A MAJOR PROJECT REPORT

ON

A SUPERVISED MACHINE LEARNING ALGORITHM FOR DETECTING FRAUDS IN BANK PAYMENTS

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

By

K.ANJANA	**(20281A0452)**
A.SHIVA KUMAR	**(21285A0412)**
A.SUSMITHA	**(20281A0436)**
S.SAITEJA	**(20281A0444)**
S.RATHAN	**(20281A0433)**

Under the guidance of

U.RAJAIAH

Assistant Professor

KAMALA INSTITUTE OF TECHNOLOGY & SCIENCE

(Sponsored by Kamala Education Society, Approved by AICTE, New Delhi, Affiliated to JNTUH, accredited by NAAC with ‘A++’ and accredited by NBA (ECE, EEE, CSE))

SINGAPUR, HUZURABAD, KARIMNAGAR, TELANGANA, INDIA-505468

2023-2024

A MAJOR PROJECT REPORT

ON

A SUPERVISED MACHINE LEARNING ALGORITHM FOR DETECTING FRAUDS IN BANK PAYMENTS

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

K.ANJANA	**(20281A0452)**
A.SHIVAKUMAR	**(21285A0412)**
A.SUSMITHA	**(20281A0436)**
S.SAITEJA	**(20281A0444)**
S.RATHAN	**(20281A0433)**

Under the guidance of

U.RAJAIAH

Assistant Professor

KAMALA INSTITUTE OF TECHNOLOGY & SCIENCE

(Sponsored by Kamala Education Society, Approved by AICTE, New Delhi, Affiliated to JNTUH, accredited by NAAC with ‘A++’ and accredited by NBA (ECE, EEE, CSE))

SINGAPUR, HUZURABAD, KARIMNAGAR, TELANGANA, INDIA-505468

2023-2024

KAMALA INSTITUTE OF TECHNOLOGY & SCIENCE

(Sponsored by Kamala Education Society, Approved by AICTE, New Delhi, Affiliated to JNTUH, accredited by NAAC with 'A++' and accredited by NBA (ECE, EEE, CSE))

SINGAPUR, HUZURABAD, KARIMNAGAR, TELANGANA, INDIA-505468

2023-2024

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the Project Stage entitled " **A SUPERVISED MACHINE LEARNING ALGORITHM FOR DETECTING FRAUDS IN BANK PAYMENT**" is a work carried out by **K. ANJANA (20281A0452), A.SHIVAKUMAR (21285A0412),A.SUSMITHA (20281A0436), S.SAITEJA (20281A0444) , S.RATHAN (20281A0433)** submitted in partial fulfillments of the requirements for the award of the degree in **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING** under Jawaharlal Nehru Technological University, Hyderabad during the academic year **2023-2024**.

Project Guide	Head of the Department	Principal
U.RAJAIAH	**Dr.B.RAMESH**	**Dr.K.ESHWARAIAH**
Assistant Professor	Professor	Professor

ACKNOWLEDGEMENT

We express our profound gratitude and deep regard to all the people who extended their support with timely suggestions and indispensable help, which made this project feasible.

We sincerely thank **Mr. U.RAJAIAH**, Assistant Professor, for assisting and monitoring as the project guide and project coordinator. With his constant encouragement, it would not have been possible to complete this work. We express unfeigned gratitude towards our project coordinator for his valuable guidance in the completion of our project.

Our special thanks to **Dr. B. RAMESH**, Professor & Head, Department of Electronics and Communication Engineering, for their suggestions and wholehearted support.

We are extremely grateful to **Dr. K.ESHWARAIAH**, principal, Kamala Institute of Technology and Science, Singapur, for providing the requisite facilities during the project work.

Finally, we would like to extend our sincere thanks to all teaching staff members of the ECE Department for their kind cooperation and valuable help towards the project in the form of encouragement and moral support.

K.ANJANA	**(20281A0452)**
A.SHIVA KUMAR	**(21285A0412)**
A.SUSMITHA	**(20281A0436)**
S.SAITEJA	**(20281A0444)**
S.RATHAN	**(20281A0433)**

ABSTRACT

As technology advanced and e-commerce services expanded, credit cards became one of the most popular payment methods, resulting in an increase in the volume of banking transactions. Furthermore, the significant increase in fraud requires high banking transaction costs. As a result, detecting fraudulent activities has become a fascinating topic. In this study, we consider the use of class weight-tuning hyperparameters to control the weight of fraudulent and legitimate transactions. We use Bayesian optimization in particular to optimize the hyperparameters while preserving practical issues such as unbalanced data. We propose weight-tuning as a pre-process for unbalanced data, as well as CatBoost and XGBoost to improve the performance of the LightGBM method by accounting for the voting mechanism.

Finally, in order to improve performance even further, we use deep learning to fine-tune the hyperparameters, particularly our proposed weight-tuning one. We perform some experiments on real-world data to test the proposed methods. To better cover unbalanced datasets, we use recall-precision metrics in addition to the standard ROC-AUC. CatBoost, LightGBM, and XGBoost are evaluated separately using a 5-fold cross-validation method. Furthermore, the majority voting ensemble learning method is used to assess the performance of the combined algorithms. LightGBM and XGBoost achieve the best level criteria of ROC-AUC = 0.95, precision 0.79, recall 0.80, F1 score 0.79, and MCC 0.79, according to the results. By using deep learning and the Bayesian optimization method to tune the hyperparameters, we also meet the ROC-AUC = 0.94, precision = 0.80, recall = 0.82, F1 score = 0.81, and MCC = 0.81. This is a significant improvement over the cutting-edge methods we compared it to.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER-1

INTRODUCTION

1.1 Problem defination

The study aims to address the escalating challenge of detecting fraudulent activities amidst the increasing volume of banking transactions, propelled by the advancement of technology and the expansion of e-commerce services. With credit cards emerging as one of the most preferred payment methods, the surge in banking transactions has been accompanied by a notable increase in fraudulent incidents, imposing substantial costs on financial institutions. Detecting and mitigating fraudulent activities have thus become imperative, sparking considerable interest in the field. This study focuses on leveraging class weight-tuning hyperparameters to regulate the balance between fraudulent and legitimate transactions. Bayesian optimization is employed to fine-tune these hyperparameters while tackling practical issues such as unbalanced data. The proposed approach entails weight-tuning as a preprocessing step for handling unbalanced data, alongside the integration of CatBoost and XGBoost algorithms to enhance the performance of the LightGBM method through a voting mechanism.

The proliferation of technology and the expansion of e-commerce services have revolutionized the way transactions are conducted in the banking sector. Credit cards have emerged as one of the most popular payment methods, leading to a significant increase in the volume of banking transactions. However, this surge in transaction volume has also been accompanied by a corresponding rise in fraudulent activities. Fraudulent incidents not only pose financial risks but also incur high transaction costs for financial institutions. Consequently, detecting and mitigating fraudulent activities have become crucial endeavors for banking institutions worldwide.

In light of the growing importance of fraud detection, this study aims to propose an innovative approach that leverages advanced techniques to enhance the effectiveness of fraud detection systems. The key focus of this approach is to address the challenge of imbalanced data, where fraudulent transactions are typically much less frequent than legitimate transactions. Class weight-tuning hyperparameters are utilized to adjust the weight assigned to fraudulent and legitimate transactions, thereby addressing the imbalance issue and improving the performance of fraud detection algorithms.

Bayesian optimization is employed as a powerful technique to optimize these weight-tuning hyperparameters while taking into account practical considerations such as unbalanced data. By iteratively exploring the hyperparameter space and evaluating the performance of the fraud detection system, Bayesian optimization aims to find the optimal configuration that maximizes the system's effectiveness in detecting fraudulent activities. Furthermore, the proposed approach integrates advanced machine learning algorithms such as CatBoost and XGBoost, alongside the LightGBM method, to further enhance the performance of the fraud detection system. These algorithms are renowned for their ability to handle complex datasets and extract meaningful patterns from data. By employing a voting mechanism that combines the outputs of multiple algorithms, the proposed approach aims to leverage the strengths of each algorithm while mitigating their individual weaknesses, resulting in a more robust and effective fraud detection system.

Overall, this study seeks to contribute to the ongoing efforts in enhancing fraud detection capabilities in the banking sector. By leveraging advanced techniques such as class weight-tuning hyperparameters, Bayesian optimization, and ensemble learning, the proposed approach aims to provide financial institutions with more effective tools to combat fraudulent activities and safeguard their assets. Through rigorous experimentation and evaluation, this study endeavors to demonstrate the efficacy of the proposed approach and its potential to significantly improve the accuracy and efficiency of fraud detection systems in real-world scenarios.

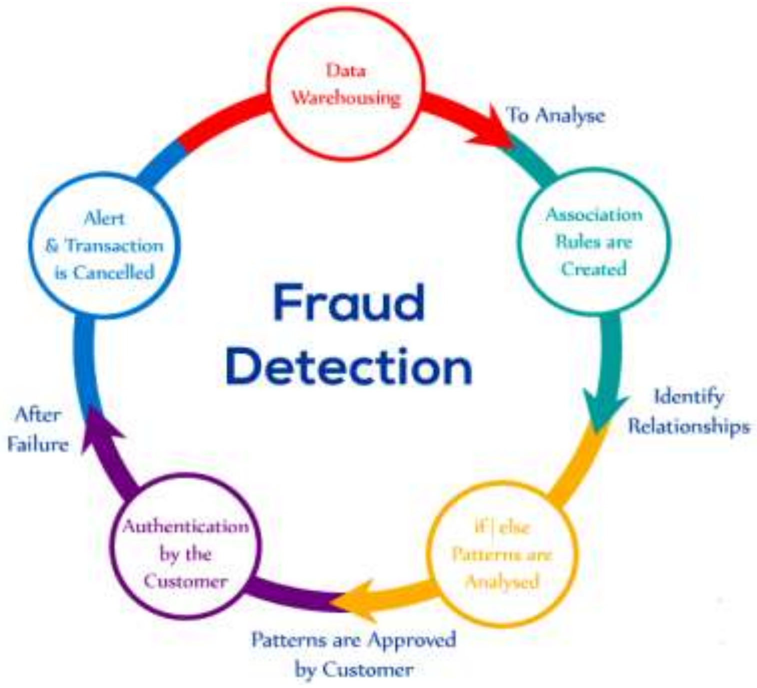

Data Warehousing
To Analyse
Association Rules are Created
Identify Relationships
if | else Patterns are Analysed
Patterns are Approved by Customer
Authentication by the Customer
After Failure
Alert & Transaction is Cancelled
Fraud Detection

Fig 1.1: Credit card fraud detection using Machine learning

1.2 Motivation

In recent years, the surge in financial transactions, driven by the expansion of financial institutions and the popularity of web-based e-commerce, has heightened the challenge of fraudulent activities, particularly in online banking. Fraudsters continually update their tactics to mimic legitimate transactions, complicating fraud detection efforts. To combat this, researchers are continuously striving to enhance existing methods or develop new ones. Fraudsters exploit security weaknesses in commercial applications, but technology can also serve as a potent tool against fraud. Detecting fraud promptly after occurrence is crucial for prevention. Credit card fraud, involving illegal use of credit card information for purchases, occurs both in physical and digital transactions, with the latter being susceptible to fraud over the phone or web. In this paper, an efficient approach for detecting credit card fraud is proposed and evaluated using publicly available datasets. The approach employs optimized algorithms like LightGBM, XGBoost, CatBoost, logistic regression, and deep learning, individually and in combined methods. Bayesian optimization is utilized for fraud detection, addressing the challenge of unbalanced data. Weight-tuning hyperparameters are suggested as a pre-processing step. The XGBoost algorithm is chosen for its speed and regularization capabilities, while CatBoost is preferred for its effectiveness without extensive hyperparameter tuning.

Fig 1.2 : Analysis on Financial fraud detection

A majority-voting ensemble learning approach is proposed, combining CatBoost, XGBoost, and LightGBM, alongside deep learning for hyperparameter fine-tuning. Performance evaluation is conducted using recall-precision, ROC-AUC, F1_score, and MCC metrics on real-world data. The proposed methods outperform existing approaches, demonstrating superior effectiveness in fraud detection. Source codes are made publicly accessible for further research. The paper's structure includes a review of related state-of-the-art in Section II, presentation of the proposed approach in Section III covering dataset, pre-processing, algorithms, and evaluation metrics, evaluation results in Section IV, and conclusion in Section V. This comprehensive approach and evaluation contribute significantly to the ongoing efforts in fraud detection, offering a robust framework for combating fraudulent activities in financial transactions.

1.3 Objectives

This project proposes an advanced approach to detect credit card fraud, focusing on the increasing volume of financial transactions, especially in online banking. Leveraging optimized algorithms such as LightGBM, XGBoost, CatBoost, logistic regression, and deep learning, the study aims to enhance fraud detection capabilities significantly.

Addressing Evolving Fraud Tactics

1. Integrating advanced technology like AI and data analytics for real-time fraud detection.
2. Enhancing security measures within commercial applications to prevent unauthorized access and data breaches.

Utilizing Technology for Security

- ➢ Leveraging AI and data analytics for fraud detection.
- ➢ Strengthening security measures in commercial applications to mitigate vulnerabilities.

Prompt Fraud Detection

- ➢ Swift fraud detection minimizes financial losses and prevents prolonged exposure to fraudulent activities, safeguarding both institutions and customers.
- ➢ Timely detection enables proactive measures, reducing the impact of fraudulent transactions and preserving financial integrity.

Focus on Credit Card Fraud

- ➢ Credit card fraud is scrutinized due to its substantial risks in physical and online transactions, requiring tailored detection strategies.
- ➢ The study highlights the heightened vulnerability of online transactions to credit card fraud, necessitating specialized preventive measures.
- ➢ The study conducts a thorough analysis of credit card fraud risks, emphasizing vulnerabilities in both physical and digital transactions.
- ➢ With a focus on online transactions, the study highlights the increased susceptibility of credit card fraud, necessitating targeted prevention strategies.

Employing Bayesian Optimization

- Bayesian optimization and hyperparameter tuning address data imbalance, improving fraud detection algorithm performance.
- These techniques optimize model parameters to better discern fraudulent from legitimate transactions, enhancing overall effectiveness.

