VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi-590018, Karnataka

A MINI PROJECT

REPORT ON

# "CREDIT CARD FRAUD DETECTION SYSTEM"

Submitted in partial fulfillment of requirements for the award of degree

**BACHELORS OF ENGINEERING IN INFORMATION SCIENCE AND ENGINEERING**

Submitted by

| | |
|---|---|
| **KUMUDA SHREE G** | **(1SB22IS021)** |
| **LAYA DR** | **(1SB22IS022)** |
| **MONIKA GS** | **(1SB22IS031)** |
| **SAHANA N** | **(1SB22IS043)** |

Under the Guidance of

**Dr. Raghavendra Rao B**

Head & Assistant Professor, Department of Information Science & Engineering

**DEPARTMENT OF INFORMATION SCIENCE&ENGINEERING**

**SRI SAIRAM COLLEGE OF ENGINEERING**

**ANEKAL, BENGALURU - 562106**

**ACADEMIC YEAR: 2024-25**

**SRI SAIRAM COLLEGE OF ENGINEERING**

**Anekal, Bengaluru – 562106**

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

## CERTIFICATE

This is to certify that the mini project work, entitled **"CREDIT CARD FRAUD DETECTION SYSTEM"** is a bona-fide work carried out by

|  |  |  |
|---|---|---|
| **1.** | **KUMUDA SHREE G** | **1SB22IS021** |
| **2.** | **LAYA DR** | **1SB22IS022** |
| **3.** | **MONIKA GS** | **1SB22IS031** |
| **4.** | **SAHANA N** | **1SB22IS043** |

in partial fulfillment for the award of degree of Bachelor of Engineering in Information Science & Engineering of the Visvesvaraya Technological University, Belagavi during the academic year 2024-25. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. This project report has been approved as it satisfies the academic requirements with respect to the project work prescribed for the Bachelor of Engineering Degree.

| Signature of the Internal Guide | Signature of the HOD | Signature of the Principal |
|---|---|---|
| **Dr. Raghavendra Rao B** | **Dr. Raghavendra Rao B** | **Dr. B.Shadaksharappa** |
| Head & Assistant Professor, | HOD | Principal |
| Dept. of ISE | Dept. of ISE | |

Name of examiners: Signature with date

1.

2.

## ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance. So, with gratitude we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

We take this opportunity to thank Our Chairman **Dr. Sai Prakash LeoMuthu**, and **Dr. Arun Kumar R**, Chief Executive Officer of Sri Sairam College of Engineering for providing us with excellent infrastructure that is required for the development of our project.

We are thankful to our Principal, **Dr. B. Shadaksharappa** and for their encouragement and support throughout the project work.

We are also thankful to our beloved HOD, **Dr Raghavendra Rao** for his incessant encouragement & all the help during the project work.

We take this opportunity to thank our Project Coordinators, **Dr. Raghavendra Rao,** Head & Assistant Professor, Dept. of ISE and**,** Head & Assistant Professor, Dept. of ISE for their inspirational guidance, valuable suggestions and providing us a chance for the completion of the Project.

We consider it a privilege and honor to express our sincere gratitude to our guide**,** Head & Assistant Professor, Dept. of ISE for his valuable guidance throughout the tenure of this project work, and whose support and encouragement made this work possible.

It is also a great pleasure to express our deepest gratitude to all the other faculty members of our department for their cooperation and constructive criticism offered, which helped us a lot during our project work.

Finally, we would like to thank all our family members and friends whose encouragement and support was invaluable.

**SRI SAIRAM COLLEGE OF ENGINEERING**

**Anekal, Bengaluru – 562106**

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

## DECLARATION

We, **KUMUDA SHREE G, LAYA DR, MONIKA GS, SAHANA N** hereby declare that the entire work titled **“CREDIT CARD FRAUD DETECTION SYSTEM**” embodied in this project report has been carried out by us during the 5th semester of BE degree at SSCE, Bangalore under the esteemed guidance of, Head & Assistant Professor, Dept. of ISE, Sri Sairam College of Engineering, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi. The work embodied in this dissertation work is original and it has not been submitted in part or full for any other degree in any University.

| | |
|---|---|
| **1. KUMUDA SHREE G (1SB22IS021)** | ______________________ |
| **2. LAYA DR (1SB22IS022)** | ______________________ |
| **3. MONIKA GS (1SB22IS031)** | ______________________ |
| **4. SAHANA N (1SB22IS043)** | ______________________ |

## ABSTRACT

Credit card fraud is without doubt has the potential to be a menace as far as institutions and consumers of the credit cards are concerned. Another approach employing machine learning comes up handy in that it is able to analyze past transactions data to look for suspicious issues. The goal of this project is to design a credit card fraud detection system using the concept of machine learning. In our case, the model will be built from a set of past credit card transactions and then tested on a set of new transactions unknown to the model. The objective is to identify which model is the best fit to detect fraud in transactional level and thus avoids more losses.

# TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# SYSTEM ANALYSIS

# CHAPTER 1

# SYSTEM ANALYSIS

## Introduction

Credit card in generally defined as a card issued to the customer to enable them to purchase goods and services on the basis of credit limit. Credit card gives equal time to their customers for repaying the amount for a prescribed period. Credit card frauds are one of the easiest methods to go after. Without any risks, it is possible to withdraw a large sum without the owner’s consent within a short time. Criminals always endeavor to make all fraudulent transactions legal. Making fraud detection very complex and even getting to be almost extremely difficult to identify.

This means that credit card companies want to know quick and certain between fraud and non-fraudulent transactions so that the right accounts will not get charged for products they did not pay for. Credit card Fraud detection is a real time problem that becomes an area of interest for Machine learning. Pecenties and other financial Companies and institutions end up draining their large and valuable resources and fraud, and fraudsters who are on the lookout for new ways of cheating, defrauding and engaging in all manner of unlawful activity. Hence, systems of fraud detection include fear relevant to all the banks that issue credit cards with a view of reducing their losses. Credit card fraud incidents in US once was 60% and multiple times 45% of the people 7% of fraud is due to theft or loss of the card and 93% remote access to person’s information in the year 2023.bed time. Credit card frauds are easy targets. Without any risks, a significant amount can be withdrawn without the owner’s knowledge, in a short period. Fraudsters always try to make every fraudulent transaction legitimate. Making fraud detection very challenging and difficult task to detect.

Credit card companies must distinguish fraudulent from non-fraudulent transactions so that their customers’ accounts will not get affected and charged for products they did not buy. Credit card fraud detection is a relevant problem that draws the attention of machine-learning. Many financial Companies and institutions lose massive amounts of money because of Fraud and fraudsters that are seeking different approaches continuously to violate the rules and commit illegal actions. Therefore, systems of fraud detection are essential for all banks that issue credit cards to decrease their losses. Multiple methods are used to detect fraudulent behaviours, such as Neural Networks, Decision Trees, K-nearest neighbor algorithms, and Support Vector Machines. Those ML methods can be applied independently or collectively with ensemble or meta-learning techniques to develop classifiers. Reports of credit card fraud in US have experienced 60% (once) and 45% of the people (multiple times) 7% fraud are due to theft or lost of the card while 93% remote access to personal information in the year 2023.

## 1.1 Literature Survey

Various supervised and semi supervised machine learning methods are employed for fraud detection but Our approach is to address three major issues with card frauds related dataset: strong class imbalance problem, mixing of labelled and unlabelled samples and lastly, to have higher capacity to handle large number of transactions.

In predicting real-time fraudulent transactions, Supervised machine learning categories including Decision Trees, Least Squares Regression, Logistic Regression, and the SVM method are applied. Two approaches of random forests used for training the behavioural characteristics of normal and anomalous transactions are identified.

They are the random-tree based, random forest and the CART based. Therefore, random forest gives quite nice result when dealing with small set data, yet there persist some issues in imbalanced data. The future work will concentrate on addressing the problem mentioned above. There is one thing that still needs to be changed in the random forest, the algorithm of the random forest.

