CREDIT CARD FRAUD DETECTION USING RANDOM FOREST ALGORITHM

A project Report submitted in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology

In

Computer Science and Engineering

By

Ch.V S G M K Nagendra (19A41A0513)

K.Sravan Kumar Reddy (19A41A0520) **T.Gowtham Reddy (19A41A0550)**

K.Sai Prasanna (19A41A0522) **A.Venkata Sathish kumar (19A41A0503)**

Under The Esteemed Guidance of

Mr.T.V.Gopal krishna

Assoc Professor

Department of Computer Science and Engineering

LOYOLA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(Approved by AICTE, New Delhi, Affiliated to JNTUK, Kakinada)

(An ISO 9001:2015 Certified & NAAC 'A' Grade Institution)

Dhulipalla-522412, Sattenapalli(M),Guntur District, AndhraPradesh

2022-2023

LOYOLA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(Approved By AICTE& affiliated to JNTUK, Kakinada)

(An ISO 9001:2015 Certified & NAAC 'A' Grade Institution)

Dhulipalla-522412, Sattenapalli(M)

Guntur District, Andhra Pradesh

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that this report is a entitled as **Credit card Fraud Detection Using Random Forest Algorithm** the bonafide record work of **Ch.V S G M K Nagendra (19A41A0513), K.Sravan Kumar Reddy (19A41A0520), T.Gowtham Reddy (19A41A0550), K.Sai Prasanna (19A41A0522), A.Venkata Sathish Kumar (19A41A0503)** under the Guidance and supervision of **Mr.T.V.Gopal Krishna** submitted to the **Department of Computer Science and Engineering, Loyola Institute of Technology and Management,** in partial fulfillment of the requirements for the award of the **Degree in Bachelor of Technology in Computer Science and Engineering.**

INTERNAL GUIDE	**Head Of The Department**
Mr.T.V.Gopal krishna	**Mr.T.G.Ramnadh Babu**
Assoc.Professor	

External Examiner

DECLARATION

We students of **LOYOLA INSTITUTE OF TECHNOLOGY AND MANAGEMENT** Dhulipalla, Guntur District, Andhra Pradesh, hereby declare that this dissertation titled "**Credit card fraud detection using random forest algorithm**", being submitted to the department of Computer Science and Engineering of this institute, affiliated to Jawaharlal Nehru Technology University Kakinada, for the award of the Degree in Bachelor of Technology in **Computer Science and Engineering** is a record of bonafide work done by us at college and it has not been submittedto any other institute or University for the award of any other Degree.

Signatures Of Students With Names And Roll no's:

Ch.V S G M K Nagendra	**(19A41A0513)**
K.Sravan Kumar Reddy	**(19A41A0520)**
T.Gowtham Reddy	**(19A41A0550)**
K.Sai Prasanna	**(19A41A0522)**
A.Venkata Sathish Kumar	**(19A41A0503)**

ACKNOWLEDGEMENT

It is great pleasure for us to express our gratitude to our honorable chairman **Sri.V. Balashowry ,** secretary **Sri K V J Kumar** and director **Sri E.Vamsi Krishna Reddy** for kindly providing facilities in accomplishing our project and their consistent work in growth of management.

We express our great pleasure to our honorable principal **Dr S.Siva Reddy ,** who had inspired a lot through his valuable message. He is only personality who had given the meaning to the technological studies.

We express our sincere and heart full thanks to our beloved **Head of the Department of Computer Science & Engineering Mr.T.G.Ramnadh Babu,** for his motivation and encouragement at the stage of this endeavor and his constant help and support throughout our project.

We express our whole heartedthanks to our beloved guide **Mr.T.V.Gopal Krishna** for her splendid effort and guidancethroughout the course of project work without whom the assignment would nothave been successfully completed.

We also owe a special thanks to all our family members and the management and administration of **LITAM.**

Project Associates

Ch. V S G M K Nagendra	**(19A41A0513)**
K. Sravan Kumar Reddy	**(19A41A0520)**
T. Gowtham Reddy	**(19A41A0550)**
K. Sai Prasanna	**(19A41A0522)**
A.Venkata Sathish Kumar	**(19A41A0503)**

CONTENTS

ABSTRACT :

The project is mainly focussed on credit card fraud detection in real world. A phenomenal growth in the number of credit card transactions, has recently led to a considerable rise in fraudulent activities. The purpose is to obtain goods without paying, or to obtain unauthorized funds from an account. Implementation of efficient fraud detection systems has become imperative for all credit card issuing banks to minimize their losses. One of the most crucial challenges in making the business is that neither the card nor the cardholder needs to be present when the purchase is being made. This makes it impossible for the merchant to verify whether the customer making a purchase is the authentic cardholder or not. With the proposed scheme, using random forest algorithm the accuracy of detecting the fraud can be improved can be improved. Classification process of random forest algorithm to analyse data set and user current dataset. Finally optimize the accuracy of the result data. The performance of the techniques is evaluated based on accuracy, sensitivity, and specificity, and precision. Then processing of some of the attributes provided identifies the fraud detection and provides the graphical model visualization. The performance of the techniques is evaluated based on accuracy, sensitivity, and specificity, and precision.

1.INTRODUCTION :

1.1 Introduction to project:

There are various fraudulent activities detection techniques has implemented in credit card transactions have been kept in researcher minds to methods to develop models based on artificial intelligence , data mining, fuzzy logic and machine learning. Credit card fraud detection is significantly difficult, but also popular problem to solve. In our proposed system we built the credit card fraud detection using Machine learning. With the advancement of machine learning techniques. Machine learning has been identified as a successful measure for fraud detection. A large amount of data is transferred during online transaction processes,resulting in a binary result: genuine or fraudulent. Within the sample fraudulent datasets, features are constructed. These are data points namely the age and value of the customer account, as well as the origin of the credit card. There are hundreds of features and each contributes, to varying extents, towards the fraud probability. Note, the level in which each feature contributes to the fraud score is generated by the artificial intelligence of the machine which is driven by the training set, but is not determined by a fraud analyst. So, in regards to the card fraud, if the use of cards to commit fraud is proven to be high, the fraud weighting of a transaction that uses a credit card will be equally so. However, if this were to shrink, the contribution level would parallel. Simply make, these models self-learn without explicit programming such as with manual review. Credit card fraud detection using Machine learning is done by deploying the classification and regression algorithms. We use supervised learning algorithm such as Random forest algorithm to classify the fraud card transaction in online or by offline. Random forest is advanced version of Decision tree. Random forest has better efficiency and accuracy than the other machine learning algorithms. Random forest aims to reduce the previously mentioned correlation issue by picking only a subsample of the feature space at each split. Essentially, it aims to make the trees de-correlated and prune the trees by fixing a stopping criteria for node splits, which I will be cover in more detail later.

1.2 Algorithm

1.Machine Learning Algorithm

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Python SKLEARN inbuilt contains support for CART with all decision trees and random forest classifier.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

2.SYSTEM STUDY

2.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and 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

2.2 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.

2.3 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.

2.4 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.

3.SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

There are various fraudulent activities detection techniques has implemented in credit card transactions have been kept in researcher minds to methods to develop models based on artificial intelligence , data mining, fuzzy logic and machine learning. Credit card fraud detection is significantly difficult, but also popular problem to solve. In our proposed system we built the credit card fraud detection using Machine learning. With the advancement of machine learning techniques. Machine learning has been identified as a successful measure for fraud detection. A large amount of data is transferred during online transaction processes, resulting in a binary result: genuine or fraudulent. Within the sample fraudulent datasets, features are constructed. These are data points namely the age and value of the customer account, as well as the origin of the credit card. There are hundreds of features and each contributes, to varying extents, towards the fraud probability.

3.2 PROPOSED SYSTEM

In proposed System, we are applying random forest algorithm for classification of the credit card dataset. Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of over fitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built, each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provide a good estimate of the generalization error and to be resistant to over fitting.

3.3 SYSTEM REQUIREMENTS

3.3.1 Hardware System Configuration:-

❖ Processor	: Intel (R) Pentium (R)
❖ Speed	: 1.1 Ghz
❖ RAM	: 2GB
❖ Hard Disk	: 57 GB
❖ Key Board	: Standard Keyboard
❖ Mouse	: Standard Mouse

3.3.2 Software System Configuration

❖ Operating system	: Windows 7 and above,Mac,Linux others
❖ Coding Language	: Python.
❖ Front-End	: Python.
❖ Designing	: Html,css,javascript.
❖ Data Base	: MySQL.

