CHAPTER-1

INTRODUCTION

Credit cards are most widely used in onsite payment processes and card-not-present transactions such as internet or online shopping. With their rising popularity of online payment transactions, credit card transactions are getting highly vulnerable to threats called credit card frauds such as eavesdropping, phishing, intrusion, denial-of-service (DoS), database stealing, etc.

Credit card frauds are classified into three categories which are card related frauds, merchant related frauds and Internet related frauds. Card related frauds include frauds such as use of lost or stolen cards and fake or counterfeit cards, shoulder surfing, account takeover etc. Merchant related frauds include merchant collusion where a person at merchant's side steals credit card information of their customers. But the most severe of all the frauds are Internet related frauds which are made during online transactions such as site-cloning, creating false merchant sites, snooping, man-in-the-middle attack [MITM], and denial-of-service. Traditionally used AVS (Address Verification System), CVM (Card Verification Method) and other security systems have proven insecure against these attacks and therefore consumers cannot fully utilize the credit card transaction system.

Credit card fraud can be defined as the illegal use of any system or criminal activity. Increase in e-commerce and the ease of online transactions and payments has led to an exponential increase in the number of people opting for online purchases. The most common method of payment for online purchase is credit cards. Credit-card-based purchases can be categorized into two types: 1) physical card and 2) virtual card. When the user is present physically with the card during the transaction, it is categorized as Physical purchasing. Especially all credit card operations are performed by web payment gateways, e.g., PayPal and Alipay.

In real life, fraudulent transaction is scattered with genuine transactions and simple pattern matching Techniques are not often sufficient to detect those frauds accurately. Outlier detection is a data mining technique commonly used for fraud detection. Anomaly based methods in which the analyzed spending pattern of each cardholder makes up the

cardholder profile. Any incoming transaction that is inconsistent with the cardholder's profile would be considered suspicious.

Every card holder having a unique pattern contains information about number of transactions, details of purchased items, merchant information, date of transaction etc. It will be the most effective method to counter fraud transaction through internet. If any deviation is noticed from the available patterns of the card holder, then it will generate an alarm to the system to stop the transaction. The aim of the fraud detection system is to detect fraud accurately and before fraud is committed. The goal is to detect least and accurate false fraud detection.

First, totally order the attributes of transaction records, and then classify the values of every attribute. Based on them, we construct a logical graph of BP (LGBP) which abstracts and covers all different transaction records. Based on LGBP, we define the path-based transition probability and diversity coefficient to characterize users' transaction behaviors and diversity. Also define a state transition probability matrix to capture temporal features of transactions, and then construct a BP for each user. A BP-based fraud detection method is proposed to determine the legality of an incoming transaction, and it considers the concept drift problem.

Outliers are used to detect fraud. While in supervised method, the models are used to differentiate between fraudulent and non-fraudulent behavior to obtain the outlier. Clustering has an application in the field of engineering and scientific disciplines like psychology, biology, medicine, computer vision, communication, and remote sensing. A set of patterns is observed by abstracting underlying structure in clustering. The patterns are clustered based on more similar features than another pattern of group. Various clustering algorithms have been proposed to fulfill different requirements. Clustering algorithms are based on the structure of abstraction and are classified into hierarchical and partition algorithms. Catalyst optimization algorithm is used to create physical plans. A catalyst is a tree composed of node objects.

CHAPTER-2
LITERATURE SURVEY

The work developed by R. Brause, T. Langsdorf, and M. Hepp, the prevention of credit card fraud is an important application for prediction techniques. One major obstacle for using neural network training techniques is the high necessary diagnostic quality: Since only one financial transaction of a thousand is invalid no prediction success less than 99.9% is acceptable. Due to these credit card transaction proportions completely, new concepts had to be developed and tested on real credit card data. This paper shows how advanced data mining techniques and neural network algorithms can be combined successfully to obtain high fraud coverage combined with a low false alarm rate.

The work developed by R. C. Chen, S. T. Luo, X. Liang, and V. C. S. Lee, Online shopping and banking has increased by the growth of internet and by use of credit card. Along with this, the number of credit card frauds has also increased. Many modern techniques based on Artificial Intelligence; Data warehousing has evolved in detecting various credit card fraudulent transactions. We proposed a system which detects fraud in credit card transaction processing using a decision tree with combination of Luhn's algorithm and Hunt's algorithm. Luhn’s algorithm is used to validate the card number. Address matching rule checks whether the Billing Address and Shipping Address match or not. This check does not guarantee whether a transaction is fraud or genuine. But if the two addresses match, the transaction can be classified as genuine with a high probability. Else, the transaction is labelled as suspect. A customer usually carries out similar types of transactions in terms of amount, which can be visualized as part of a cluster. Since a fraudster is likely to differ from the customer’s account, his transactions can be detected as exceptions to the cluster – a process known as outlier detection.

The work developed by C. Cortes and D. Pregibon Data mining technology is applied to fraud detection to establish the fraud detection model, describe the process of creating the fraud detection model, then establish data model with ID3 decision tree, and establish example of fraud detection model by using this model. As e-commerce sales continue to grow, the associated online fraud remains an attractive source of revenue for

fraudsters. These fraudulent activities impose a considerable financial loss to merchants, making online fraud detection a necessity. The problem of fraud detection is concerned with not only capturing the fraudulent activities, but also capturing them as quickly as possible. This timeliness is crucial to decrease financial losses.

V. Dheepa and R. Dhanapal Along with the great increase of internet and e-commerce, the use of credit cards is an unavoidable one. Due to the increase of credit card usage, the frauds associated with this have also increased. There are a lot of approaches used to detect frauds. In this paper, behavior-based classification approach using Support Vector Machines are employed and efficient feature extraction method also adopted. If any discrepancies occur in the behavior's transaction pattern, then it is predicted as suspicious and taken for further consideration to find the frauds. Generally, credit card fraud detection problems suffer from a large amount of data, which is rectified by the proposed method. Achieving finest accuracy, high fraud catching rate and low false alarms are the main tasks of this approach.

The work developed by S. Gupta and R. Johari, Electronic Commerce (e-Commerce) and ease in the onsite transactions have led to the exponential growth in the acceptance of credit cards among consumers of all the sections. But despite their remarkable advantages, consumers are still reluctant in their use, especially for online transactions and the reason being the increasing credit card fraud rate. Several security models have been proposed and deployed for secure online transactions but the sharing of sensitive credit card data over the Internet has made online transactions vulnerable to threats. In this paper, we discuss and analyze the current developments in online authentication procedures including biometrics, one-time-password systems and use of mobile devices and Public Switched Telephone Network for cardholder authentication. Then we propose a completely new framework for both onsite and online (Internet shopping) credit card transactions. This framework is more secure, robust, enhances user privacy and does not involve the deployment of special hardware systems at the customer's site.

The work developed by N. Abdelhamid, A. Ayesh, and F. Thabtah Phishing is serious web security problem that involves mimicking legitimate websites to deceive online

users to steal their sensitive information. Phishing can be seen as a typical classification problem in data mining where the classifier is constructed from many website's features. There are high demands on identifying the best set of features that when mined the predictive accuracy of the classifiers is enhanced. Compare two known features selection method to determine the least set of features of phishing detection using data mining.

The work developed by Although credit card fraud detection has been studied for many years, these detection models cannot effectively help financial experts handle fraud alerts since they only predict a risk degree for a transaction but are unable to provide any information to explain why the transaction is of this risk degree. This paper presents a Kernel-based Supervised Hashing (KSH) model to detect credit card fraud. KSH is based on the idea of an approximate nearest neighbor that can provide the most similar existing fraud samples for a transaction when the transaction is predicted to be fraud. These similar samples can help experts analyze the transaction, improve the detection accuracy, and lower the disturbing rate. Additionally, KSH is very suitable for the large and high-dimension dataset. To the best of our knowledge, it is the first time that KSH has been used to detect credit card fraud, and our experiments on a real large transaction dataset illustrate its advantages and effectiveness.

CHAPTER-3

PROBLEM AND SOLUTION OF THE PROJRCT

3.1 EXISTING SYSTEM:

- The volume of electronic transactions has risen significantly in recent years due to the popularization of online shopping (e.g., Amazon, eBay).
- A physical card is not required in the scenario of online shopping and only the information of the card is enough for a transaction. Therefore, it is much easier for a fraudster to commit a fraud.
- Various supervised learning methods like neural networks, decision trees, logistic regression, and support vector machines are often used to obtain the fraud patterns.
- Fraud Detecting Software are existing, but they become unstable and require a lot of data to be processed.
- There are ways in which the existing system built over supervised learning can be trained in wrong ways.
- KNN's speed was low with a medium accuracy and scored one of the lower scores. As for the cost all models built were expansive.
- Logistic regression performed poorly as the accuracy.
- The system can be trained that a fraud Credit Card holder as a true holder.
- That's where the Spark tool of splitting the Data Frames and processing it becomes easy over the existing one.

DISADVANTAGES:

- Loss track of transaction.
- Unsuitable data or transaction.
- Difficult to recover from the problem.
- Low Accuracy.
- In efficient performance due to low dimensional data.

3.2 PROPOSED SYSTEM:

- The popularization of online shopping transaction fraud is growing seriously.
- First, we totally order the attributes of transaction records, and then classify the values of every attribute.
- Behavior Profiles based on their transaction records, which is used to detect transaction fraud in the online shopping scenario.
- **Logical graph of** Behavior Profiles which is a total order-based model to represent the logical relation attributes of transaction records.
- We compare OM with other two anomaly detection methods: Bayesian learning-based fraud detection and self-organizing maps-based fraud detection. The two methods are called phase Modulation (PM).
- Transactions labeled as fraudulent. True negative (TN)is the total number of legal transactions labeled as legal. False negative (FN) is the total number of fraud transactions which are not detected.
- OM automatically classifies the values of transaction attributes so that our model can characterize the user's personalized behavior more precisely.
- Detection of the fraudulent transactions will be made by using supervised and unsupervised machine learning techniques IForest, SVM and Gradient Boost those models will be used on a credit card transaction dataset.
- Outlier detection the computation and memory required for the credit card fraud detection is much less in addition to its working faster and better in online large datasets. But their work and results showed that IForest was more accurate and efficient.