- Identify Frauds: Detecting cyclical tendencies over a period and outliers in transaction data which are hallmarks of fraudulent transactions.
- Improve Accuracy of Detection Models: Raise the efficiency of utilisation of the models, which detect fraud.
- Minimize False Positives: Reducing the number of actual transactions that are classified as potential frauds.
- Optimize Speed of Detection: Further process the transactions as much and as fast as possible to minimize the effect of its nuisance to the legitimate users.
- Keep up with Changing Threats: About to be introduced new type of fraud strategy

### 1.2 Problem Statement

Terroristic financing and money laundering are among some of the biggest threats of the modern world and many organizations have written in their reports that they were victims of such creative criminal activities. The solution was always the same, but with the development and steady growth of the internet phenomenon, the problems to fight against happen to transform as well. The percentage of fraud goes beyond the scale of the loss being taken by the firm and minimized customer losses that are still able to ensure the business sustainability is attained. But of course customers also have something to contribute here, as customers to also would love to keep their accounts opened and active and thus spending power is not wasted in the revenue in the brand's pocket. Thus the case was able to explain the big picture of things. The technology enriches people with ideas that've become a manipulated pattern of activity: The process of opening several accounts in a single site and in all other facilities accessible on the Internet.

This is how multiple reusing of accounts and loss of details starts generating. It’s where online card processors get wiped, and account utilization in any place becomes engraved, from the dark web to the ecommerce itself. The most central objective of the solution is therefore to identify where the search for the cause of fraud ends and the search for reasonable arguments and data that validate it begins.

## 1.3 Objective of the project

A credit card fraud detection system must ensure that fraudulent transactions do not occur and that the true ones are detected along with the false ones at the best end user experience pertaining to the available transaction limits. Think about what will not trigger too many risk signals while at the same time ensuring that there is enough risk signals that fraud actors might stumble upon. Alert the relevant authorities of a number of fraudulent as well as one fraudulent transaction detect the false ones while maintaining the best user experience with regards to the transaction limits.

1. Detecting Fraud

- Design a system that will not relay too many risk signals but has a sufficient number of them for potential fraud actors to get detected.
- Detect a number of fraudulent as well as a single fraudulent transaction and alarm the appropriate authorities.

2. Accurate detection of illegal fraud

- For the purpose of accurately fraud’s pattern Recognition advancing statistical analysis and ML will be deployed.
- Resultantly every next automated model is comparatively better in terms of accuracy.
- Lowering the instances of wrong prohibitions
- Avoid indicating excessive amount of client transactions as fraudulent that would need pointless customer client troubles.

3. The goal is to reduce the False Positive Ratio however reducing this ratio should not compromise the reliability of the whole system.

- Various performance enhancements made in the Algorithm
- Employ the use of blockchaining in order to afford the mapping accurateness of the client’s private data.
- Knowledge of new forms of fraud and adaptation to the new characteristics of the system.
- Improved contact between the clients and the system
- Avoid letting as little experience on legitimate clients as possible.

- Provide regular alerts and also help everyone who has been conned in one way or another fraudulent transaction from taking place and detect the false ones while maintaining the best user experience with regards to the transaction limits.

With all these in place, the cost associated with losses through a credit card fraud detection system will reduce significantly for both the people and the financial organizations, and at the same time enhance the brand value.

## 1.4 Proposed System:

Consumers markets engaging through methods of credit cards giving an open area between transactions and transfers in the account. At any given go this can be abused in various ways to defraud and embezzle the card holder funds, information and property.

### System Architecture:

### Key Components:

### 1. Data Ingestion and pre-processing

- Collect Transaction data from various sources of Distribution like POS Terminals, online merchants, and mobile devices.
- Manage and cross-check data to flag discrepancies in detail.
- Ensure ready analysis through data normalization and standardization.

### 2. Feature Engineering

- The amount that changed hands during the transaction.
- Taxpayer Identification Number (TIN).
- Date and time the transaction took place.
- The area or region where the transaction took place.
- Activities of the user defined as behavioral econometrics.
- The Device used to carry out the transaction or the purchase.
- Speed at which the transactions occur.
- Consumers buying habits.

### 3. Model Training and Deployment

- Set a history of fraudulent activities through a non-regular use of card details and images, trade counterfeiting and modifications. Train models of supervised machine learning:

- Detailed images of cardholders.
- Logistic Regression.
- All the transactions were recorded as Decision Trees.
- The acquisitions were made by Gradient GA Boosting Machines.
- Support was through Warp-potting drivers.
- Neural Networks utilized Feedforward strategies, Recurrent networks and Convolutional imaging tactics.
- Other Data Models Utilized k-clusters as supervised Learning tools.
- An addition or verification step using Hierarchical models.
- Data made an alteration through simple Isolation Forests.

# CHAPTER 2

# SYSTEM REQUIREMENTS SPECIFICATIONS

# CHAPTER 2

## SYSTEM REQUIREMENTS SPECIFICATION

System requirements specification for credit card fraud detection provides outlining of functional, non-functioning and technical requirements for the system.

**Functional Requirements:**

- Real-time Transaction Monitoring Monitor incoming transactions in real-time
- Fraudulent Transaction Detection Detect transactions with high chances of fraud
- False Positive Reduction Minimize the number of valid transactions reported as fraudulent
- Alert Generation Produce alerts for suspicious transactions.
- User Interface: Provide a user-friendly interface for monitoring and managing alerts.

**Non-Functional Requirements:**

- Performance: Process transactions in real-time with minimal latency.
- Scalability: Handle increasing transaction volumes.
- Security: Protect sensitive customer data.
- Reliability: Ensure high system availability.
- Maintainability: Facilitate updates and modifications.

## 2.1 Software and Hardware Requirements:

**Software Requirements:**

Operating system: Windows 8/10/11/12

Programming language: Python

Library: pandas, Numpy

Simulation tool: Google collab

**Hardware Requirements:**

Processor : Any processor above 500

RAM : 2 GB

Hard disk : 80GB

## 2.2 Software Description:

**Python:**

Invented in 1989 by Guido Rossum, Python is an object-oriented programming language known for its speed in creating prototypes for complex applications. Beyond its core functionality, Python interfaces with various operating system calls and libraries, and even allows for extension into C or C++. Major companies like NASA, Google, YouTube, and BitTorrent all leverage Python's capabilities. Popular in fields like Artificial Intelligence, Natural Language Generation, and Neural Networks, Python prioritizes clear, readable code. This course will guide you through Python from its fundamental concepts.

**Characteristics of Python**

- Its syntax is clear and easier to understand compared to other languages.
- It provides a diverse range of data types to work with.
- Python scripts can run on various operating systems without modifications, making them versatile.
- It allows for dynamic changes during program execution, offering more adaptability.
- Python boasts libraries with functionalities like text manipulation (similar to Perl and Awk) that work seamlessly across Linux, Mac, and Windows.

**Google collab:**

Google Colab, short for Colaboratory, is a powerful cloud-based platform that allows users to write and execute Python code in a Jupyter Notebook environment. It's particularly well-suited for machine learning and data science tasks.

**Key Features:**

- Free Access to Powerful Computing Resources: Colab provides access to Google's cloud infrastructure, including GPUs and TPUs, even if you don't have a powerful computer of your own.
- Easy to Use: No setup is required. Simply create a Google account and start coding.
- Flexible: Train and run machine learning models, process data, create visualizations, and collaborate with others.
- Collaboration: Share your notebooks with others and work together in real-time.
- Integration with Google Drive: Seamlessly save and load your notebooks from Google Drive.
- Pre-installed Libraries: Many popular machine learning libraries are pre-installed, saving you time and effort.

**The Library & Packages:**

**NumPy:**

Short for "Numeric Python" or "Numerical Python," NumPy is an open-source extension module that injects Python with a potent dose of mathematical and numerical muscle. This freely available library provides precompiled functions that excel in performing these calculations, significantly accelerating computation time. But NumPy's impact goes beyond speed. It also equips Python with powerful data structures specifically designed for efficient multidimensional arrays and matrices. These structures are built to handle even massive datasets, ensuring smooth operation regardless of data size.

Furthermore, NumPy offers a comprehensive library of high-level mathematical functions specifically tailored to operate on these arrays and matrices. This combination of optimized data structures and specialized functions makes NumPy the cornerstone of scientific computing with Python. The library boasts a rich feature set, including, but not limited to:

- A powerful N-dimensional array object, providing a robust foundation for complex data manipulation.
- Sophisticated broadcasting functions, enabling elegant and efficient operations on large datasets.
- Tools for seamless integration of C/C++ and Fortran code, allowing leverage of existing codebases and libraries written in these languages.
- Valuable functionalities for linear algebra, Fourier transforms, and random number generation, providing a comprehensive suite of tools for scientific computing tasks.