Majority-Voting Ensemble Learning

- A majority-voting ensemble learning approach is proposed to integrate multiple algorithms and deep learning for hyperparameter fine-tuning, aiming for superior performance in fraud detection.

CHAPTER-2

OVERVIEW OF THE PROJECT

2.1 Literature Survey

2.1.1: Credit Card Fraud Detection Using A New Hybrid Machine Learning Architecture

The negative effect of financial crimes on financial institutions has grown dramatically over the years. To detect crimes such as credit card fraud, several single and hybrid machine learning approaches have been used. However, these approaches have significant limitations as no further investigation on different hybrid algorithms for a given dataset were studied. This research proposes and investigates seven hybrid machine learning models to detect fraudulent activities with a real word dataset. The developed hybrid models consisted of two phases, state-of-the-art machine learning algorithms were used first to detect credit card fraud, then, hybrid methods were constructed based on the best single algorithm from the first phase. Our findings indicated that the hybrid model Adaboost + LGBM is the champion model as it displayed the highest performance. Future studies should focus on studying different types of hybridization and algorithms in the credit card domain.

2.1.2:Ecommerce Fraud Detection Through Fraud Islands And Multi-Layer Machine Learning Model

Main challenge for e-commerce transaction fraud prevention is that fraud patterns are rather dynamic and diverse. This paper introduces two innovative methods, fraud islands (link analysis) and multi-layer machine learning model, which can effectively tackle the challenge of detecting diverse fraud patterns. Fraud Islands are formed using link analysis to investigate the relationships between different fraudulent entities and to uncover the hidden complex fraud patterns through the formed network. Multi-layer model is used to deal with the largely diverse nature of fraud patterns. Currently, the fraud labels are determined through different channels which are banks' declination decision, manual review agents' rejection decisions, banks' fraud alert and customers' chargeback requests. It can be reasonably assumed that different fraud patterns could be caught though different fraud risk prevention forces (i.e. bank, manual review team and fraud machine learning model). The experiments showed that by integrating few different machine learning models which were trained using different types of fraud labels, the accuracy of fraud decisions can be significantly improved.

2.2 EXISTING MODEL:

In order to prevent fraudulent transactions and detect credit card fraud, several methods have been proposed by researchers. A review of state-of-the-art related works is presented in the following.

Halvaiee & Akbari study a new model called the AIS-based fraud detection model (AFDM). They use the Immune System Inspired Algorithm (AIRS) to improve fraud detection accuracy. The presented results of their paper show that their proposed AFDM improves accuracy by up to 25%, reduces costs by up to 85%, and reduces system response time by up to 40% compared to basic algorithms.

Bahnsen etal. developed a transaction aggregation strategy and created a new set of features based on the periodic behaviour analysis of the transaction time by using the von Mises distribution. In addition, they propose a new cost-based criterion for evaluating credit card thetransaction aggregation strategy to create new offers based on ananalysis of the periodic behaviour of transactions.

Randhawa et al. study the application of machine learning algorithms to detect fraud in credit cards. They first use Naive Bayes, stochastic forest and decision trees, neural networks, linear regression (LR), and logistic regression, as well as support vector machine standard models, to evaluate the available datasets. Further, they propose a hybrid method by applying AdaBoost and majority voting. In addition, they add noise to the data samples for robustness evaluation. They perform experiments on publicly available datasets and show that majority voting is effective in detecting credit card fraud cases.

Porwal and Mukund suggest a clustering-based method to detect outliers resilient to shifting patterns. They assume consistent spatial signatures for good user behavior. By tracking changes, they identify fraudulent behavior. They advocate for the precision-recall curve over ROC for evaluation, demonstrating its superiority in capturing model performance nuances.

The authors in, propose a group learning framework based on partitioning and clustering of the training set. Their proposed framework has two goals: 1) to ensure the integrity of the sample features, and 2) to solve the high imbalance of the dataset. The main feature of their proposed framework is that every base estimator can be trained in parallel, which improves the effectiveness of their framework.

Itoo et al. address data imbalance in fraud detection, employing logistic regression, Naive Bayes, and K-nearest neighbor with varied dataset ratios and oversampling. Their evaluation, considering accuracy, sensitivity, specificity, precision, F1-score, and AUC, highlights logistic regression's superior performance among the algorithms.

The authors in propose a framework that combines the potential of meta-learning ensemble techniques and a costsensitive learning paradigm for fraud detection. They perform some evaluations, and the results obtained from classifying unseen data show that the cost-sensitive ensemble classifier has acceptable AUC value and is efficient as compared to the performances of ordinary ensemble classifiers.

Altyeb et al. introduce a Bayesian-based hyperparameter optimization method for fraud detection in credit card transactions, applied to tune LightGBM parameters. Their study, conducted on public datasets, assesses accuracy, ROC-AUC, precision, and F1-score to evaluate the proposed approach's effectiveness.

Xiong et al. propose a learning-based approach to tackle the fraud detection problem. They use feature engineering techniques to boost the proposed model's performance. The model is trained and evaluated on the IEEE-CIS fraud dataset. Their experiments show that the model outperforms traditional machine-learning-based methods like Bayes and SVM on the used dataset .

Viram et al. evaluate the performance of Naive Bayes and voting classifier algorithms. They demonstrate that in terms of evaluated metrics, particularly accuracy, the voting classifier outperforms the Naive Bayes algorithm.

Verma and Tyagi investigate machine learning algorithms in order to determine the best supervised ML-based algorithm for credit card fraud detection in the presence of an imbalanced dataset. They evaluate five classification techniques and show that the supervised vector classifier and logistic.

TABLE 1: Features of the credit-card fraud dataset that is used in this model.

Variable Name	**Description**	**Type**
$V_1, V_2, ..., V_{28}$	Transaction feature after PCA transformation	Integer
Time	Seconds elapsed between each transaction with the first transaction	Integer
Amount	Transaction Value	Integer
Class	Legitimate or Fraudlent	0 or 1

2.3 PROPESED SYSTEM:

We start by applying the necessary preprocessing steps to the data for our proposed system. Then, we divide the data into two sections: one for training and the other for testing purposes. Following this, we employ Bayesian optimization on the training data to determine the optimal hyperparameters that enhance the system's performance. To ensure robust evaluation, we utilize the cross-validation method, which is especially crucial in dealing with unbalanced datasets. Subsequently, we assess the algorithms using a range of evaluation metrics including accuracy, precision, recall, the Matthews correlation coefficient (MCC), the F1-score, and AUC diagrams. This comprehensive evaluation framework allows us to thoroughly analyze and compare the performance of the algorithms under consideration.

Fig 2.1 :Fraud Detection Analysis

Optimized Performance:

By employing Bayesian optimization to determine optimal hyperparameters, the project enhances the performance of fraud detection algorithms. This optimization ensures that the system operates at its peak efficiency, resulting in more accurate and reliable detection of fraudulent activities.

Robust Evaluation Methodology:

Utilizing cross-validation for evaluation ensures the robustness of the results, particularly when dealing with unbalanced datasets. This methodology allows for a thorough assessment of algorithm performance across various metrics, enabling researchers to make informed decisions and comparisons to identify the most effective fraud detection approaches.

Comprehensive Analysis:

Comprehensive Analysis: Assessing algorithms using a range of evaluation metrics, including accuracy, precision, recall, MCC, F1-score, and AUC, allows for a comprehensive analysis of their performance. This enables researchers to gain insights into the strengths and weaknesses of each algorithm.

Optimized Resource Utilization:

Dividing the data into distinct training and testing sets optimizes resource usage by providing sufficient data for model training while preserving a separate dataset for evaluation. This approach ensures that the model's performance is assessed accurately without compromising the integrity of the training process.

Informed Decision-Making:

Thoroughly analyzing and comparing algorithm performance empowers informed decision-making in selecting the most effective fraud detection approach. This equips financial institutions with reliable tools to combat fraudulent activities, enhancing their ability to safeguard assets and maintain the trust of their customers.

2.3.1: Applications

- ➢ **Credit Card Fraud Detection:** Machine learning algorithms play a pivotal role in credit card fraud detection by analyzing transaction patterns, histories, and user behavior. By identifying anomalies indicative of fraudulent activities, these algorithms help prevent financial losses for banks and customers alike. Through real-time monitoring and analysis, machine learning enhances security measures and safeguards against unauthorized transactions, ensuring the integrity of credit card transactions.
- ➢ **Transaction Monitoring:** Machine learning models provide continuous monitoring of banking transactions, detecting anomalies like unusual amounts, frequencies, or locations. This real-time analysis aids in the prevention of fraudulent transactions by flagging suspicious activities promptly, thereby enhancing security measures and safeguarding against financial losses for both banks and customers.
- ➢ **Identity Theft Detection:** ML algorithms analyze user authentication patterns, biometric data, and historical transaction records to detect instances of identity theft or account takeover, enabling banks to intervene and prevent unauthorized access.
- ➢ **Loan Application Fraud Detection:** Machine learning is utilized to assess the risk associated with loan applications by analyzing various factors such as credit history, income, and financial behavior, identifying fraudulent applications and minimizing default risks.

CHAPTER-3

SOFTWARE ENVIRONMENT

3.1 Introduction to Python

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia

3.1.1 Advantages of Python

1. Extensive Libraries:

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible:

Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable:

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity:

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities:

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy:

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable:

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

8. Object-Oriented:

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source:

Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable:

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted:

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

3.1.2 Advantages of Python over other languages

1. Less Coding:

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable:

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone:

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

3.1.4 History of Python

Python's journey began in the late 1980s when Guido van Rossum, a Dutch programmer, set out to create a language that prioritized simplicity and readability. Drawing inspiration from languages like ABC and Modula-3, Python aimed to be a versatile tool suitable for various programming tasks. Its first version, Python 0.9.0, was released in February 1991, featuring fundamental elements such as classes with inheritance, exception handling, and functions. As Python matured, its user base grew, and in 1994, Python 1.0 marked a significant milestone with enhancements like lambda functions, map, filter, and reduce functions. Throughout the 1990s and early 2000s, Python steadily gained traction, attracting developers with its clean syntax, dynamic typing, and extensive standard library. The Python 2.x series, initiated in the early 2000s, solidified Python's position in the programming landscape. It introduced features such as list comprehensions, generators, and decorators, further enriching the language. Python's flexibility and ease of learning made it a popular choice for a wide range of applications, from scripting to web development to scientific computing. However, as Python evolved, it faced challenges related to backward compatibility and inconsistencies in the language design. This led to the development of Python 3.0, released in December 2008, which aimed to address these issues by introducing significant changes, including improved Unicode support, a cleaner I/O system, and syntax refinements. While the transition from Python 2.x to Python 3.x posed some initial hurdles for developers due to compatibility issues, the community rallied behind the transition, recognizing the long-term benefits of a cleaner and more efficient language.

Despite the transition challenges, Python's popularity continued to soar. Its simplicity, versatility, and extensive ecosystem of libraries and frameworks contributed to its widespread adoption in various domains, including web development, data analysis, machine learning, and automation. The formation of the Python Software Foundation (PSF) in 2001 provided organizational support for Python's development, ensuring its continued growth and maintenance. The PSF oversees the development of Python, manages its intellectual property, and supports the global Python community through initiatives such as conferences, grants, and educational programs. Today, Python stands as one of the most popular programming languages globally, embodying Guido van Rossum's vision of a language that is both powerful and easy to use. Its rich history, vibrant community, and robust ecosystem make Python a good to choose for developers across industries and skill levels.

3.2 Introduction to Machine Learning

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

3.2.1 Categories Of Machine Leaning

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more

succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

3.2.2 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

3.2.3 Challenges in Machines Learning

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

3.2.4 Applications of Machines Learning

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world application s of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention

3.3 Introduction to XGBoost Algorithm in Machine Learning

Ever since its introduction in 2014, XGBoost has been lauded as the holy grail of machine learning hackathons and competitions. From predicting ad click-through rates to classifying high energy physics events, XGBoost has proved its mettle in terms of performance – and speed. XGBoost is the first algorithm of choice for many in ML hackathon. The accuracy it consistently gives, and the time it saves, demonstrates how useful it is. But how does it actually work? What kind of mathematics power XGBoost? We'll figure out the answers to these questions soon. In this article, we will first look at the power of XGBoost, and then deep dive into the inner workings of this popular and powerful technique. It's good to be able to implement it in Python or R, but understanding the nitty-gritties of the algorithm will help you become a better data scientist.

3.3.1 Features of XGBoost Model

XGBoost is a popular implementation of gradient boosting. Let's discuss some features of XGBoost that make it so interesting:

Regularization: XGBoost has an option to penalize complex models through both L1 and L2 regularization. Regularization helps in preventing overfitting.

Handling sparse data: Missing values or data processing steps like one-hot encoding make data space. XGBoost Classifier incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data

Weighted quantile sketch: Most existing tree based algorithms can find the split points when the data points are of equal weights (using quantile sketch algorithm). However, they are not equipped to handle weighted data. XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data

Block structure for parallel learning: For faster computing, XGBoost Classifier can make use of multiple cores on the CPU. This is possible because of a block structure in its system design. Data is sorted and stored in in-memory units called blocks. Unlike other algorithms, this enables the data layout to be reused by subsequent iterations, instead of computing it again. This feature also serves useful for steps like split finding and column sub-sampling

Cache awareness: In XGBoost, non-continuous memory access is required to get the gradient statistics by row index. Hence, XGBoost has been designed to make optimal use of hardware. This is done by allocating internal buffers in each thread, where the gradient statistics can be stored

Out-of-core computing: This feature optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory.

