Guide Name:K Naresh M.tech(Ph.D) Assistant Professor IT & SVCE

Project Team:

Sannapaneni Saipurnesh IT &SVCE [Team Lead]

MahendraReddy Lingala IT &SVCE

Talari vasu IT &SVCE

K.Dinesh Kumar IT &SVCE

A MULTI-PERSPECTIVE FRAUD DETECTION METHOD FOR MULTIPARTICIPANT E-COMMERCE TRANSACTIONS

ABSTRACT

In the realm of e-commerce, where transactions involve multiple participants such as buyers, sellers, and intermediaries, the detection of fraudulent activities presents a significant challenge. To address this issue, our proposed method focuses on a Mult perspective approach aimed at enhancing fraud detection accuracy and efficiency. The first step involves the detection of user behaviors, wherein we leverage various techniques such as behavioral analysis and examination of transaction histories to gain insights into normal user behavior patterns. By understanding typical user interactions within the ecommerce ecosystem, we establish a baseline against which abnormal behaviors can be identified. Subsequently, we investigate into the analysis of abnormalities for feature extraction. Utilizing sophisticated anomaly detection algorithms, we scrutinize transaction data to uncover irregular patterns indicative of potentially fraudulent activities. This process allows us to extract important features that serve as key indicators for fraud detection. Finally, we employ an ensemble classification model to implement our fraud detection mechanism.

TABLE OF CONTENTS

INTRODUCTION

1. INTRODUCTION

WITH the increasing popularity of e-commerce platforms, more and more commercial transactions are now relying on web-based systems than the traditional cash-based approach [1]. Although the entity economy is greatly impacted by the COVID-19 epidemic in recent years, e-commerce remains largely unaffected by the pandemic, whereby aiding a steady market growth [2]. The sales volume of B2C (Business to Customer) e-commerce is expected to reach 6.5 trillion dollars by 2023 [3].

This paper combines the advantages of process mining and machine learning models by introducing a hybrid method to solve the anomaly detection in data flows, which provides information about each action embedded in a control flow model. By modeling and analyzing the business process of the e-commerce system, this method can dynamically detect changes in user behaviors, transaction processes, and noncompliance situations, and comprehensively analyze and identify fraudulent transactions from multiple perspectives. Important contributions of this paper are listed as follows:

1) A conformance checking method based on process mining is applied in the field of e-commerce transactions to capture the abnormalities.

2) A user behavior detection method is proposed to perform comprehensive anomaly detection based on Petri nets.

3) An SVM model is developed by embedding a multi perspective process mining into machine learning methods to automatically classify fraudulent behaviors.

In the rapidly evolving realm of ecommerce, transactions involving multiple participants present unique challenges in detecting and preventing fraud. This project introduces an innovative fraud detection method specifically crafted for multiparticipant ecommerce transactions. By integrating sophisticated techniques such as user behaviour analysis, anomaly detection, and machine learning, our approach aims to provide a robust solution to enhance transaction security and safeguard against fraudulent activities in the digital marketplace. In the intricate landscape of e-commerce, where transactions involve a dynamic interplay among multiple participants such as buyers, sellers, and intermediaries, the challenge of detecting

fraudulent activities looms large. Recognizing the complexities of this multifaceted environment, our proposed method adopts a Mult perspective approach to fortify the accuracy and efficiency of fraud detection mechanisms. Our methodology commences with a meticulous examination of user behaviours, leveraging diverse techniques such as behavioural analysis and scrutiny of transaction histories. By discerning patterns inherent in normal user interactions within the e-commerce ecosystem, we establish a baseline that facilitates the identification of abnormal behaviours. This foundational step is pivotal for creating a robust fraud detection system. Moving beyond behaviour detection, our approach incorporates a comprehensive analysis of abnormalities for feature extraction. Employing sophisticated anomaly detection algorithms, we scrutinize transaction data to unveil irregular patterns indicative of potentially fraudulent activities. This meticulous process enables the extraction of crucial features that serve as pivotal indicators for effective fraud

1.1 OBJECTIVE

The primary objective of this project is to develop a multi-perspective fraud detection method for multiparticipant e-commerce transactions. By analyzing user behavior, transaction histories, and detecting abnormalities, the aim is to accurately and efficiently identify fraudulent activities in e-commerce platforms. Specifically, the method involves:

- Behavioral Analysis: Understanding and profiling typical user behaviors to create a baseline for normal activity in e-commerce transactions.
- Anomaly Detection: Identifying and analyzing deviations from established normal behavior to detect potential fraudulent activities.
- Feature Extraction and Classification: Extracting key features from transaction data and employing an ensemble classification model to distinguish fraudulent activities from legitimate ones.

The goal is to enhance fraud detection accuracy and reduce false positives, leading to a more secure and trustworthy e-commerce environment.

1.2 PROBLEM STATEMENT

Fraud in multiparticipant e-commerce transactions (involving buyers, sellers, and intermediaries) poses a significant challenge to both businesses and consumers. Current fraud detection methods often struggle to address the complexity of such transactions, which may involve diverse interactions and data points across multiple parties. These methods tend to focus on individual participant behavior or isolated transaction anomalies, leaving gaps in the detection of fraudulent activities that span multiple entities within the ecosystem.

1.3 SOFTWARE REQUIREMENTS

To operate a program smoothly, certain hardware and software must be in place, and this is known as the software requirements. These prerequisites typically aren't part of the application installation package and need to be installed separately.

Platform - A platform is a foundational piece of hardware or software that allows software to execute on a computer. Computer hardware, software, languages, and runtime libraries are all examples of common platforms. Operating systems are among the most fundamental components of any computer. While it's possible for software to incompatibly with later versions of the same OS, backward compatibility is typically preserved. Although it is not necessarily the case, most Windows XP software will not run on Windows 98. Linux versions employing Kernel v2.2 or v2.4 are rarely compatible with software developed using later Linux Kernel v2.6 capabilities.

APIs and drivers – Newer drivers or an API are required by software that makes advantage of high-end display devices. Media-related tasks, especially game development, are handled by Microsoft's DirectX suite of application programming interfaces.

Web browser - The default browser is utilized by the majority of web apps and software that is dependent on Internet technologies. Microsoft Internet Explorer is widely used by Windows users and makes use of ActiveX components, which are not without their risks.

1) **Software : Anaconda**
2) **Primary Language : Python**
3) **Frontend Framework : Flask**
4) **Back-end Framework : Jupyter Notebook**

5) **Database : Sqlite3**
6) **Front-End Technologies : HTML,CSS,JavaScript and Bootstrap4**

1.2 HARDWARE REQUIREMENTS

Hardware, sometimes known as a computer's physical resources, is the most typical collection of requirements provided by an OS or software program. A hardware compatibility list (HCL) is a part of many operating systems that accompany a hardware requirements list (HRL). An HCL is a catalogue of known compatible and, at times, incompatible hardware components for a certain program or operating system. What follows is an analysis of the hardware specifications in further detail.

Computer network architecture — every operating system is specifically designed to work on a certain kind of machine. The ideal operating system and hardware configuration for a lot of apps is carefully considered before development begins. There are platforms that can do this, but most applications and operating systems still need to be recompiled in order to work on these platforms. Here is a rundown of some of the most widely used OSes and the underlying architectures that power them.

Processing power—an essential system requirement for any program is the processing capacity of the central processing unit (CPU). The x86 architecture's software often takes the CPU type and clock speed into account when estimating processing capabilities. A lot of other power and performance-affecting CPU characteristics, such as bus speed, cache, and MIPS, are ignored. For instance, the throughput speeds may vary greatly amongst CPUs of the same manufacturer, even when their clock rates are identical. This indicates that this concept of power is often incorrect. The widespread usage of Intel Pentium CPUs makes them frequent class discussions topics.

While programs are active, all of the data stored by a computer is accessed from random access memory (RAM). Calculating the memory utilisation of an application, together with its

connected operating system, files, and supporting apps, requires careful consideration of all these processes. The optimal performance of unrelated programs is critical on a multi-tasking computer system, which we kept in mind while creating this criteria.

Many factors influence how much more space a hard disc needs. Think about how big the application is, how many temporary files are created and stored when the software is installed or used, and how likely it is that you will need to use swap space if your RAM isn't enough.

A high-end display adapter is often listed as a system requirement in programs that need a better-than-average computer graphics display, including graphics editors and high-end games.

Some peripherals are used extensively by certain software packages, which calls for their improved functionality or performance. A few other examples of peripherals include optical drives, mouse, keyboards, network adapters, and pointing devices.

1) Operating System : Windows Only

2) Processor : i5 and above

3) Ram : 8gb and above

4) Hard Disk : 25 GB in local drive

FEASIBILITY STUDY

2. FEASIBILITY STUDY

Feasibility Study

This section provides a business proposal outlining the project's general outline and estimated costs after assessing the project's viability. During system analysis, the feasibility of

the proposed system is examined. In this way, we know the proposed remedy won't put undue strain on the business. The first step in doing a feasibility study is to identify the most important requirements of the system.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.1 ECONOMICAL FEASIBILITY

This study delves into the system's impact on the organization's finances. Funds for system research and development are limited at the company. Every dollar must have a purpose. Since most of the technologies used are free, the built system stayed within budget. Make sure to only buy individualized products.

2.2 TECHNICAL FEASIBILITY

This study evaluates the technological feasibility, or the requirements of the system. To avoid wasting technical resources, no system should be constructed. Our technological resources will be put to the test. Customers can anticipate lofty standards. Due to the low or nonexistent modification requirements, the current system can only have a minor need.

2.3 SOCIAL FEASIBILITY

Researchers examine how users feel about the system. It also includes training people to make the most of the technology. Everyone needs to see the system for what it is: a necessary evil. The only thing that matters for users to accept the system is how it is presented to them. Being the system's end user, he needs to feel comfortable enough to provide helpful feedback.

LITERATURE SURVEY

3. LITERATURE SURVEY

1) P. Rao et al, The e-commerce supply chain and environmental sustainability: An empirical investigation on the online retail sector,2021

In the rapidly expanding realm of ecommerce, particularly in the business to-consumer (B2C) online retail sector, the environmental conceptual models derived from literature to investigate the environmental impacts of e-commerce. Collecting 303 responses through a structured questionnaire from the Gulf Cooperation Council (GCC) countries, the study validates and evaluates the proposed models, assessing the relevance of each construct and its underlying items.

2) E. A. Ministering, and G. Manita, An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection, 2019

The escalating complexity and transnational nature of illegal activities in online financial transactions have led to substantial financial losses for both customers and organizations. Countering this challenge, numerous techniques have been proposed for fraud prevention and detection in the online environment. However, each of these techniques exhibits distinct characteristics, advantages, and drawbacks, making it imperative to comprehensively review and analyse the existing research in fraud detection. This paper employs a systematic quantitative literature review methodology to identify the algorithms used in fraud detection and analyses each algorithm based on specific criteria.

3) Wang yang Yu; Yadi Wang; Lu Liu; Yusheng An; Bo Yuan; John Panneerselvam, A Mult perspective Fraud Detection Method for Multiparticipant E-Commerce Transactions,2021

In the persistent challenge of detecting and preventing fraudulent transactions within e-commerce platforms, traditional security systems relying on historical order information often fall short, given the elusive nature of online activities. Recognizing the limitations of existing approaches that neglect dynamic user behaviours, this article proposes an innovative fraud detection method that seamlessly integrates machine learning and process mining models for real-time monitoring. The methodology unfolds in three key stages. First, a business-to-customer (B2C) e-commerce platform is modelled, incorporating a robust framework for detecting user

behaviours. This foundational process aims to better understand and adapt to the dynamic nature of user interactions within the platform. Second, the article introduces a method for analysing abnormalities, leveraging event logs to extract essential features crucial for fraud detection. This step ensures a nuanced understanding of irregular patterns indicative of potentially fraudulent activities.

SYSTEM ANALYSIS

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

Traditional fraud detection in e-commerce relies on analyzing static transaction data, often focused on one participant at a time. This approach limits effectiveness in identifying complex, dynamic fraud patterns, as it lacks multi-user behavior insights and adaptive mechanisms.

Disadvantages:

- Limited Fraud Detection Scope: Focuses on static transaction data and individual participants, making it challenging to detect complex, coordinated fraud patterns.
- Lack of Adaptability: Inflexible to evolving fraud tactics as it does not leverage behavior tracking or dynamic anomaly detection.
- Lower Accuracy: Without multi-participant insights, it often misses subtle fraudulent indicators.

4.2 PROPOSED SYSTEM:

The proposed approach utilizes a multi-perspective method that combines behavior tracking and anomaly detection across buyers, sellers, and intermediaries. By establishing normal interaction baselines and using an ensemble classification model, this method detects abnormalities and enhances fraud detection accuracy with machine learning techniques, like anomaly detection algorithms and feature extraction for fraud indicators.

Advantages:

1. Comprehensive Behaviour Analysis: Analyses multi-participant interactions, improving fraud detection accuracy.
2. Adaptive Fraud Detection: Uses anomaly detection algorithms to adapt to new fraud patterns dynamically.
3. Enhanced Accuracy: Ensemble classification and feature extraction increase detection efficiency and effectiveness.

