CREDIT CARD FRAUD DETECTION SYSTEM USING DEEP LEARNING

A Project report submitted in partial fulfilment of the requirements For the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

By
SK SHAMAN CHANDINI BEGUM
(319136410111)

Under the esteemed guidance of
Ms. Ch. Lavanya Ratna(PhD)
Asst. Professor
Department of Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Dr . L. BULLAYYA COLLEGE OF ENGINEERING
(Affiliated to Andhra University, Visakhapatnam)
New Resapuvanipalem, Visakhapatnam-530013

Year of submission : 2022

Dr. L. BULLAYYA COLLEGE OF ENGINEERING
New Resapuvanipalem, Visakhapatnam-530013
Department of Computer Science Engineering

Bonafide Certificate

This is to certify that **Ms.SK SHAMAN CHANDINI BEGUM** bearing register number **319136410111** student of Third year B. Tech in Computer Science Engineering, has carried out the project work titled **“Credit Card Fraud Detection Using Deep Learning”** at Dr. L. Bullayya College of Engineering, Visakhapatnam during the academic year **2021-22.**

Project Supervisor

Ms.Ch.Lavanya Ratna (PhD)

Asst Professor

Dept. of Computer Science Engineering
Engineering

Head of the Department

Dr. D. Madhavi

Professor

Dept. of Computer Science

ABSTRACT

Cybersecurity is becoming a crucial part of our life. The main challenge in Security in digital life is to find the abnormal activity. To make any transaction while purchasing any product online, credit cards are used frequently. The credit limit in credit cards sometimes helps in making purchases even if there is no sufficient amount at that time. But, on the other hand, these features are misused by cyber attackers. To tackle this problem, we need a system that can abort the transaction if it finds fraudulent.

This Project is focused on credit card fraud detection in real world scenarios. Nowadays credit card frauds are drastically increasing in number as compared to earlier times. Criminals are using fake identity and various technologies to trap the users and get the money out of them. Therefore, it is very essential to find a solution to these types of frauds. In this proposed project we designed a model to detect the fraud activity in credit card transactions. This system can provide most of the important features required to detect illegal and illicit transactions.

As technology changes constantly, it is becoming difficult to track the behaviour and pattern of criminal transactions. To come up with the solution one can make use of technologies with the increase of machine learning, artificial intelligence and other relevant fields of information technology, it becomes feasible to automate this process and to save some of the intensive amounts of labour that is put into detecting credit card fraud. Initially, we will collect the credit card usage data-set by users and classify it as trained and testing dataset using a random forest algorithm and decision trees. Using this feasible algorithm, we can analyse the larger data-set and user provided current data-set. Then augment the accuracy of the result data. Proceeded with the application of processing of some of the attributes provided which can find affected fraud detection in viewing the graphical model of data visualization. The performance of the techniques is gauged based on accuracy, sensitivity, and specificity, precision. The results is indicated concerning the best accuracy for Random Forest are unit 99.5% respectively.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Prof D. Deepak Chowdary,** Principal and to **Sri. S. M. Rahman**, Vice- Principal, Dr. L. Bullayya College of Engineering for giving us the opportunity to do this project.

We would like to thank **Dr. D. Madhavi,** HOD, Department of CSE, Dr. L. Bullayya College of Engineering, Visakhapatnam, for her guidance in producing this work.

We are deeply indebted to our project guide and the **Dr. Muddada Murali Krishna**, Asst Professor, head of research cluster **"Credit Card Fraud Detection Using Deep Learning"** and **Ms. Ch. Lavanya Ratna** Asst. Professor, Department of Computer Science Engineering, Dr. Lankapalli Bullayya College of Engineering, Visakhapatnam, for guiding us throughout the project in spite of her busy schedule.

Apart from the efforts of our, the success of this project depends largely on the encouragement of other faculty of CSE, Dr. L.Bullayya College of Engineering, Visakhapatnam. We take this opportunity to express my gratitude to the entire faculty who has been instrumental in the successful completion of this project.

Also deserving of thanks for our families and friends, for their support for their confidence in our achievements.

SK SHAMAN CHANDINI BEGUM

(319136410111)

DECLARATION

This is to declare that the Project work entitled “**Credit Card Fraud Detection** is a bonafide work done by us under the research cluster group “Deep Learning” with the esteemed guidance of Ms. Ch. Lavanya Ratna, Asst. Professor, Department of CSE , Dr.L. Bullayya College of Engineering. This project report is being submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science Engineering during the academic year 2021-2022

SK SHAMAN CHANDINI BEGUM

(319136410111)

Visakhapatnam

Date:

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER-1

INTRODUCTION

Nowadays Credit card usage has been drastically increased across the world. People believe in going cashless and are completely dependent on online transactions. The credit card has made the digital transaction easier and more accessible. A huge number of dollars of loss are caused every year by the criminal credit card transactions. Fraud is as old as mankind itself and can take an unlimited variety of different forms. The PwC global economic crime survey of 2017 suggests that approximately 48% of organizations experienced economic crime. Therefore, there’s positively a necessity to unravel the matter of credit card fraud detection.

Moreover, the growth of new technologies provides supplementary ways in which criminals may commit a scam. Huge Financial losses have been fraudulent effects on not only merchants and banks but also the individual person who are using the credit cards. Fraud may also affect the reputation and image of a merchant causing non-financial losses. For example, if a cardholder is a victim of fraud with a certain company, he may no longer trust their business and choose a competitor. Fraud Detection is the process of monitoring the transaction behavior of a cardholder to detect whether an incoming transaction is authentic and authorized or not otherwise it will be detected as illicit.

Background and Motivation

In this project we mainly focus on credit card fraud detection in real world. Fraud is a major problem for the whole credit card industry that grows bigger with the increasing popularity of electronic money transfers. To effectively prevent the criminal actions that lead to the leakage of bank account information leak, skimming, counterfeit credit cards, the theft of billions of dollars annually, and the loss of reputation and customer loyalty, credit card issuers should consider the implementation of advanced Credit Card Fraud Prevention and Fraud Detection methods.

Here the credit card fraud detection is based on fraudulent transactions. Generally credit card fraud activities can happen in both online and offline. But in today's world online fraud transaction activities are increasing day by day. So in order to find the online fraud transactions various methods have been used in existing system. In proposed system we are going to perform a comparative analysis based on tree based algorithms such as Random Forest Classifier, Isolation forest, Extra tree algorithm and decision trees and measure the accuracy obtained by all of these algorithms for the given dataset.

1.1 Problem Statement

We propose a Deep learning model to detect fraudulent credit card activities in online financial transactions. Analysing fake transactions manually is impracticable due to vast amounts of data and its complexity. This hypothesis will be explored in the project. In this proposed model, we are going to classify fraudulent and legitimate credit card transactions accurate result and to help us to get awareness about the of any financially sensitive data by comparing the accuracy obtained by tree based algorithm analysis such as Random forest Algorithm.

PURPOSE OF THE PROJECT

We propose a deep learning model to detect fraudulent credit card activities in online financial transactions. Analysing fake transactions manually is impracticable due to vast amounts of data and its complexity. However, adequately given informative features, could make it possible using Machine Learning. This hypothesis will be explored in the project. To classify fraudulent and legitimate credit card transactions by supervised learning algorithms such as Random Forest. To help us to get awareness about the fraudulent and without loss of any financially. data exploration, pro processing and for using random forest algorithm are:

· NumPy: For simple arrays.

· Pandas: For reading the file.

· Sci Kit: Learn- for pre-processing.

Packages Which are being used for

· Matplotlib or Seaborn: For plotting and representing confusion matrix colour format.

CHAPTER-2

REQUIREMENT ELICITATION AND ANALYSIS

To run an AI-driven strategy for Credit Card Fraud Analytics, a number of critical requirements should be met. These will ensure that the model reaches its best detection score.

Amount of data

Training high-quality Machine Learning models requires significant internal historical data. That means if you do not have enough previous fraudulent and normal transactions, it would be hard to run a Machine Learning model on it because the quality of its training process depends on the quality of the inputs. Because it is rarely the case that a training set contains an equal amount of data samples in two classes, dimensionality reduction or data augmentation techniques are used for that.

Quality of data

Models may be subject to bias based on the nature and quality of historical data. This statement means that if the platform maintainers did not collect and sort the data neatly and properly or even mixed the information of fraudulent transactions with the information of normal ones, that is likely to cause a major bias in the model’s results.

The integrity of factors

If you have enough data that is well-structured and unbiased, and if your business logic is paired nicely with the Machine Learning model, the chances are very high that fraud detection will work well for your customers and your business.

2.1 Existing System

In existing system, the methods to determine a credit card fraud such as cluster analysis, SVM, Bayesian network, Logistic Regression, Naïve bayer’s, Hidden Markov model are based on unsupervised learning and the accuracy obtained by these methods is about 60-70%.

2.2 Proposed System

In the Proposed System we are going to perform a comparative analysis based on tree-based algorithms such as Random Forest Classifier, Isolation Forest, and decision trees and measure the accuracy obtained by all of these algorithms for the given dataset.

2.3.1 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases.