**Pandas:**

Pandas is a powerful and open-source Python library specifically designed for data manipulation and analysis. It provides high-performance, easy-to-use data structures and operations for working with tabular data (like spreadsheets or SQL tables).

**Key Features:**

- Data Structures
- Data Manipulation

**Hardware Description:**

**Central Processing Unit (CPU):**

- Minimum clock speed: 500 MHz (Megahertz)
- Architecture: This specification predates widespread multi-core processors. A single-core processor would have been standard.
- Instruction set: Likely x86 architecture (most common for PCs at the time)
- Additional notes: While any processor above 500 MHz is technically acceptable, performance will improve with higher clock speeds and features like cache memory.

**Operating System:**

- Type: 64-bit operating system (e.g., Windows 7 64-bit, Linux 64-bit
- Additional notes: A 64-bit OS can address more than 4 GB of RAM, allowing it to utilize the full 2 GB available. 32-bit operating systems would only be able to use a portion of the RAM.

**Memory (RAM):**

- Capacity: 2 GB (Gigabytes)
- Type: Likely DDR SDRAM (Synchronous Dynamic Random-Access Memory) was the most common type at this time.
- Speed: Speed is not specified, but higher speeds would improve system performance.
- Number of slots: The number of RAM slots would determine how much memory could be added for

future upgrades.

**Storage:**

- Capacity: 80 GB (Gigabytes)
- Type: Likely a Hard Disk Drive (HDD) with a rotational speed of 7200 RPM (Revolutions Per Minute) - a common standard at the time.
- Interface: Likely connected via a Parallel Advanced Technology Attachment (PATA) interface, although Serial ATA (SATA) was starting to become more common.
- Additional notes: Solid State Drives (SSDs) were not widely used for personal computers at this time due to higher cost and lower capacities.

# CHAPTER 3

# SYSTEM DESIG

# CHAPTER 3

# SYSTEM DESIGN

**INTRODUCTION:**

**Understanding the Problem:**

The issue of credit card fraud poses a big risk to financial institutions as well as consumers. At the time of transaction, fraudsters use multiple tactics to obtain the card's information and conduct unauthorized transactions, leading to losses and breach in consumer trust. There is also the need to have robust systems in place in banking institutions to detect and disallow fraudulent acts in real time.

## 3.1 Algorithms:

- Description of Algorithms: A brief summary of each algorithm is presented emphasizing its advantages and possible areas of application in fraud detection.
- Conditions for the Choice of the Algorithm: The text stresses the need to take into account other aspects when selecting an algorithm including the nature of the data, performance specifications, and interpretability. This shows a realistic view with regard to the predicative model of the target variable.
- Clear Framework: The information is well and systematically arranged, thus enhancing comprehension and ease of following the writer's argument(s).

## 3.2 Network Architecture:

One of the main tasks of the banking sector is to identify credit card fraud on time. They are to look for transactions which are illegal in nature in due time so that the customers can be protected and no losses can be incurred. Machine learning models are often used for this problem which in turn will use more computing power, since processing and analyzing transaction data properly needs a rigid network architecture.

## 3.3 Data preprocessing:

1. It represents a crucial step of cleaning raw data and formatting it properly into tidy data for further analysis. It includes a number of major processes:
2. Data Cleaning: This is a process where missing data is managed, duplication is removed and other inconsistencies are corrected for purposes of ensuring accuracy of the data.
3. Data Transformation: It involves changing information into the required format which may include but not limited to expansions, scaling, normalizations and encoding of categorical variables.
4. Feature Engineering: It means taking existing features and constructing or generating additional new features with the purpose of enhancing performance of the model in search of useful information.

**Model Selection:**

The concept of model selection in ml can be generally understood as a process of choosing the right algorithm and architecture based on the specific task or data set. This involves cross-validation of and comparison of several models on the basis of fitting the model to the data and the best model for that data in terms of results achieved. Common considerations when picking a model include the complexity of the model, data processing requirements and its applicability to new data.

**Network Topology:**

Network topology means the arrangement of the links and nodes in a specific communication network and their relationships. It helps in understanding the outline of the network and the direction of data among various devices. Network topology can be physical which is the true picture of where devices and cables are placed or logical which is the picture of data placement.

## Use case:

The components of the fraud detection system commence with the processes involved right from data uploading to the successive display of the results.

Stepwise description:

1. Credit Card Dataset Upload: As the title suggests, the process begins with a user appeasing in a dataset which consists of information regarding transactions related to credit cards. This data set will assist the users in training as well as testing the fraud detection model.

2. Train and Test Model Generation: As such, the uploaded data set is divided into two sections:

- Training Set: This is used to train the machine learning model to learn the patterns of the data and identify the transactions that are likely to be fraudulent.
- Test Set: This set is used in assessing the performance of the trained model on data that the model has never seen before.

3. Fraud Detection using Test Data: The random forest model which is trained is then applied on the test dataset to predict transfer transactions that were considered to be dubious or otherwise. The set of prediction outputs is the set of values which were predicted for those transactions.

4. Clean And Fraud Transaction Graph: The previous stage results are examined. The system may discriminate and disambiguate unreasonable items of false positives/negatives and removal of them if the case.

# CHAPTER 4

# SYSTEM IMPLEMENTATION

# CHAPTER 4:

## SYSTEM IMPLEMENTATION

### Introduction

The goal of this project is to implement a real-time credit card fraud detection system based on machine learning algorithms. The main objective or goal of the project is to establish a system which will effectively identify fraudulent transactions with as few false positives as possible and with as little inconvenience as possible to legitimate cardholders.

## 4.1 System Architecture and Development

On this document, we describe the system architecture and the process of creation of a credit card fraud detection system.

### System Architecture

1. Data Ingestion Layer: It contains all the transactional information, such as payment processors, banks, and so forth.
2. Data Processing Layer: It prepares and reorganizes the data into the right form for the to be easy to analyze.
3. Machine Learning Layer: Employs ML algorithms aimed at identifying the fraudulent behavior.
4. Decision Engine Layer: Processes the model outputs, combined with some rules and makes decisions.
5. Alert and Notification Layer: After processing comes alerting and sending notifications to different stakeholders.
6. Data Storage Layer: Responsible for saving master files of transactional data, model results, and other useful files.

#### Scalability and Performance

1. Distributed Architecture: You really want to think about a distributed architecture if you're expecting high traffic and loads of data.
2. Load Balancing: Using load balancing techniques can help make sure resources are used efficiently, which is always a plus.
3. Caching Mechanism: And let's not overlook caching! It can really help cut down on latency, making everything feel snappier.

## 4.2 Security Consideration:

**1. Data Encryption:** We first have to think about the encryption of sensitive information. Now, this is credit card information and other details that would allow you to identify someone. This information must be encrypted at all times when it's traveling over networks and also just sitting there on servers.

**2. Access Control:** Then there's the whole access control thing. We should definitely set up role-based access controls and ensure that we have solid authentication methods in place.

**3. Security Audits:** Don't underestimate the importance of security audit they're a regular necessity. It's definitely a smart move to conduct these audits along with some penetration testing every now and then.

## 4.3 Explainability & Transparency:

### Explainability

Explainability techniques for credit card fraud detection provide insights into the decision-making process of machine learning models, making them more transparent, accountable, and trustworthy. Here are some explainability techniques used in credit card fraud detection.

1. Feature Importance: This tool tells us which pieces of information are most important for the box to make its predictions. It is as if finding out what ingredients are the most essential to bake a cake perfectly.
2. Partial Dependence Plots: This tool shows us how changing a single piece of information affects the box's predictions. It is like showing how adding more sugar to the cake recipe changes the taste of the cake.

### Transparency

Transparency in credit card fraud detection refers to the openness and clarity with which the fraud detection process is explained and visualized, enabling stakeholders to understand how decisions are made and how the system works.

Types of Transparency

1. Explanation: Explaining the machine learning models employed, their algorithms, data sources and feature engineering in detail.
2. Transparency in data management: Involves providing access to both the data used for model training and evaluation, as well as the information used to make predictions.
3. Clear Decision-Gattering: Definable justifications for the system's decisions, including rationales framework.

