Fraud Detection in Credit Card Transactions

May 2022

Fraud Detection in Credit Card Transactions

Submitted in partial fulfillment of the requirements

for the award of degree of

Bachelor of Technology

In

Computer Science and Engineering

Table 1: Submitted By

Roll No	*NameOfStudent*
39/CSE/18105/416	*VivekKumarYadav*
39/CSE/18036/347	*KunalGupta*

Under the guidance of

Dr. Bhaskar Biswas

Department of Computer Science and Engineering

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KALYANI

Kalyani, Nadia, West Bengal- 741235

Spring Semester 2022

Contents

Certificate

This is to certify that the Project entitled "**Fraud Detection in Credit Card Transactions** being submitted by **Mr. Vivek Kumar Yadav ,Kunal Gupta** , an undergraduate student in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, India, for the award of **Bachelor of Technology** in **Computer Science and Engineering**,

is an genuine research work carried by him under my supervision and guidance. The project has fulfilled all the requirements as par the regulation of **IIIT Kalyani** and in my opinion, has reached the standards needed for submission. The works, techniques and the results presented have not been submitted by any other student around Us.

(**Dr. Bhaskar Biswas, Ph.D**)
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Information Technology Kalyani
Kalyani, W.B.-741235, India.

Acknowledgments

First of all, I would like to take this opportunity to thank my supervisor **Dr. Bhaskar Biswas** without whose effort this project would not have been possible. I am so grateful to him for working tirelessly after me, answering my doubts whenever and wherever possible. I am most grateful to Department of Computer Science and Engineering, IIIT Kalyani, India, for providing me this wonderful opportunity to complete my bachelor project.

And last but the biggest of all, I want to thank my parents, for always believing in me and letting me do what I wanted, but keeping a continuous check that I never wandered off the track from my goal.

Vivek Kumar Yadav
Kunal Gupta

Department of Computer Science and Engineering
Indian Institute of Information Technology Kalyani
Kalyani, W.B.-741235, India.

Abstraction

The advent of online transactions has its downsides too. Each year, millions of people all across the globe fall victim to credit card fraud leading to losses in billions of dollars. Therefore, it is the need of the hour to create algorithms that can accurately detect such fraudulent transactions by analysing the various details such as the time of the transaction, the amount transacted, the account numbers, etc. Putting data science and machine learning to good use, this project uses XGBoost,Logistic and Decision tree algorithm along with anomaly detection to weed out the fraudulent transactions using a highly imbalanced dataset obtained from Kaggle.

In this project we analyze a dataset made by European cardholders. The dataset contains 284,807 transactions, of which 492 (0.17) are fraudulent.

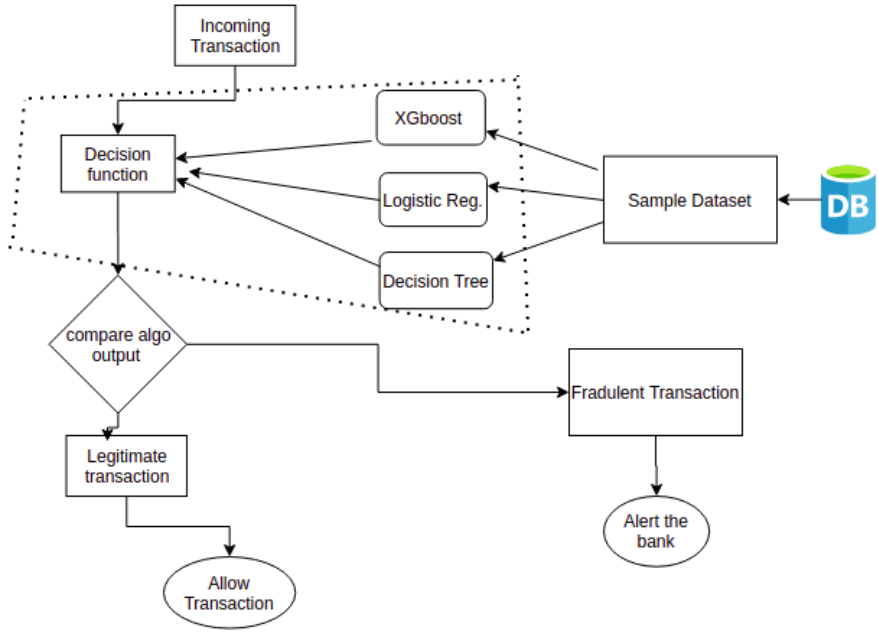
Each transaction has 30 features, all of which are numerical. The features V1, V2, ..., V28 are the result of a PCA transformation. To protect confidentiality, background information on these features is not available. The Time feature contains the time elapsed since the first transaction, and the Amount feature contains the transaction amount. The response variable, Class, is 1 in the case of fraud, and 0 otherwise.

Our 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.

0.1 Introduction

0.1.1 Background

It is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase.
Such problems can be tackled with Data Science and its importance,along with Machine Learning, cannot be overstated. This project intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud.
This model is then used to recognize whether a new transaction is fraudulent or not. Our objective here is to detect almost all of the fraudulent transactions while minimizing the incorrect fraud classifications. Credit Card Fraud Detection is a typical sample of classification.
In this process, we have focused on analysing and pre-processing data sets as well as the multiple detection algorithms such as Logistic ,XGboost,Decision tree algorithm on the PCA transformed Credit Card Transaction data.
Flow diagram shown below.



0.1.2 Problem Statement

The aim of the project is to predict fraudulent credit card transactions using machine learning models. This is crucial from the bank's as well as customer's perspective. The banks cannot afford to lose their customers money to fraudsters. Every fraud is a loss to the bank as the bank is responsible for the fraud transactions. The dataset contains transactions made over a period of two days by credit cardholders.

0.1.3 Prerequisites

Most of the libraries like sklearn, pandas, seaborn, matplotlib, numpy, scipy. We will be Training the model with various algorithm such as Logistic regression, Decision Tree, and XGboost also their comparisons. Apart from this we will use some balancing technique to improve our model.