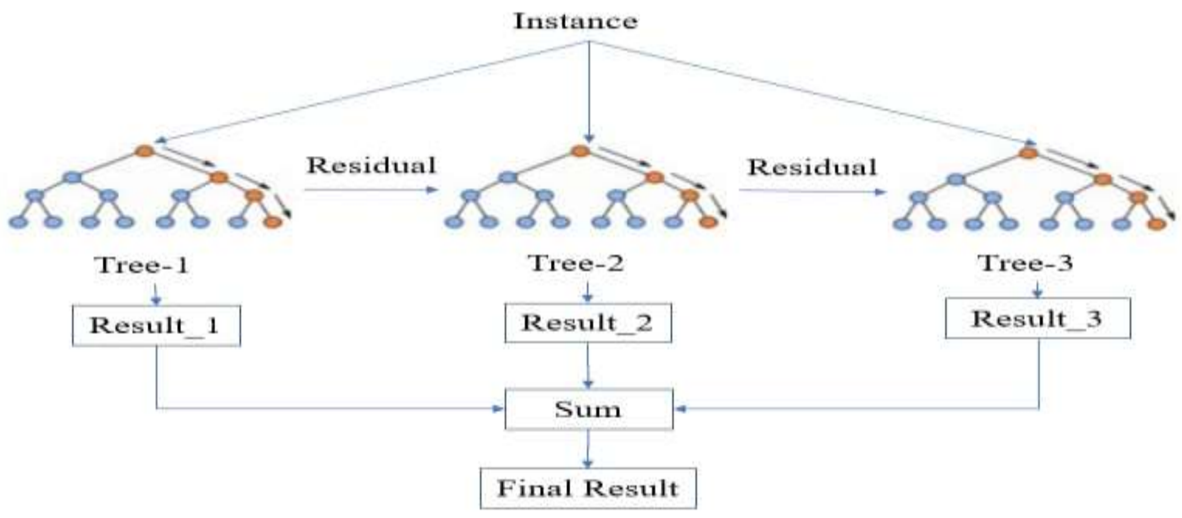


Fig 3.1: XGBoost Algorithm

3.3.2 Significance of XG-BOOST in bank transactions

XGBoost, or eXtreme Gradient Boosting, is widely employed in fraud detection within bank payments due to its robust performance and adaptability to complex patterns inherent in financial transactions. The algorithm's ability to handle imbalanced datasets, efficient computation, and feature flexibility makes it particularly well-suited for this critical application.One key advantage of XGBoost in fraud detection is its capacity to handle imbalanced data, a common characteristic of bank payment datasets where fraudulent transactions are relatively rare. XGBoost's inherent regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, mitigate overfitting and enhance the model's generalization capabilities, crucial for accurate detection of fraudulent patterns.

The algorithm's capability to manage both numerical and categorical features is crucial in the context of bank payments, where diverse types of information need to be considered. XGBoost's flexibility in handling mixed data types enables the development of comprehensive models without extensive preprocessing, allowing for a more accurate representation of the underlying patterns associated with fraudulent transactions.XGBoost's efficiency in parallel processing and optimization for large datasets aligns well with the high-dimensional and dynamic nature of bank payment data. Its speed and scalability are essential for handling the vast volume of transactions in real-time, contributing to the timely identification and prevention of fraudulent activities.Moreover, XGBoost provides interpretable insights into feature importance, aiding in the identification of key indicators of potential fraud. This interpretability is crucial for enhancing the trustworthiness of the fraud detection system and facilitating the investigation process.

In practice, data scientists often fine-tune XGBoost hyperparameters to optimize its performance for fraud detection. The algorithm's parameter customization allows for the adjustment of sensitivity and specificity levels, aligning the model with the specific risk tolerance and requirements of the bank.Overall, XGBoost's versatility, efficiency, and interpretability make it a popular choice in the field of fraud detection in bank payments. Its ability to handle complex patterns in high-dimensional datasets, coupled with effective regularization techniques, positions XGBoost as a valuable tool in the ongoing efforts to secure financial transactions and protect against fraudulent activities.

3.4 CatBoost

It a supervised machine learning method that is used by the Train Using AutoML tool and uses decision trees for classification and regression. As its name suggests, CatBoost has two main features, it works with categorical data (the Cat) and it uses gradient boosting (the Boost).

3.4.1 CatBoost in Machine Learning

We often encounter datasets that contain categorical features and to fit these datasets into the Boosting model we apply various encoding techniques to the dataset such as One-Hot Encoding or Label Encoding. But applying One-Hot encoding creates a sparse matrix which may sometimes lead to the overfitting of the model to handle this issue we use CatBoost. CatBoost automatically handles categorical features.

CatBoost or Categorical Boosting is an open-source boosting library developed by Yandex. It is designed for use on problems like regression and classification having a very large number of independent features.

Catboost is a variant of gradient boosting that can handle both categorical and numerical features. It does not require any feature encodings techniques like One-Hot Encoder or Label Encoder to convert categorical features into numerical features. It also uses an algorithm called symmetric weighted quantile sketch(SWQS) which automatically handles the missing values in the dataset to reduce overfitting and improve the overall performance of the dataset.

3.4.2 Features of CatBoost

1.**Built-in Method for handling categorical features:** CatBoost efficiently handles categorical features without requiring preprocessing. This capability eliminates the need to convert non-numeric factors into numerical values, simplifying the data preparation process.

2.**Excellent result without parameter tuning:** CatBoost aims to provide excellent results without the need for extensive parameter tuning. This feature saves time and effort for users, as they can achieve competitive performance with default parameters.

3.**Built-in methods for Handling missing values:** Unlike other Models, CatBoost can handle missing values in the input data without requiring imputation.

4.**Automatic feature scaling:** CatBoost internal scales all the columns to the same scaling whereas in other models we need to convert columns extensively.

5.**Robust to Overfitting:** CatBoost implements a variety of techniques to prevent overfitting, such as robust tree boosting, ordered boosting, and the use of random permutations for feature combinations. These techniques help in building models that generalize well to unseen data.

6.**Built-in cross-validation** – CatBoost internally applies a cross-validation method to choose the best hyperparameters for the model.

7.**Fast and scalable GPU version:** CatBoost offers a GPU-accelerated version of its algorithm, allowing users to train models quickly on large datasets. The GPU implementation enhances scalability and performance, especially when dealing with multi-card configurations.

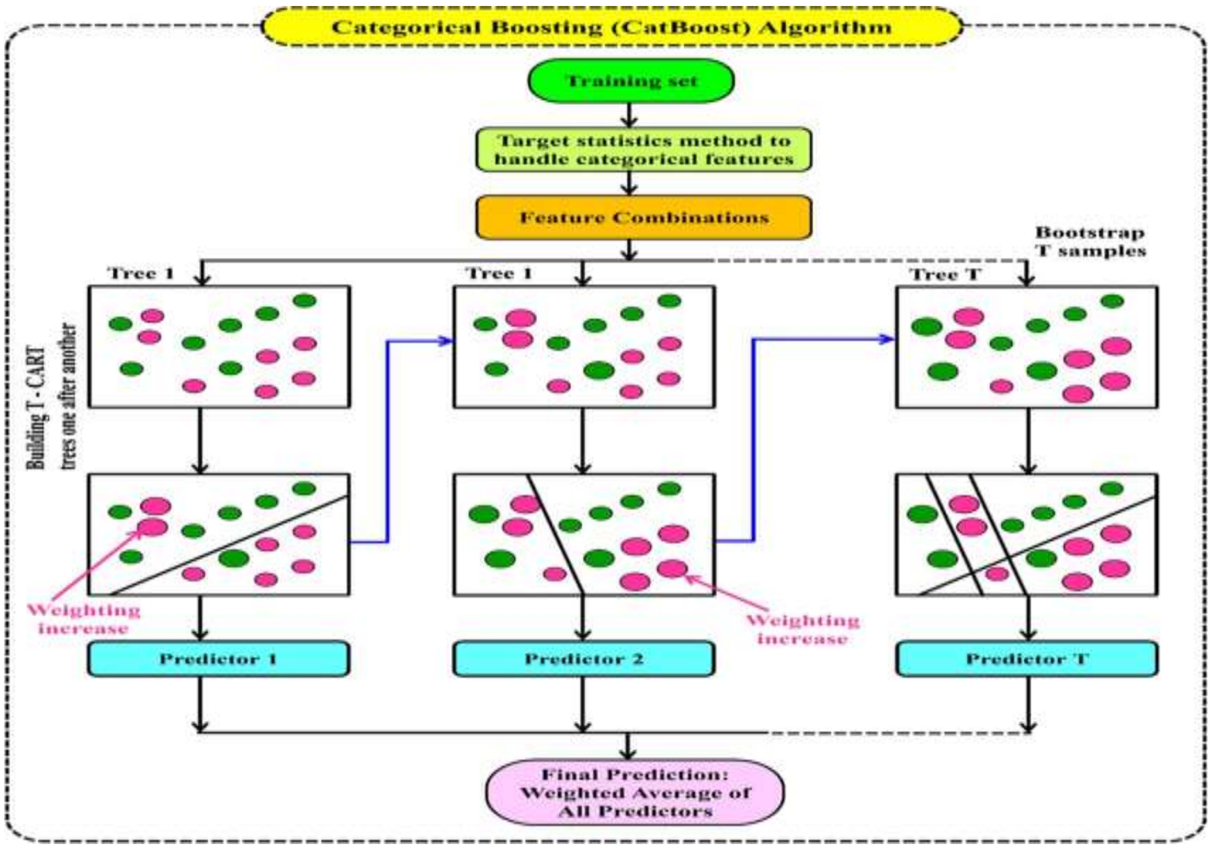

Categorical Boosting (CatBoost) Algorithm
Training set
Target statistics method to handle categorical features
Feature Combinations
Bootstrap T samples
Tree 1
Tree 1
Tree T
Building T - CART trees one after another
Weighting increase
Weighting increase
Predictor 1
Predictor 2
Predictor T
Final Prediction: Weighted Average of All Predictors

Fig 3.2 : Cat Boost Algorithm

3.4.3 Significance of CATBOOST in bank transactions

CatBoost, developed by Yandex, finds valuable applications in the domain of bank payments, particularly in tasks related to fraud detection, risk assessment, and transaction classification. Its unique strengths make it well-suited for handling the intricacies of financial data.One notable advantage of CatBoost in bank payments is its efficient handling of categorical features without the need for preprocessing. In the context of transaction data, where categorical features such as merchant categories and transaction types are prevalent, CatBoost's ability to directly work with such features preserves the richness of information, contributing to more accurate models.Furthermore, in the realm of fraud detection, CatBoost's robustness and regularization techniques play a crucial role. Fraud detection often deals with imbalanced datasets where genuine transactions far outnumber fraudulent ones. CatBoost's ability to prevent overfitting and generalize well to new instances makes it effective in identifying patterns associated with fraudulent activities.

CatBoost's compatibility with large datasets and its computational efficiency are particularly advantageous for processing the vast volume of transactions that occur in the context of bank payments. Its speed and scalability contribute to timely decision-making, a critical factor in preventing fraudulent transactions and ensuring the security of financial transactions.Additionally, CatBoost's integration with Python and its compatibility with scikit-learn make it accessible to data scientists and machine learning practitioners involved in developing and deploying models for bank payments. The seamless integration allows for a smooth workflow, from data preprocessing to model training and deployment.

In summary, CatBoost proves to be a valuable tool in the realm of bank payments, especially in tasks related to fraud detection and risk assessment. Its ability to handle categorical features, prevent overfitting, and efficiently process large datasets positions it as a reliable and effective choice for developing machine learning models to enhance the security and efficiency of financial transactions.

3.5 LightGBM

It is s a gradient boosting ensemble method that is used by the Train Using AutoML tool and is based on decision trees. As with other decision tree-based methods, LightGBM can be used for both classification and regression. LightGBM is optimized for high performance with distributed systems.

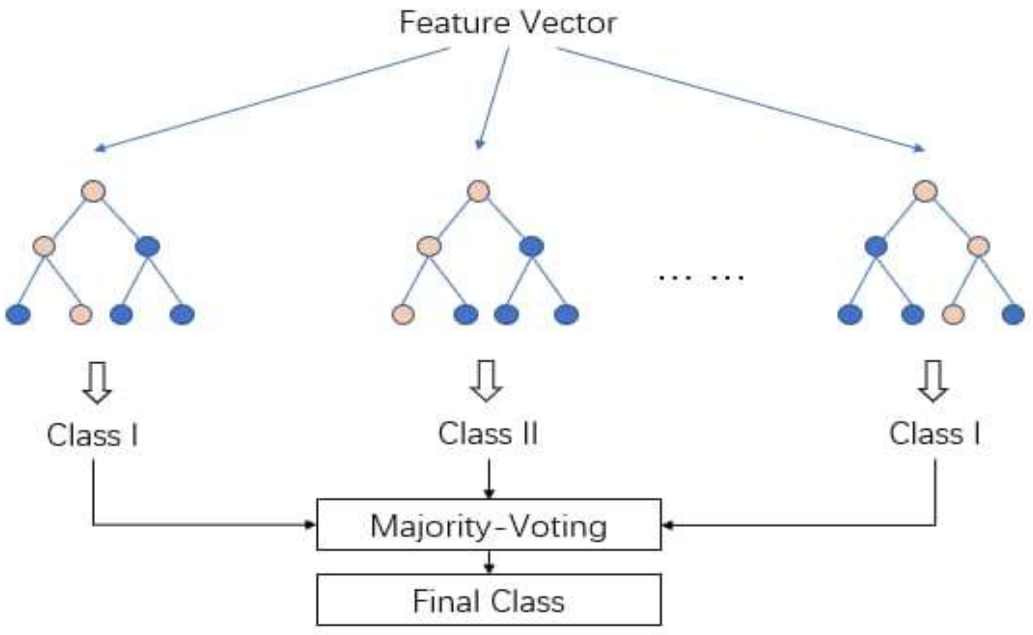


Fig 3.3 :LightGBM Model

3.5.1 LightGBM (Light Gradient Boosting Machine)

LightGBM is an ensemble learning framework, specifically a gradient boosting method, which constructs a strong learner by sequentially adding weak learners in a gradient descent manner. It optimizes memory usage and training time with techniques like Gradient-based One-Side Sampling (GOSS).

LightGBM is an open-source, distributed, high-performance gradient boosting framework developed by Microsoft. It is designed for efficiency, scalability, and accuracy. It is based on decision trees designed to improve model efficiency and reduce memory usage. It incorporates several novel techniques, including Gradient-based One-Side Sampling (GOSS), which selectively retains instances with large gradients during training to optimize memory usage and training time. Additionally, LightGBM employs histogram-based algorithms for efficient tree construction. These techniques, along with optimizations like leaf-wise tree growth and efficient data storage formats, contribute to LightGBM's efficiency and give it a competitive edge over other gradient boosting frameworks.

