ANALYZING THREE DIFFERENT TUNING STRATEGIES FOR RANDOM FOREST HYPERPARAMETERS FOR FRAUD DETECTION

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Kulwinder Kaur Sarao

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ANALYZING THREE DIFFERENT TUNING STRATEGIES FOR RANDOM FOREST HYPERPARAMETERS FOR FRAUD DETECTION

By

Kulwinder Kaur Sarao

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Saeed Salem

Dr. Maria Alfonseca

Approved:

11/12/2021
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

Technology is advancing rapidly, and more tasks are becoming online than ever. Along with the benefits comes the disadvantages of this great advancement. While online services relieve from the struggle of in person activities, it also puts you on the risk of getting deceived by the fraudsters. This paper aims to detect the fraudulent transactions made online from a bank using a synthetically produced dataset. A random forest model has been trained to predict the fraudulent transactions. To achieve the best performance, the hyperparameters of the model have been tuned using three different tuning methods. As it turns out, grid search proved to be the best tuning strategy in terms of the mean cv score, precision, recall, f1-score and accuracy. It only lacked in providing the best run time, where Bayesian Optimization scored well than the others.

ACKNOWLEDGMENTS

I would like to thank all the people who in some way have contributed to the work done in this paper. Firstly, I would like to thank my advisor, Dr. Simone Ludwig, for the immense support she has provided me during my journey of master's program. I would also like to thank my committee members Dr. Saeed Salem and Dr. Maria Alfonseca for their interest in my work and serving as my committee members.

I would like to thank my parents for their constant love and affection that encouraged me to work hard. Finally, I would like to thank a big time to my mentor, my brother, for always being there for me and putting all the faith in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

1. INTRODUCTION

The banking sector is the most common sector, as every single human has or is going to interact with a bank in their lifetime. Banking has made it very easy for humans to manage their finances. With the advancement in technology, most of the people tend to use online banking rather the physical one. Online banking has great benefits, skipping the hectic wait in lines for physical banking. Thus, making the processes much easier and faster. It does not matter, if you must pay your bills, buying something, pay tuition etc., everything can be done online.

There is always a risk of getting deceived by fraudsters when browsing online. Fraud is increasing at a rapid rate day by day. Fraudsters are looking for a way to deceive anyone they can find. It happens both offline and online and in almost all the business sectors. But the banking sector is more prone to fraud, as people tend to save a huge portion of their money in the bank rather than at home. There have been a lot of work going on to solve this issue. Governments and private companies are spending tons of money to invent a permanent solution to this problem. But fraudsters tend to evolve alongwith the technologies. They are coming up with new methods and tricks to fool the people and loot their money. It is analogous to a police man chasing the thief. Fraud detection is a hot topic in the computer science world. More and more techniques have been developed to catch the fraud behaviour in a service.

Machine learning (ML) has played a huge role in this domain. Researchers have developed really smart machine learning models that are able to detect the fraud transactions to a great extent. But due to lesser data available in this domain, these models are far from being perfect, although they do quite a good job. Researchers are always coming up with new strategies to enhance the performance of their machine learning model. Almost every machine learning model has hyperparameters associated with it.

Hyperparameters are the parameters that need to be set before training the machine learning model. The values of these hyperparameters cannot be altered after the model has been trained. They play a vital role in the performance of the model. To understand it in an easy way, let us take an example of the radio. Assume the radio as a machine learning model and the frequency setting knob of the radio as the hyperparameter of the model. You can achieve the best sound quality of the radio by adjusting the knob to a specific setting. A slight change in either direction of the knob will distort the sound quality of the radio. Similar is the case with the machine learning models. Only by setting the optimal value of the hyperparameters, you can achieve the best performance of the machine learning model. Selecting even a slightly higher or lower value than the optimal value, can provide us a poorly working machine learning model.

There is ton of research going on in this domain as well. Machine learning hyperparameters can have a wide range of domain. Therefore, it is impossible to achieve the best results from the model by manually changing the parameters each time. This becomes even harder when the ML model has multiple hyperparameters. Because an optimal combination of the hyperparameter values is extremely hard to find manually. There are various methods that scientists have developed to automatically find the best set of hyperparameters for a ML model. But not all the methods work to tune each and every model. Therefore, one must have a good understanding of both the tuning strategies and the model they are going to train, in order to achieve the best results possible.

2. RELATED WORK

There have been ton of research on detecting fraud cases beforehand, so that the losses to the banks and the individuals can be minimized. Frauds happen in most of the sectors of the business: telecom, bank, medical etc. There is an urgent need to develop more and more ways to stop the fraudsters playing with the people's hard work and their money.

Sarma et al. proposed a community detection algorithm to identify the patterns that could catch the fraud attempts [1]. A web application was built to detect the fraud using agile methodology that acted as a central hub between the customers and the bank. A graph database system, Neo4j, was used to search and filter the fraud cases. Finally, the test experiments were detected successfully by the built application and then represented to the user graphically.

Another paper by John [1] used the data mining techniques to address the fraud detection in banking. This paper talks about using the data mining techniques: clustering, association, classification and forecasting to find the patterns by analyzing the customer data, that could lead to detecting the fraud.

The paper in [2] proposed a smart approach, using optimized light gradient boosting machine (OLightGBM), to detect the credit card fraud. The hyperparameters of this model were tuned by Bayesian-based optimization. To test the performance of the developed model, two real-world publicly available transaction data set for credit cards were used. The above-mentioned method outperformed the other approaches by achieving the highest accuracy (98.40%).

Chomiak [3] tried to detect fraud in the telecom industry. They used deep learning techniques for this purpose, on a real mobile communication carrier data. They evaluated the performance of different deep learning algorithms against each other. Deep convolutional neural

networks turned out to be the best technique with an accuracy of 82% as compared to other algorithms (Support vector machines, random forest, gradient boosting).

In [4], a new generative adversarial network (GAN) that calculates the probability of a large transfer which could be fraudulent was proposed. If the probability exceeds a threshold, the bank can get notified and potentially stop the transaction. This model used a deep denoising autoencoder to learn the probabilistic relationship between the input features and then plays a minmax game between a generator and a discriminator to classify the positive and negative samples accurately.

