CHAPTER 1

INTRODUCTION

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.

CHAPTER 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

2.1.1 ECONOMICAL FEASIBILITY

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

2.1.2 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 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.

2.1.3 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 3

LITERATURE SURVEY

3.1 The Use of Predictive Analytics Technology to Detect Credit Card Fraud in Canada.

Authors: “Kosemani Temitayo Hafiz, Dr. Shaun Aghili, Dr. Pavol Zavarsky.”

Abstract: This research paper focuses on the creation of a scorecard from relevant evaluation criteria, features, and capabilities of predictive analytics vendor solutions currently being used to detect credit card fraud. The scorecard provides a side-by-side comparison of five credit card predictive analytics vendor solutions adopted in Canada. From the ensuing research findings, a list of credit card fraud PAT vendor solution challenges, risks, and limitations was outlined.

3.2 BLAST-SSAHA Hybridization for Credit Card Fraud Detection.

Authors: “Amlan Kundu, Suvasini Panigrahi, Shamik Sural.”

Abstract: This paper proposed to use two-stage sequence alignment in which a profile Analyser (PA) first determines the similarity of an incoming sequence of transactions on a given credit card with the genuine cardholder’s past spending sequences. The unusual transactions traced by the profile analyser are next passed on to a deviation analyser (DA) for possible alignment with past fraudulent behavior. The final decision about the nature of a transaction is taken on the basis of the observations by these two analyzers. In order to achieve online response time for both PA and DA, we suggest a new approach for combining two sequence alignment algorithms BLAST and SSAHA.

3.3 Research on Credit Card Fraud Detection Model Based on Distance Sum.

Authors: “Wen-Fang YU, Na Wang”.

Abstract: Along with increasing credit cards and growing trade volume in China, credit card fraud rises sharply. How to enhance the detection and prevention of credit card fraud becomes the focus of risk control of banks. It proposes a credit card fraud detection model using outlier detection based on distance sum according to the infrequency and unconventionality of fraud in credit card transaction data, applying outlier mining into credit card fraud detection. Experiments show that this model is feasible and ac

3.4 Fraudulent Detection in Credit Card System Using SVM & Decision Tree.

Authors: "Vijayshree B. Nipane, Poonam S. Kalinge,

Abstract: With growing advancement in the electronic commerce field, fraud is spreading all over the world, causing major financial losses. In current scenario, Major cause of financial losses is credit card fraud; it not only affects trades person but also individual clients. Decision tree, Genetic algorithm, Meta learning strategy, neural network, HMM are the presented methods used to detect credit card frauds. In contemplate system for fraudulent detection, artificial intelligence concept of Support Vector Machine (SVM) & decision tree is being used to solve the problem. Thus by implementation of this hybrid approach, financial losses can be reduced to greater extend.

3.5 Supervised Machine (SVM) Learning for Credit Card Fraud Detection.

Authors: "Sitaram Patel, Sunita Gond".

Abstract: This thesis proposed the SVM (Support Vector Machine) based method with multiple kernel involvement which also includes several fields of user profile instead of only spending profile. The simulation result shows improvement in TP (true positive), TN (true negative) rate, & also decreases the FP (false positive) & FN (false negative) rate.

CHAPTER 4

RESEARCH METHODOLOGY

4.1 EXISTING SYSTEM

In existing System, are search about a case study involving credit card fraud detection, where data normalization is applied before Cluster Analysis and with results obtained from the use of Cluster Analysis and Artificial Neural Networks on fraud detection has shown that by clustering attributes neuronal inputs can be minimized. And promising results can be obtained by using normalized data and data should be MLP trained. This research was based on unsupervised learning.

Significance of this paper was to find new methods for fraud detection and to increase the accuracy of results. The data set for this paper is based on real life transactional data by a large European company and personal details in data is kept confidential. Accuracy of an algorithm is around 50%. Significance of this paper was to find an algorithm and to reduce the cost measure. The result obtained was by 23% and the algorithm they find was Bayes minimum risk.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. In this paper a new collative comparison measure that reasonably represents the gains and losses due to fraud detection is proposed.

2. A cost sensitive method which is based on Bayes minimum risk is presented using the proposed cost measure.

4.2 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.

4.2.1 ADVANTAGES OF PROPOSED SYSTEM

- The importance of variables in a regression or classification problem in a natural way can be done by Xgboost.
- The transaction amount Feature 'class' is the target class for the binary classification and it takes value 1 for positive case (fraud) and 0 for negative case (not fraud)

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 Hardware System Configuration:

➢ Processor	-	Pentium –IV
➢ RAM	-	4 GB (min)
➢ Hard Disk	-	20 GB
➢ Key Board	-	Standard Windows Keyboard
➢ Mouse	-	Two or Three Button Mouse
➢ Monitor	-	SVGA

5.2 Software System Configuration:

➢ Operating System	-	Windows 7 Ultimate
➢ Coding Language	-	Python
➢ Front-End	-	Python
➢ Back-End	-	Django-ORM
➢ Designing	-	Html, CSS, JavaScript.
➢ Data Base	-	MySQL (WAMP Server).

5.3 FUNCTIONAL REQUIREMENTS:

1.Data Collection
2.Data Preprocessing
3.Training and Testing
4.Modiling
5.Predicting

5.4 NON-FUNCTIONAL REQUIREMENTS:

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are>10000. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement

CHAPTER 6

SYSTEM ARCHITECTURE

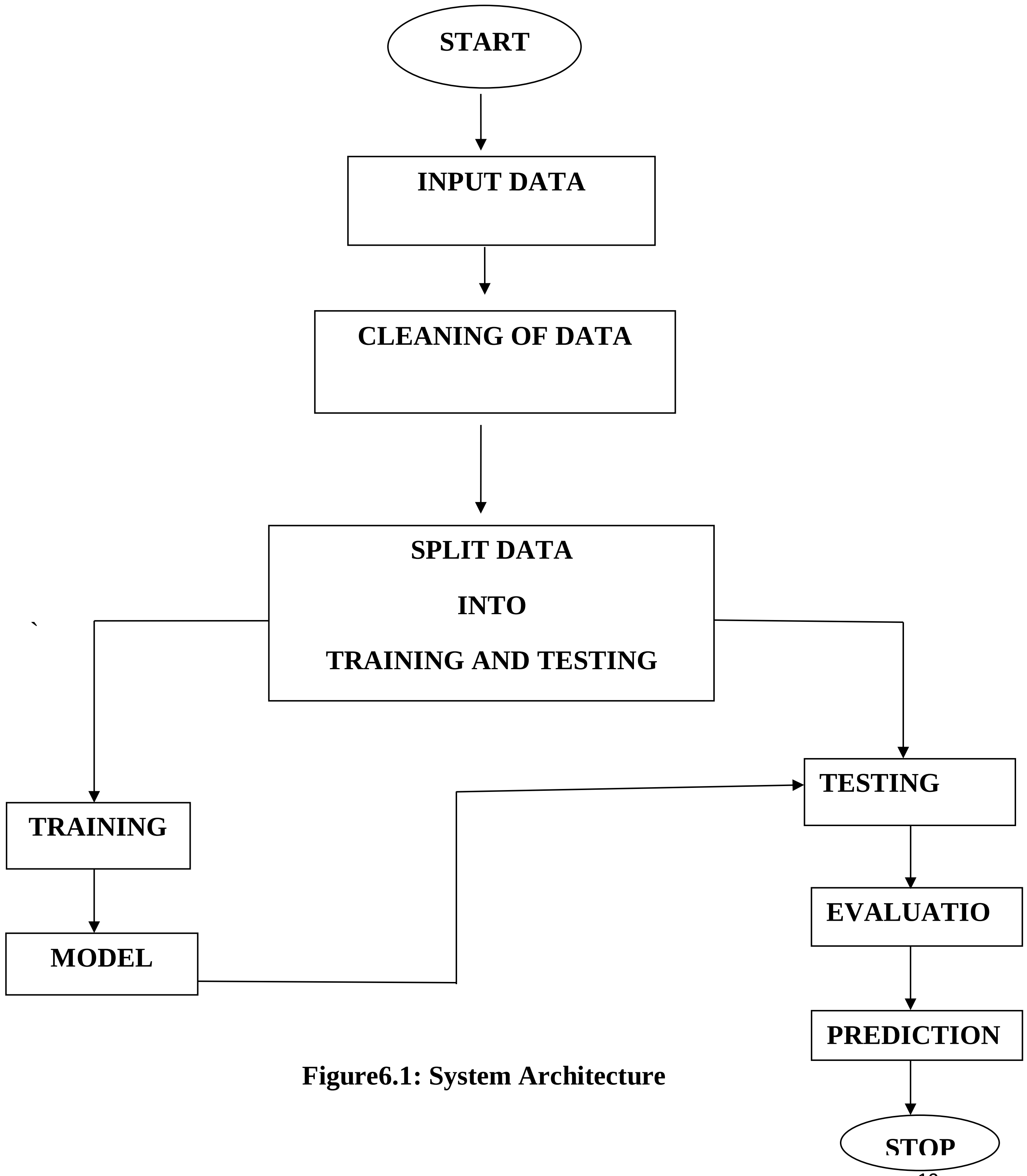


Figure6.1: System Architecture

CHAPTER 7

MODULES

7.1 UPLOAD DATA

The data resource to database can be uploaded by both administrator and authorized user. The data can be uploaded with key in order to maintain the secrecy of the data that is not released without knowledge of user. The users are authorized based on their details that are shared to admin and admin can authorize each user. Only Authorized users are allowed to access the system and upload or request for files.

7.2 ACCESS DETAILS

The access of data from the database can be given by administrators. Uploaded data are managed by admin and admin is the only person to provide the rights to process the accessing details and approve or unapproved users based on their details.

7.3 USER PERMISSIONS

The data from any resources are allowed to access the data with only permission from administrator. Prior to access data, users are allowed by admin to share their data and verify the details which are provided by user. If user is access the data with wrong attempts then, users are blocked accordingly. If user is requested to unblock them, based on the requests and previous activities admin is unblock users.

