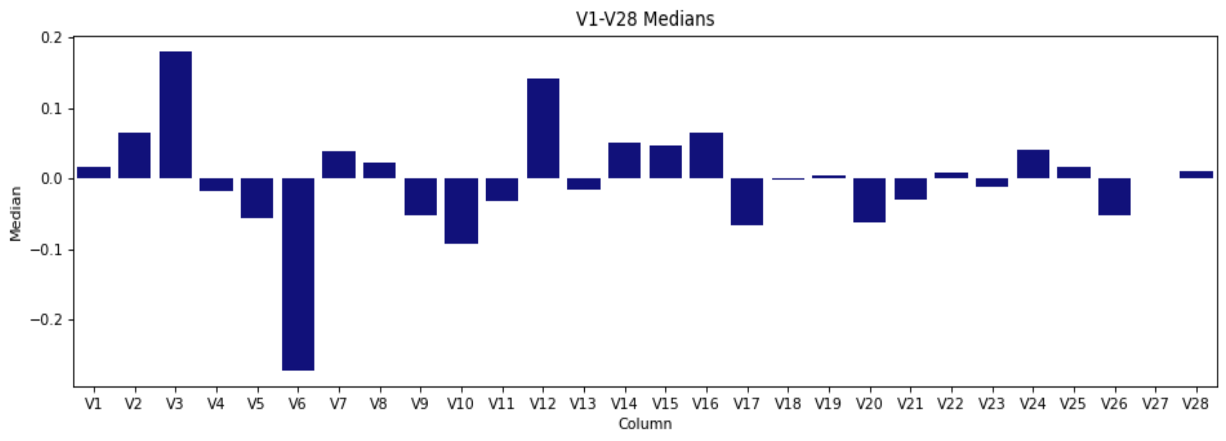


The PCA variables have roughly unit variance, but as low as ~0.3 and as high as ~1.9.

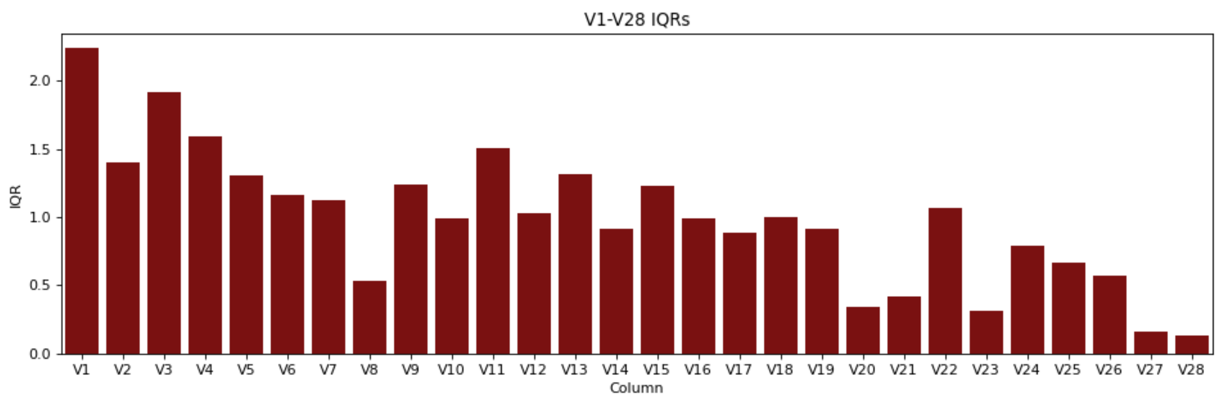
The boxplot is also hard to read due to the large number of outliers, which indicates high k urtosis in V8. This motivates us to plot the kurtoses of the PCA variables. The kurtosis meth od employed in pandas is Fisher's definition, for which the standard normal distribution ha s kurtosis 0.