LightGBM installations

LightGBM installations involve setting up the LightGBM gradient boosting framework on a local machine or server environment. This typically includes installing necessary dependencies such as compilers and CMake, cloning the LightGBM repository from GitHub, building the framework using CMake, and installing the Python package using pip.

Proper installations ensure that users can utilize LightGBM’s efficient algorithms and functionalities for machine learning tasks effectively.

3.5.2 Significance of CATBOOST in bank transactions

Light GBM (Gradient Boosting Machine) plays a significant role in fraud detection within bank payments using machine learning algorithms. Its efficacy stems from several key attributes. Firstly, Light GBM is renowned for its efficiency and speed, particularly suited for large datasets common in financial transactions. This enables rapid processing and real-time detection of fraudulent activities, crucial in preventing financial losses.

Moreover, Light GBM's ability to handle categorical features effectively enhances its applicability in fraud detection, where diverse transaction characteristics require nuanced analysis. Its capability to handle imbalanced datasets is vital in detecting fraudulent transactions, which often represent a minority class.

Additionally, Light GBM's interpretability facilitates understanding the model's decision-making process, aiding in the identification of fraudulent patterns. Its ensemble learning approach, combining multiple weak learners to create a robust model, enhances detection accuracy by capturing intricate fraud patterns while minimizing false positives.

In summary, Light GBM's efficiency, feature handling capabilities, interpretability, and ensemble learning methodology make it indispensable in fraud detection within bank payments, providing financial institutions with a powerful tool to safeguard against fraudulent activities.

CHAPTER-4

LIBRARIES USED AND INSTALLATION

4.1 Libraries used

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

4.2 Installation of Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

4.2.1 How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheethere.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

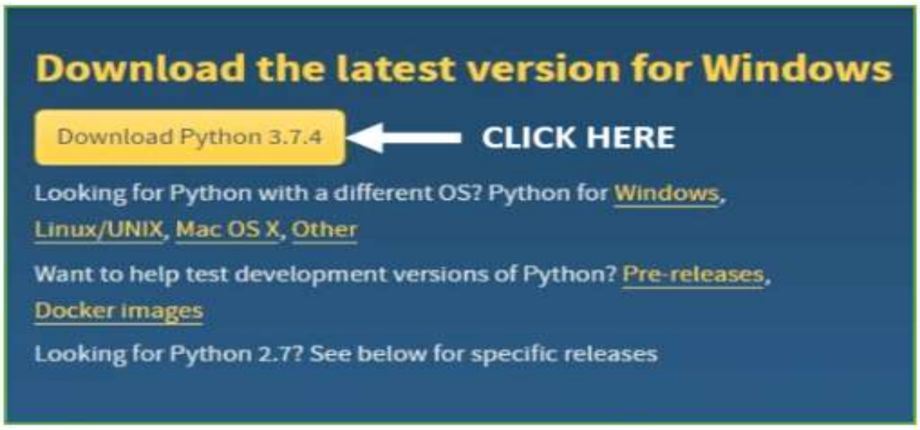
4.3 Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://www.python.org**

Fig 4.1: Installation of python using google chrome

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.


Download the latest version for Windows

Download Python 3.7.4 CLICK HERE

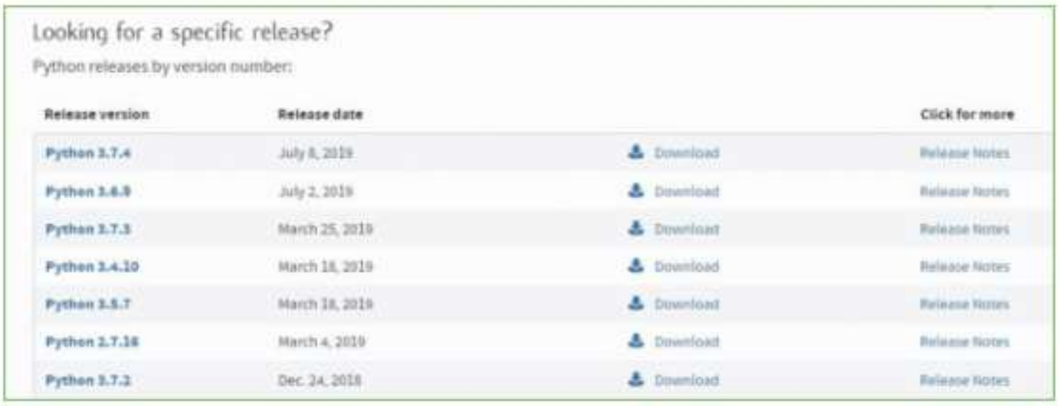
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Fig 4.2: Download tab

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4.


Looking for a specific release?

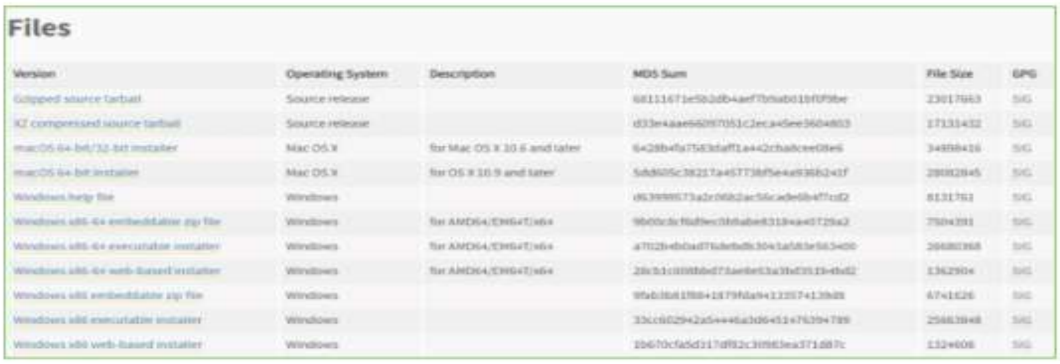
Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Fig 4.3: Python versions

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131763	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

Fig 4.4: Python versions along with operating system

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

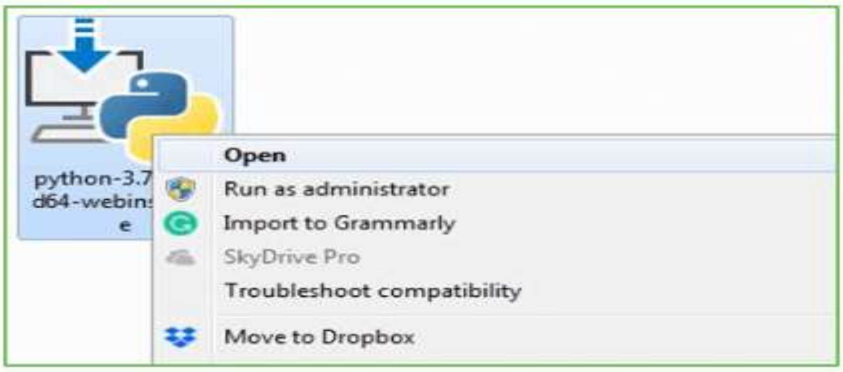
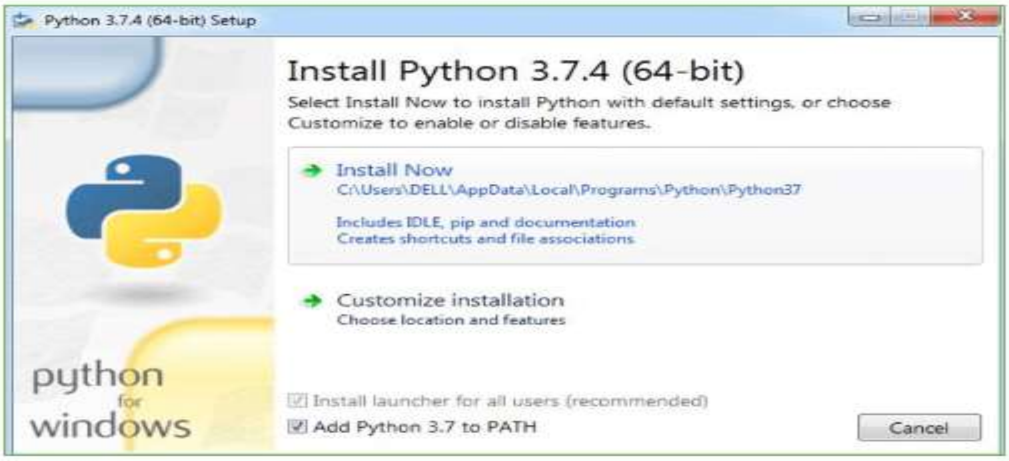


Fig 4.5: Downloaded python version

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

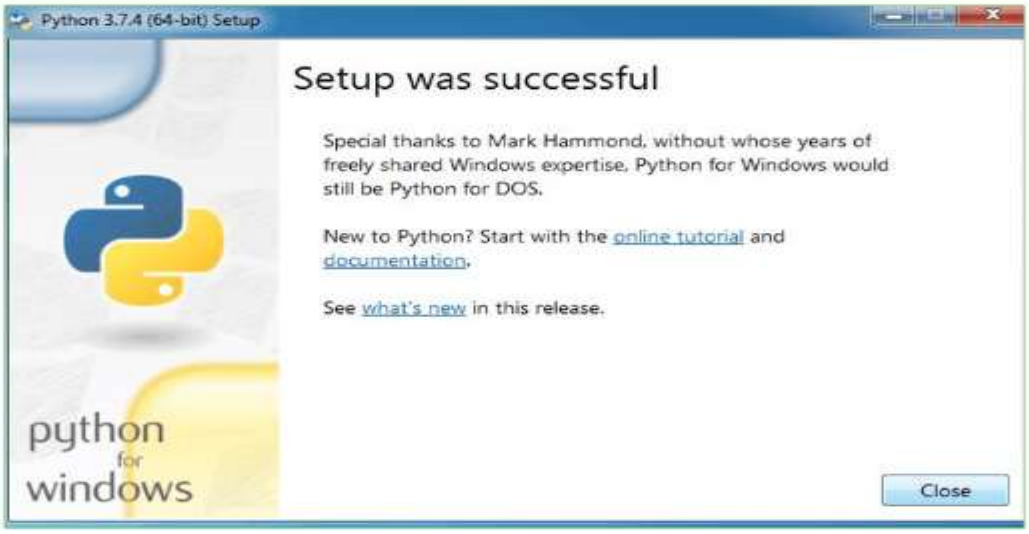
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Fig 4.6: Installation of downloaded python

Step 3: Click on Install NOW After the installation is successful. Click on Close.


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Fig 4.7: Installation successful

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd".

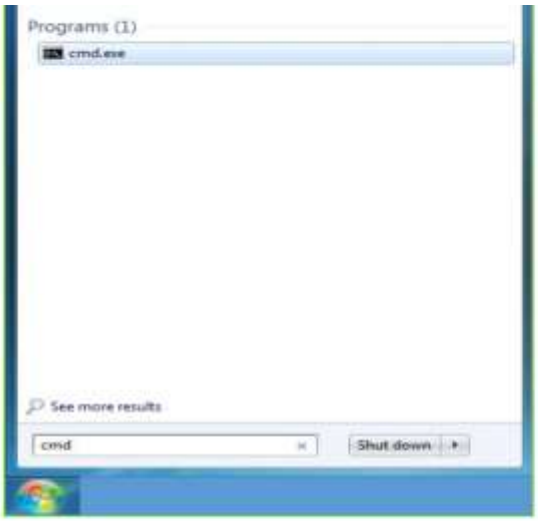
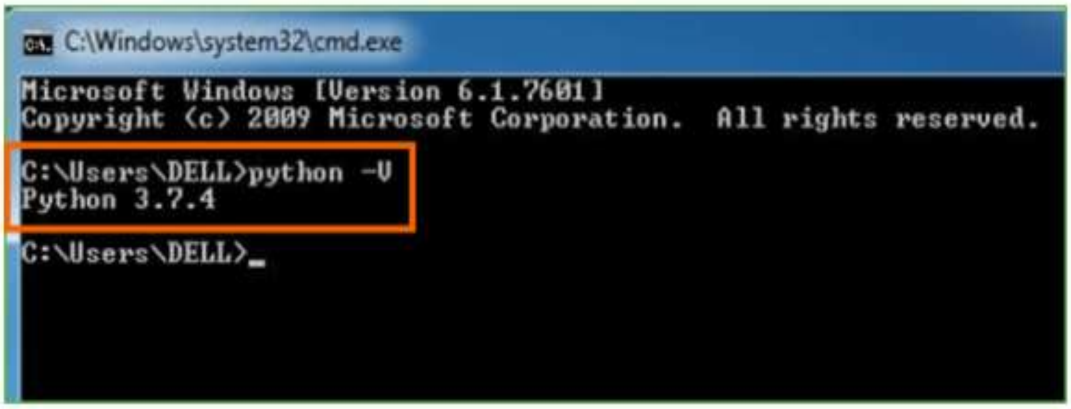


Fig 4.8: Windows run command

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

Fig 4.9: Command prompt

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".

