A Internship report on

“ **Make Skilled**“

Submitted in partial fulfillment of the requirements for the award of degree of B.Tech

By

M.Nandini(20KE1A4426)

Year: III semester : I

A.Y: 2022-23

<u>Department of CSE-DATASCIENCE</u>

MALINENI LAKSHMAIAH WOMEN’S ENGINEERING COLLEGE
(Approved by AICTE, Affiliated to JNTUK, *(An ISO9001:2008 Certified Institution)*
Pulladigunta (Village), Vatticherukuru (Mandal),
Guntur-522017, Andhra Pradesh, India

MAKE SKILLED
EARN THE SKILL WITH MAKING

INTERNSHIP REPORT SUBMITTED TO

MAKE SKILLED

MALINENI LAKSMAIAH WOMEN'S ENGINEERING COLLEGE

EDUCATIONAL SOCIETY'S GROUP OF INSITIUTION

PULLADIGUNTA, GUNTUR-522 017.

MOVIE RECOMMEDIATION SYSTEM USING MACHINE LEARING

TEAM NAME:OUT OF THE BOX

NAME: M. NANDINI

ROLL NO:20KE1A4426

COURSE: CSE-DATASCIENCE

Introduction to Python in AI

Python is a key part of AI programming languages due to the fact that it has good frameworks, such as SCIKIT-learn-Machine Learning in Python that meets almost all requirements as well as D3.

SCIKIT-learn is a Python module integrating a wide range supervised and unsupervised problems.

Python standard library

The Python Standard Library contains the exact syntax, semantics, and tokens of Python.

1. **Matplotlib:** This library is responsible for plotting numerical data. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

2. **Pandas:** Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools.

3. **Numpy:** The name “Numpy” stands for “Numerical Python”. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data.

4. **SciPy:** The name “SciPy” stands for “Scientific Python”. It is an open-source library.

5. Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

PYTHON LISTS:

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and **usage.**

PYTHON TUPLES:

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

DICITIONARY ITEMS:

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key: value pairs, and can be referred to by using the key name.

Exception	Description
Attribute error	Raised on the attribute assignment or reference fails.
Floating point error	Raised when a floating pint operation fails.
Index error	Raised when the index of a sequence is out of range.
Key error	Raised when a key is not in a dictionary.
Name error	Raised when a variable is not found in the local or global scope.
Syntax error	Raised by the parser when a syntax error is encountered.
Type error	Raised when a function or operation is applied to an object of an incorrect type.
Zero division error	Raised when the second operand of a division or module operation is zero
Indentation error	Raised when there is an incorrect indentation.

Python Install

Many PCs and Macs will have python already installed.