The above mentioned were some of the research papers published in the fraud detection domain. Now, let us discuss some of the papers that shows the work done on tuning the hyperparameters of a machine learning model and how the tuning benefits the model.

Wu in 2019 [5] used the gaussian process to construct the relationship between the performance of a machine learning model and their hyperparameters. This way, the problem of tuning the hyperparameters was considered an optimization problem and was solved using Bayesian optimization, that is based on the bayes theorem. The results of the experiments for the proposed method were successful in finding the best hyperparameters for various machine learning algorithms, including random forest and neural networks.

The paper in [6] introduced a generic approach to utilize the knowledge from previously performed experiments while tuning the hyperparameters of ML model that is working on new problems. They demonstrated a surrogate-based collaborative tuning (SCoT) technique in two experiments. SCoT is a combination of surrogate-based ranking and optimization methods. The proposed technique outperformed the standard practices.

A Simple and Robust Hyperparameter Tuning Algorithm (ASHA) [7] was introduced in this paper. This algorithm is said to exploit the features like early-stopping and parallelism, making it outstand the state-of-the-art hyperparameter optimization techniques. This algorithm have a linear relationship with the distributed number of workers. Moreover, it is efficient as compared to Vizier (internal hyperparameter tuning service by google), as it takes only half of the time to process.

There is a problem of time constraint while performing hyperparameter tuning jobs, as it increases the financial cost associated with it. In order to address this problem, researchers have created a framework called Rubberband [8]. It is claimed to be the first framework that provides elastic and cost-effective execution of the hyperparameter jobs in the cloud. This framework has successfully and efficiently reduced the financial cost by the factor of 2 as compared to the static allocation baselines.

With the massive increase in the data, hyperparameter tuning methods needs to be applicable to the big data as well. A novel framework was introduced to optimize the hyperparameters for the big data [9]. Big data was divided into smaller pieces and then Bayesian optimization was considered to generate the different configurations for the hyperparameters of the model. After being tested with two machine learning algorithms, it was evaluated that the framework reduced the computational time in comparison to the modern tuning strategies.

3. APPROACH

For this project, a Random Forest Classifier model has been trained for detecting fraud. To get the best results possible, the hyperparameters of the model have then been tuned using various strategies. Below is a detailed description of the terminologies used in this project.

3.1. Random Forest Classifier

Random forest (RF) comes under the classification of supervised learning algorithms. Supervised learning is a subclass of machine learning that uses a predefined training set to train the model, such that it could predict the desired output.

As the name suggests, random forest builds a forest of random decision trees. It is used to solve both classification and regression problems. It is an ensemble learning model, that means it uses many classifiers to obtain solutions for many complex problems. It takes the average/mean of multiple trees to predict the output.

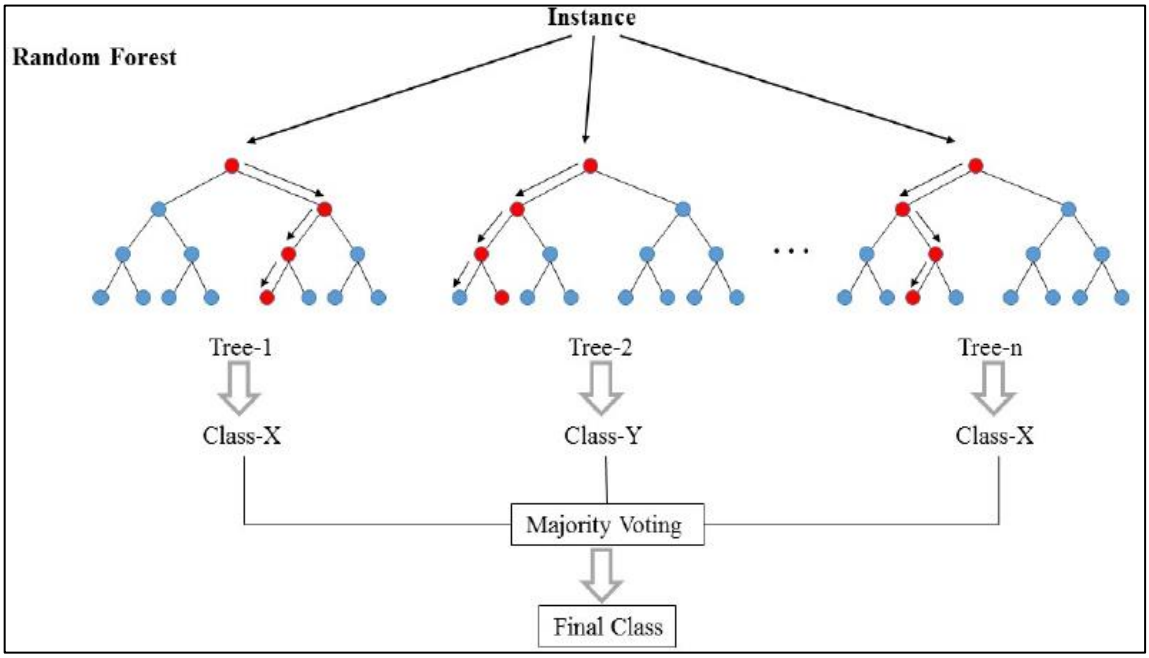

Figure 1. Working of Random Forest [10]

3.1.1. Hyperparameters of Random Forest

A hyperparameter is a parameter whose value is set before training the model and cannot be determined from the data. Only by specifying the specific value of hyperparameters for a problem, one could achieve the best performance of the model. Therefore, it becomes truly important to determine the best hyperparameters of a model for a given problem to achieve the best results possible. The hyperparameters of random forest in consideration for this project are as follows:

3.1.1.1. N_estimators

This hyperparameter decides the number of decision trees generated for the random forest model. Increasing the number of trees might help achieve the best results, but that is not always true. Increasing the value of this parameter do not contribute to overfitting but might increase the time complexity.

3.1.1.2. Max_depth

This is one of the most important hyperparameter for the RF model. It decides the threshold of the height, the trees inside the RF model could grow. With the increase in value for this parameter, the accuracy increases up to a certain limit and then starts decreasing gradually because of the overfitting.