SYSTEM REQUIREMENT SPECIFICATION

S.NO.	**Requirement**	**Requirement no.**	**Essential/Desirable**	**Description**
1	Loading of dataset	RS1	Essential	Data set should be loaded without any errors.
2	Data Cleaning	RS2	Essential	Data should be cleaned properly
3	Feature Selection	RS3	Essential	Feature selection is Performed using heatmap.
4	Splitting of dataset into train and test data	RS4	Essential	According to the given ratio the dataset should be split into train and test data.
5	RFA, and decision tree algorithms are called for given dataset	RS5	Essential	Each of the model applied should work without any errors
6	Analysis	RS6	Essential	Algorithms are compared.

2.3.2 Non-functional requirements

In addition to the obvious features and functions that you will provide in your system, there are other requirements that don't actually do anything, but are important characteristics, nevertheless. These are called "non-functional requirements" or sometimes "Quality Attributes." For example, attributes such as performance, security, usability, compatibility, aren't a "feature" of the system but, are a required characteristic. You can't write a specific line of code to implement them, rather they are "emergent" properties that arise from the entire solution. The specification needs to describe any such attributes the customer requires. You must decide the kind of requirements that apply to your project and include those that are appropriate.

Each requirement is simply stated in English. Each requirement must be objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.

Often deciding on quality attributes requires making trade-offs, e.g., between performance and maintainability. Here are some examples of non-functional requirements:

Requirements

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

Operating constraints

List any run-time constraints. This could include system resources, people, needed software.

Platform constraints

Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

Accuracy and Precision

Requirements about the accuracy and precision of the data. Beware of 100% requirements, they often cost too much.

Modifiability

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Portability

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Reliability

It is the ability of a system or components to perform its required function under stated conditions for a specified period of time.

Usability

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

Execution Qualities

Such as security and usability, which are observable at run time.

Evolution Qualities

such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

SYSTEM SPECIFICATIONS

Hardware Specifications

- Processor : Intel core i3 or above
- Hard disk: 1GB or above
- Memory : 1GB RAM

Software Specifications

- Operating System: Windows 7 or above
- Languages: Python
- Any Python IDE

CHAPTER-3

SYSTEM DESIGN

- System Design is a solution to how to approach to the creation of a system. This important phase provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. The design step produces a data design, an architectural design and a procedural design. The data design transforms the information domain model created during analysis in to the data structures that will be required to implement the software.
- The architectural design defines the relationship among major structural components into a procedural description of the software. Source code generated and testing is conducted to integrate and validate the software.
- From a project management point of view, software design is conducted in two steps. Preliminary design is connected with the transformation of requirements into data and software architecture. Detailed design focuses on refinements to the architectural representation that leads the detailed data structure and algorithmic representations of software.

3.1. Object Oriented Design and Analysis

- In the system analysis or object-oriented analysis phase of software development, the system requirements are determined, the classes are identified and the relationships among classes are identified.

- The three analysis techniques that are used in conjunction with each other for object-oriented analysis are object modelling, dynamic modelling, and functional modelling.

- Object modelling develops the static structure of the software system in terms of objects. It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class.
- After the static behaviour of the system is analysed, its behaviour with respect to time and external changes needs to be examined. This is the purpose of dynamic modelling. Dynamic Modelling can be defined as "a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world".

- Functional Modelling is the final component of object-oriented analysis. The functional model shows the processes that are performed within an object and how the data changes as it moves between methods. It specifies the meaning of the operations of object modelling and the actions of dynamic modelling. The functional model corresponds to the data flow diagram of traditional structured analysis.

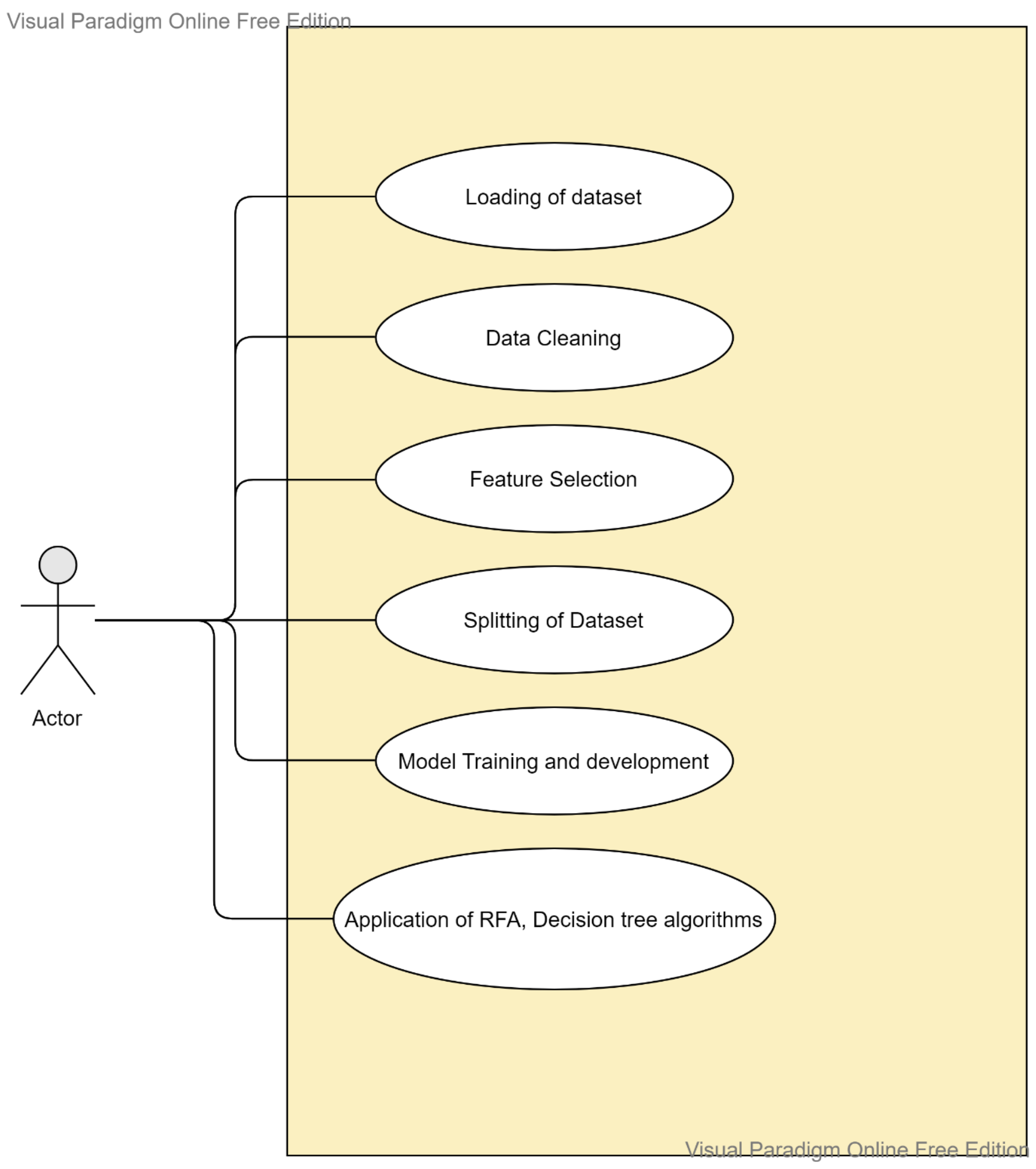
- After the analysis phase, the conceptual model is developed further into an object-oriented model using object-oriented design (OOD). In OOD, the technology-independent concepts in the analysis model are mapped onto implementing classes, constraints are identified, and interfaces are designed, resulting in a model for the solution domain. In a nutshell, a detailed description is constructed specifying how the system is to be built on concrete technologies.

3.1.1Use Case Diagram

A use case diagram is a Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. A use case is a set of scenarios that describes an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Use case diagrams are used to gather the requirements are mostly design requirements. So, when a system is analysed to gather its functionalities use cases are prepared and actors are identified.
So, in brief, the purpose of use case diagrams:

1) Used to gather requirements of a system.
2) Used to get an outside view of a system.
3) Identify external and internal factors influencing the system.
4) Show the interacting among the requirements are actors.


Loading of dataset
Data Cleaning
Feature Selection
Splitting of Dataset
Model Training and development
Application of RFA, Decision tree algorithms
Actor
Visual Paradigm Online Free Edition
Visual Paradigm Online Free Edition

3.1.2Class Diagram

Class diagrams are used to depict the system's structure. Classes are abstractions that define a group of objects' shared structure and behaviour. During the execution of the system, objects are instances of classes that are generated, changed, and destroyed. An object's state consists of the values of its properties as well as its relationships with other things.

The class diagram depicts a class's attributes and operations, as well as the system's limitations. Because class diagrams are the only UML diagrams that can be directly mapped with object-oriented languages, they are frequently utilised in the modelling of object-oriented systems. A collection of classes, interfaces, relationships, and collaborations are depicted in the class diagram. A structural diagram is another name for it.