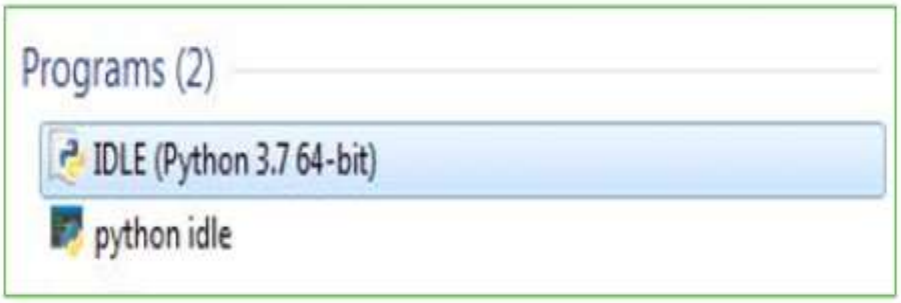
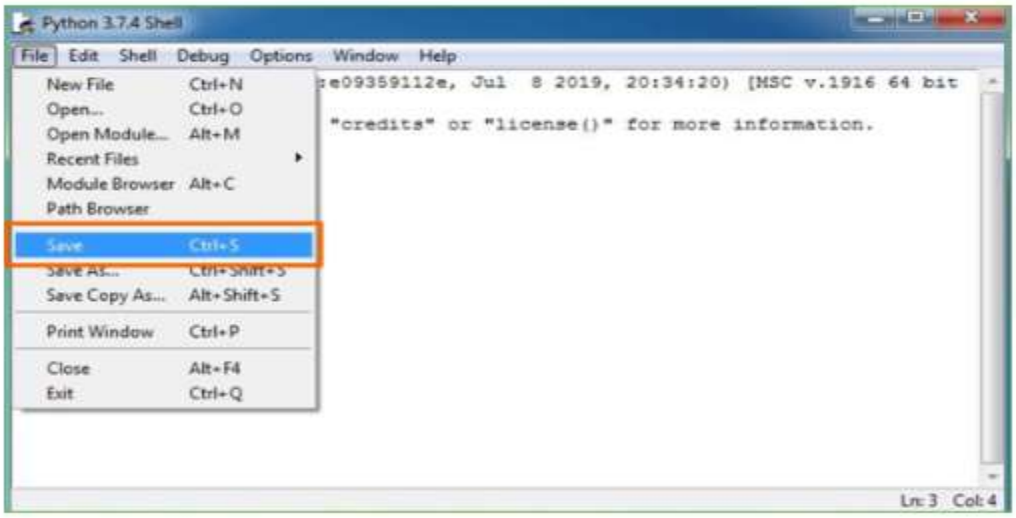


