

A Mini Project Report

On

CREDIT CARD FRAUD DETECTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

Sk.Farheen

(20NN1A0546)

R.Usha

(20NN1A0542)

V.Cahitanya Sarojitha

(20NN1A0553)

P.Jhansi Sravani

(20NN1A0537)

Under the Esteemed Guidance of

Mrs.P.Ganga Bhavani

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN

PEDAPALAKALURU, GUNTUR-522005

(Approved by AICTE, NEW DELHI, and Affiliated to JNTUK, Kakinada.)

2020-2024

VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN

(Approved by AICTE, NEW DELHI, and Affiliated to JNTUK, Kakinada)

PEDAPALAKALURU, GUNTUR-522005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the mini project entitled “**Credit Card Fraud Detection Using ML**”, is a bonafide work of **Sk.Farheen(20NN1A0546), R.Usha(20NN1A0542), V. Chaitanya Sarojitha (20NN1A0553) and P.Jhansi Sravani(20NN1A0537)** submitted for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from **VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN, GUNTUR.**

Project Guide

Mrs.P.Ganga Bhavani

Assistant Professor

Head of the Department

Dr. V Lakshman Narayana

Associate Professor

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work described in this mini project work, entitled “**Credit Card Fraud Detection Using ML**” which is submitted by us in partial fulfillment for the award of **Bachelor of Technology** in the Department of **Computer Science and Engineering** to the **Vignan’s Nirula Institute of Technology and Science for women**, affiliated to Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, is the result of work done by us under the guidance of **Mrs.P.Ganga Bhavani**, Assistant Professor.

The work is original and has not been submitted for any Degree/ Diploma of this or any other university.

Place:

Date:

Sk.Farheen (20NN1A0546)

R.Usha (20NN1A0542)

V.Chaitanya Sarojitha (20NN1A0553)

P.Jhansi Sravani (20NN1A0537)

ACKNOWLEDGEMENT

We express our heartfelt gratitude to our beloved principal **Dr. P. Radhika** for giving a chance to study in our esteemed institution and providing us all the required resources.

We would like to thank **Dr. V Lakshman Narayana, Associate Professor, Head of the Department of Computer Science and Engineering**, for his extended and continuous support, valuable guidance, and timely advices in the completion of this mini-project thesis.

We wish to express our profound sense of sincere gratitude to our Project Guide **Mrs.P.Ganga Bhavani, Assistant Professor, Department of Computer Science and Engineering**, without her help, guidance, and motivation this mini-project thesis could not have been completed the mini project successfully.

We also thank all the faculty of the Department of Computer Science and Engineering for their help and guidance on numerous occasions, which has given us the cogency to build- up adamant aspirations over the completion of our mini-project thesis.

Finally, we thank one and all who directly or indirectly helped us to complete our mini-project thesis successfully.

Sk.Farheen (20NN1A0546)

R.Usha (20NN1A0542)

V.Chaitanya Sarojitha (20NN1A0553)

P.Jhansi Sravani (20NN1A0537)

TABLE OF CONTENTS

Chapter 1 Introduction	1
Chapter 2 Literature Survey	3
Chapter 3 System Analysis	9
3.1 Existing System	10
3.2 Proposed System	10
Chapter 4 Feasability Study	11
4.1 Economical Feasability	12
4.2 Technical Feasability	12
Chapter 5 System Requirments	13
5.1 Hardware Requirements	14
5.2 Software Requirements	14
Chapter 6 System Design	15
6.1 System Architecture	16
6.2 Data Flow Diagram	16
6.3 UML Diagrams	17
Chapter 7 Implementation	25
7.1 Modules	26
7.2 Sample Code	27
Chapter 8 System Testing	28
8.1 Unit Testing	30
8.2 Integration Testing	31
8.3 Acceptance Testing	31
Chapter 9 Input Design And Output Design	32
9.1 Input Design	33
9.2 Output Design	33
Chapter 10 Screenshots	35
Chapter 11 Future Enhancement	44
Chapter 12 Conclusion	46
Chapter 13 Bibliography	48

List of figures

Fig.6.1. System Architecture	16
Fig.6.2 Data Flow Diagram	17
Fig .6.3. UML Diagrams	18
Fig.6.3.1. Use case Diagram	19
Fig.6.3.2. Class Diagram	20
Fig.6.3.3. Sequence diagram	20
Fig.6.3.4 Sequence diagram	21
Fig.6.3.5. Activity Diagram	22
Fig.6.3.6 Component Diagram	23
Fig.6.3.7 Deployment Diagram	24
Fig.10.1.1 Screenshot of Transaction Details Of Credit Cards	36
Fig.10.1.2 Screenshots of Transaction Details Of Credit Cards	37
Fig.10.1.3 Screenshot of Classification Of Fraud and Non Fraud	38
Fig.10.2 Screenshot of Graph of detecting fraud and normal transactions	39
Fig.10.3 Screenshot of Graph of Detecting Fraud Transactions	40
Fig.10.4.Screenshot of Graph of Detecting Fraud and Normal Transactions	41
Fig.10.5 Screenshot Of Confusion Matrix	42
Fig.10.6 Screenshot of Final Output	43

Acronyms

ML	Machine Learning
AI	Artificial Intelligence
BDFL	Benevolent Dictator For Life
OSI	Open Source License
GUI	Graphical User Interface
DFD	Data Flow Diagram
UML	Unified Model Language

CREDIT CARD FRAUD DETECTION USING ML

ABSTRACT

Billions of dollars of loss are caused every year by fraudulent credit card transactions. The design of efficient fraud detection algorithms is key for reducing these losses, and more and more algorithms rely on advanced machine learning techniques to assist fraud investigators. The design of fraud detection algorithms is however particularly challenging due to the non-stationary distribution of the data, the highly unbalanced class distributions, and the availability of few transactions labeled by fraud investigators. At the same time, public data are scarcely available for confidentiality issues, leaving unanswered many questions about what is the best strategy. In this thesis, we aim to provide some answers by focusing on crucial issues such as: i) why and how under-sampling is useful in the presence of class imbalance (i.e. frauds are a small percentage of the transactions), ii) how to deal with unbalanced and evolving data streams (non-stationarity due to fraud evolution and change of spending behavior), iii) how to assess performances in a way which is relevant for detection and iv) how to use feedbacks provided by investigators on the fraud alerts generated. Finally, we design and assess a prototype of a Fraud Detection System able to meet real-world working conditions and that is able to integrate investigators' feedback to generate accurate alerts.

Index Terms— credit card, fraud detection, online shopping, e-commerce, logistic regression.

1. INTRODUCTION

INTRODUCTION

The online shopping growing day to day. Credit cards are used for purchasing goods and services with the help of virtual cards and physical card where as virtual card for online transaction and physical card for offline transaction. In a physical-card based purchase, the cardholder presents his card physically to a merchant for making a payment. To carry out fraudulent transactions in this kind of purchase, an attacker has to steal the credit card. If the cardholder does not realize the loss of card, it can lead to a substantial financial loss to the credit card company. In online payment mode, attackers need only little information for doing fraudulent transaction (secure code, card number, expiration date etc.). In this purchase method, mainly transactions will be done through Internet or telephone. To commit fraud in these types of purchases, a fraudster simply needs to know the card details. Most of the time, the genuine cardholder is not aware that someone else has seen or stolen his card information. The only way to detect this kind of fraud is to analyse the spending patterns on every card and to figure out any inconsistency with respect to the “usual” spending patterns. Fraud detection based on the analysis of existing purchase data of cardholder is a promising way to reduce the rate of successful credit card frauds. Since humans tend to exhibit specific behavioristic profiles, every cardholder can be represented by a set of patterns containing information about the typical purchase category, the time since the last purchase, the amount of money spent, etc. Deviation from such patterns is a potential threat to the system.

Credit card fraud is a growing concern in the present world with the growing fraud in the government offices, corporate industries, finance industries, and many other organizations. In the present world, the high dependency on the internet is the reason for an increased rate of credit card fraud transactions but the fraud has increased not only online but also offline transactions. Though the data mining techniques are used the result is not much accurate to detect these credit card frauds. The only way to minimize these losses is the detection of the fraud using efficient algorithms which is a promising way to reduce credit card frauds. As the use of the internet is increasing, a credit card is issued by the finance company. Having a credit card means that we can borrow the funds. The funds can be used for any of the purposes. When coming to the issuance of the card, the condition involved is that the cardholder will pay back the original amount they borrowed along with the additional charges they agreed to pay.

A credit card is said to be a fraud when some other person uses your credit card instead of you without your authorization. Fraudsters steal the credit card PIN or the account details to perform any of the unauthorized transactions without stealing the original physical card. Using the credit card fraud detection we could find out whether the new transactions are fraud one or a genuine one. The fraud that is committed may involve the card such as a credit card or debit card. In this, the card itself acts as a fraudulent source in the transaction. The purpose of committing the crime may be to obtain the goods without paying money or to obtain the unauthorized fund. Credit cards are a nice target for fraud. The reason is that in a very short time a lot of money can be earned without taking many risks and even the crime will take many weeks to be detected.

As the use of the internet nowadays is very much increasing there may be many chances for the fraudsters to commit the credit card frauds. The main fraud cases that are ongoing in the present world are in those of the e-commerce sites. In the present generation, people are showing much interest in getting things online rather than going and purchasing them, and due to this, the growth of the ecommerce sites is increasing and thereby there is a huge chance of credit card fraud. So to avoid such credit card frauds, we need to find out the best algorithm that reduces credit card frauds.

