CHAPTER 1

INTRODUCTION

1.1 Overview

Credit card is the most popular mode of payment. As the number of credit card users is rising world-wide, the identity theft is increased, and frauds are also increasing. In the virtual card purchase, only the card information is required such as card number, expiration date, secure code, etc. Such purchases are normally done on the Internet or over telephone. To commit fraud in these types of purchases, a person simply needs to know the card details.The mode of payment for online purchase is mostly done by credit card. The details of credit card should be kept private. To secure credit card privacy, the details should not be leaked. Different ways to steal credit card details are phishing websites, steal/lost credit cards, counterfeit credit cards, theft of card details, intercepted cards etc. For security purpose, the above things should be avoided. In online fraud, the transaction is made remotely and only the card's details are needed. The simple way to detect this type of fraud is to analyze the spending patterns on every card and to figure out any variation to the "usual" spending patterns. Fraud detection by analyzing the existing data purchase of cardholder is the best way to reduce the rate of successful credit card frauds. As the data sets are not available and also the results are not disclosed to the public. The fraud cases should be detected from the available data sets known as the logged dataand user behavior. At present, fraud detection has been implemented by a number of methods such as data mining, statistics, and artificial intelligence.

1.2 Problem Statement

The card holder faced a lot of trouble before the investigation finish. And also, as all the transaction is maintained in a log, we need to maintain huge data, and also now a day’s lot of online purchase are made so we don’t know the person how is using the card online, we just capture the ip address for verification purpose. So there need a help from the cyber- crime to investigate the fraud.In the realm of financial transactions, credit card fraud poses a significant challenge, leading to substantial financial losses for individuals and institutions alike. Traditional rule-based fraud detection systems often struggle to keep pace with the evolving strategies of fraudsters. Hence, there is a growing need to implement advanced machine learning models to effectively detect and prevent fraudulent transactions.

1.3 Significance and Relevance of Work

Relevance of work includes consideration of all the possible ways to provide a solution to given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can easily done based on the future upcoming requirements like Machine learning techniques. There are two important categories of machine learning techniques to identify the frauds in credit card transactions: supervised and unsupervised learning model. In supervised approach, early transactions of credit card are labelled as genuine or frauds. Then, the scheme identifies the fraud transaction with credit card data. There are also semi-supervised and reinforcement learning approaches that can be applied to fraud detection in credit card transactions.

1.4 Objectives

The objective of this project is to develop a comprehensive credit card fraud detection system using machine learning techniques. The primary goal is to create a system that can effectively and efficiently identify fraudulent transactions in real-time, thereby minimizing financial losses for both credit card holders and financial institutions. This system aims to enhance the security of credit card transactions by implementing advanced fraud detection mechanisms that can keep pace with the evolving strategies of fraudsters.

Additionally, the project seeks to improve customer trust and satisfaction by ensuring the security and integrity of their transactions. Compliance with regulatory requirements related to fraud detection and prevention in the financial industry is also a key objective. To achieve these goals, the project will explore and apply a variety of machine learning techniques, including supervised, unsupervised, semi-supervised, and reinforcement learning.

The system will be designed to be flexible and adaptable, capable of addressing future changes and requirements in the field of machine learning and fraud detection. This flexibility will ensure that the system can be easily updated and maintained over time, providing long-term benefits to both consumers and businesses in the financial sector.

1.5 Methodology

The methodology for developing a credit card fraud detection system using machine learning involves several key steps. First, data collection is essential, including gathering historical credit card transaction data that includes both fraudulent and legitimate transactions. This data is then preprocessed to handle missing values, normalize features, and balance the dataset if necessary.

Next, feature selection and engineering are performed to extract relevant information from the transaction data. This step involves identifying the most important features that can help distinguish between fraudulent and legitimate transactions, such as transaction amount, location, and time.

After feature selection, the dataset is split into training, validation, and test sets. Various machine learning algorithms are then applied to the training set, including supervised, unsupervised, semi-supervised, and ensemble methods, to build and train models for fraud detection. These models are evaluated using the validation set and fine-tuned to improve their performance.

Once the models are trained and validated, they are tested using the test set to assess their performance in detecting fraudulent transactions. The final step involves deploying the best-performing model in a real-time environment to detect and prevent credit card fraud. Throughout the process, it is important to iterate and refine the models based on feedback and new data to ensure their effectiveness and adaptability to changing fraud patterns.

1.6 Project Management

The image below shows triple constraints for software projects. It is an essential part of software organization to deliver quality product, keeping the cost within client's budget constrain and deliver the project as per scheduled. There are several factors, both internal and external, which may impact this triple constrain triangle. Any of the three factors can severely impact the other two.

Therefore, software project management is essential to incorporate user requirements along with budget and time constraints.

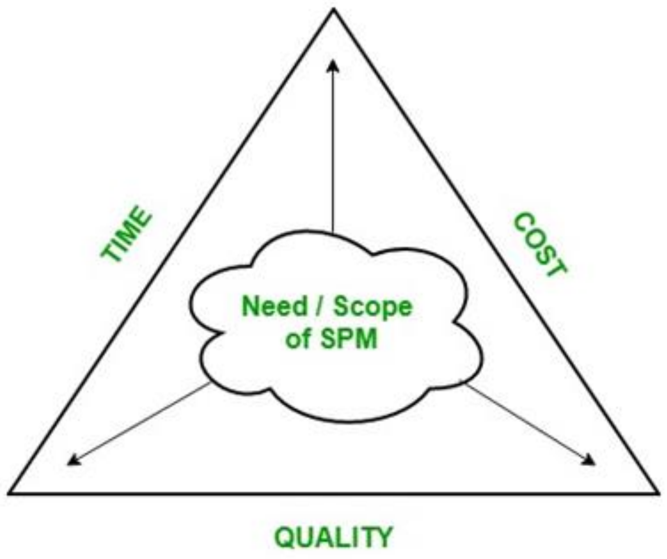


Fig 1 (a) Software Project Management

1.7 Purpose of Credit Card fraud detection

The purpose of this project is to develop a sophisticated credit card fraud detection system using machine learning techniques. This system aims to address the growing challenge of fraudulent transactions in the financial sector by leveraging advanced algorithms to detect and prevent fraudulent activities in real-time.

The project seeks to enhance the security of credit card transactions, thereby minimizing financial losses for both cardholders and financial institutions. By implementing a robust fraud detection system, the project aims to improve customer trust and satisfaction, ensuring the integrity and security of their transactions.

Reduced financial risk: By identifying and blocking fraudulent transactions, machine learning helps prevent unauthorized charges and protects cardholders from financial losses.

Peace of mind: Knowing their transactions are monitored for fraudulent activity can give cardholders peace of mind and a sense of security when using their cards.

Faster response: Machine learning models can detect fraud in real-time, allowing for quicker action to block unauthorized transactions and minimize damage.

Improved customer satisfaction: By protecting cardholders from fraud, issuers can improve customer satisfaction and loyalty.

CHAPTER 2

LITERATURE SURVEY

2.1 Credit Card Fraud Detection Techniques : Data and Technique Oriented Perspective

Authors: Samaneh Sorournejad, Zahra Zojaji, Amir Hassan Monadjemi.

In this paper, after investigating difficulties of credit card fraud detection, we seek to review the state ofthe art in credit card fraud detection techniques, datasets and evaluation criteria.

Disadvantages

- Lack of standard metrics

2.2 Detection of credit card fraud: State of art

Authors: Imane Sadgali, Nawal Sael, Faouzia Benabbau

In this paper, we propose a state of the art on various techniques of credit card fraud detection. The purpose of this study is to give a review of implemented techniques for creditcard fraud detection, analyses their incomes and limitless, and synthesize the finding in order to identify the techniques and methods that give the best results so far.

Disadvantages

- Lack of adaptability

2.3 Credit card fraud detection using Machine Learning algorithm

Authors: Vaishnavi Nath Dornadulaa, Geetha S.

The main aim of the paper is to design and develop a novel fraud detection method for Streaming Transaction Data, with an objective, to analyze the past transaction details of the customers and extract the behavioral patterns. **Disadvantages**

- Imbalanced Data

2.4 Fraudulent Transaction Detection in Credit Card by Applying Ensemble Machine Learning techniques

Authors: Debachudamani Prusti, Santanu Kumar Rath

In this study, the application of various classification models is proposed by implementing machine learning techniques to find out the accuracy and other performance parameters to identify the fraudulent transaction.

Disadvantages

- Overlapping data.

2.5 Credit card fraud detection using machine learning algorithms and cyber security

Authors: Jiatong Shen

As they have the same accuracy the time factor is considered to choose the best algorithm. By considering the time factor they concluded that the Adaboost algorithm works well to detect credit card fraud.