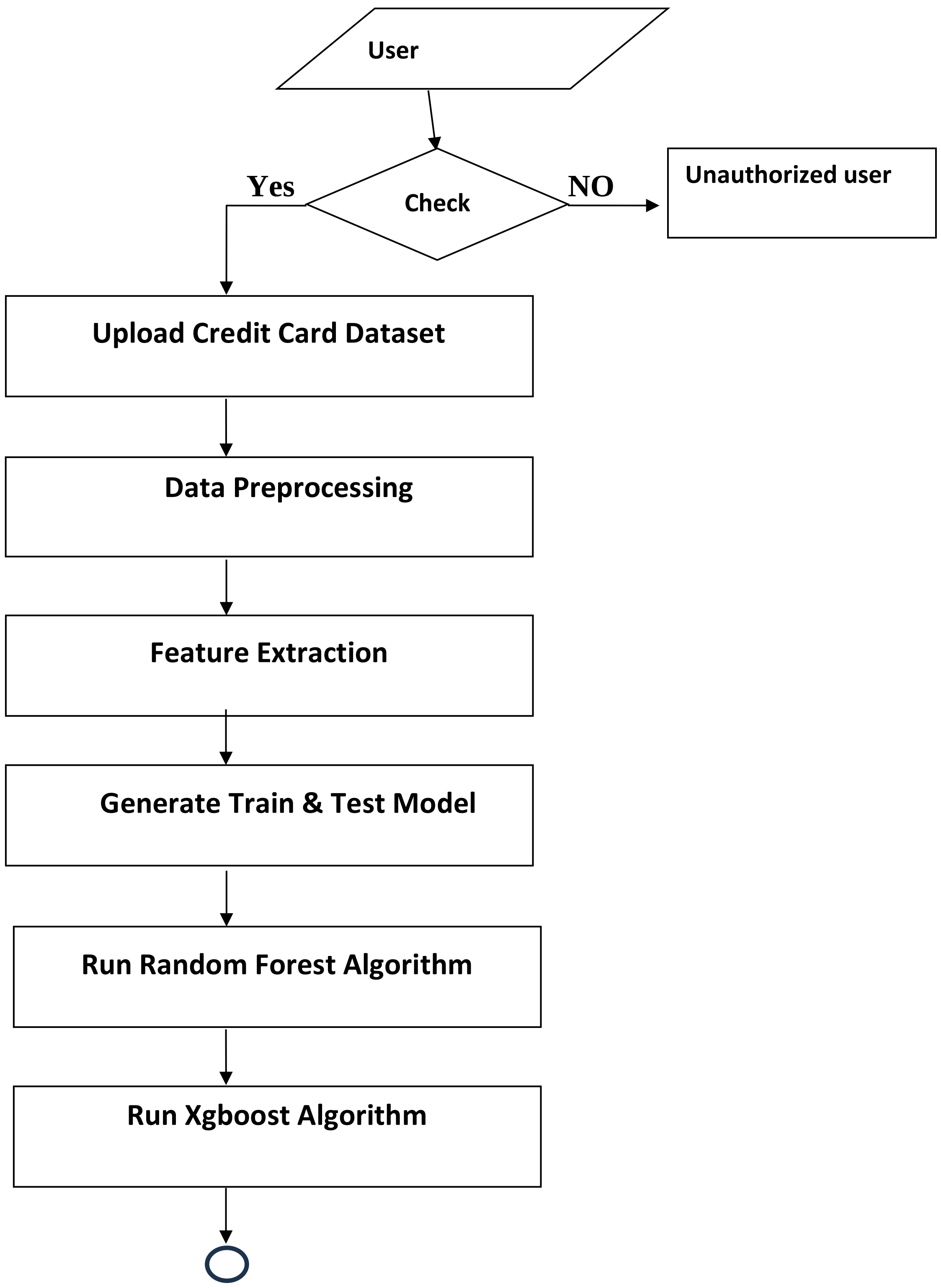
7.4 DATA ANALYSIS

Data analyses are done with the help of graph. The collected data are applied to graph in order to get the best analysis and prediction of dataset and given data policies.

CHAPTER 8

UML DIAGRAMS

8.1 Data Flow Diagram



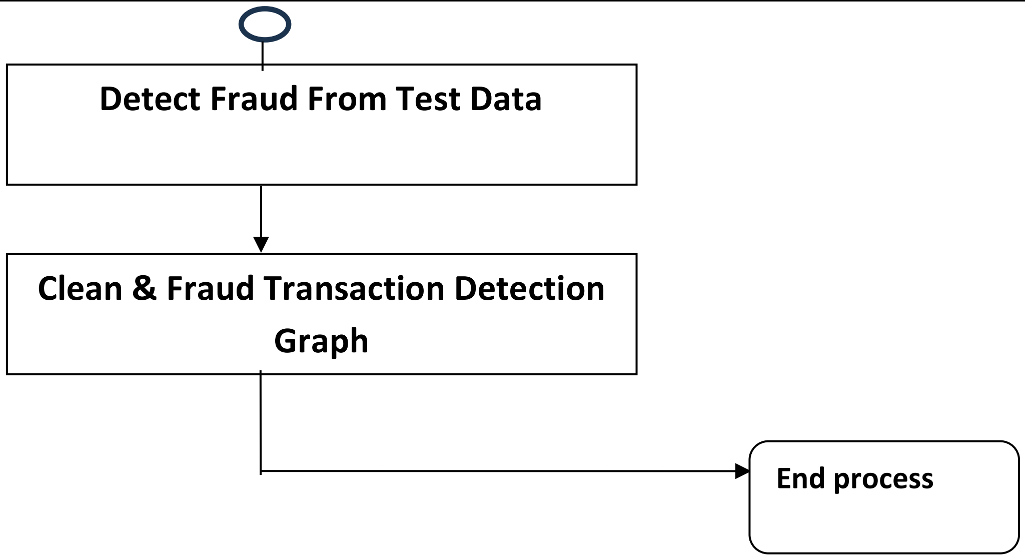

Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
End process

Figure 8.1: Data Flow Diagram

8.2 Use Case Diagram:

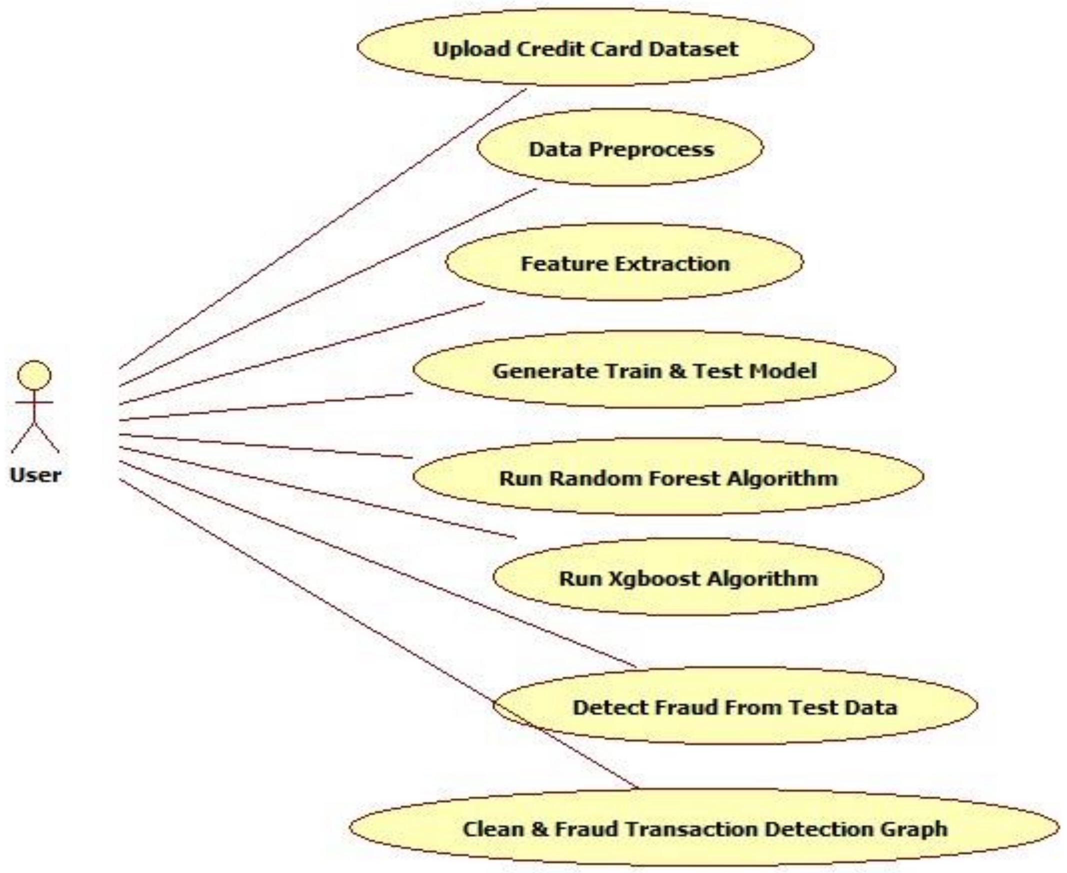

User
Upload Credit Card Dataset
Data Preprocess
Feature Extraction
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph

Figure 8.2: Use Case Diagram

Class Diagram:

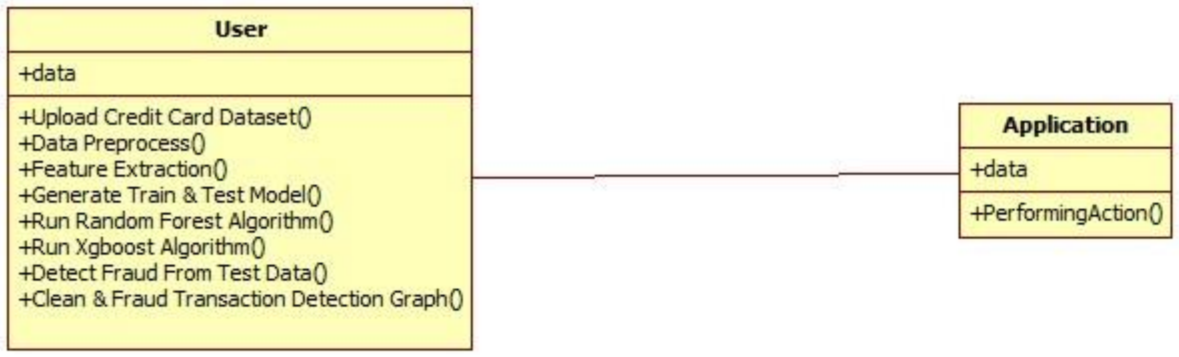

User
+data
+Upload Credit Card Dataset()
+Data Preprocess()
+Feature Extraction()
+Generate Train & Test Model()
+Run Random Forest Algorithm()
+Run Xgboost Algorithm()
+Detect Fraud From Test Data()
+Clean & Fraud Transaction Detection Graph()
Application
+data
+PerformingAction()

Figure 8.3: Class Diagram

8.3 Sequence Diagram:

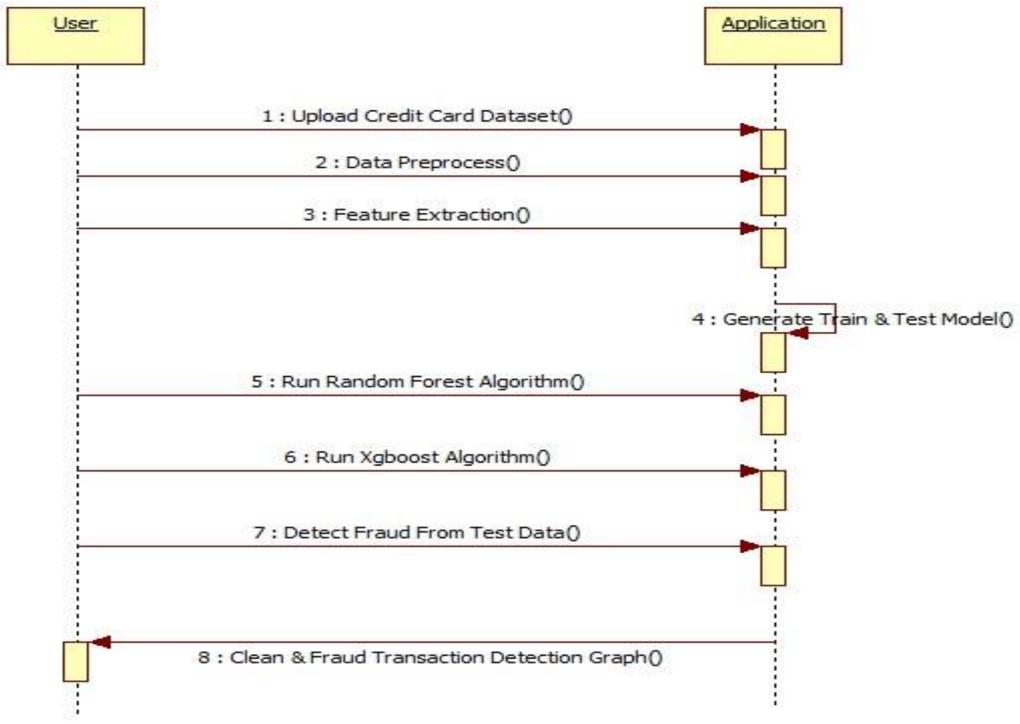

User
Application
1 : Upload Credit Card Dataset()
2 : Data Preprocess()
3 : Feature Extraction()
4 : Generate Train & Test Model()
5 : Run Random Forest Algorithm()
6 : Run Xgboost Algorithm()
7 : Detect Fraud From Test Data()
8 : Clean & Fraud Transaction Detection Graph()

