



A
Project Report
on
CREDIT CARD FRAUD DETECTION
submitted as partial fulfillment for the award of
BACHELOR OF TECHNOLOGY
DEGREE
SESSION 2024-25
in
COMPUTER SCIENCE AND ENGINEERING

By
Mayank Mishra (2100290100095)
Vanshika Mittal (2100290100182)
Yash Kumar Jhunjunwala (2100290100198)
Suryansh Gupta (2100290100171)

Under the supervision of

Dr. Neha Yadav

KIET Group of Institutions, Ghaziabad

Affiliated to
Dr. A.P.J. Abdul Kalam Technical University, Lucknow
(Formerly UPTU)
MAY, 2025

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature

Name: Mayank Mishra

Roll No.:2100290100095

Date: 25th February 2025

Signature

Name: Yash Kumar Jhunhunwala

Roll No.:2100290100198

Date: 25th February 2025

Signature

Name: Vanshika Mittal

Roll No.:2100290100182

Date: 25th February 2025

Signature

Name: Suryansh Gupta

Roll No.:2100290100171

Date: 25th February 2025

CERTIFICATE

This is to certify that Project Report entitled “Credit Card Fraud Detection” which is submitted by Mayank Mishra, Vanshika Mittal, Yash Kumar Jhunjhunwala, Suryansh Gupta in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Dr. Neha Yadav
(Associate Professor)

Dr. Vineet Sharma
(Dean, CSE Department)

Date:

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Dr. Neha Yadav, Department of Computer Science & Engineering, KIET, Ghaziabad, for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only her cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Vineet Sharma, Head of the Department of Computer Science & Engineering, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members, especially faculty/industry person/any person, of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature :

Name: Mayank Mishra
Roll No.:210029010095
Date:25th February 2025

Signature :

Name: Yash Kumar Jhunjhunwala
Roll No.: 2100290100198
Date: 25th February 2025

Signature:

Name: Vanshika Mittal
Roll No.: 2100290100182
Date:25th February 2025

Signature :

Name: Suryansh Gupta
Roll No.:2100290100171
Date: 25th February 2025

ABSTRACT

Credit card fraud usage has been increased greatly with the upcoming never-stopping evolution of communication and in modern technology. Fraud in credit card is an issue as it costs consumers and financial companies quite a few billions yearly, as well as fraudsters always try to research and develop new methods in order to perform illegal actions. Fraud is considered as a major issue that is present in credit card companies. Credit card fraud refers to the physical loss of credit card or loss of sensitive credit card information. As these frauds are happening on a large scale; we need efficient methods like Artificial Intelligence and Machine Learning techniques.

As detecting and preventing frauds is costly, labor-intensive tasks and time-consuming. The most commonly techniques used in detecting fraud are Naïve Bayes (NB), Support Vector Machines (SVM), K-Nearest Neighbor algorithms (KNN). Various weaknesses include unpublished literature and an imbalance classification. Imbalance classification is caused due to lower data entries of observations of minority class compared with the majority in the data set. In this project, we suggest to add a new module of face recognition and detection as a confirmatory test in case of doubtful transactions. Detection of fraud is a set of methodologies that have been taken in order to prevent money or property to be obtained through false instances.

Fraud detection is used many industries like insurance or banking. In banking, frauds can be done by forging checks or by use of lost and robbed credit cards. Other forms of card frauds may involve huge losses or accident while the payout. There are different types of fraud: insurance fraud, credit card fraud, statement fraud, securities fraud etc. In our work we would be focusing on credit card frauds and workout a number of classifier algorithms. We would be considering sampling the data using various algorithms and performing a comparative analysis on the results of classifiers post sampling. Resultant, we hope to retrieve a classification algorithm working best with a particular sampling methodology dependent on the dataset and parameters provided. Lastly addition of a face recognition model is performed to act as a confirmatory test in case of fraud detection by the classifier.

LIST OF FIGURES

| FIGURE NO. | FIGURE NAME | PAGE NO. |
|------------|---|----------|
| 1.1 | Existing Architecture | 2 |
| 1.2 | Activity Diagram | 3 |
| 1.3 | Proposed Architecture | 4 |
| 1.4 | Data Available | 5 |
| 1.5 | Facet plot of state and credit line vs. Fraud Risk | 6 |
| 1.6 | Facet Plot of cardholder and his/her balance vs. Fraud Risk | 6 |
| 1.7 | Heatmap of all variables | 7 |
| 3.1 | Normal Distribution Curve | 9 |
| 3.2 | DT Representation | 10 |
| 3.3 | Random Forest Representation | 11 |
| 3.6 | MNB Algorithm representation | 13 |
| 3.7 | LR representation | 13 |
| 3.8 | SMOTE workflow | 15 |
| 3.9 | Under-sampling workflow | 16 |
| 3.10 | Boosting Algorithm Workflow | 17 |
| 4.1 | Confusion Matrix | 18 |

| | | |
|------|---|----|
| 4.2 | Sample ROC curve | 20 |
| 4.3 | Sample GUI for taking user input | 21 |
| 4.4 | Sample output obtained for above input | 21 |
| 5.1 | Sample Mail Sent | 22 |
| 6.1 | Comparative Analysis of algorithms | 23 |
| 6.2 | ROC Curves of various algorithms without sampling | 24 |
| 6.3 | ROC Curve of various algorithms post SMOTE | 24 |
| 6.4 | GNB Classification Report | 25 |
| 6.5 | RF Classification Report | 25 |
| 6.6 | DT Classification Report | 26 |
| 6.7 | LR Classification Report | 26 |
| 6.8 | LDA Classification Report | 27 |
| 6.9 | QDA Classification Report | 27 |
| 6.10 | RF Confusion Matrix and ROC Curve | 28 |
| 6.11 | GNB Confusion Matrix and ROC Curve | 28 |
| 6.12 | LR Confusion Matrix and ROC Curve | 28 |
| 6.13 | DT Confusion Matrix and ROC Curve | 29 |

LIST OF ABBREVIATIONS

| | | |
|-----|---|----------------------------------|
| KNN | - | K- Nearest Neighbor |
| LR | - | Logistic Regression |
| NB | - | Naïve Bayes |
| MLP | - | Multilevel Perceptron |
| ANN | - | Artificial Neural Network |
| CNN | - | Convolutional Neural Network |
| RF | - | Random Forest |
| DT | - | Decision Tree |
| NN | - | Neural Networks |
| GNB | - | Gaussian Naïve Bayes Multinomial |
| MNB | - | Naïve Bayes Principal Component |
| PCA | - | Analysis Recurrent Neural |
| RNN | - | Networks Linear Discriminant |
| LDA | - | Analysis Quadratic Discriminant |
| QDA | - | Analysis Attribute Selection |
| ASM | - | Measures |

TABLE OF CONTENTS

Page No.

| | |
|---|-----|
| DECLARATION..... | i |
| CERTIFICATE..... | ii |
| ACKNOWLEDGEMENTS..... | iii |
| ABSTRACT..... | iv |
| LIST OF FIGURES..... | v |
| LIST OF ABBREVIATIONS..... | vii |
| | |
| CHAPTER 1 (INTRODUCTION)..... | 1 |
| | |
| 1.1 Inference from Literature Survey | 2 |
| 1.2 Project Description | 3 |
| 1.3 Architectural Design..... | 4 |
| 1.4 Data Loading and Analysis | 5 |
| 1.5 Inference From Data..... | 7 |
| | |
| CHAPTER 2 (LITERATURE REVIEW)..... | 8 |
| | |
| CHAPTER 3 (CLASSIFICATION ALGORITHMS' IMPLEMENTATIONS)..... | 9 |
| | |
| 3.1 Gaussian Naïve Bayes..... | 9 |
| 3.2 Decision Tree Classifier | 10 |
| 3.3 Random Forest Classifier..... | 11 |
| 3.4 Linear Discriminant Analysis..... | 12 |
| 3.5 Quadratic Discriminant Analysis..... | 12 |
| 3.6 Multinomial Naïve Bayes Algorithm..... | 12 |
| 3.7 Logistic Regression..... | 13 |
| 3.8 Sampling Data Algorithm(SMOTE)..... | 14 |
| 3.9 Near Miss Algorithm | 15 |
| 3.10 Boosting Algorithm | 16 |

| | | |
|--|----|----|
| 3.11 Use of Different Performance Metrics..... | 17 | |
| CHAPTER 4 (PERFORMANCE METRICS) | 18 | |
| 4.1 Confusion Matrix | 18 | |
| 4.2 Accuracy | 19 | |
| 4.3 Precision | 19 | |
| 4.4 Recall | 19 | |
| 4.5 F1 Score..... | 19 | |
| 4.6 Area Under Curve | 20 | |
| 4.7 Graphic User Interface..... | 21 | |
| CHAPTER 5 (EXTENSION)..... | 22 | |
| 5.1 SMTP(Simple Mail Transfer Protocol)..... | 22 | |
| CHAPTER 6(RESULTS AND DISCUSSION)..... | 23 | |
| CHAPTER 7 (CONCLUSIONS AND FUTURE SCOPE)..... | 30 | 31 |
| REFERENCES..... | 31 | |
| APPENDICES..... | 32 | |

CHAPTER 1

INTRODUCTION

Fraud detection comprises methods used to prevent fraudulent activities, primarily in banking, finance, and insurance. Common frauds include unauthorized credit card use and forging checks. Increasingly complex fraud methods necessitate advanced detection techniques, such as real-time monitoring.

Statistical data analysis and AI are pivotal in identifying fraud patterns. To evaluate classification algorithms, we performed a 70/30 train-test split on our imbalanced dataset, compared performance parameters, and applied sampling methodologies. A comparative analysis helped identify the best algorithm and sampling technique. Face recognition was added for user identity verification in fraud detection.

Key algorithms include:

Logistic Regression: Classifies binary data based on the relationship between dependent and independent variables.

Random Forest: An ensemble learning method for regression and classification that constructs multiple decision trees.

Decision Trees: Supervised learning algorithm for both classification and regression, using a tree-like structure.

AdaBoost: Boosting algorithm that combines weak classifiers to create a strong classifier.

Gaussian Naïve Bayes: Linear classification method based on supervised learning, assuming data follows a normal distribution.

Sampling methods used:

SMOTE: Oversampling technique creating synthetic class samples to handle data imbalance.

Near Miss Algorithm: Balances class distribution by eliminating majority class instances.

The most accurate algorithm and sampling technique were determined by comparing accuracies, recall, and precision values. This approach ensures effective fraud detection and alerts banks in case of suspicious transaction.

1.1 Inference from Literature Survey:

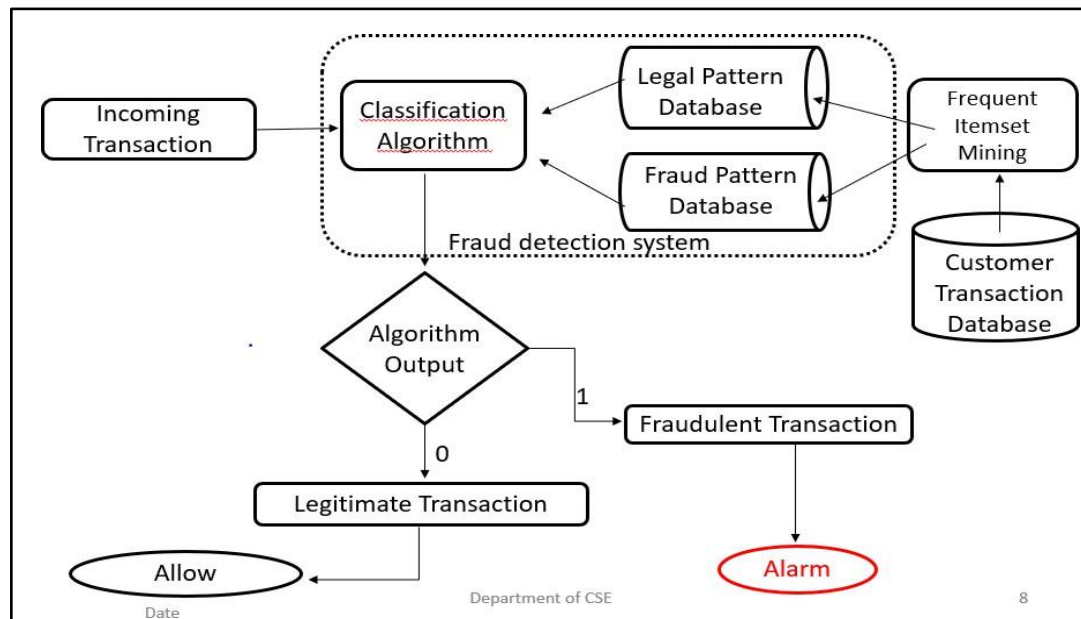


Figure 1.1: Existing Architecture

The image depicts an existing architecture for fraud detection in transactions. It involves several key steps:

- Incoming transactions are processed by a classification algorithm.
- The algorithm uses legal pattern and fraud pattern databases, along with frequent itemset mining from customer transaction data.
- The algorithm outputs a result :0 for a legitimate transaction, which is allowed, or 1 for a fraudulent transaction, which triggers an alarm.
- This system aims to automatically identify and flag potentially fraudulent transactions in real-time

1.2 PROJECT DESCRIPTION:

In this work, we propose a credit card fraud detection model with an added layer of security — face recognition as a confirmatory step. Our focus is on online transactions, given their widespread use and higher vulnerability to fraud. We began with a thorough analysis of transaction data, followed by a comparative study of various machine learning algorithms and the effect of sampling techniques on model performance. To improve reliability, we integrated a face recognition module that activates when a transaction appears suspicious, helping verify the cardholder's identity. In addition to building the model, we identified key limitations in current approaches to guide future research. The architecture of our system is shown in Figure 2, along with structural and behavioral diagrams that explain its working in detail.

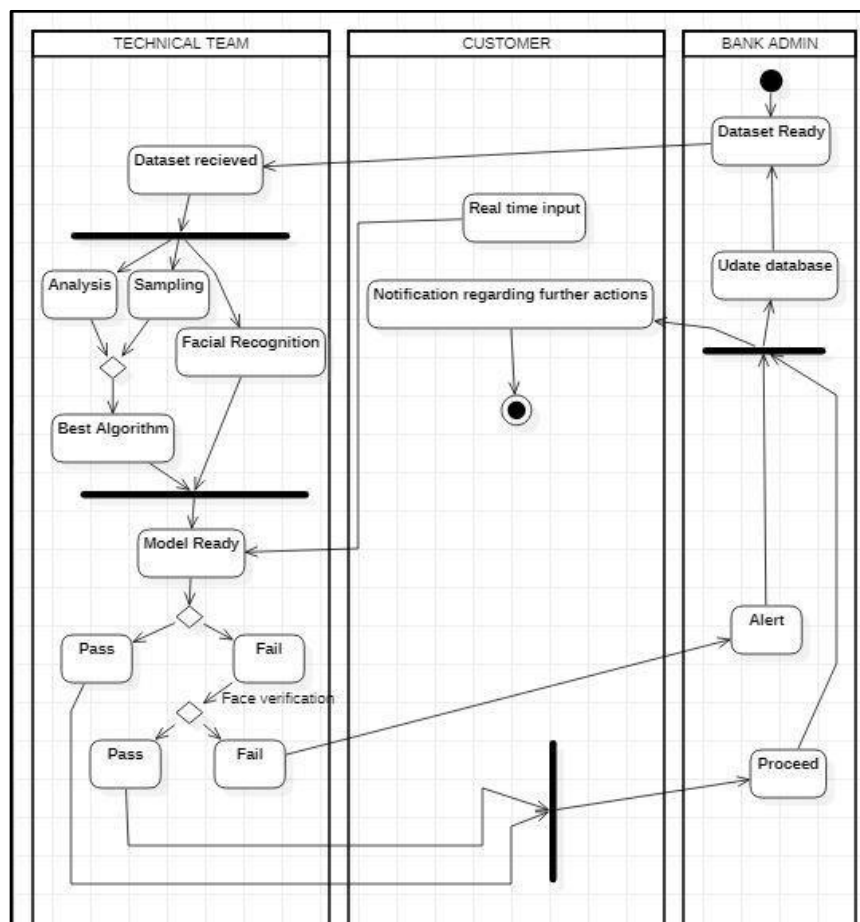


Figure 1.2 Activity Diagram

In the figure 1.2 shows an activity diagram for the project. The starting point is marked with the retrieving of dataset which is analyzed by us, followed by performing classification and sampling algorithms for performing a deep detailed analysis. Face recognition model is also developed. Sample GUI for testing purposes is created. If users' real time input is suspicious the bank is notified and thereon bank can take required steps.