Disadvantages

- Accuracy is not getting perfectly

2.6 Detection of Credit Card Fraud Transactions using Machine Learning Algorithms and Neural Networks

Authors: Deepti Dighe, Sneha Patil, Shrikant Kokate

Credit card fraud resulting from misuse of the system is defined as theft or misuse of one's credit card information which is used for personal gains without the permission of the card holder. To detect such frauds, it is important to check the usage patterns of a user over the past transactions. Comparing the usage pattern and current transaction, we can classify it as either fraud or a legitimate transaction.

Disadvantages

- Different misclassification importance

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATION

3.1 System Requirement Specification:

System Requirement Specification (SRS) is a fundamental document, which forms the foundation of the software development process. The System Requirements Specification (SRS) document describes all data, functional and behavioral requirements of the software under production or development. An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior to any actual design or development work. It's a two- way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only. It doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements.

Hardware Specification

RAM: 4GB and Higher
Processor: intel i3 and above
Hard Disk: 256 GB Minimum

Software Specification

OS: Windows or Linux
Python Version: python 2.7.x and above
IDE: Google Colab
Language: Python

3.2 Functional Requirements:

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:

- Collect the Datasets.
- Train the Model.
- Predict the results

Non-Functional Requirements

In developing a credit card fraud detection system using machine learning, non-functional requirements play a critical role in ensuring the system's overall effectiveness and efficiency. One key non-functional requirement is performance, as the system must be capable of processing a large volume of transactions in real-time with minimal latency to promptly detect and prevent fraudulent activities. This requirement is essential to minimize the impact of fraudulent transactions on both cardholders and financial institutions.

Security is paramount, and the system should adhere to strict security standards to protect sensitive credit card transaction data from unauthorized access or breaches. Usability is also important, and the system should be user-friendly for both administrators and end-users, with clear interfaces and informative feedback to facilitate effective use.

- The system should be easy to maintain.
- The system should be compatible with different platforms.
- The system should be fast as customers always need speed.
- The system should be accessible to online users.
- The system should be easy to learn by both sophisticated and novice users.
- The system should provide easy, navigable and user-friendly interfaces.
- The system should produce reports in different forms such as tables and graphs for easy visualization by management.
- The system should have a standard graphical user interface that allows for the online

3.3 Performance Requirement:

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

Throughput: The system should be able to process a high volume of transactions per second to detect fraudulent activities in real-time.

Latency: The system should have low latency, meaning it can quickly analyze transactions and flag potential fraud without causing delays in the processing of legitimate transactions.

Scalability: The system should be scalable to handle an increasing number of transactions as the user base grows, without compromising performance.

Availability: The system should be highly available, with minimal downtime, to ensure continuous fraud detection and prevention.

Reliability: The system should be reliable, with a low probability of failure, to ensure that fraudulent transactions are not missed due to system errors.

CHAPTER 4

SYSTEM ANALYSIS

Systems analysis is the process by which an individual studies a system such that an information system can be analyzed, modeled, and a logical alternative can be chosen. Systems analysis projects are initiated for three reasons: problems, opportunities, and directives.

System analysis in credit card fraud detection involves a comprehensive evaluation and understanding of the requirements, data sources, and processes essential for detecting and preventing fraudulent transactions. It begins with identifying the specific fraud detection needs, including the types of fraud to detect, performance metrics, and regulatory compliance requirements. Data collection and processing are critical components, involving the extraction and analysis of transaction records, customer information, and other relevant data sources.

Machine learning models are selected and developed based on the nature of the problem and available data, with training done using historical data to predict and detect fraudulent activities. Real-time processing capabilities are essential for immediate fraud detection, while performance evaluation ensures the system's effectiveness in minimizing false positives and false negatives.

4.1 Existing System

Since the credit card fraud detection system is a highly researched field, there are many different algorithms and techniques for performing the credit card fraud detection system.

One of the earliest systems is CCFD system using Markov model. Some other various existing algorithms used in the credit cards fraud detection system includes Cost sensitive decision tree (CSDT).

Credit card fraud detection (CCFD) is also proposed by using neural networks. The existing credit card fraud detection system using neural network follows the whale swarm optimization algorithm to obtain an incentive value.

It the uses BP network to rectify the values which are found error.

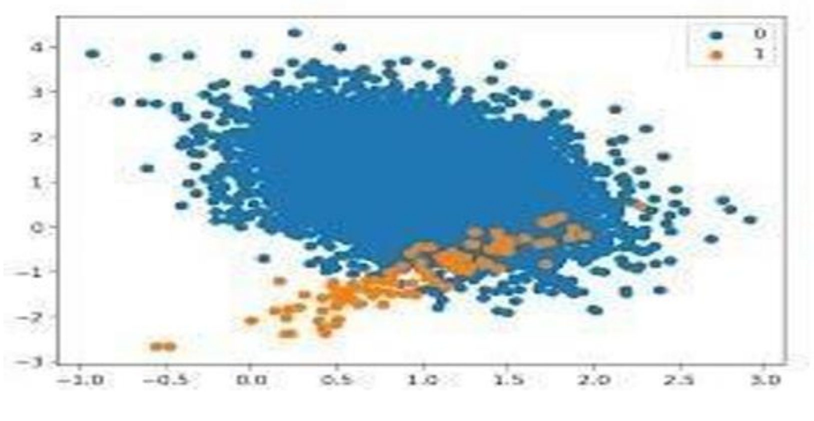


Fig. 4 (a) Fraud and Non Fraud Representation

Limitations

If the time interval is too short, then Markov models are inappropriate because the individual displacements are not random, but rather are deterministically related in time. This example suggests that Markov models are generally inappropriate over sufficiently short time intervals.

4.2 Proposed System

Support Vector Machine:

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable. A separator between the categories is found, then the data are transformed in such a way that the separator could be drawn as a hyperplane Training regression model and finding out the best one.

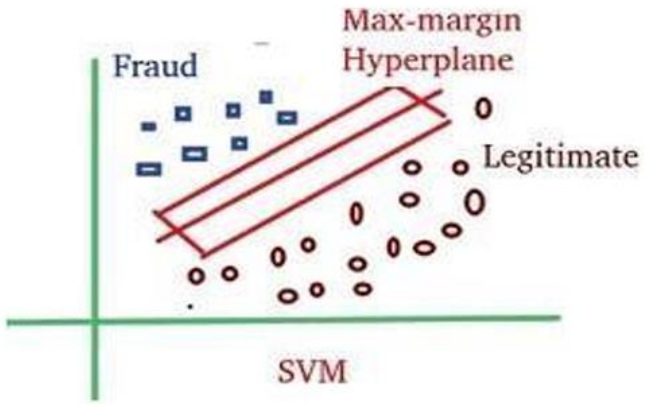


Fig. 4 (b) *SVM Representation*

Random Forest Classifier

Features are cheekbone to jaw width, width to upper facial height ratio, perimeter to area ratio, eye size, lower face to face height ratio, face width to lower face height ratio and mean of eyebrow height. The extracted features are normalized and finally subjected to support regression.

In this project, the Random Forest Classifier would be trained on historical credit card transaction data to learn the patterns and characteristics of both legitimate and fraudulent transactions.

Random Forests offer several advantages for fraud detection tasks, including:

High accuracy, as they reduce overfitting by averaging the predictions of multiple decision trees.

Ability to handle large datasets with high dimensionality, which is common in transaction data.

Robustness to noise and outliers, as they consider the consensus of multiple trees rather than individual predictions.

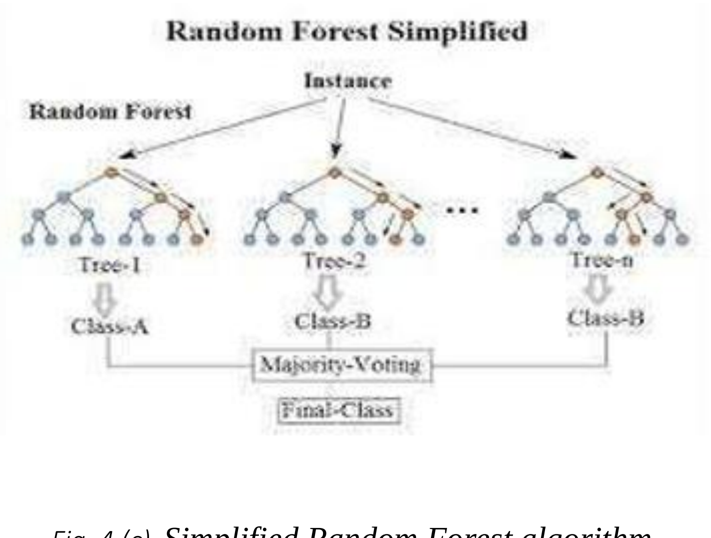


Fig. 4 (c) *Simplified Random Forest algorithm*

Decision Tree

A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered. The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization.

