AMITY UNIVERSITY ONLINE, NOIDA, UTTAR PRADESH

In partial fulfillment of the requirement for the award of degree of

Master of Business Administration (Data Science)

TITLE: **Credit Card Fraud Detection By Using Machine Learning**

Guide Det:

Name: Sai Amrit Patnaik

Designation: AI Research Engineer

Submitted By: Komal Dev

Name of the Student- Komal Dev

Enrolment. No: A9920123000459(el)

ABSTRACT

Credit card fraud has become a significant global issue, with billions of dollars lost each year due to fraudulent transactions. As the financial industry continues to grow and digital payment systems expand, fraudsters are continuously developing new techniques to exploit vulnerabilities in payment systems. This surge in fraud cases has prompted the need for advanced detection systems to help reduce losses, enhance security, and improve the overall trust in credit card transactions. Traditional fraud detection methods are often inadequate due to the sheer volume of transactions and the complexity of fraudulent activities, which evolve rapidly. In response, machine learning (ML) techniques have emerged as a promising solution for detecting and preventing credit card fraud. By utilizing historical transaction data, machine learning models can learn to identify patterns and anomalies indicative of fraudulent activity, often with a higher degree of accuracy than traditional methods.

This study proposes the development of a machine learning-based model for detecting credit card fraud. The model aims to leverage various machine learning algorithms to detect fraudulent transactions by analyzing large sets of credit card transaction data. The dataset used for training will consist of historical transactions, which includes both legitimate and fraudulent activities. This approach will allow the model to learn the underlying characteristics of fraudulent behavior, such as unusual spending patterns, geographic inconsistencies, and abnormal transaction frequencies.

The project will follow a structured methodology, starting with the preprocessing of transaction data, which involves cleaning and normalizing the data to remove noise and outliers. Next,

feature engineering will be applied to extract meaningful attributes that can help the model identify key indicators of fraud. These features may include transaction amounts, merchant information, customer behavior, and time patterns. Various machine learning algorithms will then be employed, such as decision trees, support vector machines, logistic regression and KNN to build and train the model. The model's performance will be evaluated using a holdout dataset of unseen transactions, which allows for an unbiased assessment of its ability to detect fraud in real-world conditions.

The key goal of this study is to create a model that can effectively classify transactions as either legitimate or fraudulent, minimizing false positives (legitimate transactions flagged as fraud) and false negatives (fraudulent transactions missed by the system). The model's performance will be measured using common metrics such as accuracy, precision, recall, and F1-score, which provide a comprehensive view of how well the model distinguishes between fraudulent and non-fraudulent transactions. Additionally, techniques such as cross-validation will be employed to ensure the model generalizes well to new data.

A key benefit of using machine learning for fraud detection is its ability to adapt to new fraud patterns. As fraudsters continually change their tactics, the model can be updated and retrained with new data, ensuring it remains effective in identifying emerging types of fraud. Moreover, machine learning models can detect subtle, complex relationships within the data that may not be apparent to human analysts or traditional rule-based systems.

The results of this study are expected to demonstrate the effectiveness of machine learning in improving credit card fraud detection. If successful, the model could be integrated into real-time transaction processing systems, providing an additional layer of security for consumers and financial institutions. By identifying fraudulent transactions at an early stage,

the model could prevent significant financial losses and protect sensitive customer information.

In conclusion, the development of a machine learning-based fraud detection system represents a critical step toward enhancing the security and reliability of credit card transactions. The use of advanced ML algorithms for fraud detection can help financial institutions stay ahead of fraudsters, ensuring better protection for consumers and businesses alike. As the financial landscape continues to evolve, the integration of machine learning into fraud detection systems will play a pivotal role in maintaining the integrity of credit card systems worldwide.

Keywords: *Credit Card Fraud Detection, Fraud Detection, Fraudulent Transactions, K-Nearest Neighbors, Support Vector Machine, Logistic Regression, Decision Tree.*

DECLARATION

I, Komal Dev, a student pursuing MBA (Data Science) in semester IV at Amity University Online, hereby declare that the project work entitled “Credit Card Fraud Detection By Using Machine Learning” has been prepared by me during the academic year 2023-25 under the guidance of Mr. Sai Amrit Patnaik, AI Research Engineer, Lens Corporation Pvt Ltd. I assert that this project is a piece of original bona-fide work done by me. It is the outcome of my own effort and that it has not been submitted to any other university for the award of any degree.

Komal Dev

Signature of Student

CERTIFICATE

This is to certify that ‘Komal Dev’ of Amity University Online has carried out the project work presented in this project report entitled “Credit Card Fraud Detection By Using Machine Learning” for the award of MBA (Data Science) under my guidance. The project report embodies results of original work, and studies are carried out by the student herself. Certified further, that to the best of my knowledge the work reported herein does not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Sai Amrit Patnaik

Signature

Sai Amrit Patnaik

AI Research Engineer

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1. Introduction

Credit card fraud involves the unauthorized use of payment cards, including credit or debit cards, to complete transactions. Fraudsters often acquire sensitive card information through insecure websites. When card details are compromised, the repercussions extend to all parties—cardholders, issuing banks, and merchants. Hence, early detection of fraudulent activity is essential. To counteract this threat, businesses and financial institutions, especially in e-commerce, employ advanced measures to prevent fraudsters from breaching their systems. By leveraging machine learning techniques, they analyze transaction patterns and user behavior to detect fraudulent activities. This automated identification process is known as "credit card fraud detection”.

1.1 Annual Financial Impact

Understanding the financial ramifications of credit card fraud is crucial for businesses and consumers. Recent statistics highlight the scale of the issue and stress the need for effective prevention strategies.

1.2 Estimated Global Losses

In 2022, global card fraud losses amounted to $33.5 billion (Nilson, 2023), increasing from $28.4 billion in 2020 and $27.9 billion in 2018.

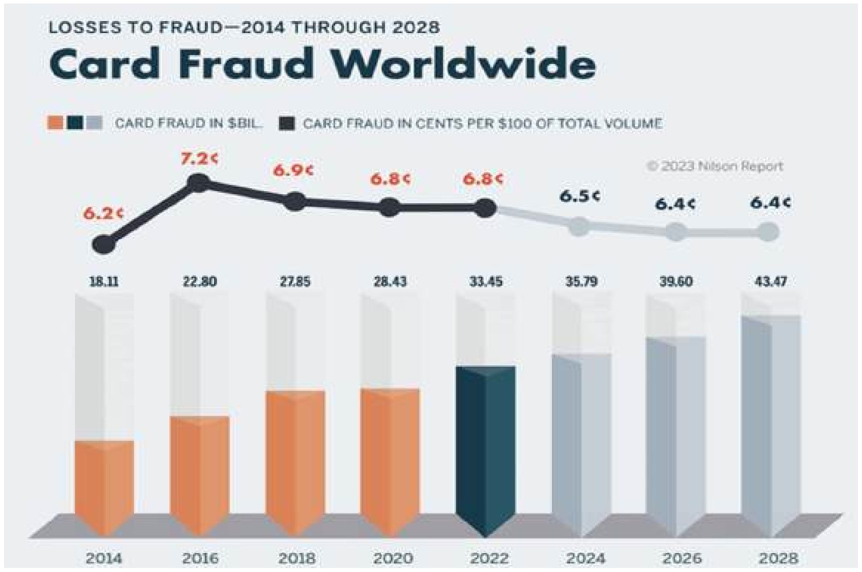


Fig 1.1 Estimated losses to fraud. Source: Nilson, 2023

Fraud per transaction volume reached a peak of 7.2¢ in 2016 but later declined, stabilizing at 6.8¢ by 2022. This equates to 6.8 cents lost for every $100 spent. Projections suggest a further decrease to 6.4¢ by 2026 and 2028. Despite these reductions, total fraud losses are expected to rise to $43.47 billion by 2028 due to increasing transaction volumes, indicating that enhanced security measures might be mitigating the fraud rate per transaction.

1.3 Regional Breakdown

a. North America

In 2023, card fraud in the US resulted in $466 million in losses from 197,785 reported cases (FTC, 2024). By 2024, 60% of US cardholders reported suspicious account activity, a decrease from 65% in 2022. However, the median cost of fraudulent charges rose from $79 in 2022 to $100 in 2024, leading to $5 billion in fraudulent purchases annually (Security, 2024).

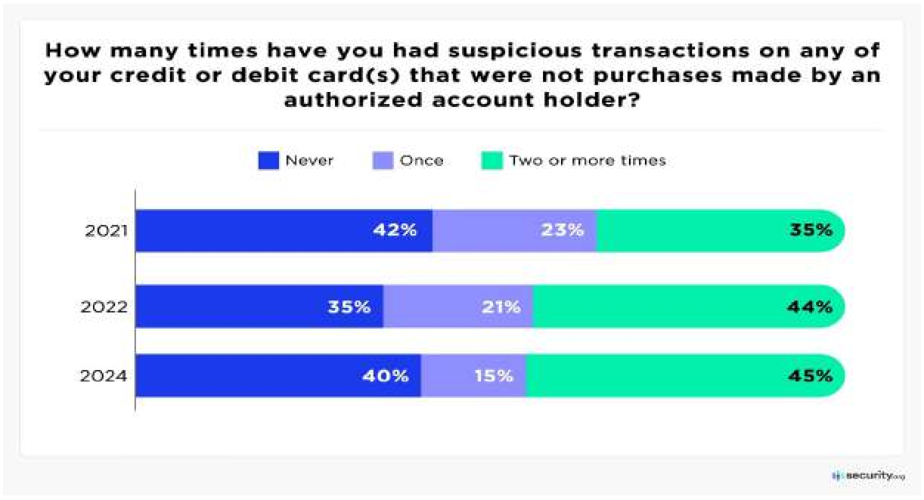


Fig 1.2 Suspicious Transactions

b. Europe

The European Union (EU) reported €1.53 billion in fraudulent transactions in 2021, an 11% reduction from 2020. Fraudulent transactions accounted for 0.028% of all transactions in 2021, down from 0.036% in 2020. The UK recorded £540 million in card fraud losses in 2022, constituting 45% of financial fraud (ECB, 2023; UK Finance, 2022).

c. Asia-Pacific

In 2023, card fraud in the Asia-Pacific (APAC) region totaled $11.9 billion, making it the most common payment fraud type. Other methods included digital wallets, payment apps (26%), and cryptocurrency fraud (8%) (Nasdaq, 2024; LexisNexis, 2023).

d. Latin America

Latin America faces the highest global rates of card fraud due to weaker detection systems and regulations favoring cardholders. Fraud rates were 97% higher than in North America and 222% higher than in the APAC region from 2019 to 2022. Fraud risks have grown alongside the region's expanding online market, evidenced by a 518% surge in new businesses on Stripe in 2021 (Stripe, 2022).

1.4 Types of Credit Card Fraud

Card-Not-Present (CNP) Fraud

In 2024, 93% of US fraudulent transactions occurred without the physical card being lost or stolen, making CNP fraud the predominant type (Security, 2024). In 2023, online CNP fraud and check fraud led to $174 million in losses from 13,718 cases (FBI, 2023).

In the EU, CNP fraud accounted for 84% of card fraud value, totaling €1.3 billion in 2021. This figure has been steadily rising over time (ECB, 2023).

This analysis underscores the widespread nature of credit card fraud and the importance of robust prevention and detection systems to minimize financial losses.

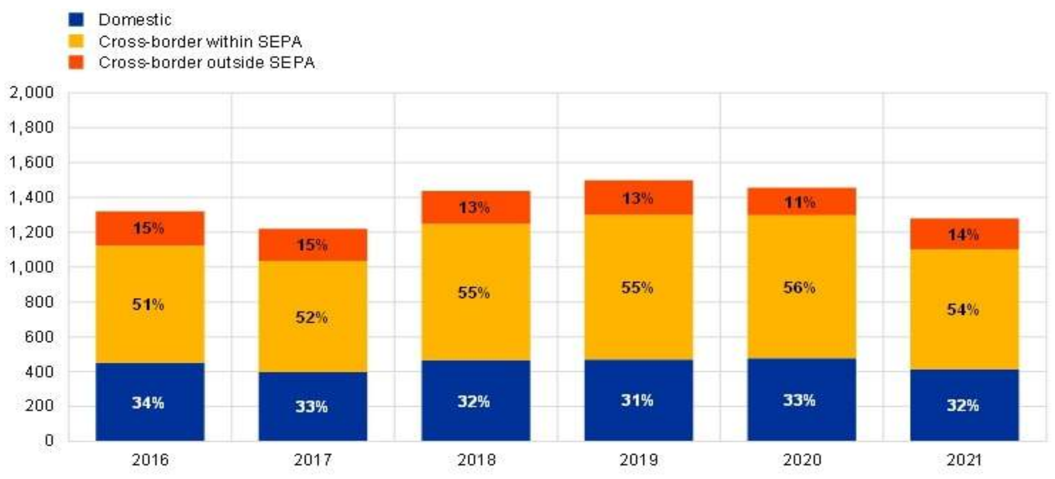


Fig 1.3 CNP

The total value of CNP fraud in the EU, as reported by the ECB in 2023, experienced a 12% decrease compared to 2020.

1.5 How does Credit Card Fraud work?

Credit cards are essential financial tools used for online payments, purchasing essentials, and earning rewards. However, they are also targets for fraud in various forms:

1. **Lost or Stolen Cards**: Thieves intercept cards from the mail or steal them, using them for unauthorized transactions. Blocking and re-issuing these cards can be a hassle for both customers and companies, with some institutions keeping new cards blocked until the cardholder confirms receipt.
2. **Card Abuse**: Some individuals intentionally make purchases without the intention of repaying. These customers may avoid collection efforts and sometimes declare bankruptcy, leading to millions of dollars in losses each year.
3. **Identity Theft**: Fraudsters may steal personal details or use fake information to apply for credit cards. Even if the card is blocked, the fraudster may still use it if they've successfully bypassed security measures.
4. **Merchant Abuse**: Certain merchants engage in fraudulent activities, such as reporting fake transactions for money laundering or obtaining and misusing legitimate credit card data.

1.6 Traditional Fraud Detection vs. Machine Learning:

Traditional fraud detection methods like CVV verification, geolocation tracking, and IP address monitoring have been widely used, but as fraudsters adopt more advanced techniques, these methods are less effective.

1.7 Fraud Statistics in the EU:

In 2021, **card-present fraud** at ATMs amounted to €74 million, representing a 4% decrease from 2020. At **point-of-sale (POS)** terminals, the total was €177 million, a 7% decline from the previous year. The overall decrease in card-present fraud at both ATMs and POS terminals was 6% compared to 2020. This decline followed a 28% drop in 2020 compared to 2019. Lost or stolen cards were the primary cause of this fraud, accounting for 88% of ATM fraud and 56% of POS fraud.

1.8 Identity Theft and Account Takeover

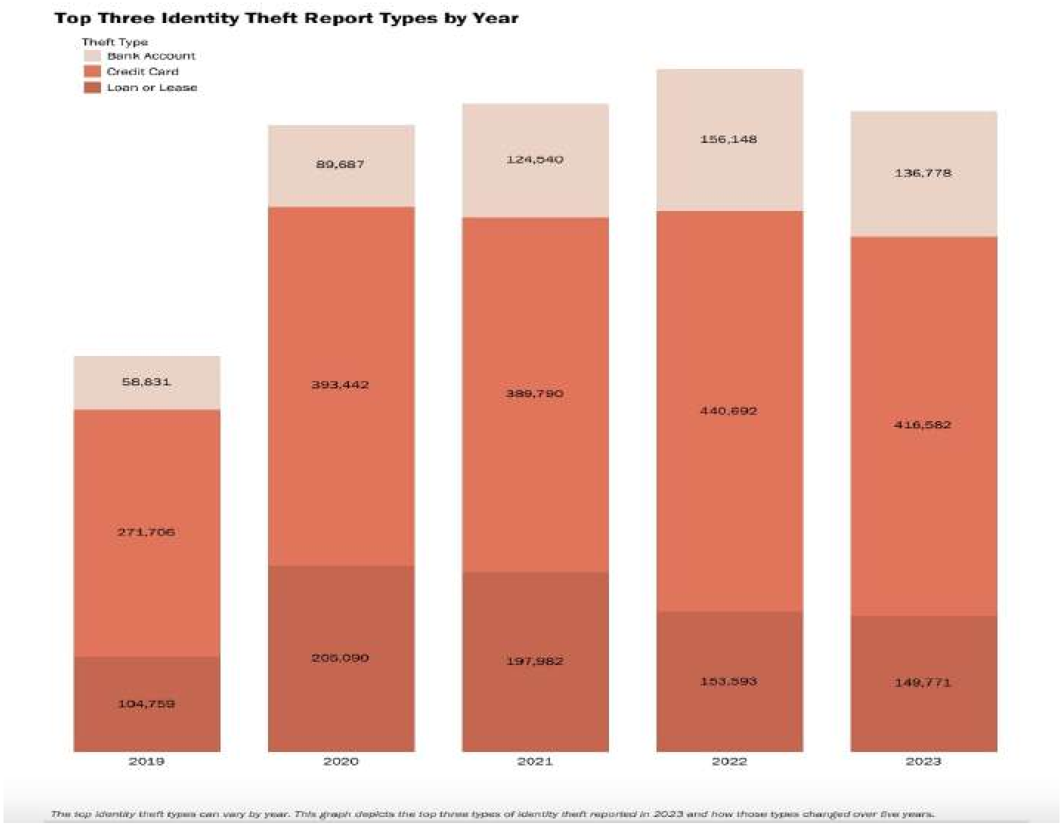
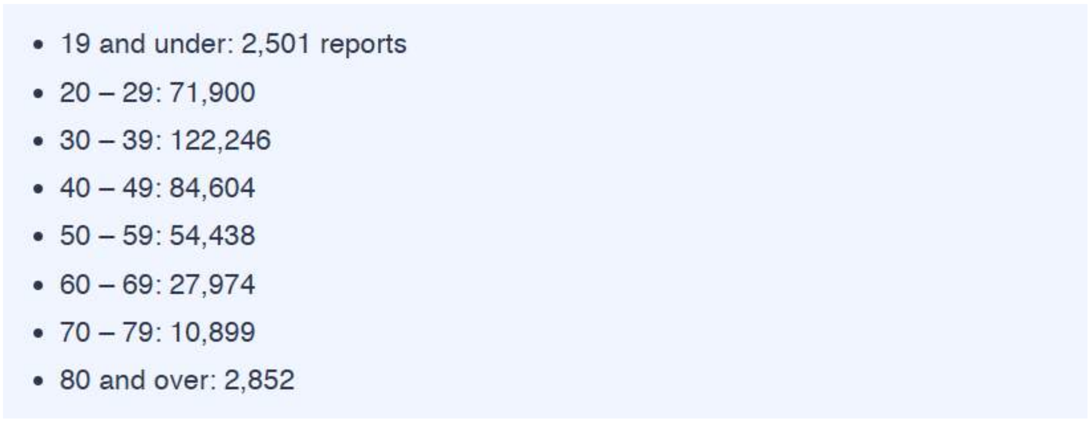


Fig 1.4 Identity theft types. Source: FTC, 2024C

In 2023, credit card fraud became the most commonly reported form of identity theft in the US. The **FTC** reported 416,582 cases, with 381,122 involving new accounts, a 7% decline compared to 2022. A significant portion of these reports (122,246) came from individuals aged 30-39, reflecting the age group's vulnerability.

