MSc Data Analytics

Credit Card Fraud Detection using Python

Dissertation
by
Mehedi Hasanat Apu

Supervisor

Hamidreza Khaleghzadeh

September 2022

Abstract:

With the entire world under lockdown amid the Corona pandemic, digital transactions in terms of shopping and transactions have increased exponentially and are constantly increasing. Credit card is one of the means of this digital transaction. But the problem is not being able to ensure the security of card transactions. As the use of credit cards increases, so does the amount of credit card fraud. Therefore, to solve these fraud problems, models have been created using different algorithms of machine learning in our proposed project. In this case, we have used private transaction data of European Union card holders. Decision tree, Random forest, XGBoost, and Logistic regression classifiers are used here. GridsearchCV is used in the model as cross-validation. The dataset used is balanced to an imbalanced dataset using undersampling and upsampling. Finally, the best model has been found through the comparative analysis of the models to achieve the desired results. In our case study, multiple models are prepared separately through unbalanced dataset and through balanced dataset prepared after using Undersampling/Upsampling Technique. Great performance has been achieved from each implemented model, but XGBoost model built on balanced dataset by using upsampling technique provides the best performance.

Acknowledgment:

First of all, I would like to thank my supervisor Hamidreza Khaleghzadeh for providing me with all kinds of support and instruction in all challenging situations from the beginning of the dissertation. I am grateful to my family for always supporting and inspiring me. I also express my gratitude to my university, from where I have been able to receive all kinds of learning outcomes, and library facilities. I would like to thank my classmates for their help.

Table of Contents

List of Tables

List of Figures

Chapter-1: Introduction

Along with the advancement of modern technology and communication across the world, the volume of transactions through electronics is also increasing at a great rate. And one of these online transactions is a credit card. Basically, a credit card is a kind of small plastic card by using which customers can make transactions without cash money. This card allows the card user to transact money to purchase goods and services.

According to a published report by Nilson in 2022, there were more than 438 billion transactions worldwide in 2021, which is 7.7% more than the transactions in 2020 and is expected to exceed 800.41 billion by 2026 [21]. As a result, the issue of economic security has become more of a concern. Security is the most important thing when it comes to electronic transactions, and nowadays the security of this credit cards is often threatened. Due to the development of technology, various unscrupulous people are using technology for unethical activities. Therefore, analysts are working all over the world to protect against credit card fraud. Credit card fraud can take many forms. There are two main things to follow to avoid cheating.

1. Resistance.
2. Identification.

We will also work on fraud detection to reduce credit card fraud in our proposed project. In the case of credit card fraud detection, the models created through different algorithms of machine learning are getting success as expected, so we will create models with the help of algorithms of machine learning in the proposed project. In the next chapters, we will know in detail all techniques, tools, and processes to detect fraud.

1.1 Research motivation:

The general issue of the proposed project is to detect any fraud activity in any credit card. Nowadays people are using the digital platform in every field with the touch of modernity. Therefore, people are relying on digital platforms to transact money and save money. And all banks and financial institutions have also made all their transaction activities online based. In other words, both ordinary customers and financial institutions are saving their money on digital platforms. So it goes without saying that these platforms need to have the highest shackles of security and detect any kind of fraud. If this security is not maintained, everyone will suffer. Unfortunately, credit card fraud is increasing through the use of technology in spite of keeping the security issue at the fore. Which is now a cause of concern for all customers and organizations?

In 2020, Nelson released a report that showed credit card fraud costing $42.274 trillion worldwide in 2019, which is more than 4.2% than 2018. In 2014, the Canadian Banking Association published a report that said losses increased 18 percent in 2014 compared to 2013, despite improved EMV chip security.[22]

Therefore, it is important to detect credit card fraud for the safety of both customers and organizations. Fraudulent causes huge losses to both banks and customers, which requires a dependable solution. Currently, it is possible to get the desired success through data mining and machine learning algorithm. As a result, in this project, an attempt has been made to find out the best solution by creating different models using data mining and machine learning algorithms.

There are various types of credit card fraud today [23], some of the notable ones are-

1. Bankruptcy fraud.
2. A theft fraud/counterfeit fraud.
3. An application fraud.
4. A behavioral fraud.

1.2 Scope of the research:

With this increase in credit card fraud day by day, analysts are very concerned and are working to reduce and prevent fraud. Various types of technology are being researched to solve this problem. Machine learning and data mining techniques are the most effective to solve this problem. Classification algorithms like logistic regression, random forest, decision tree, KNN, SVM, and naive Bayes are the most used to detect fraud. By using these classification algorithms it is possible to detect fraud in real-time transactions by creating high-class models. So analysts are strengthening research to detect fraud through machine learning.

So our main content of the proposed project is to detect credit card fraud activity using machine learning algorithms. Also, data mining is used here to detect fraud. Different models will be created using different algorithms of machine learning and the best model will be selected considering the performance of the models according to the dataset.

We use the random forest, logistic regression, XGBoost, and decision tree classifier to create our fraud detection model.

In this case, the comparative analysis and results of each model will be presented. In the next chapters of our project, we will see all the tools, processes, and techniques used in detail.

1.3 Objectives:

This section will show in detail the main aims of the proposed project throughout the given time frame.

1. Establish a background in fraud detection novelty techniques and machine learning methodologies that need to be used in terms of achieving the solution.
    - Identify a software framework and tools needed to develop the engine for the project (e.g. programming language, tools, framework, and techniques).

- Gain an understanding of the software to be used throughout the solution development.

2. Implementing the Credit Card Fraud Detection engine.
    - Design a Reader for the input, a model for transaction data, and card data.
    - Design a data store for transactions and cards.
    - Design the machine learning algorithm that classifies the data into potential fraud and otherwise by training the machine with the given data set.
    - Implement the algorithm using chosen framework and tools.
    - Validate the Implementation.
3. Documentation and evaluation of the established solution.
    - Test the program with different datasets to verify the solution works and trained properly.
    - Introduce any possible code enhancement and improvements to the solution.
    - Performance Analysis
4. Produce the final report, which includes the previous background research on the project.
    - Identify evaluation analysis and results from the implementation testing and improvements throughout the different stages.

1.4 Chapter Overview:

Literature review:
In this chapter, we have analyzed and explained several papers related to credit card fraud detection, through which we have gained an understanding of machine learning and data mining tools/techniques and proposed our necessary models.

Methodology:
The Methodology chapter explains our project planning through the most popular method C.R.S.P-D.M method.

Demand and Inspection:
All demands of clients, goal, and purpose of the project, tools/software used in the project, constraints of the project, and different types of algorithms are discussed in this chapter.

Project Outline:
All processes used for the project, algorithm selection, and design of the entire project will be implemented.

Data visualization and Exploration:
Analytical information of the dataset used for our project is presented in this chapter through data visualization and exploration.

Implementation:
The implementation of all the proposed models has been implemented in this chapter.

Evaluation:
After implementing all the proposed models, all the output results are analyzed here. And after collecting the results obtained from all the models, the best model is selected from them.

Conclusion:
The complete project summary has been made and the future plan and maintenance plan of the project will be discussed.

1.5 Summary:

This chapter discusses various topics related to credit cards and credit card fraud. What is a credit card, how its use is increasing day by day, what is card fraud, how frauds are organized and various types of fraud are briefly explained. Moreover, the technology used for fraud detection has been highlighted and a brief plan of our proposed project has been presented.

Chapter 2: Literature Review

A literature review is a research where science or any specific topic is analyzed or observed previously several writers or researchers were working on that specific topic and published various academic papers. Through this literature review, a specific topic is very well studied. Knowledge can be gained and later it becomes much easier to act on it. Moreover, logical and various critical topics are also highlighted in the literature review. Therefore, it is considered logical to call it a literature review without thinking of it as a literature report.

2.1 Discussion:

We will work on the content of credit card fraud detection in our proposed project. So, in this chapter, we have tried to work with several papers related to our project, which were previously implemented by different writers and researchers. In this chapter, we have to observe and analyze all the papers properly so that we can understand the frauds well and work on them. Here we will get a very good understanding of many types of machine learning techniques by reviewing the papers through which we can gather important ideas to detect credit card fraud.

Finally, after reviewing all the papers, we will select the best techniques to build our models.

2.2 Project Base:

Credit card fraud detection is a process through which to find out all the unwanted activities or transactions by identifying various types of fraudulent transactions or fake transactions. This type of fraud can be detected by using different techniques such as machine learning or the use of different types of models etc.
To reach our appropriate results, we will therefore first review the credit card fraud-related research and papers and determine our requirements, aims, and analysis process.

2.3 Individual paper Review:

2.3.1 Basic Machine Learning Algorithms:

Linda Delamare [1], Fraud is one of the biggest problems in credit. So to eliminate this problem one should first master the techniques to detect credit card fraud, then carefully observe and analyze the techniques that are used for credit card fraud. In this project, it is proposed to use JEL classification Algorithms like C49, G21, G24, and K42. Here, some machine learning Algorithms for Fraud detection are also mentioned, such as DT, Genetic Algorithm, Clustering Techniques, and Neural networks. Moreover, different types of credit card fraud are highlighted here, such as - Bankruptcy fraud, theft fraud, application fraud, and behavioral fraud.

Gajendra Singh [2], In this paper SVM is proposed to detect fraud where different types of kernels are used. Due to privacy reasons, this type of data is not available, here in this paper, some data has been created, where some types of mean and variance have been created using true and false transactions. They are then combined using different probabilities. MATLAB is chosen to implement the algorithm in this paper. Here 3 types of a kernel are used- Linear, Quadratic, and RBF where the accuracy of the RBF kernel was the best compared to the Linear and Quadratic kernel which was 97%.

Ibtissam Benchaji [3], Day by day credit score cards have become more and more popular in the online world. Due to this fraudulent activities are increasing day by day. To cope with this situation, it is mandatory for financial institutions to further improve their Fraud detection framework to reduce their losses. And for that purpose, the main content of this project is to create a model for credit card fraud detection that is based on sequential modeling of records and uses LSTM deep recurrent neural network and attention mechanism. This model better considers the sequential nature of transitional records than previous studies and allows understanding of the most significant transactions in the enter sequence, thereby better predicting fraud transactions.

The price structure of this model is based on three sub-strategies.

1. The uniform manifold approximation to understand the most useful future properties and projection (UMAP).
2. Big quick time memory (LSTM) network for linking transaction sequences together.
3. The eye mechanism to enhance an LSTM performance.

Two datasets are used in this project. The 1st dataset contains 284,807 transactions, including 492 fraud transactions, and the 2nd dataset contains 594,643 transactions, including 7200 fraud transactions.
By comparing the proposed model with some other machine learning models such as GRU, SVM, KNN, and ANN, it can be seen that the accuracy, precision, and recall results of LSTM are the best, whereas for dataset 1 and dataset 2, the accuracy is 0.9672 and 0.9748, and the precision is 0.9885. And 0.9769, Recall 0.9191 and 0.9722.

Asoke K. Nandi et al [4], Today, with the advancement of machine learning, analysts are trying to master and apply effective and critical techniques to detect fraud in economic fields. Businesses are facing billions of dollars in losses due to this fraud every year. Therefore, a multi-classifier framework has been designed for this project by considering all these issues. Also, several ML Classification Algorithms like DT, RF, and XGBoost is used. Here python is used as a programming language that is used in the google lab environment. Here an ensemble model is designed, and Behaviour-Knowledge-space(BKS) is used for future content aggregation.
To get the maximum performance of this model publicly available dataset is used and the record of real financial activities is shown. The dataset used here shows 60,595 transactions

of 9685 customers, which took place in 23 countries. And among these transactions, there were 28 fraudulent transactions.
Using benchmark data, it was found that the performance of Bks was the best compared to GEP and CUSBoost. Bks had the highest F1 score in 4 out of 6 datasets. On the other hand, according to the different rations of minority and majority samples, the accuracy of Bks was the highest at 0.993.

Kuldeep Randhawa [5], A model using MI is proposed in this project to detect and solve the credit card fraud problem. Here the hybrid method is used from the cross-breed models found using Adaboost. However, the standard model has been used before that. A real-world credit dataset was used to get the maximum performance of the proposed model of this project, which belonged to an institution in Malaysia. There were 2,87,224 transactions out of which 102 were fraud transactions. Moreover, a publicly used dataset has also been used where there are 2,87,807 transactions out of which 492 are fraud transactions. Mathews correlation coefficient (MCC) is used here to explain True Positive, False Positive, True Negative, and False Negative. Here the accuracy of different types of ML Algorithms such as (NB, DT, RE, GBT, DS, RT, DL, NN, MLP, LIR, LOR, and SVM) can be seen by applying Adaboost. Where seen the accuracy of RF is at best 95%. In the case of the real-world dataset, it can be seen that the use of Adaboost results in a lot of improvement. The LIR Algorithm using Adaboost results in 7.4% to 94.1% accuracy.

Fayaz Itoo [6], Currently financial fraud has become a very serious problem and is affecting the economy in a very practical way. So analysts are researching to eliminate this problem. This project uses three types of ML Algorithms to overcome such problems. They are Logistic Regression, Naive Bayes, and k-nearest Neighbor, and their complete performance is comparatively analyzed here. This project is done in python. This performance is measured as their best accuracy. The dataset of this project is used from Kaggle, there are 284807 transactions of which 0.172% are fraud transactions. Among the algorithms of this project, accuracy, sensitivity, specificity, precision, and F-measure has been used to choose the best model. Here, logistic regression shows a higher accuracy of 95% which is higher than Naïve Buyes (91%) and KNN(75%)

RONG-CHANG CHEN [7], In terms of fraud detection, this project has some problems, but this project has revealed promising results. The model proposed in this project cannot accurately detect Fraud transactions from some skewed datasets. Here a Binary support vector system (BSVS) model is used to detect fraud. BSVS mainly relies on a support vector machine (SVM) here and several genetic algorithms (GA) are implemented. Here, self-organizing mapping (SOM) is applied to the input data of the distribution model to obtain a high true negative rate. Then the high detection input data is used to properly train the BSVS. The outcome of this model shows that BSVS is very efficient for estimating a high true negative rate. A model has been proposed in this project but no publicly used dataset has been used.

Wen-Fang YU et al [8], with the increase in the use of credit cards in China and its use, is increasing day by day, fraud is also increasing at a large rate. Therefore, to eliminate this problem, an outlier detection model is proposed in this paper, where the main basis of this model is distance sum, which is found in credit card fraud transaction data from infrequency and unconventionality. The main feature of this model is to identify the contents of Fraud transactions using outlier mining. The experimental outcome shows that it is possible to accurately detect fraud transactions through this model. Through this model, a series of anti-fraud techniques can be used, so that banks can reduce many losses.

V.Mareeswari et al [9], One of the major problems in the banking sector is fraudulent transactions. With the development of e-commerce and online sectors, the use of credit cards is increasing, but the bigger problem is that fraud is increasing more and more using new technologies. So prevention is very important along with fraud detection. There were some limitations in the previous proposals so a new Algorithm has been proposed for this project. The previous algorithm had some unbalanced classes, scalability issues, and many limitations. A Hybrid support vector machine (HSVM) is proposed here to overcome these limitations. HSVM is the most used method for pattern recognition and classification. Scalability has been increased in this project by updating the data evaluation.

Maja Puh et al [10], With the development of e-commerce, credit score card fraud has become a serious problem worldwide. Currently, there is an increasing trend of practicing machine learning through data mining to detect such fraud. In this case study, three machine learning algorithms have been worked on. These are Random forest, Support Vector machine, and logistic regression. Fraud has been extracted from here using real-life data. To reduce the imbalanced dataset the dataset is sampled and in this case, SMOTE sampling method is used. The dataset used in this project contains 2,84,807 transactions, there are 492 fraud transactions or 0.1728%. The performance of the Algorithms proposed here has been considered using precision and recall. Here in this project, Logistic regression shows good performance. Based on AUC and AP result, the score for incremental learning was 0.9107 and 0.8413. Although the dataset here was small. If there is a big dataset, the result may change.

K. VENGATESAN[11], At present, in the banking sector, banks provide various facilities to their customers such as ATMs, Online Banking, loans, and debit and credit card facilities to get more publicity. But there are some difficulties in using these cards. And that is card fraud. So to eliminate this problem some Algorithm of Machine learning is used here. Usually, customers use their credit cards 24 hours and 7 days. In this case, the use of machine learning can always monitor e-transactions. In this project, the author has proposed KNN and LR where it is possible to find out the best model by comparing their accuracy. Here, KNN shows the best performance where accuracy was 0.99, precision 0.95, recall 0.72, and f-1 score was 0.82.

Cuizhu Meng [12], In this paper, one of the machine learning algorithms XGboost model has been developed to detect credit card fraud. The unbalanced dataset here has only 0.172%

fraud transactions. The performance of this model is measured according to recall and ROC values. Undersampling, Oversampling, and XGboost Algorithms are implemented here by reviewing the unbalanced dataset. Moreover, SMOTE method is used here to balance the dataset.

Fayyomi [13], The main point of this project is to create some structure through Algorithm through which all the fraudulent transactions in credit cards are stopped. In this paper, various types of machine learning algorithms are discussed such as logistic regression, the decision tree, the random forest, the artificial neural network, the k-nearest neighbor and k-means clustering. After implementing all the models, the best model is selected here by comparing their performance, cost, time, etc.

Abdou [14], In this research paper the researcher discussed JEL classification. Here the contents of different classifications of JEL such as C49, G21, G24, and K42 are highlighted. In this paper, the researcher has described all the factors that can lead to credit card fraud and various techniques to detect fraud. Moreover, in this paper, he has discussed machine learning algorithms like a Decision tree and Genetic Algorithm. Even automated mechanism has been highlighted in this paper. Moreover, neural network and clustering methods have also been analyzed here. Although no real-life dataset or any kind of dataset was used in this paper. Because this is a review paper but using this paper will be very effective to carry out further research work.

2.3.2 An Association Rules:

Vinaya [15], To detect credit card fraud in many cases to find out how the behavior of fraud can be. Association rules are used. The dataset used in this project is based on the transactions of Chilean retail companies. The data contained in the dataset was de-fuzzified with the help of the fuzzy query 2+ data mining tool and all processing operations were done using the same tool. The outcome of adopting this approach was to reduce the number of rules significantly, thereby making fraud detection easier for analysts.

This was the main overview of the project

1. Data gathering
2. Data stabilization
3. Attribute removal.
4. Outlier spotting.
5. GBT classification.

The final result of this project ie accuracy was 92.94%.

2.3.3 The Neural-Network:

Ghosh [16], The data of a credit card provider is used here and the large sample of transactions of these data is trained to utilize a system based on a neural network ie Model. The proposed network has been able to detect a large amount of fraud (an order of magnitude

more) and show fewer FP over rule-based fraud detection. The mentioned system is set up in an IBM 3090. This network is implemented in 2000000 transactions taking about 2 months.

2.3.4 The Hidden Markov Model:

Srivastava [17], In this case, study, with the help of a Hidden Markov Model, a sequence Model Base is formed of all the transactions that take place on the credit card. Later it is presented how this model can play a role in detecting frauds. The model is initially trained with the normal behavior of a card user such as cardholder profiles, pricing cues for application monitoring, and initial estimation of model parameters When attempting to commit a transaction, if it is not accepted with probability by the trained HMM, then the transaction is identified as a Fraud. At the same time, it is also specified here that genuine transactions should not be rejected. Analysis shows that the spread of differences in the input data invalidates the validity of system-2 by 80 percent. The model proposed in this project is also effective for large-scale transactions.

2.3.5 An Artificial Immune System (AIS):

Halvaiee [18], Nowadays online transactions are also expanding along with online expansion. A large number of transactions are organized through credit cards. But the problem is that card fraud is increasing significantly. Therefore, analysts are working with many types of models and technologies to detect fraud. As a result, Artificial Immune System is also being worked on. Now it is said that in the case of fraud detection, it is important to have the ability to detect accurately along with speed, which has not been achieved yet. In this paper, a model using AIS is constructed called the AIS-based Fraud detection Model (AFDM). Here the accuracy is increased by about 25%, while the cost is reduced by 85% and the response time of the system is reduced by 40%.

2.3.6 The Game theory-based approach (NRT pool):

Gianini [19], A pool of classification rules is used to detect fraud through which the set of member rules is modified according to their performance. In this case, a near-real-time (NRT) classification is used here. Several topics have been highlighted in this project, first of all, what kind of rules can be kept in the NRT pool based on their previous performance. Based on the Shapley Value (SV) each rule is evaluated by its overall performance within the pool. Here the model is validated using real-world credit card data. The dataset contains 3x105 transactions.

The analysis shows that the SV-based approach is more effective than the conventional approach.

2.3.7 The Self-Organizing map visualization:

Olszewski [20], This case study describes the SOM technique for fraud detection. Using this technique, the user accounts have been visualized here. One of the distinguishing features of this paper is the use of SOM. Because customer accounts are represented numerically in the

form of metrics and the SOM technique visualizes a single vector. After setting user accounts in SOM, threshold type binary classification is used to detect any type of fraud transaction. For practical fields such as telecommunication, computer network, and credit card fraud detection this technique shows promising results. It is a very attractive technique as it is done through visualization because it is very easy to analyze the data by spreading the high-dimensional data on the 2-dimensional data for fraud detection.