Figure 8.4: Sequence Diagram

8.4 Activity Diagram:

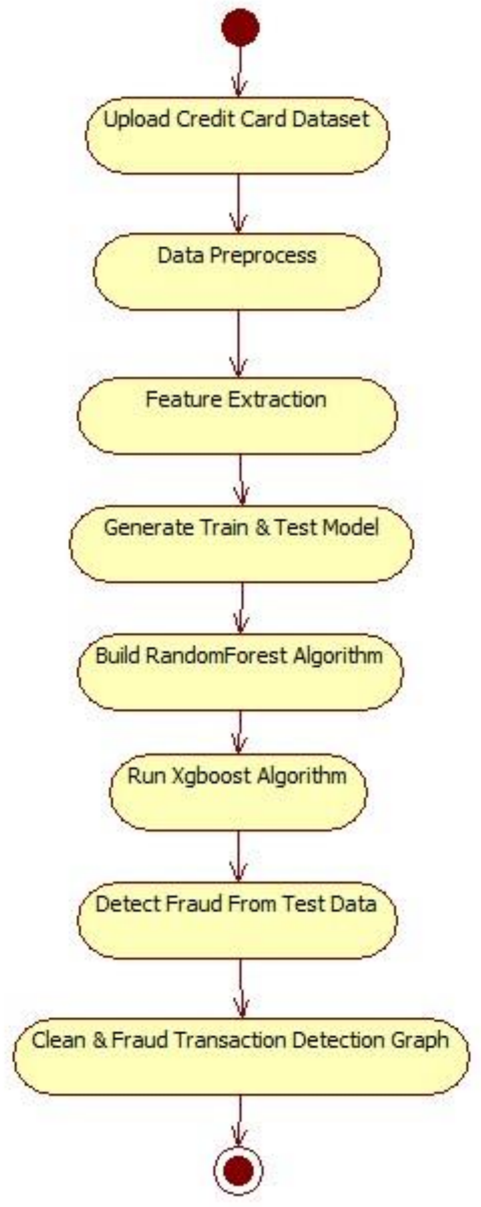

Upload Credit Card Dataset
Data Preprocess
Feature Extraction
Generate Train & Test Model
Build RandomForest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph

Figure 8.5: Activity Diagram

8.5 Collaboration Diagram:

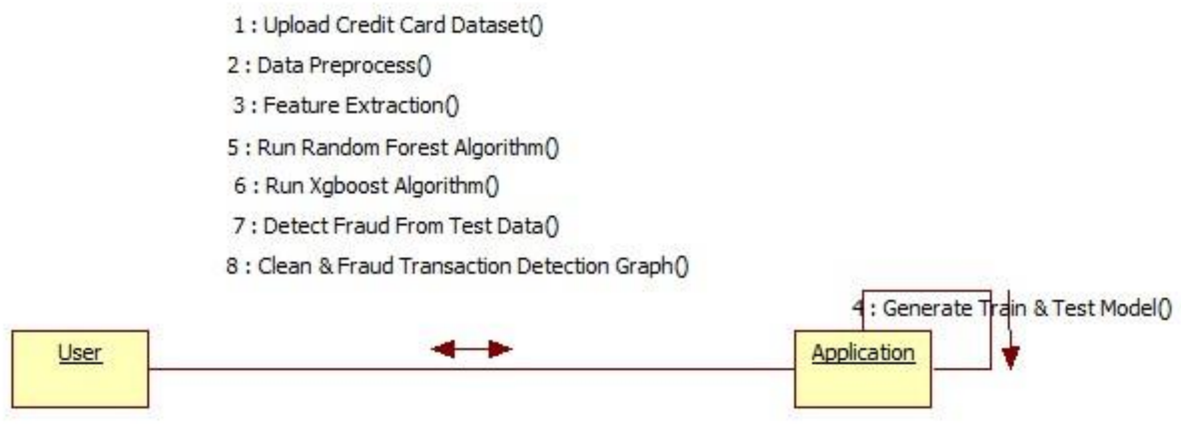

1 : Upload Credit Card Dataset()
2 : Data Preprocess()
3 : Feature Extraction()
5 : Run Random Forest Algorithm()
6 : Run Xgboost Algorithm()
7 : Detect Fraud From Test Data()
8 : Clean & Fraud Transaction Detection Graph()
4 : Generate Train & Test Model()
User
Application

Figure 8.6: Collaboration Diagram

CHAPTER 9
SYSTEM TESTING

9.1 TESTING METHODOLOGIES

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- Validation Testing.

9.1.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

9.1.2 Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

- **1.Top-Down Integration**

  - This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

➢ 2. Bottom-up Integration

- This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom-up integration strategy may be implemented with the following steps:
- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

9.1.3 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

9.1.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format.

Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

9.1.5 Validation Checking

Validation checks are performed on the following fields.

Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true system test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

User Training

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

Maintenance

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing are simple and easy to understand which will make maintenance easier.

9.1.7 System Testing

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

CHAPTER 10

ALGORITHMS

10.1 XGBoost:

XGBoost, short for Extreme Gradient Boosting, is a powerful machine learning algorithm known for its efficiency and effectiveness in handling a variety of structured data problems. It belongs to the ensemble learning family and is based on the gradient boosting framework. XGBoost builds a series of decision trees sequentially, each correcting the errors of its predecessors, with a focus on optimizing a specific objective function. It utilizes a combination of regularization techniques to prevent overfitting, including shrinkage (or learning rate) and tree pruning. Additionally, XGBoost employs parallel computing techniques to scale efficiently to large datasets, making it a popular choice for both academic research and industrial applications.

One key feature of XGBoost is its flexibility in handling diverse types of data and tasks, such as regression, classification, ranking, and user-defined objectives. It supports a wide range of customization options, including different tree learning algorithms, loss functions, and split finding methods, allowing practitioners to fine-tune the model according to the specific requirements of their problem domain. With its superior performance, scalability, and versatility, XGBoost has become a staple tool in machine learning pipelines across various domains, from finance and healthcare to marketing and cybersecurity, where accurate predictions and interpretability are paramount.

10.2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient-boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg. An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).

The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman to construct a collection of decision trees with controlled variance. Random forests are frequently used as "black box" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

CHAPTER 11

PYTHON

11. 1 Introduction

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted**: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

11.1.1 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

11.1.2 Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

• **A broad standard library:** Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh. FRAUD AUDITOR: A VISUAL ANALYSIS APPROACH FOR COLLUSIVE FRAUD IN HEALTH INSURANCE VISM-Department of CSE (2020-24) 24

• **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

• **Databases:** Python provides interfaces to all major commercial databases.

• **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

• It supports functional and structured programming methods as well as OOP.

• It can be used as a scripting language or can be compiled to byte-code for building large applications.

• It provides very high-level dynamic data types and supports dynamic type checking. • IT supports automatic garbage collection.

• It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

11.2 LIST

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type. Creating a list is as simple as putting different comma-separated values between square brackets. For example –

list1 =['physics','chemistry',1997,2000];

list2 =[1,2,3,4,5];

list3 =["a","b","c","d"]

Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

Python Expression	**Result**	**Description**
Len ([1, 2, 3])	**3**	Length
[1, 2, 3] + [4, 5, 6]	] [1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x	1 2 3	Iteration

Built-in List Functions & Methods:

Python includes the following list functions –

S.NO	**FUNCTION DESCRIPTION**
1	Cmp (list1, list2) Compares elements of both lists.
2	Len(list) Gives the total length of the list.
3	Max(list) Returns item from the list with max values.
4	Min(list) Returns item from the list with min value.
5	List(seq) Converts a tuple into list.

Python has the following list methods

S.NO	**Method With Description**
1	List.apend(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list

3	list. extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list. Insert(index, obj) Inserts object obj into list at offset index
6	list.pop(obj=list[-1]) Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list

11.3 TUPLES:

In Python, a tuple is an immutable sequence of values. Once created, you cannot modify its contents. Tuples are created by placing comma-separated values inside parentheses `()`. Here's an example:

python

my_tuple = (1, 2, 3, 'a', 'b', 'c')

Tuples can contain elements of different data types, including numbers, strings, lists, or even other tuples.

Operations on Tuples:

1. Accessing Elements: You can access elements of a tuple using indexing. Indexing starts from 0.

```
python
print(my_tuple[0])  # Output: 1
```

2. Slicing: You can also slice tuples to extract a portion of the tuple.

~~~
```'python
print(my_tuple[2:5])  # Output: (3, 'a', 'b')
```
~~~

3. Concatenation: You can concatenate tuples using the `+` operator.

~~~
```python
~~~

```python
new_tuple = my_tuple + (4, 5, 6)
```

4. Repetition: You can repeat a tuple using the `*` operator.

```python
repeated_tuple = my_tuple * 2
```

5. Length: You can find the length of a tuple using the `len()` function.

```python
print(len(my_tuple))  # Output: 6
```

6. Membership Test: You can check if an element is present in a tuple using the `in` keyword.

```python
print('a' in my_tuple)  # Output: True
```

7. Iteration: You can iterate over a tuple using a loop.

```python
for item in my_tuple:
    print(item)
```

8. Unpacking: You can unpack a tuple into individual variables.

```python
a, b, c = my_tuple[:3]
```

Tuples are commonly used when you have a fixed collection of items that you don't want to change. They're particularly useful for returning multiple values from a function, as they're immutable and therefore safer for certain operations.

11.4 DICTIONARY:

In Python, a dictionary is a powerful data structure used to store collections of items as key-value pairs. Each key in a dictionary must be unique, and it is associated with a value. Dictionaries are mutable, meaning their contents can be modified after creation. They are defined using curly braces `{}` and consist of comma-separated key-value pairs.

Here are some common operations and descriptions of dictionaries in Python:

1. Creating a Dictionary:

```python
my_dict = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

2. Accessing Values:

You can access the value associated with a key using square brackets `[]` or the `get()` method.

```python
print(my_dict['key1'])  # Output: value1
print(my_dict.get('key2'))  # Output: value2
```

3. Adding or Modifying Entries:

You can add new key-value pairs or modify existing ones.

```python
my_dict['new_key'] = 'new_value'  # Adding a new entry
my_dict['key1'] = 'modified_value'  # Modifying an existing entry
```

4. Removing Entries:

Use the `del` keyword or the `pop()` method to remove an entry by key.

```python
del my_dict['key3']  # Removing an entry using del
value = my_dict.pop('key2')  # Removing an entry using pop and storing the removed value
```

5. Checking Membership:

You can check if a key exists in a dictionary using the `in` keyword.

```python
if 'key1' in my_dict:
    print("Key 'key1' exists")
```

6. Iterating Over a Dictionary:

You can iterate over keys, values, or key-value pairs using loops.

```python
for key in my_dict:
    print(key, my_dict[key])
for value in my_dict.values():
    print(value)
for key, value in my_dict.items():
    print(key, value)
```

7. Dictionary Length:

- Get the number of key-value pairs in a dictionary using the `len()` function.

```python
print(len(my_dict))
```

Dictionaries are commonly used in Python due to their flexibility and efficiency in storing and accessing data. They are suitable for tasks like representing mappings, storing settings, and handling various data structures.