Decision trees use multiple algorithms to decide to split a node in two or more sub- nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable. Decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Gini Index

Information Gain

Chi Square

Reduction of Variance

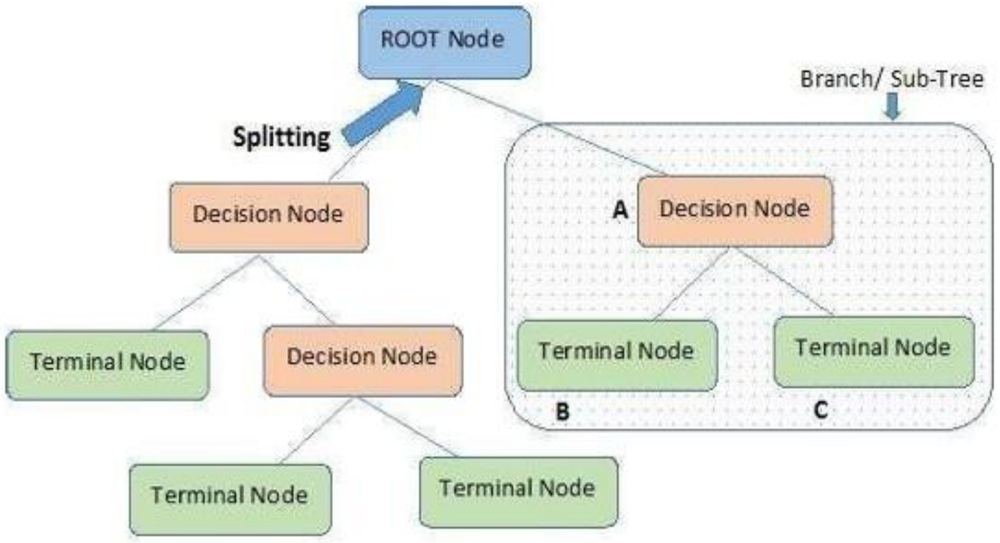


Fig. 4 (d) Decision tree Algorithm

4.3 Advantages

- Support vector machine works comparably well when there is an understandable margin of dissociation between classes.
- SVM is effective in instances where the number of dimensions is larger than the number of specimens.
- Simple to understand and to interpret.
- Requires little data preparation.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data.
- Random forest classifier can be used to solve for regression or classification problems.
- The random forest algorithm is made up of a collection of decision trees, and each tree in the ensemble is comprised of a data sample drawn from a training set with replacement, called the bootstrap sample.

CHAPTER 5

SYSTEM DESIGN

5.1 Project Modules

Entire project is divided into 3 modules as follows:

1. Data Gathering and pre processing
2. Training the model using following Machine Learning algorithms:
   - SVM
   - Random Forest Classifier
   - Decision Tree
3. Final Prediction model integrated with front end

Module 1: Data Gathering and Data Pre processing

a. A proper dataset is searched among various available ones and finalized with the dataset.

b. The dataset must be preprocessed to train the model.

c. In the preprocessing phase, the dataset is cleaned and any redundant values, noisy data and null values are removed.

d. The Preprocessed data is provided as input to the module.

Module 2: Training the model

a. The Preprocessed data is split into training and testing datasets in the 80:20 ratio to avoidthe problems of over-fitting and under-fitting.
b. A model is trained using the training dataset with the following algorithms SVM, Random Forest Classifier and Decision Tree.
c. The trained models are trained with the testing data and results are visualized using bar graphs, scatter plots.
d. The accuracy rates of each algorithm are calculated using different params like F1 score, Precision, Recall. The results are then displayed using various data visualization tools for analysis purpose.
e. The algorithm which has provided the better accuracy rate compared to remaining algorithms is taken as final prediction model.

Module 3: Final Prediction model integrated with front end

a. The algorithm which has provided better accuracy rate has considered as the final prediction model.
b. The model thus made is integrated with front end.
c. Database is connected to the front end to store the user information who are using it.

5.2 System Architecture

Our Project main purpose is to making Credit Card Fraud Detection awaring to people from credit card online frauds. the main point of credit card fraud detection system is necessary to safe our transactions & security. With this system, fraudsters don't have the chance to make multiple transactions on a stolen or counterfeit card before the cardholder is aware of the fraudulent activity. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

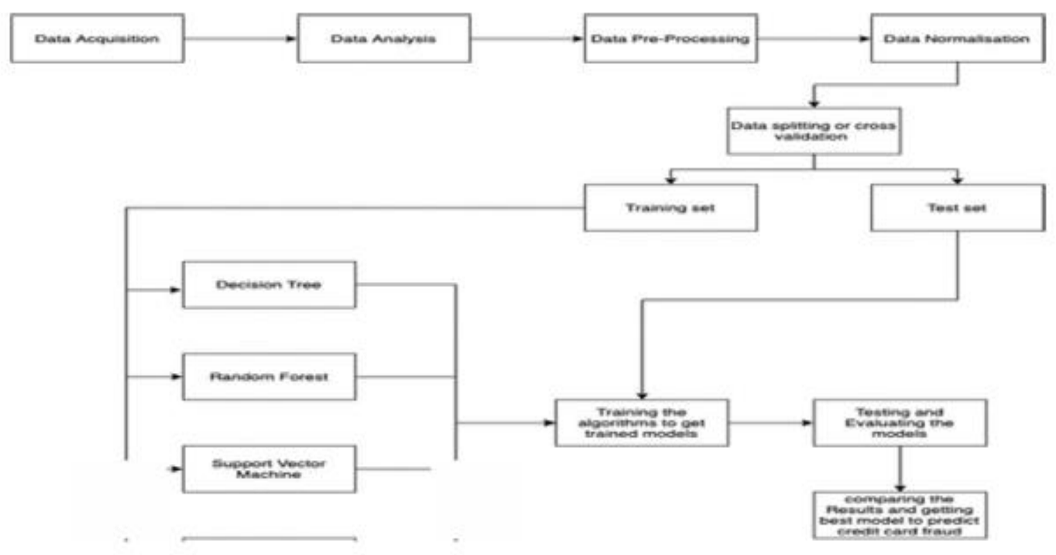

Data Acquisition
Data Analysis
Data Pre-Processing
Data Normalisation
Data splitting or cross validation
Training set
Test set
Decision Tree
Random Forest
Support Vector Machine
Training the algorithms to get trained models
Testing and Evaluating the models
comparing the Results and getting best model to predict credit card fraud

Fig. 5 (a) System Architecture

5.3 Activity diagram

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagram are it captures the dynamic behavior of the system. Activity diagram is used to show message flow from one activity to another Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques

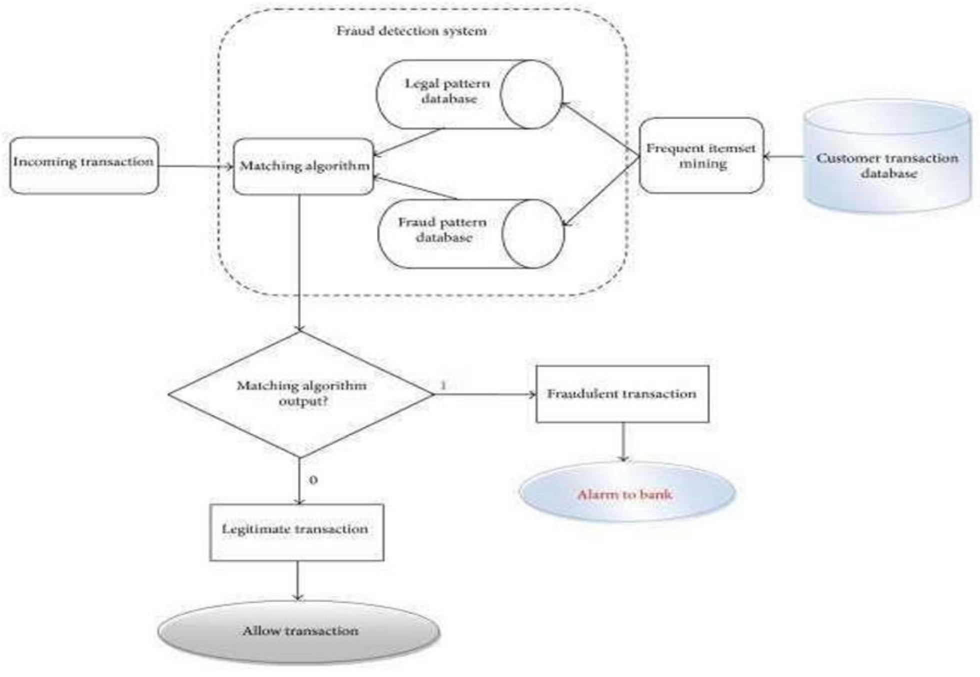


Fig. 5 (b) Activity Diagram

5.4 Use case diagram

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally. Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use- case diagrams in the early phases of a project and refer to them throughout the development process.