2.4 Summary of Literature Review:

In the literature review chapter, we reviewed several papers suggested by different writers. After analyzing and observing all the papers, I have been able to collect some important information related to credit card fraud and to work on our next research and project. In our observed papers we have used several Algorithms for machine learning such as - The Decision Tree, The Naive Bayes, The Logistic Regression, The K-nearest Neighbor, The Random Forest, The Support-Vector-Machine, XDR, and XG-Boost. I was able to get an idea about the processes and procedures for credit card fraud detection. From these papers, I was able to gain knowledge about different types of models where there were models as The Hidden Markov Model (HMM), SOM, The game theory-based approach, and the Artificial immune system (AIS) through which everything about the processes of fraud detection was visible in detail. Moreover, the concepts of fraud detection have been achieved by implementing The Neural Network (perceptron), The Association rules, The JEL Classification, and clustering techniques. The genetic Algorithm and SMOTE sampling method to detect fraud can be seen in the mentioned papers. Various tools and software used for machine learning such as Jupyter notebook, Matlab, rapid miner, and google collab have been seen working in these projects. For how to achieve the results after model creation, here I got the idea about Mathematical equations and calculations such as Mean, Variance, Probability, etc. Moreover, I have mastered details about Accuracy, Precision, recall, F1-score, AUC, AP, Sensitivity, and specificity to analyze the results.

Through all this information and knowledge gained through the literature review, I will later present our proposed models in our design chapter through which I can expect an effective result to detect credit card fraud.

Chapter 3: Methodology

To implement or maintain a project, the methodology is a set of methods and plans where what kind of tools and techniques will be used to make the plans, the entire process of project implementation, and all the forecasts of management methods are highlighted. Following the methodology, leading the project team, collaborating with the team members, and managing the project becomes easier. The most followed C.R.I.S.P-D.M method is used for our proposed fraud detection project and our project plan is illustrated in this chapter.

3.1 Method:

A popular industry-standard method for data mining worldwide is the CRISP-DM method. It is a method for data mining, which is used by data analysts to solve a specific problem. While working on a data mining project, this method divides the project into different phases, thereby allowing the project to be worked on in a well-planned manner. These stages are discussed below.

3.2 Why CRISP-DM method:

It is a data mining method or methodology through which it is possible to get the blue print of a data mining project. It provides a well-planned roadmap for working on our project. CRISP-DM provides a well-planned approach to each step of our project, from problem identification, dataset understanding, data pre-processing, designing, model implementation, model evaluation, and deployment.

3.3 Implementation of C.R.I.S.P- D.M:

As C.R.I.S.P- D.M based on the process flow in Figure 3.1 below.
The model suggests that flow or steps to procure:

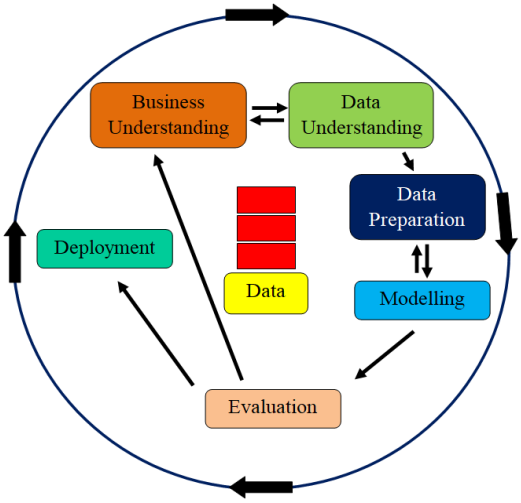


Figure 3.1: CRISP-DM method

1. An Understanding the Business - Understanding the rules and regulations of a particular organization along with its business objectives and goals.
2. An Understanding of data - Data collection and analysis
3. A Data Preparation Phase - Identifying the data needed to use the data in the model
4. A Modelling Phase - Building models according to datasets and targets
5. The Rating Mechanism - Verifying that the models developed are effective for solving a problem
6. The Final Deployment - Validation by users after successfully building the model

3.3.1 Business understanding:

The CRISP-DM method begins with understanding a specific business. In this research, we will try to understand the business objective of the bank. Our proposed aim is to detect fraud in the bank's credit card transactions. We need to note here that we need to mine data to better understand the transactions that can drive fraud. At the same time, it is important to understand the contents of the bank. Credit card fraud hurts both the customer and the bank. So in this case it is important to know about the bank's overall database and business policy to develop a model for credit card fraud. After building the model it is important to ensure that the desired results are achieved. Moreover, project planning and risk assessment are essential at this stage.

The business Understanding phase follows some steps

1. Consumer demand.
2. Project scheme.
3. Goal and Purpose.
4. Tools and software requirements.
5. Constraints.
6. Expected success.

We will discuss the business understanding phase of the CRISP-DM method later in more detail in the demand and requirement chapter

3.3.2 Understanding the Data:

The second thing is to observe the data. For that, primary data must first be collected, these data must be analyzed and data quality must be verified. We have worked on time of the fraud, the number of fraud, and the types of fraud in the dataset we are using in our project to understand credit card fraud.

The data understanding phase monitors the following issues

1. Source of data.
2. The procedure of collection.
3. Set of data.
4. Tools for visualization.

5. Standard of data.
6. The morality of data.

This will be discussed in detail in our data visualization and exploration chapter.

3.3.3 Data Preparation Stage:

In the next step of this step, we will use the data in our model to detect fraud, so in this step, we will prepare the data for use. In our project, the data will be processed for use in various machine learning algorithms. At this stage the fields are calculated, necessary data is cleaned, an external dataset is added and the dataset is classified according to the required information.

The following points are important for the data preparation phase

1. Missing value.
2. Relavancing of data.
3. Inaccuracy in data.
4. Data scaling.
5. Procedure challenges.

The data preparation phase will be dealt with in the data visualization and exploration chapter.

3.3.4 Modeling phase:

At this stage, we select the appropriate modeling techniques to achieve the desired results and goals and use the necessary algorithms to test the dataset. In our case study, we use some machine learning algorithms for dataset training, validation, and testing. Through this Algorithm, we can reach our goal.

Later, our proposed model and algorithms will be formulated in detail in our modeling chapter.

3.3.5 Assessment criteria:

At this stage, we observe whether we have used appropriate tools in our project for data mining and whether our developed model based on the business object is providing the best results. If we need to test some more models for our business content then we don't decide in this step. After creating the model we need for our case study Modeling Phase, we evaluate it in this step because it is a review paper, but using this paper for further research. It will be very effective to perform the task.

In our Implementation and evaluation chapter, we will perform in detail the assessment criteria of the CRISP-DM method.

3.3.6 Final deployment:

In this phase of the CRISP-DM method, the final decision is taken on the results of the entire project. The final result of our credit card fraud detection project will be accepted in this phase, that is, the Conclusion chapter will be the content of the final deployment. The best model will be selected from our used models based on this phase. Finally, the success of the project will be confirmed by taking a review through the client.

3.4 Summary of Methodology:

The Methodology chapter discusses the all steps of the CRISP-DM method and how the steps of the CRISP-DM method will work with our entire project. As a result of using this method, a preliminary plan of our project has been established and through which it will become easier to implement the project later.

Chapter 4: Demand and Inspection

In this chapter, the overall demand and analytical content of the project are presented. The first phase of the CRISP-DM method was Business understanding. This chapter will be observed with all the contents of business understanding. Consumer demand for this project, our goal, and purpose, project scheme, tools and software, constraints, our expected success, and various types of machine learning algorithms are depicted here.

4.1 Consumer demand:

The project requires protecting the clients from major economic losses by identifying any kind of unwanted or fraudulent transactions in the used credit cards. In this case, the main goal will be to find the best model for the project.

4.2 Goal and purpose:

The first goal or aim of the project is to detect all types of fraudulent transactions and the purpose is to prevent fraudulent transactions by providing a successful model to clients.

a. **Goal:**
   In this project, fraud detection has been done using data mining and machine learning tools. In that case, the best model has been selected by creating a fraud detection model to get the desired outcome using different types of machine learning algorithms.
b. **Purpose:**

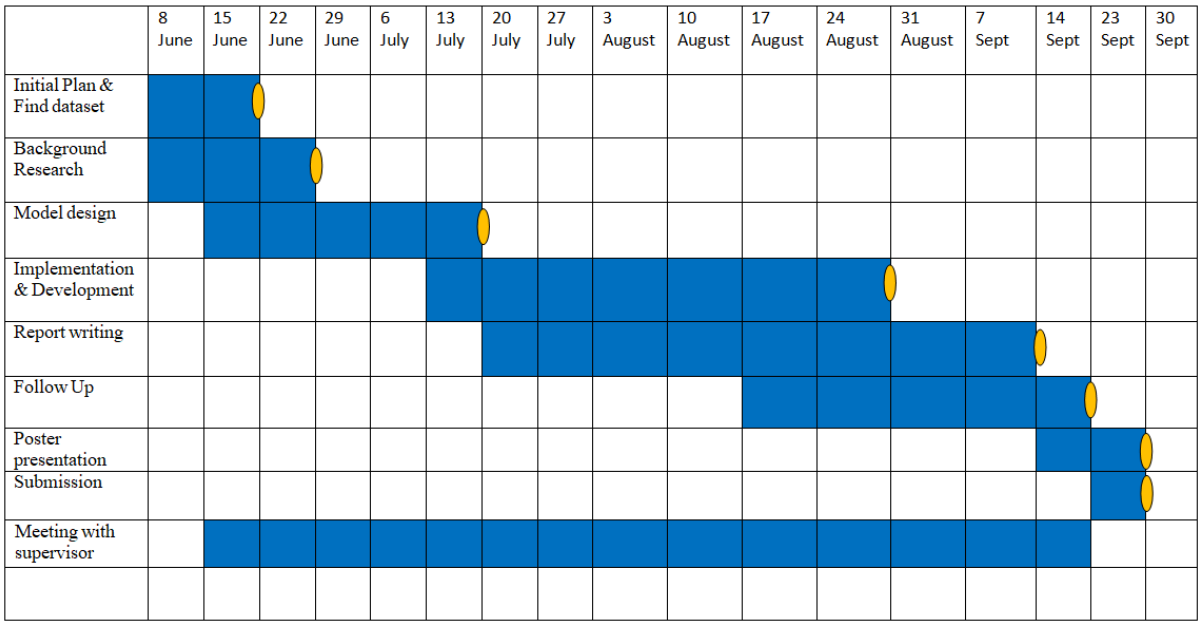
   1. Detecting fraud by using data mining tools, different techniques, and all types of processes to the maximum success.

   2. To achieve the desired results by comparing the performance of each classifier through the use of classifier algorithms required for machine learning.

   3. Prevent fraudulent activity by selecting the best model.

4.3 Project scheme:

Table 4.1: Gantt chart of proposed project scheme for our dissertation.



4.4 Tools and software:

4.4.1 Software:

An operating system: Windows 8.1/10/11.

An IDE tool: We used the Jupyter Notebook platform to work on this project. It is a web-based application through which various tasks including live code, data visualization, and calculation tasks are done. As Jupyter Notebook is an open-source application, there is no extra payment for it.

4.4.2 Programming Language:

Moreover, in this project, we have used Python version 3.9.7 as a programming language.

```
from platform import python_version
print(python_version())
```

```
3.9.7
```

4.4.3 Hardware:

The following tools are required as hardware:

- Processor: Pentium i3 or higher.
- 4.00 GB of RAM or higher
- Hard disk drive: 20 GB (free).
- Peripheral devices: Monitor, Mouse, and Keyboard.

4.4.4 API's :

NumPy: For simple operations on array-type structures.
Pandas: For reading and updating the CSV file or dataset.
SciKit Library: Learn- for pre-processing of data in our project.
Matplotlib/Seaborn: For plotting and illustrating confusion matrix in color format.

4.5 Constraints:

There are no significant constraints in this project, but a certain deadline has been fixed for the completion of this project within which it is desirable to complete the project. Moreover, due to credit card data being highly secured, the availability of such datasets in open source platforms is less.

4.6 Expected success:

The expected success after completing this project is to select the desired best model through which maximum fraud transactions can be brought out. Here the real success is the goal of detecting credit card fraud.

4.7 Algorithms:

In our dissertation, we implemented the model using 4 types of machine learning classifiers algorithm and then compare different models. The classifiers use are the followings.

- Logistic regression
- XGboost
- Decision tree
- Random forest
  -

Why not use SVM and KNN?

Currently, there are 284807 data points in the dataset, and in the event of Oversampling, there would be an even greater amount of data points. With a big number of data points, SVM is not particularly efficient since it requires a significant amount of processing power and resources to do the transformation.

KNN is not a memory-efficient algorithm. With an increase in the number of data points, the model becomes very sluggish since it must store all the data points to function properly. A computationally intensive technique is required because for a single data point, the program must compute the distance between all the data points and discover the data points closest to it.

4.8 Techniques:

Since the dataset used by us is imbalanced, we have used two types of sampling methods through which the dataset is balanced.

- Undersampling
- Upsampling

4.9 Chapter summary:

The main content is explained in this chapter to understand business based on any specific topics. Before implementing the credit card fraud detection model, in this chapter, the business understanding phase of the CRISP-DM method is followed. Moreover, detailed ideas about various machine learning algorithms have been given.

Chapter-5: Modeling

The approach we have for our proposed project is mentioned in this chapter. The complete project design is shown here. Moreover, the algorithms used to create the models are explained. All types of measurements to verify the performance of the models are explained in this chapter.

5.1 Data Mining:

It is a process by which all possible patterns are examined from a large source dataset. Data mining is much like extracting precious metals from mines. Data analysts use data mining techniques to extract useful information from deep datasets. Data mining can also be called by other names, such as - Knowledge discovery, information harvesting, etc. Various tasks of machine learning, artificial intelligence, statistical information, and database are performed through data mining. Through this problem like fraud detection is solved.

There are generally 8 types of data mining techniques.

1. Classification.
2. Clustering.
3. Regression.
4. Outer.
5. Decision Trees.
6. Association.
7. Prediction.
8. Sequence patterns.

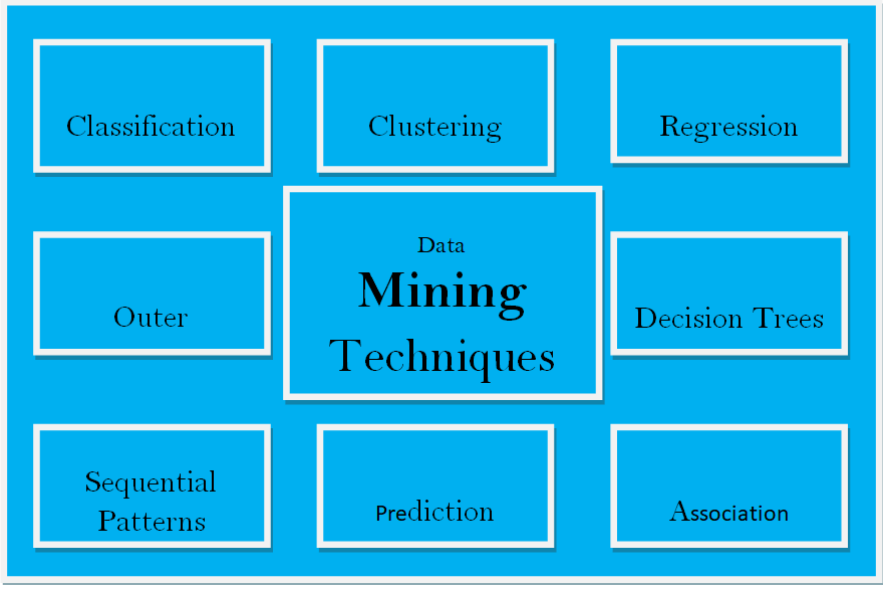


Figure 5.1: Data mining techniques.

In our project, we will work with the classification techniques of data mining to detect credit card fraud activity.

5.2 Machine Learning:

Machine learning is a special part of artificial intelligence, with the help of which an acceptable prediction of the result is obtained without programming any software or application. In this case, machine learning uses previously used databases as input.
There are three types of machine learning techniques.

1. Supervised.
2. Unsupervised.
3. Reinforcement.

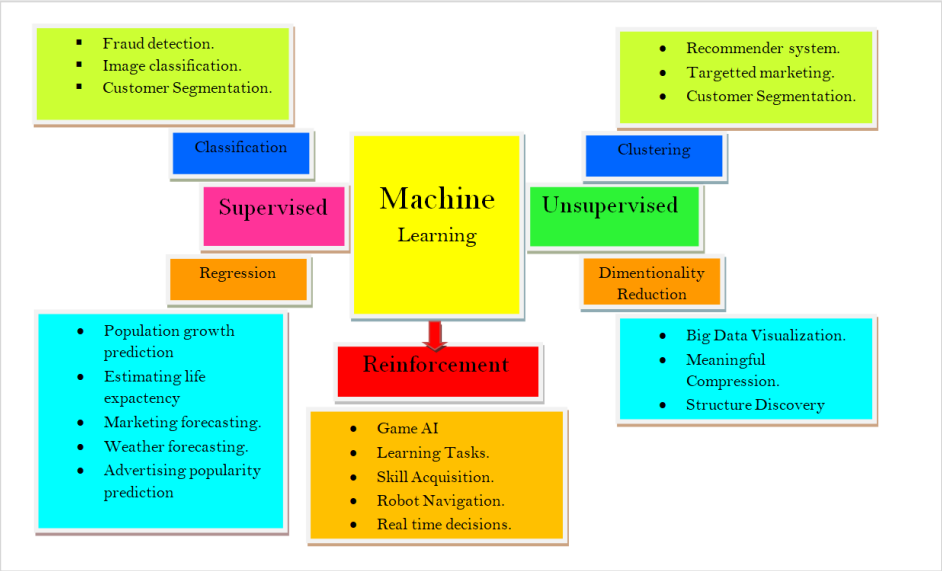


Figure 5.2: Machine Learning

5.2.1 Supervised:

Supervised learning is a special part of machine learning or artificial intelligence. Algorithms used in supervised learning are trained using labeled datasets, which then predict the results.
Supervised learning for data mining can be divided into two ways-

1. Classification.
2. Regression.

5.2.2 Unsupervised:

This is sometimes called unsupervised machine learning. The datasets that are not labeled are clustered and analyzed through unsupervised learning. It can analyze data without imparting any human knowledge.
The two main aspects of unsupervised machine learning are-

1. Clustering.
2. Dimensionality reduction.

5.2.3 Reinforcement:

This learning is the science of acquiring the ability to make decisions. By learning the whole of the environment, it acquires the knowledge to obtain the highest reward. Capabilities like GameAI, Learning tasks, and real-time decision-making can be achieved through it.

In our proposed project we have tried to achieve our desired results by using different algorithms of supervised learning. We proposed our classification approach in this chapter

5.3 Proposed Approach:

Our project is divided into two steps. In step 1 we implemented the model by our proposed logistic regression, random forest, decision tree, and XGboost algorithms, but here the dataset is not balanced. The dataset is divided into 0.75 train and 0.25 test portions.

In step 2 we first balanced the dataset using upsampling and downsampling methods. Just like before, the balanced dataset has been divided into 0.75 train and 0.25 test portions. Then the model is implemented with the help of suggested logistic regression, random forest, decision tree, and XGboost algorithm.

Finally, the best model has been selected considering its performance through a comparative analysis of the outcomes of all implemented models.

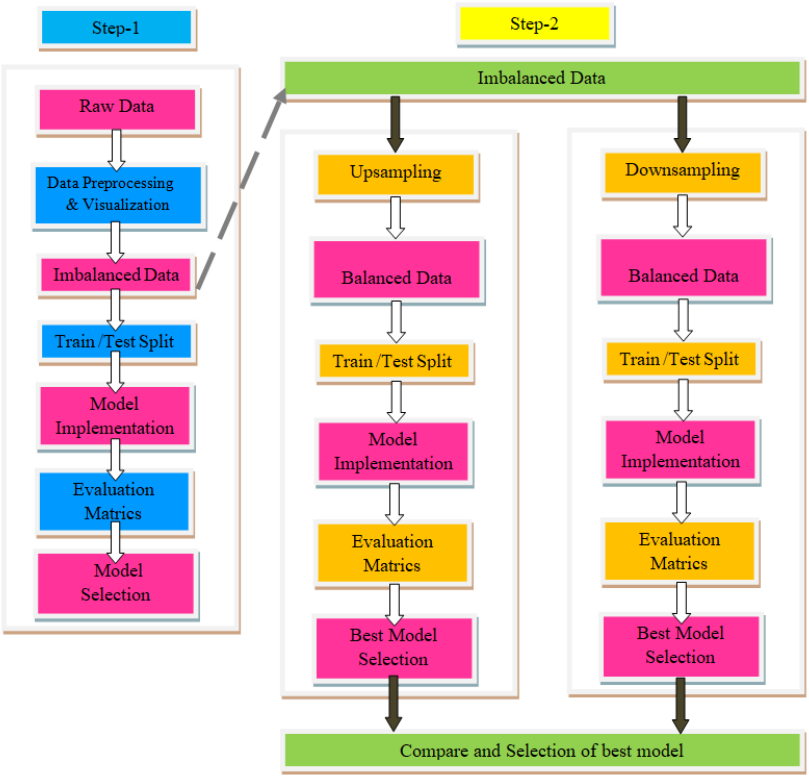


Figure 5.3: Proposed project approach

5.4 Individual classifier:

5.4.1 Logistic Regression:

Logistic regression is one of the common machine learning algorithms, which is capable of rapid analysis and easy analysis of the class characteristics of a dataset. Its capabilities depend on several features, such as the ability to add robust dataset classes. The ability to find discrepancies between different factors depends on the predictor variables and the results obtained. The logistic regression algorithm detects values greater than 1 and less than 0 to resolve any inconsistencies found in the dataset. This algorithm not only works for binomial classification and prediction but also uses a sigmoid function to extract multinominal results and predict the required value of the parameter. When a transaction occurs, logistic regression justifies that transaction and notifies whether the transaction should be continued, because this time it works for clustering.