# CHAPTER 5

# SYSTEM TESTING

CHAPTER 5

# SYSTEM TESTING

System testing in the general sense is the final step in credit card fraud detection where the investigators have to ensure that the implementation of the designed system has a verified functionality with required level of security, performance and reliability. This stage also takes care in assuring whether the implemented system does not incorrectly classify any of the complete data sets that were submitted as being fraudulent, while also providing a sufficient number of correctly fraudulent transactions. It also assesses the interconnection of the system with such components as payment gateways or external sources, such as fraud databases, for the performance of sharing data and real time fraud prevention.

## 5.1 Functional Testing:

Every single platform and branch of banking related to a credit card has a possible risk of fraud, thus multiple subtopics come under credit card fraud detection, one of which is functional testing.

### Transaction Validation:

A fraud detection system's most important aspect is to confirm each transaction and whether it is fraudulent or not. The present system undergoes functional testing which makes sure all the rules are exercised in evaluating the transaction. These may include anomalies like a transaction that involves high dollar amounts, several transactions occurring in a short time, geographic locations that are not familiar, or the card itself being used in different places.

### Detection Accuracy:

Functional Testing enables the system to detect fraudulent behavior with a high degree of accuracy. The system should have a very low cut-off in their false positive rate, as well as a false negative rate .During a number of tests, various situations are simulated including varying degrees of fraud patterns while normal user behaviors are espoused to see how the system can differentiate between the two.

### Real-Time Fraud Detection:

An important aspect of fraud detection functional testing is how well the system runs in real-time or nearly real-time. Credit card fraud detection needs to flag the suspicious transactions before they actually happen to prevent losses, and the system needs to analyze transaction data quick enough to make decisions almost instantly. It tests if the system can handle huge transaction volumes at the same time, process them in real-time, and identify fraud without delay.

### Pattern Recognition:

Modern fraud detection systems rely on sophisticated algorithms that help them recognize fraudulent patterns. Functional testing will prove whether the system can actually identify these patterns, for example, a change in spending pattern or multiple transactions in a very short period or usage of a card in a new or unusual location. For example, it should detect suspicious behavior, such as several large purchases from various countries within a short period. Testing employs known fraud patterns as test cases to confirm that the system can easily identify such cases. Adaptability: The system should adapt to changes in fraud patterns by revising rules and algorithms according to emerging threats.

## 5.2 Performance Testing:

Credit card fraud detection performance testing is significant in that it will test whether the system performs effectively and efficiently in all aspects, especially with regard to a high number of transactions. It monitors the response, stability, scalability, and tolerance of the system when there is a heavy transaction load and does not collapse. In this case, the most significant performance testing areas include load testing, stress testing, scalability testing, and latency testing. Let's explore each of these in more detail:

### Throughput Testing:

It looks into how many transactions a given fraud detection system is able to process in a given period. This aspect has special relevance to huge operation scenarios where the system should be able to deal with thousands of credit card transactions spread across several regions or channels of payment. Thus, the throughput test essentially attempts to check whether thousands and even millions of transactions could be processed by the system per minute or even a second with the required precision in detection. This will prevent the system from slowing down and not missing fraud as transaction volumes go up.

### Safety Testing:

Credit card fraud detection safety testing focuses on the security of the system to ensure it is resilient and capable of protecting sensitive data from unauthorized access, misuse, or attacks. It evaluates security measures in place to prevent, detect, and respond to security breaches that may compromise the integrity of the system or expose customers to fraud.

### Environment Testing:

Environment testing in credit card fraud detection refers to validating how well the fraud detection system works and integrates within its intended operating environment, including hardware, software, network configurations, and other system dependencies. The aim is that the system works as it should in different environments and performs reliably under real conditions. This kind of testing identifies the issues that might come from other environmental factors like hardware configurations, software dependencies, or conditions of a network.

## 5.3 Data Encryption and Privacy Protection:

The most critical aspect of testing for fraud detection systems involves safety testing. All the sensitive data about customers should be encrypted in both transit and rest forms. It includes credit card numbers, personal information, and transaction history. It should be checked that the encryption algorithms applied are powerful enough to guard against interception or unauthorized decryption. In addition, safety testing is conducted to ensure that no personal or financial information is leaked during the transaction processing or system operations. Encryption protocols such as TLS for data in transit and AES for data at rest need to be tested to be sure they are up to industry standards for data protection.

## 5.4 Authentication and Access Control:

Safety testing ascertains that the fraud detection system is well-equipped to authenticate all users, administrators, and systems accessing any sensitive data. This implies testing of multi-factor authentications which add an additional layer beyond just passwords for security, and role-based access control, which helps ensure only authorized personnel accesses specific data or system functionality. Testing should also guarantee that unauthenticated users cannot bypass the mechanisms of authentication or get high levels of privilege in the system. This is mainly about ensuring that fraud analysts, among other users, gain access only to data which they require to execute their roles.

## 5.5 Hardware Configuration and Resource Availability:

Fraud detection systems require certain hardware configurations to process huge amounts of data, complex fraud detection algorithms, and managing high transaction loads. Environmental testing ensures that the system works well across different hardware setups, such as servers, CPUs, and memory configurations. This testing will verify if the hardware can take the peak workload, especially when running intensive machine learning algorithms, processing transactions, or monitoring real-time data streams. Testing also verifies whether the system consumes resources efficiently without overloading the hardware, which might have an impact on performance or cause downtime.

## Network Configuration and Bandwidth:

A very important aspect of environment testing is checking how well the fraud detection system performs under different network conditions. This would be to test how it can work with multiple network configurations such as corporate networks, cloud environments, and hybrid systems. Network bandwidth, latency, and stability would all have a major effect on the performance of the system, especially when used for real-time fraud detection. For example, a slow network may result in delayed transaction processing or delayed fraudulent activity detection. Environmental testing ensures that the system will support a variety of network speeds and continue to process information in real-time, even when network conditions are bad or high volumes of traffic exist.

## Transaction Processing and Fraud Detection Modules:

A key integration in credit card fraud detection is between the transaction processing system and the fraud detection algorithms. These modules must operate together seamlessly, with the transaction processing system sending relevant data to the fraud detection system for analysis. Integration testing verifies that this data is transmitted correctly and that the fraud detection system can make prompt decisions based on it. The system should be capable of flagging suspicious transactions while allowing legitimate ones to proceed. Testing should ensure that all necessary parameters for the fraud detection algorithm are provided accurately and that the responses are correct.

## Database Integration:

Fraud detection systems usually depend on databases to hold essential information like transaction records, user profiles, fraud history, and detection outcomes. Integration testing is crucial to confirm that data is accurately retrieved from and stored in the database. This testing verifies whether the system can access and update records in real time when fraud is identified or when more information is required to assess transactions.