0.1.4 Steps To Be Taken

1. Reading, understanding and visualising the data (EDA)
    - It is a CSV file, contains 31 features, the last feature is used to classify the transaction whether it is a fraud or not.

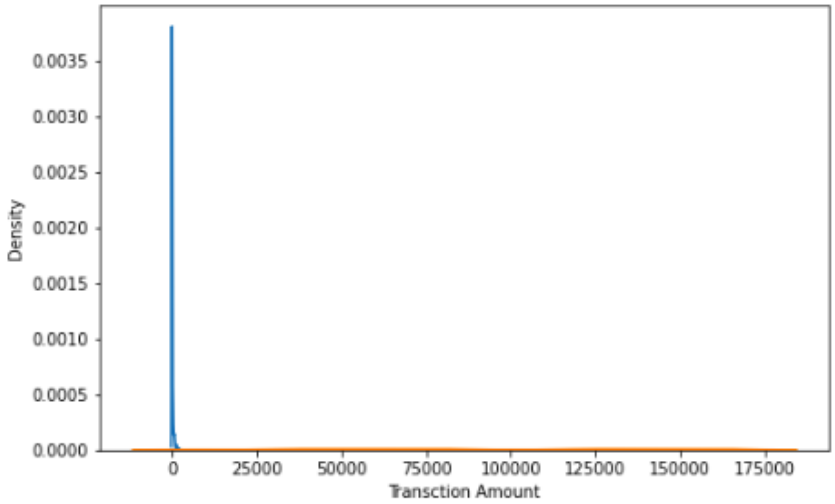
2. Preparing the data for modelling
3. Building the model (with different model)
4. Evaluate the model with different balancing techniques.

0.1.5 DataSet

It is a CSV file, contains 31 features, the last feature tells whether the transaction is a fraud or not. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. Which is 0.17%. It contains only numeric input variables. Due to secrecy issues original features are not provided Features V1, V2, . . . , V28 are the principal components obtained is the only features.
LINK : https://www.kaggle.com/mlg-ulb/creditcardfraud

0.1.6 The features have not been transformed with PCA is 'Time' and 'Amount'

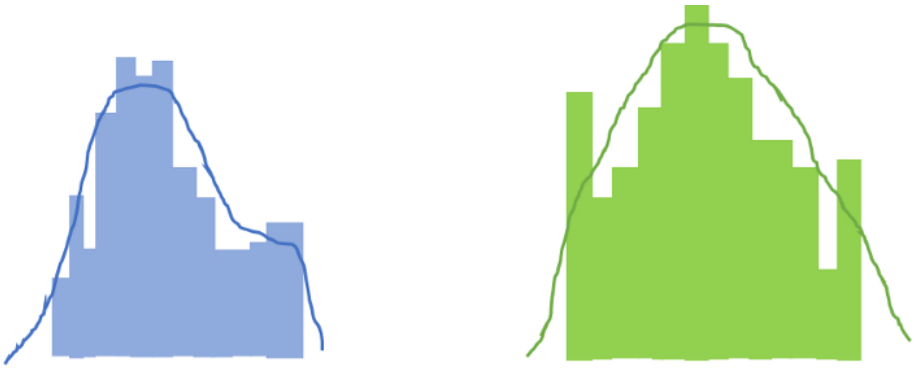
Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-senstive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.
We do not see any specific pattern for the fraudulent and non-fraudulent transctions with respect to Time. Hence, we can drop the Time column. **We have verified there is no missing values in any of the columns. Hence, there is no problem with null values in the entire dataset**



We can see that the fraudulent transctions are mostly densed in the lower range of amount, whereas the non-fraudulent transctions are spreaded throughout low to high range of amount.

0.1.7 To remove skewness in this dataset

we need to remove skewness in this dataset We are Using Power Tranformation to check and remove the skewness in dataset
POWER TRANSFORMATION Power transforms is a class of techniques that use a power function (like a logarithm or exponent) to make the probability distribution of a variable more-Gaussian. Because many machine learning algorithms prefer or perform better when numerical variables have a Gaussian probability distribution. Gaussian probability distribution This is often described as removing a skew in the distribution, although more generally is described as stabilizing the variance of the distribution



Before Transformation **After Transformation**

Basically a technique for transforming numerical input or output variables to have a Gaussian or more-Gaussian-like probability distribution.

We can apply a power transform directly by calculating the log or square root of the variable, although this may or may not be the best power transform for a given variable.

Instead, we can use a generalized version of the transform that finds a parameter (lambda) that best transforms a variable to a Gaussian probability distribution.

There are two popular approaches for such automatic power transforms they are (here we are using Yeo-Johnson):

1. Box-Cox Transform
2. Yeo-Johnson Transform
   Unlike the Box-Cox transform, it does not require the values for each input variable to be strictly positive.

   It supports zero values and negative values.

   PROS : We can apply it to our data set without scaling it first.

As per our information we are not going with SVM ,Random forest and KNN because of the reasons.

