Credit Card Fraud Detection

ML model using Random Forest Classification and XG Boost

THE ISSUE

As technology is drastically growing, we all are shifting yo e-commerce, e-pay and so surveys say that 90% of the people have started e-payments or online transactions, say Paytm, Gpay or any other random surfed site of clothing or groceries. Hackers got their hand red in it, and can easily hack into the local sites to get the credit card informations, and we never know how much lossy it could be!

A Tool which is today’s necessity

- The major problem nowadays is the misusing of the data, the cyber crimes that are tremendously increasing. As the Hacker’s are getting their hand black in this field, the chances of data breaching, leaking or misusing again drastically increases.

- Then comes this special type of crime of credit card frauds, as everybody is switching to net banking and online payment methods, it is a bit risky for the people who do not know the tactics of frauds.

- It is very important for us nowadays, to at least make sure that these types of frauds do not happen. And so we have this deployed model for the public use.

Solution

- We are using Deep learning techniques and Algorithms to train a model, which will help a user to identify whether the transaction was fraudulent or not based on some regress patterns.

- Also We have taken the help of AWS (amazon web services), The fraud detector that AWS provides is well trained from 20 years of data, and so it is quite accurate to rely upon.
- The deployed model can be used as a tool by the end users.

Solution 1: AWS - The Fraud Detector

- Amazon Fraud Detector is a fully managed service that makes it easy to identify potentially fraudulent online activities such as online payment fraud and the creation of fake accounts.
- Amazon Fraud Detector uses user’s data, machine learning (ML), and more than 20 years of fraud detection expertise from Amazon to automatically identify potentially fraudulent online activity so we can catch more fraud faster.
- So we have created our model, trained it and tested it on their platform.
- The process includes-

A. Model creation, Detector creation

B. Uploading the data on which one have to check the frauds

C. User can add the rules as well, according to his/her wants and needs

D. Lastly, Deployment and we have publish it.

Dataset Analysis

- 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 numeric 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.