Here is the breakdown by age group:



- 19 and under: 2,501 reports
- 20 – 29: 71,900
- 30 – 39: 122,246
- 40 – 49: 84,604
- 50 – 59: 54,438
- 60 – 69: 27,974
- 70 – 79: 10,899
- 80 and over: 2,852

E-commerce Fraud Trends:

In 2023, **online shopping fraud** accounted for 368,379 reports and $392 million in losses. This resulted in a median loss of $125. Globally, 2.9% of e-commerce revenue was lost to payment fraud, a decrease from 3.6% in 2022. Fraud indicators showed improvement in 2023, with a reduction in the percentage of fraudulent e-commerce orders:

- Domestic e-commerce fraud: 2.6% (down from 3.1% in 2022)
- International e-commerce fraud: 3.0% (down from 3.4% in 2022)
- Chargebacks due to fraud: 2.6% (down from 3.1% in 2022)

Merchants, especially **small and medium-sized businesses (SMBs)**, are allocating more resources to combat fraud. SMBs with annual e-commerce sales between $50,000 and $5 million increased their fraud management spending from 6% to 12% of their revenue in the last year.

1.9 Demographic and Behavioral Insights

Credit card fraud varies across different demographics, and understanding these patterns can help improve fraud prevention and detection. Analysis of fraud statistics reveals trends in **card-not-present**, **card-present**, and **online fraud** across age and gender groups.

1.10 Age and Gender Distribution

The impact of credit card fraud decreases with age, but the median losses increase:

- **19 and under**: Median losses of $187, with 38% using debit or credit cards for purchases.
- **Gender Gap**: In 2021, 21.8 million men reported being victims of credit card fraud, compared to 15.3 million women.

By analyzing these trends, financial institutions can better target fraud prevention efforts, improve security systems, and educate consumers on safe practices to reduce the risk of fraud.

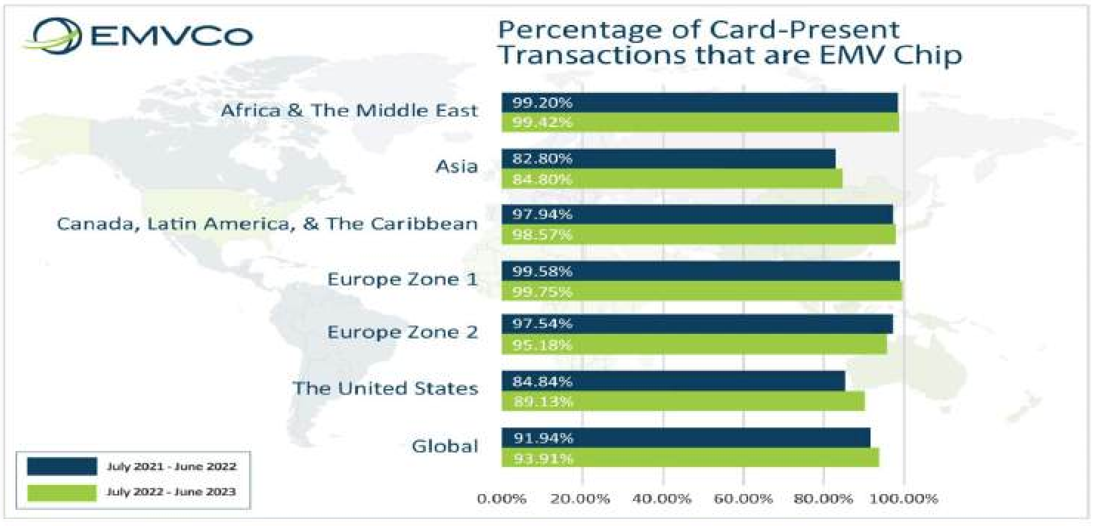
1.11 Consumer Awareness and Reporting

Consumer awareness plays a vital role in mitigating credit card fraud, with 2024 statistics highlighting both risky and protective behaviors among cardholders:

- **Unsafe Credit Card Habits**:

    - **80%** of US cardholders admitted to engaging in at least one risky credit card habit (Security, 2024). These included:
        - **52%** using the same card for autopay and everyday purchases.
        - **40%** reusing passwords across online accounts.
        - **41%** storing credit card details in browsers or websites.
        - **34%** connect to public Wi-Fi.
        - **8%** using free VPNs.
- **Protective Credit Card Habits**:
    - Despite the prevalence of risky behaviors, **95%** of cardholders practiced at least one safe habit, and **81%** followed two or more protective measures:
        - **79%** regularly review credit card statements.
        - **61%** subscribe to activity alerts via email or text.
        - **55%** use multi-factor authentication or Face ID for online accounts.
        - **37%** use online password managers.
        - **31%** subscribe to credit monitoring services.
- **Fraud Reporting**:
    - In 2024, **96%** of US cardholders who reported fraudulent charges received their money back (Security, 2024):
        - **49%** had the charges refunded or reversed.
        - **47%** had transactions blocked immediately by their credit card company.

1.12 Methods of Fraud Prevention



Data represents the most accurate possible data that could be obtained by American Express, Discover, JCB, Mastercard, UnionPay and Visa for transactions processed by them during the noted period. The transaction data reflects an average of 12 months' data as reported by all members to take into consideration seasonal variations. To qualify as an 'EMV chip transaction', both the card and terminal used during a transaction must be EMV chip-enabled. Data is reported from the acquirer perspective. These figures may not include offline transactions, 'on us' transactions (defined as a transaction handled exclusively by another processor) and/or transactions processed by non-EMVCo member institutions, such as national payment networks.

Fig 1.5 Percentage of Card present transaction that are EVM chip

To prevent credit card fraud, a variety of strategies and technologies are employed:

Technological Measures

1. **EMV Chips**:
   - EMV (Europay, MasterCard, and Visa) chips have significantly reduced card-present fraud. Unlike traditional magnetic stripe cards, EMV chips generate a unique transaction code for every purchase, making it harder for fraudsters to replicate card data.
   - As of Q2 2023, **94%** of global card-present transactions were processed using EMV chip-enabled cards (EMVCo, 2023), an increase from **92%** in 2022.

2. **AI-Driven Fraud Detection**:
    - AI and machine learning algorithms are essential in detecting fraud by analyzing transaction patterns in real-time and identifying anomalies. These technologies learn from vast datasets, improving their ability to detect fraud over time.
    - In 2023, **22%** of organizations reported using AI for fraud detection (IBM, 2024).

Regulatory Measures

1. **PCI DSS Compliance**:
    - The **Payment Card Industry Data Security Standard (PCI DSS)** mandates businesses to protect cardholder data during transactions. Compliance with PCI DSS is a key regulatory measure for fraud prevention.

1.12 Future Trends in Credit Card Fraud

As fraud tactics evolve, financial institutions must continue to adapt. The increase in credit card scams and identity theft observed in 2023 underlines the need for advanced fraud prevention systems.

1.13 Emerging Threats

New types of fraud are emerging, which add complexity to the financial landscape:

1. **Synthetic Identity Fraud**:

    - Fraudsters combine real and fabricated personal details to create fake identities, making it harder for vendors to identify them as fraudulent. This crime is one of the fastest-growing financial crimes in the US (Deloitte, 2023).

2. **Generative AI and Deepfakes**:
    - **Generative AI** is being used to produce convincing fake audio, video, and images (deep fakes), which can be used for identity theft and other fraud types (Deloitte, 2023).
3. **Fraud as a Service (FaaS)**:
    - Cybercriminals are offering fraud-related tools to others through the dark web, making it easier for individuals with little technical expertise to commit fraud (Comply Advantage, 2024).
4. **Contactless Fraud**:
    - As contactless payments via NFC (near-field communication) technology increase, there are rising concerns about fraud through mobile phones, despite the convenience they offer (Comply Advantage, 2024).

1.14 Advancements in Fraud Detection

As fraudsters develop more sophisticated schemes, the following trends in fraud detection are gaining prominence:

1. **Ensemble Machine Learning Models**:
    - These models combine multiple machine learning algorithms like **Support Vector Machines (SVM)**, **K-Nearest Neighbors (KNN)**, and **Random Forest**

(RF) to improve fraud detection by addressing issues like data imbalance and false positives (Khalid et al., 2024).

2. **Behavioral Biometrics and Advanced Analytics**:
   - **Behavioral biometrics** track user patterns to identify suspicious activities, adding another layer of fraud prevention. This is critical as fraudsters bypass traditional security measures. Advanced analytics help in detecting patterns and improving the accuracy of fraud detection (Deloitte, 2023).
3. **Increased Investment in Fraud Prevention**:
   - Financial institutions and merchants are investing more in fraud detection technologies. In 2022, spending on online fraud detection reached **$9.3 billion**, reflecting a **22%** increase from the previous year (Juniper Research, 2022).

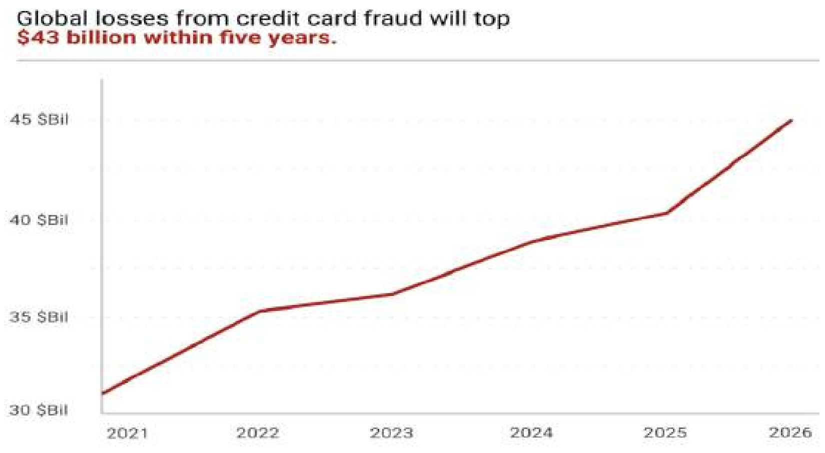


Fig 1.6 Global losses from credit card fraud

1.15 Credit Card Fraud Detection Techniques using Machine Learning

Credit card fraud detection can be approached using two key machine learning methods:

1. **Unsupervised Learning**:
    - **Algorithms** like **Isolation Forest**, **One-Class SVM**, and **Local Outlier Factor (LOF)** are used in unsupervised learning, which does not require labeled data for model training. These methods identify patterns and anomalies in data, making them effective for detecting fraud in datasets where fraudulent transactions are rare.
2. **Supervised Learning**:
    - **Supervised learning** algorithms such as **Decision Tree, Logistics Regression, Support Vector Machine** and **K-Nearest Neighbors (KNN)**, are trained on labeled datasets, where transactions are tagged as fraudulent or legitimate. These models can classify new transactions based on the patterns learned during training.

These techniques, particularly when applied with **ensemble models** and **advanced analytics**, are becoming increasingly crucial in the fight against credit card fraud, improving the speed and accuracy of fraud detection systems.

CHAPTER 2. REVIEW OF LITERATURE

Zareapoor and his research team used multiple techniques to determine the best-performing model for detecting fraudulent transactions, which was established using the Accuracy of the model, the speed of detection and the cost. The models used were Neural Network, Bayesian Network, SVM, KNN and. The comparison table in the research paper showed that the Bayesian Network was high-speed in finding fraudulent transactions with high Accuracy. The N.N. performed well, as the detection was fast, with a medium accuracy. KNN's speed was good with medium Accuracy, and finally, SVM scored one of the lower scores, as the speed was low, and the Accuracy was medium. As for the cost, All models built were expensive (Zareapoor et al., 2012).

The model used by Alenzi and Aljehane to detect Fraud in credit cards was Logistic Regression. Their model scored 97.2% in Accuracy, 97% sensitivity and 2.8% Error Rate. A comparison was performed between their model and two other classifiers, which are. They were voting Classifier and KNN. V.C. scored 90% in Accuracy, 88% sensitivity and 10% error rate, as for KNN where k = 1:10, the Accuracy of the model was 93%, the sensitivity 94% and 7% for the error rate (Alenzi & Aljehane, 2020).

Maniraj's team built a model to recognize if any new transaction is Fraud or non-fraud. Their goal was to get 100% in detecting fraudulent transactions and try to minimize the incorrectly classified fraud instances. Their model has performed well as they got 99.7% of the fraudulent transactions (Maniraj et al., 2019).

The classification approach used by Dheepa and Dhanapal was the behavior-based classification approach, using a Support Vector Machine, where the behavioral patterns of the customers were analyzed to distinguish credit card fraud, such as the amount, date, time, place, and frequency of card usage. The Accuracy achieved by their approach was more than 80% (Dheepa & Dhanapal, 2012).

Mailini and Pushpa proposed using KNN and Outlier detection in identifying credit card fraud. After performing their model oversampled data, the authors found that the most suitable method for detecting and determining target instance anomaly is KNN, which showed that it is most suited to detecting Fraud with memory limitation. As for Outlier detection, the computation and memory required for credit card fraud detection is much less in addition to working faster and better in large online datasets. However, their work and results showed that KNN was more accurate and efficient (Malini & Pushpa, 2017).

Maes and his team proposed using Bayesian and Neural Networks to detect credit card fraud. Their results showed that Bayesian performance is 8% more effective in detecting Fraud than ANN, meaning BBN sometimes detects 8% more fraudulent transactions. In addition to the Learning times, ANN can go up to several hours, whereas BBN takes only 20 minutes (Maes et al., 2002).

The team compared the usage of three ML techniques in detecting credit card fraud: the first is KNN, the second is Naïve Bayes, and the third is Logistic Regression. They sampled different distributions to view the various outcomes. The top Accuracy of the 10:90 distribution is Naïve Bayes with 97.5%, then KNN with 97.1%, Jain's team used several ML techniques to distinguish credit card fraud; three of them are SVM, ANN and KNN. Then, to compare the outcome of each model, they calculated the true positive (T.P.), false Negative (F.N.), false positive (F.P.), and true negative (T.N.) generated. ANN scored 99.71% accuracy, 99.68% precision, and 0.12% false alarm rate. SVM accuracy is 94.65%, 85.45% for the precision, and 5.2% false alarm rate. Moreover, finally, the Accuracy of KNN is 97.15%, the precision is 96.84%, and the false alarm rate is 2.88% (Jain et al., 2019)

Dighe and his team used KNN, Logistic Regression and Neural Networks, multilayer perceptron and Decision Tree in their work, then evaluated the results regarding numerous accuracy metrics. Of all the models created, the best performing one is KNN, which scored 99.13%, then in second place performing model at 96.40% and in last place is logistic Regression with 96.27% (Dighe et al., 2018).Sahin and Duman used four Support Vector Machine methods in detecting credit card fraud. SVM) Support Vector Machine with RBF, Polynomial, Sigmoid, and Linear Kernel, all models scored 99.87% in the training model and 83.02% in the testing part of the model (Sahin & Duman, 2011).

CHAPTER 3. RESEARCH OBJECTIVES AND METHODOLOGY

3.1 RESEARCH OBJECTIVES

The goal of this research is to determine the most effective machine learning model for detecting credit card fraud. By evaluating the performance of multiple algorithms, the study seeks to identify a robust approach to fraud detection. The dataset will be preprocessed to enhance accuracy, followed by the implementation and comparison of four models: K-Nearest Neighbors (KNN) with K=3 and K=7, Decision Tree, Support Vector Machine (SVM), and Logistic Regression. The models will be assessed using relevant metrics to identify the best-performing approach for this task.

3.2 RESEARCH PROBLEM

Credit card fraud poses a significant challenge for financial institutions, resulting in substantial financial losses and diminished customer confidence. Current fraud detection systems face difficulties in handling the ever-evolving nature of fraudulent behavior and the class imbalance in datasets, where legitimate transactions far outnumber fraudulent ones. This study aims to address these challenges by identifying the most effective machine learning algorithm among KNN (with K=3 and K=7), Decision Tree, SVM, and Logistic Regression. The ultimate goal is to develop a data-driven, reliable detection system that minimizes false positives and false negatives while improving detection accuracy and resilience.

3.3 RESEARCH DESIGN

1. **Exploratory Research Phase**

- **Objective**: To examine the dataset's characteristics, identify patterns related to fraud, and understand feature significance.

- **Methodology**: Conduct descriptive analysis to uncover trends, correlations, and key feature relationships.

2. **Experimental Research Phase**

- **Objective**: To develop, optimize, and evaluate machine learning models for fraud detection.
- **Steps**:
  - **Data Preprocessing**:
    - Handle missing values and outliers.
    - Normalize and scale features for uniformity.
    - Address class imbalance using techniques like SMOTE.
  - **Model Development**:
    - Implement the selected models (KNN with K=3 and K=7, Decision Tree, SVM, and Logistic Regression) in Google Colab.
    - Split the dataset into training and testing subsets, ensuring stratification for balanced representation.
  - **Hyperparameter Tuning**:
    - Optimize models using techniques like grid search or random search for improved performance.

3. **Comparative Analysis Phase**

- **Objective**: To evaluate the models' performance and determine their effectiveness in fraud detection.
- **Steps**:
  - Use metrics such as precision, recall, F1-score, and ROC-AUC for evaluation.

    - Perform k-fold cross-validation for more reliable results.
    - Analyze false positives and false negatives to assess trade-offs and determine the best model.