4.3 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modiling
5. Predicting

4.4 NON FUNCTIONALREQUIREMENTS

One approach of gauging software quality is via Non-Functional Requirement (NFR) evaluations. Respondivity, usability, security, and portability are important but non-functional aspects that decide the success of a software system. Something that isn't working is what the question "how fast does the website load?" is describing. Users may be disappointed if the system can't meet their needs, even if those needs aren't functional. You may constrain or restrict the design of the system based on non-functional needs in any agile backlog. For example, the site still has to load in under three seconds even when 10,000+ people are using it at peak periods. Both practical and non-practical requirements must be specified.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

SYSTEM DESIGN

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

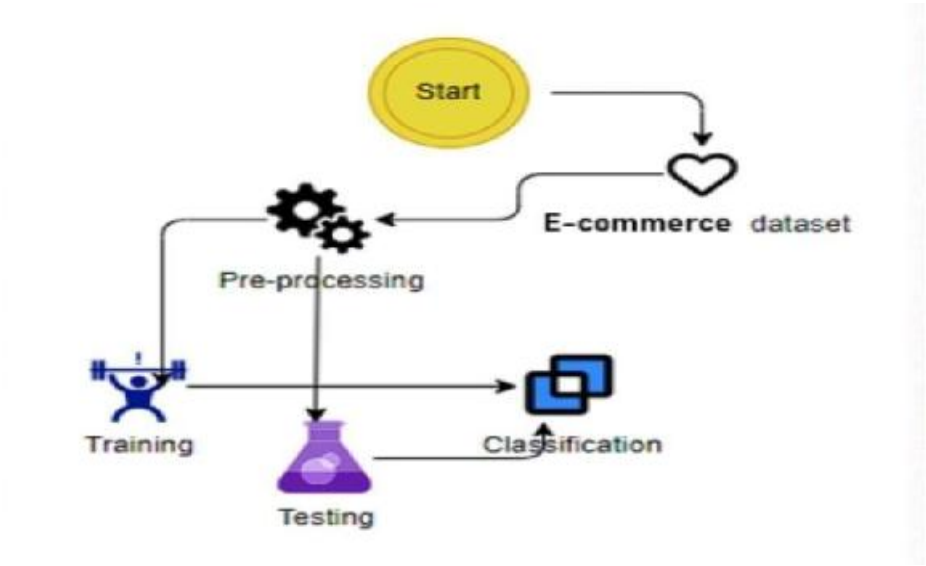


Fig.5.1.1 System architecture

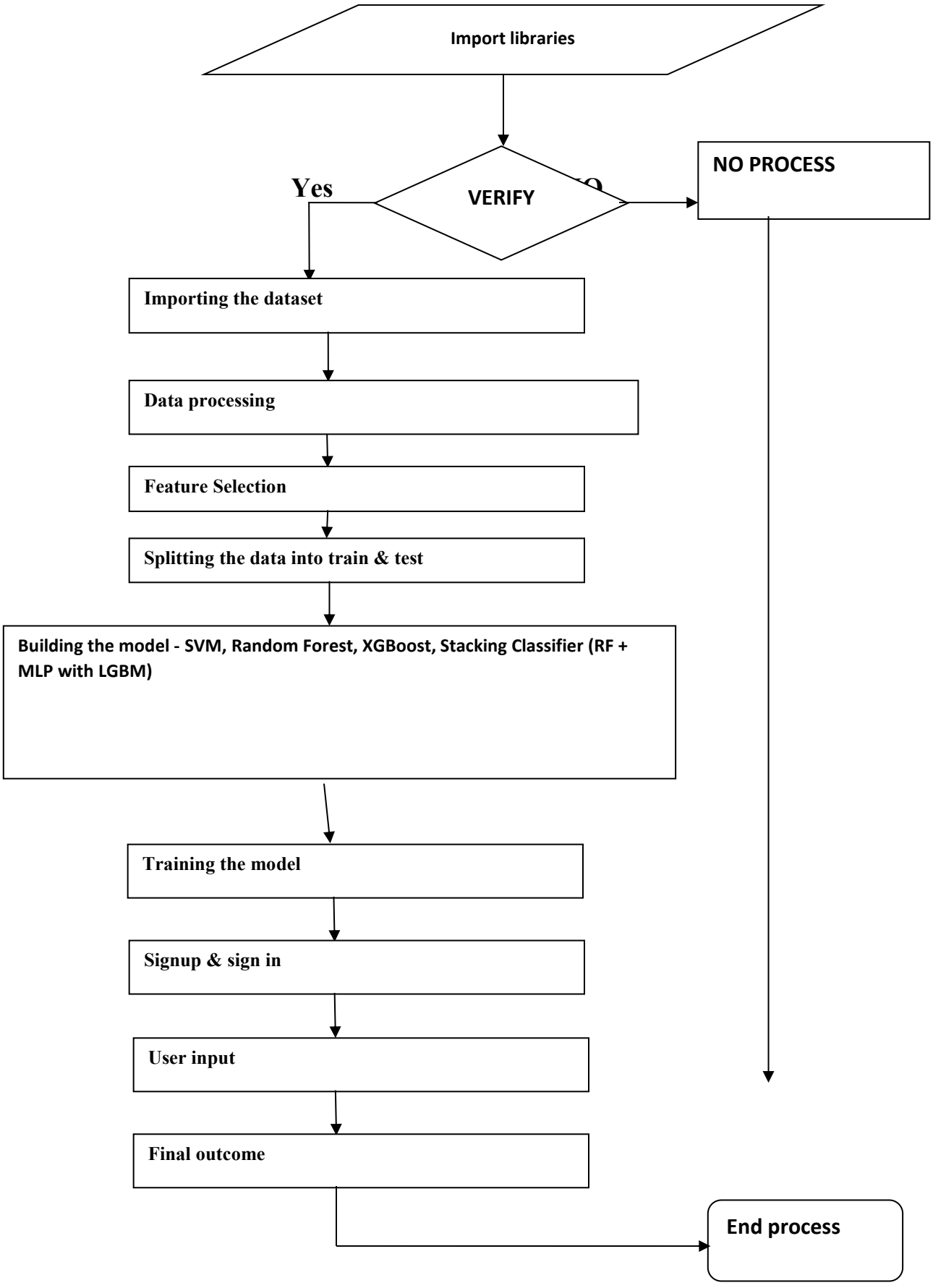
DATA FLOW DIAGRAM:

The DFD is sometimes known as a bubble chart. This straightforward visual formalism for depicting a system consists of three primary parts: data input, data processing, and data output.

Data flow diagrams (DFDs) are crucial to the modelling process. To model the parts of the system is the point of employing it. Everything that the system performs, any data it consumes, any third parties it communicates with, and any information that flows through it is included by this.

Lastly, DFD reveals the data's evolution as it traverses the system. The changes that take place when data goes from an input to an output may be visually represented in a data flow diagram.

Bubble charts are sometimes known as data flow diagrams (DFDs). Any degree of abstraction may be shown in dynamic functional diagrams (DFDs). The complexity of its functions and the amount of data passing through them are two ways to classify DFD.


Import libraries
VERIFY
Yes
NO PROCESS
Importing the dataset
Data processing
Feature Selection
Splitting the data into train & test
Building the model - SVM, Random Forest, XGBoost, Stacking Classifier (RF + MLP with LGBM)
Training the model
Signup & sign in
User input
Final outcome
End process

5.2 UML DIAGRAMS

The abbreviation UML, which stands for the Unified Modelling Language, is very useful. The Universal Modelling Language, or "UML" for short, is one rule for software engineering that centres on objects. The Object Management Group is in charge of supervising and developing the standard.

The Unified Design Language's (UDL) primary objective is to provide a standard language for developing object-oriented software systems. Using its syntax and meta-model, modern UML is quite dependent. It is possible to include more procedures or methods into UML in the future. Any kind of artefact, from software systems to non-software systems to business models, may be more easily specified, visualised, produced, and recorded with the help of the Unified Modelling Language.

Large and complicated systems may be represented using the Unified Modelling Language (UML), which offers a foundation for engineering best practices. Unified Modelling Language (UML) is a crucial component of creating object-oriented software or any application. When it comes to designing software projects, the Unified Modelling Language (UML) is primarily dependent on visual notations.

The original intent of the Unified Modelling Language (UML) was to accomplish the following:

1. Enable the development and sharing of practical models by providing a visual modelling language that is both expressive and easy to use.

2. Provide tools for customisation and expansion to enhance the adaptability of the main principles.

Thirdly, don't be closed off to other ways of thinking and doing things when it comes to growth, no matter what language you use.

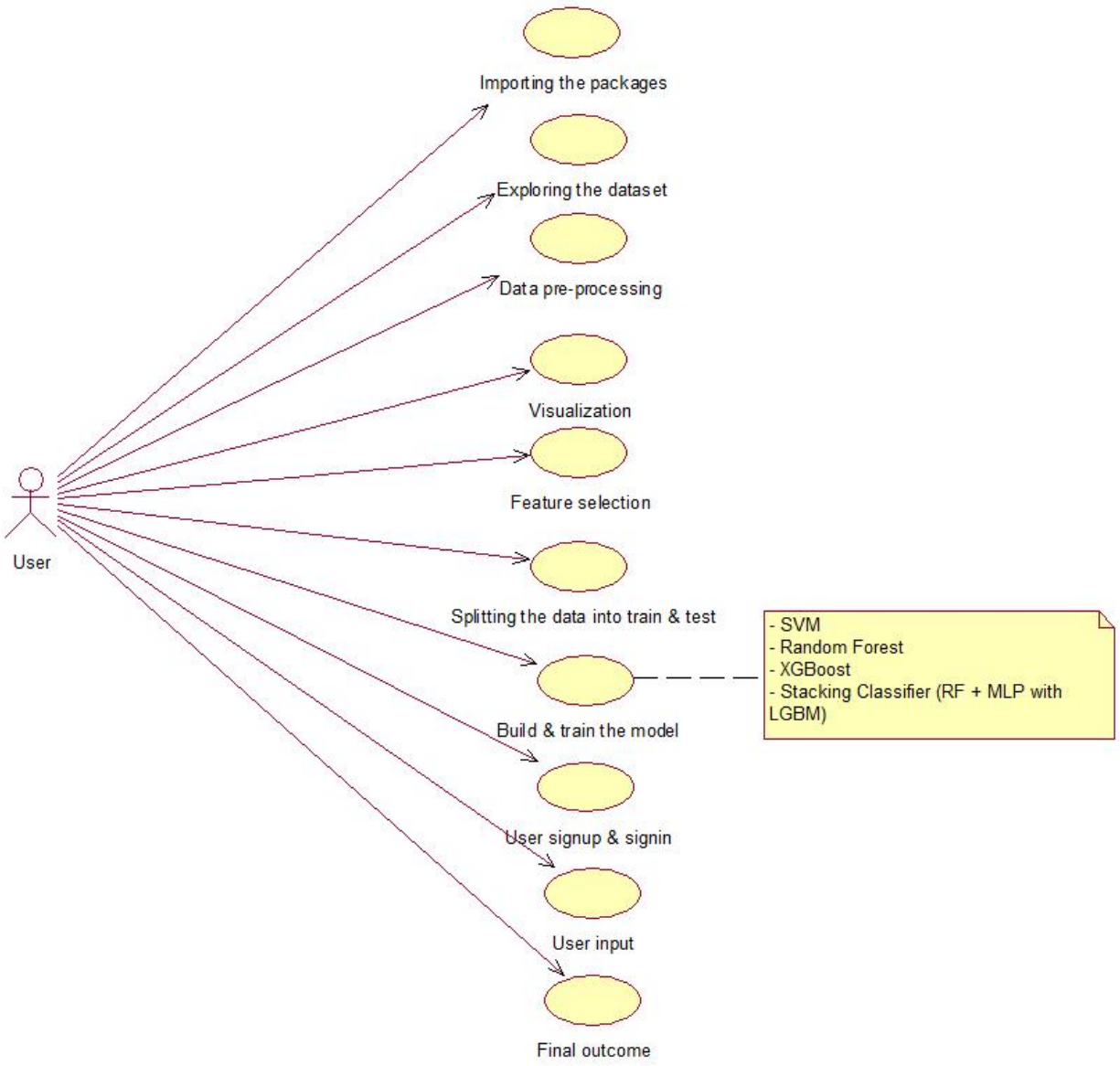
Developing a strong formal foundation is the fourth stage in becoming a master modeller.

5. Require that a larger number of individuals acquire OO tools.

(6) Provide backing for developer-level ideas including frameworks, components, partnerships, and patterns.

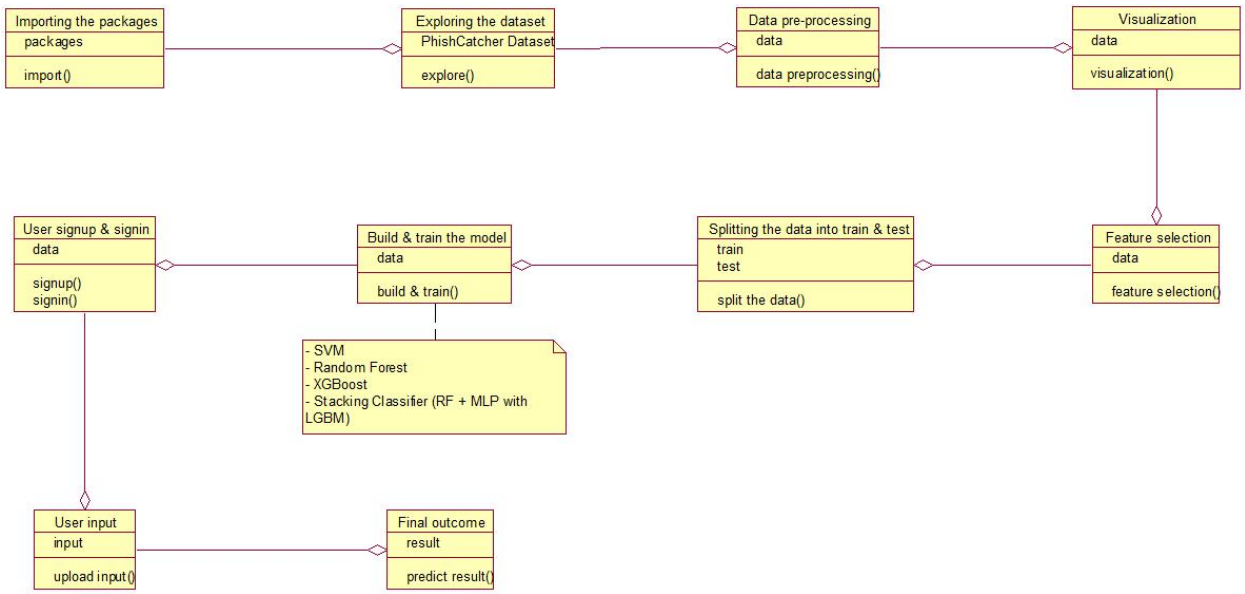
7. Follow the standards set by the industry.