Fig 4.10: Windows run command python IDLE

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
 "credits" or "license()" for more information.
```

Fig 4.11: Working in IDLE

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print

CHAPTER-5

METHODOLOGY

5.1 System Requirements

5.1.1 Hardware Requirements

• System	: Pentium IV 2.4 GHz.
• Hard Disk	: 40 GB.
• Floppy Drive	: 1.44 Mb.
• Monitor	: 15 VGA Colour.
• Mouse	: Logitech.
• Ram	: 512 Mb.

5.1.2 Software Requirements

- Operating System: Windows
- Coding Language: Python 3.7

5.2 Proposed Approach To Detecting Credit Card Fraud

The proposed framework for fraud detection is presented in Fig. 2. As this figure shows, we first apply the desired pre-processing on the data and further divide the data into two sections: training and testing, followed by performing Bayesian optimization on the training data to find the best hyperparameters that lead to the improvement of the performance. We use the cross-validation method to obtain performance comparison in an unbalanced set and then examine the algorithms using different evaluation metrics, including accuracy, precision, recall, the Matthews correlation coefficient (MCC), the F1-score, and AUC diagrams. These steps are explained in detail as follows:

Dataset

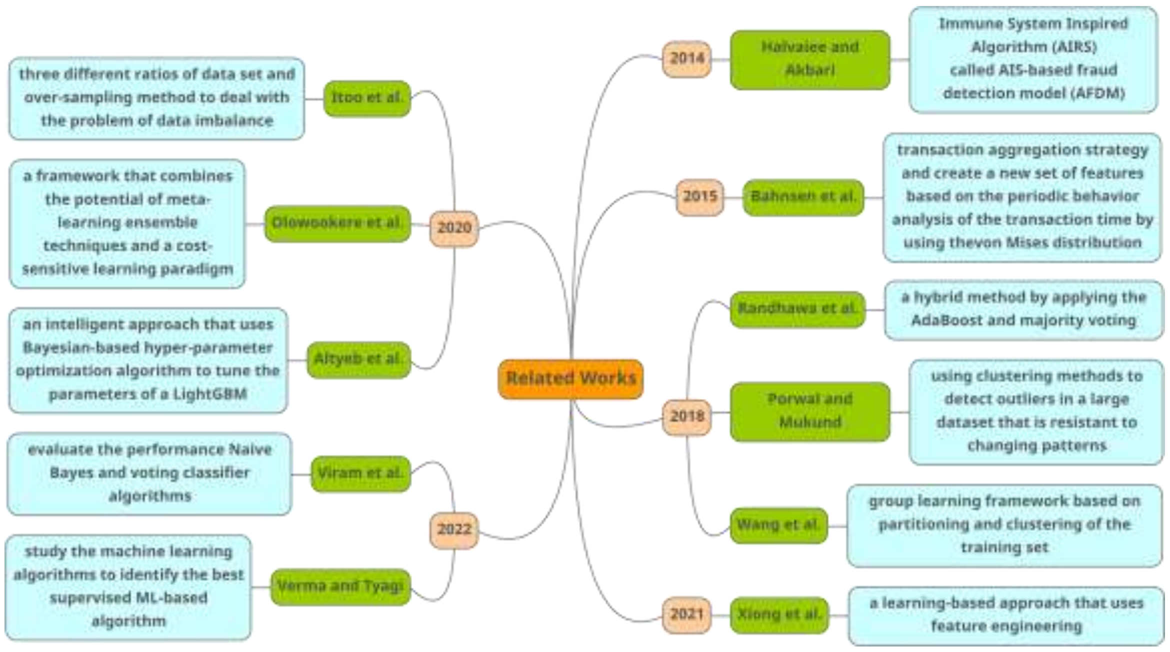

Related Works
2020
Itoo et al.
three different ratios of data set and over-sampling method to deal with the problem of data imbalance
Olowookere et al.
a framework that combines the potential of meta-learning ensemble techniques and a cost-sensitive learning paradigm
Altyeb et al.
an intelligent approach that uses Bayesian-based hyper-parameter optimization algorithm to tune the parameters of a LightGBM
2022
Viram et al.
evaluate the performance Naive Bayes and voting classifier algorithms
Verma and Tyagi
study the machine learning algorithms to identify the best supervised ML-based algorithm
2014
Halvaiee and Akbari
Immune System Inspired Algorithm (AIRS) called AIS-based fraud detection model (AFDM)
2015
Bahnsen et al.
transaction aggregation strategy and create a new set of features based on the periodic behavior analysis of the transaction time by using thevon Mises distribution
2018
Randhawa et al.
a hybrid method by applying the AdaBoost and majority voting
Porwal and Mukund
using clustering methods to detect outliers in a large dataset that is resistant to changing patterns
Wang et al.
group learning framework based on partitioning and clustering of the training set
2021
Xiong et al.
a learning-based approach that uses feature engineering

Fig 5.1: Summary of the related works on fraud detection in banking industry with machine learning techniques

5.2.1: Block diagram

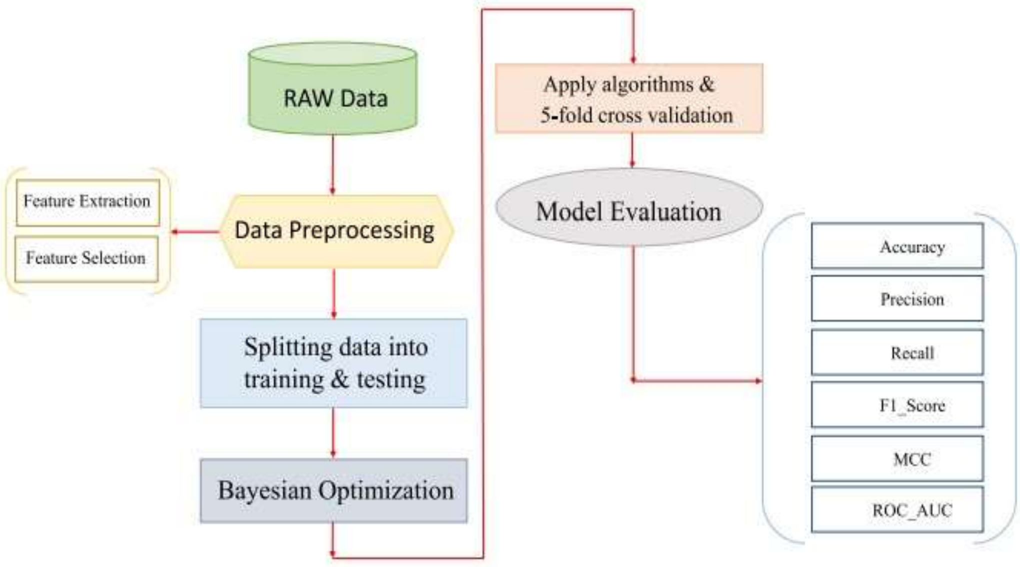


Fig 5.2: Proposed framework for credit card fraud detection.

In this project, we use a real dataset so that the outcome of the proposed algorithm can be used in practice. We consider a dataset named “creditcard” that contains 284,807 records of two days of transactions made by credit card holders in September 2013. There are 492 fraudulent transactions, and the rest of the transactions are legitimate. The positive class (frauds) accounts for 0.172% of all transactions; hence, the dataset is highly imbalanced. This dataset is available and can be accessed through

This dataset contains only numerical input variables resulting from a principle component analysis (PCA)transformation. Unfortunately, the original features and background information about the data are not given due to confidentiality and privacy considerations. PCA yielded the following principal components: V1,V2,V28. The untransformed features with PCA are ‘‘time’’ and ‘‘amount.’’ The ‘‘Time’’ column contains the time (in seconds) elapsed between each transaction and the first transaction in the dataset. The feature ‘‘Amount’’ shows the transaction amount.Feature ‘‘Class’’ is the response variable, and it takes the value 1 in case of fraud and 0 otherwise. The summary of the variables and features is presented in Table 1.

Data Pre-Processing

As illustrated in Table 2, the total number of fraudulent transactions is significantly lower than the total number of legitimate transactions, indicating that the data distribution is unbalanced. In real datasets for credit card fraud detection, unbalanced data is expected. This data imbalance causes performance issues in machine learning algorithms, and having a class with the majority of the samples influences the evaluation results. Therefore, in many studies, undersampling and over-sampling methods are used to solve the data imbalance problem. Using under-sampling methods leads to data loss. Besides, using over-sampling methods leads to the production of duplicate data that doesn’t provide information (the data and information are different, and the subject is discussed under the ‘‘Entropy’’). Some researchers use synthetic minority oversampling (SMOTE) as a solution, which avoids the drawbacks of under and over sampling. However, the SMOTE method causes an increase in the false-positive rate, which is not acceptable in banking for customer orientation. To solve this problem, in this study, we use class weight tuning hyperparameter to solve the mentioned disadvantages. However, the SMOTE method causes an increase in the false-positive rate, which is not acceptable in banking for customer orientation. To solve this problem, in this study, we use class weight tuning hyper parameter to solve the mentioned disadvantages.

TABLE 2. Transaction label distribution in the ‘‘credit card’’ dataset this unbalanced data is expected in real-life datasets.

No. of Trans-actions	**No. of legit-imate Trans-actions**	**No. of fraudulent Transac-tions**	**Legitimate (%)**	**fraudulent (%)**
284,807	284,315	492	99.83%	0.17%

Feature Extraction

The ‘‘time’’ feature includes the time (in seconds) elapsed between each transaction and the first transaction. To make the most of the feature, we expand it to extract the transaction hour feature, which gives us more information than the time feature itself.

Feature Selection

The features are unknown except for ‘‘Time’’ and ‘‘Amount’’, and we have no additional information. Feature selection tries to find a subset of features that improve the classifier’s performance on effectively detecting credit card fraud. The information gain (IG) method is used to select the most important features that lead to a dimension reduction of the training data. Information gain functions by extracting similarities between credit card transactions and then awarding the greatest weight to the most significant features based on the class of legitimate and fraudulent credit card transactions. The information gain method has been proven to be computationally efficient and shows leading performance in terms of precision. Therefore, we also consider the IG method for feature selection in the proposed framework. Figure 3 shows the diagram of the IG, and the top six features extracted by this method have been used to evaluate the proposed algorithm.

Splitting Data into Training & Testing Sets

To evaluate the performance of the machine learning model, it's essential to separate the dataset into two parts: a training set and a testing set. The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data. Typically, a random split is used, with a majority of the data allocated to the training set and a smaller portion to the testing set.

Bayesian Optimization

Bayesian optimization is a technique used to search for the optimal hyperparameters of machine learning algorithms. Hyperparameters are parameters that are set prior to training and cannot be learned from the data. Bayesian optimization efficiently explores the hyperparameter space by iteratively selecting promising parameter configurations based on past evaluations of the objective function (e.g., model performance).

5.2.2 Algorithms

Hyperparameters have a significant effect on the performance of machine learning models. We refer to optimization as the process of finding the best set of hyperparameters that configure a machine learning algorithm during its training. Recently, it was shown that the Bayesian method is capable of finding the optimised values in a much smaller number of training courses compared with evolutionary optimization methods. In this model, we use the Bayesian optimization algorithm to tune the hyperparameters that lead to computational time reduction and performance improvement.

1) Logistic Regression

Logistic regression is a predictive analysis that finds out if two or more variables are related to each other. This method determines whether there is a relationship between one binary dependent variable and one or more ordinal, nominal, interval, or ratio-level independent variables.

This algorithm could not be used for unbalanced data. Therefore, we used hyperparameter class weight to solve the class imbalance prior to applying logistic regression. We show that the ROC-AUC curve cannot be used for the evaluation of unbalanced data and leads to false interpretations.

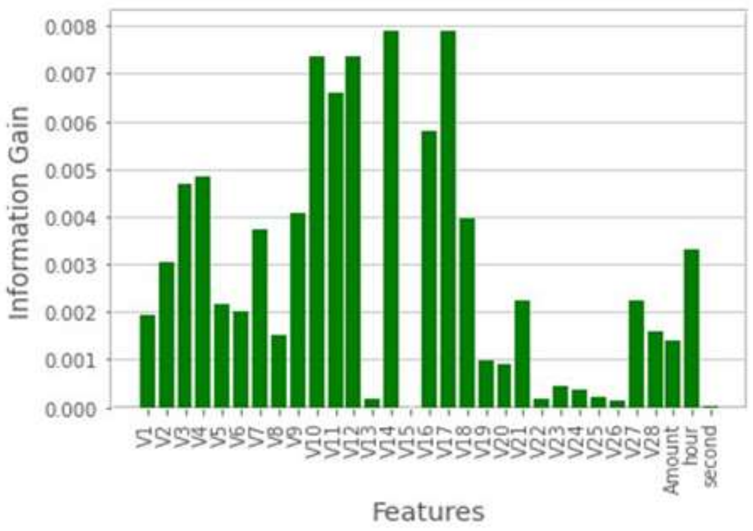


Fig 5.3: Feature importance diagram that shows the IG for the unknown features of the "creditcard" dataset. The top six features are used in evaluations.

2) LightGBM

The LightGBM algorithm is built on the GBDT framework and aims to improve computational efficiency, particularly on big data prediction problems . The high performance LightGBM algorithm can quickly handle large amounts of data, and the distributed processing of data. In LightGBM, the histogram-based algorithm and trees' leaf-wise growth strategy with a maximum depth limit are adopted to increase the training speed and reduce memory consumption. The tuned hyperparameters include the "num_leaves", which is the number of leaves per tree, "max_depth", which denotes the maximum depth of the tree, and "learning_rate" which is also balanced by tuning the weight of the class. With the excessive increase of the leaves, the problem fits horizontally. Therefore, we need to consider a suitable range for this algorithm to obtain good optimization results.

3) XGBoost

eXtreme Gradient Boosting (XGBoost) has become a dominant algorithm in the field of applied machine learning. XGBoost is a type of decision tree algorithm with boosted gradients. It is preferred over other gradient boosting machines (GBMs) due to its fast execution speed, model performance, and memory resources. This algorithm is a hybrid technique in which new models are added to fix errors caused by existing models. XGBoost includes parallel computation to construct trees using all the CPUs during training. Instead of traditional stopping criteria (i.e., criterion first), it makes use of the “max depth” parameter and starts tree pruning from the backward direction, which significantly improves the computational performance and speed of XGBoost. XGBoost employs a more regularised technique called “formalization” to control over-fitting and achieve better performance. The tuned hyperparameters include learning rate, number of trees, and maximum tree depth, as well as applying weight to classes

4) CatBoost

Category Boosting (CatBoost) is a new gradient boosting algorithm proposed by Prokhorenkova et al. CatBoost is a competitive candidate in the realm of classifiers for highly unbalanced data. CatBoost machine learning algorithm is a particular type of Gradient boosting on the decision trees as it can handle categorical, ordered features, and the overfitting of the model is taken care of by Bayesian estimators. CatBoost doesn’t require extensive data training like other machine learning models and can be successfully applied to diverse types and formats of data. CatBoost has both CPU and GPU implementations, the GPU implementation allows for much faster training and is faster than both state-of-the-art open-source GBDT GPU implementations, XGBoost and LightGBM, on ensembles of similar sizes. CatBoost uses a more efficient strategy hat reduces over-fitting and allows the use of the whole dataset for training. We perform a random permutation of the dataset, and also, for data imbalance problems, we use a class weight hyperparameter.

5) Majority Voting