GridsearchCV is used to increase the generalizability of the model to implement logistic regression in our case study.

Table 5.1: GridsearchCV settings for logistic regression

Parameter	**Description**	**Value**
Param_grid	This is the list of dictionaries for the parameter value.	0.01 Upsampling=1000 Undersampling=0.1
classifier	Chosen algorithm for implementing the model	LogisticRegression( )
scoring	It's assumed to implement a sci-kit-learn estimator interface	Roc_auc = 0.98
cv	Cv for generating cross-validation	5
verbose	It controls the verbosity of model	1

[25]

5.4.2 Decision Tree:

A decision tree is basically a type of algorithm where the goal is classified through calculation and the future outcome is predicted. This process can be likened to a tree. Because this algorithm has a root node, leaf node, and branch. First, a test is performed based on the properties of all the nodes in it, then the test results are assigned to each of its branches and the class label is attached to all the leaf nodes. When a similar tree is defined, this process happens continuously and all nodes are labeled with their attribute names and edges. Values associated with this label satisfy certain conditions and satisfy all its leaves that contain an intensity factor. This intensity factor is expressed as a ratio of the number of transactions that meet the conditions for all valid transactions. One of our algorithms used in the proposed dissertation is this decision tree through which we have created a model to detect fraud. Before implementing the model, the required part parameter value of gridsearchCV has been set.

Table 5.2: GridsearchCV settings for Decision tree

Parameter	Description	Value
max-depth	Maximum depth per tree	5 Upsampling=10 Undersampling=10
classifier	Chosen algorithm for implementing the model	DecisionTreeClassifier()
min_samples_leaf	Minimum required samples for leaf node	100 Upsampling =50 Undersampling =50
cv	Cv for generating cross-validation	5
verbose	It controls the verbosity of model	1
min_samples_split	Minimum required samples to split an internal node	100 Upsampling =50 Undersampling =50

[25]

5.4.3 Random Forest:

This algorithm can also be called random decision forest as another name. Random forest can performs classification, regression, or other tasks by creating multiple types of decision trees. Random forest is a supervised learning algorithm. The best part of this algorithm is that it can be used for both classification and regression tasks. Through this, it is possible to get very good accuracy.

A random forest algorithm is used for classification in our project While implementing the model, the required parameters and values of GridsearchCV have been set for cross-validation.

Table 5.3: GridsearchCV settings for Random forest

Parameter	Description	Value
max-depth	Maximum depth per tree	9 Upsampling = 9 Undersampling = 9
classifier	Chosen algorithm for implementing the model	RandomForestClassifier( )
scoring	It's assumed for implement sci-kit-learn estimator interface	Roc_auc
cv	Cv for generating cross-validation	5
n_estimator	Evaluate the cross-validation model on the test set	20 Upsampling=50 Undersampling=50
n_jobs	The number of jobs to run in parallel	-1

[25]

5.4.4 XGBoost:

A widely used machine learning algorithm is XGBoost through which models are implemented in tasks like fraud detection. It controls the imbalance of all classes by overfitting. It is an open-source implementation of the Gradient boost algorithm. Gradient boost is a supervised learning algorithm that combines all the predictions of the weak model to predict a target variable.

XGBoost minimizes the regularized objective function and adds penalty terms to reduce the convex loss function and complexity. Training creates a new tree and corrects any mistakes in the previous tree and adds it to the next tree. This is called gradient boosting because it involves the use of a gradient descent algorithm to reduce the loss when adding new models.

One of the most important algorithms for our project is XGBoost. Hyperparameter is used while implementing the model.

Table 5.4: GridsearchCV settings for XGboost

Parameter	**Description**	**Value**
max-depth	Maximum depth per tree	7 Upsampling =5 Undersampling =3
classifier	Chosen algorithm for implementing the model	XGBClassifier ( )
n_estimator	Evaluate the cross-validation model on the test set	130 Upsampling=130 Undersampling=150
min_child_weight	the minimum sum of instance weight (hessian) needed in a child [26]	2 Upsampling=1 Undersampling=2
n_jobs	The number of jobs to run in parallel	-1

[25]

5.5 Measures used for evaluating models:

Various types of measurement methods are used to understand the performance of the classification model. However, the classification report and confusion matrix are the most used in this case. With the help of a classification report and confusion matrix, it is possible to extract values like precision, F1-score, recall, sensitivity, and specificity, through which the efficiency of any machine learning model can be verified. Moreover, roc_curve is used to visually represent the accuracy of a model.

5.5.1 Confusion matrix:

The confusion matrix basically shows the performance of the classified model for test data. After creating a model, the confusion matrix compares the predicted class value with the actual class value, explaining how much a model can predict correctly and how much it can predict incorrectly.

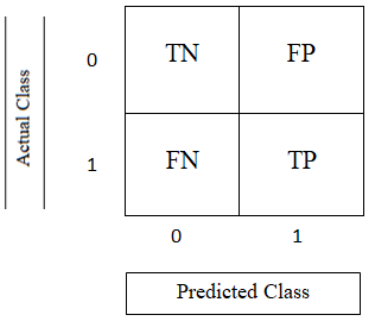


Figure 5.4: Confusion matrix

- **True positive (TP):**
  True positive value can correctly predict the positive value (fraud case) of the True class.
  For example: to predict the fraud case as fraud.
- **True Negative (TN):**
  True Negative value can correctly predict the Negative value (real case) of the True class.
  For example: to predict the real case as real.
- **False Negative (FN):**
  False Negative value incorrectly predicts positive value (fraud case) of true class as Negative (real case). For example: to predict the fraud case as real.
- **False Positive (FP):**
  False positive value predicts negative value (real case) of true class as incorrectly positive (fraud case). For example: predicting the real case as fraud.

5.5.2 Accuracy:

Accuracy is the ability to predict according to the class of the dataset records after implementing a model. Accuracy is expressed as a percentage. Accuracy can be measured from the confusion matrix.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

5.5.3 Precision:

Precision is a type of classification measurement by which negative values are not labeled as positive. Precision is a measure of the performance of a model.

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

5.5.4 Recall:

In machine learning, recall means how many True Positive (TP) values can be obtained from the outcome after implementing a model. In other words how many correct hits are achieved.

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

5.5.5 F1-score:

The F1-score is also called F score. Through this, the accuracy of a model is determined. F1-score is used to evaluate a binary classification model.

$$\text{F1-score} = 2\,\frac{\text{Precision x Recall}}{\text{Precision + Recall}}$$

5.5.6 ROC curve:

ROC curve is also known as receiver operating characteristics. It is used to measure the performance of a classifier model. This curve plots the False Positive (FN) value relative to the False Positive (FP) value in a graph, thereby revealing the sensitivity of the classifier model.

5.5.7 Classification Report:

A model's precision, recall, F1-score, and support value are published in the classification report. The classification report is mainly used to select robust models, through which all models are compared.

5.6 Summary:

An overall picture of the entire project has been highlighted in the chapter. The modeling part of the proposed model is adopted from here. Moreover, all measurements required to evaluate the project have been explained.

Chapter 6: Data visualization & Exploration

Data visualization is trying to understand all the details of the data by presenting the data graphically, through which the pattern, trend, and correlation of the data can be easily identified. In this chapter, we have reviewed our dataset in detail. Jupyter notebook platform is used for data visualization and exploration and the dataset is collected from Kaggle. All required libraries have been imported into the Jupyter notebook platform. Some popular libraries include

1. Matplot library - has a lot of freedom and is low-level.
2. Pandas visualization – It is built on matplot library and its UI is easy to use.
3. Seaborn - has a higher level UI and lots of default styles.

This chapter represents the data understanding and preparation phase of C.R.I.S.P-DM

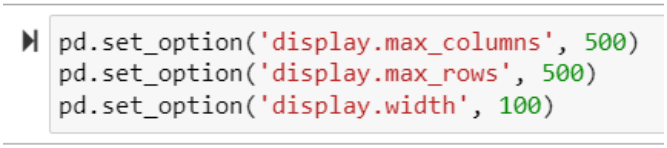
6.1 Importing libraries:

Before starting the data visualization process, all the necessary visualization libraries have been imported. Python provides several excellent graphical libraries. Here params have been imported from pandas, NumPy, matplotlib, seaborn, and pylab to visually represent all the data.

Figure 6.1: Importing libraries

6.2 Default settings:

We use it to change the default number of rows/columns and width to be displayed



6.3 Dataset:

The dataset used in the project is collected from Kaggle. Basically, all credit card transaction data is stored in this dataset. In 2013, all the transactions used by all card users of the European Union for 2 days are in this dataset.

According to the data mentioned in Kaggle, there are 284807 transactions, of which 492 are fraudulent transactions.

6.3.1 Reading dataset:

The chosen dataset is first uploaded to the Jupyter notebook platform.

Reading the dataset

```
c_data = pd.read_csv('creditcard.csv')
c_data.head()
```

9]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852

Reading the dataset

```
c_data = pd.read_csv('creditcard.csv')
c_data.head()
```

[9]:

V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26
-0.991390	-0.311169	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115
0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895
0.717293	-0.165946	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097
0.507757	-0.287924	-0.631418	-1.059647	-0.684093	1.965775	-1.232622	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929
1.345852	-1.119670	0.175121	-0.451449	-0.237033	-0.038195	0.803487	0.408542	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292

V27	V28	Amount	Class
0.133558	-0.021053	149.62	0
-0.008983	0.014724	2.69	0
-0.055353	-0.059752	378.66	0
0.062723	0.061458	123.50	0
0.219422	0.215153	69.99	0

Figure 6.2: Dataset structure

It is seen that there are 31 features in the dataset, all of which are numerical data. Here, out of 31 columns, 24 columns have been transformed through principle component analysis (PCA) and are presented as V1 to V28. These data are real transaction data, so they have been transformed into the source dataset for security purposes. The main purpose of this

transformation is to hide the original information of the user. The remaining three features, time, class, and amount, have not been transformed. Class column 0 indicates real data and 1 indicates fraud data.

6.3.2 Total records:

The dataset shows 284807 records and 31 columns.

Dataset shape

```
# Checking data set shape
c_data.shape
```

```
12]: (284807, 31)
```

6.3.3 Dataset information:

The dataset used 67.4 Mb of memory space. Each PCA transformed column from V1 to V28 has type float and each feature has 284807 transactions. Without PCA transform, there are 284807 records in the three columns Time, Amount, and Class feature and among them, the data type of amount and time is float 64. The data type of the class feature is an integer.

```
#Checking data set full information
c_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Figure 6.3: Dataset information

6.3.4 Describe dataset:

Here we have tried to describe some statistical information of our dataset ie count, mean, standard deviation, minimum, maximum value of each feature of the dataset has been shown.

```
c_data.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01

```
c_data.describe()
```

	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
	e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	284807.000000	284807.000000
	e-16	1.473120e-16	8.042109e-16	5.282512e-16	4.456271e-15	1.426896e-15	1.701640e-15	-3.662252e-16	-1.217809e-16	88.349619	0.001727
	e-01	7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	5.212781e-01	4.822270e-01	4.036325e-01	3.300833e-01	250.120109	0.041527
	e+01	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	-1.029540e+01	-2.604551e+00	-2.256568e+01	-1.543008e+01	0.000000	0.000000
	e-01	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.171451e-01	-3.269839e-01	-7.083953e-02	-5.295979e-02	5.600000	0.000000
	e-02	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	1.659350e-02	-5.213911e-02	1.342146e-03	1.124383e-02	22.000000	0.000000
	e-01	1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	3.507156e-01	2.409522e-01	9.104512e-02	7.827995e-02	77.165000	0.000000
	e+01	2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	7.519589e+00	3.517346e+00	3.161220e+01	3.384781e+01	25691.160000	1.000000

Figure 6.4: Statistical information

The class feature shows minimum value 0 and maximum value 1 by which real and fraudulent transactions are meant.

6.3.5 Checking duplicated values:

It is seen that there are 1081 duplicate rows in 31 columns in the dataset.

Check Duplicated rows

```
c_data[c_data.duplicated()]
```

Out[17]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
33	26.0	-0.529912	0.873892	1.347247	0.145457	0.414209	0.100223	0.711206	0.176066	-0.286717	-0.484688	0.872490	0.851636	-0.571745
35	26.0	-0.535388	0.865268	1.351076	0.147575	0.433680	0.086983	0.693039	0.179742	-0.285642	-0.482474	0.871800	0.853447	-0.571822
113	74.0	1.038370	0.127486	0.184456	1.109950	0.441699	0.945283	-0.036715	0.350995	0.118950	-0.243289	0.578063	0.674730	-0.534231
114	74.0	1.038370	0.127486	0.184456	1.109950	0.441699	0.945283	-0.036715	0.350995	0.118950	-0.243289	0.578063	0.674730	-0.534231
115	74.0	1.038370	0.127486	0.184456	1.109950	0.441699	0.945283	-0.036715	0.350995	0.118950	-0.243289	0.578063	0.674730	-0.534231
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
282987	171288.0	1.912550	-0.455240	-1.750654	0.454324	2.089130	4.160019	-0.881302	1.081750	1.022928	0.005356	-0.541998	0.745036	-0.375165
283483	171627.0	-1.464380	1.368119	0.815992	-0.601282	-0.689115	-0.487154	-0.303778	0.884953	0.054065	-0.828015	-1.192581	0.944989	1.372532
283485	171627.0	-1.457978	1.378203	0.811515	-0.603760	-0.711883	-0.471672	-0.282535	0.880654	0.052808	-0.830603	-1.191774	0.942870	1.372621
284191	172233.0	-2.667936	3.160505	-3.355984	1.007845	-0.377397	-0.109730	-0.667233	2.309700	-1.639306	-1.449823	-0.508930	0.600035	-0.627313
284193	172233.0	-2.691642	3.123168	-3.339407	1.017018	-0.293095	-0.167054	-0.745886	2.325616	-1.634651	-1.440241	-0.511918	0.607878	-0.627645

1081 rows × 31 columns

Figure 6.5: Duplicate rows

So the duplicate rows have been removed from the dataset.

remove duplicated rows

```
c_data.drop_duplicates(keep='first', inplace=True)
```

I checked whether there is any duplicate row.

Check Duplicated rows again

```
c_data.duplicated().sum()
```

```
19]: 0
```

After removing the duplicate rows, there are 283726 transaction data in the dataset.

Dataset shape after drop duplicated column

```
c_data.shape
```

```
20]: (283726, 31)
```

6.3.6 Missing values:

It turns out that there are no missing values in the dataset used, which makes our preprocessing task a bit easier.

Check missing values

```
[21]: c_data.isnull().sum()
```

```
Out[21]: Time      0
         V1        0
         V2        0
         V3        0
         V4        0
         V5        0
         V6        0
         V7        0
         V8        0
         V9        0
         V10       0
         V11       0
         V12       0
         V13       0
         V14       0
         V15       0
         V16       0
         V17       0
         V18       0
         V19       0
         V20       0
         V21       0
         V22       0
         V23       0
         V24       0
         V25       0
         V26       0
         V27       0
         V28       0
         Amount    0
         Class     0
         dtype: int64
```

Figure 6.6: Missing values

6.3.7 Histogram:

We can see here that most of the feature variables lie around 0 only the time feature is very broad and not only around 0.

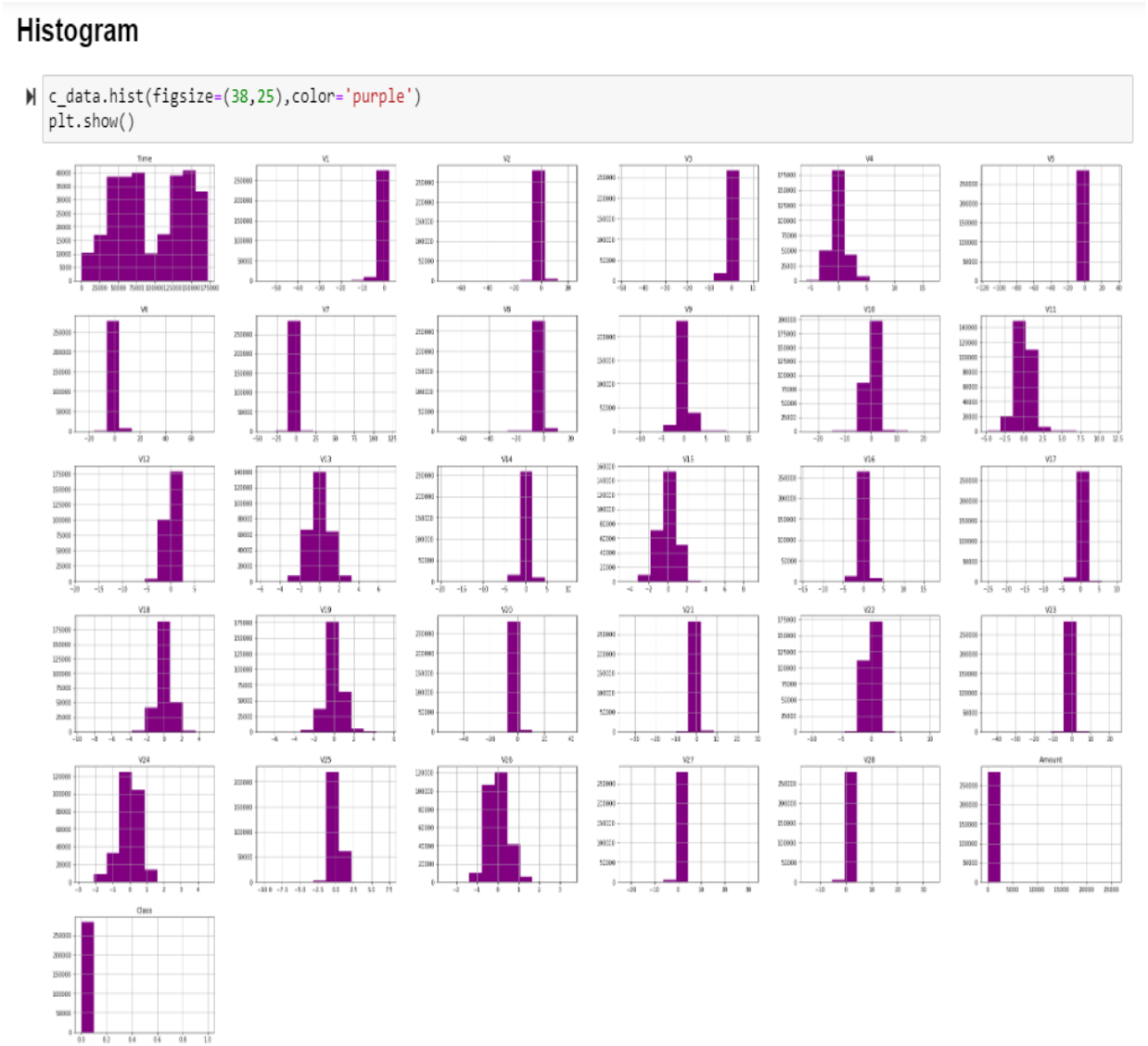


Figure 6.7: Histogram

6.4 Correlation:

The correlation between each feature located within the dataset is shown. Here it is seen that there is no correlation between amount/class and time/class.

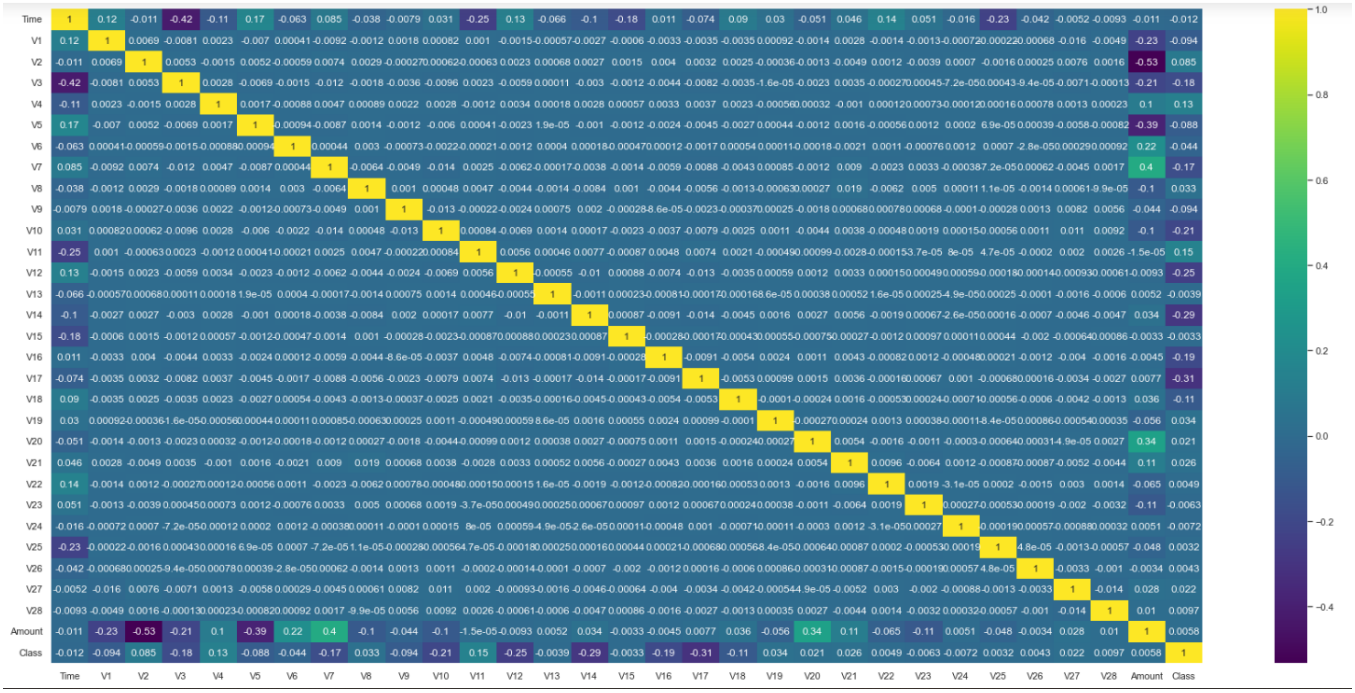


Figure 6.8: Correlation among columns

Figure 6.9: Correlation among Amount/Class/Time feature

6.4.1 Describe Amount/Class/Time feature:

The amount feature has a minimum value of 0 and maximum value of 25691.16, mean of 88.47, and standard deviation of 250.39.