3.1.1.3. Min_samples_split

This hyperparameter decides the number of minimum samples a decision tree inside the RF can hold before splitting into a new tree. Keeping the value low for this parameter means that the tree will continue to grow and can end up getting over-fitted. On the other hand, higher values for this parameter can cause underfitting.

3.1.1.4. Min_samples_leaf

It is the minimun number of samples a decision tree must hold in order to split. Extremely higher or lower values of this parameter could cause over-fitting or under-fitting of the model.

3.1.1.5. Max_features

This parameter helps in finding the number of features needed to form the best split of the trees possible. It can take four values: auto, sqrt, log2 and None. Auto and sqrt are mostly same as they take the square root of the n estimators. While log2 takes the log2 of the n estimators and none considers taking the n estimators as is.

3.2. Hyperparameter Optimization

Hyperparameter tuning or optimization is the process of choosing optimal value for a hyperparameter of the machine learning algorithm. Only with selecting the optimal value for the hyperparameters of a machine learning algorithm, RF in our case, we can achieve the best performance of the model. There are various techniques available to tune the hyperparameters. For the scope of this project, we will be focusing on the following three techniques:

3.2.1. Grid Search

This could be considered as one of the simplest algorithms for tuning the hyperparameters. As the name suggests, the whole domain of the possible values of the hyperparameters are divided into discrete grids. Then, the performance metrics are evaluated using different combinations of the grid using cross-validation. The optimal combination of the values of the hyperparameters is the point where the average of the cross-validation is the maximum.

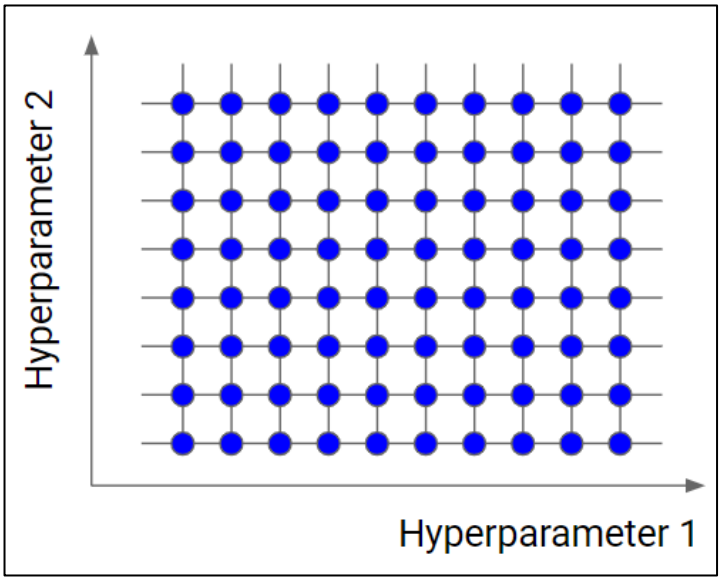

Figure 2. Grid Search Tuning [11]

Grid search evaluates all the possible combinations of the grid and usually finds the best set of hyperparameters. Because of spanning across all the combinations, it becomes an exhaustive search and takes larger processing times.

3.2.2. Genetic Algorithm

Genetic algorithm (GA) is a classic evolutionary algorithm which is inspired by the natural process of evolution. It is considered a stochastic search algorithm because it applies random changes to the current solutions to generate the new solutions. The following diagram depicts the cycle process of an evolutionary algorithm:

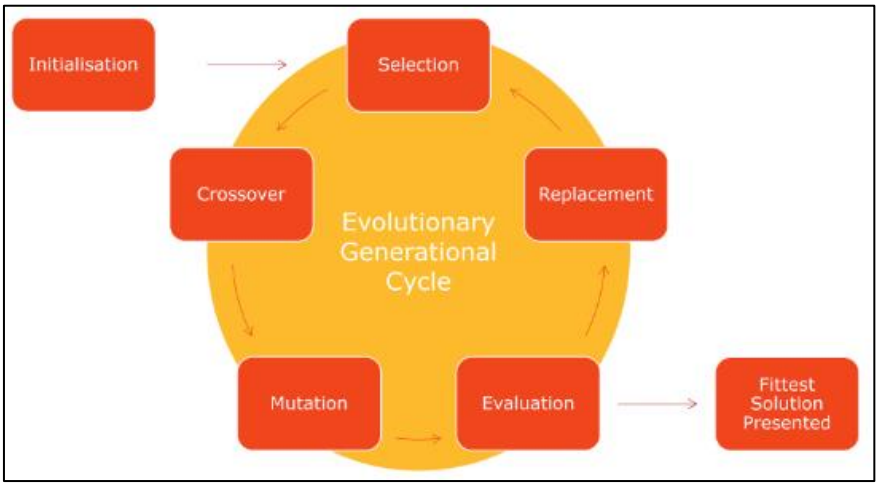

Figure 3. Working of Evolutionary Algorithms [12]

3.2.2.1. Basic Terminologies of GA

- Population is the subset of all the possible solutions of the problem we are trying to solve.
- Chromosome is one of the candidate solutions out of the population. It can be mutated and altered as per needs.
- Gene is further an element of the chromosome.
- Genetic operators help changing the composition of the next generation.
- Fitness function is a function that takes specific inputs to produce child/offspring which is better than its parents.

