IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING ALGORITHMS

Mrs. PINNAMARAJU.T.S.PRIYA[*1], Ms. MADDI HINDUDEVI [*2]

[*1]Assistant Professor, Department of Computer Science and Application,
Sanketika Vidhya Parishad Engineering College, P.M. Palem,
Visakhapatnam, Andhra Pradesh

[*2] MCA Student, Department of Computer Science and Application,
Sanketika Vidhya Parishad Engineering College, P.M. Palem,
Visakhapatnam, Andhra Pradesh
maddihindudevi@gmail.com [#2]

ABSTRACT

Credit card fraud is a big issue in the financial services industry. Every year, billions of dollars are lost due to credit card theft. Due to confidentiality concerns, there is a scarcity of research studies analysing real-world credit card data. Machine learning techniques are employed to detect credit card fraud in this article. Standard models are initially employed. Then, hybrid methods based on Ada Boost and majority voting are used. A publicly available credit card data set is used to assess the model's performance. Following that, a real-world credit card data set obtained from a financial institution is analysed. In addition, noise is introduced into the data samples to test the robustness of the algorithms. The experimental results show that the majority voting method provides high accuracy rates in detecting credit card fraud.

KEY WORDS:

Credit card fraud, Ada Boost, Hybrid Models, Machine Learning Algorithm, Confidentiality, Financial Services.

1. INTRODUCTION

Fraud is defined as improper or criminal deception with the intent of gaining financial or personal gain. Two strategies can be employed to avoid fraud losses: fraud prevention and fraud detection. Fraud prevention is a proactive approach that prevents fraud from occurring in the first place. On the other hand, fraud detection is required when a fraudster attempts a fraudulent transaction. Credit card fraud is the unauthorized use of credit card information to make purchases. Credit card transactions can be done both physically and digitally. When conducting physical transactions, the credit card is used. This can happen over the phone or the internet in digital transactions. Cardholders commonly supply the card number, expiration date, and card verification number over the phone or in the mail.

With the advent of e-commerce over the last decade, the use of credit cards has skyrocketed. In 2011, there were over 320 million credit card transactions in Malaysia, which climbed to approximately 360 million in 2015. The number of fraud instances has steadily climbed in tandem with the rise in credit card usage. Despite numerous authorization techniques in place, credit card fraud cases have not been effectively thwarted. Fraudsters prefer the internet because it conceals their identity and location. The surge in credit card fraud has a significant influence on the financial sector. In 2015, global credit card theft totaled a whopping USD $21.84 billion.

2. LITERATURE SURVEY

In this section we will mainly discuss about the background work that is carried out in order to prove the performance of our proposed Method. Literature survey is the most important step in software development process. For any software or application development, this step plays a very crucial role by determining the several factors like time, money, effort, lines of code and company strength. Once all these several factors are satisfied, then we need to determine which operating system and language used for developing the application. Once the programmers start building the application, they will first observe what are the pre-defined inventions that are done on same concept and then they will try to design the task in some innovated manner.

MOTIVATION

BORA MEHAR SRI SATYA TEJA et al (2022) have reported a study using supervised ML techniques. They have used random forest technique to find fraudulent transactions and Accuracy. For classification of the dataset decision trees are used they also defined that random forest would work efficiently by increasing the Accuracy after oversampling and SMOTE is used to increase glasses in data set in balanced way to increase the Accuracy.

Zahra Faraji (2022) has examined Logistic Regression, Decision Tree, Random Forest, XGBoost, KNN and Ensemble, reported that Logistic Regression execution is superior to Decision Tree. XGBoost is quickest and have superior execution. KNN and Logistic Regression have better execution in detecting better and concluded that the model complexity doesn't ensure great performance.

B. N. V. Madhubabu et al (2021) have reported using supervised ML techniques. They have used random forest algorithm for tracking the fraudulent transactions and the Accuracy. For classification of the dataset decision trees are used. Using confusion matrix performance of RFA is evaluated. Although RFA produced good results with small data set, it is not accurate for imbalanced dataset.