Purpose

The class diagram is used to represent an application's static view. Class diagrams are the only diagrams that can be directly mapped with object-oriented languages, making them popular during construction.
The purpose of the class diagrams is:

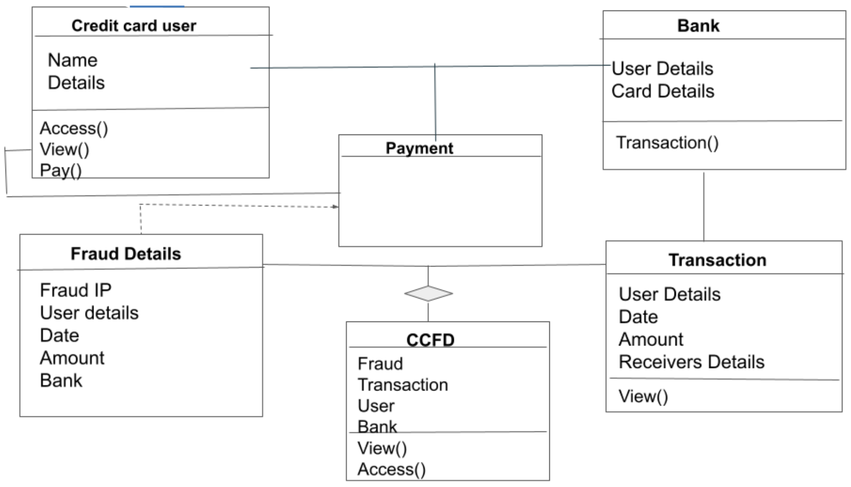
1) Analysis and design of an application's static view.

2) Describe a system's responsibilities.

3) Provides a foundation for component and deployment diagrams.

4) The class diagram is a fundamental component of object-oriented modelling. It is used for both overall conceptual modelling of the application's structure and specific modelling of the models' translation into programming code. Data modelling can also be done with class diagrams. In a class diagram, the classes reflect both the major features of the application, as well as the classes that will be programmed.

In the diagram, classes contain three compartments.
1) The name of the class is written on the top compartment. The initial letter is capitalised and displayed in bold and in the centre.
2) The class's characteristics are kept in the intermediate container. They're aligned to the left, and the first letter is in lowercase.
3) The operations that the class can perform are kept in the bottom container. They're also positioned to the left, and the initial letter is in lowercase.


Credit card user
Name
Details
Access()
View()
Pay()
Bank
User Details
Card Details
Transaction()
Payment
Fraud Details
Fraud IP
User details
Date
Amount
Bank
CCFD
Fraud
Transaction
User
Bank
View()
Access()
Transaction
User Details
Date
Amount
Receivers Details
View()

3.1.3State Chart Diagram

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

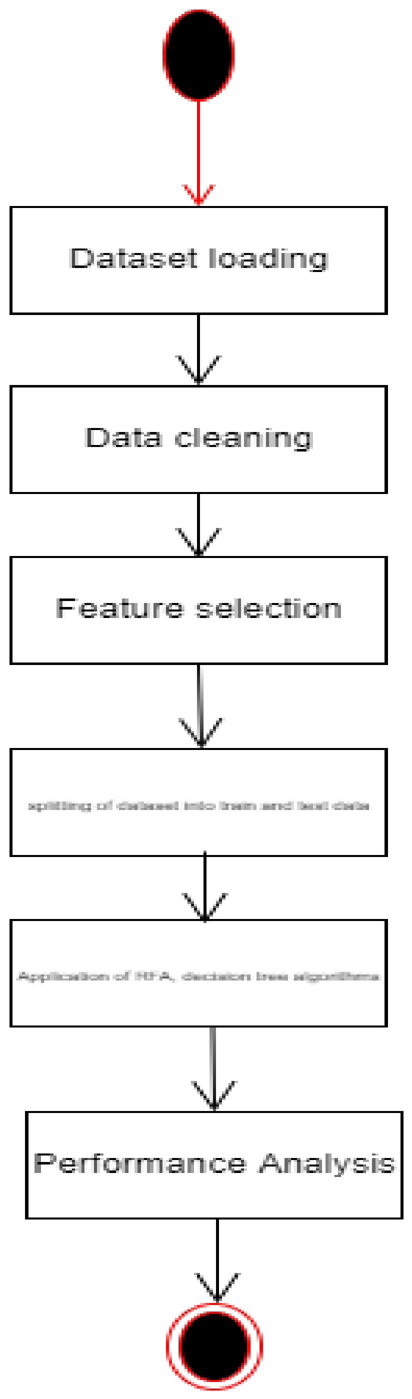
Purpose of state chart diagram

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using State chart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.


Dataset loading
Data cleaning
Feature selection
splitting of dataset into train and test data
Application of RFA, decision tree algorithms
Performance Analysis

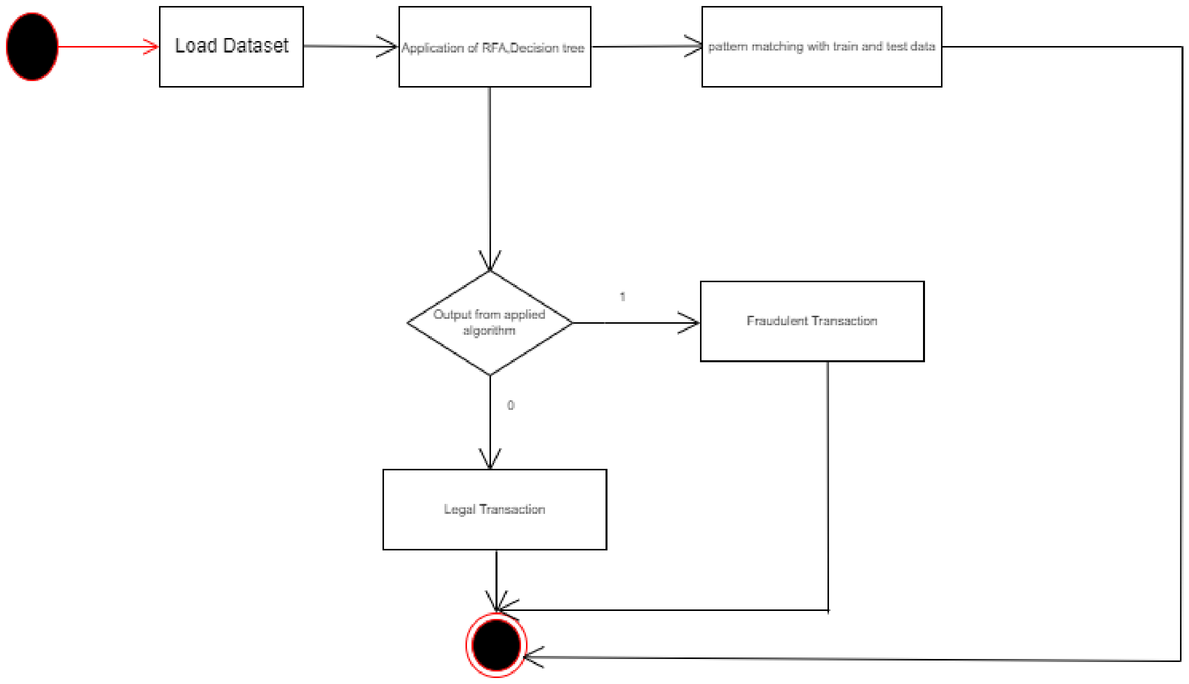
3.1.4Activity Diagram

An activity diagram describes the behaviour of a system in terms of activities. Activities are modelling elements that represent the execution of a set of operations. The completion of these operations triggers a transition to another activity. Activity diagrams are similar to flowchart diagrams in that they can be used to represent control flow (i.e., the order in which operations occur) and data flow (i.e., the objects that are exchanged among operations).

In this Activity Diagram, the faculty visit the site gives his/her details to view their profile and if the details are valid then profile will be displayed and checks for leaves if leaves are completed then faculty will logout otherwise, he can apply for leave and after applying the leave he can logout from the site.

Purpose

The basic purpose of activity diagrams is similar to other four diagrams. It captures the dynamic behaviour of the system. Other diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. The diagrams are not only for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing is the message port.
It can be described as:
1) Draw the activity flow of a system.
2) Describe the sequence from one activity to another.
3) Describe the parallel, branched and concurrent flow of the system.