As credit card transactions become the most prevailing mode of payment for both online and offline transaction, credit card fraud rate also accelerates. Credit card fraud can come in either inner card fraud or external card fraud. Inner card fraud occurs as a result of consent between cardholders and bank by using false identity to commit fraud while the external card fraud involves the use of stolen credit card to get cash through dubious means. A lot of researches have been devoted to detection of external card fraud which accounts for majority of credit card frauds. Detecting fraudulent transactions using traditional methods of manual detection is time consuming and inefficient, thus the advent of big data has made manual methods more impractical. However, financial institutions have focused attention to recent computational methodologies to handle credit card fraud problem.

2. LITERATURE SURVEY

LITERATURE SURVEY

2.1 Automatic credit card fraud detection based on non-linear signal processing

Authors: Salazar, Addison, et al.

Abstract: Fraud detection is a critical problem affecting large financial companies that has increased due to the growth in credit card transactions. This paper presents a new method for the automatic detection of fraud in credit card transactions based on non-linear signal processing. The proposed method consists of the following stages: feature extraction, training and classification, decision fusion, and result presentation. Discriminant-based classifiers and an advanced non-Gaussian mixture classification method are employed to distinguish between legitimate and fraudulent transactions. The posterior probabilities produced by classifiers are fused by means of order statistical digital filters. Results from data mining of a large database of real transactions are presented. The feasibility of the proposed method is demonstrated for several datasets using parameters derived from receiver characteristic operating analysis and key performance indicators of the business

2.2 Credit card fraud and detection techniques: a review

Authors: John Pointon, Hussein A Abdou, Linda Delamaire

Abstract: Fraud is one of the major ethical issues in the credit card industry. The main aims are, firstly, to identify the different types of credit card fraud, and, secondly, to review alternative techniques that have been used in fraud detection. The sub-aim is to present, compare, and analyze recently published findings in credit card fraud detection. This article defines common terms in credit card fraud and highlights key statistics and figures in this field. Depending on the type of fraud faced by banks or credit card companies, various measures can be adopted and implemented. The proposals made in this paper are likely to have beneficial attributes in terms of cost savings and time efficiency. The significance of the application of the techniques reviewed here is in the minimization of credit card fraud. Yet there are still ethical issues when genuine credit card customers are misclassified as fraudulent.

2.3 Generation and Interpretation of Temporal Decision Rules

Authors: K. Karimi and H.J. Hamilton

Abstract: We present a solution to the problem of understanding a system that produces a sequence of temporally ordered observations. Our solution is based on generating and interpreting a set of temporal decision rules. A temporal decision rule is a decision rule that can be used to predict or retrodict the value of a decision attribute, using condition attributes that are observed at times other than the decision attribute's time of observation. A rule set, consisting of a set of temporal decision rules with the same decision attribute, can be interpreted by our Temporal Investigation Method for Enregistered Record Sequences (TIMERS) to signify an instantaneous, an acausal or a possibly causal relationship between the condition attributes and the decision attribute. We show the effectiveness of our method, by describing a number of experiments with both synthetic and real temporal data.

2.4 Simplifying decision trees

Authors: Quinlan, J. R

Abstract: Many systems have been developed for constructing decision trees from collections of examples. Although the decision trees generated by these methods are accurate and efficient, they often suffer the disadvantage of excessive complexity and are therefore incomprehensible to experts. It is questionable whether opaque structures of this kind can be described as knowledge, no matter how well they function. This paper discusses techniques for simplifying decision trees while retaining their accuracy. Four methods are described, illustrated, and compared on a test-bed of decision trees from a variety of domains.

2.5 SOFTWARE ENVIRONMENT

Python is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt. Sources [newsgroup](#) in 1991, and version 1.0 was released in 1994.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008.

At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions.

Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But

in general, they remain not quite compatible.

Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of Life date of January 1, 2020, has been established for Python 2, after which time it will no longer be maintained. If you are a newcomer to Python, it is recommended that you focus on Python 3, as this tutorial will do.

Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.

2.6 WHY CHOOSE PYTHON

If you're going to write programs, there are literally dozens of commonly used languages to choose from. Why choose Python? Here are some of the features that make Python an appealing choice.

Python is Popular

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.

According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the Popularity of Programming Language Index.

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you'd like to dig deeper into Python salary statistics and job opportunities, you can do so here.

Python is interpreted

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step. One potential downside to interpreted languages is execution speed.

Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smartphones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way. A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

But It's Not That Simple For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming. Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

3. SYSTEM ANALYSIS

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

This was on k-means Algorithm implementation, Only the two features with the most variance were used to train the model. The model was set to have 2 clusters, 0 being non-fraud and 1 being fraud. We also experimented with different values for the hyperparameters, but they all produced similar results. Changing the dimensionality of the data (reducing it to more dimensions than 2) also made little difference on the final values.

DISADVANTAGES OF THE EXISTING SYSTEM:

The Clustering doesn't produce less accuracy when compared to Regression methods in scenarios like credit card fraud detection. Comparatively, with other algorithms, k-means produce less accurate scores in prediction in this kind of scenario

3.2 PROPOSED SYSTEM:

Our goal is to implement machine learning model in order to classify, to the highest possible degree of accuracy, credit card fraud from a dataset gathered from Kaggle. After initial data exploration, we knew we would implement a logistic regression model for best accuracy reports.

Logistic regression, as it was a good candidate for binary classification. Python sklearn library was used to implement the project, We used Kaggle datasets for Credit card fraud detection, using pandas to data frame for class ==0 for no fraud and class==1 for fraud, matplotlib for plotting the fraud and non fraud data, train_test_split for data extraction (Split arrays or matrices into random train and test subsets) and used Logistic Regression machine learning algorithm for fraud detection and print predicting score according to the algorithm. Finally Confusion matrix was plotted on true and predicted.

ADVANTAGES OF PROPOSED SYSTEM:

- The results obtained by the Logistic Regression Algorithm is best compared to any other Algorithms.
- The Accuracy obtained was almost equal to cent percent which proves using of the Logistic algorithm gives best results.
- The plots that were plotted according to the proper data that is processed during the implementation

4. FEASIBILITY STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3 SOCIAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5. SYSTEM REQUIREMENTS

SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram : 1 GB

5.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows 10
- Coding Language : python
- Tool : PyCharm
- Database : MYSQL
- Server : Flask

6. SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE:

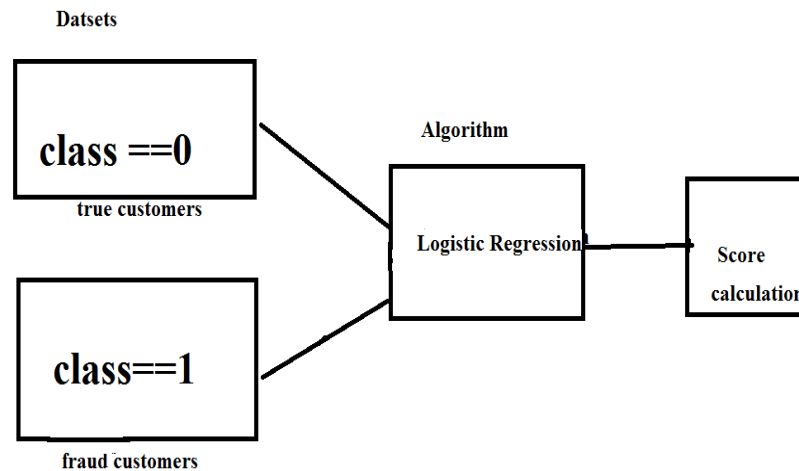


Fig.6.1. System Architecture

6.2 DATA FLOW DIAGRAM:

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- DFD is also known as a bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