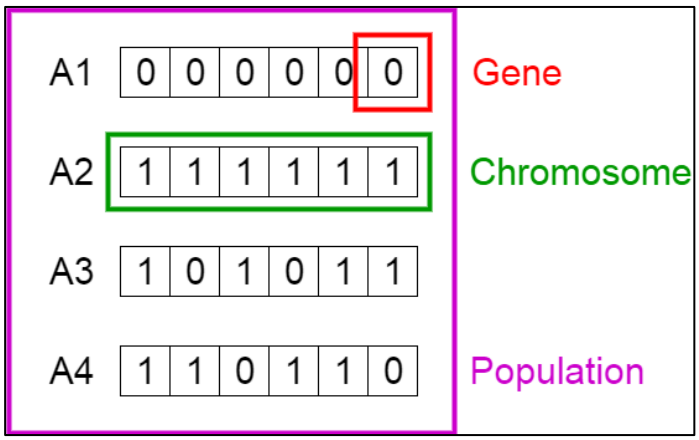

Figure 4. Genetic Algorithm Terminologies [13]

3.2.2.2. Working of GA

GA follows this process to work on optimization problems:

- Initialization: the first step in the process is to generate an initial population that comprises of the possible solutions to the problem in hand.
- Fitness assignment: All the individuals in the population are assigned a fitness score by the fitness function. The fitness score determines the probability of being selected for reproduction. The higher the fitness score, the higher is the probability of getting selected.
- Selection: This is the phase where individuals are selected to produce the offspring. The individual who has been chosen are then arranged in a pair of two in order to enhance the reproduction, where they pass their genes on to the next generation (offspring).
- Reproduction: A population of children is created in this step. The two main operators of this phase are crossover and mutation. Crossover swaps the genetic

information between two parents to produce the offspring. Whereas, new genetic information is added to the newly created child population in mutation.

- Replacement: in this step the old population is replaced with the new population that consists of higher fitness scores than their parents, thus, improving the results.
- Termination: this is the last step, used to terminate the algorithm. Once, a threshold fitness score has been achieved after the replacement, then the algorithm terminates itself. The best solution obtained is then provided.

3.2.3. Bayesian Optimization

Black box is an approach, where we can give an input and observe the output without having knowledge of internal workings of the system. In machine learning, the problems where objective function $f(\mathrm{x})$, acts as a black box, are considered as black box problems [14].

Bayesian optimization (BO) techniques tend to work around this terminology. The goal of BO is to find the global optimal solution in the least number of attempts possible. In BO, a prior belief is built around f, that gets updated using the samples that are drawn from f. This is done in order to achieve a posterior that could better approximate f. The model used for this purpose, is known as surrogate model. Another terminology, acquisition function, is used to direct the sampling to the areas where there is a chance of improvement over the current best observation.

3.2.3.1. Surrogate Model

One of the most popular surrogate models is the Gaussian Process (GP). GP is a random process to assign a random variable to the objective function $f(\mathrm{x})$. The variable can be any random variable chosen from $\mathrm{x} \in \mathbb{R}^{\mathrm{d}}$.

GPs define the prior functions that can be used to incorporate prior beliefs of the $f(x)$ such as smoothness. It is comparatively cheaper to evaluate the posterior of GP. It can be used to propose the sampling points in the search space, where there is a chance for improvement.

3.2.3.2. Acquisition Function

This function proposes the sampling points in the search space. They work on two terminologies: exploitation, the sampling point where surrogate model predicts the higher value of objective and exploration, is the sampling location where there is high uncertainty for prediction.

The objective function f, is sampled at

$$X_t = \text{argmax}_x \, u(x| Đ_{1:t-1})$$

Where u represents the acquisition function and $Đ_{1:t-1}$ are the t-1 samples that have been drawn from the objective function.

3.3. Dataset

It is quite obvious that we need some dataset to work on for detecting fraud. The dataset used for this project is taken from Kaggle [15]. The dataset created is a synthetic dataset that was generated by a simulator called PaySim. The size of the data is about 0.5 GB consisting of more than 6 million rows and 10 columns which contains 9 feature and 1 label column.

The label column is a Boolean type, that represents if the transaction made was fraud or not (0 represents no and 1 represents yes). As it is very normal that most transactions that happen in daily life are usually normal and only a small number of them are fraud (as compared to the authentic ones). The same is the case with this synthetic dataset. Out of the 6 million transactions, about 8 thousand were labelled as fraud. Since, this is a classification problem and we need to train our RF model accordingly, the problem that arises is class imbalance.

3.3.1. Class Imbalance

In comparison to 6 million authentic transactions, the number of fraud transactions becomes negligible. This is known as class imbalance as the data for the labels become skewed. This affects the performance of the machine learning model, as it will not be able to make sound predictions from this data. Therefore, there is a need to balance the data. There are various techniques to balance the imbalance of the data and reduce the bias of the machine learning algorithms. The techniques are discussed as follows:

3.3.1.1. Under-sampling

This technique is useful for the larger datasets. Under-sampling reduces the size of the majority class to match the size of the minority class in the data to balance the dataset. It does so by keeping the samples of the minority class as is and selecting random samples from the majority class.

3.3.1.2. Over-sampling

As the name suggests, this technique is the opposite to under-sampling. It works best when the dataset is smaller in size. To balance the dataset, it increases the size of the minority class to match the size of majority class. To generate more samples of the minority class, different techniques: bootstrapping, repetition, or Synthetic Minority Over-Sampling Technique (SMOTE) are used.

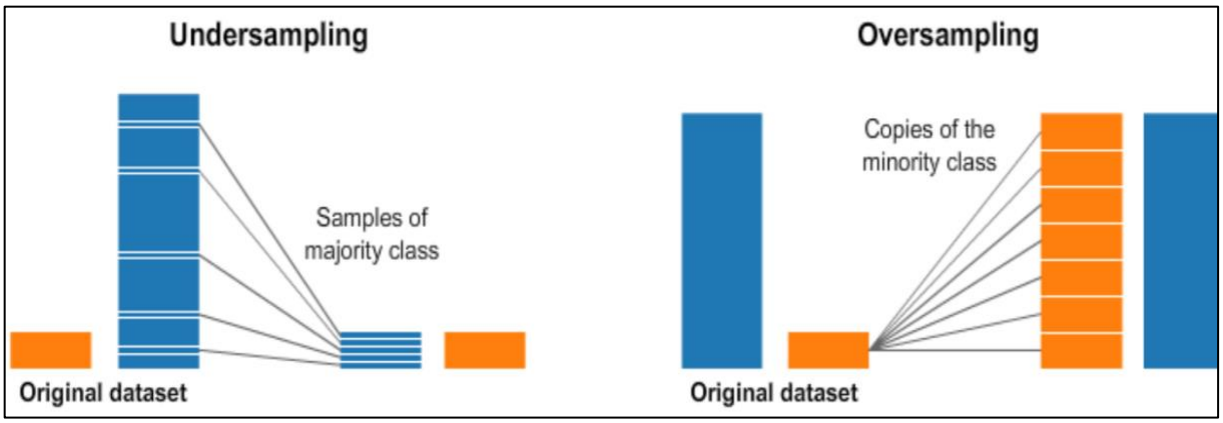

Figure 5. Class Balance: Under sampling vs Oversampling [16]

3.3.2. Cross-Validation

Cross-validation (CV), also known as k-fold CV, is an approach to evaluate the performance of machine learning models. In this procedure, the training set is split into k smaller sets. K here signifies the number of groups that a dataset can be split into. If the value of k is set to 2, we can call it 2-fold CV, k=10 is 10-fold CV and so on.