Andhavarapu Bhanusri et al (2020) have proposed a study where various machine learning algorithms are compared with each other to evaluate the best classifier. They implemented various machine learning techniques on an imbalanced dataset such as Logistic Regression, naive bayes, random forest with ensemble classifiers utilizing boosting technique. This evaluation is done in view of the calculation's quantitative estimates like Accuracy, Precision, Recall, f1 score, support, confusion matrix. By contrasting the above three techniques, it was concluded that random forest classifier with boosting technique is superior to Logistic Regression and naive bayes techniques. In spite of the fact that Random Forest with Boosting technique performs best in this situation,

utilizing these three techniques we cannot decide the names of fraud and legitimate exchanges for the given dataset using machine learning.

S P Maniraj et al (2019) have reported a study using ML techniques. They have used Local outlier factor to measure the local length of the sample with respect to its neighbors and isolation forest algorithm for the feature selection and to split the data set between the least and greatest qualities for finding fraudulent transactions and we compare the techniques for the testing purpose to determine the Accuracy and Precision of those transactions. They used Jupyter notebook to program in python.

3. EXISTING METHODOLOGY

Three fraud detection approaches are described. To begin, a clustering model is employed to categorise legitimate and fraudulent transactions based on data parameter values. Second, Gaussian mixture models of past and present behaviour can be calculated to detect any deviations from the past. Finally, Bayesian networks are utilised to describe the statistics of a certain user as well as the statistics of other fraud scenarios.

LIMITATION OF EXISTING SYSTEM

1. The high amount of losses due to fraud and the awareness of the relation between loss and the available limit has to be reduced.
2. Testing credit card FDSs using real data set is a difficult task.
3. The fraud has to be deducted in real time and the number of false alert.

4. PROPOSED SYSTEM & ITS ADVANTAGES

For identifying credit card fraud, a total of twelve machine learning algorithms are deployed. Standard neural networks and deep learning models are among the algorithms used. Furthermore, the AdaBoost and majority voting methods are used to create hybrid models. The main contribution of this study is the evaluation of a range of machine learning models for fraud detection using a real-world credit card data set.

ADVANTAGES OF PROPOSED SYSTEM:

The following are the benefits of the proposed system. They are:

1. The results obtained by the several ML algorithms are accurate.
2. The performance of each and every individual ML algorithm differs with one another.
3. The proposed method greatly optimize the processing time.
4. We can finally predict the best algorithm after comparing several ML algorithms.