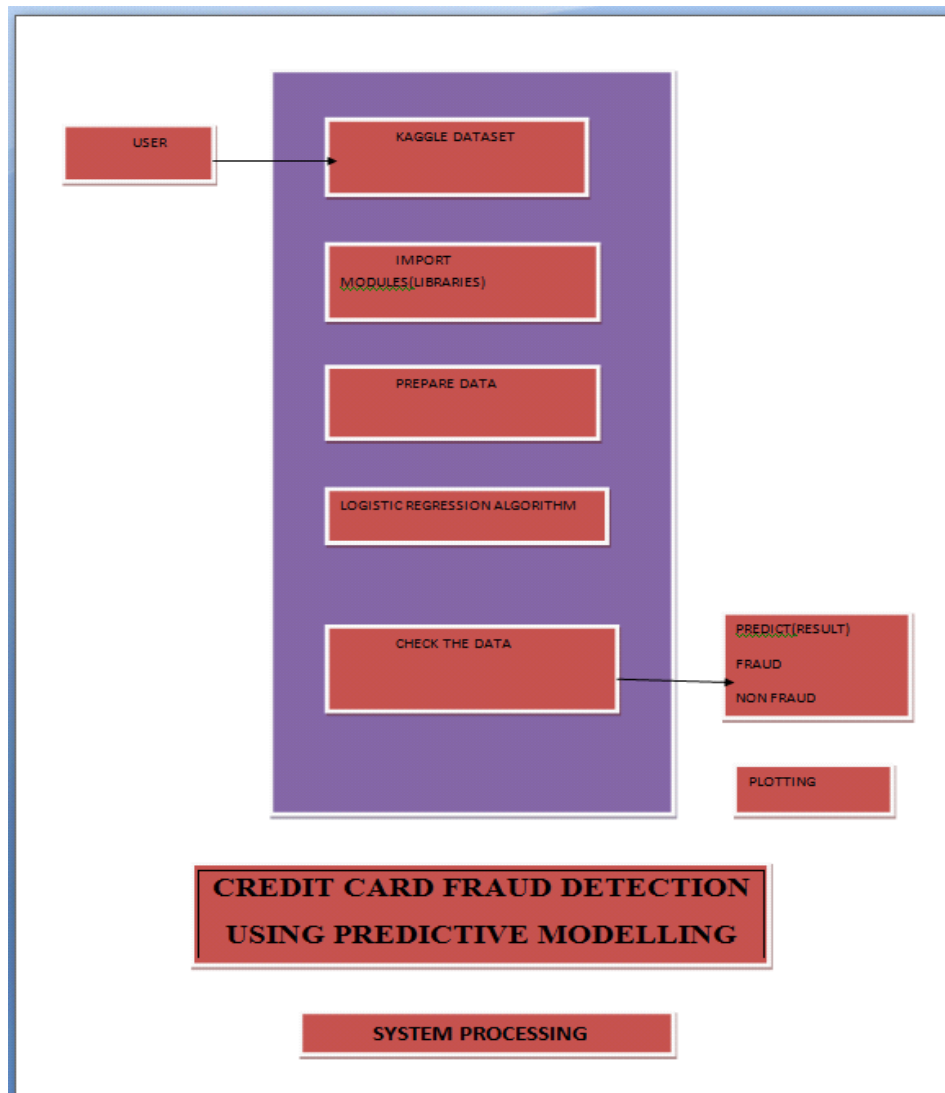


Fig 6.2. Data Flow Diagram

6.3 UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form, UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software

systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Integrate best practices.

USE CASE DIAGRAM:

A use-case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a 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. The roles of the actors in the system can be depicted.

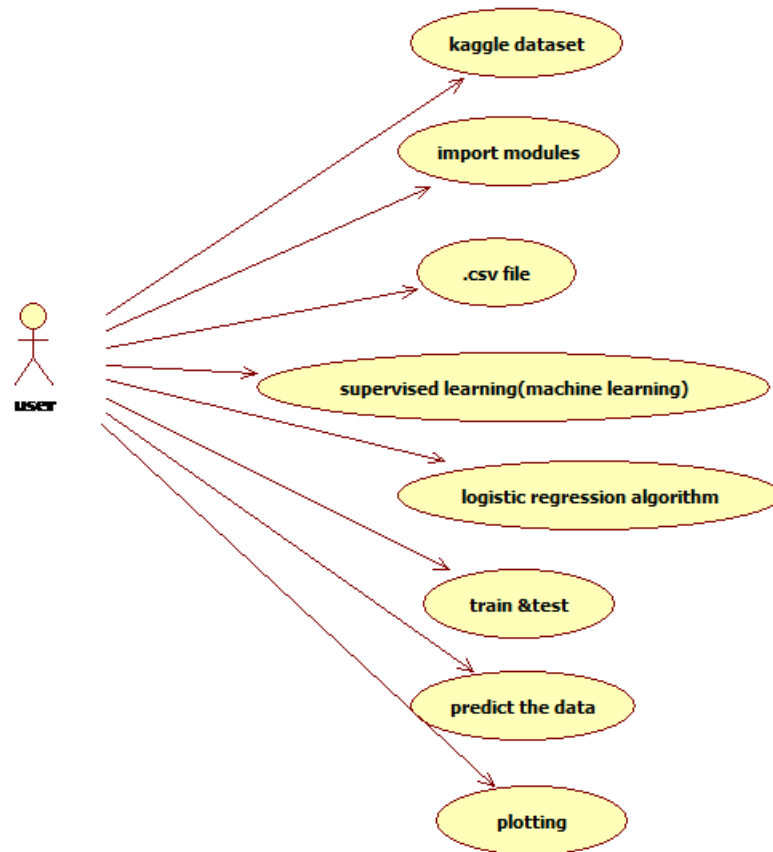


Fig 6.3.1. Use Case Diagram

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. Transaction data is processed by DataPreprocessing and used by MLModel for training and prediction. User information is utilized by ML Model for training.