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

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

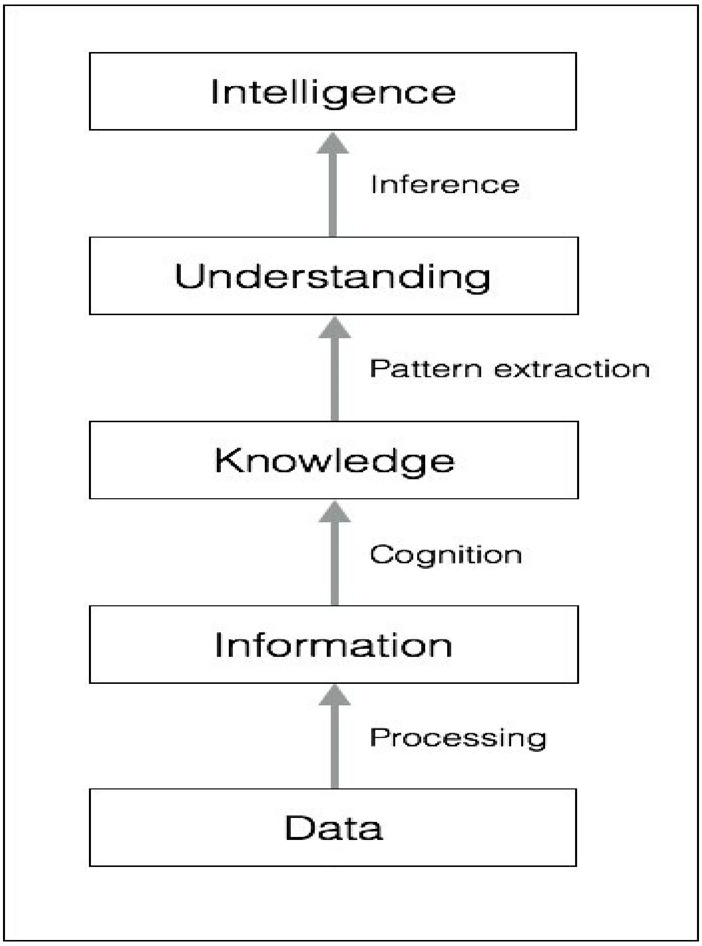
To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python –version
```

Artificial intelligence

1. Artificial intelligence is **the simulation of human intelligence processes by machines.**
2. Applications of AI include expert systems, natural language processing, speech recognition and machine vision.
3. AI is closely related to the study of human brain. Researchers believe that AI can be accomplished by understanding how the human brain works. By mimicking the way the human brain learns, thinks, and takes action, we can build a machine that can do the same. This can be used as a platform to develop intelligent systems that are capable of learning.
4. One of the main reasons we want to study AI is to automate many things. We live in a world where:

- We deal with huge and insurmountable amounts of data. The human brain can't keep track of so much data.
- Data originates from multiple sources simultaneously.