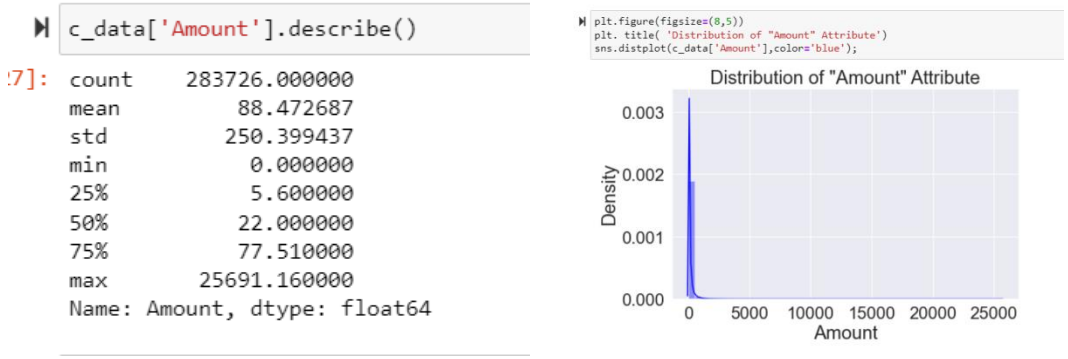


Figure 6.10: Distribution of amount feature

The class feature has a minimum value of 0 and maximum value of 1, a mean of 0.0017, and a standard deviation of 0.04.

Figure 6.11: Distribution of class feature

The time feature has a minimum value of 0 and maximum value of 172792, a mean of 94811.08, and a standard deviation of 47481.05.

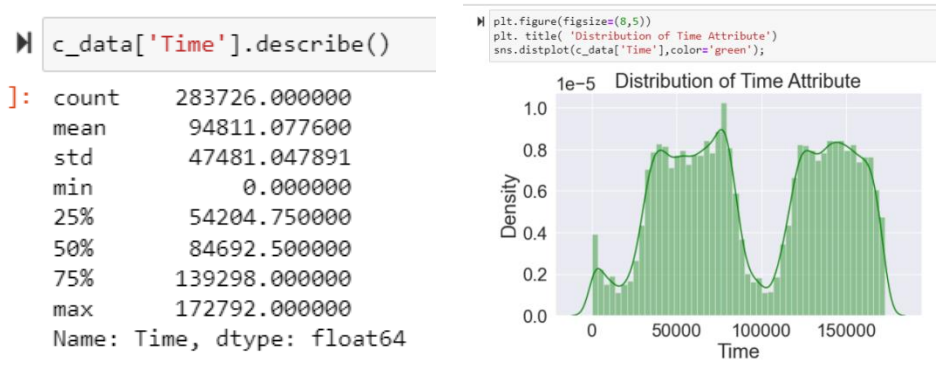


Figure 6.12: Distribution of time feature

6.5 Fraud and real transaction:

Credit card fraud and real transactions are shown through the class feature. The real case is indicated by 0 and the fraud case by 1.

There are 283253 real transactions and 473 fraudulent transactions.

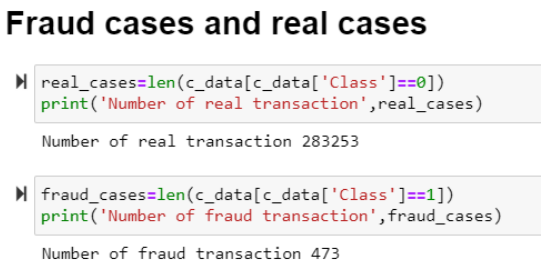


Figure 6.13: Total fraud and real case

6.6 Pie Chart:

Through the pie chart, it can be seen that the dataset contains 99.83% real cases and 0.17% fraud cases.

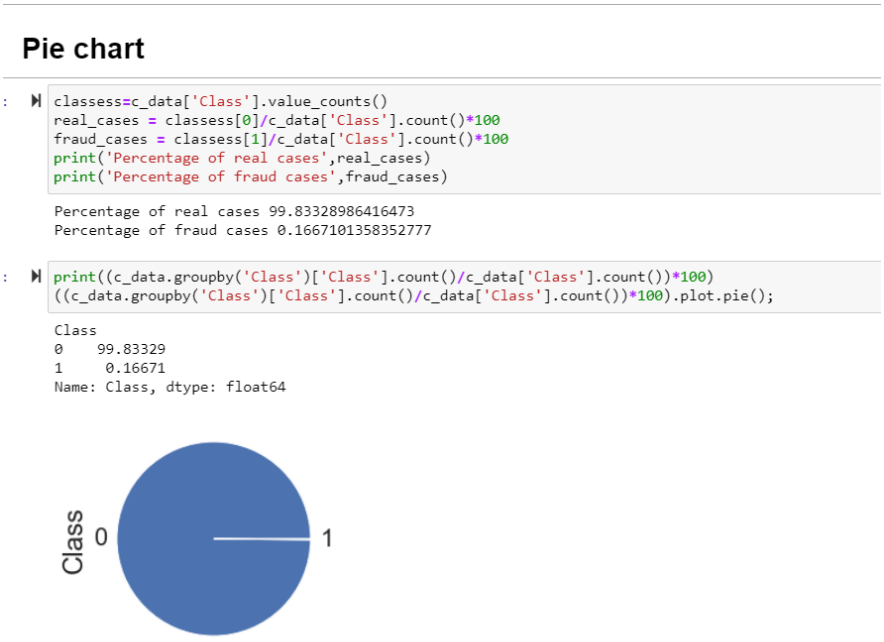


Figure 6.14: Percentage pie chart of fraud/real transaction

6.7 Comparing fraud and real cases:

A very large gap is observed in the class feature between Fraud and real cases. There is a huge difference between the values of V1, V3, V7, V16, and V17 and the amount column. So it can be said that the dataset is an Imbalanced dataset.

Figure 6.15: Difference between fraud/real transaction

6.8 Subplotting:

6.8.1 Time vs amount:

In the case of fraudulent transactions, it is seen that there is not much fraudulent activity in a very large amount of transactions. Fraudulent activity is very high in transactions of 0 to 1000 amounts. A small number of fraudulent activities are around 2000.

Moreover, in the case of a real transaction, it is seen that the concentration of real transactions is the highest between 0 and 10000. Between 25000 and 90000 seconds, fraud and real transactions are seen a little more. Between 100,000 and 120,000 second time, the transaction decreased a little and exited again from 125,000 second time

```
rcParams['figure.figsize']= 15, 8
f,(ax1, ax2) = plt.subplots(2, 1, sharex=True)
f. suptitle('Time of transaction vs Amount by Class')
ax1.scatter (fraud_cases.Time, fraud_cases.Amount)
ax1.set_title('Fraud Transaction')
ax2.scatter (real_cases.Time, real_cases.Amount)
ax2.set_title('Real Transaction')
plt.xlabel("Time (in seconds)")
plt.ylabel('Amount')
plt.show()
```

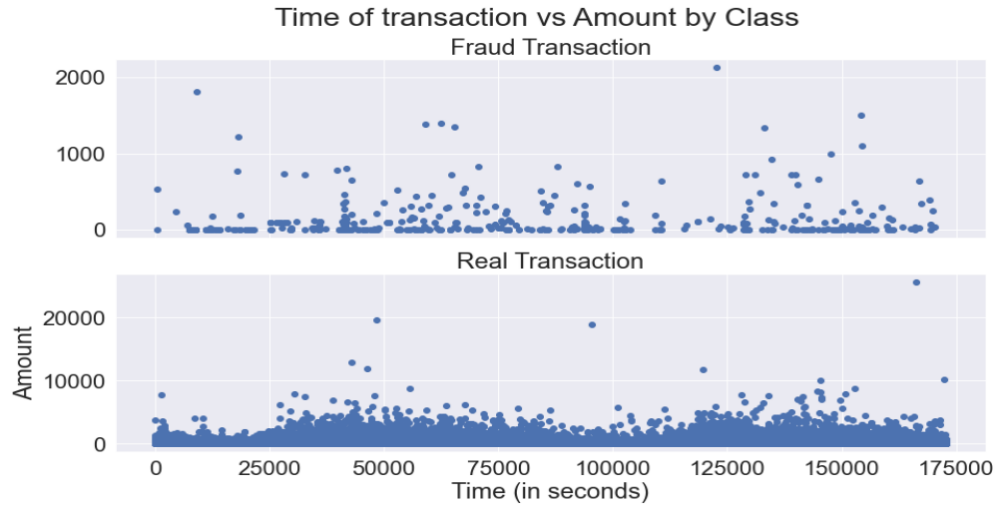


Figure 6.16: Time of transaction vs amount by class

6.8.2 Amount per transaction by class:

From the subplot, it is seen that the fraudulent activities are more under 100$ in the case of fraudulent transactions, where the number is around 300. On the other hand, real transactions are more than 0 to 500 $ whose number is more than 100000.

```
f, (ax1, ax2) = plt. subplots(2, 1, sharex=True)
f. suptitle('Amount per transaction by class')
bins = 50
ax1.hist (fraud_cases.Amount, bins = bins)
ax1.set_title(' Fraud Transaction')
ax2.hist (real_cases.Amount, bins = bins)
ax2.set_title('Real Transaction')
plt.xlabel('Amount')
plt.ylabel (' Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt. show();
```

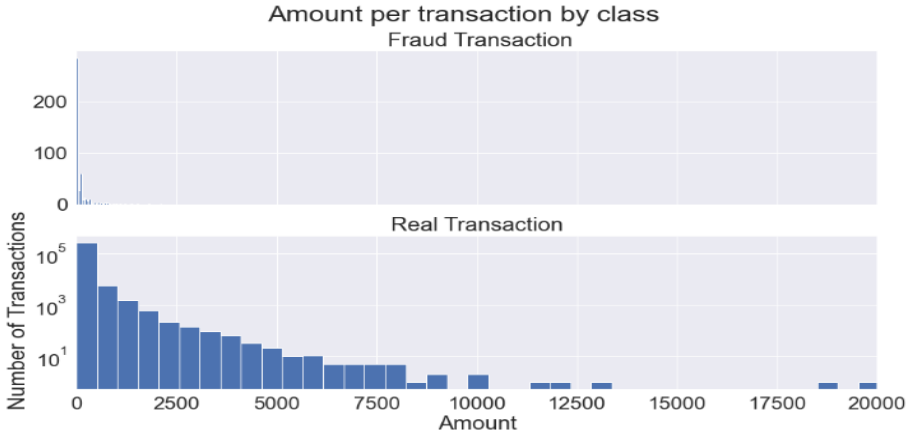


Figure 6.17: Amount per transaction by class

6.9 Imbalanced dataset:

The dataset contains 283253 real cases and 473 fraud cases out of 283726 records. Here it is clearly understood through the count plot that the dataset is completely unbalanced.

```
real_cases = c_data[c_data.Class==0]
fraud_cases = c_data[c_data.Class==1]
```

```
real_cases.shape
```

```
]: (283253, 31)
```

```
fraud_cases.shape
```

```
]: (473, 31)
```

```
plt.figure(figsize = (10,6))
sns.countplot(c_data['Class'], hue=c_data['Class'], palette = "Set1")
sns.set(font_scale=2)
plt.title('Real Vs Fraud transcations(Imbalanced Data)');
```

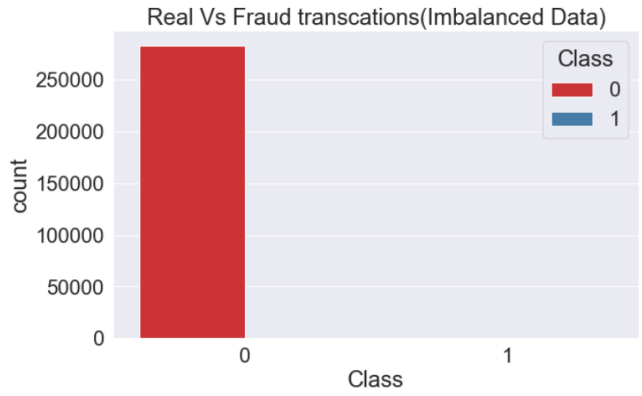


Figure 6.18: Imbalanced dataset

6.10 Summary:

In this chapter, the data understanding and data preparation phase of C.R.I.S.P-DM is completely completed. Here the dataset used in our project has been reviewed and analyzed very deeply by using various features and libraries of python. Data cleaning, missing value, and duplicate value has been removed in this part. That is, all the tasks for understanding the data and implementing the model have been completed in this chapter.

Chapter 7: Implementation

The assessment criteria phase of the CRISP-DM method is highlighted in this chapter. Fraud detection models have been implemented with the help of machine learning algorithms selected for our project in the previous modeling chapter. Our prescribed machine learning algorithms were-

- Logistic regression
- XGBoost
- Random forest
- Decision tree

First, we implemented the model through the imbalanced dataset and collected the outcomes. Then sampling method is used to balance the dataset. In this case, by applying upsampling and downsampling methods, the dataset is balanced and the outcome is saved by implementing the model. Later we will analyze all these outcomes in the evaluation chapter and try to find the best model.

7.1 Importing libraries:

With the help of our determined machine learning algorithm, an environment has been created for model processing and output by importing the required libraries before implementing the models. Here metrics, confusion_matrix, ConfusionMatrixDisplay, make_classification, f1-score, and classification report have been imported from sklearn. Moreover, required libraries fold, cross_val_score, and GridSearchCv have been imported for validation.

```
# Impoting metrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.datasets import make_classification
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report

# Importing libraries for cross validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

Figure 7.1: Importing necessary libraries

The given function is used to create a ROC curve function from our models,

```
# ROC Curve function
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(8, 8))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')#or[1 - True Negative Rate]
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    sns.set(font_scale=2)
    plt.show()

    return None
```

Figure 7.2: ROC curve function

7.2 Models:

We use four different machine learning classifiers and then compare different models. The classifiers used are the followings.

- Logistic regression
- XGBoost
- Decision tree
- Random forest

7.3 Model implementation without sampling:

7.3.1 Split dataset for an imbalanced dataset:

1. First the train_test_split library is imported.

```
# Import library
import sklearn
from sklearn.model_selection import train_test_split
```

2. Then our dataset is divided into dependent and independent features to split it into train and test portions.

```
# Putting feature variables into X
X = c_data.drop(['Class'], axis=1)
X.shape
```

```
(283726, 30)
```

```
# Putting target variable to y
y = c_data['Class']
y.shape
```

```
(283726,)
```

3. After splitting, there are 212794 feature variable and target variable records for training and 70932 records for testing.

```
# Splitting data into train and test set 75:25
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, test_size=0.25, random_state=123)
```

```
print('Total number of train data of feature variable:', X_train.shape)
print('Total number of test data of feature variable:', X_test.shape)
```

```
Total number of train data of feature variable: (212794, 30)
Total number of test data of feature variable: (70932, 30)
```

```
print('Total number of train data of target variable:', y_train.shape)
print('Total number of test data of target variable:', y_test.shape)
```

```
Total number of train data of target variable: (212794,)
Total number of test data of target variable: (70932,)
```

```
X_train.shape
```

```
62]: (212794, 30)
```

```
X_test.shape
```

```
63]: (70932, 30)
```

7.3.2 Feature scaling:

All columns of the dataset except the Amount column are PCA transformed. So here the amount column has been transformed using the standard scale.

```
from sklearn.preprocessing import StandardScaler
```

```
#before scaling amount data be like
c_data.Amount.head()
```

```
]: 0    149.62
   1      2.69
   2    378.66
   3    123.50
   4     69.99
   Name: Amount, dtype: float64
```

```
# Instantiate the Scaler
scaler = StandardScaler()
```

Scaling the train set

```
# Fit the train data into scaler and transform
X_train['Amount'] = scaler.fit_transform(X_train[['Amount']])
X_train['Amount'].head()
```

```
65]: 35255    -0.343448
     130762    0.272267
     21015     0.044806
     275966   -0.144066
     53182    -0.229717
     Name: Amount, dtype: float64
```

Scaling the test set

```
#I don't fit scaler on the test set. I only transform the test set.
```

```
# Transform the test set
X_test['Amount'] = scaler.transform(X_test[['Amount']])
X_test['Amount'].head()
```

```
7]: 66926     0.835156
    234206   -0.318191
    163840    4.244214
    236675   -0.127751
    210159   -0.072886
    Name: Amount, dtype: float64
```

Figure 7.3: Feature scaling

7.3.3 Checking the skewness:

If there is skewness in the dataset, it may be difficult to get a 100% outcome from the models, so the skewness of the dataset is checked first.

Checking the Skewness