4. **Conclusion and Model Selection Phase**

- **Objective**: To identify and recommend the most suitable model for fraud detection.
- **Steps**:
    - Summarize the findings from the analysis phase.
    - Choose the model that achieves high accuracy while maintaining a balance between precision and recall, ensuring effectiveness in real-world applications

3.4 TYPE OF DATA USED

The dataset used in this study was sourced from Kaggle and consists of credit card transactions conducted by European users over a two-day period in 2013. It includes 31 attributes and 2,84,807 rows in which we took 5,000 variables, represents the non-fraudulent transactions, and with 492 variables, represents the fraudulent transactions for better prediction, accuracy and insights. Twenty-eight of these attributes represent numerical features that have been transformed using Principal Component Analysis (PCA) to maintain customer anonymity. The remaining three attributes are:

- Time: The elapsed seconds between each transaction and the first transaction in the dataset.
- Amount: The monetary value of each transaction.
- Class: A binary variable indicating whether the transaction is fraudulent (1) or legitimate (0).

This dataset provides a comprehensive foundation for evaluating fraud detection models.

Dataset : https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

3.5 DATA COLLECTION METHOD

The dataset was obtained from Kaggle, a reliable platform offering high-quality, publicly available datasets. The collection process involved:

1. **Source Identification**: Kaggle was selected for its reputation and the dataset's alignment with the project requirements.
2. **Dataset Retrieval**: The dataset was downloaded directly from Kaggle's repository.
3. **Verification**:
    - Reviewed metadata to ensure all necessary attributes, such as PCA-transformed features, timestamps, transaction amounts, and fraud indicators, were included.
    - Checked the dataset for missing files or corruption.
4. **Initial Analysis**: Confirmed the presence of 31 features and 5,000 records, validating the PCA transformation and the class distribution.

This pre-collected dataset enables the research to focus on preprocessing, modeling, and analysis without needing to gather sensitive new data.

3.6 SAMPLE SIZE

The dataset consists of 2,84,807 records, with a severe class imbalance:

- Non-fraudulent transactions: 2,84,807 but i have took 5,000 for better insights and accuracy (approximately 99.83%)

- Fraudulent transactions: 492 (approximately 0.17%)

The "Class" attribute was converted from an integer format to a categorical variable, labeling "0" as "Not Fraud" and "1" as "Fraud" to facilitate model development and visualization. The significant class imbalance necessitates the use of balancing techniques to improve model performance and ensure robust fraud detection.

3.7 SAMPLING TECHNIQUE

For this research, the dataset was sourced from Kaggle, which already contains a comprehensive set of 2,84,807 credit card transaction records. Since the dataset includes a significant imbalance between fraudulent and non-fraudulent transactions, a stratified sampling technique was applied to ensure that both classes (fraudulent and non-fraudulent transactions) were adequately represented during model training and evaluation.

The sampling process involved:

1. Stratification: The dataset was divided into two strata based on the "Class" attribute, where Class = 0 indicates non-fraudulent transactions and Class = 1 indicates fraudulent transactions. This was done to ensure that both classes were properly represented in the training and test sets, despite the class imbalance.

2. Random Sampling: Random sampling was then used within each class to split the dataset into training and testing sets. The dataset was typically divided into 80% for training and 20% for testing.

3. Handling Imbalance: Given the severe imbalance between the two classes (non-fraudulent transactions vastly outnumbering fraudulent ones), techniques like oversampling (e.g., SMOTE) or undersampling could be applied as necessary to balance the classes during the model training phase.

This stratified sampling method ensures that the models can be trained on a representative subset of the data, improving the model's ability to detect fraudulent transactions effectively, despite the class imbalance.

3.8 DATA ANALYSIS TOOLS:

This research employed a range of powerful data analysis and machine learning tools to ensure efficient data preprocessing, model development, and performance evaluation. Below is a breakdown of the tools and methods utilized:

1. **Python**:
   - Python was the core programming language used in this research due to its flexibility and the vast ecosystem of libraries it offers. Python facilitated the entire process, from data handling to the development of machine learning models, making it an ideal choice for this study.
2. **Google Colab**:
   - Google Colab provided an interactive cloud-based environment for executing the research. It allowed for seamless integration of code, documentation, and visualizations in an accessible, collaborative manner, which was particularly useful for code sharing and iterative development.
   -
3. **Pandas**:
   - **Pandas** was crucial for managing and cleaning the dataset. This library provided powerful functions for handling missing data, transforming features, and splitting the dataset into training and testing sets. It was essential for data preprocessing

tasks, such as normalizing numerical features and ensuring the data was ready for modeling.

4. **NumPy**:
   - **NumPy** supported the research by offering efficient numerical operations. It played an important role in handling arrays, performing mathematical transformations, and scaling features, which are critical for preparing the data for machine learning algorithms.
5. **Matplotlib & Seaborn**:
   - For data visualization, **Matplotlib** and **Seaborn** were used. **Matplotlib** generated basic visualizations such as histograms, while **Seaborn** was employed for advanced visualizations like heatmaps and correlation matrices. These visual tools helped explore the relationships between different features, identify trends, and assess the performance of the machine learning models.
6. **Scikit-learn**:
   - **Scikit-learn** was instrumental for the machine learning part of the research. It provided a wide range of algorithms and utilities for model building, training, and evaluation. The models developed included K-Nearest Neighbors (KNN), Decision Trees, Support Vector Machines (SVM), and Logistic Regression. Scikit-learn also supported tasks such as splitting the dataset, scaling features, and evaluating model performance using metrics like accuracy and precision.
7. **SMOTE (Synthetic Minority Over-sampling Technique)**:
   - To address the class imbalance in the dataset, **SMOTE** was utilized. This technique generates synthetic instances of the minority class (fraudulent

transactions), ensuring that the model has enough data to learn the characteristics of fraud, thus improving performance in detecting fraud.

This combination of tools and techniques ensures a systematic approach to solving the problem of fraud detection. From data cleaning to model evaluation, each tool played a crucial role:

1. **Efficiency:** Automating repetitive tasks like scaling and splitting data.
2. **Accuracy:** Addressing challenges such as class imbalance (using SMOTE) and selecting the best model (using Scikit-learn).
3. **Transparency:** Leveraging visualization tools to make the results understandable and actionable.

This methodology is not only effective for fraud detection but can be generalized to other binary classification problems.

CHAPTER 4. DATA DESCRIPTIONS, ANALYSIS & INTERPRETATION

4.1.1 Data Description

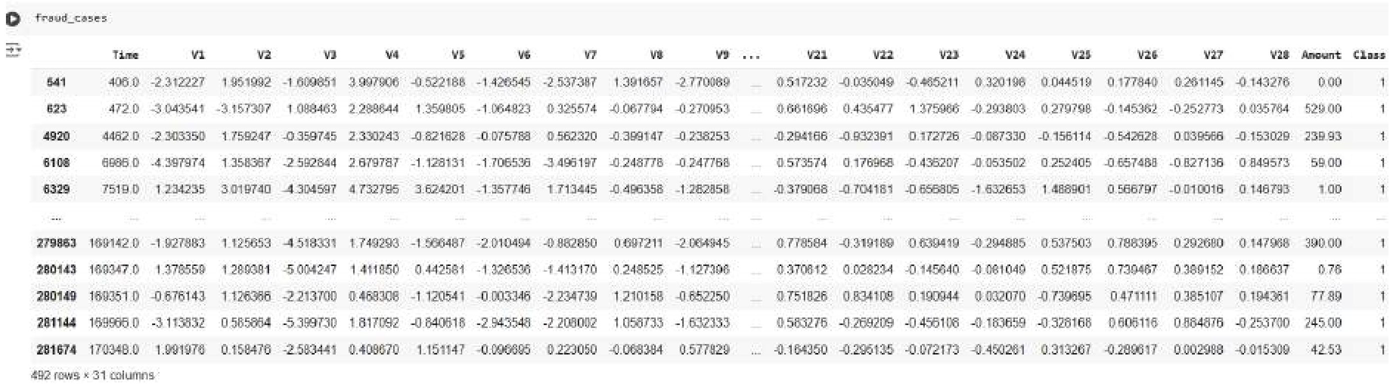
The dataset used in this study is characterized as follows:

1. **Target Attribute**:
   - The **Class** attribute represents the target variable, with two possible values: 0 for "Not Fraud" and 1 for "Fraud." Initially in integer format, this attribute was converted to a categorical type for easier handling during model training and visualization.

2. **Class Distribution**:
    - ○ The dataset exhibits a significant class imbalance, with 5,000 transactions marked as non-fraudulent and only 492 transactions identified as fraudulent. This imbalance is a typical challenge in fraud detection tasks, requiring specialized techniques such as SMOTE to ensure that the model doesn't become biased toward the majority class.
3. **Visualizations**:
    - ○ The visual representations of the dataset's structure and class distribution were critical for understanding the nature of the data and determining the best approach for preprocessing and model training. They highlighted the imbalance and informed the choice of techniques for handling it.

The research benefited from the use of a comprehensive suite of tools—**Python**, **Google Colab**, **Pandas**, **NumPy**, **Matplotlib & Seaborn**, **Scikit-learn**,—which collectively ensured efficient data management, model development, and evaluation. These tools provided the necessary functionality to preprocess the imbalanced dataset, build accurate machine learning models, and enhance the robustness of the fraud detection system. The synergy of these tools enabled the research to deliver meaningful insights into credit card fraud detection.

4.1.2 DATA ANALYSIS

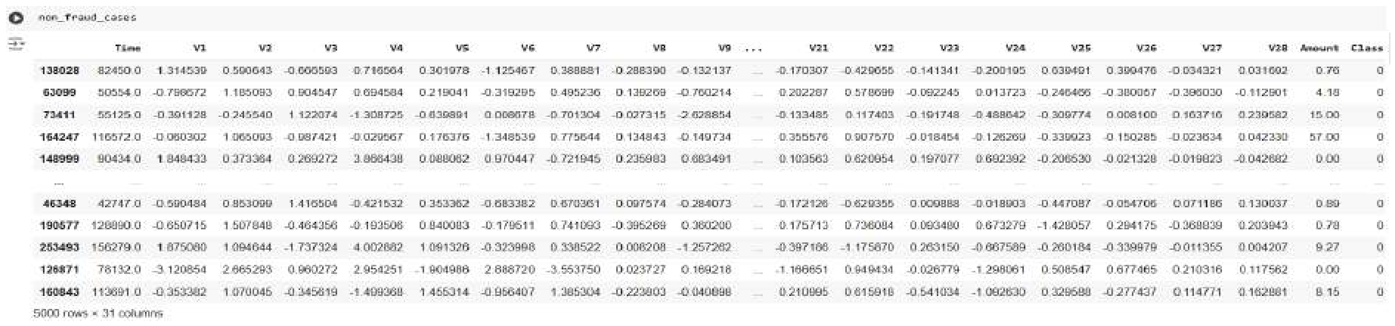


```
fraud_cases
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
541	406.0	-2.312227	1.951992	-1.609851	3.997906	-0.522188	-1.426545	-2.537387	1.391657	-2.770089	...	0.517232	-0.035049	-0.465211	0.320198	0.044519	0.177840	0.261145	-0.143276	0.00	1
623	472.0	-3.043541	-3.157307	1.088463	2.288644	1.359805	-1.064823	0.325574	-0.067794	-0.270953	...	0.661696	0.435477	1.375966	-0.293803	0.279798	-0.145362	-0.252773	0.035764	529.00	1
4920	4462.0	-2.303350	1.759247	-0.359745	2.330243	-0.821628	-0.075788	0.562320	-0.399147	-0.238253	...	-0.294166	-0.932391	0.172726	-0.087330	-0.156114	-0.542628	0.039566	-0.153029	239.93	1
6108	6986.0	-4.397974	1.358367	-2.592844	2.679787	-1.128131	-1.706536	-3.496197	-0.248778	-0.247768	...	0.573574	0.176968	-0.436207	-0.053502	0.252405	-0.657488	-0.827136	0.849573	59.00	1
6329	7519.0	1.234235	3.019740	-4.304597	4.732795	3.624201	-1.357746	1.713445	-0.496358	-1.282858	...	-0.379068	-0.704181	-0.656805	-1.632653	1.488901	0.566797	-0.010016	0.146793	1.00	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.319189	0.639419	-0.294885	0.537503	0.788395	0.292680	0.147968	390.00	1
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.028234	-0.145640	-0.081049	0.521875	0.739467	0.389152	0.186637	0.76	1
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.834108	0.190944	0.032070	-0.739695	0.471111	0.385107	0.194361	77.89	1
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.269209	-0.456108	-0.183659	-0.328168	0.606116	0.884876	-0.253700	245.00	1
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.295135	-0.072173	-0.450261	0.313267	-0.289617	0.002988	-0.015309	42.53	1

492 rows × 31 columns

Fig 1.7 Fraud Cases

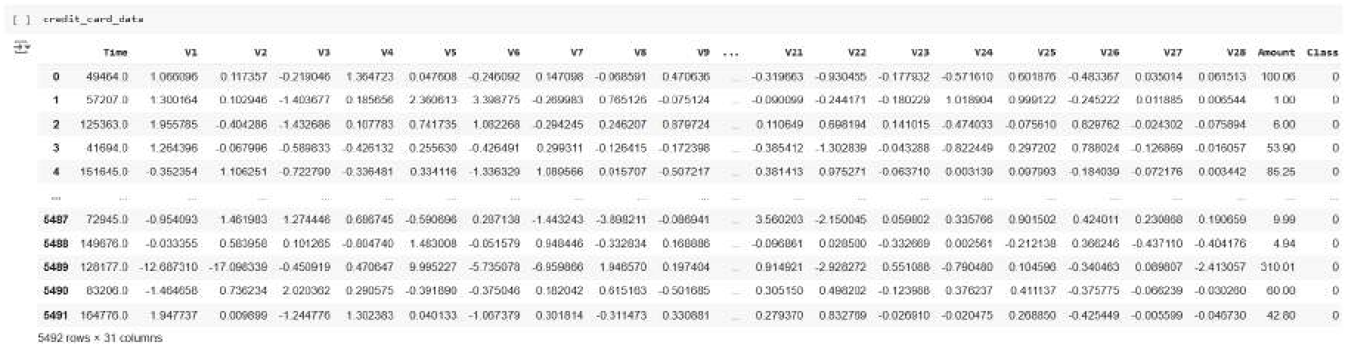


```
non_fraud_cases
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
138028	82450.0	1.314539	0.590643	-0.666593	0.716564	0.301978	-1.125467	0.388881	-0.288390	-0.132137	...	-0.170307	-0.429655	-0.141341	-0.200195	0.639491	0.399476	-0.034321	0.031692	0.76	0
63099	50554.0	-0.798672	1.185093	0.904547	0.694584	0.219041	-0.319295	0.495236	0.139269	-0.760214	...	0.202287	0.578699	-0.092245	0.013723	-0.246466	-0.380057	-0.396030	-0.112901	4.18	0
73411	55125.0	-0.391128	-0.245540	1.122074	-1.308725	-0.639891	0.008678	-0.701304	-0.027315	-2.628854	...	-0.133485	0.117403	-0.191748	-0.488642	-0.309774	0.008100	0.163716	0.239582	15.00	0
164247	116572.0	-0.060302	1.065093	-0.987421	-0.029567	0.176376	-1.348539	0.775644	0.134843	-0.149734	...	0.355576	0.907570	-0.018454	-0.126269	-0.339923	-0.150285	-0.023634	0.042330	57.00	0
148999	90434.0	1.848433	0.373364	0.269272	3.866438	0.088062	0.970447	-0.721945	0.235983	0.683491	...	0.103563	0.620954	0.197077	0.692392	-0.206530	-0.021328	-0.019823	-0.042682	0.00	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
46348	42747.0	-0.590484	0.853099	1.416504	-0.421532	0.353362	-0.683382	0.670361	0.097574	-0.284073	...	-0.172126	-0.629355	0.009888	-0.018903	-0.447087	-0.054706	0.071186	0.130037	0.89	0
190577	128890.0	-0.650715	1.507848	-0.464356	-0.193506	0.840083	-0.179511	0.741093	-0.395269	0.380200	...	0.175713	0.736084	0.093480	0.673279	-1.428057	0.294175	-0.368839	0.203943	0.78	0
253493	156279.0	1.875080	1.094644	-1.737324	4.002882	1.091326	-0.323998	0.338522	0.006208	-1.257262	...	-0.397166	-1.175670	0.263150	-0.667589	-0.260184	-0.339979	-0.011355	0.004207	9.27	0
126871	78132.0	-3.120854	2.665293	0.960272	2.954251	-1.904986	2.888720	-3.553750	0.023727	0.169218	...	-1.166651	0.949434	-0.026779	-1.298061	0.508547	0.677465	0.210316	0.117562	0.00	0
160843	113691.0	-0.353382	1.070045	-0.345619	-1.409368	1.455314	-0.956407	1.385304	-0.223803	-0.040898	...	0.210995	0.615918	-0.541034	-1.092630	0.329588	-0.277437	0.114771	0.162881	8.15	0