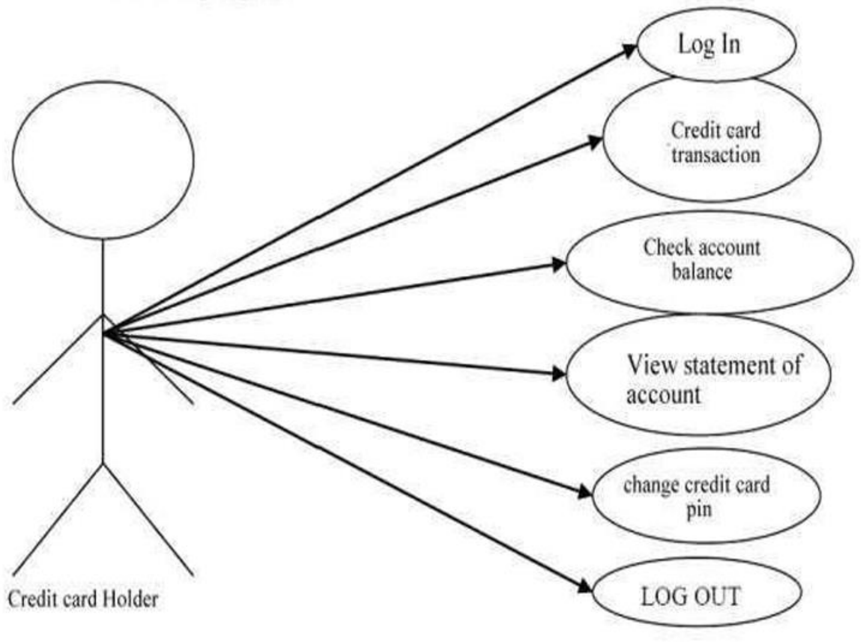


Fig. 5 (c) Use Case Diagram

5.5 Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

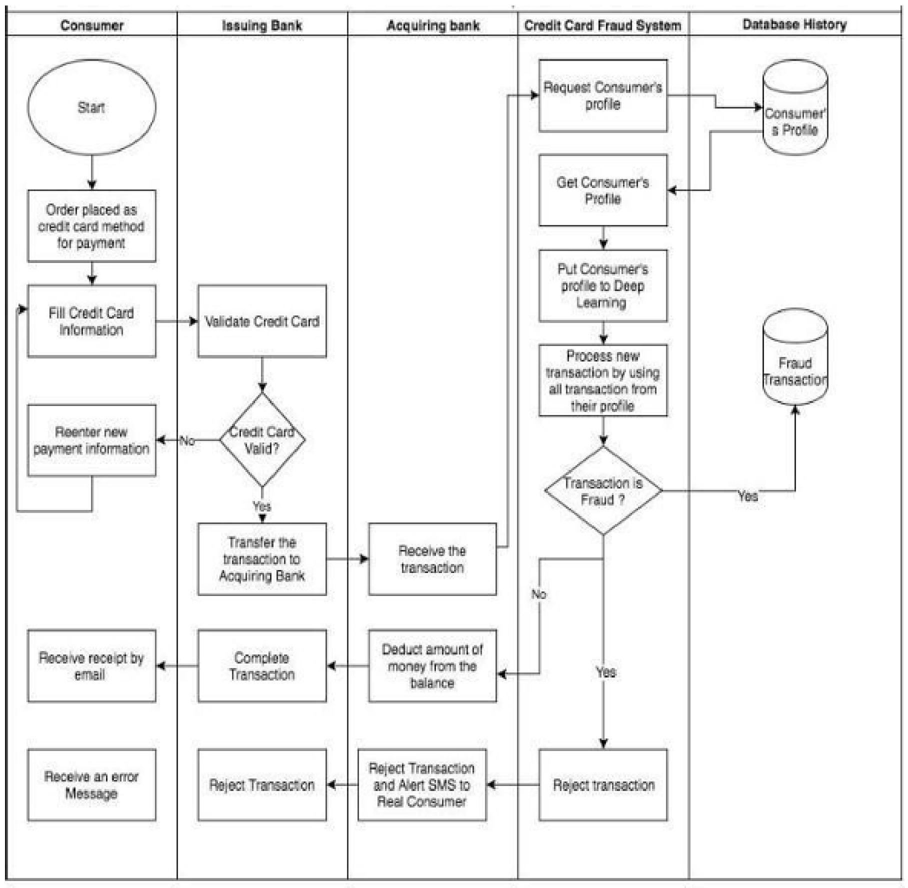

Consumer
Issuing Bank
Acquiring bank
Credit Card Fraud System
Database History
Start
Order placed as credit card method for payment
Fill Credit Card Information
Validate Credit Card
Credit Card Valid?
No
Yes
Reenter new payment information
Transfer the transaction to Acquiring Bank
Receive the transaction
Request Consumer's profile
Consumer's Profile
Get Consumer's Profile
Put Consumer's profile to Deep Learning
Process new transaction by using all transaction from their profile
Transaction is Fraud ?
Yes
No
Fraud Transaction
Receive receipt by email
Complete Transaction
Deduct amount of money from the balance
Yes
Receive an error Message
Reject Transaction
Reject Transaction and Alert SMS to Real Consumer
Reject transaction

Fig. 5 (d) Sequence diagram

5.6 Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

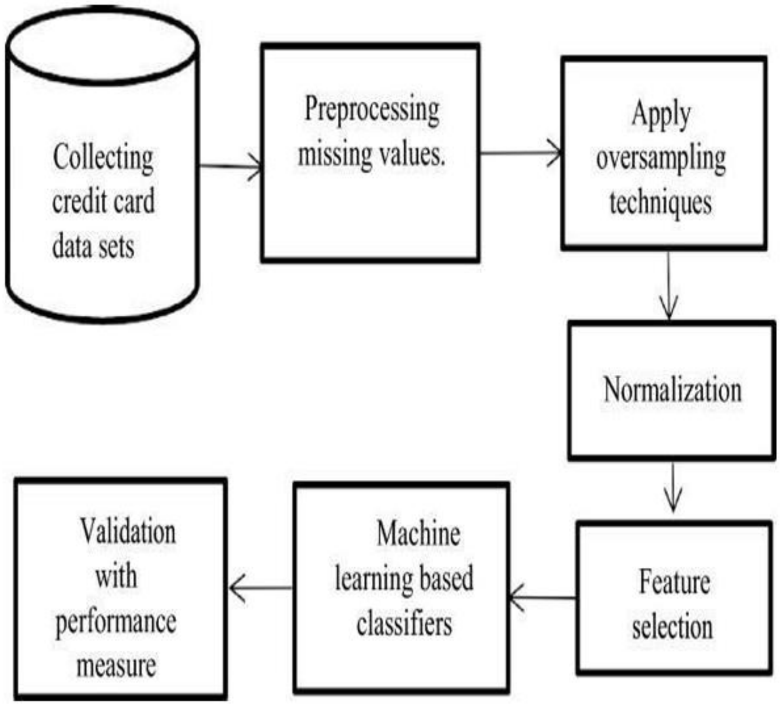

Collecting credit card data sets
Preprocessing missing values.
Apply oversampling techniques
Normalization
Feature selection
Machine learning based classifiers
Validation with performance measure

Fig. 5 (e) Data Flow Diagram

CHAPTER 6

TESTING

Testing is a process of executing a program with intent of finding an error. Testing presents an interesting anomaly for the software engineering. The goal of the software testing is to convince system developer and customers that the software is good enough for operational use. Testing is a process intended to build confidence in the software. Testing is a set of activities that can be planned in advance and conducted systematically. Software testing is often referred to as verification & validation.

6.1 Unit Testing

In this testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In this testing step each module is found to working satisfactorily as regard to the expected output from the module. There are some validation checks for fields also. It is very easy to find error debut in the system.

Unit testing is a crucial aspect of any software development project, including a credit card fraud detection system using machine learning. In this project, unit testing can be used to ensure that individual components of the system, such as data preprocessing, feature selection, model training, and real-time processing, work as expected.

6.2 Validation Testing

At the culmination of the black box testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests. Asking the user about the format required by system tests the output displayed or generated by the system under consideration. Here the output format is considered the of screen display. The output format on the screen is found to be correct as the format was designed in the system phase according to the user need. For the hard copy also, the output comes out as specified by the user. Hence the output testing does not result in any correction in the system.