- The data is unorganized and chaotic.
- Knowledge derived from this data has to be updated constantly because the data itself keeps changing.
- The sensing and actuation has to happen in real time with high precision.

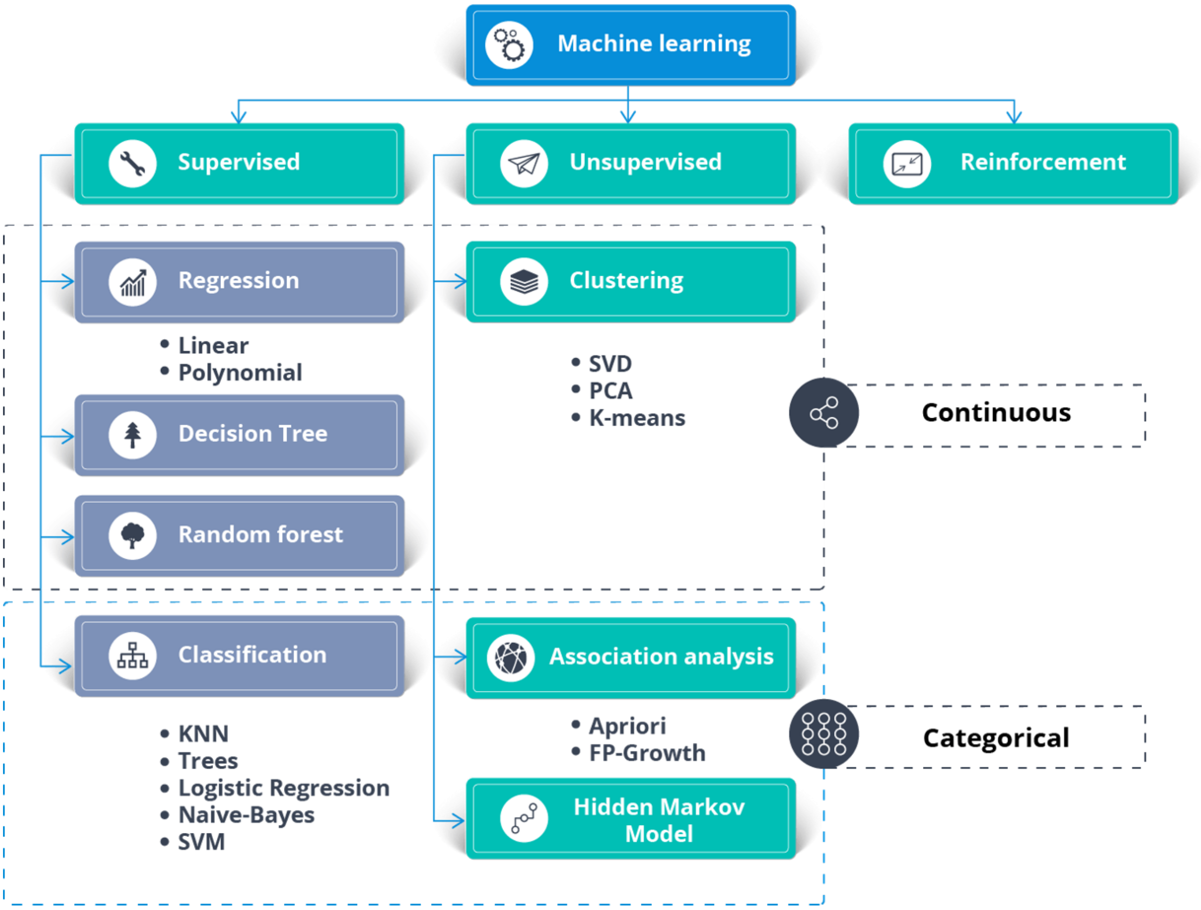


Machine Learning

Machine Learning enables a machine to automatically learn from data, improve performance from experiences and predict things without being explicitly programmed.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it.

The accuracy of the predicted output depends upon the amount of data, as the huge of amount of data helps to build a better model which predicts the output more accurately.



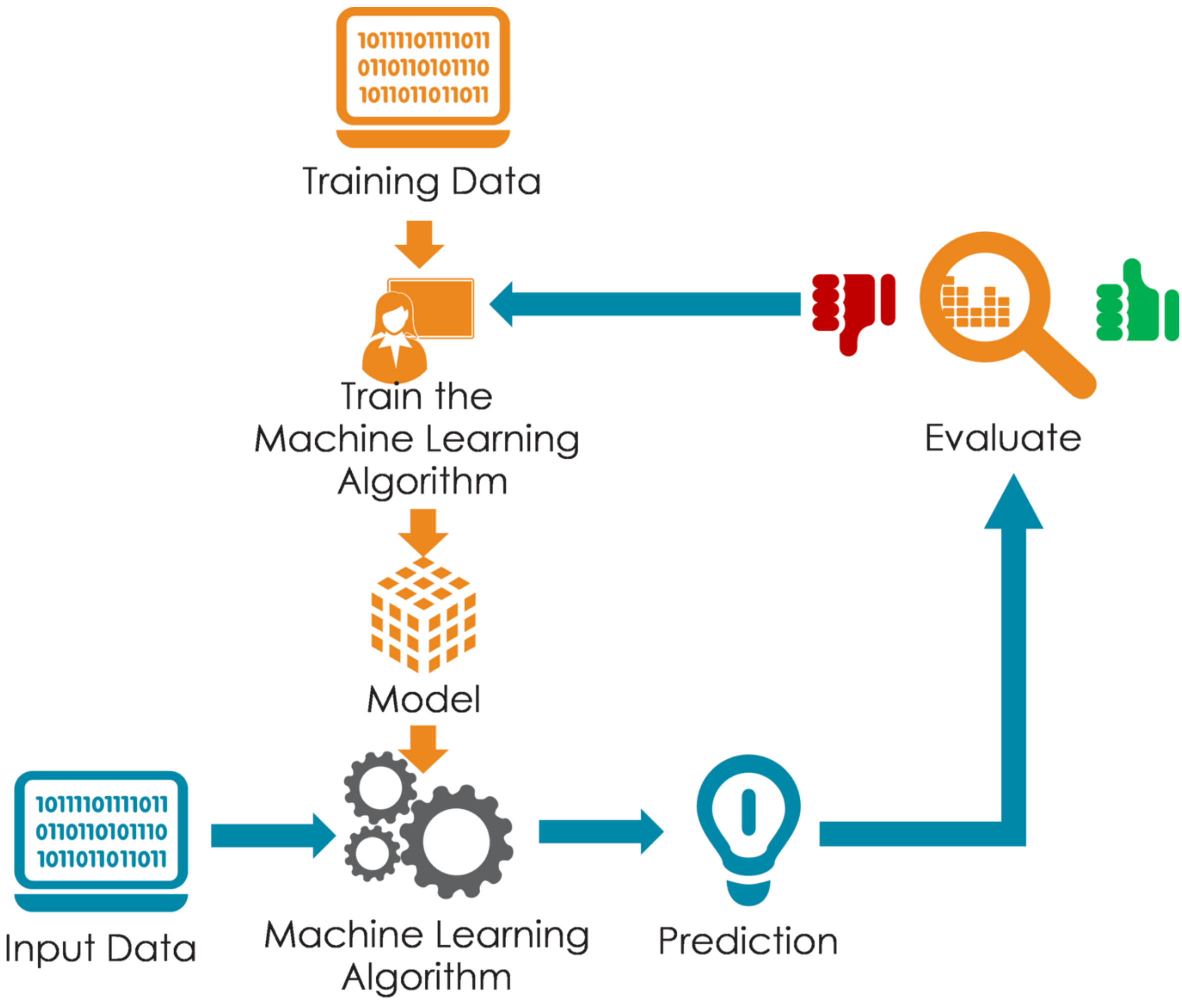
Features of Machine learning:

Machine Learning uses data to detect various patters in a given dataset

→ It can learn from past data and improve automatically.

→ It is a data-driven technology.

→ Machine Learning is much similar to data mining as it also deals with the huge amount of the data



Importance of Machine Learning:

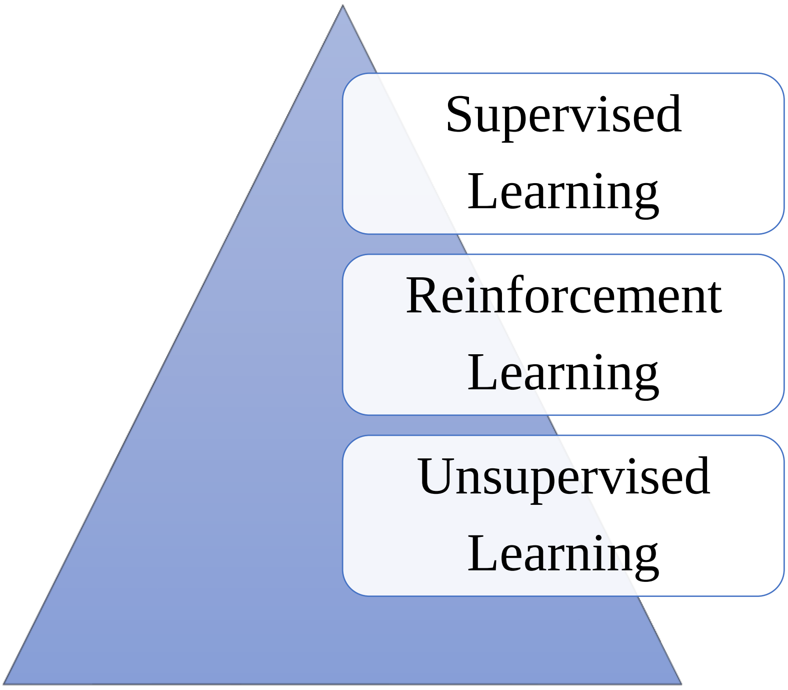
