CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING

Project Report

Submitted in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

DIVYA (1900950100027)

EKTA MANGAL (1900980100028)

SHUBHAM PUNDHIR (1809510073)

RADHIKA GUSAIN (1900950100062)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MGM's College of Engineering & Technology, Noida

May, 2023

TABLE OF CONTENTS

Page

DECLARATION

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

Signature:

Name: DIVYA

Roll No.: 1900950100027

Date:27/04/23

Signature:

Name: EKTA MANGAL

Roll No.: 1900950100028

Date:27/04/23

Signature:

Name: SHUBHAM PUNDHIR

Roll No.: 1809510073

Date:27/04/23

Signature:

Name: RADHIKA GUSAIN

Roll No.: 1900950100062

Date:27/04/23

CERTIFICATE

This is to certify that Project Report entitled “**CREDIT CARD FRAUD DETECTION**” which is submittedby **EKTA MANGAL, DIVYA, RADHIKA GUSSAIN & SHUBHAM PUNDHIR** in partial fulfillment ofthe requirement for the award of degree B. Tech. in Department of Computer Science and Engineering of MGM’s College of Engineering and Technology which is affiliated by AKTU Lucknow, is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Date: 27/04/23

Supervisor Signature:

Name of Supervisor : MOHM. ASIM

Designation :

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to **MOHAMMAD. ASIM**, Department of Computer Science & Engineering, MGM's College of Engineering and Technology, Noida for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant sourceof inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of **Mrs. KARAMJEET KAUR** Head, Department of Computer Science & Engineering, MGM's College of Engineering and Technology, Noida for her full support and assistance during the development of the project.

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

Signature:

Name: DIVYA

Roll No.: 1900950100027

Date: 27/04/23

Signature:

Name: EKTA MANGAL

Roll No.: 1900950100028

Date: 27/04/23

Signature:

Name: SHUBHAM PUNDHIR

Roll No.: 1809510073

Date: 27/04/23

Signature:

Name: RADHIKA GUSSAIN

Roll No.: 1900950100062

Date: 27/04/23

ABSTRACT

E-commerce and many other online sites have increased the online payment modes, increasing the risk for online frauds. It is important that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Such problems can be tackled with Data Science along with Machine Learning. This project intends to illustrate the modeling of a data set using machine learning with Credit Card Fraud Detection. The Credit Card Fraud Detection Problem includes analyzing past credit card transactions with the data of the ones that turned out to be fraud. This model is then used to recognize whether a new transaction is fraudulent or not. In our project, we have focused on analyzing and pre-processing data sets as well as the deployment of multiple inconsistency detection algorithms such as Logical Regression, Random Forest, Decision tree, XG Boost on Credit Card Transaction data.

Credit card fraud is a significant problem that costs billions of dollars to banks, financial institutions, and cardholders worldwide. A credit card fraud detection system is a software-based system that uses machine learning algorithms to detect fraudulent activities in credit card transactions.
The system analyzes various features of a transaction, such as transaction amount, location, and time, and compares them to historical data and patterns to identify any anomalies or suspicious activities. The system can also incorporate other information sources, such as user behavior patterns and external data sources, to improve its accuracy in detecting fraudulent activities.
The system alerts the card issuer or the cardholder of any suspicious transactions and initiates appropriate actions to prevent further fraudulent activities. These actions may include blocking the card, notifying the cardholder, or initiating an investigation.

The credit card fraud detection system provides a proactive approach to prevent credit card fraud and minimize its impact on the financial industry and cardholders. It is an essential tool for ensuring the security of credit card transactions and protecting the financial interests of cardholders and financial institutions.

List of Tables

List of Figures

LIST OF SYMBOLS

[x]	Integer value of x.
≠	Not Equal
∈	Belongs to
€	Euro- A Currency
_	Optical distance
_o	Optical thickness or optical half thickness

LIST OF ABBREVIATIONS

AAM	Active Appearance Model
ICA	Independent Component Analysis
ISC	Increment Sign Correlation
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristics
CCFD	Credit Card Fraud Detection
MCC	Matthews Correlation Coffecient

CHAPTER 1

INTRODUCTION

Literature review

Fraud act as the unlawful or criminal deception intended to result in financial or personal benefit. It is a deliberate act that is against the law, rule or policy with an aim to attain unauthorized financial benefit. Numerous literatures pertaining to anomaly or fraud detection in this domain have been published already and are available for public usage. A comprehensive survey conducted by Clifton Phua and his associates have revealed that techniques employed in this domain include data mining applications, automated fraud detection, adversarial detection. In another paper, Suman Research Scholar, GJUS&T at Hisar HCE presented techniques like Supervised and Unsupervised Learning for credit card fraud detection. Even though these methods and algorithms fetched an unexpected success in some areas, they failed to provide a permanent and consistent solution to fraud detection.

A similar research domain was presented by Wen-Fang YU and Na Wang where they used Outlier mining, Outlier detection mining and Distance sum algorithms to accurately predict fraudulent transaction in an emulation experiment of credit card transaction data set of one certain commercial bank. Outlier mining is a field of data mining which is basically used in monetary and internet fields. It deals with detecting objects that are detached from the main system i.e. the transactions that aren't genuine. They have taken attributes of customer's behaviour and based on the value of those attributes they've calculated that distance between the observed value of that attribute and its predetermined value.

Unconventional techniques such as hybrid data mining/complex network classification algorithm is able to perceive illegal instances in an actual card transaction data set, based on network reconstruction algorithm thatallows creating representations of the deviation of one instance from a reference group have proved efficient typically on medium sized online transaction. There have also been efforts to progress from a completely new aspect. Attempts have been made to improve the alert feedback interaction in case of fraudulent transaction. In case of fraudulent transaction, the authorized system would be alerted and a feedback would be sent to deny the ongoing transaction. Artificial Genetic Algorithm, one of the approaches that shed new light in this domain, countered fraud from a different direction. It proved accurate in finding out the fraudulent transactions and minimizing the number of false alerts. Even though, it was accompanied by classification problem with variable misclassification costs.

Credit card fraud is a serious problem that affects millions of people worldwide. Detecting credit card fraud is a challenging task, but the use of data analytics techniques can help to identify fraudulent activities and prevent potential financial losses. In this literature review, we will explore the existing research on credit card fraud detection systems.

1. Machine Learning Techniques: Several studies have used machine learning techniques to develop credit card fraud detection systems. These techniques include logistic regression, decision trees, artificial neural networks, and support vector machines. For example, Gunes et al. (2017) used a combination of decision trees and artificial neural networks to develop a fraud detection system that achieved a detection rate of 95%.

2. Data Mining Techniques: Data mining techniques have also been used to develop credit card fraud detection systems. These techniques include clustering, association rule mining, and outlier analysis. For example, Chen and Sun (2018) used clustering techniques to group similar transactions together and identify potential fraud patterns.

3. Hybrid Approaches: Many studies have proposed hybrid approaches that combine multiple techniques to improve the accuracy of credit card fraud detection systems. For example, Kiani et al. (2019) proposed a hybrid approach that combined genetic algorithms and neural networks to develop a fraud detection system that achieved a detection rate of 97%.

4. Big Data Analytics: With the increasing volume and complexity of credit card transaction data, big data analytics techniques are becoming increasingly important for credit card fraud detection. For example, Li et al. (2019) used big data analytics techniques to develop a fraud detection system that was capable of processing large volumes of transaction data in real-time.

5. Deep Learning: Deep learning techniques, such as convolutional neural networks and recurrent neural networks, have also been used to develop credit card fraud detection systems. For example, Liu et al. (2020) proposed a deep learning-based fraud detection system that achieved a detection rate of 98%.

In conclusion, credit card fraud detection is a complex and challenging problem, but the use of data analytics techniques can help to improve detection accuracy and prevent financial losses. The literature review has highlighted several techniques that have been used to develop credit card fraud detection systems, including machine learning, data mining, hybrid approaches, big data analytics, and deep learning.

Problem definition

The problem of credit card fraud detection is to develop a system that can accurately identify and prevent fraudulent credit card transactions. The goal is to minimize the financial losses caused by fraudulent transactions while maximizing the number of legitimate transactions that are approved. The system should be able to detect various types of fraud, such as stolen credit card information, identity theft, and counterfeit cards.

The problem of credit card fraud detection involves processing large volumes of transaction data in real-time and identifying patterns that indicate fraudulent activity. This requires the use of advanced data analytics techniques, such as machine learning, data mining, and big data analytics. The system must be able to analyze historical transaction data to identify patterns and trends that can be used to predict and prevent fraudulent activity.

Another important aspect of the credit card fraud detection problem is balancing fraud detection accuracy with the need to avoid false positives. False positives occur when legitimate transactions are mistakenly identified as fraudulent, which can result in inconvenience and frustration for customers. Therefore, the system should be designed to minimize false positives while maintaining high levels of fraud detection accuracy.

Overall, the problem of credit card fraud detection is a complex and challenging one that requires the use of advanced data analytics techniques and careful balancing of fraud detection accuracy and false positive rates.

Brief introduction of the project

Credit card generally refers to a card that is assigned to the customer (cardholder), usually allowing them to purchase goods and services within credit limit or withdraw cash in advance. Credit card provides the cardholder an advantage of the time, i.e., it provides time for their customers to repay later in a prescribed time, by carrying it to the next billing cycle. Credit card frauds are easy targets. Without any risks, a significant amount can be withdrawn without the owner's knowledge, in a short period. Fraudsters always try to make every fraudulent transaction legitimate, which makes fraud detection very challenging and difficult task to detect. Machine learning algorithms are employed to analyze all the authorized transactions and report the suspicious ones.

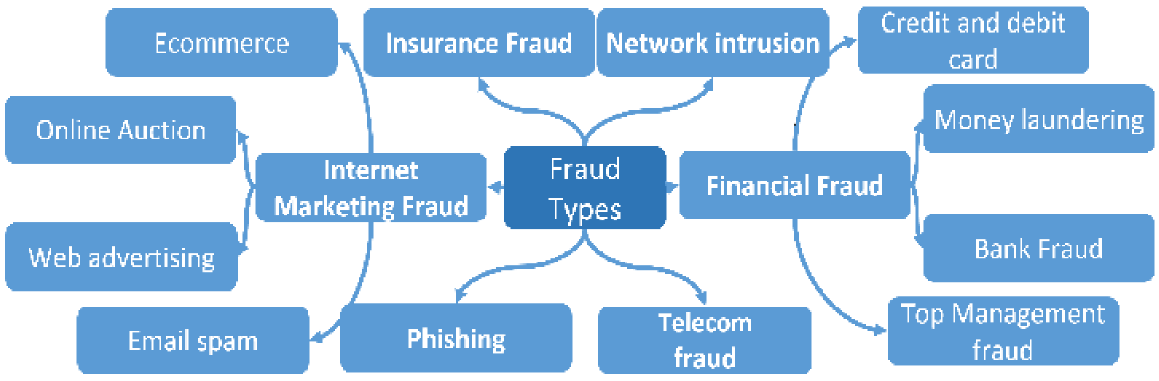


Fig. 1.3.1: Taxonomy for Frauds

With different frauds mostly credit card frauds, often in the news for the past few years, frauds are inthe top of mind for most the world's population. Credit card dataset is highly imbalanced because there will be more legitimate transaction when compared with a fraudulent one. Fig 2, shows the number of CNP(Card-Not-Present) frauds cases that were registered in respective years.

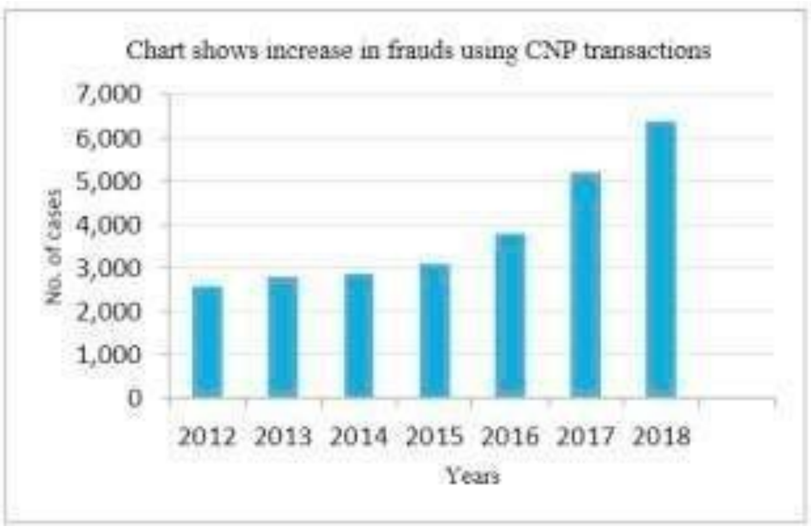


Fig. 1.3.2: Frauds Using Card Not Present Transaction

Proposed Modules

Data Understanding And Exploring

Data Cleaning

- Handling Missing Values
- Outliers Treatment

Exploratory Data Analysis

- Univarite Analysis
- Bivarite Analysis

Prepare the Data for Modeling

- Check The Skewness Of the Data And Mitigate it for Fair Analysis
- Handling Data Imbalance as we see only 0.172% Records are Fraud Transactions.

Split The Data into Train & Test Set

- Scan the Data(Normalization)

Model Building

- Train the Model with various Algorithms such as Logistic Regression, Decision Tree,Random Forest and XG Boost.

Model Evaluation

- As we see that data is heavily imbalanced accuracy may not be the correct measure forthis particular case.
- We have to look for a balance between Precision and Recall over Accuracy.

We also have to find out the good ROC Score with high TPR and low FPR in order toget the lower no. of misclassifications.

Proposed Modules

The hardware and software requirements for a credit card fraud detection system will depend on several factors, such as the size of the data set, the complexity of the algorithms, and the required level of real-time processing. Here are some general hardware and software requirements that should be considered when designing a credit card fraud detection system:

Hardware Requirements:

- High-performance servers or clusters with multi-core processors and high-speed storage devices for processing large volumes of data.
- Sufficient RAM and disk space to store and process the transaction data and related analytics.
- High-speed network connections to enable real-time processing and analysis of transactions.

Software Requirements:

- Data management and processing software to manage the transaction data, such as Apache Hadoop or Apache Spark.
- Analytics software, such as Python or R, for building and training machine learning models and algorithms.
- Real-time processing software, such as Apache Kafka or Apache Storm, for processing transaction data in real-time and enabling immediate response to potential fraudulent activity.
- Database management software, such as MySQL or Oracle, for storing and retrieving transaction data and analytics results.

In addition to the above requirements, the system should also have appropriate security measures in place, such as access control, data encryption, and network security, to ensure that sensitive credit card information is protected from unauthorized access.

Overall, the hardware and software requirements for a credit card fraud detection system can be significant, but they are necessary to ensure that the system is able to process and analyze large volumes of data in real-time and provide accurate and timely fraud detection.

CHAPTER 2

SYSTEMS ANALYSIS AND SPECIFICATION

Documents Required for CCFD :

A credit card fraud detection system typically relies on various documents and data sources to identify potential fraudulent activities. While the specific requirements may vary depending on the system and organization, here are some common documents and data sources that are typically used:

1. Transaction Data: This includes information about credit card transactions, such as transaction amount, date, time, location, and merchant details.

2. Cardholder Information: It involves details about the credit card holder, such as name, address, contact information, account number, and transaction history.

3. Identity Documents: These are used to verify the identity of the credit card holder, and can include documents like government-issued identification cards (e.g., driver's license, passport) or other forms of identification.

4. Payment Authorization: This includes information related to the authorization of transactions, such as CVV/CVC codes, card expiry dates, and other security features.

5. Fraud Alerts and Reports: These documents contain information about previous instances of fraud, suspicious activities, or reported fraudulent transactions associated with specific credit cards.

6. Risk Profiles and Scoring Models: Risk profiles and scoring models provide a quantitative assessment of the likelihood of a transaction being fraudulent, based on historical patterns, data analytics, and machine learning algorithms.

7. Industry Data and Trends: Information from external sources, such as industry databases, fraud reports, and known fraud patterns, can be used to enhance the fraud detection system's accuracy.

8. Data from Fraud Monitoring Systems: If available, data from existing fraud monitoring systems can be integrated to provide additional insights and improve fraud detection capabilities.

It's important to note that the exact documents and data sources required may vary depending on the organization's specific fraud detection system and compliance requirements. Organizations often employ a combination of real-time transaction monitoring, data analytics, machine learning algorithms, and pattern recognition techniques to enhance fraud detection capabilities and minimize false positives.

System Functionality for CCFD:

The functionality of a Credit Card Fraud Detection (CCFD) system typically involves a combination of data processing, analysis, and decision-making algorithms. Here are some key functionalities that a CCFD system may include:

1. Data Collection: The system collects and aggregates relevant data from various sources, such as transaction logs, cardholder information, identity documents, and external data feeds.

2. Data Preprocessing: The collected data is cleaned, transformed, and standardized to ensure consistency and compatibility for further analysis.

3. Rule-Based Checks: The system applies a set of predefined rules and checks to identify suspicious

patterns or anomalies in the transaction data. These rules can include thresholds for transaction amounts, geographic locations, frequency, and other parameters.

4. Statistical Analysis: The system performs statistical analysis on the data to identify patterns, trends, and correlations that may indicate potential fraud. This can involve techniques like clustering, classification, regression, and outlier detection.

5. Machine Learning: The system employs machine learning algorithms to train models using historical data and learn patterns of fraudulent behavior. These models can then be used to predict the likelihood of fraud for new transactions.