3.4 MODULES DESCRIPTION

3.4.1 Upload dataset credit card data

Upload the dataset from credit card folder

3.4.2 Generate Train &test model

In this generating model we can see total records available in dataset and then application using how many records for training and how many for testing.

3.4.3 Random Forest Algorithm

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Python SKLEARN inbuilt contains support for CART with all decision trees and random forest classifier.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

3.4.4 Detect Fraud From Test Data

It can detect the fraud data from the test and trained data

3.5 PRELIMINARY INVESTIGATION

The first and foremost strategy for development of a project starts from the thoughtof designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine ,address book and also including someentertaining games. When it is approved by the organization and our project guide the first activity, ie. preliminary investigation begins. The activity has three parts:

Request Clarification

Feasibility Study

Request Approval

3.6 REQUEST CLARIFICATION

After the approval of the request to the organization and project guide, withan investigation being considered, the project request must be examined to determine preciselywhat the system requires.Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network(LAN).

3.7 FEASIBILITY ANALYSIS

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

Operational Feasibility

Economic Feasibility

Technical Feasibility

4.SYSTEM DESIGN

4.1 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 maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

4.2 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.

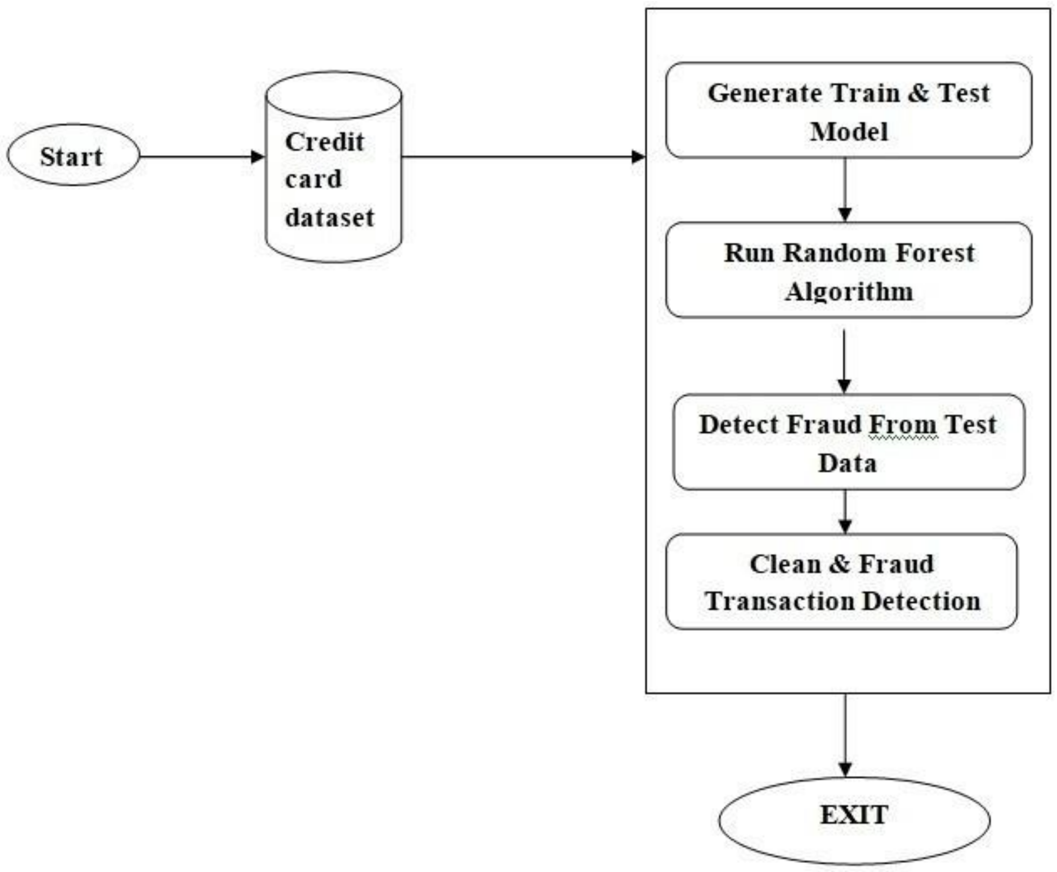
2.Select methods for presenting information.

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.

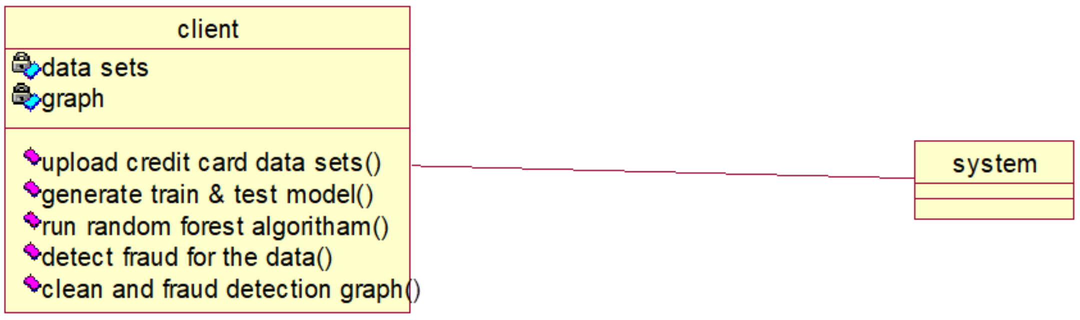
4.3 SYSTEM ARCHITECTURE:


Start
Credit card dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection
EXIT

4.4 UML DIAGRAMS:

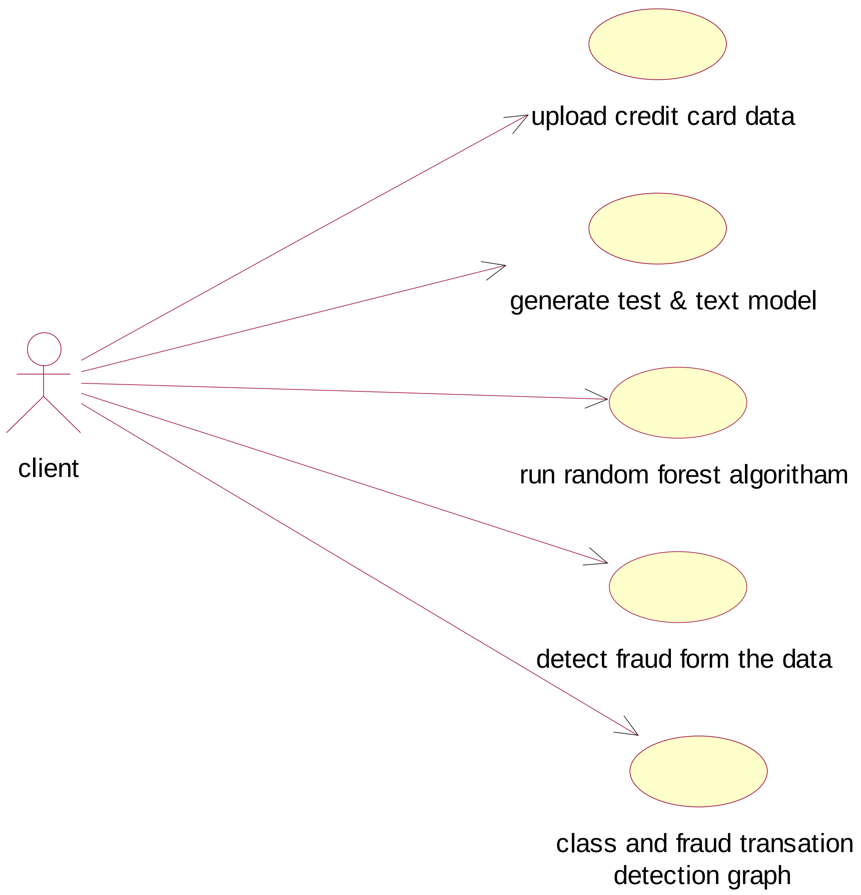
4.4.1 CLASS DIAGRAM:

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.