Rapid increment in the production of data

→ Solving complex problems, which are difficult for a human.

→ Decision making in various sector including finance.

→ Finding hidden patterns and extracting useful information from data.

Classification of Machine Learning



Supervised Learning:

data to the ML system in order to train it, and on that basis, it predicts the output. It is a type of machine learning method in which we provide labeled

- Classification
- Regression

Reinforcement Learning:

- It is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action.
- The agent learns automatically with these feedbacks and improves its performance.

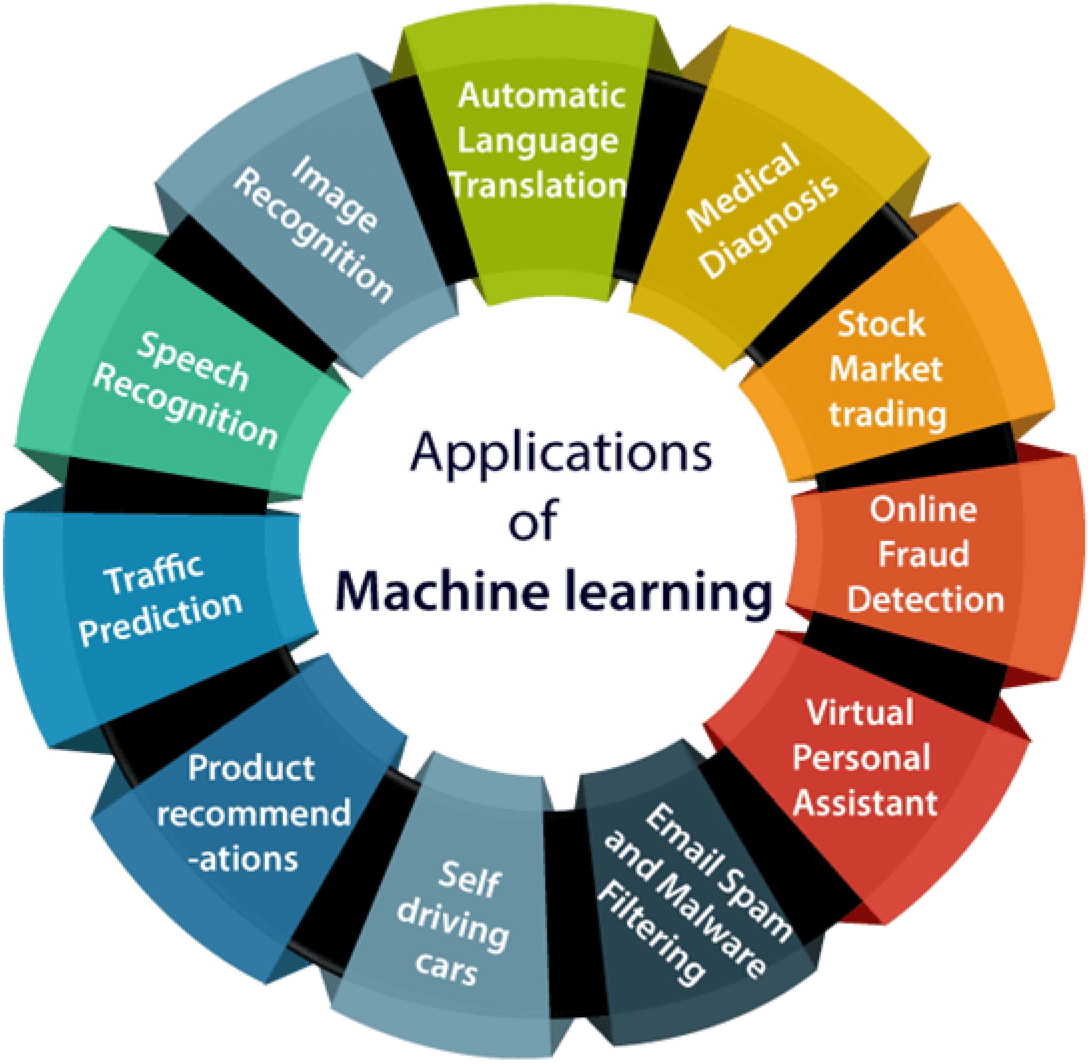
Unsupervised Learning:

- Unsupervised learning is a learning method in which a machine learns without any supervision.
- In unsupervised learning, we don't have a predetermined result.

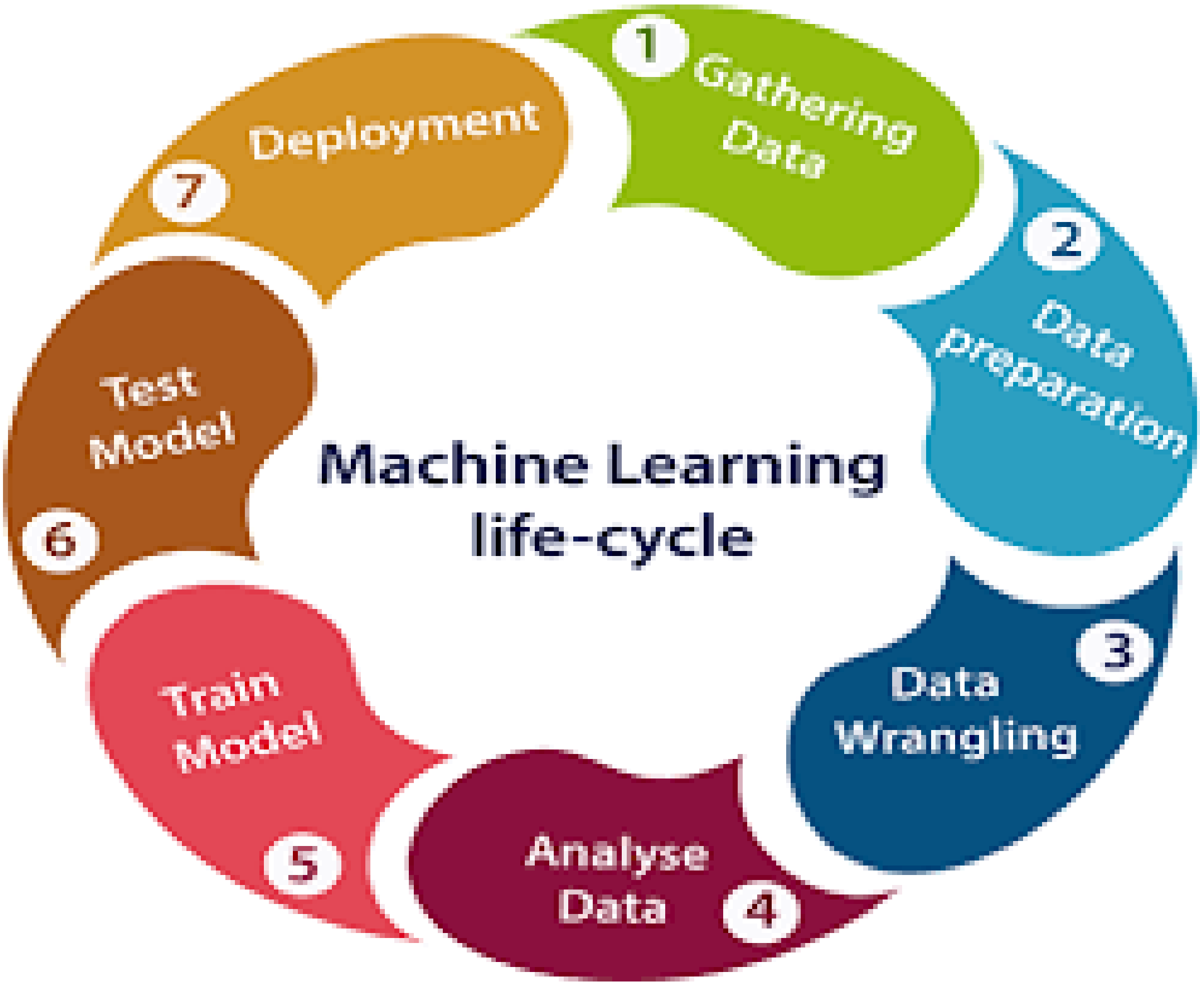
The machine tries to find useful insights from the huge amount of data.

- Clustering
- Association

Applications of Machine Learning



Machine Learning Lifecycle