Load Dataset
Application of RFA,Decision tree
pattern matching with train and test data
Output from applied algorithm
1
Fraudulent Transaction
0
Legal Transaction

Database Design

Entity Relationship Diagram(ER)

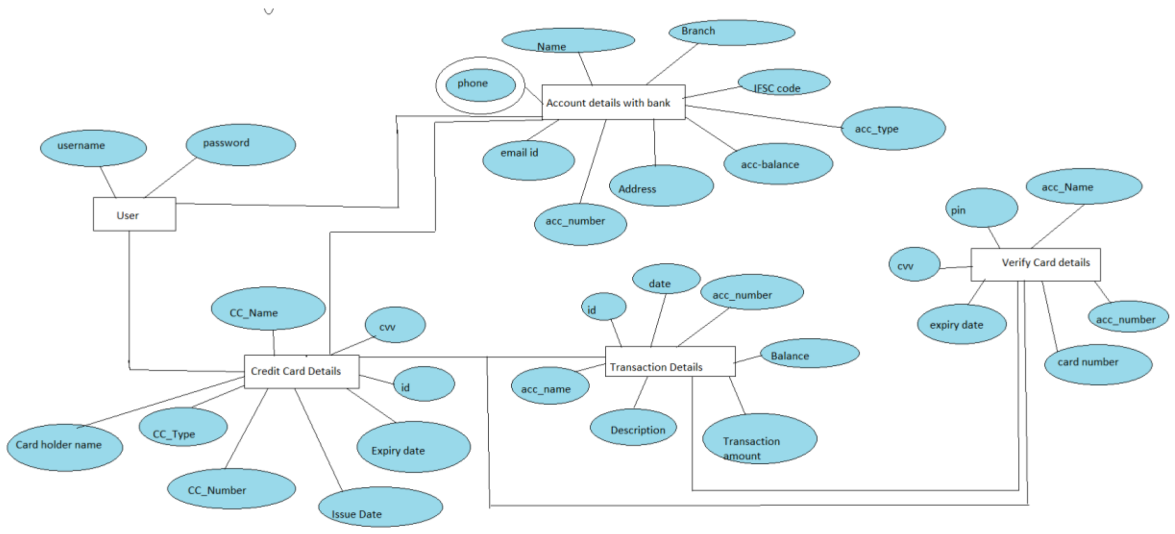


Fig. ER Diagram

CHAPTER-4

IMPLEMENTATION DETAILS

4.1Software Technologies

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

It has a huge number of libraries and frameworks. Different frameworks and packages used include:

- Numpy
- Pandas
- Seaborn
- python

IDE: IDE stands for Integrated Development Environment. It's a coding tool which allows to write, test, and debug your code in an easier way, as they typically offer code completion or code insight by highlighting, resource management, debugging tools, etc.

Here we are using **google colab ,**which is python open source distribution containing different IDEs. The IDE that is used is Google Colab notebook which is a classic notebook interface of Google Colab. It is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. The below images show Google Colab notebook launching. The former image shows that Google colab is launched which is in turn opened in the default (chrome) browser which is shown in the later image.

4.2Algorithm

Random Forest Classifier

Random forest is a supervised deep learning algorithm based on ensemble learning. Ensemble learning is an algorithm where the predictions are derived by assembling or bagging different models or similar model multiple times. The random forest algorithm works in a similar way and uses multiple algorithm i.e. multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

Advantages of using random forest algortihm

· The random forest algorithm is not biased and depends on multiple trees where each tree is trained separately based on the data, therefore biasedness is reduced overall.

· It's a very stable algorithm. Even if a new data point is introduced in the dataset it doesn't affect the overall algorithm rather affect the only a single tree.

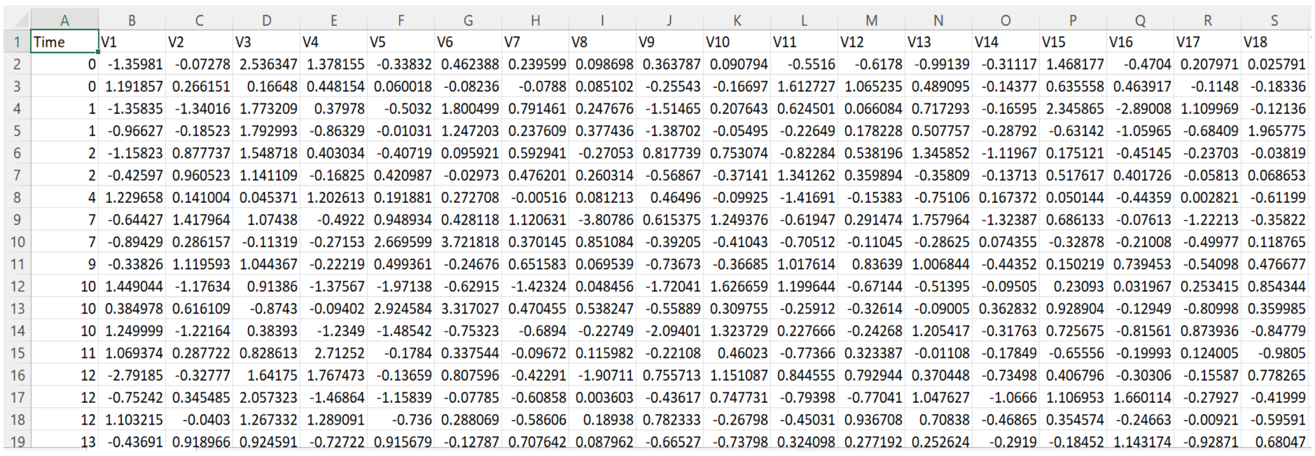
· It works well when one has both categorical and numerical features.

The random forest algorithm also works well when data possess missing values, or when it's not been scaled properly. Thus, using this Random forest algorithm and decision trees algorithm we have extracted the accurate percentage of detection of fraud from the given dataset by studying its behaviour.

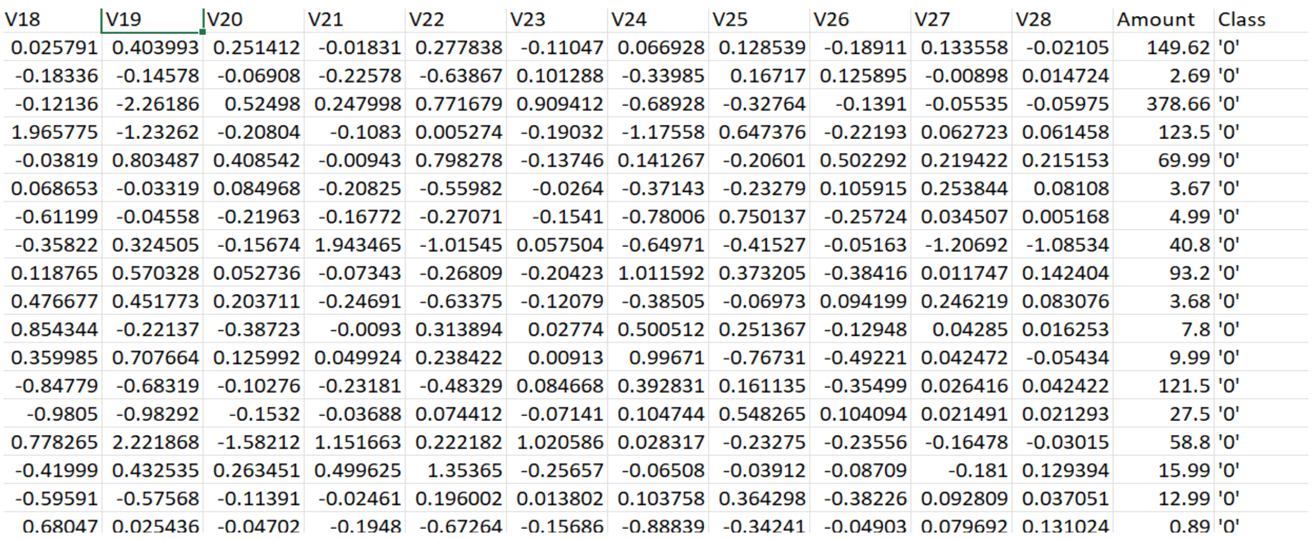
4.3Dataset

The dataset contains transactions made by credit cards by European cardholders which was made available on Kaggle website. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Due to confidentiality issues, original features and more background information about the data are not available in genuine format . Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18
2	0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791
3	0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336
4	1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969	-0.12136
5	1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409	1.965775
6	2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703	-0.03819
7	2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813	0.068653
8	4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821	-0.61199
9	7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.291474	1.757964	-1.32387	0.686133	-0.07613	-1.22213	-0.35822
10	7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977	0.118765
11	9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098	0.476677
12	10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.67144	-0.51395	-0.09505	0.23093	0.031967	0.253415	0.854344
13	10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998	0.359985
14	10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729	0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936	-0.84779
15	11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005	-0.9805
16	12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587	0.778265
17	12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731	-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927	-0.41999
18	12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798	-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24663	-0.00921	-0.59591
19	13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798	0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871	0.68047