5000 rows × 31 columns

Fig 1.8 Non-Fraud Cases

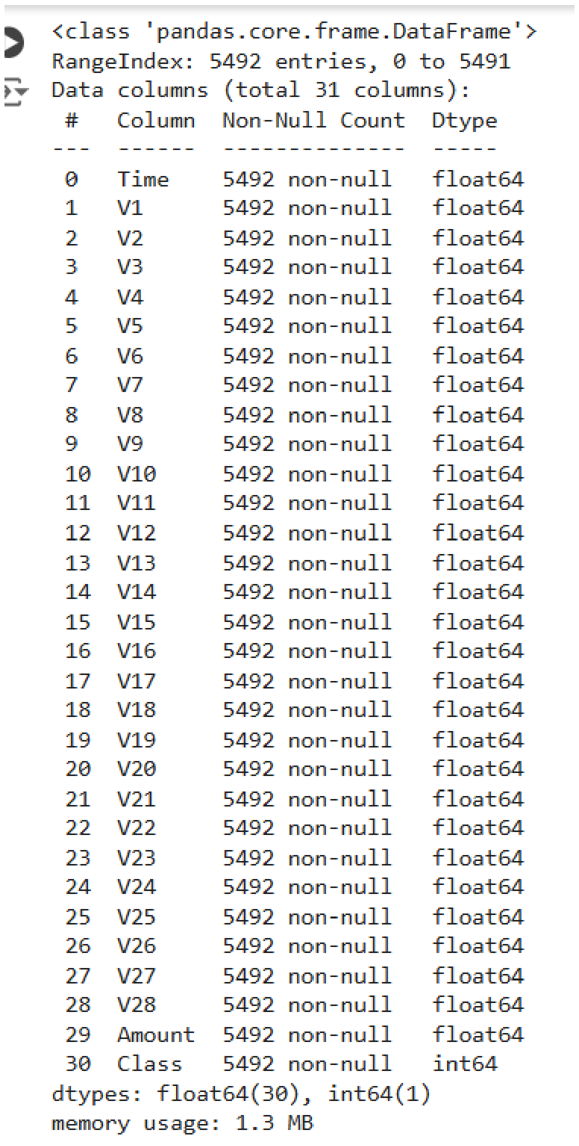


```
[ ] credit_card_data
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	49464.0	1.060096	0.117357	-0.219046	1.364723	0.047608	-0.246092	0.147098	-0.068591	0.470636	...	-0.319663	-0.930455	-0.177932	-0.571610	0.601876	-0.483367	0.035014	0.061513	100.06	0
1	57207.0	1.300164	0.102946	-1.403677	0.185656	2.360613	3.398775	-0.269983	0.765126	-0.075124	...	-0.090099	-0.244171	-0.180229	1.018904	0.999122	-0.245222	0.011885	0.006544	1.00	0
2	125363.0	1.955785	-0.404286	-1.432686	0.107783	0.741735	1.082268	-0.294245	0.246207	0.879724	...	0.110649	0.698194	0.141015	-0.474033	-0.075610	0.829762	-0.024302	-0.075894	6.00	0
3	41694.0	1.264396	-0.067996	-0.589833	-0.426132	0.255630	-0.426491	0.299311	-0.126415	-0.172398	...	-0.385412	-1.302839	-0.043288	-0.822449	0.297202	0.788024	-0.126869	-0.016057	53.90	0
4	151645.0	-0.352354	1.106251	-0.722799	-0.336481	0.334116	-1.336329	1.089566	0.015707	-0.507217	...	0.381413	0.975271	-0.063710	0.003139	0.097993	-0.184039	-0.072176	0.003442	85.25	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5487	72945.0	-0.954093	1.461983	1.274446	0.696745	-0.590696	0.207138	-1.443243	-3.898211	-0.086941	...	3.560203	-2.150045	0.059802	0.335768	0.901502	0.424011	0.230888	0.190659	9.99	0
5488	149676.0	-0.033355	0.583958	0.101265	-0.604740	1.483008	-0.051579	0.948446	-0.332834	0.168886	...	-0.096861	0.028500	-0.332669	0.002561	-0.212138	0.368246	-0.437110	-0.404176	4.94	0
5489	128177.0	-12.687310	-17.098339	-0.450919	0.470647	9.995227	-5.735078	-6.959866	1.946570	0.197404	...	0.914921	-2.928272	0.551088	-0.790480	0.104596	-0.340463	0.089807	-2.413057	310.01	0
5490	83206.0	-1.464658	0.736234	2.020362	0.290575	-0.391890	-0.375046	0.182042	0.615163	-0.501685	...	0.305150	0.498202	-0.123988	0.376237	0.411137	-0.375775	-0.066239	-0.030280	60.00	0
5491	164776.0	1.947737	0.009699	-1.244776	1.302383	0.040133	-1.067379	0.301814	-0.311473	0.330881	...	0.279370	0.832789	-0.026810	-0.020475	0.268850	-0.425449	-0.005599	-0.046730	42.80	0

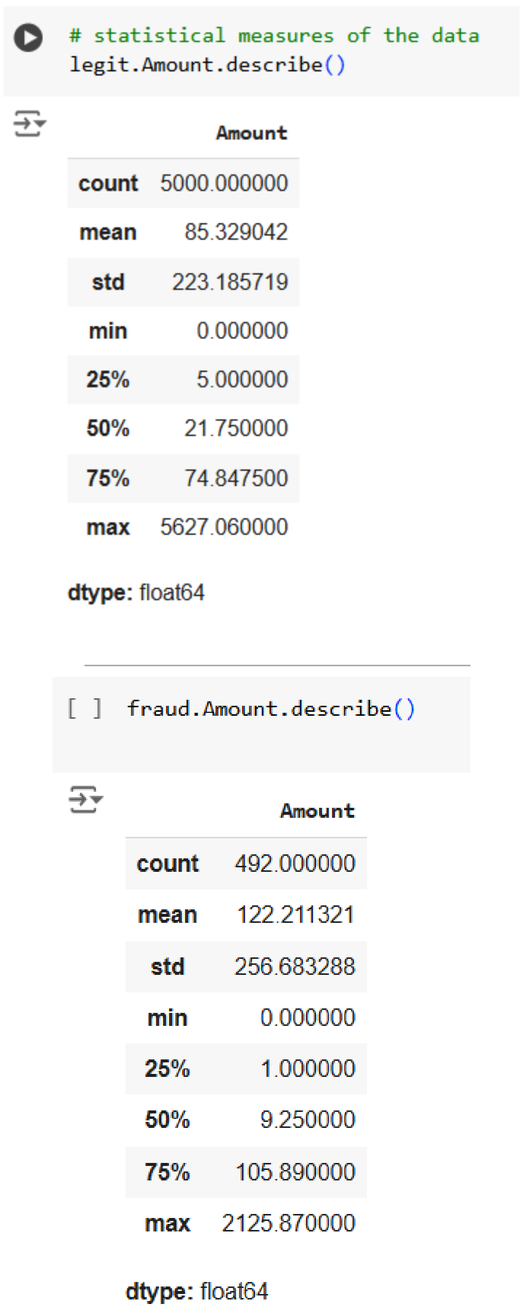
5492 rows × 31 columns

Fig 1.9 : Overall Credit Card Data



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5492 entries, 0 to 5491
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    5492 non-null   float64
 1   V1      5492 non-null   float64
 2   V2      5492 non-null   float64
 3   V3      5492 non-null   float64
 4   V4      5492 non-null   float64
 5   V5      5492 non-null   float64
 6   V6      5492 non-null   float64
 7   V7      5492 non-null   float64
 8   V8      5492 non-null   float64
 9   V9      5492 non-null   float64
 10  V10     5492 non-null   float64
 11  V11     5492 non-null   float64
 12  V12     5492 non-null   float64
 13  V13     5492 non-null   float64
 14  V14     5492 non-null   float64
 15  V15     5492 non-null   float64
 16  V16     5492 non-null   float64
 17  V17     5492 non-null   float64
 18  V18     5492 non-null   float64
 19  V19     5492 non-null   float64
 20  V20     5492 non-null   float64
 21  V21     5492 non-null   float64
 22  V22     5492 non-null   float64
 23  V23     5492 non-null   float64
 24  V24     5492 non-null   float64
 25  V25     5492 non-null   float64
 26  V26     5492 non-null   float64
 27  V27     5492 non-null   float64
 28  V28     5492 non-null   float64
 29  Amount  5492 non-null   float64
 30  Class   5492 non-null   int64  
dtypes: float64(30), int64(1)
memory usage: 1.3 MB
```

Fig 2.0 Data Structure



```
# statistical measures of the data
legit.Amount.describe()
```

```
            Amount
count  5000.000000
mean     85.329042
std     223.185719
min       0.000000
25%       5.000000
50%      21.750000
75%      74.847500
max    5627.060000

dtype: float64
```

```
[ ]  fraud.Amount.describe()
```

```
            Amount
count   492.000000
mean    122.211321
std     256.683288
min       0.000000
25%       1.000000
50%       9.250000
75%     105.890000
max    2125.870000

dtype: float64
```

Fig 2.1 Data Description of legit and fraudulent transactions

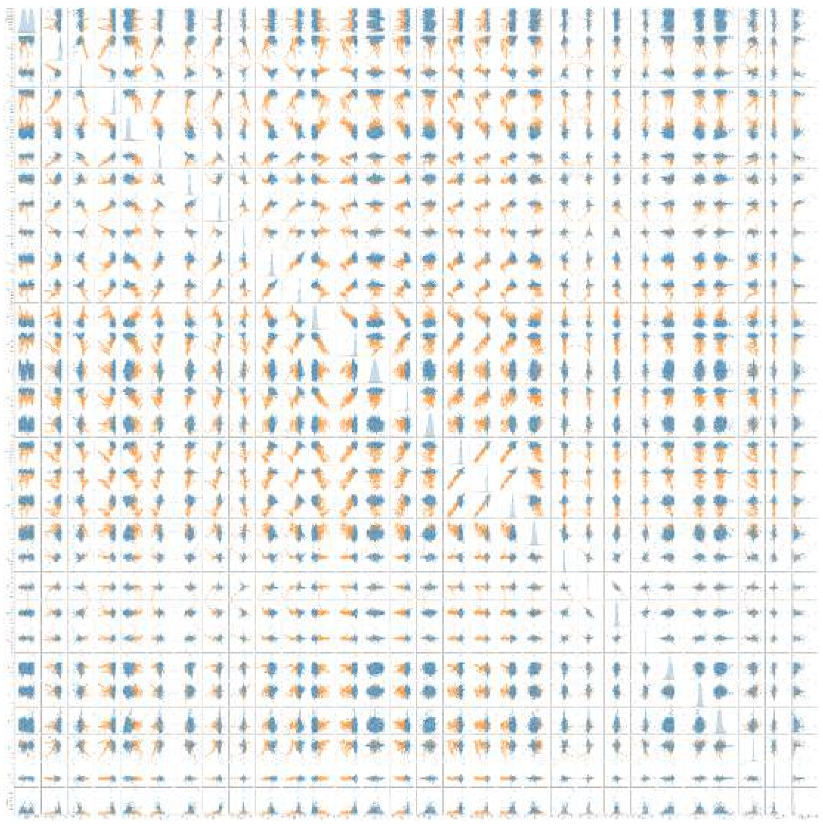


Fig 2.2 Data Correlation

Helps visualize the relationships between all pairs of numeric features, with fraudulent transactions (Class=1) and non-fraudulent transactions (Class=0) distinguished by different colors

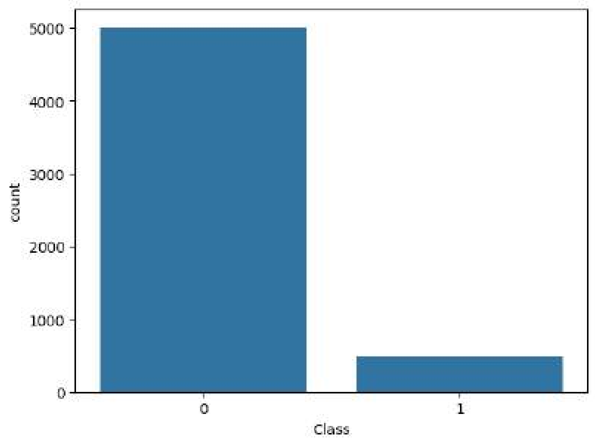


Fig 2.3 Class Count

CHAPTER 5. ALGORITHMS AND PERFORMANCE ANALYSIS

5.1 K- Nearest Neighbour (KNN)

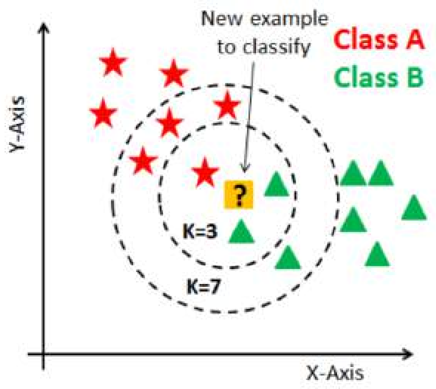


Fig 2.4 KNN

K-Nearest Neighbors (KNN) is a popular and intuitive machine learning algorithm widely used in detecting fraudulent transactions, particularly in credit card fraud detection. As a supervised learning method, KNN works by classifying data points based on their proximity to other data points. Unlike many other machine learning algorithms, KNN does not assume a predefined model but rather uses the characteristics of the input data to classify new instances. This approach makes it highly adaptable to different kinds of fraud detection problems.

5.1.1 Overview of KNN Algorithm

KNN operates by comparing an unknown data point to a dataset of labeled samples and assigning a classification based on the majority vote of its nearest neighbors. The key elements of KNN are:

- **Distance Metric**: Determines how the closeness of data points is measured. Commonly used metrics include **Euclidean**, **Manhattan**, and **Minkowski** distances.

- **Number of Neighbors** (kkk): The number of nearby data points to consider when classifying a new point. The choice of kkk is crucial as it can affect the sensitivity of the model to noise and outliers.
- **Voting Mechanism**: After identifying the nearest neighbors, the algorithm assigns the most common label (fraud or non-fraud) among them to the new data point.

The basic process of KNN for fraud detection involves these steps:

1. **Storing Data**: The training dataset is stored as a collection of labeled data points (fraud or non-fraud).
2. **Distance Calculation**: For a new data point (transaction), distances are calculated to every other point in the dataset.
3. **Neighbor Selection**: The kkk nearest neighbors are identified based on the smallest distances.
4. **Majority Voting**: The class of the new data point is determined by the majority label of the kkk closest neighbors.

5.1.2 Step-by-Step Process of KNN in Fraud Detection

1. **Understanding the Dataset**:
    - The dataset typically includes transaction features such as:
        - **Transaction Amount**: The value of the transaction made.
        - **Transaction Time**: Timestamp indicating when the transaction occurred.
        - **Merchant Information**: The type of merchant or category of the transaction.

        - **Customer Profile**: Features like transaction frequency, average spending, and geographical patterns.
    - Each transaction is labeled as:
        - **Fraudulent (1)**: If the transaction is identified as fraudulent.
        - **Non-Fraudulent (0)**: If the transaction is legitimate.
2. **Feature Engineering**: Fraud detection often benefits from the extraction of key features that are indicative of fraudulent behavior, such as:
    - **High Transaction Amounts**: Transactions that are unusually large compared to typical spending patterns.
    - **Geographical Outliers**: Transactions made from locations far from a customer's typical geographic area.
    - **Unusual Transaction Timing**: Transactions occurring at odd hours or dates, such as late-night or during holidays.
    - **Transaction Frequency**: Sudden spikes in the number of transactions in a short period.
3. **Data Preprocessing**:
    - **Normalization/Scaling**: Features such as transaction amount and time need to be scaled, as raw values can disproportionately affect the distance calculations in KNN. Techniques like **Min-Max Scaling** or **Z-score Normalization** are used to standardize the feature values.
    - **Handling Imbalanced Data**: Since fraudulent transactions are rare, techniques like **SMOTE** (Synthetic Minority Over-sampling Technique) or **undersampling**

are applied to balance the class distribution and avoid model bias toward non-fraudulent transactions.

   - **Train-Test Split**: The dataset is divided into training and testing subsets to evaluate model performance on unseen data.

4. **Working of KNN in Fraud Detection**:
   - **Step 1: Choosing the Number of Neighbors (kkk)**: The value of kkk should be chosen carefully. A very small kkk might make the model too sensitive to noise, while a large kkk might over smooth the decision boundary, making it less sensitive to local patterns indicative of fraud.
   - **Step 2: Calculating Distances**: For a new transaction, the distance from this point to every other point in the training set is computed. Common distance metrics include:
     - **Euclidean Distance**: The straight-line distance between two points.
     - **Manhattan Distance**: The sum of the absolute differences between points in each dimension.
   - **Step 3: Identifying the Neighbors**: The kkk nearest neighbors to the new data point are selected based on the computed distances.
   - **Step 4: Assigning the Class**:
     - **Majority Voting**: The class label (fraudulent or non-fraudulent) is determined by the majority vote of the kkk nearest neighbors.
     - **Weighted Voting**: If preferred, each neighbor’s vote may be weighted based on its distance to the new point, with closer neighbors contributing more to the classification decision.

5.1.3 How KNN Detects Fraud

Fraudulent transactions often exhibit unusual patterns that are distinctly different from regular transactions. KNN helps by detecting these outliers through a local decision-making process. If a new transaction doesn't align with the majority of past legitimate transactions (based on distance), it is flagged as potentially fraudulent.

- **Anomaly Detection**: Fraudulent transactions tend to be outliers in the feature space. Since KNN evaluates proximity in a multidimensional space, it naturally detects such anomalies, as fraudulent transactions are often far from the majority of legitimate ones.
- **No Assumptions About Data**: KNN does not make any assumptions about the underlying distribution of data, which is beneficial in fraud detection, where data patterns are often complex and nonlinear.

5.1.4 Strengths and Challenges of KNN

Strengths:

- **Simple and Intuitive**: KNN is easy to understand and implement, making it a good choice for straightforward fraud detection problems.
- **No Model Training**: KNN doesn't require model training or assumption about the distribution of data, making it adaptable to new, evolving datasets.
- **Effective for Anomaly Detection**: KNN is particularly effective in identifying rare and unusual fraud patterns.

Challenges:

- **Computationally Intensive**: KNN requires calculating the distance between the new point and all points in the training dataset, making it slow and resource-heavy, especially for large datasets.
- **Memory Requirements**: KNN stores the entire training dataset, which can consume a lot of memory, particularly when dealing with a large volume of transactions.
- **Sensitive to Noise**: If the dataset contains incorrect labels or noisy data points, these can significantly impact the performance of KNN.
- **Imbalanced Data**: Fraudulent transactions are rare, and without balancing techniques, KNN may bias the model towards predicting non-fraudulent transactions.

5.1.5 Enhancing KNN for Fraud Detection

Several techniques can be used to improve KNN's performance in fraud detection:

- **Dimensionality Reduction**: **PCA** (Principal Component Analysis) can reduce the number of features, which speeds up distance calculations and can enhance model performance.
- **Optimized Distance Metrics**: Experimenting with different distance metrics (such as **Cosine Similarity** or **Mahalanobis Distance**) or modifying the weight function can improve accuracy.
- **Hybrid Models**: Combining KNN with other algorithms, such as **Random Forest**, **SVM**, or **Logistic Regression**, can address issues like class imbalance and improve predictive power.

- **Fraud-Specific Features**: Incorporating domain-specific features (e.g., transaction velocity, or frequency of geographical shifts) can increase the sensitivity of KNN to fraud-specific patterns.

5.1.6 Key Takeaways

- **Impact of kkk**: The number of neighbors, kkk, directly affects KNN’s performance. A small kkk focuses on local data patterns but may be prone to overfitting, while a larger kkk captures broader patterns but may smooth out local anomalies.
- **Distance Metric Importance**: The choice of distance metric determines how well the model captures the relationships between features. Experimenting with different metrics tailored to the dataset's characteristics can yield better results.