11.5 Defining a Function

In Python, defining a function allows you to encapsulate a block of code that performs a specific task, making your code more modular and reusable. Here's how you define a function in Python:

```python
def function_name(parameters):
    """
    Docstring: Description of what the function does.
    """
    # Body of the function
    # Perform operations using parameters
    # Return a value if necessary
    pass  # This line is optional if the function has no code yet
```

Let me break down the components:

-`def`: This keyword is used to define a function.

- `function_name`: This is the name of the function. Choose a descriptive name that indicates what the function does.

- `(parameters)`: These are the inputs to the function. Parameters are optional; if the function doesn't require any input, you can leave the parentheses empty.

- `:`: This colon marks the beginning of the function body.

- `"""Docstring: Description of what the function does."""`: This is a docstring, which is a string literal that appears as the first statement in a function and provides documentation about the function's purpose, parameters, and return values.

- `# Body of the function`: This is where you write the actual code that the function will execute.

- `pass`: This is a placeholder that allows the function to have an empty body. It's optional and can be removed once you start implementing the function.

Here's an example of a simple function that adds two numbers and returns the result:

```python
def add_numbers(a, b):
    '''
     Adds two numbers and returns the result.
 Parameters:
        a (int): The first number.
        b (int): The second number.
 Returns:
        int: The sum of a and b.
    """
    return a + b
```

You can call this function elsewhere in your code like this:

```python
result = add_numbers(5, 3)
print(result)  # Output: 8
```

This is the basic syntax for defining functions in Python.

11.6 Scope of Variables

In Python, the scope of a variable refers to the region of the code where that variable is accessible. Python has four main types of variable scopes:

1. Local scope: Variables defined inside a function have local scope. They are accessible only within that function.

```python
def my_function():
    x = 10  # Local variable
    print(x)
my_function()  # Output: 10
print(x)  # Error: NameError: name 'x' is not defined
```

2. Enclosing (nonlocal) scope: Variables defined in an enclosing function (i.e., a function that contains other functions) are accessible within nested functions. This applies when you have nested functions and you want to access a variable from an outer function within an inner function.

```python
def outer_function():
    x = 10  # Enclosing scope
    def inner_function():
        print(x)  # Accessing x from the enclosing scope
    inner_function()
```

```
  outer_function()  # Output: 10
```

3. Global scope: Variables defined outside of all functions or classes have global scope. They are accessible throughout the entire module.

```python
x = 10  # Global variable
def my_function():
    print(x)  # Accessing x from the global scope
my_function()  # Output: 10
print(x)  # Output: 10
```

4. Built-in scope: Python provides a set of built-in functions and exceptions that are always available. These are in the built-in scope.

```python
print(max([1, 2, 3]))  # Output: 3
```

Variables can be accessed within their scope and any inner scopes, but if you try to access a variable outside of its scope, you'll encounter a `NameError`.
It's important to note that while you can access global variables from within functions, modifying them requires you to explicitly declare them as global within the function. Otherwise, Python will create a new local variable instead.

```python
x = 10  # Global variable
def my_function():
    global x  # Declare x as global
    x = 20  # Modify the global variable
    print(x)
my_function()  # Output: 20
print(x)  # Output: 20
```

11.4 How to Install Python on Windows and Mac:

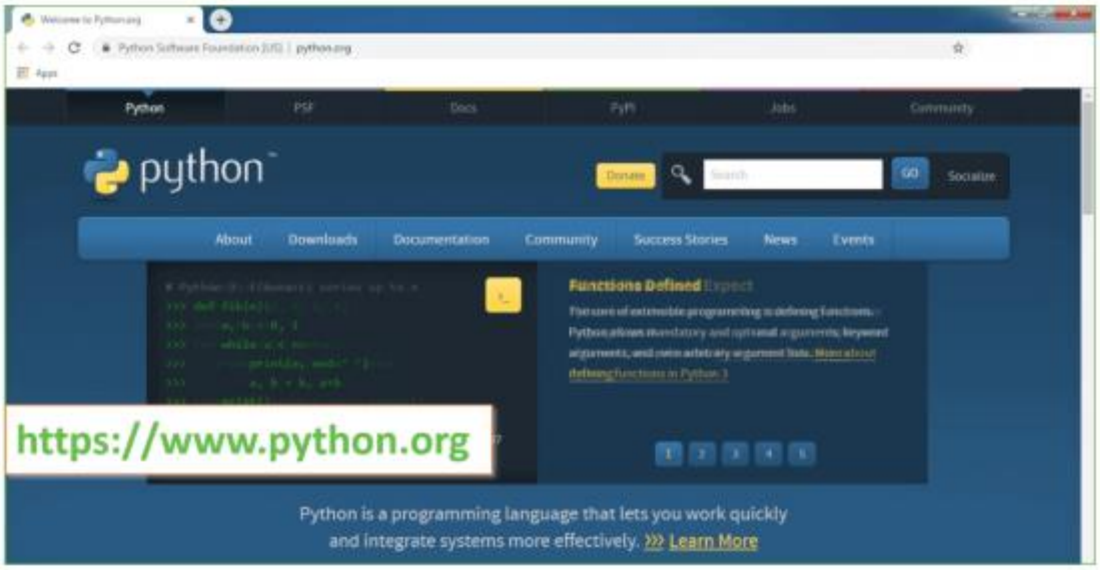
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

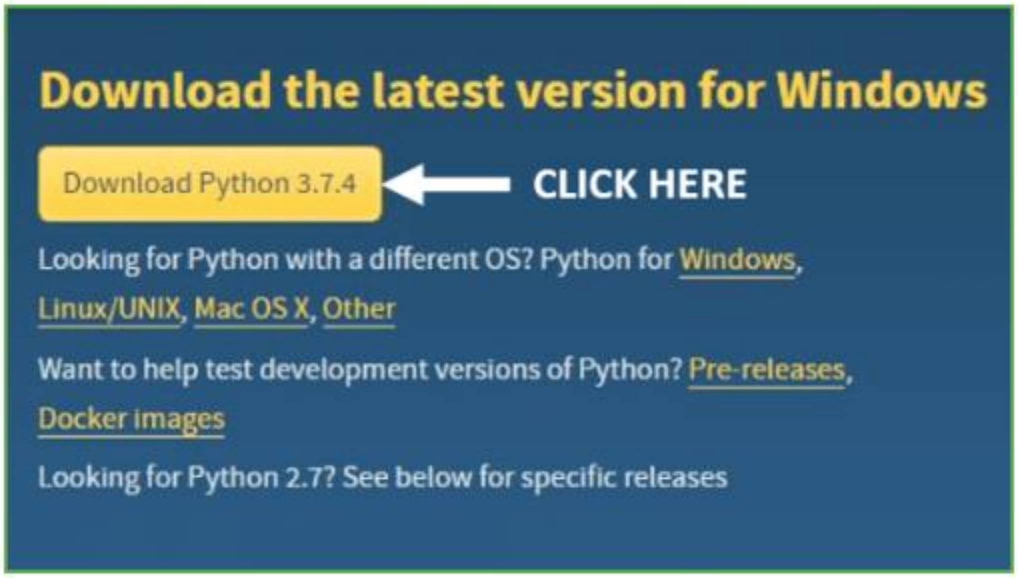
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://www.python.org**



Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Step 2: Click on the Download Tab.



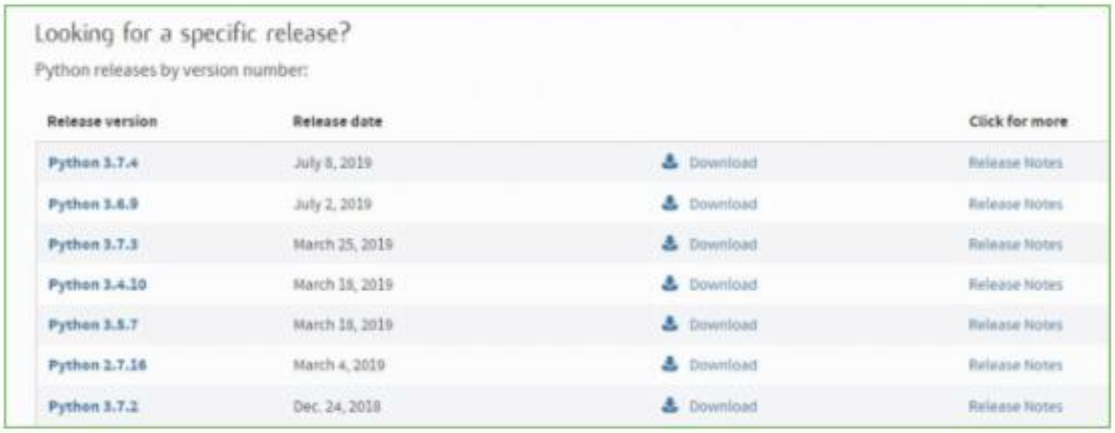
Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



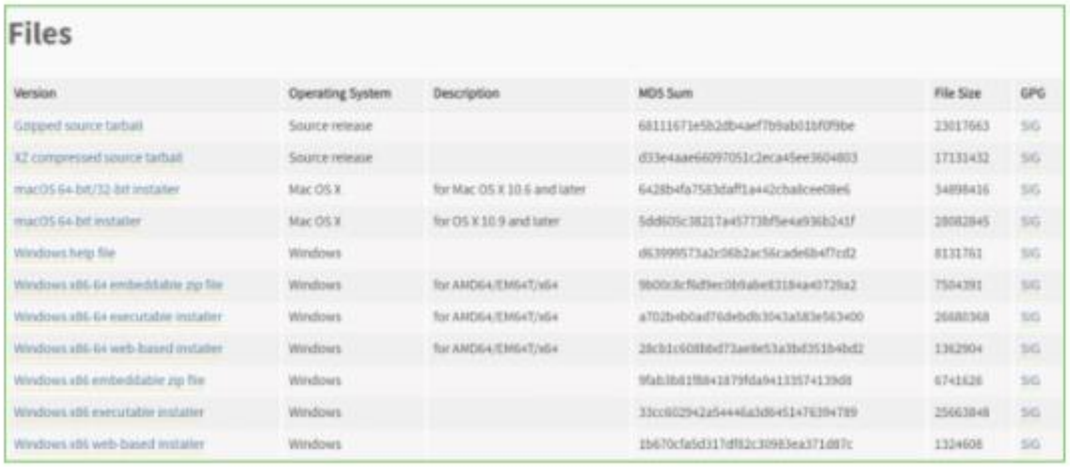
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