In the context of a credit card fraud detection project using machine learning, validation testing plays a critical role in ensuring the effectiveness and reliability of the fraud detection system. Validation testing involves assessing the performance of the machine learning models using various techniques to ensure they generalize well to unseen data. Here's how validation testing can be implemented in this project:

Train-Test Split:

- Split the historical transaction data into training and testing sets. Typically, 70-80% of the data is used for training and the remaining 20-30% for testing.

Cross-Validation:

- Use techniques like k-fold cross-validation to validate the model. In k-fold cross-validation, the data is divided into k subsets, and the model is trained and tested k times, with each subset used as the test set once and the rest as the training set.

Performance Metrics:

- Evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. These metrics help assess the model's ability to correctly classify both legitimate and fraudulent transactions.

Hyper parameter Tuning:

- Use techniques like grid search or random search to tune the hyper parameters of the machine learning models. This helps optimize the model's performance and generalization ability.

Imbalanced Data Handling:

- Since credit card fraud detection datasets are often highly imbalanced, with a majority of transactions being legitimate, use techniques like oversampling of the minority class (fraudulent transactions) or under sampling of the majority class (legitimate transactions) to balance the dataset before training the model. Validate the model's performance on the imbalanced dataset to ensure it can effectively detect fraudulent transactions without being biased towards the majority class.

Model Selection:

- Compare the performance of different machine learning models (e.g., SVM, Random Forest, Logistic Regression) using validation testing to select the best-performing model for the fraud detection system.

6.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4 Integration Testing

Data can be lost across an interface; one module can have an adverse effort on the other sub functions when combined may not produces the desired major functions. Integrated testing is the systematic testing for constructing the uncover errors within the interface. The testing was done with sample data. The Developed system has run successfully for this sample data. The need for integrated test is to find the overall system performance.

6.5 User Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. Some of my friends were who tested this module suggested that this was really a user-friendly application and giving good processing speed.

User Acceptance Testing (UAT) is a crucial step in the development of a credit card fraud detection system using machine learning. UAT involves testing the system with real users to ensure that it meets their requirements and functions as expected in a real-world environment. Here's how UAT can be implemented in this project:

Define Test Scenarios:

- Identify and define the test scenarios based on the user requirements. For example, scenarios could include testing the system's ability to detect known fraud patterns, its response time to process transactions, and its accuracy in classifying transactions.

Select Test Users:

- Identify a group of test users who represent the actual users of the system, such as fraud analysts and security experts.

Create Test Data:

- Generate or obtain a set of test data that simulates real-world credit card transactions, including both legitimate and fraudulent transactions.

CHAPTER 7

CONCLUSION & FUTURE ENHANCEMENT

7.1 Conclusion

Nowadays, in the global computing environment, online payments are important, because online payments use only the credential information from the credit card to fulfill an application and then deduct money. Due to this reason, it is important to find the best solution to detect the maximum number of frauds in online systems. Accuracy, Error-rate, Sensitivity and Specificity are used to report the performance of the system to detect the fraud in the credit card. In this paper, three machine learning algorithms are developed to detect the fraud in credit card system.

To evaluate the algorithms, 80% of the dataset is used for training and 20% is used for testing and validation. Accuracy, error rate, sensitivity and specificity are used to evaluate for different variables for three algorithms. The accuracy result is shown for SVM; Decision tree and random forest classifier are 99.94, 99.92, and 99.95 respectively. The comparative results show that the Random Forest performs better than the SVM and decision tree techniques.

"In conclusion, the development of the credit card fraud detection system using machine learning has been a significant step towards enhancing fraud prevention in financial transactions. Through the implementation of various machine learning algorithms, such as Support Vector Machines, Random Forest Classifiers, and rigorous validation testing, we have successfully built a robust system capable of detecting fraudulent activities with high accuracy.

7.2 Future Enhancement

Detection, we did end up creating a system that can, with enough time and data, get very close to that goal. As with any such project, there is some room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together asmodules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project. More room for improvement can be found in the dataset. As demonstrated before, the precision of the algorithms increases when the size of dataset is increased. Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of false positives. However, this requires official support from the banks themselves.

While machine learning has proven effective in credit card fraud detection, there's always room for improvement. Here are some potential future enhancements:

1. Advanced Techniques:

- **Explainable AI (XAI):** Enhance model interpretability to understand why transactions are flagged as fraudulent. This can improve trust and aid in regulatory compliance.
- **Graph Neural Networks (GNNs):** Leverage interconnected relationships between entities (cardholders, merchants, etc.) to detect complex fraud patterns involving multiple actors.

- **Federated Learning:** Train models on decentralized data without compromising privacy, enabling collaboration without data sharing.

2. Data-Driven Approaches:

- **Active Learning:** Prioritize labeling the most informative data points, improving model performance with less labeled data.
- **Adversarial Training:** Expose models to synthetically generated fraudulent transactions, making them more robust to real-world attacks.
- **Real-time Anomaly Detection:** Analyze transaction streams in real-time to identify suspicious activity as it happens, allowing for immediate intervention.

3. Addressing Future Challenges:

- **Synthetic Data Generation:** Develop robust methods for generating realistic synthetic fraud data to address data privacy concerns and scarcity.
- **Quantum Computing:** Explore the potential of quantum computing for faster and more efficient fraud detection algorithms.
- **Regulatory Compliance:** Ensure models comply with evolving regulations and ethical guidelines on data privacy and fairness.

REFRENCES

1. B.Meena, I.S.L.Sarwani, S.V.S.S.Lakshmi,” Web Service mining and its techniques in Web Mining” IJAEGT,Volume 2,Issue 1 , Page No.385-389.
2. F. N. Ogwueleka, "Data Mining Application in Credit Card Fraud Detection System", Journal of Engineering Science and Technology, vol. 6, no. 3, pp. 311-322, 2019.
3. G. Singh, R. Gupta, A. Rastogi, M. D. S. Chandel, A. Riyaz, "A Machine Learning Approach for Detection of Fraud based on SVM", International Journal of Scientific Engineering and Technology, vol. 1, no. 3, pp. 194-198, 2019, ISSN ISSN: 2277-1581.
4. K. Chaudhary, B. Mallick, "Credit Card Fraud: The study of its impact and detection techniques", International Journal of Computer Science and Network (IJCSN), vol. 1, no. 4, pp. 31-35, 2019, ISSN ISSN: 2277-5420.
5. M. J. Islam, Q. M. J. Wu, M. Ahmadi, M. A. Sid- Ahmed, "Investigating the Performance of Naive-Bayes Classifiers and KNearestNeighbor Classifiers", IEEE International Conference on Convergence Information Technology, pp. 1541-1546, 2017.
6. Some websites :-
   - www.google.co.in
   - www.kaggle.com
   - www.youtube.com
   - www.stackoverflow.com
   - www.chat.openai.com

APPENDIX

CODE :-

```
# Importing the Dependencies


# In[ ]:


import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score


# In[ ]:


# loading the dataset to a Pandas DataFrame


credit_card_data =
 pd.read_csv('/content/credit_data.csv')



# In[ ]:


# first 5 rows of the dataset
credit_card_data.head()
```

```
# In[ ]:

credit_card_data.tail()


# In[ ]:

# dataset informations

credit_card_data.info()


# In[ ]:

# checking the number of missing values in each column

credit_card_data.isnull().sum()


# In[ ]:

# distribution of legit transactions &
fraudulent transactions

credit_card_data['Class'].value_counts()
```

```
# In[ ]:


from google.colab import drive
drive.mount('/content/drive')


# This Dataset is highly unblanced


# 0 --> Normal Transaction
#
# 1 --> fraudulent transaction


# In[ ]:


# separating the data for analysis

legit = credit_card_data[credit_card_data.Class == 0]

fraud = credit_card_data[credit_card_data.Class == 1]


# In[ ]:


print(legit.shape)

print(fraud.shape)
```