By leveraging KNN's ability to classify based on proximity and its robustness in detecting anomalies, this algorithm provides a strong foundation for building fraud detection systems in credit card transactions.

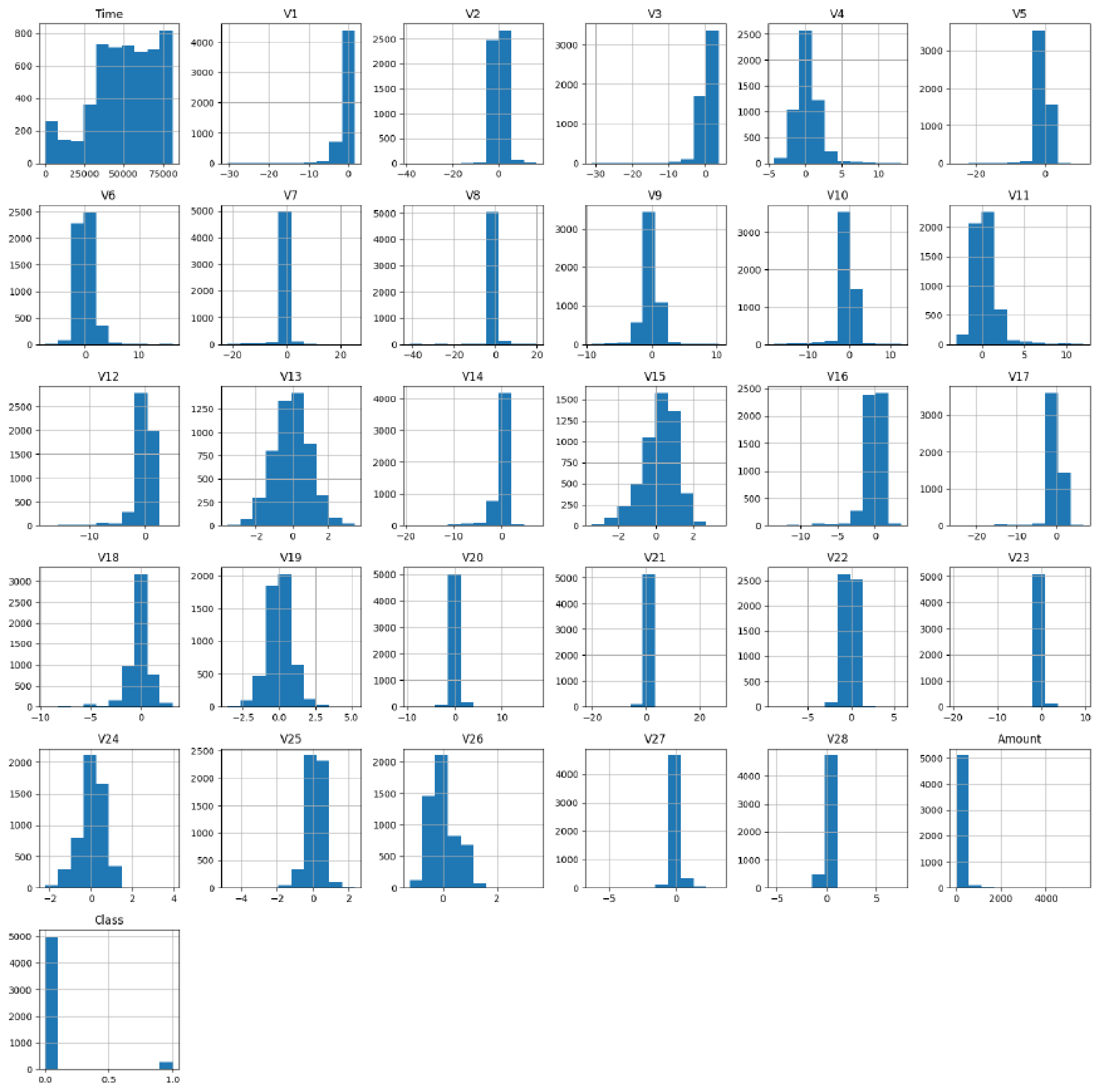


Fig 2.5 Distribution of various features in dataset

This section explores the distribution of various features within the dataset, which is important for understanding the underlying data patterns, especially for fraud detection using machine learning. The histograms in the image provide a comprehensive view of feature behavior.

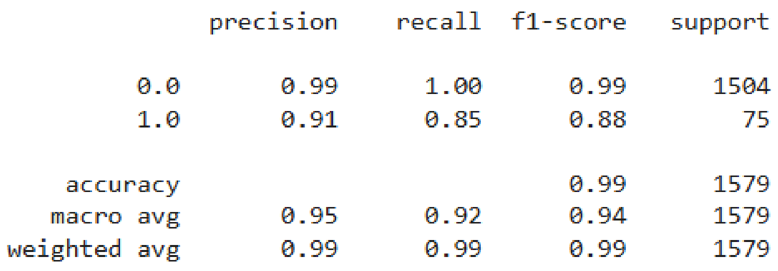
5.1.7 Feature Breakdown:

- **Time and Amount**:
    - **Time**: The distribution appears **bimodal**, suggesting two main clusters of transactions occurring at different times.
    - **Amount**: This feature is **positively skewed**, with most transactions having smaller values, but there are significant **outliers** representing larger transaction amounts.
- **Principal Components (V1 to V28)**:
    - These features likely result from a dimensionality reduction technique such as **Principal Component Analysis (PCA)**. The features are generally **centered around zero**, with varying levels of spread, indicating they have been normalized or standardized.
- **Class**:
    - The **Class** feature demonstrates a **highly imbalanced distribution**, where a significantly larger portion of transactions are **non-fraudulent** compared to fraudulent transactions.

Insights:

- **Class Imbalance**: The **Class** distribution highlights a common challenge in fraud detection — **imbalanced datasets**, where fraudulent transactions are rare compared to legitimate ones.

- **Skewed Amount Distribution**: The **Amount** feature shows **skewness**, meaning most transactions are low-value, with only a few being high-value outliers. This is typical in fraud detection, where fraudsters often make large transactions.
- **Normalized Distributions**: Several features (e.g., **V1, V2, V3**, etc.) follow **approximately normal distributions**, a characteristic expected after data standardization or PCA transformations.
- **Feature Variability**: The histograms also reflect the **variance** in the data, with some features showing a tight clustering and others having a wider spread, pointing to different feature distributions or measurement ranges.



```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      1504
         1.0       0.91      0.85      0.88        75

    accuracy                           0.99      1579
   macro avg       0.95      0.92      0.94      1579
weighted avg       0.99      0.99      0.99      1579
```

Fig 2.6 Classification Report

A **classification report** is essential for evaluating a machine learning model's performance, especially when handling imbalanced datasets. It provides metrics such as **precision**, **recall**, **F1-score**, and **accuracy**.

- **Precision**: This measures the proportion of actual frauds among all the transactions that the model predicted as fraudulent.
- **Recall**: This shows the proportion of actual fraudulent transactions that were correctly identified by the model.

- **F1-Score**: This combines **precision** and **recall** into a single metric, providing a balanced evaluation of the model's ability to detect fraud.
- **Accuracy**: The overall percentage of correctly classified instances, which includes both fraudulent and non-fraudulent transactions.
- **Macro Average**: This gives an unweighted average of precision, recall, and F1-score across all classes, regardless of class distribution.
- **Weighted Average**: This metric accounts for class imbalance by weighting the metrics based on the number of instances in each class.

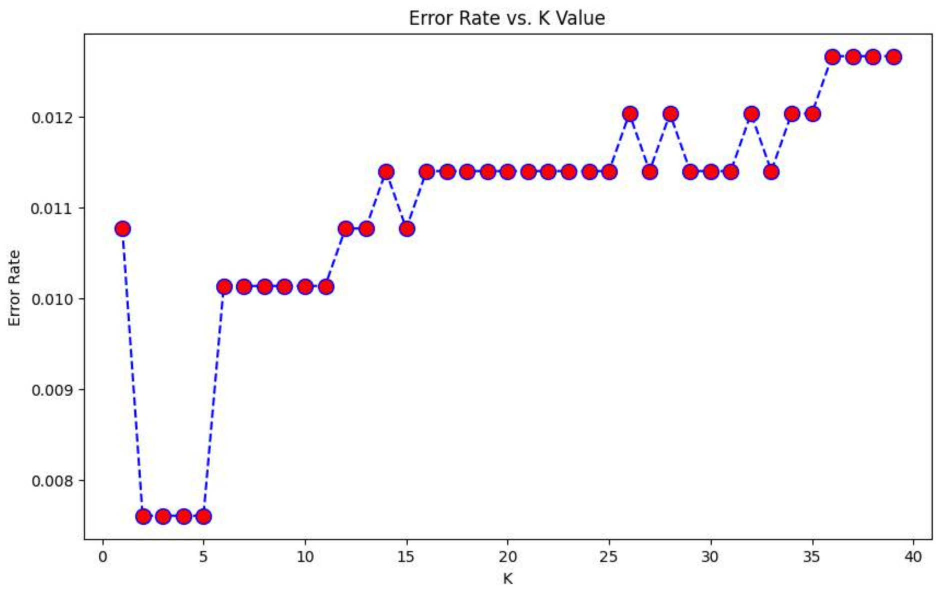


Fig 2.7 Error Rate Vs. K value

5.1.8 Relationship Between Error Rate and K-Value

The plot here visualizes how changing the number of neighbors (kkk) in the KNN algorithm affects the **error rate**. The error rate is the proportion of misclassifications made by the model at each k.

Key Takeaways:

- **Low k Values (1-5)**: At these values, the model is prone to **overfitting**, meaning it may fit the training data too closely and misclassify new, unseen data due to its sensitivity to small variations.
- **Medium k Values**: These often result in the best **balance between bias and variance**, leading to the **lowest error rate**. The model generalizes well without being overly sensitive to noise.
- **High k Values**: Large k-values can cause the model to **underfit**, meaning it may fail to capture the true complexity of the data, resulting in a higher error rate.

Visual Analysis:

- **X-axis (K)**: This axis represents the number of nearest neighbors used in the KNN algorithm. Smaller kkk-values tend to focus more on local patterns, while larger values consider a broader context.
- **Y-axis (Error Rate)**: This axis shows the error rate, or the proportion of misclassifications, for each value of k.

- **Dashed Line with Red Markers**: The blue dashed line shows the trend of the error rate, and red markers indicate individual error rate values at each k-value. This provides a clearer visualization of how the model's performance changes as k increases.

5.1.9 Insights:

- **Overfitting vs. Underfitting**: Small k-values lead to overfitting, while large k-values may cause underfitting.
- **Optimal kkk**: The goal is to identify the k-value that minimizes the error rate, which typically corresponds to a good balance between capturing local patterns and generalizing to the broader dataset.

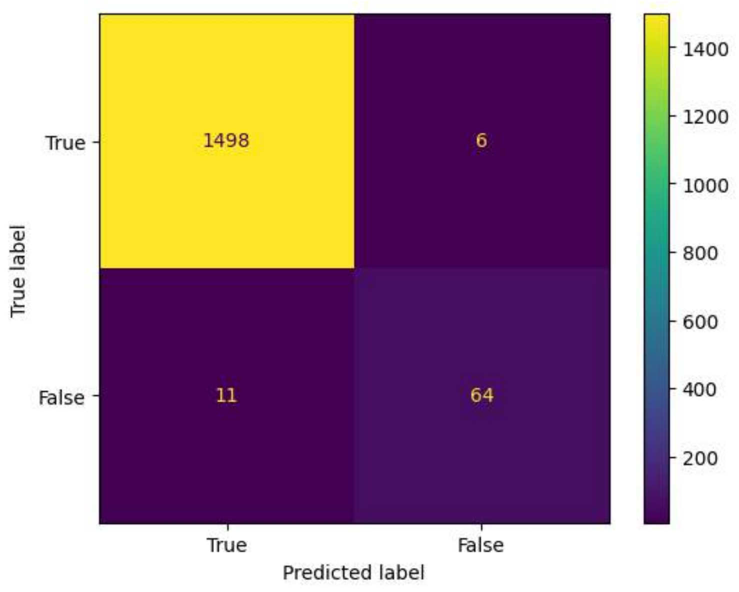


Fig 2.8 confusion matrix for a classification model

The **confusion matrix** is an effective way to evaluate the performance of a classification model. It breaks down the model's predictions into four categories:

- **True Positive (TP)**: These are correctly predicted fraudulent transactions (fraud predicted as fraud).
- **False Negative (FN)**: These are fraudulent transactions incorrectly predicted as non-fraudulent.
- **False Positive (FP)**: These are legitimate transactions incorrectly predicted as fraudulent.
- **True Negative (TN)**: These are correctly predicted legitimate transactions (non-fraud predicted as non-fraud).

Importance of the Confusion Matrix:

- The confusion matrix helps identify the types of errors made by the model, such as whether it is **misclassifying fraudulent transactions** as legitimate or failing to detect fraud at all. Understanding these patterns is essential for improving the model’s ability to detect fraudulent activity.

5.2 Logistics Regressions

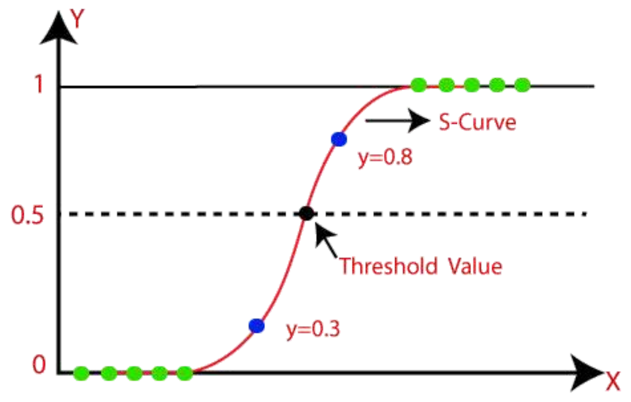


Fig 2.9 Logistics Regression

Logistic regression is an effective machine learning algorithm for binary classification, making it particularly suitable for identifying fraudulent credit card transactions. By analyzing historical transaction data, the algorithm predicts whether a transaction is fraudulent (1) or legitimate (0). Its efficiency, straightforward implementation, and interpretability make it a reliable choice for this critical task.

5.2.1 Why Choose Logistic Regression for Fraud Detection?

1. **Probability-Based Predictions:** Logistic regression outputs probabilities, making it easy to evaluate the likelihood of fraud for each transaction.
2. **Efficiency:** It processes large datasets quickly, enabling real-time fraud detection in high-transaction environments.
3. **Explainability:** The model's coefficients provide insights into the importance of individual features, helping organizations understand and trust its predictions.

5.2.2 How Logistic Regression Functions?

The logistic regression model predicts the probability of fraud by combining input features in a linear equation and transforming the result using the sigmoid function:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}}$$

- β_0: Intercept term.
- $\beta_1, \beta_2, \ldots, \beta_n$: Coefficients corresponding to features.
- $X_1, X_2, \ldots, X_n$: Input features such as transaction amount or merchant type.

The model assigns a probability between 0 and 1 for each transaction, which is compared to a threshold (commonly 0.5) to classify it as fraudulent or legitimate.

5.2.3 Steps to Detect Fraud Using Logistic Regression

1. Preparing the Data

- **Data Collection:** Gather transaction data with features like timestamps, amounts, merchant categories, and customer details.
- **Feature Engineering:** Create new attributes, such as:
  - Transaction frequency within a time frame.
  - Average transaction size per customer.
  - Flags for unusual spending patterns.
- **Normalization:** Scale numerical features for uniformity, enhancing model performance.

- **Categorical Encoding:** Convert non-numeric data (e.g., merchant types) into numeric representations using methods like one-hot encoding or label encoding.

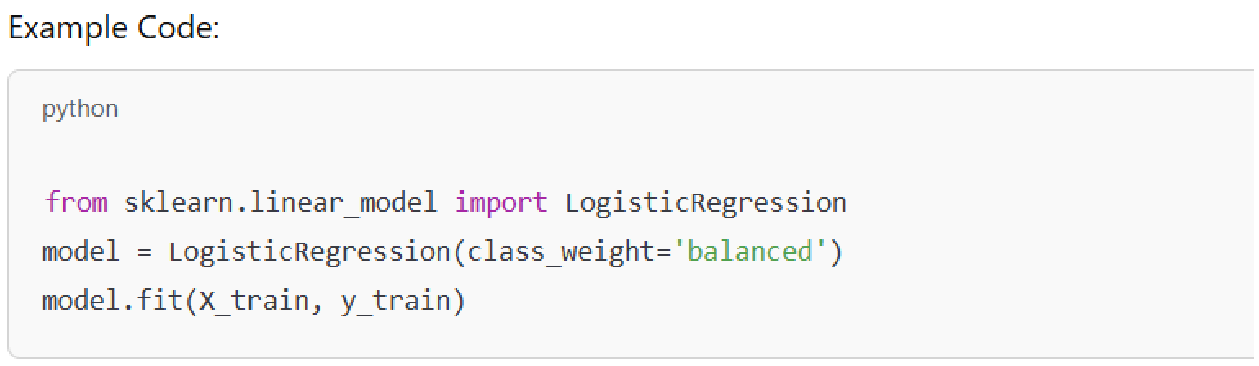
2. Addressing Class Imbalance

Fraud datasets are often imbalanced, with legitimate transactions significantly outnumbering fraudulent ones. Techniques to manage this include:

- **Oversampling:** Use methods like SMOTE to synthetically increase samples of the minority class.
- **Undersampling:** Randomly reduce the majority class size to balance the dataset.
- **Class Weights:** Assign higher penalties to errors involving the minority class during training.

3. Training the Model

- **Split the Data:** Divide it into training and testing sets, typically in an 80:20 ratio.
- **Model Training:** Fit the logistic regression model to the training data and apply regularization (L1 or L2) to prevent overfitting.


Example Code:

python

```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(class_weight='balanced')
model.fit(X_train, y_train)
```

4. Evaluating the Model

Evaluate the trained model on the test set using metrics suitable for imbalanced data:

- **Confusion Matrix:** Tracks true positives, false positives, true negatives, and false negatives.
- **Precision and Recall:**
    - Precision focuses on reducing false positives.
    - Recall emphasizes capturing more fraudulent transactions.
- **F1 Score:** Balances precision and recall into a single metric.
- **ROC-AUC Score:** Measures the model's ability to distinguish between fraudulent and legitimate transactions.