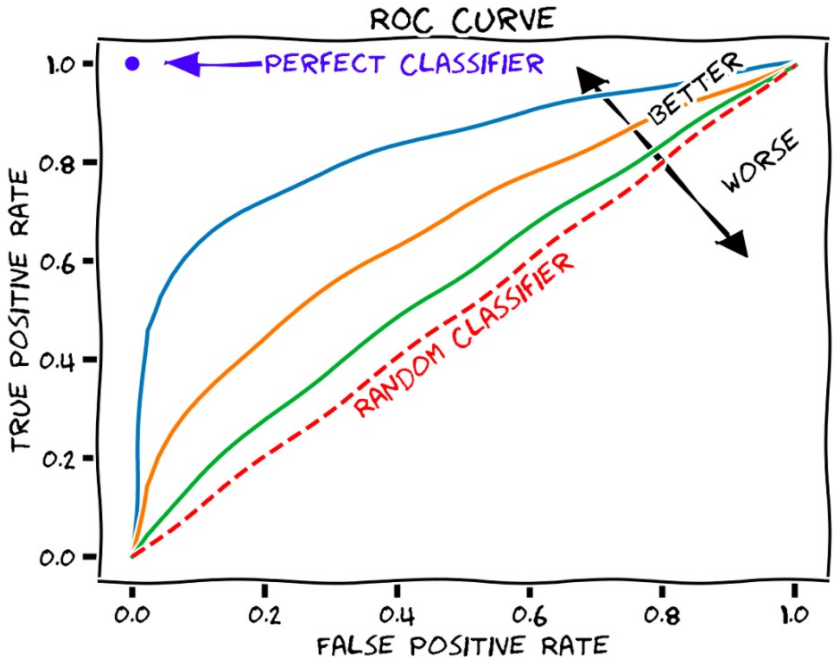
SVM is not very efficient with large number of datapoints beacuse it takes lot of computational power and resources to make the transformation.When we perform the cross validation with K-Fold for hyperparameter tuning, it takes lot of computational resources and it is very time consuming. Hence, because of the unavailablity of the required resources and time SVM was not tried. For the same reason Random forest was also not tried for model building in few of the hyperparameter tuning for oversampling technique.

KNN was not used for model building .As it is not memory efficient. It becomes very slow as the number of datapoints increases as the model needs to store all the data points. It is computationally heavy because for a single datapoint the algorithm has to calculate the distance of all the datapoints and find the nearest neighbors.

0.1.8 Using ROC-AUC (Receiver Operating Characteristic Curve)

As we have seen that the data is heavily imbalanced, where only 0.17% transactions are fraudulent, we should not consider accuracy as a good measure for evaluating the model. Because in the case of all the data points return a particular class(1/0) irrespective of any prediction, still the model will result very high accuracy.
Hence, we have to measure the **ROC-AUC** score for fair evaluation of the model. The ROC curve is used to understand the strength of the model by evaluating the performance of the model at all the classification thresholds.
Overall accuracy is based on one specific threshold, while ROC tries all of the threshold and plots the sensitivity and specificity. So when we compare the overall accuracy, we are comparing the accuracy

based on some threshold.



The ROC curve is measured at all thresholds, the best threshold would be one at which the TPR(True Positive Rate) is high and FPR(False Positive Rate) is low, i.e., misclassifications are low.
After determining the optimal threshold, we can calculate it. F1 Score(is a measure of a model's accuracy on a dataset) of the classifier to measure the precision and recall at the selected threshold.
F1 Score = 2*(Recall * Precision) / (Recall + Precision)

0.1.9 Confusion Matrix -Accuracy,Precision and Recall

Confusion Matrix:is used to find the amount of values which are predicted correctly and wrongly. Here **False Positive** FP is called as Type I error and **False Negative** FN are called as Type II error.
Where both the Type I and II errors are the values which are predicted wrongly.
Similarly **True Positive** and **True Negative** are the values which are predicted correctly.
Accuracy : It is simply a ratio of correctly predicted observation to the total observations.
Accuracy = TP+TN/TP+FP+FN+TN
Precision : Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
Precision = TP/TP+FP
Recall (Sensitivity) : Recall is the ratio of correctly predicted positive observations to the all observations in actual class
Recall = TP/TP+FN

0.2 MODEL Demonstration

Classification Models we are using over imbalanced data set.

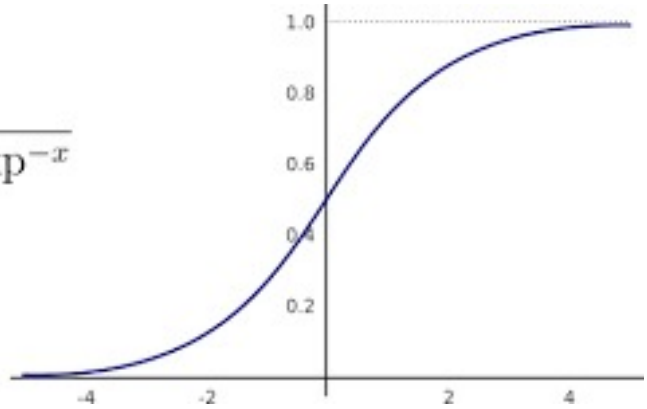
1. LOGISTIC Regression
2. DECISION Tree
3. XGBOOST
4. RANDOM FOREST

Our goal is to find best suited model that consumes less computational resources and build on infrastructure that takes less deploying cost.

0.3 LOGISTIC Regression

- It is the Supervised Learning technique used for predicting the categorical dependent variable(predicted variable (class)) using a given set of independent variables(predictors (our independent variables)(v1 to v28)
- It predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$