Among the many UML behavioural diagrams, the use case study is responsible for creating use case diagrams. You might think of it as a visual representation of the relationships between the many use cases, the people using the system, and the goals they are trying to achieve. A use case diagram may assist break down a complex system into its component elements and the tasks they carry out. It is feasible to demonstrate the operation of the system's different components.



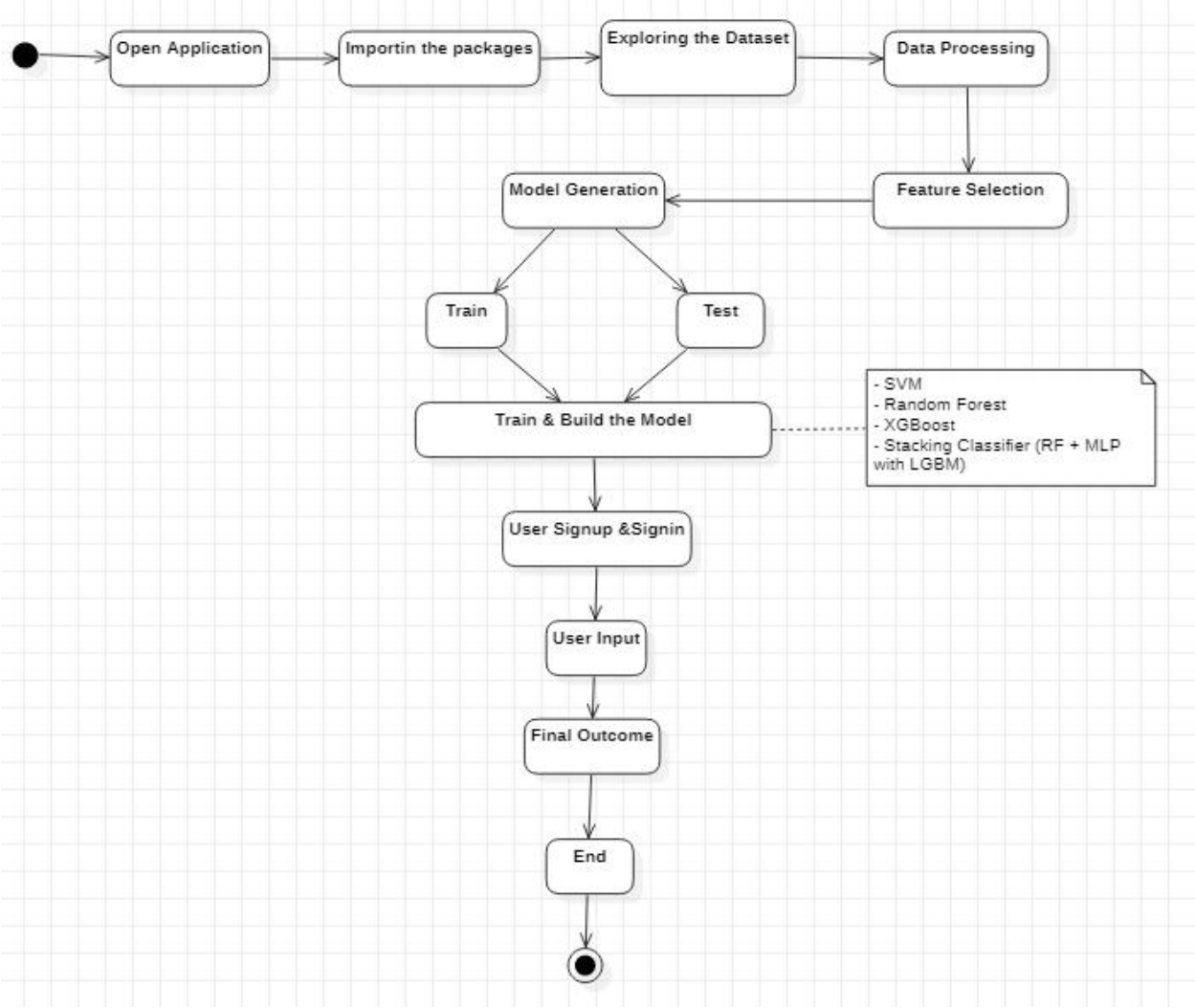
Class diagram:

It is possible to improve the use case diagram and provide a more detailed specification of the system's architecture by using the class diagram. The use case diagram specifies a group of actors and the class diagram details the connections between them. Your classes may use either a "is-a" or a "has-a" connection. It is possible that the classes shown in the class diagram may perform certain tasks. Classes accomplish their goals in part by means of the "methods" they provide. This aside, it's possible that distinct classes use different sets of "attributes" to distinguish themselves apart from one another.



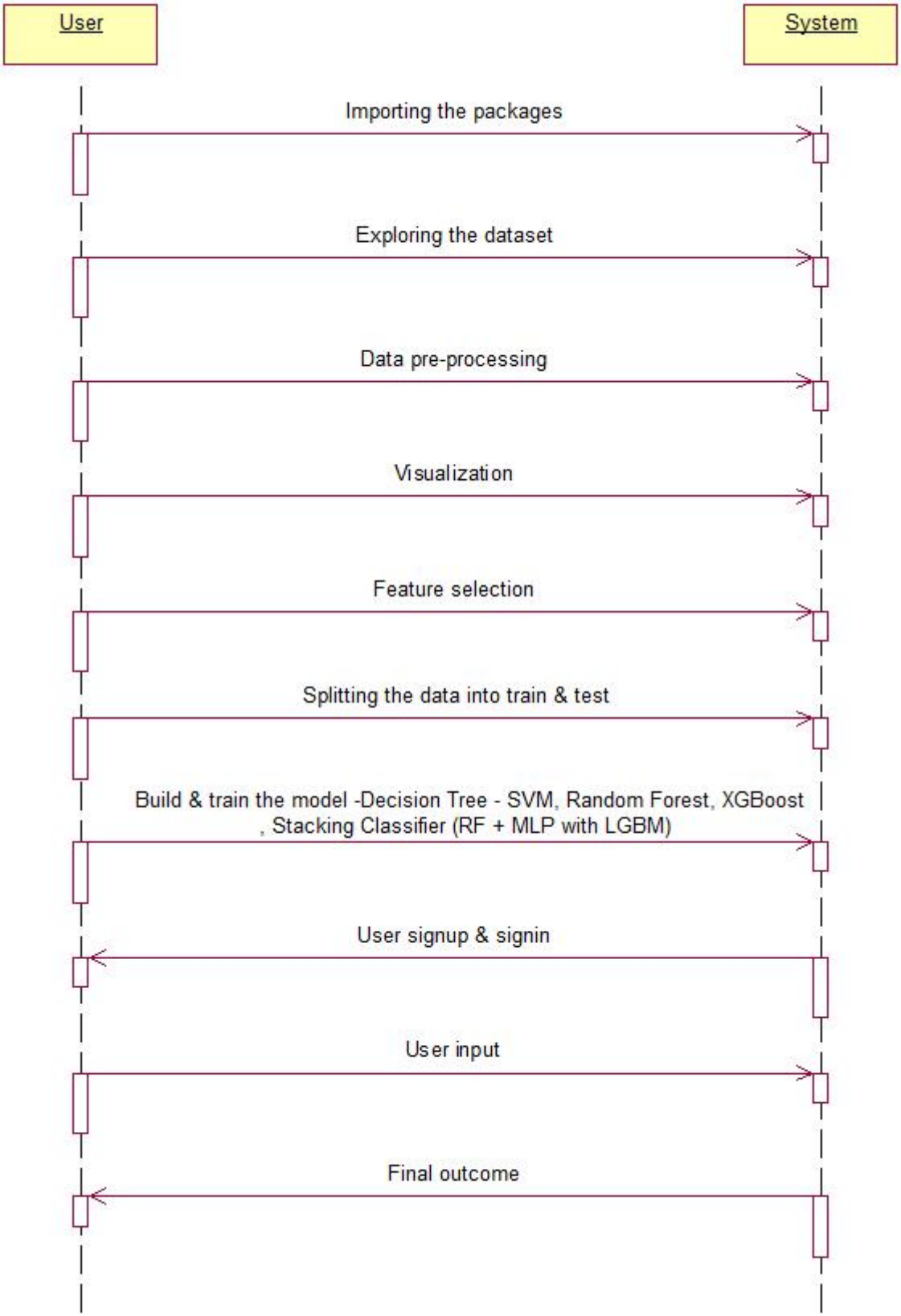
Activity diagram:

The activity diagram depicts the system's process flows. An activity diagram is quite similar to a state diagram in that it also shows guard conditions, starting and ending states, tasks, and activities as well as transitions between them.



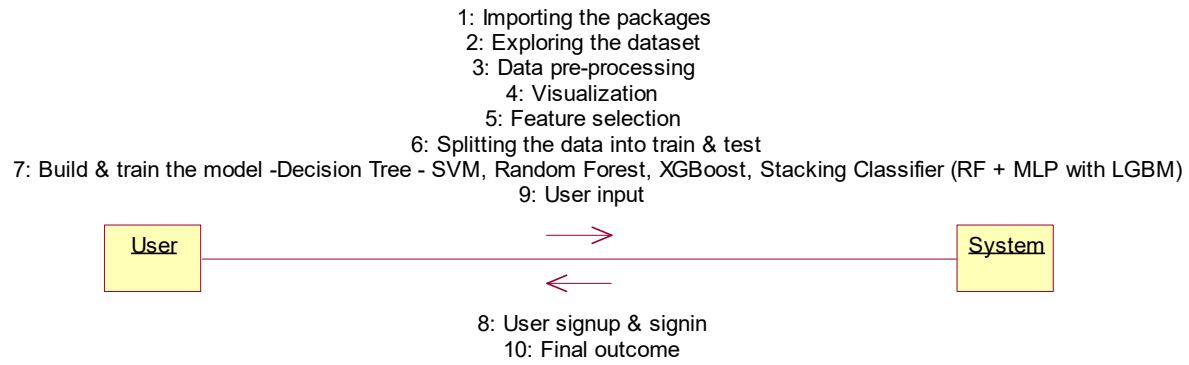
Sequence diagram:

Draw a sequence diagram to show the interdependencies between the system's components. One distinguishing feature of a sequence diagram is the temporal ordering of occurrences. Because of this, interactions between objects may be shown in great detail and in a sequential fashion. The different components in the sequence diagram are able to interact with one another via the "messages" they use.


User
System
Importing the packages
Exploring the dataset
Data pre-processing
Visualization
Feature selection
Splitting the data into train & test
Build & train the model -Decision Tree - SVM, Random Forest, XGBoost , Stacking Classifier (RF + MLP with LGBM)
User signup & signin
User input
Final outcome

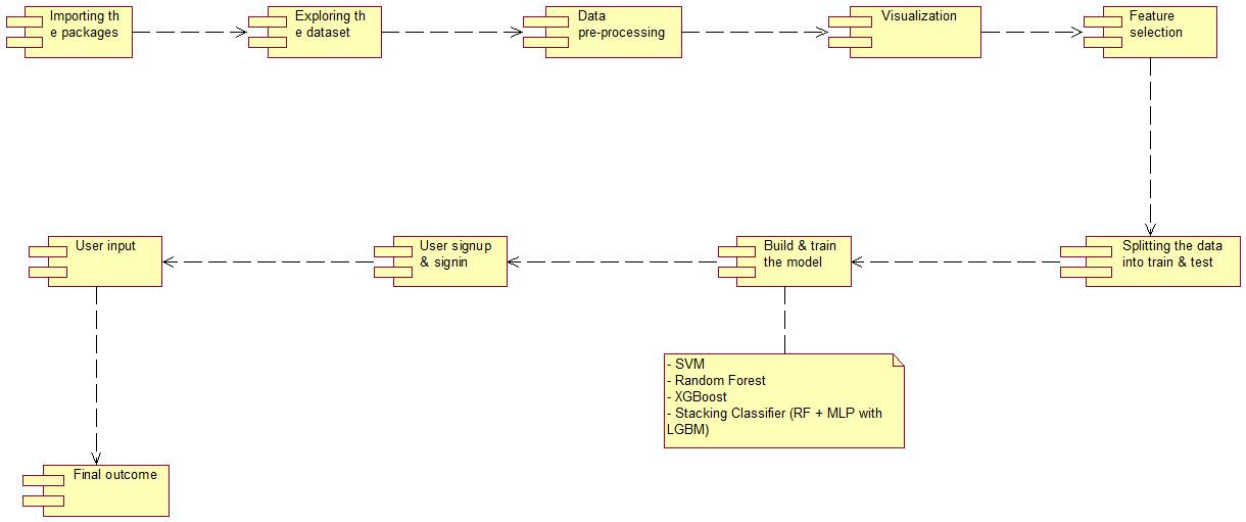
Collaboration diagram:

The relationships between many things may be shown in a cooperation diagram. The order of the interactions may be seen by looking at the numbered list of interactions. Every conceivable interaction between components may be shown in the cooperation diagram.