V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0.025791	0.403993	0.251412	-0.01831	0.277838	-0.11047	0.066928	0.128539	-0.18911	0.133558	-0.02105	149.62	'0'
-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.101288	-0.33985	0.16717	0.125895	-0.00898	0.014724	2.69	'0'
-0.12136	-2.26186	0.52498	0.247998	0.771679	0.909412	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	'0'
1.965775	-1.23262	-0.20804	-0.1083	0.005274	-0.19032	-1.17558	0.647376	-0.22193	0.062723	0.061458	123.5	'0'
-0.03819	0.803487	0.408542	-0.00943	0.798278	-0.13746	0.141267	-0.20601	0.502292	0.219422	0.215153	69.99	'0'
0.068653	-0.03319	0.084968	-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.105915	0.253844	0.08108	3.67	'0'
-0.61199	-0.04558	-0.21963	-0.16772	-0.27071	-0.1541	-0.78006	0.750137	-0.25724	0.034507	0.005168	4.99	'0'
-0.35822	0.324505	-0.15674	1.943465	-1.01545	0.057504	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	'0'
0.118765	0.570328	0.052736	-0.07343	-0.26809	-0.20423	1.011592	0.373205	-0.38416	0.011747	0.142404	93.2	'0'
0.476677	0.451773	0.203711	-0.24691	-0.63375	-0.12079	-0.38505	-0.06973	0.094199	0.246219	0.083076	3.68	'0'
0.854344	-0.22137	-0.38723	-0.0093	0.313894	0.02774	0.500512	0.251367	-0.12948	0.04285	0.016253	7.8	'0'
0.359985	0.707664	0.125992	0.049924	0.238422	0.00913	0.99671	-0.76731	-0.49221	0.042472	-0.05434	9.99	'0'
-0.84779	-0.68319	-0.10276	-0.23181	-0.48329	0.084668	0.392831	0.161135	-0.35499	0.026416	0.042422	121.5	'0'
-0.9805	-0.98292	-0.1532	-0.03688	0.074412	-0.07141	0.104744	0.548265	0.104094	0.021491	0.021293	27.5	'0'
0.778265	2.221868	-1.58212	1.151663	0.222182	1.020586	0.028317	-0.23275	-0.23556	-0.16478	-0.03015	58.8	'0'
-0.41999	0.432535	0.263451	0.499625	1.35365	-0.25657	-0.06508	-0.03912	-0.08709	-0.181	0.129394	15.99	'0'
-0.59591	-0.57568	-0.11391	-0.02461	0.196002	0.013802	0.103758	0.364298	-0.38226	0.092809	0.037051	12.99	'0'
0.68047	0.025436	-0.04702	-0.1948	-0.67264	-0.15686	-0.88839	-0.34241	-0.04903	0.079692	0.131024	0.89	'0'

CHAPTER-5

TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product is the process of exercising software with the intent of ensuring that the Software system meets s requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. The testing of individual software units of the application. This is a structural testing, that relies on knowledge of s construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

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 centred on the following items:

Valid Input

Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of valid input must be accepted.

Invalid: identified classes of application outputs must be exercised.

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

System Testing

System testing ensures that the entire integrated software system meets requirements. Tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points

Integration Testing

Integration tests are designed to test integrated software components to determine if the user actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Algorithm/ Parameter	**Random forest**	**Decision tree**
Accuracy	0.99957866647	0.99922755521
Precision	0.97435897435	0.76470588235
Recall	0.77551020408	0.79591836734
F1-score	0.86363636363	0.77999999999

Comparison of Algorithms

- We followed with a Random forest classifier and obtained an accuracy of **0.99957**.
- Finally, the dataset is presented with decision and we obtained an accuracy of **0.99922**.

S. No	REQUIRE-MENTS	REQ No.	ESSENTIAL/ DESIRABLE	DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1.	Loading of dataset	RS1	ESSENTIAL	Dataset is downloaded from kaggle and then loaded	The data set should be loaded without any errors	The data set is loaded without any errors	SUCCESS
2.	Data Cleaning	RS2	ESSENTIAL	Dataset is checked for null values; if any null values are found they should be handled.	Data should be cleaned and should contain no null values.	No null values found	SUCCESS
3.	Feature Selection	RS3	ESSENTIAL	Feature selection is done using heatmap	Heatmap shows whether all columns are relevant or not to the target	Heatmap is generated and all the features are found to be significant	SUCCESS
4.	Splitting of the dataset into train and test data.	RS4	ESSENTIAL	Data set is split into train and test data in the ratio 8:2.	According to the given ratio the dataset should be split into train and test data.	The given dataset is split into train and test data in 8:2 ratio	SUCCESS
5.	RFA and decision tree algorithms - are called for given dataset	RS5	ESSENTIAL	RFA, Decision tree are applied.	Each of the model applied should work without any errors.	Results are classified into fraud and non-fraud transaction	SUCCESS
6.	Evaluation and comparison of models	RS6	ESSENTIAL	Accuracy, precision, Recall and F1-score are measured.	All the evaluation properties are measured properly.	All the properties are measured and compared	SUCCESS

CHAPTER-6

CONCLUSION

Here, a comparative study is made on different tree-based algorithms on a European credit card transactions dataset of size (284807x31) which was taken from Kaggle website. On raw and pre-processed data, there are three four techniques such as Random Forest Classifier and decision tree are applied in Python. Based on certain parameters like precision, accuracy, F1-score, recall and performances of these techniques are evaluated. It is seen through the achieved results that in comparison to these four techniques, the performance of Random Forest classifier is better.

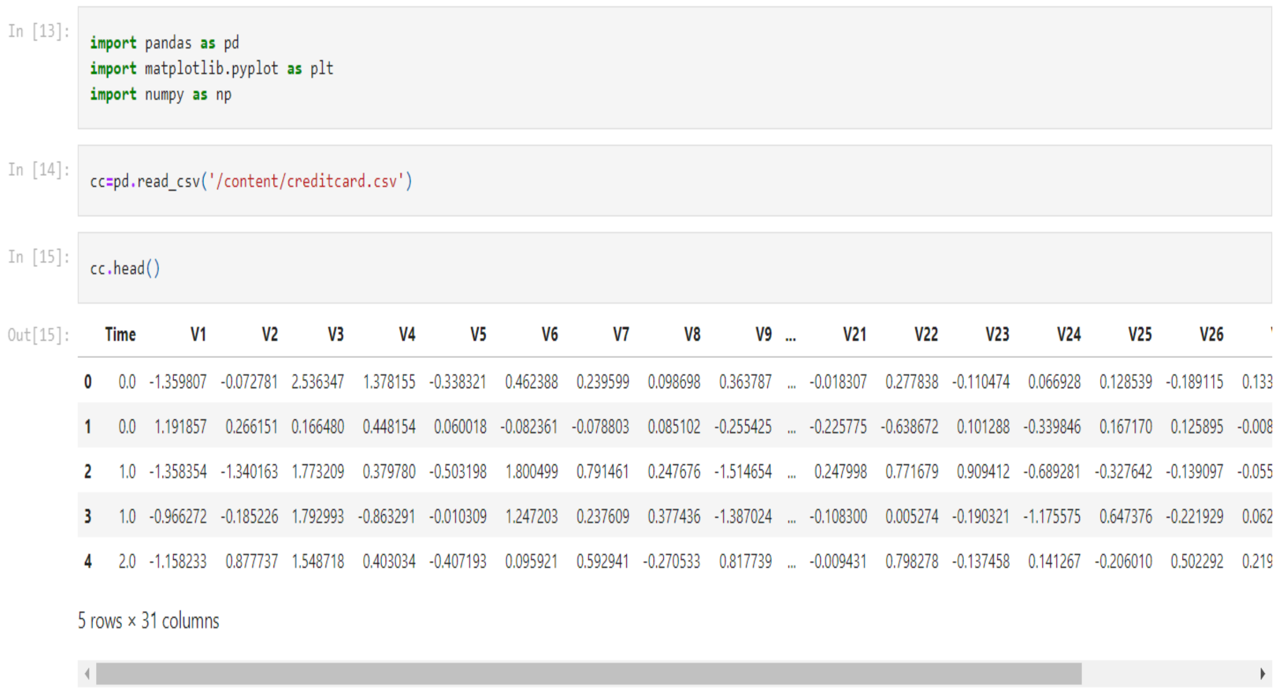
REFERENCES

[1].https://www.researchgate.net/publication/336800562_Credit_Card_Fraud_Detection_usin
g_Deep learning
September 2019International Journal of Engineering and Technical Research 08(09)
DOI:10.17577/IJERTV8IS090031
Authors :Maniraj S P,SRM Institute of Science and Technology
Aditya Saini
[2]. Xuan, Shiyang, et al.
"Random Forest for Credit Card Fraud Detection."
2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, doi:10.1109/icnsc.2018.8361343.

APPENDIX

A. INPUT/OUTPUT SCREENS

Step 1 – Importing required libraries for Credit Card Fraud Detection.



In [13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [14]:
```
cc=pd.read_csv('/content/creditcard.csv')
```

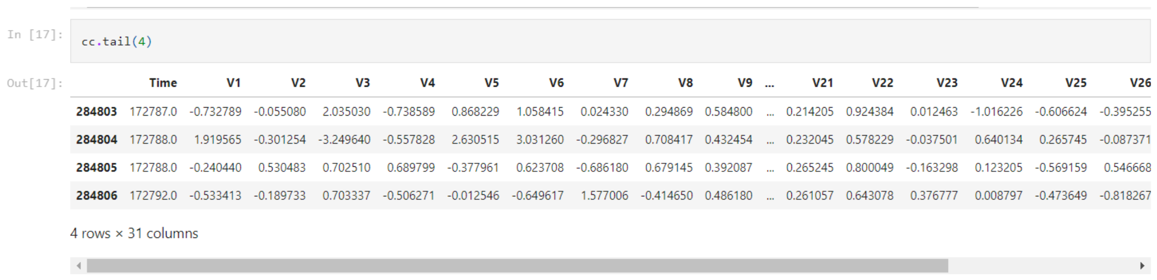
In [15]:
```
cc.head()
```

Out[15]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	'
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219

5 rows × 31 columns

Step 2 – Reading our input data for Credit Card Fraud Detection.