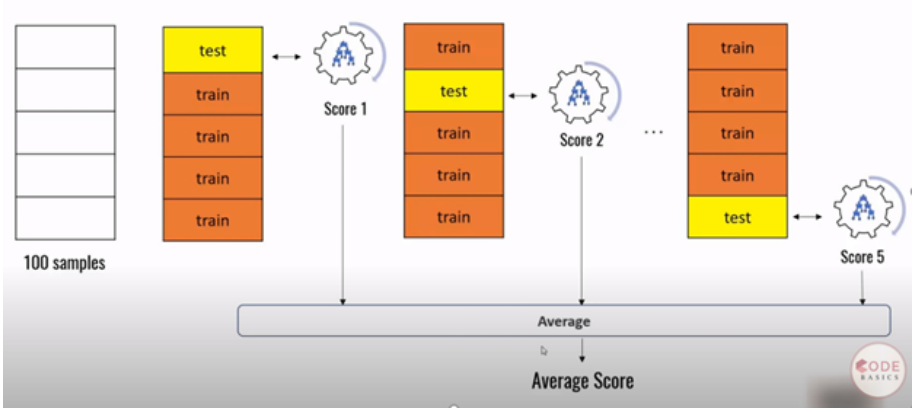


0.3.1 Logistic Function (Sigmoid Function)

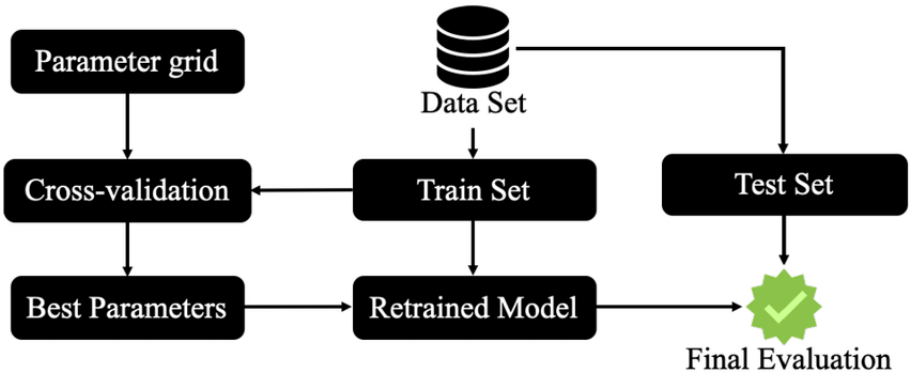
The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function. The mathematical steps to get Logistic Regression equations are given below:

0.3.2 Cross validation AND GridSearchCV

The most common is the k-fold cross-validation technique, where you divide your dataset into k distinct subsets. By choosing 1 of the k subsets to be the validation set(testing set), and the rest k-1 subsets to be the training set, we can repeat this process k times by choosing a different subset to be the validation set every time. This makes it possible to repeat the training-validation process k times, eventually going through the entire original training set as both training and validation set.



GridSearchCV : It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model.



0.3.3 Prediction on the Train Set AND Test Set (LOGISTIC)

Train Set Accuracy = 0.99
Sensitivity = 0.70
Specificity = 0.99
F1-Score = 0.76
ROC = 0.99
Test Set Accuracy = 0.99
Sensitivity = 0.77
Specificity = 0.99
F1-Score = 0.65
ROC = 0.97

0.4 DECISION TREE

Decision Tree's are an excellent way to classify classes, unlike a Random forest they are a transparent or a whitebox classifier which means we can actually find the logic behind decision tree's classification.
Next we're going to initialise our classifier and GridSearchCv which is the main component which will help us find the best hyperparameters.
We simply create a tuple (kind of non edit list) of hyperparameters we want the machine to test with as save them as parameters.
We then give the classifier and list of params as paramters to gridsearchcv.

0.4.1 GridSearchCV AND Tuning Of Hyperparameters

GridSearchCV : It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model.
The performance of a model significantly depends on the value of hyperparameters. Note that there is no way to know in advance the best values for hyperparameters so ideally, we need to try all possible values to know the optimal values.
Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyperparameters.

Create the parameter grid which are max depth, min samples leaf,min samples split: After we instantiate the grid search model GridSearchCV estimator dtree, paramgrid, scoring= 'rocauc ,cv is 3, verbose is 1 and fit the grid search to the data

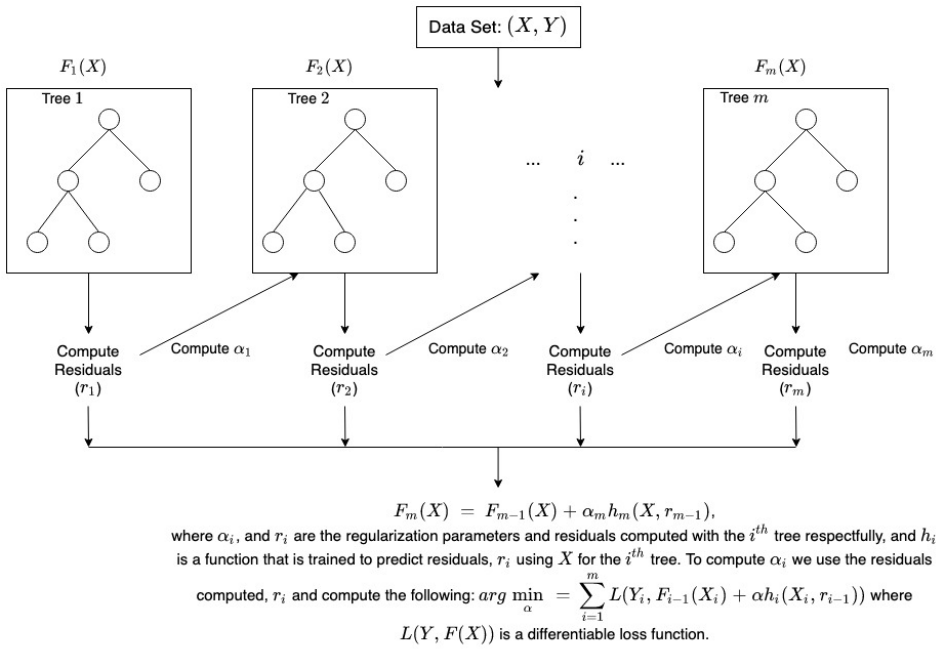
0.4.2 Prediction on the Train Set AND Test Set(DECISION TREE)

Train Set Accuracy = 0.99
Sensitivity = 1.0
Specificity = 1.0
F1 Score = 0.75

ROC-AUC = 0.95
Test Set Accuracy = 0.99
Sensitivity = 0.58
Specificity = 0.99
F1 Score = 0.75
ROC-AUC = 0.92

0.5 XGBOOST

XGBoost is boosting sequential technique which works on the principle of an ensemble(combine several base model to produce one optimal predictive model). It combines a set of weak learners and delivers improved prediction accuracy.
When it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.
This learning algorithm whose individual learning units are decision trees and trees have two favorable features which, again, render feature engineering unnecessary.
How it works internally ?