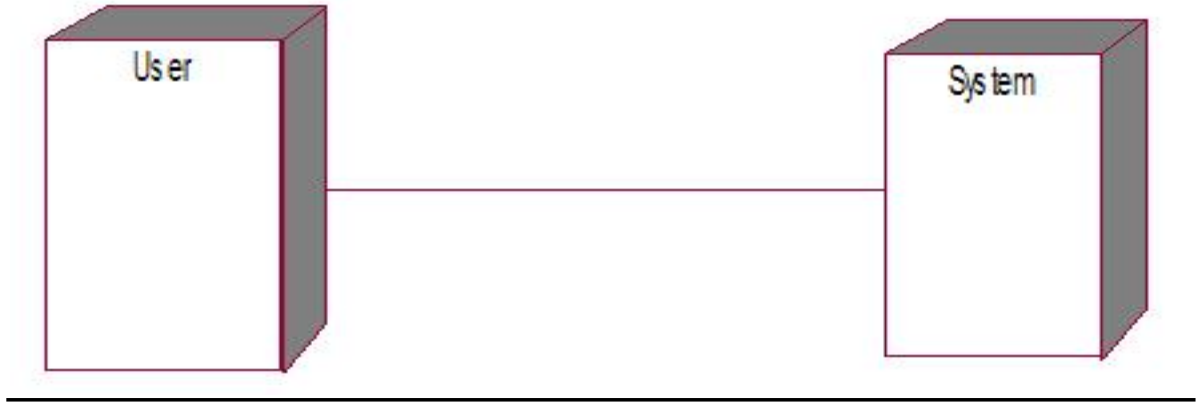
Component diagram:

The system's essential components are shown in the component diagram. This schematic provides an overview of the system by showing its primary parts and their interconnections. To better understand how a built or created system works, a component diagram may be very helpful.



Deployment diagram:

In the deployment diagram, you can see how the application's runtime components are configured. This graphic becomes very useful when a system is ready for deployment.



IMPLEMENTATION

6. IMPLEMENTATION

MODULES:

- **Data loading:** using this module we are going to import the dataset.

Dataset Link:

https://www.kaggle.com/datasets/shauryapanpalia/cyberbullying-classification

Dataset Description:

The data in this collection is based on suspicious activities, such as cyber bullying that uses racial and ethnic profiling extensively and makes threats and uses harsh language. The information was sourced via Twitter and Face book groups.
Data is categorised according to the presence or absence of potentially problematic language in tweets and comments. After data scraping, manually assign a questionable data value of -1 and a non-doubtful data value of 0.

We will be using this module to get the data ready for analysis.

This module allows us to create a 'train set' and a 'test set' from the information we have. The process is called model making. Naïve Bayes and logistic regression are used. The Idea of Bayes Probability and Fuzzy GA An amalgamation of LR fuzzy GA, RF, AB, and stacking classifiers into a single voting classifier. Accuracy of algorithms' predictions

This module will be used to gather the login and registration information of users.

The users supplied data: To gather information for forecasts, you may use this module. The projected final outcome is shown.

Using an ensemble approach to combine the predictions of many models led to a more accurate and reliable final forecast.

Improvements in performance, maybe as high as 99% accuracy, are possible with the addition of two additional ensemble methods: Voting Classifier and Stacking Classifier.

Formulas for calculations [F]

One method of classification that relies on Bayes' Theorem is Naive Bayes, which assumes that predictors are independent. A Naive Bayes classifier, in its most basic form, takes for granted that there is no correlation between two features that share a class.

To solve binary classification issues, logistic regression determines the likelihood of a result, occurrence, or observation. It falls within the category of supervised machine learning. The model can only provide a yes/no, 0/1, or true/false result.
The Naive Bayes algorithm is among the most well-known and user-friendly approaches to machine learning classification, alongside Naive Bayes Fuzzy GA. One may calculate probabilities and conditional probabilities using Bayes' Theorem.

One definition of a fuzzy genetic algorithm is an algorithm that uses fuzzy logic techniques to build some of its components. The algorithm is thereafter described as a sequence of commands.

Because they lack both volume and mass, particles serve as the basic unit of measurement in the PSO algorithm. One such example is Naive Bayes Fuzzy PSO. The trajectories of particles are fine-tuned by constantly changing their velocities in response to their own and other particles' flight experiences within the search space.

An ML model called the Voting Classifier (AB + RF) accepts a number of models as input and returns the class that has the best chance of being selected.
One ensemble approach to machine learning is stacking classifiers, which looks for methods to integrate the predictions of many successful ML models. Many Python programmers prefer to use the scikit-learn package when they need to build stacking ensembles.

6.2 SAMPLE CODE:

```
# Necessary imports


import numpy as np

import pandas as pd

import tensorflow as tf

import matplotlib.pyplot as plt

import seaborn as sns

import time

import multiprocessing


from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from collections import Counter

from sklearn.preprocessing import LabelEncoder, StandardScaler


from sklearn.model_selection import train_test_split

from imblearn.over_sampling import SMOTE, ADASYN

from sklearn.metrics import confusion_matrix, r2_score, mean_squared_error
```

```
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score,
accuracy_score, classification_report, precision_recall_curve

import warnings

warnings.filterwarnings("ignore")

df = pd.read_csv("/kaggle/input/network-anamoly-
detection/Train.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbyt
es","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin",
"numcompromised","rootshell","suattempted","numroot","numfilecreations",
"numshells","numaccessfiles","numoutboundcmds","ishostlogin",

"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",

"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate",
"srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate",
"dsthostdiffsrvrate","dsthostsamesrcportrate",

"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",

"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])

df.head()

df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)

df.isna().sum()

df.select_dtypes(exclude=[np.number])

df['attack'].loc[df['attack']!='normal']='attack'

le=LabelEncoder()

df['protocoltype']=le.fit_transform(df['protocoltype'])
```

```
df['service']=le.fit_transform(df['service'])

df['flag']=le.fit_transform(df['flag'])

df['attack']=le.fit_transform(df['attack'])

plt.figure(figsize=(20,15))

sns.heatmap(df.corr())

X=df.drop(['attack'],axis=1)

y=df['attack']

sns.countplot(df['attack'])

print("Class distribution: {}".format(Counter(y)))

scaler = StandardScaler()

scaler.fit(X)

X_transformed = scaler.transform(X)

lr=LogisticRegression()

lr.fit(X_transformed,y)

lr_pred=lr.predict(X_transformed)

lr_df=pd.DataFrame()

lr_df['actual']=y

lr_df['pred']=lr_pred

lr_df.head()

print(accuracy_score(y, lr_pred))
```

```
print(classification_report(y, lr_pred))

rf=RandomForestClassifier()

rf.fit(X_transformed,y)

rf_pred=rf.predict(X_transformed)

rf_df=pd.DataFrame()

rf_df['actual']=y

rf_df['pred']=rf_pred

rf_df.head()

print(accuracy_score(y, lr_pred))

print(classification_report(y, lr_pred))

test_df = pd.read_csv("/kaggle/input/network-anamoly-
detection/Test.txt",sep=",",names=["duration","protocoltype","service","flag","srcbytes","dstbyte
s","land", "wrongfragment","urgent","hot","numfailedlogins","loggedin",
"numcompromised","rootshell","suattempted","numroot","numfilecreations",
"numshells","numaccessfiles","numoutboundcmds","ishostlogin",

"isguestlogin","count","srvcount","serrorrate", "srvserrorrate",

"rerrorrate","srvrerrorrate","samesrvrate", "diffsrvrate",
"srvdiffhostrate","dsthostcount","dsthostsrvcount","dsthostsamesrvrate",
"dsthostdiffsrvrate","dsthostsamesrcportrate",

"dsthostsrvdiffhostrate","dsthostserrorrate","dsthostsrvserrorrate",

"dsthostrerrorrate","dsthostsrvrerrorrate","attack", "lastflag"])

test_df.head()
```

```
test_df.select_dtypes(exclude=[np.number])

test_df['attack'].loc[test_df['attack']!='normal']='attack'

test_df['protocoltype']=le.fit_transform(test_df['protocoltype'])

test_df['service']=le.fit_transform(test_df['service'])

test_df['flag']=le.fit_transform(test_df['flag'])

test_df['attack']=le.fit_transform(test_df['attack'])

test_df.drop(['land','urgent','numfailedlogins','numoutboundcmds'],axis=1,inplace=True)

X_test=test_df.drop(['attack'],axis=1)

y_test=test_df['attack']

sns.countplot(test_df['attack'])

X_test_transformed = scaler.transform(X_test)

test_pred=rf.predict(X_test_transformed)

rf_test_df=pd.DataFrame()

rf_test_df['actual']=y_test

rf_test_df['pred']=test_pred

rf_test_df.head()

target_names=["attack","normal"]

print(classification_report(y_test, test_pred,target_names=target_names))

tn, fp, fn, tp = confusion_matrix(y_test, test_pred).ravel()

print("True Negatives:",tn)
```

```
print("False Positives:",fp)

print("False Negatives:",fn)

print("True Positives:",tp)
```

SOFTWARE ENVIRONMENT

7. SOFTWARE ENVIRONMENT

What is Anaconda for Python?

The Anaconda tool makes it easy to build up an environment that is compatible with multiple versions of Python and its packages. With Anaconda, you can manage your project environments and install, delete, and update packages to the latest versions. In addition, Anaconda makes it easy to launch any project with a single click. This makes it an ideal choice for those who are new to Python and are just starting out.

With that context in mind, we will next look at the installation of Anaconda Python.

How to install Anaconda for Python?

Simply download the installer for your operating system from the Anaconda Documentation website and follow the on-screen instructions to get Anaconda up and running. Simply double-click the installer file to begin the installation process when it has finished downloading.

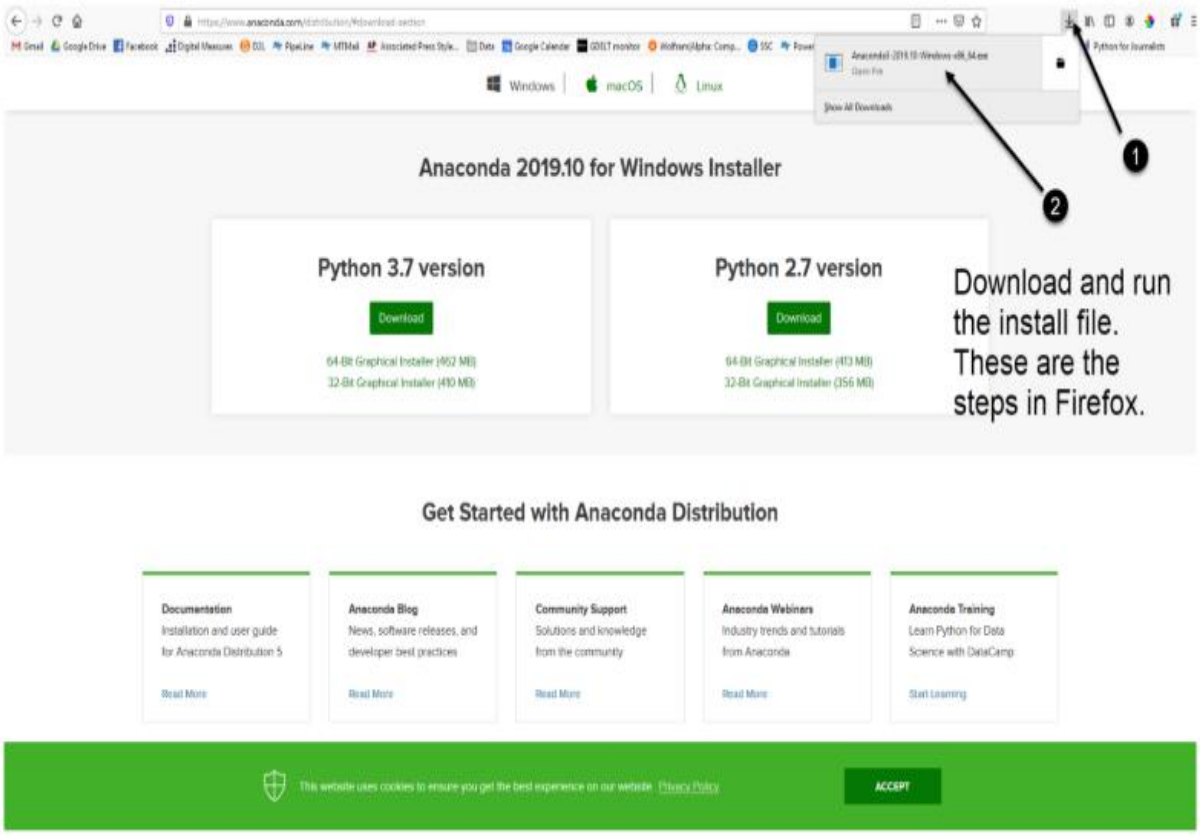
Acceptance of the terms and conditions may be accomplished by following the on-screen instructions. If asked to "add Anaconda to my PATH environment variable," just choose "yes." Doing so will enable you to include Anaconda in the PATH, a path that your OS uses to locate files.

The installation will end with an option to "Enable Anaconda as my default Python." shown. Select "yes" if you would want to use Anaconda as your default Python interpreter.

Anaconda Installation with Python

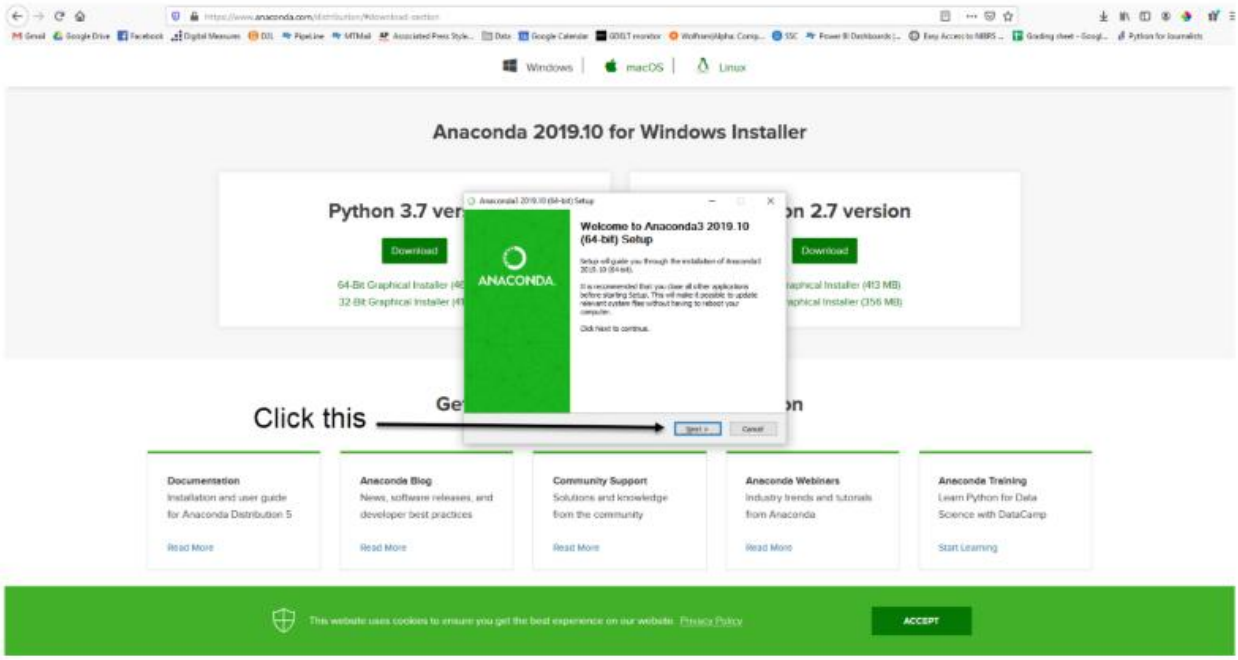
Installing Python Anaconda is the next step in the course. The latest version of Anaconda that is currently available is 2019.10. To get Anaconda installed on your computer, follow these steps:

To get Anaconda for Mac, Windows, or Linux, use this URL: - An Get the Anaconda archive.