To start with k-fold CV, the whole dataset is first shuffled randomly and then split into k smaller groups. For each unique group, one group is taken as a hold on the side while the rest of the groups work as a training data. Then, the machine learning model is trained on the training data and evaluated on the test data. Finally, the evaluation score is stored, and the model is discarded. This process repeats k number of times. At the last, we get a CV for each fold and the average of all the CV scores is considered the final result.

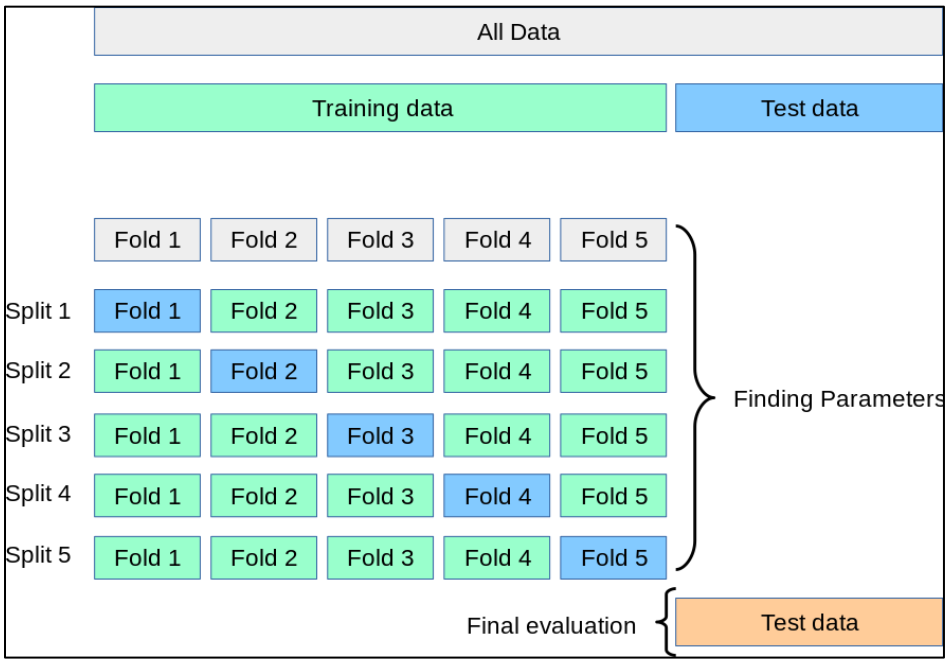

Figure 6. K-fold Cross Validation [17]

3.4. Evaluation Metrics

After the model is trained, the next step is to evaluate how well the model has performed. For this purpose, various performance metrics can be used. The RF model, for the scope of this project, has been evaluated on the following six metrics.

3.4.1. Confusion Matrix

A confusion matrix is a performance metrics that is useful in determining the performance of machine learning models for classification problems. It is N * N matrix, where N represents the number of class labels. In our case, N = 2, as we have two class labels 0 and 1. Following diagram shows how a 2 * 2 confusion matrix would look like.

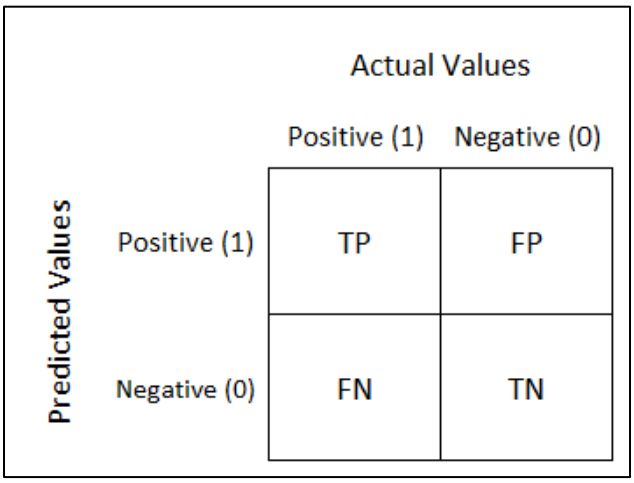

Figure 7. Confusion-matrix [18]

Let us understand what all the values inside the matrix represents.

3.4.1.1. True positive (TP)

If you predict a value to be true and it is actually true, that is considered as a true positive.

3.4.1.2. False positive (FP)

If you predict a value to be true but it is actually false, then you have a false positive.

3.4.1.3. True negative (TN)

You predicted a value to be false and it is infact false, then you have a true negative.

3.4.1.4. False negative (FN)

You predicted a value to be false, but it is actually true, then you have a false negative.

3.4.2. Precision

This measure tells us out of all the classes that our model has predicted as positive/true, are actually true. Basically, this measure tells us how many positive classes has been correctly predicted by our model. We can calculate the precision by the following formula:

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

The higher is the precision, the better the model performs.

3.4.3. Recall

This measure gives us the number of correctly predicted positive classes out of the all the positive classes.

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

The higher is the recall, the better the model performs.

3.4.4. F1-Score

Above, we learned about precision and recall and that the higher values are the better. But what if the model has higher precision and lower recall or vice versa. This is the situation, where F1 score helps in the evaluation.

F1-score, also called F-measure, is the harmonic mean of the precision and recall and can be calculated by this formula:

$$F1 - Score = \frac{2 * Precison * Recall}{Precision + Recall}$$

3.4.5. ROC-AUC

Receiver operating characteristic (ROC) curve is a graph based on TP and FP values. It selects the best classifications models that have good rates of TP and FP. ROC curve is used for the models that could predict the class probabilities. The area under the curve characterizes the performance of the classification model. The more the area under the curve, the better is the performance of the model.