ADVANTAGES:

- Frequently Access and Used Data.
- More efficient result for detecting Fraud Detection discovery.
- High accuracy.
- Less detection time.
- Get user-based historical records.
- Avoid complex problems.

CHAPTER-4

SYSTEM STUDY

4.1 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:

1. Economical feasibility
2. Technical feasibility
3. Social feasibility

4.1.1Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system is 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.

4.1.2 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 demand for 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.

4.1.3 Social Feasibility

The aspect of 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.

CHAPTER-5

SYSTEM REQUIREMENTS AND ANALYSIS

5.1 SYSTEM REQUIREMENTS

The system requirement is a main part in the analyzing phase of the project. The of the project must properly analyze the hardware and the software requirements, otherwise in future the project designer will face more trouble with the hardware and software required. Below specified are the project hardware and software requirements.

5.1.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by the software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

- System : Pentium dual core
- Hard disk : 120 GB
- RAM : 1GB

5.1.1 SOFTWARE REQUIREMENTS

The software requirements are the software specification of the system. It should include both a definition and a specification of a requirements. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's progress throughout the development activity.

- Operating system : Windows
- Programming Language : Python
- Tool : Anaconda
- Database : TensorFlow, Keras

5.2 SYSTEM ANALYSIS

5.2.1 PYTHON

Python features a dynamic type of system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional, and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. C, Python the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of Python's other implementations. Python and C are managed by the non-profit Python software foundation. Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming. Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle detecting garbage collector for memory management. It also features dynamic name resolution, which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has filter( ), map( ) and reduce( ) functions; list comprehensions, dictionaries and sets; and generator expressions. The standard library has two modules that implement functional tools borrowed from Haskell and Standard ML.

5.2.2 KERAS

Kerasisan Opensource Neural Network library written in Python that runs on top of Theano or TensorFlow. It is designed to be modular, fast, and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Kerasis high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or setup multiple input-output models. In this level Kerasal compiles our model with loss and

optimizes functions training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

5.2.1 ANACONDA

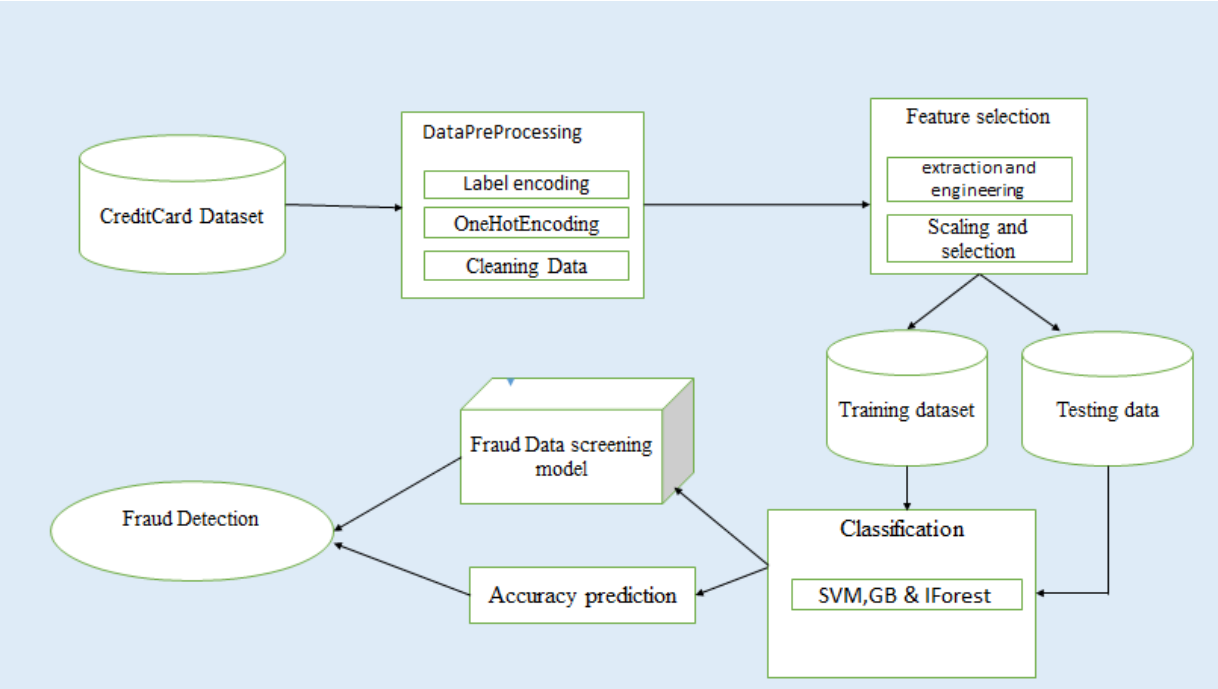
Anaconda is a free and open-source distribution of the Python programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the system Canada. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda is a Python-based data processing and scientific computing platform. It has built in many very useful third-party libraries. Installing Anaconda is equivalent to automatically installing Python and some commonly used libraries such as NumPy, Pandas, Scrip, and Matplotlib, so it makes the installation so much easier than regular Python installation. It is painful and you need to consider compatibility, thus it is highly recommended to directly install Anaconda.

CHAPTER-6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE



System Architecture

6.2 MODULES

1. Data Collection
2. Pre-Processing
3. Feature Extraction
4. Splitting Data
5. Classification
6. Accuracy prediction

6.2.1 Data Collection

The outer overall functionality of the proposed system applications this indicates that have collected the credit card dataset from various source and analysis and processing of data have done by predicting the farmland and classifying the rate of accuracy from this information by plotting the graph credit card fraud is predicted.

6.2.2 Pre-Processing

The data is cleaned before loading to the classifier. Import the Label Encoder class from the sclera library, fit and transform the first column of the data, and then replace the existing text data with the new encoded data. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. This work fitting and transforming Standard Scaler method on train data.

6.2.3 Feature Extraction

Feature extraction by Layering of data. For feature extraction, first select the data and process data for original feature. Then reduce the feature by transforming refining data. This can be done by selecting the data by CSV file and processing the data for original feature and it was transformed and refining data to reduced feature, finally it predicts the accuracy of credit card.

6.2.4 Splitting Data

The Splitting data by two ways are Training dataset and Testing dataset. In statistics and machine learning we usually split our data into two subsets: training data and testing data

(and sometimes to three: train, validate and test), and fit our model on the train data, to make predictions on the test data. When this work does that, one of two things might happen: we overfit our model or we underfit our model. It doesn't want any of these things to happen, because they affect the predictability of our model- Work might be using a model that has lower accuracy and/or is ungeneralized.

6.2.5 Classification

Classification of data through IForest (Isolation Forest) Algorithm uses randomness by design to ensure they effectively learn the function being approximated for the problem. Randomness is used because this class of machine learning algorithm performs better with it than without. The most common form of randomness used in neural networks is the random initialization of the network weights. An iforest with a certain level of complexity, a network with more than two layers. IForest uses sophisticated mathematical modeling to process data in complex ways. Each layer performs specific types of sorting and ordering in a process that some refer to as "feature hierarchy."

6.2.5 Accuracy Prediction

The accuracy agriculture prediction occurs using the classifier, list with predicted values and plot the farmland graph. Predicting the test set result. The predicted result will give you the probability of finding the farmland. This will convert that probability into binary 0 and 1. 1 for Fraud and 0 for Non-Fraud. This is the final step where this work evaluates our model performance. It predicts the Confusion matrix to check the accuracy of model.

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 system, 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. 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 programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.

6.3.1 USE CASE DIAGRAM

A use case is a set of scenarios that describe 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. An actor represents a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform to complete a task.

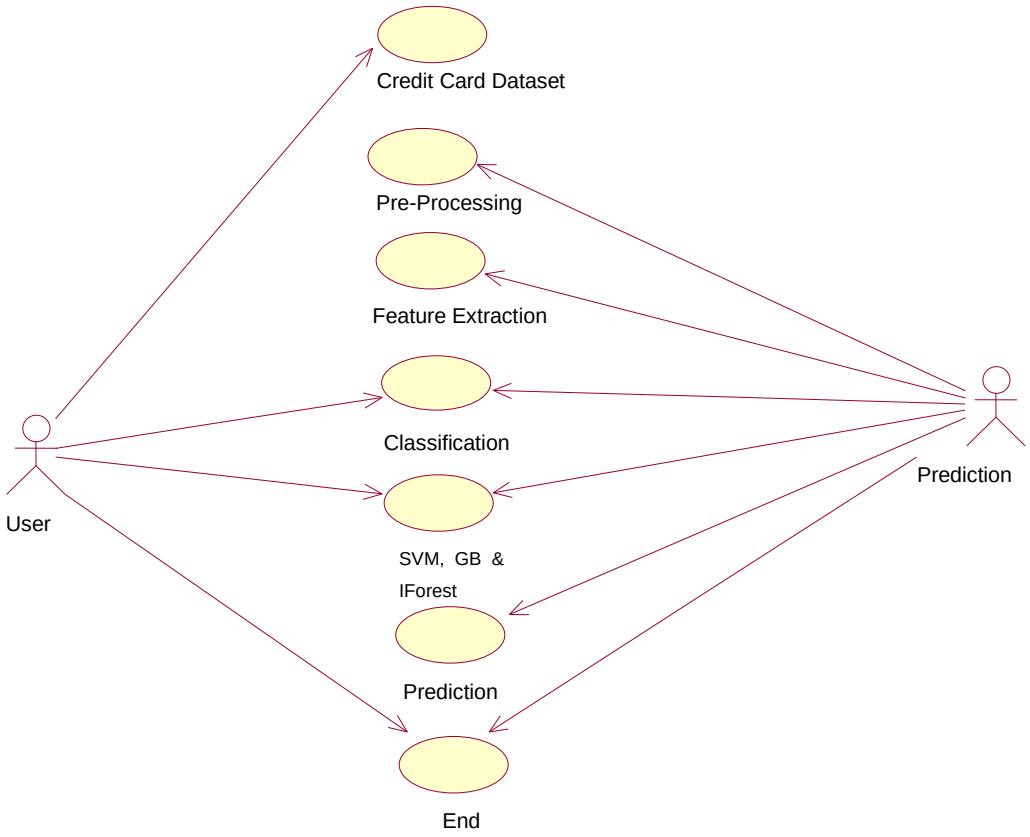


Fig 6.3.1 Use Case Diagram

6.3.2 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 workflow of components in a system. An activity diagram shows the overall flow of control.