0.5.1 Hyperparameter Tuning With XGBoost

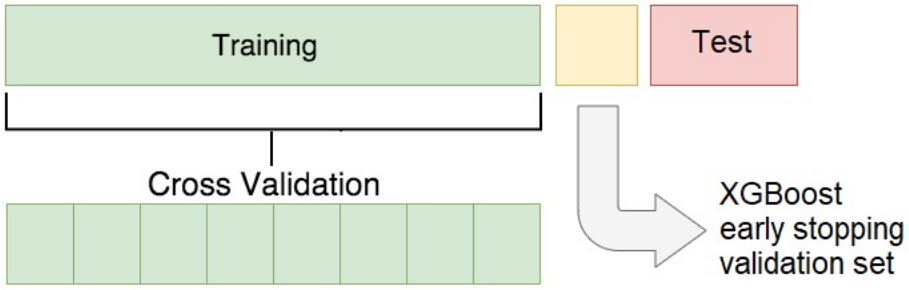
Hyper-parameter tuning can considerably improve the performance of learning algorithms.
XGBoost has many hyper-parameters which make it powerful and flexible, but also very difficult to tune due to the high-dimensional parameter space. Instead of the more traditional tuning methods(i.e. grid search and random search) that perform a search through the parameter space.

0.5.2 Challenges AND Solution

The main challenge in fraud detection is the extreme class imbalance in the data which makes it difficult for many classification algorithms to effectively separate the two classes. Only 0.172% of transactions are labeled as fradulent in this dataset. So we will address the class imbalance by re-weighting the data before training XGBoost.

0.5.3 Tuning By Cross Validation

Here we are using most common is the k-fold cross-validation technique, where you divide your dataset into k distinct subsets. By choosing 1 of the k subsets to be the validation set, and the rest k-1 subsets to be the training set, we can repeat this process k times by choosing a different subset to be the validation set every time. This makes it possible to repeat the training-validation process k times, eventually going through the entire original training set as both training and validation set.



0.5.4 Why tuning by Cross Validation ?

To prevent the early stopping feature from over-fitting, in addition to splitting the data into a training and a test set, we also set aside a small portion of the training set to be used as the early stopping validation set. The figure below shows how the data is split. In above figure Splitting the dataset. split the dataset into train and test sets. Then use a small part (of data) of the training set as the early stopping validation set.

0.5.5 Hyperparameter Tuning with GridSearchCV

It is the process of performing hyperparameter tuning in order to determine the optimal values for a given model cause the performance of a model significantly depends on the value of hyperparameters. So we use GridSearchCV to automate the tuning of hyperparameters.
This helps us to loop through predefined hyperparameters and fit estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.
Specifying the range of hyperparameters ('learning_rate': [0.2, 0.6], 'subsample' : [0.3, 0.6, 0.9])

0.5.6 Arguments taken by GridSearchCV

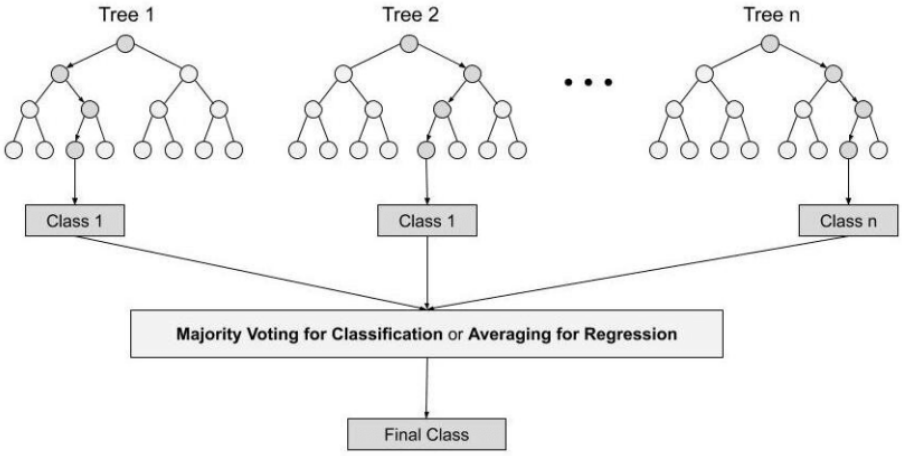
1. **estimator** : Pass the model instance for which you want to check the hyperparameters.
2. **params-grid** : The dictionary object that holds the hyperparameters you want to try
3. **scoring** : evaluation metric that you want to use, you can simply pass a valid string/ object of evaluation metric
4. **cv** : number of cross-validation you have to try for each selected set of hyperparameters
5. **verbose** : you can set it to 1 to get the detailed print out while you fit the data to GridSearchCV
6. **n-jobs** : number of processes you wish to run in parallel for this task if it -1 it will use all available processors.

0.6 RANDOM FOREST

It is a supervised ML algorithm used in classification problems and regression problems. It builds decision trees on different samples and takes their majority vote. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing categorical variables.

0.6.1 Steps involved in Random Forest:

Step 1: In Random forest n number of random records are taken from the data set having k number of records. Step 2: Individual decision trees are constructed for each sample. Step 3: Each decision tree will generate an output. Step 4: Final output is considered based on Majority Voting .



0.6.2 Hyperparameter tuning in Random Forest:

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster. Hyperparameters that increase predictive power: Number of trees the algorithm builds before averaging the predictions. Maximum number of features considered while splitting a node. Minimum number of nodes required to split an internal node. Hyperparameters that make the model fast: Number of processors it can use. Randomness while selecting from training data. Size of test dataset.