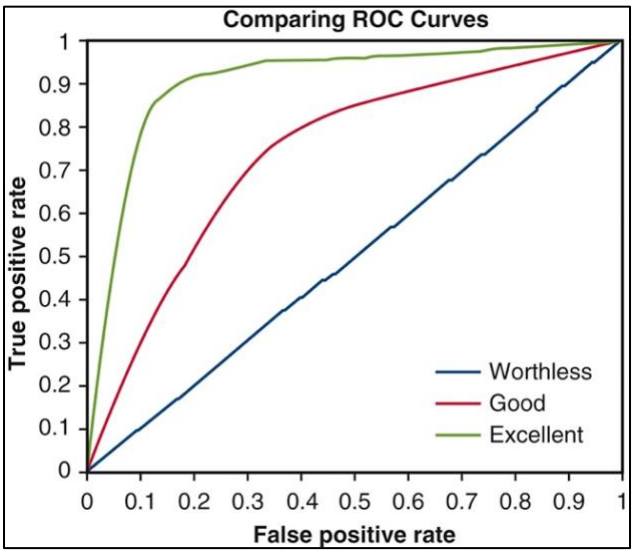

Figure 8. ROC-AUC [19]

4. EXPERIMENTS AND RESULTS

4.1. Experiment

To start with the experiment, the first step is to load and manipulate the data if needed. Then set the base model for the class prediction. As we want to see if the hyperparameters can provide better performance of the model, therefore the next step is to tune the hyperparameters. Finally, after tuning is done, evaluate the performance of the model using various performance metrics. The experiment is split into four parts, which are described in the detail below.

4.1.1. Data Preparation

First, import all of the required libraries and then load the dataset. Count the number of labels for each class. The data is skewed with higher number of samples for the authentic class and lower samples for the fraud class. Therefore, we need to resample the data in order to balance the classes. Because of the large size of the dataset, I have used under-sampling technique to balance the data.

Once, the data has been balanced the next step is to look for numerical vs categorical columns in the data. As categorical variable does not work quite well with the RF classifier, we need to encode them using numbers.

After encoding the categorical variables, finally our data is ready to be used to train the machine learning model. I have used 70:30 ratio to split the training and testing data.

4.1.2. Set the Base Model

After the splitting the dataset into test and train data, the next step is to prepare the base model without any hyperparameter tuning. For doing this, first call the RF classifier and then fit the train data on to the model. This model will serve as the base model for classification with default set of hyperparameters.

4.1.3. Tune the Hyperparameters

After the base model is ready, we will now tune it using three optimization strategies we have discussed before. I have used grid search, Bayesian optimization and genetic algorithm to find the best set of hyperparameters that could provide the best performance of the RF classifier. Thus, having a higher rate of success in detecting fraud.

4.1.4. Evaluate the Performance

Once all the tuning methods have been used to tune the model, the final step is to evaluate how well our model has performed. Also, determine the impact of tuning the hyperparameters and compare it with the base model. As discussed before, we will be using metrics like precision, recall, f-1 score, confusion matrix, roc-auc to evaluate the performance.

4.2. Results

4.2.1. Base Model

The below diagrams shows the confusion matrix for the base model. From the diagram, we can see that, out of total 16,426 samples 14,958 have been predicted correctly. It successfully predicted 8,117 true positives and 6,841 true negatives. This gives us an accuracy of 91.06%.

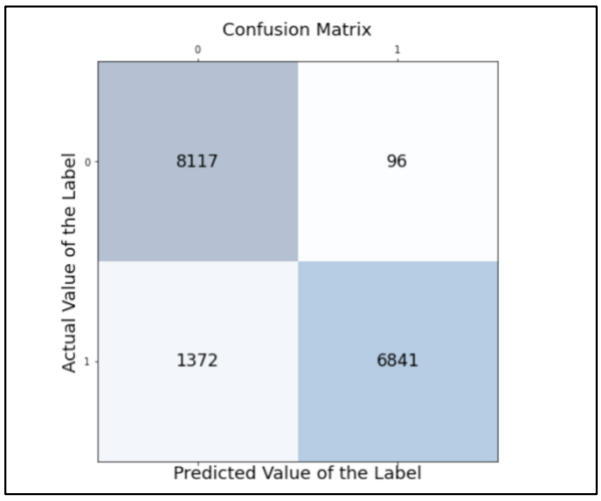

Figure 9. Base Model Confusion-matrix

The following table shows the results for each of the performance metrics analyzed for the base model. Accuracy for the base model turned out to be quite good. For the bigger dataset, CV score becomes more reliable measure, as it gives you the mean of accuracy across the k-folds.

Our base model, shown in Table 1, was successful in achieving an accuracy of 91.06% and a mean CV score of 88.58%. The CV score has a standard deviation of 10.56%. It also achieved close values for recall, precision, and F1-score measures, along with a great area under the ROC curve value of 99.97%. This model had a run time of 15.89 seconds.

Table 1. Results of Base Model.

Metrics	Value
Mean CV Score	0.8858
CV Score Standard Deviation	0.1056
Precision	0.92
Recall	0.91
F1-Score	0.91
ROC	0.9997
Accuracy	0.9106
Duration of run (seconds)	15.89

4.2.2. Bayesian Optimization

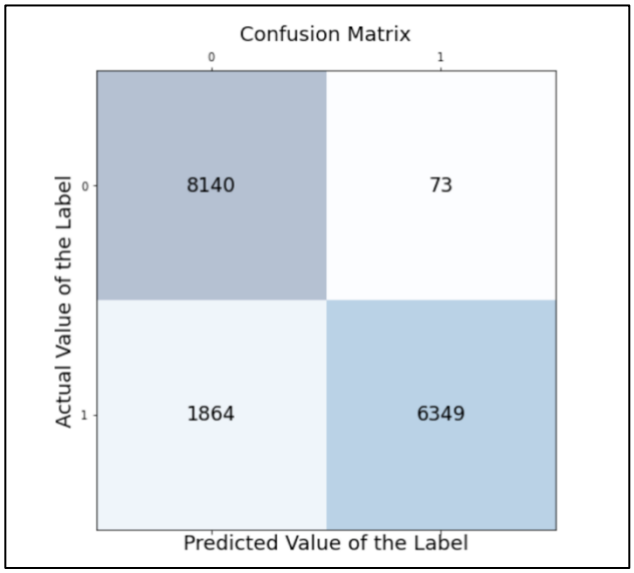

Figure 10. Bayesian Optimization Confusion-matrix

The confusion matrix for the Bayesian optimization as shown in Figure 11, correctly predicted 8,140 true positives and 6,349 true negatives. This gives us an accuracy of 88.20%. All the other measure also achieved lesser success rate as compared to the base model (results shown in Table 2). Therefore, in contrast to our expectation, it turned that BO did not improve the performance of our base model even with hyperparameter tuning.