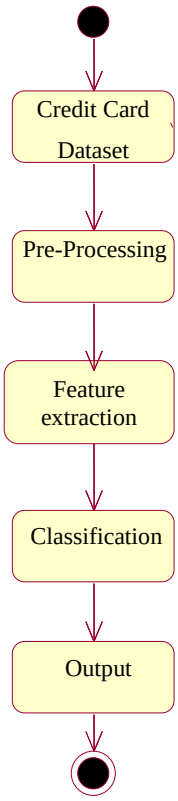


Fig 6.3.2 Activity Diagram

6.3.3 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-trace diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

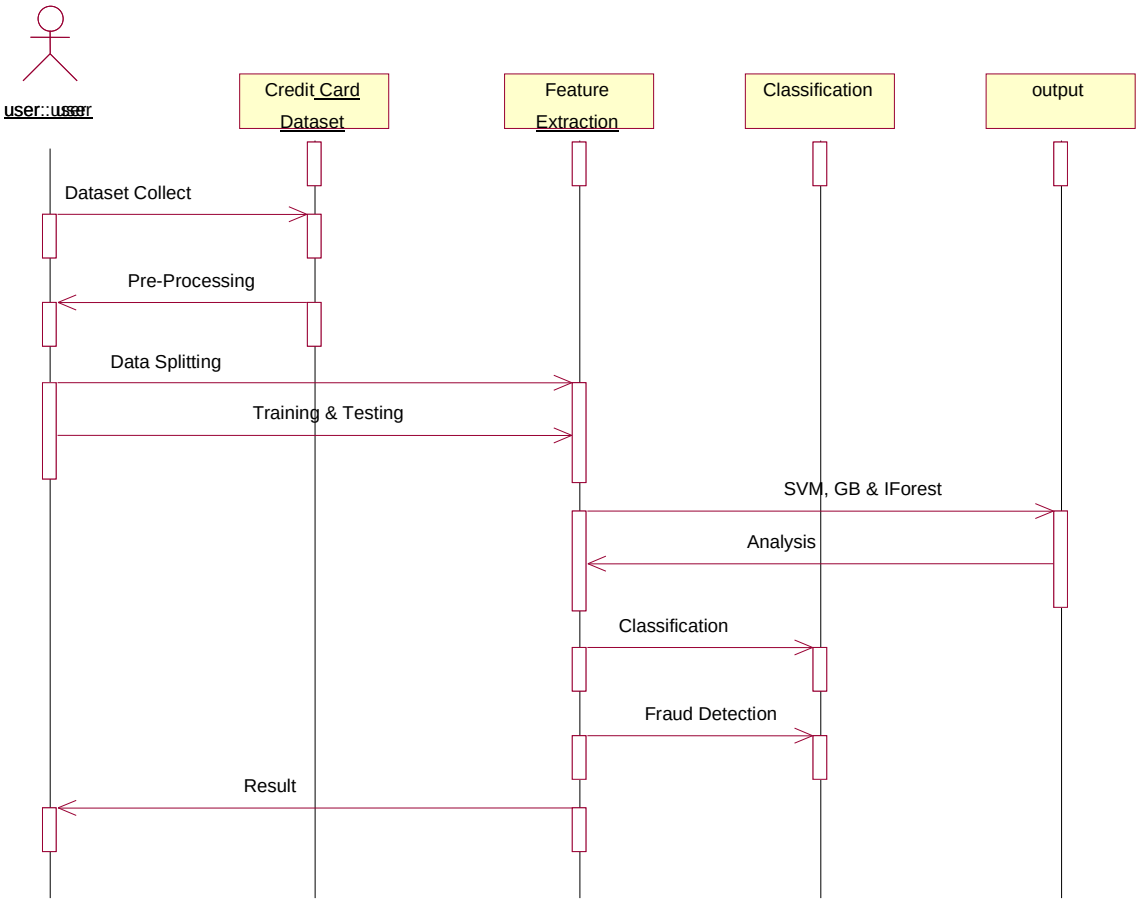


Fig 6.3.3 Sequence Diagram

6.3.4 COLLABRATION DIAGRAM

Collaboration diagrams belong to a group of UML diagrams called Interaction Diagrams. Collaboration diagrams, like Sequence Diagrams, show how objects interact over the course of time. However, instead of showing the sequence of events by the layout on the diagram, collaboration diagrams show the sequence by numbering the messages on the diagram. This makes it easier to show how the objects are linked together, but harder to see the sequence briefly.

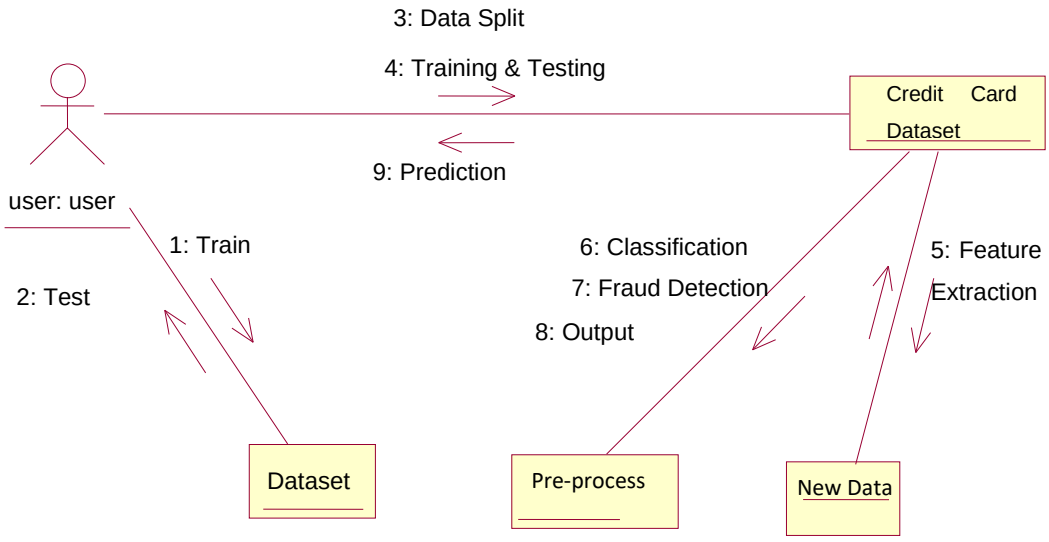


Fig 6.3.4 Collaboration Diagram

6.4 DATA FLOW DIAGRAM

DFDs are used to Specify Functions of the Information System and how data flows from function to function. It's a collection of functions that manipulates data. On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel.

6.4.1 Data Flow Diagram Level 0 For Credit Card Fraud Detection

DFD level 0 for credit card fraud detection explains the overall functionality of the entire project. Below shows the data flow diagram level 0 for credit card fraud detection.

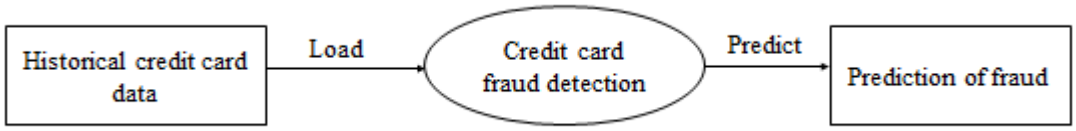


Fig 6.4.1 Data Flow Diagram Level 0 For Credit Card Fraud Detection

The collected data is loaded to the database and finally, the accuracy of fraud is predicted.

6.4.2 Data Flow Diagram Level 1 For Credit Card Fraud Detection

The below diagram shows the overall representation of each module and their functionality in this project.

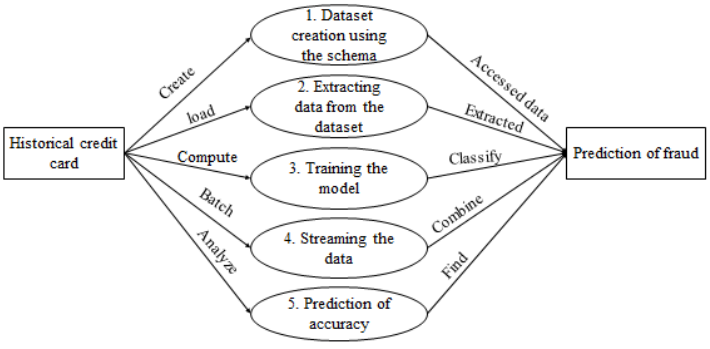


Fig 6.4.2 Data Flow Diagram Level 1 For Credit Card Fraud Detection

6.4.3 Data Flow Diagram Level 2 For Credit Card Fraud Detection Data Set Creation Using Schema

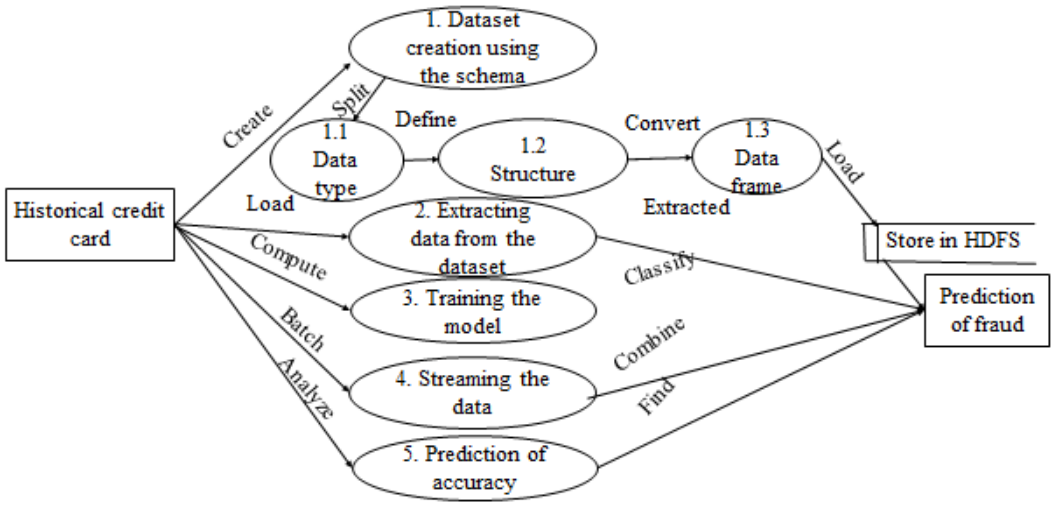


Fig 6.4.3 Data Flow Diagram Level 2 For Credit Card Fraud Detection Data Set Creation Using Schema

In this module, first split the dataset based on the data type and convert them as a data frames.