1.3 ARCHITECTURAL DESIGN :

The basic architecture of the proposed model is depicted in figure 1.3. The design comprises of 5 modules. These modules can be listed as under:

- Data Loading and Data Analysis
- Classification Algorithms Implementation
- Sampling Data
- Selecting the most valid algorithm and taking input
- Facial Recognition

It involves thorough study of dataset and parameters, to find out what parameters affect the results maximum. The study helps in figuring out various positive and negative aspects of data. Deep knowledge of dataset in terms of balance between parameters and their importance helps in moving to further steps of choosing rightful algorithms for classification and otherwise. Studying the performances of algorithm without tampering the dataset we get to results of best algorithms. (In case of balanced datasets, completion of module 2 can lead to overlooking of module 3). Module 3 involves sampling and re-applying of classifiers to study affects of various sampling methodologies on the various classification algorithms.

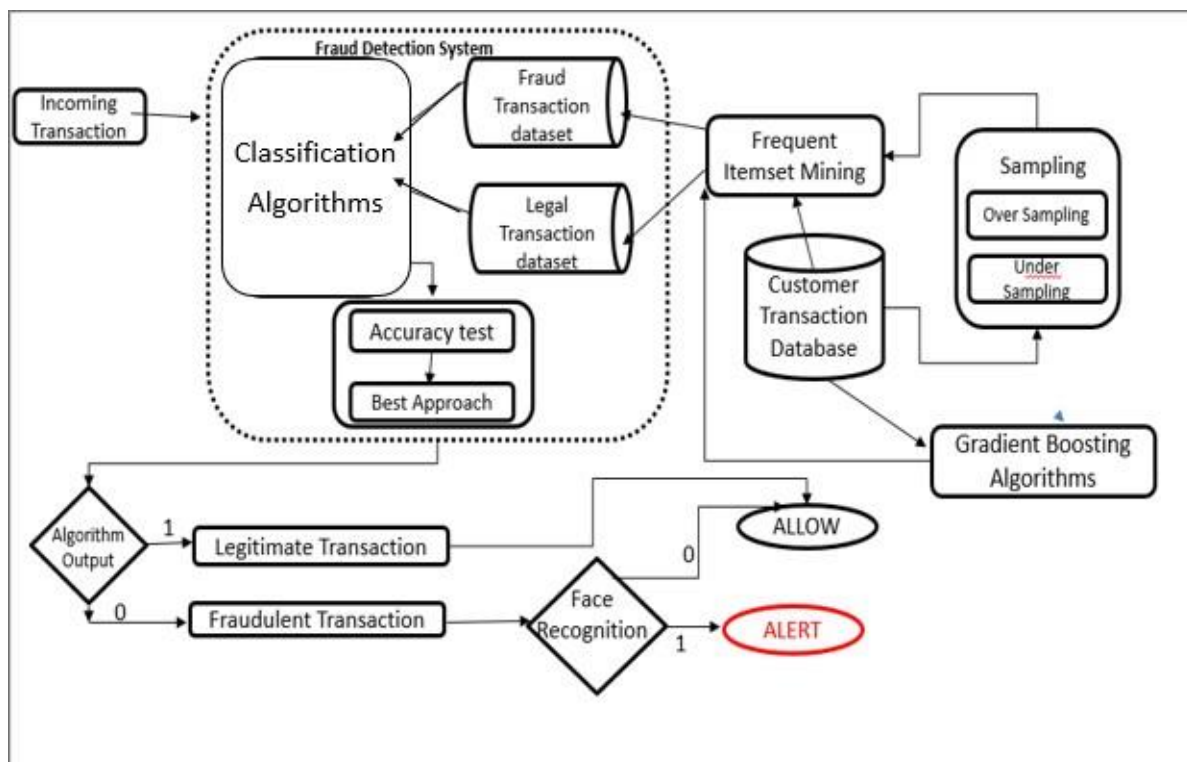


Figure 1.3 Proposed Architecture

A comparative analysis of algorithms is performed. Resultant, we have the best suited sampling methodology and the classifier with maximum performance rate on that sampling

technique. Finally, we create a model to select the best classifier and work with sampled data on the real-time input. If the classifier results into fraud detection, user would be redirected to face testing module to confirm identity. Face recognition and testing is performed in this module. Passing this, leads to normal continuation of transaction and failing in passing this generate an alert to bank.

1.4 DATA LOADING AND ANALYSIS

First the data is downloaded which contains various parameters. The dataset has 9 columns, as shown in figure 1.4, these are:

- custId: A unique customer ID that auto-increments for each entry. It is later dropped from the analysis as it doesn't contribute to fraud detection.
- Gender: Indicates the gender of the customer.
- State: Represents the U.S. state where the customer resides.
- cardHolder: Shows how many credit cards the customer holds, with a maximum of two.
- Balance: The remaining available credit on the customer's card, measured in USD.
- numTrans: The total number of domestic (within the country) transactions made by the customer.
- numIntlTrans: The number of international transactions performed by the customer so far.
- CreditLine: Reflects the credit limit assigned to the customer.
- fraudRisk: This is the target variable. It is binary — a value of 0 indicates a legitimate transaction, while 1 indicates a fraudulent one.

This dataset includes both legitimate and fraudulent transaction records, providing a solid foundation for training and evaluating the fraud detection model.

| | custID | gender | state | cardholder | balance | numTrans | numIntlTrans | creditLine | fraudRisk |
|---|--------|--------|-------|------------|---------|----------|--------------|------------|-----------|
| 0 | 1 | 1 | 35 | 1 | 3000 | 4 | 14 | 2 | 0 |
| 1 | 2 | 2 | 2 | 1 | 0 | 9 | 0 | 18 | 0 |
| 2 | 3 | 2 | 2 | 1 | 0 | 27 | 9 | 16 | 0 |
| 3 | 4 | 1 | 15 | 1 | 0 | 12 | 0 | 5 | 0 |
| 4 | 5 | 1 | 46 | 1 | 0 | 11 | 16 | 7 | 0 |
| 5 | 6 | 2 | 44 | 2 | 5546 | 21 | 0 | 13 | 0 |
| 6 | 7 | 1 | 3 | 1 | 2000 | 41 | 0 | 1 | 0 |
| 7 | 8 | 1 | 10 | 1 | 6016 | 20 | 3 | 6 | 0 |
| 8 | 9 | 2 | 32 | 1 | 2428 | 4 | 10 | 22 | 0 |
| 9 | 10 | 1 | 23 | 1 | 0 | 18 | 56 | 5 | 0 |

Figure 1.4 Data Available

We have used pandas and matplotlib to analyze the data and plotting graphs. The facet plots of variables are drawn followed by heatmap for observation of data. The figure 1.5 shows the Facet plot of state and credit line vs. Fraud Risk. Followed by it, is figure 1.6 showcasing the Facet Plot of cardholder and his/her balance vs. Fraud Risk. These plots are used to study the dependency of fraud risk on the given variables.

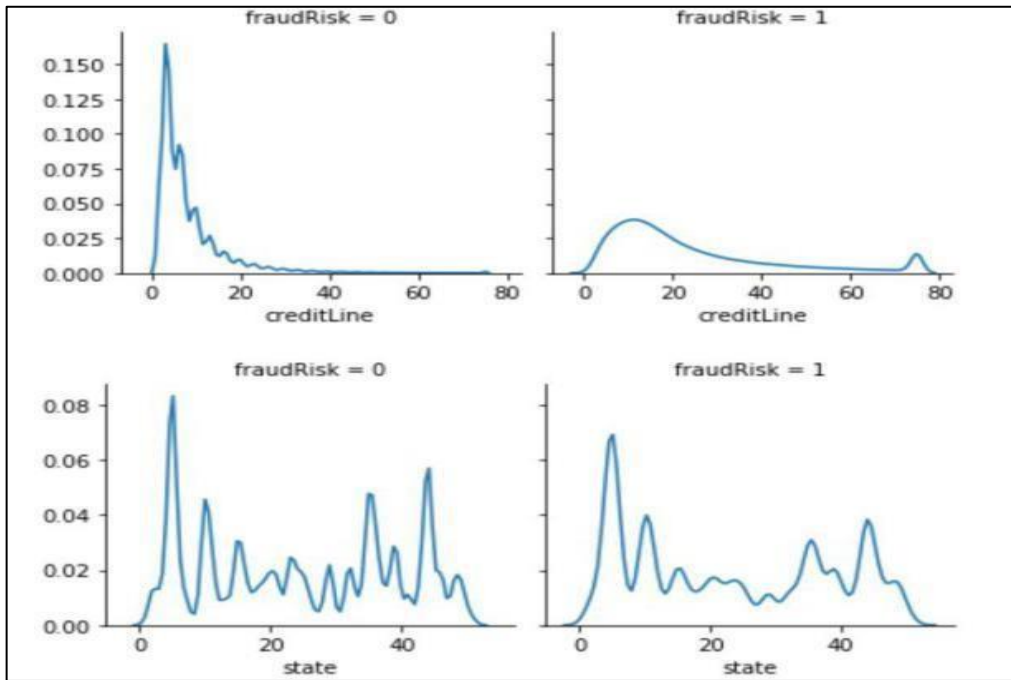


Figure 1.5 Facet plot of state and credit line vs. Fraud Risk

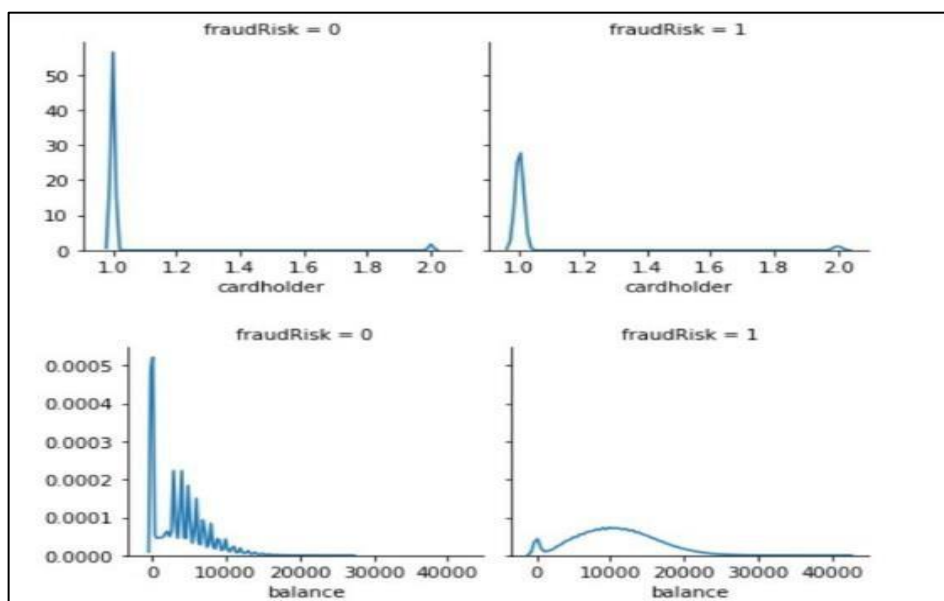


Figure 1.6 Facet Plot of cardholder and his/her balance vs. Fraud Risk

The heatmap below in figure 1.7, shows the correlation of various variables available to us in dataset. We graphically represent data to visualize most impacting areas in dataset. We studied the performances of various classification algorithms without tampering the dataset we get to results of best algorithms.

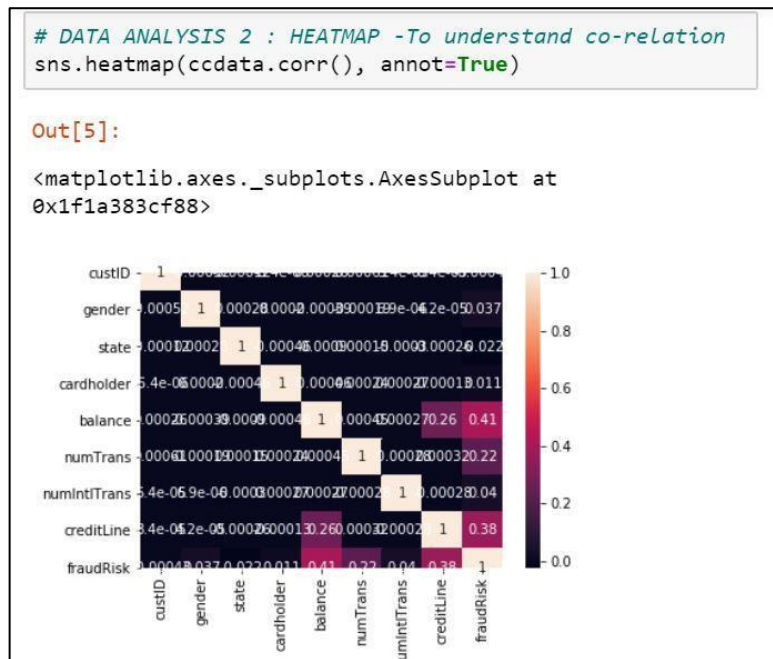


Figure 1.7 Heatmap of all variables

1.5 INFERENCE FROM DATA:

The data that is available is a highly imbalanced data as the no. of legitimate transactions based on the fraud risk is more compared to the no. of fraudulent transactions so it makes certain algorithms to predict the fraud risks becomes difficult. So, it can't be made using normal classifiers but we have to use models in which the precision to check the transaction should be high.

This problem arises when one class of data has very few entries compared to the other. Machine learning algorithms aim at improving accuracy and performance by reducing the error rate, but do not consider imbalance in data. Certain algorithms are biased towards the majority class, making it tougher to detect frauds, which seldom is a majority class. Minority class tends to be ignored and thus are generally misclassified.

CHAPTER 2

LITERATURE REVIEW

A significant amount of research has been dedicated to addressing class imbalance issues in machine learning and data analysis. Several algorithms have been developed to mitigate imbalance effects, though it remains a major challenge [2], [4], [6]. Face recognition presents additional difficulties due to its specificity and complexity, influenced by factors such as noise and camera distortion. Despite advancements in methodologies like OpenCV and face recognition libraries, achieving high accuracy is still challenging. Face recognition holds great promise for verification processes, making it a key area of research [7].