# CHAPTER 6

# RESULTS AND DISCUSSION

# CHAPTER 6

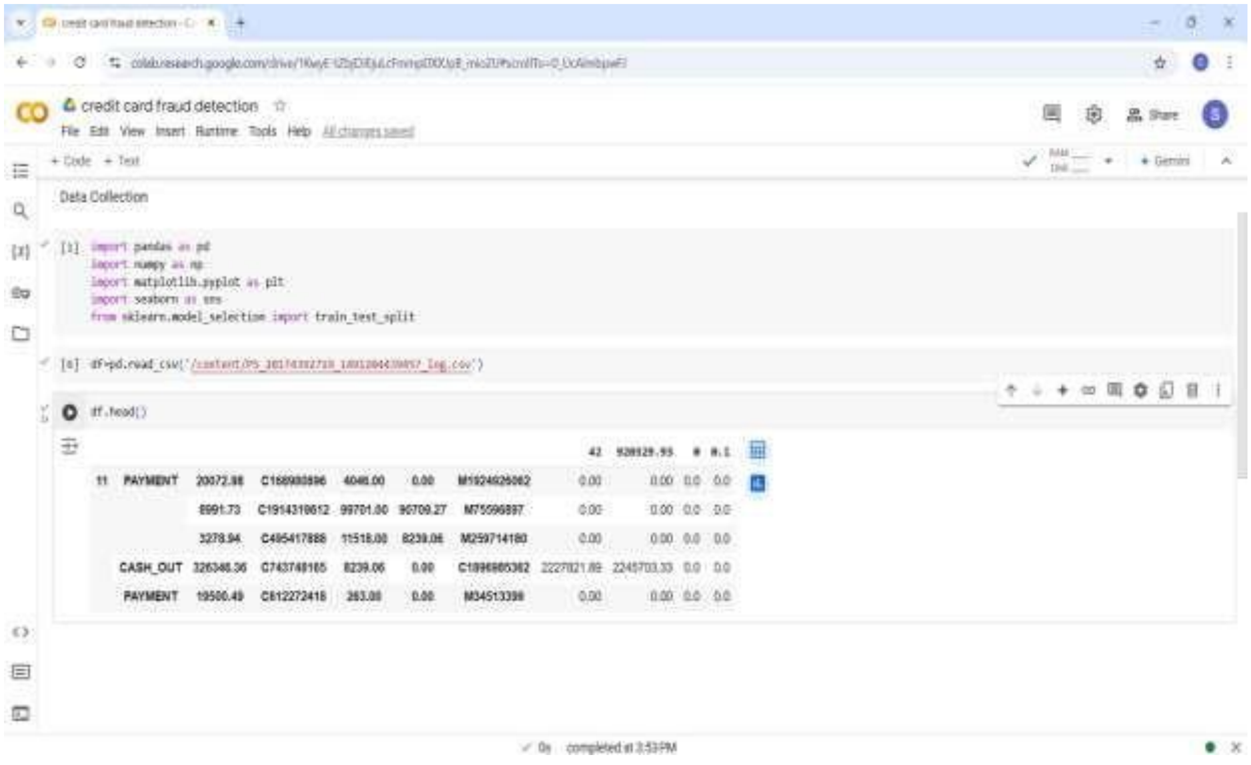
# RESULTS AND DISCUSSION

## Introduction:

In this chapter, we delve into the results and discussion of our research. The foundation of any successful machine learning project lies in the quality and quantity of the data used. Therefore, we begin by outlining the data collection process employed in this study.

## 6.1 Data Collection:

Data used in this paper is a set of product reviews collected from credit card transactions records. This step is concerned with selecting the subset of all available data that you will be working with. ML problem start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called labelled data.



Data Collection

```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
```

```
[6] df=pd.read_csv('/content/PS_20174392719_1491204439457_log.csv')
```

```
df.head()
```

```
                                                                        42  928129.95  0  0.1
11  PAYMENT   20072.88  C168980896  4046.00  0.00      M1924926062  0.00        0.00  0.0  0.0
              8991.73   C1914310612 99701.00 90709.27  M75596897    0.00        0.00  0.0  0.0
              3278.94   C495417888  11518.00 8239.06   M259714180   0.00        0.00  0.0  0.0
    CASH_OUT  326346.36 C743748165  8239.06  0.00      C1896985362  2227821.89  2245703.33  0.0  0.0
    PAYMENT   19500.49  C612272418  263.00   0.00      M34513398    0.00        0.00  0.0  0.0
```

credit card fraud detection
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
completed at 3:53 PM

**Fig 6.1 Importing python packages for data exploration**

in the above fig 6.1 we are importing the packages for data exploration to run the modules.

## 6.2 Data Pre-processing:

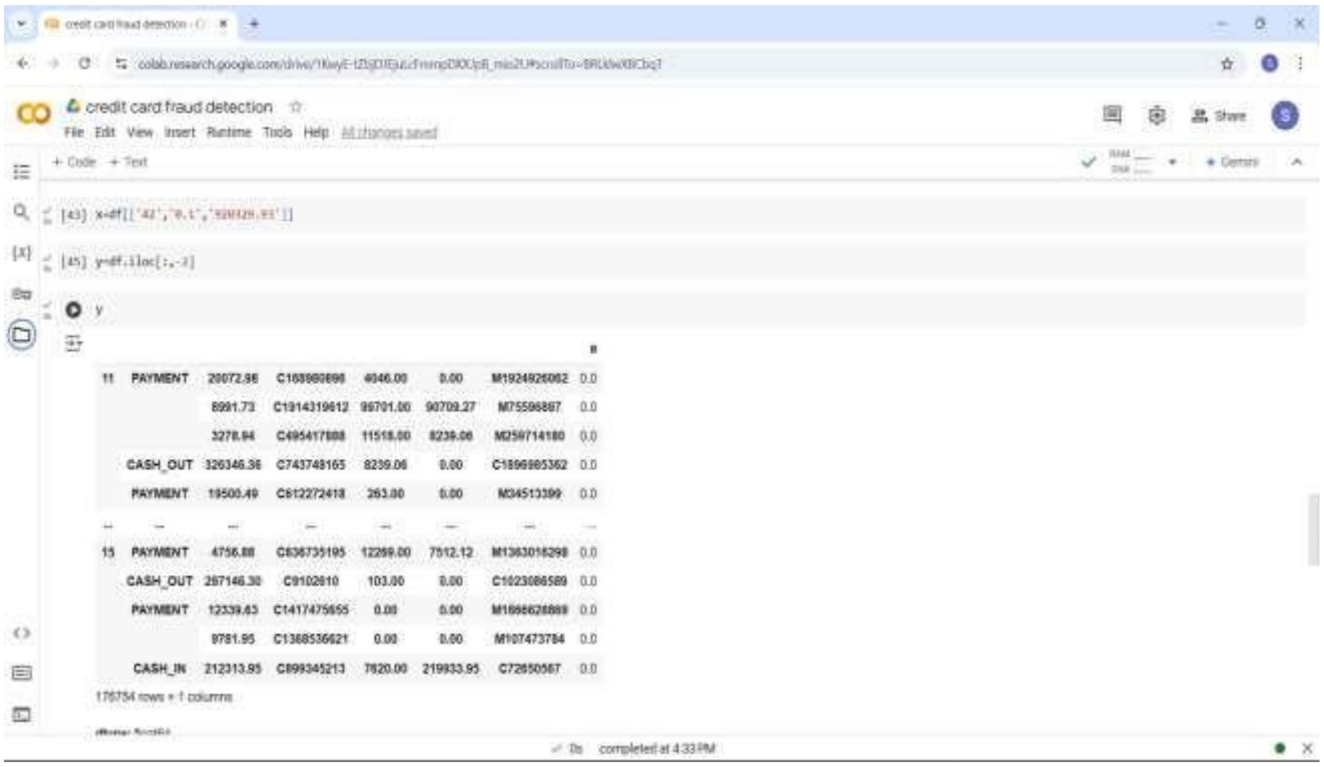
Pre-processing is the process of three important and common steps as follows:

- Formatting: It is the process of putting the data in a legitimate way that it would be suitable to work with. Format of the data files should be formatted according to the need. Most recommended format is .csv files.
- Cleaning: Data cleaning is a very important procedure in the path of data science as it constitutes the major part of the work. It includes removing missing data and complexity with naming category and

so on. For most of the data scientists, Data Cleaning continues of 80% of work.

- Sampling: This is the technique of analyzing the subsets from whole large datasets, which could provide a better result and help in understanding the behavior and pattern of data in an integrated way.