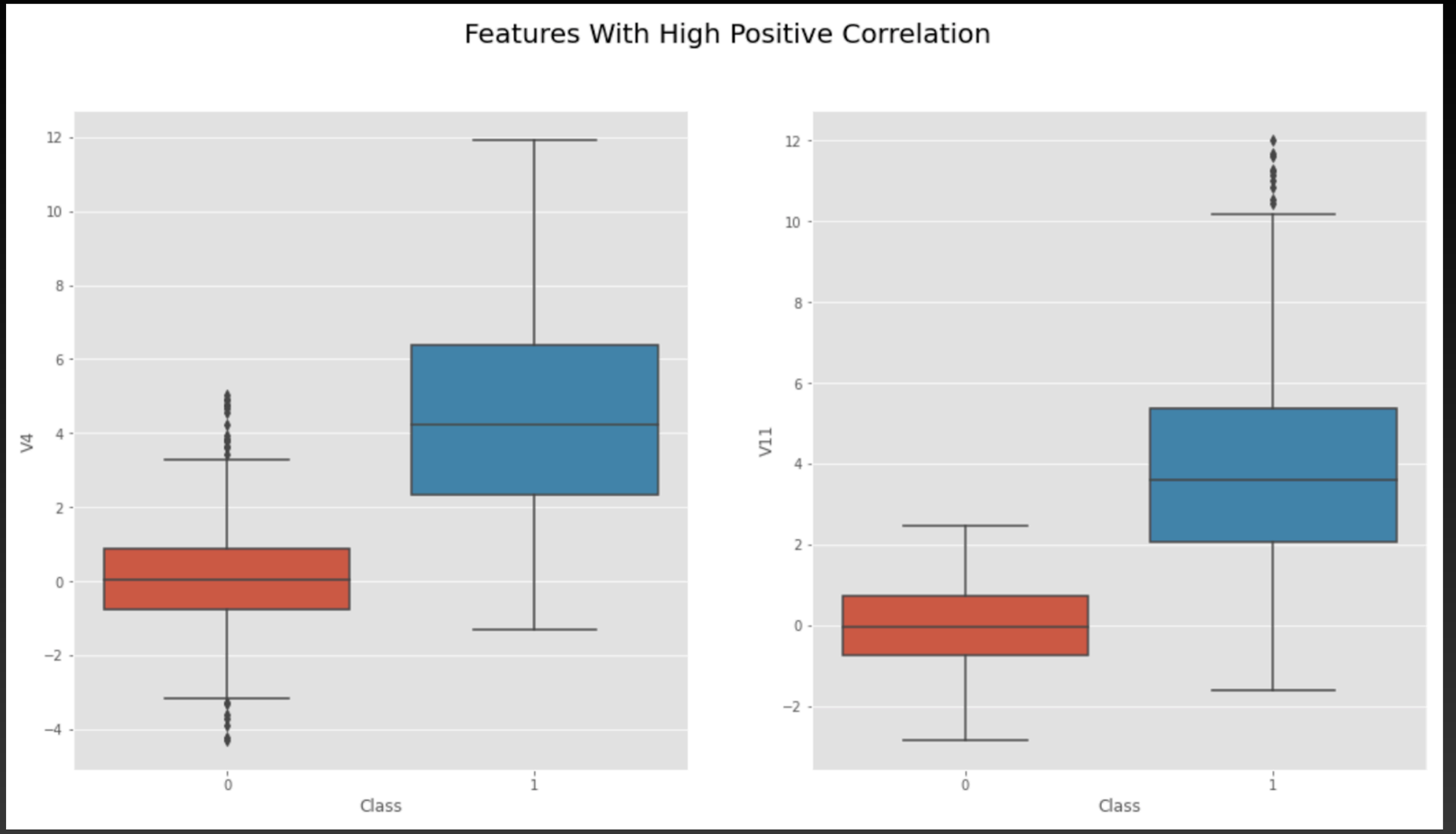
Data Preprocessing

- Feature scaling is applied to the dataset so that all the features have values in same scale and the training process is efficient.
- The output and input are separated from the dataset given. The column “Class” which takes binary values (0/1) based on whether the transaction is fraudulent (1) or not (0), is stored in y. Whereas, rest of the columns are stored in X.
- X and y are split into (X_{train}, y_{train}), (X_{test}, y_{test}) and ($X_{crossval}$, $y_{crossval}$) in suitable proportions.
- The model is trained on (X_{train}, y_{train}) with various algorithms.

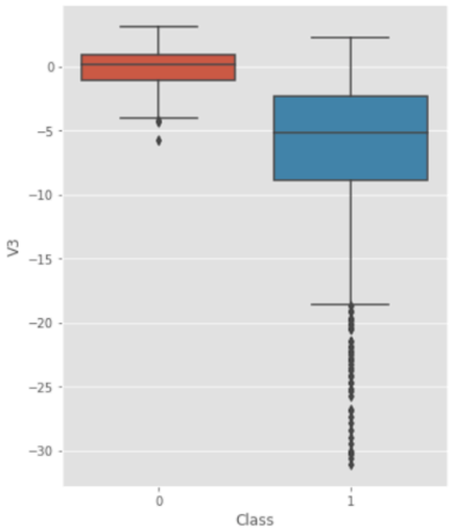
Data Modification

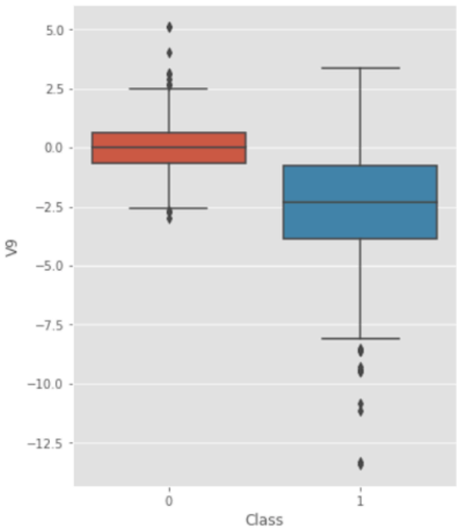
(creating a Training Set from a Heavily Imbalanced Data Set)