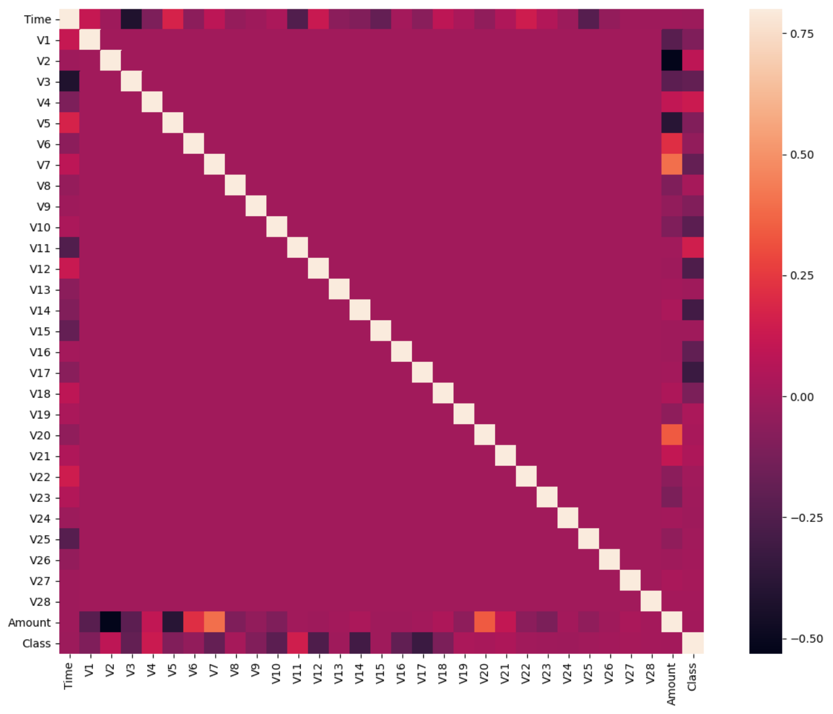
Many of the PCA variables are heavy-tailed. The large numbers of outliers in V1-V28 motiva tes us to consider robust descriptive statistics. Let's plot the medians.



So the IQRs can be visualized as follows. The IQRs of V1-V28 are on a similar scale as the sta ndard deviations as above.



The correlation matrix related to the scenario is,



Now comes the difficult part, making a coaching knowledge set which will permit our algori thms to select up the precise characteristics that build a dealing a lot of or less possible to b e dishonorable. Victimization of the initial knowledge set wouldn't encourage be a decent p lan for a straightforward reason: Since over ninety-nine of our transactions square measure nonfraudulent, AN algorithmic program that invariably predicts that the dealing is non-frau dulent would accomplish AN accuracy more than the ninety-nine. However, that's the alter native to what we wish. We don't need a ninety-nine accuracy that's achieved by ne'er labe ling a dealing as dishonorable, we wish to discover dishonorable transactions and label the m in and of itself.

3.3. Random forest algorithm as the model

I used the random forest algorithm with some tuning techniques for increasing the accurac y of the model.
Random forests or random decision forests are an ensemble learning method for classificat ion, regression, and other tasks that operate by constructing a multitude of decision trees a t training time and outputting the class that is the mode of the classes (classification)/mean prediction (regression) of the individual trees. The training algorithm for RF applies the gen eral technique of bootstrap aggregating, or bagging, to tree learners.

Given a training set X = x1, .. xn with responses Y = y1.. yn, bagging repeatedly (B times) sel ects a random sample with replacement of the training set and fits trees to these samples:
For b = 1, ..., B.

- Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
- Train a classification or regression tree fb on Xb, Yb.

After training is done as above, predictions for unseen samples x' can be made by averagin g the predictions from all the individual regression trees on x'.

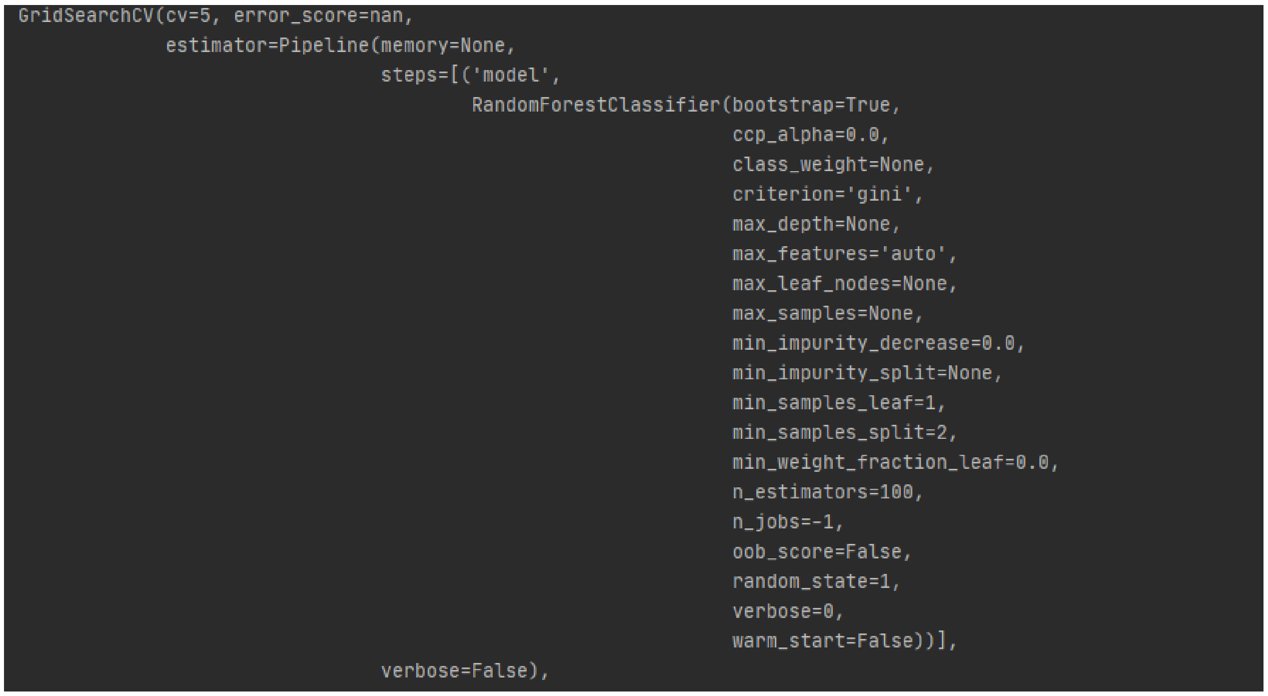
Random decision forests correct for decision trees' habit of overfitting to their training set.