Research highlights various approaches to tackling class imbalance and face recognition accuracy. One study focuses on the imbalance problem, conducting rigorous experiments with classifiers and sampling techniques [1]. Another proposes a creative method for fraud detection in skewed data distributions using Minority class sampling, combining the C4.5 decision tree algorithm, backpropagation, and Naïve Bayes [2]. Advanced methods like false-positive rates are suggested to enhance fraud detection performance [3].

Several studies analyze transaction behavior to predict fraud, discussing machine learning methodologies for credit card fraud detection. Techniques include Bayes' belief networks and neural networks [4]. A comparative analysis of machine learning algorithms such as logistic regression, Naïve Bayes, random forest, and multilayer perceptron determines their suitability for fraud detection based on the dataset [5]. Another study aims to improve processing time and memory cost by focusing on regression methodologies to detect fraud [6].

Outlier detection is addressed by a scoring mechanism for data points within clusters. This methodology shows high potential for identifying outliers and inliers, with precision-recall curves performing better than ROC curves [7]. Another study examines anomaly detection in transactions, using multiple algorithms to recognize fraudulent activities [8]. Recent research also discusses setting up facial recognition models using convolutional neural networks, highlighting improvements in accuracy [9].

Comparative studies of logistic regression, decision trees, and random forests for fraud detection in credit card transactions identify the best-performing algorithms [10]. Credit card fraud often involves the loss or theft of card information. Research data on transactions, including card type, account number, location, time, amount, balance, and credit limit, is used to distinguish fraud from legitimate use and detect noisy data or outliers [12].

Web-based fraud detection models applying machine learning are proposed, aiming to create a live system for transaction verification [12]. Genetic programming, fuzzy Darwinian models, and Hidden Markov Models (HMM) are suggested for training normal user behavior and detecting fraud [13], [14]. Studies also explore artificial immune systems and association rules to identify legal and illegal transactions [19]. Decision trees with cost sensitivity are proposed for more accurate fraud detection [20].

This literature review underscores the complexity and ongoing challenges in fraud detection and face recognition, highlighting the diverse methodologies and algorithms being explored to improve accuracy and Efficiency.

CHAPTER 3

CLASSIFICATION ALGORITHMS' IMPLEMENTATIONS

In this module we add classification algorithms. The algorithms included in model are as follows:

3.1 GAUSSIAN NAIVE BAYES:

Every continuous value which is associated along feature is presumed to be distributed as per Gaussian distribution is done in Gaussian Naive Bayes. A normal distribution is a bell- shaped curve which has a symmetry of the mean of all feature's values which is shown figure 3.1 below.

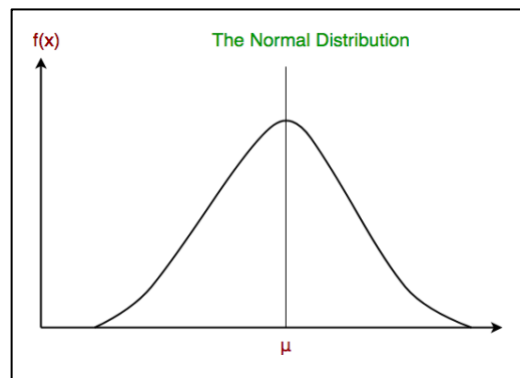


Figure 3.1 Normal Distribution Curve.

The conditional probability is given as likelihood which is considered to be Gaussian. The following equation is used to calculate the probability or likelihood of a situation given another situation/condition. Using scikit-learn library we implement Gaussian Naive Bayes Classifier.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi}\sigma_y} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \quad (4.1)$$

The equation is an extension of Bayes theorem, where $P(x_i|y)$ represents conditional probability. Here, x_i refers to features, variable y is reference to class and variable μ refers to mean value of the given class and σ represents the variance of the class.

3.2 DECISION TREE CLASSIFIER [CART]:

It consists of a tree like structure where the decision rule is represented in a branch, an internal node is represented as a feature (or attribute), the outcome is depicted on each leaf node. The first parent node of the tree is called root node. It self-learns to differentiate depending on the value of attributes. It partitions the tree in a recursively manner called as recursive partitioning. Decision making is done by the help of flowchart like structure shown in figure

3.2.1 The visualization that is formed looks like the representation of DT, because of which DT can be easily interpreted.

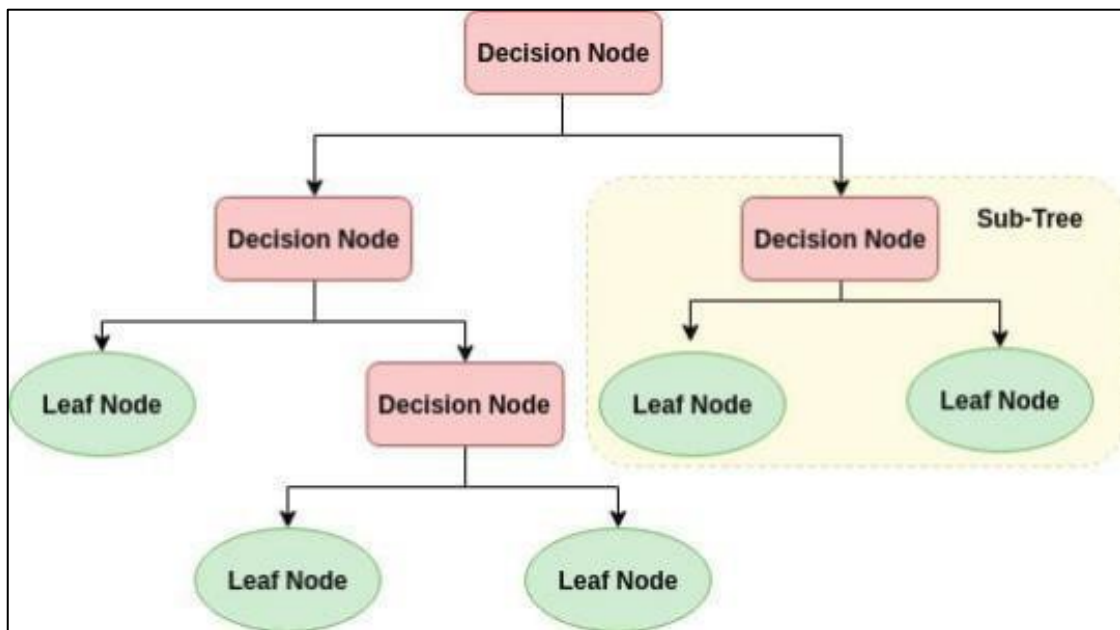


Figure 3.2 DT Representation

Decision Trees (DT) are machine learning algorithms where internal decisions are visible, unlike Neural Networks (NN). DTs train quickly and handle high-dimensional data accurately.

The entropy equation is:

$$H = - \sum p(x) \log p(x)$$

The time complexity of DTs depends on the number of records and features. They are non-distributive and non-parametric, independent of probabilistic assumptions. DTs use Attribute Selection Measures (ASM) to choose the best attribute, split records, and form decision nodes. This process repeats recursively until a pre-condition is met. That should be more concise!

- The attribute value is same and will be present in all the tuples.
- There are no more remaining instances and attributes.

The diagram in figure 3.3 below represents the functioning of decision tree algorithm for classifying the data.

3.3 RANDOM FOREST CLASSIFIER:

It takes a number of steps to execute. Using a given dataset it chooses random samples. Constructs a DT for each sample that has been previously selected to get a predicted outcome for each sample. Check for the highest predicted result. The Final result is selected that has the highest number of votes.

$$RFf_i = \frac{\sum_j \text{norm}f_{ij}}{\sum_{j \in \text{all Features}, k \in \text{all trees}} \text{norm} f_{jk}} \quad (4.3)$$

Here, RFf_i is representing the significance of the particular feature i , as a result of the RF model and $\text{norm}f_{ij}$ is significance of attribute i in the respective tree j , after normalization. This method is part of a group of techniques called ensemble algorithms, which are used to make predictions — like figuring out whether a transaction is fraud or not. Here’s how it works: instead of using just one decision tree (which is like a flowchart that makes decisions), this method creates many of them. Each tree looks at different parts of the data and makes its own guess. Then, all the trees “vote” on the final answer. The answer that gets the most votes is what the system picks. The idea is simple but powerful — it’s like asking a group of people instead of just one. Even if some trees make mistakes, the others can help correct them. Since the trees don’t all think the same way, they balance each other out and make the final result more accurate. A diagram showing how this works is included in the figure below.

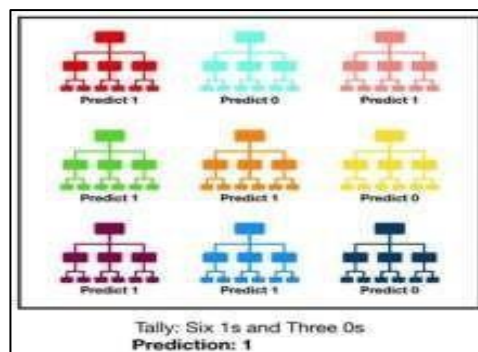


Figure 3.3 Random Forest Representation

3.4. LINEAR DISCRIMINANT ANALYSIS:

LDA is used for deduction of dimensionality in reduced form. It helps in reducing the dimensionality by reducing the dimensionality (i.e. variables) that are dataset also it retains information as much it is possible. The steps involved start from the process to compute inside one class and between two classes scatter matrices. Then calculate the eigenvectors and respective eigen-values for the scatter matrices. Sorting of the eigenvalues and selecting the top k ones. Make a new matrix that have eigen vectors which maps to eigenvalues. In order to get the new feature (i.e. LDA components) which is done by calculating cartesian product of matrix from previous step and data.

3.5 QUADRATIC DISCRIMINANT ANALYSIS:

In QDA no assumption has been made that the covariance of every class will be identical. It requires more computation and data that is required like in the case of linear discrimination. Basically, used for two main objectives: Either we want to check the accuracy of classifiers, given the dependency of the considered objects; or we want to refer to the similar objects under on particular group or in other words, known group of objects. In either case, some dependent assignments are to be highlighted and observed before performing the Discriminant Analysis. Therefore, discriminant analysis can be having both descriptive or predictive objective. Such group assignments, or labeling, may be reached in any way. Thus, this may be worked upon as a functional tool to Principal Components Analysis or Analysis of clusters (in order to observe differences in the results of the latter).

3.6 MULTINOMIAL NAIVE BAYES ALGORITHM:

This classifier works best when the features (or inputs) are in the form of clear, separate values — usually whole numbers that count something. However, it can also handle decimal or fractional values without much trouble. What makes this method unique is that it looks at how each feature is distributed across the data. Instead of just using the values directly, it pays attention to how often certain values appear. As shown in Figure 3.6, the algorithm studies the distribution of all the input variables before making a decision.

$$P(i|j) = \frac{x_{ij} + \alpha}{x_j + |v| + 1}$$

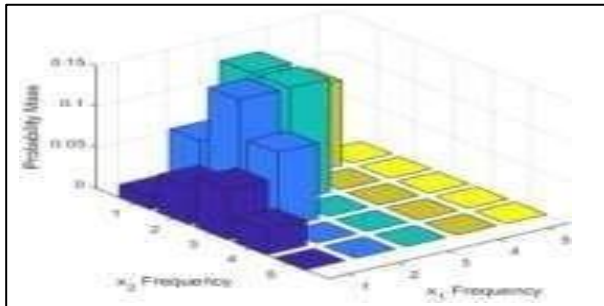


figure 3.6 MNB rep.

3.7 LOGISTIC REGRESSION:

Prediction of binary class for given input using a statistical method, is the basis for LR. The output or target variable that has been formed will have a dichotomous nature, as shown in figure 3.7. Dichotomous indicates that there are only two outcome classes. For instance, it is used in detection of cancer. This is done by computing the probability of happening or presence of an event. Linear regression has a special case in which the target variable will be in categorical in nature. Log of odds are used as a dependent variable. Logistic Regression generally works by making predictions on the probability of presence of any binary event by help of logit function. The equation used in this algorithm is as follows:

$$p = \frac{e^{\sum_{i=0}^n \beta_i X_i}}{1 + e^{\sum_{i=0}^n \beta_i X_i}}$$

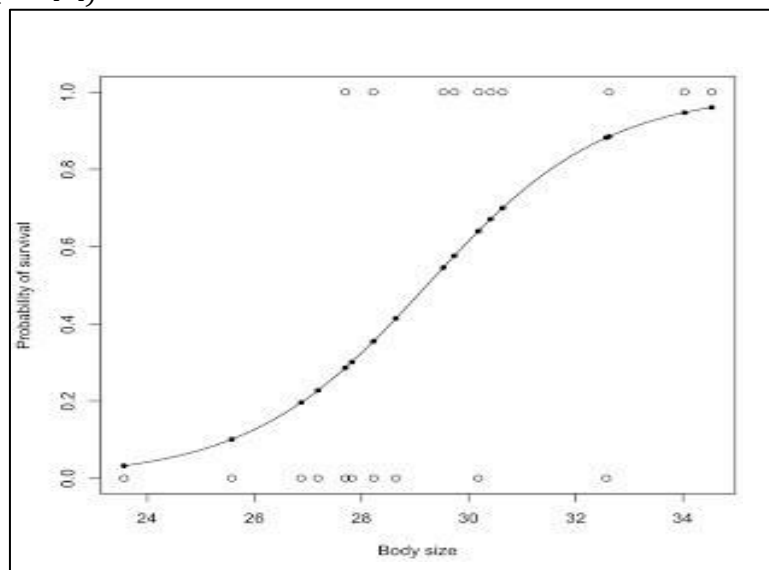


Figure 3.7 Logical Regression representation

3.8 SAMPLING DATA ALGORITHM:

Since this data set which is a very imbalanced one. Thus, we turn to sampling it in order to enhance the results. A dataset is called an imbalanced one if categories in it are having high difference in frequency. The algorithms used are:

SMOTE

This technique (Synthetic Minority Oversampling Technique) has been used to help the in working on the problem of imbalance, by generating synthetic samples of a minority class commonly called as oversampling technique. For training of a classifier this technique obtains a class balanced training set and synthetically class. Instead of recreating the minority observations like defaulters, fraudsters, (SMOTE) helps by creating synthetic observations which is based upon existing minority observations. Shortlisting k number of nearest samples for all observations of the outnumbered class, synthetic samples are created. The data is then re made and many classification models are applied to process it after the process of oversampling has been done. The steps (figure 3.8) involved begin with the process to set the minority class (set A) for each of them we calculate the k number of neighbors for every class by calculating the distance using Euclidean's formula, among all other observations in set A. Due to imbalanced proportion of data a rate of sampling (say, N) has to be set. For every, N observations, we have been randomly selecting, from its nearest k number of neighbors, and set A1 is constructed.

To deal with imbalanced data — where one class has a lot more examples than the other — we can increase the number of entries in the smaller group so it matches the larger one. One smart way to do this is by using a method called SMOTE. Instead of just copying the existing examples from the smaller group, SMOTE creates new, similar ones based on the existing data. It picks a few real examples from the minority class, generates new ones that are close to them, and adds these to the dataset. This process is shown in the figure. The result is a more balanced dataset that gives both classes equal importance. This helps the model learn better, reduces the chances of overfitting, and keeps all the original information intact.