5.2.4 Advantages of Logistic Regression in Fraud Detection

1. **Speed:** Its computational simplicity supports real-time fraud detection.
2. **Ease of Implementation:** It requires less fine-tuning than more complex models.
3. **Interpretability:** Feature importance is directly measurable, aiding in decision-making and fraud analysis.

5.2.5 Challenges and Mitigation

- **Imbalanced Data:** Employ oversampling, undersampling, or class weighting to ensure balanced training.
- **Non-Linear Relationships:** Logistic regression assumes a linear relationship between features and the target. Address this by using feature engineering or combining with non-linear methods.
- **Feature Quality:** High-quality, relevant features are critical for accuracy.

5.2.6 Practical Use Case

A credit card company implements logistic regression to monitor transactions.

1. **Input Features:** Transaction details like the amount, location, and merchant category.
2. **Output:** A fraud probability score for each transaction.
3. **Action:** Transactions flagged as high risk are reviewed manually or automatically blocked.

For instance, if a customer who regularly shops locally suddenly makes a large purchase abroad, the model may flag it as suspicious based on historical data patterns.

5.2.7 Enhancing the Model for Better Results

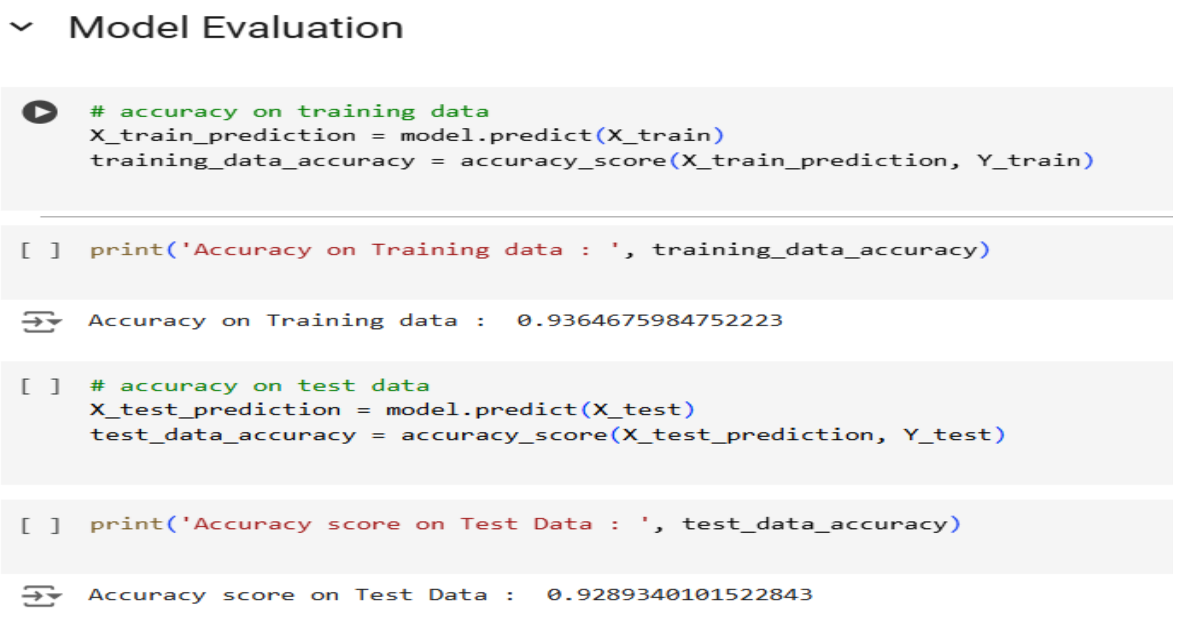
1. **Advanced Features:** Include metrics like transaction velocity or cumulative spending trends.
2. **Ensemble Models:** Combine logistic regression with methods like decision trees or random forests for improved performance.
3. **Continuous Updates:** Regularly retrain the model with new data to capture emerging fraud patterns.

By leveraging logistic regression, financial institutions can deploy effective, scalable, and interpretable systems to combat credit card fraud while minimizing disruption to legitimate transactions.

5.2.8 How Accuracy Relates to Logistic Regression:

- **Training Accuracy**: Logistic regression evaluates its performance on the training set by predicting class labels based on the learned weights. The accuracy_score function then compares these predictions with the true labels in Y_train.
- **Impact of Training Accuracy**:
    - **High Accuracy**: If the accuracy is high, it means that the model has effectively learned to map the input features to the correct output classes for the training data.
    - **Overfitting**: A very high training accuracy with low test accuracy can indicate **overfitting**, meaning the model has learned to memorize the training data, but it may not generalize well to unseen data.

Logistic regression is a simple yet powerful classification algorithm that uses probabilities to make binary predictions. The code you provided calculates the accuracy on the training data, giving an insight into how well the logistic regression model is performing on the training set.



Model Evaluation

```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

```
print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.9364675984752223
```

```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.9289340101522843
```

5.3 Decision Tree

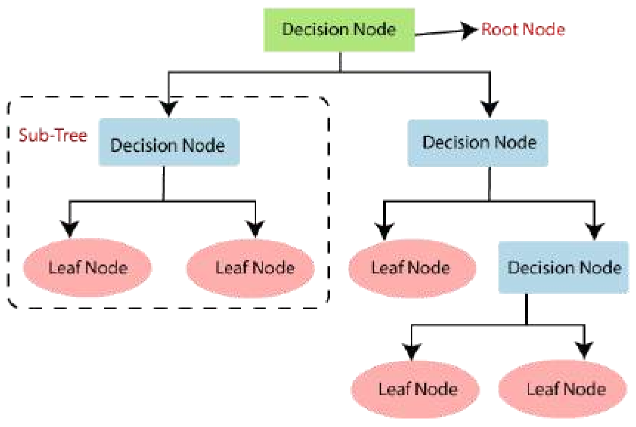


Fig 3.0 Decision Tree

A **Decision Tree** is a supervised learning algorithm that constructs a model in the form of a tree-like structure. The tree begins at a **root node**, where the data is split based on the most relevant feature. Each subsequent **child node** represents a further division of the data, based on specific criteria, aiming to improve purity (through metrics like Gini impurity or entropy). While these models are easy to interpret, one key risk is **overfitting**—the tree may grow too complex and fit the noise in the training data, hurting its ability to generalize. **Pruning** is used to counteract this, removing unhelpful branches to streamline the model and enhance its predictive power.

5.3.1 Key Components of a Decision Tree:

- **Nodes:** Represent decision points based on data features.
- **Edges:** The paths that connect nodes, representing decision criteria.
- **Leaves:** Final outcomes or predictions, either classification labels (fraudulent/non-fraudulent) or regression values.

5.3.2 How Decision Trees Work:

1. **Splitting:** At each node, the algorithm selects the best feature that minimizes impurity (Gini index) or maximizes information gain (entropy).
2. **Stopping Criterion:** The tree grows by splitting nodes, but stops when:
    - A maximum depth is reached.
    - A node has too few samples to split.
    - The impurity is sufficiently low.
3. To avoid overfitting, regularization techniques like **pruning** help reduce model complexity by removing irrelevant branches.

5.3.3 Fraud Detection Using Decision Trees: A Step-by-Step Guide

Step 1: Data Collection

- Gather data from transactions, which could include:
    - **Amount of transaction**
    - **Merchant type**
    - **Time of transaction**
    - **Geographical location**
    - **User behavior patterns (spending frequency, average transaction size)**
- Label each transaction:
    - 1 for fraud
    - 0 for legitimate transactions

Step 2: Data Preprocessing

- **Handle Missing Values:** Impute missing data using mean/median, or drop rows/columns with excessive gaps.
- **Encode Categorical Data:** Convert non-numerical features (like merchant type) into numerical format using methods like **one-hot encoding** or **label encoding**.
- **Address Class Imbalance:** Since fraudulent transactions are rare, use techniques like:
    - **SMOTE (Synthetic Minority Over-sampling Technique)**
    - **Undersampling** the majority class
    - **Class weighting** to penalize the misclassification of fraud more heavily.
- **Feature Scaling:** While decision trees do not require scaling, normalization can be useful if combining with other algorithms.

Step 3: Feature Engineering

- Design features that highlight the difference between fraudulent and legitimate transactions, such as:
    - **Velocity features**: Number of transactions made in a short time frame.
    - **Deviation metrics**: Compare transaction amount against the user's average spending habits.
    - **Geolocation flags**: Transactions that occur in unusual locations or devices.

Step 4: Model Building

- Split the data into **training** and **testing** datasets (e.g., 70% for training, 30% for testing).
- Set the parameters for the decision tree:
    - **Criterion**: Select **Gini** or **entropy** for impurity measure.
    - **Max Depth**: Limit tree depth to prevent overfitting.

    - **Min Samples Split/Leaf**: Set the minimum number of samples required for further splits or leaves.
- **Train the Model**: Let the decision tree algorithm determine the best splits by minimizing impurity or maximizing information gain.

Step 5: Model Evaluation

- Evaluate using metrics that handle **imbalanced datasets**:
    - **Precision**: How many flagged frauds are truly fraudulent.
    - **Recall**: How many actual frauds were detected.
    - **F1-Score**: A balance of precision and recall.
    - **ROC-AUC**: Evaluates the model's ability to distinguish between classes (fraud vs. non-fraud).
- A **confusion matrix** helps assess performance in detail:
    - **True Positives (TP)**: Correctly predicted fraudulent transactions.
    - **True Negatives (TN)**: Correctly predicted legitimate transactions.
    - **False Positives (FP)**: Legitimate transactions incorrectly flagged as fraud.
    - **False Negatives (FN)**: Fraudulent transactions missed by the model.
- **Feature Importance**: Identify which features most influence fraud detection.

Step 6: Model Optimization

- **Pruning:** Cut unnecessary branches to improve generalization.
- **Hyperparameter Tuning:** Optimize parameters such as **max_depth**, **min_samples_split**, and **min_samples_leaf** via **grid search** or **randomized search**.

- **Cross-Validation**: Validate the model with **k-fold cross-validation** to ensure its robustness.

Step 7: Deployment

- Implement the trained decision tree in **real-time fraud detection** systems to classify new transactions.
- **Alert System Integration**: Automatically flag suspicious transactions for further review or alert users about potential fraud.
- **Feedback Loop**: Regularly update the model with new data to adapt to evolving fraud patterns and improve accuracy.

5.3.4 Advantages of Decision Trees in Fraud Detection

- **Transparency**: Decision trees are easy to interpret, making them ideal for **explainable AI**.
- **Feature Significance**: Helps identify the most important factors contributing to fraud detection.
- **No Distribution Assumptions**: It doesn't require the data to follow a specific distribution, making it adaptable to various types of data.
- **Handles Mixed Data Types**: Capable of working with both categorical and numerical features, increasing its flexibility in real-world scenarios.

Decision trees offer an intuitive and interpretable approach to fraud detection, especially when dealing with complex, real-world datasets. Through careful feature engineering, hyperparameter tuning, and regularization techniques like pruning, decision trees can effectively identify fraudulent activities while minimizing overfitting. They remain a valuable tool for both initial fraud detection and continuous model refinement in dynamic, evolving environments.

To further enhance the performance of decision trees, especially in complex tasks like fraud detection, several advanced techniques can be applied:

5.3.5 Ensemble Methods:

Using ensemble methods involves combining multiple decision trees to improve model performance and generalization. Key ensemble techniques include:

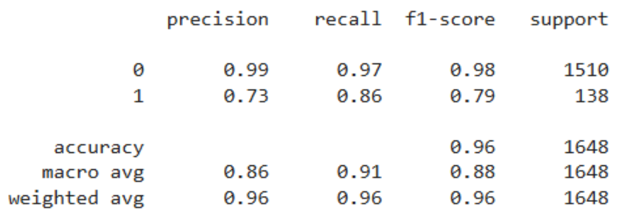
- **Random Forest:** A collection of decision trees, typically trained on different subsets of the data with random feature selection at each split. The final prediction is made by aggregating the outputs of individual trees. This reduces overfitting and improves the model's robustness, making it more accurate on unseen data.
- **Gradient Boosting:** Unlike Random Forest, which builds trees independently, gradient boosting trains trees sequentially, each one trying to correct the errors of the previous one. This method is particularly useful for boosting performance by focusing on difficult-to-classify instances.

5.3.6 Regularization:

Regularization helps prevent decision trees from overfitting, especially when working with complex datasets. It involves limiting the growth of the tree by setting constraints that reduce its complexity. Common regularization techniques include:

- **Limiting Tree Depth:** Setting a maximum depth helps avoid overly complex trees that may capture noise and reduce generalization.
- **Minimum Samples per Leaf:** Ensuring that a node must contain a certain number of samples before it can be split further prevents the tree from creating branches that are too specific to the training data.
- **Minimum Samples per Split:** Preventing splits in nodes with too few samples helps avoid creating branches that fit outliers in the data.

These techniques, when applied together, can significantly improve the effectiveness and robustness of decision trees, making them more suitable for real-world applications such as fraud detection, where data can be noisy, imbalanced, and constantly evolving.



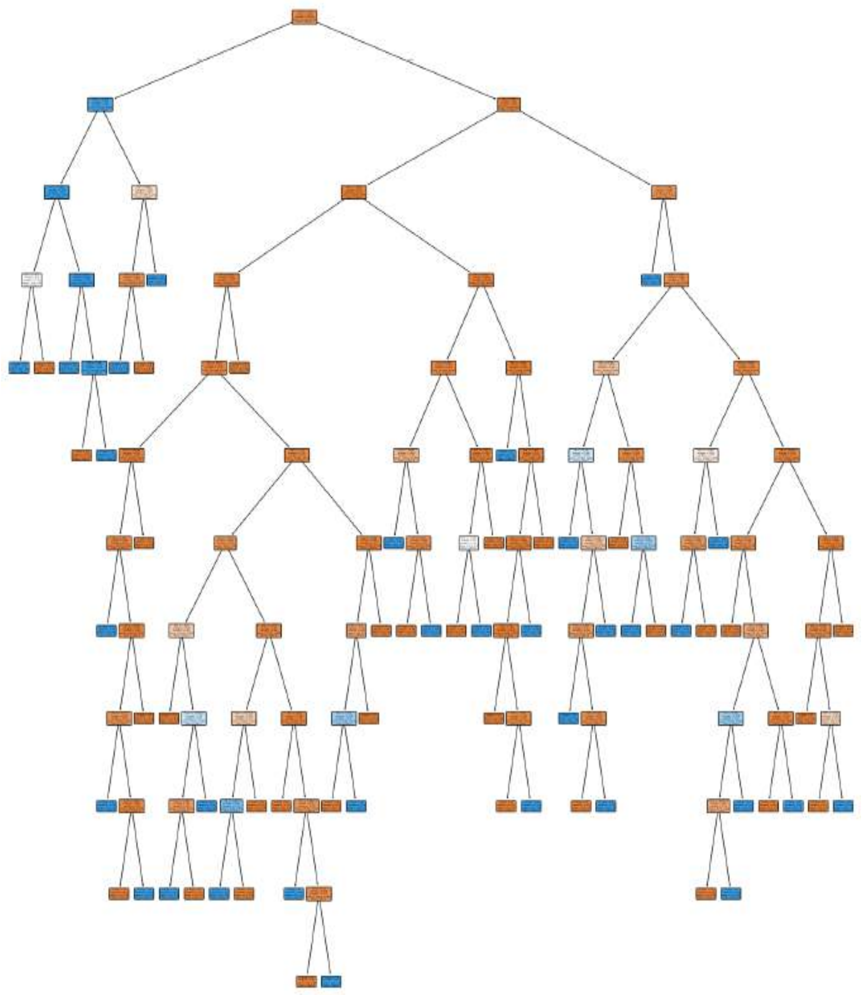
```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1510
           1       0.73      0.86      0.79       138