I imported the basic randomforestclassifier() from the sklearn to create my model. We do n ot need to rescale the data for tree-based models, so our pipeline will simply consist of the random forest model. We'll leave the pipeline implementation in place in case if we want t o add preprocessing steps in the future.

The random forest takes much longer to train on this fairly large dataset like this, so we do some hyper-parameter grid search, only specifying the number of estimators.
We'll leave the grid search implemented in case we decide to try different hyper-paramete r values in the future. hyperparameters include the number of decision trees in the forest a nd the number of features considered by each tree when splitting a node. (The parameters of a random forest are the variables and thresholds used to split each node learned during training). Scikit-Learn implements a set of sensible default hyperparameters for all models, but these are not guaranteed to be optimal for a problem.

The best hyperparameters are usually impossible to determine ahead of time, and tuning a model is where machine learning turns from science into trial-and-error based engineering.

After we perform the grid search



```
GridSearchCV(cv=5, error_score=nan,
             estimator=Pipeline(memory=None,
                                steps=[('model',
                                        RandomForestClassifier(bootstrap=True,
                                                               ccp_alpha=0.0,
                                                               class_weight=None,
                                                               criterion='gini',
                                                               max_depth=None,
                                                               max_features='auto',
                                                               max_leaf_nodes=None,
                                                               max_samples=None,
                                                               min_impurity_decrease=0.0,
                                                               min_impurity_split=None,
                                                               min_samples_leaf=1,
                                                               min_samples_split=2,
                                                               min_weight_fraction_leaf=0.0,
                                                               n_estimators=100,
                                                               n_jobs=-1,
                                                               oob_score=False,
                                                               random_state=1,
                                                               verbose=0,
                                                               warm_start=False))],
                                verbose=False),
```

All the nodes, except the leaf nodes (colored terminal nodes), have 5 parts:

- The question asked about the data based on the value of a feature.
  Each question has either a True or False answer that splits the node.
  Based on the answer to the question, a data point moves down the tree.
- The Gini Impurity of the node.
  The average weighted Gini Impurity decreases as we move down the tree.
- Samples or The number of observations in the node.
- Value or The number of samples in each class.
  For example, the top node has 2 samples in class 0 and 4 samples in class 1.
- Class or the majority classification for points in the node.
  In the case of leaf nodes, this is the prediction for all samples in the node.

The random forest algorithm is combined with various features to increase the accuracy of the model. We can see the parameters such as max depth, number of estimators, class, the criterion, etc.

CV determines the cross-validation splitting strategy used in cross_val_predict to train final _est-imator. None, to use the default 5-fold cross-validation. Criterian is important because it’s a function to measure the quality of a split. In_samples_split counts the minimum num ber of samples required to split an internal node. Supported criteria are “gini” for the Gini i mpurity and “entropy” for the information gain.
Gini Importance or Mean Decrease in Impurity (MDI) calculates each feature's importance a s the sum over the number of splits(across all tress)that include the feature, proportionally to the number of samples it splits.