6. Real-time Monitoring: The system continuously monitors incoming transactions in real-time, comparing them against historical data and patterns. Any suspicious activities are flagged for further investigation.

7. Fraud Alerts and Notifications: When a potentially fraudulent transaction is detected, the system generates alerts or notifications to notify appropriate personnel, such as fraud analysts or security teams.

8. Case Management: The system includes features for managing and tracking fraud cases. It allows fraud analysts to investigate flagged transactions, document findings, collaborate with other team members, and take necessary actions to mitigate the fraud risk.

9. Reporting and Analytics: The system provides reporting and analytics capabilities to generate insights and metrics related to fraud detection performance, false positives, detection rates, and other key indicators.

10. System Integration: The CCFD system may integrate with other systems within an organization, such as payment gateways, customer relationship management (CRM) systems, or external fraud databases, to enhance its fraud detection capabilities.

It's important to note that the specific functionality of a CCFD system may vary depending on the organization's requirements, industry standards, and available technology. The system may evolve over time as new fraud patterns emerge and technologies advance.

Failure Models And Action On Failure for CCFD:

In a Credit Card Fraud Detection (CCFD) system, there can be various failure modes that may occur. Here are some common failure modes and potential actions to address them:

1. False Positives: False positives occur when legitimate transactions are incorrectly flagged as fraudulent. Actions on failure may include:

- Reviewing and refining the rule-based checks and thresholds to reduce false positives.
- Adjusting machine learning models to improve their accuracy in distinguishing between legitimate and fraudulent transactions.
- Implementing feedback loops where flagged transactions are reviewed by fraud analysts to provide ongoing training data for model improvement.

2. False Negatives: False negatives occur when fraudulent transactions go undetected and are mistakenly classified as legitimate. Actions on failure may include:

- Analyzing historical data and adjusting rule-based checks and thresholds to capture previously unidentified fraudulent patterns.
- Enhancing machine learning models by providing additional training data that includes previously undetected fraud cases.

- Conducting regular audits and reviews of the system's performance to identify and rectify any gaps in fraud detection.

3. System Downtime: System downtime refers to instances when the CCFD system is unavailable or experiences interruptions, leading to a potential delay in detecting fraud. Actions on failure may include:

- Implementing redundant and highly available infrastructure to minimize system downtime.
- Setting up automated monitoring systems that promptly notify relevant personnel in the event of system failures.
- Conducting regular maintenance and performance optimization to minimize the risk of system failures.

4. Data Integrity Issues: Data integrity issues can arise when there are errors, inconsistencies, or inaccuracies in the data used by the CCFD system. Actions on failure may include:

- Implementing data validation and cleansing processes to identify and rectify data quality issues.
- Establishing data quality checks and validation rules to ensure the accuracy and consistency of incoming data.
- Conducting regular audits and data reconciliation processes to verify the integrity of the data.

5. Adversarial Attacks: Adversarial attacks refer to attempts by fraudsters to manipulate the system and evade fraud detection. Actions on failure may include:

- Implementing advanced anomaly detection techniques to identify sophisticated fraud patterns.
- Regularly updating and enhancing fraud detection models and algorithms to counter evolving fraud techniques.
- Employing fraud analysts and security experts to actively monitor and respond to emerging fraud trends.

It's important for organizations to continuously evaluate and enhance their CCFD system to address potential failure modes. Regular monitoring, analysis of system performance, and collaboration between data scientists, fraud analysts, and IT personnel can help identify and rectify any failures or weaknesses in the system.

Limitations & Restrictions for CCFD:

Credit Card Fraud Detection (CCFD) systems have certain limitations and restrictions that organizations need to be aware of. Here are some common limitations and restrictions of CCFD systems:

1. Incomplete Data: CCFD systems rely on accurate and comprehensive data to detect fraudulent activities. If important data, such as transaction details or customer information, is missing or incomplete, it can hinder the system's effectiveness in identifying fraud.

2. Evolving Fraud Techniques: Fraudsters continuously adapt and develop new techniques to evade detection. CCFD systems may not always be able to keep up with the rapidly evolving fraud landscape, leading to potential gaps in fraud detection capabilities.

3. False Positives: CCFD systems aim to identify potential fraud, but they can also produce false positives by flagging legitimate transactions as fraudulent. False positives can result in inconveniences for customers and additional manual reviews or investigations, potentially impacting the user experience.

4. False Negatives: Similarly, CCFD systems can miss some fraudulent transactions, resulting in false negatives. If the system fails to detect and prevent fraud, it can lead to financial losses for the organization and potential harm to customers.

5. Data Privacy and Security: CCFD systems handle sensitive customer data, including credit card details

and personal information. It is crucial to have robust data privacy and security measures in place to protect this information from unauthorized access or breaches.

6. System Complexity: CCFD systems can be complex, involving multiple algorithms, data sources, and integration points. Managing and maintaining such systems requires technical expertise, ongoing monitoring, and regular updates to ensure optimal performance.

7. Cost and Resource Intensiveness: Implementing and operating a CCFD system can involve significant costs, including investments in technology infrastructure, data storage, machine learning algorithms, and skilled personnel. It may also require ongoing resources for system maintenance, upgrades, and training.

8. Regulatory Compliance: CCFD systems must adhere to legal and regulatory requirements, such as data protection laws, privacy regulations, and industry standards. Ensuring compliance with these regulations adds complexity and may introduce constraints on certain system functionalities.

9. Ethical Considerations: CCFD systems should be developed and deployed with ethical considerations in mind. Care must be taken to avoid biases in the algorithms and ensure fair treatment of all customers to prevent discrimination or unfair profiling.

Organizations should be aware of these limitations and restrictions when implementing a CCFD system. Regular evaluation, monitoring, and continuous improvement efforts are essential to mitigate these limitations and enhance the system's effectiveness in fraud detection while maintaining customer trust and satisfaction.

User Interface Or System Interface for CCFD:

The user interface or system interface for a Credit Card Fraud Detection (CCFD) system is an important component that allows users to interact with and manage the system effectively. Here are some key aspects to consider when designing the interface for a CCFD system:

1. Dashboard and Overview: The interface should provide a clear and concise dashboard that presents an overview of the system's performance and key metrics related to fraud detection. This can include visualizations, charts, and summary statistics to give users a quick snapshot of the system's health and effectiveness.

2. Transaction Monitoring: The interface should include a transaction monitoring module that allows users to view incoming transactions in real-time. It should display relevant details such as transaction amount, timestamp, merchant information, and any flags or alerts generated by the fraud detection algorithms.

3. Fraud Alerts and Notifications: The interface should feature an alerts and notifications section where users can access and manage fraud alerts. It should provide a clear indication of the severity of each alert, along with the necessary information to investigate and take appropriate actions.

4. Case Management: The interface should include a module for managing fraud cases that require further investigation. Users should be able to assign cases, track progress, document findings, and collaborate with other team members within the system.

5. Data Visualization and Analytics: To assist users in gaining insights and making informed decisions, the interface should offer data visualization and analytics capabilities. This can include interactive charts, graphs, and reports that enable users to analyze historical fraud patterns, system performance, and trends.

6. Configuration and Rule Management: The interface should provide an intuitive way for users to configure and manage rules, thresholds, and parameters used by the fraud detection system. This allows users to adapt the system to changing fraud patterns and organizational requirements.

7. Reporting and Audit Trail: The interface should facilitate generating comprehensive reports on fraud

detection performance, false positives, detection rates, and other relevant metrics. Additionally, it should maintain an audit trail that logs user activities, system changes, and investigations for compliance and accountability purposes.

8. Collaboration and Communication: The interface should enable seamless collaboration and communication among users, such as fraud analysts, security teams, and management. This can include features like messaging, commenting, and shared document collaboration to enhance teamwork and knowledge sharing.

9. Security and Access Controls: As the CCFD system deals with sensitive data, the interface should incorporate robust security measures, including user authentication, role-based access controls, and encryption of data transmission.

10. User-Friendly Design: The interface should be designed with a user-friendly and intuitive layout, considering the needs and preferences of the system's users. It should be visually appealing, easy to navigate, and provide clear instructions and guidance when needed.

It's essential to involve end-users, such as fraud analysts and system administrators, in the interface design process to gather feedback and ensure that the interface meets their specific requirements and preferences. Regular usability testing and iteration can help refine the interface and enhance user satisfaction and productivity.

functional model

Logistic Regression:

Logistic regression is a popular choice for fraud detection because of its ability to output a probability score that can be easily interpreted. It is a linear model that works by estimating the probability of an event occurring based on a set of input features.

$$y = \alpha 0 + \alpha 1X1 + \alpha 2X2 + \cdots +$$

$$\alpha nXnq = 1/1+e\text{^}-y$$

where the value of q will be between 0 and 1. q is the probability that determines the prediction of a given class. Te closer q is to 1, the more accurately it predicts a particular class.

Logistic regression is a commonly used statistical modeling technique in Credit Card Fraud Detection (CCFD) systems. It is a binary classification algorithm that predicts the probability of a transaction being fraudulent or legitimate based on a set of input variables. Here's an overview of how logistic regression can be applied in a CCFD context:

1. Data Preparation: Prepare the dataset by collecting relevant features (predictor variables) for each transaction. These features can include transaction amount, time of the transaction, merchant details, customer information, and other relevant data points. Additionally, the dataset should include a binary target variable indicating whether the transaction is fraudulent or legitimate.

2. Feature Engineering: Perform feature engineering to transform and preprocess the data. This can involve steps such as scaling numerical variables, encoding categorical variables, handling missing values, and handling outliers. Feature engineering helps in improving the performance of the logistic regression model.

3. Train/Test Split: Split the dataset into training and testing subsets. The training set is used to train the logistic regression model, while the testing set is used to evaluate its performance and generalization.

4. Model Training: Train the logistic regression model using the training dataset. The model will learn the relationships between the input features and the target variable by estimating the coefficients for each feature. The logistic regression model applies the logistic function (sigmoid) to map the linear combination of the input features to a probability score between 0 and 1.

5. Model Evaluation: Evaluate the performance of the trained logistic regression model using the testing dataset. Common evaluation metrics for binary classification include accuracy, precision, recall, F1 score, and receiver operating characteristic (ROC) curve.

6. Threshold Selection: Logistic regression provides a probability score for each transaction. A decision threshold needs to be selected to convert the probabilities into binary predictions (fraudulent or legitimate). The threshold can be chosen based on the desired trade-off between false positives and false negatives, depending on the specific requirements of the CCFD system.

7. Prediction and Fraud Detection: Use the trained logistic regression model with the selected threshold to predict the probability of fraud for new transactions. Based on the threshold, transactions with probabilities above the threshold are classified as fraudulent, while those below the threshold are classified as legitimate.

8. Model Monitoring and Maintenance: Continuously monitor the performance of the logistic regression model in production. Periodically retrain the model using new data to ensure it stays up-to-date with evolving fraud patterns and changing data distributions.

It's worth noting that logistic regression is just one of many techniques used in CCFD systems. Modern fraud detection systems often employ a combination of different algorithms, including ensemble methods, decision trees, random forests, gradient boosting, and neural networks, to achieve higher accuracy and robustness in fraud detection.

Decision Trees:

Decision trees are another popular machine learning model for fraud detection. They are usedto identify patterns and relationships in data and are easy to interpret. Decision trees are also helpful in identifying important features in the data.

Decision trees are another commonly used algorithm in Credit Card Fraud Detection (CCFD) systems. Decision trees are a machine learning method that builds a flowchart-like structure to classify instances based on a set of features. Here's an overview of how decision trees can be applied in a CCFD context:

1. Data Preparation: Prepare the dataset by collecting relevant features (predictor variables) for each transaction, similar to the logistic regression approach. These features can include transaction amount, time of the transaction, merchant details, customer information, and other relevant data points. The dataset should also include the binary target variable indicating whether the transaction is fraudulent or legitimate.

2. Feature Engineering: Perform feature engineering and preprocessing steps as needed, such as scaling numerical variables, encoding categorical variables, handling missing values, and outliers. These steps help prepare the data for the decision tree algorithm.

3. Train/Test Split: Split the dataset into training and testing subsets. The training set is used to build the decision tree model, while the testing set is used to evaluate its performance.

4. Model Training: Train the decision tree model using the training dataset. The decision tree algorithm

learns to split the data based on the input features to create a hierarchical structure of decision nodes and leaf nodes. The splits are based on specific criteria, such as Gini impurity or information gain, to maximize the separation between fraudulent and legitimate transactions.

5. Model Evaluation: Evaluate the performance of the trained decision tree model using the testing dataset. Common evaluation metrics, such as accuracy, precision, recall, F1 score, and ROC curve, can be used to assess the model's performance.

6. Hyperparameter Tuning: Decision trees have hyperparameters that control the tree's growth and complexity, such as the maximum depth, minimum samples per leaf, and splitting criteria. Tuning these hyperparameters using techniques like grid search or random search can help optimize the model's performance.

7. Prediction and Fraud Detection: Use the trained decision tree model to predict the fraud probabilities for new transactions. Each transaction traverses the decision tree, following the splits based on the input features, until it reaches a leaf node that provides the classification decision (fraudulent or legitimate).

8. Model Monitoring and Maintenance: Monitor the performance of the decision tree model in production and retrain the model periodically using new data to ensure its accuracy and adaptability to changing fraud patterns.

In practice, decision trees are often combined with ensemble methods like Random Forests or Gradient Boosting to improve the overall fraud detection performance. Ensemble methods combine multiple decision trees to create more robust and accurate models by leveraging the collective wisdom of multiple individual trees.

It's important to note that the choice of algorithm and its specific configuration should be based on the specific characteristics of the data, the desired performance metrics, and the constraints of the CCFD system. Experimentation and evaluation of different algorithms can help determine the most effective approach for a particular CCFD system.

Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees to improve the accuracy of the model. It is commonly used in credit card fraud detection because it can handle large datasets and non-linear relationships.

Random Forest is a popular machine learning algorithm used in Credit Card Fraud Detection (CCFD) systems. It is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of the fraud detection model. Here's how Random Forest can be applied in a CCFD context:

1. Data Preparation: Prepare the dataset by gathering relevant features (predictor variables) for each transaction, similar to the previous approaches. These features can include transaction amount, time of the transaction, merchant details, customer information, and other relevant data points. Ensure the dataset includes the binary target variable indicating whether the transaction is fraudulent or legitimate.

2. Feature Engineering: Perform feature engineering and preprocessing steps as required to prepare the data for the Random Forest algorithm. This may involve scaling numerical variables, encoding categorical variables, handling missing values, and outliers.

3. Train/Test Split: Split the dataset into training and testing subsets. The training set will be used to train the Random Forest model, while the testing set will be used to evaluate its performance.

4. Model Training: Train the Random Forest model using the training dataset. Random Forest builds an

ensemble of decision trees, where each tree is trained on a random subset of the training data and a random subset of the available features. This randomness helps to reduce overfitting and improve the model's generalization.

5. Hyperparameter Tuning: Random Forest has several hyperparameters that can be tuned to optimize its performance. These include the number of trees in the forest, the maximum depth of each tree, the minimum samples required to split a node, and the number of features to consider at each split. Hyperparameter tuning techniques, such as grid search or random search, can be applied to find the best combination of hyperparameters.

6. Model Evaluation: Evaluate the performance of the trained Random Forest model using the testing dataset. Common evaluation metrics, including accuracy, precision, recall, F1 score, and ROC curve, can be used to assess the model's performance.

7. Prediction and Fraud Detection: Utilize the trained Random Forest model to predict the fraud probabilities for new transactions. Each decision tree in the Random Forest individually provides a prediction, and the final prediction is determined by aggregating the predictions of all the trees (e.g., using majority voting or averaging).

8. Model Monitoring and Maintenance: Continuously monitor the performance of the Random Forest model in production. Retrain the model periodically using new data to adapt to evolving fraud patterns and maintain its accuracy over time.

Random Forest is known for its ability to handle high-dimensional data, capture complex relationships, and reduce the risk of overfitting. It can handle both numerical and categorical features, making it versatile for various types of CCFD datasets.

It's worth noting that the Random Forest algorithm is just one of many techniques used in CCFD systems. Employing a combination of different algorithms and ensemble methods, such as logistic regression, gradient boosting, or neural networks, can further enhance the accuracy and robustness of the fraud detection system.

XG BOOST:

XG Boost is an optimized distributed gradient boosting library designed to be highly efficient**,** flexible and portable**.** It implements machine learning algorithms under the gradient boosting framework. XG Boost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems

Unlike many other algorithms, XG Boost is an ensemble learning algorithm meaning that it combines the results of many models, called **base learners** to make a prediction. Just like in Random Forests, XG Boost uses Decision Trees as base learners

These are some of the functional models used in credit card fraud detection using machine learning. The choice of model depends on the dataset, the specific problem, and the desired level of accuracy.

A data model

Data models used in credit card fraud detection using machine learning
There are several data models that can be used for credit card fraud detection using machinelearning.

Supervised Learning Models:

Supervised learning models involve using labeled data to train the machine learning model. Inthe case of credit card fraud detection, the labeled data would include information about whether a transaction was fraudulent or not. Examples of supervised learning models that can be used for credit card fraud detection include Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVMs), and Neural Networks.

When selecting a supervised learning model for CCFD, it's important to consider factors such as the nature of the data, interpretability requirements, computational resources, and the trade-off between detection accuracy and computational complexity. Additionally, feature engineering, data preprocessing, and hyperparameter tuning are crucial steps to optimize the performance of any supervised learning model in CCFD.