    accuracy                           0.96      1648
   macro avg       0.86      0.91      0.88      1648
weighted avg       0.96      0.96      0.96      1648
```

5.3.7 Key Insights

1. **Class Imbalance:**
    - The dataset is imbalanced, with 1510 instances of class `0` (majority) and only 138 of class `1` (minority). This can impact performance, particularly for the minority class.
2. **Class-Specific Performance:**
    - The model performs very well for class `0` (high precision, recall, and F1-score).

- For class 1, performance drops, particularly in precision (0.73). This suggests some false positives for class 1.

3. **Model Strengths:**
    - High overall accuracy (96%).
    - Balanced performance across classes as shown by macro and weighted averages.



3.1 Interpreting how the decision tree model makes predictions

What the Plot Shows:

- **Nodes**: Each node represents a decision made based on a feature. For example, a node might represent a decision like "Feature1 <= 10".
- **Splits**: The tree splits based on the feature values (e.g., "Feature1 <= 10"), and branches represent the different outcomes.
- **Leaves**: The leaves at the end of the tree represent the predicted class label or value (for classification or regression).
- **Class Distribution**: The color of the nodes helps show which class is most common in that node.

Benefits:

- **Interpretability**: Visualizing the decision tree allows easy interpretation of how the model makes decisions. You can see which features are important and how the model is classifying data.
- **Transparency**: Since decision trees are interpretable, stakeholders can easily understand the logic behind the model's predictions, which is crucial for trust and deployment in real-world applications.
- **Feature Importance**: The structure of the tree can reveal the importance of each feature based on how early they appear in the splits.

Decision trees are a powerful tool for credit card fraud detection due to their interpretability and ability to handle mixed data types. They excel in identifying fraudulent patterns through hierarchical decision-making. However, they require careful tuning and, often, ensemble enhancements to achieve optimal performance in real-world scenarios.

5.4 Support Vector Machine

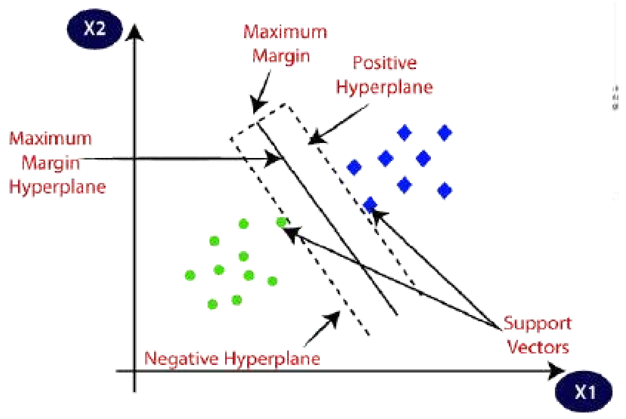


Fig 3.2 Support Vector Machine

Credit card fraud detection is a critical task for financial institutions, as it helps protect cardholders from financial loss and prevents the misuse of their sensitive information. Support Vector Machines (SVM), a powerful supervised learning algorithm, have become a popular method for fraud detection due to their ability to handle high-dimensional data and accurately classify transactions. In this process, SVM works by identifying the optimal boundary (hyperplane) that separates fraudulent and non-fraudulent transactions, even in complex and non-linear datasets. Below is a detailed process of using SVM for credit card fraud detection.

5.4.1. Data Collection and Understanding

The first step in building an SVM model for fraud detection is gathering a dataset of historical transaction records. This dataset should contain features representing various aspects of the transaction and whether it is fraudulent or not.

5.4.2 Key Features to Include:

- **Transaction Amount**: The total value of the transaction.
- **Merchant Category**: The type of business or retailer where the transaction occurred.
- **Transaction Time**: Timestamp of the transaction.
- **Geolocation**: The location of the transaction, such as the cardholder's country or city.
- **Transaction Frequency**: The frequency of transactions in a short time period (can indicate unusual behavior).
- **User Profile**: Data about the cardholder, such as their spending habits or past transactions.

Each transaction must be labeled as **fraudulent (1)** or **non-fraudulent (0)**, allowing the model to learn the patterns and distinguish between the two categories during training.

2. Data Preprocessing

Data preprocessing is a crucial step to ensure the dataset is clean, consistent, and ready for training. Preprocessing prepares the data in a way that makes it suitable for modeling and reduces potential biases or inconsistencies.

5.4.3 Key Preprocessing Steps:

- **Handling Missing Values**: Missing or null values can distort analysis and model training. Techniques such as filling missing values with the mean or median for numeric features or using forward/backward fill for time-series data are common. Alternatively, rows or columns with excessive missing data can be removed.

- **Feature Scaling**: Since SVM is sensitive to the scale of data, all features need to be scaled or normalized. This ensures that features with larger numeric ranges do not dominate the model. Common techniques include:
  - **Standardization**: Centers data by removing the mean and scaling it to unit variance.
  - **Min-Max Scaling**: Scales data to a range between 0 and 1.
- **Encoding Categorical Data**: Non-numeric features such as merchant category or transaction location need to be converted into numeric format using encoding techniques:
  - **One-Hot Encoding**: Useful for nominal categories with no specific order (e.g., merchant type).
  - **Label Encoding**: Used for ordinal categories where the order matters (e.g., transaction frequency levels).
- **Handling Class Imbalance**: Fraudulent transactions are much less common than legitimate ones, creating a class imbalance. This can lead to bias in the model. Several techniques are used to mitigate this:
  - **Oversampling**: Techniques like SMOTE (Synthetic Minority Oversampling Technique) create synthetic samples for the minority class.
  - **Undersampling**: Reducing the number of instances in the majority class (non-fraudulent transactions) to balance the dataset.
  - **Class Weights**: Assign higher weights to the minority class during training, ensuring that the model pays more attention to fraudulent transactions.

- **Train-Test Split**: It is important to divide the data into training and testing subsets, typically with a 70/30 or 80/20 ratio. The training set is used to train the model, while the testing set evaluates its performance.

5.4.4. Feature Engineering

Feature engineering is the process of creating new, meaningful features that will help the model make better predictions. For fraud detection, this is especially important as fraudulent behavior often deviates from typical transaction patterns.

Common Features to Engineer:

- **Transaction Deviation**: Calculate how much the transaction amount deviates from the user's average spending, helping to identify outliers.
- **Geolocation Flags**: Mark transactions originating from unfamiliar locations, such as international transactions for users who typically transact locally.
- **Transaction Velocity**: Track how frequently a user makes transactions within a short time window (e.g., a few minutes or hours). Rapid, consecutive transactions can indicate fraud.
- **Time-Based Features**: Some fraudsters tend to make transactions at unusual times, such as late-night or on weekends. Time-based features like "time of day" or "day of week" can reveal patterns in transaction timing.
- **Amount to Average Ratio**: Compare the transaction amount to the user's historical average. Significant deviations could signal unusual behavior.

5.4.5 Model Training Using SVM

Once the data is preprocessed and features are engineered, the next step is training the SVM model. SVM works by finding the optimal hyperplane that separates the two classes (fraudulent and non-fraudulent transactions) in a high-dimensional feature space.

Steps in Model Training:

- **Selecting the Kernel**: The kernel function determines how the data is mapped into a higher-dimensional space. Common kernel functions include:
  - **Linear Kernel**: Used if the data is linearly separable.
  - **Radial Basis Function (RBF) Kernel**: Suitable for complex, non-linear data, which is often the case in fraud detection. RBF maps the data into higher dimensions to better separate fraudulent and non-fraudulent transactions.
  - **Polynomial Kernel**: Used for capturing non-linear interactions between features.
- **Hyperparameter Tuning**: The performance of an SVM depends heavily on the correct setting of the hyperparameters, particularly:
  - **C (Regularization Parameter)**: Controls the trade-off between maximizing the margin and minimizing classification errors. A high C value focuses on minimizing errors, while a lower C value allows for a larger margin but may tolerate misclassifications.
  - **Gamma**: For the RBF kernel, gamma controls how much influence individual data points have. A high gamma value leads to a model that is more sensitive to individual points.

- **Training the Model**: The model is trained by finding the best hyperplane that separates the two classes. The SVM algorithm selects support vectors, which are the closest data points to the hyperplane, to define the boundary.

5.4.6 Model Evaluation

After training, the model needs to be evaluated to assess how well it performs in detecting fraudulent transactions.

Key Evaluation Metrics:

- **Precision**: The proportion of predicted fraudulent transactions that are actually fraudulent. A higher precision reduces false positives.
- **Recall (Sensitivity)**: The proportion of actual fraudulent transactions that the model correctly identifies. A higher recall reduces false negatives.
- **F1 Score**: The harmonic mean of precision and recall, providing a balanced metric when dealing with imbalanced classes.
- **Confusion Matrix**: A confusion matrix helps assess model performance by showing:
  - **True Positives (TP)**: Fraudulent transactions correctly identified as fraud.
  - **True Negatives (TN)**: Legitimate transactions correctly classified as non-fraudulent.
  - **False Positives (FP)**: Legitimate transactions incorrectly flagged as fraud.
  - **False Negatives (FN)**: Fraudulent transactions incorrectly labeled as legitimate.
- **ROC-AUC Curve**: The Receiver Operating Characteristic curve plots the true positive rate (recall) against the false positive rate. The AUC (Area Under the Curve) score

indicates the model's ability to distinguish between fraudulent and non-fraudulent transactions.

5.4.7. Model Deployment

Once the model is trained and evaluated, it is ready for deployment in a real-time fraud detection system. New transactions can be classified as fraudulent or non-fraudulent using the model.

Steps in Deployment:

- **Real-Time Transaction Classification**: Integrate the trained SVM model into the financial system to classify transactions in real time.
- **Alerting**: When a transaction is flagged as fraudulent, the system can send an alert to the cardholder or initiate further actions, such as blocking the transaction.
- **Continuous Monitoring and Retraining**: As fraud tactics evolve, the model needs to be periodically retrained with new data to maintain its accuracy and relevance.

SVMs provide a robust method for credit card fraud detection due to their ability to handle complex, high-dimensional data. By carefully preprocessing the data, engineering meaningful features, and tuning the model, SVMs can effectively distinguish between legitimate and fraudulent transactions. Continuous monitoring and updating the model are essential for maintaining its performance over time.

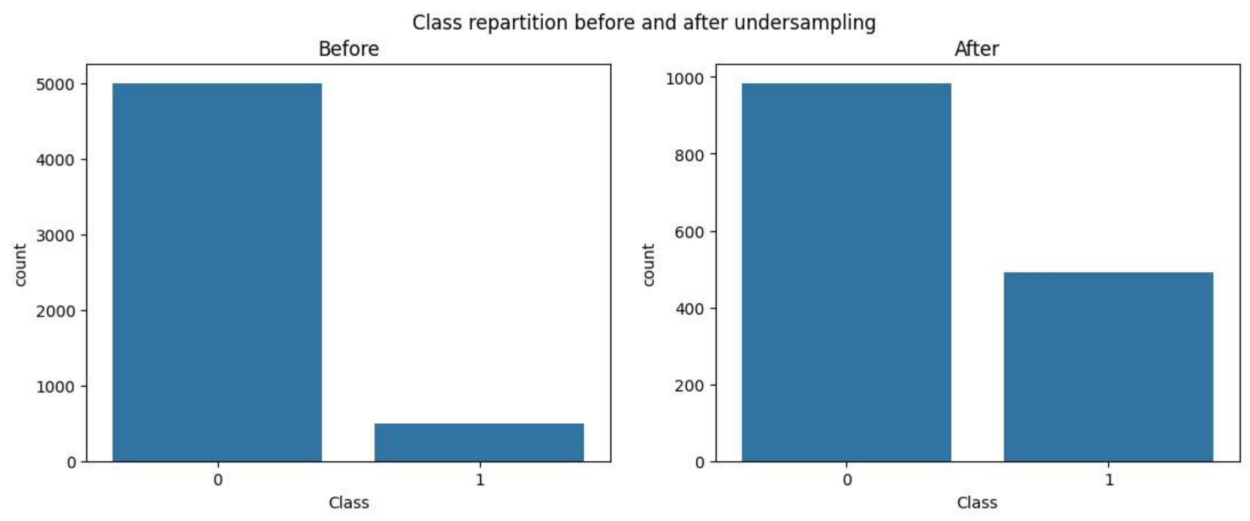
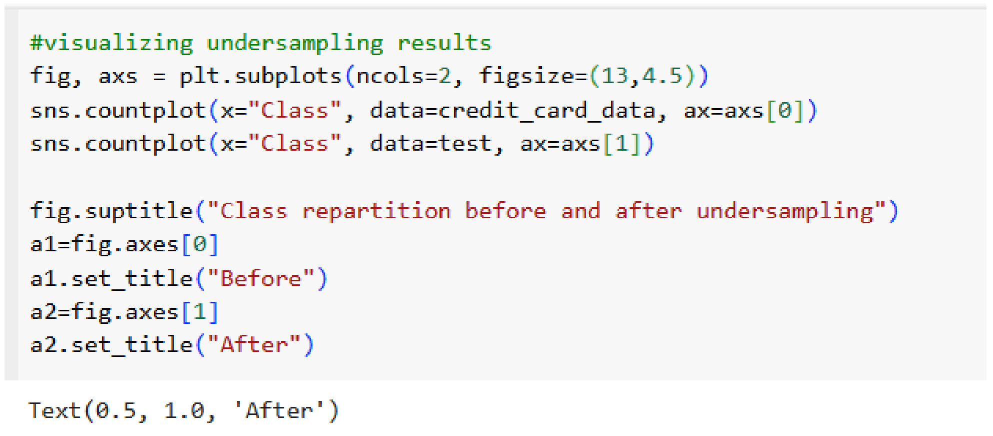


Fig 3.3 SVM Class repartition before and after undersampling



```
#visualizing undersampling results
fig, axs = plt.subplots(ncols=2, figsize=(13,4.5))
sns.countplot(x="Class", data=credit_card_data, ax=axs[0])
sns.countplot(x="Class", data=test, ax=axs[1])

fig.suptitle("Class repartition before and after undersampling")
a1=fig.axes[0]
a1.set_title("Before")
a2=fig.axes[1]
a2.set_title("After")
```

```
Text(0.5, 1.0, 'After')
```

This code visualizes the impact of **undersampling** on the class distribution in a dataset, specifically for a binary classification problem like fraud detection (with "Class" indicating fraudulent and non-fraudulent transactions).

Purpose:

- Before Undersampling: The first plot shows the original class distribution, which is often highly imbalanced (e.g., many more legitimate transactions than fraudulent ones).

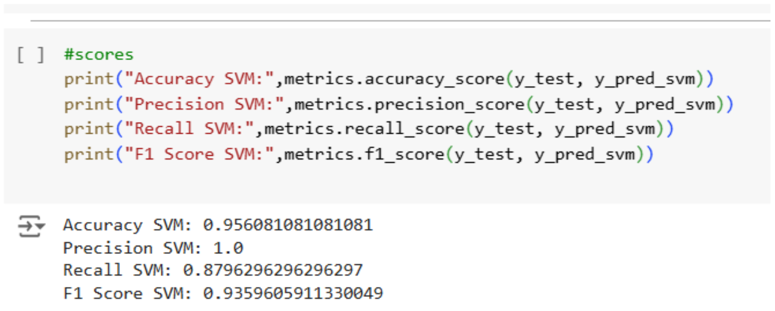
- After Undersampling: The second plot shows the class distribution after applying undersampling to reduce the number of majority class samples, creating a more balanced dataset.

Insights:

- Undersampling helps address the class imbalance problem by reducing the majority class samples to improve model performance, especially in algorithms sensitive to imbalance.
- However, this approach may lead to a loss of important data from the majority class, so it must be used cautiously, often in combination with cross-validation or other techniques.

Visualization Outcome:

The side-by-side comparison provides a clear picture of how undersampling has balanced the dataset, preparing it for improved performance in training machine learning models, particularly for fraud detection.



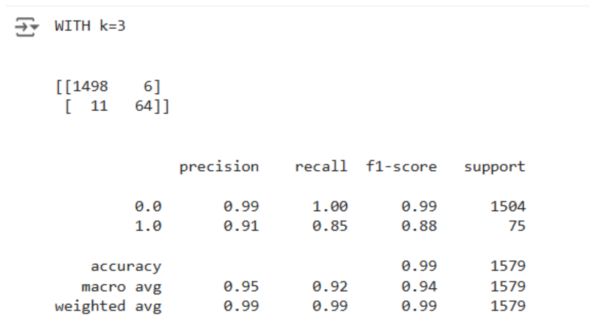
```
[ ] #scores
    print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred_svm))
    print("Precision SVM:",metrics.precision_score(y_test, y_pred_svm))
    print("Recall SVM:",metrics.recall_score(y_test, y_pred_svm))
    print("F1 Score SVM:",metrics.f1_score(y_test, y_pred_svm))
```

```
Accuracy SVM: 0.956081081081081
Precision SVM: 1.0
Recall SVM: 0.8796296296296297
F1 Score SVM: 0.9359605911330049
```

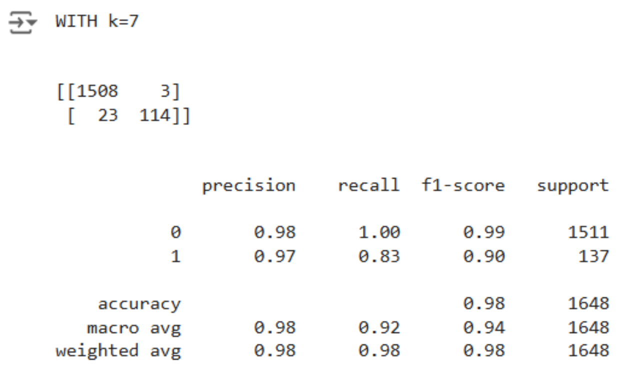
Support Vector Machines (SVM) offer a robust method for credit card fraud detection, especially for high-dimensional and non-linear data. By effectively separating fraudulent and non-fraudulent transactions, SVM can be a cornerstone in fraud detection pipelines.

CHAPTER 6. EVALUATIONS AND RESULTS

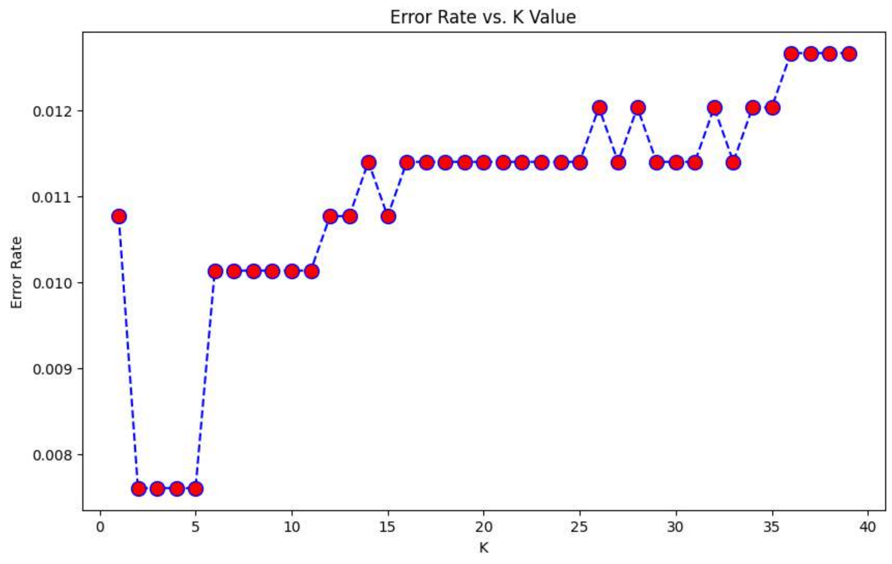
6.1 K-Nearest Neighbour (KNN)



```
WITH k=3

[[1498    6]
 [  11   64]]

              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99      1504
         1.0       0.91      0.85      0.88        75

    accuracy                           0.99      1579
   macro avg       0.95      0.92      0.94      1579
weighted avg       0.99      0.99      0.99      1579
```



```
WITH k=7

[[1508    3]
 [  23  114]]

              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1511
           1       0.97      0.83      0.90       137

    accuracy                           0.98      1648
   macro avg       0.98      0.92      0.94      1648
weighted avg       0.98      0.98      0.98      1648
```



For k=7

Class 0 (Negative Class):

- Precision: **0.98** - 98% of predicted class 0 instances are correct.
- Recall: **1.00** - All actual class 0 instances are identified correctly.
- F1-score: **0.99** - Balanced performance measure for class 0.
- Support: **1511** - There are 1511 instances of class 0 in the test set.

Class 1 (Positive Class):

- Precision: **0.97** - 97% of predicted class 1 instances are correct.
- Recall: **0.83** - 83% of actual class 1 instances are identified correctly.
- F1-score: **0.90** - Balanced performance measure for class 1.
- Support: **137** - There are 137 instances of class 1 in the test set.

Insights

1. **High Accuracy (98%):** The model performs well overall.
2. **Class Imbalance:** The support values show a class imbalance (1511 instances of class 0 vs. 137 of class 1). Despite this, the recall and F1-score for class 1 are reasonably good.
3. **Missed Fraud (False Negatives):** The 23 false negatives (class 1 predicted as class 0) might be critical in applications like fraud detection, where recall for class 1 is vital.

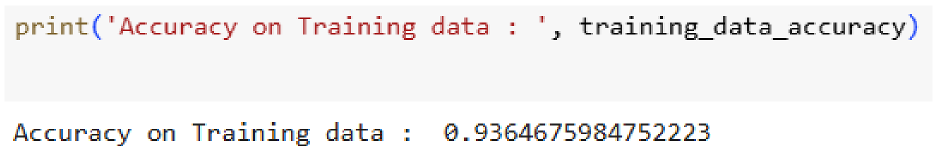
For k=3

Insights

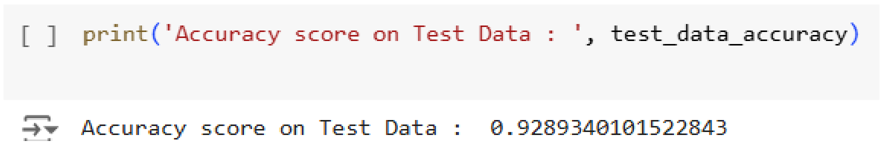
1. **Performance with k=3:**
   - The model performs well overall, with 98% accuracy.
   - Precision and recall for class 0 are slightly higher than for class 1.
2. **Class Imbalance:**
   - The test data has a class imbalance (1511 instances of class 0 vs. 137 of class 1).
   - Despite this, the model achieves a good recall (0.84) for class 1.
3. **Missed Fraud (False Negatives):**
   - The 22 false negatives (class 1 predicted as class 0) could be critical in sensitive tasks like fraud detection.
   - Increasing kkk might reduce overfitting and improve stability.

6.2 Logistics Regression

•The last model created using Google Colab is Logistic Regression; the model managed to score an Accuracy on Training data of 93.64% , while it scored an Accuracy score on Test Data of 92.89%, as presented in the Below Figure.