6.4.4 Data Flow Diagram Level 2 For Extracting Data from the Data Set

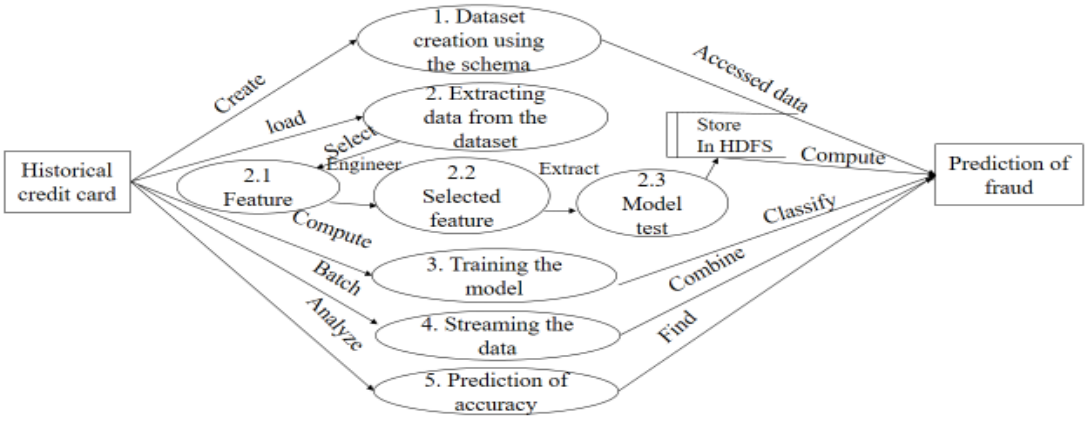


Fig 6.4.4 Data Flow Diagram Level 2 For Extracting Data from the Data set

In this module extract the features which are needed to create model test.

6.4.5 Data Flow Diagram Level 2 For Training the Model

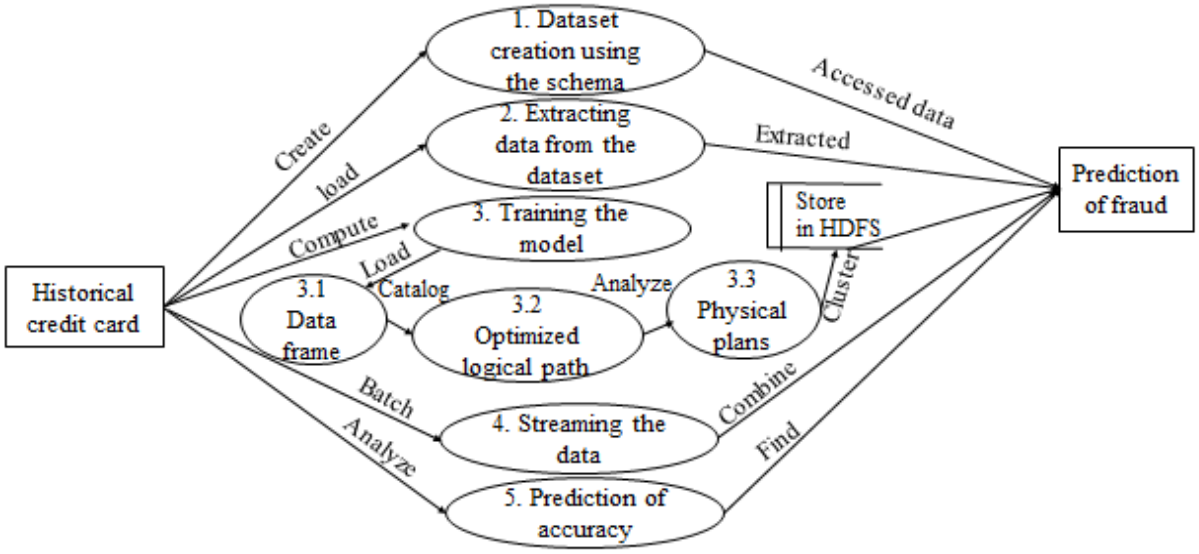


Fig 6.4.5 Data Flow Diagram Level 2 For Training the Model

In this module using catalyst optimization algorithm create data frame. Then catalog the optimized logical path. Analyze the physical plans and store the data in the database.

6.4.6 Data Flow Diagram Level 2 For Streaming Data

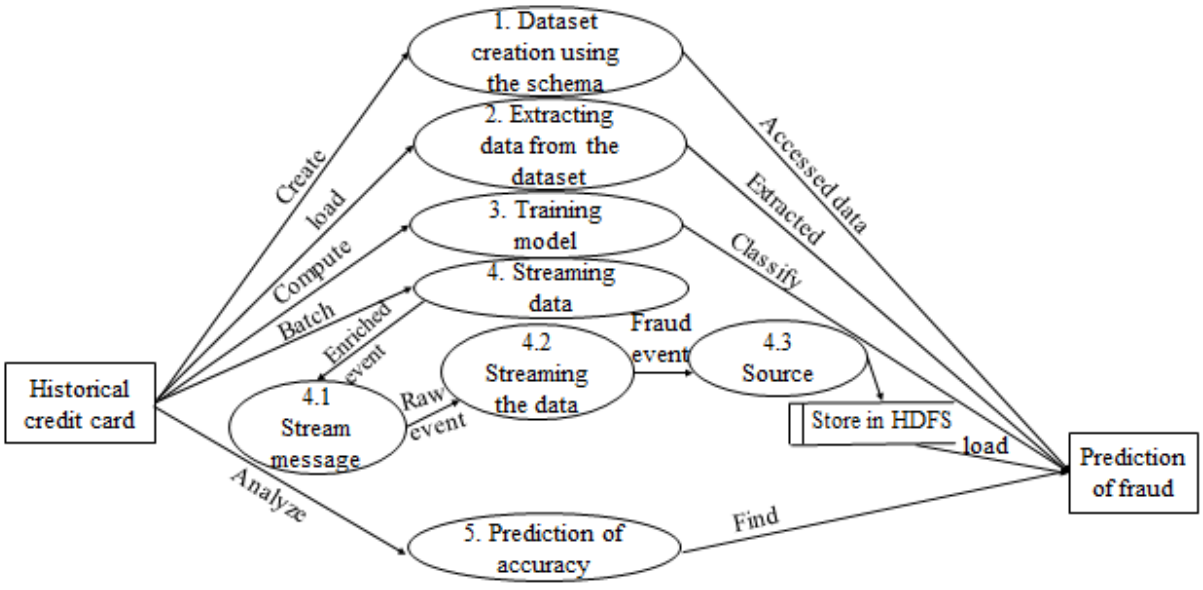


Fig 6.4.6 Data Flow Diagram Level 2 For Streaming Data

6.4.7 Data Flow Diagram Level 2 For Streaming Data

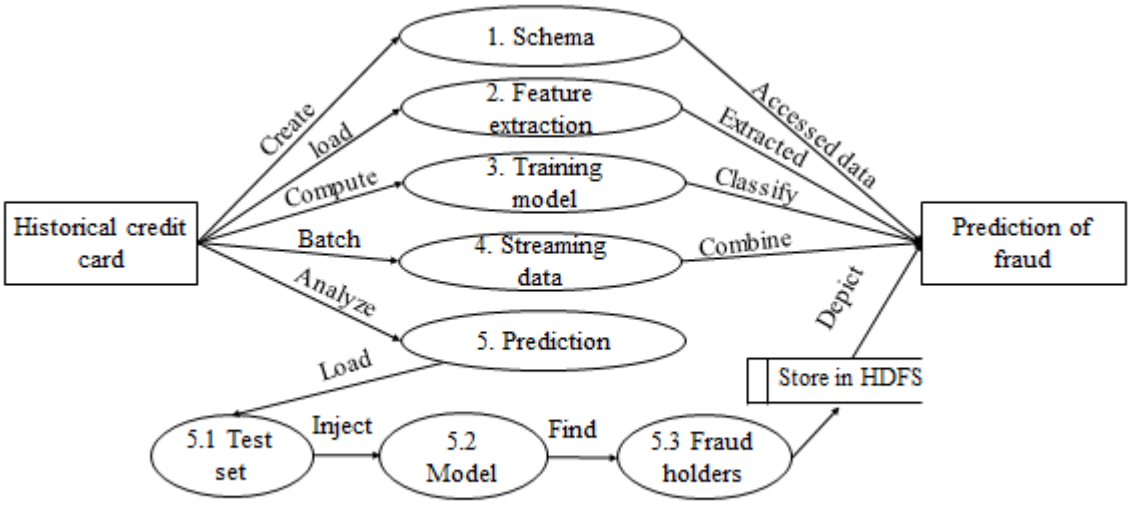


Fig 6.4.7 Data Flow Diagram Level 2 For Streaming Data

CHAPTER-7

SOURCE CODE

#ISOLATION FOREST

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.tokenize import sent_tokenize, word_tokenize

from sklearn.model_selection import train_test_split

from sklearn.pipeline import Pipeline

from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

from imblearn.over_sampling import SMOTE

from sklearn.svm import LinearSVC

from sklearn.linear_model import SGDClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Reading the data from data set
```

```
data = pd.read_csv(r'creditcard.csv',  encoding='latin-1')

data.head()

#Total data Row and column wise

data.shape

data = data.sample(frac = 0.2, random_state = 1)

print(data.shape)

# Plot the histogram of each parameter

data.hist(figsize = (20, 20))

plt.show()

# Determine the number of fraud cases

fraud = data[data['Class'] == 1]

valid = data[data['Class'] == 0]

#Data cleaning – Remove unnecessary variables

outlier_fraction = len(fraud) / float(len(valid))

print(outlier_fraction)

print ('Fraud Cases: {}'.format(len(fraud)))

print ('Valid Cases: {}'.format(len(valid)))

# correlation matrix

corrmat = data.corr()

fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)
```

```
plt.show()

#RENAME COLUMNS:

# Get the columns from the dataframe

columns = data.columns.tolist()