In [17]:
```
cc.tail(4)
```

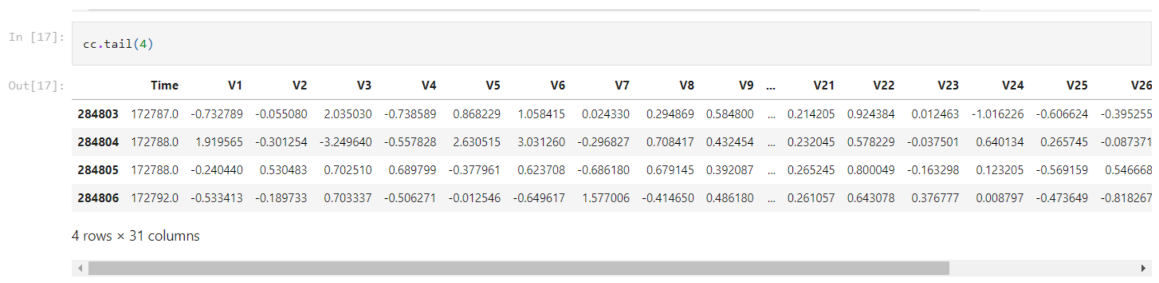
Out[17]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267

4 rows × 31 columns

3. Step 3 – Describing our data for Credit Card Fraud Detection.

- Here we can see that all the columns are normalized except the Time and Amount columns.
- So we will normalize them further.

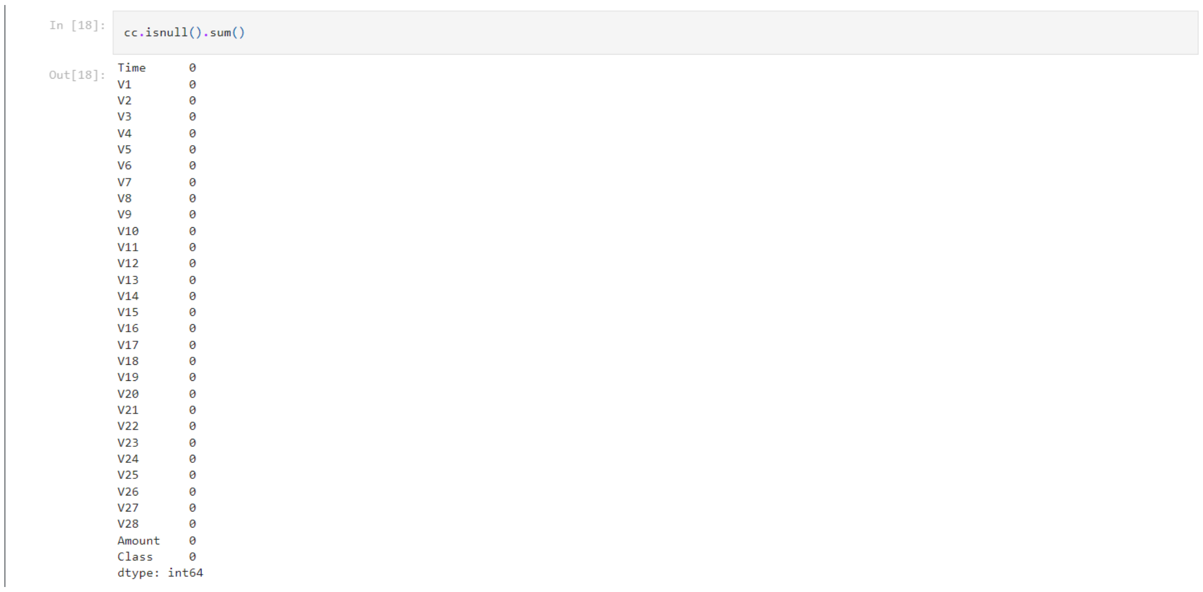


```
In [17]: cc.tail(4)
```

Out[17]:

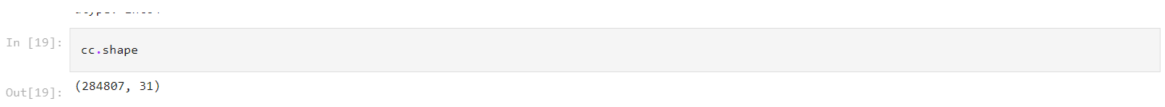
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267

4 rows × 31 columns



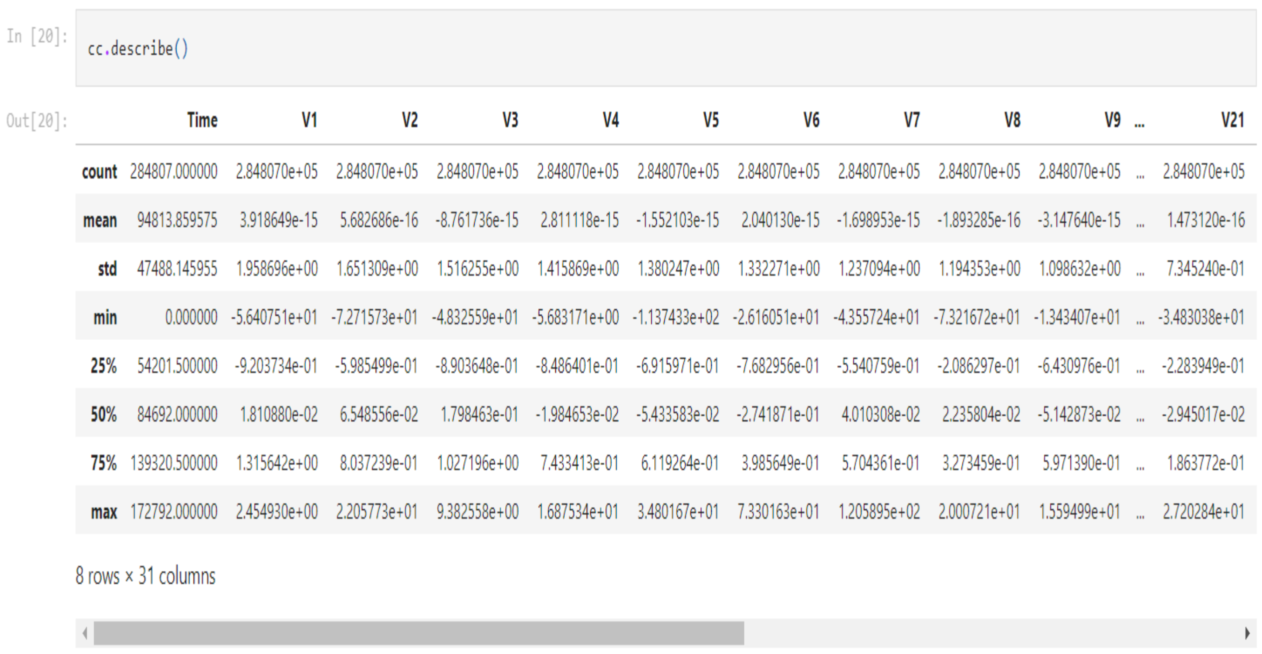
```
In [18]: cc.isnull().sum()

Out[18]: Time      0
         V1        0
         V2        0
         V3        0
         V4        0
         V5        0
         V6        0
         V7        0
         V8        0
         V9        0
         V10       0
         V11       0
         V12       0
         V13       0
         V14       0
         V15       0
         V16       0
         V17       0
         V18       0
         V19       0
         V20       0
         V21       0
         V22       0
         V23       0
         V24       0
         V25       0
         V26       0
         V27       0
         V28       0
         Amount    0
         Class     0
         dtype: int64
```

1.



```
In [19]: cc.shape

Out[19]: (284807, 31)
```

5.



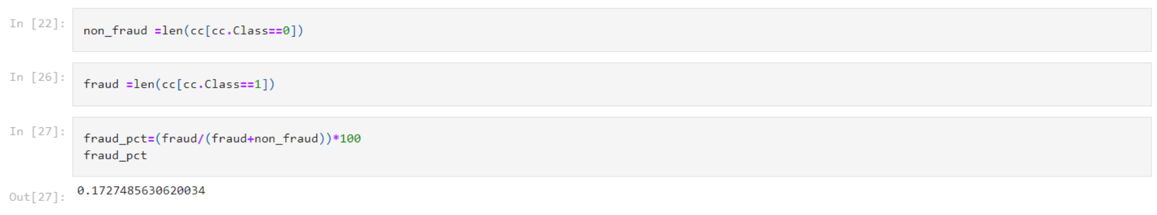
```
In [20]: cc.describe()
```

Out[20]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15	...	1.473120e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01