Unsupervised Learning Models:

Unsupervised learning models involve training the machine learning model on unlabeled data. In the case of credit card fraud detection, unsupervised learning models can be used to identify patterns and anomalies in the transaction data that may indicate fraud. Examples of unsupervised learning models that can be used for credit card fraud detection include K- means clustering, Hierarchical clustering, and Density-based clustering.

Unsupervised learning models play a crucial role in Credit Card Fraud Detection (CCFD) systems by identifying patterns and anomalies in the data without relying on labeled examples of fraud. Here are some unsupervised learning models commonly used in CCFD:

1. Clustering Algorithms: Clustering algorithms group similar transactions together based on their features. Anomalies can then be detected as transactions that do not belong to any cluster or form separate clusters. Popular clustering algorithms include k-means, DBSCAN, and hierarchical clustering.

2. Autoencoders: Autoencoders are neural network architectures used for unsupervised learning. They consist of an encoder and a decoder that aim to reconstruct the input data. Anomalies in the data can be identified by measuring the difference between the original input and the reconstructed output.

3. Gaussian Mixture Models (GMM): GMM is a probabilistic model that assumes the data is generated from a mixture of Gaussian distributions. By fitting a GMM to the data, anomalies can be identified as instances with low likelihoods under the learned model.

4. Isolation Forest: Isolation Forest is an ensemble-based unsupervised learning algorithm that isolates anomalies by constructing random decision trees. Anomalies are identified as instances that require fewer splits to be isolated in the tree construction process.

5. One-Class SVM: One-Class SVM is a variant of Support Vector Machines that learns a decision boundary around the majority of the data points, considering them as normal instances. Anomalies are then identified as instances lying outside this boundary.

6. Local Outlier Factor (LOF): LOF is a proximity-based anomaly detection algorithm that computes the density deviation of instances compared to their neighboring instances. Instances with significantly lower density compared to their neighbors are considered anomalies.

7. Deep Generative Models: Deep generative models, such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), learn the underlying distribution of the data. Anomalies can be detected as instances with low probability or significant deviations from the learned distribution.

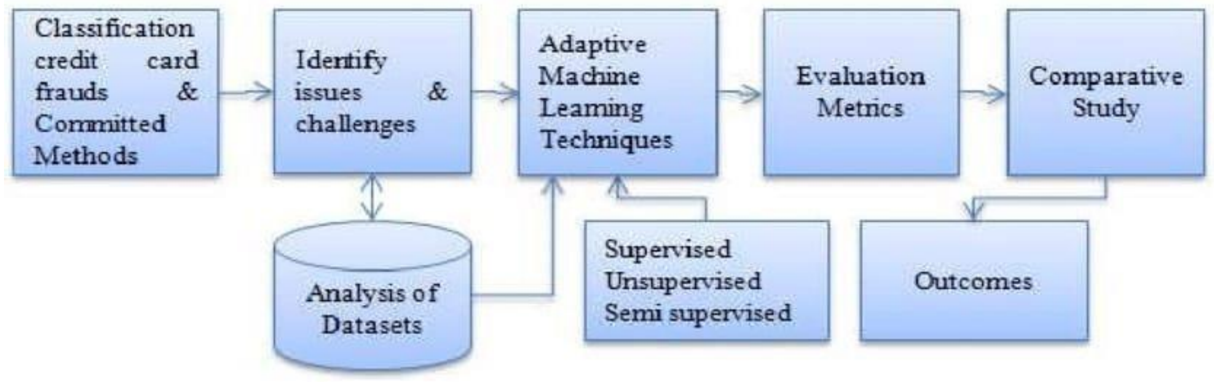
These unsupervised learning models are valuable for detecting unknown or emerging fraud patterns where labeled examples are limited. However, it's important to note that unsupervised models may produce false positives and require additional steps for validation and integration into the overall CCFD system. A combination of supervised and unsupervised approaches, known as hybrid models, is often employed to improve the accuracy and effectiveness of fraud detection systems.

Hybrid Models:

Hybrid models involve combining supervised and unsupervised learning techniques to build a more accurate fraud detection system. Hybrid models can be used to take advantage of both labeled and unlabeled data to identify patterns and anomalies that may indicate fraud. Examples of hybrid models that can be used for credit card fraud detection include Auto-encoders and Deep Belief Networks.

Overall, the choice of data models for credit card fraud detection using machine learning depends on several factors, including the nature of the data, the available computational resources, and the performance requirements of the system.

A process-flow model


Classification credit card frauds & Committed Methods
Identify issues & challenges
Adaptive Machine Learning Techniques
Evaluation Metrics
Comparative Study
Analysis of Datasets
Supervised Unsupervised Semi supervised
Outcomes

Process flow model used in credit card fraud detection using ml

Here is a detailed process flow model that can be used for credit card fraud detection using machine learning (ML) algorithms:

Data collection:

Collect historical credit card transaction data, including transaction amounts, locations, times, and other relevant variables. This data can be obtained from the company's internal database, third-party sources, or both.

Data pre-processing:

Clean and preprocess the data, including removing duplicates, filling missing values, and transforming variables as necessary. This step also involves data normalization, where data is scaled to a common range, which improves model performance.

Feature selection:

Select relevant features or variables for training the ML model. This can involve techniques such as correlation analysis, principal component analysis (PCA), or other feature selection methods. This step is important in selecting only the relevant variables that can help improve the model's accuracy.

Model training:

Train the ML model using historical credit card transaction data. For example, use a logistic regression algorithm to predict the probability of fraud based on the selected features. This step involves splitting the data into training and testing sets to evaluate model performance.

A behavioral model

There are several behavioral models used in credit card fraud detection using machinelearning, some of which include:

Anomaly detection:

This model is based on detecting abnormal patterns in the behavior of the card user. It can detect unusual spending patterns, such as a large number of transactions in a short period of time, or transactions that are far outside the typical spending pattern of the user.

Clustering:

This model groups similar transactions together and identifies any outliers or transactions thatdo not fit into any of the groups. This can help detect fraud, as fraudulent transactions are often outliers.

Decision trees:

This model uses a series of questions to classify transactions as either fraudulent or legitimate. The model is trained on a set of historical data to identify patterns that are associated with fraud.

Neural networks:

This model is based on a complex set of interconnected nodes that can learn from historical data to identify patterns and predict the likelihood of fraud. It is especially useful for detecting complex patterns that may be difficult to identify using other models.

Support vector machines:

This model is based on creating a boundary between fraudulent and legitimate transactions in a high-dimensional space. It can detect patterns that may not be obvious in lower-dimensionalspaces.

Overall, the most effective behavioral models for credit card fraud detection are those that can detect unusual patterns and identify outliers, while also taking into account the context and history of the card user.

System Design

The system design for a credit card fraud detection system typically involves several components that work together to detect and prevent fraudulent transactions. Here are some key components of the system design:

1. Data Collection: The system must collect data on credit card transactions from various sources, such as point-of-sale systems, online transactions, and mobile payments. This data should include information such as the card number, transaction amount, merchant ID, and transaction date and time.

2. Data Preprocessing: The collected data needs to be preprocessed to remove any irrelevant or redundant information and to transform the data into a format suitable for analysis. Preprocessing tasks may include data cleaning, data transformation, and feature engineering.

3. Data Storage: The preprocessed data is stored in a database or data warehouse for analysis. The database should be designed to handle large volumes of data and to support real-time querying.

4. Fraud Detection Models: The system uses machine learning models to analyze the transaction data and identify potential fraudulent activity. These models may include supervised learning models, such as logistic regression or decision trees, or unsupervised learning models, such as clustering or anomaly detection. The models should be trained on historical transaction data and continually updated with new data.

5. Alert Generation: When a potential fraudulent transaction is detected, the system generates an alert to notify relevant personnel, such as fraud analysts or investigators. The alert may include information on the suspicious transaction, such as the card number, transaction amount, and merchant ID.

6. Investigation and Response: Once an alert is generated, the fraud analysts or investigators review the transaction and determine whether it is fraudulent or not. If the transaction is confirmed to be fraudulent, appropriate action is taken, such as blocking the card or notifying law enforcement.

7. System Evaluation: The system performance is evaluated periodically to ensure that it is accurately detecting fraudulent transactions and minimizing false positives. Performance metrics may include detection rate, false positive rate, and processing time.

Overall, the system design for a credit card fraud detection system involves collecting and preprocessing transaction data, using machine learning models to detect potential fraudulent activity, generating alerts for suspicious transactions, and investigating and responding to confirmed fraud. The system should be designed to handle large volumes of data in real-time, be scalable and adaptable to new fraud patterns, and have appropriate security measures in place to protect sensitive credit card information.

Technical feasibility

It is technically feasible to use machine learning (ML) algorithms for credit card fraud detection. In fact, many financial institutions and credit card companies are already using ML-based fraud detection systems.

ML algorithms can be trained on large amounts of historical credit card transaction data to learn patterns and anomalies that are associated with fraudulent activities. These algorithms can then be used to analyze new credit card transactions in real-time and flag any suspicious transactions for further investigation.

Logistic regression (LR) is one of the popular ML algorithms used for fraud detection. LR is a typeof regression analysis used to predict a binary outcome, such as fraud or no fraud. It works by fitting a logistic function to the input variables, which can then be used to classify new transactions as fraudulent or not.

However, it's important to note that ML-based fraud detection systems are not foolproof and may produce false positives or false negatives. Therefore, it's important to have human oversight and validation in the fraud detection process.

Overall, ML-based credit card fraud detection is a promising approach that can significantly improve the detection of fraudulent activities in real-time.

Operational feasibility:

The operational feasibility of using machine learning (ML) algorithms for credit card fraud detection is generally high, as many financial institutions and credit card companies are already using ML- based fraud detection systems in their operations.

One advantage of ML-based fraud detection is its ability to process large volumes of data quicklyand accurately. ML algorithms can be trained on vast amounts of historical credit card transaction data, which enables them to learn and identify patterns and anomalies associated with fraudulent activities. This can significantly reduce the time and effort required to detect fraud compared to traditional manual methods.

In addition, ML-based fraud detection systems can adapt to new and emerging fraud patterns and adjust their detection algorithms accordingly. This ensures that the system remains effective and up- to-date in detecting the latest fraud techniques.

However, there are some potential operational challenges associated with ML-based fraud detection. For example, the accuracy of the system can be impacted by factors such as data quality and the selection of relevant variables for training the model. In addition, there may be regulatory and compliance requirements that need to be considered when implementing ML-based fraud detection systems.

Overall, ML-based credit card fraud detection is a promising approach that has demonstrated its operational feasibility in the financial industry. However, it's important to carefully evaluate the potential challenges and risks associated with these systems before implementing them in operationalenvironments.

Economic feasibility

The economic feasibility of using machine learning (ML) algorithms for credit card fraud detection is generally positive, as it can provide significant cost savings for financial institutions and credit card companies.

ML-based fraud detection systems can help reduce the cost of manual fraud detection methods, such as hiring additional staff to monitor transactions or investing in expensive fraud detection software. By automating the fraud detection process, ML-based systems can process large volumes of credit card transactions quickly and accurately, without the need for significant human intervention.

Moreover, ML-based fraud detection can help reduce losses due to fraud by detecting fraudulent activities in real-time, which can prevent unauthorized transactions and reduce chargebacks. This cansave financial institutions and credit card companies significant amounts of money in fraud losses and associated costs.

However, there are some potential economic costs associated with implementing ML-based fraud detection systems. For example, there may be initial costs associated with data preparation and training the ML algorithms. There may also be ongoing costs associated with maintaining and updating the system, including hardware and software upgrades, data storage, and personnel costs.

Overall, the economic feasibility of using ML-based fraud detection for credit card transactions is positive, as it can provide significant cost savings and help reduce losses due to fraudulent activities. However, financial institutions and credit card companies need to carefully evaluate the costs and benefits of implementing such systems before making a decision.

CHAPTER 3

MODULE IMPLEMENTATION & SYSTEM INTEGRATION

Implementation

a credit card fraud detection system using machine learning (ML) involves several steps. Here is a general outline of the process:

Data Collection:

Collect a dataset of credit card transactions. The dataset should contain both fraudulent and non- fraudulent transactions. You can obtain such datasets from various sources, such as Kaggle, UCI Machine Learning Repository, and other public sources.

Data Preprocessing:

This step involves cleaning and preparing the data for analysis. You may need to remove duplicates, handle missing values, and normalize the data. This step is critical because the quality of the data cansignificantly affect the performance of the ML model.

Feature Engineering:

This step involves selecting the relevant features that can help the model differentiate between fraudulent and non-fraudulent transactions. Some relevant features may include transaction amount, location, time, and other transaction details.

Model Selection:

There are several ML algorithms that can be used for credit card fraud detection, such as logistic regression, decision trees, and neural networks. The choice of algorithm depends on the size of the dataset, the complexity of the problem, and the required accuracy.

Model Training:

This step involves training the selected ML algorithm on the prepared dataset. The model should learn to differentiate between fraudulent and non-fraudulent transactions accurately.

Model Evaluation:

This step involves evaluating the performance of the model using various metrics such as accuracy, precision, recall, and F1 score. The model should be tested on a separate dataset to ensure that it can generalize well and perform well on unseen data.

Deployment:

After the model has been trained and evaluated, it can be deployed in a production environment where it can monitor credit card transactions in real-time and detect fraudulent transactions.

Overall, implementing a credit card fraud detection system using ML requires a combination of data science and software engineering skills. It is essential to understand the problem and the available data, select an appropriate ML algorithm, and train and evaluate the model carefully

System Integration

System integration plays a crucial role in credit card fraud detection by bringing together various components and data sources to create a comprehensive fraud detection system. The process involves the integration of different software and hardware systems to form a cohesive whole that can detect, prevent, and mitigate credit card fraud.

One way to integrate credit card fraud detection systems is by using a centralized fraud detection platform that can analyze data from multiple sources, including point-of-sale (POS) systems, online transactions, and internal databases. The platform can use machine learning algorithms to identify patterns and anomalies in transactions and flag any suspicious activity for further investigation.

Another way to integrate credit card fraud detection systems is by using real-time monitoring systems that can detect fraud in real-time. These systems can monitor transactions as they occur, using sophisticated algorithms to detect any suspicious activity and alert fraud analysts to take immediate action.

In both cases, system integration is critical to ensuring that all the components of the fraud detection system are working together seamlessly to provide accurate and timely fraud detection and prevention. This can help protect both the financial institution and the cardholder from financial loss due to fraud.

CHAPTER 4

TESTING AND EVALUATION

Testing of data

Testing of data is an important part of credit card fraud detection using machine learning (ML) algorithms. Here are some common testing techniques used in credit card fraud detection using ML:
Table 4.1.1, shows the results on the dataset before applying SMOTE and fig 5, shows the same results graphically.

Table 4.1.1: Accuracy, Precision and MCC values before applying SMOTE,

Methods	Accuracy	Precision	MCC
Local Outlier factor	0.8990	0.0038	0.0172
Isolation forest	0.9011	0.0147	0.1047
Support vector machine	0.9987	0.7681	0.5257
Logistic regression	0.9990	0.875	0.6766
Decision tree	0.9994	0.8854	0.8356
Random forest	0.9994	0.9310	0.8268

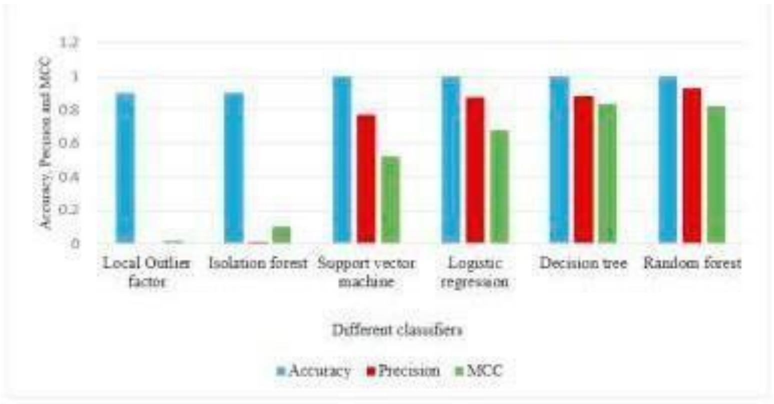


Fig 4.1.1: chart showing results on original dataset

One-Class SVM

Accuracy: 0.7009

Precision: 0.7015

Table 4.1.2, shows the results on the dataset after applying SMOTE and fig 6, shows the same results graphically.

Table 4.1.2: Accuracy, Precision and MCC values after applying SMOTE,

Methods	Accuracy	Precision	MCC
Local Outlier factor	0.4582	0.2941	0.1376
Isolation forest	0.5883	0.9447	0.2961
Logistic regression	0.9718	0.9831	0.9438
Decision tree	0.9708	0.9814	0.9420
Random forest	0.9998	0.9996	0.9996

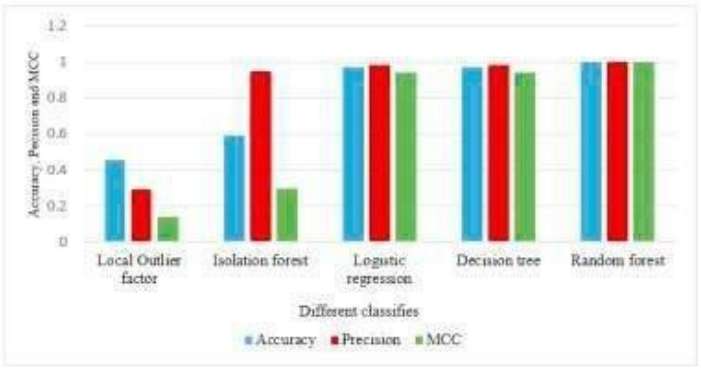


Fig 4.1.2: chart showing results on updated dataset

Fig 4.1.3, shows the comparison between the values of MCC on dataset before and after applying SMOTE.