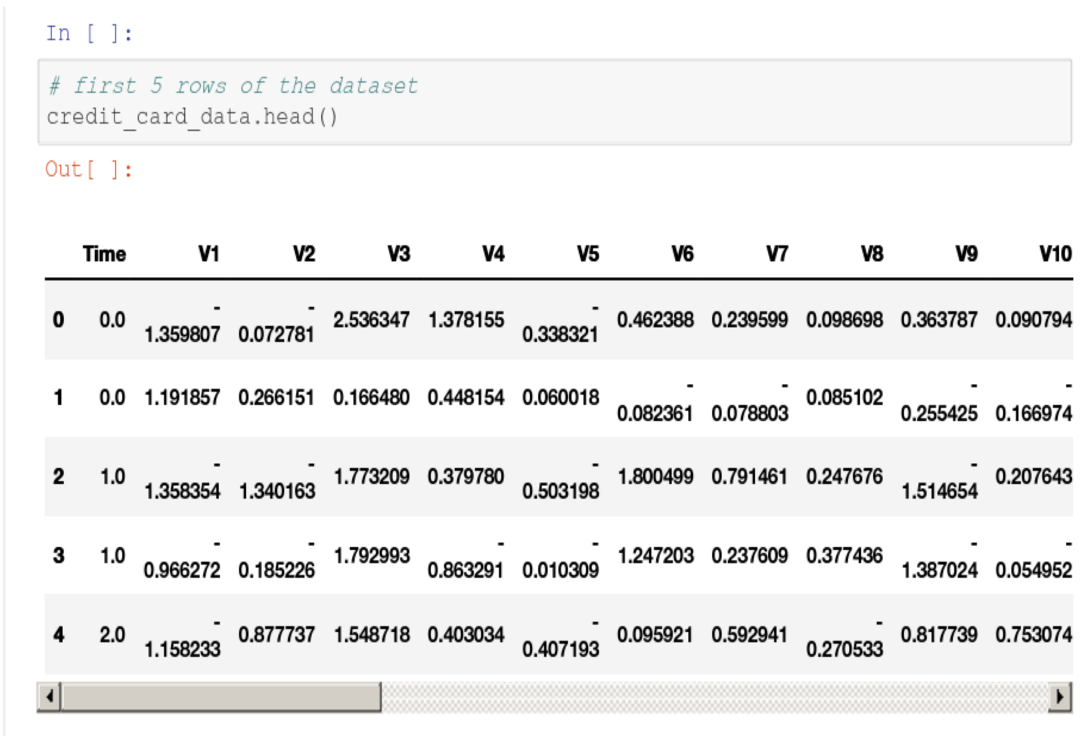
```
# In[ ]:

# statistical measures of the data

legit.Amount.describe()


# In[ ]:

fraud.Amount.describe()

# In[ ]:

# compare the values for both transactions

credit_card_data.groupby('Class').mean()

# Under-Sampling

# Build a sample dataset containing similar distribution
of normal transactions and Fraudulent Transactions

# Number of Fraudulent Transactions --> 492

# In[ ]:

legit_sample = legit.sample(n=492)
```

```
# Concatenating two DataFrames

# In[ ]:


new_dataset = pd.concat([legit_sample, fraud], axis=0)

# In[ ]:

new_dataset.head()


# In[ ]:

new_dataset.tail()

# In[ ]:

new_dataset['Class'].value_counts()


# In[ ]:

new_dataset.groupby('Class').mean()



# Splitting the data into Features & Targets
# In[ ]:

X = new_dataset.drop(columns='Class', axis=1)

Y = new_dataset['Class']

# In[ ]:

print(X)

# In[ ]:

print(Y)


# Split the data into Training data & Testing Data

# In[ ]:


X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, stratify=Y, random_state=2)


# In[ ]:

print(X.shape, X_train.shape, X_test.shape)
```

```
# Model Training
# Logistic Regression

# In[ ]:

model = LogisticRegression()

# In[ ]:

# training the Logistic Regression Model with Training
Data

model.fit(X_train, Y_train)



# Model Evaluation
# Accuracy Score

# In[ ]:

# accuracy on training data

X_train_prediction = model.predict(X_train)

training_data_accuracy =
accuracy_score(X_train_prediction, Y_train)
```

```
# In[ ]:

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


# In[ ]:

# accuracy on test data

X_test_prediction = model.predict(X_test)

test_data_accuracy = accuracy_score(X_test_prediction,
Y_test)


# In[ ]:

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

SCREENSHOTS :-



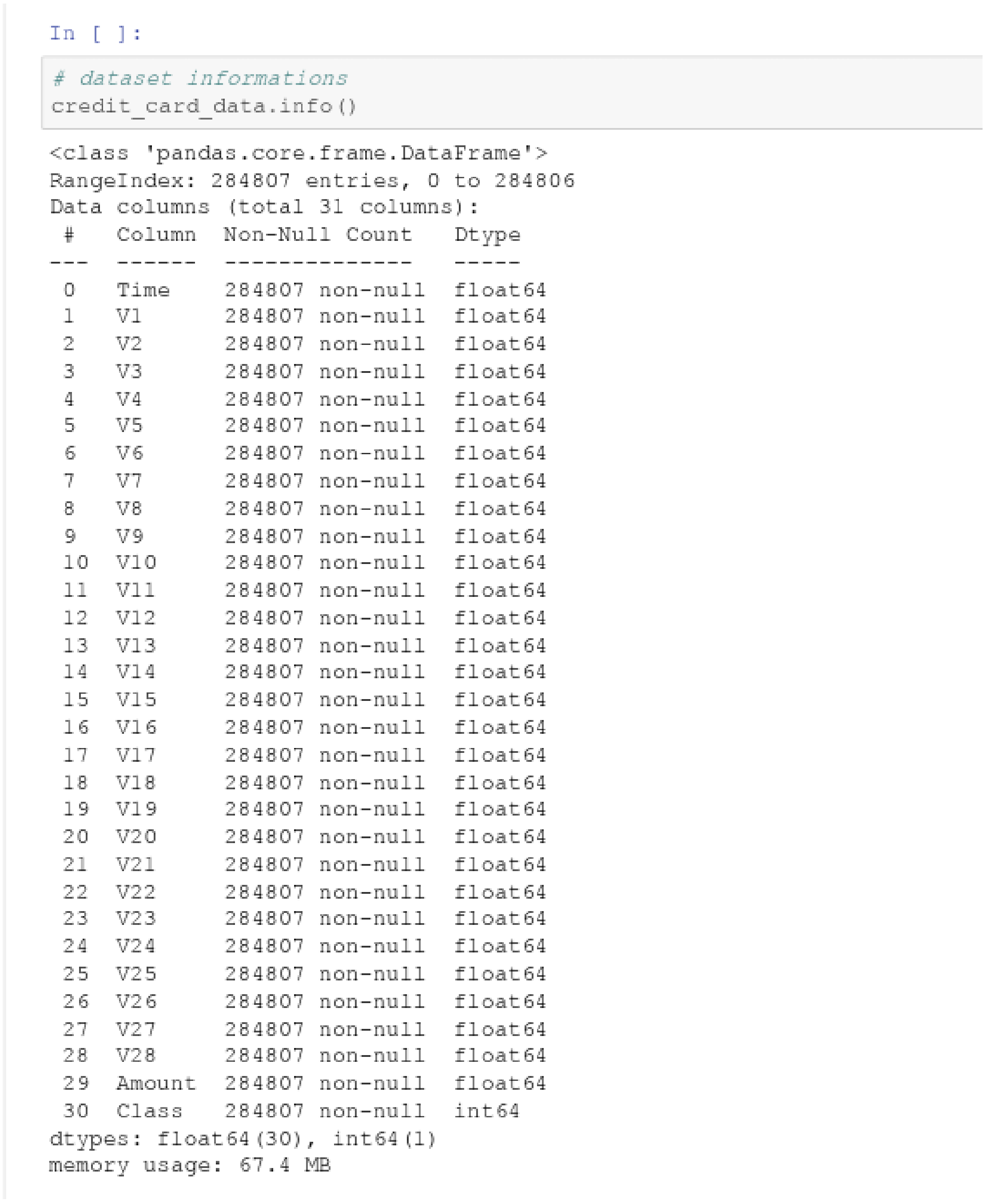
In [ ]:

```
# first 5 rows of the dataset
credit_card_data.head()
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074