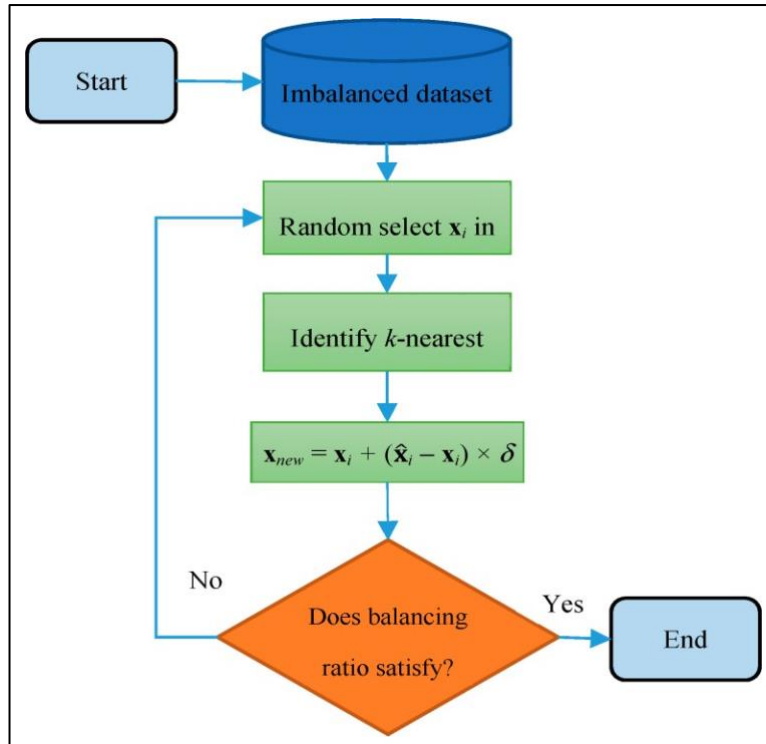


Figure 3.8 SMOTE workflow

3.9 UNDER-SAMPLING TECHNIQUES - NEAR MISS ALGORITHM:

Near Miss is a mathematical model that is basically an under-sampling technique. Its main purpose is to bring equilibrium in the class distributions that is caused due to class imbalance by eliminating majority class examples. We eliminate a number of observations from the outnumbering class, for the sake of increasing the spacing or distances between any randomly selected 2 samples from the same class. This helps in the classification process. Near-neighbor methods are used in order to handle the problem of losing the information in most of the under-sampling techniques.

The basic purpose for using least distant neighbors is explained as follows: The foremost process helps in finding closeness that exists among all observations present between the instances of both the minority and majority class. Particularly in this case, the major task is that the higher-numbered class has to be reduced in number, or in other words, under-sampled. After that, the given number of instances of the majority class having the smallest distance with respect to the minority classes have to be selected. Finally, the higher-ordered class ends up having the number of instances that is a multiple of the observation of the outnumbered class. The basic workflow is shown in figure 3.9 below.

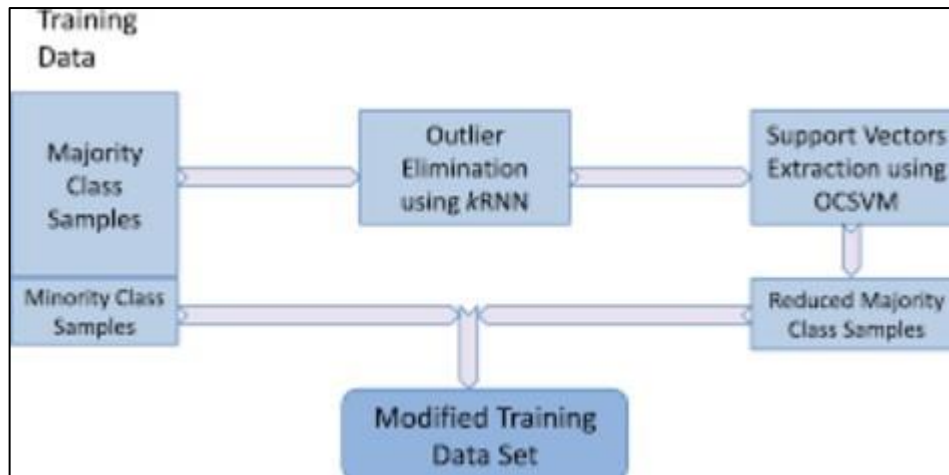


Figure 3.9 Under-sampling workflow

To find a particular number, say n , closest instances of higher ordered class various changes are applied on Near-Miss Algorithm:

- 3.9.1 NearMiss - Version 1: The average distances of the k closest instances that is present in the minority instances is least which is done by selecting samples of majority class.
- 3.9.2 NearMiss - Version 2: The average distance of k number of most distant instances of minority class will be smallest which is done by selecting samples of majority class.
- 3.9.3 NearMiss - Version 3: There are 2 steps for it to work. Firstly, M nearest-neighbors will have to be stored in each minority class instance. The mean of the distance between closest neighbors is observed to be highest after selecting majority class instances.

3.10 BOOSTING ALGORITHMS:

These types of algorithms improve performance by combining several simple models to create a stronger, more accurate one. A simple or “weak” model on its own can easily make mistakes or be affected by small changes in the data, as shown in Figure 3.10. To fix this, the algorithm focuses on the errors made during training. It gives more importance to the cases that were wrongly predicted, so the next model can learn from those mistakes. This process is repeated several times, with each new model trying to improve on the last. Over time, this step-by-step learning reduces errors and helps the final model make better predictions. It works well when the training is done carefully and the rules used in learning are kept straightforward.

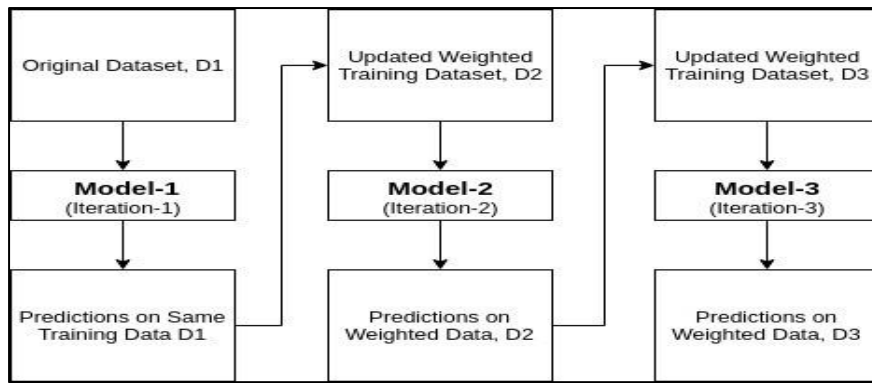


Figure 3.10 Boosting Algorithm Workflow

3.11 USE OF DIFFERENT PERFORMANCE METRICS:

When the data is imbalanced, a classifier might show high accuracy, but this can be misleading. The model may be good at predicting the majority class, but it can easily misclassify the minority class, which is the smaller group. In such cases, accuracy alone doesn't give a true picture of the model's performance. The recall for the minority class might be very low compared to the majority class. To get a better sense of how well the model is really doing, we should focus on other metrics like recall, precision, F1 score, and ROC-AUC. These metrics help us understand how well the model handles both the majority and minority classes. The higher these values are, the better the model is at correctly identifying the minority class and avoiding misclassification. In this paper, we will use these metrics to evaluate the performance of our classifiers, rather than relying on accuracy alone.

To evaluate classifiers post sampling methods, following algorithms are re-run:

3.11.1 GNB

3.11.2 LR

3.11.3 RF

3.11.4 DT

These algorithms are compared based on their accuracies, precision score, recall value and F1 scores. Further, their confusion matrices and ROC plots are used to depict their performances. Precision of a method or a model is found by finding frequency of true positives then dividing it by the summation of all true and false positives. Recall is calculated by finding the total true positives and dividing by adding all true and false negatives, that represents all the elements which belong to the positive class. F1-score it is a very commonly used measure for determining a test's accuracy. It uses both precision as well as recall of the test to compute the score of a model. Confusion matrix is used to check the performance result of a classifier by constructing a simple NxN matrix, where N is the number of class labels. It provides us the values of positive as well as negative cases that were both correctly and incorrectly identified for calculating accuracy. ROC graphs are drawn. The ROC curve generally consists of a graphical plot that is made between sensitivity and (1- specificity). (1- specificity) is considered as false positive rate and sensitivity represents true positive rate. The area under the ROC curve known as (AUC) is also calculated which helps in depicting how well the test can separate the group being tested into the two class labels. Greater the area, the better it will be.

CHAPTER 4

PERFORMANCE METRICS

4.1 CONFUSION MATRIX:

This is a method of visually representing the performance of an algorithm. The classifiers classify the data, then the results are compared and the accuracy and other judgement parameters are calculated using the values depicted by the confusion matrix. Predicted classes are compared against actual classes and the confusion matrix as shown in figure 4.1 are calculated. The 4 values represent:

- 4.1.1 TP (True Positive):** These are the values or classes of instances that were actually true and were correctly classified by the algorithm as positive or true. The estimation and actual values of the class match as positive. This improves the accuracy of the algorithm. For example, the when a fraud is correctly recognized by the algorithm as fraud.
- 4.1.2 TN (True Negative):** These are the negative classes that are estimated correctly to be negative. The estimation and by the classifier rightly classified the instances as negative, thus, matching the actual values. This also improves the accuracy of the algorithm. For example, the when a legal user is correctly recognized by the algorithm as non-fraudulent user.
- 4.1.3 FP (False Negative):** These are the positive class instances that are classified as negative by the classifier. This reduces the accuracy of the classifier. The estimation of the algorithm is observed to be incorrect due to misclassification of a positive class as negative. This is also referred to as the Type 2 error. For example, fraud transaction is passed as a legal one.

| | | Predicted Class | | |
|--------------|----------|-------------------------------------|---|---|
| | | Positive | Negative | |
| Actual Class | Positive | True Positive (TP) | False Negative (FN) Type II Error | Sensitivity $\frac{TP}{(TP + FN)}$ |
| | Negative | False Positive (FP) Type I Error | True Negative (TN) | Specificity $\frac{TN}{(TN + FP)}$ |
| | | Precision $\frac{TP}{(TP + FP)}$ | Negative Predictive Value $\frac{TN}{(TN + FN)}$ | Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$ |

Figure 4.1 Confusion Matrix

4.1.4 FP (False Positive): These are actual negative class is predicted to be positive. The estimation turns out to be incorrect in this case, since, the negative instance is misclassified as positive. This also reduces the accuracy of the algorithm. This is also referred to as the type 1 error. For example, a non-fraudster is classified as fraud.

When these four parameters are comprehended then Accuracy, Precision, Recall and F1 score can be computed. In the equations given below (6.1 to 6.4), TP, TN, FP, FN refer to their usual meanings.

4.2 ACCURACY:

This is a performance measure for algorithms. It represents the correctness of the algorithm. The higher the value the better it is. It marks the correct estimations of positive and negative classes against all estimations. It is most used way to check correctness of algorithms. However, due major imbalance in class in dataset, we cannot rely solely on this performance metric.

$$\text{ACCURACY} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (6.1)$$

4.3 PRECISION:

Precision is a performance metric to judge the actual positives that are estimated correctly with respect to all actual positive values. This is defined as the ratio of correct estimation of positive class instances to all instances of positive class as per their estimated class.

$$\text{PRECISION} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (6.2)$$

4.4 RECALL:

Recall is the proportion of effectively anticipated positive perceptions to all perceptions in real class – yes. This is defined as the ratio of correct estimation of positive class instances to actual instances of positive class as per their actual class.

$$\text{RECALL} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (6.3)$$

4.5 F1 SCORE:

F1 Score is the measure of performance of algorithm. This is most reliable mechanisms it

considers both precision and recall values to calculate the performance of the classifier. It gives exactness of performance as it overcomes the uneven class problem while measuring the performance. Along these lines, at whatever point you fabricate a model, this article should

assist with figuring out what these parameters mean and how great the model has performed. The mathematical formula is as follows:

$$F1 = 2 * \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \quad (6.4)$$

4.6 AREA UNDER CURVE:

It is a greatly advised metric that is been used for evaluation and is mostly used in binary classification problems. The area under curve of a classifying algorithm is calculated by finding the probability that a algorithm will give output randomly for a chosen positive instance more compared to a negative example. We need to calculate two basic terms before calculating AUC.

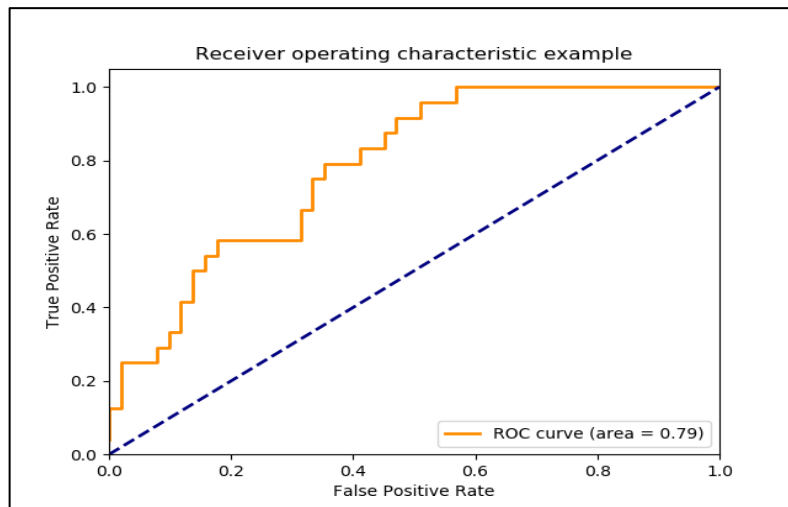


Figure 4.2 Sample ROC curve

Both true and false positive rates lie within the range of [0,1], AUC is basically the area that is under the curve of the plot that is formed False Positive Rate True Positive Rate at different points between [0, 1]. The figure 4.2 above shows a sample ROC curve.

4.7 GRAPHICAL USER INTERFACE:

It is a visual way in which the user interacts with the computer. GUI increases the interaction between the user and computer as it is an easy approach in which users communicate with the system. In our System we have used Tkinter as our GUI so that the user can interact with our application.

TKINTER:

It is a standard GUI library that is present in python. When python is used with tkinter it creates a fast and easy way to interact with the applications that a user has designed. Example as shown in the figure.

This model has 2 parts.

- Selection of best algorithm for the given dataset and parameters available.
- Taking real time input and classifying the transaction as fraud or legitimate. In case of fraud detection by the classifier the user will be redirected to face recognition and detection to confirm identity. Otherwise, the normal transaction continues.