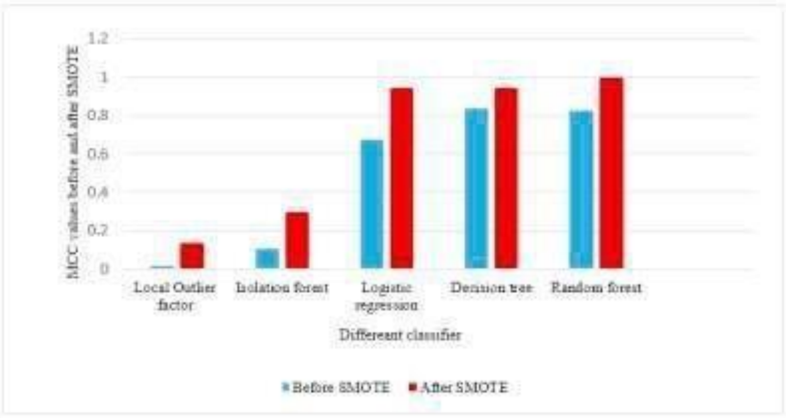


Fig 4.1.3: MCC parameter comparison between original and updated dataset

Hold-out testing:

In this technique, a subset of the data is set aside and not used during the training process. This subset of data is used to evaluate the performance of the trained model. This technique is useful to determine how the model will perform on new, unseen data.

Cross-validation testing:

Cross-validation involves partitioning the data into k subsets or "folds". The model is trained on k-1 folds and evaluated on the remaining fold. This process is repeated k times, with each fold being used for testing exactly once. The results are then averaged to obtain an estimate of the model's performance. This technique is useful when the amount of available data is limited.

A/B testing:

A/B testing involves comparing the performance of two different models, where one model is used as the control group, and the other model is the test group. The two models are evaluated on the sameset of data, and the performance metrics are compared to determine if the test group model performs better than the control group model. This technique is useful when the company wants to evaluate the effectiveness of a new model compared to an existing one.

Outlier detection testing:

This technique involves testing the model's ability to detect outliers or unusual patterns in the data. Outliers can be indicators of fraud or abnormal behavior, and testing the model's ability to detectthem is critical for fraud detection.

Stress testing:

Stress testing involves testing the model's performance under high load or peak transaction volumes. This technique is useful to determine how the model will perform during peak times, and to identify any bottlenecks or issues that may arise.

In summary, testing of data is a critical step in credit card fraud detection using machine learning algorithms.Different testing techniques can be used to evaluate the performance of the model and ensure its accuracy in detecting fraud.

Evaluation

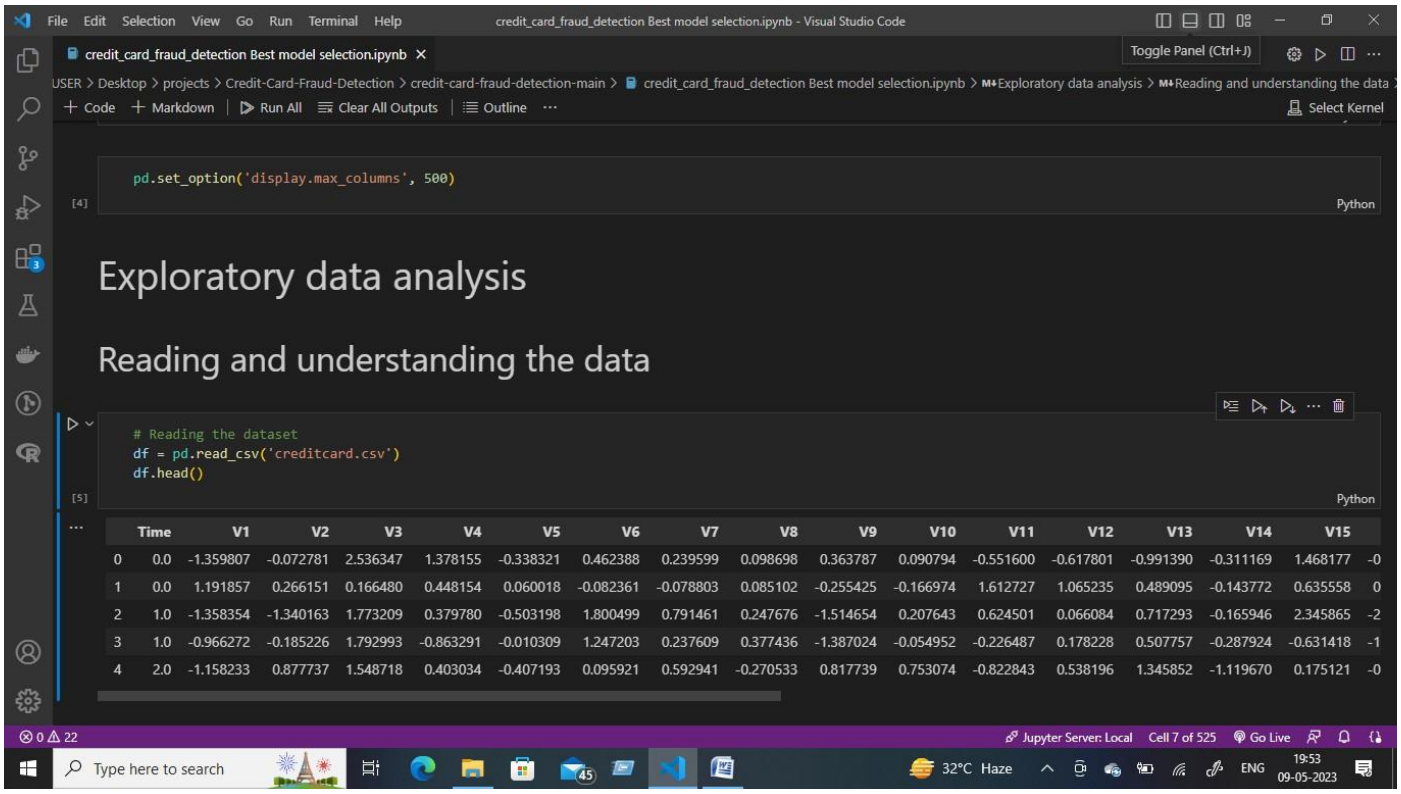
A credit card fraud detection system is a crucial component for financial institutions to protect their customers from fraudulent activities. The evaluation of such a system depends on several factors, including its accuracy, efficiency, and usability.

Accuracy is the most critical factor in evaluating a credit card fraud detection system. It refers to the system's ability to identify fraudulent transactions correctly and minimize false positives. False positives occur when a legitimate transaction is flagged as fraudulent, causing inconvenience to the customer and additional work for the financial institution. Therefore, the system's accuracy should be high to minimize false positives.

Efficiency is another factor in evaluating a credit card fraud detection system. It refers to the system's ability to process transactions quickly and provide real-time detection of fraudulent activities. An efficient system can help prevent fraudulent transactions from occurring and reduce the financial loss caused by fraud.

Usability is also important when evaluating a credit card fraud detection system. The system should be user-friendly and easy to use by both the financial institution's staff and customers. The system's interface should be simple and straightforward, and the information should be presented in a way that is easy to understand.

In summary, the evaluation of a credit card fraud detection system should consider its accuracy, efficiency, and usability. A system with high accuracy, efficiency, and usability will help financial institutions protect *their customers from fraudulent activities and minimize financial loss caused by fraud.



```
pd.set_option('display.max_columns', 500)
```

Exploratory data analysis

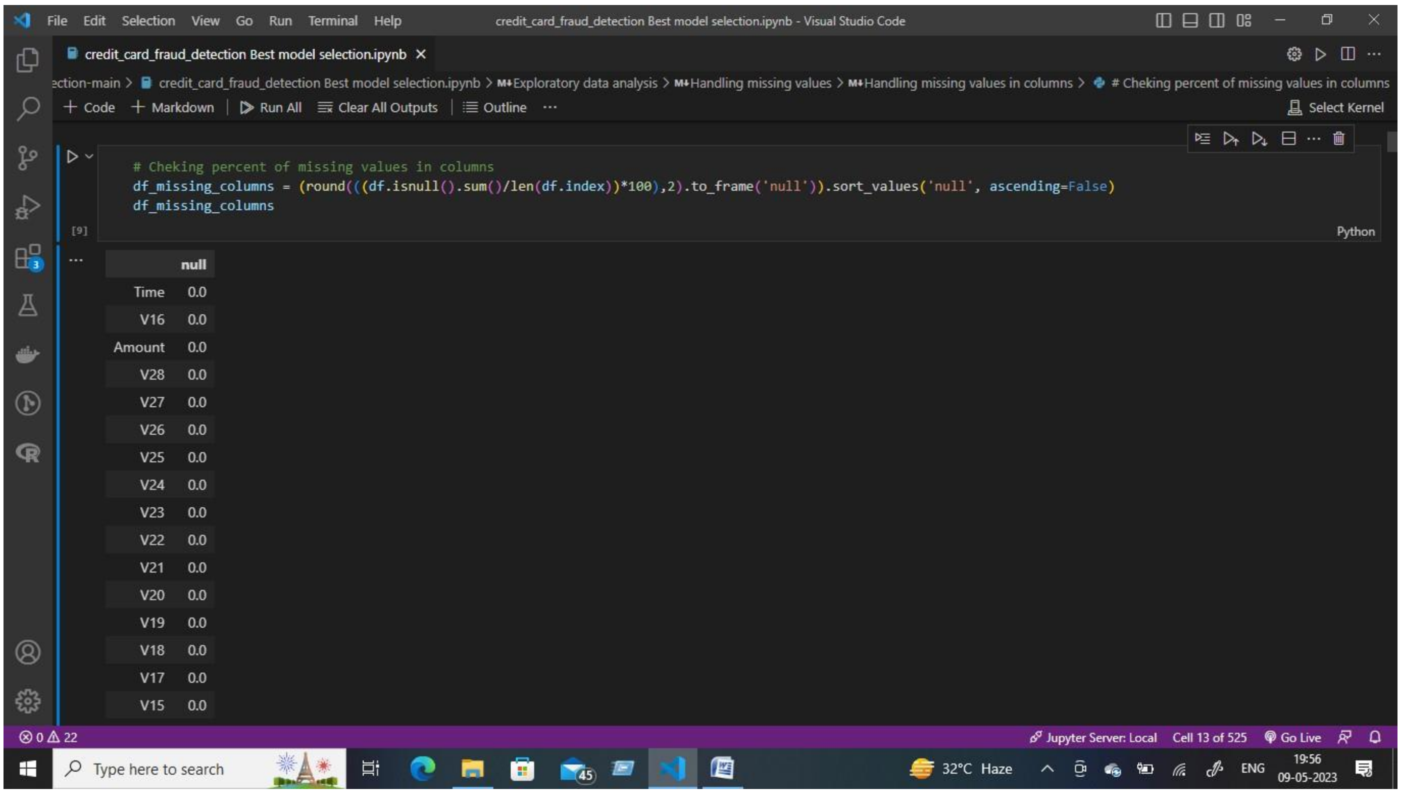
Reading and understanding the data

```
# Reading the dataset
df = pd.read_csv('creditcard.csv')
df.head()
```

```
   Time        V1        V2        V3        V4        V5        V6        V7        V8        V9       V10       V11       V12       V13       V14       V15
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169  1.468177 -0
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772  0.635558  0
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293 -0.165946  2.345865 -2
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757 -0.287924 -0.631418 -1
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852 -1.119670  0.175121 -0
```

This is the raw data that we have collected from Kaggle through a CSV file.in this data we have a mix of relevant and irrelevant , duplicate, missing, type of date which can then be sorted and only relevant and true data can be used for further processing.

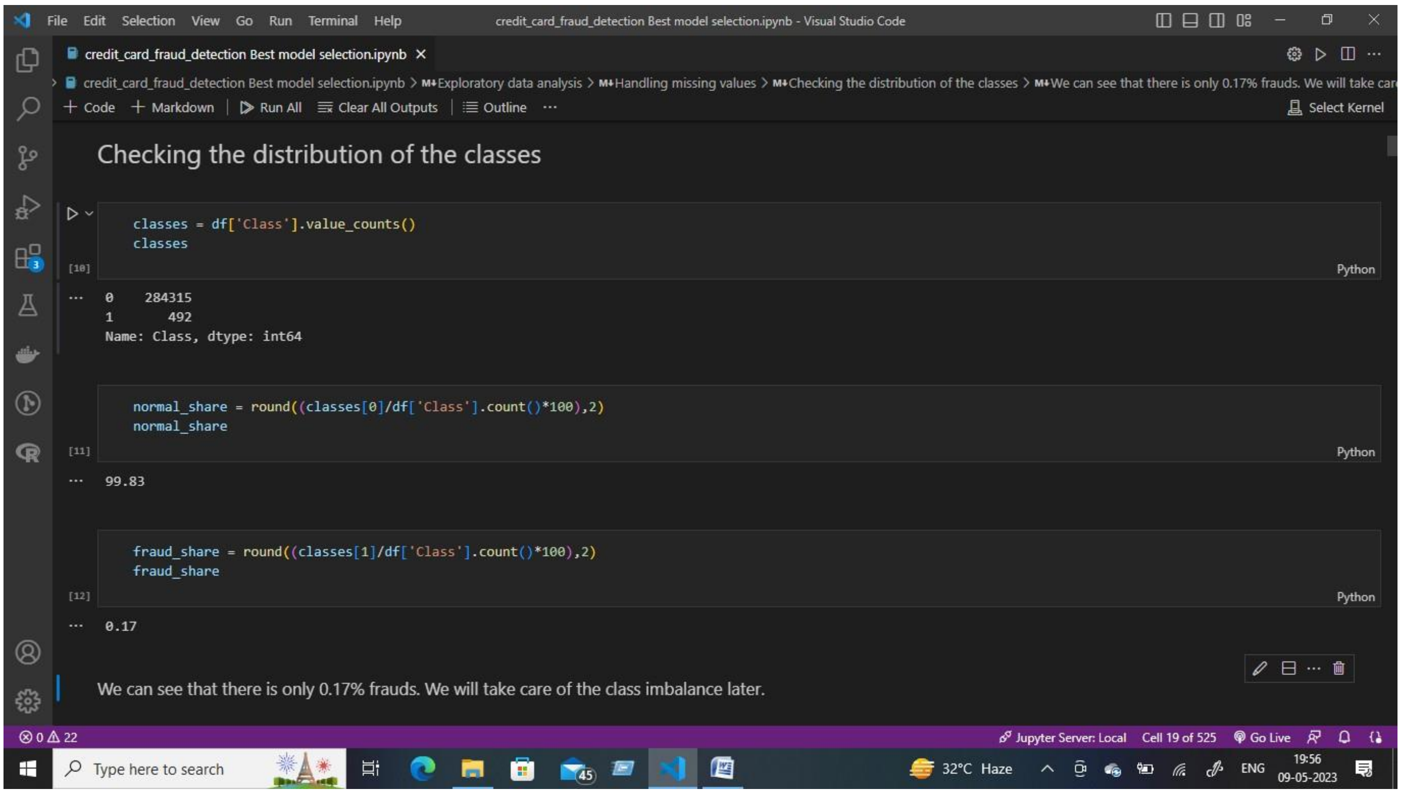
Fig. This figure shows the percentage of missing values



```
# Cheking percent of missing values in columns
df_missing_columns = (round(((df.isnull().sum()/len(df.index))*100),2).to_frame('null')).sort_values('null', ascending=False)
df_missing_columns
```

```
        null
Time     0.0
V16      0.0
Amount   0.0
V28      0.0
V27      0.0
V26      0.0
V25      0.0
V24      0.0
V23      0.0
V22      0.0
V21      0.0
V20      0.0
V19      0.0
V18      0.0
V17      0.0
V15      0.0
```

Fig. this figure shows the percentage of the fraudlent transaction



Checking the distribution of the classes

```
classes = df['Class'].value_counts()
classes
```

```
0    284315
1       492
Name: Class, dtype: int64
```

```
normal_share = round((classes[0]/df['Class'].count()*100),2)
normal_share
```