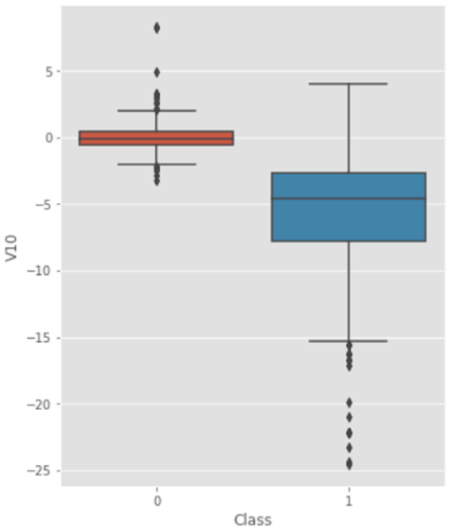
- The dataset is analysed. The number of fraudulent transactions vs the non - fraudulent ones are found out.
- Also, since our input data is sensitive, the features are encrypted using PCA. So, we find the correlation of each of these features, that is how much do these contribute for a transaction to be fraudulent.
- Using the above correlation data, we find the features with high positive and negative correlation which in turn helps us to find out the outliers.
- Presence of outliers may lead to incorrect predictions. Hence, we remove the outliers and find that our dataset size is now reduced.
- Dimensionality reduction is done for the features obtained by PCA.
- t-SNE algorithm is used to do the dimensionality reduction.
-

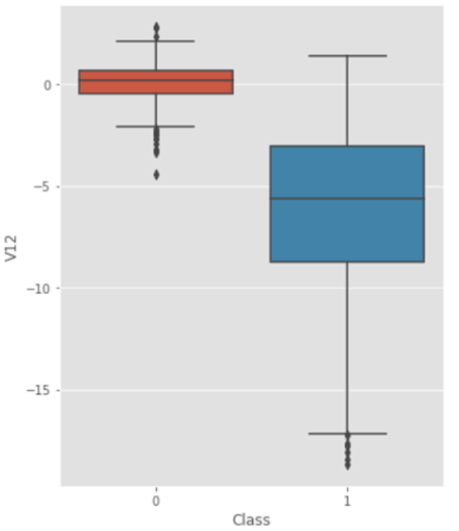

Features With High Positive Correlation
V4
12
10
8
6
4
2
0
−2
−4
0
1
Class
V11
12
10
8
6
4
2
0
−2
0
1
Class