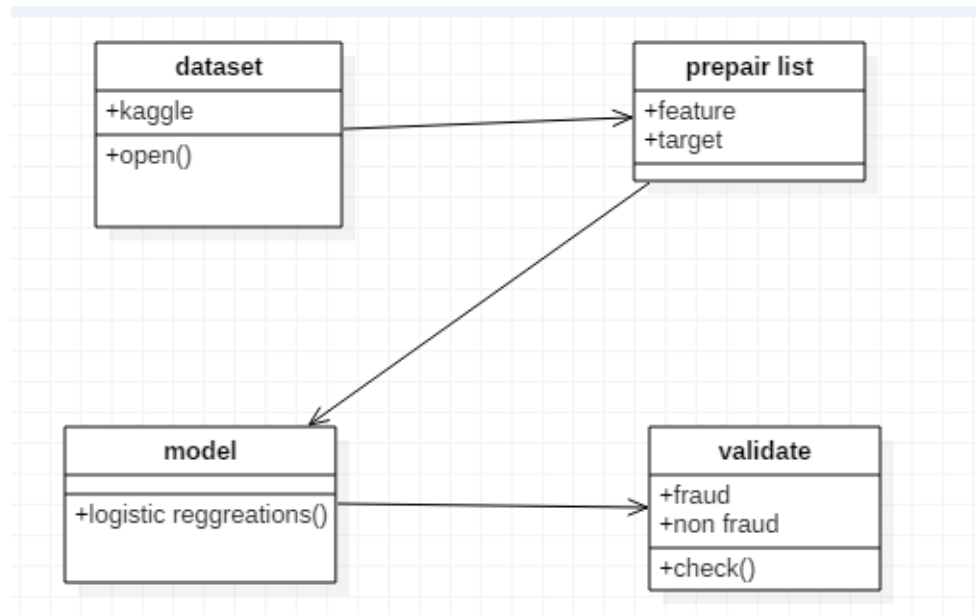


Fig 6.3.2.Class Diagram

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

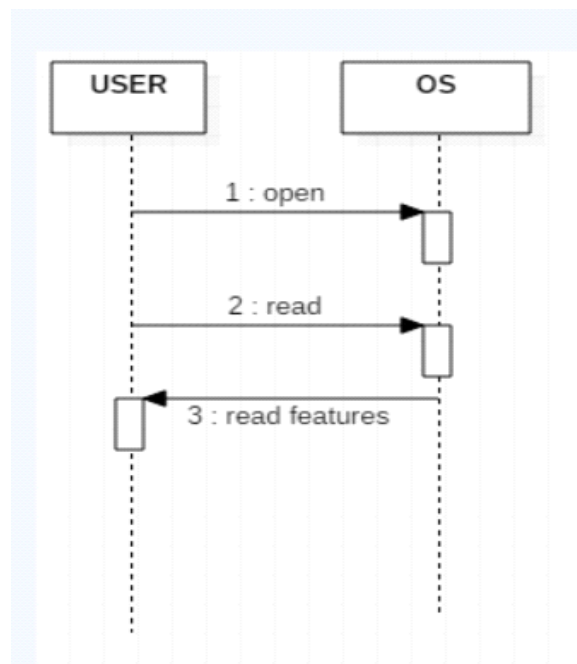


Fig 6.3.3. Sequence Diagram

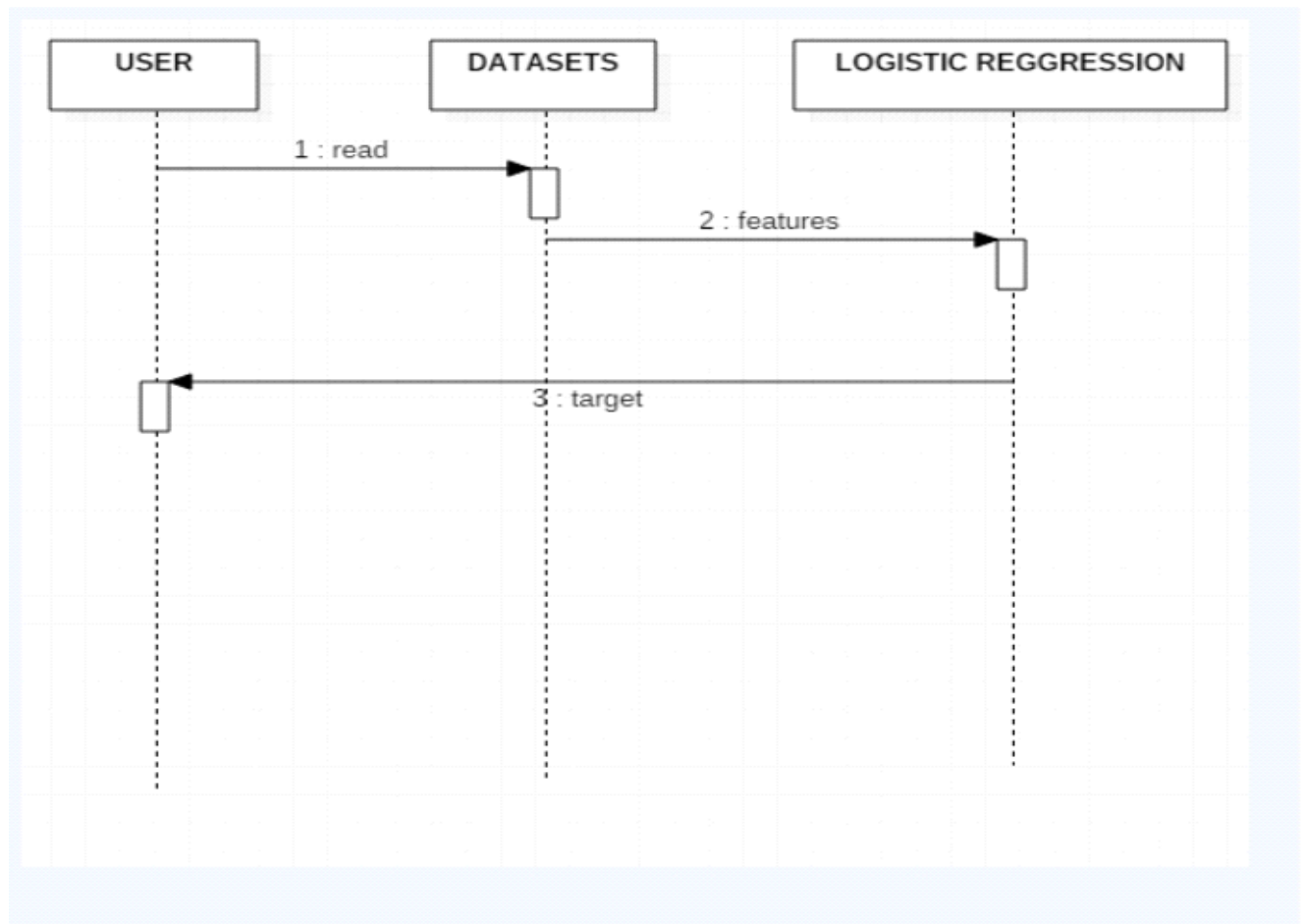


Fig 6.3.4 Sequence Diagram

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

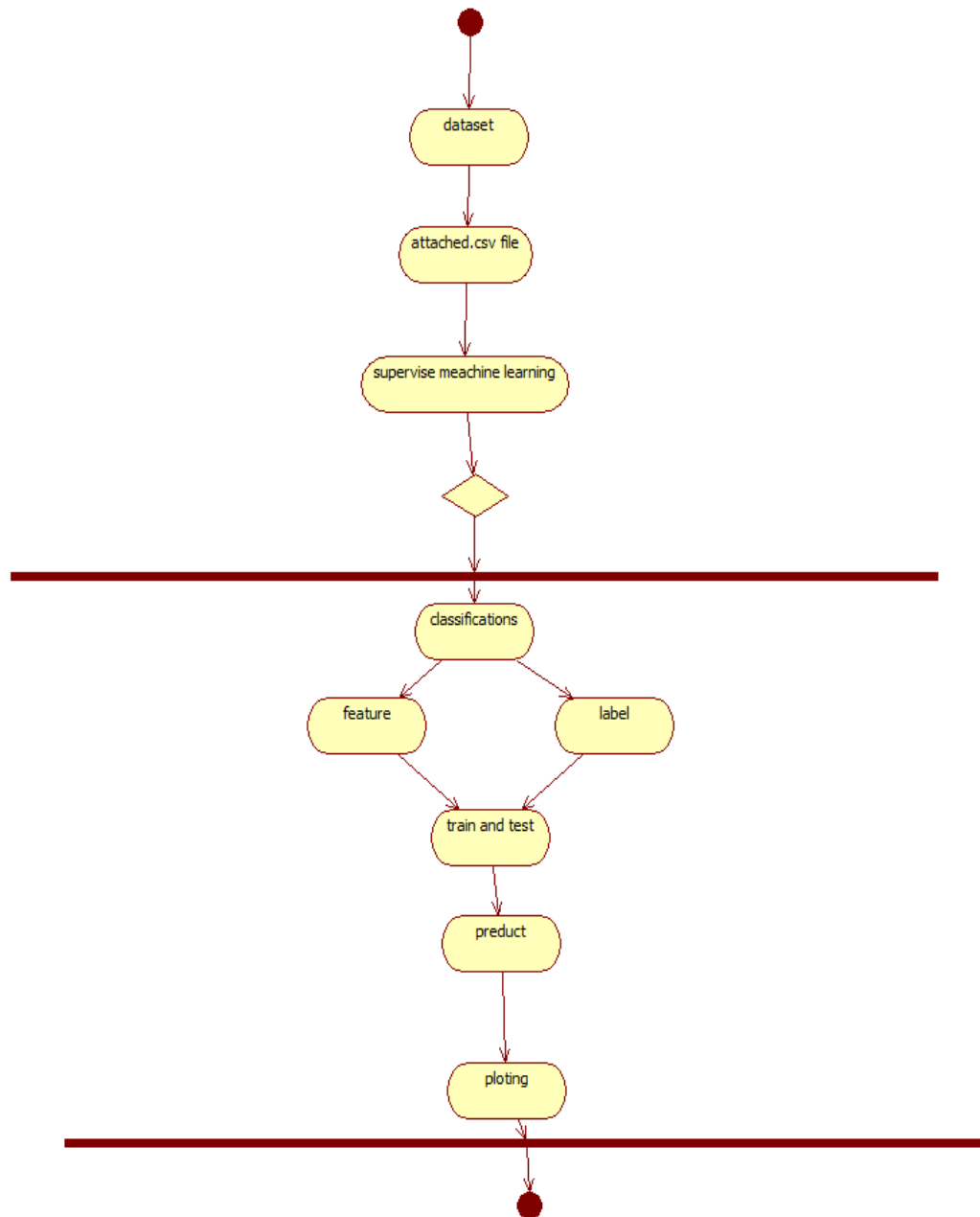


Fig 6.3.5.Activity Diagram

Involve data preprocessing steps, such as handling missing values, normalization, or scaling. Train the machine learning model using labeled data to learn patterns associated with fraudulent and then the non-fraudulent transactions. Display the final decision or action, such as approving or blocking the transaction.

Component diagram:

A component diagram illustrates the organization and relationships among software components in a system. Pandas represents the pandas library, which is commonly used for data manipulation and the analysis. The logistic regression algorithm, which is part of the machine learning model the scikit-learn library, which includes tools for machine learning tasks

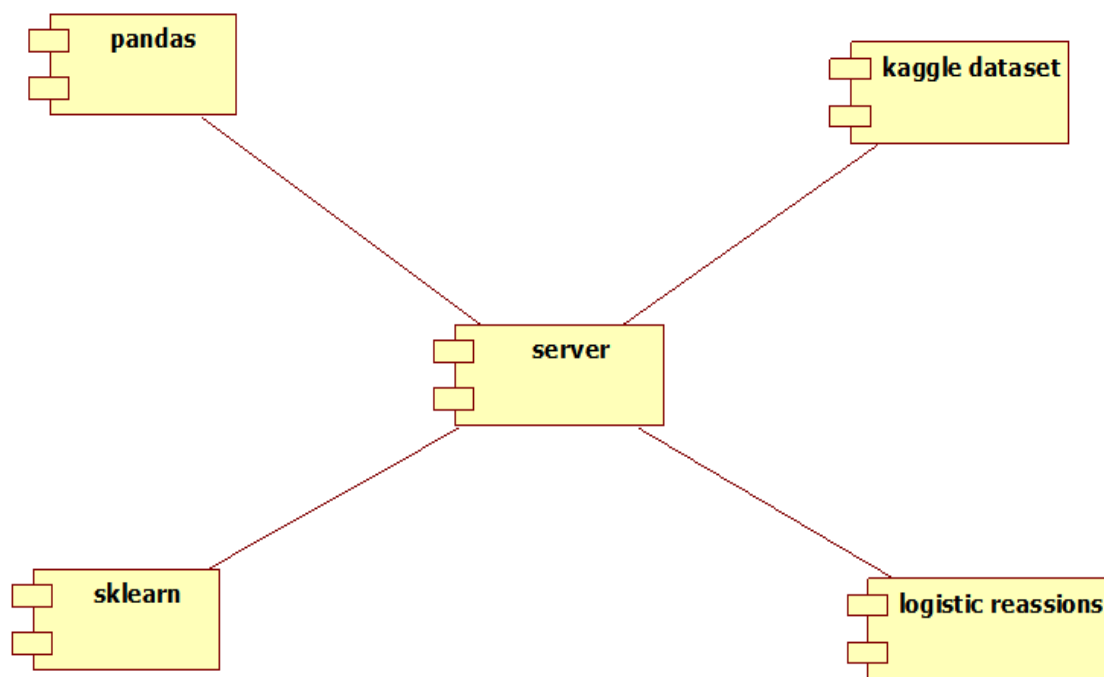


Fig 6.3.6. Component Diagram

It including logistic regression arrows that might indicate the flow of data from the Kaggle dataset to the Pandas component for preprocessing, then to the scikit-learn component for model training finally to the logistic regression component for the actual logistic regression algorithm.

Deployment diagram:

A deployment diagram visualizes the deployment of software components across different nodes in a system. Depicts the node where the pandas library is deployed, handling data manipulation and preprocessing. Represents the deployment of scikit-learn, the machine learning library, possibly on a server or cloud-based environment.