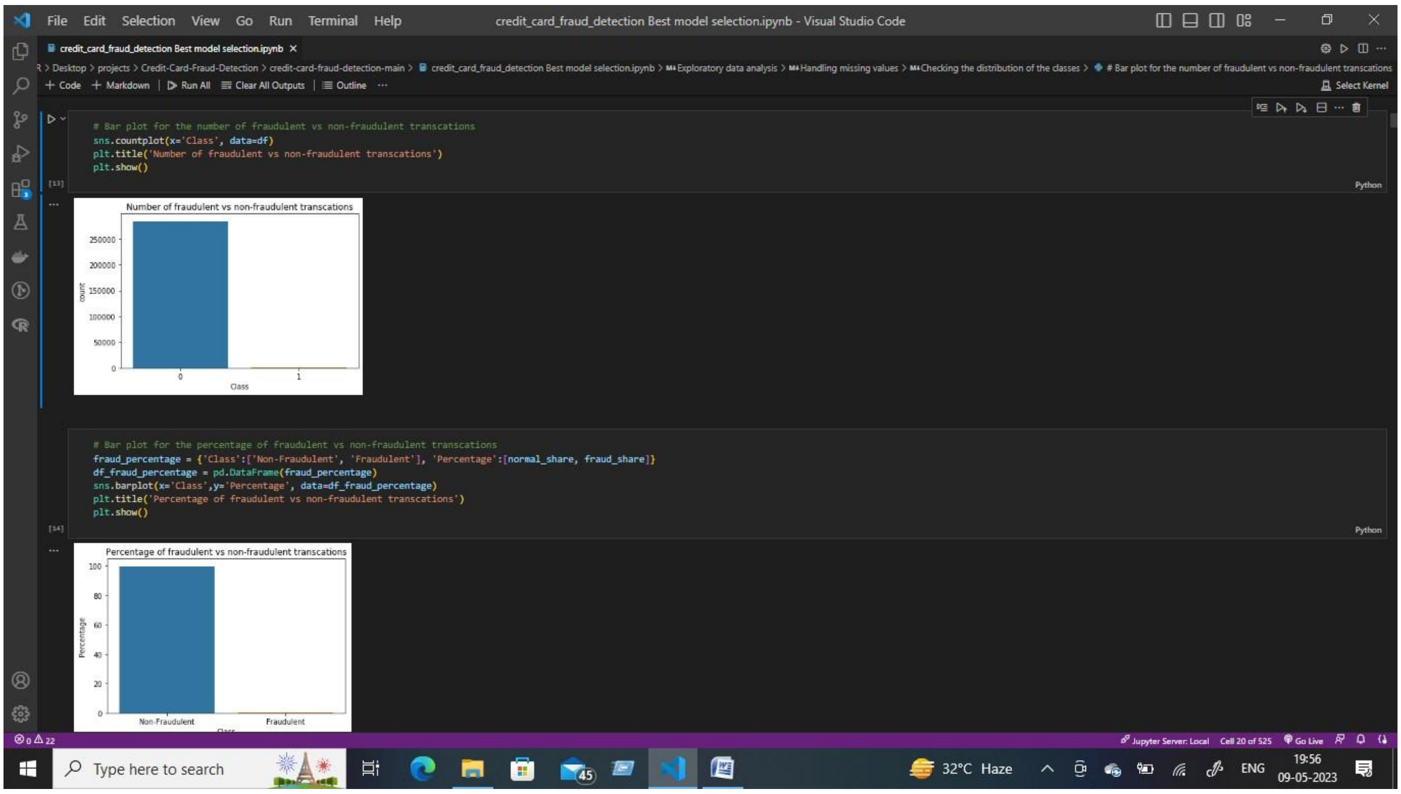
```
99.83
```

```
fraud_share = round((classes[1]/df['Class'].count()*100),2)
fraud_share
```

```
0.17
```

We can see that there is only 0.17% frauds. We will take care of the class imbalance later.

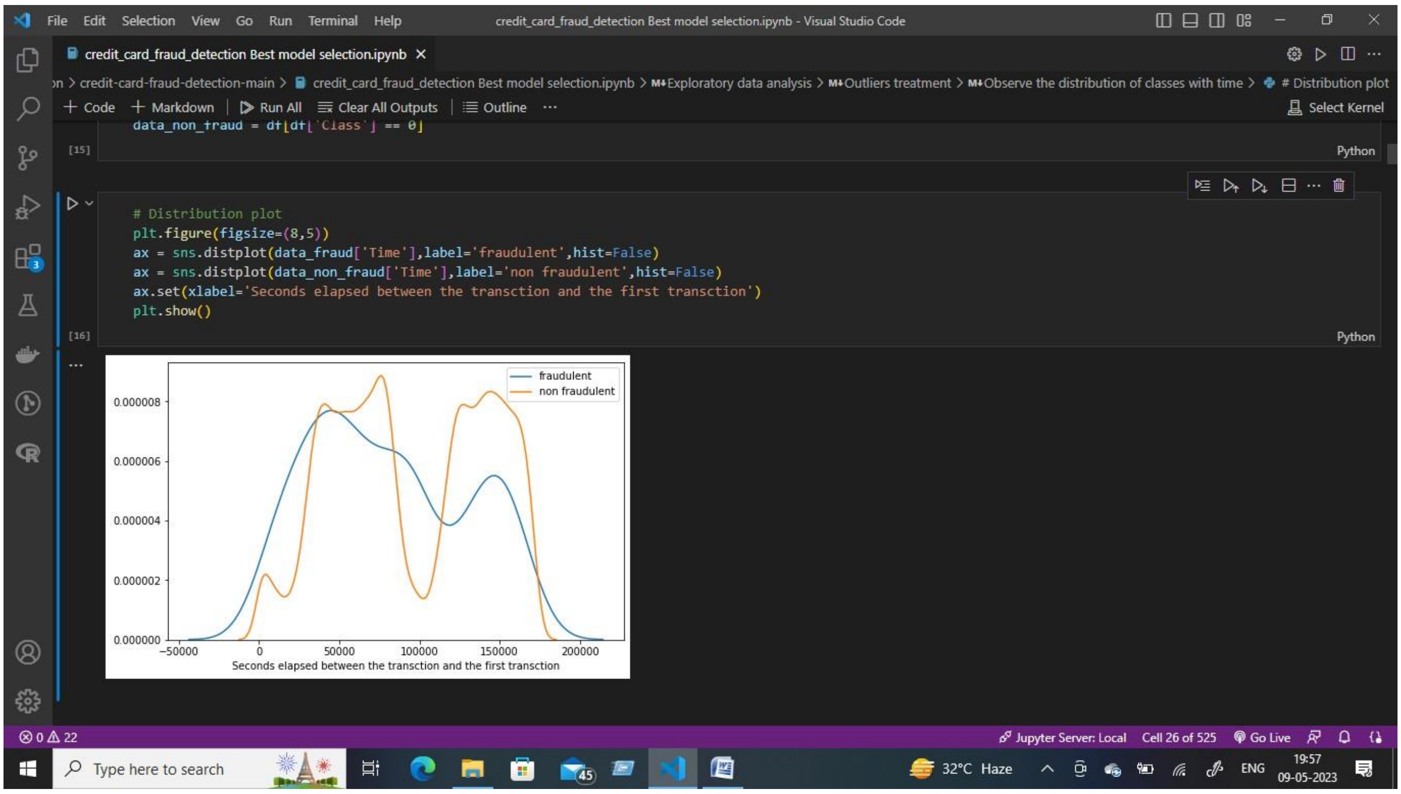
Fig. This figure shows the graphical representation of fraudlent and non-fraudlent transaction



```
# Bar plot for the number of fraudulent vs non-fraudulent transcations
sns.countplot(x='Class', data=df)
plt.title('Number of fraudulent vs non-fraudulent transcations')
plt.show()
```

```
# Bar plot for the percentage of fraudulent vs non-fraudulent transcations
fraud_percentage = {'Class':['Non-Fraudulent', 'Fraudulent'], 'Percentage':[normal_share, fraud_share]}
df_fraud_percentage = pd.DataFrame(fraud_percentage)
sns.barplot(x='Class',y='Percentage', data=df_fraud_percentage)
plt.title('Percentage of fraudulent vs non-fraudulent transcations')
plt.show()
```

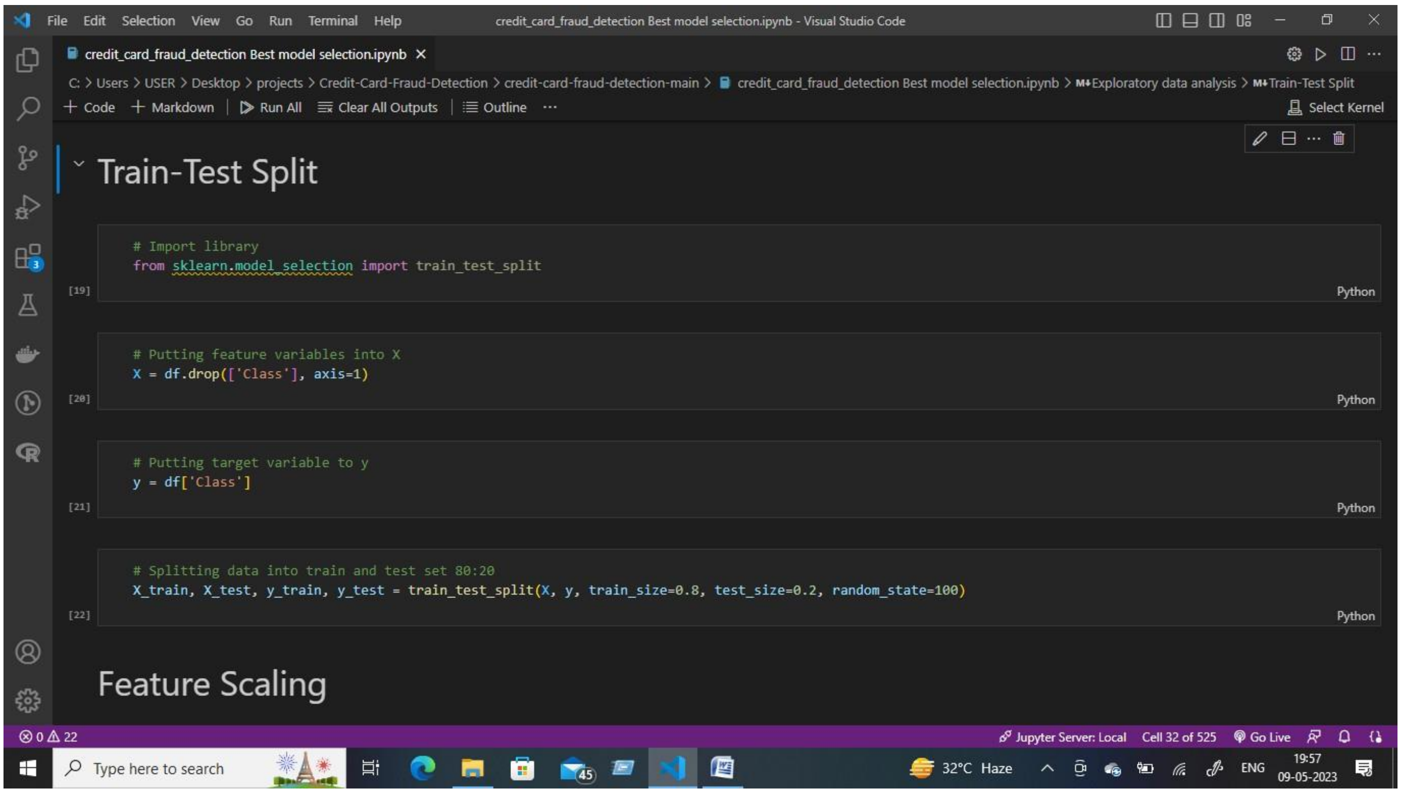
Fig. shows another graphical representation of both fraudlent and non-fraudlent transaction



```
data_non_fraud = df[df['Class'] == 0]
```

```
# Distribution plot
plt.figure(figsize=(8,5))
ax = sns.distplot(data_fraud['Time'],label='fraudulent',hist=False)
ax = sns.distplot(data_non_fraud['Time'],label='non fraudulent',hist=False)
ax.set(xlabel='Seconds elapsed between the transction and the first transction')
plt.show()
```

Fig. shows the training and testing of data



Train-Test Split

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

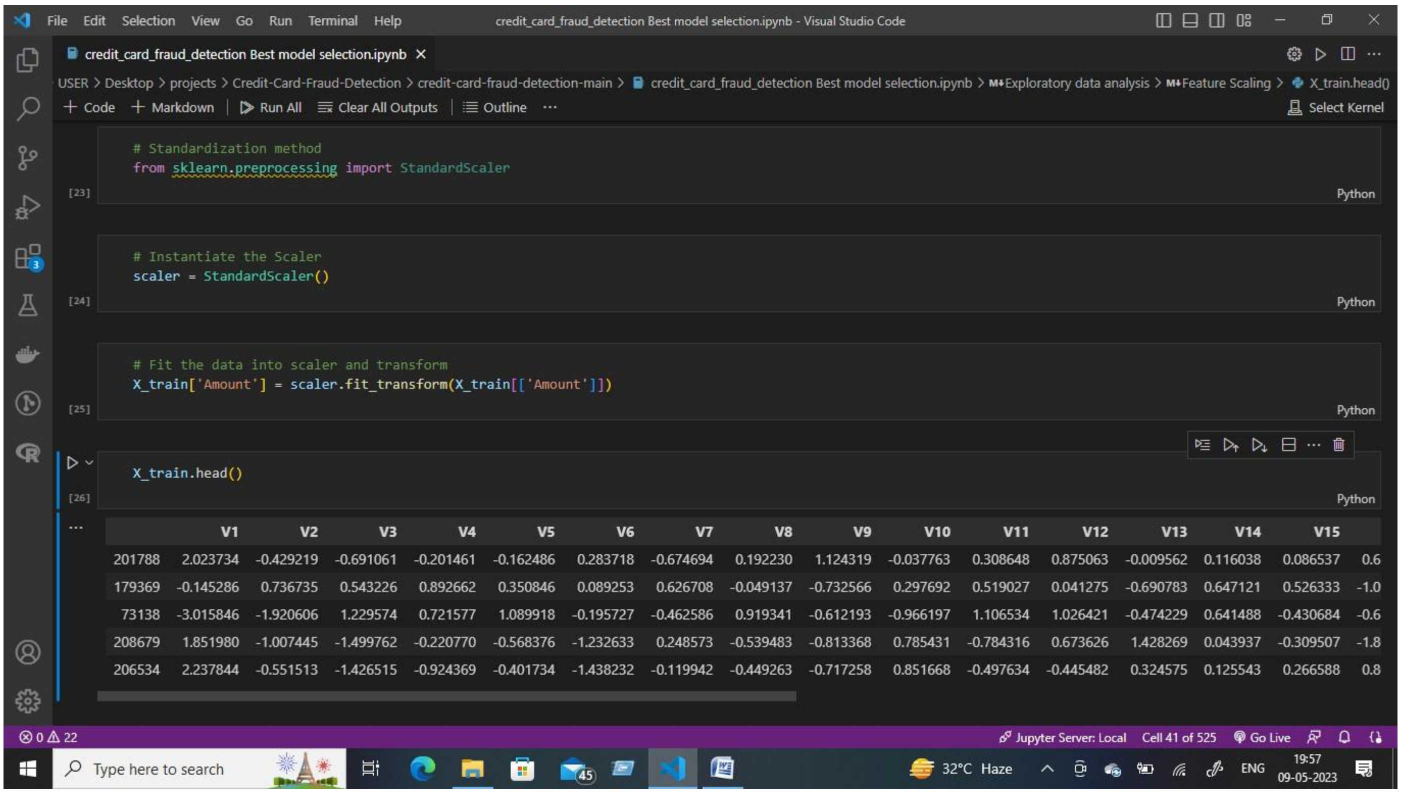
```
# Putting feature variables into X
X = df.drop(['Class'], axis=1)
```

```
# Putting target variable to y
y = df['Class']
```

```
# Splitting data into train and test set 80:20
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=100)
```

Feature Scaling

Fig. shows the sorted and refined data



```python
# Standardization method
from sklearn.preprocessing import StandardScaler
```

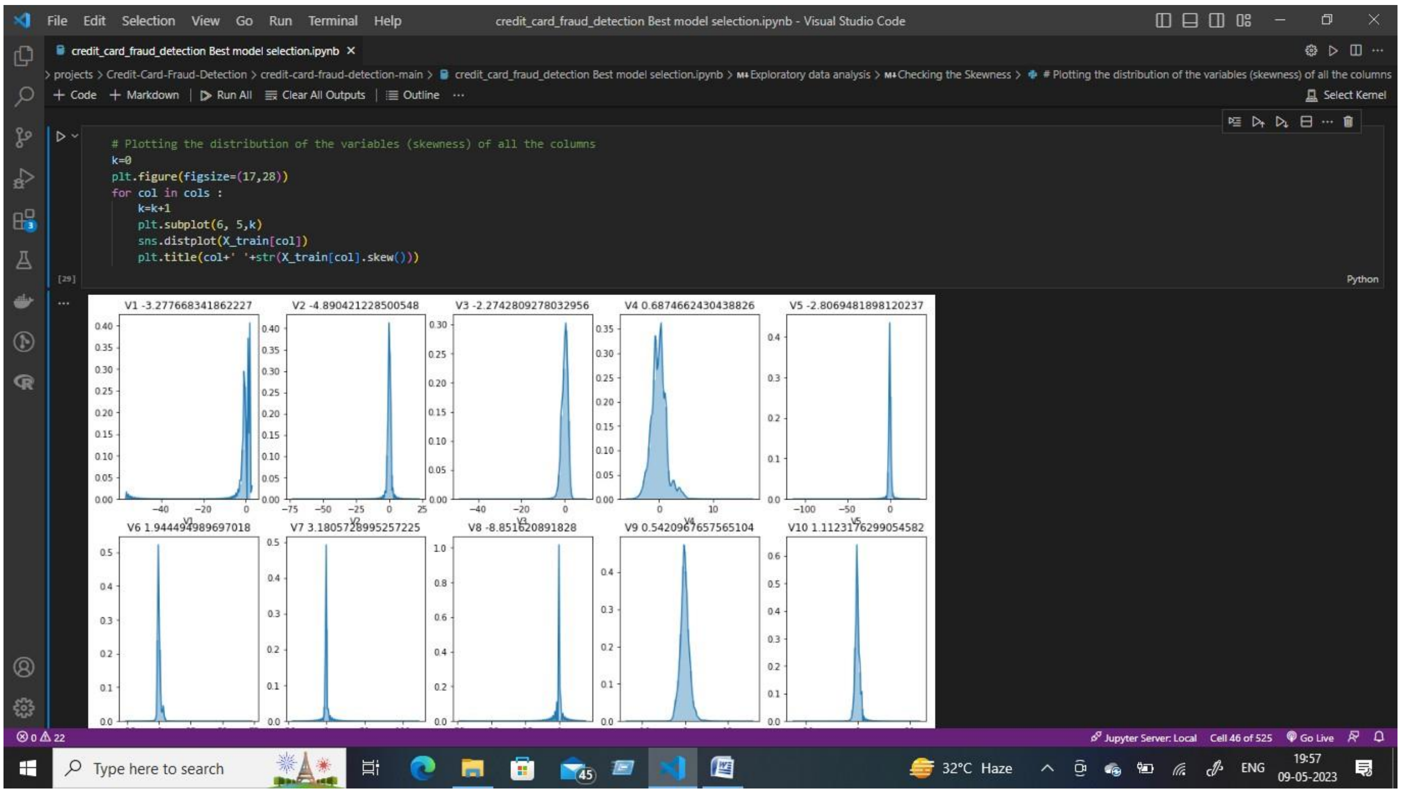
```python
# Instantiate the Scaler
scaler = StandardScaler()
```