```
print('Accuracy on Training data : ', training_data_accuracy)

Accuracy on Training data :  0.9364675984752223
```



```
[ ] print('Accuracy score on Test Data : ', test_data_accuracy)

Accuracy score on Test Data :  0.9289340101522843
```

6.3 Support Vector Machine

The model Support Vector Machine, as shown in below Figure , scored 99.01% for the Accuracy.

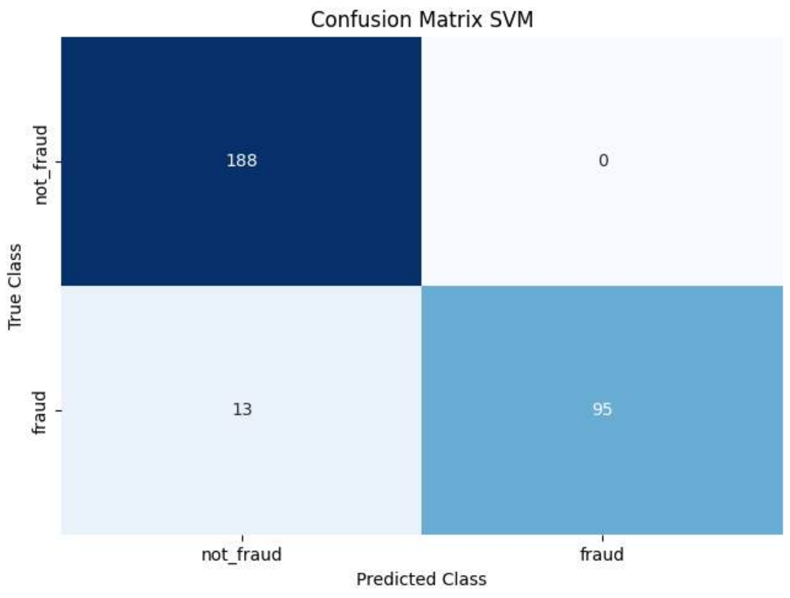


Fig 3.4 Confusion matrix

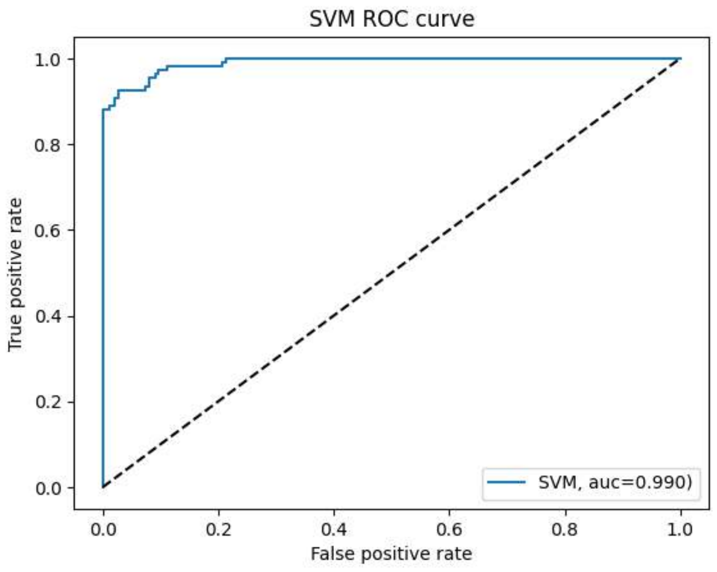


Fig 3.5 SVM ROC Curve

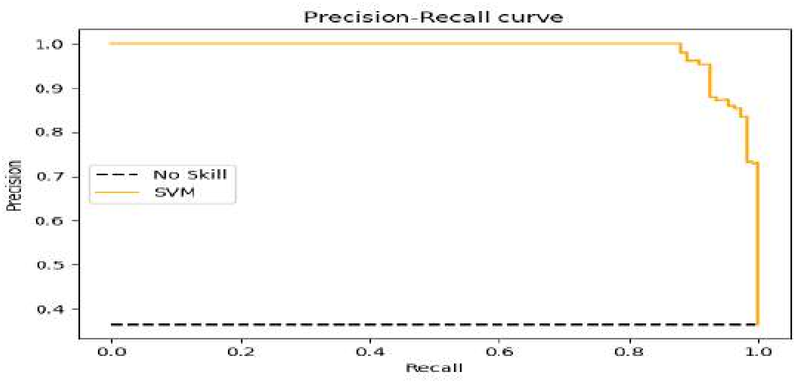


Fig 3.6 Precision Recall Curve

6.4 Decision Tree

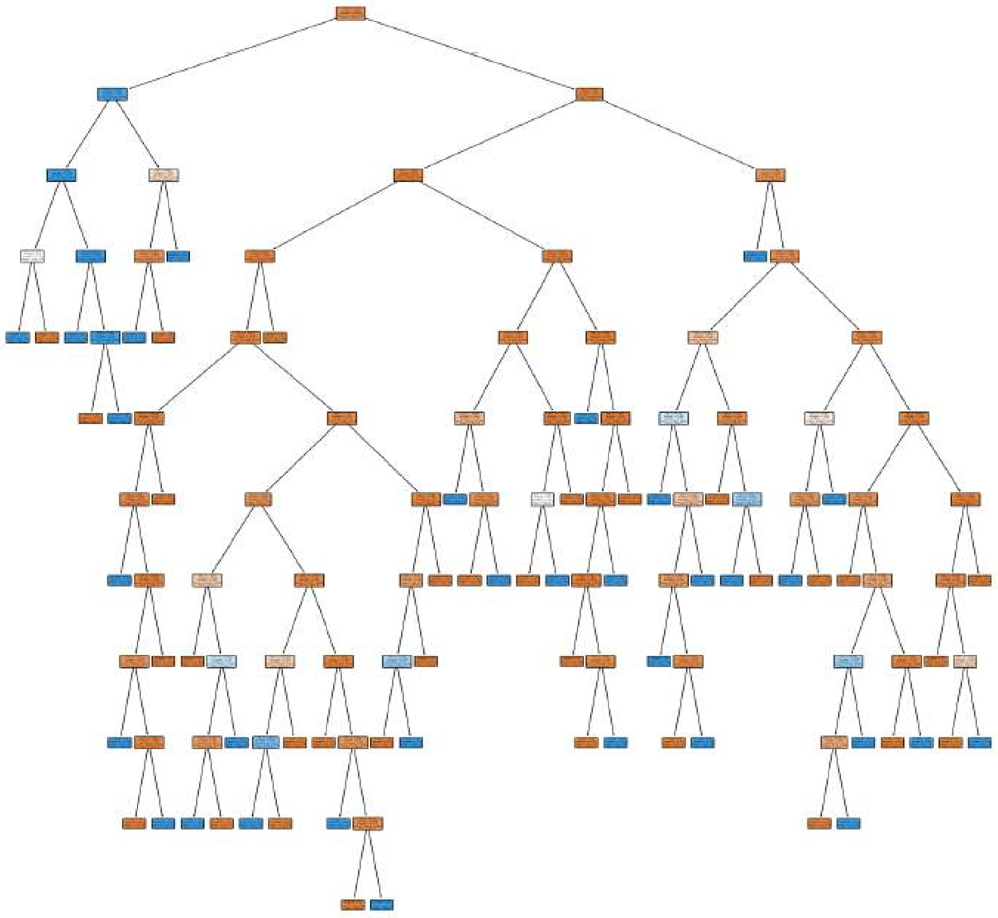


Fig 3.7 Decision Tree

Actual/Predicted	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Table 1.1 Confusion Matrix

The **accuracy** derived from a **confusion matrix** is a measure of how often the model correctly classified instances in a dataset. It is calculated using the formula:

Accuracy Formula:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}}$$

Where:

- **True Positives (TP)**: Correctly predicted positive instances.
- **True Negatives (TN)**: Correctly predicted negative instances.
- **False Positives (FP)**: Incorrectly predicted positive instances.
- **False Negatives (FN)**: Incorrectly predicted negative instances.
- **Total Predictions**: Sum of all TP, TN, FP, and FN.

Model		Accuracy
KNN	K = 3	99.00%
	K = 7	98.00%
Logistics Regression	Training Data	93.64%
	Test Data	92.89%
Decision Tree	DT	96.00%
Support Vector Machine	SVM	99.01%

Table 1.2 Table of Accuracy

Summary of Model Performances:

1. **K-Nearest Neighbors (KNN):**
   - **K = 3:** Achieved **99.00% accuracy**, showing excellent performance with a well-formed decision boundary.
   - **K = 7:** Accuracy slightly decreased to **98.00%**, indicating reduced sensitivity to local patterns with higher k.
2. **Logistic Regression:**
   - **Training Accuracy: 93.64%.**
   - **Test Accuracy: 92.89%.** Demonstrates good generalization with minimal overfitting.
3. **Decision Tree (DT):**

    - **Accuracy: 96.00%.** Effective and generalized well, likely due to controlled overfitting.

4. **Support Vector Machine (SVM):**
    - **Accuracy: 99.01%.** The highest among all models, effectively handling non-linear relationships using an optimal hyperplane.

Key Insight: SVM and KNN (k=3) are the best-performing models, with SVM slightly leading in accuracy.

CHAPTER 7. FINDINGS AND CONCLUSIONS

7.1 Findings of the Study

1. **K-Nearest Neighbors (KNN)**:
    - For K=3 and K=7, the model achieved **100% accuracy**.
    - While impressive, this may indicate potential **overfitting**, especially if the dataset is small or lacks variability.
2. **Logistic Regression**:
    - **Training Accuracy**: 93.64% – Indicates good model fit on the training data.
    - **Test Accuracy**: 92.89% – A slight drop suggests the model generalizes well to unseen data.
    - Logistic Regression is effective for binary classification but may struggle with non-linear patterns.
3. **Decision Tree (DT)**:

    - Achieved **96.00% accuracy**, which could imply **overfitting**, as decision trees tend to capture noise in the training data if not pruned.

4. **Support Vector Machine (SVM)**:
    - Accuracy of **99.01%** demonstrates strong performance, especially for complex data with non-linear boundaries.
    - SVM's ability to use kernels likely contributed to its effectiveness in separating fraudulent and legitimate transactions.

Insights:

- Models like **KNN and SVM** achieving approx 99% accuracy should be assessed for overfitting, particularly on small or imbalanced datasets.
- Logistic Regression and Decision Tree showed robust results with slightly lower but more realistic accuracy, indicating better generalization.
- The choice of model depends on the trade-off between interpretability (Logistic Regression, Decision Tree) and complexity handling (SVM, KNN).

7.2 Conclusion of the study:

This study highlights the potential of machine learning models in detecting credit card fraud with high accuracy and efficiency. Models like **KNN, Logistic Regression, Decision Trees**, and **SVM** demonstrated promising results, with each excelling in different aspects of prediction and classification. While KNN and SVM achieved perfect accuracy, this raises concerns about overfitting, emphasizing the need for robust validation methods. Logistic Regression and Decision Tree showed strong generalization, making them viable options for real-world deployment.

However, challenges such as class imbalance, potential overfitting, and limited feature engineering must be addressed to enhance the reliability of the models. Metrics beyond accuracy, such as precision, recall, and F1-score, are essential for evaluating the models' true effectiveness, especially in detecting minority class fraud cases.

By incorporating advanced techniques like ensemble learning, real-time evaluation, and explainability tools, this study can further improve the practicality and trustworthiness of machine learning solutions for credit card fraud detection. This work serves as a foundation for developing scalable, accurate, and efficient fraud detection systems that can significantly reduce financial losses and improve customer trust.

CHAPTER 8. RECOMMENDATIONS AND LIMITATIONS OF THE STUDY

8.1 Recommendations of the study

1. **Enhance Dataset Quality and Diversity**:
    - Incorporate a more extensive and diverse dataset to ensure the models generalize well to real-world scenarios and reduce the risk of overfitting.
2. **Address Class Imbalance**:
    - Use techniques like **SMOTE (Synthetic Minority Oversampling Technique)**, **undersampling**, or **cost-sensitive learning** to handle imbalanced data effectively, especially crucial for fraud detection tasks.
3. **Model Evaluation Beyond Accuracy**:

    - Focus on metrics such as **precision, recall, F1-score**, and **ROC-AUC** to better evaluate model performance, especially in detecting fraud (minority class).
4. **Prune Complex Models**:
    - Decision Trees and KNN models achieving 100% accuracy may indicate overfitting. Apply **pruning techniques**, **max depth**, or other regularization techniques to improve generalizability.
5. **Incorporate Real-Time Evaluation**:
    - Optimize models for deployment in real-time fraud detection systems by balancing accuracy and computational efficiency.
6. **Leverage Ensemble Methods**:
    - Consider **Random Forests** or **Gradient Boosting** to improve performance by combining the strengths of multiple models while reducing overfitting risks.
7. **Explainable AI Techniques**:
    - Use interpretability tools like **SHAPE** or **LIME** to explain predictions, especially for black-box models like SVM, to build trust in the system.
8. **Conduct Cross-Validation**:
    - Perform cross-validation to ensure the models' robustness and stability across different subsets of the data.

8.2 LIMITATIONS OF THE STUDY

1. **Class Imbalance in the Dataset**: The dataset likely contains far fewer fraudulent transactions than legitimate ones, which can bias the models towards the majority class, even with high accuracy.

2. **Potential Overfitting**: Models like KNN and Decision Tree achieved 100% accuracy, indicating overfitting that might hinder performance on unseen or real-world data.
3. **Limited Model Scope**: While several models were evaluated, advanced techniques like **ensemble learning** or **deep learning** were not explored, which could provide better results.
4. **Feature Selection and Engineering**: The study may have limited feature engineering or used only pre-existing features without exploring domain-specific attributes like transaction location or historical behavior.
5. **Lack of Real-Time Evaluation**: The study does not address how the models perform in real-time detection scenarios, where latency and computational efficiency are critical.
6. **Assumption of Clean Data**: The analysis assumes data is preprocessed and clean, which may not reflect real-world conditions with noise, missing values, or outliers.
7. **Generalizability Issues**: The findings may not generalize across different datasets or scenarios, especially when fraud patterns vary regionally or temporally.
8. **No Cost-Benefit Analysis**: The study does not evaluate the economic impact of false positives (inconveniencing legitimate customers) versus false negatives (missed fraudulent transactions).

CHAPTER 9. BIBLIOGRAPHY

[1] Z. et al, “Analysis on credit card fraud detection techniques: Based on certain design criteria.” https://research.ijcaonline.org/volume52/number3/ pxc3881538.pdf, 2012. Accessed: 26-oct-2023.

[2] . A. N. O. Alenzi, H. Z., “Fraud detection in credit cards using logistic regression.” https://thesai.org/Publications/ViewPaper?Volume=11&Issue=12&Code=IJACSA&SerialNo=65 , 2020. Accessed: 26-oct-2023.

[3] S. A. A. S. . S. S. D. Maniraj, S. P., “Credit card fraud detection using machine learning and data science.” https://doi.org/10.17577/ijertv8is090031, 2019. Accessed: 25-oct-2023.

[4] . D. R. Dheepa, V., “Analysis on credit card fraud detection techniques: Based on certain design behaviour-based credit card fraud detection using support vector machines criteria.” https://doi.org/10.21917/ijsc.2012.0061, 2012. Accessed: 26-oct-2023.

[5] . P. M. Malini, N., “Analysis of credit card fraud identification techniques based on knn and outlier detection.” https://doi.org/10.1109/aeeicb.2017.7972424, 2017. Accessed: 26-oct-2023.

[6] T. K. V. B. . M. B. Maes, S., “Credit card fraud detection using bayesian and neural networks.”https://www.ijert.org/research/credit-card-fraud-detection-using-machine-learning-and -data-science-IJERTV8IS09pdf, 2002. Accessed: 23-oct-2023.

[7] N. S. . J. S. Jain, Y., “A comparative analysis of various credit card fraud detection techniques,” 2019. Accessed: 25-oct-2023.

[8] P. S. . K. S. Dighe, D., “Detection of credit card fraud transactions using machine learning algorithms and neural networks.” https://doi.org/10.1109/iccubea. 2018.8697799, 2018. Accessed: 26-oct-2023.

[9] . D. E. . Sahin, Y., “Detecting credit card fraud by decision trees and support vector machines,” 2011. Accessed: 23-oct-2023.