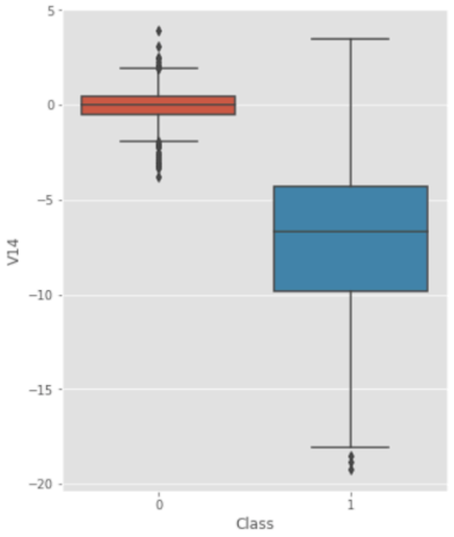
Features With High Negative Correlation

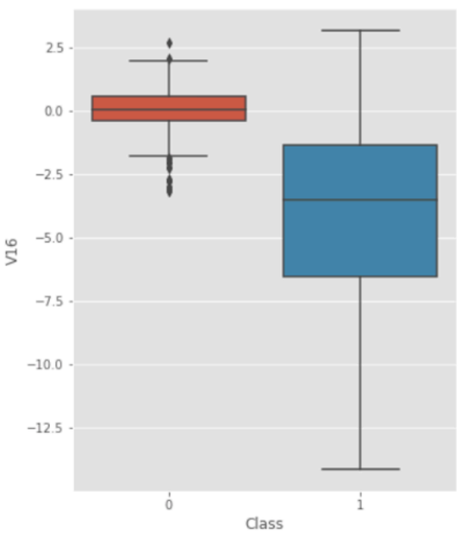


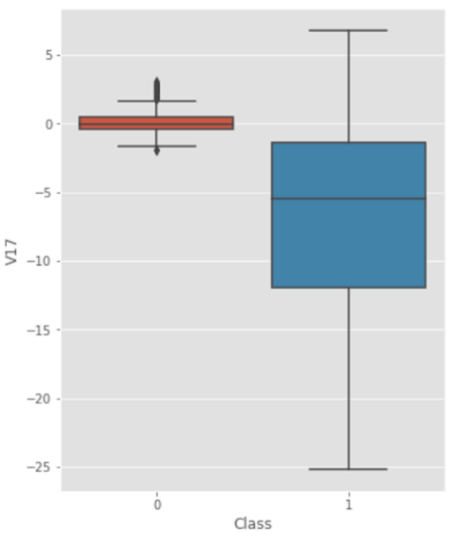












Selecting the best suitable model

- We train the model with the preprocessed clean data and fit various algorithms like random forest classification, logistic regression, naive bayes, etc. to it.
- Predictions are made on the test set and test set accuracy is calculated in each case.
- The model in which test set accuracy is comparatively higher and the number of false negatives are less is finally fit to the model and is used to make real life predictions.
- And so we have compared our model’s accuracy with all these different algorithms.

Test accuracy comparison of classification algorithms

localhost:8888/notebooks/creditcardfraud.ipynb

jupyter creditcardfraud Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
models.append(('SVM', SVC()))
models.append(('XGB', XGBClassifier()))
models.append(('RF', RandomForestClassifier()))

#testing models

results = []
names = []

for name, model in models:
    kfold = KFold(n_splits=10, random_state=42)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='roc_auc')
    results.append(cv_results)
    names.append(name)
    msg = '%s: %f (%f)' % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

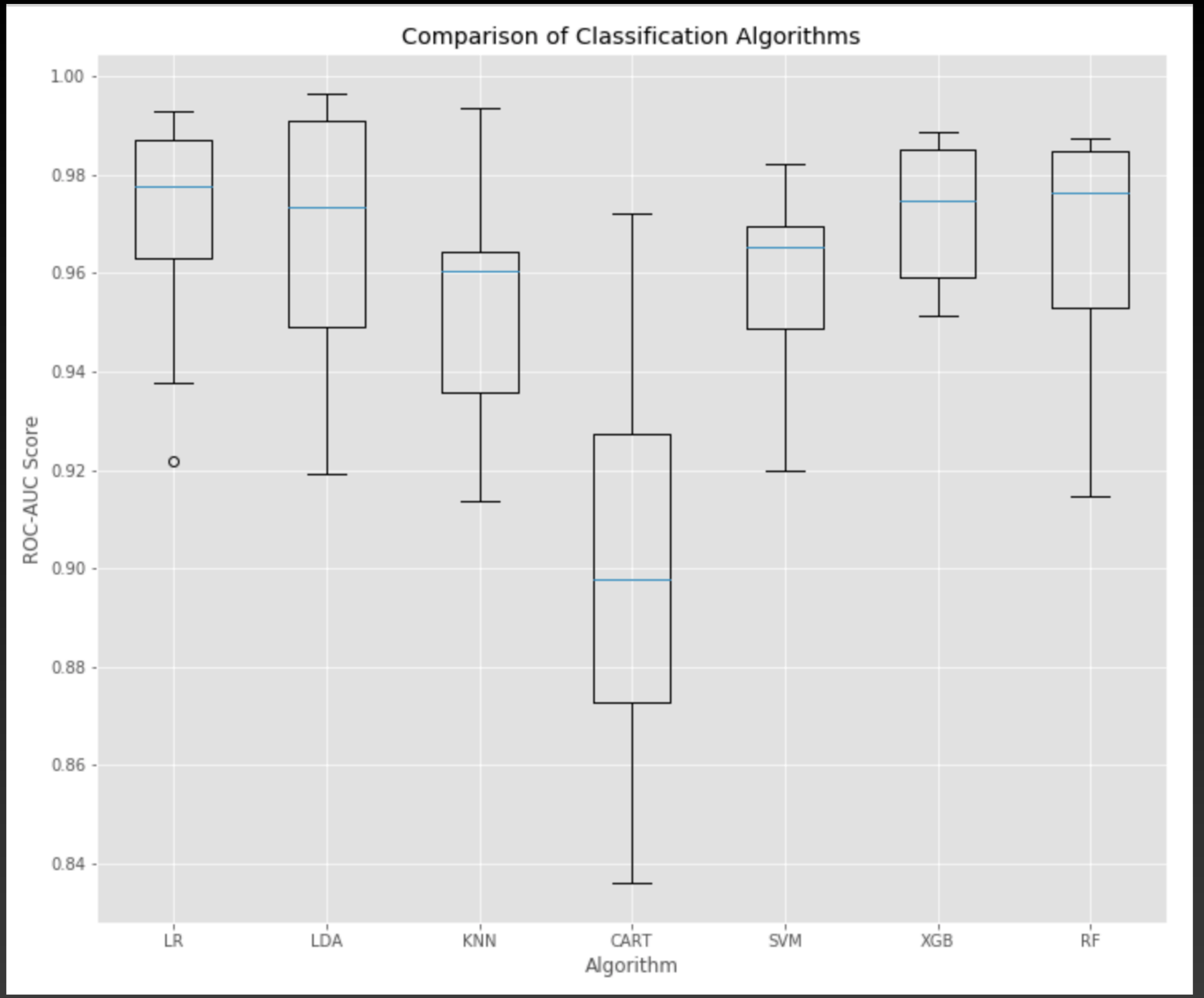
```
LR: 0.968559 (0.021046)
LDA: 0.971412 (0.020097)
KNN: 0.967261 (0.022617)
CART: 0.899285 (0.043165)
SVM: 0.971311 (0.019924)
XGB: 0.962028 (0.026296)
RF: 0.969424 (0.019367)
```