```
# Listing the columns
cols = X_train.columns
```

```
# Plotting the distribution of the variables (skewness) of all the columns
k=0
plt.figure(figsize=(17,32))
sns.set(font_scale=1)
for col in cols :
    k=k+1
    plt.subplot(6, 5,k)
    sns.distplot(X_train[col])
    plt.title(col+' '+str(X_train[col].skew()))
```

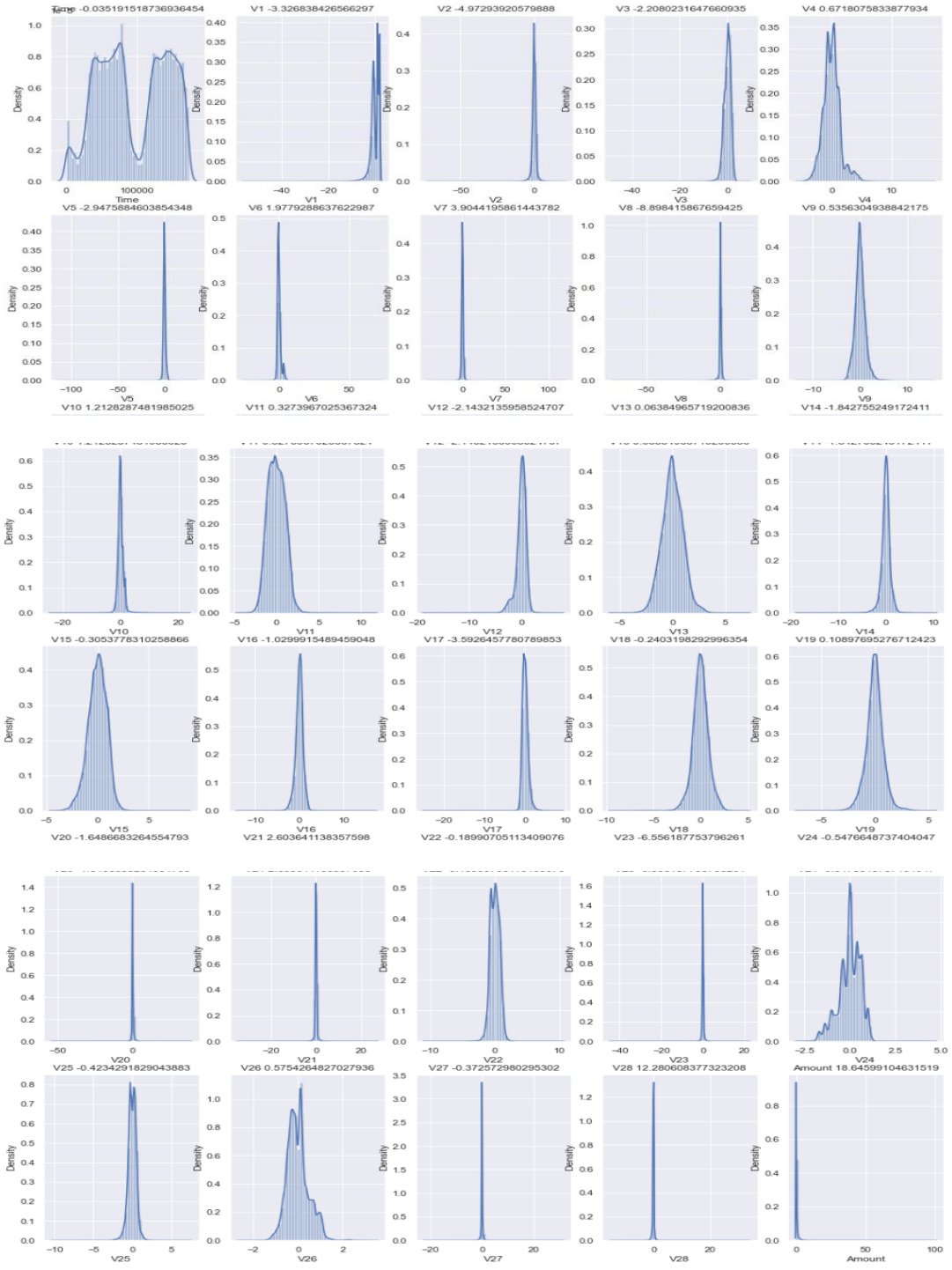


Figure 7.4: Features skewness

7.3.4 Reduce skewness:

We can see that there are several factors in our dataset that are highly skewed.

The power transformer is used here to reduce this skewness and bring the variables to normal distribution. A power transformer is a parametric or monotonic transformation set in which the data can be represented in a more Gaussian way.

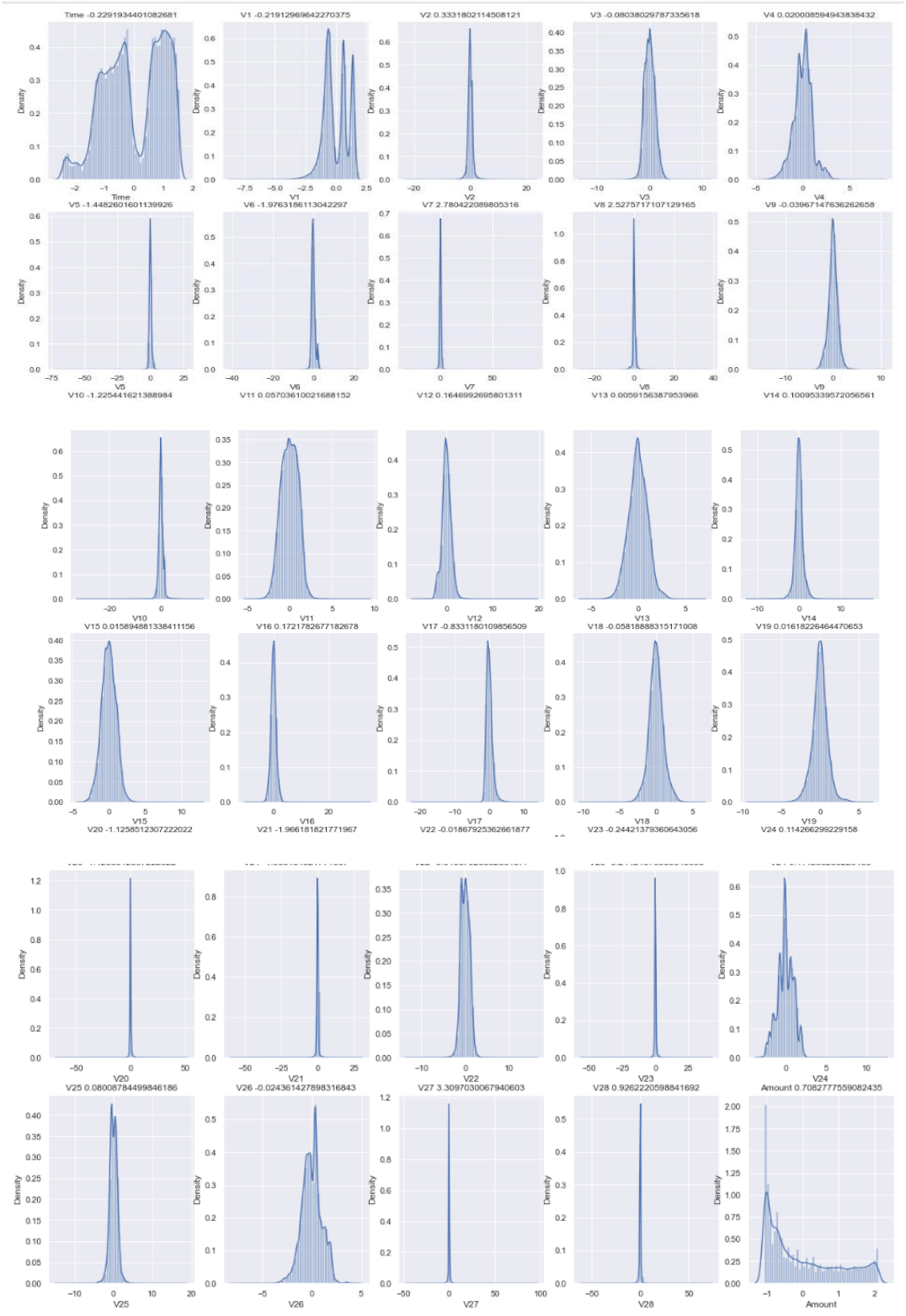


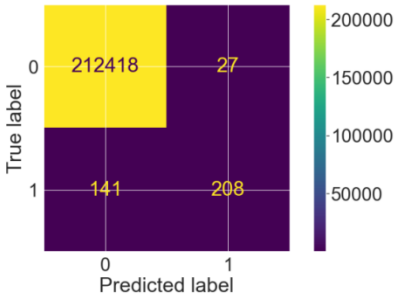
Figure 7.5: Features visualization after removing the skewness

7.3.5 Logistic Regression (without sampling):

GridsearchCV has been used to build this model through hyperparameter tuning, which can increase the generalizability of the model. Hyperparameter C is tuned here. C is the inverse of regularization strength in logistic regression. The optimal C value is used as 0.01 and 5 folds are kept. Later, while using the undersampling and upsampling method, the logistic regression model has been implemented in the same way.

Training performance of LR (without sampling):

Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by L.R is 208.
The true negative value predicted by L.R is 212418.
The false positive value predicted by L.R is 27.
The false negative value predicted by L.R is 141.

In the case of the training dataset wrong prediction of false positive (fraud but predicted as real) 27 and False negative (real but predicted as fraud) 141.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9992105040555654
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.5959885386819485
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9998729082821436
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.7123287671232876
```

Figure 7.6: Outputs of LR (Train model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    212445
           1       0.89      0.60      0.71       349

    accuracy                           1.00    212794
   macro avg       0.94      0.80      0.86    212794
weighted avg       1.00      1.00      1.00    212794
```

Figure 7.7: Classification result of LR (Train model without sampling)

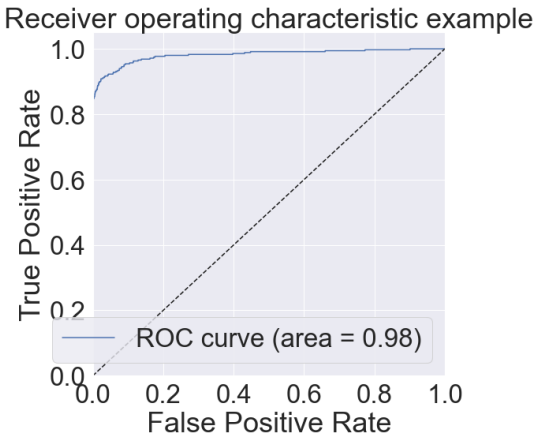
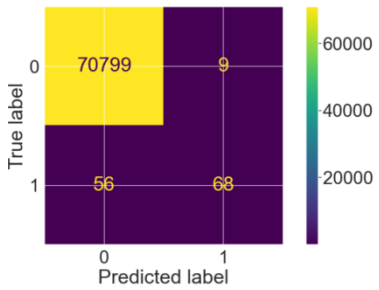


Figure 7.8: ROC curve of LR (Train model without sampling)

Testing performance of LR (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by L.R is 68.
The true negative value predicted by L.R is 70799.
The false positive value predicted by L.R is 9.
The false negative value predicted by L.R is 56.

In the testing dataset wrong prediction of false positive (fraud but predicted as real) 9 and False negative (real but predicted as fraud) 56.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.99908362939153
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.5483870967741935
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.999872895717998
```

```
# F1 score
print("F1-Score:-", f1_score(y_test, y_test_pred))
```

```
F1-Score:- 0.6766169154228856
```

Figure 7.9: Outputs of LR (Test model without sampling)

Classification report:

```
#classification_report
print(classification_report(y_test, y_test_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	70808
1	0.88	0.55	0.68	124
accuracy			1.00	70932
macro avg	0.94	0.77	0.84	70932
weighted avg	1.00	1.00	1.00	70932

Figure 7.10: Classification report of LR (Test model without sampling)

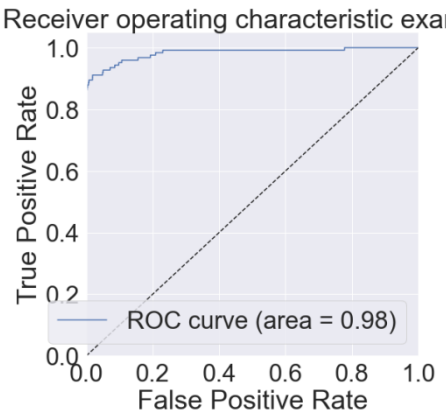
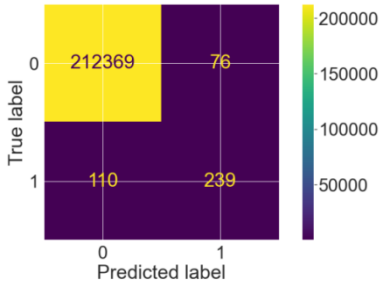


Figure 7.11: ROC curve of LR (Test model without sampling)

7.3.6 Decision tree (without sampling):

Before creating this model, a param_grid dictionary is first set for use in GridsearchCV where max_depth = 5, min_samples_leaf=100 and min_samples_split=100. Value 3 is kept as a fold value. Then the decision tree model is implemented. The same approach has been adopted in implementing the model using undersampling and upsampling methods.

Training performance of DT (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by D.T is 239.
The true negative value predicted by D.T is 212369.
The false positive value predicted by D.T is 76.
The false negative value predicted by D.T is 110.

The learning ability of the decision tree model is higher than that of logistic regression. The model has a false positive (fraud but predicted as real) prediction 76 and a false negative (real but predicted as fraud) prediction 110.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9991259152043761
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.6848137535816619
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9996422603497376
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.7198795180722891
```

Figure 7.12: Outcomes of DT (Train model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    212445
           1       0.76      0.68      0.72       349

    accuracy                           1.00    212794
   macro avg       0.88      0.84      0.86    212794
weighted avg       1.00      1.00      1.00    212794
```

Figure 7.13: Classification report of DT (Train model without sampling)

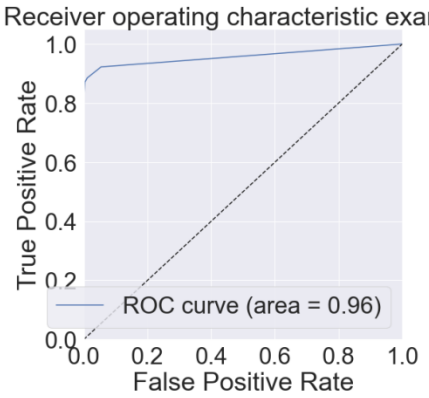
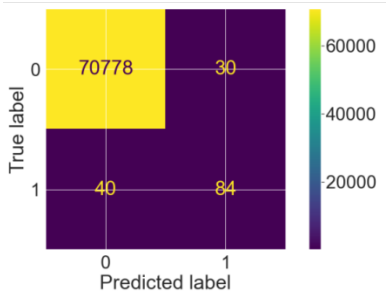


Figure 7.14: Roc curve of DT (Train model without sampling)

Testing performance of DT (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by D.T is 84.
The true negative value predicted by D.T is 70778.
The false positive value predicted by D.T is 30.
The false negative value predicted by D.T is 40.

In the case of the Test model, the model has a false positive (fraud but predicted as real) prediction 30 and a false negative (real but predicted as fraud) prediction 40.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9990131393447246
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.6774193548387096
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9995763190599932
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.7198795180722891
```

Figure 7.15: Outcomes of DT (Test model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     70808
           1       0.74      0.68      0.71       124

    accuracy                           1.00     70932
   macro avg       0.87      0.84      0.85     70932
weighted avg       1.00      1.00      1.00     70932
```

Figure 7.16: Classification report of DT (Test model without sampling)

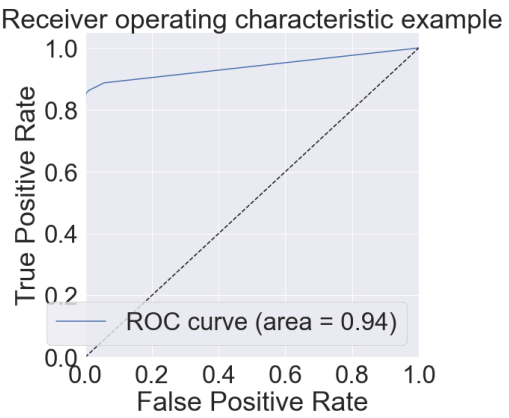
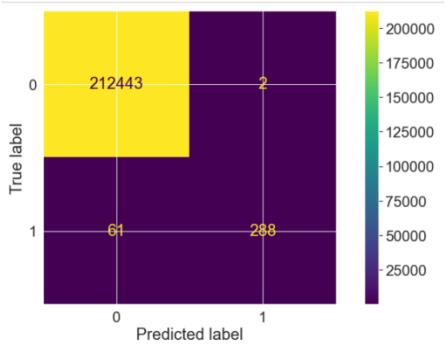


Figure 7.17: Roc curve of DT (Test model without sampling)

7.3.7 Random forest (without sampling):

GridsearchCV is used for cross-validation in this model. First, a param dictionary is used. GridsearchCV has CV set to 5 and n_jobs set to 1. From here max_depth 9 and n_estimator 20 are determined as the best parameters. Then the model is fitted using the best parameters.

Training performance of RF (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by R.F is 288.
The true negative value predicted by R.F is 212443.
The false positive value predicted by R.F is 2.
The false negative value predicted by R.F is 61.

False positive (fraud but predicted as real) prediction is less than false negative (real but predicted as fraud) prediction of random forest in the training model. FP is 2 here and FN is 61.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9997039390208371
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.8252148997134671
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9999905857986773
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9014084507042254
```

Figure 7.18: Outcomes of RF (Train model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    212445
           1       0.99      0.83      0.90       349

    accuracy                           1.00    212794
   macro avg       1.00      0.91      0.95    212794
weighted avg       1.00      1.00      1.00    212794
```

Figure 7.19: Classification report of RF (Train model without sampling)

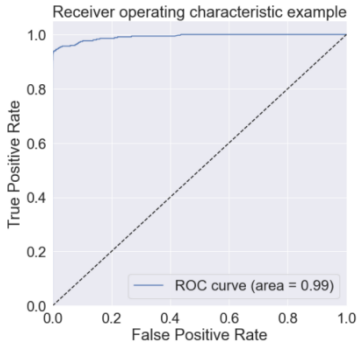
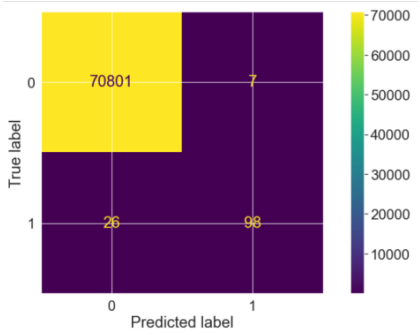


Figure 7.20: Roc curve of RF (Train model without sampling)

Testing performance of RF (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by R.F is 98.
The true negative value predicted by R.F is 70801.
The false positive value predicted by R.F is 7.
The false negative value predicted by R.F is 26.

The performance of random forest in the test model is better than logistic regression and decision tree. There are 7 false positives (fraud but predicted as real) predictions and 26 false negatives (real but predicted as fraud) predictions.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9995347656910845
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.7903225806451613
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9999011411139984
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9014084507042254
```

Figure 7.21: Outcomes of RF (Test model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     70808
           1       0.93      0.79      0.86       124

    accuracy                           1.00     70932
   macro avg       0.97      0.90      0.93     70932
weighted avg       1.00      1.00      1.00     70932
```

Figure 7.22: Classification report of RF (Test model without sampling)

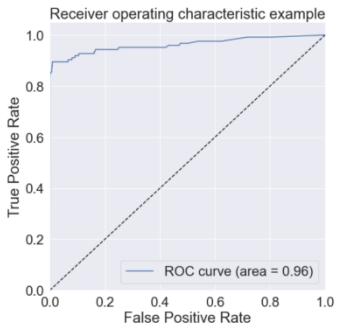
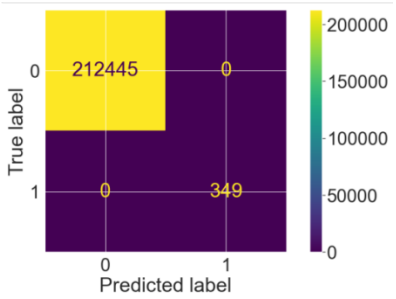


Figure 7.23: Roc curve of RF (Test model without sampling)

7.3.8 XGBoost (without sampling):

To implement the XGB model, first, a param dictionary is set using n_estimator, max_depth, and min_child_weight. Here CV 3 is taken in GridsearchCV for cross-validation and n_jobs=1. Then max_depth = 7, min_child_weight = 2 and n_estimator = 130 as the best parameters and the model is fitted.

Training performance of XGB (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by X.G.B is 349.
The true negative value predicted by X.G.B is 212445.
The false positive value predicted by X.G.B is 0.
The false negative value predicted by X.G.B is 0.

The learnability of XGBoost is higher than all the models here. No false positive (fraud but predicted as real) prediction and false negative (real but predicted as fraud) prediction here.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```
```
Accuracy:- 1.0
```
```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```
```
Sensitivity:- 1.0
```
```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```
```
Specificity:- 1.0
```
```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```
```
F1-Score:- 1.0
```

Figure 7.24: Outcomes of XGB (Train model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    212445
           1       1.00      1.00      1.00       349

    accuracy                           1.00    212794
   macro avg       1.00      1.00      1.00    212794
weighted avg       1.00      1.00      1.00    212794
```

Figure 7.25: Classification report of XGB (Train model without sampling)

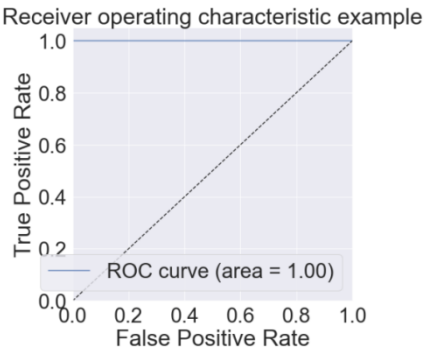
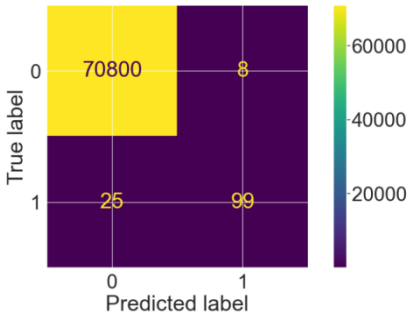


Figure 7.26: Roc curve of XGB (Train model without sampling)

Testing performance of XGB (without sampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by X.G.B is 99.
The true negative value predicted by X.G.B is 70800.
The false positive value predicted by X.G.B is 8.
The false negative value predicted by X.G.B is 25.

XGBoost's test model has false positive (fraud but predicted as real) prediction 8 and false negative (real but predicted as fraud) prediction 25. Here the performance of XGBoost and the random forest is similar.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9995347656910845
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.7983870967741935
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9998870184159981
```

```
# F1 score
print("F1-Score:-", f1_score(y_test, y_test_pred))
```

```
F1-Score:- 0.8571428571428571
```

Figure 7.27: Outcomes of XGB (Test model without sampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	70808
1	0.93	0.80	0.86	124
accuracy			1.00	70932
macro avg	0.96	0.90	0.93	70932
weighted avg	1.00	1.00	1.00	70932

Figure 7.28: Classification report of XGB (Test model without sampling)

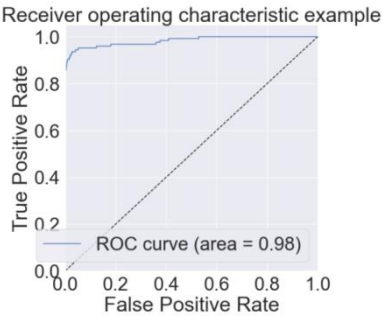


Figure 7.29: Roc curve of XGB (Test model without sampling)

7.4 Model implementation with Undersampling Method:

By using this method the dataset is balanced from imbalance. It is seen that an equal number of fraud and real data have been kept in the dataset. Now there is a total of 473 fraud data and 473 real data.

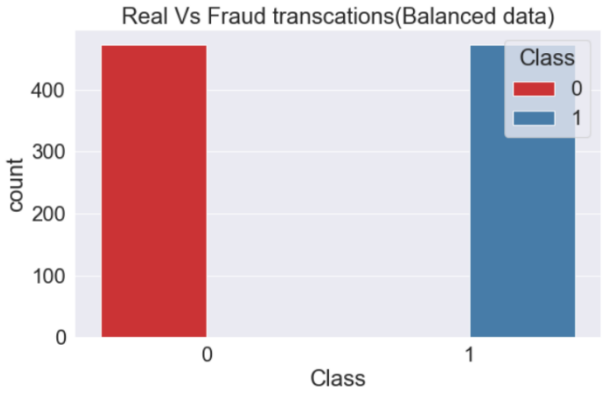


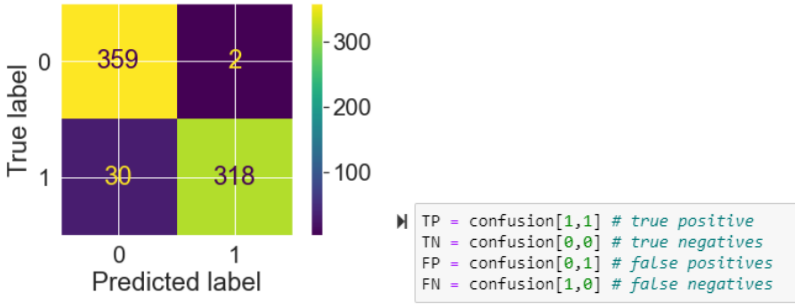
Figure 7.30: Balanced dataset using an undersampling technique

Here it is seen that the dataset is completely balanced. The dataset is divided into 0.75 train and 0.25 test portions in the same way as before and scaling and skewness are reduced. Then the models are implemented using a balanced dataset.