Ensemble learning (EL), which is a type of machine learning, combines several classifiers, minimises the error of the classifiers, and achieves more reasonable results than a single technique. A voting majority classifier is not a real classifier, but a method that is trained and evaluated in parallel in order to use the different features of each algorithm. We can train the data using different hybrid algorithms to predict the final output. The final result of the prediction is determined by a majority of votes according to two different strategies: hard voting and soft voting. If voting is hard, it uses the predicted class labels to vote for the majority law. Otherwise, if the vote is soft, it predicts the class label based on “Argmax,” the sum of the predicted probabilities, which is recommended for a set of well-calibrated classifiers. In this case, the probability vector is calculated on average for each predicted class (for all classifiers). The winning class is the one with the highest value .

$$y^{\wedge} = \text{argmax} \frac{1\,X}{\text{NClassifiers}}, (p1,...,pn) \qquad (1)$$

$$\text{Classifiers}$$

6) Deep Learning

Deep learning algorithms are a class of machine learning algorithms where multiple hidden layers are used to improve the outcome. Deep learning is shown to be a very promising solution to deal with fraud in financial transactions, making the best use of banks’ big data. [34]. Deep learning is a generic term that refers to machine learning using a deep multi-layer artificial neural network (ANN). It is a biologically inspired model of human neurons, composed of multilevel hidden layers of nonlinear processing units, where each neuron is able to send data to a connected neuron within the hidden layers. These processing units discover intermediate representations in a hierarchical manner. The features discovered in one layer form the basis for the processing of the succeeding layer. In this way, deep learning algorithms learn intermediate concepts between raw input and target knowledge .

In this project, we use a sequential model, which is a linear stack of layers to construct an artificial neural network model. Our model has a dense class, which is a very common layer and is often used. In the neural network, the activation function is used to increase the predictive power. This function divides input signals into output signals. We use the Relu activation function, and in the last layer, we use “Sigmoid”, since our output is binary. The Sigmoid function generates values in a range of zero and one. In the “Relu” function, if the value x is smaller than or equal to zero, the output is zero. The function of the Relu activation function is in many ways similar to the function of our biological neurons.

Neural networks require initial weighting. We use kernelinitializer, which defines the method of determining the random weights of the primary Keras layers. To overcome the unbalanced data problem, we consider the ratio of 1 to 4 for the weight of the majority class to the minority class. This causes an increase in the processing speed as well as increasing the efficiency of the model. The size of the input layer is equal to the number of features plus the extracted features. We also remove the “time” feature. To build the Keras model, we optimise the number of layers and neurons, the number of epochs, and the batch size, which leads to an increase in speed. Commonly, batch size is set to 32 or 128. However, our dataset is highly unbalanced, and by choosing the common batch size, there may be no fraud cases in the batch during training. Therefore, our range is chosen so that we can see fraudulent samples in each batch. Also, by choosing a larger batch size, the processing is faster, and we also need less memory. Large epoch sizes can result in either over- or underfitting. Therefore, selecting the appropriate range for optimization not only increases the efficiency of the algorithm but also reduces the time required to find the optimal points. By performing Bayesian optimization, the number of neurons in the first hidden layer is set to 86, the number of epochs is set to 117, and the batch size is set to 1563. The details of our model are presented in Table 3.

Following Keras and with the help of the compile method and Adam's optimizer, we perform weight updates and use binary-cross entropy for the loss function that finalises the configuration of the learning and training process.

TABLE 3. Details of our deep learning model used in the paper are provided. The total parameters are set to 7593, and all are trainable.

Layer(Type)	**Output Shape**	**Param No.**
dense (Dense)	(None, 86)	2752
dense-1 (Dense)	(None, 44)	3828
dense-2 (Dense)	(None, 22)	990
dense-3 (Dense)	(None, 1)	23

Applying Algorithms & 5-fold Cross Validation

5.2.3 Evaluation Metrics

We apply a cross-validation test to evaluate the performance of the proposed model for credit card fraud detection. Similar to [6], [17], We use a stratified 5-fold validation test to obtain a reliable performance comparison in the unbalanced set. The dataset is divided randomly into five separate subsets of equal size, where the number of samples in each class is divided into equal proportions in each category. In all steps of validation, a single subset (20% of the dataset) is reserved as the validation data to test the performance of the proposed approach, while the remaining four subsets (80% of the dataset) are employed as the training data. We repeat this process five times until all subsets are used. The average performances of the five test subsets are calculated, and the final result is the performance of the proposed approach on a 5-fold cross-validation test.

To be fair in our comparisons, we use the common metrics for our evaluations, including accuracy, precision, recall, the Matthews correlation coefficient (MCC), the F1-score, and AUC diagrams. Positive numbers represent fraudulent transactions in our experiments, while negative numbers represent legitimate ones. True positive (TP) represents fraudulent transactions that have been classified as such. False positives (FP) indicate the number of legitimate transactions misclassified as fraudulent. The true negative (TN) represents legitimate transactions classified as legitimate, and the false negative (FN) indicates the misclassified fraudulent transactions as legitimate [15]. The mathematical expressions for the metrics used are given in Eq. (2) to Eq. (6).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + TN} \tag{3}$$

$$\text{Precision} = \frac{TP}{TP + TN + FP + FN} \tag{4}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \tag{5}$$

$$\text{MCC} = \frac{TP \times FP - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{6}$$

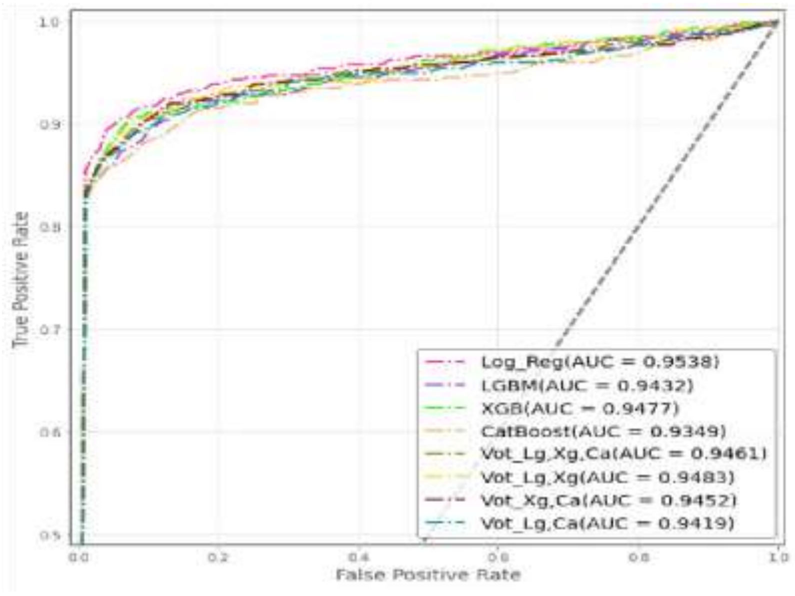


Fig 5.4: ROC_AUC curve

Accuracy :Accuracy quantifies the total performance of the classifier and is defined as the number of correct predictions made by the model. When dealing with data that isn't balanced, this criterion doesn't give good results because it also gives a high value if even one fraudulent transaction is found. Recall shows the efficiency of the classifier in detecting actual fraudulent transactions. Precision measures the reliability of the classifier and F1-Score is the harmonic average of recall and precision measures, that considers both false negatives and positives.

ROC-AUC is a measure of separability that demonstrates the model's ability to differentiate between classes. ROC-AUC is a graphical plot of the false positive rate (FPR) and the true positive rate (TPR) at different possible levels [17]. The area under the ROC curve is not a suitable criterion for evaluating fraud detection methods since it only considers positive values. The precision and recall curves are commonly used to compare classifiers in terms of precision and recall. Usually, in this two-dimensional graph, the precision rate is plotted on the y-axis and the recall is plotted on the x-axis. There is no good way to describe the true and false positives and negatives using one indicator. One good solution is to use MCC, which measures the quality of a two-class problem, taking into account the true and false positives and negatives. It is a balanced measure, even when the classes are of different sizes .

Model Evaluation

Finally, the performance of the trained machine learning models is evaluated using a set of evaluation metrics. These metrics include accuracy, precision, recall, F1 score, Matthews correlation coefficient (MCC), and receiver operating characteristic area under the curve (ROC AUC). These metrics provide insights into how well the model is performing in terms of correctly identifying fraudulent transactions while minimizing false positives and false negatives.

We use the stratified 5-fold cross validation method and the boosting algorithms with the Bayesian optimization method to evaluate the performance of the proposed framework. We extract the hyperparameters and evaluate each algorithm individually before using the majority voting method. We examine the algorithms in triple and double precision.The comparison results are presented in Table 5.

Most studies in the literature rely on AUC diagrams to evaluate performance. However, as can be seen from the ROC-AUC curve in Fig. 4, the value of AUC in severely unbalanced data is not a good evaluation metric. It is influenced by the real positives and considers the negatives irrelevant. According to the ROC-AUC Fig. 4, the logistic regression algorithm 0.9583 has the highest number of fraud detection, but it has the lowest value in other criteria.

TABLE 4. Performance evaluation of algorithms.

Model	Accuracy	AUC	Recall	Precision	F1-score	MCC
Log Reg	0.97477	0.9578	0.8730	0.0617	0.1143	0.2248
LGBM	0.99919	0.9472	0.7990	0.7534	0.7699	0.7727
XGB	0.99923	0.9517	0.7949	0.7862	0.7830	0.7864
CatBoost	0.99880	0.9390	0.8096	0.6431	0.7066	0.7158
Vot Lg, Xg, Ca	0.99924	0.9501	0.8033	0.7720	0.7825	0.7847
Vot Lg, Xg	0.99927	0.9522	0.8012	0.7901	0.7901	0.7925
Vot g, Ca	0.99923	0.9492	0.8097	0.7681	0.7823	0.7852
Vot Lg, Ca	0.99912	0.9459	0.8075	0.7260	0.7581	0.7620

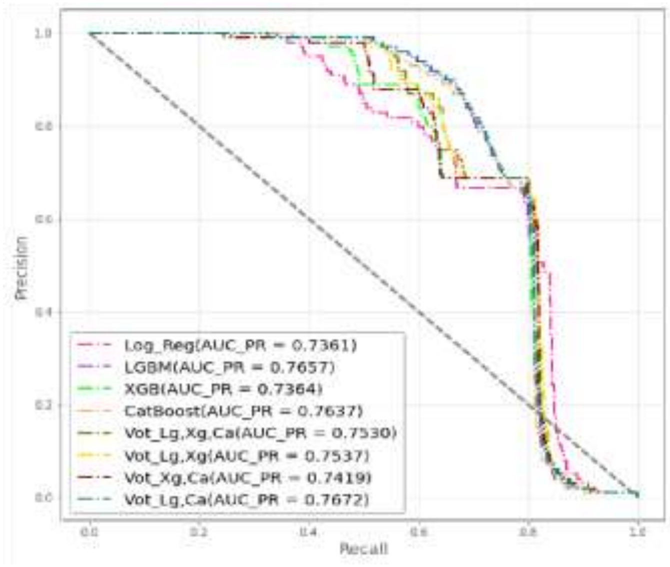


Fig 5.5 :Precision_Recall curve

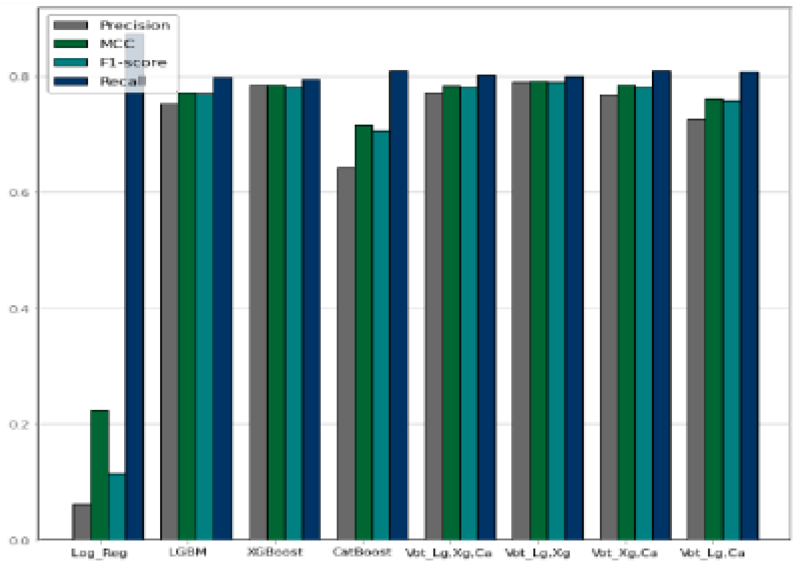


Fig 5.6: Performance comparing algorithms with different evaluation criteria.

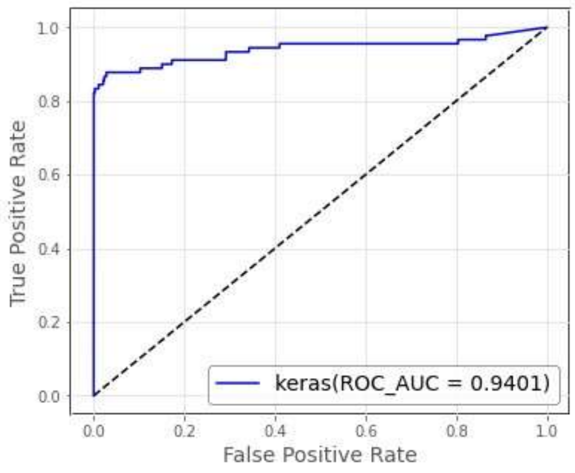


Fig 5.7: ROC curve of deep learning

TABLE 5. Deep learning model results**.**

Model	Accuracy	AUC	Recall	Precision	F1-score	MCC
Keras	0.9994	0.9401	0.8222	0.8043	0.8132	0.8129

The precision-recall curve is illustrated in Fig. 5 and shows the system performance in a more precise manner compared with the ROC-AUC curve. However, the results cannot be cited because false negatives are far from the view of this diagram. As Fig. 5 shows, the highest value belongs to the combination of the CatBoost and LightGBM algorithms with a value of 0.7672, and the lowest value belongs to logistic regression and is 0.7361.

Comparing the precision, recall, and F1-score as well as the MCC, the algorithms used are shown in Fig. 6. The best performance is related to the combination of lightGBM and XGBoost algorithms, which have an MCC value of 0.79 and an F1-score of 0.79. In individual algorithms, XGBoost has the highest values.

According to the digits obtained in Table 5, deep learning has achieved better performance compared with individual algorithms and majority voting ensemble learning. The MCC and F1-score metrics have values of 0.8129 and 0.8132, respectively. The area under the ROC curve in the deep learning method is illustrated in Fig. 7 and shows a value of 0.9401.

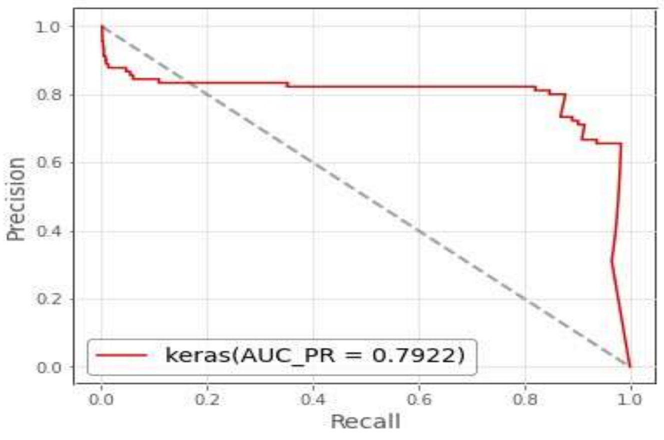


Fig 5.8: Precision- recall curve of deep learning

The diagram of the Precision-Recall curve is shown in Fig. 8, and shows the value as 0.7922. The evaluation results of the proposed approach using different pre-processing and class weight hyperparameter tuning to deal with the problem of data unbalance compared to previous data are shown in Fig. 9. The results show improvement of both methods compared to the previous observation method .

According to the Table 6, it is shown that the proposed methods outperform the intelligence method presented in using common metrics and a public dataset.

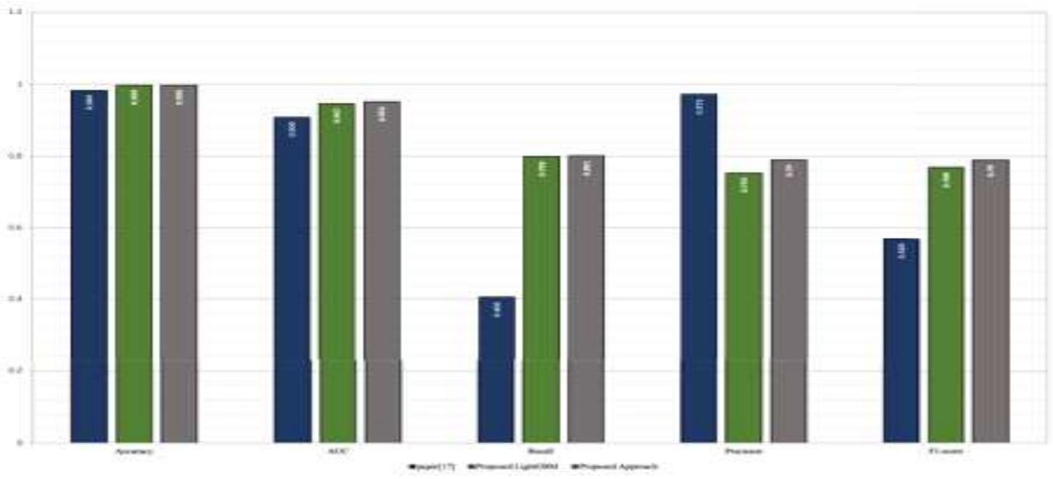


Fig 5.9:Performance comparison of the proposed approach based on the different evaluation criteria.

CHAPTER-6

RESULTS AND SIMULATION ANALYSIS

6.1 Result description

- ➢ Firstly we have to choose a .CSV file from files
- ➢ Click on Upload
- ➢ After uploading the file we have to split the data by clicking on Split Data

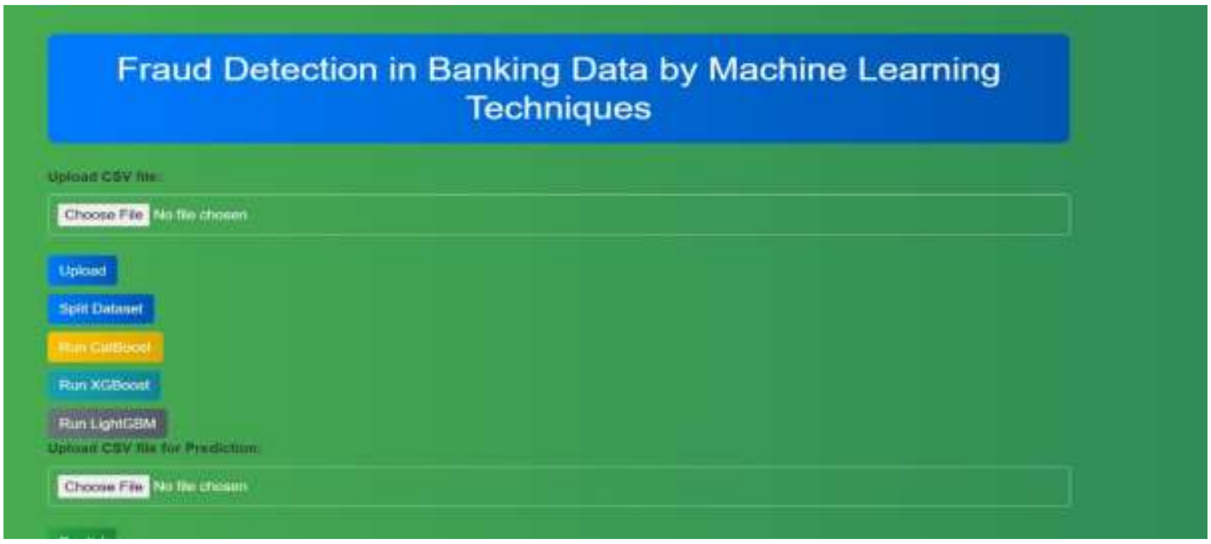
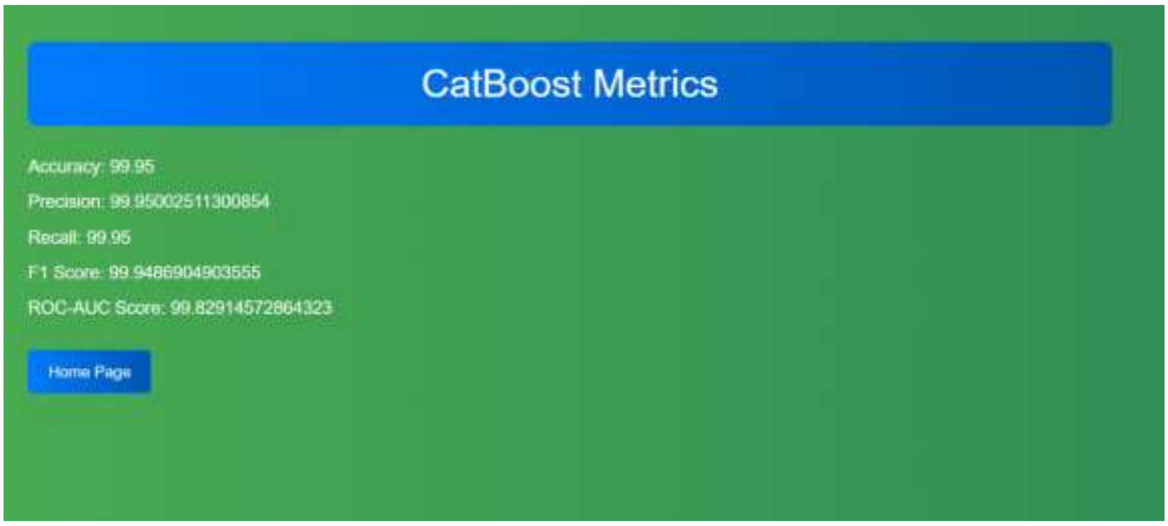


Fig 6.1:Uploading Window

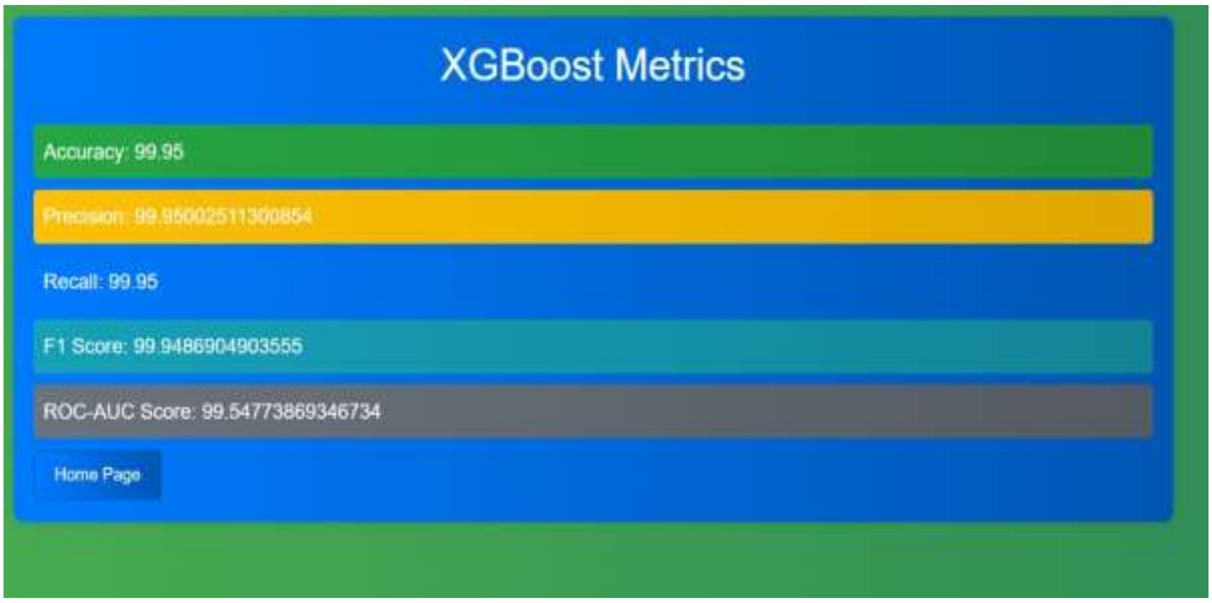
- ➢ After splitting data we have to run the data by using CatBoost algorithm.(click-Run CatBoost)