Gini impurity

The Gini Impurity of a node is the probability that a randomly chosen sample in a node woul d be incorrectly labeled if it was labeled by the distribution of samples in the node. For exam ple, in the top (root) node, there is a 44.4% chance of incorrectly classifying a data point cho sen at random based on the sample labels in the node. We arrive at this value using the foll owing equation

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

The Gini Impurity of a node n is 1 minus the sum over all the classes J(for binary classificatio n task this is 2) of the fraction of examples in each class pi squared.

Adding one further step of randomization yields extremely randomized trees. While similar to ordinary random forests in that they are an ensemble of individual trees, there are two main differences: first, each tree is trained using the whole learning sample (rather than a b ootstrap sample), and second, the top-down splitting in the tree learner is randomized. Inst ead of computing the locally optimal cut-point for each feature under consideration.
This algorithm has an overfitting feature to get the maximum accuracy of the model.

Overfitting feature of random forest algorithm

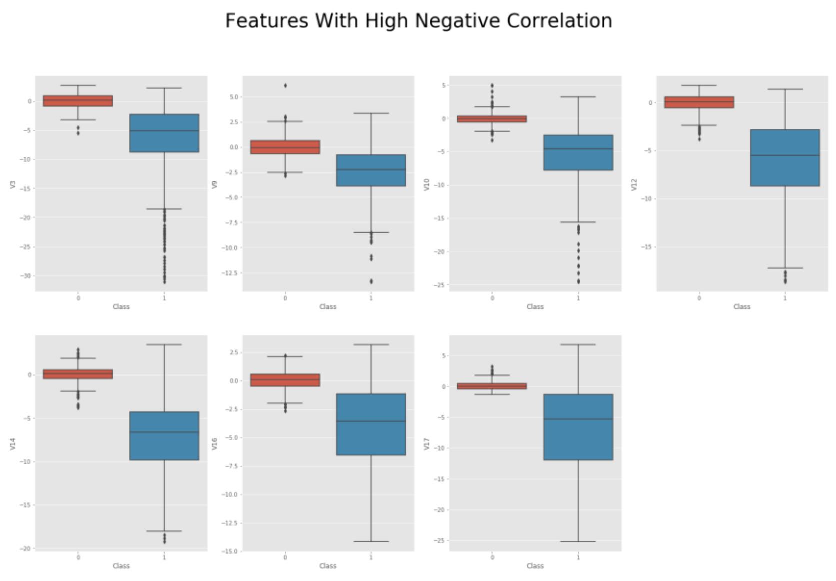
Overfitting occurs when we have a very flexible model which essentially memorizes the tra ining data by fitting it closely. The problem is that the model learns not only the actual relat ionships in the training data but also any noise that is present.
A flexible model is said to have high variance because the learned parameters (such as the structure of the decision tree) will vary considerably with the training data.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic. bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

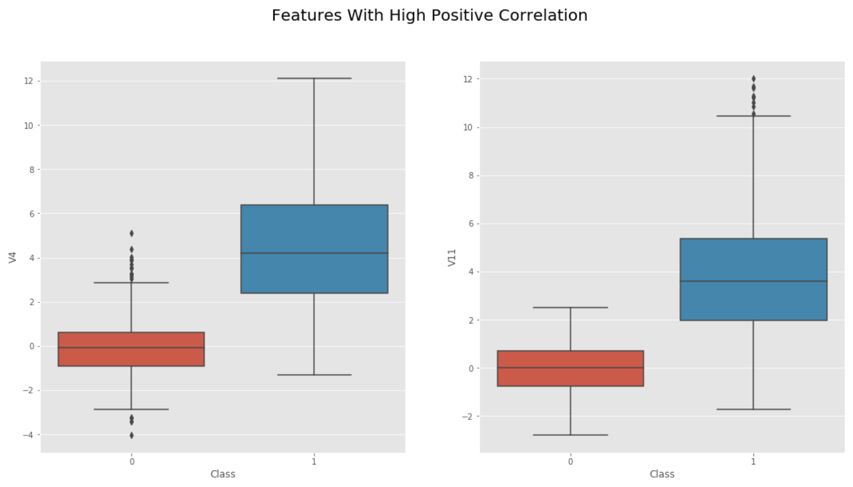
Additionally, an estimate of the uncertainty of the prediction can be made as to the standard deviation of the predictions from all the individual regression trees on *x′*.