7.4.1 Logistic regression (undersampling):

In this case, 5 folds are kept for greadsearchCV. The param dictionary is specified with several values. Verbose is set to 1 in GridsearchCV. Then the value of best C is found to be 0.1 through GridsearchCV. The model is fitted with this optimal C.

Training performance of LR (undersampling):
Confusion matrix:



Here,

The true positive value predicted by L.R is 318.
The true negative value predicted by L.R is 359.
The false positive value predicted by L.R is 2.
The false negative value predicted by L.R is 30.

In the training phase of the model, the value of false positive (fraud but predicted as real) prediction is much lower than that of FN. There were only 2 FP and 30 FN values.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9548660084626234
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.9137931034482759
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9944598337950139
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9520958083832336
```

Figure 7.31: Outcomes of LR (Train model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.99      0.96       361
           1       0.99      0.91      0.95       348

    accuracy                           0.95       709
   macro avg       0.96      0.95      0.95       709
weighted avg       0.96      0.95      0.95       709
```

Figure 7.32: Classification report of LR (Train model with undersampling)

Roc curve:

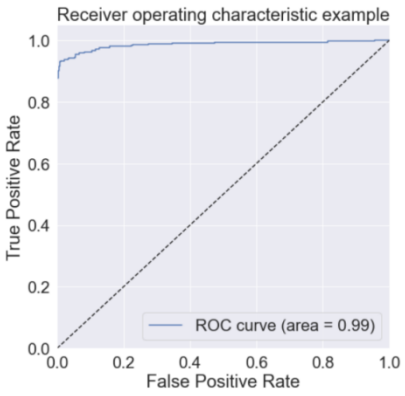
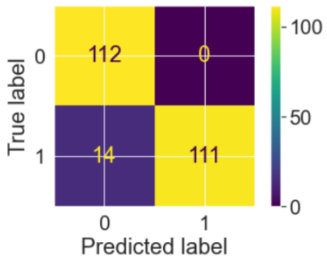


Figure 7.33: Roc curve of LR (Train model with undersampling)

Testing performance of LR (undersampling):
Confusion matrix:




```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

Here,

The true positive value predicted by L.R is 111.
The true negative value predicted by L.R is 112.
The false positive value predicted by L.R is 0.
The false negative value predicted by L.R is 14.

In terms of testing, logistic regression has done very well here. The false positive (fraud but predicted as real) prediction value is only 0 and the false negative (real but predicted as fraud) prediction is only 14.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```
```
Accuracy:- 0.9409282700421941
```
```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```
```
Sensitivity:- 0.888
```
```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```
```
Specificity:- 1.0
```
```
# F1 score
print("F1-Score:-", f1_score(y_test, y_test_pred))
```
```
F1-Score:- 0.940677966101695
```

Figure 7.34: Outcomes of LR (Test model with undersampling)

Classification report:

```
#classification_report
print(classification_report(y_test, y_test_pred))
```
```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       112
           1       1.00      0.89      0.94       125

    accuracy                           0.94       237
   macro avg       0.94      0.94      0.94       237
weighted avg       0.95      0.94      0.94       237
```

Figure 7.35: Classification report of LR (Test model with undersampling)

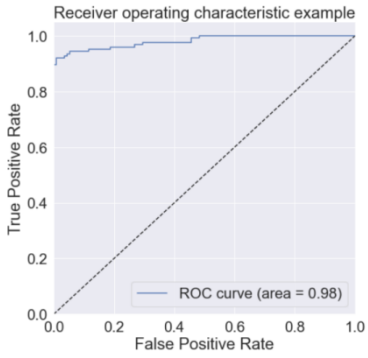
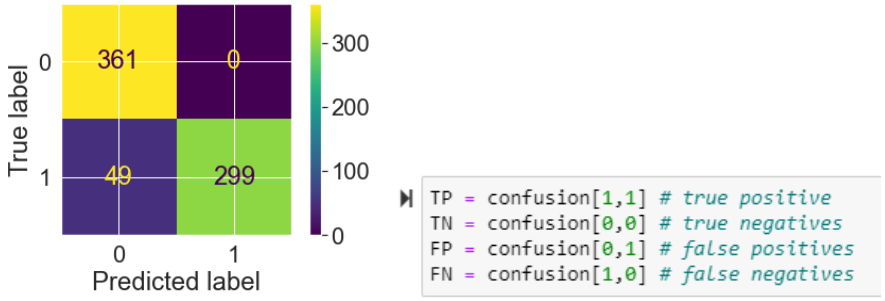


Figure 7.36: Roc curve of LR (Test model with undersampling)

7.4.2 Decision tree (undersampling):

In implementing the model through the undersampling technique, here a param dictionary has been set for GridsearchCV. GridsearchCV holds 3 CVs. The model is then fitted with max_depth=10, min_sample_split=50 and random_state=100.

Training performance of DT (undersampling):
Confusion matrix:



Here,

The true positive value predicted by D.T is 299.
The true negative value predicted by D.T is 361.
The false positive value predicted by D.T is 0.
The false negative value predicted by D.T is 49.

The train set had no value of false positive (fraud but predicted as real) prediction and false negative (real but predicted as fraud) prediction was 49. That is, the model has learned well.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9308885754583921
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.8591954022988506
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 1.0
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9242658423493045
```

Figure 7.37: Outcomes of DT (Train model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

	precision	recall	f1-score	support
0	0.88	1.00	0.94	361
1	1.00	0.86	0.92	348
accuracy			0.93	709
macro avg	0.94	0.93	0.93	709
weighted avg	0.94	0.93	0.93	709

Figure 7.38: Classification report of DT (Train model with undersampling)

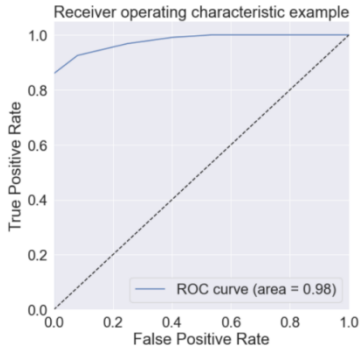
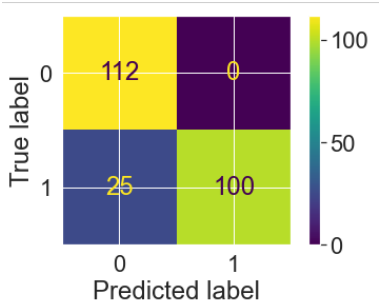


Figure 7.39: Roc curve of DT (Train model with undersampling)

Testing performance of DT (undersampling):

Confusion matrix:



Here,

The true positive value predicted by D.T is 100.
The true negative value predicted by D.T is 112.
The false positive value predicted by D.T is 0.
The false negative value predicted by D.T is 25.

In the case of the test model like the trained model false positive (fraud but predicted as real) prediction has no value and here false negative (real but predicted as fraud) prediction is 25, which is effective to reveal the good performance of the model.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.8945147679324894
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.8
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 1.0
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9242658423493045
```

Figure 7.40: Outcomes of DT (Test model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      1.00      0.90       112
           1       1.00      0.80      0.89       125

    accuracy                           0.89       237
   macro avg       0.91      0.90      0.89       237
weighted avg       0.91      0.89      0.89       237
```

Figure 7.41: Classification report of DT (Test model with undersampling)

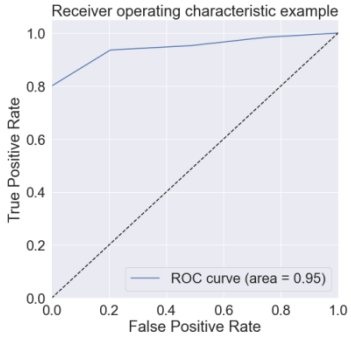
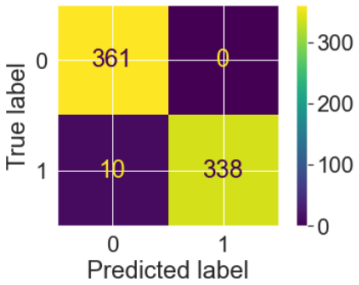


Figure 7.42: Roc curve of DT (Test model with undersampling)

7.4.3 Random forest (Undersampling):

Here the model is set to max_depth=9 and n_estimator = 50 using GridsearchCV. The random forest model is implemented using these parameter values.

Training performance of RF (undersampling):
Confusion matrix:



Here,

The true positive value predicted by R.F is 338.
The true negative value predicted by R.F is 361.
The false positive value predicted by R.F is 0.
The false negative value predicted by R.F is 10.

Compared to logistic regression and decision tree, the random forest has learned better. There are no false positive (fraud but predicted as real) predictions and only 10 false negatives (real but predicted as fraud) predictions.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9858956276445698
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.9712643678160919
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 1.0
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9854227405247813
```

Figure 7.45: Outcomes of RF (Train model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

	precision	recall	f1-score	support
0	0.97	1.00	0.99	361
1	1.00	0.97	0.99	348
accuracy			0.99	709
macro avg	0.99	0.99	0.99	709
weighted avg	0.99	0.99	0.99	709

Figure 7.46: Classification of RF (Train model with undersampling)

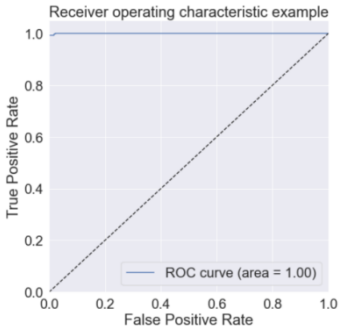
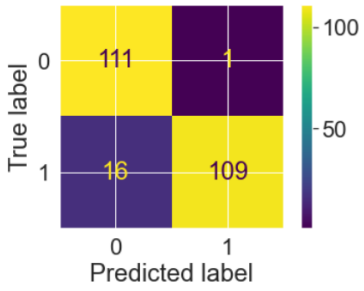


Figure 7.47: Roc curve of RF (Train model with undersampling)

Testing performance of RF (undersampling):
Confusion matrix:



Here,

The true positive value predicted by R.F is 109.
The true negative value predicted by R.F is 111.
The false positive value predicted by R.F is 1.
The false negative value predicted by R.F is 16.

Here false positive (fraud but predicted as real) prediction 1 and false negative (real but predicted as fraud) prediction 16 of random forest for the test model. The performance of random forest lags behind logistic regression.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9282700421940928
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.872
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9910714285714286
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9854227405247813
```

Figure 7.48: Outcomes of RF (Test model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       112
           1       0.99      0.87      0.93       125

    accuracy                           0.93       237
   macro avg       0.93      0.93      0.93       237
weighted avg       0.94      0.93      0.93       237
```

Figure 7.49: Classification report of RF (Test model with undersampling)

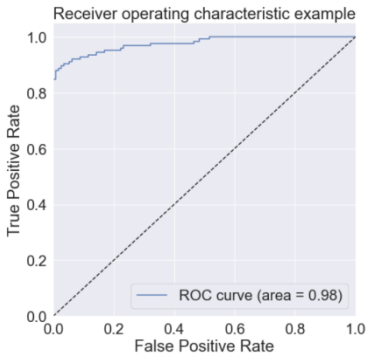
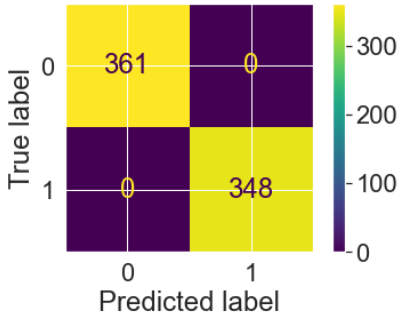


Figure 7.50: Roc curve of RF (Test model with undersampling)

7.4.4 XGBoost (undersampling):

GridsearchCV is used as cross-validation to implement XGBoost model through which max_depth=3, min_child_weight=2 and n_estimator=50 are obtained. Using these parameter values the model is implemented to get the best outcome.

Training performance of XGB (undersampling):
Confusion matrix:



Here,

The true positive value predicted by X.G.B is 348.
The true negative value predicted by X.G.B is 361.
The false positive value predicted by X.G.B is 0.
The false negative value predicted by X.G.B is 0.

The XGBoost model learned the best compared to all other models. No false positive (fraud but predicted as real) prediction and false negative (real but predicted as fraud) prediction here.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 1.0
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 1.0
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 1.0
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 1.0
```

Figure 7.51: Outcomes of XGB (Training model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       361
           1       1.00      1.00      1.00       348

    accuracy                           1.00       709
   macro avg       1.00      1.00      1.00       709
weighted avg       1.00      1.00      1.00       709
```

Figure 7.52: Classification report of XGB (Training model with undersampling)

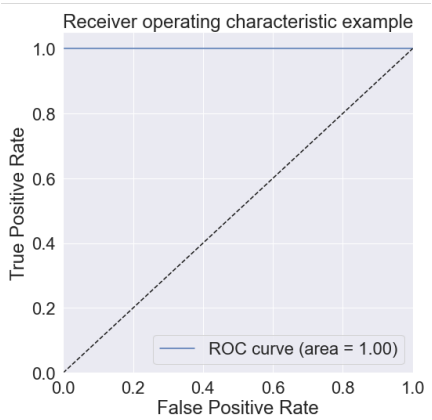
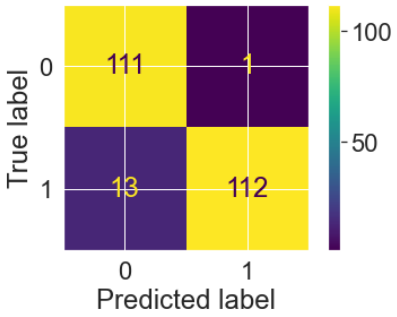


Figure 7.53: Roc curve of XGB (Training model with undersampling)

Testing performance of XGB (undersampling):
Confusion matrix:



Here,

The true positive value predicted by X.G.B is 112.
The true negative value predicted by X.G.B is 111.
The false positive value predicted by X.G.B is 1.
The false negative value predicted by X.G.B is 13.

Besides learning the dataset, XGBOOST is also the best for testing models. Here false positive (fraud but predicted as real) prediction is only 1 and false negative (real but predicted as fraud) prediction is only 13.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```
```
Accuracy:- 0.9409282700421941
```
```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```
```
Sensitivity:- 0.896
```
```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```
```
Specificity:- 0.9910714285714286
```
```
# F1 score
print("F1-Score:-", f1_score(y_test, y_test_pred))
```
```
F1-Score:- 0.9411764705882352
```

Figure 7.54: Outcomes of XGB (Testing model with undersampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```
```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       112
           1       0.99      0.90      0.94       125

    accuracy                           0.94       237
   macro avg       0.94      0.94      0.94       237
weighted avg       0.95      0.94      0.94       237
```

Figure 7.55: Classification report of XGB (Testing model with undersampling)

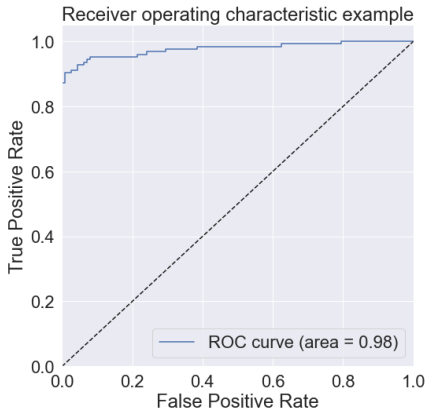


Figure 7.56: Roc curve of XGB (Testing model with undersampling)

7.5 Model implementation with Upsampling method:

Our dataset has only 0.17% fraud data and 0.99% real data, that is, the dataset was completely unbalanced. So the upsampling method is used to balance the dataset.

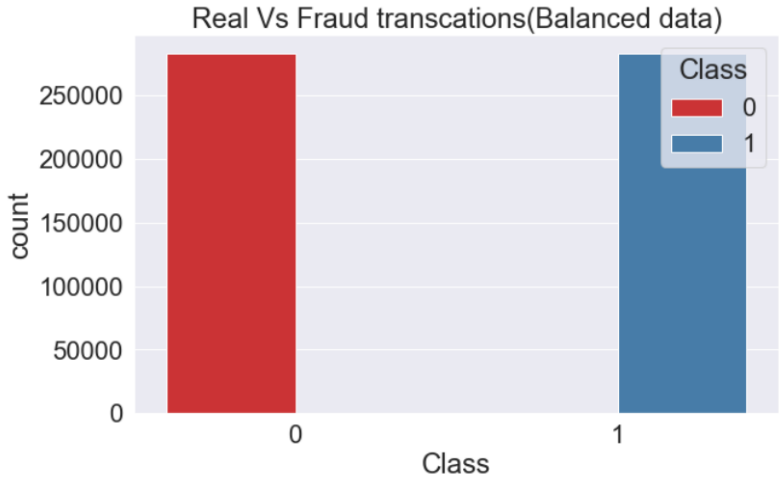


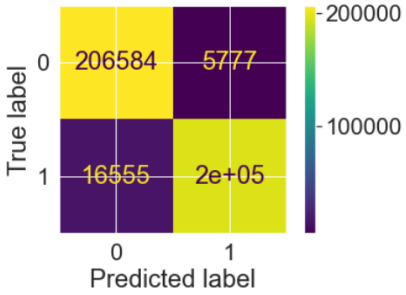
Figure 7.57: Balanced dataset using an upsampling technique

Here, the model is implemented by splitting 283253 fraud and 283253 real data into 0.75% train and 0.25% test portions in the balanced dataset and skewness have been removed from the dataset.

7.5.1 Logistic Regression (Upsampling):

In this case, using GridsearchCV cross-validation, the best C is 1000. Then the dataset is fitted with the value of beat C.

Training performance of LR (upsampling):
Confusion matrix:



Here,

The true positive value predicted by L.R is 200000.
The true negative value predicted by L.R is 206584.
The false positive value predicted by L.R is 5777.
The false negative value predicted by L.R is 16555.

The value of false positive (fraud but predicted as real) prediction is much higher than FP in the model. Here FP is 5777 while false negative (real but predicted as fraud) prediction is more than double at 16555.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```

```
Accuracy:- 0.9474391532648119
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.92210071617463
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9727963232420266
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.946091566125458
```

Figure 7.58: Outcomes of LR (Training model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95    212361
           1       0.97      0.92      0.95    212518

    accuracy                           0.95    424879
   macro avg       0.95      0.95      0.95    424879
weighted avg       0.95      0.95      0.95    424879
```

Figure 7.59: Classification report of LR (Training model with upsampling)

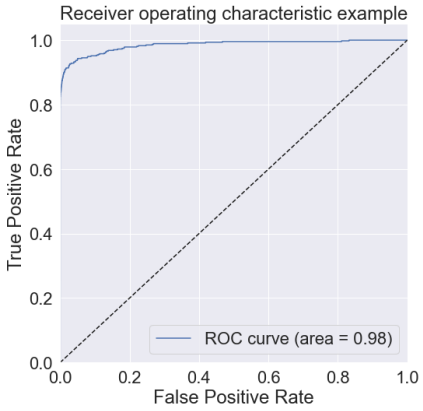
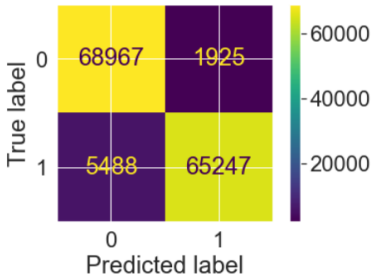


Figure 7.60: Roc auc of LR (Training model with upsampling)

Testing performance of LR (upsampling):
Confusion matrix:



Here,

The true positive value predicted by L.R is 65247.
The true negative value predicted by L.R is 68967.
The false positive value predicted by L.R is 1925.
The false negative value predicted by L.R is 5488.