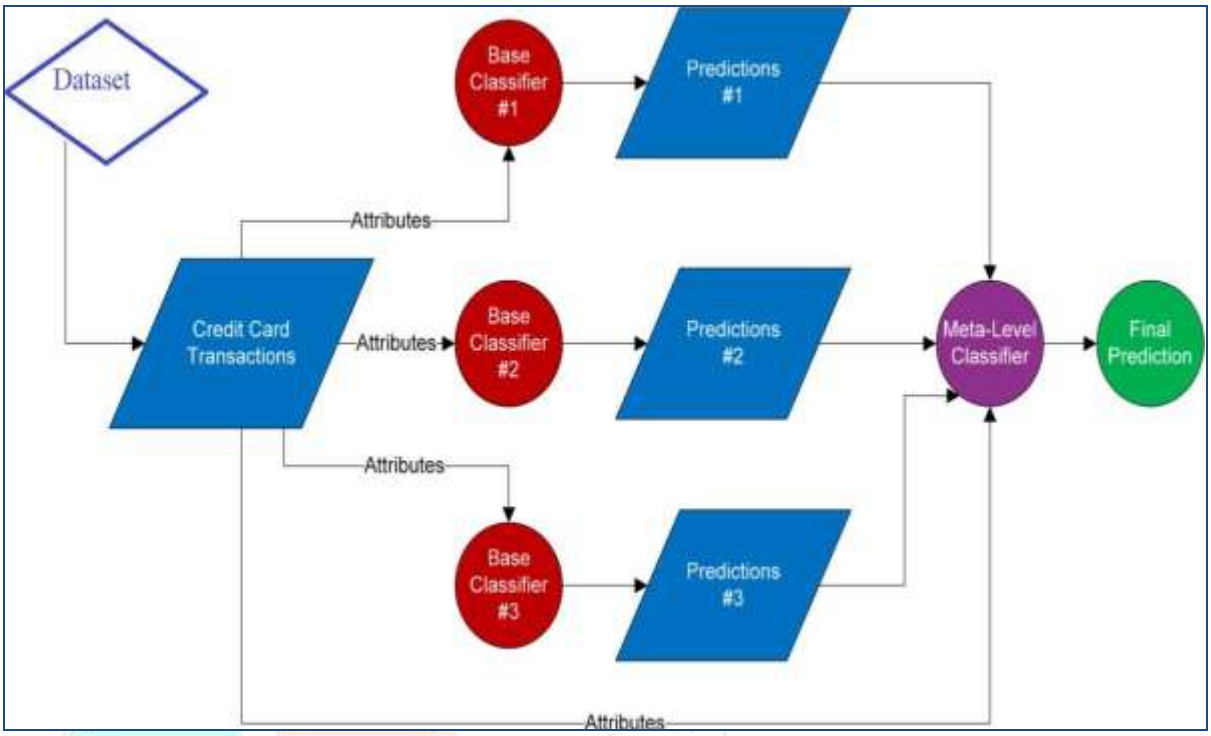


Figure 1. Denotes the Proposed Architecture

Figure 1 clearly represents the proposed architecture in which credit card fraud detection is predicted based on several classifiers. Here multiple classifiers are used to predict and based on the predictions we finally calculated the final prediction.

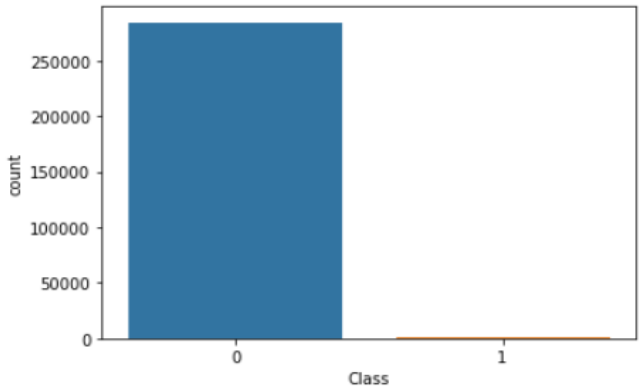
5. PROPOSED METHODOLOGY

In this proposed application we try to implement some ML algorithms to find out the credit card fraud transaction from public dataset.

A. ABOUT THE DATASET

We gathered the data from a public website, Kaggle. This dataset contains 284807 total records. These records are based on two days duration time data. Only a small amount of record is fraud. In the dataset 28 columns are already transformed by principal component analysis (PCA). The amount and time are the features that need to be scaled for acquiring better model. The dataset consists of not fraud class is 284315 and fraud class is 492. The count plot visualizes the imbalance data. The class 1 comprises of 0.172% of total dataset.

```
In [6]: sns.countplot('Class', data=data)
Out[6]: <AxesSubplot:xlabel='Class', ylabel='count'>
```



B. TRAINING AND TESTING

The highly imbalanced data that is typical in such applications, data from the two classes are sampled at different rates to obtain training data with reasonable proportion of fraud to non-fraud cases. As noted earlier, random under sampling of the majority class has been found to be generally better than other sampling approaches. We use random under sampling to obtain training dataset with varying proportions of fraud cases. Performance is observed on a separate Test dataset having 0.5% fraudulent transactions. As described in the data section, dataset has 492 observed fraudulent transactions. We Sampled legitimate transactions from dataset to create varying fraud rates in the modeling and test datasets. In other words, we kept the same number of fraudulent transactions in the modeling datasets, but varied the number of legitimate transactions from dataset to create varying fraud rates. Similarly, the actual fraud rates in the test dataset is 0.5%

ADAPTIVE BOOSTING

Adaptive Boosting is a machine learning meta-algorithm. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner. Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset, AdaBoost with decision trees as the weak learners is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm.

```
In [68]: from sklearn.ensemble import AdaBoostClassifier
         adaboost = AdaBoostClassifier()
         adaboost.fit(X_train,y_train)

Out[68]: AdaBoostClassifier
         AdaBoostClassifier()

In [69]: Y_pred_ad = adaboost.predict(X_test)

In [74]: ad_acc=sklearn.metrics.accuracy_score(y_test, Y_pred_ad)*100
         ad_prec = sklearn.metrics.precision_score(y_test,Y_pred_ad)*100
         ad_recall = sklearn.metrics.recall_score(y_test,Y_pred_ad)*100
         ad_f1 = sklearn.metrics.f1_score(y_test,Y_pred_ad)*100
         ad_Roc_auc = sklearn.metrics.roc_auc_score(y_test,Y_pred_ad)
         print("ADABOOST:")
         print("Accuracy:",ad_acc);
         print("Precision:",ad_prec);
         print("recall:",ad_recall);
         print("f1-score:",ad_f1);
         print("roc_auc:",ad_Roc_auc);
         cm5 = confusion_matrix(y_test,Y_pred_ad)
         print(cm5)
         specificity  = cm5[0,0]/(cm5[0,0]+cm5[0,1])
         sensitivity = cm5[1,1]/(cm5[1,0]+cm5[1,1])
         print("sensitivity:",sensitivity)
         print("specificity:",specificity)

         ADABOOST:
         Accuracy: 96.41666666666666
         Precision: 96.67832167832168
         recall: 95.84055459272098
         f1-score: 96.25761531766753
         roc_auc: 0.9639539768159323
         [[604  19]
          [ 24 553]]
         sensitivity: 0.9584055459272097
         specificity: 0.9695024077046549
```