Outlier Detection & Removal Outlier detection

Outlier Detection & Removal Outlier detection is a complex topic. The trade-off between reducing the number of transactions and thus volume of information available to my algorithms and having extreme outliers skew the results of your predictions is not easily solvable and highly depends on your data and goals. In my case, I decided to focus exclusively on features with a correlation of 0.5 or higher with the class variable for outlier removal. Before getting into the actual outlier removal, let’s take a look at the visualizations of those features.

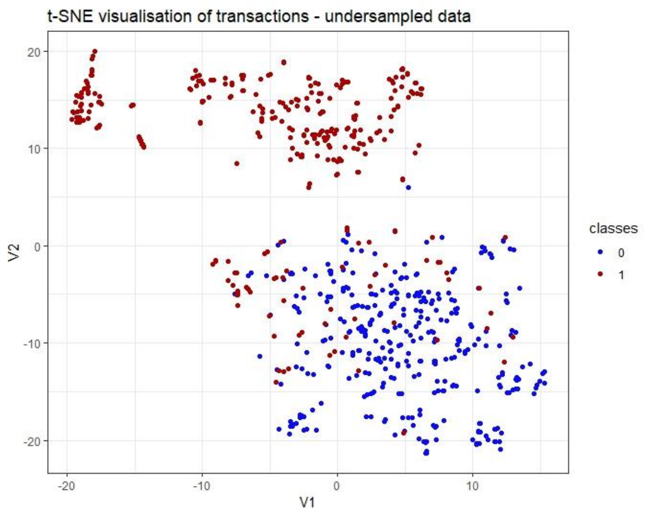


The positive correlation can be plot as follows.



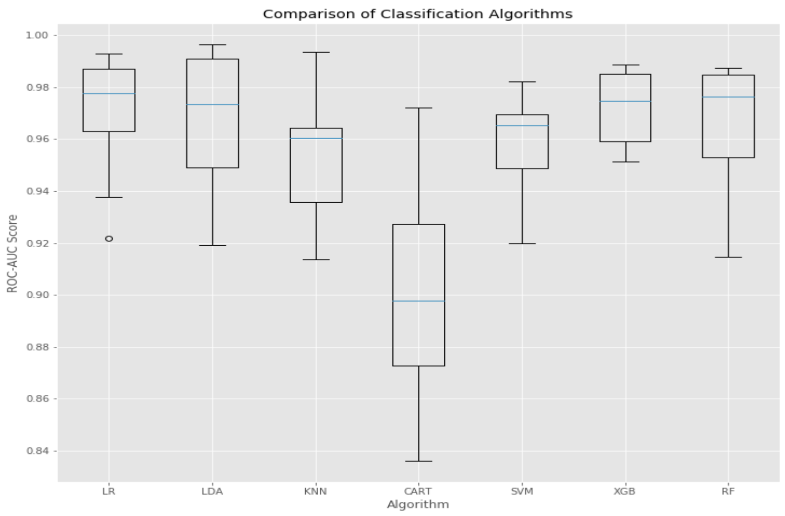
Dimensionality Reduction with t-SNE for Visualization

Visualizing our categories would influence be quite attention-grabbing and show if they're severable. However, it's unfeasible to provide 30-dimensional plot exploitation of all of our predictors. Instead, employing a spatiality reduction technique like t-SNE, we have a tendency to area unit able to project these higher dimensional distributions into lower-dimensional visualizations. For this project, I decided to use t-SNE, AN rule that I had not been operating with before. If you'd prefer to apprehend additional concerning however this rule works, see here. Projecting our information set into a two-dimensional area, we have a tendency to area unit able to manufacture a scatter plot showing the clusters of dishonest and non-fraudulent transactions.