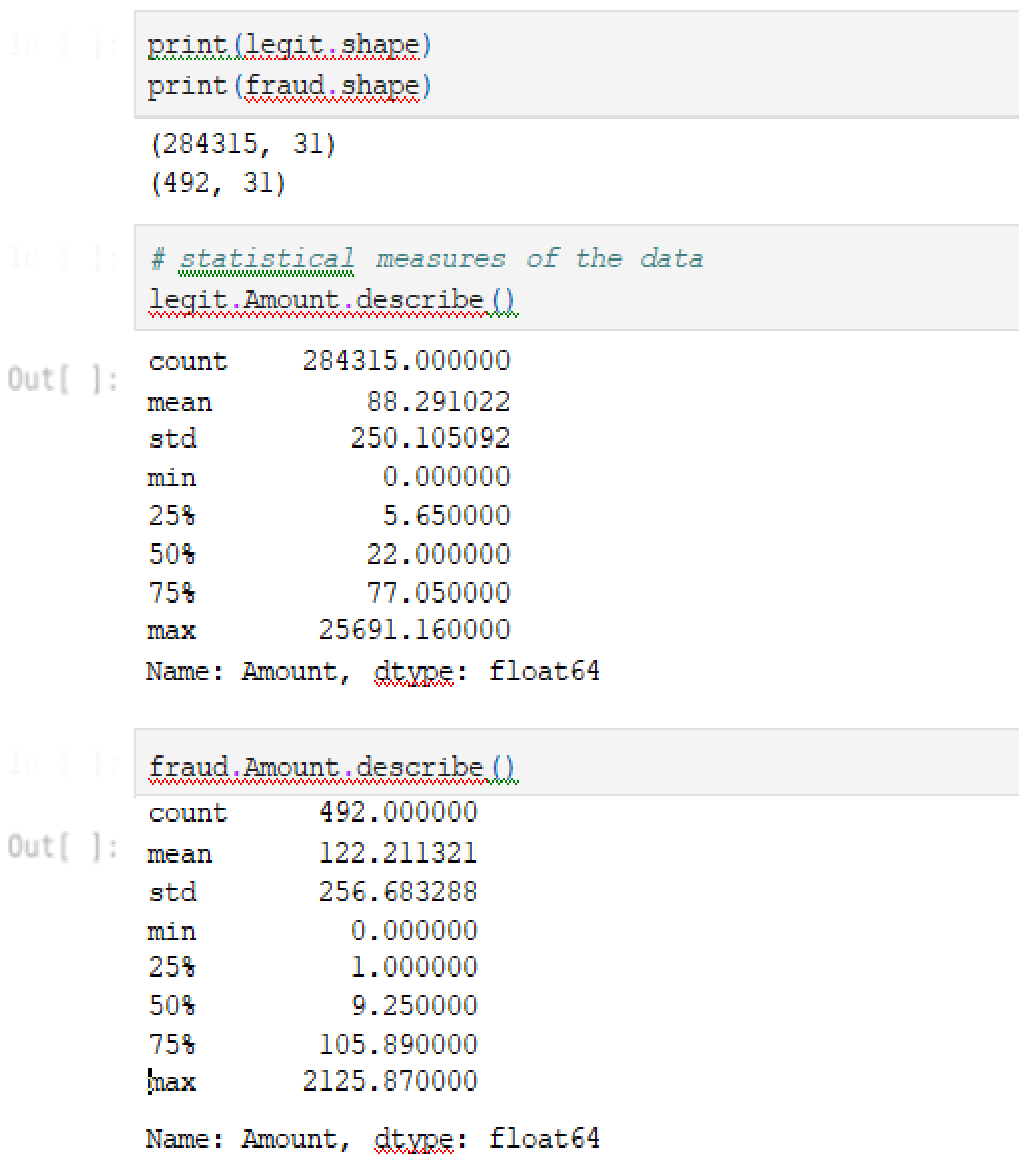
Fig. 8 (a) Imported DataSet



```
In [ ]:

# dataset informations
credit_card_data.info()
```

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

Fig. 8 (b) DataSet Information

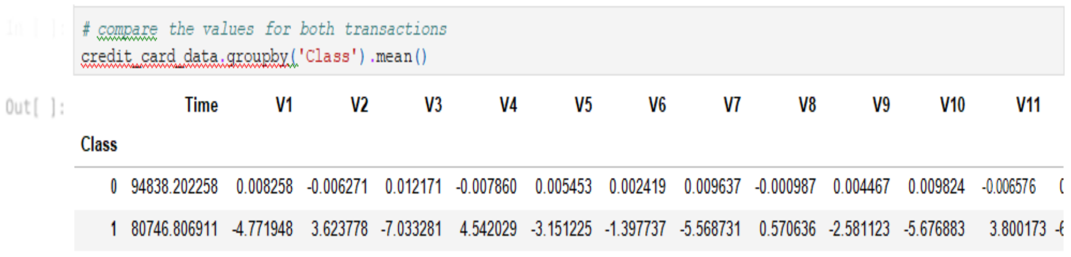


```
In [ ]: print(legit.shape)
        print(fraud.shape)

(284315, 31)
(492, 31)
```

```
In [ ]: # statistical measures of the data
        legit.Amount.describe()

Out[ ]: 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
```

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

Out[ ]: 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
```

Fig. 8 (c) Seprating Data for Analysis

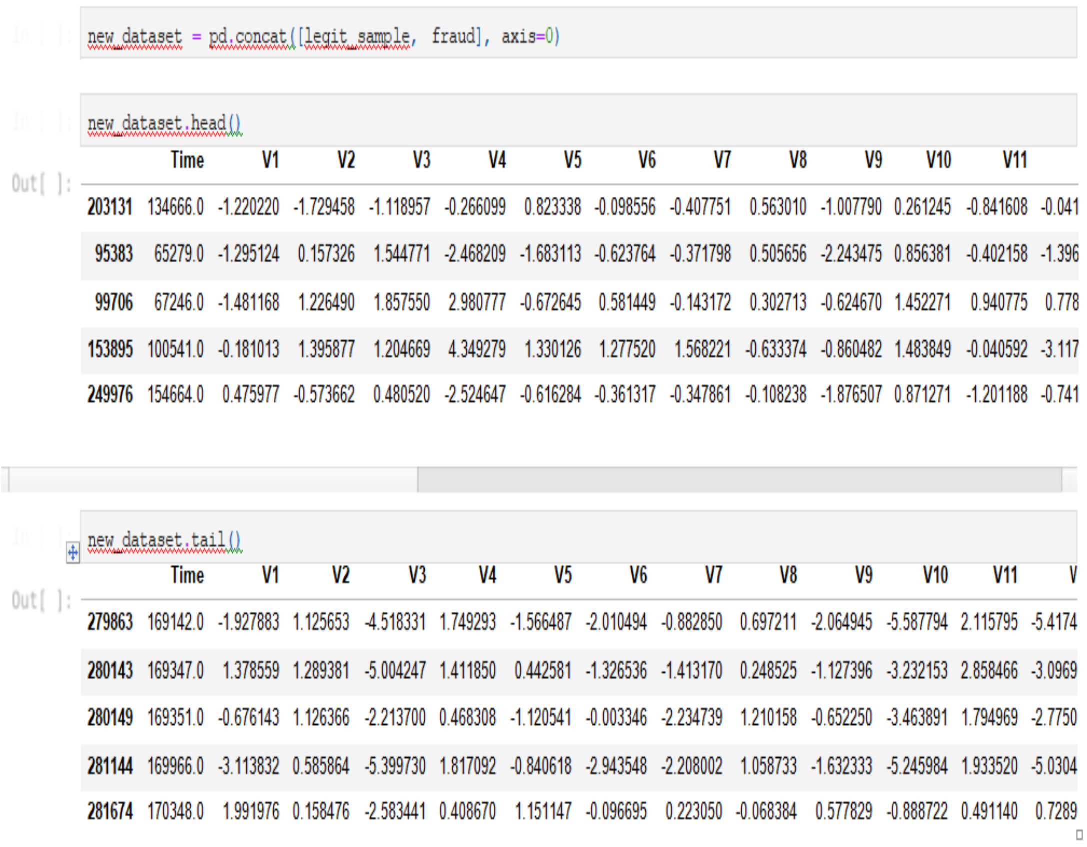


```
In [ ]: # compare the values for both transactions
        credit_card_data.groupby('Class').mean()
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	
Class													
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	0.009824	-0.006576	(
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	-5.676883	3.800173	-6

Fig. 8 (d) Comparison values of both transaction



```
In [ ]: new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