client
data sets
graph
upload credit card data sets()
generate train & test model()
run random forest algoritham()
detect fraud for the data()
clean and fraud detection graph()
system

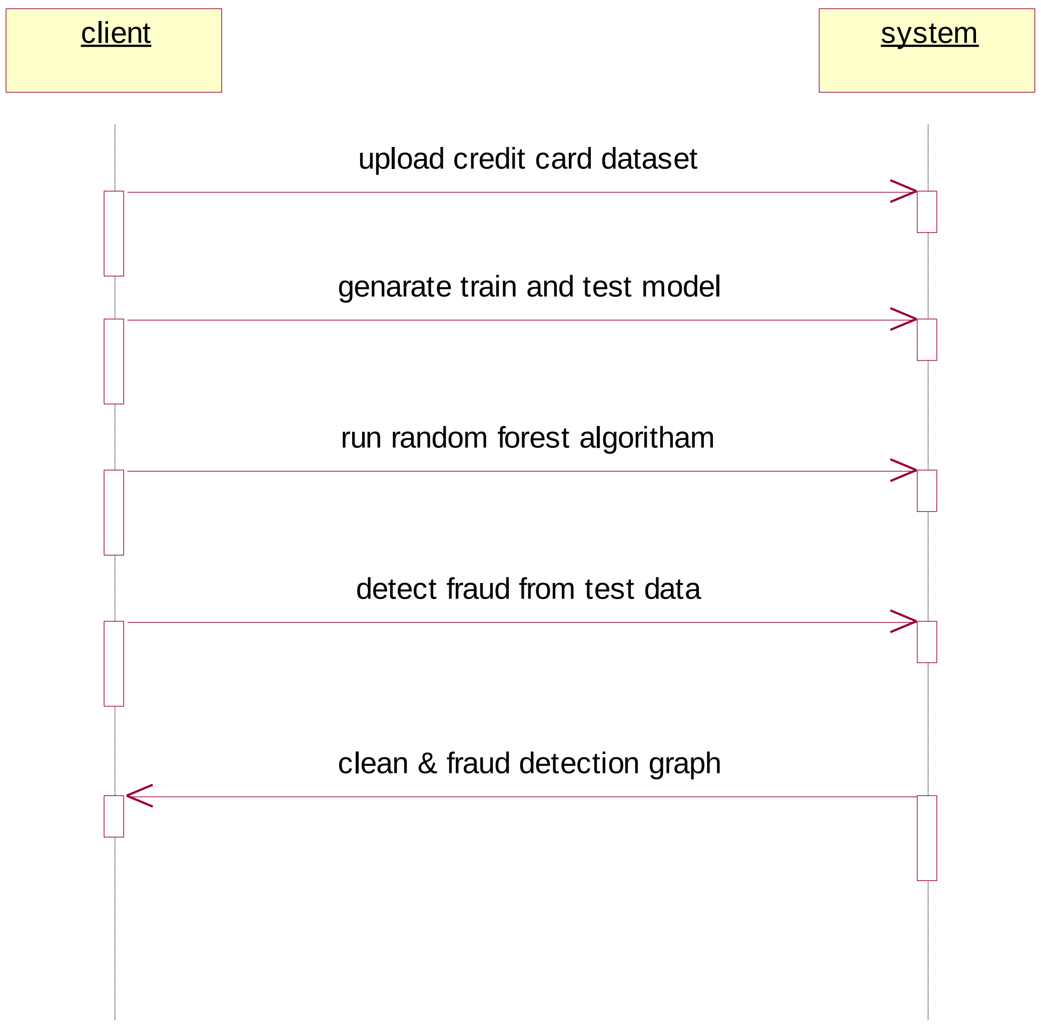
4.4.2 USE CASE DIAGRAM:

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. Roles of the actors in the system can be depicted.


upload credit card data
generate test & text model
client
run random forest algoritham
detect fraud form the data
class and fraud transation detection graph

4.4.3 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 diagrams, event scenarios, and timing diagrams.


client
system
upload credit card dataset
genarate train and test model
run random forest algoritham
detect fraud from test data
clean & fraud detection graph

5.SOFTWARE ENVIRONMEMNT

5.1 INTRODUCTION TO PYTHON:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type 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

What is Python:

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do:

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.

- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages:

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install:

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

```
print("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

Or, if the "python" command did not work, you can try "py":

```
C:\Users\Your Name>py
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

 >>> print("Hello, World!")
```

```
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Virtual Environments and Packages:

Introduction:

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

```
python3 -m venv tutorial-env
```

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

```
tutorial-env\Scripts\activate.bat
```

On Unix or MacOS, run:

```
source tutorial-env/bin/activate
```

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate
(tutorial-env) $ python
```

```
    Python 3.5.1 (default, May  6 2016, 10:59:36)

  ...
>>> import sys
>>> sys.path
['', '/usr/local/lib/python35.zip', ...,
'~/envs/tutorial-env/lib/python3.5/site-packages']
>>>
```

12.3. Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature:

```
 (tutorial-env) $ pip search astronomy
skyfield              - Elegant astronomy for Python
gary                  - Galactic astronomy and gravitational dynamics.
novas                 - The United States Naval Observatory NOVAS astronomy library
astroobs              - Provides astronomy ephemeris to plan telescope observations
PyAstronomy           - A collection of astronomy related tools for Python.
...
```

pip has a number of subcommands: "search", "install", "uninstall", "freeze", etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package's name:

```
 (tutorial-env) $ pip install novas
Collecting novas
  Downloading novas-3.1.1.3.tar.gz (136kB)
Installing collected packages: novas
  Running setup.py install for novas
Successfully installed novas-3.1.1.3
```

You can also install a specific version of a package by giving the package name followed by == and the version number:

```
 (tutorial-env) $ pip install requests==2.6.0
Collecting requests==2.6.0
 Using cached requests-2.6.0-py2.py3-none-any.whl
```

```
Installing collected packages: requests

Successfully installed requests-2.6.0
```

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

```
(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0
```

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

```
(tutorial-env) $ pip show requests

---

Metadata-Version: 2.0
```

```
Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:
```

pip list will display all of the packages installed in the virtual environment:

```
(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)
```

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

```
(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt
```

```
novas==3.1.1.3

numpy==1.9.2

requests==2.7.0
```

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

```
(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

  ...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

  ...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

  ...