# Filter the columns to remove the data we do not want

columns = [c for c in columns if c not in ['Class']]

# Store the variable we will be predicting on which is class

target = 'Class'

# X includes everything except our class column

X = data[columns]

# Y includes all the class labels for each sample

# This is also one-dimensional

Y = data[target]

# Print the shapes of X and Y

print(X.shape)

print(Y.shape)

from sklearn.metrics import classification_report, accuracy_score

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

# define a random state

state = 1
```

```
# define the outlier detection methods
classifiers = {
    # contamination is the number of outliers we think there are
    'Isolation Forest': IsolationForest(max_samples = len(X),
                                        contamination = outlier_fraction,
                                        random_state = state),
    # number of neighbors to consider, the higher the percentage of outliers the higher you
#want to make this number
    'Local Outlier Factor': LocalOutlierFactor(
    n_neighbors = 20,
    contamination = outlier_fraction)
}
n_outliers = len(fraud)
for i, (clf_name, clf) in enumerate(classifiers.items()):
    # fit the data and tag outliers
    if clf_name == 'Local Outlier Factor':
        y_pred = clf.fit_predict(X)
        scores_pred = clf.negative_outlier_factor_
    else:
        clf.fit(X)
        scores_pred = clf.decision_function(X)
```

```
        y_pred = clf.predict(X)
    # reshape the prediction values to 0 for valid and 1 for fraud
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1
    # calculate the number of errors
    n_errors = (y_pred != Y).sum()
    # classification matrix
    print('{}: {}'.format(clf_name, n_errors))
    print(accuracy_score(Y, y_pred))
    print(classification_report(Y, y_pred))
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state = 0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
smote = SMOTE()
X_train_sm,y_train_sm = smote.fit_resample(X_train,y_train)
print(X_train_sm.shape)
print(y_train_sm.shape)
```

#GRADIENT BOOSTING

```
model_gb = Pipeline([('tfidf', TfidfTransformer()),
                     ('model', GradientBoostingClassifier(random_state=100,
n_estimators=150,min_samples_split=100, max_depth=6)), ])
model_gb.fit(X_train_sm,y_train_sm)
ytest = np.array(y_test)
y_pred = model_gb.predict(X_test)
print ('accuracy %s' % accuracy_score(y_pred, y_test))
print (classification_report(ytest, y_pred))
```

#SUPPORT VECTOR MACHINE