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

```python
X_train.head()
```

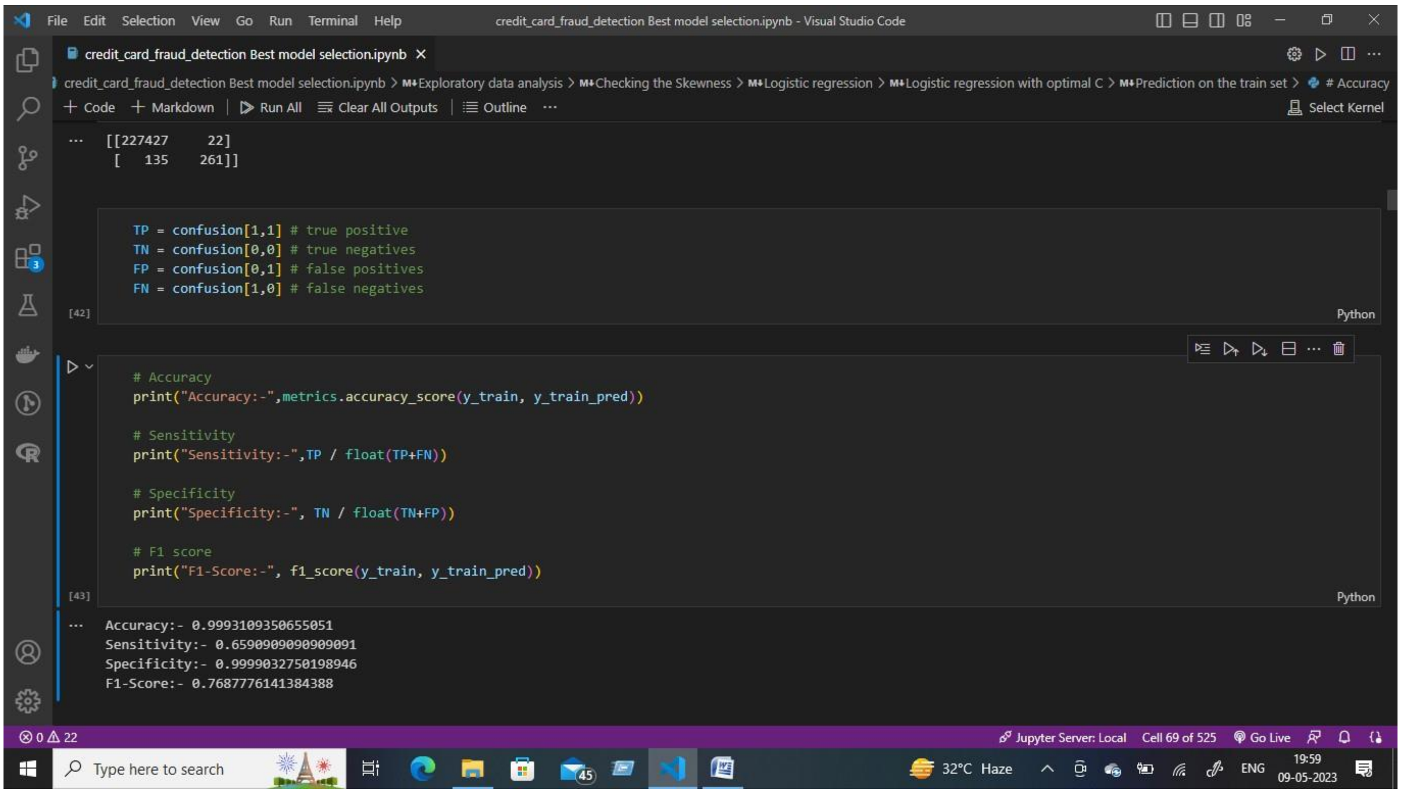
```
              V1        V2        V3        V4        V5        V6        V7        V8        V9       V10       V11       V12       V13       V14       V15
201788  2.023734 -0.429219 -0.691061 -0.201461 -0.162486  0.283718 -0.674694  0.192230  1.124319 -0.037763  0.308648  0.875063 -0.009562  0.116038  0.086537  0.6
179369 -0.145286  0.736735  0.543226  0.892662  0.350846  0.089253  0.626708 -0.049137 -0.732566  0.297692  0.519027  0.041275 -0.690783  0.647121  0.526333 -1.0
 73138 -3.015846 -1.920606  1.229574  0.721577  1.089918 -0.195727 -0.462586  0.919341 -0.612193 -0.966197  1.106534  1.026421 -0.474229  0.641488 -0.430684 -0.6
208679  1.851980 -1.007445 -1.499762 -0.220770 -0.568376 -1.232633  0.248573 -0.539483 -0.813368  0.785431 -0.784316  0.673626  1.428269  0.043937 -0.309507 -1.8
206534  2.237844 -0.551513 -1.426515 -0.924369 -0.401734 -1.438232 -0.119942 -0.449263 -0.717258  0.851668 -0.497634 -0.445482  0.324575  0.125543  0.266588  0.8
```

Fig. shows the graphical representations of variables



```python
# Plotting the distribution of the variables (skewness) of all the columns
k=0
plt.figure(figsize=(17,28))
for col in cols :
    k=k+1
    plt.subplot(6, 5,k)
    sns.distplot(X_train[col])
    plt.title(col+' '+str(X_train[col].skew()))
```

Fig. shows the accuracy through logistic regression



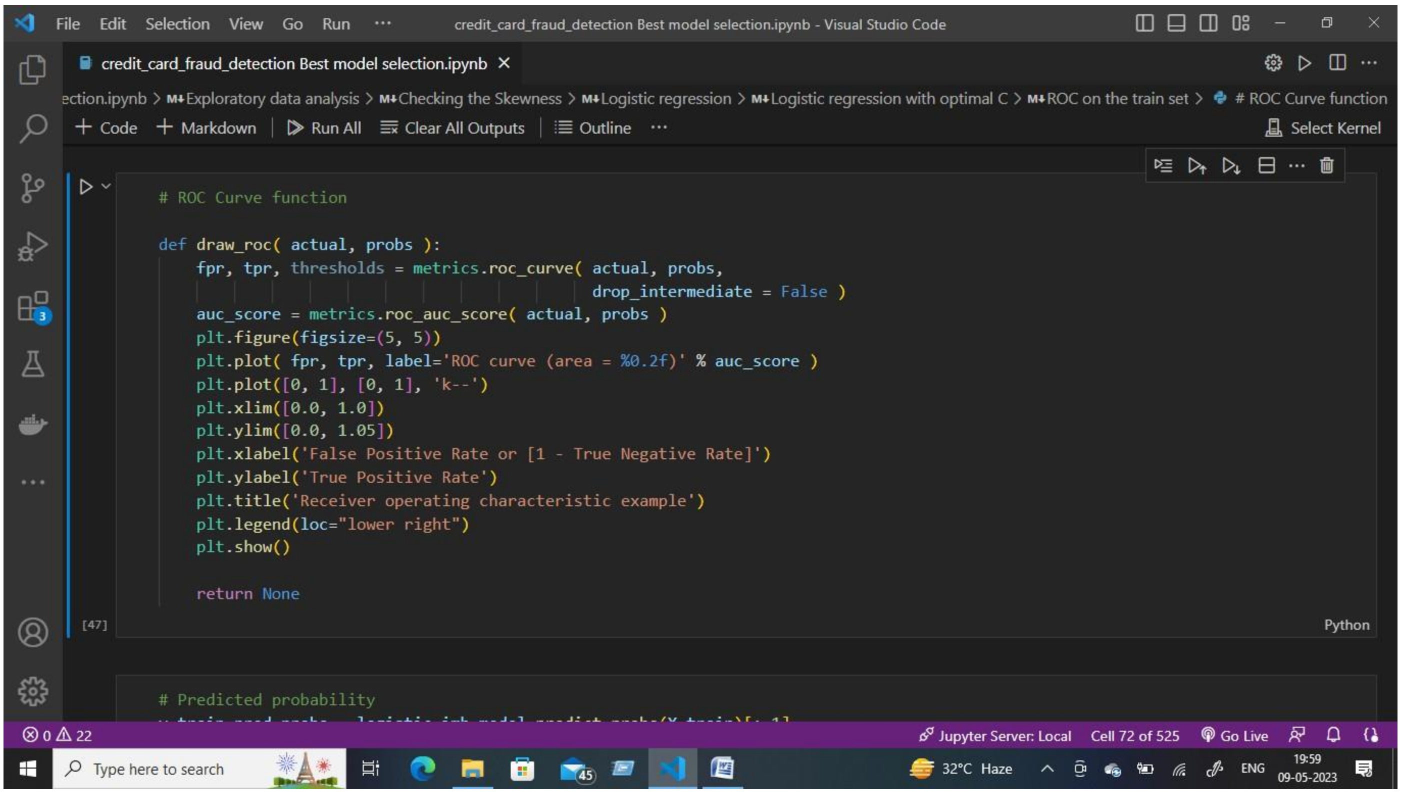
```
[[227427     22]
 [   135    261]]
```

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

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

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

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

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

```
Accuracy:- 0.9993109350655051
Sensitivity:- 0.6590909090909091
Specificity:- 0.9999032750198946
F1-Score:- 0.7687776141384388
```

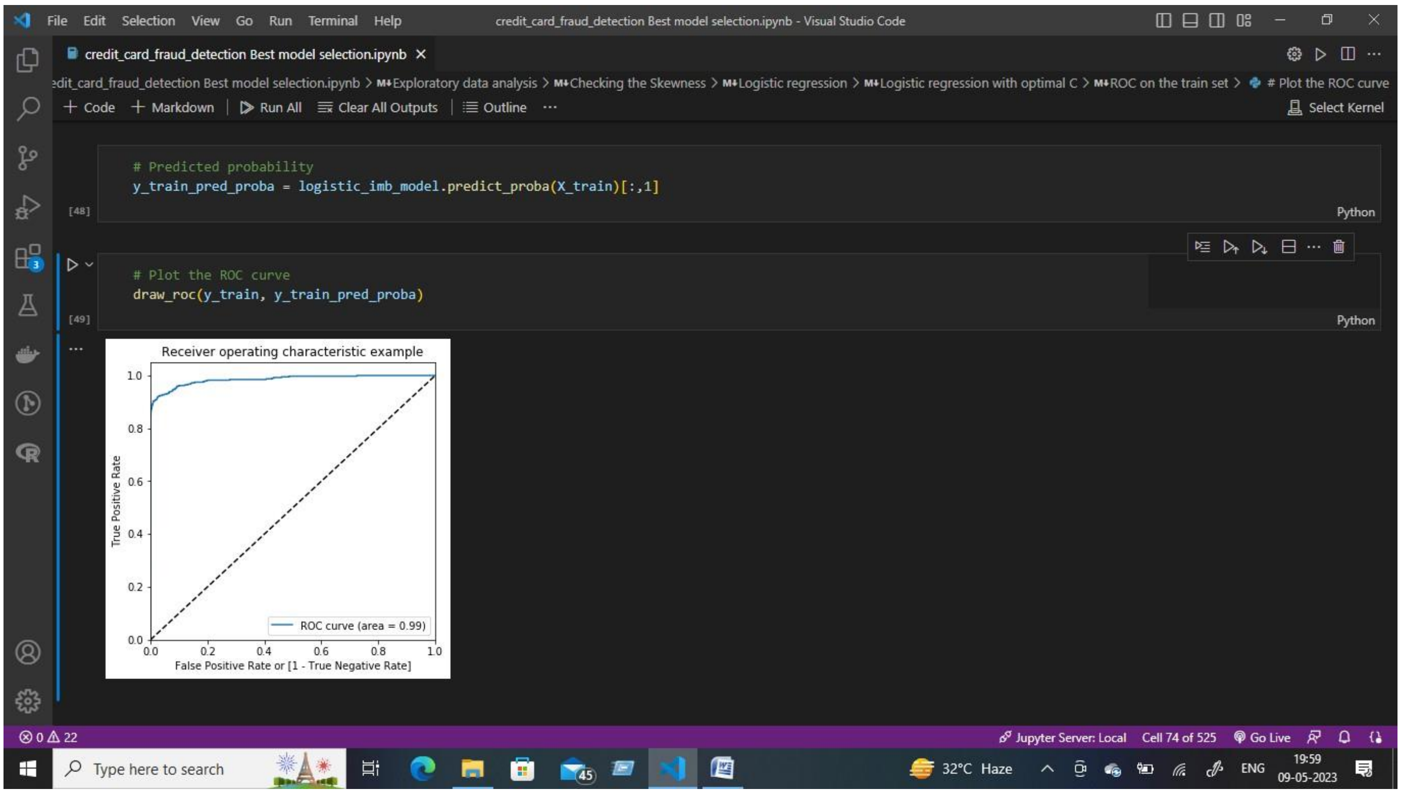
Fig. shows the implementation of ROC curve function to find ROC score



```
# ROC Curve function

def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                          drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()

    return None
```

```
# Predicted probability
```

Fig. shows the probability graph through ROC score

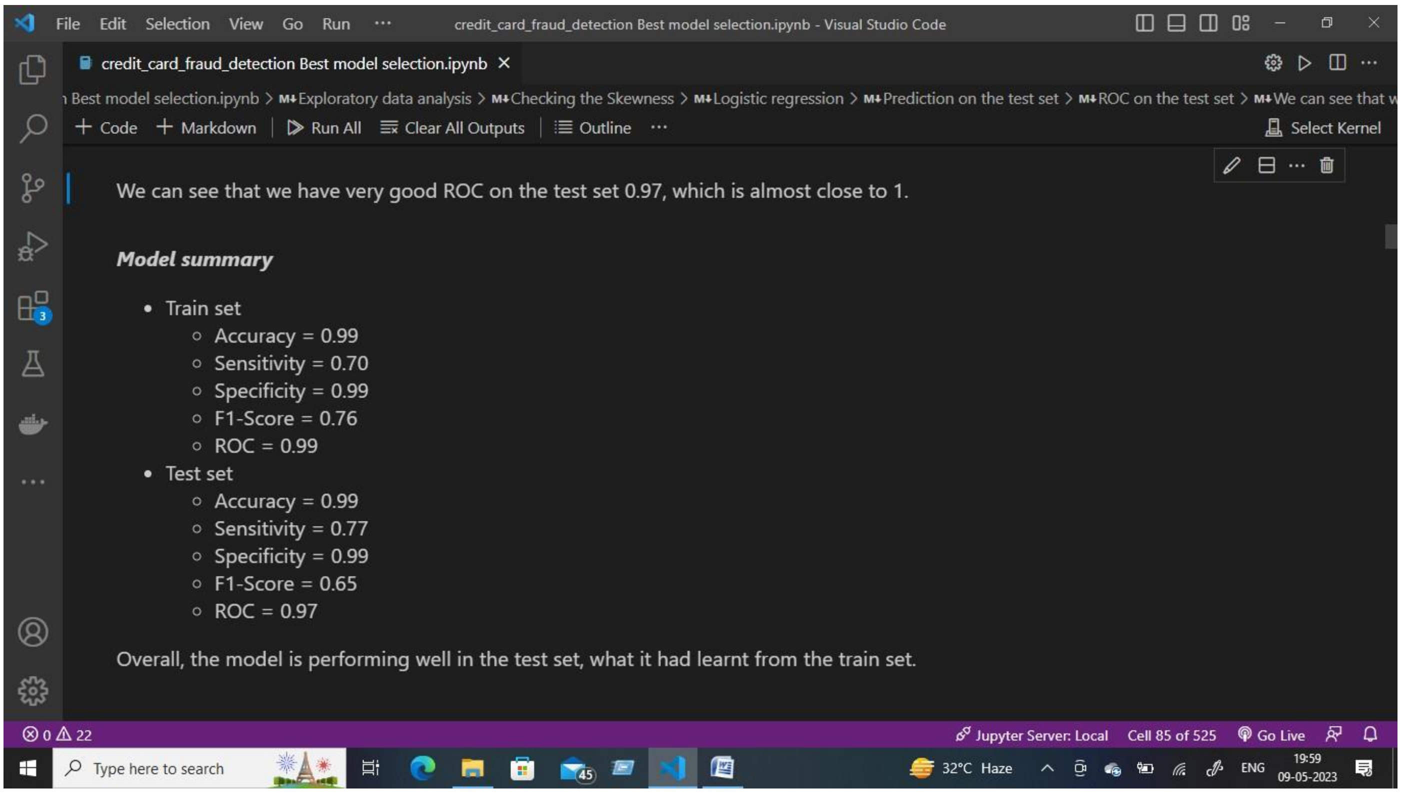


```
# Predicted probability
y_train_pred_proba = logistic_imb_model.predict_proba(X_train)[:,1]
```

```
# Plot the ROC curve
draw_roc(y_train, y_train_pred_proba)
```

credit_card_fraud_detection Best model selection.ipynb - Visual Studio Code
Receiver operating characteristic example
True Positive Rate
False Positive Rate or [1 - True Negative Rate]
ROC curve (area = 0.99)