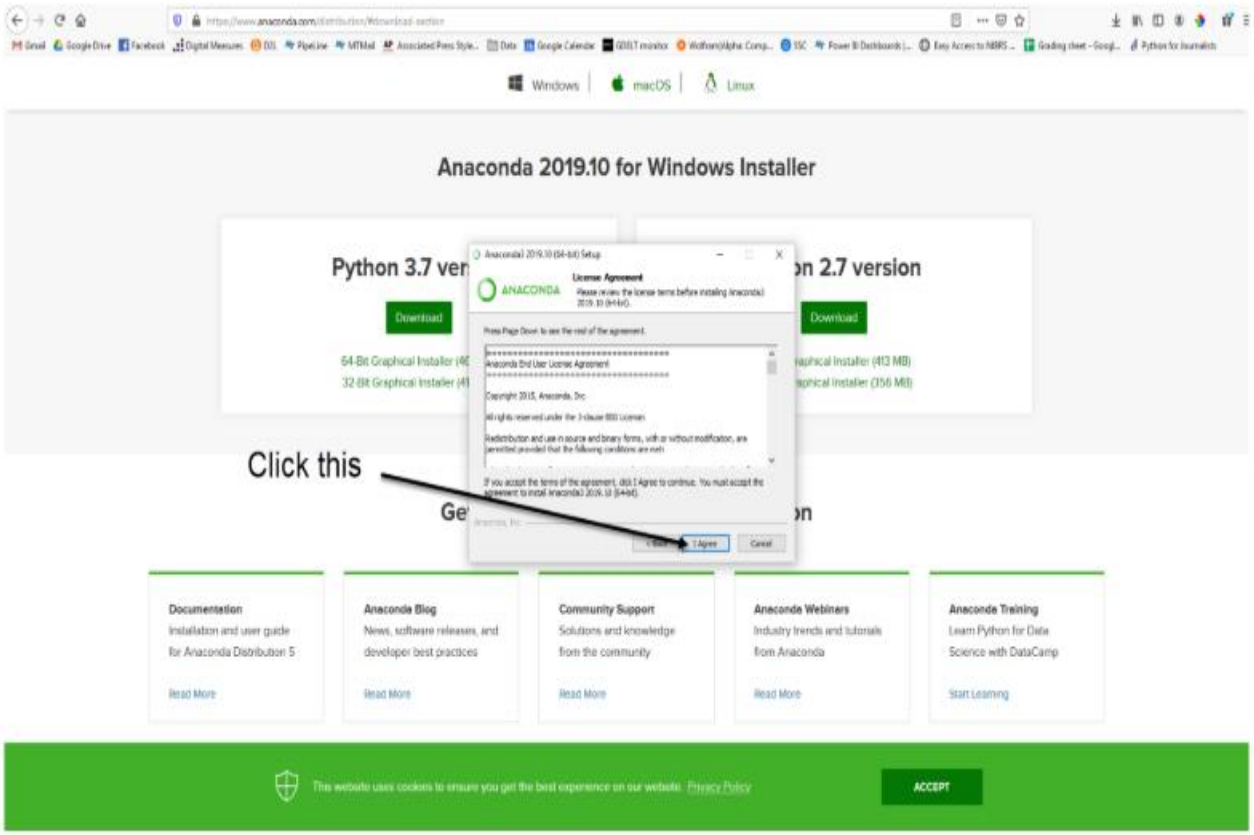


The installers for Python 3.7 and Python 2.7 are both available for download at the moment. Not to mention that it's free for both 32-bit and 64-bit laptops.

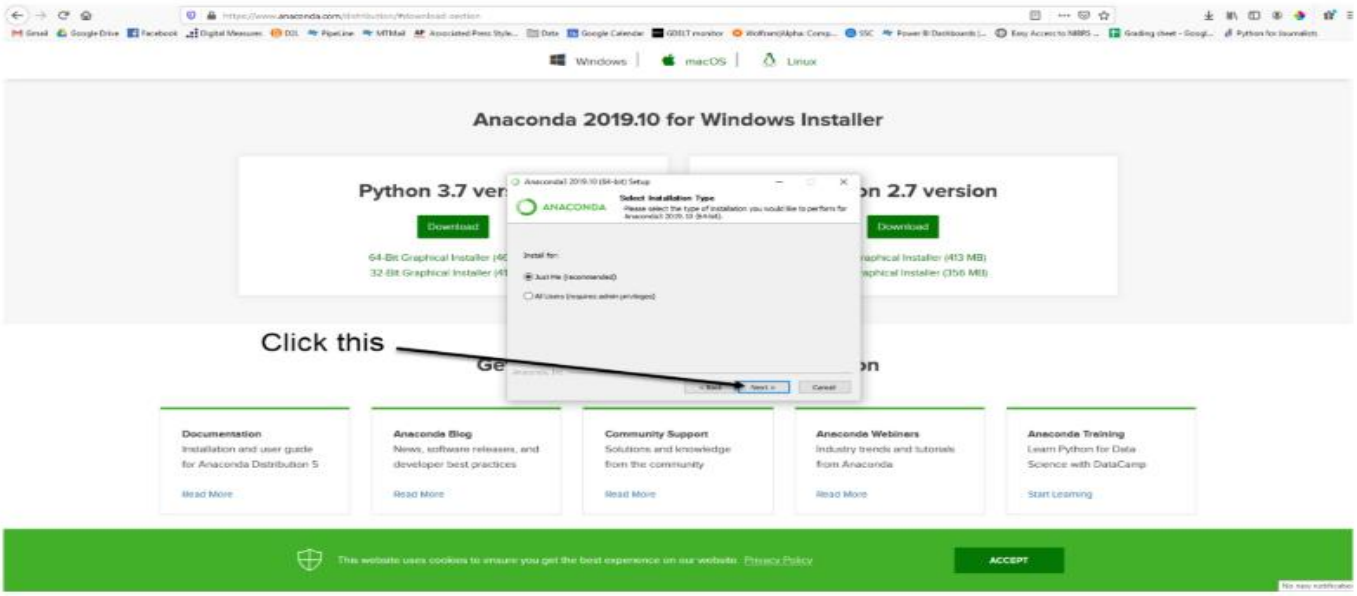
Step 2: Run the application you downloaded by double-clicking the.exe file. Here is how Anaconda is set up. Moving on, press the button.



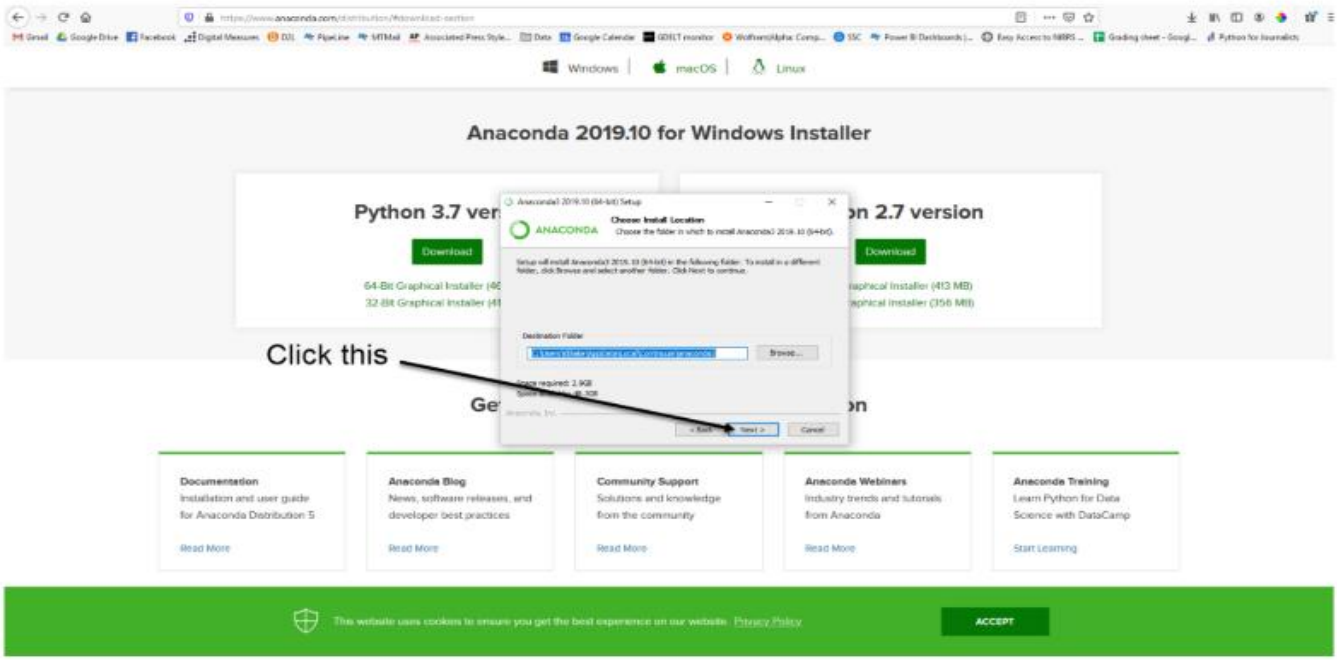
Step 3: The licence agreement will now be shown. Tap the "I Agree" button.



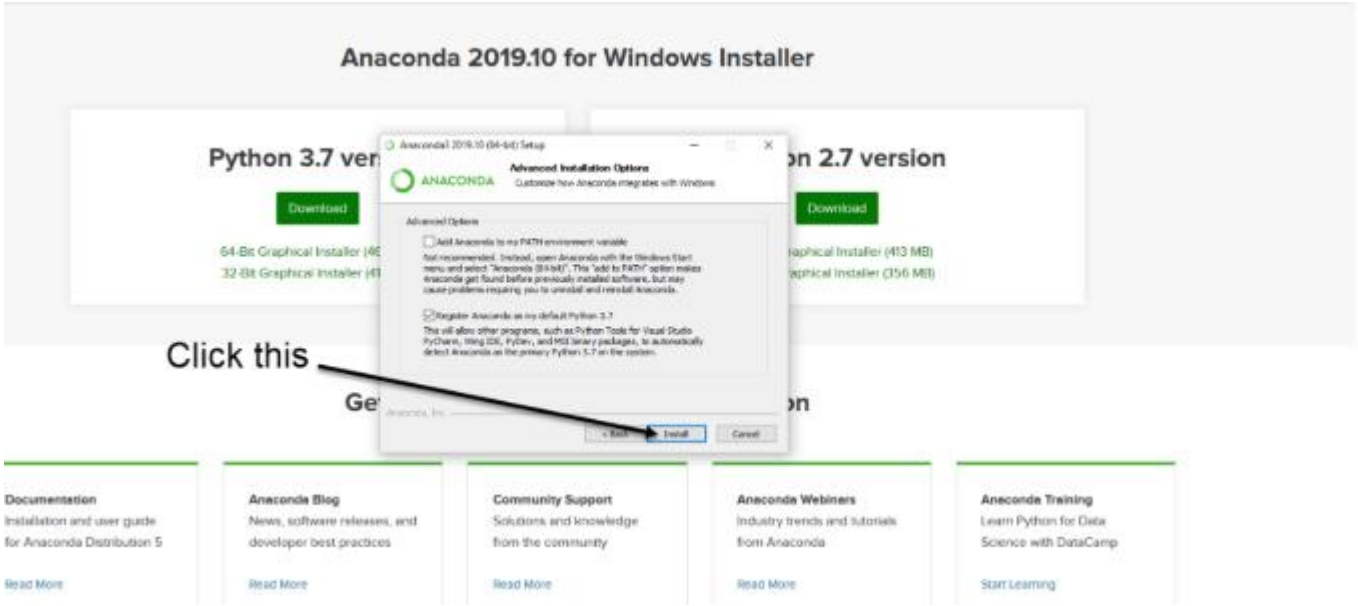
4. You have the option to install it for your own use or for all users. You will need administrative rights to install it for every user.