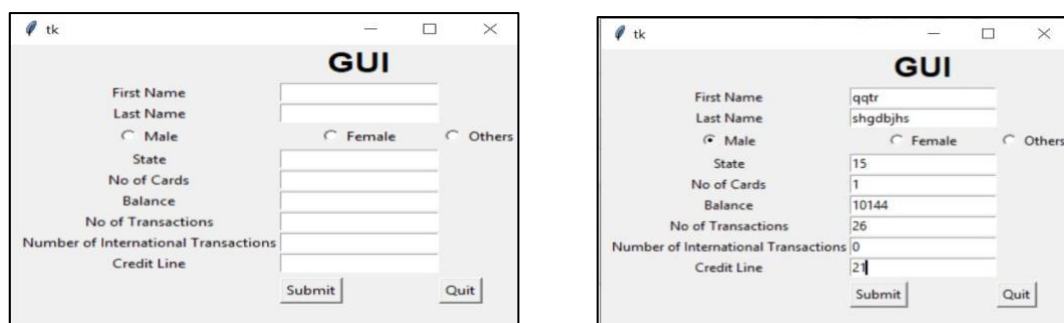


Figure 4.3 Sample GUI for taking user input

For the given input above in figure 4.3 the classifier gives a fraudulent transaction as a result. Thus, Face recognition module is followed.

```
[[1, 15, 1, 10144, 26, 0, 21]]  
[1]
```

Figure 4.4 Sample output obtained for above input

Output 1 represents fraud entry as shown in figure 4.4 above.

CHAPTER 5

EXTENSION

5.1 SMTP: Simple Mail Transfer Protocol:

This protocol is used to handle transfer of emails and routing between mail servers. The python module has a SMTP module/library helps us to define an object for the client session via the protocol, that is performs the function of sending the mail or communicate via other mail services. The following module consist of three things:

5.1.1 Host: It specifies the Internet Protocol address host or a domain name.

5.1.2 Port: In this place the SMTP server will listen to the request.

5.1.3 Local_Hostname: It's an optional thing and is to be specified when the protocol is being run on a sever which is local.

The SMTP object has an instance send mail which has three parameters:

5.1.4 Sender: The sender's mail Id to be specified

5.1.5 Receiver: This has receiver's e-mail Id

5.1.6 Message: The message that we want to send

The following figure 9.1 shows sample email sent as an alert message to the bank.

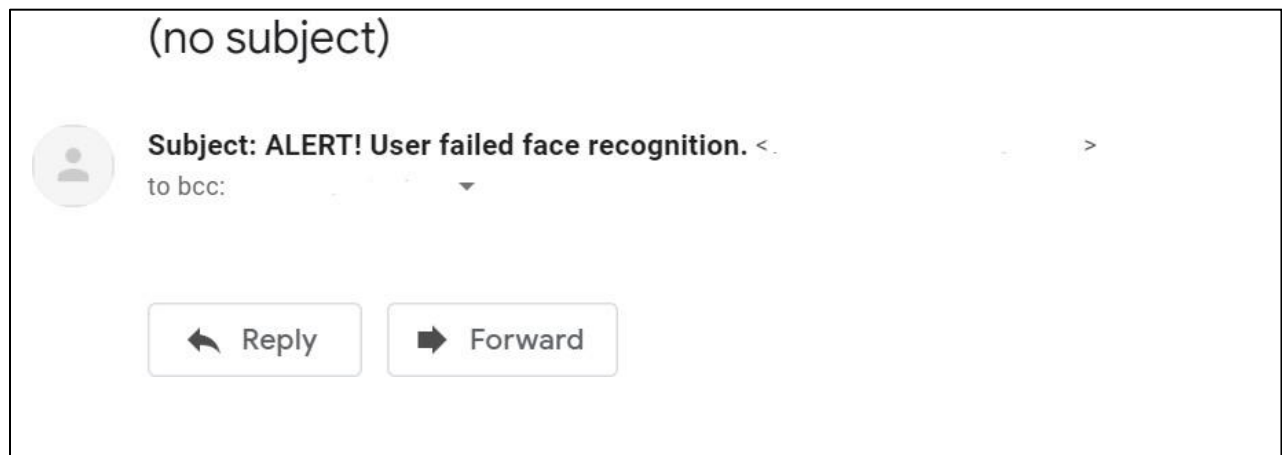


Figure 5.1 Sample Mail Sent

CHAPTER 6

RESULTS AND DISCISSION

One of the biggest challenges we faced was the lack of enough research and reliable datasets. Since fraud data is highly sensitive and can easily be misused, it's tough to get access to real-time data for analysis. On top of that, our dataset didn't have images of users, so we had to create dummy data to fill that gap.

Another major issue we encountered was the class imbalance problem, which can have a big impact on the results. Even though we took steps to improve our classifiers' performance, the imbalance in the data remains a significant challenge.

In this paper, we've looked at several classification algorithms, such as decision trees, random forests, AdaBoost, and autoencoders. We also address the class imbalance problem and propose a real-time fraud detection system.

Fraud is a global issue that affects individuals, businesses, organizations, and banks. Detecting credit card fraud is especially tough because it's hard to tell which transactions are legitimate and which are fraudulent. Our system helps detect fraud in real-time, which can significantly reduce financial losses and help prevent fraud from spreading.

Through our analysis, we found that Multinomial Naïve Bayes gave the highest recall value for detecting fraud compared to other classifiers. Autoencoders also performed well at detecting fraud but had lower accuracy and precision. Neural Networks showed fairly balanced performance, although they had a lower recall for fraud. Logistic Regression and Neural Networks provided the highest accuracy for the given data. You can see the results for these classifiers in Figure 6.1.

| Classifier | Accuracy | P(0) | P(1) | R(0) | R(1) | F(0) | F(1) |
|--|----------|------|------|------|------|------|------|
| GNB | 94 | 97 | 48 | 95 | 56 | 97 | 51 |
| RF | 95 | 97 | 67 | 99 | 45 | 98 | 54 |
| DT | 95 | 96 | 70 | 99 | 36 | 98 | 47 |
| LR | 96 | 97 | 75 | 99 | 44 | 98 | 56 |
| NN | 96 | 97 | 73 | 99 | 48 | 98 | 58 |
| AE | 92 | 97 | 40 | 94 | 76 | 96 | 52 |
| MNB | 71 | 97 | 13 | 71 | 69 | 82 | 22 |
| P(0)- Precision for Legal Transactions | | | | | | | |
| P(1)- Precision for Fraud Transactions | | | | | | | |
| R(0)- Recall for Legal Transactions | | | | | | | |
| R(1)- Recall for Fraud Transactions | | | | | | | |
| F(0)- F1 Score for Legal Transactions | | | | | | | |
| F(1)- F1 Score for Fraud Transactions | | | | | | | |

Figure 6.1 Comparative Analysis of algorithms

The ROC curve shows that Logistic Regression and LDA performed well on this dataset. The confusion matrices further illustrate how accurately each model classified the transactions. in figure 6.2 below.

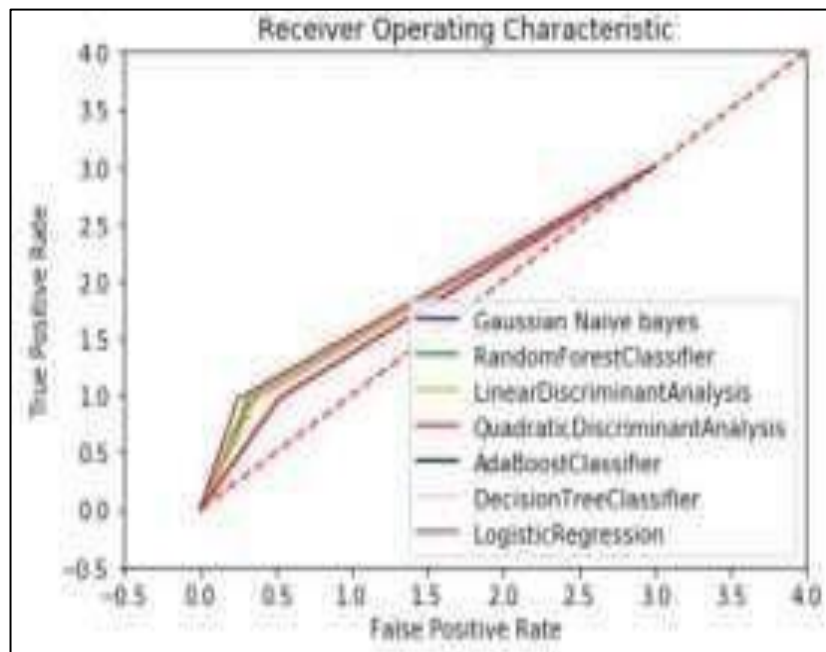


Figure 6.2 ROC Curves of various algorithms without sampling

Because of the class imbalance issue, we focused on metrics like recall, precision, and F1 score to evaluate the classifiers more accurately. To address the imbalance, we applied sampling techniques like SMOTE and NearMiss. The effect of SMOTE on different algorithms can be seen in the ROC curve shown in Figure 6.

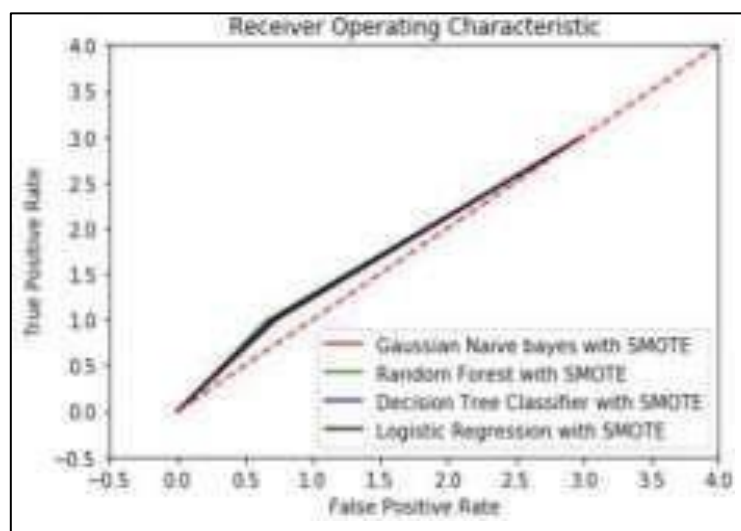


Figure 6.3 ROC Curve of various algorithms post SMOTE

The classification report for the following algorithms are given below. GNB report in figure 6.4 depicts moderate value for precision and recall for fraud transactions.

Followed by diagram 6.5 showing RF results.

| GAUSSIAN NAIVE BAYES | | | | | |
|----------------------|-----------|--------|----------|---------|--|
| | precision | recall | f1-score | support | |
| 0 | 0.97 | 0.96 | 0.97 | 2821165 | |
| 1 | 0.48 | 0.56 | 0.51 | 178835 | |
| micro avg | 0.94 | 0.94 | 0.94 | 3000000 | |
| macro avg | 0.73 | 0.76 | 0.74 | 3000000 | |
| weighted avg | 0.94 | 0.94 | 0.94 | 3000000 | |
| Accuracy: 93.76% | | | | | |

Figure 6.4 GNB Classification Report

| RANDOM FOREST | | | | | |
|------------------|-----------|--------|----------|---------|--|
| | precision | recall | f1-score | support | |
| 0 | 0.97 | 0.99 | 0.98 | 2821165 | |
| 1 | 0.67 | 0.45 | 0.54 | 178835 | |
| micro avg | 0.95 | 0.95 | 0.95 | 3000000 | |
| macro avg | 0.82 | 0.72 | 0.76 | 3000000 | |
| weighted avg | 0.95 | 0.95 | 0.95 | 3000000 | |
| Accuracy: 95.39% | | | | | |

Figure 6.5 RF Classification Report

The recall and precision of DT is highest among other classifiers as seen in classification report of DT in figure 6.6. LR Classification report in figure 6.7 shows a quite low value for recall of fraud.

| DECISION TREE CLASSIFIER | | | | | |
|--------------------------|-----------|--------|----------|---------|--|
| Accuracy: 95.23% | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.96 | 0.99 | 0.98 | 2821165 | |
| 1 | 0.70 | 0.36 | 0.47 | 178835 | |
| micro avg | 0.95 | 0.95 | 0.95 | 3000000 | |
| macro avg | 0.83 | 0.67 | 0.72 | 3000000 | |
| weighted avg | 0.94 | 0.95 | 0.94 | 3000000 | |

Figure 6.6 DT Classification Report

| LOGISTIC REGRESSION | | | | | |
|---------------------|-----------|--------|----------|---------|--|
| Accuracy: 95.83% | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.97 | 0.99 | 0.98 | 2821165 | |
| 1 | 0.75 | 0.44 | 0.56 | 178835 | |
| micro avg | 0.96 | 0.96 | 0.96 | 3000000 | |
| macro avg | 0.86 | 0.72 | 0.77 | 3000000 | |
| weighted avg | 0.95 | 0.96 | 0.95 | 3000000 | |

Figure 6.7 LR Classification Report

| LINEAR DISCRIMINANT ANALYSIS | | | | | |
|------------------------------|-----------|--------|----------|---------|--|
| Accuracy: 95.38% | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.97 | 0.98 | 0.98 | 2821165 | |
| 1 | 0.62 | 0.56 | 0.59 | 178835 | |
| micro avg | 0.95 | 0.95 | 0.95 | 3000000 | |
| macro avg | 0.80 | 0.77 | 0.78 | 3000000 | |
| weighted avg | 0.95 | 0.95 | 0.95 | 3000000 | |

Figure 6.8 LDA Classification Report

| QUADRATIC DISCRIMINANT ANALYSIS | | | | | |
|---------------------------------|-----------|--------|----------|---------|--|
| Accuracy: 93.68% | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.97 | 0.96 | 0.97 | 2821165 | |
| 1 | 0.48 | 0.58 | 0.52 | 178835 | |
| micro avg | 0.94 | 0.94 | 0.94 | 3000000 | |
| macro avg | 0.72 | 0.77 | 0.74 | 3000000 | |
| weighted avg | 0.94 | 0.94 | 0.94 | 3000000 | |

Figure 6.9 QDA Classification Report

We compared the performance of different classifiers both with and without sampling techniques. The results are shown through ROC curves and confusion matrices in the figures below.

These figures help provide a deeper insight into how each algorithm performs. From Figure 6.13, we observe that Logistic Regression without any sampling actually performs better than with sampling. It also has the lowest number of false positives and true negatives among all the models. Interestingly, from Figures 6.11 and 6.13, we can see that Random Forest, which is usually expected to perform better than Decision Trees, shows higher values for both false positives and true negatives in this case.