Installing collected packages: novas, numpy, requests

  Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0
```

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

Cross Platform:platform.architecture(executable=sys.executable, bits='', linkage='') Queries the given executable (defaults to the Python interpreter binary) for various architecture information.

Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cannot be determined are returned as given by the parameter presets. If bits is given as '', the sizeof(pointer) (or sizeof(long) on Python version < 1.5.2) is used as indicator for the supported pointer size.

The function relies on the system's file command to do the actual work. This is available on most if not all Unix platforms and some non-Unix platforms and then only if the executable points to the Python interpreter. Reasonable defaults are used when the above needs are not met.

Note On Mac OS X (and perhaps other platforms), executable files may be universal files containing multiple architectures.

To get at the "64-bitness" of the current interpreter, it is more reliable to query the sys.maxsize attribute:

is_64bits = sys.maxsize > 2**32

platform.machine()

Returns the machine type, e.g. 'i386'. An empty string is returned if the value cannot be determined.

platform.node()

Returns the computer's network name (may not be fully qualified!). An empty string is returned if the value cannot be determined.

platform.platform(aliased=0, terse=0)

Returns a single string identifying the underlying platform with as much useful information as possible.

The output is intended to be human readable rather than machine parseable. It may look different on different platforms and this is intended.

If aliased is true, the function will use aliases for various platforms that report system names which differ from their common names, for example SunOS will be reported as Solaris. The system_alias() function is used to implement this.

Setting terse to true causes the function to return only the absolute minimum information needed to identify the platform.

platform.processor()

Returns the (real) processor name, e.g. 'amdk6'.

An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

platform.python_build()

Returns a tuple (buildno, builddate) stating the Python build number and date as strings.

platform.python_compiler()

Returns a string identifying the compiler used for compiling Python.

platform.python_branch()

Returns a string identifying the Python implementation SCM branch.

New in version 2.6.

platform.python_implementation()

Returns a string identifying the Python implementation. Possible return values are: ‘CPython’, ‘IronPython’, ‘Jython’, ‘PyPy’.

New in version 2.6.

platform.python_revision()

Returns a string identifying the Python implementation SCM revision.

New in version 2.6.

platform.python_version()

Returns the Python version as string 'major.minor.patchlevel'.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to 0).

platform.python_version_tuple()

Returns the Python version as tuple (major, minor, patchlevel) of strings.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to '0').

platform.release()

Returns the system’s release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cannot be determined.

platform.system()

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cannot be determined.

platform.system_alias(system, release, version)

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion.

platform.version()

Returns the system’s release version, e.g. '#3 on degas'. An empty string is returned if the value cannot be determined.

platform.uname()

Fairly portable uname interface. Returns a tuple of strings (system, node, release, version, machine, processor) identifying the underlying platform.

Note that unlike the os.uname() function this also returns possible processor information as additional tuple entry.

Entries which cannot be determined are set to ''.

Java Platform

platform.java_ver(release='', vendor='', vminfo=('', '', ''), osinfo=('', '', ''))

Version interface for Jython.

Returns a tuple (release, vendor, vminfo, osinfo) with vminfo being a tuple (vm_name, vm_release, vm_vendor) and osinfo being a tuple (os_name, os_version, os_arch). Values which cannot be determined are set to the defaults given as parameters (which all default to '').

Windows Platform

platform.win32_ver(release='', version='', csd='', ptype='')

Get additional version information from the Windows Registry and return a tuple (release, version, csd, ptype) referring to OS release, version number, CSD level (service pack) and OS type (multi/single processor).

As a hint: ptype is 'Uniprocessor Free' on single processor NT machines and 'Multiprocessor Free' on multi processor machines. The ‘Free’ refers to the OS version being free of debugging code. It could also state ‘Checked’ which means the OS version uses debugging code, i.e. code that checks arguments, ranges, etc.

Note This function works best with Mark Hammond’s win32all package installed, but also on Python 2.3 and later (support for this was added in Python 2.6). It obviously only runs on Win32 compatible platforms.

Win95/98 specific:

platform.popen(cmd, mode='r', bufsize=None)

Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

Mac OS Platform:

platform.mac_ver(release='', versioninfo=('', '', ''), machine='')

Get Mac OS version information and return it as tuple (release, versioninfo, machine) with version info being a tuple (version, dev_stage, non_release_version).

Entries which cannot be determined are set to ''. All tuple entries are strings.

Unix Platforms:

platform.dist(distname='', version='', id='', supported_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...))

This is an old version of the functionality now provided by linux_distribution(). For new code, please use the linux_distribution().

The only difference between the two is that dist() always returns the short name of the distribution taken from the supported_dists parameter.

Deprecated since version 2.6.

platform.linux_distribution(distname='', version='', id='', supported_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...), full_distribution_name=1)

Tries to determine the name of the Linux OS distribution name.

supported_dists may be given to define the set of Linux distributions to look for. It defaults to a list of currently supported Linux distributions identified by their release file name.

If full_distribution_name is true (default), the full distribution read from the OS is returned. Otherwise the short name taken from supported_dists is used.

Returns a tuple (distname,version,id) which defaults to the args given as parameters. id is the item in parentheses after the version number. It is usually the version codename.

Note This function is deprecated since Python 3.5 and removed in Python 3.8. See alternative like the distro package.

New in version 2.6.

platform.libc_ver(executable=sys.executable, lib='', version='', chunksize=2048)

Tries to determine the libc version against which the file executable (defaults to the Python interpreter) is linked. Returns a tuple of strings (lib, version) which default to the given parameters in case the lookup fails.

Note that this function has intimate knowledge of how different libc versions add symbols to the executable is probably only usable for executables compiled using gcc. The file is read and scanned in chunks of chunksize bytes.

Using the Python Interpreter:

Invoking the Interpreter

The Python interpreter is usually installed as /usr/local/bin/python3.8 on those machines where it is available; putting /usr/local/bin in your Unix shell’s search path makes it possible to start it by typing the command:

python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn't work, you can exit the interpreter by typing the following command: quit().

The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

All command line options are described in Command line and environment.

Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c command is used, sys.argv[0] is set to '-c'. When -m module is used, sys.argv[0] is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter's option processing but left in sys.argv for the command or module to handle.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$ python3.8

Python 3.8 (default, Sep 16 2015, 09:25:04)

[GCC 4.8.2] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>>

>>> the_world_is_flat = True

>>> if the_world_is_flat:

...     print("Be careful not to fall off!")

...

Be careful not to fall off!
```

For more on interactive mode, see Interactive Mode.

The Interpreter and Its Environment

Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

```
# -*- coding: encoding -*-
```

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

```
# -*- coding: cp1252 -*-
```

One exception to the first line rule is when the source code starts with a UNIX “shebang” line. In this case, the encoding declaration should be added as the second line of the file. For example:

```
#!/usr/bin/env python3

# -*- coding: cp1252 -*-
```

5.2 Introduction to Artificial Intelligence:

“The science and engineering of making intelligent machines, especially intelligent computer programs”. -John McCarthy-

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems.The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

Trending AI Articles:

1. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data
2. Data Science Simplified Part 1: Principles and Process
3. Getting Started with Building Realtime API Infrastructure
4. AI & NLP Workshop

Applications of AI:

Gaming − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess,river crossing, N-queens problems and etc.

Natural Language Processing − Interact with the computer that understands natural language spoken by humans.

· Expert Systems − Machine or software provide explanation and advice to the users.

· Vision Systems − Systems understand, explain, and describe visual input on the computer.

· Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.

· Handwriting Recognition − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.

· Intelligent Robots − Robots are able to perform the instructions given by a human.

Major Goals:

- Knowledge reasoning
- Planning
- Machine Learning
- Natural Language Processing
- Computer Vision
- Robotics

IBM Watson:

“Watson” is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a “question answering” machine. The supercomputer is named for IBM’s founder, Thomas J. Watson.

IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that “more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations.” recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and authorities. It isn’t any longer completely a request answering figuring system arranged from Q&A joins yet can now ‘see’, ‘hear’, ‘read’, ‘talk’, ‘taste’, ‘translate’, ‘learn’ and ‘endorse’.

Machine Learning

5.3 Introduction Machine Learning

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we'll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We'll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we'll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

Supervised Learning:

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as `fish` and images of oceans labeled as `water`. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as `fish` and unlabeled ocean images as `water`.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Unsupervised Learning:

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

Approaches:

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. **Correlation** is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

Approaches to machine learning are continuously being developed. For our purposes, we'll go through a few of the popular approaches that are being used in machine learning at the time of writing.

Decision Tree Learning:

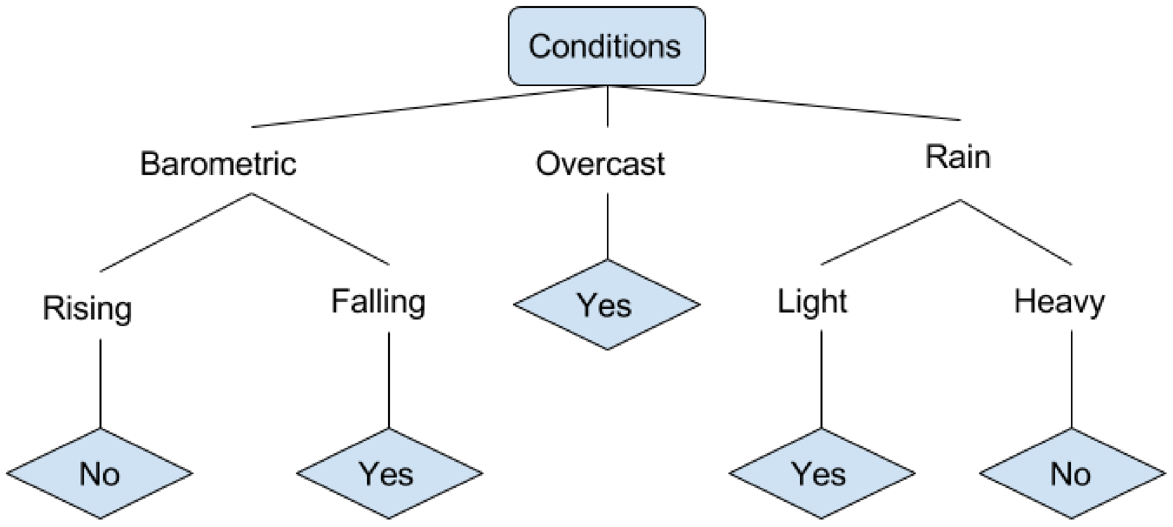
For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data's target value.