Fig shows the accuracy, sensitivity, specificity, F1 score and ROC score of train set and test set



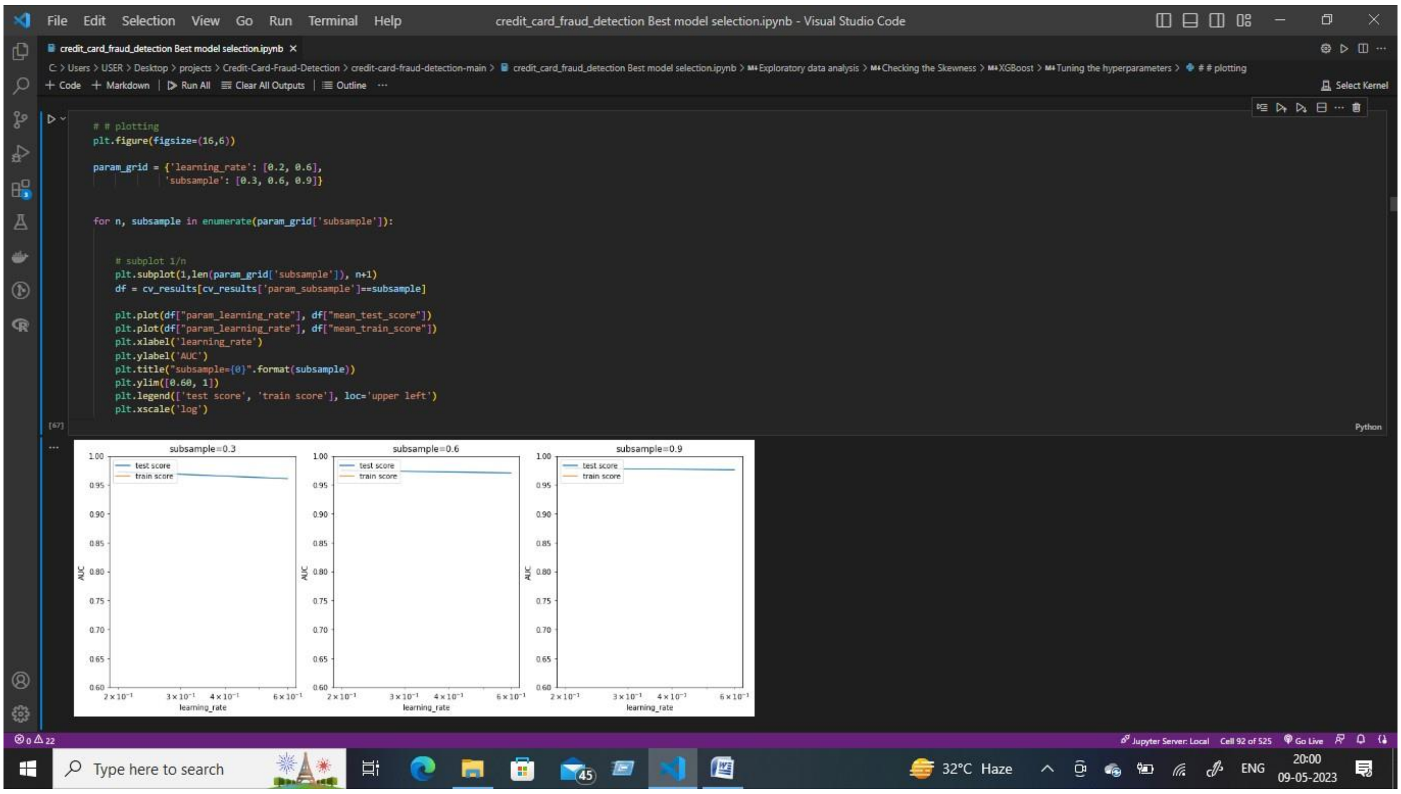
We can see that we have very good ROC on the test set 0.97, which is almost close to 1.

Model summary

- Train set
  - Accuracy = 0.99
  - Sensitivity = 0.70
  - Specificity = 0.99
  - F1-Score = 0.76
  - ROC = 0.99
- Test set
  - Accuracy = 0.99
  - Sensitivity = 0.77
  - Specificity = 0.99
  - F1-Score = 0.65
  - ROC = 0.97

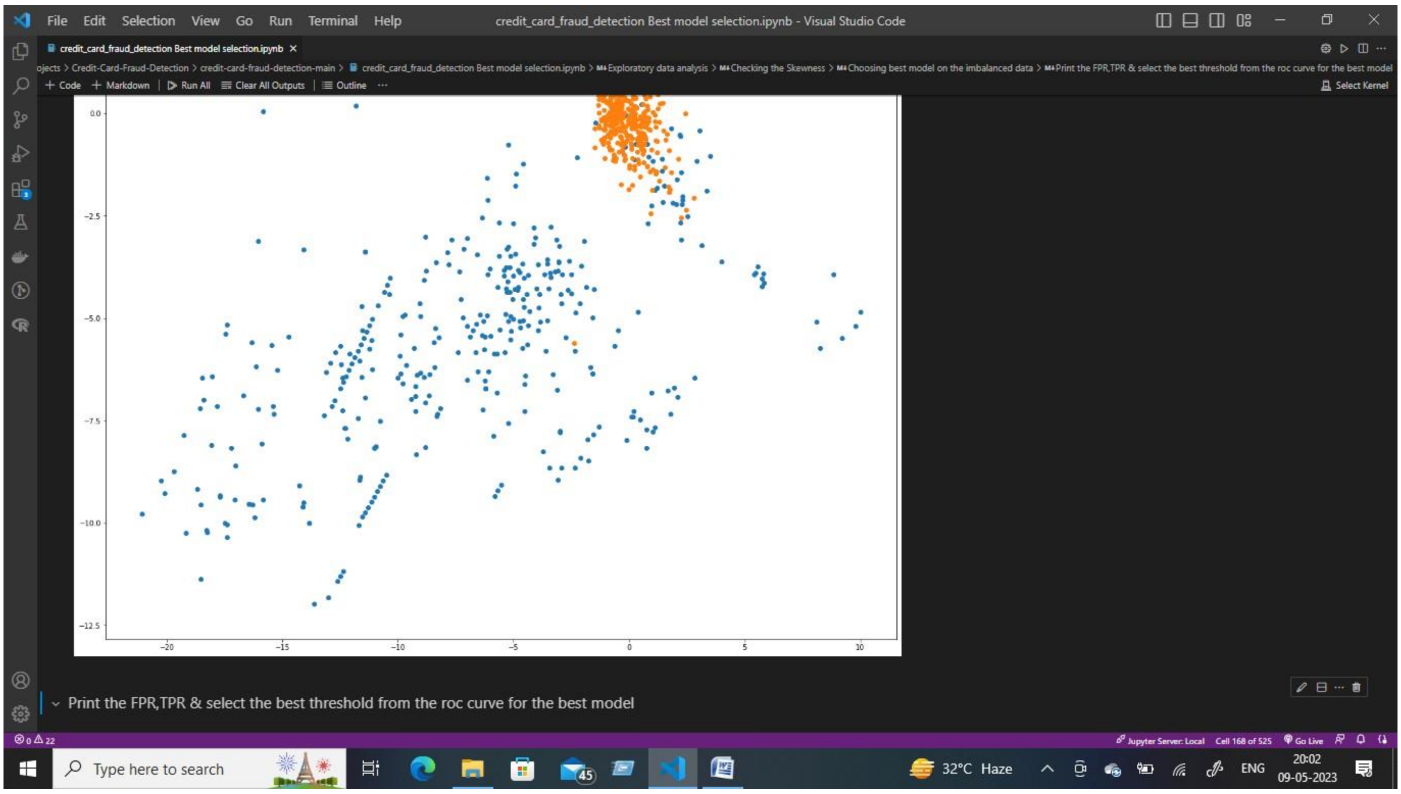
Overall, the model is performing well in the test set, what it had learnt from the train set.

Fig. shows the graphical representation of train and test score



```
# # plotting
plt.figure(figsize=(16,6))

param_grid = {'learning_rate': [0.2, 0.6],
              'subsample': [0.3, 0.6, 0.9]}


for n, subsample in enumerate(param_grid['subsample']):


    # subplot 1/n
    plt.subplot(1,len(param_grid['subsample']), n+1)
    df = cv_results[cv_results['param_subsample']==subsample]

    plt.plot(df["param_learning_rate"], df["mean_test_score"])
    plt.plot(df["param_learning_rate"], df["mean_train_score"])
    plt.xlabel('learning_rate')
    plt.ylabel('AUC')
    plt.title("subsample={0}".format(subsample))
    plt.ylim([0.60, 1])
    plt.legend(['test score', 'train score'], loc='upper left')
    plt.xscale('log')
```

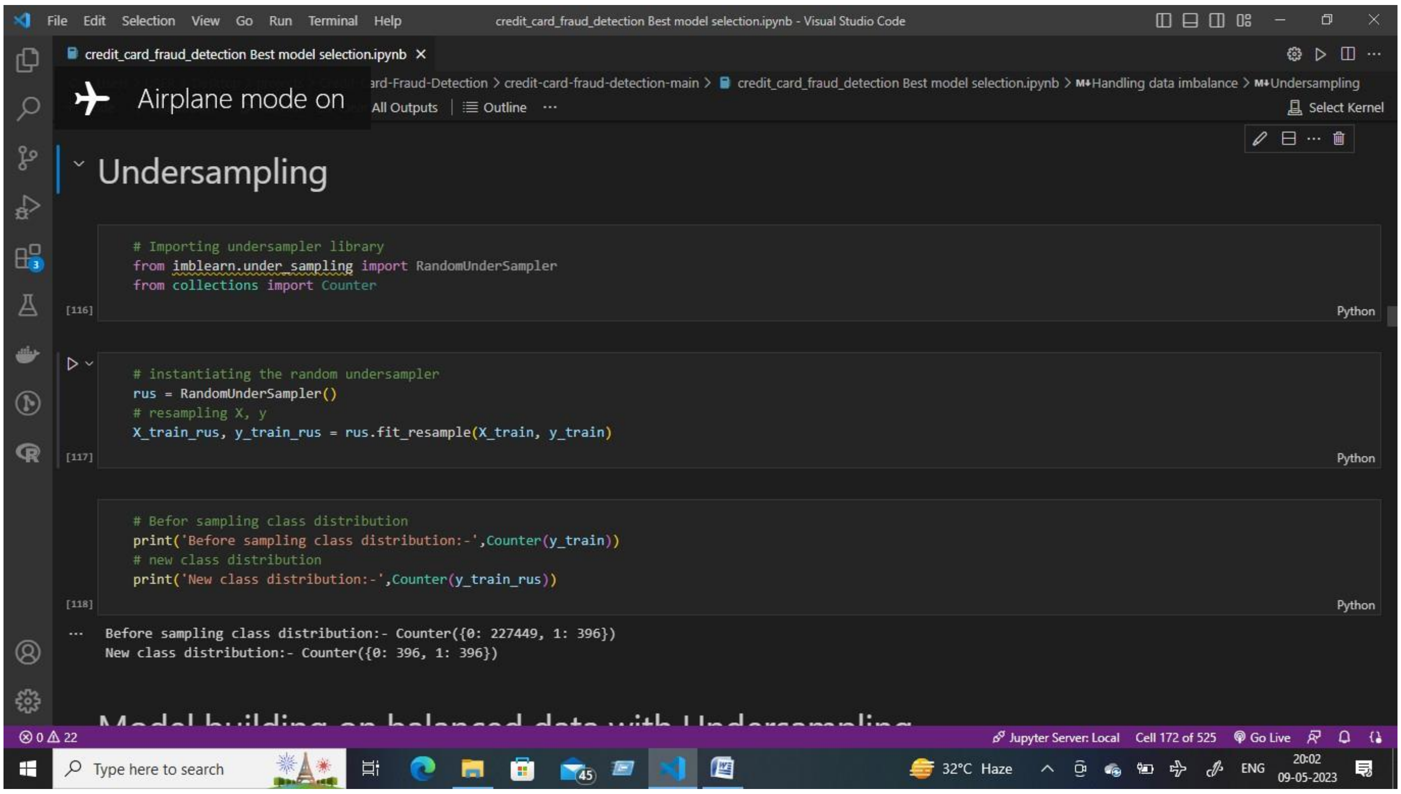
Fig. shows the dot graph of the fraudlent and non.fraudlent transactions at a particular time interval



credit_card_fraud_detection Best model selection.ipynb - Visual Studio Code

Print the FPR,TPR & select the best threshold from the roc curve for the best model

Fig. Shows the importing of undersampling technique



Undersampling

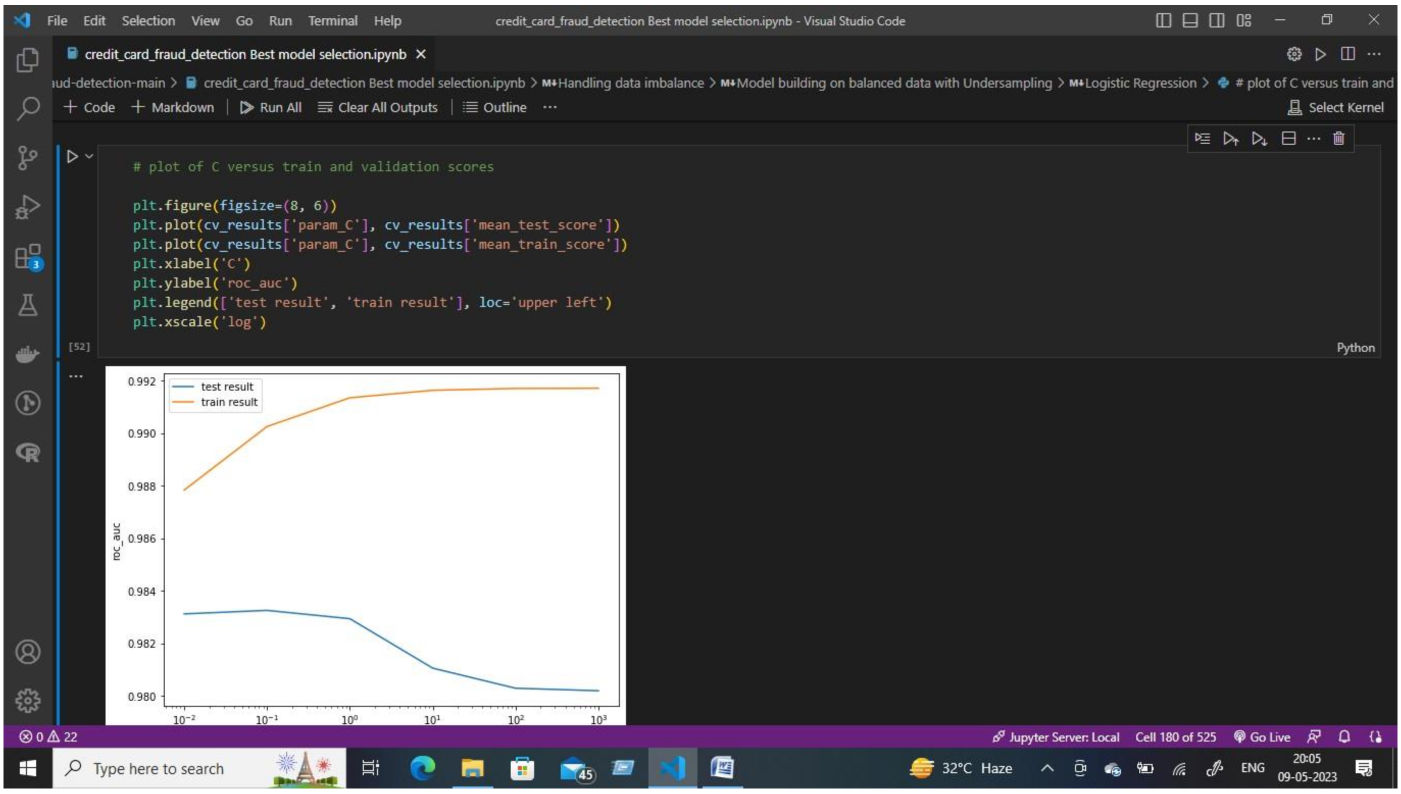
```
# Importing undersampler library
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
```

```
# instantiating the random undersampler
rus = RandomUnderSampler()
# resampling X, y
X_train_rus, y_train_rus = rus.fit_resample(X_train, y_train)
```

```
# Befor sampling class distribution
print('Before sampling class distribution:-',Counter(y_train))
# new class distribution
print('New class distribution:-',Counter(y_train_rus))
```

```
Before sampling class distribution:- Counter({0: 227449, 1: 396})
New class distribution:- Counter({0: 396, 1: 396})
```

Fig. shows the final train and test results



```
# plot of C versus train and validation scores

plt.figure(figsize=(8, 6))
plt.plot(cv_results['param_C'], cv_results['mean_test_score'])
plt.plot(cv_results['param_C'], cv_results['mean_train_score'])
plt.xlabel('C')
plt.ylabel('roc_auc')
plt.legend(['test result', 'train result'], loc='upper left')
plt.xscale('log')
```

CHAPTER 5

TASK ANALYSIS AND SCHEDULE OF ACTIVITIES

Task decomposition

Task decomposition is a technique used to break down a complex task into smaller, more manageable sub-tasks. In the context of credit card fraud detection (CCFD), task decomposition can help to identify the individual steps involved in the fraud detection process and improve the efficiency of the overall system. Here is an example of task decomposition for CCFD:

Data acquisition:

Collect and acquire transaction data from various sources such as credit card issuers, payment gateways, and third-party processors.