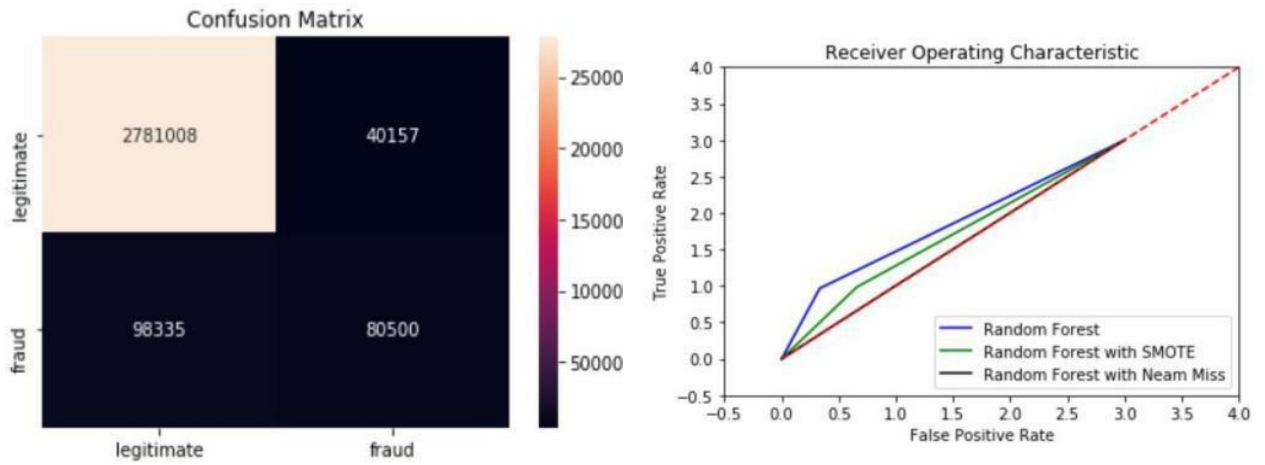


Figure 6.10 RF Confusion Matrix and ROC Curve

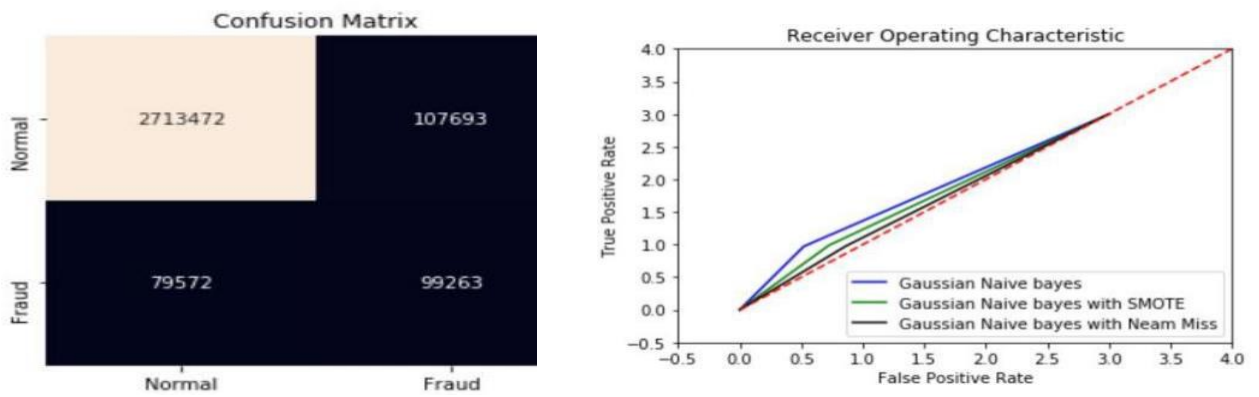


Figure 6.11 GNB Confusion Matrix ROC Curve

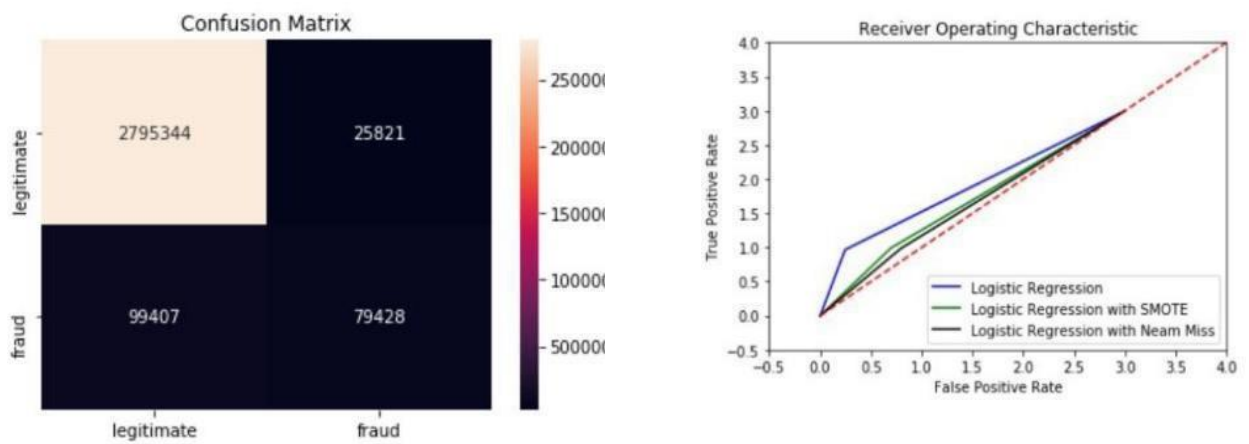


Figure 6.12 LR Confusion Matrix ROC Curve

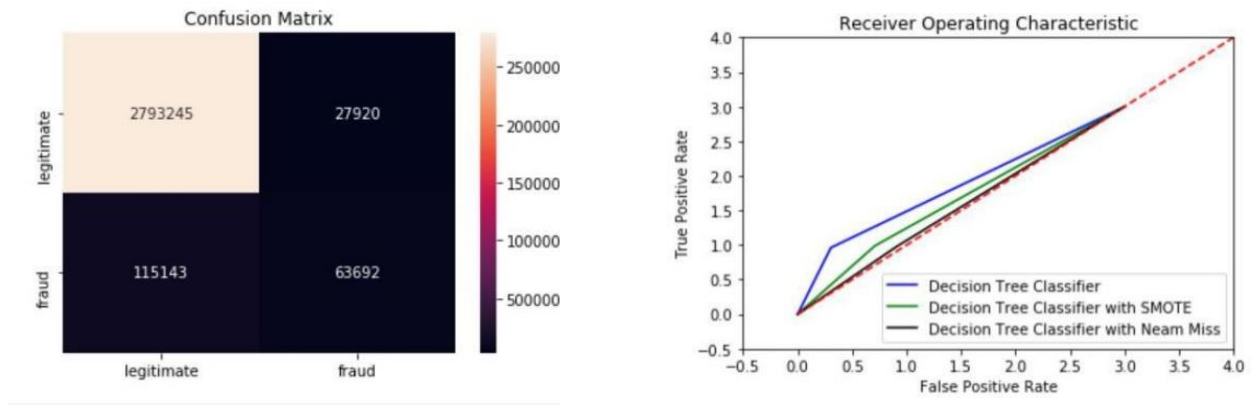


Figure 6.13 DT Confusion Matrix and ROC Curve

The Gaussian Naïve Bayes algorithm shows the lowest error when it comes to wrongly classifying fraudsters as legitimate users. However, it does have a higher rate of mistakenly flagging genuine users as fraudsters, as seen in Figure 6.13. Our system is designed to automatically handle fraud detection. When a transaction is flagged as potentially fraudulent by the classifier, it is sent through an additional step — face detection. If the transaction is considered safe, it moves ahead without any extra checks. If face detection fails, the system alerts the bank via email for further action. This extra layer of verification is done using a face recognition library, which helps confirm the identity of the user and adds more security to the process.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

In this project, we explored and compared several classification algorithms, including decision trees, random forests, AdaBoost, and autoencoders. Our goal was to address the class imbalance problem and build a real-time fraud detection system.

One of the major challenges we faced was the lack of proper datasets and supporting literature. Since fraud-related data is highly sensitive and prone to misuse, accessing real-time data is difficult. Our dataset also lacked user image entries, so we had to generate dummy images to simulate the face recognition component. Additionally, the imbalance between legitimate and fraudulent transactions had a strong impact on the results. Although we applied methods to improve performance, the limited and unbalanced data remained a significant drawback.

Fraud is a serious concern worldwide, affecting individuals, businesses, and financial institutions. Detecting credit card fraud is especially challenging, as it's often hard to tell which transactions are legitimate and which are not. That's why real-time detection is so important — it can reduce the chances of financial loss and help prevent fraud before it spreads.

To improve fraud detection further, a larger and more complete dataset would be needed — one that combines transaction details with face recognition data. Unfortunately, the absence of such a dataset made it harder to train and test our models effectively.

While accuracy is an important metric in our system — especially when deciding whether to trigger face verification — the class imbalance made it clear that we couldn't rely on accuracy alone. Instead, we had to consider other metrics like recall and precision to better evaluate how well our models performed.

Thus, use of other performance metrics became utmost important. With better techniques to resolve class imbalance the outputs can be more precise and reliable. The observation of lower performance metrics of lighter algorithms, if improved could enhance the functioning of the model. Smart integrated GUI at client level would be of great benefit.

Also, higher the accuracy of face verification the better it will be. Face recognition is a developing field and we are expecting better mechanisms for improved accuracy of face verification by the model.

REFERENCES

1. Sara Makki, Zainab Assaghir, IEEE Transaction paper ,“An Experimental Study with Imbalanced Classification Approaches for Credit Card Fraud Detection” ,[2019].
2. C. Phua, D. Alahakoon, and V. Lee, “Minority report in fraud detection: Classification of skewed data,” ACM SIGKDD Explorations Newslett., vol. 6, no. 1, pp. 50–59, 2004.
3. P. Richhariya and P. K. Singh, “Evaluating and emerging payment card fraud challenges and resolution,” Int. J. Comput. Appl., vol. 107, no. 14, pp. 5–10, Jan. 2014.
4. Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, “Credit card Fraud Detection Using Bayesian and Neural Network”’s by Vrije Universiteit Brussel [2015].
5. Dejan Varmedja, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, “Credit Card Fraud Detection- Machine Learning methods”, University of Novi Sad Novi Sad, Serbia,IEEE [2019].
6. An article by Linda Delamaire (UK), Hussein Abdou (UK), John Pointon (UK) “Credit card fraud and detection techniques: a review” [2009].
7. Research paper by Utkarsh Porwal, Smruthi Mukund from eBay Inc “Credit Card Fraud Detection in e-Commerce: An Outlier Detection Approach” [May,2019].
8. Ishu Trivedi , Monika , Mrigya Mridushi ,“ Credit Card Fraud Detection”, published in International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue [January, 2016].
9. Master Thesis by Xavier SERRA , “Face recognition using Deep Learning”, [January,2017].
10. Lakshmi S.V.S.S., Selvani Deepthi Kavila , Research paper on “Machine Learning For Credit Card Fraud Detection System” [2018].
11. H. Moon, "Biometrics Person Authentication Using Projection-Based Face Recognition System in Verification Scenario," in International Conference on Bioinformatics and its Applications. Hong Kong, China, 2004, pp.207-213.
12. Debachudamani Prusti, Santanu Kumar Rath, “Web service based credit card fraud detection by applying machine learning techniques”, IEEE Region10 Conference, TENCON 2019.
13. P. J. Bentley, J. Kim, G.-H. Jung, and J.-U. Choi, “Fuzzy darwinian detection of credit card fraud”, inProc. 14th Annu. Fall Symp. Korean Inf. Process. Soc., Oct. 2000.
14. A. Srivastava, A. Kundu, S. Sural, and A. K. Majumdar, “Credit card fraud detection using hidden Markov model,” IEEE Trans. Depend. Sec. Comput., vol. 5, no. 1, pp. 37–48, Jan./Mar. 2008.
15. Xiao Han, Qingdong Du, “Research on Face Recognition Based on Deep Learning”, IEEE, 2018.
16. J.S.Mishra, S.Panda,and A.K.Mishra, “A novel approach for credit card fraud detection targeting the Indian market,” Int. J. Comput. Sci., vol. 10, no. 3, pp. 172–179, May 2013.
17. A.Brabazon, J.Cahill, P.Keenan,and D.Walsh, “Identifying online credit card fraud using artificial immune systems,” in Proc. IEEE Congr. Evol. Comput., Jul. 2010, pp. 1–7.

APPENDICES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss
ccdata = pd.read_csv('ccFraud.csv')
print(ccdata.info())
ccdata.head(10)
g = sns.FacetGrid(ccdata, col='fraudRisk')
g = g.map(sns.kdeplot, 'creditLine')
g = sns.FacetGrid(ccdata, col='fraudRisk')
g = g.map(sns.kdeplot, 'state')
g = sns.FacetGrid(ccdata, col='fraudRisk')
g = g.map(sns.kdeplot, 'cardholder')
g = sns.FacetGrid(ccdata, col='fraudRisk')
g = g.map(sns.kdeplot, 'balance')
g = sns.FacetGrid(ccdata, col='fraudRisk')
g = g.map(sns.kdeplot, 'numTrans')
g = sns.FacetGrid(ccdata, col='fraudRisk')
g = g.map(sns.kdeplot, 'numIntlTrans')
sns.heatmap(ccdata.corr(), annot=True)
targ_cnt=ccdata.fraudRisk.value_counts()
print('Legal Class 0: ',targ_cnt[0])
print('Fraud Class 1: ',targ_cnt[1])
print('Ratio: ',round(targ_cnt[0]/targ_cnt[1],2),':1')
targ_cnt.plot(kind='bar',title='Count(fraudRisk)')
def plot_data(hue,data):
    for i, col in enumerate(data.columns):
        plt.figure(i)
        sns.set(rc={'figure.figsize':(5,5)})
        ax=sns.countplot(x=data[col],hue=hue, data=data)
hue=ccdata['fraudRisk']
plot_data(hue,ccdata[ccdata.columns[1:8]])
labels = ccdata.columns[1:8]
l1 = ccdata.columns[8]
X = ccdata[labels]
y = ccdata[l1]
X_trn, X_tst, y_trn, y_tst = train_test_split(X, y, test_size = 0.3, random_state = 0)
g = GaussianNB()
g.fit(X_trn, y_trn.ravel())
ypredg = g.predict(X_tst)
rfc = RandomForestClassifier()
rfc.fit(X_trn, y_trn.ravel())
ypredrf = rfc.predict(X_tst)
lda=LinearDiscriminantAnalysis()
```

```

lda.fit(X_trn,y_trn.ravel())
yprb=lda.predict_proba(X_tst)[:,1]
ypredlda=np.where(yprb>0.5,1,0)
qda=QuadraticDiscriminantAnalysis()
qda.fit(X_trn,y_trn.ravel())
yprb1=qda.predict_proba(X_tst)[:,1]
ypredqda=np.where(yprb1>0.5,1,0)
abc = AdaBoostClassifier(n_estimators=20, random_state=0, algorithm='SAMME.R')
abc.fit(X_trn, y_trn)
ypredabc = abc.predict(X_tst)
dtc=tree.DecisionTreeClassifier(criterion='gini',max_depth=3)
dtc.fit(X_trn,y_trn.ravel())
ypreddtc=dtc.predict(X_tst)
lr = LogisticRegression()
lr.fit(X_trn, y_trn.ravel())
ypredlr = lr.predict(X_tst)
print('Gaussian Naive Bayes')
cmg=confusion_matrix(y_tst,ypredg)
print(cmg)
print('#####')
print('Random Forest')
cmr=confusion_matrix(y_tst,ypredrf)
print(cmr)
print('#####')
print('Linear Discriminant Analysis')
cml=confusion_matrix(y_tst,ypredlda)
print(cml)
print('#####')
print('Quadratic Discriminant Analysis')
cmq=confusion_matrix(y_tst,ypredqda)
print(cmq)
print('#####')
print('Ada Boost Algorithm')
cma=confusion_matrix(y_tst,ypredabc)
print(cma)
print('#####')
print('Decision Tree Classifier')
cmd=confusion_matrix(y_tst,ypreddtc)
print(cmd)
print('#####')
print('Logistic Regression')
cml=confusion_matrix(y_tst,ypredlr)
print(cml)
print('#####')
print('GAUSSIAN NAIVE BAYES')
print(classification_report(y_tst, ypredg))
ag = accuracy_score(y_tst, ypredg)
print("Accuracy: %.2f%%" % (ag * 100.0))
print('#####')
print('RANDOM FOREST')
print(classification_report(y_tst, ypredrf))
ar = accuracy_score(y_tst, ypredrf)
print("Accuracy: %.2f%%" % (ar * 100))
print('#####')
print('LINEAR DISCRIMINANT ANALYSIS')
al = accuracy_score(y_tst, ypredlda)
print("Accuracy: %.2f%%" % (al * 100.0))
print(classification_report(y_tst, ypredlda))
print('#####')
print('QUADRATIC DISCRIMINANT ANALYSIS')

```