Best model

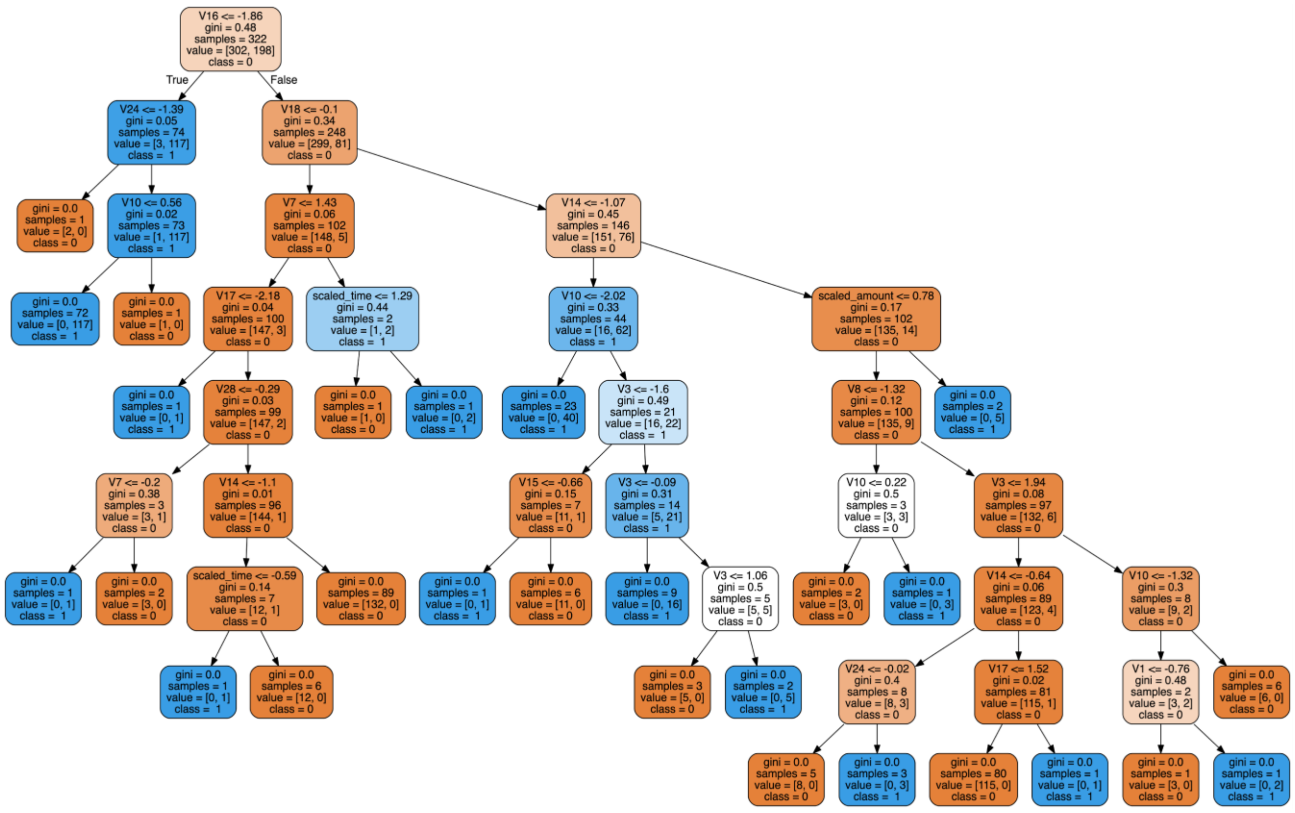
To avoid overfitting, I used the quite common resampling technique of k-fold cross-validati on. This merely means you separate your coaching knowledge into k elements (folds) and s o suit your model on k-1 folds before creating predictions for the kth hold-out fold.
You then repeat this method for every single fold and average the ensuing predictions. The results can be visualized as follows.



As we can see, there are a few algorithms that quite significantly outperformed the others. Now, what algorithm do we choose? As mentioned above, this project had not only the foc us of achieving the highest accuracy but also to create business value. Therefore, choosing Random Forest over XGBoost might be a reasonable approach to achieve a higher degree o f comprehensiveness while only slightly decreasing performance.

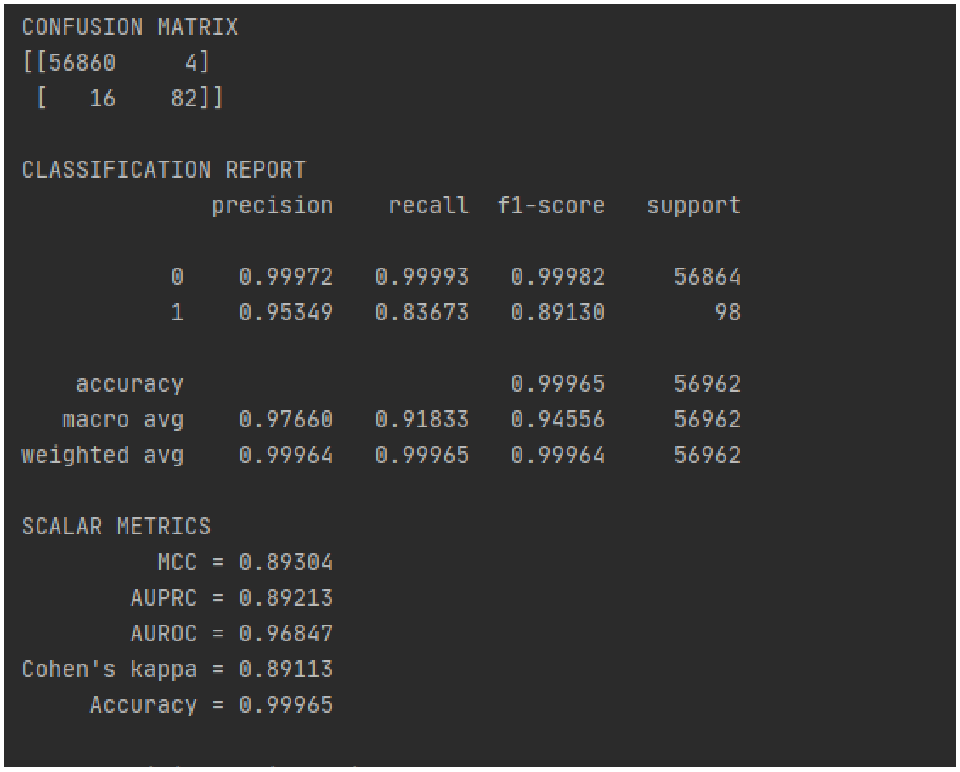
To further illustrate what I mean by this, here is a visualization of our Random Forest mode l that could easily be used to explain very simply why a certain decision was made.
So we can classify the random forest related to credit card fraud detection as follows.