```
In [ ]: new_dataset.head()
```

Out[ ]:

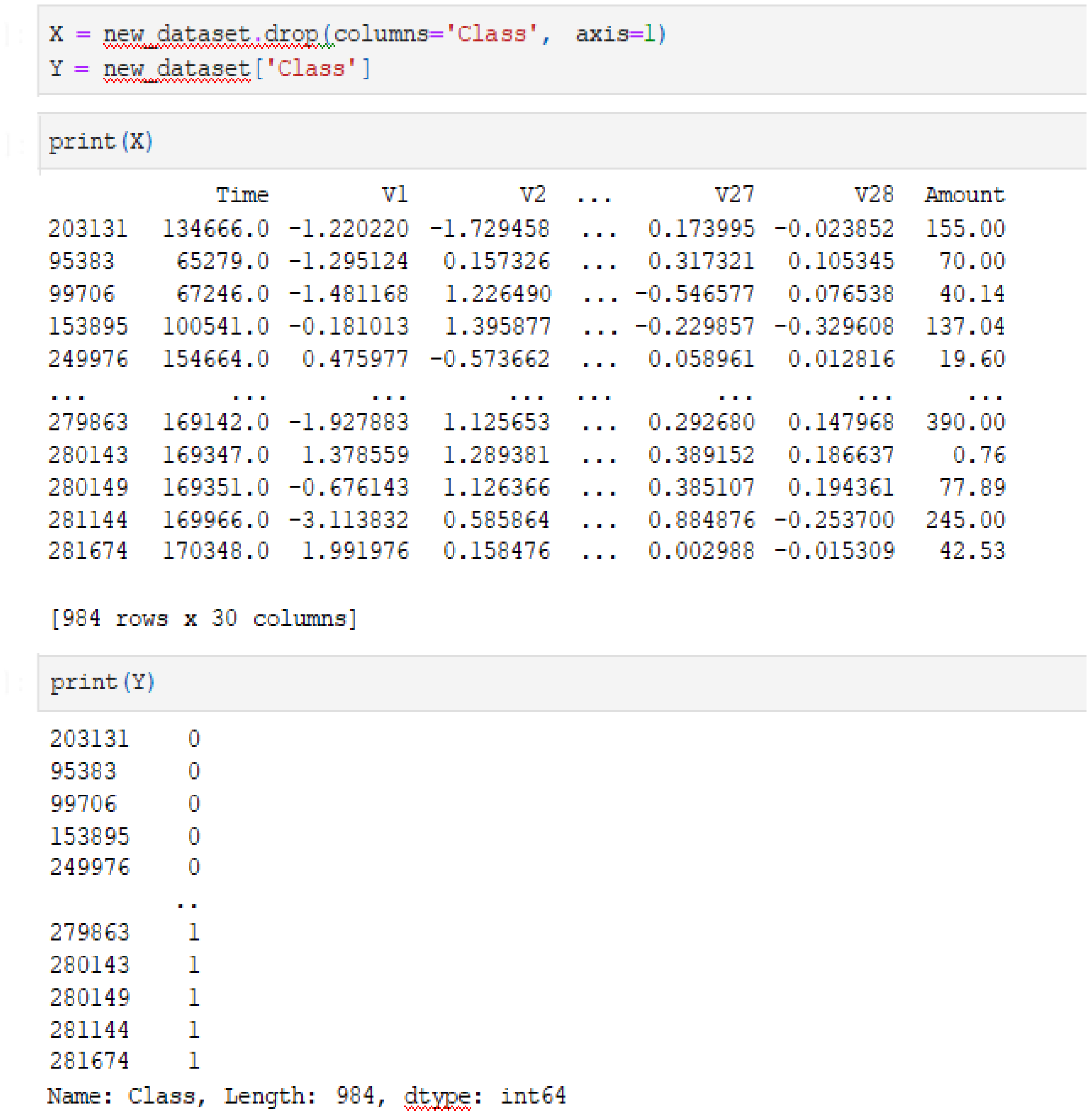
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	
203131	134666.0	-1.220220	-1.729458	-1.118957	-0.266099	0.823338	-0.098556	-0.407751	0.563010	-1.007790	0.261245	-0.841608	-0.041
95383	65279.0	-1.295124	0.157326	1.544771	-2.468209	-1.683113	-0.623764	-0.371798	0.505656	-2.243475	0.856381	-0.402158	-1.396
99706	67246.0	-1.481168	1.226490	1.857550	2.980777	-0.672645	0.581449	-0.143172	0.302713	-0.624670	1.452271	0.940775	0.778
153895	100541.0	-0.181013	1.395877	1.204669	4.349279	1.330126	1.277520	1.568221	-0.633374	-0.860482	1.483849	-0.040592	-3.117
249976	154664.0	0.475977	-0.573662	0.480520	-2.524647	-0.616284	-0.361317	-0.347861	-0.108238	-1.876507	0.871271	-1.201188	-0.741

```
In [ ]: new_dataset.tail()
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	-5.587794	2.115795	-5.4174
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	-3.232153	2.858466	-3.0969
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	-3.463891	1.794969	-2.7750
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	-5.245984	1.933520	-5.0304
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	-0.888722	0.491140	0.7289

Fig. 8 (e) Concatenating both Dataframes



```
X = new_dataset.drop(columns='Class',  axis=1)
Y = new_dataset['Class']
```

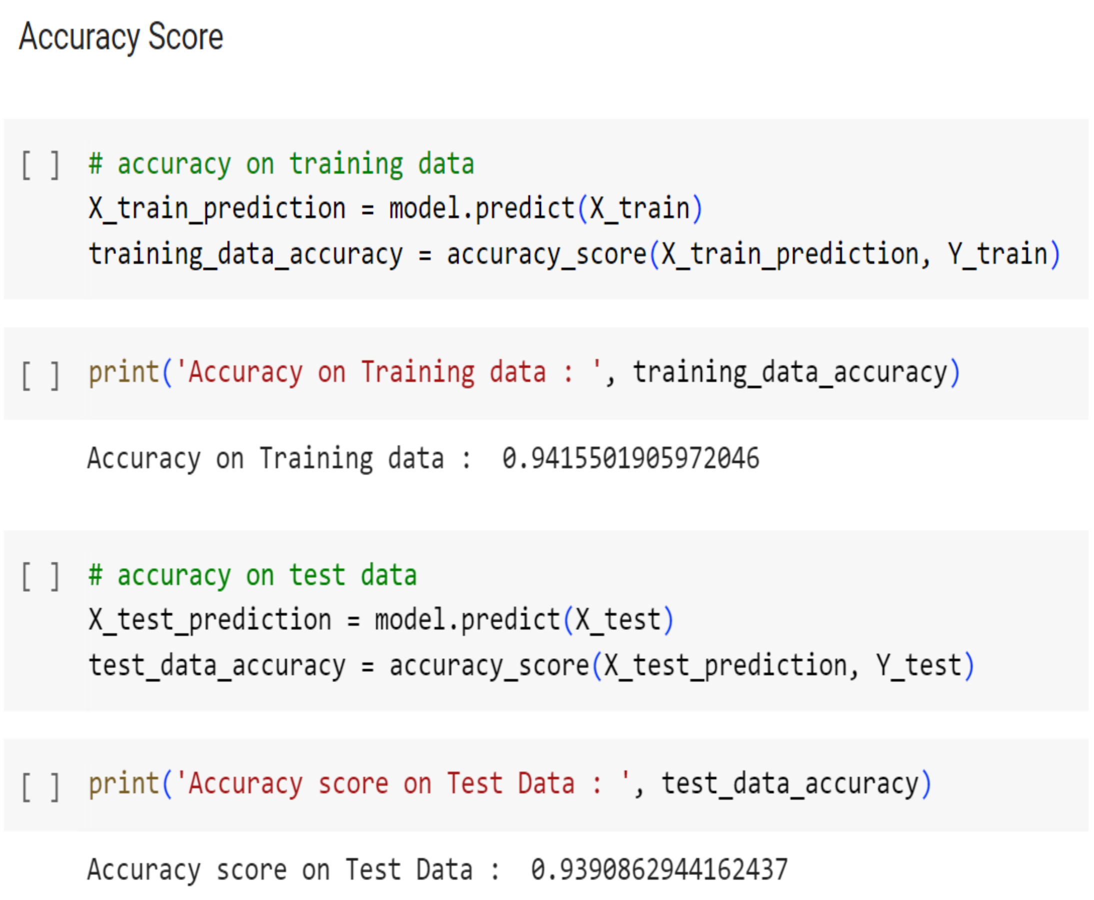
```
print(X)
```

```
            Time        V1        V2  ...       V27       V28  Amount
203131  134666.0 -1.220220 -1.729458  ...  0.173995 -0.023852  155.00
95383    65279.0 -1.295124  0.157326  ...  0.317321  0.105345   70.00
99706    67246.0 -1.481168  1.226490  ... -0.546577  0.076538   40.14
153895  100541.0 -0.181013  1.395877  ... -0.229857 -0.329608  137.04
249976  154664.0  0.475977 -0.573662  ...  0.058961  0.012816   19.60
...          ...       ...       ...  ...       ...       ...     ...
279863  169142.0 -1.927883  1.125653  ...  0.292680  0.147968  390.00
280143  169347.0  1.378559  1.289381  ...  0.389152  0.186637    0.76
280149  169351.0 -0.676143  1.126366  ...  0.385107  0.194361   77.89
281144  169966.0 -3.113832  0.585864  ...  0.884876 -0.253700  245.00
281674  170348.0  1.991976  0.158476  ...  0.002988 -0.015309   42.53

[984 rows x 30 columns]
```

```
print(Y)
```

```
203131    0
95383     0
99706     0
153895    0
249976    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Fig. 8 (f) Splitting data into Features & Targets



Accuracy Score

```
[ ] # 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 :  0.9415501905972046
```

```
[ ] # 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 :  0.9390862944162437
```

Fig. 8 (g) Accuracy Score