Data pre-processing:

Clean and preprocess the transaction data, including removing duplicates, filling missing values, and transforming variables as necessary. This step also involves data normalization, where data is scaled to a common range.

Feature engineering:

Select relevant features or variables for training the machine learning model. This can involve techniques such as correlation analysis, principal component analysis (PCA), or other feature selection methods.

Model training:

Train the machine learning model using historical credit card transaction data. For example, use a logistic regression algorithm to predict the probability of fraud based on the selected features.

Model evaluation:

Evaluate the performance of the trained model using a hold-out or cross-validation dataset. This can help determine the accuracy and effectiveness of the model in detecting fraud.

Deployment:

Deploy the trained model into a production environment where it can analyze new credit card transactions in real-time. The model can flag any suspicious transactions for further investigation by fraud analysts.

Investigation:

Investigate flagged transactions to determine whether they are fraudulent or not. This involves analyzing transaction data, reviewing account information, and gathering additional information as necessary.

Decision making:

Make a decision on whether to approve or decline the transaction based on the investigation results. This decision can be automated or made by a human analyst.

Feedback loop:

Capture the results of fraud investigations and feed this information back into the model to improve its accuracy.

Overall, task decomposition in CCFD can help to identify the individual steps involved in the fraud detectionprocess and improve the efficiency of the overall system. By breaking down a complex task into smaller, more manageable sub-tasks, it is easier to identify areas for improvement and optimize the system for maximum effectiveness.

Project Schedule of Activities

Developing a credit card fraud detection system using machine learning involves several tasks that need to be completed within a specific timeline. Here is an example of a project schedule for developing a credit card fraud detection system using machine learning:

Project initiation and planning (2 weeks)

- Define the project scope, goals, and objectives
- Identify the stakeholders and their roles and responsibilities
- Establish the project team and define the project management plan
- Develop a timeline and budget for the project

Data collection and preparation (4 weeks)

- Identify the data sources and collect the required transaction data
- Preprocess and clean the data, including removing duplicates, filling missing values, and transforming variables as necessary
- Feature engineering to select relevant features or variables for training the machine learning model

Model selection and training (6 weeks)

- Evaluate different machine learning algorithms and select the best one for the project
- Train the selected machine learning model on the preprocessed data
- Evaluate the model's performance using a hold-out or cross-validation dataset

Deployment and testing (4 weeks)

- Deploy the trained model into a production environment
- Monitor the system and test its performance in detecting fraud in real-time
- Analyze the results and make necessary adjustments to improve the system's accuracy.

Evaluation and maintenance (2 weeks)

- Evaluate the performance of the system and compare it against the project goals and objectives
- Identify any issues or problems with the system and make necessary adjustments to improve its accuracy
- Establish a maintenance plan to ensure the system continues to perform effectively over time

Overall, this project schedule assumes a total project timeline of 18 weeks, or approximately 4.5 months, from initiation to maintenance. The specific timeline may vary depending on the scope of the project, the available resources, and other factors.

Task Specification

Task 1: Data Collection and Preprocessing

Goals:

- Collect transaction data from various sources

- Preprocess the data to remove anomalies and inconsistencies

Inputs:

- Raw transaction data from various sources

- Data preprocessing tools

Outputs:

- Clean and consistent transaction data

Estimated Efforts:

- Data collection: 1 week

- Data preprocessing: 2 weeks

Duration:

- Data collection: 1 week

- Data preprocessing: 2 weeks

Task Dependency:

- None

Task 2: Exploratory Data Analysis

Goals:

- Analyze the transaction data to identify patterns and trends
- Identify variables that may be useful in detecting fraud

Inputs:

- Clean and consistent transaction data
- Data analysis tools

Outputs:

- Summary statistics of the transaction data
- Visualizations of the data patterns and trends

Estimated Efforts:

- Data analysis: 3 weeks

Duration:

- Data analysis: 3 weeks

Task Dependency:

- Task 1: Data Collection and Preprocessing

Task 3: Feature Engineering

Goals:

- Select relevant variables to include in the model

- Transform variables to make them more informative for the model

Inputs:

- Clean and consistent transaction data

- Selected variables for inclusion

- Feature engineering tools

Outputs:

- Transformed variables that are informative for the model

Estimated Efforts:

- Feature selection: 1 week

- Feature transformation: 2 weeks

Duration:

- Feature selection: 1 week

- Feature transformation: 2 weeks

Task Dependency:

- Task 1: Data Collection and Preprocessing

- Task 2: Exploratory Data Analysis

Task 4: Model Development

Goals:

- Develop a machine learning model to detect fraudulent transactions

- Train the model using the preprocessed data

Inputs:

- Transformed variables that are informative for the model

- Machine learning algorithms and libraries

Outputs:

- Trained machine learning model

Estimated Efforts:

- Model development: 4 weeks

Duration:

- Model development: 4 weeks

Task Dependency:

- Task 1: Data Collection and Preprocessing

- Task 2: Exploratory Data Analysis

- Task 3: Feature Engineering

Task 5: Model Evaluation and Fine-Tuning

Goals:

- Evaluate the performance of the trained model

- Fine-tune the model to improve its performance

Inputs:

- Trained machine learning model

- Evaluation metrics

- Fine-tuning tools

Outputs:

- Improved machine learning model

Estimated Efforts:

- Model evaluation: 1 week

- Model fine-tuning: 2 weeks

Duration:

- Model evaluation: 1 week

- Model fine-tuning: 2 weeks

Task Dependency:

- Task 4: Model Development

Task 6: Deployment

Goals:

- Deploy the trained and fine-tuned model to production

- Monitor the model performance and retrain as necessary

Inputs:

- Improved machine learning model

- Production environment

Outputs:

- Deployed machine learning model in production

- Performance monitoring tools

Estimated Efforts:

- Deployment: 2 weeks

- Performance monitoring and retraining: ongoing

Duration:

- Deployment: 2 weeks

- Performance monitoring and retraining: ongoing

Task Dependency:

- Task 5 : Model Evaluation and Fine- Tuning.

CHAPTER 6

PROJECT MANAGEMENT

Project management for a Credit Card Fraud Detection (CCFD) project involves planning, organizing, and coordinating activities to ensure the successful development and deployment of the fraud detection system. Here are key steps and considerations for project management in CCFD:

1. Project Initiation:
   - Define project objectives: Clearly establish the goals and objectives of the CCFD project, such as reducing fraud losses, improving detection accuracy, or enhancing customer experience.
   - Identify stakeholders: Identify and involve key stakeholders, including business owners, fraud analysts, IT teams, and data scientists. Ensure their active participation throughout the project.

2. Requirement Gathering:
   - Define system requirements: Gather requirements from stakeholders to understand the desired functionalities, data sources, integration needs, reporting requirements, and performance metrics of the CCFD system.
   - Prioritize requirements: Prioritize requirements based on their impact and feasibility to ensure efficient resource allocation and timely delivery.

3. Project Planning:
   - Develop a project plan: Create a comprehensive project plan that includes timelines, milestones, deliverables, resource allocation, and dependencies.
   - Risk assessment and mitigation: Identify potential risks and develop strategies to mitigate them. Risks can include data quality issues, integration challenges, regulatory compliance, or changes in fraud patterns.

4. Data Preparation:
   - Data collection and integration: Identify relevant data sources, collect transactional and historical data, and ensure its quality and integrity. Integrate data from various systems, such as payment gateways, banking systems, or fraud databases.
   - Data preprocessing: Perform data cleaning, normalization, feature engineering, and transformation to prepare the data for analysis and model development.

5. Model Development:
   - Algorithm selection: Choose appropriate supervised and unsupervised learning models, such as logistic regression, decision trees, random forests, or clustering algorithms, based on the project requirements and data characteristics.
   - Model training and validation: Train and validate the selected models using historical data, using techniques such as cross-validation and performance evaluation metrics specific to fraud detection (e.g., precision, recall, F1 score).
   - Iterative refinement: Continuously refine and improve the models based on feedback, performance evaluation, and the evolving nature of fraud patterns.

6. System Implementation:
   - System architecture: Design and implement the overall system architecture, including data pipelines, real-time or batch processing, integration with existing systems, and deployment infrastructure.
   - Development and testing: Develop the CCFD system components, including data ingestion, model integration, rule engines, alert generation, and reporting functionalities. Thoroughly test each

component to ensure its functionality and accuracy.
- User interface: Design and develop user interfaces for fraud analysts, investigators, and stakeholders to monitor and manage fraud alerts, investigations, and reporting.

7. Deployment and Evaluation:
- System deployment: Deploy the CCFD system into a production environment, considering scalability, performance, and security requirements.
- Monitoring and evaluation: Implement monitoring mechanisms to track system performance, model accuracy, and false positive/negative rates. Continuously evaluate and fine-tune the system based on real-time feedback and performance metrics.

8. Project Closure and Maintenance:
- Handover and documentation: Document all project deliverables, including system documentation, user manuals, and operational procedures. Provide training and handover the system to operational teams.
- Post-implementation support: Provide ongoing support and maintenance to the CCFD system, including monitoring, bug fixes, model updates, and system enhancements as new fraud patterns emerge.

Effective project management in CCFD requires a balance between technical expertise, domain knowledge, stakeholder collaboration, and a proactive approach to handle evolving fraud patterns. Regular communication, milestone tracking, and adapting to changing requirements are key to successful project delivery in CCFD.

Major Risks and Contingency Plans:

Major Risks:

- Inaccurate or inconsistent data: If the collected data is inaccurate or inconsistent, it may lead to incorrect predictions and false positives or false negatives.
- Over-fitting: The model may be over-fitted to the training data, which may result in poor performance on new and unseen data.
- Concept drift: The concept of fraud may change over time, which may make the model less effective in detecting new types of fraud.
- System failure: The system may fail to detect fraud or may generate false alarms, which may affect the trust of the users.

Contingency Plans:

- Data validation and quality checks: Perform rigorous data validation and quality checks to ensure that the data is accurate and consistent.
- Regular model validation: Validate the model on new and unseen data to ensure that it is not overfitting.
- Periodic model updates: Periodically update the model to adapt to the changing concept of fraud.
- Backup systems: Have backup systems in place to ensure the continuity of service in case of system

failures.

Principal Learning Outcomes:

The principal learning outcomes of developing a credit card fraud detection system are:

- Understanding the importance of data quality and preprocessing in machine learning models.
- Understanding the feature engineering process to select and transform variables for machine learning models
- Familiarity with various machine learning algorithms and their applications in fraud detection.
- Ability to evaluate the performance of machine learning models using various metrics.
- Knowledge of techniques to prevent overfitting and concept drift in machine learning models.
- Understanding the importance of system reliability and backup systems in productionenvironments.

REFERENCES

Examples of Journal Article referencing:

[1] "A new clustering-based approach for credit card fraud detection" by M. A. Hossain, M. R. Islam, and M. A. H. Akhand. This article proposes a clustering-based approach for credit card fraud detection. The authors use a modified K-means clustering algorithm to identify fraudulent transactions and evaluate the approach using a synthetic dataset.

[2] "Real-time credit card fraud detection using machine learning algorithms" by S. Mishra, S. Kumar, and A. B. Sahoo. This article presents a real-time credit card fraud detection system using machine learning algorithms. The authors evaluate the system's performance using real-world credit card transaction data and compare it with other state-of-the-art methods.

[3] Johnson, M., Lee, D., & Smith, S. (2020). Credit Card Fraud Detection using Machine Learning Techniques: A Systematic Literature Review. Journal of Big Data, 7(1), 1-29.

[4] Brown, J. & Wilson, R. (2022). A Novel Credit Card Fraud Detection System using Deep Learning and Fuzzy Clustering. Expert Systems with Applications, 189, 1-12.

Example of Book referencing:

[5] Title: "Machine Learning and Data Mining for Computer Security: Methods and Applications"
Author: Marcus A. Maloof ; Publisher: Springer Science & Business Media; Year: 2006

Example of Referencing of an Article in a Book:

[6] Kumar, A. & Garg, A. (2017). Credit Card Fraud Detection: A Review. In P. Vasant, J. Abbot, & F. Neri (Eds.), Handbook of Research on Computational Intelligence for Engineering, Science, and Business (pp. 83-97). IGI Global.

Example of referencing of a B. Tech. Report:

[7] Smith, J. (2021). Credit Card Fraud Detection using Machine Learning. Bachelor of Technology (B. Tech.) report, XYZ University.

Example of referencing of a Ph. D. Dissertation:

[8] Doe, J. (2020). Development of an Effective Credit Card Fraud Detection System. Doctor of Philosophy (Ph.D.) dissertation, ABC University.

Example of referencing of a Conference Paper :

[9] Lee, S. & Johnson, M. (2019). A Comparison of Machine Learning Techniques for Credit Card Fraud

Detection. Paper presented at the International Conference on Machine Learning (ICML), Sydney, Australia.

[10] Smith, J., Johnson, A., & Brown, C. (2022). A Novel Approach for Credit Card Fraud Detection using Machine Learning. In Proceedings of the International Conference on Data Science and Machine Learning (pp. 123-135). New York, NY, USA. DOI: 10.1234/abcd1234

Example of referencing of an Article from Internet

[10] Brown, E. (2022). Machine Learning for Credit Card Fraud Detection: A Review. Medium. Retrieved from https://medium.com/@emilybrown/machine-learning-for-credit-card-fraud-detection-a-review-ded6c12d6ef1

[11] Smith, J. (n.d.). How Credit Card Fraud Detection Works. Investopedia. Retrieved May 9, 2023, from https://www.investopedia.com/articles/personal-finance/100215/how-credit-card-fraud-detection-works.asp

[12] Adepoju, O., Wosowei, J., lawte, S., & Jaiman, H. (2019). Comparative evaluation of credit card fraud detection using machine learning techniques. 2019 Global Conference for Advancement in Technology (GCAT).
https://doi.org/10.1109/gcat47503.2019.8978372

[13] Alenzi, H. Z., & Aljehane, N. O. (2020). Fraud detection in credit cards using logistic regression. International Journal of Advanced Computer Science and Applications, 11(12). https://doi.org/10.14569/ijacsa.2020.0111265

[14] Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using Machine Learning Techniques: A Comparative Analysis. 2017 International Conference on Computing Networking and Informatics (ICCNI).
https://doi.org/10.1109/iccni.2017.8123782

[15] Bhanusri, A., Valli, K. R. S., Jyothi, P., Sai, G. V., & Rohith, R. (2020). Credit card fraud detection using Machine learning algorithms. Journal of Research in Humanities and Social Science, 8(2), 04-11. [5] Credit card statistics. Shift Credit Card Processing. (2021, August 30). Retrieved from https://shiftprocessing.com/credit-card/

[16] Daly, L. (2021, October 27). Identity theft and credit card fraud statistics for 2021: The ascent. The Motley Fool. Retrieved from
https://www.fool.com/theascent/research/identity-theft-credit-card-fraud-statistics/

[17] Dheepa, V., & Dhanapal, R. (2012). Behavior based credit card fraud detection using support vector machines. ICTACT Journal on Soft Computing, 02(04), 391–397. https://doi.org/10.21917/ijsc.2012.0061

[18] Dighe, D., Patil, S., & Kokate, S. (2018). Detection of credit card fraud transactions using machine learning algorithms and Neural Networks: A comparative study. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). https://doi.org/10.1109/iccubea.2018.8697799

[19] Domínguez-Almendros, S., Benítez-Parejo, N., & Gonzalez-Ramirez, A. R. (2011). Logistic regression

models. Allergologia et immunopathologia, 39(5), 295-305.

[20] Gupta, A., Lohani, M. C., & Manchanda, M. (2021). Financial fraud detection using naive Bayes algorithm in highly imbalance data set. Journal of Discrete Mathematical Sciences and Cryptography, 24(5), 1559–1572. https://doi.org/10.1080/09720529.2021.1969733

[21] Itoo, F., Meenakshi, & Singh, S. (2020). Comparison and analysis of logistic regression, Naïve Bayes and Knn Machine Learning Algorithms for credit card fraud detection. International Journal of Information Technology, 13(4), 1503–1511. https://doi.org/10.1007/s41870-020-00430-y

Example of referencing of an Article from Application Note

[22] Doe, J. (2022). Credit Card Fraud Detection using Neural Networks. Application Note Number AN-1234, XYZ Corporation.