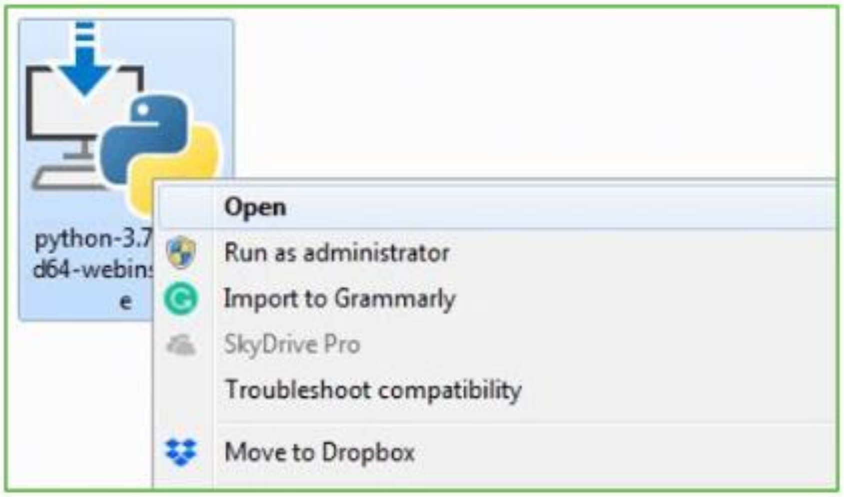
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

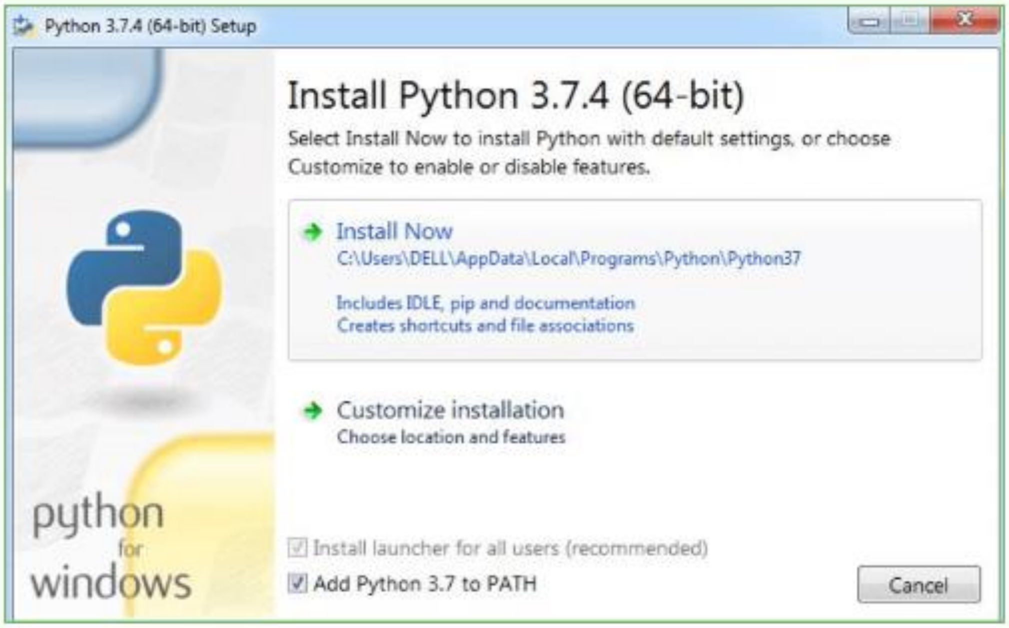
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

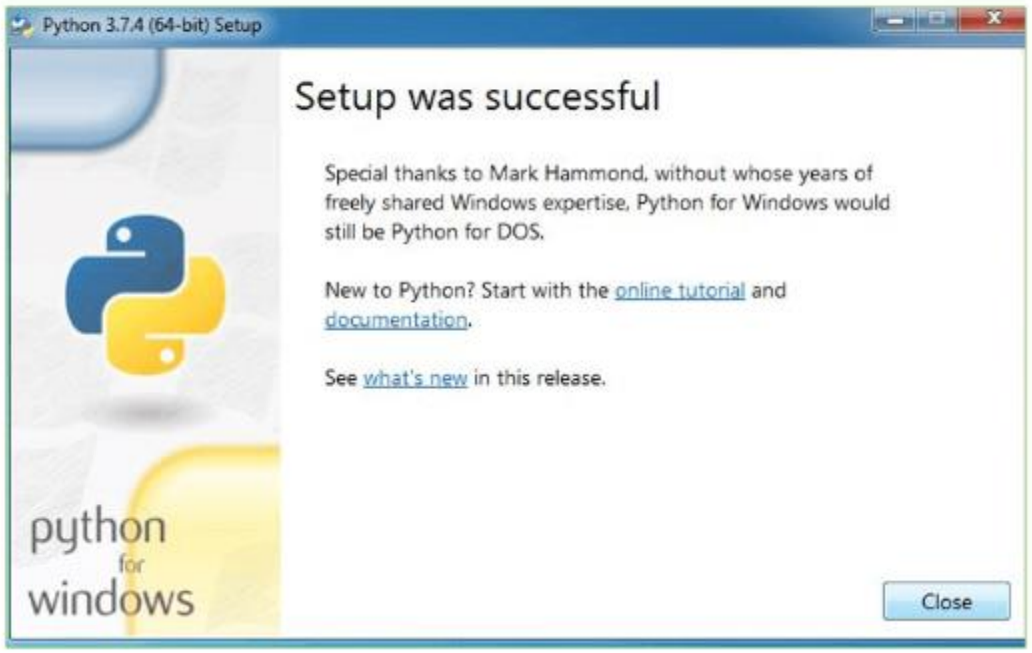
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

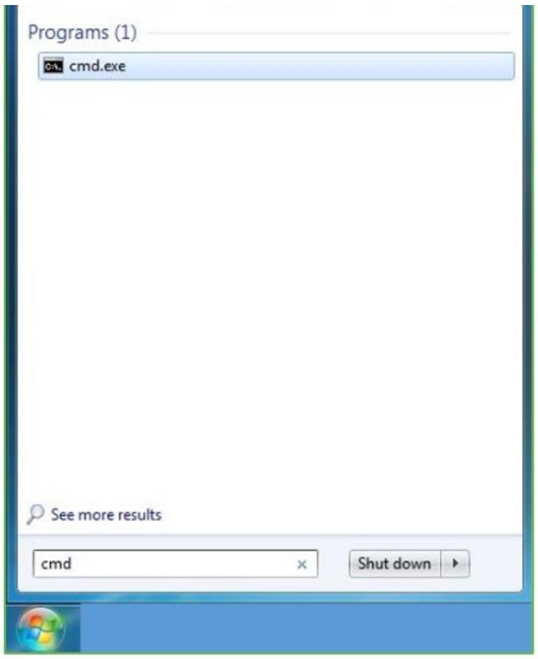
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

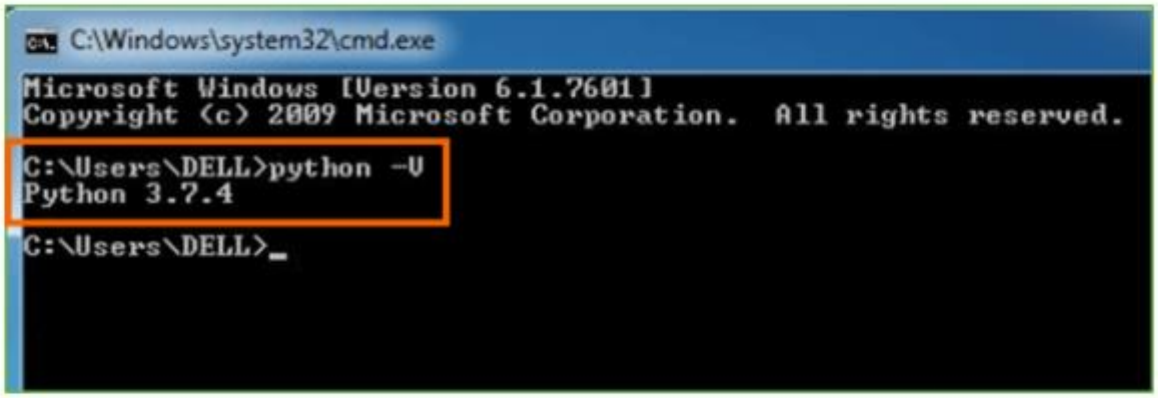
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

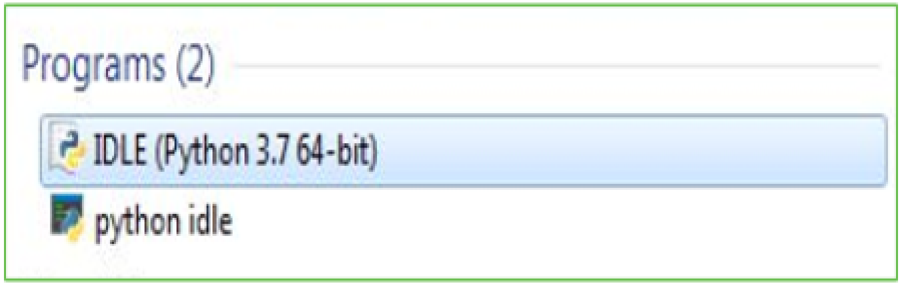
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

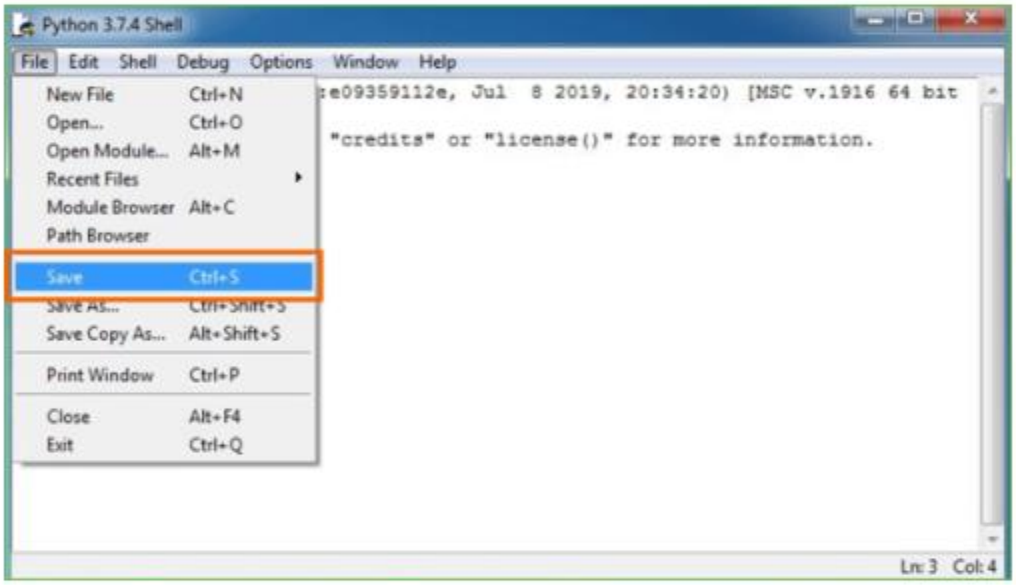
Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit

 "credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

CHAPTER 12
SOURCE CODE

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
import numpy as np
import pandas as pd
from sklearn import *
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

main = tkinter.Tk()
main.title("Credit Card Fraud Detection Using Machine Learning") #designing main screen
main.geometry("1300x1200")

global filename
global cls
global X, Y, X_train, X_test, y_train, y_test
global random_acc # all global variables names define in above lines
global clean
global attack
global total
```

```
def traintest(train):     #method to generate test and train data from dataset
    X = train.values[:, 0:29]
    Y = train.values[:, 30]
    print(X)
    print(Y)
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 0)
    return X, Y, X_train, X_test, y_train, y_test

def generateModel(): #method to read dataset values which contains all five features data
    global X, Y, X_train, X_test, y_train, y_test
    train = pd.read_csv(filename)
    X, Y, X_train, X_test, y_train, y_test = traintest(train)
    text.insert(END,"Train & Test Model Generated\n\n")
    text.insert(END,"Total Dataset Size : "+str(len(train))+"\n")
    text.insert(END,"Split Training Size : "+str(len(X_train))+"\n")
    text.insert(END,"Split Test Size : "+str(len(X_test))+"\n")

def upload(): #function to upload
    global filename
    filename = filedialog.askopenfilename(initialdir="dataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");

def prediction(X_test, cls):  #prediction done here
    y_pred = cls.predict(X_test)
    for i in range(50):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred
```

```
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details):
    accuracy = accuracy_score(y_test,y_pred)*100
    text.insert(END,details+"\n\n")
    text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
    return accuracy

def runRandomForest():
    headers =
["Time","V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","
V15","V16","V17","V18","V19","V20","V21","V22","V23","V24","V25","V26","V27","V28",
"Amount","Class"]
    global random_acc
    global cls
    global X, Y, X_train, X_test, y_train, y_test
    cls =
RandomForestClassifier(n_estimators=50,max_depth=2,random_state=0,class_weight='balanced
')
    cls.fit(X_train, y_train)
    text.insert(END,"Prediction Results\n\n")
    prediction_data = prediction(X_test, cls)
    random_acc = cal_accuracy(y_test, prediction_data,'Random Forest Accuracy')
    #str_tree = export_graphviz(cls, out_file=None, feature_names=headers,filled=True,
special_characters=True, rotate=True, precision=0.6)
    #display.display(str_tree)

def runXgboost():
    headers =
["Time","V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","
V15","V16","V17","V18","V19","V20","V21","V22","V23","V24","V25","V26","V27","V28",
"Amount","Class"]
```

```
    global xgboost_acc
    global cls
    global X, Y, X_train, X_test, y_train, y_test
    cls = XGBClassifier(learning_rate=0.1,n_estimators=100, n_jobs=-1)
    cls.fit(X_train, y_train)
    text.insert(END,"Prediction Results\n\n")
    prediction_data = prediction(X_test, cls)
    xgboost_acc = cal_accuracy(y_test, prediction_data,'Xgboost Accuracy')

def predicts():
    global clean
    global attack
    global total
    clean = 0;
    attack = 0;
    text.delete('1.0', END)
    filename = filedialog.askopenfilename(initialdir="dataset")
    test = pd.read_csv(filename)
    test = test.values[:, 0:29]
    total = len(test)
    text.insert(END,filename+" test file loaded\n");
    y_pred = cls.predict(test)
    for i in range(len(test)):
        if str(y_pred[i]) == '1.0':
            attack = attack + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Contains Fraud Transaction
Signature')+"\n\n")
        else:
            clean = clean + 1
            text.insert(END,"X=%s, Predicted = %s" % (test[i], 'Transaction Contains Cleaned
Signatures')+"\n\n")
```

```
def graph():
    height = [total,clean,attack]
    bars = ('Total Transactions','Normal Transaction','Fraud Transaction')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()
font = ('times', 16, 'bold')
title = Label(main, text='Credit Card Fraud Detection Using Machine Learning')
title.conFigure(bg='greenyellow', fg='dodger blue')
title.conFigure(font=font)
title.conFigure(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.conFigureure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.conFigure(font=font1)
font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Upload Credit Card Dataset", command=upload)
uploadButton.place(x=50,y=550)
uploadButton.conFigure(font=font1)
modelButton = Button(main, text="Generate Train & Test Model", command=generateModel)
modelButton.place(x=350,y=550)
modelButton.conFigure(font=font1)
runrandomButton = Button(main, text="Run Random Forest Algorithm",
command=runRandomForest)
runrandomButton.place(x=650,y=550)
runrandomButton.conFigure(font=font1)