Credit card Fraud detection system


V16 <= -1.86 gini = 0.48 samples = 322 value = [302, 198] class = 0
True
False
V24 <= -1.39 gini = 0.05 samples = 74 value = [3, 117] class = 1
V18 <= -0.1 gini = 0.34 samples = 248 value = [299, 81] class = 0
gini = 0.0 samples = 1 value = [2, 0] class = 0
V10 <= 0.56 gini = 0.02 samples = 73 value = [1, 117] class = 1
V7 <= 1.43 gini = 0.06 samples = 102 value = [148, 5] class = 0
V14 <= -1.07 gini = 0.45 samples = 146 value = [151, 76] class = 0
gini = 0.0 samples = 72 value = [0, 117] class = 1
gini = 0.0 samples = 1 value = [1, 0] class = 0
V17 <= -2.18 gini = 0.04 samples = 100 value = [147, 3] class = 0
scaled_time <= 1.29 gini = 0.44 samples = 2 value = [1, 2] class = 1
V10 <= -2.02 gini = 0.33 samples = 44 value = [16, 62] class = 1
scaled_amount <= 0.78 gini = 0.17 samples = 102 value = [135, 14] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
V28 <= -0.29 gini = 0.03 samples = 99 value = [147, 2] class = 0
gini = 0.0 samples = 1 value = [1, 0] class = 0
gini = 0.0 samples = 1 value = [0, 2] class = 1
gini = 0.0 samples = 23 value = [0, 40] class = 1
V3 <= -1.6 gini = 0.49 samples = 21 value = [16, 22] class = 1
V8 <= -1.32 gini = 0.12 samples = 100 value = [135, 9] class = 0
gini = 0.0 samples = 2 value = [0, 5] class = 1
V7 <= -0.2 gini = 0.38 samples = 3 value = [3, 1] class = 0
V14 <= -1.1 gini = 0.01 samples = 96 value = [144, 1] class = 0
V15 <= -0.66 gini = 0.15 samples = 7 value = [11, 1] class = 0
V3 <= -0.09 gini = 0.31 samples = 14 value = [5, 21] class = 1
V10 <= 0.22 gini = 0.5 samples = 3 value = [3, 3] class = 0
V3 <= 1.94 gini = 0.08 samples = 97 value = [132, 6] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 2 value = [3, 0] class = 0
scaled_time <= -0.59 gini = 0.14 samples = 7 value = [12, 1] class = 0
gini = 0.0 samples = 89 value = [132, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 6 value = [11, 0] class = 0
gini = 0.0 samples = 9 value = [0, 16] class = 1
V3 <= 1.06 gini = 0.5 samples = 5 value = [5, 5] class = 0
gini = 0.0 samples = 2 value = [3, 0] class = 0
gini = 0.0 samples = 1 value = [0, 3] class = 1
V14 <= -0.64 gini = 0.06 samples = 89 value = [123, 4] class = 0
V10 <= -1.32 gini = 0.3 samples = 8 value = [9, 2] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 6 value = [12, 0] class = 0
gini = 0.0 samples = 3 value = [5, 0] class = 0
gini = 0.0 samples = 2 value = [0, 5] class = 1
V24 <= -0.02 gini = 0.4 samples = 8 value = [8, 3] class = 0
V17 <= 1.52 gini = 0.02 samples = 81 value = [115, 1] class = 0
V1 <= -0.76 gini = 0.48 samples = 2 value = [3, 2] class = 0
gini = 0.0 samples = 6 value = [6, 0] class = 0
gini = 0.0 samples = 5 value = [8, 0] class = 0
gini = 0.0 samples = 3 value = [0, 3] class = 1
gini = 0.0 samples = 80 value = [115, 0] class = 0
gini = 0.0 samples = 1 value = [0, 1] class = 1
gini = 0.0 samples = 1 value = [3, 0] class = 0
gini = 0.0 samples = 1 value = [0, 2] class = 1