0.6.3 Prediction on the Train Set AND Test Set(RANDOM FOREST)

Train Set Accuracy = 0.99
Sensitivity = 1
Specificity = 1
F1-Score = 0.8
ROC = 0.98
Test Set Accuracy = 0.99
Sensitivity = 0.62
Specificity = 0.99
F1-Score = 0.75
ROC = 0.96

0.6.4 Model With Optimal Hyperparameters

We see that the train score almost touches to 1. Among the hyperparameters, we can choose the best parameters as learning rate : 0.2 and subsample: 0.3

We can see that between the models we tried (Logistic,, Decision Tree, and XGboost), almost all of them have performed well. More specifically Logistic regression and XGBoost performed best in terms of ROC-AUC score. But as we have to choose one of them, we can go for the best as XGBoost, which gives us ROC score of 1.0 on the train data and 0.98 on the test data. Keep in mind that XGBoost requires more resource utilization than Logistic model. Hence building XGBoost model is more costlier than the Logistic model. But XGBoost having ROC score 0.98, which is 0.01 more than the Logistic model. The 0.01 increase of score may convert into huge amount of saving for the bank. Finally our task is to predict accuracy using ROC-AUC curve after choosing best balancing technique mentioned. So we are gearing up for that.

0.7 Goal to be achieve

Choosing best model on the balanced data. Here we are trying to balance the data with various approach such as

- Undersampling
- Oversampling
- SMOTE : Synthetic minority oversampling technique
- Adasyn

With every data balancing technique we built several models such as Logistic, XGBoost, Decision Tree, and Random Forest. **WHY NEED TO BALANCE THE DATA** We have seen that the data is heavily imbalanced, where only 0.17% transactions are fraudulent, we should not consider simply a accuracy as a good measure for evaluating the model. Because in the case of all the data points return a particular class(1/0) irrespective of any prediction, still the model will result very high Accuracy.
Hence, we have to measure the ROC-AUC score for fair evaluation of the model,along with Balancing techniques given below.

0.8 Goal to Choose Best Balancing Technique

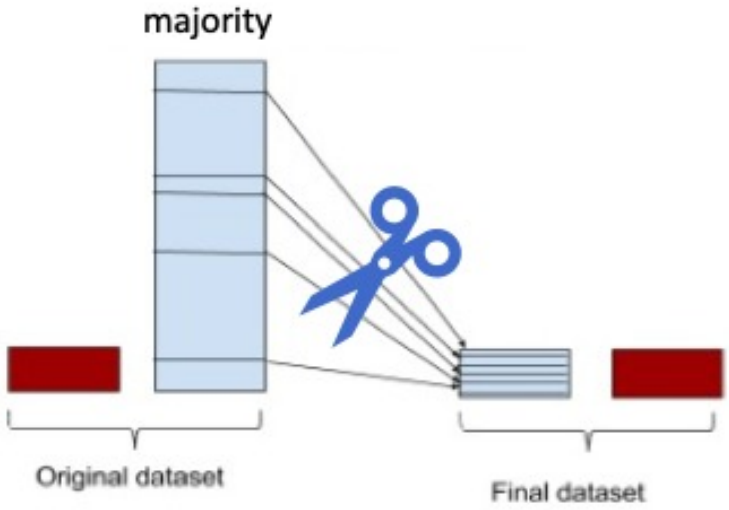
Undersampling : Here for balancing the class distribution, the non-fraudulent transactions count will be reduced to 396 (similar count of fraudulent transactions)
Oversampling : Here we will make the same count of non-fraudulent transactions as fraudulent transactions.
SMOTE : Synthetic minority oversampling technique. It is another oversampling technique, which uses nearest neighbor algorithm to create synthetic data.
Adasyn : This is similar to SMOTE with minor changes that the new synthetic data is generated on the region of low density of imbalanced data points.

0.9 Predict Accuracy Of Model And Feasibility

Finally our task is to predict accuracy using ROC-AUC curve after choosing best balancing technique mentioned above.
Like we can perform all three classification models in **Undersampling, Oversampling, SMOTE, and in Adasyn** respectively. Only to get best suited models over tried classification model.
Reason : We also have to consider that for little change of the ROC score how much monetary loss of gain the bank incur.

0.10 UNDERSAMPLING

In this we reduces data in majority class so that it can match with data of minority class. Basically we will shrink down the majority to minority, If we don't do this model will be biased towards majority data set so we are doing balancing. It will increase classification performance.



Working with libraries. Imblearn - provides tools when dealing with classification with imbalanced classes. Using undersampler library which under-sample the majority class(es) by randomly picking samples with or without replacement.

0.10.1 Accuracy (ROC-AUC) Comparison

Logistic : without any balancing technique Train set :Accuracy = 0.99 Sensitivity = 0.70 Specificity = 0.99 ROC = 0.99
Test set :Accuracy = 0.99 Sensitivity = 0.77 Specificity = 0.99 ROC = 0.97
Decision Tree - without any balancing technique
Train set:Accuracy = 0.99 Sensitivity = 1.0 Specificity = 1.0 ROC-AUC = 0.95
Test set:Accuracy = 0.99 Sensitivity = 0.58 Specificity = 0.99 ROC-AUC = 0.92
XGboost - without any balancing technique
Train set Accuracy = 0.99 Sensitivity = 0.85 Specificity = 0.99 ROC-AUC = 0.99 F1-Score = 0.90
Test set Accuracy = 0.99 Sensitivity = 0.75 Specificity = 0.99 ROC-AUC = 0.98 F-Score = 0.79

0.10.2 Logistic With Undersampling

Train Set
Accuracy = 0.95
Sensitivity = 0.92
Specificity = 0.98
ROC = 0.99
Test Set Accuracy = 0.97
Sensitivity = 0.86
Specificity = 0.97
ROC = 0.96

0.10.3 Decision Tree With Undersampling

Train Set Accuracy = 0.93
Sensitivity = 0.88
Specificity = 0.97
ROC-AUC = 0.98
Test Set Accuracy = 0.96
Sensitivity = 0.85
Specificity = 0.96
ROC-AUC = 0.96