runXgboostButton = Button(main, text="Run Xgboost Algorithm", command=runXgboost)
runXgboostButton.place(x=950,y=550)
runXgboostButton.conFigure(font=font1)
predictButton = Button(main, text="Detect Fraud From Test Data", command=predicts)
predictButton.place(x=50,y=600)
predictButton.conFigure(font=font1)
graphButton = Button(main, text="Clean & Fraud Transaction Detection Graph",
command=graph)
graphButton.place(x=350,y=600)
graphButton.conFigure(font=font1)
exitButton = Button(main, text="Exit", command=exit)
exitButton.place(x=770,y=600)
exitButton.conFigure(font=font1)
main.conFigure(bg='orange')
main.mainloop()
```

CHAPTER 13

OUTPUT SCREENS

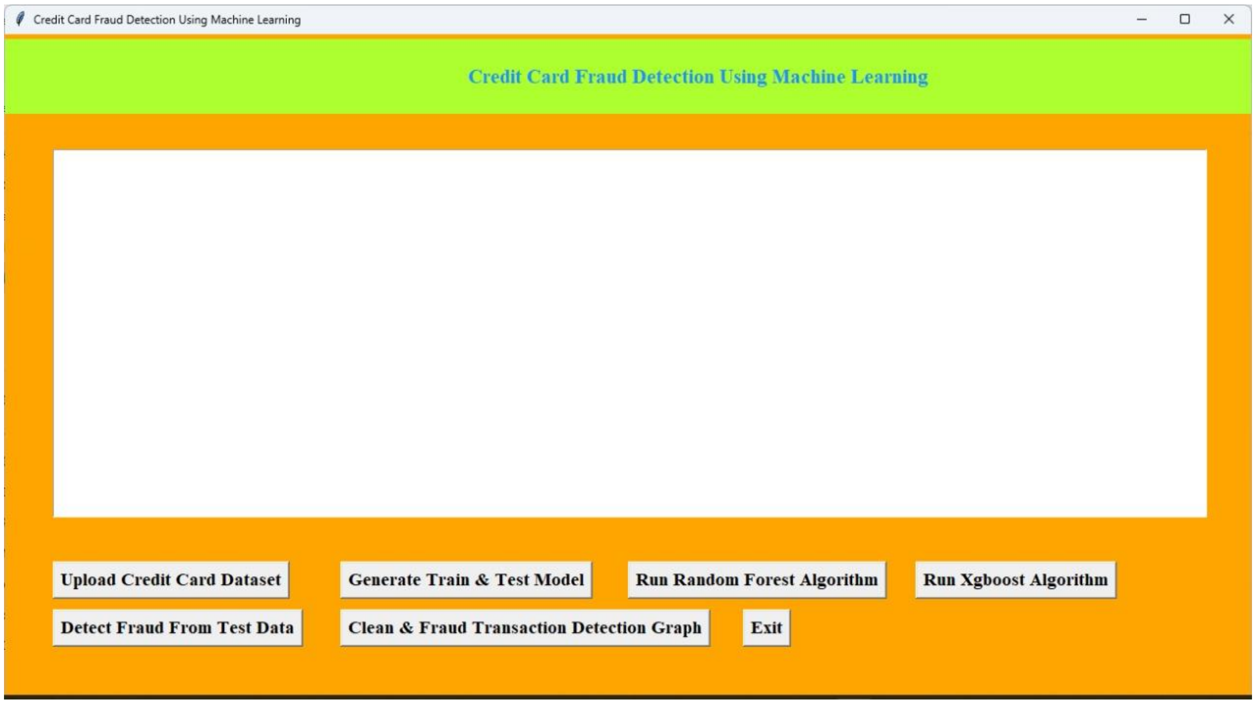
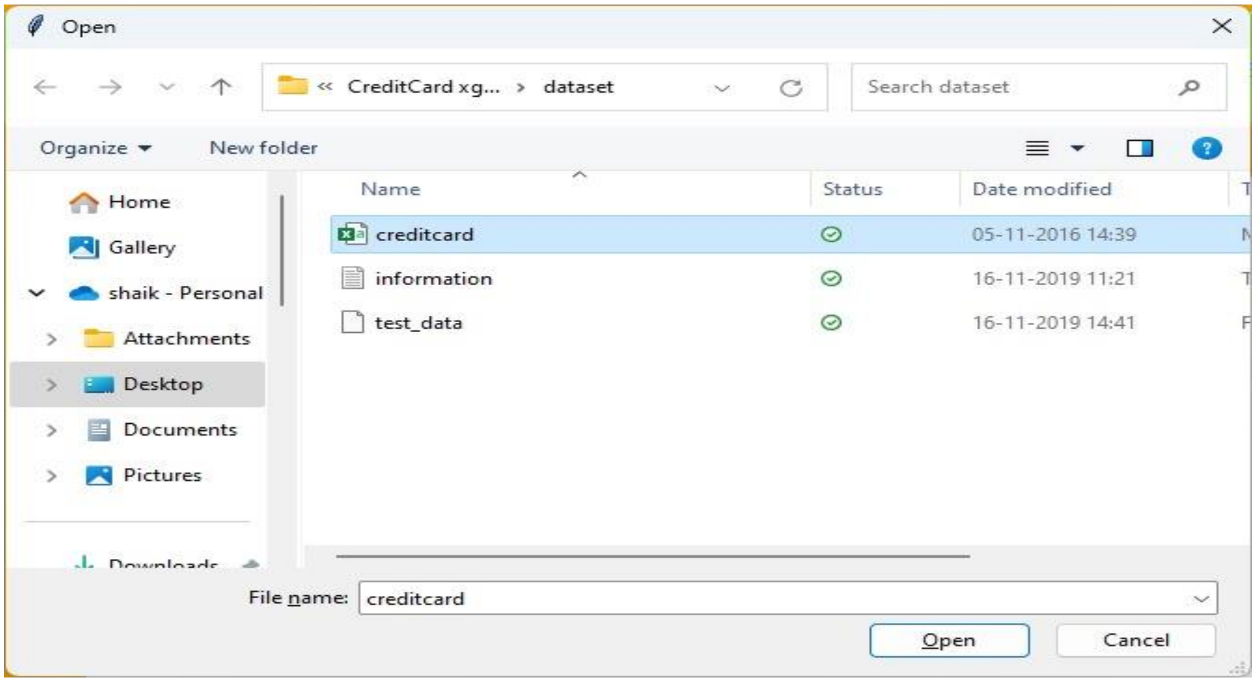
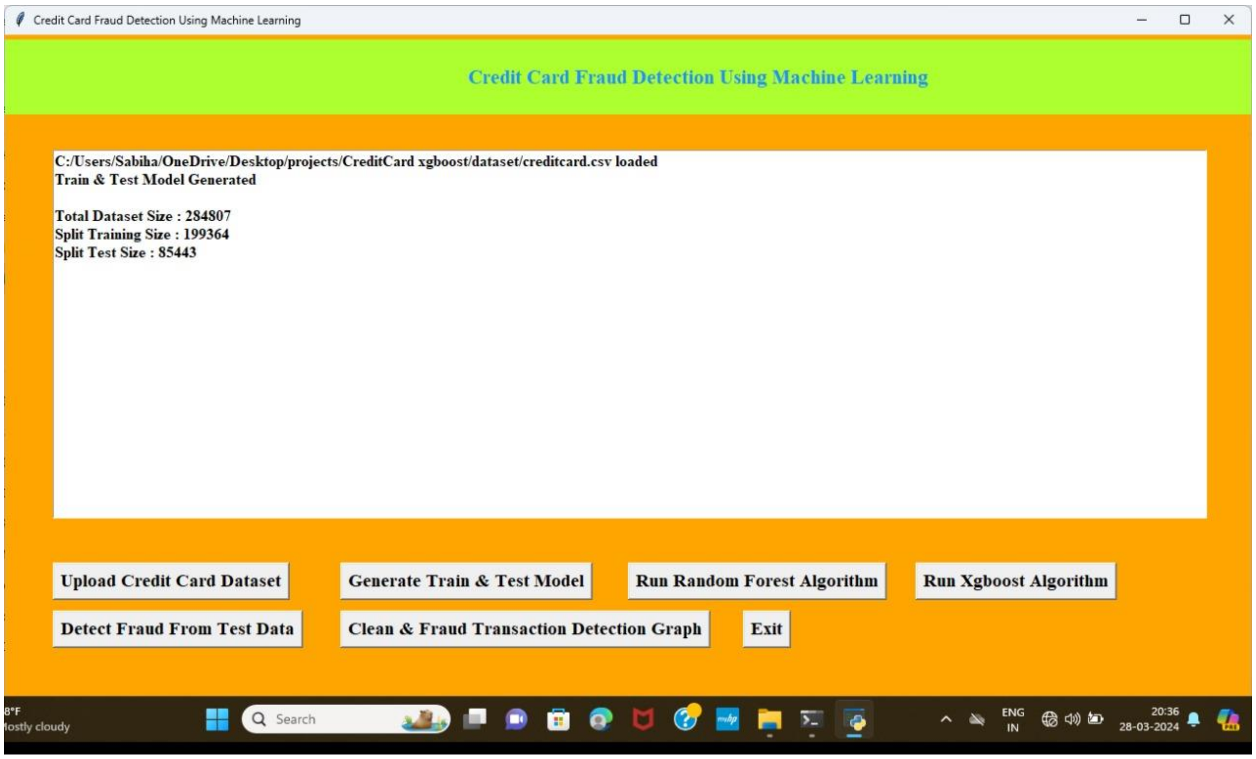

Credit Card Fraud Detection Using Machine Learning
Credit Card Fraud Detection Using Machine Learning
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

Figure 13.1: HOME PAGE



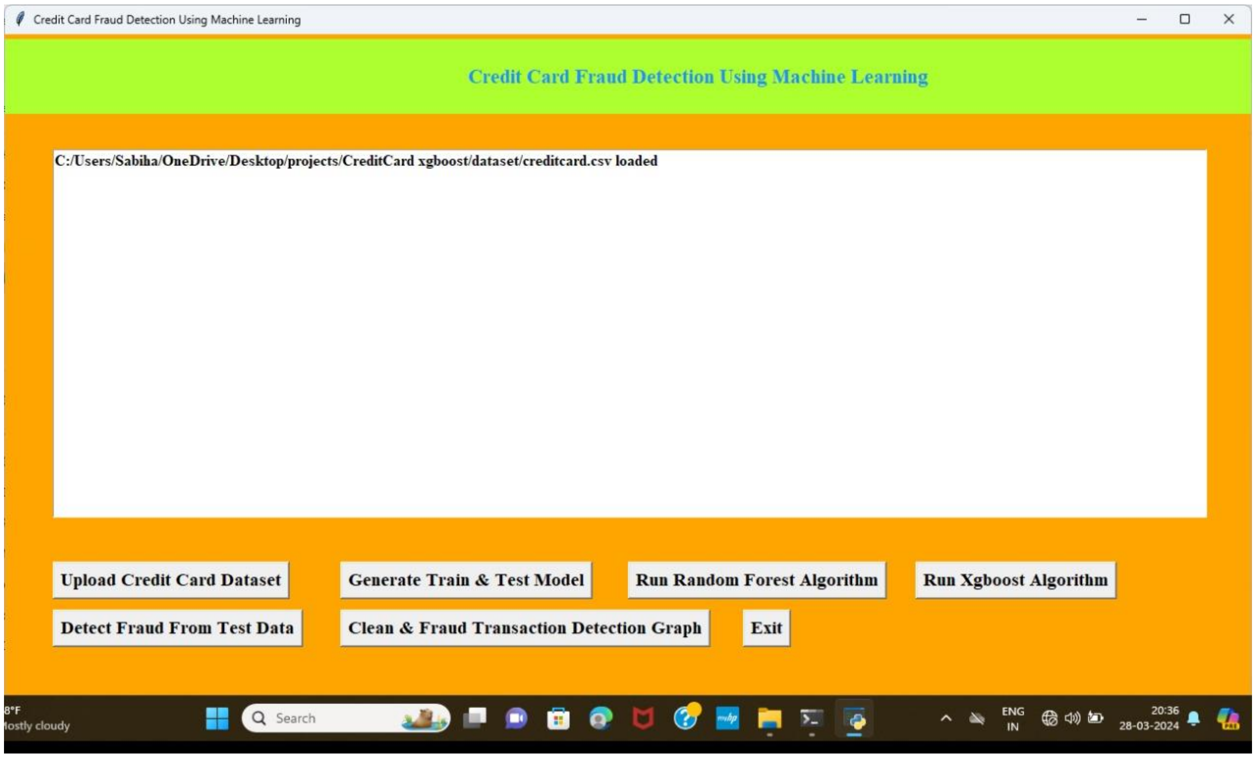
Name	Status	Date modified
creditcard	✓	05-11-2016 14:39
information	✓	16-11-2019 11:21
test_data	✓	16-11-2019 14:41

Figure 13.2: DATASET UPLOAD


Credit Card Fraud Detection Using Machine Learning
Credit Card Fraud Detection Using Machine Learning
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

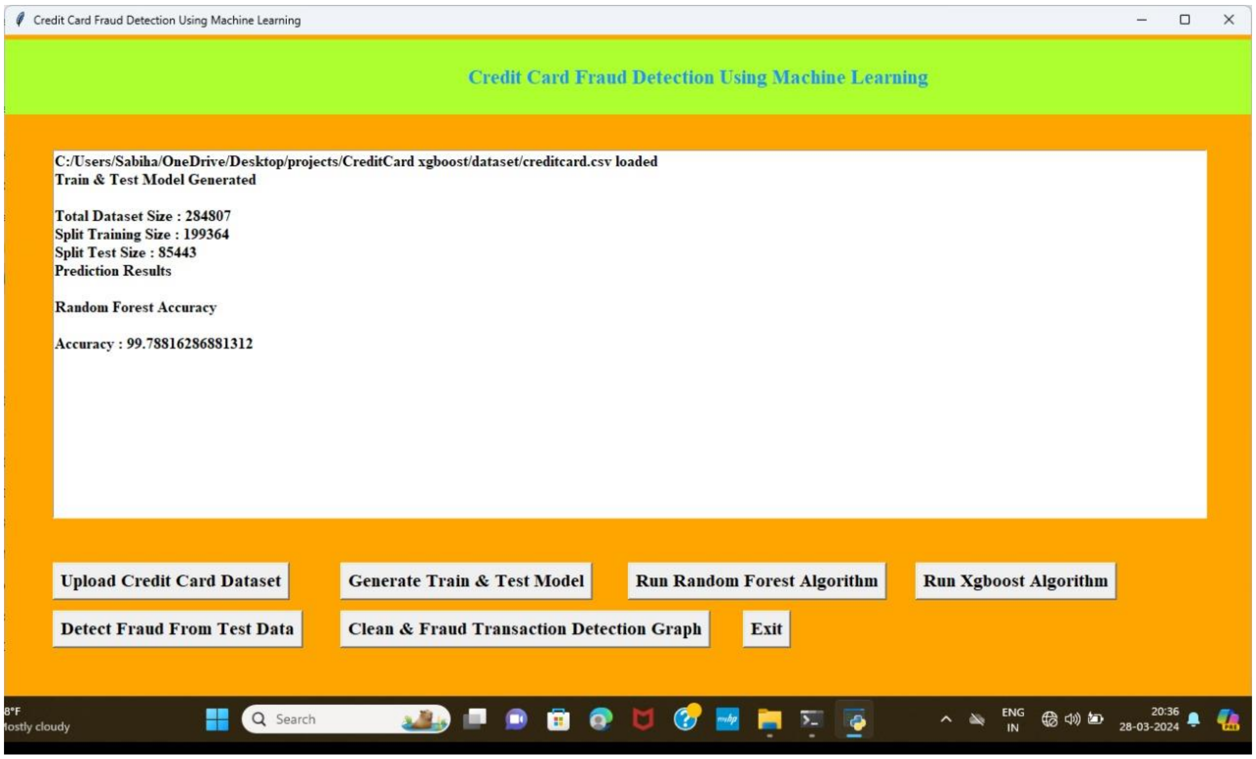
```
C:/Users/Sabiha/OneDrive/Desktop/projects/CreditCard xgboost/dataset/creditcard.csv loaded
Train & Test Model Generated

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

Figure 13.3: DATASET DETAILS


Credit Card Fraud Detection Using Machine Learning
Credit Card Fraud Detection Using Machine Learning
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

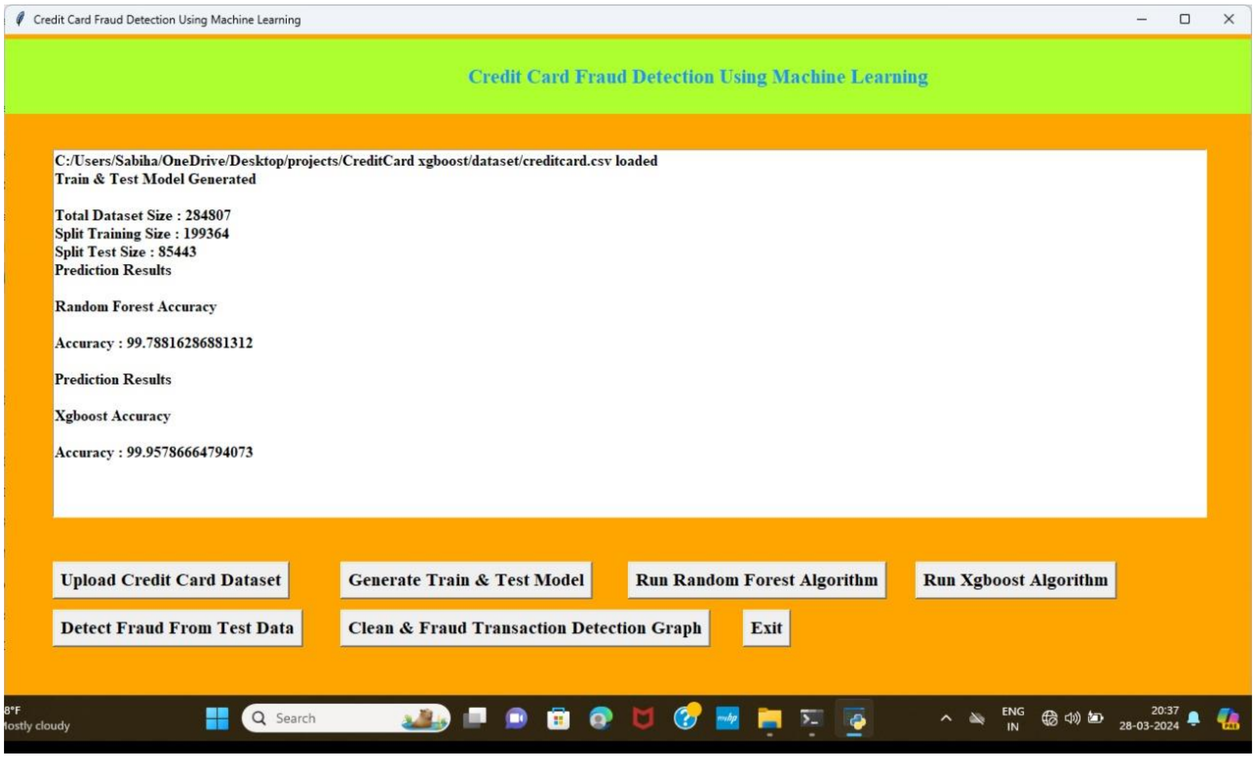
```
C:/Users/Sabiha/OneDrive/Desktop/projects/CreditCard xgboost/dataset/creditcard.csv loaded
```