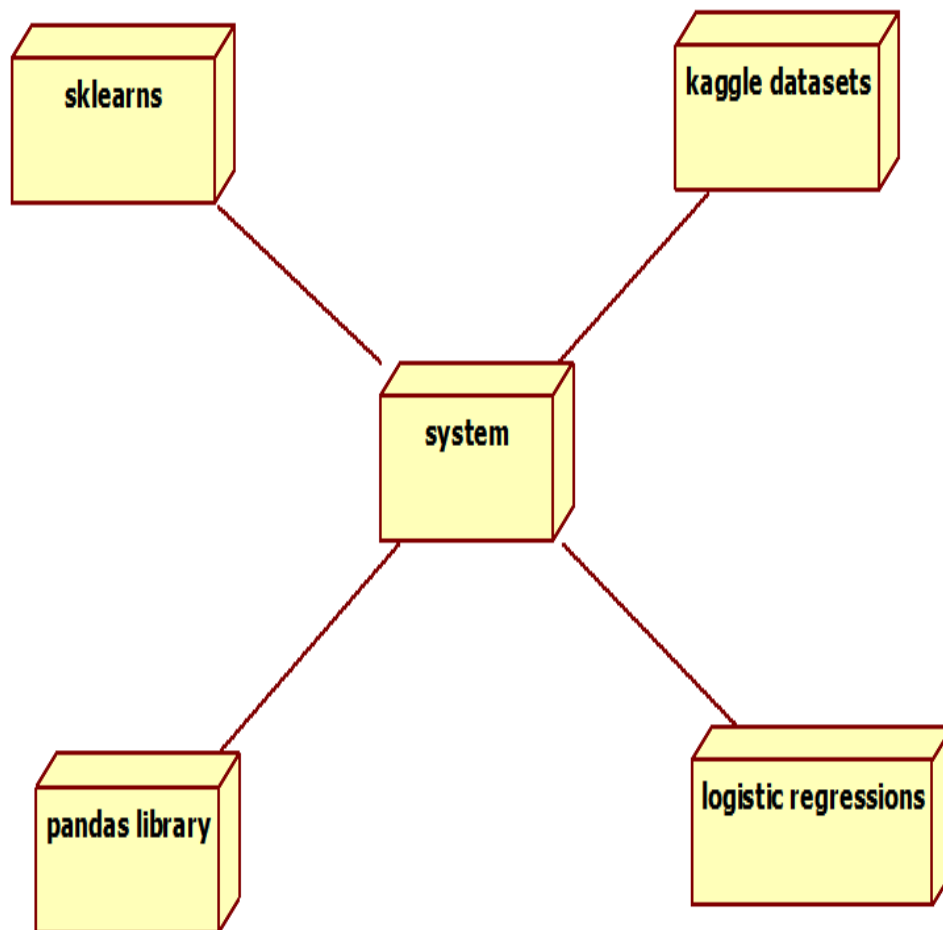


Fig 6.3.7 Deployment Diagram

Represents the overall system where the components interact, possibly a server or cloud-based platform. The system node represents the overall system where the components interact, possibly a server or cloud-based platform.

7. IMPLEMENTATION

IMPLEMENTATION

7.1 MODULES:

- **Numpy**
- **Pandas**
- **Matplotlib**
- **Seaborn**
- **Scipy**

MODULES DESCRIPTION:

1. Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

2. Pandas:

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

3. Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication-quality plots. Make interactive figures that can zoom, pan, and update.

4. Seaborn:

Python Seaborn library is a widely popular data visualization library that is commonly used for data science and machine learning tasks. You build it on top of the matplotlib data visualization library and can perform exploratory analysis. You can create interactive plots to answer questions about your data

5. Scipy:

SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats, and signal processing.

7.2 SAMPLE CODE

```
#importing the modules
import numpy as np
#import sklearn python machine learning modules
import sklearn as sk
#import pandas dataframes
import pandas as pd
#import matplotlib for plotting
import matplotlib.pyplot as plt
#import datasets and linear_model from sklearn module
from sklearn import datasets, linear_model
#import Polynomial features from sklearn module
from sklearn.preprocessing import PolynomialFeatures
#import train_test_split data classification
from sklearn.model_selection import train_test_split
#import ConfusionMatrix from pandas_ml
from pandas_ml import ConfusionMatrix
#reading the csv file from C:/Python27
dataframe = pd.read_csv('C:/Python27/creditcard.csv', low_memory=False)
#dataframe.sample Returns a random sample of items from an axis of object.
#The frac keyword argument specifies the fraction of rows to return in the random sample, so frac=1
means return all rows (in random order).
# If you wish to shuffle your dataframe in-place and reset the index
dataframe = dataframe.sample(frac=1).reset_index(drop=True)
#dataframe.head(n) returns a DataFrame holding the first n rows of dataframe.
dataframe.head()
print dataframe
```

8. SYSTEM TESTING

INTRODUCTION TO 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. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its 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.

TYPES OF TESTS

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they 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 satisfactory, 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.

Functional test:

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 are focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It 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.

White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.3 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.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9. INPUT DESIGN AND OUTPUT DESIGN

9.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maze of instant. Thus the objective of input design is to create an input layout that is easy to follow

9.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output

must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

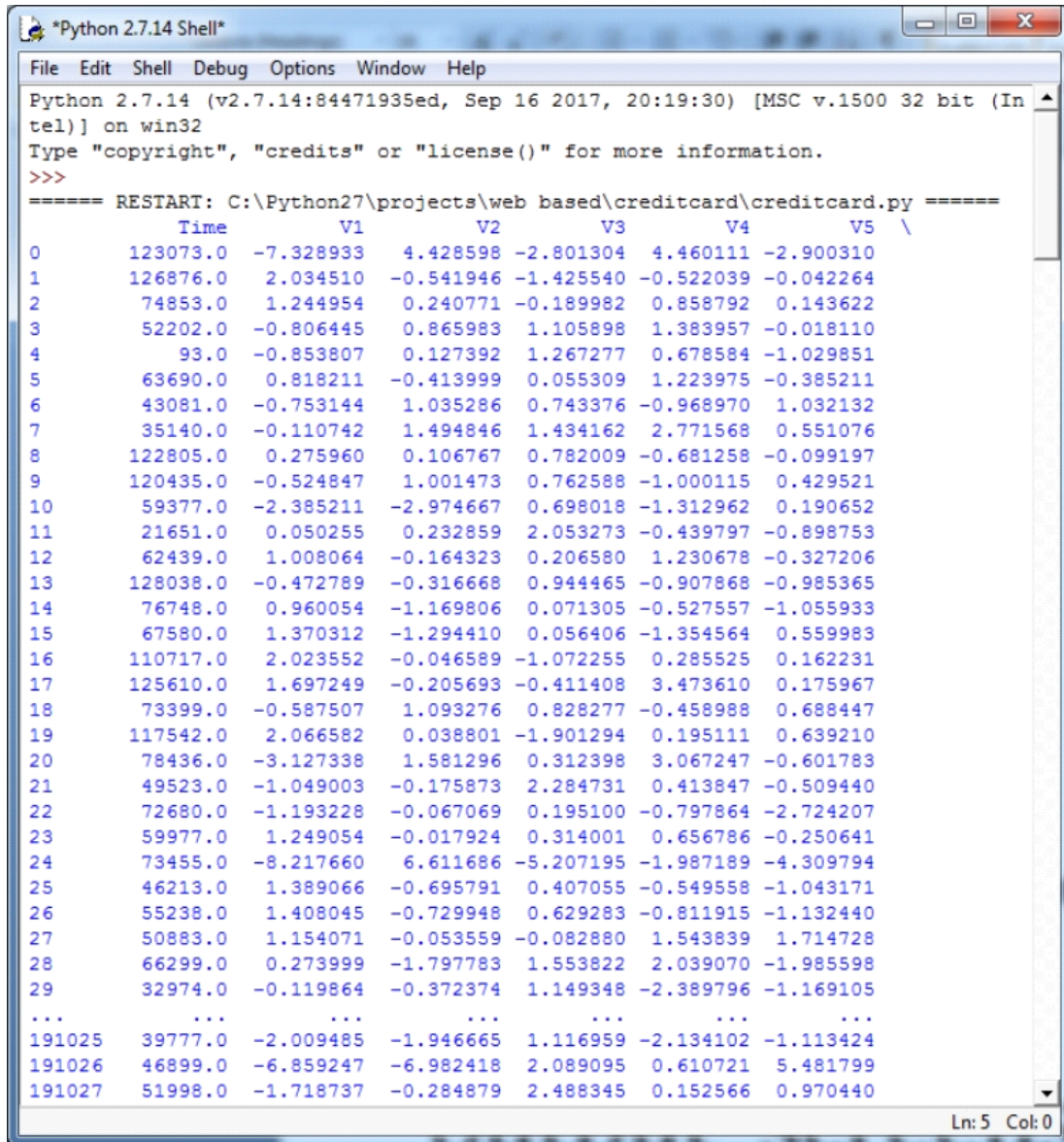
The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