0.10.4 XGboost With Undersampling

Train Set Accuracy = 1.0
Sensitivity = 1.0
Specificity = 1.0
ROC-AUC = 1.0
Test Set Accuracy = 0.96
Sensitivity = 0.92
Specificity = 0.96
ROC-AUC = 0.98

0.10.5 Random Forest With Undersampling

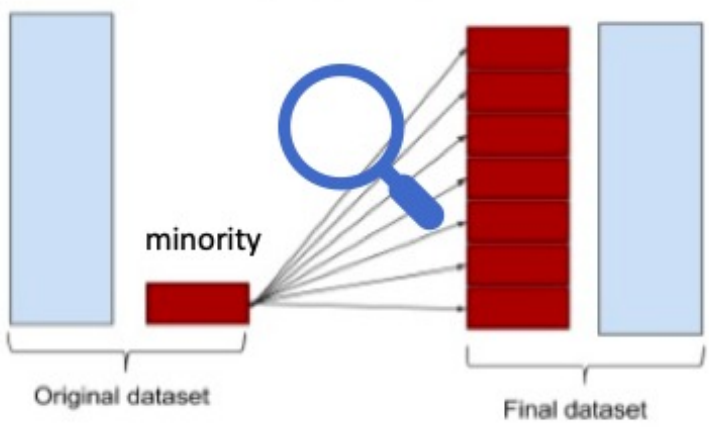
Train Set Accuracy = 0.94
Sensitivity = 0.89
Specificity = 0.98
ROC = 0.98
Test Set Accuracy = 0.98
Sensitivity = 0.83
Specificity = 0.98
ROC = 0.97

0.11 OVERSAMPLING

It basically means duplicating samples from the minority class. Random Oversampling includes selecting random examples from the minority class and creating multiple copies of this instance, hence it is possible that a single instance may be selected multiple times. This random selection and duplication may increase the likelihood of overfitting.

0.11.1 Working Procedure

In this we increase the data of minority class so that it can match with data of majority class. Basically we will expand the minority to majority. If we don't do this, model will be biased towards majority data set so we are doing balancing. It will increase classification performance.



0.11.2 Logistic With Oversampling

Train Set
Accuracy = 0.95
Sensitivity = 0.92
Specificity = 0.97

ROC = 0.99
Test Set Accuracy = 0.97
Sensitivity = 0.87
Specificity = 0.97
ROC = 0.97

0.11.3 Decision Tree With Oversampling

Train Set Accuracy = 0.99
Sensitivity = 1
Specificity = 0.99
ROC-AUC = 1
Test Set Accuracy = 0.99
Sensitivity = 0.79
Specificity = 0.99
ROC-AUC = 0.89

0.11.4 XGboost With Oversampling

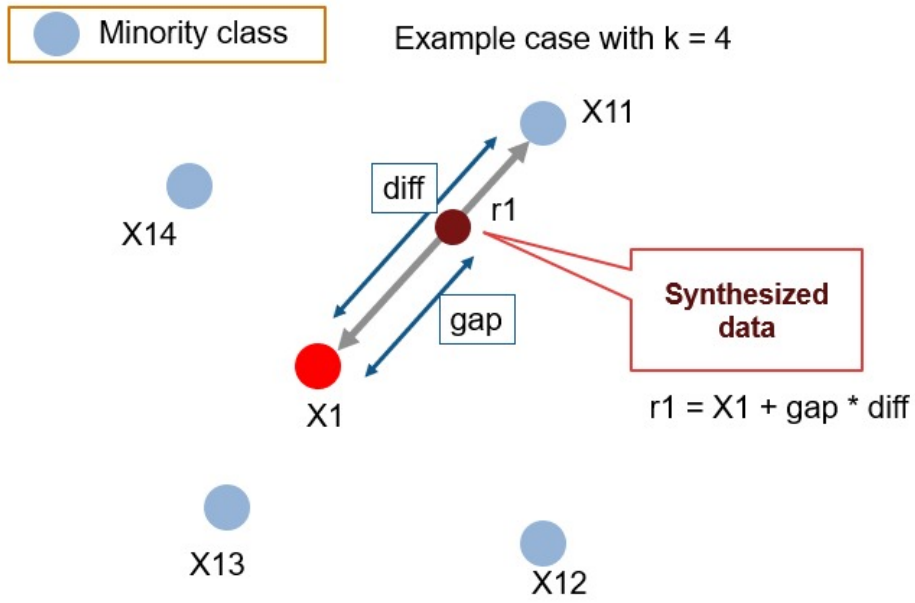
Train Set Accuracy = 1.0
Sensitivity = 1.0
Specificity = 1.0
ROC-AUC = 1.0
Test Set Accuracy = 0.99
Sensitivity = 0.79
Specificity = 0.99
ROC-AUC = 0.98

0.12 SMOTE

Synthetic Minority Overlapping Technique uses nearest neighbor algorithm to create synthetic data. Smote takes the entire dataset as input and increases the percentage of only the minority cases. To double the percentage of minority set the module property to 200% To triple set to 300% and so on.

0.12.1 Working Procedure

At first the total no. of oversampling observations, N is set up. Generally, it is selected such that the binary class distribution is 1:1. But that could be tuned down based on need. Then the iteration starts by first selecting a positive class instance at random. Next, the KNN's (by default 5) for that instance is obtained. At last, N of these K instances is chosen to interpolate new synthetic instances. To do that, using any distance metric the difference in distance between the feature vector and its neighbors is calculated. Now, this difference is multiplied by any random value in (0,1] and is added to the previous feature vector.
This is pictorially represented below.