NAÏVE BAYES

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

LOGISTIC REGRESSION Logistic Regression is a supervised technique helps in classification. It provides output with the help of probabilistic values. The value will be 0 or 1. Logistic Regression follows the cost function called Sigmoid Function.

$$Sigmoid\ Function\ f(x) = \frac{1}{1 + e^{-x}}$$

SUPPORT VECTOR MACHINE

One of the supervised learning method is SVM. We use SVM classification technique called SVC. SVC is called Support Vector Classifier. For the given dataset Support Vector Machine technique creates its hyperplane. The main objective of SVM is creating hyperplane. For the given data, there will be many hyperplanes. Finding the optimized hyperplane is the task of SVM. Hyperplane is created with closest points of different classes. The points which are finalized are called support vectors. Every transaction undergoes through this hyperplane equation to classify the class it belongs to. This is about Support Vector Machine.

CATBOOST

In CatBoost, cat is category and boost is boosting. CatBoost does not required extensive training for providing better output. For credit card fraud detection we need classification so we use CatBoost Classifier. CatBoost develops symmetric trees. When we use the CatBoost Classifier without any parameters it creates 1000 decision trees from 0 to 999. These decision trees are used when we want to predict the output. Each input undergoes these decision trees to predict the class.

```
In [68]: from catboost import CatBoostClassifier
         model = CatBoostClassifier()
         catboost=model.fit(X_train, y_train)
```

```
Learning rate set to 0.015991
0:      learn: 0.6582874        total: 168ms    remaining: 2m 47s
1:      learn: 0.6288097        total: 176ms    remaining: 1m 27s
2:      learn: 0.5998417        total: 184ms    remaining: 1m 1s
3:      learn: 0.5740697        total: 192ms    remaining: 47.8s
4:      learn: 0.5500055        total: 198ms    remaining: 39.5s
5:      learn: 0.5253607        total: 205ms    remaining: 33.9s
6:      learn: 0.5046221        total: 211ms    remaining: 29.9s
7:      learn: 0.4855414        total: 217ms    remaining: 26.9s
8:      learn: 0.4655448        total: 225ms    remaining: 24.8s
9:      learn: 0.4463715        total: 235ms    remaining: 23.2s
10:     learn: 0.4304139        total: 242ms    remaining: 21.7s
11:     learn: 0.4154122        total: 248ms    remaining: 20.4s
12:     learn: 0.3991800        total: 256ms    remaining: 19.4s
13:     learn: 0.3845007        total: 263ms    remaining: 18.5s
14:     learn: 0.3692573        total: 271ms    remaining: 17.8s
15:     learn: 0.3543913        total: 279ms    remaining: 17.1s
16:     learn: 0.3410315        total: 286ms    remaining: 16.5s
17:     learn: 0.3289320        total: 293ms    remaining: 16s
```

```
In [69]: y_pred_cb=model.predict(X_test)
```

```
In [77]: cb_acc=sklearn.metrics.accuracy_score(y_test, y_pred_cb)*100
         cb_prec = sklearn.metrics.precision_score(y_test,y_pred_cb)*100
         cb_recall = sklearn.metrics.recall_score(y_test,y_pred_cb)*100
         cb_f1 = sklearn.metrics.f1_score(y_test,y_pred_cb)*100
         cb_Roc_auc = sklearn.metrics.roc_auc_score(y_test,y_pred_cb)
         print("CATBOOSTING:")
         print("Accuracy:",cb_acc);
         print("Precision:",cb_prec);
         print("recall:",cb_recall);
         print("f1-score:",cb_f1);
         print("roc_auc:",cb_Roc_auc);
         cm4 = confusion_matrix(y_test,y_pred_cb)
         print(cm4)
         specificity = cm4[0,0]/(cm4[0,0]+cm4[0,1])
         sensitivity = cm4[1,1]/(cm4[1,0]+cm4[1,1])
         print("sensitivity:",sensitivity)
         print("specificity:",specificity)
```

```
CATBOOSTING:
Accuracy: 97.58333333333333
Precision: 98.40989399293287
recall: 96.53379549393414
f1-score: 97.46281714785651
roc_auc: 0.9754458635049836
[[614   9]
 [ 20 557]]
sensitivity: 0.9653379549393414
specificity: 0.985553772070626
```