Inputs and Outputs in Machine Learning:

→Inputs in ML are called as features variables or independent variables or input variables and are denoted with 'X→Outputs in ML are also called as target variable or predicted variables or dependent variables and are denoted with 'Y'.

Y=F(X).

What is Dataset?

A dataset is a collection of data in which data is arranged in some order.

	A	B
1	**Name**	**Height (cm)**
2	Harry the Horse	181
3	Dana the Deer	175
4	Fran the Fox	159
5	Bob the Buffalo	177
6	Gracie the Goat	165

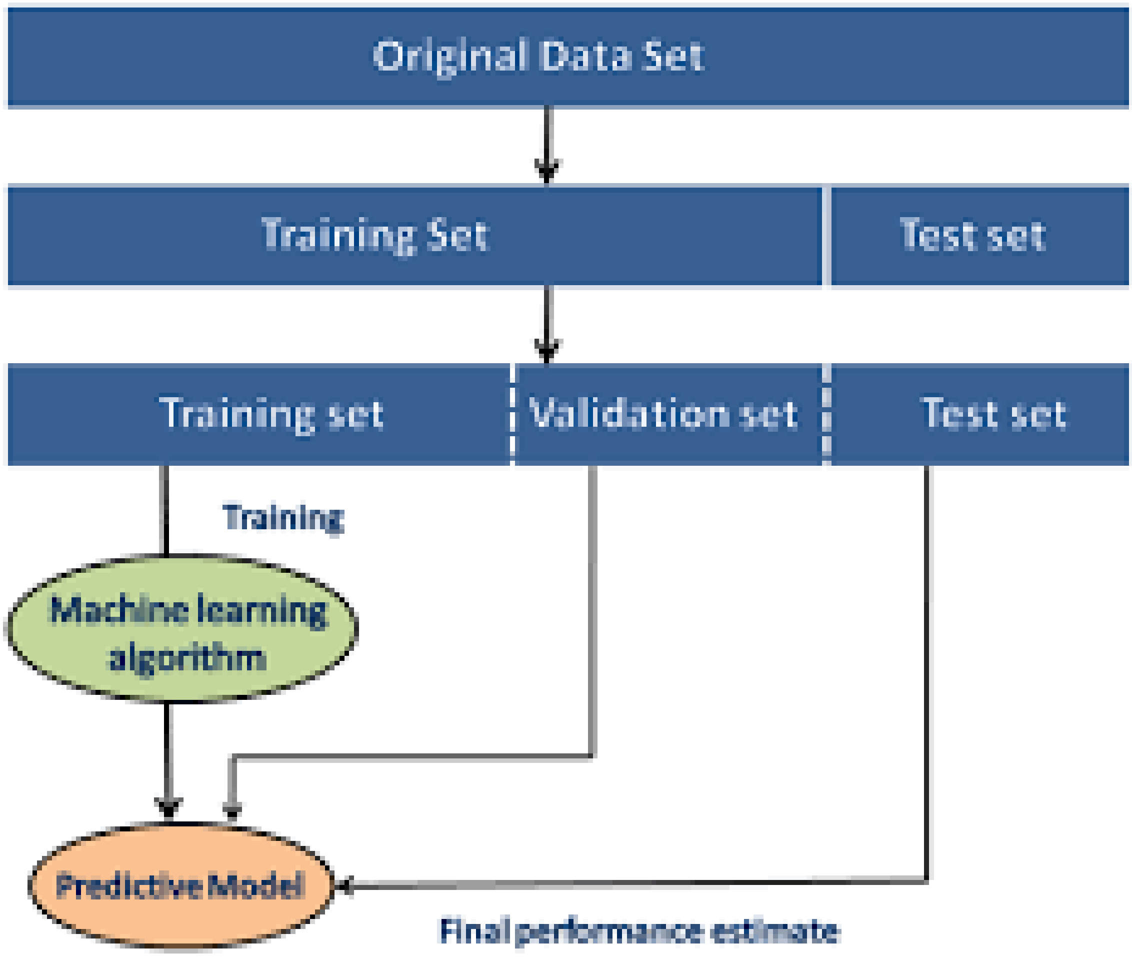
Need of Dataset

During the development of the ML project, the developers completely rely on the datasets.

- In building ML applications, datasets are divided into two parts
- Training dataset
- Test Dataset

Training dataset: A training dataset is **an initial dataset that teaches the ML models to identify desired patterns or perform a particular task**.

Testing dataset: The test dataset is **a subset of the training dataset that is utilized to give an objective evaluation of a final model**.



Data Gathering:

→It is the collection of data from different sources.

Eg: MQTT protocol. (Message Queue Telemetry Transport)

→MQTT is used in data communication.

Communication Protocols

There are mainly used in mobile applications to exchange information between people, between applications and also between application and cloud.

Most popular communication protocol is MQTT.

MQTT is created by IBM and they made it open source.

MQTT is managed by eclipse foundation.

MQTT follows PUBSUB model.

Sender is called publisher.

Receiver is called subscriber.

Server connect both publisher & subscriber which is called brokers.

Server should be always a cloud server.

A server will have IP address and port number.

Port number for MQTT protocol is 1883.

MQTT topic works like user id.

PIP: Python package installer. It is used to install packages in python.

Pip install package name

PROJECT-1

Create a Group Chat within your team using MQTT.

Objective: The of group chart is to create communication between one or more persons us MQTT.

Installation of MQTT:

pip install paho-mqtt

Publisher.py