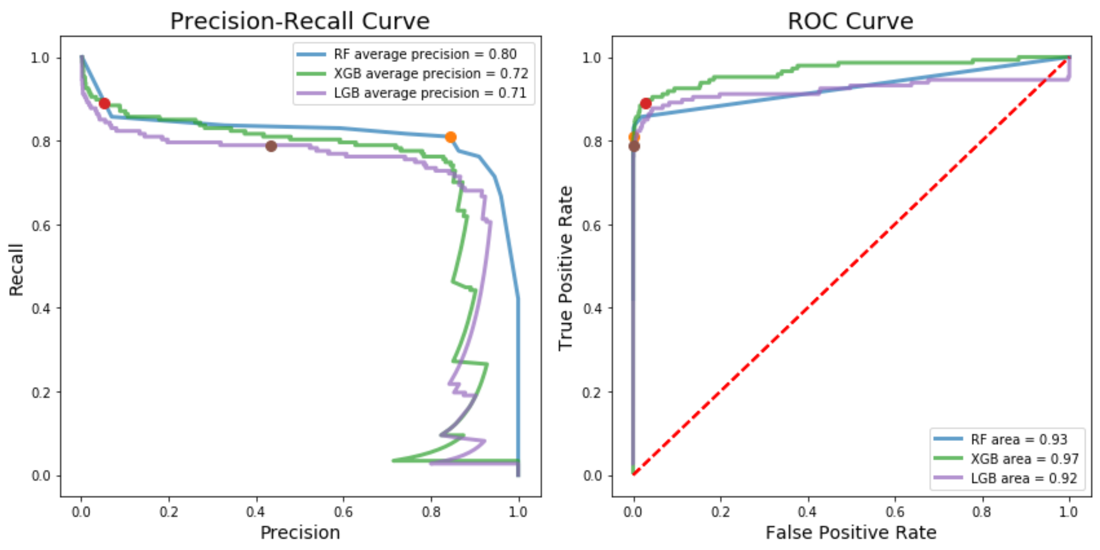
3.4. Test and evaluate the model

According to the cross-validated MCC scores, the random forest is one of the best-performing model, so now let's evaluate its performance on the test set.



```
CONFUSION MATRIX
[[56860     4]
 [   16    82]]

CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0    0.99972   0.99993   0.99982     56864
           1    0.95349   0.83673   0.89130        98

    accuracy                        0.99965     56962
   macro avg    0.97660   0.91833   0.94556     56962
weighted avg    0.99964   0.99965   0.99964     56962

SCALAR METRICS
          MCC = 0.89304
        AUPRC = 0.89213
        AUROC = 0.96847
Cohen's kappa = 0.89113
     Accuracy = 0.99965
```

Speaking of performance, we have a tendency to don't seem to be about to have confidence accuracy. Instead, we have a tendency to ar} about to build use of the Receiver in operation Characteristics-Area beneath the Curve or ROC-AUC performance measure (I have connected any reading below this article). Primarily, the ROC-AUC outputs a price between zero and one, whereby one could be an excellent score and nil the worst. If AN algorithmic program encompasses a ROC-AUC score of higher than zero.5, it's achieving a better performance than random estimation.


Precision-Recall Curve
RF average precision = 0.80
XGB average precision = 0.72
LGB average precision = 0.71
Recall
Precision
ROC Curve
RF area = 0.93
XGB area = 0.97
LGB area = 0.92
True Positive Rate
False Positive Rate

4. Conclusion

Fraud detection is a complex issue that requires a substantial amount of planning before throwing machine learning algorithms at it. Nonetheless, it is also an application of data science and machine learning for the good, which makes sure that the customer’s money is safe and not easily tampered with. Having a data set with non-anonymized features would make this particularly interesting as outputting the feature importance would enable one to see what specific factors are most important for detecting fraudulent transactions.