In [47]:

```
#Compare Algorithms

fig = plt.figure(figsize=(12,10))
plt.title('Comparison of Classification Algorithms')
plt.xlabel('Algorithm')
plt.ylabel('ROC-AUC Score')
plt.boxplot(results)
ax = fig.add_subplot(111)
ax.set_xticklabels(names)
plt.show()
```

We concluded that will get the best accuracy from Random Forest Classifier and XGboost classifier.

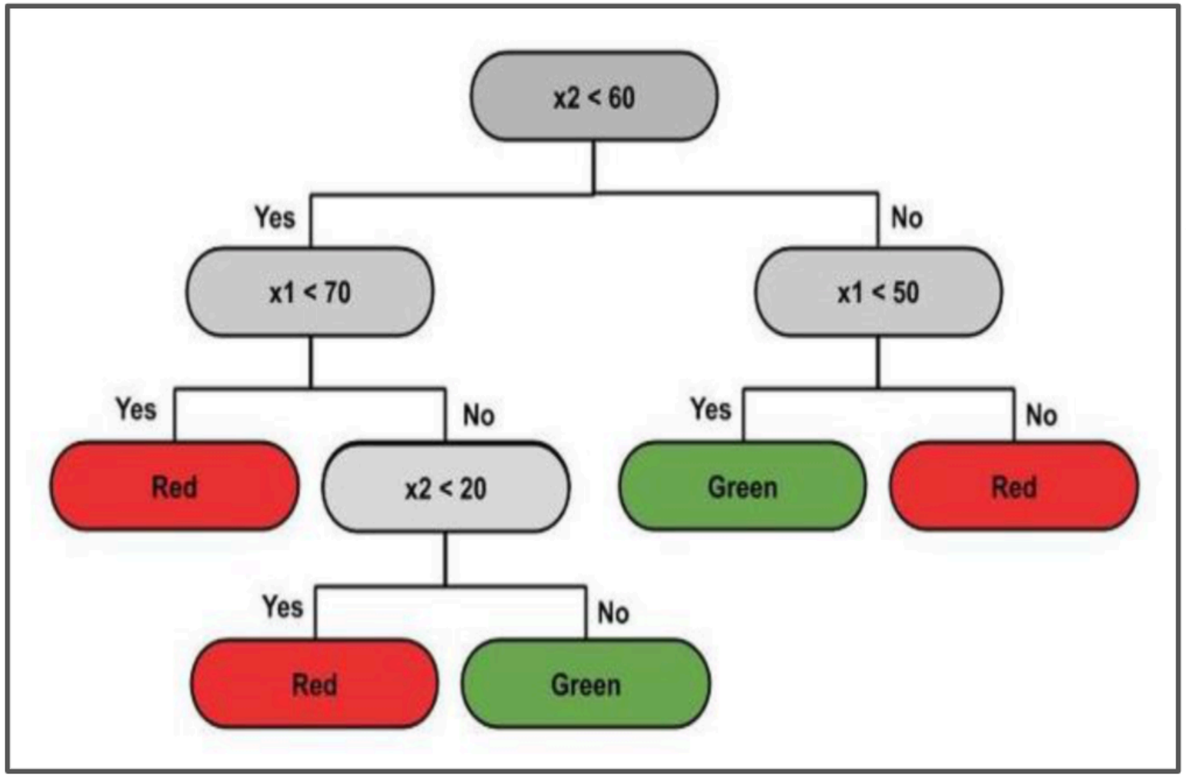

Comparison of Classification Algorithms
ROC-AUC Score
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
LR
LDA
KNN
CART
SVM
XGB
RF
Algorithm

RF Classification - How does it work ?