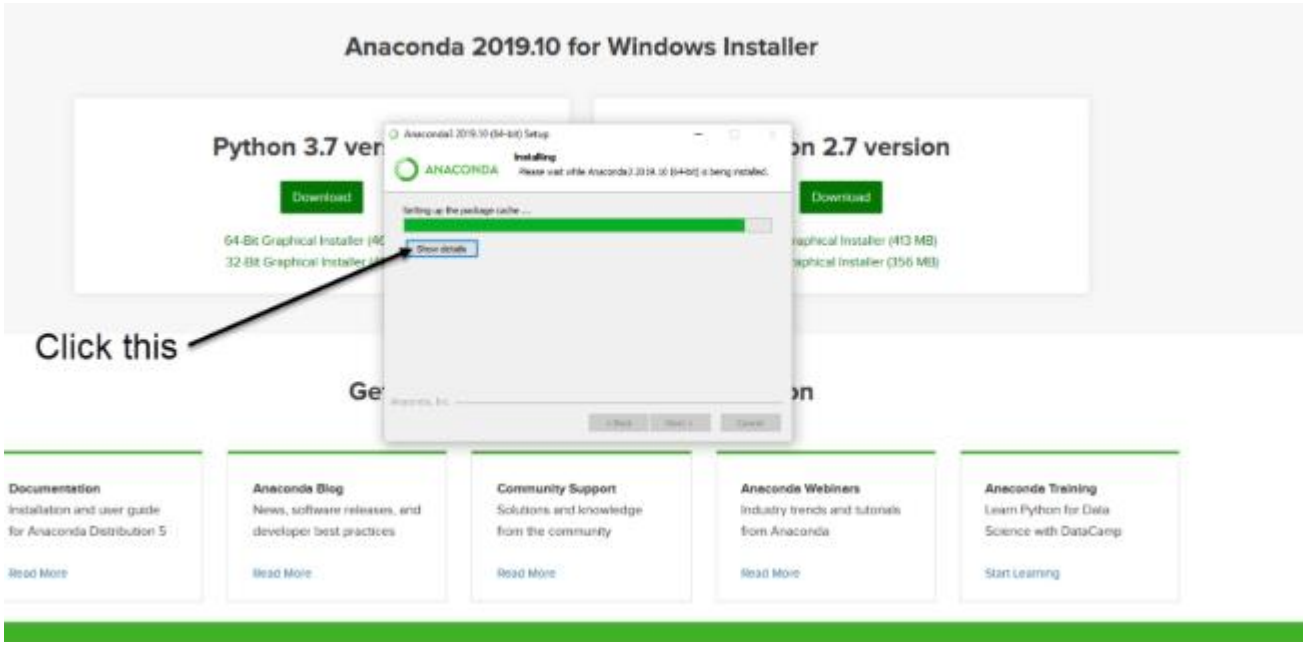
5. Go ahead and choose the installation location. The amount of space you require and the amount that is available are both shown here.



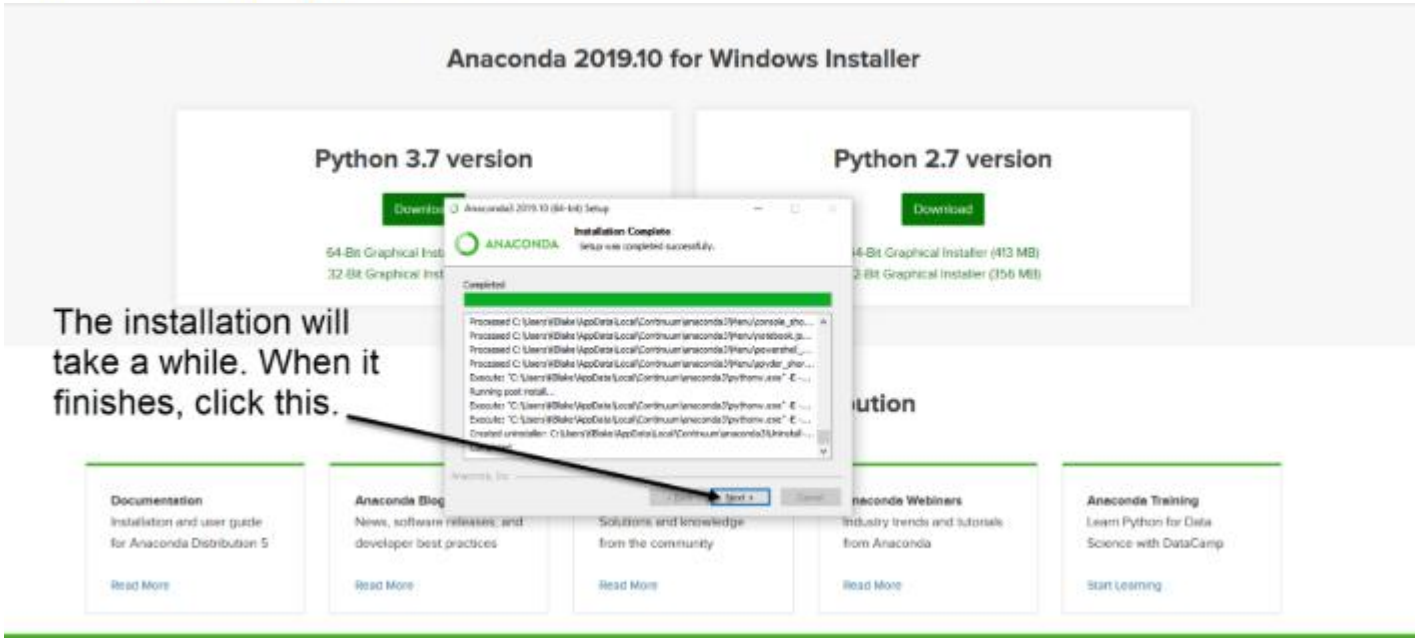
6. Several advanced choices will be presented to you now. Adding Anaconda to the PATH environment variable will make it the main executable for Python 3.7. If you include it in the PATH, it will be available before other installations. Select "Install" from the menu.



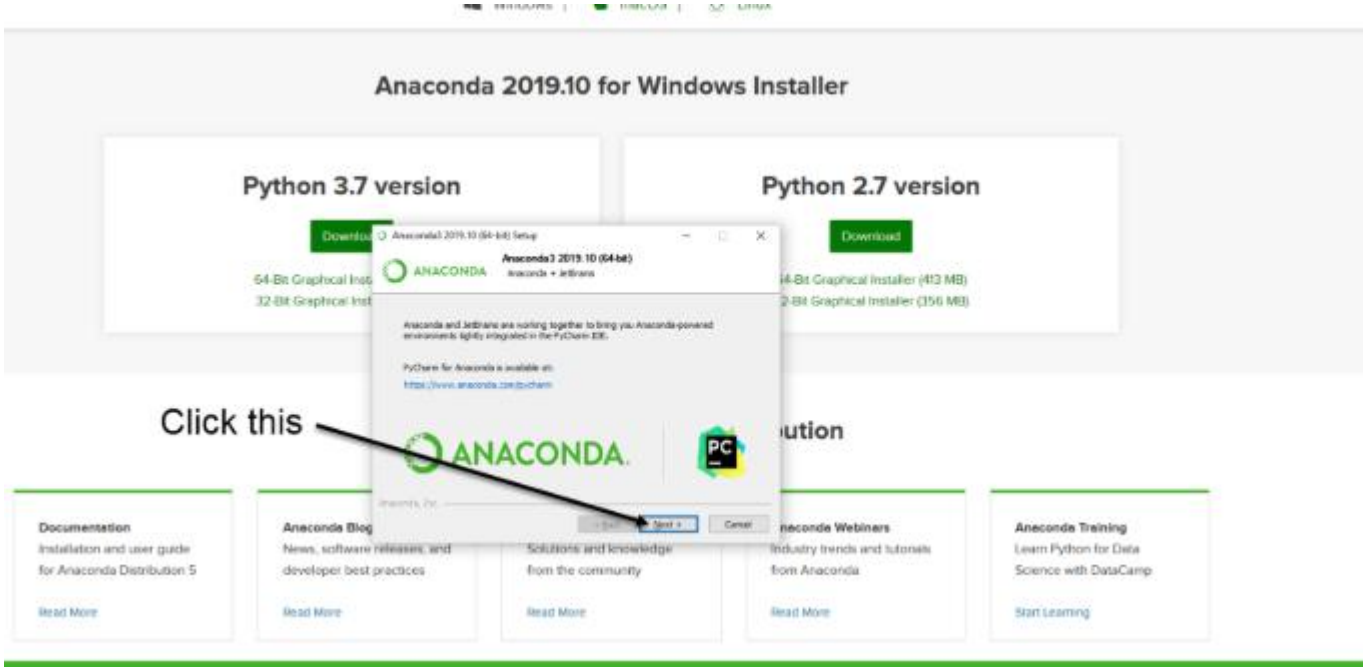
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



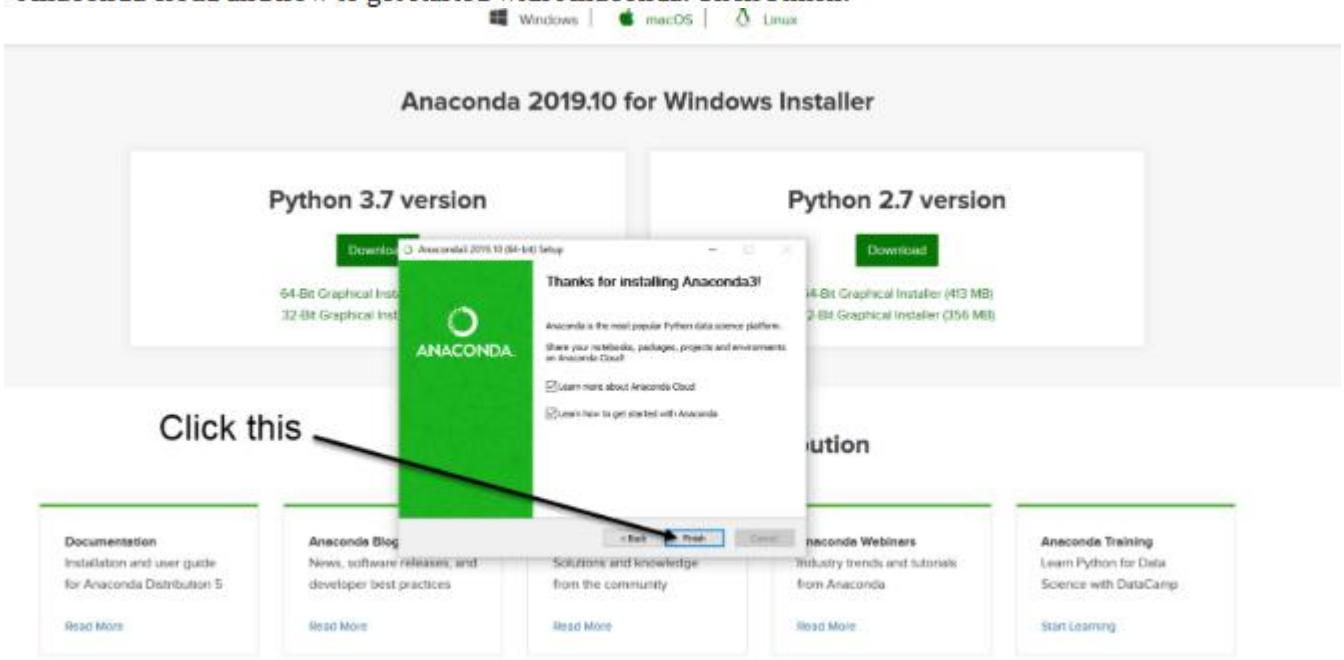
8. The installation is complete. Click Next.



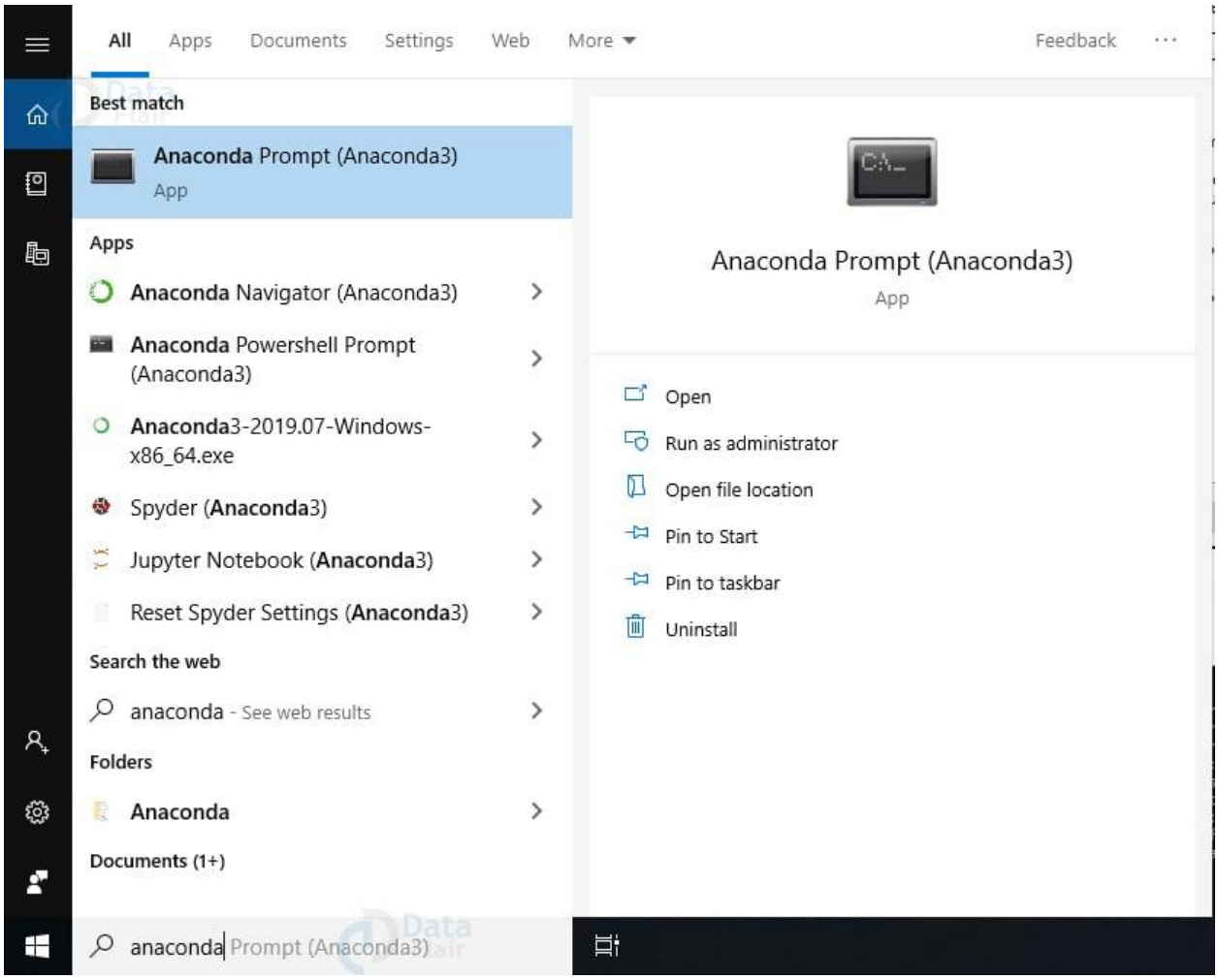
9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.