```
model_svc = Pipeline([('tfidf', TfidfTransformer()),
                      ('model',LinearSVC()) ])
model_svc.fit(X_train_sm,y_train_sm)
ytest = np.array(y_test)
predict = model_svc.predict(X_test)
print ('accuracy %s' % accuracy_score(predict, y_test))
print (classification_report(ytest, predict))
```

CHAPTER-8

SYSTEM TESTING AND IMPLEMENTATION

8.1 TESTING PROCESS

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 tests. Each test type addresses a specific testing requirement.

8.1.1 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 input produces 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 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.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify 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 Testing

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 test 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, like most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a test 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.

8.1.2 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 in the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

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.

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.

8.2 SYSTEM IMPLEMENTATION

Unit Testing

Unit testing comprises the set of tests performed by an individual programmer prior to integration of the unit into a larger system. A program unit is usually small enough that the programmer who developed it can test it in great detail and certainly is greater detail than will be possible when the unit is integrated into an evolving software product.

Test case Id	Test Case	Expected Result	Actual result	Success/ failure
UTC_test1	Registration	The user gives their details, login id, and login password to server.	The user details stored in server system. That cannot view any other user. if any fault means error report display	Success
UTC_test2	Login	Using server login id, login password user moves to next step.	If it is correct login details the message will display. Otherwise, the user cannot login to next step.	Success
UTC_test3	Upload File	The client creates the file and uploads in the cloud.	The file is being uploaded and saved in the Google App Engine.	Success

Validation Testing

Test condition	Test Input	Expected output	Actual output	Result
Checking for login process	Correct username and password	Perspective main pages should appear	Main page should appear	Success
	Correct username and wrong password	Main page should appear	Display the message that the password is incorrect	Fail
	Wrong username and correct password	Main page should appear	Display the message that the username is incorrect	Fail
	Wrong username and password	Main page should appear	Display the message that the username and password are incorrect	Fail
File upload activity	The file is browsed and uploaded	The file is uploaded.	The file is uploaded	Success
	File name not entered	Prompt user to enter the file name	File got stored in the persistent storage.	Fail
File Download Activity	File download if correct user	File download successfully	File Download	Success
	File download failure	If not in member list	File Download Not Successful	Fail.

CHAPTER-9

PERFORMANCE ANALYSIS

9.1 PROCESSING SPEED