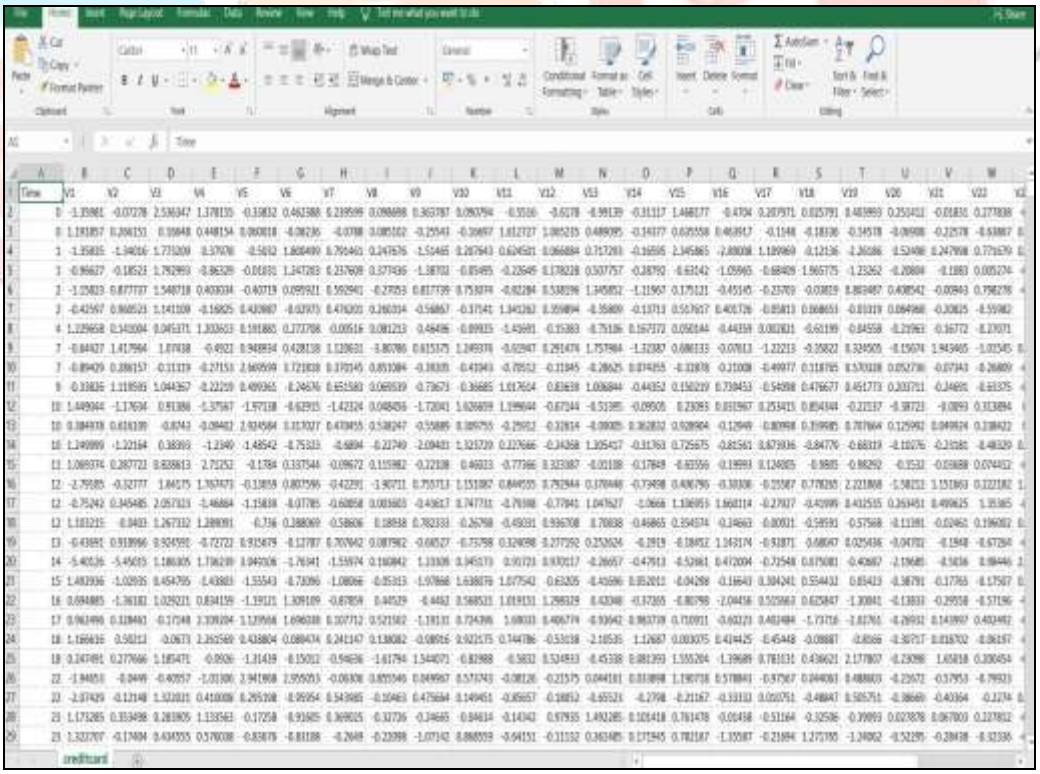
RESULTS

S. No	Algorithm	Accuracy	Precision	Recall	F1-Score	ROC_AUC
1	LOGISTIC REGRESSION	97.25	99.09	95.15	97.08	97.17
2	SUPPORT VECTOR MACHINE	96.42	98.9	93.59	96.17	96.31
3	ADABOOST	96.42	96.68	95.84	96.26	96.4
4	**CATBOOSTING**	**97.58**	**98.41**	**96.53**	**97.46**	**97.54**