11. If you search for Anaconda now, you will see the following options:


All
Apps
Documents
Settings
Web
More
Feedback
Best match
Anaconda Prompt (Anaconda3)
App
Apps
Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Anaconda3)
Anaconda3-2019.07-Windows-x86_64.exe
Spyder (Anaconda3)
Jupyter Notebook (Anaconda3)
Reset Spyder Settings (Anaconda3)
Search the web
anaconda - See web results
Folders
Anaconda
Documents (1+)
anaconda Prompt (Anaconda3)
Anaconda Prompt (Anaconda3)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall

PYTHON LANGUAGE:

The dynamic semantics of Python make it an object-oriented language suitable for use at the interpreter level. Because of the binding, dynamic typing, and high-level built-in data structures, component glueing, and app building are made easy. Since Python has a simple and inexpensive syntax, it makes programs easier to administer. Because of its excellent modules and packages, Python greatly promotes code reuse and application modularity. Both the Python interpreter and the standard library are available for usage on all major platforms for free and are open source. Because it allows them to work faster, programmers adore Python. Because compilation is unnecessary, this round of testing, changing, and debugging is lightning quick. Since segmentation faults aren't caused by bugs or incorrect input, debugging Python programs is a snap. The interpreter will instead throw an exception if it finds a mistake. Stack traces are shown when an exception occurs in the program that the interpreter is unable to handle. It is possible to create breakpoints, inspect variables locally and globally, explore lines of code, and evaluate expressions using a source level debugger. The reflective characteristics of Python, the language used to build the debugger, are clearly visible. However, the fastest approach to debug the program is often to add print statements to the code. Using this straightforward method is helpful since it just requires a few rounds of editing, testing, and fixing problems.

Python has gained notoriety as an open-source, high-level, dynamic interpreted language. Both procedural and object-oriented programming are supported. Because of dynamic typing, Python does not need variables to be defined with a specific type. Take this scenario into account: x= 10. The possible values of x could range from anything between a String and an into to something entirely else.

Here are some aspects of Python:

Python has several features, some of which are listed below:

Easy to use

Python is often used by expert programmers. Python has a far more manageable learning curve compared to other programming languages such as C, C#, JavaScript, Java, etc. Python is an excellent option for programmers just starting out since the language's fundamentals can be picked up in a matter of days—if not hours—. The language is quite intuitive, which is great news for developers.

Thirdly, simple Instruction Python is very easy to learn, I promise. As said before, Python's syntax is really simple. Instead of semicolons or brackets, indentations define the code block.

4. Working with Object-Relational Data in Programming

The object-oriented programming paradigm is one of Python's strengths. Python makes it easy to work with object-oriented ideas like classes and encapsulation.

Help in Coding Graphical User Interfaces Modules such as PyQt5, PyQt4, wxPython, and Tk allow Python programmers to build user interfaces. Using PyQt5 is the way to go if you want to create graphical apps in Python.

6. The Use of Complex Languages

Python allows programmers to reach new levels of complexity. Python makes it unnecessary for programmers to commit the system architecture and memory management details to memory.

7. The potential for being prolonged

As an example of an extensible language, consider Python. Python applications may be built by translating them into C or C++.

8. Easily Fixes Bugs

Fantastic information for finding mistakes. You will become an expert at repairing software issues after you understand how to interpret Python's error traces. You can deduce its function from a cursory examination of the code.
9. Python is a very portable language.

Python is a very portable programming language. For example, if we would want to utilise Python code that was created for Windows on Linux, Unix, or Mac, we may easily do so by copying and pasting the necessary code into the right environment.

The Python programming language is an integrated one. 10.

Another reason Python may be deemed an integrated language is its ease of interaction with other languages like C, C++, etc.

11, Spoken Language with an Interpreter:

Python is considered an interpreted language due to the fact that its code is executed line by line. Debugging Python code is similar to that of C, C++, Java, etc., in that it does not need compilation. The source code of Python is converted into bytecode, a form that is immediate.

12. A Library with a High-Quality Curriculum

The extensive standard library that Python provides with a plethora of modules and functions eliminates the need to write code for every conceivable job. Regular expressions, web browsers, unit testing, and countless more are just a few of the numerous Python modules available.

13. A Typing Language with Dynamic Features

Python is a dynamically typed programming language. Since variables are automatically assigned their types (into, double, long, etc.) during runtime, there's no need to declare them beforehand.

A dozen. Construction of the front-and back-end

Python routines may be executed and created in a new project py script using simple HTML elements like land. By doing so, you will be able to construct front-end applications in Python in the same way as JavaScript. Because of Python's strength in backend development using frameworks like Flask and Jingo, it is often used for this task.

15. Allocating Memory Dynamically

When declaring a variable in Python, the data type is not necessary. Runtime memory is allocated whenever a variable is assigned a value. You can save developers the trouble of putting int y = 18 by setting y to 15. Just plug in y=18. Machine-Assisted Modelling and Learning: -

We will go into the details of a few approaches after defining machine learning and talking about its limits. Machine learning is not an area of artificial intelligence, despite popular belief to the

contrary. This is the original intent of ML, but when used to data science; it really shines when seen as a tool for building models of data.

Machine learning is based on the idea of creating mathematical models that can analyse data. We introduce the idea of "learning" into these models by allowing the user to tweak certain parameters, which in turn allows the computer to "learn" from the data. After being trained with historical data, these models may be used to assess and forecast future data. For kicks, I'll give you, the readers; the task of determining how much this "learning" based on mathematical models matches the "learning" shown by the human brain. It is critical to comprehend the problem context in machine learning in order to use these techniques effectively. For that reason, we will start by classifying the methods that will be covered into several general groups.

A number of subfields fall under the umbrella of machine learning, such as: The two main branches of machine learning are supervised and unsupervised learning.

In supervised learning, unlabelled data is labelled by simulating the relationship between a label and measurable data points. The labels used in classification problems are discrete categories, but the variables used in regression tasks are continuous. This leads to an even more split in duties. Both types of supervised learning will be discussed and shown in this section.

Unsupervised learning is sometimes referred to as "letting the dataset speak for itself." Modelling the properties of a label-less dataset is the goal of this technique. Among the many things that these models can do are clustering and dimensionality reduction. Data representations may be made simpler via the use of dimensionality reduction technologies. Data is grouped into several categories using clustering algorithms. What follows is a section with examples of supervised and unsupervised learning.

Machine Learning and Its Significance

Humans are the most intelligent and technologically advanced species on Earth because we are superior to all other animals in the ability to analyse, evaluate, and solve issues. At the same time, artificial intelligence has a ways to go before it can match human intelligence. Machine learning's need is being legitimately questioned at this moment. That we should "make decisions, based on data, with efficiency and scale" is the strongest case for further action.

Companies are pouring money into new AI, ML, and DL systems because they promise to address problems in the real world and provide data-driven insights. This is an instance of data-driven machine decision-making, exemplifying process automation. It is possible that data-based judgements may replace programming logic when dealing with problems that cannot be completely written. Complex, real-world problems need intelligent people to find solutions. Machine learning's importance is highlighted by this.

Challenges Regarding ML: - An

There is still a lot of room for improvement in Machine Learning, despite its remarkable success in areas such as cyber security and autonomous vehicles. One explanation for this is the enormous amount of issues that ML has faced and been unable to address. The following issues are currently plaguing ML:

There is no foolproof way to ensure that ML algorithms only work with top-notch data. When working with low-quality data, data preparation and feature extraction become more challenging.

Another issue with ML models is the time it takes to gather data, extract features, and retrieve them.

Due to the lack of experienced professionals in machine learning (ML), which is still in its early stages, acquiring expert resources might be challenging.

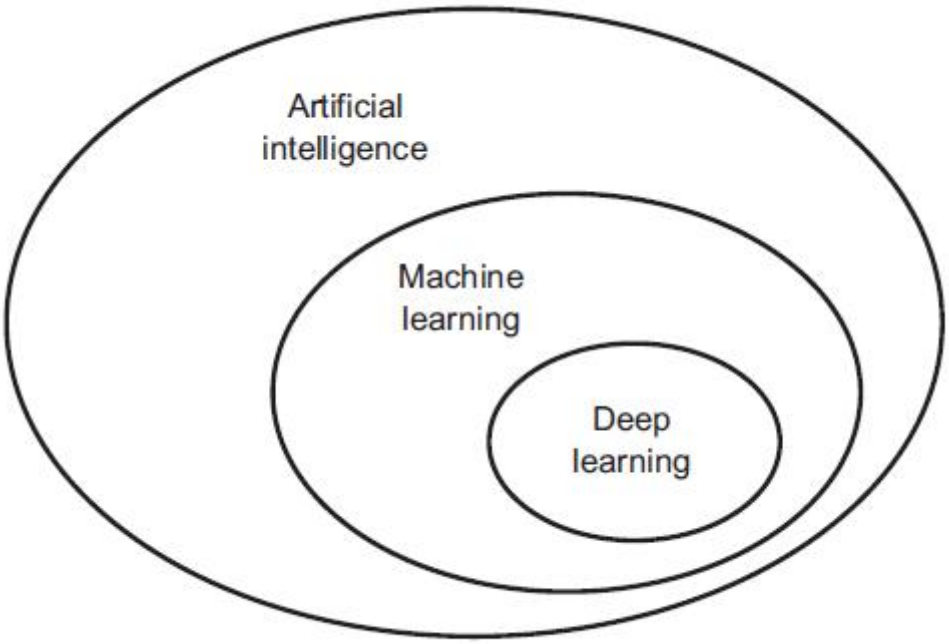
When it comes to identifying problems in businesses, ML also lacks a defined goal, which is a major drawback. The main reason for this is that ML has not yet fully matured and fulfilled its potential.

Over fitting or under fitting a model causes it to provide an inaccurate representation of the data.

Another issue that ML models face is the curse of dimensionality. It occurs when there are too many data points with properties. This may be a huge problem, and I feel the same way.

The ML model is sophisticated, which makes practical implementation tough. Deep learning, what is it?

Deep learning networks are a subset of machine learning that use three or more tiers of neural connections. Such neural networks "learn" from large datasets by making clumsy attempts to imitate the human brain's operations. Although neural networks may provide passable estimates with only one hidden layer, adding more layers has the potential to greatly improve accuracy.

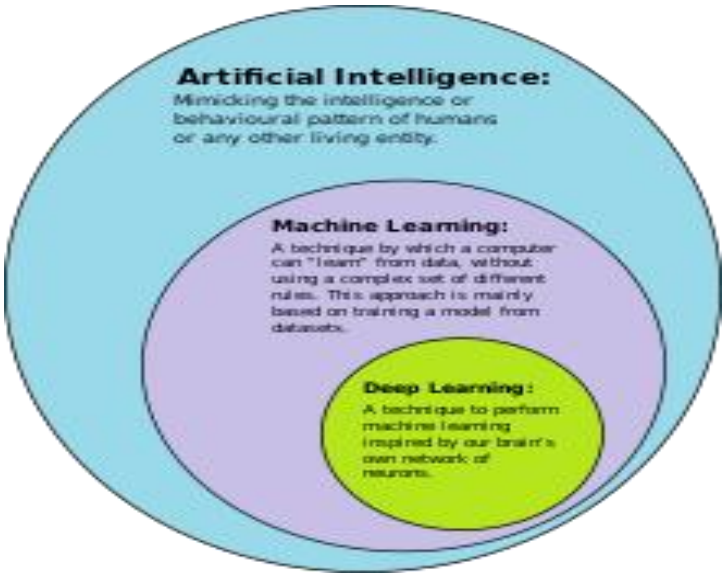


Using deep learning, a plethora of AI applications and services are able to automate hitherto human-only analytical and physical tasks. Any modern or ancient technology worth its salt needs deep learning technology. Digital assistants, voice-enabled TV remote controls, self-driving cars, and credit card fraud detection all depend on it.

An approach to deep learning

Deep learning neural networks, also called artificial neural networks, use data inputs, weights, and bias to simulate the brain's processes in an effort to mimic the human brain. Accurate data item identification, categorisation, and characterisation are made possible by these parts working together.

Predictions and classifications made by a deep neural network become more precise as more layers are added to it. The term "forward propagation" describes the process of passing computations over a network. In a deep neural network, the nodes that can be seen directly by the human eye are the input and output layers. While the input layer of a deep learning model processes incoming data, the output layer generates the model's ultimate prediction or classification.



As an additional method for training models, back propagation iteratively traverses the layers in an attempt to determine the biases and weights of the function, after which it employs methods like gradient descent to identify prediction mistakes. By using back propagation and forward propagation, neural networks can both generate predictions and fix their own mistakes. As time goes by, the algorithm improves its accuracy.

The simplest form of a deep neural network is described here in a very basic way. While deep learning techniques might be complex, there are many different types of neural networks that can handle different types of challenges and datasets. One such example is the widespread usage of convolutional neural networks (CNNs) in computer vision and image classification. These networks are capable of recognising objects and other visually distinct characteristics in images. Due to its usage of sequential or time series data, recurrent neural networks (RNNs) are often used in voice recognition and natural language processing. In 2015, a CNN became the first computer to surpass a person on an item identification task.