The fig 6.2 shows a sample of training and testing data for a credit card fraud detection.


credit card fraud detection
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Share

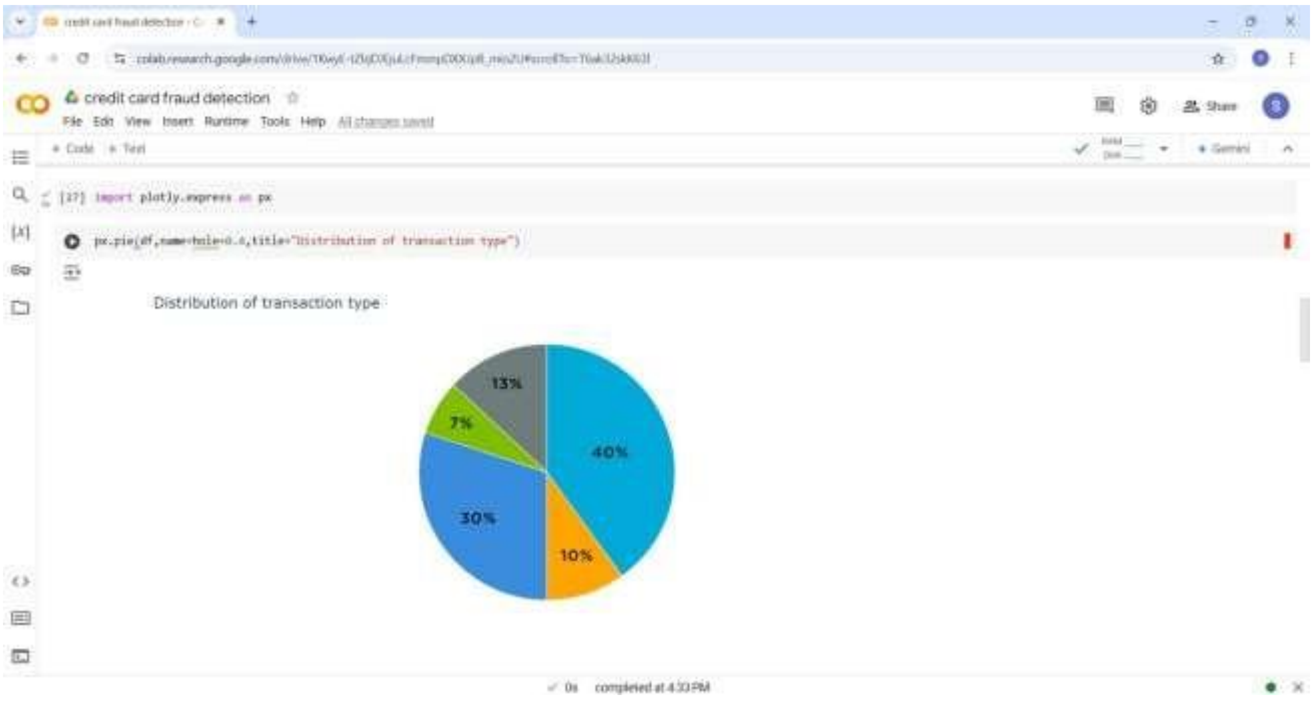
```
[43] x=df[['42','0.1','920129.93']]

[45] y=df.iloc[:,-1]

y

                                                                           0
11  PAYMENT   20072.98   C188980898   4046.00      0.00   M1924926062   0.0
                8991.73  C1914319612  99701.00  90709.27   M75596887    0.0
                3278.94  C495417888   11518.00   8239.06   M259714180   0.0
   CASH_OUT  326346.36   C743748165    8239.06      0.00   C1896985362  0.0
    PAYMENT   19500.49   C612272418     263.00      0.00   M34513399    0.0
 ...    ...        ...          ...        ...       ...          ...   ...
15  PAYMENT    4756.88   C636735195   12289.00   7512.12   M1363016298  0.0
   CASH_OUT  287146.30   C9102610       103.00      0.00   C1023086589  0.0
    PAYMENT   12339.63   C1417475655      0.00      0.00   M1898628889  0.0
                9781.95  C1368536621      0.00      0.00   M107473784   0.0
    CASH_IN  212313.95   C899345213    7620.00 219933.95   C72650567    0.0
176754 rows × 1 columns
```

**Fig 6.2 Training and Testing data**

**Data Visualization:**

- Data Visualization is the method of representing the data in a graphical and pictorial way, data scientists depict a story by the results they derive from analysing and visualising the data.
- The pie chart visualizes the distribution of transaction types in the dataset.
- Payment transaction constitute the largest proportion, accounting for approximately 40% of the total transactions.
- CASH_OUT transactions are the second most frequent, making up about 30% of the total.
- CASH_IN transactions represent 10% of the total, and the remaining 20% belong to the “other” category.
- It is important to consider the potential impact of this class imbalance on the model’s performance and to use appropriate techniques to address it.
- The dominance of PAYMENT transactions suggests that the dataset may be biased towards this type of activity.

In the fig 6.3 it is important to consider the potential impact of this class imbalance on the model's performance and to use appropriate techniques to address it.

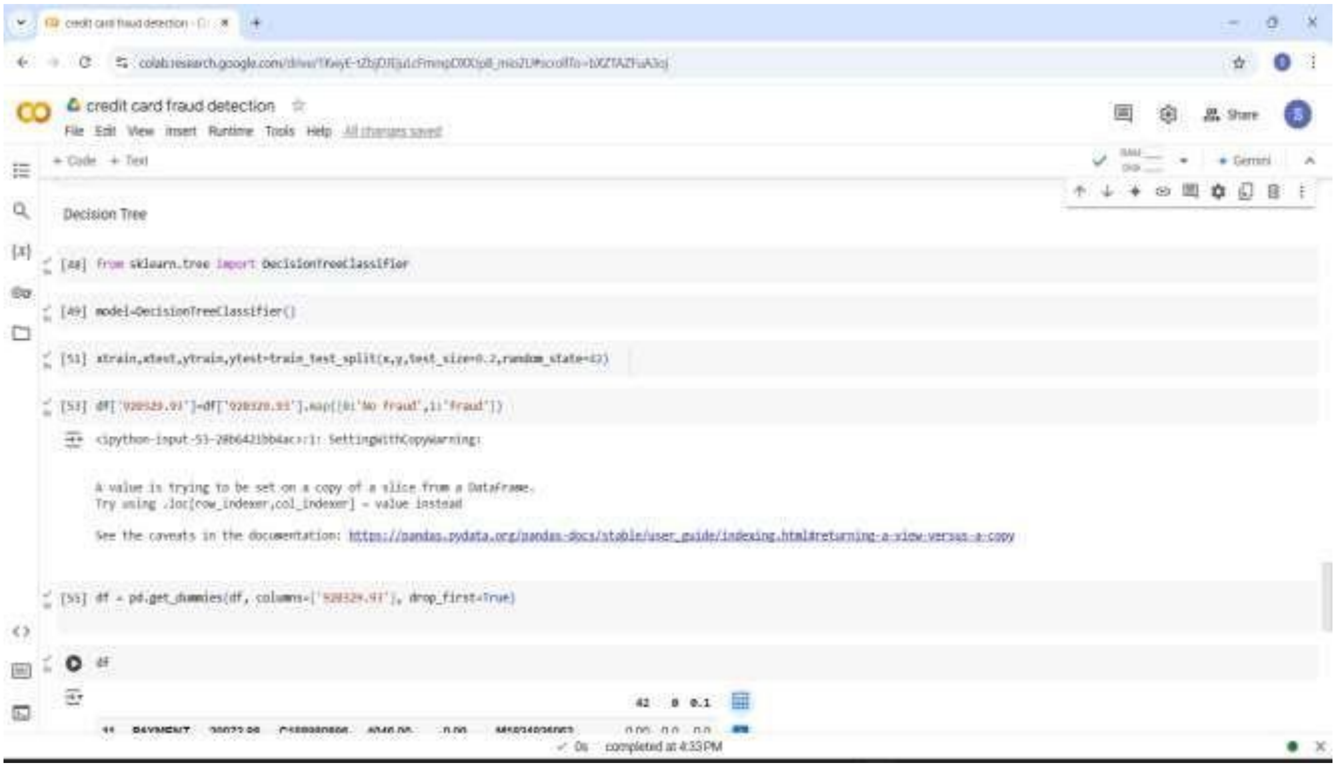

credit card fraud detection
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Share
Gemini
Distribution of transaction type
13%
7%
40%
30%
10%
completed at 4:33 PM

```
[27] import plotly.express as px

px.pie(df,name=hole=0.4,title="Distribution of transaction type")
```

Fig 6.3: Data visualization of distribution of transaction type

**Decision Tree (D.T.):**

Supervised learning performs as a decision tree that takes the form of a tree structure such that it consists of a root node and other nodes that are split in a binary or multi-split manner into child nodes whereby each tree employs its algorithm in performing the splitting process.


credit card fraud detection
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Share
Gemini
completed at 4:33 PM

Decision Tree

```
[48] from sklearn.tree import DecisionTreeClassifier

[49] model=DecisionTreeClassifier()

[51] xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=42)

[53] df['920529.93']=df['920320.93'].map({0:'No fraud',1:'Fraud'})

<ipython-input-53-28b6421bb4ac>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

[55] df = pd.get_dummies(df, columns=['920529.93'], drop_first=True)

df

                                                                42    0   0.1
 44  PAYMENT   20072.08   C[illegible]   4046.08   0.00   M[illegible]   0.00  0.0  0.0
```

Fig 6.4: implementing decision tree

Fig 6.4 shows the implementation of a decision tree model for credit card fraud detection in python.