False positive (fraud but predicted as real) prediction 1925 and false negative (real but predicted as fraud) prediction 5488 in the logistic regression model for testing.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9476582854964096
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.9224146462147452
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9728460192969588
```

```
# F1 score
print("F1-Score:-", f1_score(y_test, y_test_pred))
```

```
F1-Score:- 0.946246383432313
```

Figure 7.61: Oucomes of LR (Testing model with upsampling)

Classification report:

```
#classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     70892
           1       0.97      0.92      0.95     70735

    accuracy                           0.95    141627
   macro avg       0.95      0.95      0.95    141627
weighted avg       0.95      0.95      0.95    141627
```

Figure 7.62: Classification report of LR (Testing model with upsampling)

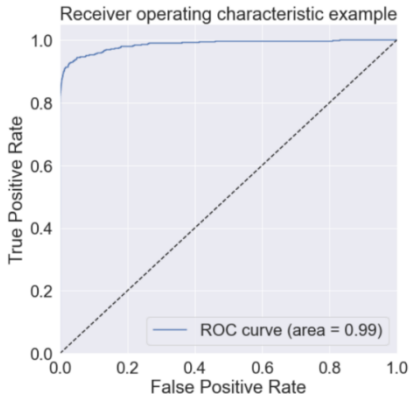
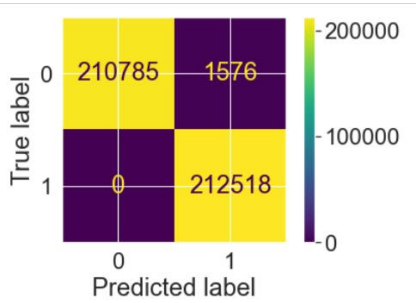


Figure 7.63: Roc auc of LR (Testing model with upsampling)

7.5.2 Decision tree (upsampling):

In implementing the decision tree model, max_depth=10, min_sample_leaf =50 and min_samples_split = 50 are determined through GridsearchCV. The model is implemented using these values.

Training performance of DT (upsampling):
Confusion matrix:



Here,

The true positive value predicted by D.T is 212518.
The true negative value predicted by D.T is 210785.
The false positive value predicted by D.T is 1576.
The false negative value predicted by D.T is 0.

In the case of the training decision tree has no false negative (real but predicted as fraud) prediction and false positive (fraud but predicted as real) prediction is only 1576, which is better than logistic regression.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
Accuracy:- 0.9962907086488153

# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
Sensitivity:- 1.0

# Specificity
print("Specificity:-", TN / float(TN+FP))
Specificity:- 0.9925786749921125

# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
F1-Score:- 0.9963057766776368
```

Figure 7.64: Outcomes of DT (Training model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00    212361
           1       0.99      1.00      1.00    212518

    accuracy                           1.00    424879
   macro avg       1.00      1.00      1.00    424879
weighted avg       1.00      1.00      1.00    424879
```

Figure 44: Classification report of DT (Training model with upsampling)

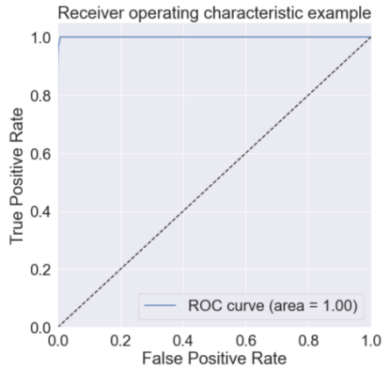
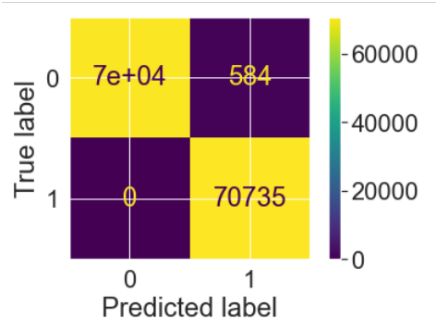


Figure 7.65: Roc curve of DT (Training model with upsampling)

Testing performance of DT (upsampling):
Confusion matrix:



Here,

The true positive value predicted by D.T is 70735.
The true negative value predicted by D.T is 70000.
The false positive value predicted by D.T is 584.
The false negative value predicted by D.T is 0.

Here decision tree performed better than logistic regression. There is only 1576 false positive (fraud but predicted as real) prediction and no false negative (real but predicted as fraud) prediction.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9958764924767171
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 1.0
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9917621170230774
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9963057766776368
```

Figure 7.66: Outcomes of DT (Testing model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

	precision	recall	f1-score	support
0	1.00	0.99	1.00	70892
1	0.99	1.00	1.00	70735
accuracy			1.00	141627
macro avg	1.00	1.00	1.00	141627
weighted avg	1.00	1.00	1.00	141627

Figure 7.67: Classification report of DT (Testing model with upsampling)

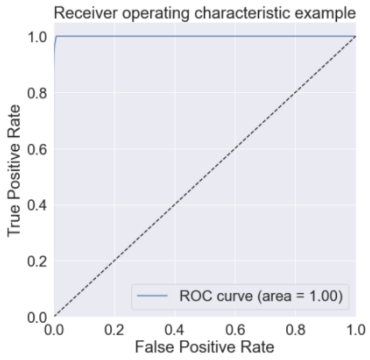
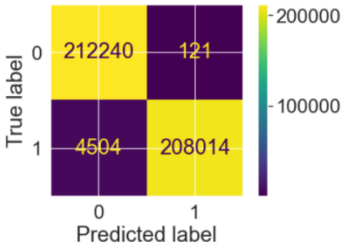


Figure 7.68: Roc curve of DT (Testing model with upsampling)

7.5.3 Random forest (Upsampling):

Max_depth =9 and n_estimator=50 selected by GridsearchCV. Here the random forest model is implemented through these values

Training performance of RF (upsampling):
Confusion matrix:



Here,

The true positive value predicted by R.F is 208014.
The true negative value predicted by R.F is 212240.
The false positive value predicted by R.F is 121.
The false negative value predicted by R.F is 4504.

Here in the case of the training dataset wrong prediction of false positive (fraud but predicted as real) 121 and False negative (real but predicted as fraud) 4504.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```
```
Accuracy:- 0.9891145479065805
```
```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```
```
Sensitivity:- 0.9788065010963777
```
```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```
```
Specificity:- 0.9994302155292167
```
```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```
```
F1-Score:- 0.9890051895505322
```

Figure 7.69: Outcomes of RF (Training model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```
```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99    212361
           1       1.00      0.98      0.99    212518

    accuracy                           0.99    424879
   macro avg       0.99      0.99      0.99    424879
weighted avg       0.99      0.99      0.99    424879
```

Figure 7.70: Classification report of RF (Training model with upsampling)

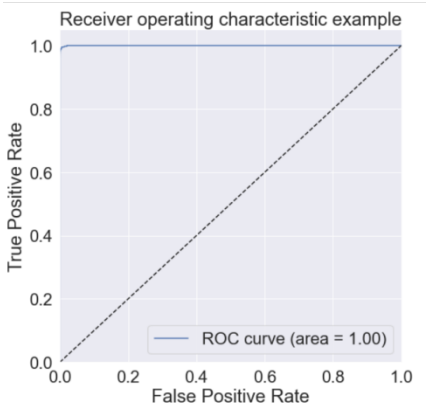
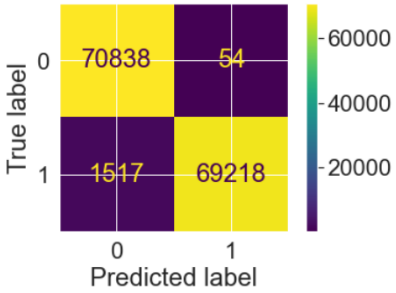


Figure 7.71: Roc curve of RF (Training model with upsampling)

Testing performance of RF (upsampling):
Confusion matrix:



Here,

The true positive value predicted by R.F is 69218.
The true negative value predicted by R.F is 70838.
The false positive value predicted by R.F is 54.
The false negative value predicted by R.F is 1517.

False positive (fraud but predicted as real) 54 and false negative (real but predicted as fraud) prediction 1517 of random forest model in case of the Test model.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9889074823303466
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 0.9785537569802785
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9992382779439147
```

```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```

```
F1-Score:- 0.9890051895505322
```

Figure 7.72: Outcomes of RF (Testing model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     70892
           1       1.00      0.98      0.99     70735

    accuracy                           0.99    141627
   macro avg       0.99      0.99      0.99    141627
weighted avg       0.99      0.99      0.99    141627
```

Figure 7.73: Classification report of RF (Testing model with upsampling)

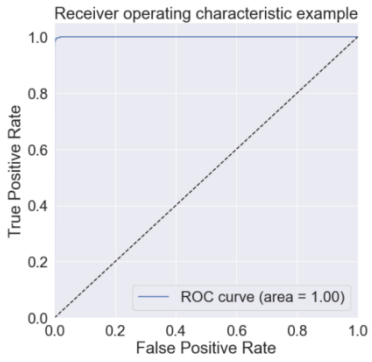
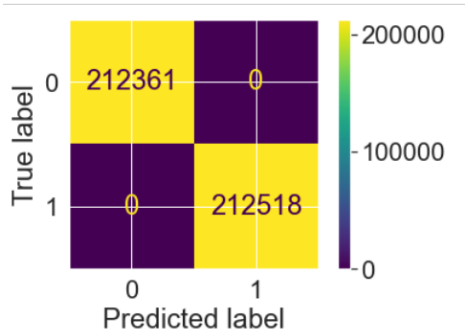


Figure 7.74: Roc curve of RF (Testing model with upsampling)

7.5.4 XGBoost (Upsampling):

Before fitting the XGBoost classifier, max_depth = 5, min_child_weight = 1 and n_estimator = 130 were set using GridsearchCV.

Training performance of XGB (upsampling):
Confusion matrix:



Here,

The true positive value predicted by X.G.B is 212518.
The true negative value predicted by X.G.B is 212361.
The false positive value predicted by X.G.B is 0.
The false negative value predicted by X.G.B is 0.

XGboost predicts the best among the rest of the models. XGBoost has no false positive (fraud but predicted as real) and false negative (real but predicted as fraud) prediction.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_train, y_train_pred))
```
```
Accuracy:- 1.0
```
```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```
```
Sensitivity:- 1.0
```
```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```
```
Specificity:- 1.0
```
```
# F1 score
print("F1-Score:-", f1_score(y_train, y_train_pred))
```
```
F1-Score:- 1.0
```

Figure 7.75: Outcomes of XGB (Training model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_train, y_train_pred))
```
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    212361
           1       1.00      1.00      1.00    212518

    accuracy                           1.00    424879
   macro avg       1.00      1.00      1.00    424879
weighted avg       1.00      1.00      1.00    424879
```

Figure 7.76: Classification report of XGB (Training model with upsampling)

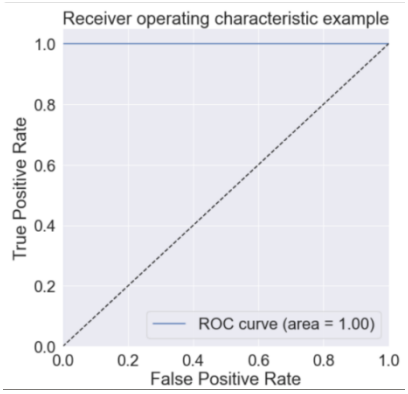
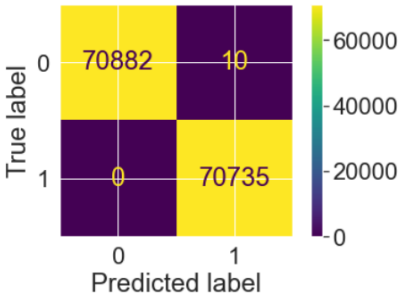


Figure 7.77: Roc curve of XGB (Training model with upsampling)

Testing performance of XGB (upsampling):
Confusion matrix:



Here,

The true positive value predicted by X.G.B is 70735.
The true negative value predicted by X.G.B is 70882.
The false positive value predicted by X.G.B is 10.
The false negative value predicted by X.G.B is 0.

XGBoost shows the best performance for test models like the Train model. There are only 10 false positives (fraud but predicted as real) and no false negatives (real but predicted as fraud) prediction.

Outcomes:

```
# Accuracy
print("Accuracy:-",metrics.accuracy_score(y_test, y_test_pred))
```

```
Accuracy:- 0.9999293919944643
```

```
# Sensitivity
print("Sensitivity:-",TP / float(TP+FN))
```

```
Sensitivity:- 1.0
```

```
# Specificity
print("Specificity:-", TN / float(TN+FP))
```

```
Specificity:- 0.9998589403599842
```

```
# F1 score
print("F1-Score:-", f1_score(y_test, y_test_pred))
```

```
F1-Score:- 0.9999293186316087
```

Figure 7.78: Outcomes of XGB (Testing model with upsampling)

Classification report:

```
# classification_report
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     70892
           1       1.00      1.00      1.00     70735