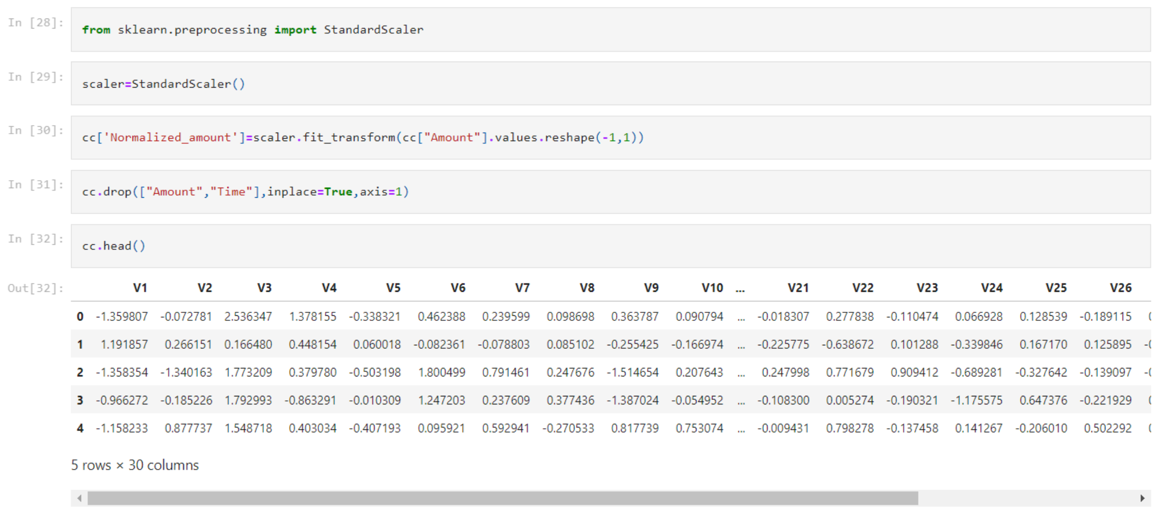
8 rows × 31 columns

6. Line 1 – Taking out all the nonfraud transactions from our dataset.
Line 2 – Taking out all the fraud transactions from our dataset.



```
In [22]: non_fraud =len(cc[cc.Class==0])

In [26]: fraud =len(cc[cc.Class==1])

In [27]: fraud_pct=(fraud/(fraud+non_fraud))*100
         fraud_pct

Out[27]: 0.1727485630620034
```

7.



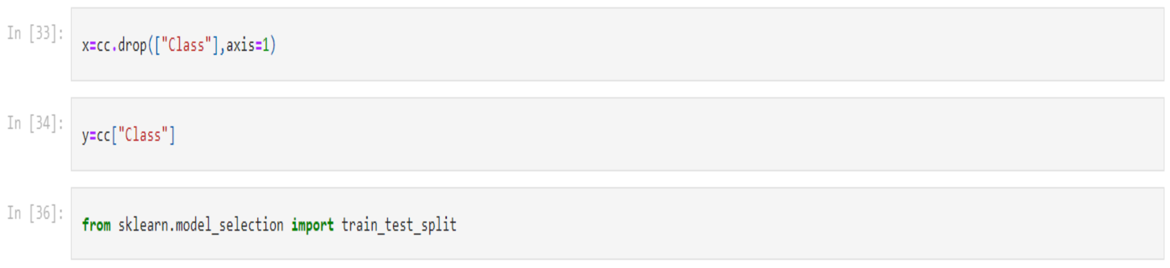
```
In [28]: from sklearn.preprocessing import StandardScaler

In [29]: scaler=StandardScaler()

In [30]: cc['Normalized_amount']=scaler.fit_transform(cc["Amount"].values.reshape(-1,1))

In [31]: cc.drop(["Amount","Time"],inplace=True,axis=1)

In [32]: cc.head()
```

Out[32]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292

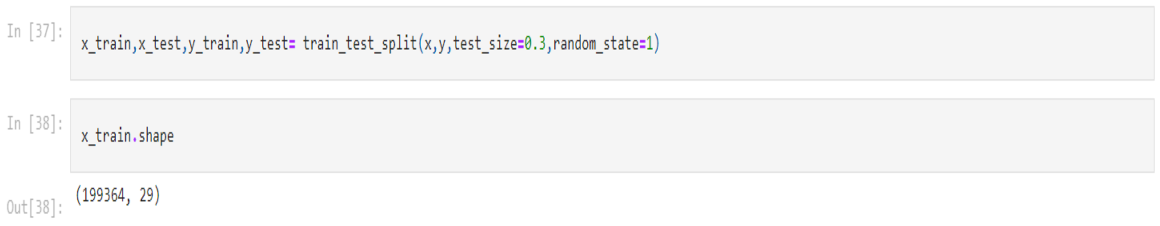
5 rows × 30 columns

8.



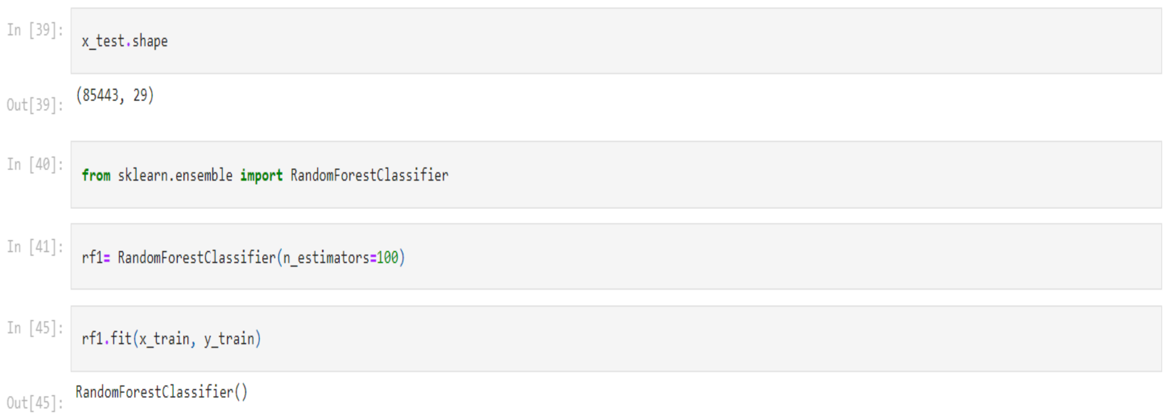
```
In [33]: x=cc.drop(["Class"],axis=1)

In [34]: y=cc["Class"]

In [36]: from sklearn.model_selection import train_test_split
```

9. Step 9 – train-test-split our data for Credit Card Fraud Detection.



```
In [37]: x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3,random_state=1)

In [38]: x_train.shape

Out[38]: (199364, 29)
```

10.



```
In [39]: x_test.shape

Out[39]: (85443, 29)

In [40]: from sklearn.ensemble import RandomForestClassifier

In [41]: rf1= RandomForestClassifier(n_estimators=100)

In [45]: rf1.fit(x_train, y_train)

Out[45]: RandomForestClassifier()
```

11. Performance of our model

Accuracy - 0.99 %

```
In [46]: prediction_rf=rf1.predict(x_test)
```

```
In [47]: rf_score=rf1.score(x_test,y_test)*100
```

```
In [48]: rf_score

Out[48]: 99.95201479348805
```

B.CODE

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

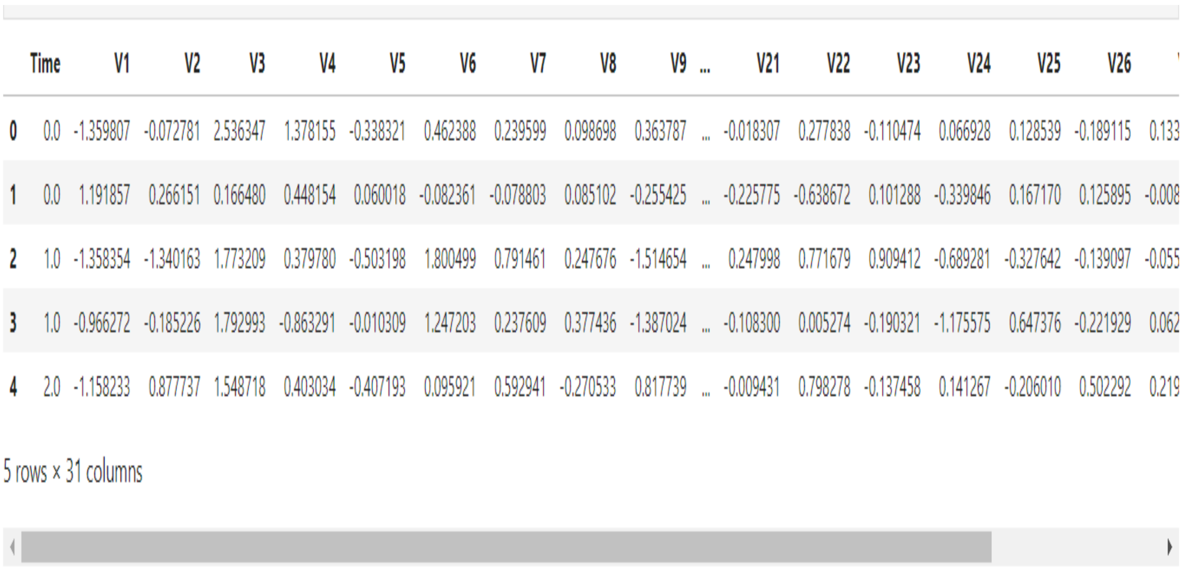
In [14]:

```
cc=pd.read_csv('/content/creditcard.csv')
```

In [15]:

```
cc.head()
```

Out[15]:



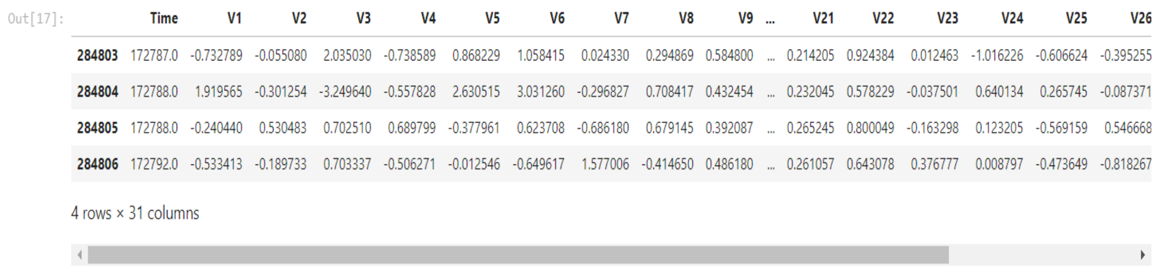
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219

5 rows × 31 columns

In [17]:

```
cc.tail(4)
```

Out[17]:



Out[17]:

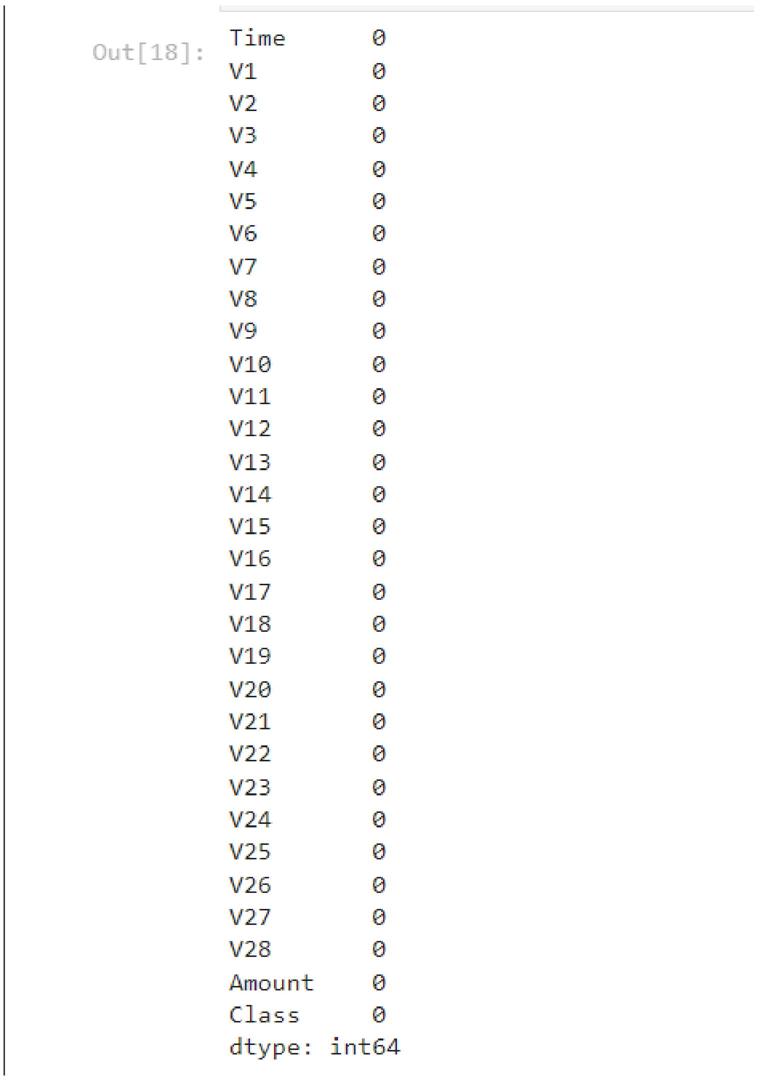
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267

4 rows × 31 columns

In [18]:

```
cc.isnull().sum()
```

Out[18]:



```
Out[18]: Time      0
         V1        0
         V2        0
         V3        0
         V4        0
         V5        0
         V6        0
         V7        0
         V8        0
         V9        0
         V10       0
         V11       0
         V12       0
         V13       0
         V14       0
         V15       0
         V16       0
         V17       0
         V18       0
         V19       0
         V20       0
         V21       0
         V22       0
         V23       0
         V24       0
         V25       0
         V26       0
         V27       0
         V28       0
         Amount    0
         Class     0
         dtype: int64
```

In [19]:

```
cc.shape
```

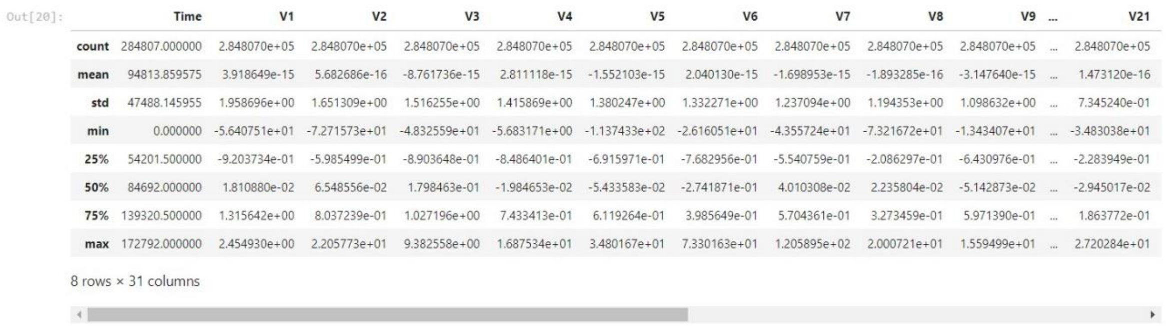
Out[19]:

```
(284807, 31)
```

In [20]:

```
cc.describe()
```

Out[20]:



Out[20]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15	...	1.473120e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01

8 rows × 31 columns

In [26]:

```
non_fraud =len(cc[cc.Class==0])

fraud =len(cc[cc.Class==1])
```

In [27]:

```
fraud_pct=(fraud/(fraud+non_fraud))*100
fraud_pct
```

Out[27]:

```
0.1727485630620034
```

In [28]:

```
from sklearn.preprocessing
import StandardScaler
```

In [29]:

```
scaler=StandardScaler()
```

In [30]:

```
cc['Normalized_amount']=scaler.fit_transform(cc["Amount"].values.reshape(-
1,1))
```

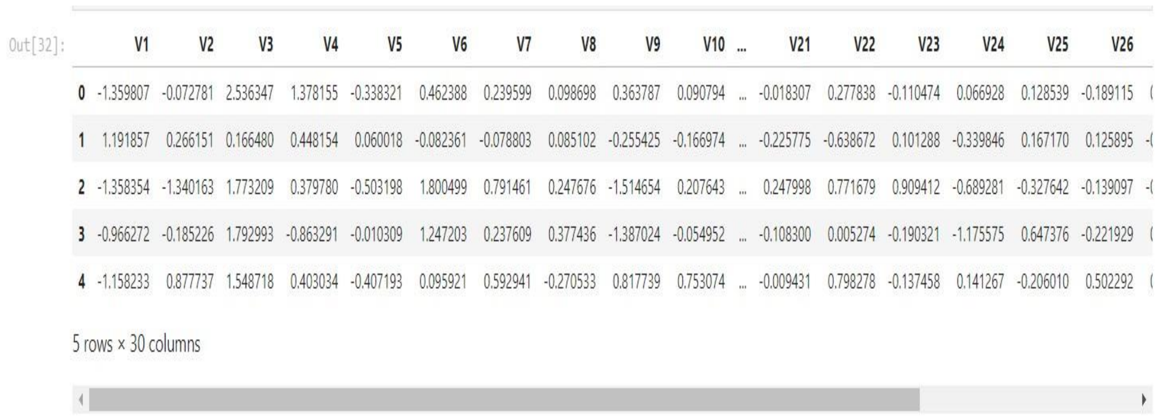
In [31]:

```
cc.drop(["Amount","Time"],inplace=True,axis=1)
```

In [32]:

```
cc.head()
```

Out[32]:



Out[32]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292

5 rows × 30 columns

In [33]:

```
x=cc.drop(["Class"],axis=1)
```

In [34]:

```
y=cc["Class"]
```

In [36]:

```
from sklearn.model_selection
import train_test_split
```

In [37]:

```
x_train,x_test,y_train,y_test=
train_test_split(x,y,test_size=0.3,random_state=1)
```

In [38]:

```
x_train.shape
```

Out[38]:

```
(199364, 29)
```

In [39]:

```
x_test.shape
```

Out[39]:

```
(85443, 29)
```

In [40]:

```
from sklearn.ensemble
import RandomForestClassifier
```

In [41]:

```
rf1= RandomForestClassifier(n_estimators=100)
```

In [45]:

```
rf1.fit(x_train, y_train)
```

Out[45]:

```
RandomForestClassifier()
```

In [46]:

```
prediction_rf=rf1.predict(x_test)
```

In [47]:

```
rf_score=rf1.score(x_test,y_test)*100
```

In [48]:

```
rf_score
```

Out[48]:

```
0.99
```