10. SCREENSHOTS

SCREENSHOTS

step1:-



```
*Python 2.7.14 Shell*
File Edit Shell Debug Options Window Help
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:19:30) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python27\projects\web based\creditcard\creditcard.py =====
      Time      V1      V2      V3      V4      V5      \
0      123073.0    -7.328933    4.428598    -2.801304    4.460111    -2.900310
1      126876.0     2.034510    -0.541946    -1.425540    -0.522039    -0.042264
2       74853.0     1.244954     0.240771    -0.189982     0.858792     0.143622
3       52202.0    -0.806445     0.865983     1.105898     1.383957    -0.018110
4         93.0    -0.853807     0.127392     1.267277     0.678584    -1.029851
5      63690.0     0.818211    -0.413999     0.055309     1.223975    -0.385211
6      43081.0    -0.753144     1.035286     0.743376    -0.968970     1.032132
7      35140.0    -0.110742     1.494846     1.434162     2.771568     0.551076
8      122805.0     0.275960     0.106767     0.782009    -0.681258    -0.099197
9      120435.0    -0.524847     1.001473     0.762588    -1.000115     0.429521
10      59377.0    -2.385211    -2.974667     0.698018    -1.312962     0.190652
11      21651.0     0.050255     0.232859     2.053273    -0.439797    -0.898753
12      62439.0     1.008064    -0.164323     0.206580     1.230678    -0.327206
13      128038.0    -0.472789    -0.316668     0.944465    -0.907868    -0.985365
14       76748.0     0.960054    -1.169806     0.071305    -0.527557    -1.055933
15      67580.0     1.370312    -1.294410     0.056406    -1.354564     0.559983
16      110717.0     2.023552    -0.046589    -1.072255     0.285525     0.162231
17      125610.0     1.697249    -0.205693    -0.411408     3.473610     0.175967
18       73399.0    -0.587507     1.093276     0.828277    -0.458988     0.688447
19      117542.0     2.066582     0.038801    -1.901294     0.195111     0.639210
20       78436.0    -3.127338     1.581296     0.312398     3.067247    -0.601783
21      49523.0    -1.049003    -0.175873     2.284731     0.413847    -0.509440
22       72680.0    -1.193228    -0.067069     0.195100    -0.797864    -2.724207
23      59977.0     1.249054    -0.017924     0.314001     0.656786    -0.250641
24       73455.0    -8.217660     6.611686    -5.207195    -1.987189    -4.309794
25      46213.0     1.389066    -0.695791     0.407055    -0.549558    -1.043171
26      55238.0     1.408045    -0.729948     0.629283    -0.811915    -1.132440
27      50883.0     1.154071    -0.053559    -0.082880     1.543839     1.714728
28      66299.0     0.273999    -1.797783     1.553822     2.039070    -1.985598
29      32974.0    -0.119864    -0.372374     1.149348    -2.389796    -1.169105
...      ...      ...      ...      ...      ...      ...
191025   39777.0    -2.009485    -1.946665     1.116959    -2.134102    -1.113424
191026   46899.0    -6.859247    -6.982418     2.089095     0.610721     5.481799
191027   51998.0    -1.718737    -0.284879     2.488345     0.152566     0.970440
Ln: 5 Col: 0
```

Fig10.1.1 Screenshot of Transaction Details of Credit Cards

Python 2.7.14 Shell

	V6	V7	V8	V9	...	V21	V22	\
0	0.860848	-3.264272	3.387505	-0.678122	...	0.517120	0.793954	
1	-0.554310	-0.061190	-0.213010	1.208486	...	0.058205	0.400362	
2	-0.379393	0.188256	-0.025890	-0.156274	...	-0.032685	-0.162469	
3	0.140882	0.124651	0.603672	-0.853074	...	0.173074	0.494319	
4	-0.487614	1.836071	-0.298566	-0.922127	...	0.252358	0.179725	
5	-0.312591	0.242221	-0.042756	-0.159928	...	0.215567	0.135244	
6	0.179043	0.630513	0.217137	-0.305976	...	-0.313708	-1.005632	
7	-0.323673	1.150141	-0.406856	-1.607967	...	0.014948	0.417099	
8	0.601896	-0.550635	-0.536688	0.636462	...	0.880972	0.116895	
9	-0.280322	0.540743	-0.039551	1.038476	...	-0.358302	-0.723072	
10	0.660131	-1.525946	1.218576	-0.356994	...	0.349349	0.452242	
11	0.138100	-0.888775	-0.707806	2.586276	...	0.591876	-0.300709	
12	-0.395308	0.209822	-0.126912	0.344534	...	-0.169161	-0.654151	
13	1.270817	-0.501582	0.321978	-2.425153	...	-0.481924	-0.646595	
14	-0.601747	-0.234903	-0.141296	-0.815393	...	0.160223	-0.113911	
15	4.381076	-2.136257	1.219148	0.574333	...	0.085114	0.463230	
16	-0.137493	-0.531162	-0.076383	2.537730	...	-0.449375	-0.935900	
17	1.339410	-0.631000	0.466104	-0.399152	...	0.232849	0.316123	
18	0.358313	0.578649	0.194290	-0.044353	...	-0.328960	-0.797696	
19	-0.297948	0.017728	-0.013159	0.387878	...	-0.354218	-0.986282	
20	1.656185	-1.824034	-1.160688	-1.683612	...	-1.468608	-0.439224	
21	1.612317	0.483134	0.398399	1.038750	...	0.042743	0.218921	
22	2.082131	2.237805	0.279369	0.186022	...	-0.077393	-0.077975	
23	-0.167691	-0.104377	-0.051658	0.578203	...	-0.342899	-0.850398	
24	-1.033386	-4.195913	5.674681	0.886528	...	-0.103259	-0.815974	
25	-0.489942	-0.687088	-0.056695	-0.222781	...	-0.120725	-0.340315	
26	-0.256349	-1.033760	-0.008066	-0.321571	...	0.332826	0.885971	
27	4.415031	-1.119840	1.160933	-0.022263	...	-0.131255	-0.407110	
28	0.769695	-0.697361	0.273927	1.894263	...	0.336687	0.565503	
29	0.602393	-1.920177	-2.476358	-2.671414	...	-1.559949	-0.245757	
...	
191025	-0.687275	-0.854937	0.617364	-2.386114	...	0.315558	0.383476	
191026	-3.018185	-2.479407	0.291017	0.663596	...	-0.330619	0.044042	
191027	-0.190548	-0.605851	0.387687	0.492221	...	-0.010143	0.042408	
191028	-0.398494	-0.663356	0.085062	1.307200	...	-0.226139	-0.482545	
191029	-1.047888	-1.525645	1.096137	0.810332	...	-0.530157	-0.515465	
191030	-0.932457	-0.141720	0.551792	0.143824	...	0.273434	0.490586	
191031	-0.802823	-0.599786	-0.095845	3.329477	...	-0.048467	0.347211	
191032	0.653854	0.148550	0.030807	0.733073	...	0.288138	0.827187	

Ln: 5 Col: 0

Fig10.1.2.Screenshot of Transaction Details of Credit Cards

```
*Python 2.7.14 Shell*
File Edit Shell Debug Options Window Help
191004 -0.001250 0.002000 0.000000 -0.400001 0.010107 0.010100 0.90
...
Class
0 0
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
29 0
...
191025 0
191026 0
191027 0
191028 0
191029 0
191030 0
191031 0
Ln: 5 Col: 0
```