```
import paho.mqtt.client as mqtt
broker='broker.hivemq.com'
port=1883
pub=mqtt.Client()
pub.connect(broker,port)
print('Broker Connected')
pub.publish('laya','hello')
```

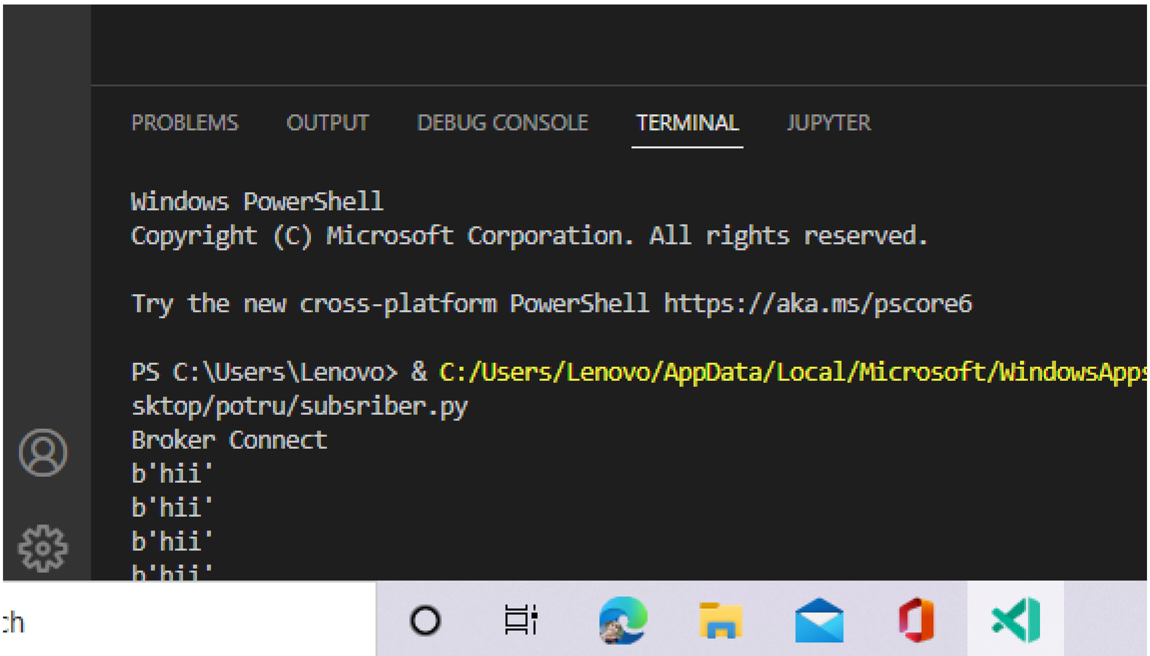
Subscriber.py