- Random Forest Classification is an example of Ensemble learning, where multiple machine learning algorithms are put together to create one bigger and better performance ML algorithm.

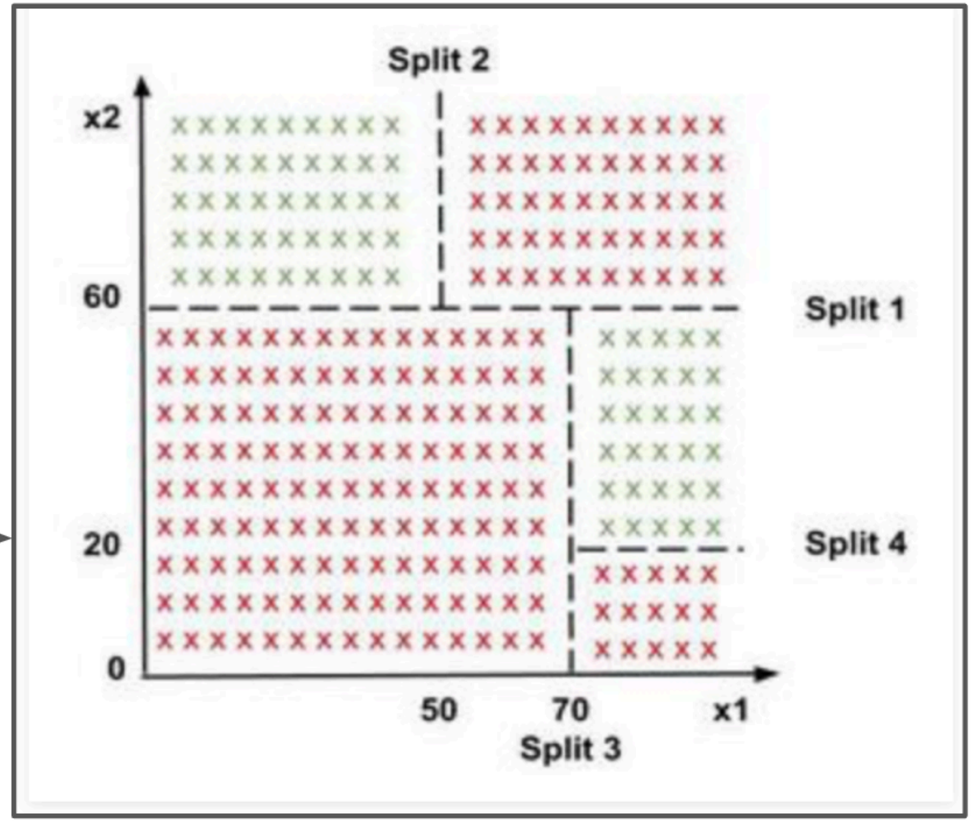
- We randomly pick ‘k’ data points from the training set, build a decision tree associated with these k points. Then, we choose the number of trees ‘n’ we want to build and repeat. For a new data point, we take the predictions of each of the ‘n’ decision trees and and assign it to the majority vote category.

- A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute (a condition), each branch represents an outcome of the test (True or False), and each leaf node (terminal node) holds a class label. Based on this tree, splits are made to differentiate classes in the original dataset given.