Fig10.1.3 Screenshot of Classification of Fraud and Non Fraud

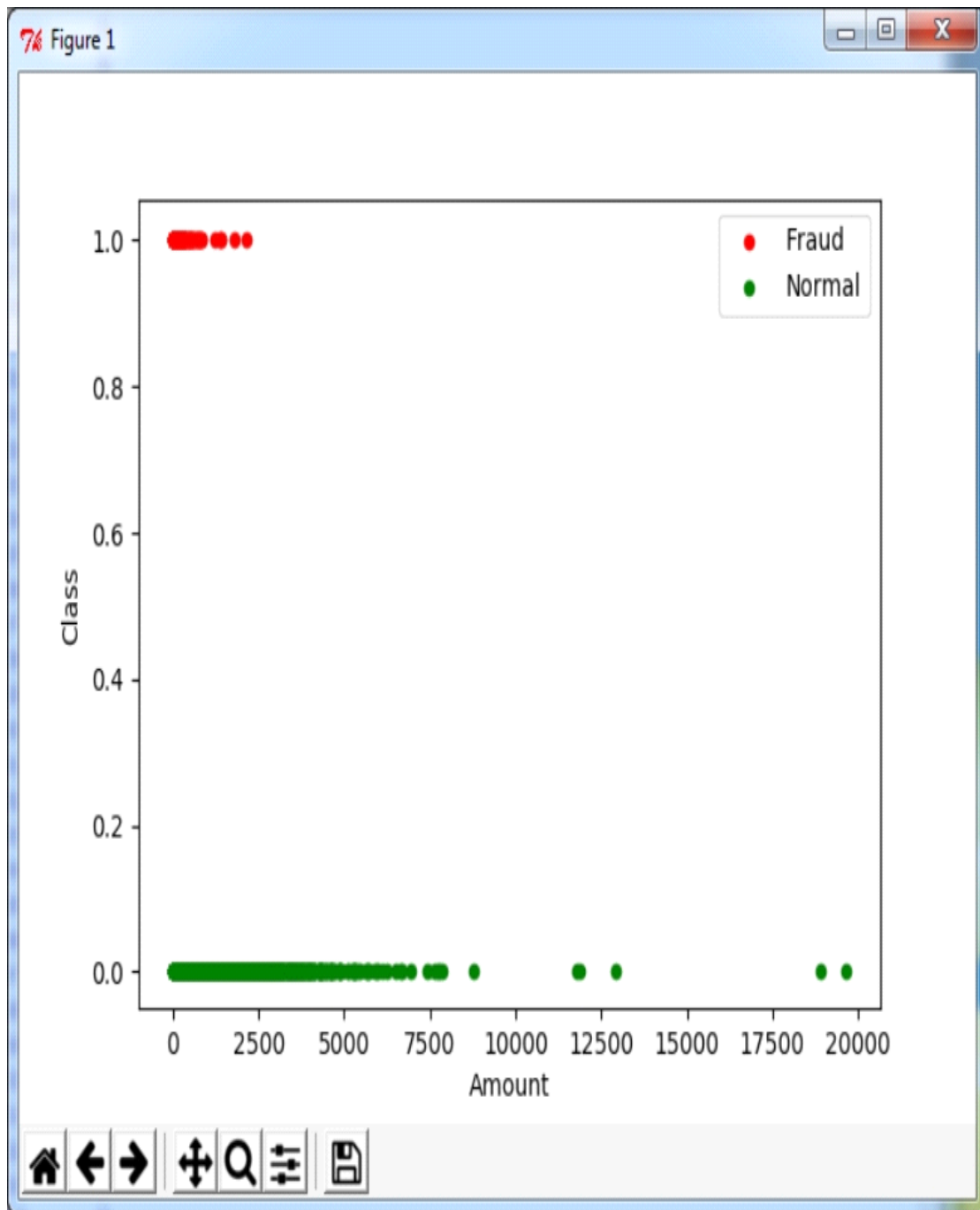
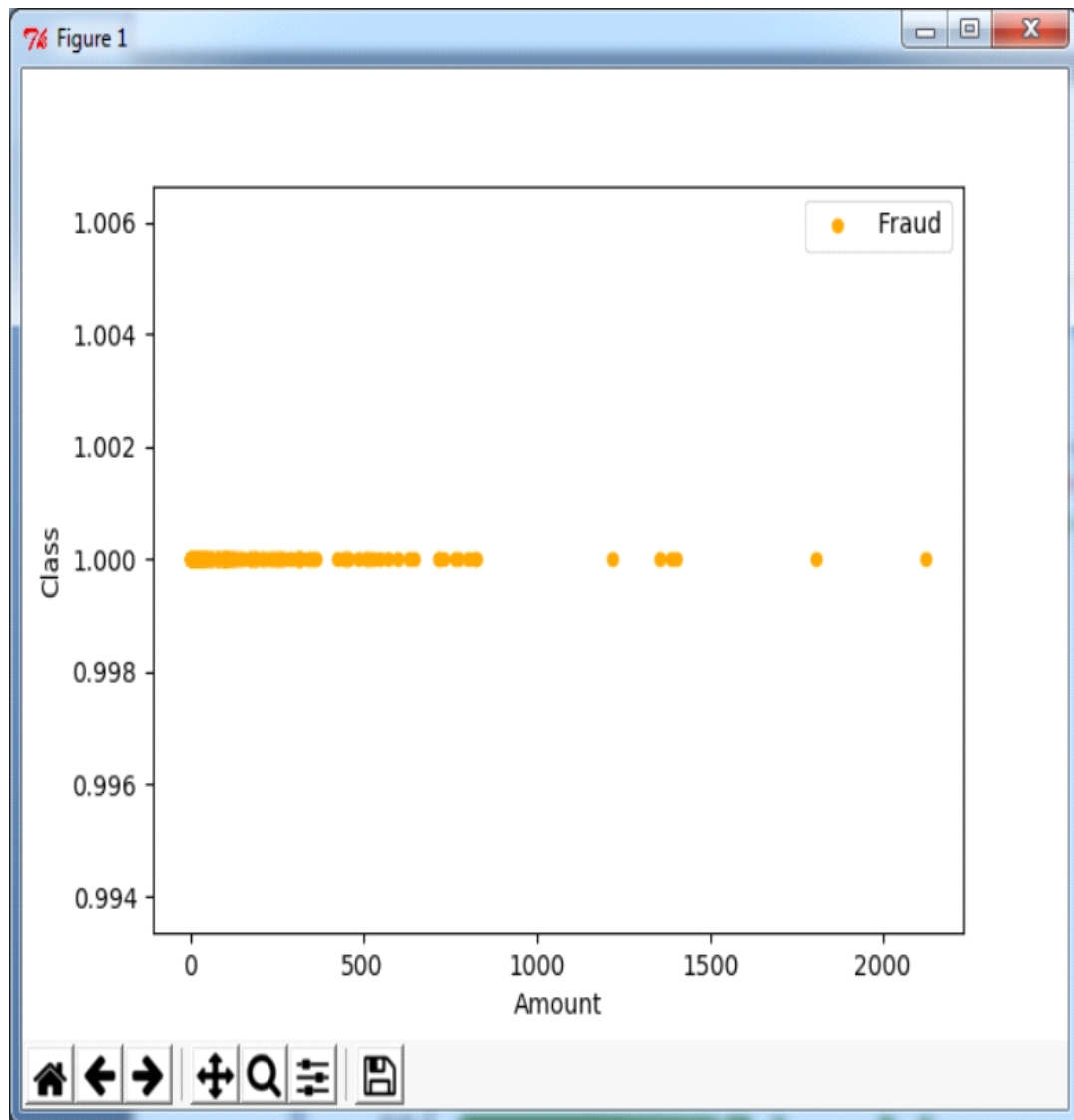