## 6.3 Feature extraction:

Feature extraction is the process of studying the behavior and pattern of the analyzed data and draw the features for further testing and training. Finally, our models are trained using the Classifier algorithm. We use classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre-processed data.

## 6.4 Evaluation model:

Model Evaluation is an essential part of the model development process. It helps to find the best model that represents our data and how well the selected model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can effortlessly generate overoptimistically and over fitted models. To avoid overfitting, evaluation methods such as hold out and cross-validations are used to test to evaluate model performance. The result will be in the visualized form. Representation of classified data in the form of graphs. Accuracy is well-defined as the proportion of precise predictions for the test data. It can be calculated easily by mathematical calculation i.e. dividing the number of correct predictions by the number of total predictions.

# CHAPTER 7

# CONCLUSION

**CHAPTER 7**

# CONCLUSION AND FUTURE ENHANCEMENT

## INTRODUCTION:

This introduction provides a comprehensive overview of the “Conclusion and Future Enhancement” section, emphasizing the project’s goals, methodology, key findings, and the scope of the future enhancements.

## 7.1 CONCLUSION:

Hence, we have acquired the result of an accurate value of credit card fraud detection i.e. 0.9994802867383512 (99.93%) using a decision tree algorithm with new enhancements. In comparison to existing modules, this proposed module is applicable for the larger dataset and provides more accurate results. The decision tree algorithm will provide better performance with many training data, but speed during testing and application will still suffer. Usage of more pre-processing techniques would also assist.

## 7.2 FUTURE ENHANCEMENT:

Using the model with different datasets that differ in dimensions and nature, modifying the split ratio, as well as using a varying algorithm can be ways in which the model can be improved. Our future work will try to represent this into a software application and provide a solution for credit card fraud using the new technologies like Machine Learning, Artificial Intelligence and Deep Learning.

## BIBLIOGRAPHY:

## REFERENCE:

**[1]** https://www.xoriant.com/blog/productengineering/decision-trees-machine-learningalgorithm.html

**[2]** Quah, J. T. S., and Sriganesh, M. (2020). Real-time credit card fraud detection using computational intelligence. Expert Systems with Applications, 35(4), 1721-1732.

**[3]** 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.

**[4]** 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-IJERTV8IS090031.pdf, 2002. Accessed: 23-oct-2023

## APPENDIX:

## Source Code:

### Imports:

```
#Import Necessary Packages
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
from pylab import rcParams
```

### Access Data:

### First set of steps followed:

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive')

 import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content/gdrive/My Drive/"

  %cd /content/gdrive/My Drive/

  dataFrame = pd.read_csv("./creditcard.csv")
```

### Second set of steps followed:

```
# dataFrame = pd.read_csv("/content/sample_data/creditcard.csv")
```

### Load Data

The data includes 492 fraudulent transactions out of 284,807 total transactions using European credit cards over the course of two days. A Principle Component Analysis (PCA) has been used to reduce everything except the time and amount due to privacy concerns.

```
dataFrame.isnull().values.any()
False

dataFrame.shape
(284807, 31)

dataFrame.head()
```

#### Data Analysis

Let's see some graphic proof that the data in this fraud dataset is uneven.

```
countClasses = pd.value_counts(dataFrame['Class'], sort = True)
countClasses.plot(kind = 'pie', rot=0)
plt.title("Distribution of Transaction class")
plt.xticks(range(2), LABELS)
([<matplotlib.axis.XTick at 0x7f15f17d8a90>,
  <matplotlib.axis.XTick at 0x7f15f17d8a60>],
[Text(0, 0, 'Normal'), Text(0, 0, 'Fraud')])
```

**Overview of Transaction Amount Data Statistics**
Two data frames, one for legitimate transactions and the other for fraud, will be created from the dataset.

```
fraudData = dataFrame[dataFrame.Class == 1]
normalData = dataFrame[dataFrame.Class == 0]

normalData.shape
(284315, 31)

fraudData.shape
(492, 31)

normalData.Amount.describe()
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64

fraudData.Amount.describe()
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Even while the mean in the fraudulent transactions is a little higher, it is still within a standard deviation, making it difficult to distinguish between the classes with high precision using only statistical approaches.

**Visual Exploration of Transaction Amount vs. Hour**
Again, this is insufficient to create a reliable classifier. For instance, it would be challenging to clearly delineate between legitimate business transactions and fraud.

```
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
fig.suptitle('Transaction time vs Amount')

ax1.scatter(fraudData.Time, fraudData.Amount)
ax1.set_title('Fraud')

ax2.scatter(normalData.Time, normalData.Amount)
ax2.set_title('Normal')

plt.xlabel('Time(in s)')
plt.ylabel('Amount(in $)')
plt.show()
```

**Examination of the Transaction Amount Data Visually**

The areas of interest in datasets for anomaly identification frequently become "washed away" by an abundance of data. Many low-value transactions that are often boring can be found in this dataset (buying cups of coffee, lunches, etc). Looking at transactions that are $200 or more will likely provide information that the rest of the data is likely to obscure.

```
fig, (ax1, ax2) = plt.subplots(1,2, sharex=True)
fig.suptitle('Amount per transaction by class')

bins = 5

ax2.hist(normalData.Amount, bins = bins)
ax2.set_title('Normal')

ax1.hist(fraudData.Amount, bins = bins)
ax1.set_title('Fraud')

plt.xlabel('Amount($)')
plt.ylabel('No. of Transactions')
plt.xlim((0, 25000))
plt.yscale('log')
plt.show();
```

**Setup of the Model: Basic Autoencoder**

We are justified in investigating our autoencoder to see if it performs a little bit better now that more straightforward ways are not showing to be all that effective.

**Scale and normalize data**

Since time and quantity have extremely different magnitudes, the high magnitude value will probably "wash out" the small magnitude value. As a result, scaling the data to comparable magnitudes is frequent. As the majority of the data (aside from "time" and "amount") are the end result of a PCA analysis. The dataset was transformed into standard-normal form by the PCA performed on it. The columns for "time" and "amount" will get the same treatment.

```
data = dataFrame.drop(['Time'], axis=1)

data['Amount'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1, 1))
```

**Dividing the test and training sets**

Now, using a random seed that we wrote at the beginning of the code, we divided the data into training and testing sets based on the percentage.

```
X_train, X_test = train_test_split(data, test_size=0.3, random_state=42)
X_train = X_train[X_train.Class == 0]
X_train = X_train.drop(['Class'], axis=1)

y_test = X_test['Class']
X_test = X_test.drop(['Class'], axis=1)

X_train = X_train.values
X_test = X_test.values

X_train.shape
```

```
(199008, 29)

inputDim = X_train.shape[1]
encodingDim = 14

input_layer = Input(shape=(inputDim, ))

encoder = Dense(encodingDim, activation="tanh",
            activity_regularizer=regularizers.l1(10e-5))(input_layer)
encoder = Dense(int(encodingDim / 2), activation="relu")(encoder)

decoder = Dense(int(encodingDim / 2), activation='tanh')(encoder)
decoder = Dense(inputDim, activation='relu')(decoder)

autoencoder = Model(inputs=input_layer, outputs=decoder)
```