This section extends the analysis of the techniques proposed previously for the big data forensic investigation. There are different factors that can influence the selection and performance of the forensic techniques. These factors should be analysed closely before carrying out the implementation. The following table has been populated based on the available resources and the sensitivity of the data to be used for investigation.

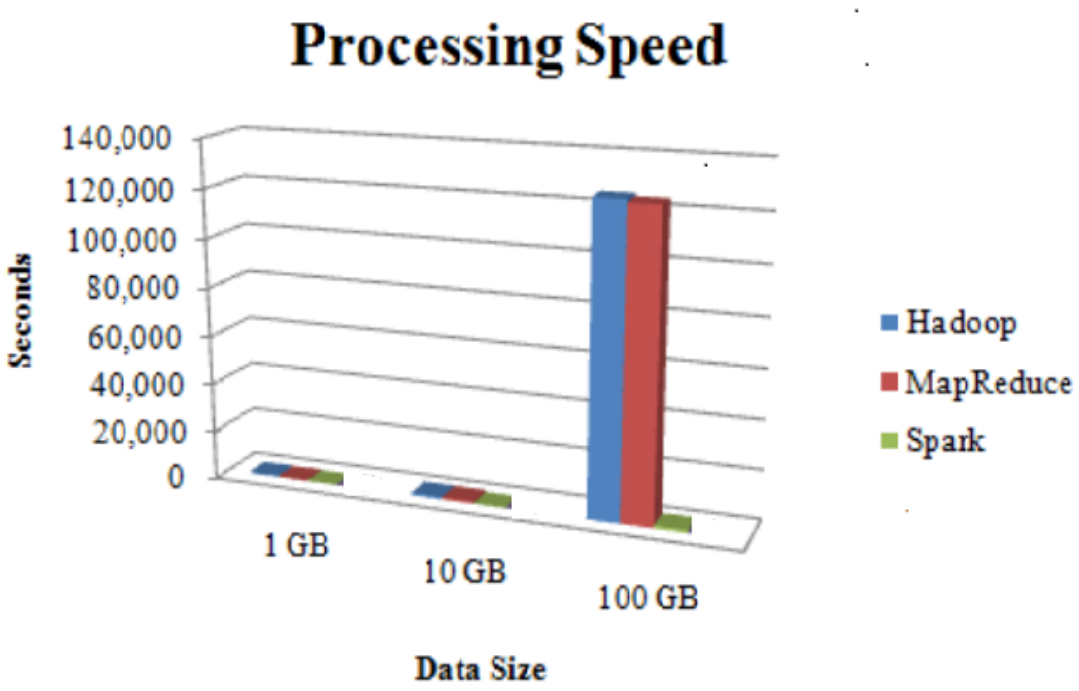


Fig 9.1 Processing Speed

In Figure 9.1 shows that the processing speed of Apache Spark lower even though the data size increases. But Hadoop and MapReduce increase its processing time if the data size is too big to handle.

9.2 LATENCY

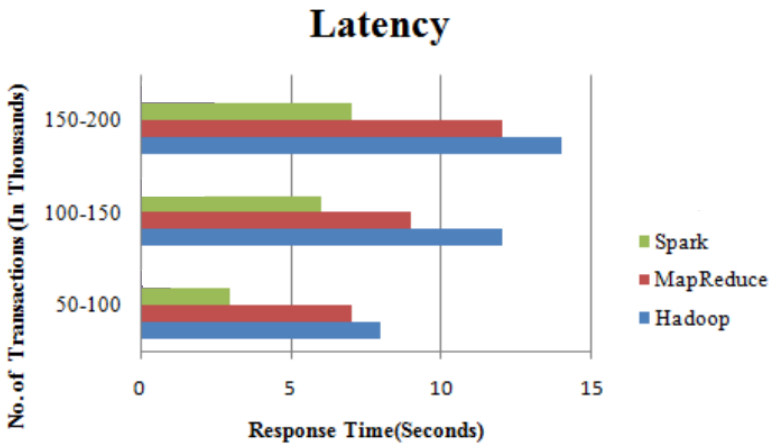


Fig 9.2 Latency

The response time should be at minimum while executing the data/transactions. Among the three techniques, Apache spark has low latency while processing Big Data. While analyzing the Fault tolerance factor, Spark system replicates the input data in memory which is a most useful solution for handling faults in between the execution of transactions. Data lost due to failure can be recomputed from replicated input data.

9.3 PERFORMACE

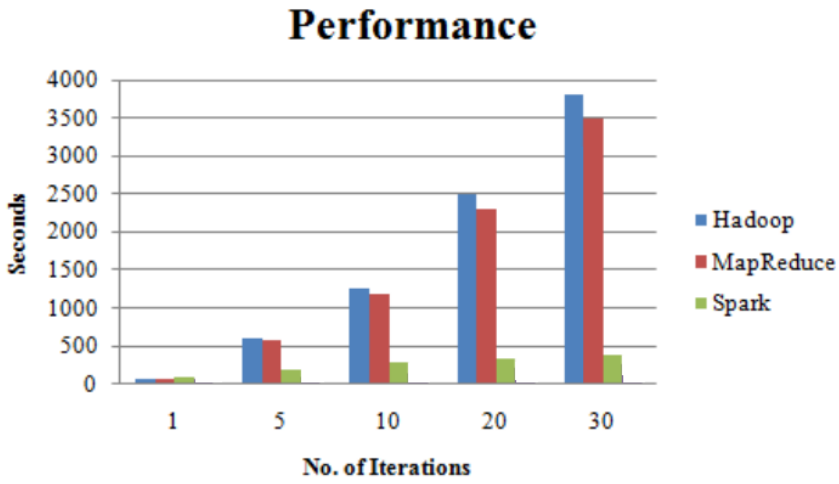
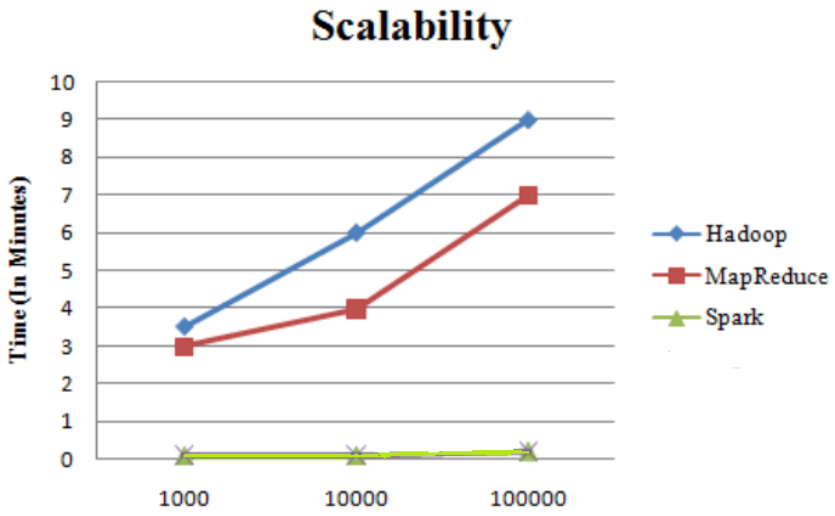


Fig 9.3 Performance

In Fig 9.3 shows that Spark has a short execution time when compared to other techniques.

9.4 SCALABILITY



9.4 Scalability

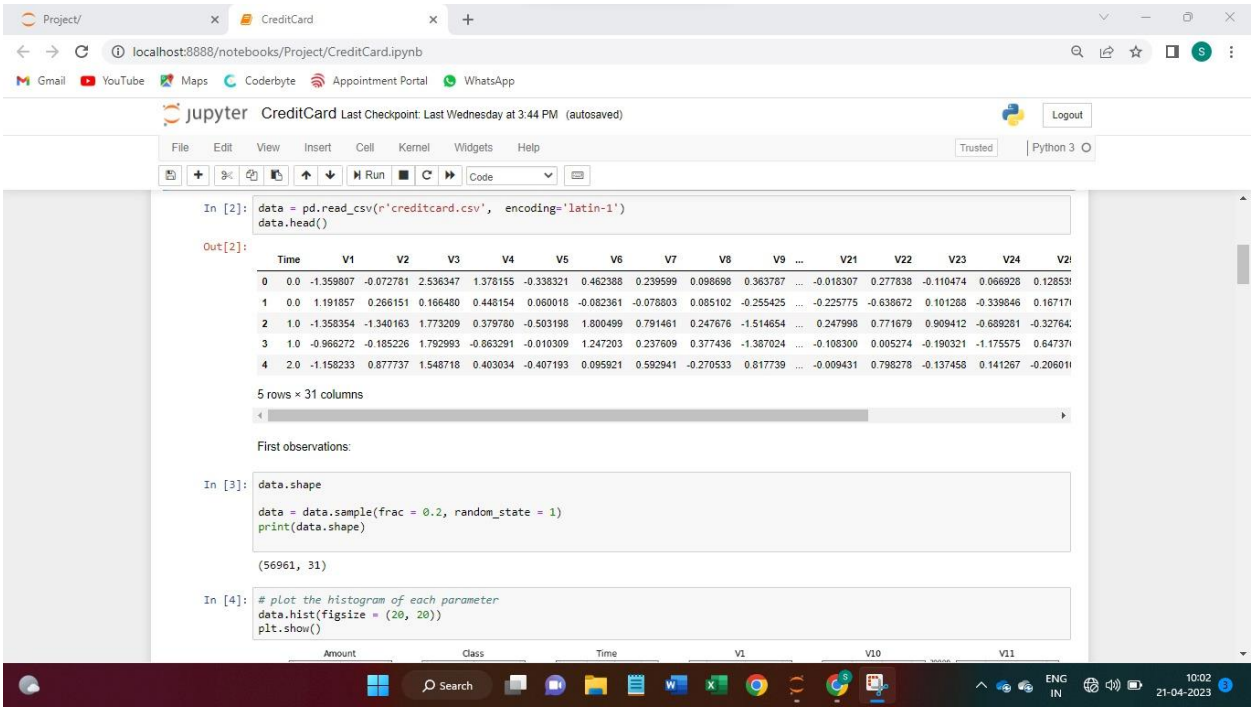
In Fig 9.4 shows the scalability factor, Spark processes the data smoothly if there is need to increase nodes for processing Big Data. Although the number of node increases, Spark performs better with Big Data.

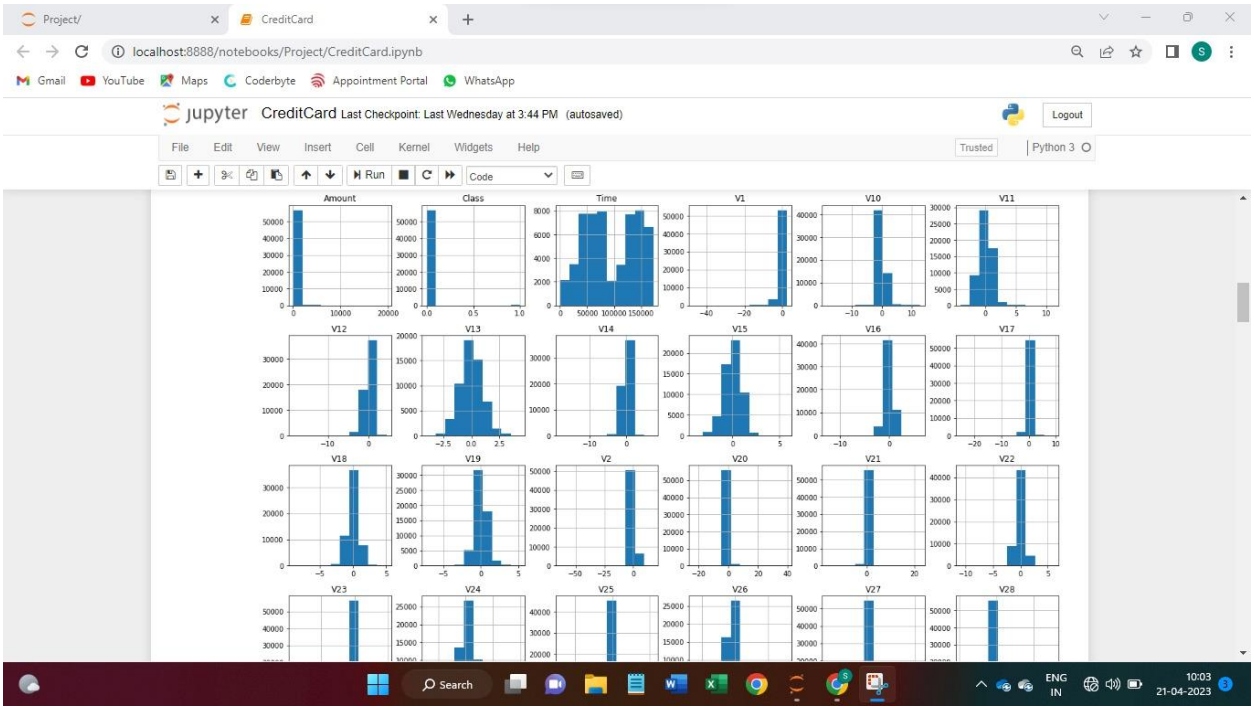
Techniques	Processing speed	Latency	Fault Tolerance	performance	Scalability
Hadoop	Medium	High	High	Slow	Medium
Mapreduce	Slow	High	High	Slow	Medium
Spark	Fast	Low	High	Fast	High

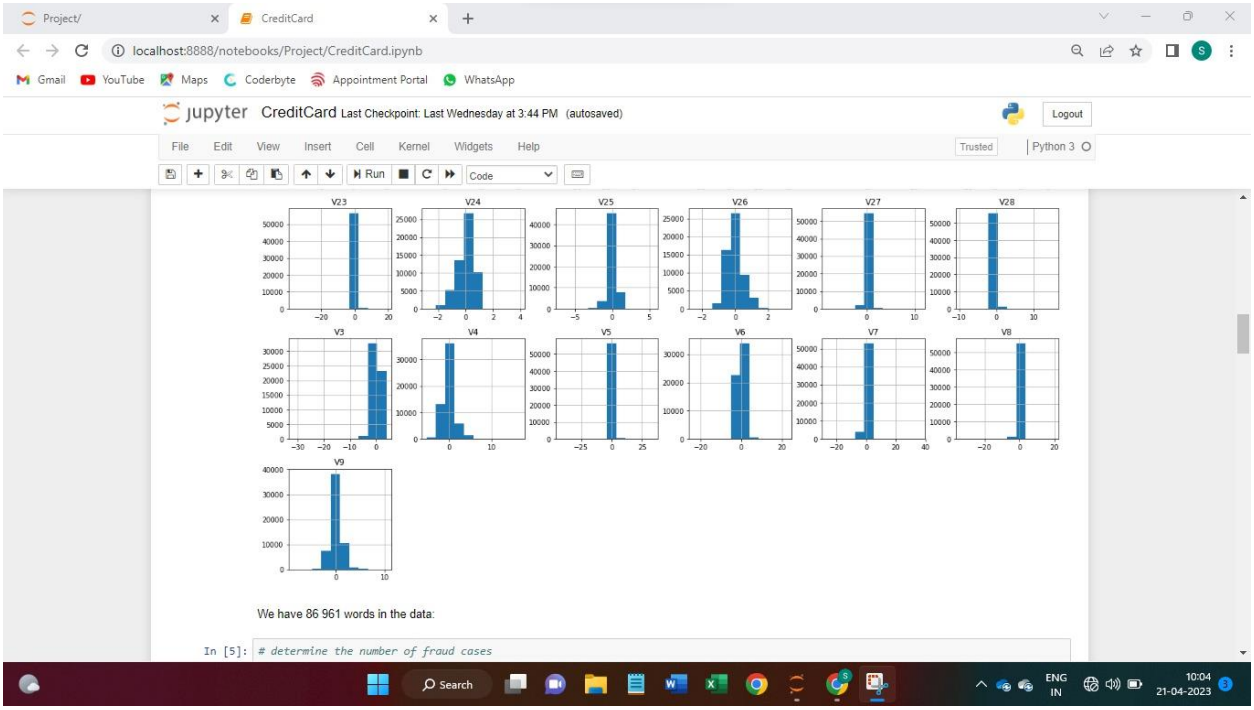
Analysis of Techniques shows the overall comparison of other techniques.

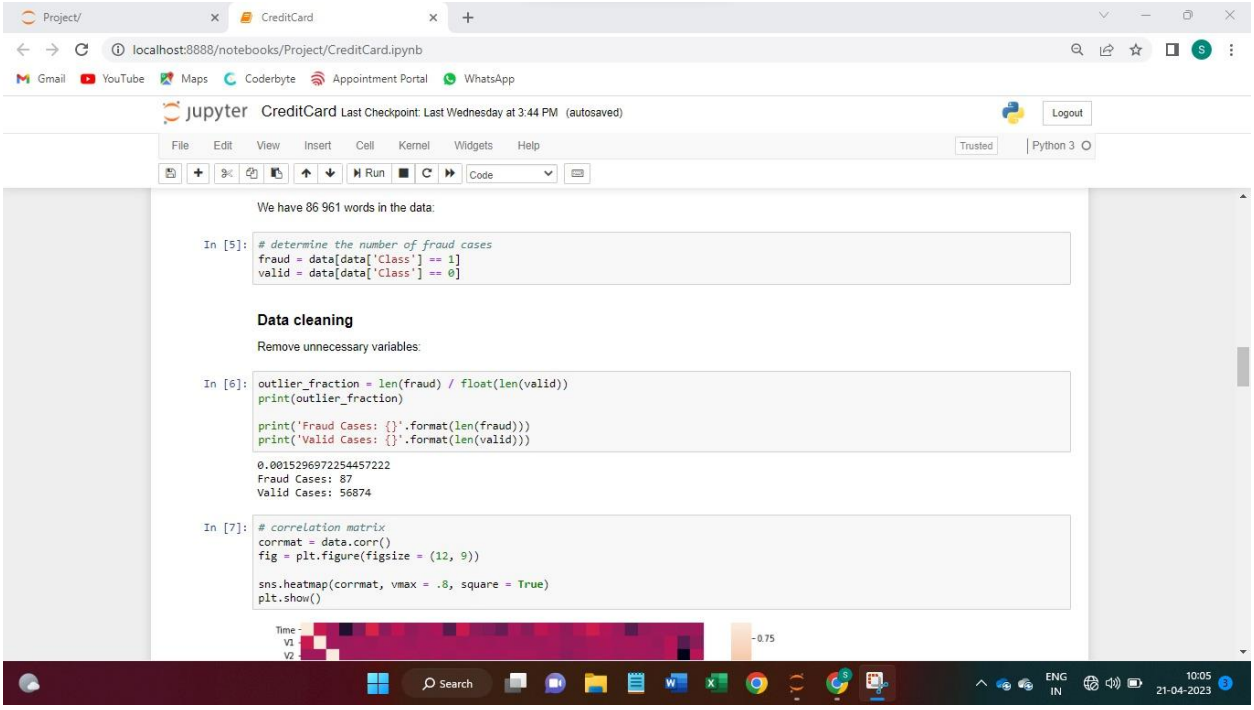
CHAPTER-10

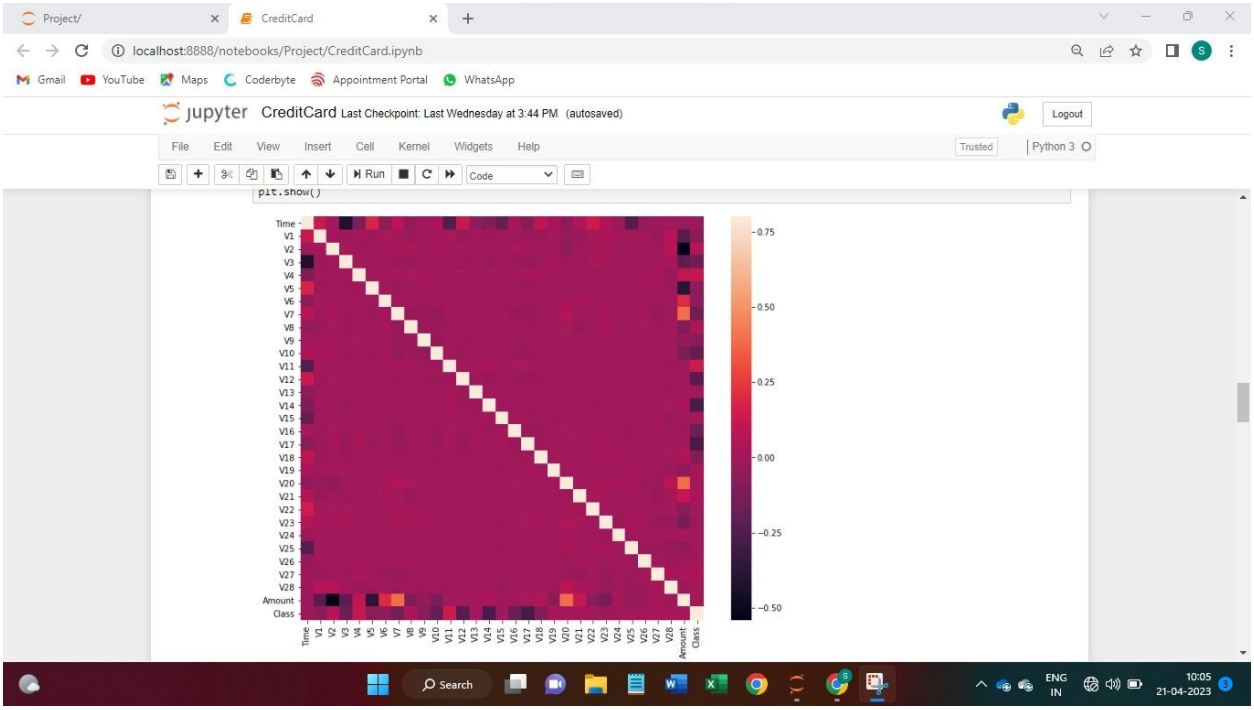
SCREENSHOTS






Project/
CreditCard
localhost:8888/notebooks/Project/CreditCard.ipynb
Gmail
YouTube
Maps
Coderbyte
Appointment Portal
WhatsApp
jupyter CreditCard Last Checkpoint: Last Wednesday at 3:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
V23
V24
V25
V26
V27
V28
V3
V4
V5
V6
V7
V8
V9
We have 86 961 words in the data:
In [5]:
determine the number of fraud cases
Search
ENG
IN
10:04
21-04-2023

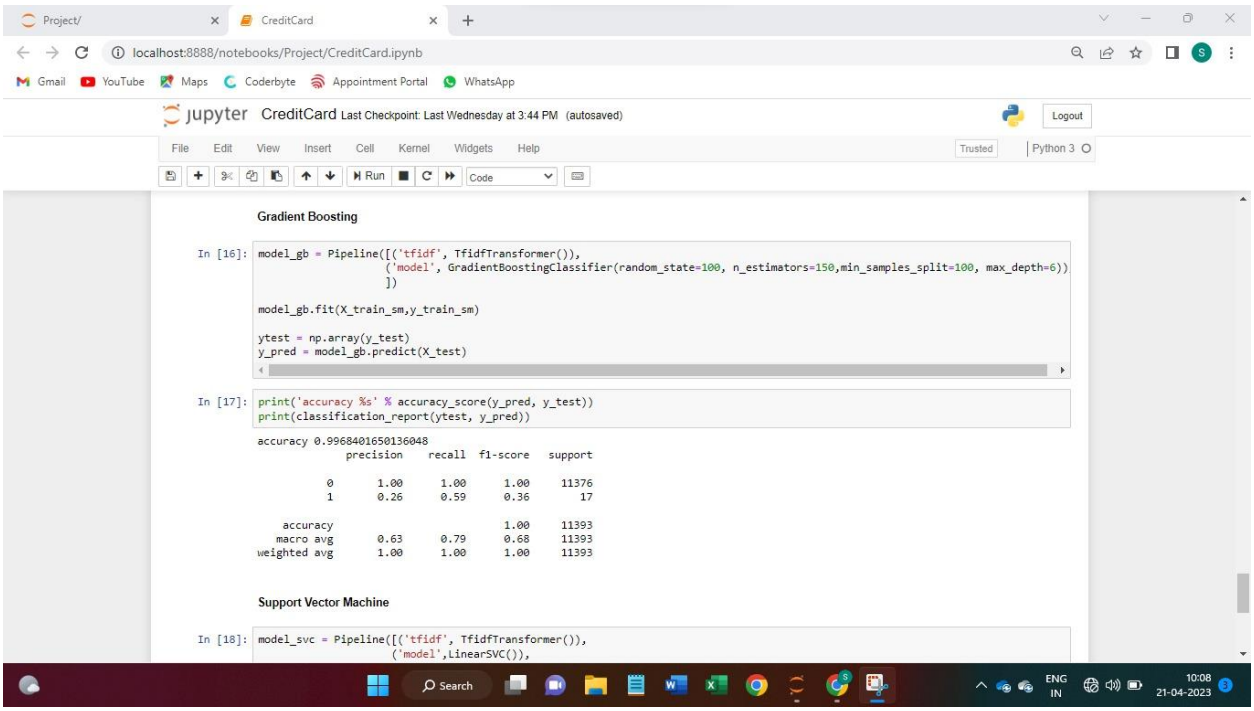

Project/
CreditCard
localhost:8888/notebooks/Project/CreditCard.ipynb
Gmail
YouTube
Maps
Coderbyte
Appointment Portal
WhatsApp
jupyter CreditCard Last Checkpoint: Last Wednesday at 3:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
We have 86 961 words in the data:
In [5]:
# determine the number of fraud cases
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
Data cleaning
Remove unnecessary variables:
In [6]:
outlier_fraction = len(fraud) / float(len(valid))
print(outlier_fraction)
print('Fraud Cases: {}'.format(len(fraud)))
print('Valid Cases: {}'.format(len(valid)))
0.0015296972254457222
Fraud Cases: 87
Valid Cases: 56874
In [7]:
# correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
Time
V1
V2
0.75
Search
ENG
IN
10:05
21-04-2023

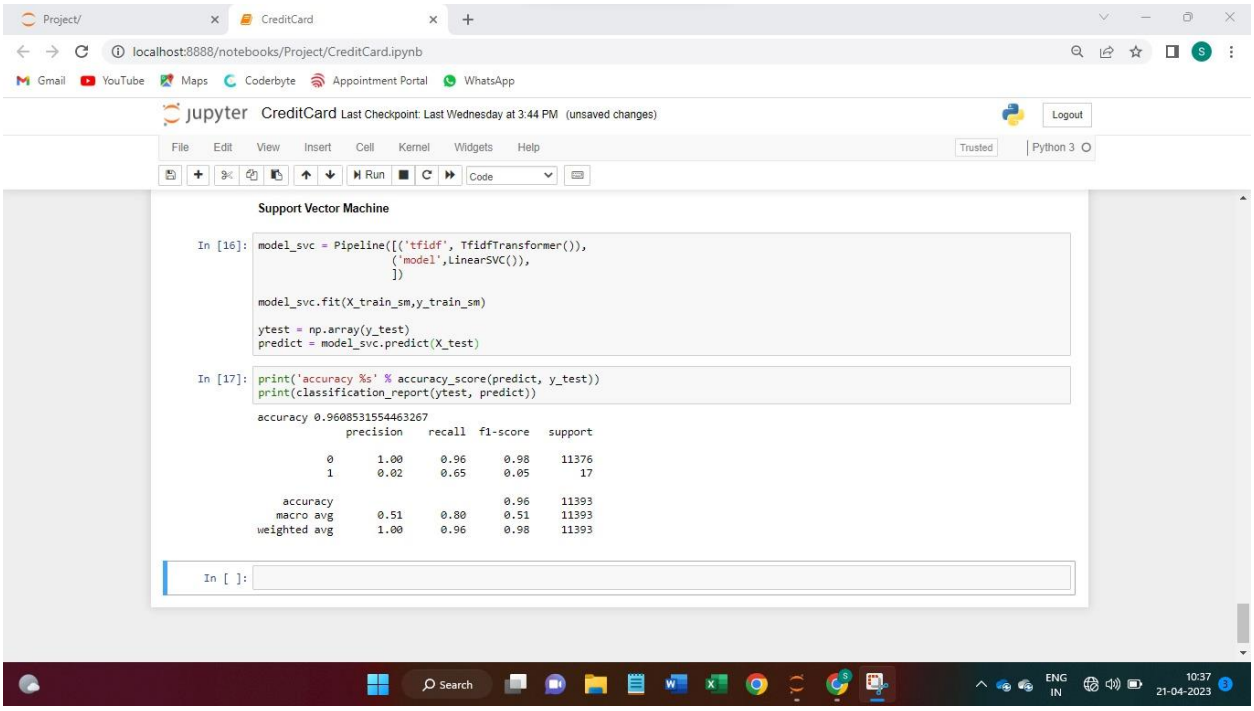