6. EXPERIMETAL REPORTS

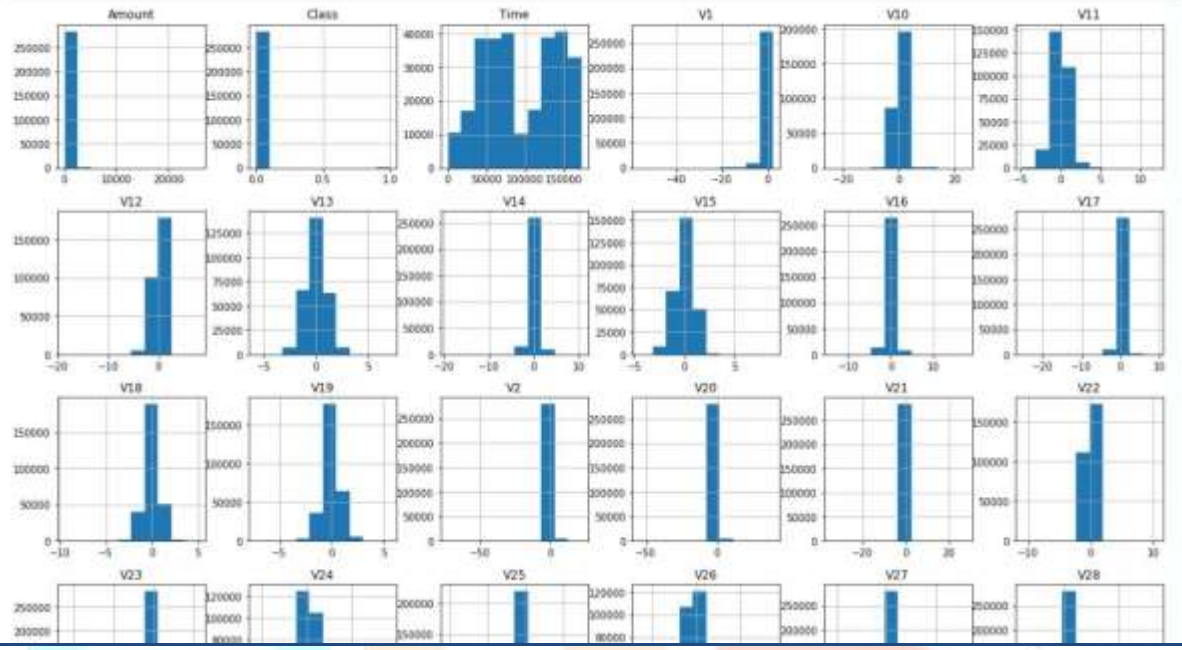
In this proposed application, we try to use google collab as working platform and try to show the performance of our proposed application.

1) DATASET



2) PLOT HISTAGRAM FOR EACH PARAMETER

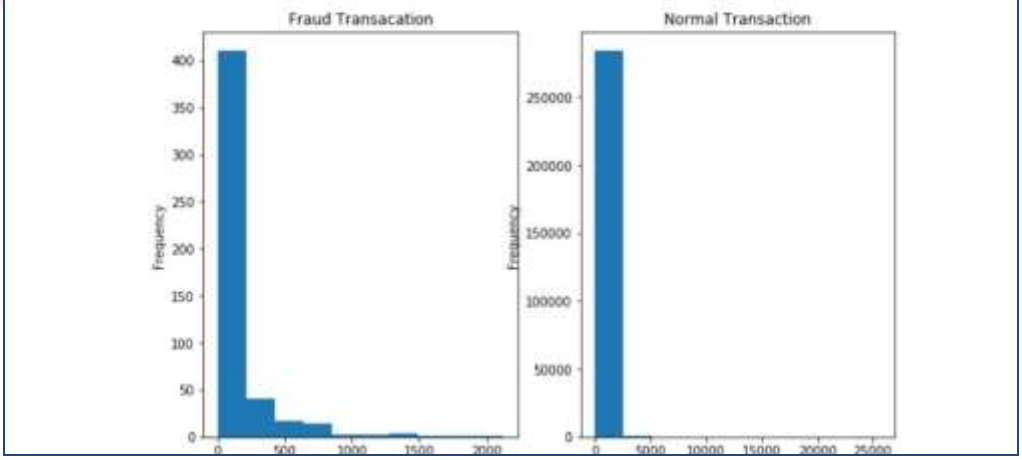
```
In [4]: #plot histogram for each parameter
        data.hist(figsize=(20,20))
        plt.show()
```