**Model Creation**
**Structure and parameters of the autoencoder layer:**
Layers for symmetric encoding and decoding are "dense" in the autoencoder. The input is being compressed into a kind of streamlined encoding before being expanded once more. The feature space is the input and output dimension (for example, 30 columns), therefore the encoding layer should be thinner by the expected quantity of features it expects to represent. High-level characteristics should be represented by about two columns in this situation because I am encoding 30 columns into 14 dimensions (30/14 = 2.1). I anticipate that each of those high-level features will correspond to around seven latent or hidden features in the data.
Also tuned to experimentally sound settings were the epochs, batch size, learning rate, learning strategy, and activation functions.

```
#Training and Logging Models
#The complete setup for the run, including
nb_epoch = 50
batch_size = 15

autoencoder.compile(optimizer='adam',
               loss='mean_squared_error',
               metrics=['accuracy'])

checkpointer = ModelCheckpoint(filepath="fraudDetection.h5",
                      verbose=0,
                      save_best_only=True)
tensorboard = TensorBoard(log_dir='./logs',
                    histogram_freq=0,
                    write_graph=True,
                    write_images=True)

history = autoencoder.fit(X_train, X_train,
               epochs=nb_epoch,
               batch_size=batch_size,
               shuffle=True,
               validation_data=(X_test, X_test),
               verbose=1,
               callbacks=[checkpointer, tensorboard]).history

Epoch 1/50
13268/13268 [==============================] - 30s 2ms/step - loss: 0.8101 - accuracy:
```

```
0.5782 - val_loss: 0.7700 - val_accuracy: 0.6703
Epoch 2/50
13268/13268 [==============================] - 29s 2ms/step - loss: 0.7381 - accuracy:
0.6819 - val_loss: 0.7526 - val_accuracy: 0.6890
Epoch 3/50
13268/13268 [==============================] - 30s 2ms/step - loss: 0.7271 - accuracy:
0.6922 - val_loss: 0.7456 - val_accuracy: 0.6993
Epoch 4/50
13268/13268 [==============================] - 30s 2ms/step - loss: 0.7220 - accuracy:
0.6947 - val_loss: 0.7435 - val_accuracy: 0.6996
Epoch 5/50
13268/13268 [==============================] - 31s 2ms/step - loss: 0.7192 - accuracy:
0.6971 - val_loss: 0.7390 - val_accuracy: 0.7029
Epoch 6/50
13268/13268 [==============================] - 31s 2ms/step - loss: 0.7165 - accuracy:
0.6988 - val_loss: 0.7374 - val_accuracy: 0.7070
Epoch 7/50
13268/13268 [==============================] - 33s 2ms/step - loss: 0.7147 - accuracy:
0.7011 - val_loss: 0.7349 - val_accuracy: 0.7106
Epoch 8/50
13268/13268 [==============================] - 34s 3ms/step - loss: 0.7118 - accuracy:
0.7046 - val_loss: 0.7345 - val_accuracy: 0.7130
Epoch 9/50
13268/13268 [==============================] - 33s 3ms/step - loss: 0.7084 - accuracy:
0.7116 - val_loss: 0.7310 - val_accuracy: 0.7181
Epoch 10/50
13268/13268 [==============================] - 31s 2ms/step - loss: 0.7057 - accuracy:
0.7191 - val_loss: 0.7283 - val_accuracy: 0.7271
Epoch 11/50
13268/13268 [==============================] - 35s 3ms/step - loss: 0.7043 - accuracy:
0.7205 - val_loss: 0.7282 - val_accuracy: 0.7223
Epoch 12/50
13268/13268 [==============================] - 31s 2ms/step - loss: 0.7039 - accuracy:
0.7204 - val_loss: 0.7258 - val_accuracy: 0.7309
Epoch 13/50
13268/13268 [==============================] - 30s 2ms/step - loss: 0.7030 - accuracy:
0.7215 - val_loss: 0.7268 - val_accuracy: 0.7202
Epoch 14/50
```

```
autoencoder = load_model('fraudDetection.h5')
```

**Model assessment**

More training epochs are probably not going to assist because the loss of our present model appears to be convergent. Let's investigate this visually to make sure.

```
 plt.title(' Loss in the Model')
plt.legend(['training Data', 'testing Data'], loc='upper right');
plt.ylabel('loss')
plt.xlabel('epochs')
plt.plot(history['loss'])
plt.plot(history['val_loss'])

[<matplotlib.lines.Line2D at 0x7f15eabfb970>]

predictions = autoencoder.predict(X_test)
```

```
2671/2671 [==============================] - 3s 1ms/step

mse = np.mean(np.power(X_test - predictions, 2), axis=1)
errorDf = pd.DataFrame({'reconstruction_error': mse,
                        'true_class': y_test})

#Check for Reconstruction Errors
#As we demonstrate below, autoencoders are trained to minimize reconstruction error.
errorDf.describe()
```

```
reconstruction_error      true_class

count                     85443.000000        85443.000000

mean                      0.717792            0.001592

std                       3.329529            0.039865

min                       0.032391            0.000000

25%                       0.235207            0.000000

50%                       0.380212            0.000000

75%                       0.611760            0.000000

max                       263.686291          1.000000
```

**ROC**

Most binary classifiers are intended to produce receiver operating characteristic curves as their output. They are somewhat less useful because the data set we have is unbalanced. Why? Because there are so relatively few examples of fraud, you may create a really good-looking curve by assuming everything is normal.

```
normalErrorDf = errorDf[(errorDf['true_class']== 0) & (errorDf['reconstruction_error'] < 10)]
fraudErrorDf = errorDf[errorDf['true_class'] == 1]

fig, (ax1, ax2) = plt.subplots(2,1, sharex=True)
fig.suptitle('Normal error and Fraud error')

bins = 10

ax2.hist(normalErrorDf.reconstruction_error.values, bins = bins)
ax2.set_title('Normal')

ax1.hist(fraudErrorDf.reconstruction_error.values, bins = bins)
ax1.set_title('Fraud')

plt.show();

fpr, tpr, thresholds = roc_curve(errorDf.true_class, errorDf.reconstruction_error)
roc_auc = auc(fpr, tpr)

plt.title('ROC')
plt.ylim([0, 1.001])
plt.xlim([-0.001, 1])
plt.ylabel('TPR')
plt.xlabel('FPR')
plt.plot(fpr, tpr, label='AUC = %0.4f'% roc_auc)
plt.plot([0,1],[0,1],'r--')
plt.show();
```

**Precision Thresholding vs. Recall**
Let's now examine the trade-off between recall and precision.

```
precision, recall, th = precision_recall_curve(errorDf.true_class, errorDf.reconstruction_error)
plt.plot(recall, precision, 'b', label='Precision-Recall curve')
plt.title('Recall vs Precision')
plt.ylabel('Precision')
plt.xlabel('Recall')
plt.show()
```

In data science, precision and recall are a constant tradeoff, therefore you eventually have to define an arbitrary boundary or threshold. It is simply a business decision where this boundary will be drawn. In this situation, you are weighing the costs of either missing a fraudulent transaction or incorrectly labeling a legitimate transaction as fraudulent.

```
plt.plot(th, recall[1:], 'b', label='Threshold-Recall curve')
plt.xlabel('Reconstruction error')
plt.title('Recall for different threshold values')
plt.ylabel('Recall')
plt.show()

plt.plot(th, precision[1:], 'b', label='Threshold-Precision curve')
plt.xlabel('Threshold')
plt.ylabel('Precision')
plt.title('Precision for different threshold values')
plt.show()

threshold = 2.9
```

**Threshold Check vs. Reconstruction**

```
groups = errorDf.groupby('true_class')
f, ax = plt.subplots()

for name, group in groups:
   ax.plot(group.index, group.reconstruction_error, marker='+', ms=5, linestyle='',
          label= "Fraud" if name == 1 else "Normal")
ax.hlines(threshold, ax.get_xlim()[0], ax.get_xlim()[1], colors="r", zorder=100, label='Threshold')
ax.legend()
plt.title("Reconstruction Error for Normal and Fraud Classes")
plt.ylabel("Reconstruction Error")
plt.xlabel("Data Points")
plt.show();
```

We then examine a conventional confusion matrix for the 20% of the data that we had purposefully withheld from the testing set. Here, I particularly focus on the proportion of fraud cases that have been identified to false positives. If there are no business regulations or cost considerations that dominate that decision, a 1:10 ratio is a reasonably common benchmark.

*#Confusion Matrix*

```
y_pred = [1 if e > threshold else 0 for e in errorDf.reconstruction_error.values]
conf_matrix = confusion_matrix(errorDf.true_class, y_pred)

plt.figure(figsize=(10, 10))
sns.heatmap(conf_matrix,
            xticklabels=LABELS,
            yticklabels=LABELS,
            annot=True,
            fmt=".1f",
            linewidth=.5,
            cmap=sns.cubehelix_palette(as_cmap=True));
plt.title("CONFUSION MATRIX")
plt.ylabel('True Class')
plt.xlabel('Predicted Class')
plt.show()
```