    accuracy                           1.00    141627
   macro avg       1.00      1.00      1.00    141627
weighted avg       1.00      1.00      1.00    141627
```

Figure 7.79: Classification report of XGB (Testing model with upsampling)

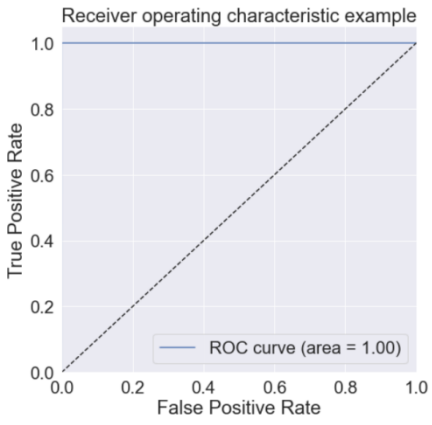


Figure 7.78: Roc curve of XGB (Testing model with upsampling)

Chapter 8: Evaluation

In this chapter, the results obtained from all the models implemented by us are discussed in comparison. Here the outcome of classifier models created separately through the imbalanced dataset and balanced datasets created through oversampling/upsampling has been investigated.

8.1 Result and discussion:

8.1.1 Model with Imbalanced Dataset:

Table 8.1: Output of train model with imbalanced dataset

Classifier	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
				F1-score	Precision	Recall	
Logistic regression	0.9992	0.5959	0.9998	0.7123	0.89	0.60	0.9813
Decision Tree	0.9991	0.6848	0.9996	0.7198	0.76	0.68	0.9576
Random forest	0.9997	0.8252	0.9999	0.9014	0.99	0.83	0.9921
XGBoost	**1**	**1**	**1**	**1**	**1**	**1**	**1**

The XGBoost model learned the best among the models we built. XGBoost's accuracy, sensitivity, specificity, F1-score, and precision value are the best among the rest of the models. Moreover, the roc_auc and recall value are 1 here which is higher than all other models. The random forest model implemented through an unbalanced dataset has learned well here in the training phase. The accuracy of R.F here was 0.9997 and roc_auc was 0.9921.

Table 8.2: Output of test model with imbalanced dataset

Classifier	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
				F1-score	Precision	Recall	
Logistic regression	0.9990	0.5483	0.9998	0.6766	0.88	0.55	**0.9830**
Decision Tree	0.9990	0.6774	0.9995	0.7198	0.74	0.68	0.9391
Random forest	**0.9995**	0.7903	**0.9999**	**0.9014**	**0.93**	0.79	0.9627
XGBoost	**0.9995**	**0.7983**	0.9998	0.8571	**0.93**	**0.80**	0.9823

The XGB model on the Imbalanced dataset learned the best in the training phase and the second best model was the random forest. XGboost and random forest perform well even in the testing phase of the model. Both RF and XGB have the best accuracy of 0.9995 in the testing phase. XGB obtained the best sensitivity (0.7983), precision (0.93), and recall (0.80) values among the models and R.F obtained the best specificity (0.9999), F1-score (0.9014). Logistic regression's roc_auc (0.9830) value is the best here, but all other values are relatively low. XGB's roc_auc is also very good at 0.9823.

8.1.2 Best model for Imbalanced dataset:

All the above models provide very good performance. However, random forest and XGBoost had the best performance among all the models. But if it is considered between XGBoost and random forest, then the XGBoost model is the best in an unbalanced dataset with 0.9995 accuracies, 0.80 recall, and 0.9823 roc_auc value.

8.1.3 Model with Undersampling:

Table 8.3: Output of train model with the balanced dataset (undersampling)

Classifier	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
				F1-score	Precision	Recall	
Logistic regression	0.9548	0.9137	0.9944	0.9520	0.99	0.91	0.9864
Decision Tree	0.9308	0.8591	**1**	0.9242	**1**	0.86	0.9786
Random forest	0.9858	0.9712	**1**	0.9854	**1**	0.97	0.9998
XGBoost	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Using the undersampling technique, our imbalanced dataset has been balanced. Then four models are implemented. All four models had good learning abilities. However, the XGboost model learns best in this case. In terms of train data, XGBoost's accuracy (1.0), sensitivity (1.0), specificity (1.0), F1-score (1.0), precision (1.0), recall (1.0) and roc_auc (1.0) were higher than the other three models. Decision tree and random forest also had better specificity (1.0) and precision (1.0) values among the models.

Table 8.4: Output of test model with the balanced dataset (undersampling)

Classifier	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
				F1-score	Precision	Recall	
Logistic regression	**0.9409**	**0.888**	**1**	0.9406	**1**	**0.89**	**0.9806**
Decision Tree	0.8945	0.8	**1**	0.9242	**1**	0.80	0.9519
Random forest	0.9282	0.872	0.9910	**0.9854**	0.99	0.87	0.9760
XGBoost	0.9282	**0.888**	0.9738	0.9288	0.97	**0.89**	0.9778

The models are created through a balanced dataset using the undersampling technique. Here it is seen that XGboost provides the best performance during training. But in terms of testing the models, it has been seen that the performance of logistic regression is the highest. Logistic regression accuracy (0.9409) and roc_auc (0.9806) are higher than all models. Moreover, specificity (1.0), precision (1.0), and sensitivity (0.888) are more than all. Recall (0.89) of logistic regression and recall (0.89) of XGBoost were the best. Although the random forest has the best F1-score (0.9854) among these models, random forest lags behind in terms of overall performance.

8.1.4 Best model for undersampling technique:

All four models implemented through the undersampling technique gave a good performance. In terms of the training model, XGboost was able to learn the best where accuracy, sensitivity, specificity, F1-score, precision, recall, and roc_auc value is better than all. But logistic regression provides the best performance in terms of the test model. The model had the highest accuracy of 0.9409.

Moreover recall (0.89) and roc_auc (0.9806) were better than all. So considering the performance of the testing model, the best model for the undersampling technique is logistic regression.

8.1.5 Model with upsampling:

Table 8.5: Output of train model with the balanced dataset (upsampling)

Classifier	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
				F1-score	Precision	Recall	
Logistic regression	0.9474	0.9221	0.9727	0.9460	0.97	0.92	0.9849
Decision Tree	0.9962	**1**	0.9925	0.9963	0.99	**1**	0.9997
Random forest	0.9891	0.9788	0.9994	0.9890	**1**	0.98	0.9998
XGBoost	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Another technique used in our project is an upsampling method.

Here also each model gives very good performance in terms of the training model. Accuracy and roc_auc of each model are more than 90%. However, the XGBoost model has done the best in learning the training model. Here its accuracy (1.0), sensitivity (1.0), specificity (1.0), F1-score (1.0), precision (1.0), recall (1.0), roc_auc(1.0) gives the best. Decision tree sensitivity (1.0) and recall (1.0) and random forest precision (1.0) compete with XGBoost, but in terms of overall performance, XGboost is ahead of all in the training model.

Table 8.6: Output of test model with the balanced dataset (upsampling)

Classifier	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
				F1-score	Precision	Recall	
Logistic regression	0.9476	0.9224	0.9728	0.9462	0.97	0.92	0.9856
Decision Tree	0.9958	**1**	0.9917	0.9963	0.99	**1**	0.9995
Random forest	0.9889	0.9785	0.9992	0.9890	**1**	0.98	0.9998
XGBoost	**0.9999**	**1**	**0.9998**	**0.9999**	**1**	**1**	**0.9999**

In terms of performance, XGBoost was able to learn better than the rest of the models for the training model. Similarly, XGBoost provides the best performance among the test models as well. Here XGBoost has accuracy 0.9999, recall 1.0 and roc_auc 0.9999. Moreover, XGBoost's sensitivity (1.0), specificity (0.9998), F1-score (0.9999) and precision (1.0) are better than all. Like the training model, the sensitivity (1.0) and recall (1.0) of the decision tree and the precision (1.0) of the random forest come into consideration with XGBoost.

8.1.6 Best model for Upsampling technique:

In terms of upsampling method, XGBoost provides good performance in both the training and testing of models. Whereas XGBoost's accuracy (0.9999), recall (1.0), and roc_auc (0.9999) are the best for the testing model.

Although the performance of the decision tree and the random forest is good, two models lag behind XGBoost. So XGBoost is best for upsampling technique as well.

8.1.7 Comparative Analysis for overall techniques:

Table 8.7: Outputs of all train models.

	Method	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
					F1-score	Precision	Recall	
LR	No sampling	0.9992	0.5959	0.9998	0.7123	0.89	0.60	0.9813
	Under-sampling	0.9548	0.9137	0.9944	0.9520	0.99	0.91	0.9864
	Upsampling	0.9474	0.9221	0.9727	0.9460	0.97	0.92	0.9849
DT	No sampling	0.9991	0.6848	0.9996	0.7198	0.76	0.68	0.9576
	Under-sampling	0.9308	0.8591	**1**	0.9242	**1**	0.86	0.9786
	Upsampling	0.9962	**1**	0.9925	0.9963	0.99	**1**	0.9997
RF	No sampling	0.9997	0.8252	0.9999	0.9014	0.99	0.83	0.9921
	Under-sampling	0.9858	0.9712	**1**	0.9854	**1**	0.97	0.9998
	Upsampling	0.9891	0.9788	0.9994	0.9890	**1**	0.98	0.9998
XGB	No sampling	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	Under-sampling	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	Upsampling	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Table 8.8: Outputs of all test models.

	Method	Accuracy	Sensitivity	Specificity	Classification report			Roc_Auc
					F1-score	Precision	Recall	
LR	No sampling	0.9990	0.5483	0.9998	0.6766	0.88	0.55	0.9830
	Under-sampling	0.9409	0.888	**1**	0.9406	**1**	0.89	0.9806
	Upsampling	0.9476	0.9224	0.9728	0.9462	0.97	0.92	0.9856
DT	No sampling	0.9990	0.6774	0.9995	0.7198	0.74	0.68	0.9391
	Under-sampling	0.8945	0.8	**1**	0.9242	**1**	0.80	0.9519
	Upsampling	0.9958	**1**	0.9917	0.9963	0.99	**1**	0.9995
RF	No sampling	0.9995	0.7903	0.9999	0.9014	0.93	0.79	0.9627
	Under-sampling	0.9282	0.872	0.9910	0.9854	0.99	0.87	0.9760
	Upsampling	0.9889	0.9785	0.9992	0.9890	**1**	0.98	0.9998
XGB	No sampling	0.9995	0.7983	0.9998	0.8571	0.93	0.80	0.9823
	Under-sampling	0.9282	0.888	0.9738	0.9288	0.97	0.89	0.9778
	Upsampling	**0.9999**	**1**	0.9998	**0.9999**	**1**	**1**	**0.9999**

In our project, we have created several models through source unbalanced dataset, upsampling and undersampling techniques using logistic regression, decision tree, Random forest, and XGBoost algorithms, from where I tried to find the best model considering the performance of the models. First, we check the performance of the models for the training dataset. In this case source unbalanced dataset, upsampling technique and undersampling technique for each XGBOOST learned best. XGBoost achieved 100% accuracy, recall, and roc_auc in each.

But our most important thing is to find the best model for the testing dataset. In this case, the performance of XGboost is better than all. XGBoost's best model is obtained by using the upsampling technique, where XGBoost's accuracy is higher than all 0.9999. Moreover F1-score (0.9999), precision (1.0), and recall (1.0) are more than all other models. XGBoost also achieved the highest roc_auc (0.9999) value in this case.

Using upsampling technique gives better performance in the decision tree and random forest. Where the accuracy of the decision tree is 0.9958 and roc_auc is 0.9995. The accuracy of random forest is 0.9889 and roc_auc is 0.9998.

Finally, considering the overall performance, the XGBoost model implemented by using the upsampling technique is the best in detecting credit card fraud.

8.1.8 Summary:

This chapter supports the assessment criteria phase of the CRISP-DM method. The outcome and performance of each model implemented in our project have been compared mathematically in this chapter, from which we have selected the best model. Each implemented model has performed very well due to the use of the upsampling technique. Here the XGBoost model shows the best performance.

Chapter 9: Conclusion

In this chapter, our selected best model, our thoughts on the project, and the plans to manage the project are discussed. Moreover, this chapter represents the deployment phase of the CRISP-DM method.

The main goal of our project was to create the best model to detect the maximum amount of credit card fraud. This project has been started by following the CRISP-DM method. In the literature review chapter, an understanding of tools, techniques, and technology is gained by reviewing fraud detection-related papers. A well-planned design of the project is done in the methodology chapter through the CRISP-DM method. Then the demand and inspection chapter covers all the topics for business understanding. A full model of the project was proposed in the modeling chapter. In the Data visualization and exploration chapter, all the tasks of data understanding and data preprocessing have been done.

The project is implemented in the implementation chapter and the best model is selected through the evaluation chapter.

9.1 Best model:

We have used four machine learning algorithms logistic regression, decision tree, random forest, and XGBoost to implement the model in the project. As our source dataset was unbalanced, the dataset was balanced by upsampling and downsampling techniques. Then using our source dataset, a new balanced dataset obtained through upsampling and undersampling, model four is implemented. Each model provided good performance here and the sampling techniques used were both effective. However, the XGBoost model implemented through the use of the upsampling technique can show the best performance.

So we chose the XGBoost model as the best model for the project.

9.2 Challenges:

The biggest challenge in creating fraud detection models is the use of large datasets. Every day millions of transactions are done all over the world. However, we used two-day transactions of European Union card holders from Kaggle. Moreover, the availability of such data is very low due to customer security issues. The dataset we used was unbalanced, so balancing it was a big challenge.

9.3 Future scope:

Our project is based on a small dataset which is not effective for global fraud detection. Later, by using large datasets and more realistic data, our models can be updated and used to detect fraud around the wor

REFERENCES

1. Delamaire, Linda, Abdou, Hussein and Pointon, John (2009) Credit card fraud

2. and detection techniques: a review. Banks and Bank Systems, 4 (2). pp. 57-68. ISSN 1816-7403.This version is available at http://eprints.hud.ac.uk/id/eprint/19069/

3. International Journal of Scientific Engineering and Technology (ISSN : 2277-1581) www.ijset.com, Volume No.1, Issue No.3, pg : 194-198 01 July 2012

4. Benchaji et al. Journal of Big Data (2021) 8:151 https://doi.org/10.1186/s40537-021-00541-8

5. Nandi AK, Randhawa KK, Chua HS, Seera M, Lim CP (2022) Credit card fraud detection using a hierarchical behavior-knowledge space model.PLoS ONE 17(1): e0260579. https://doi.org/10.1371/journal.pone.0260579

6. Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit card fraud detection using AdaBoost and majority voting. IEEE access, 6, 14277-14284.

7. Itoo, F., Meenakshi & Singh, S. Comparison and analysis of logistic regression, Naïve Bayes and KNN machine learning algorithms for credit card fraud detection. Int. j. inf. tecnol. 13, 1503–1511 (2021). https://doi.org/10.1007/s41870-020-00430-y

8. Chen, R. C., Chen, T. S., & Lin, C. C. (2006). A new binary support vector system for increasing detection rate of credit card fraud. International Journal of Pattern Recognition and Artificial Intelligence, 20(02), 227-239.

9. W. -F. Yu and N. Wang, "Research on Credit Card Fraud Detection Model Based on Distance Sum," 2009 International Joint Conference on Artificial Intelligence, 2009, pp. 353-356, doi: 10.1109/JCAI.2009.146.

10. V. Mareeswari and G. Gunasekaran, "Prevention of credit card fraud detection based on HSVM," 2016 International Conference on Information Communication and Embedded Systems (ICICES), 2016, pp. 1-4, doi: 10.1109/ICICES.2016.7518889.

11. M. Puh and L. Brkić, "Detecting Credit Card Fraud Using Selected Machine Learning Algorithms," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019, pp. 1250-1255, doi: 10.23919/MIPRO.2019.8757212.

12. Vengatesan, K., Kumar, A., Yuvraj, S., Kumar, V., & Sabnis, S. (2020). Credit card fraud detection using data analytic techniques. Advances in Mathematics: Scientific Journal, 9(3), 1185-1196.

13. Meng, C., Zhou, L., & Liu, B. (2020, August). A case study in credit fraud detection with SMOTE and XGboost. In Journal of Physics: Conference Series (Vol. 1601, No. 5, p. 052016). IOP Publishing.

14. Fayyomi, A. M., Eleyan, D., & Eleyan, A. A Survey Paper On Credit Card Fraud Detection Techniques.

15. Delamaire, L., Abdou, H., & Pointon, J. (2009). Credit card fraud and detection techniques: a review. Banks and Bank systems, 4(2), 57-68.

16. Vinaya, D. S., Basapur, S. B., Abhay, V., & Natesh, N. (2020). Credit Card Fraud Detection Systems (CCFDS) using Machine Learning (Apache Spark).

17. Ghosh, Sushmito, and Douglas L. Reilly. "Credit card fraud detection with a neural-network." *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*. Vol. 3. IEEE, 1994.

18. Srivastava, Abhinav, et al. "Credit card fraud detection using hidden Markov model." *IEEE Transactions on dependable and secure computing* 5.1 (2008): 37-48.

19. Halvaiee, Neda Soltani, and Mohammad Kazem Akbari. "A novel model for credit card fraud detection using Artificial Immune Systems." *Applied soft computing* 24 (2014): 40-49.

20. Gianini, Gabriele, et al. "Managing a pool of rules for credit card fraud detection by a Game Theory based approach." *Future Generation Computer Systems* 102 (2020): 549-561.

21. Olszewski, Dominik. "Fraud detection using self-organizing map visualizing the user profiles." *Knowledge-Based Systems* 70 (2014): 324-334.

22. Nilson report: https://nilsonreport.com/mention/1542/1link/

23. K. T. Hafiz, S. Aghili and P. Zavarsky, "The use of predictive analytics technology to detect credit card fraud in Canada," 2016 11th Iberian Conference on Information Systems and Technologies (CISTI), 2016, pp. 1-6, doi: 10.1109/CISTI.2016.7521522.

24. Different types of fraud- Delamaire, L., Abdou, H. and Pointon, J., 2009. Credit card fraud and detection techniques: a review. *Banks and Bank systems, 4*(2), pp.57-68.

25. https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

26. https://stats.stackexchange.com/questions/317073/explanation-of-min-child-weight-in-xgboost-algorithm

Appendix

Appendix A

UNIVERSITY OF PORTSMOUTH

Certificate of Ethics Review

Project title: Credit card fraud detection using python

Name:	Mehedi Hasanat Apu	**User ID**:	2083580	**Application date**:	06/06/2022 15:49:20	**ER Number**:	TETHIC-2022-103347

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Haythem Nakkas, David Williams**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Postgraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Hamidreza Khaleghzadeh**
Is the study likely to involve human subjects (observation) or participants?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No
Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: No
Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No

I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc)
I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs
I confirm that I will act ethically and honestly throughout this project

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor's signature: Khaleghzadeh Date: 06/06/2022

Appendix B

School of Computing
Postgraduate Programme

MSc in Data Analytics

Project Specification

Mehedi Hasanat Apu

Project Specification

1. Basic details

Student name:	Mehedi Hasanat Apu
Draft project title:	Credit Card Fraud Detection Using Python
Course and year:	Msc in Data Analytics (2021-2022)
Client organisation:	Unknown
Client contact name:	Unknown
Project supervisor:	Hamidreza Khaleghzadeh

2. Outline of the project environment

Suppose company XYZ.We proposed a MLAI based model for fiscal company or bank to descry malicious credit card conditioning in online fiscal deals assaying fake transaction manually is inoperable due to larger quantities of data and its complexity.Still, Adequately given a instructional features of labels in dataset, could make it's possible using evolving Machine Learning. This thesis will be explored in the design. To classify fraudulent and licit credit card sale by supervised literacy Algorithm similar as Random timber. To help us to get mindfulness about the fraudulent and without loss of any financially.

3. The problem to be solved

The end of this design is to prognosticate whether a credit card sale is fraudulent or not, grounded on the sale quantum, position and other sale related data. It aims to track down credit card scale of data, which is done by detecting anomalies in it, the large scale of data.Regarding unborn work, we motivate the operation of simple and transparent machine literacy styles for credit card fraud discovery and we sketch a simple stoner- concentrated modeling approach.

4. Breakdown of tasks

The huge number of deals makes it difficult for a mortal critic to descry fraudulent transaction patterns. In real world massive scenarios deals were scrutinized by automated tools transaction, Anatomized by classifiers who in turn find and warn the suspicious bone cautions are also audited by professional investigators.However, also the massive deals can not be vindicated by professional investigators due to his negligence, time and cost factor challenges, If we put fraud-lent transaction

discovery performance as research consideration..So, In related research situation, we need to guarantee a high accuracy and delicacy of the system.

Thirdly, Veritably grueling a well designed strategy to descry fraudulent transaction patterns since the data scarcely available due to non-public issues and mostly they were in-equilibrium records of transactions through our system.Chancing the right measures and strategies is a pivotal asset to assiduity for further enhancement of fraud discovery.

Which credit card fraud discovery concept has been using now?

Most of the time a multiple techniques used for fraud discovery and done in multiple situations, not just one for illustration.Payment gate way will get a list of fraud cards(a separate train having these figures streamlined daily) which will be used to check if the card is blacklisted.

Bank issuer will have medium/ systems to identify the abnormal exertion of the card, like swiping the card in transnational position other than the country of the card holder(alert will be entered and the client service cautions the client)

5. Bank authority will have medium to large scale systems to warn deals within reinforced period of time like(2 – 3 deals with in blink of interval)
6. Many banks also get alert in case of a quantum scale of data regarding transactions.
7. These are many fraud discovery ways.

5.Project deliverables

a. Project Documentation and Deliverable
b. A complete SRS of our imposed System
c. Requirement Document
d. Project PPT
e. Project Report
f. User Manual
g. Code Implementation
h. Project Objectives
i. Project Constrain
j. Stakeholders Internal & External
k. In-scope activities of credit card
l. Out of project scope activities and interest
m. Project Assumptions & Dependencies
n. Beneficiaries in project

o. Risk Analysis

p. Budget high-level report

6. Requirements

Requirement Overview

Nowadays everyone use the credit card for payment across the world.Online and in store purchases or payments, due to actuality of wide point of trade(POS). The reason behind the fraud is negligence of stoner. When details were exposed to other person than he steal the most important information about credit card and stoner details fluently fraud can be achieved. To descry what type of fraud do during sale, we need to face Several challenges.Costing that among all the deals is passed and which one is real could be a task. Among the standard and veritably common ways of making payment encyclopedia,especially in South America, because of the presence of a far reaching point of trade.

The large number of individuals around the horizon use charge cards to buy products and services by getting a credit for half of month. Any helpful frame could be mishandled and charge card is no impunity to this. Alongside the ascent of charge card use, highway robbery is on the ascent. Monetary Institutions(FIs) endure meliorated fake exercises and bear a large number of bone mischances every time. In light of statistics frauds regard to further than$1.5 billion every time for Visa and MasterCard around the world. Credit card companies and their part banks essay to discover better approaches to avert swindles. A portion of the preventative measures on the cards are glamorous stripes, 3D badges, and CVC.

Credit card companies are likewise taking way to have alternate for credit cards as Smart Cards, be that as it may, in view of assessments this negotiation will be over the top precious because of the broad POS network in USA and the gigantic no. of cards available for use in those premises. FIS also exercising an multifariousness of calculating medium, similar as Neural Networks(NNs), to follow and distinguish dubious exchanges and ban them for fresh examination.

PROBLEM STATEMENT

Not all the doubtful transactions or deals consider as fraudulent. It's generally called as false positive(FP) which means that the case wasn't fraud although it was flagged as being potentially fiddle

. This process of affirming each sale those outliers from the cardholder's normal routine brings mistrustfulness about possible customer disappointment. also, the charges related with exploring an enormous numbers of false cons are high.

Motivation

As of now, a considerable quantum of time is given for examining innumerous genuine cases(FPs). On the off chance that the volume of examination on FPs could be dropped down, fiddle judges can invest further energy and time in genuine fraud deals that restricts the losses to the FIs.

SYSTEM ANALYSIS

ProposedSystem:

The crucial ideal of current exploration is clapping the procedure of particular follow up on a large number of suspicious deals and to discover a path to preprocess the flagged records to fete the probable genuine entries from the list of genuine/ falsified entries. Then, the volume of dispensable analysis is dropped leading to significant savings for the fiscal institutions.The current FDS threshold also be lowered than a number of fraudulent cases, being missed under this position, Can be detected.

As a result, the fraud is discovered before and the overall losses may be reduced. For addressing these challenges, outlier discovery and GBT Classifier is used, i.e. among the veritably common used operations of Machine Learning for addressing the pattern recognition and bracket problems. The results indicate that the habituated system has a veritably good possibility to extemporize the present system.

Advantages

The proposed system overcomes the low delicacy cast problem.

Exercising rearmost AI styles, the fraudulent deals are honored and the false cautions are reduced.

Fast and dependable result is attained.

Functional Conditions

Functional conditions are the characteristics of the product. All the features anticipated from any development are mentions as functional conditions.

Non-Functional Conditions

Non-Functional conditions list out the customer prospects from product design, security, availability, and trustability or performance standpoint.

Performance related condition:

Its about the software capability to respond on druggies ' action, similarly as upon running the operation, it should not take more than 2's seconds.

Data confirmation should not take more over 5 seconds.

Result generation should be achieved within 5 seconds.

Design Constraints- The design is to be developed in python which should get executed in a Windows OS.

Norms of Compliance- There should be uniformity while defining variable names. The GUI shall have a poleasent look and feel. The graphical stoner interface should be stoner friendly.

Trustability- The product shouldn't fail in medial of any operations carrying out.

Vacuity- software can be used anytime.

Security- Security is veritably important for any operation that holds stoner sensitive data.

Portability- The design should be executable on an Operating System.

Maintainability- The software admin should be suitable to manage the data.

HardwareRequirements:

Processor : Pentium i3 or higher. RAM : 4 GB or higher. Hard Disk Drive : 20 GB (free). Peripheral Devices : Monitor, Mouse and Keyboard.

Soft Requirements:

An operating system : Windows 8.1/10/11.

Major Coding Language : Python 3.6

An IDE Tool : Python jupyter3 notebook

Our API's : Data structures Numpy, Pandas,PySpark 2.0, Matplotlib3.0

Packages: Which are being used for data exploration or illusion, pre- processing and for using random forest algorithm in our project were:

NumPy: For simple operations on array type structures.

Pandas: For reading and updating the csv file or dataset.

SciKit Library: Learn- for pre-processing of data in our project.

Matplotlib/Seaborn: For plotting and illustrated confusion matrix in colour format.

Tensor flow: A matrix formatting and tracing of machine.

Stake holder characteristics

User/Customer The person who willing to use credit card for buy purpose

Authorization services will validates the current credit card transactions must be ensured card number was validated and the card has passed its expiry..

Deposit processing fee to be applied on deposit payment through card

Prepare card transaction log reports and receipts that show authorization unique codes table, logs, amounts, messages and errors.

Stake holder constraints

a. Trusted if using a well known third-party processor or intermediaries
b. Must be suited for large-volume of data
c. A low transaction cost as compared to market
d. Money transferred speed must be robust
e. Must be provided credit false transaction preventive measures.

The pivotal ideal of current disquisition is clapping the procedure of particular follow up on a large number of suspicious deals and to discover a path to preprocess the flagged records to recognize the probable genuine entries from the list of genuine/ falsified entries.The volume of gratuitous analysis dropped leading to significant savings by financial institutions.The currently FDS given threshold can lowered and a number of fraudulent cases go strike up, being missed under this position, can be detected. As a result, the fraud is discovered before and the overall losses may be reduced. For addressing these challenges, outlier discovery and GBT Classifier is used, i.e. among the truly common used operations of Machine Learning for addressing the pattern recognition and type problems. The results indicate that the accustomed system has a truly good possibility to improvise the present system.

Advantages

The proposed system overcomes the low delicacy cast problem. Exercising most of the time MLAI like, the fraudulent deals are recognized and the false cautions are reduced. Fast and the reliable results attained.

Functional Conditions

Functional conditions are the characteristics of the product. All the features anticipated from any development are mentions as functional conditions.

Non-Functional Conditions

Non-Functional conditions list out the client prospects from product design, security, vacuity, and responsibility or performance viewpoint.

A performance measures

Performance Conditions tells us about the software capability to respond on malicious action analogous as Upon running the operation, it should n't take further than 3 seconds.

Data evidence should n't take over 3 seconds.

Result generation would be acquired within 3 seconds.

Design Constraints- The design is to be developed in python which should get executed in a Windows OS.Python editor py-charm used as an integrated environment during the development phase.

Moral Compliance- There should be uniformity while defining variable names. The GUI shall have a poleasent look and feel. The graphical user interface should be stoner friendly.

A responsibility- product should not fail in medium of operation executions.

The Vacuity- The software can be used anytime.

Security- Security is truly important for any operation that holds user sensitive data.

The maintainability- The software admin should be suitable to manage the data.

Portability- The design should be executable on an Operating System.

7. Legal, ethical, professional, social issues

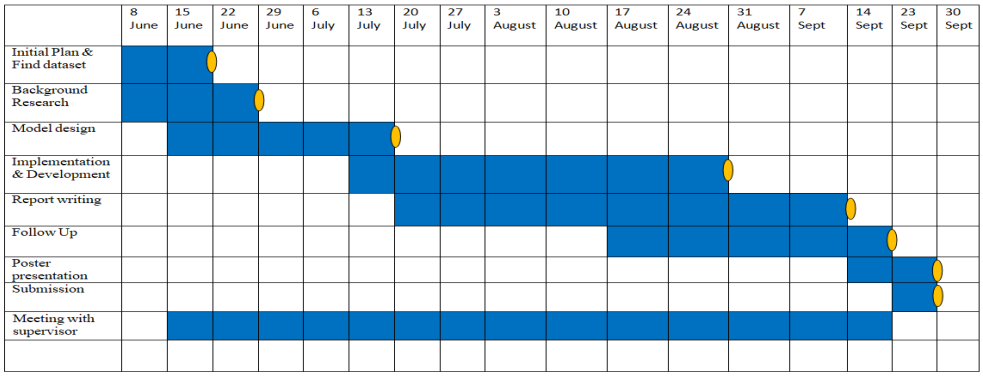
Utmost ethical and legal issues arises in our system research and development phase, In the area of existence's right to sequestration versus the lesser good of a larger reality themultinational companiesand high business shop keepers. For illustration, tracking how druggies use credit card, crowd surveillance, managing client biographies, tracking a person's

trip with passport and so on. A crucial conception in resolving this issues is to find out, what's a person's anticipation of sequestration.This deals with the right of an individual to control particular information. It's the protection of particular or sensitive informationsequestration is private. Different people have different ideas of what sequestration is and how important sequestration they will trade for safety or convenience, delicacy talks about the responsibility for the authenticity, dedication an delicacy of the information.Property determines who the proprietor of the information is and who controls access,and availability of it,To dealt with that issue of information exposure an association has the right to collect it.The expert data gather or scientist resolve by properly managed integrity and privacy compliance.

8. Facilities and resources

we use labs and online resources like Wikipedia and many books related to that genre of project. Human resource, books and case studies and banking journals regrading the credit card frauds.

9. Project plan



10.Supervision meetings

I discussed this with my supervisor. I will try to have weekly meetings with my supervisor. It can be offline or online, depending on the convenience of my supervisor.

Moreover, whenever I face any problem, I will try to have an urgent meeting with my supervisor.

Moreover, I will always try to inform my supervisor about the updates of my project

11.Project mode

If there are two possibilities for your project mode, after negotiation, please record your planned duration and submission date. It is also helpful to record your initial registration mode (i.e. are you a full time or a part time student). Remember, the exact dates will be announced through Moodle – these represent a generic guideline.

Registration mode	Full Time	
Project mode	Full Time	
Planned submission deadline	20/9/22	

Signatures

	Signature:	Date:
Student	Mehedi Hasanat Apu	07/06/22
Client		
Project supervisor		07/06/22