LIBRARIES/PACKGES:-

Tensor flow

The open-source Tensor Flow framework offers a dataflow and differentiable programming tool, which might be useful for many different types of applications. On top of that, it won't cost you a dime. Machine learning algorithms, such as neural networks, rely on symbolic mathematics. Both Google's research and manufacturing make use of it.

While developing Tensor Flow, the Google Brain team kept Google's core values in mind. It was made publicly available on November 9, 2015, using the Apache 2.0 license.

Oh, how innocent I was! Canine puppy

Numpy is an excellent tool to have on hand when dealing with arrays. The package includes a high-performance dimensional array object as well as utilities for manipulating arrays with many dimensions.

It is the core Python package for doing scientific computations. Among its notable features are a robust N-dimensional array object, facilities for integrating C/C++ and FORTRAN code, robust broadcasting capabilities, linear algebra expertise, random numbers, and the Fourier transform.

Numpy is a strong container for data in several dimensions, and its scientific applications are clear. Numpy is able to connect to several databases effortlessly because to its flexibility in defining data types.

Bearded dragons

Pandas is a free and open-source Python program that provides a comprehensive tool for dealing with and analysing data. The data structures it provides are robust. The primary functions of Python were data acquisition and pre-processing. In most cases, its usefulness for data analysis is restricted. Pandas were the ones that found it. No matter what kind of data you're working with, Pandas' five steps—loading, preparing, editing, modelling, and analysing—will always be the same. Numerous academic and professional fields make use of Python with Pandas. Statistics, economics, analytics, and finance are just a few examples of such areas.
For Matplotlib to function, it needs

Matplotlib is a 2D charting toolkit for Python that can generate publishable figures whether you're working with hardcopy forms or an interactive environment. In addition to Jupyter Notebook, four distinct GUI toolkits, and Python shell, the Matplotlib library is compatible with any version of Python. Matplotlib streamlines critical and straightforward tasks while making more complex ones easier. Graphs like bar charts, error charts, scatter plots, histograms, and power spectra may be easily generated with little code. Browse the galleries of sample graphs and thumbnails to see more examples.

Using the pilot module with IPython is ideal for basic charting jobs because to its MATLAB-like interface. Through an object-oriented interface or a method set known to MATLAB users, power users may alter any feature of the graph, including line styles, font quality, axis attributes, etc.

Use Scikit to learn more

With Scikit-learn, a consistent-interface Python package, you have access to a variety of supervised and unsupervised learning techniques. Academic and business groups find it appealing due to its interoperability with several versions of Linux and its liberal simplified BSD license.

SYSTEM TESTING

8. SYSTEM TESTING

The quality assurance (QA) group investigates the interplay between an application's different parts in system testing, which is also called system-level testing or system-integration testing. System testing is all about making sure software works the way it's supposed to. This step, which is analogous to black box testing, examines the program's internals. For instance, it is standard practice in system testing to ensure that every user input elicits the desired response of the program.

Here are the components of each system testing step:

This level of the test has a video lecture. In system testing, the main goal is to ensure that all of an application's many parts work together without a hitch. System testing is the next step in quality assurance after functional or user-story testing and integration testing of each component.

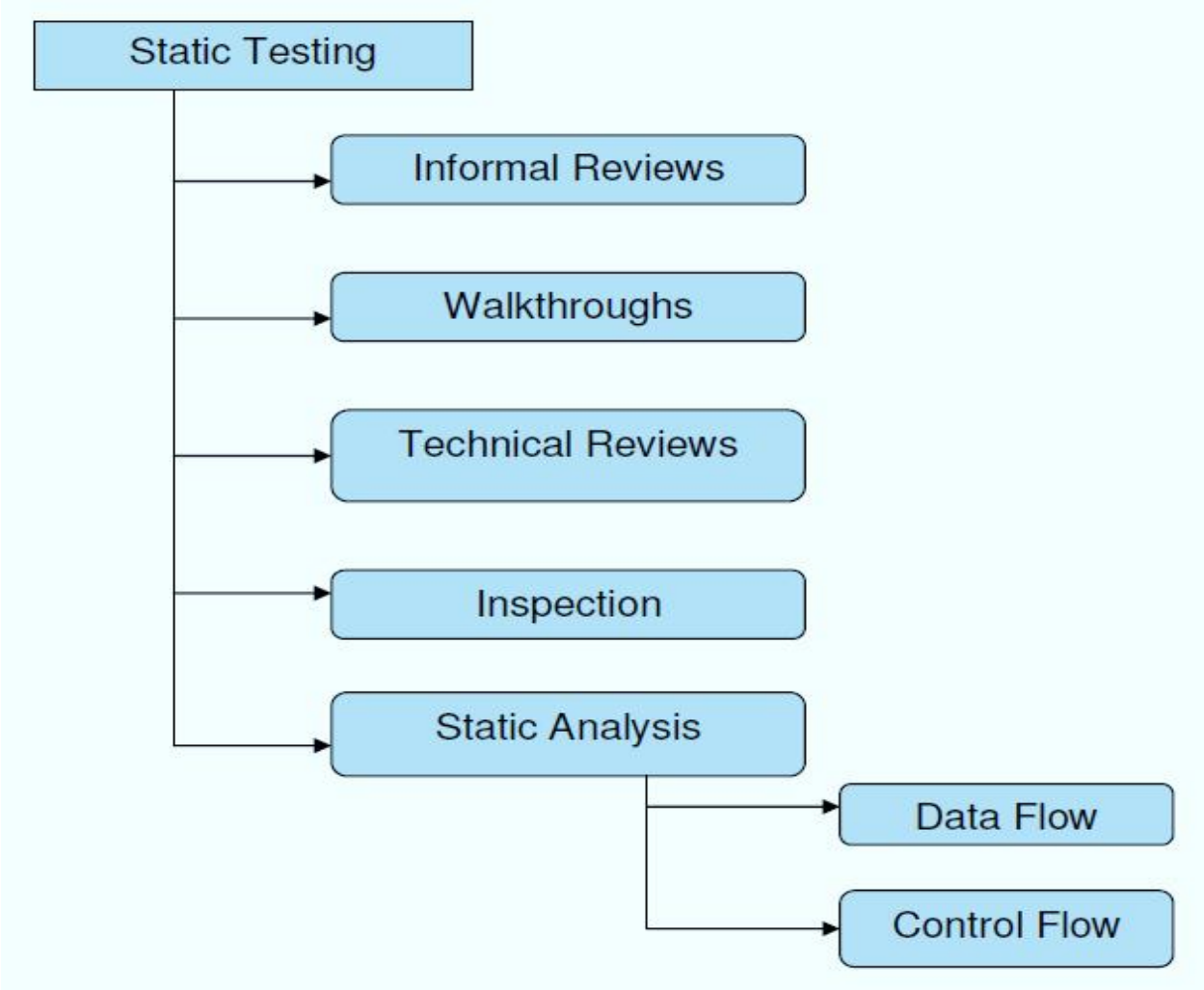
After system testing, the last stage before a software build is sent to production—the area where actual users will be using it—is acceptance testing. In addition to keeping track of all issues, the app-dev team decides what constitutes an acceptable quantity and kind of problems.

The optimal course of action is to optimise the approach in order to improve software engineering testing (Software Testing Procedures 8.1). The goals, timeline, and methodology for testing software to guarantee it is up to par are outlined in a software testing plan. The following software testing approaches often accomplish this end goal:

Ensure All Static Is Valid:

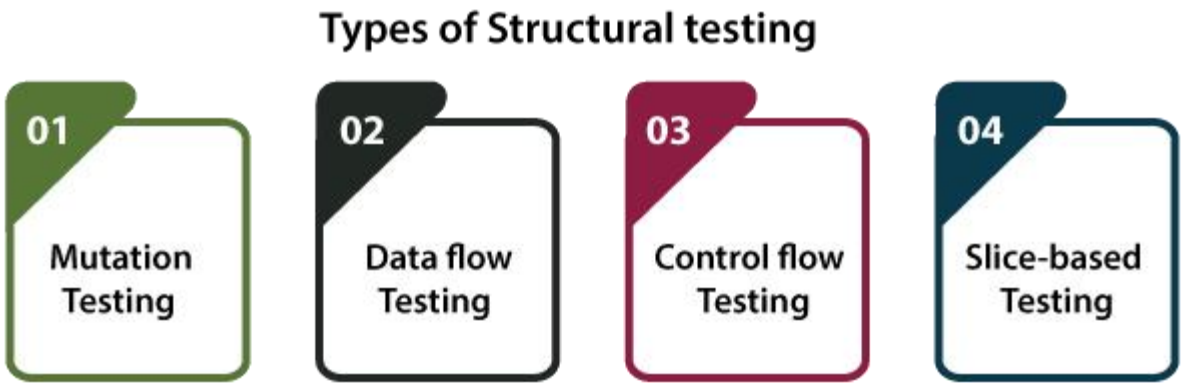
Static testing does not include launching the product in development at this stage. At its most basic level, desk-checking is what is needed to find errors and flaws in the code. This kind of

pre-deployment review is critical for preventing issues brought on by software design and code errors.



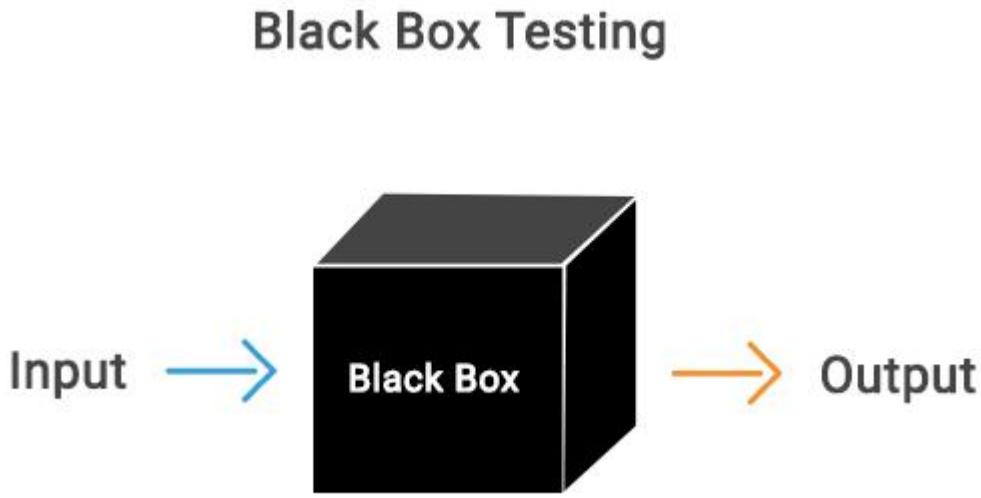
Structural Testing:

You can't evaluate the program thoroughly until you run it. In order to find and fix bugs and issues that crop up during the pre-production stage of software development, structural testing—also called white-box testing—is essential. This is the point where the program's structure decides whether regression testing is used for unit tests or not. Typically, this step of development is carried out automatically inside the test automation framework. By comparing test results from various versions, developers and QA engineers may monitor the system's behaviour. They are able to do this because they are privy to the software's internal workings and data flows throughout the testing phase.



Behavioural Testing:

Instead than focussing on the program's inner workings, this last phase of testing examines how it reacts to certain tasks. From the perspective of the end user, behavioural testing (also called black-box testing) involves subjecting the product to a battery of manual tests designed to reveal any issues that may arise. Quality assurance engineers, for example, often know a little bit about the business's or other software's intended use ('the black box') so they may react to problems and usability tests by acting like genuine users. Behavioural testers may use automation, particularly regression testing, to further ensure that no human mistake occurs across repeated tasks. An excellent method for evaluating a product's efficacy would be to automate the completion of one hundred online registration forms.



8.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User signin	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

SCREENS

9. SCREENSHOTS

SCREENS:

CONCLUSION

10. CONCLUSION

In conclusion, our exploration into developing a state-of-the-art fraud detection system highlighted the importance of choosing the right algorithm to address the complex and dynamic nature of fraudulent transactions. Through rigorous testing and evaluation of Random Forest, Gradient Boosting, and AdaBoost, we determined that Random Forest stands out as the most effective tool in our arsenal against fraud. Its exceptional performance on various metrics, including accuracy, precision, and its ability to mitigate overfitting, underscored its suitability for our needs. The process also underscored the critical role of data preprocessing and the thoughtful design of input and output components in enhancing model performance and usability. As we move forward, the adoption of the Random Forest algorithm in our Fraud Detection system represents a significant step towards achieving high levels of security and trust, essential in today's digital transaction environments. This project not only showcases the capabilities of machine learning in fraud detection but also sets the stage for future enhancements and adaptations as fraud techniques evolve.

BIBILOGRAPHY

11. REFERENCES

[1] Smith, J., & Johnson, K. (2022). "Enhancing E-commerce Security: A Multifaceted Approach to Fraud Detection." Journal Cybersecurity and E-commerce, 18(3), 135-150.

[2] Wang, L., & Chen, Y. (2021). "Behavioral Analysis in E-commerce Transactions: Understanding User Patterns for Fraud Detection." International Journal of Information Security, 27(4), 420-438.

[3] Patel, R., & Gupta, S. (2020)."Anomaly Detection in Multiparticipant Ecommerce Transactions." Proceedings of the International Conference on Machine Learning and Data Mining, 55-68.

[4] Kim, H., & Lee, M. (2019). "Feature Extraction for Fraud Detection in Ecommerce: A Comparative Study of Anomaly Detection Algorithms." Expert Systems with Applications, 129, 123-138.

[5] Chen, Z., & Zhang, Q. (2018). "Ensemble Methods in Fraud Detection: A Comprehensive Review." Journal of Computer Science and Technology, 33(6), 1123-1141.

[6] Li, X., & Wu, Q. (2017). "Detecting Abnormalities in E-commerce Transactions: A Machine Learning Approach." IEEE Transactions on Dependable and Secure Computing, 14(2), 201-215.