```

aq = accuracy_score(y_tst, ypredqda)
print("Accuracy: %.2f%%" % (aq * 100.0))
print(classification_report(y_tst, ypredqda))
print('#####')
print('ADABOOST CLASSIFIER')
aa = accuracy_score(y_tst, ypredabc)
print("Accuracy: %.2f%%" % (aa * 100.0))
print(classification_report(y_tst, ypredabc))
print('#####')
print('DECISION TREE CLASSIFIER')
ad = accuracy_score(y_tst, ypreddtc)
print("Accuracy: %.2f%%" % (ad * 100.0))
print(classification_report(y_tst, ypreddtc))
print('#####')
print('LOGISTIC REGRESSION')
al = accuracy_score(y_tst, ypredlr)
print("Accuracy: %.2f%%" % (al * 100.0))
print(classification_report(y_tst, ypredlr))
print('#####')
TP1 = cmg[0][0]
FP1 = cmg[0][1]
FN1 = cmg[1][0]
TN1 = cmg[1][1]
TP2 = cmr[0][0]
FP2 = cmr[0][1]
FN2 = cmr[1][0]
TN2 = cmr[1][1]
TP3 = cml[0][0]
FP3 = cml[0][1] FN3 = cml[1][0]
TN3 = cml[1][1]
TP4 = cmq[0][0] FP4 = cmq[0][1]
FN4 = cmq[1][0]
TN4 = cmq[1][1] TP5 = cma[0][0]
FP5 = cma[0][1]
FN5 = cma[1][0] TN5 = cma[1][1]
TP6 = cmd[0][0] FP6 = cmd[0][1]
FN6 = cmd[1][0]
TN6 = cmd[1][1] FP7 = cml[0][1]
FN7 = cml[1][0] TN7 = cml[1][1]
TP7 = cml[0][0]
tpr1 = TP1 / (TP1+FN1)
fpr1 = 1- (TN1 / (TN1+FP1))
tpr2 = TP2 / (TP2+FN2)
fpr2 = 1- (TN2 / (TN2+FP2))
tpr3 = TP3 / (TP3+FN3)
fpr3 = 1- (TN3 / (TN3+FP3))
tpr4 = TP4 / (TP4+FN4)
fpr4 = 1- (TN4 / (TN4+FP4))
tpr5 = TP5 / (TP5+FN5)
fpr5 = 1- (TN5 / (TN5+FP5))
tpr6 = TP6 / (TP6+FN6)
fpr6 = 1- (TN6 / (TN6+FP6))
tpr7 = TP7 / (TP7+FN7)
fpr7 = 1- (TN7 / (TN7+FP7))
plt.title('Receiver Operating Characteristic')
plt.plot([0,fpr1,3],[0, tpr1,3], 'b', label='Gaussian Naive bayes')
plt.plot([0,fpr2,3],[0, tpr2,3], 'g', label='RandomForestClassifier')
plt.plot([0,fpr3,3],[0, tpr3,3], 'y', label='LinearDiscriminantAnalysis')
plt.plot([0,fpr4,3],[0, tpr4,3], 'r', label='QuadraticDiscriminantAnalysis')
plt.plot([0,fpr5,3],[0, tpr5,3], 'black', label='AdaBoostClassifier')

```

```

plt.plot([0,fpr6,3],[0, tpr6,3], 'pink', label='DecisionTreeClassifier')
plt.plot([0,fpr7,3],[0, tpr7,3], 'brown', label='LogisticRegression')
plt.legend(loc='lower right')
plt.plot([0,4],[0,4],'-r-')
plt.xlim([-0.5,4.0])
plt.ylim([-0.5,4.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
print("Before OverSampling, counts of label '1': {}".format(sum(y_trn == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_trn == 0)))
sm = SMOTE(random_state = 2)
X_trn_res, y_trn_res = sm.fit_sample(X_trn, y_trn.ravel())
ccdata_sm = pd.concat([X_train_res, y_train_res ])
print(type(X_trn_res))
ccdata_sm = np.concatenate((X_trn_res, y_trn_res[:,None]), axis=1)
ccdata_sm
print('After OverSampling, the shape of train_X: {}'.format(X_trn_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_trn_res.shape))
print("After OverSampling, counts of label '1': {}".format(sum(y_trn_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_trn_res == 0)))
g_s = GaussianNB()
g_s.fit(X_trn_res, y_trn_res.ravel())
ypredgnbs = g_s.predict(X_tst)
rf_s = RandomForestClassifier()
rf_s.fit(X_trn_res, y_trn_res.ravel())
ypredrfs = rf_s.predict(X_tst)
d_s=tree.DecisionTreeClassifier(criterion='gini',max_depth=3)
d_s.fit(X_trn_res, y_trn_res.ravel())
ypreddtcs=d_s.predict(X_tst)
l_s = LogisticRegression()
l_s.fit(X_trn_res, y_trn_res.ravel())
ypredlrs = l_s.predict(X_tst)
nr = NearMiss()
X_trn_miss, y_trn_miss = nr.fit_sample(X_trn, y_trn.ravel())
print('After Undersampling, the shape of train_X: {}'.format(X_trn_miss.shape))
print('After Undersampling, the shape of train_y: {} \n'.format(y_trn_miss.shape))
print("After Undersampling, counts of label '1': {}".format(sum(y_trn_miss == 1)))
print("After Undersampling, counts of label '0': {}".format(sum(y_trn_miss == 0)))
g_nm = GaussianNB()
g_nm.fit(X_trn_miss, y_trn_miss.ravel())
ypredgnbnm = g_nm.predict(X_tst)
r_nm = RandomForestClassifier()
r_nm.fit(X_trn_miss, y_trn_miss.ravel())
ypredrfnm = r_nm .predict(X_tst)
d_nm=tree.DecisionTreeClassifier(criterion='gini',max_depth=3)
d_nm.fit(X_trn_miss, y_trn_miss.ravel())
ypreddtcnm=d_nm.predict(X_tst)
l_nm = LogisticRegression()
l_nm.fit(X_trn_miss, y_trn_miss.ravel())
ypredlrnm = l_nm.predict(X_tst)
print('I. GAUSSIAN NAIVE BAYES')
print('Before Sampling : ')
print(classification_report(y_tst, ypredg))
print("Accuracy: %.2f%%" % (ag * 100.0))
print('After Sampling [SMOTE]: ')
cmgs=confusion_matrix(y_tst,ypredgnbs)
print(cmgs)
print(classification_report(y_tst, ypredgnbs))
ags = accuracy_score(y_tst, ypredgnbs)

```

```

print("Accuracy: %.2f%%" % (ags * 100.0))
print('After Sampling [NEAR MISS]: ')
cmgn=confusion_matrix(y_tst,ypredgnbnm)
print(cmgn)
print(classification_report(y_tst, ypredgnbnm))
agn = accuracy_score(y_tst, ypredgnbnm)
print("Accuracy: %.2f%%" % (agn * 100.0))
print('#####')
print('II. RANDOM FOREST ')
print('Before Sampling : ')
print(classification_report(y_tst, ypredrf))
print("Accuracy: %.2f%%" % (ar * 100))
print('After Sampling [SMOTE]: ')
cmrs=confusion_matrix(y_tst,ypredrfs)
print(cmrs)
print(classification_report(y_tst, ypredrfs))
ars = accuracy_score(y_tst, ypredrfs)
print("Accuracy: %.2f%%" % (ars* 100))
print('After Sampling [NEAR MISS]: ')
cmrn=confusion_matrix(y_tst,ypredrfnm )
print(cmrn )
print(classification_report(y_tst, ypredrfnm ))
arn = accuracy_score(y_tst, ypredrfnm )
print("Accuracy: %.2f%%" % (arn * 100))
print('#####')
print('III. DECISION TREE ')
print('Before Sampling : ')
print("Accuracy: %.2f%%" % (ad * 100.0))
print(classification_report(y_tst, ypreddtc))
print('After Sampling [SMOTE]: ')
cmds=confusion_matrix(y_tst,ypreddtcs)
print(cmds)
ads = accuracy_score(y_tst, ypreddtcs)
print("Accuracy: %.2f%%" % (ads * 100.0))
print(classification_report(y_tst, ypreddtcs))
print('After Sampling [NEAR MISS]: ')
cmdn=confusion_matrix(y_tst,ypreddtcnm)
print(cmdn)
adn = accuracy_score(y_tst, ypreddtcnm)
print("Accuracy: %.2f%%" % (adn * 100.0))
print(classification_report(y_tst, ypreddtcnm))
print('#####')
print('IV. LOGISTIC REGRESSION ')
print('Before Sampling : ')
print("Accuracy: %.2f%%" % (al * 100.0))
print(classification_report(y_tst, ypredlr))
print('After Sampling [SMOTE]: ')
cmls=confusion_matrix(y_tst,ypredlrs)
print(cmls)
als = accuracy_score(y_tst, ypredlrs)
print("Accuracy: %.2f%%" % (als * 100.0))
print(classification_report(y_tst, ypredlrs))
print('After Sampling [NEAR MISS]: ')
cmln=confusion_matrix(y_tst,ypredlrnm)
print(cmln)
aln= accuracy_score(y_tst, ypredlrnm)
print("Accuracy: %.2f%%" % (aln* 100.0))
print(classification_report(y_tst, ypredlrnm))
print('#####')
TP1 = cmg[0][0]

```

```

FP1 = cmg[0][1]
FN1 = cmg[1][0]
TN1 = cmg[1][1]
TP1s = cmgs[0][0]
FP1s = cmgs[0][1]
FN1s = cmgs[1][0]
TN1s = cmgs[1][1]
TP1nm = cmgn[0][0]
FP1nm = cmgn[0][1]
FN1nm = cmgn[1][0]
TN1nm = cmgn[1][1]
tpr1 = TP1 / (TP1+FN1)
fpr1 = 1- (TN1 / (TN1+FP1))
tpr1s = TP1s / (TP1s+FN1s)
fpr1s = 1- (TN1s / (TN1s+FP1s))
tpr1n = TP1nm / (TP1nm+FN1nm)
fpr1n = 1- (TN1nm / (TN1nm+FP1nm))
plt.title('Receiver Operating Characteristic')
plt.plot([0,fpr1,3],[0,tpr1,3], 'b', label='Gaussian Naive bayes')
plt.plot([0,fpr1s,3],[0,tpr1s,3], 'g', label='Gaussian Naive bayes with SMOTE')
plt.plot([0,fpr1n,3],[0,tpr1n,3], 'black', label='Gaussian Naive bayes with Neam Miss')
plt.legend(loc='lower right')
plt.plot([0,4],[0,4],r--)
plt.xlim([-0.5,4.0])
plt.ylim([-0.5,4.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# In[39]:
TP2 = cmr[0][0]
FP2 = cmr[0][1]
FN2 = cmr[1][0]
TN2 = cmr[1][1]
TP2s = cmrs[0][0]
FP2s = cmrs[0][1]
FN2s = cmrs[1][0]
TN2s = cmrs[1][1]
TP2nm = cmrn[0][0]
FP2nm = cmrn[0][1]
FN2nm = cmrn[1][0]
TN2nm = cmrn[1][1]
tpr2 = TP2 / (TP2+FN2)
fpr2 = 1- (TN2 / (TN2+FP2))
tpr2s = TP2s / (TP2s+FN2s)
fpr2s = 1- (TN2s / (TN2s+FP2s))
tpr2n = TP2nm / (TP2nm+FN2nm)
fpr2n = 1- (TN2nm / (TN2nm+FP2nm))
plt.title('Receiver Operating Characteristic')
plt.plot([0,fpr2,3],[0,tpr2,3], 'b', label='Random Forest')
plt.plot([0,fpr2s,3],[0,tpr2s,3], 'g', label='Random Forest with SMOTE')
plt.plot([0,fpr2n,3],[0,tpr2n,3], 'black', label='Random Forest with Neam Miss')
plt.legend(loc='lower right')
plt.plot([0,4],[0,4],r--)
plt.xlim([-0.5,4.0])
plt.ylim([-0.5,4.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# In[40]:
TP6 = cmd[0][0]

```

```

FP6 = cmd[0][1]
FN6 = cmd[1][0]
TN6 = cmd[1][1]
TP6s = cmds[0][0]
FP6s = cmds[0][1]
FN6s = cmds[1][0]
TN6s = cmds[1][1]
TP6nm = cmdn[0][0]
FP6nm = cmdn[0][1]
FN6nm = cmdn[1][0]
TN6nm = cmdn[1][1]
tpr6 = TP6 / (TP6+FN6)
fpr6 = 1 - (TN6 / (TN6+FP6))
tpr6s = TP6s / (TP6s+FN6s)
fpr6s = 1 - (TN6s / (TN6s+FP6s))
tpr6n = TP6nm / (TP6nm+FN6nm)
fpr6n = 1 - (TN6nm / (TN6nm+FP6nm))
plt.title('Receiver Operating Characteristic')
plt.plot([0,fpr6,3],[0,tpr6,3], 'b', label='Decision Tree Classifier')
plt.plot([0,fpr6s,3],[0,tpr6s,3], 'g', label='Decision Tree Classifier with SMOTE')
plt.plot([0,fpr6n,3],[0,tpr6n,3], 'black', label='Decision Tree Classifier with Neam Miss')
plt.legend(loc='lower right')
plt.plot([0,4],[0,4],r--)
plt.xlim([-0.5,4.0])
plt.ylim([-0.5,4.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# In[41]:
TP7 = cml[0][0]
FP7 = cml[0][1]
FN7 = cml[1][0]
TN7 = cml[1][1]
TP7s = cmls[0][0]
FP7s = cmls[0][1]
FN7s = cmls[1][0]
TN7s = cmls[1][1]
TP7nm = cmln[0][0]
FP7nm = cmln[0][1]
FN7nm = cmln[1][0]
TN7nm = cmln[1][1]
tpr7 = TP7 / (TP7+FN7)
fpr7 = 1 - (TN7 / (TN7+FP7))
tpr7s = TP7s / (TP7s+FN7s)
fpr7s = 1 - (TN7s / (TN7s+FP7s))
tpr7n = TP7nm / (TP7nm+FN7nm)
fpr7n = 1 - (TN7nm / (TN7nm+FP7nm))
plt.title('Receiver Operating Characteristic')
plt.plot([0,fpr7,3],[0,tpr7,3], 'b', label='Logistic Regression')
plt.plot([0,fpr7s,3],[0,tpr7s,3], 'g', label='Logistic Regression with SMOTE')
plt.plot([0,fpr7n,3],[0,tpr7n,3], 'black', label='Logistic Regression with Neam Miss')
plt.legend(loc='lower right')
plt.plot([0,4],[0,4],r--)
plt.xlim([-0.5,4.0])
plt.ylim([-0.5,4.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
# In[42]:
plt.title('Receiver Operating Characteristic')

```