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data's attributes that are determined through observation are represented by the branches, while the conclusions about the data's target value are represented in the leaves.

When "learning" a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let's look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.



In the simplified decision tree above, an example is classified by sorting it through the tree to the appropriate leaf node. This then returns the classification associated with the particular leaf, which in this case is either a `Yes` or a `No`. The tree classifies a day's conditions based on whether or not it is suitable for going fishing.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

5.4 Introduction to Deep Learning:

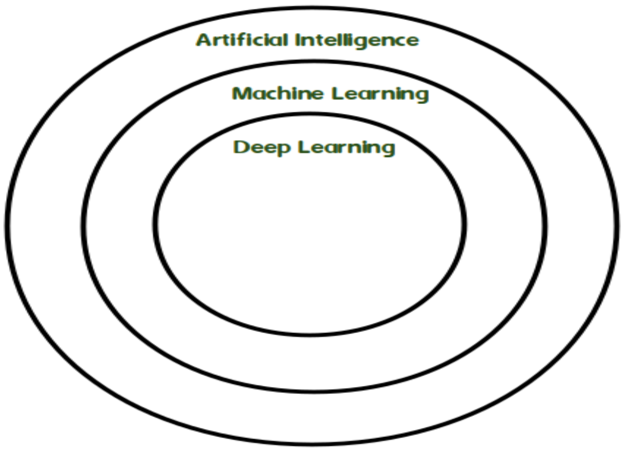
What is deep learning

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don't need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and a lot of data.

As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.
A formal definition of deep learning is- neurons

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours.
The question here is how do we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.



Difference between Machine Learning and Deep Learning :

MACHINE LEARNING	DEEP LEARNING
Works on small amount of Dataset for accuracy.	Works on Large amount of Dataset.
Dependent on Low-end Machine.	Heavily dependent on High-end Machine.
Divides the tasks into sub-tasks, solves them individually and finally combine the results.	Solves problem end to end.
Takes less time to train.	Takes longer time to train.
Testing time may increase.	Less time to test the data.

5.5 SOURCE CODE

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import collections
from datetime import datetime
import hashlib
import os.path
import random
import re
import sys
import tarfile
import numpy as np
from six.moves import urllib
```

```
import tensorflow as tf

from tensorflow.python.framework import graph_util
from tensorflow.python.framework import tensor_shape
from tensorflow.python.platform import gfile
from tensorflow.python.util import compat

FLAGS = None
MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1  # ~134M
def create_image_lists(image_dir, testing_percentage, validation_percentage):
  if not gfile.Exists(image_dir):
    tf.logging.error("Image directory '" + image_dir + "' not found.")
    return None
  result = collections.OrderedDict()
  sub_dirs = [
    os.path.join(image_dir,item)