Decision Tree

Classification using the decision tree constructed

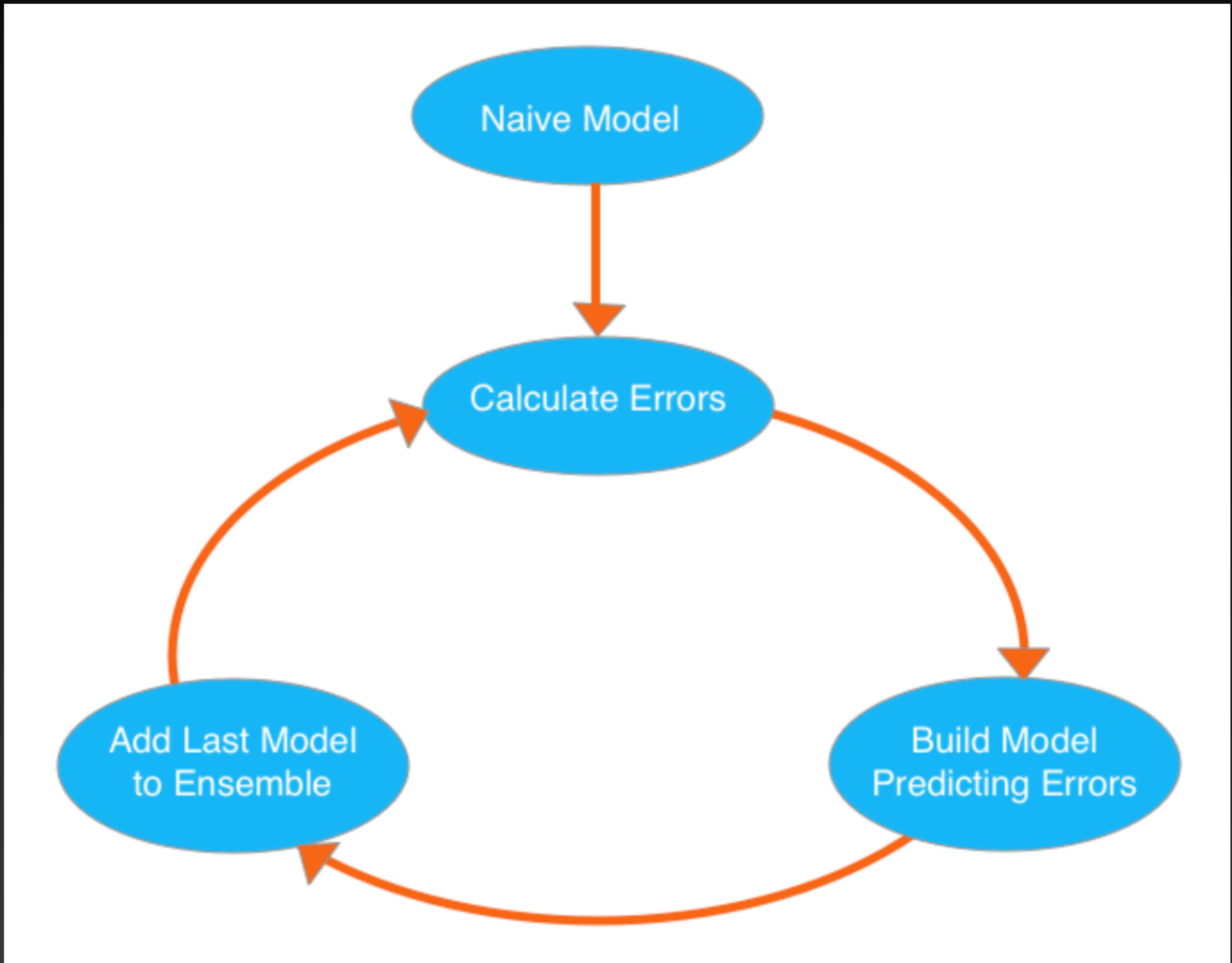


XG - Boost

- XGBoost is an implementation of the Gradient Boosted Decision Trees algorithm.

- We go through cycles that repeatedly builds new models and combines them into an ensemble model. We start the cycle by calculating the errors for each observation in the dataset. We then build a new model to predict those. We add predictions from this error-predicting model to the "ensemble of models.”
- To make a prediction, we add the predictions from all previous models. We can use these predictions to calculate new errors, build the next model, and add it to the ensemble.
- XG - Boost gives high quality performance on large datasets with fast execution speed.