```
Accuracy: 99.95
Precision: 99.95002511300854
Recall: 99.95
F1 Score: 99.9486904903555
ROC-AUC Score: 99.82914572864323
```

Fig 6.2:Output screen of CatBoost

➢ After getting output screen of CarBoost , go to Home page then we have to run the date in XG Boost algorithm for that click on Run XGBoost


XGBoost Metrics

Accuracy: 99.95

Precision: 99.95002511300854

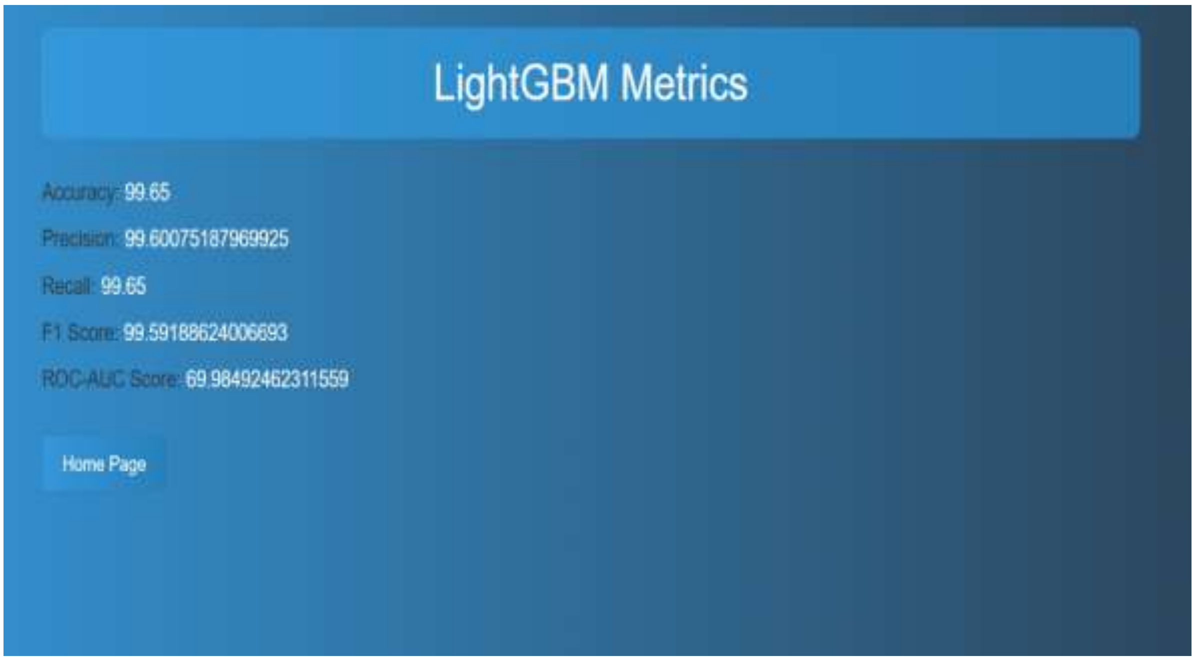
Recall: 99.95

F1 Score: 99.9486904903555

ROC-AUC Score: 99.54773869346734

Fig 6.3: Output screen of XGBoost

➢ After the result of XGBoost we have to click on Home page then for final test the splitted date is run on LifhtGBM .(click-Run LightGBM)


LightGBM Metrics

Accuracy: 99.65

Precision: 99.60075187969925

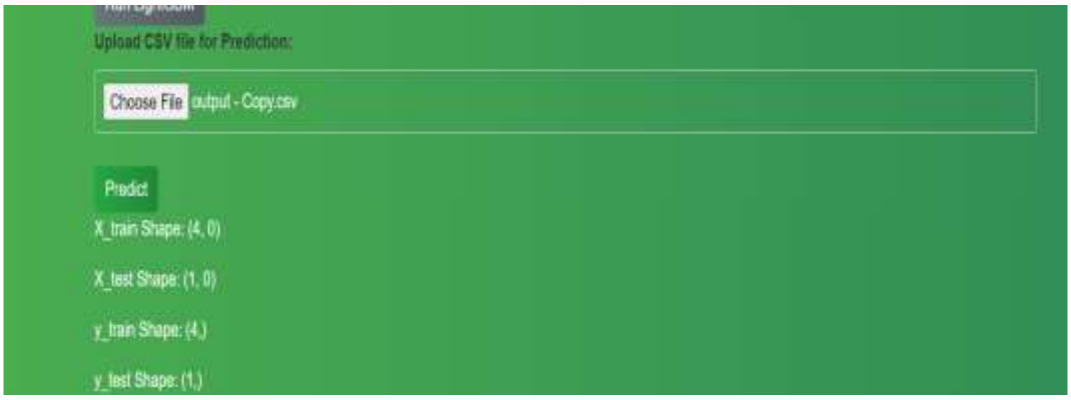
Recall: 99.65

F1 Score: 99.59188624006693

ROC-AUC Score: 69.98492462311559

Fig 6.4:Output screen of LightGBM

- After running the data in three algorithms then we have to predict the output by camparing those three algorithms
- For predication choose the output.copy.CSV file from files
- Click on predict



Upload CSV file for Prediction:

output - Copy.csv

X_train Shape: (4, 0)

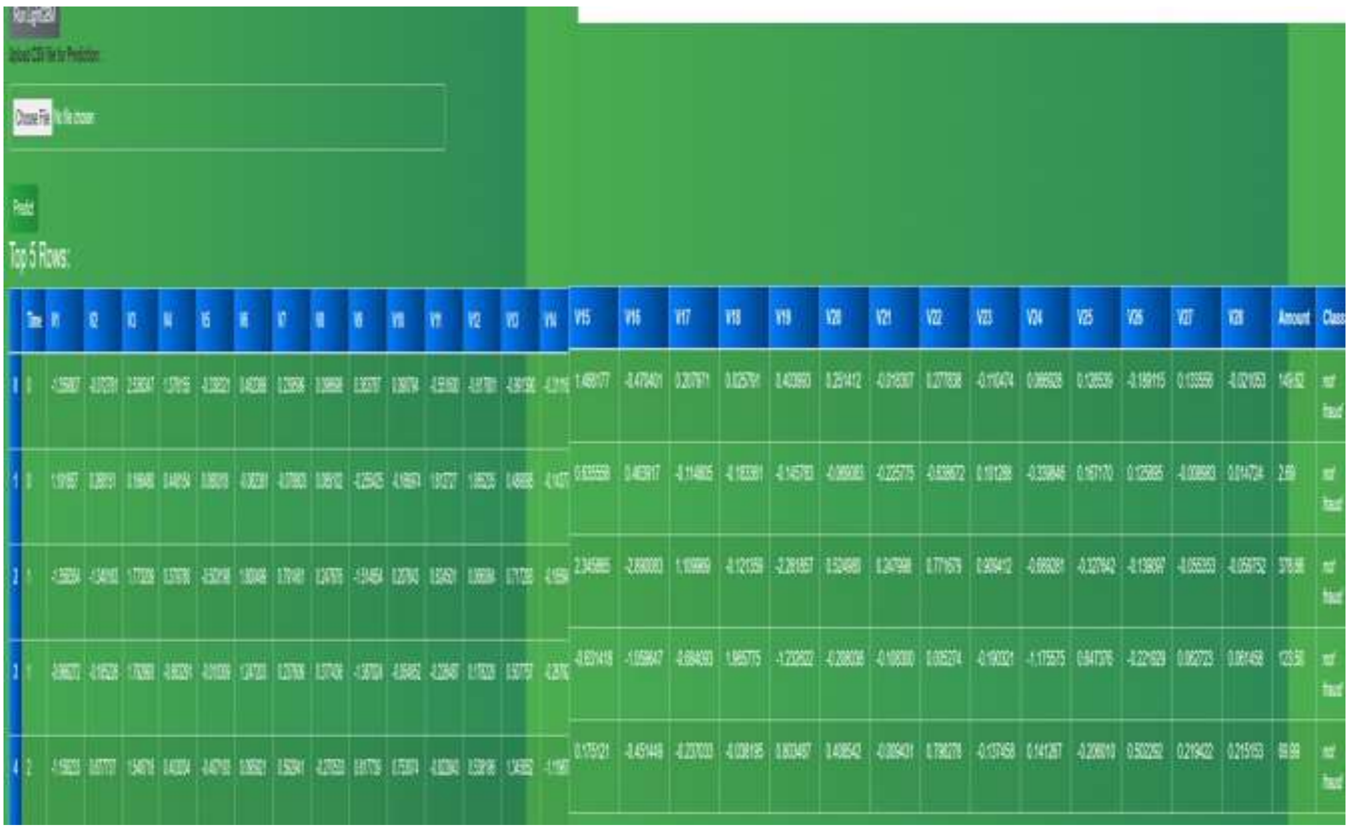
X_test Shape: (1, 0)

y_train Shape: (4,)

y_test Shape: (1,)

Fig 6.5:Predecting the algorithms

- The final output window is displayed showing whether the uploaded data is fraud and non fraud



Upload CSV file for Prediction:

No file chosen

Top 5 Rows:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
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	-0.311169
1	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	-0.143772
2	1	-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	-0.165946
3	1	-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	-0.287924
4	2	-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	-1.119670

	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	1.468177	-0.470401	0.207971	0.025791	0.403993	0.251412	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	not fraud
1	0.635558	0.463917	-0.114805	-0.183361	-0.145783	-0.069083	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	not fraud
2	2.345865	-2.890083	1.109969	-0.121359	-2.261857	0.524980	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	not fraud
3	-0.631418	-1.059647	-0.684093	1.965775	-1.232622	-0.208038	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	not fraud
4	0.175121	-0.451449	-0.237033	-0.038195	0.803487	0.408542	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	not fraud

Fig 6.6: Output Window

CHAPTER-7

APPLICATIONS ADVANTAGES AND LIMITATIONS

7.1 Applications

Anomaly Detection:

Anomaly detection in transaction data is critical for fraud detection. Using clustering, isolation forests, or autoencoders, banks identify unusual patterns like large transactions or unfamiliar locations, helping to prevent fraudulent activity.

Classification of Fraudulent Transactions:

Classification models, such as logistic regression, decision trees, and ensemble methods, classify transactions as fraudulent or legitimate based on features like amount, frequency, time, and user behavior, enhancing fraud detection.

Real-time Monitoring:

Real-time monitoring uses machine learning to analyze transaction data instantly, detecting suspicious activities and triggering alerts to prevent fraudulent transactions, crucial for securing bank payments and minimizing losses.

7.2 Advantages

- Real-time fraud detection capability.
- Continuous model evolution.
- Pattern detection capabilities.
- Cost-effective fraud detection.
- Enhanced security and trust.

7.3 Limitations

- Risk of false positives inconvenience.
- Privacy concerns with data analysis.
- Risk of oversight gaps.

CONCLUSION

In this project, we studied the credit card fraud detection problem in real unbalanced datasets. We proposed a machinelearning approach to improve the performance of fraud detection. We used a publicly available ‘‘credit card’’ dataset with 28 features and 0.17 percent of the fraud data. We proposed two methods. In the proposed LightGBM, we used class weight tuning to choose the proper hyperparameters. We used the common evaluation metrics, including accuracy, precision, recall, F1-score, and AUC. Our experimental results showed that the proposed LightGBM method improved the fraud detection cases by 50% and the F1-score by 20% compared with the recently presented method in [17]. We improve the performance of the algorithm with the help of the majority voting algorithm. We also improved the criteria by using the deep learning method. The assurance of the results of MCC for unbalanced data proved that, compared to other criteria of evaluation, it’s stronger. In this paper, by combining the LightGBM and XGBoost methods, we obtained 0.79 and 0.81 for the deep learning method. Using hyper parameters to address data unbalance compared to sampling methods, in addition to reducing memory and time needed to evaluate algorithms, also has better results.For future studies and work, we propose using other hybrid models as well as working specifically in the field of CatBoost by changing more hyperparameters, especially the hyperparameter number of trees. Also, due to hardware limitations in this study, the use of stronger and better hardware may bring better results that can ultimately be compared with the results of this study.

BIBLIOGRAPHY

[1] E. F. Malik, K. W. Khaw, B. Belaton, W. P. Wong, and X. Chew, ‘‘Credit card fraud detection using a new hybrid machine learning architecture,’’ Mathematics, vol. 10, no. 9, p. 1480, Apr. 2022.

[2] K. Gupta, K. Singh, G. V. Singh, M. Hassan, G. Himani, and U. Sharma, ‘‘Machine learning based credit card fraud detection—A review,’’ in Proc. Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC), 2022, pp. 362–368.

[3] R. Almutairi, A. Godavarthi, A. R. Kotha, and E. Ceesay, ‘‘Analyzing credit card fraud detection based on machine learning models,’’ in Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS), Jun. 2022

[4] A. A. Taha and S. J. Malebary, ‘‘An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine,’’ IEEE Access, vol. 8, pp. 25579–25587, 2020.

[5] H. Feng, ‘‘Ensemble learning in credit card fraud detection using boosting methods,’’ in Proc. 2nd Int. Conf. Comput. Data Sci. (CDS), Jan. 2021, pp. 7–11.

[6] I. Matloob, S. A. Khan, R. Rukaiya, M. A. K. Khattak, and A. Munir, ‘‘A sequence mining-based novel architecture for detecting fraudulent transactions in healthcare systems,’’ IEEE Access, vol. 10, pp. 48447–48463, 2022.

[7] H. Feng, ‘‘Ensemble learning in credit card fraud detection using boosting methods,’’ in Proc. 2nd Int. Conf. Comput. Data Sci. (CDS), Jan. 2021, pp. 7–11.

[8]J. Nanduri, Y.-W. Liu, K. Yang, and Y. Jia, ‘‘Ecommerce fraud detection through fraud islands and multi-layer machine learning model,’’ in Proc. Future Inf. Commun. Conf., in Advances in Information and Communication. San Francisco, CA, USA: Springer, 2020, pp. 556–570.

[9] U. Porwal and S. Mukund, ‘‘Credit card fraud detection in e-commerce: An outlier detection approach,’’ 2018, arXiv:1811.02196.

[10] A. C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, ‘‘Feature engineering strategies for credit card fraud detection,’’ Expert Syst. Appl., vol. 51, pp. 134–142, Jun. 2016.

[11] T. A. Olowookere and O. S. Adewale, ‘‘A framework for detecting credit card fraud with cost-sensitive meta-learning ensemble approach,’’ Sci. Afr., vol. 8, Jul. 2020, Art. no. e00464.