for item in gfile.ListDirectory(image_dir)]
  sub_dirs = sorted(item for item in sub_dirs
                    if gfile.IsDirectory(item))
  for sub_dir in sub_dirs:
    extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
    file_list = []
    dir_name = os.path.basename(sub_dir)
    if dir_name == image_dir:
      continue
    tf.logging.info("Looking for images in '" + dir_name + "'")
    for extension in extensions:
      file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
      file_list.extend(gfile.Glob(file_glob))
    if not file_list:
      tf.logging.warning('No files found')
      continue
    if len(file_list) < 20:
      tf.logging.warning(
          'WARNING: Folder has less than 20 images, which may cause issues.')
    elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
      tf.logging.warning(
          'WARNING: Folder {} has more than {} images. Some images will '
          'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
    label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
    training_images = []
    testing_images = []
    validation_images = []
    for file_name in file_list:
      base_name = os.path.basename(file_name)
      hash_name = re.sub(r'_nohash_.*$', '', file_name)
      hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
      percentage_hash = ((int(hash_name_hashed, 16) %
                          (MAX_NUM_IMAGES_PER_CLASS + 1)) *
                         (100.0 / MAX_NUM_IMAGES_PER_CLASS))
      if percentage_hash < validation_percentage:
        validation_images.append(base_name)
      elif percentage_hash < (testing_percentage + validation_percentage):
        testing_images.append(base_name)
      else:
        training_images.append(base_name)
    result[label_name] = {
        'dir': dir_name,
        'training': training_images,
        'testing': testing_images,
        'validation': validation_images,
    }
  return result
```

6.SYSTEM TEST

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.

6.1TYPES OF TESTS

6.1.1Unit 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.

6.1.2 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 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.

6.1.3 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 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.

6.1.4 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 in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is 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, 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.

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.

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.

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.

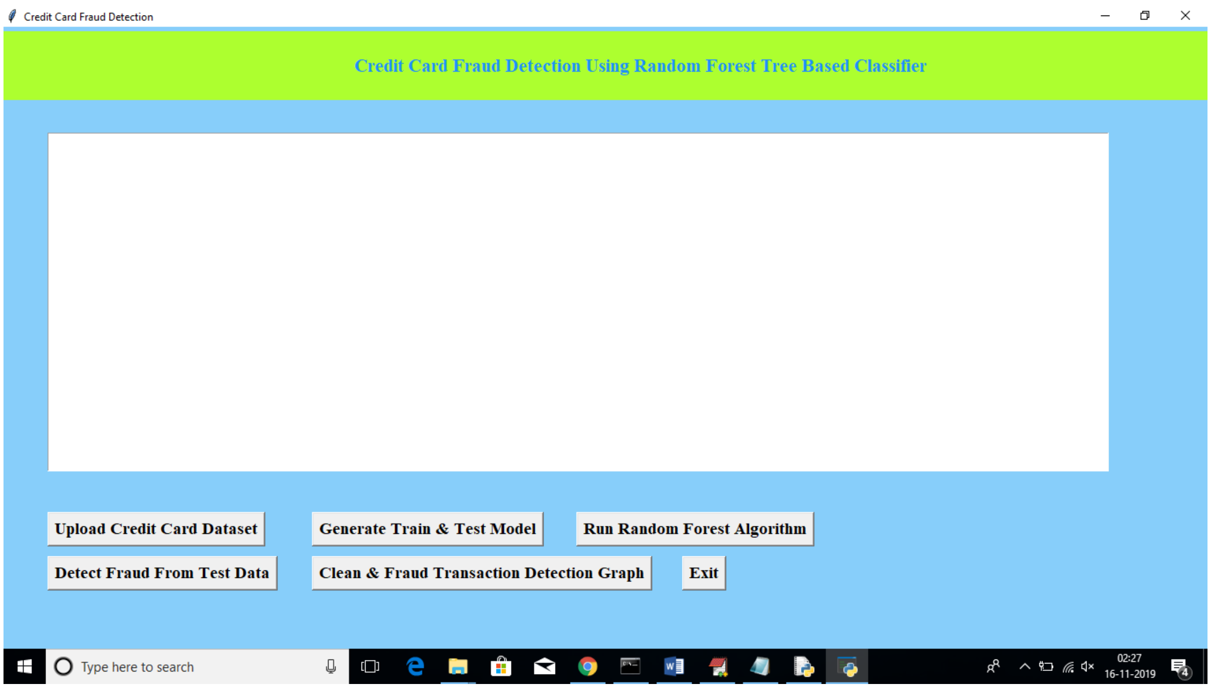
6.1.5 Test Case

Test Case Id	Test Case Name	Test Case Description	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Start the Application	Host the application and test if it starts making sure the required software is available	If it doesn't Start	We cannot run the application	The application hosts success.	High	High
02	Home Page	Check the deployment environment for properly loading the application.	If it doesn’t load.	We cannot access the application	The application is running successfully .	High	High
03	User Mode	Verify the working of the application in freestyle mode	If it doesn’t Respond	We cannot use the Freestyle mode.	The application displays the Freestyle Page	High	High
04	Data Input	Verify if the application takes input and updates	If it fails to take the input or store in The Database	We cannot proceed further	The application updates the input to application	High	High

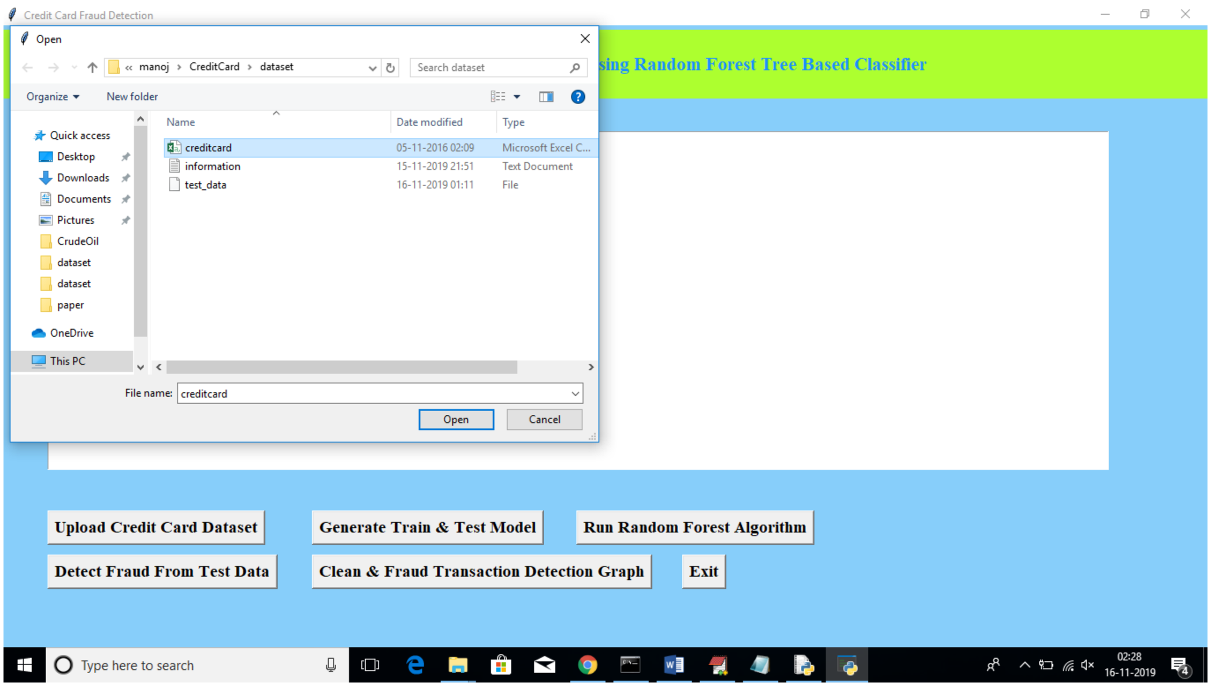
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.OUTPUT SCREENS

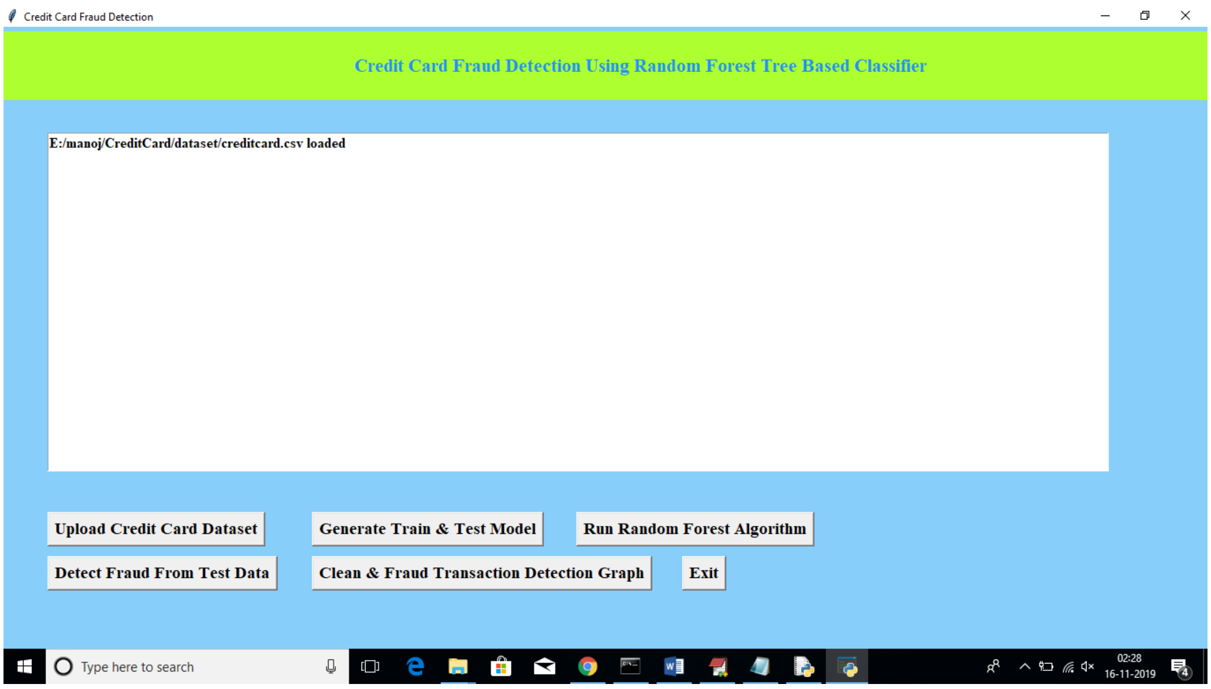
Double click on 'run.bat' file to get below screen


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:27
16-11-2019

In above screen click on 'Upload Credit Card Dataset' button to upload dataset

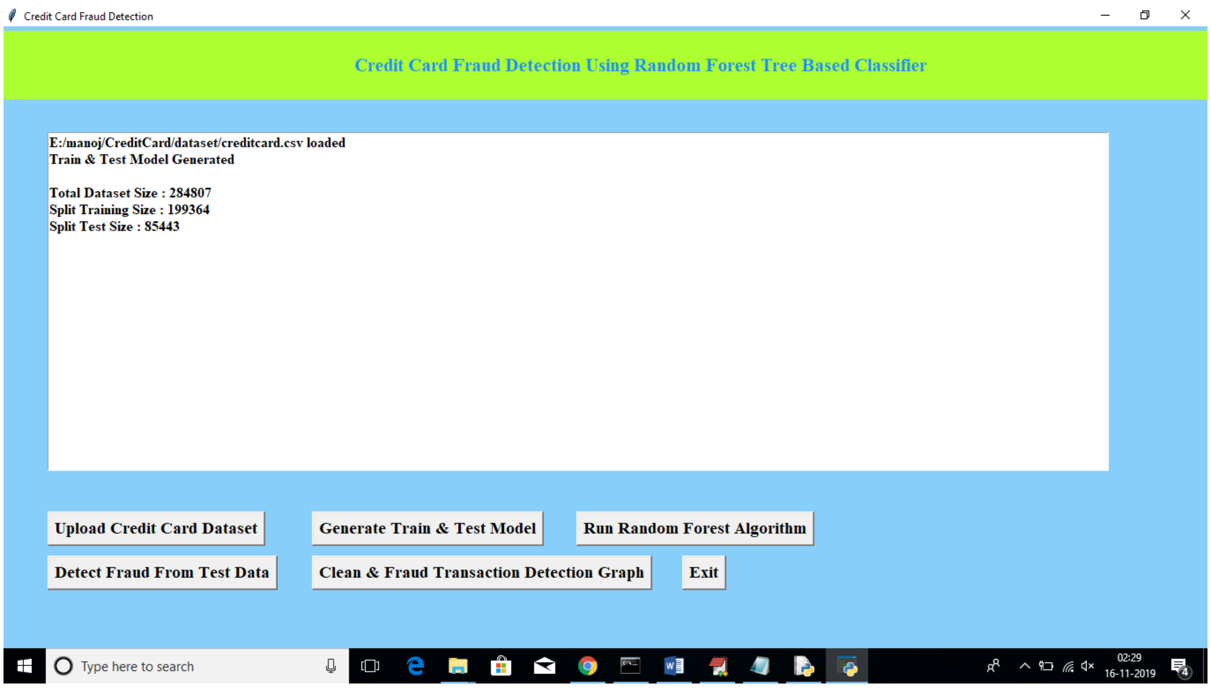

Credit Card Fraud Detection
Open
« manoj › CreditCard › dataset
Search dataset
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
CrudeOil
dataset
dataset
paper
OneDrive
This PC
Name
Date modified
Type
creditcard 05-11-2016 02:09 Microsoft Excel C...
information 15-11-2019 21:51 Text Document
test_data 16-11-2019 01:11 File
File name: creditcard
Open
Cancel
sing Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:28
16-11-2019

After uploading dataset will get below screen


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:28
16-11-2019

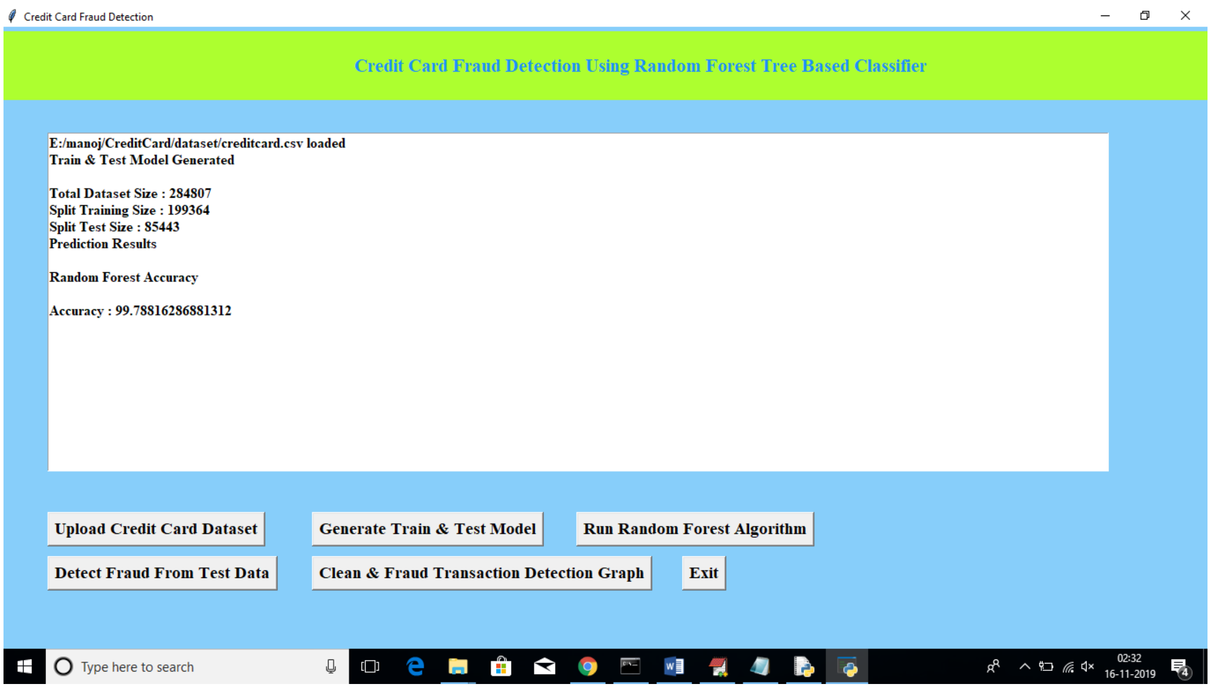
```
E:/manoj/CreditCard/dataset/creditcard.csv loaded
```

Now click on 'Generate Train & Test Model' to generate training model for Random Forest Classifier


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:29
16-11-2019