Table 2. Results of Bayesian Tuning.

Metrics	Value
Mean CV Score	0.8820
CV Score Standard Deviation	0.1680
Precision	0.90
Recall	0.88
F1-Score	0.88
ROC	0.9998
Accuracy	0.8820
Duration of run (seconds)	263.04

4.2.3. Genetic Algorithm

The results of the genetic algorithm were also comparable to that of Bayesian optimization. There was only a slight difference in the values achieved for the performance metrics. The confusion matrix (Figure 11) for the GA, was able to correctly predict 8,125 true positives and 6,361 true negatives, providing the accuracy of 88.18% (which is slightly lesser than the prediction of BO). All the other measures (Table 3) also achieved lesser success rate as compared to the base model. Therefore, in contrast to our expectation, it turned that BO did not improve the performance of our base model even with hyperparameter tuning.

It was able to achieve a mean CV score of 88.26% with a standard deviation of 16.82%. It had similar recall and F1-score of 0.88 along with the precision of 0.90. this method took a run duration of 23,975.11 seconds, which is much higher as compared to the above discussed methods.

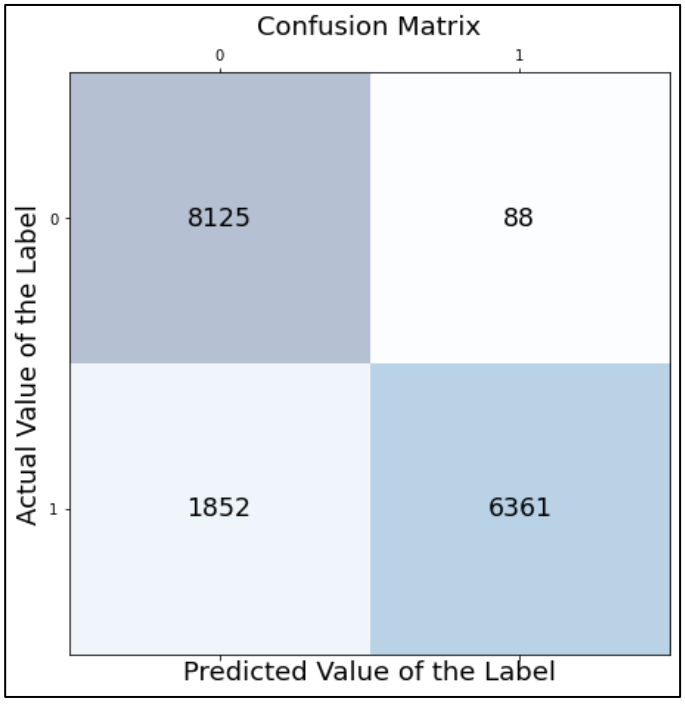

Figure 11. Genetic Algorithm Confusion-matrix

Table 3. Results of Genetic Algorithm.

Metrics	Value
Mean CV Score	0.8826
CV Score Standard Deviation	0.1682
Precision	0.90
Recall	0.88
F1-Score	0.88
ROC	0.9998
Accuracy CV	0.8818
Duration of run (seconds)	23975.11

4.2.4. Grid Search

The confusion matrix for Grid Search in Figure 13 shows a prediction of 8,028 true positives and 7,419 true negatives, that is an accuracy of 94.03%. This approach achieved the highest accuracy of all the other methods. It also achieved a mean CV score of 94.90% with a very less standard deviation of 4.22%.

Grid search had the same score for the precision, recall and F1-score measures (Table 4). Clearly, this method produced the best performance of all the above-described methods. Unfortunately, it was not that good in terms of the run duration, as it was only able to beat the GA. It had a run duration of 12,557.45 seconds, which is nearly half of that of GA, but it is significantly greater than the run duration of BO.

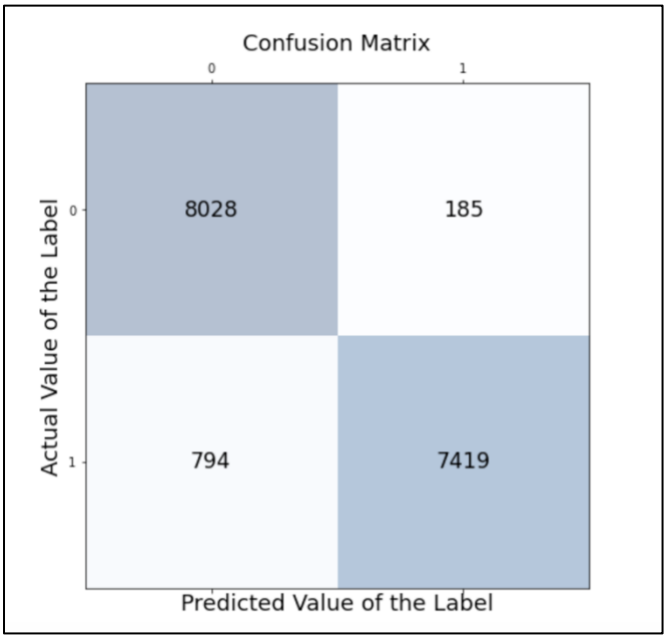

Figure 12. Grid Search Confusion-matrix

Table 4. Results of Grid Search.