Fig 10.2.Graph of detecting fraud and normal transactions

Step 2:-



Step 3:-

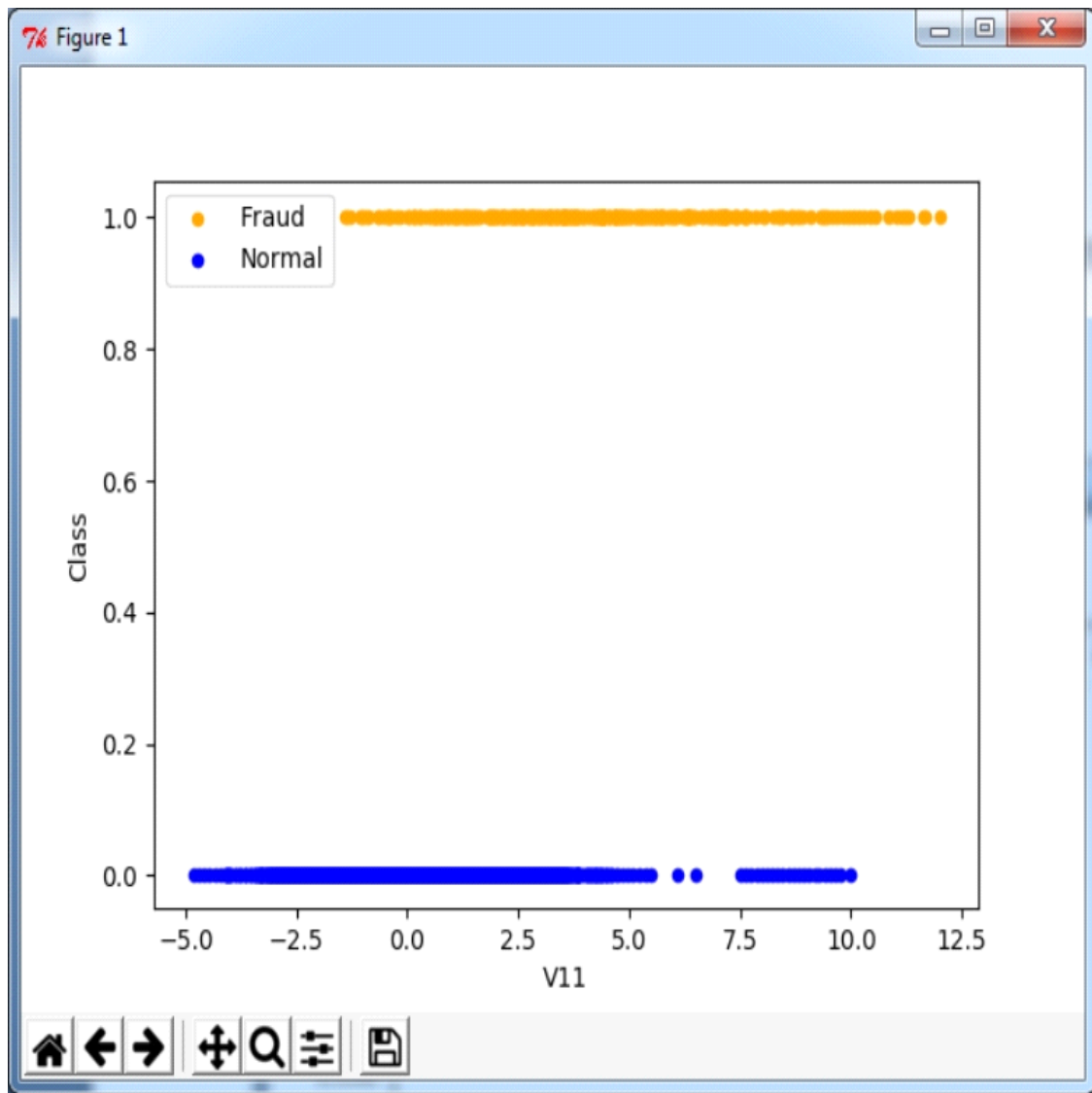


Fig 10.4.Graph of detecting fraud and normal transactions

Step 4:-

Finally got confusion metrics and high accuracy

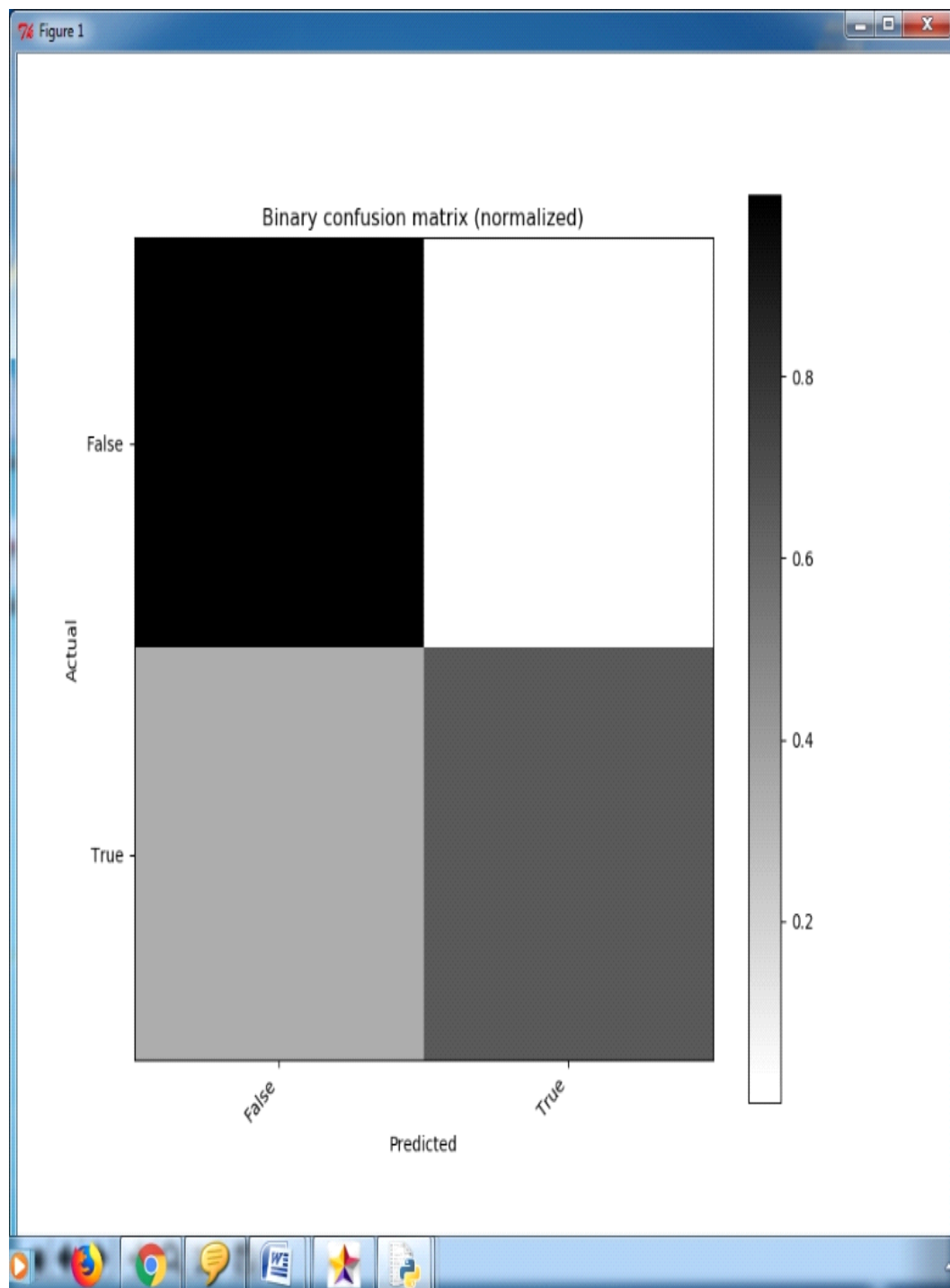


Fig.10.5 Confusion Matrix

```
Python 2.7.14 Shell
File Edit Shell Debug Options Window Help

data class points.')
This Feature what is mentioned is based on the class Distribution.
('X and y sizes, respectively:', 191055, 191055)
('Train and test sizes, respectively:', 124185, 124185, '|', 66870, 66870)
('Total number of frauds:', 372, 0)
('Number of frauds on y_test:', 140, 0)
('Number of frauds on y_train:', 232, 0)
('Score: ', 0.9990728278749813)
('Confusion matrix:', Predicted False True __all__
Actual
False      66716      14      66730
True        48       92       140
__all__    66764     106     66870)
population: 66870
P: 140
N: 66730
PositiveTest: 106
NegativeTest: 66764
TP: 92
TN: 66716
FP: 14
FN: 48
TPR: 0.6571428571428571
TNR: 0.9997901993106548
PPV: 0.8679245283018868
NPV: 0.9992810496674854
FPR: 0.00020980068934512214
FDR: 0.1320754716981132
FNR: 0.34285714285714286
ACC: 0.9990728278749813
F1_score: 0.7479674796747967
MCC: 0.7547820949843432
informedness: 0.6569330564535121
markedness: 0.8672055779693721
prevalence: 0.0020936144758486614
LRP: 3132.224489795918
LRN: 0.34292908961654095
DOR: 9133.738095238094
FOR: 0.0007189503325145288
>>> |
```

Ln: 297 Col: 4

Fig.10.6.Final Output

11. FUTURE ENHANCEMENT

FUTURE ENHANCEMENT

The prediction model predicts continuous valued functions. We have to detect 148 may be fraud and other are genuine. In decision tree generate a tree with root node, decision node and leaf nodes. The leaf node may be 1 becomes fraud and 0 otherwise. Logistic Regression is same as linear regression but interpret curve is different. To generalize the linear regression model, when dependant variable is categorical and analyzes the relationship between multiple independent variables.

12. CONCLUSION

CONCLUSION

This process is used to detect credit card transactions, which are fraudulent or genuine. Data mining techniques of Predictive modeling, Decision trees, and Logistic Regression are used to predict the fraudulent or genuine credit card transactions. In predictive modeling to detect and check output class distribution. The prediction model predicts continuous valued functions. We have to detect that 148 may be fraud and that other are genuine. In decision tree generate a tree with root node, decision node, and leaf nodes. The leaf node may be 1 becomes fraud and 0 otherwise. Logistic Regression is the same as linear regression but interpret curve is different. To generalize the linear regression model, when the dependant variable is categorical and analyzes the relationship between multiple independent variables.

13. BIBLIOGRAPHY

BIBLIOGRAPHY

References

- [1] Salazar, Addisson, et al. "Automatic credit card fraud detection based on non-linear signal processing." *Security Technology (ICCST), 2012 IEEE International Carnahan Conference on*. IEEE, 2012.
- [2] Delamaire, Linda, H. A. H. Abdou, and John Pointon. "Credit card fraud and detection techniques: a review." *Banks and Bank systems* 4.2 (2009): 57-68.
- [3] Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1.1 (1986): 81-106.
- [4] Quinlan, J. R. (1987). "Simplifying decision trees". *International Journal of Man-Machine Studies*. 27 (3): 221. doi:10.1016/S0020-7373(87)80053-6.
- [5] K. Karimi and H.J. Hamilton (2011), "Generation and Interpretation of Temporal Decision Rules", *International Journal of Computer Information Systems and Industrial Management Applications*, Volume 3.
- [6] Aggarwal, Charu C. "Outlier analysis." *Data mining*. Springer International Publishing, 2015.
- [7] Salazar, Addisson, Gonzalo Safont, and Luis Vergara. "Surrogate techniques for testing fraud detection algorithms in credit card operations." *Security Technology (ICCST), 2014 International Carnahan Conference on*. IEEE, 2014.
- [8] Ogwueleka, Francisca Nonyelum. "Data mining application in credit card fraud detection system." *Journal of Engineering Science and Technology* 6.3 (2011): 311-322.
- [9] Jiang, Changjun et al. "Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism." *IEEE Internet of Things Journal* 5 (2018): 3637-3647.
- [10] Pumsirirat, A. and Yan, L. (2018). Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine. *International Journal of Advanced Computer Science and Applications*, 9(1).
- [11] Mohammed, Emad, and Behrouz Far. "Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study." *IEEE Annals of the History of Computing*, IEEE, 1 July 2018,

doi.ieeecomputersociety.org/10.1109/IRI.2018.00025.

- [12]Randhawa, Kuldeep, et al. “Credit Card Fraud Detection Using AdaBoost and Majority Voting.” IEEE Access, vol. 6, 2018, pp. 14277–14284., doi:10.1109/access.2018.2806420.
- [13]Roy, Abhimanyu, et al. “Deep Learning Detecting Fraud in Credit Card Transactions.” 2018 Systems and Information Engineering Design Symposium (SIEDS), 2018, doi:10.1109/sieds.2018.8374722.
- [14]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.
- [15]Awoyemi, John O., et al. “Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis.” 2017 International Conference on Computing Networking and Informatics (ICCNI), 2017,doi:10.1109/iccni.2017.8123782.
- [16]Melo-Acosta, German E., et al. “Fraud Detection in Big Data Using Supervised and Semi-Supervised Learning Techniques.” 2017 IEEE Colombian Conference on Communications and Computing (COLCOM), 2017,doi:10.1109/colcomcon.2017.8088206.
- [17] Quah, J. T. S., and Sriganesh, M. (2020). Real-time credit card fraud detection using computational intelligence. Expert Systems with Applications, 35(4), 1721-1732.
- [18] Gupta, Shalini, and R. Johari. ”A New Framework for Credit Card Transactions Involving Mutual Authentication between Cardholder and Merchant.” International Conference on Communication Systems and Network Technologies IEEE, 2021:22-26.
- [19] Y. Gmbh and K. G. Co, “Global online payment methods: the Full year 2020,” Tech. Rep., 3 2020.
- [20] Alenzi, H. Z., & Aljehane, N. O. (2020). Fraud detection in credit cards using logistic regression. International Journal of Advanced Computer Science and Applications