```
E:/manoj/CreditCard/dataset/creditcard.csv loaded
Train & Test Model Generated

Total Dataset Size : 284807
Split Training Size : 199364
Split Test Size : 85443
```

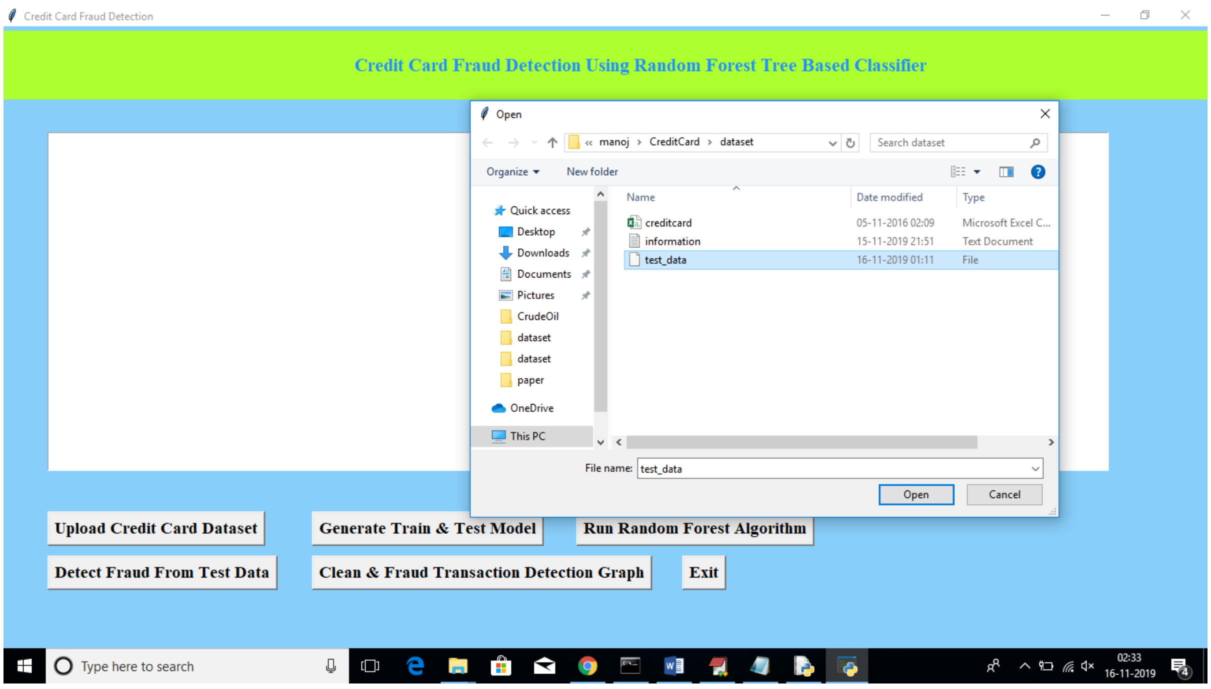
In above screen after generating model we can see total records available in dataset and then application using how many records for training and how many for testing. Now click on "Run Random Forest Algorithm' button to generate Random Forest model on train and test data.



```
E:/manoj/CreditCard/dataset/creditcard.csv loaded
Train & Test Model Generated

Total Dataset Size : 284807
Split Training Size : 199364
Split Test Size : 85443
Prediction Results