3) PLOT FOR NORMAL AND FRAUD TRANSACTIONS

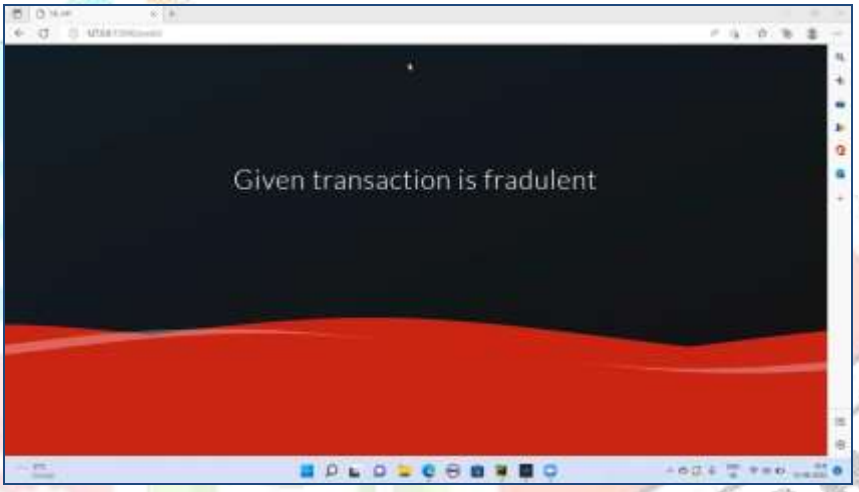
```
In [12]: Fraud_transacation = data[data["Class"]==1]
         Normal_transacation= data[data["Class"]==0]
         plt.figure(figsize=(10,6))
         plt.subplot(121)
         Fraud_transacation.Amount.plot.hist(title="Fraud Transacation")
         plt.subplot(122)
         Normal_transacation.Amount.plot.hist(title="Normal Transaction")

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1d5bd51bd30>
```



4) MAJORITY VOTING

5) TEST THE INPUT



7. CONCLUSION

This project offered credit card fraud detection using machine learning techniques. A variety of standard models, including Nave Bayes and decision trees, were applied. Individual (standard) models and hybrid models using AdaBoost and majority voting combination methods were evaluated using a publicly available credit card data set. Because it considers true and false positive and negative predicted outcomes, the confusion metric has been adopted as a performance measure. For evaluation, an actual credit card data set from a financial institution was also utilised. The same individual and hybrid models were used. Using AdaBoost and majority voting methods, a perfect score of 1 was obtained.

8. REFERENCES

[1] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8292883

[2] J. T. Quah and M. Sriganesh, ―Real-time credit card fraud detection using computational intelligence,‖ Expert Systems with Applications, pp. 1721-1732, 2008.

[3] https://en.wikipedia.org/wiki/Machine_learning

[4] https://en.wikipedia.org/wiki/AdaBoost

[5]https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[6]https://www.kdnuggets.com/2016/04/datacamp-learning-python-data-analysis- data-science.html

[7]https://opensource.com/article/18/9/top-3-python-libraries-data-science

[8]https://www.researchgate.net/publication/323213023_Credit_card_fraud_detection_using_AdaBoost_and_majority_voting

[9] N. Laleh and A. M. Azgomi, ―A Taxonomy of Frauds and Fraud Detection Techniques,‖ICISTM, vol. 31, pp. 256-267, 2009.

[10] E. Aleskerov, B. Fieisleben and R. Bharat, ―CARDWATCH: A Neural Network Based Database Mining System for Credit Card Fraud Detection,‖ Department of Electrical Engineering and Computer Science, University of Siegen, pp. 220-226, 1997.

[11] N. F. R. Centre, ―Bank card and cheque fraud,‖ National Fraud Authority, UK.

9. ABOUT THE AUTHORS

Mrs. PINNAMARAJU.T.S.PRIYA working as Assistant professor in Master of computer application(MCA) in Sanketika Vidya Parishad Engineering College, Visakhapatnam Andhra Pradesh. with 6 years of experience in Masters of Computer Applications (MCA) , accredited by NAAC. With her area of interests in C, DBMS,Computer Organization, Software Engineering, IOT.

Ms. MADDI HINDUDEVI is currently pursuing her 2 years MCA in Department of Computer Science and Applications at Sanketika Vidhya Parishad Engineering College, P.M. Palem, Visakhapatnam, Andhra Pradesh. Her area of interest includes Python, Java, C, and C++.