```
    print(classification_report(Y, y_pred))
```

```
C:\Users\jyoth\Anaconda3\lib\site-packages\sklearn\base.py:451: UserWarning: X does not have valid feature names, but IsolationForest was fitted with feature names
  "X does not have valid feature names, but"
```

```
Isolation Forest: 127
0.997770404311722
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56874
           1       0.27      0.28      0.27        87

    accuracy                           1.00     56961
   macro avg       0.64      0.64      0.64     56961
weighted avg       1.00      1.00      1.00     56961

Local Outlier Factor: 173
0.9969628342199048
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56874
           1       0.01      0.01      0.01        87

    accuracy                           1.00     56961
   macro avg       0.50      0.50      0.50     56961
weighted avg       1.00      1.00      1.00     56961
```


Project/
CreditCard
localhost:8888/notebooks/Project/CreditCard.ipynb
jupyter CreditCard Last Checkpoint: Last Wednesday at 3:44 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Gradient Boosting
In [16]:
model_gb = Pipeline([('tfidf', TfidfTransformer()),
('model', GradientBoostingClassifier(random_state=100, n_estimators=150,min_samples_split=100, max_depth=6))
])
model_gb.fit(X_train_sm,y_train_sm)
ytest = np.array(y_test)
y_pred = model_gb.predict(X_test)
In [17]:
print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(ytest, y_pred))
accuracy 0.9968401650136048
precision recall f1-score support
0 1.00 1.00 1.00 11376
1 0.26 0.59 0.36 17
accuracy 1.00 11393
macro avg 0.63 0.79 0.68 11393
weighted avg 1.00 1.00 1.00 11393
Support Vector Machine
In [18]:
model_svc = Pipeline([('tfidf', TfidfTransformer()),
('model',LinearSVC()),
Search
ENG
IN
10:08
21-04-2023


Project/
CreditCard
localhost:8888/notebooks/Project/CreditCard.ipynb
jupyter CreditCard Last Checkpoint: Last Wednesday at 3:44 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Support Vector Machine
In [16]:
model_svc = Pipeline([('tfidf', TfidfTransformer()),
('model',LinearSVC()),
])
model_svc.fit(X_train_sm,y_train_sm)
ytest = np.array(y_test)
predict = model_svc.predict(X_test)
In [17]:
print('accuracy %s' % accuracy_score(predict, y_test))
print(classification_report(ytest, predict))
accuracy 0.9608531554463267
precision recall f1-score support
0 1.00 0.96 0.98 11376
1 0.02 0.65 0.05 17
accuracy 0.96 11393
macro avg 0.51 0.80 0.51 11393
weighted avg 1.00 0.96 0.98 11393
In [ ]:
Search
ENG
IN
10:37
21-04-2023

CHAPTER-11

CONCLUSION

Detection of credit card fraud is an intent part of testing for the researchers over a long time and will be an interesting part of testing in the coming time. We are introducing a fraud detection system for credit cards by applying three different algorithms and training our machine using these algorithms with the transaction records we have. The model that we built helps the authorities to get notified of the fraud in credit-cards and take the further necessary steps over the transaction and label the transaction as fraud or legitimate transaction. These algorithms show us whether the given transaction tends to be a type of fraud or not. These algorithms were selected using experimentation, discussion and feature important techniques as shown in methodology. It is real-time transaction data from European credit-card holders which explains the skewness of data. Therefore, we can infer that there is a requirement of applying feature selection techniques. We used supervised and unsupervised algorithms like IForest, SVM and GB to select the features from our transaction dataset which uses correlation and variance as parameters to select the features. We have set the summation of variance as 95% for selecting the features using the IForest, SVM and GB algorithm. We also applied the IForest, SVM and GB feature selection algorithm as there are no features which have high variance and correlation with the class column which determines the transaction as fraud or not. We have applied the three machine learning algorithms as stated in methodology and the models indicate a high accuracy score for each one of them. The scores of each model were 99.7%, 99.8%, 99.7% for the Iforest, support vector machine and gradient boosting classifier algorithms respectively. These models have high accuracy and get the best results with high precision in determining the fraud detections in credit card transaction records.

CHAPTER-12

FUTURE ENHANCEMENT

Reaching a Goal of 100% should be the target of our accuracy score for our machine model which detects the fraud detection of credit card transactions. But reaching a 100% accuracy score we can infer that our model is being overfitted with data which is giving us the output which is already being trained for it. So, for future enhancements we can convey that the precision and confusion matrix values can be improved with a high range. We can furthermore implement new algorithms to our Indian credit-card holders transaction dataset and combine the results for these algorithms for precision and confusion matrix to get more legitimate values. The data set can be improved too with replacing the highly skewed values to normalized values and bringing a pattern to it which helps us in building a more accurate model. The outliers can be minimized in the present data set as they confuse the model while training it. The correlation and variance are also less in the dataset which can be improved by minimizing the skewness of the data. These will increase the modularity and versatility of our project and make it more accurate to predict whether the transactions are tending to be fraudulent or not. These improvements require the knowledge and support of experts from different sectors like machine learning and artificial intelligence experts, data scientists and bankers who will help us to provide a better form of data.

REFERENCES

[1]. Li Yingjiu and Xinwen Zhang, "A security-enhanced one-time payment scheme for credit cards", in Proc. of 14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications, 2004, Page(s): 40 - 47.

[2] W. Stallings, Cryptography, and network security: principles and practices, 3rd edition.

[3] Leila Seyedhossein, mahmoudrezahashemi “Mining Information from Credit Card Time Series for Timelier Fraud Detection”. IEEE-5th International Symposium on Telecommunications. (2010).

[4] E.W.T. Ngai, Yong Hu, Y.H. Wong, Yijun Chen, Xin Sun “The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature”. Elsevier-Decision Support Systems (2011). 50; (559–569).

[5] A. Abdallah, M. A. Maarof, and A. Zainal, “Fraud detection system: A survey,” J. Netw. Comput. Appl., vol. 68, pp. 90–113, Jun. 2016

[6] Abhinav Srivastava, Amlan Kundu, Shamik Sural and Arun K. Majumdar, "CreditCard Fraud Detection Using Hidden Markov Model" IEEE, Transactions on Dependable and Secure Computing, Vol. 5, No 1., January-March 2008.

[7] V. Bhusari, and S. Patil, “Study of Hidden Markov Model in Credit Card Fraudulent Detection”, International Journal of Computer Applications (0975 – 8887) Volume 20– No.5, April 2011.

[8] N. Abdelhamid, A. Ayesh, and F. Thabtah, “Phishing detection based associative classification data mining,” Expert Syst. Appl., vol. 41, no. 13, pp. 5948–5959, 2014.

[9] R. Brause, T. Langsdorf, and M. Hepp, “Neural data mining for credit card fraud detection,” in Proc. IEEE Int. Conf. Tools Artif. Intell., 1999, pp. 103–106.

[10] R. C. Chen, S. T. Luo, X. Liang, and V. C. S. Lee, "Personalized approach based on SVM and ANN for detecting credit card fraud," in Proc. Int. Conf. Neural Netw. Brain, Oct. 2005, pp. 810–815

[11] C. Cortes and D. Pregibon, "Signature-based methods for data streams," Data Mining Knowl. Discovery, vol. 5, no. 3, pp. 167–182, 2001.