Random Forest Accuracy

Accuracy : 99.78816286881312
```

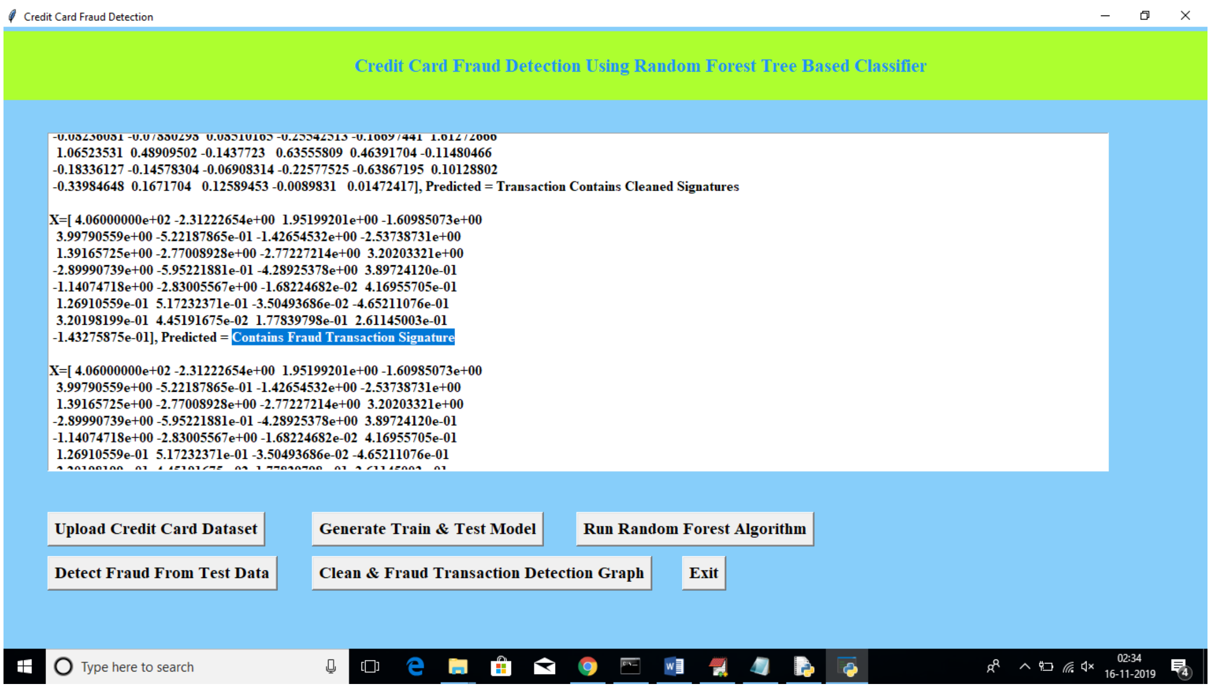
Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

In above screen we can see Random Forest generate 99.78% percent accuracy while building model on train and test data. Now click on ‘Detect Fraud From Test Data’ button to upload test data and to predict whether test data contains normal or fraud transaction.



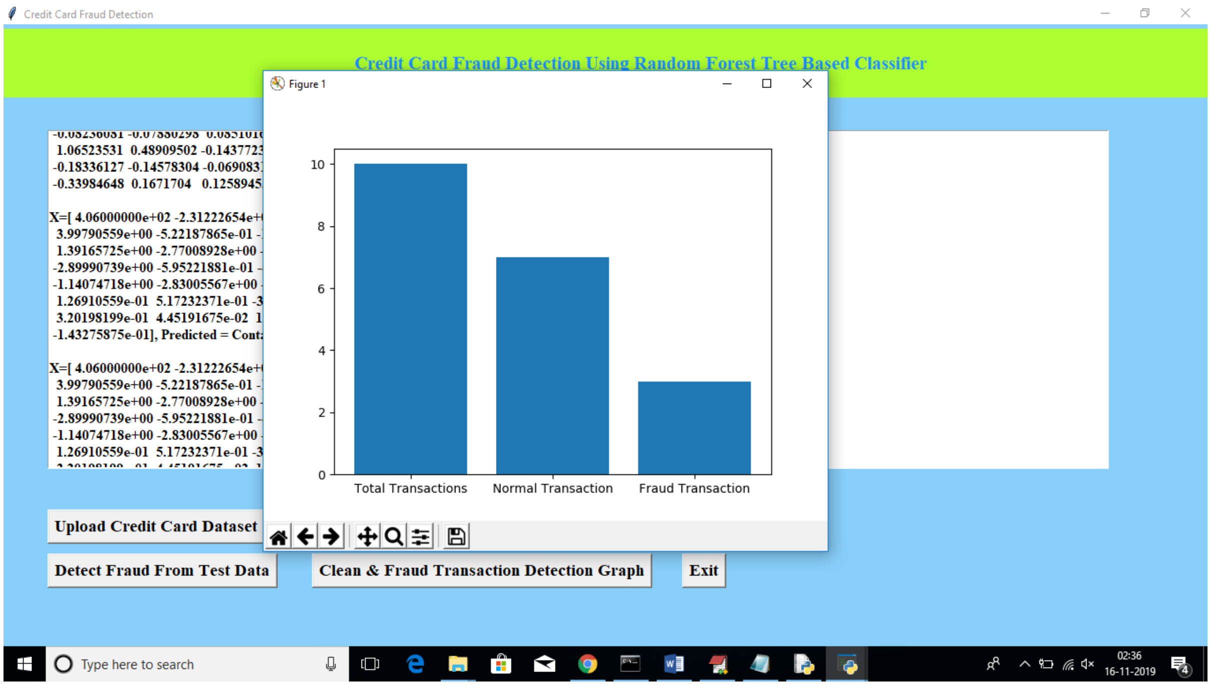
Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Open
manoj › CreditCard › dataset
Search dataset
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
CrudeOil
dataset
dataset
paper
OneDrive
This PC
Name
Date modified
Type
creditcard 05-11-2016 02:09 Microsoft Excel C...
information 15-11-2019 21:51 Text Document
test_data 16-11-2019 01:11 File
File name: test_data
Open
Cancel
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

In above screen I am uploading test dataset and after uploading test data will get below prediction details


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:34
16-11-2019

```
-0.08236081 -0.07880298  0.08510165 -0.25542513 -0.16697441  1.61272666
 1.06523531  0.48909502 -0.1437723   0.63555809  0.46391704 -0.11480466
-0.18336127 -0.14578304 -0.06908314 -0.22577525 -0.63867195  0.10128802
-0.33984648  0.1671704   0.12589453 -0.0089831   0.01472417], Predicted = Transaction Contains Cleaned Signatures

X=[ 4.06000000e+02 -2.31222654e+00  1.95199201e+00 -1.60985073e+00
 3.99790559e+00 -5.22187865e-01 -1.42654532e+00 -2.53738731e+00
 1.39165725e+00 -2.77008928e+00 -2.77227214e+00  3.20203321e+00
-2.89990739e+00 -5.95221881e-01 -4.28925378e+00  3.89724120e-01
-1.14074718e+00 -2.83005567e+00 -1.68224682e-02  4.16955705e-01
 1.26910559e-01  5.17232371e-01 -3.50493686e-02 -4.65211076e-01
 3.20198199e-01  4.45191675e-02  1.77839798e-01  2.61145003e-01
-1.43275875e-01], Predicted = Contains Fraud Transaction Signature

X=[ 4.06000000e+02 -2.31222654e+00  1.95199201e+00 -1.60985073e+00
 3.99790559e+00 -5.22187865e-01 -1.42654532e+00 -2.53738731e+00
 1.39165725e+00 -2.77008928e+00 -2.77227214e+00  3.20203321e+00
-2.89990739e+00 -5.95221881e-01 -4.28925378e+00  3.89724120e-01
-1.14074718e+00 -2.83005567e+00 -1.68224682e-02  4.16955705e-01
 1.26910559e-01  5.17232371e-01 -3.50493686e-02 -4.65211076e-01
```

In above screen beside each test data application will display output as whether transaction contains cleaned or fraud signatures. Now click on ‘Clean & Fraud Transaction Detection Graph’ button to see total test transaction with clean and fraud signature in graphical format. See below screen


Credit Card Fraud Detection
Credit Card Fraud Detection Using Random Forest Tree Based Classifier
Figure 1
10
8
6
4
2
0
Total Transactions
Normal Transaction
Fraud Transaction
Upload Credit Card Dataset
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit
Type here to search
02:36
16-11-2019

In above graph we can see total test data and number of normal and fraud transaction detected. In above graph x-axis represents type and y-axis represents count of clean and fraud transaction

8. CONCLUSION :

The Random forest algorithm will perform better with a larger number of training data, but speed during testing and application will suffer. Application of more pre-processing techniques would also help. The SVM algorithm still suffers from the imbalanced dataset problem and requires more preprocessing to give better results at the results shown by SVM is great but it could have been better if more preprocessing have been done on the data.

9. REFERENCES :

1) Sudhamathy G: Credit Risk Analysis and Prediction Modelling of Bank Loans Using R, vol. 8, no-5, pp. 1954-1966.

2) LI Changjian, HU Peng: Credit Risk Assessment for ural Credit Cooperatives based on Improved Neural Network, International Conference on Smart Grid and Electrical Automation vol. 60, no. - 3, pp 227-230, 2017.

3) Wei Sun, Chen-Guang Yang, Jian-Xun Qi: Credit Risk Assessment in Commercial Banks Based On Support Vector Machines, vol.6, pp 2430-2433, 2006.

4) Amlan Kundu, Suvasini Panigrahi, Shamik Sural, Senior Member, IEEE,"BLAST-SSAHA Hybridization for Credit Card Fraud Detection", vol. 6, no. 4 pp. 309-315, 2009.

5) Y. Sahin and E. Duman, "Detecting Credit Card Fraud by Decision Trees and Support Vector Machines, Proceedings of International Multi Conference of Engineers and Computer Scientists, vol. I, 2011.