```

plt.plot([0,fpr1s,3],[0, tpr1s,3], 'r', label='Gaussian Naive bayes with SMOTE')
plt.plot([0,fpr2s,3],[0, tpr2s,3], 'g', label='Random Forest with SMOTE')
plt.plot([0,fpr6s,3],[0, tpr6s,3], 'b', label='Decision Tree Classifier with SMOTE')
plt.plot([0,fpr7s,3],[0, tpr7s,3], 'black', label='Logistic Regression with SMOTE')
plt.legend(loc='lower right')
plt.plot([0,4],[0,4],r--')
plt.xlim([-0.5,4.0])
plt.ylim([-0.5,4.0])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import operator
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import seaborn as sns
ccdata = pd.read_csv('ccFraud.csv')
print(ccdata.info())
ccdata.head(10)
labels = ccdata.columns[1:8]
l1 = ccdata.columns[8]
X = ccdata[labels]
y = ccdata[l1]
X_trn, X_tst, y_trn, y_tst = train_test_split(X, y, test_size = 0.4, random_state = 5)
mg = MultinomialNB()
mg.fit(X_trn, y_trn.ravel())
ypredmg = mg.predict(X_tst)
print(confusion_matrix(y_tst,ypredmg))
print(classification_report(y_tst,ypredmg))
amg = accuracy_score(y_tst, ypredmg)
print("Accuracy: %.2f%%" % (amg* 100.0))
with open('output.txt') as f:
    content = f.readlines()
# you may also want to remove whitespace characters like '\n' at the end of each line
content = [x.strip() for x in content]
#content has first name and last name also at index 0 and 1
content2 = []
for i in content[2:]:
    content2.append(int(i))
print(content2)
# In[ ]:
xpred = [content2]
#xpred=[[2,11,1,17656,16,3,25]]
#pred=mg.predict(xpred)
pred=1
if (pred==1):
    print("SUSPECTED FRAUD\nWe request the user to remove any kind of obstacles like sungalsses or
spectacles for face verification.")
    import face_recognition
    import cv2
    import numpy as np
    from PIL import Image
    import os
    import smtplib, ssl
    import time

```

```

print("Image will be captured automatically at end of 10 seconds. Kindly keep the camera right in front of your
face.")
time.sleep(0.04)
cmra = cv2.VideoCapture(0)
dtr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
sampNum=0
while(True):
    r, pic = cmra.read()
    gry = cv2.cvtColor(pic, cv2.COLOR_BGR2GRAY)
    fcs = dtr.detectMultiScale(gry, 1.3, 5)
    for (x,y,w,h) in fcs:
        cv2.rectangle(pic,(x,y),(x+w,y+h),(255,0,0),2)
        sampNum=sampNum+1
        cv2.imwrite("dataset/user" + ".jpg", gry[y:y+h,x:x+w])
        cv2.imshow('frame',pic)
    if cv2.waitKey(5) & 0xFF == ord('q'):
        break
    elif sampNum>0:
        break
cmra.release()
cv2.destroyAllWindows()
pic1=face_recognition.load_image_file('./dataset/ref.jpg')
imgfenc=face_recognition.face_encodings(pic1)[0]
imgch=face_recognition.load_image_file('./dataset/user.jpg')
imgfencch=face_recognition.face_encodings(imgch)[0]
face_recognition.api.face_locations(pic1, number_of_times_to_upsample=1, model='cnn')
rslt=face_recognition.compare_faces([imgfenc],imgfencch,tolerance=0.6 )
if rslt[0]:
    print ('PASSED FACE RECOGNITION\nNOW PROCEED NORMALLY')
else :
    print ('FAILED FACE VERIFICATION\nTHE BANK WOULD BE GETTING IN TOUCH WITH YOU
IN FEW SECONDS\nINCONVENIENCE REGRETTE\nTRANSACTION TEMPORARILY PAUSED')
    port = 465 # For SSL
    password = "Vandita541998@"
    # Create a secure SSL context
    context = ssl.create_default_context()
    semail = "vanditachadda@gmail.com"
    remail = "vanditachadda598@gmail.com"
    msg = """      Subject: ALERT!
User failed face recognition."""
    with smtplib.SMTP_SSL("smtp.gmail.com", port, context=context) as server:
        server.login(semail, password)
        server.sendmail(semail, remail, msg)
    # TODO: Send email here
else:
    print("LEGAL TRANSACTION... NO SUSPICION\nPROCEED NORMALLY")
from tkinter import *
def sel():
    arr = []
    f_name=e1.get()
    l_name=e2.get()
    gender=str(var.get())
    state = e3.get()
    no_of_card=e4.get()
    balance=e5.get()
    no_of_transac=e6.get()
    no_of_intl_transac=e7.get()
    credit_line=e8.get()
    arr.append(f_name)
    arr.append(l_name)

```

```

        arr.append(gender)
        arr.append(state)
        arr.append(no_of_card)
        arr.append(balance)
        arr.append(no_of_transac)
        arr.append(no_of_intl_transac)
        arr.append(credit_line)
        with open('output.txt', 'w') as f:
            for item in arr:
                f.write("%s\n" % item)
        f.close()

    print(arr)
f_name = None
l_name = None
gender = None
state = None
no_of_card = None
balance = None
no_of_transac = None
no_of_intl_transac = None
credit_line = None
root = Tk()
lbl1 = Label(root, text="GUI", font=("Arial Bold", 20) )
lbl1.grid(column=2,row=0)
var = IntVar()
label = Label(root)
label.grid(row=20)
Label(root, text="First Name").grid(row=4,column=1)
Label(root, text="Last Name").grid(row=5,column=1)
R1 = Radiobutton(root, text="Male", variable=var, value=1).grid(row=6,column=1)
R1 = Radiobutton(root, text="Female", variable=var, value=2).grid(row=6,column=2)
R1 = Radiobutton(root, text="Others", variable=var, value=3).grid(row=6,column=3)
Label(root,text="State").grid(row=7,column=1)
Label(root,text="No of Cards").grid(row=8,column=1)
Label(root,text="Balance").grid(row=9,column=1)
Label(root,text="No of Transactions").grid(row=10,column=1)
Label(root,text="Number of International Transactions").grid(row=11,column=1)
Label(root,text="Credit Line").grid(row=12,column=1)
e1=Entry(root)
e2=Entry(root)
e3=Entry(root)
e4=Entry(root)
e5=Entry(root)
e6=Entry(root)
e7=Entry(root)
e8=Entry(root)
e1.grid(row=4, column=2)
e2.grid(row=5, column=2)
e3.grid(row=7,column=2)
e4.grid(row=8,column=2)
e5.grid(row=9,column=2)
e6.grid(row=10,column=2)
e7.grid(row=11,column=2)
e8.grid(row=12,column=2)
Button(root,text="Submit",command=sel).grid(row=13,column=2,sticky=W,pady=4)
Button(root, text='Quit', command=root.quit).grid(row=13, column=3, sticky=W, pady=4)
mainloop()

```


Credit Card Fraud Detection

ORIGINALITY REPORT

| | | | |
|------------------|------------------|--------------|----------------|
| 26% | 23% | 14% | 17% |
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|----|--|-----|
| 1 | sersc.org Internet Source | 5% |
| 2 | Submitted to KIET Group of Institutions, Ghaziabad Student Paper | 3% |
| 3 | www.geeksforgeeks.org Internet Source | 2% |
| 4 | thuvienso.hoasen.edu.vn Internet Source | 1% |
| 5 | nefy.org Internet Source | 1% |
| 6 | www.coursehero.com Internet Source | 1% |
| 7 | Submitted to Universiti Brunei Darussalam Student Paper | 1% |
| 8 | Submitted to HTM (Haridus- ja Teadusministeerium) Student Paper | 1% |
| 9 | Submitted to University of Southern California Student Paper | 1% |
| 10 | www.rccit.org Internet Source | <1% |
| 11 | Submitted to University of Bradford Student Paper | <1% |

| | | |
|----|---|------|
| 12 | www.researchgate.net Internet Source | <1 % |
| 13 | Submitted to University of Warwick Student Paper | <1 % |
| 14 | Submitted to American University of the Middle East Student Paper | <1 % |
| 15 | blog.csdn.net Internet Source | <1 % |
| 16 | Hatiboğlu, Anıl. "Investigation of the Permeability of the Cell Membrane for Different Cryoprotectant Agents in a Continuous Thermo-Fluidic Micro-Channel System", Middle East Technical University (Turkey), 2024 Publication | <1 % |
| 17 | Submitted to SUNY, Binghamton Student Paper | <1 % |
| 18 | Submitted to Oxford Brookes University Student Paper | <1 % |
| 19 | www.programcreek.com Internet Source | <1 % |
| 20 | Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper | <1 % |
| 21 | commons.und.edu Internet Source | <1 % |
| 22 | "Proceedings of International Joint Conference on Computational Intelligence", Springer Science and Business Media LLC, 2020 Publication | <1 % |

| | | |
|----|---|------|
| 23 | Submitted to Nottingham Trent University Student Paper | <1 % |
| 24 | docksci.com Internet Source | <1 % |
| 25 | manqingzhou.github.io Internet Source | <1 % |
| 26 | H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication | <1 % |
| 27 | github.com Internet Source | <1 % |
| 28 | www.jetir.org Internet Source | <1 % |
| 29 | Submitted to Eastern University Student Paper | <1 % |
| 30 | Xavier Vasques. "Machine Learning Theory and Applications", Wiley, 2024 Publication | <1 % |
| 31 | Submitted to nith Student Paper | <1 % |
| 32 | python-forum.io Internet Source | <1 % |
| 33 | Pawan Singh Mehra, Dharendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security - Volume 2", CRC Press, 2023 Publication | <1 % |
| 34 | pastebin.com Internet Source | <1 % |

35 Duarte, Tomás Fróis. "Exploring the Influence Factors on Card Fraud", Universidade Catolica Portuguesa (Portugal), 2024
Publication

36 Submitted to Queen Mary and Westfield College
Student Paper

37 Submitted to Georgia Institute of Technology Main Campus
Student Paper

38 Submitted to Multimedia University
Student Paper

39 docshare.tips
Internet Source

40 Al-Zadid Sultan Bin Habib, Tanpia Tasnim, Md. Muktadir Billah. "A Study on Coronary Disease Prediction Using Boosting-based Ensemble Machine Learning Approaches", 2019 2nd International Conference on Innovation in Engineering and Technology (ICIET), 2019
Publication

41 Submitted to International Islamic University Malaysia
Student Paper

42 S. Vijayalakshmi, Magesh Kumar, M. Arun. "A study of various classification techniques used for very high-resolution remote sensing [VHRRS] images", Materials Today: Proceedings, 2020
Publication

43 Submitted to University of North Texas
Student Paper

| | | |
|----|--|------|
| 44 | www.cxymm.net Internet Source | <1 % |
| 45 | Submitted to University of Oklahoma Student Paper | <1 % |
| 46 | repository.ju.edu.et Internet Source | <1 % |
| 47 | Durgesh Kumar Mishra, Nilanjan Dey, Bharat Singh Deora, Amit Joshi. "ICT for Competitive Strategies", CRC Press, 2020 Publication | <1 % |
| 48 | Submitted to University of Nottingham Student Paper | <1 % |
| 49 | ijircce.com Internet Source | <1 % |
| 50 | stackoverflow.com Internet Source | <1 % |
| 51 | www.biorxiv.org Internet Source | <1 % |
| 52 | hdl.handle.net Internet Source | <1 % |
| 53 | www.ir.juit.ac.in:8080 Internet Source | <1 % |
| 54 | Submitted to CSU, San Jose State University Student Paper | <1 % |
| 55 | Submitted to International School of Geneva Student Paper | <1 % |
| 56 | Submitted to Johnson County Community College Student Paper | <1 % |

| | | |
|----|--|------|
| 57 | Submitted to Liverpool John Moores University Student Paper | <1 % |
| 58 | programmingtrick.com Internet Source | <1 % |
| 59 | Submitted to University of Auckland Student Paper | <1 % |
| 60 | elbruno.com Internet Source | <1 % |
| 61 | Huijian Dong. "Data Analytics in Finance", CRC Press, 2025 Publication | <1 % |
| 62 | Submitted to Coventry University Student Paper | <1 % |
| 63 | Submitted to Istanbul Bilgi University Student Paper | <1 % |
| 64 | pure.hw.ac.uk Internet Source | <1 % |
| 65 | Submitted to University of Missouri, Kansas City Student Paper | <1 % |
| 66 | Submitted to University of Queensland Student Paper | <1 % |
| 67 | machinelearning.recipes Internet Source | <1 % |
| 68 | www.eurchembull.com Internet Source | <1 % |
| 69 | Kun Zhu, Nana Zhang, Weiping Ding, Changjun Jiang. "An Adaptive Heterogeneous Credit Card Fraud Detection Model Based on | <1 % |

Deep Reinforcement Training Subset Selection", IEEE Transactions on Artificial Intelligence, 2024

Publication

| | | |
|----|---|------|
| 70 | research.library.mun.ca Internet Source | <1 % |
| 71 | www.diva-portal.se Internet Source | <1 % |
| 72 | Roshani Raut, Salah-ddine Krit, Prasenjit Chatterjee. "Machine Vision for Industry 4.0 - Applications and Case Studies", CRC Press, 2022 Publication | <1 % |
| 73 | Sanjeev J. Wagh, Manisha S. Bhende, Anuradha D. Thakare. "Fundamentals of Data Science", CRC Press, 2021 Publication | <1 % |
| 74 | bobrupakroy.medium.com Internet Source | <1 % |
| 75 | c.coek.info Internet Source | <1 % |
| 76 | www.analyticsvidhya.com Internet Source | <1 % |
| 77 | www.hindawi.com Internet Source | <1 % |
| 78 | www.slideshare.net Internet Source | <1 % |
| 79 | Submitted to Beykent Universitesi Student Paper | <1 % |
| 80 | Fernandez, A.. "A study of the behaviour of linguistic fuzzy rule based classification | <1 % |

systems in the framework of imbalanced data-sets", Fuzzy Sets and Systems, 20080916

Publication

81 Mohammed Abdulrahman, Abdallah Abdulfattah. "Deepfake Image Detection Using Explainable AI and Deep Learning", Rochester Institute of Technology

Publication

82 Natasa Kleanthous, Abir Hussain. "Machine Learning in Farm Animal Behavior using Python", CRC Press, 2025

Publication

83 Submitted to University of Essex

Student Paper

84 fliphtml5.com

Internet Source

85 stax.strath.ac.uk

Internet Source

86 upcommons.upc.edu

Internet Source

87 eprints.nottingham.ac.uk

Internet Source

88 Mohamed Abdel-Basset, Hossam Hawash, Laila Abdel-Fatah. "Artificial Intelligence and Internet of Things in Smart Farming", CRC Press, 2024

Publication

89 Nazerke Baisholan, J. Eric Dietz, Sergiy Gnatyuk, Mussa Turdalyuly, Eric T. Matson, Karlygash Baisholanova. "FraudX AI: An Interpretable Machine Learning Framework

for Credit Card Fraud Detection on Imbalanced Datasets", Computers, 2025

Publication

90

researchspace.ukzn.ac.za

Internet Source

<1%

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off