Metrics	Value
Mean CV Score	0.9490
CV Score Standard Deviation	0.0422
Precision	0.94
Recall	0.94
F1-Score	0.94
ROC	0.9995
Accuracy CV	0.9403
Duration of run (seconds)	12557.45

4.2.5. Summary

We can see the best hyperparameter values suggested by each tuning method in Table 5. While the best value of N_estimator suggested by GA and BO was close, 140 and 150 respectively, Grid Search provided the best results of the model with only 40 N_estimators. On the other hand, Grid search suggested a higher value for Max_features as compared to GA and

BO. For Max_samples_split, Grid search and GA's best values were quite close, whereas BO chose the value to be extremely less (2).

Table 5. Best Values of Hyperparameters Suggested.

Hyperparameter	Grid Search	Genetic Algorithm	Bayesian Optimization
N_estimators	40	140	150
Bootstap	True	False	False
Max_features	1.0	0.6012	0.6605
Min_samples_split	12	11	2
Min_samples_ leaf	1	6	1

Table 6 summarizes the results of all the methods of experiments for tuning the hyperparameters for fraud detection. From the table below we can see that the highest mean value of the CV score was achieved by Grid Search, 0.94, which is significantly higher than the other methods. Grid Search again performed better than the rest of the techniques in terms of recall (0.94) and F1-score (0.94), with a noticeable difference in the value. All the tuning methods showed comparable results in terms of precision. Grid search outperformed the other methods in terms of accuracy (94.03%) as well, while GA and BO achieved nearly closed accuracies, 88.18% and 88.20%, respectively. The last factor to analyze was the duration of run. BO performed the best in terms of run time, as it only took 263.04 seconds to finish processing, while Grid Search took 12,557.45 seconds and the highest run time was for GA, 23,975.11 seconds.

Table 6. Summary of the Evaluation Metrics.

Metric	Base Model	Grid Search	Genetic Algorithm	Bayesian Optimization
Mean CV Score	0.8858	0.9490	0.8826	0.8820
CV Score Std. Dev.	0.1056	0.0422	0.1682	0.1680
Precision	0.92	0.94	0.90	0.90
Recall	0.91	0.94	0.88	0.88
F1-Score	0.91	0.94	0.88	0.88
ROC	0.9997	0.9995	0.9998	0.9998
Accuracy	0.9106	0.9403	0.8818	0.8820
Run Duration (sec)	15.89	12557.45	23975.11	263.04

5. CONCLUSION

This paper has tried to enhance the performance of the RF model in order to better detect the fraudulent transaction. It used a synthetic dataset of the size of nearly half a GB to train and test the random forest model. To compare if tuning the hyperparameters of the model can provide better performance in fraud detection, we used three techniques to tune the hyperparameters of the model.

Before getting started with the experiments for tuning the hyperparameter, first the dataset was manipulated. Since this dataset represents whether a transaction made online was fraud or not, it is common to have an imbalance between the class labels. This caused the model to be biased in predicting the class labels correctly. Therefore, in order to balance the dataset, under-sampling class rebalancing was used. It was noticed that after the class was balanced, the model was unbiased in predicting the class labels.

All of the three hyperparameter tuning techniques: grid search, genetic algorithm and Bayesian optimization were run to get the results from the random forest classifier. The results showed that the grid search outperformed the rest of the methods in terms of the performance measures except the run time.

Grid search was able to achieve an accuracy of 94.03% and a mean CV score of 94.90%, which is significantly higher than the other tuning methods. It also outperformed in terms of precision, recall and F1-score. The other two methods, GA and BO decreased the accuracy and mean CV score of the base model. On the other hand, they had a better precision, recall and F1-score than the base model. Lastly, grid search did better in all the measures except the run duration but was still better than GA.

6. REFERENCES

[1] S. John, "Realtime Fraud Detection in the Banking Sector Using Data Mining Techniques/Algorithm," *IEEE,* 2016.

[2] A. A. Taha, "An Intelligent Approach to Credit Card Fraud Detection Using an Optimized Light Gradient Boosting Machine," *IEEE,* 2020.

[3] A. Chouiekh, "ConvNets for Fraud Detection analysis," *Procidia Computer Science,* 2018.

[4] Y.-J. Zheng, "Generative adversial network based telecom fraud detection at the receiving bank," *Elsevier,* 2018.

[5] X.-Y. C. Jia Wu, "Hyperparameter Optimization for Machine Learning Models based on Bayesian Optimization," *Journal of ELECTRONIC AND TECHNOLOGY,* 2019.

[6] R. Bardenet, "Collaborative hyperparameter tuning," *Proceedings of Machine Learning Research,* 2013.

[7] L. Li, "A System for Massively Parallel Hyperparameter Tuning," *Conference on Machine Learning and Systems 2020,* 2020.

[8] U. Misra, "RubberBand: Cloud-based Hyperparameter Tuning," *Association for Computing Machinery,* 2021.

[9] T. T. Joy, "Hyperparameter tuning for big data using Bayesian optimisation," *IEEE,* 2016.

[10] W. contributors, "Random forest," *Wikipedia, The Free Encyclopedia.,* 2021.

[11] G. MALATO, "Hyperparameter tuning. Grid search and random search," *Your Data Teacher,* 2021.

[12] Pico, "How does a Genetic Algorithm work?," *Knowledgebase ,* 2021.

[13] V. Mallawaarachchi, "Introduction to Genetic Algorithms — Including Example Code," *Towards Data Science,* 2017.

[14] M. Krasser, "Bayesian optimization," Github, 2018.

[15] E. A. Lopez-Rojas, "PaySim: A financial mobile money simulator for fraud detection," The 28th European Modeling and Simulation Symposium-EMSS, 2016.

[16] R. Agarwal, "The 5 Most Useful Techniques to Handle Imbalanced Datasets," *KDnuggets,* 2020.

[17] F. Pedregosa, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research,* 2011.

[18] S. Narkhede, "Understanding Confusion Matrix," *Towards Data Science,* 9 May 2018.

[19] V. A. Ferraris, "Commentary: Should we rely on receiver operating characteristic curves? From submarines to medical tests, the answer is a definite maybe!," *Plum X Metrics,* 2019.

[20] D. Sarma, "Bank fraud Detection using Community Detection Algorithm," *IEEE,* 2020.

[21] X. Zhuo, "A state of the art survey of data mining-based fraud detection and credit scoring," *EDP Sciences,* 2018.