Figure 13.4: RUN RANDOMFOREST ALGORITHM


Credit Card Fraud Detection Using Machine Learning
Credit Card Fraud Detection Using Machine Learning
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

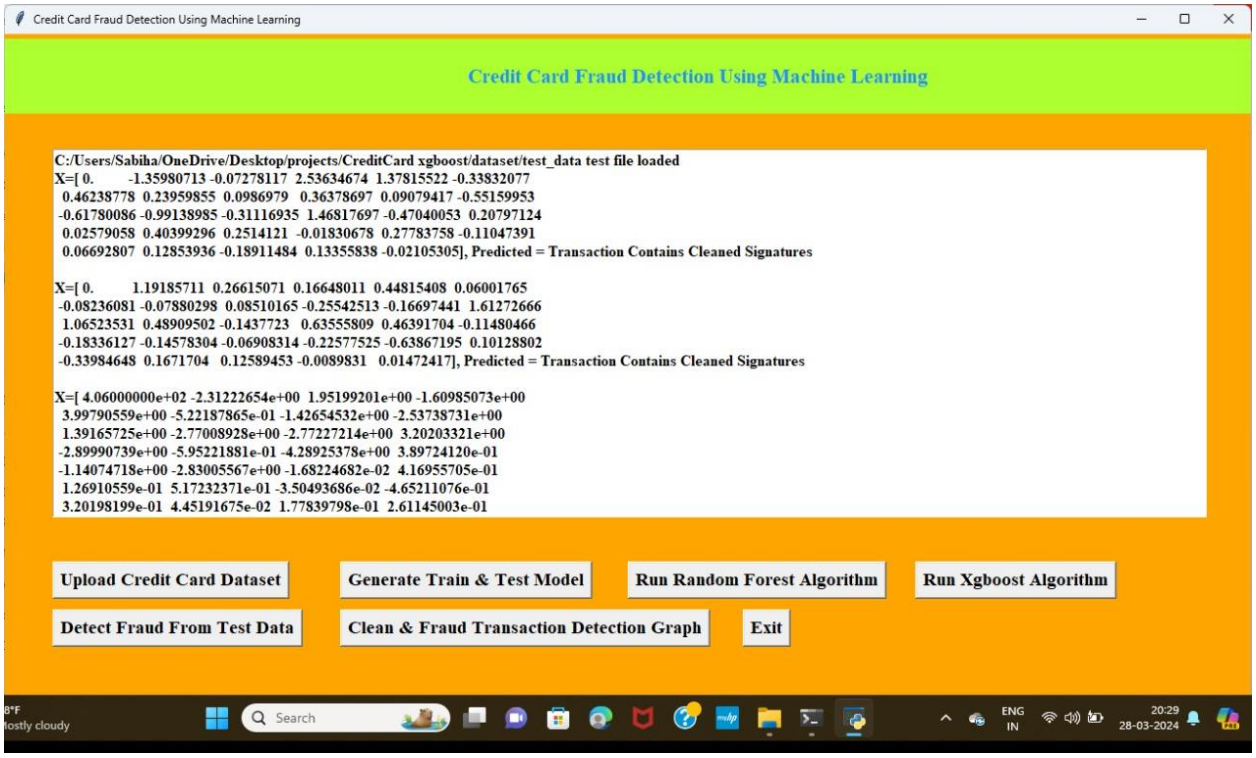
```
C:/Users/Sabiha/OneDrive/Desktop/projects/CreditCard xgboost/dataset/creditcard.csv loaded
Train & Test Model Generated

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

Random Forest Accuracy

Accuracy : 99.78816286881312
```

Figure 13.5: Random Forest Accuracy


Credit Card Fraud Detection Using Machine Learning
Credit Card Fraud Detection Using Machine Learning
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

```
C:/Users/Sabiha/OneDrive/Desktop/projects/CreditCard xgboost/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

Prediction Results

Xgboost Accuracy

Accuracy : 99.95786664794073
```

Figure 13.6: Xgboost Accuracy


Credit Card Fraud Detection Using Machine Learning
Credit Card Fraud Detection Using Machine Learning
Upload Credit Card Dataset
Generate Train & Test Model
Run Random Forest Algorithm
Run Xgboost Algorithm
Detect Fraud From Test Data
Clean & Fraud Transaction Detection Graph
Exit

```
C:/Users/Sabiha/OneDrive/Desktop/projects/CreditCard xgboost/dataset/test_data test file loaded
X=[ 0.         -1.35980713 -0.07278117  2.53634674  1.37815522 -0.33832077
  0.46238778  0.23959855  0.0986979   0.36378697  0.09079417 -0.55159953
 -0.61780086 -0.99138985 -0.31116935  1.46817697 -0.47040053  0.20797124
  0.02579058  0.40399296  0.2514121  -0.01830678  0.27783758 -0.11047391
  0.06692807  0.12853936 -0.18911484  0.13355838 -0.02105305], Predicted = Transaction Contains Cleaned Signatures

X=[ 0.          1.19185711  0.26615071  0.16648011  0.44815408  0.06001765
 -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
```

Figure 13.7: Detect Fraud from Test Data

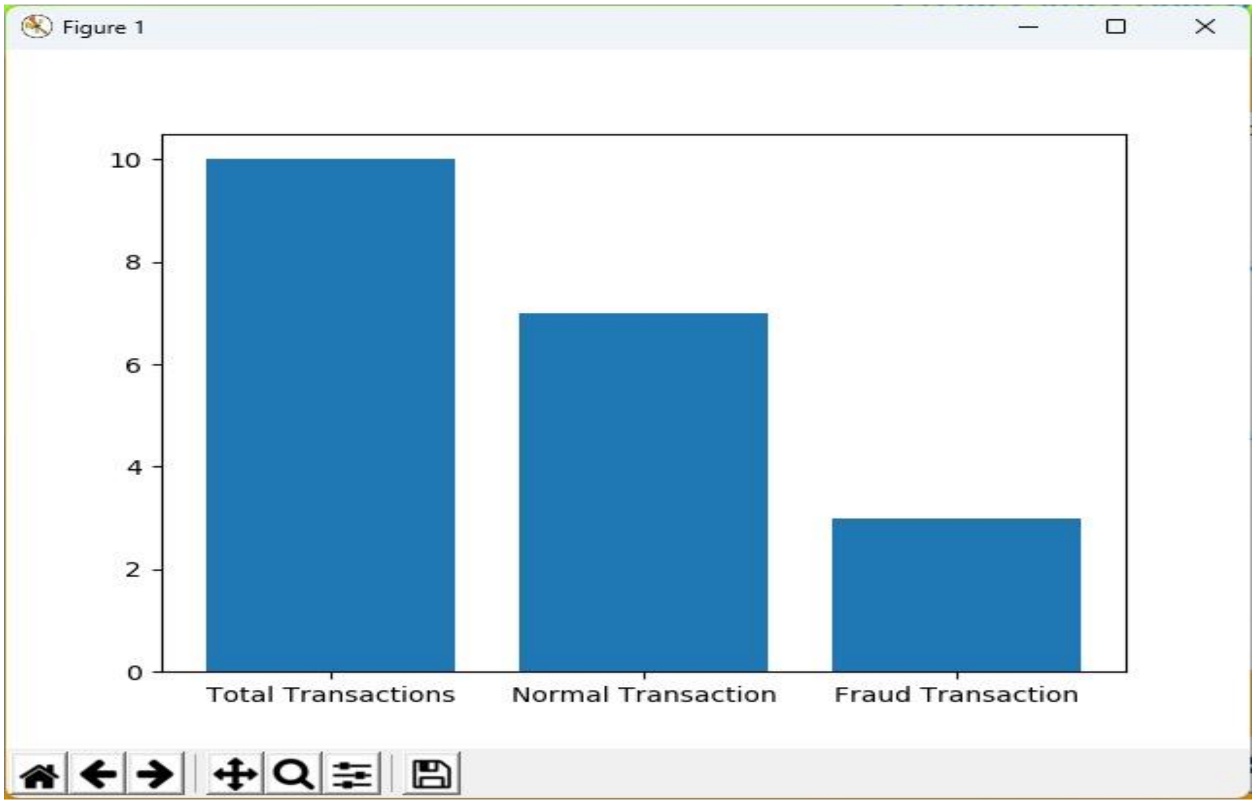


Figure 13.8: Clean and Fraud Transaction Detection Graph

CHAPTER 14

CONCLUSION

The xgboost 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.

CHAPTER 15

FUTURE ENHANCEMENT

For future work, the methods studied in this paper will be extended to online learning models. In addition, other online learning models will be investigated. The use of online learning will enable rapid detection of fraud cases, potentially in real-time. This in turn will help detect and prevent fraudulent transactions before they take place, which will reduce the number of losses incurred every day in the financial sector.

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 URL 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.

[6] Sitaram patel, Sunita Gond , "Supervised Machine (SVM) Learning for Credit Card Fraud Detection, International of engineering trends and technology, vol. 8, no. -3, pp. 137- 140, 2014.

[7] Snehal Patil, Harshada Somavanshi, Jyoti Gaikwad, Amruta Deshmane, Rinku Badgujar," Credit Card Fraud Detection Using Decision Tree Induction Algorithm, International Journal of Computer Science and Mobile Computing, Vol.4 Issue.4, April- 2015, pg. 92-95

[8] Dahee Choi and Kyungho Lee, "Machine Learning based Approach to Financial Fraud Detection Process in Mobile Payment System", vol. 5, no. - 4, December 2017, pp. 12-24.

9. C. Alippi, G. Boracchi, and M. Roveri, "Just-in-time classifiers for recurrent concepts," IEEE Trans. Neural Netw. Learn. Syst., vol. 24, no. 4, pp. 620–634, Apr. 2013.

10. B. Baesens, V. Van Vlasselaer, and W. Verbeke, Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection. Hoboken, NJ, USA: Wiley, 2015.