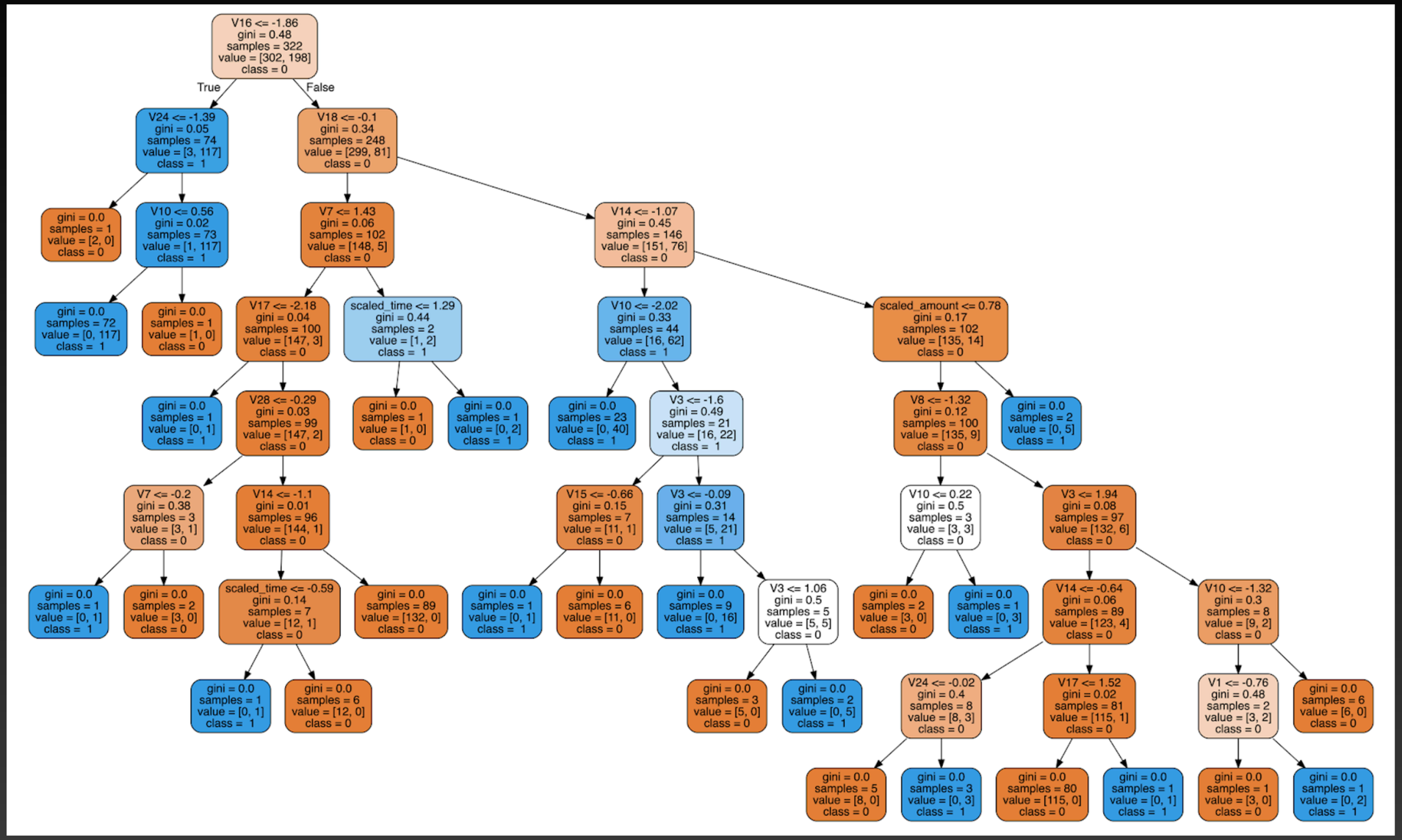
Working of XG - Boost



Training the model with RF and XG Boost

- As found, the most efficient algorithm for given dataset is Random Forest Classification and XG Boost. Therefore, the model is trained on (X_{train}, y_{train}) and the above classifiers are fit to it.
- Once, the model is ready, predictions are made on the test set and graphs are plotted to demonstrate the error or loss function (a measure of difference between the actual and predicted value - usually Root mean square error is used)
- The test accuracy of RF Classification using XG Boost is found out to be - 96.94% which is pretty good.
- Brief working of these algorithms is explained in the upcoming slides.

Visualising the RF model applied to our dataset



Deployment of the model

- The major part in any Deep learning and machine learning project is the part of Model Deployment. So that the end users or the naive users can take the advantage of what the solution in real is.
- So, we have used Amazon web Services for the model deployment part.
- The Deployed model will work as a tool for the users and they can easily and accurately check whether the transactions were fraud or not.
- One thing which every project and solution search for is - Security, and it is 100% secured to use.