0.12.2 Logistic with SMOTE

Train Set Accuracy = 0.95
Sensitivity = 0.92
Specificity = 0.98
ROC = 0.99
Test Set Accuracy = 0.97
Sensitivity = 0.90
Specificity = 0.99
ROC = 0.97

0.12.3 Decision Tree With SMOTE Balancing Technique

Train Set Accuracy = 0.99
Sensitivity = 0.98
Specificity = 0.99
ROC-AUC = 0.99
Test Set Accuracy = 0.98
Sensitivity = 0.80
Specificity = 0.98
ROC-AUC = 0.86

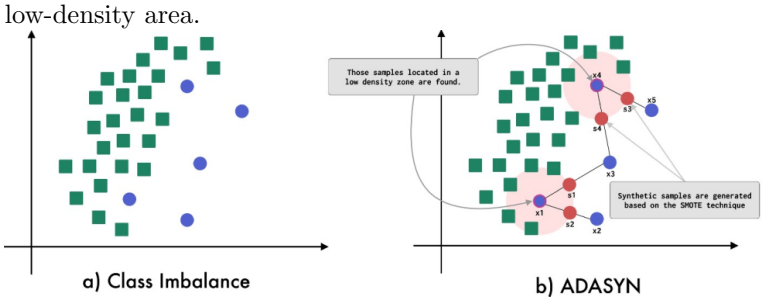
0.12.4 XGboost with SMOTE Balancing Technique

Train Set Accuracy = 0.99
Sensitivity = 1.0
Specificity = 0.99
ROC-AUC = 1.0
Test Set Accuracy = 0.99
Sensitivity = 0.79
Specificity = 0.99
ROC-AUC = 0.96
Overall, the model is performing well in the test set, what it had learnt from the train set.

0.13 ADASYN(Adaptive Synthetic)

ADASYN implements a methodology that detects samples from the minority class found in spaces dominated by the majority class, in order to generate samples in the lower density areas of the minority class. That is, ADASYN focuses on those samples of the minority class that are difficult to classify because they are in a low-density area.

It is an algorithm that generates synthetic data and its advantages are not copying the same minority data and generating more data for "harder to learn" examples.
After creating those samples it adds random values so as to make it more realistic.
ADASYN implements a methodology that detects those samples of the minority class found in spaces dominated by the majority class.
Cause it focuses on those samples of the minority class that are difficult to classify because they are in a low-density area.



0.13.1 Logistic With Adasyn

Train Set
Accuracy = 0.88
Sensitivity = 0.86
Specificity = 0.91
ROC = 0.96
Test Set Accuracy = 0.99
Sensitivity = 0.54
Specificity = 0.99
ROC = 0.97

0.13.2 Decision Tree With Adasyn

Train Set Accuracy = 0.99
Sensitivity = 1
Specificity = 1
ROC-AUC = 0.95
Test Set Accuracy = 0.99
Sensitivity = 1
Specificity = 1
ROC-AUC = 1

0.13.3 XGboost With Adasyn

Train Set Accuracy = 0.99
Sensitivity = 1.0
Specificity = 1.0
ROC-AUC = 1.0
Test Set Accuracy = 0.99
Sensitivity = 0.78
Specificity = 0.99

ROC-AUC = 0.96

0.14 Choosing Best Model On The Balanced Data

Here we balanced the data with various approach such as Undersampling, Oversampling, (can also apply SMOTE and Adasy). With every data balancing technique we built several models such as Logistic, XGBoost, Decision Tree.
We can see that almost all the models performed more or less good. But we should be interested in the best model.Though the Undersampling technique models performed well, we should keep mind that by doing the undersampling some information were lost. Hence, it is better not to consider the undersampling models.
Whereas the SMOTE performed well. Among those models the simplest model Logistic regression has ROC score 0.98 in the train set and 0.97 on the test set. We can consider the Logistic model as the best model to choose because of the easy interpretation of the models and also the resource requirements to build the model is lesser than the other heavy models such as Random forest(NOT USING) or XGBoost. Hence, we can conclude that the Logistic regression model with SMOTE is the best model for its simplicity and less resource requirement.

0.15 Cost Benefit Analysis

We have tried several models till now with both balanced and imbalanced data. We have noticed most of the models have performed more or less well in terms of ROC score, Precision and Recall.
But while picking the best model we should consider few things such as whether we have required infrastructure, resources or computational power to run the model or not.
For the models such as Random forest, SVM, XGBoost we require heavy computational resources and eventually to build that infrastructure the cost of deploying the model increases. On the other hand the simpler model such as Logistic regression requires less computational resources, so the cost of building the model is less.
We also have to consider that for little change of the ROC score how much monetary loss of gain the bank incur. If the amount if huge then we have to consider building the complex model even though the cost of building the model is high.

0.16 Summary AND Conclusion

For banks with smaller average transaction value, we would want high precision because we only want to label relevant transactions as fraudulent for every transaction that is flagged as fraudulent we can add the human element to verify whether the transaction was done by calling the customer. However, when precision is low, such tasks are a burden because the human element has to be increased.
For banks having a larger transaction value, if the recall is low, i.e., it is unable to detect transactions that are labelled as non-fraudulent. So we have to consider the losses if the missed transaction was a high-value fraudulent one.
So here, to save the banks from high-value fraudulent transactions, we have to focus on a high recall(in those model we worked) in order to detect actual fraudulent transactions.
After performing several models, we have seen that in the balanced dataset with SMOTE technique the simplest Logistic regression model has good ROC score and also high Recall.
Hence, we can go with the logistic model here. It is also easier to interpret and explain to the business.

Bibliography

[1] https://www.tutorialspoint.com/Machinelearning.HTML

[2] :https://towardsdatascience.com/

[3] https://spd.group/machinelearning/creditcardfrauddetection/https://www.kaggle.com/mlgulb/creditcardfraud

[4] https://pianalytix.com/machinelearning

[5] https://www.dataschool.io/

[6] Bergstra, James, Daniel Yamins, and David Daniel Cox 2013Making a science of model search: Hyperparameter optimization in hundreds of dimensions forvision architectures

[7] Chawla, Nitesh V, et al. 2002. SMOTE synthetic minority oversampling technique. Journal of artificial intelligence research Davis, Jesse, and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd international conferenceon Machine learning, 233240 ACM.