```
sub=mqtt.Client()
broker='broker.hivemq.com'
port=1883
sub.connect(broker,port)
print('Broker Connected')
sub.subscribe('laya')

def notification(sub,userdata,msg):
    print(msg.payload)

sub.on_message=notification
sub.loop_forever()
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\Lenovo> & C:/Users/Lenovo/AppData/Local/Microsoft/WindowsApps
sktop/potru/subsriber.py
Broker Connect
b'hii'
b'hii'
b'hii'

PROJECT-2

Publish virtual sensory feed to subscriber

Publisher.py

```
import paho.mqtt.client as mqtt
import random
import time
broker='broker.hivemq.com'
port=1883
pub=mqtt.Client()
pub.connect(broker,port)
print('Broker Connect')
while -5:
    humidity=random.randint(10,100)
    temp=random.randint(20,50)
    k='{"Humidity":'+str(humidity)+',"Temperature":'+str(temp)+'}'
    pub.publish('datapirates',k)
    print(k)
    time.sleep(4)
```

Subscriber.py

```
import paho.mqtt.client as mqtt
import random
import time
broker='broker.hivemq.com'
port=1883
sub=mqtt.Client()
sub.connect(broker,port)
print('Broker Connect')
sub.subscribe('laya')

def notification(sub,userdata,msg):
    print(msg.payload)

sub.on_message=notification
sub.loop_forever()
```

Output

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Lenovo\OneDrive\Desktop\virtual iot.L> & C:/Users/Lenovo/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/Lenovo/OneDrive/Desktop/virtual iot.L/publisher.py"

Broker Connect

{"Humidity":59,"Temperature":39}

{"Humidity":76,"Temperature":23}

{"Humidity":53,"Temperature":25}

{"Humidity":91,"Temperature":43}

{"Humidity":95,"Temperature":25}

{"Humidity":33,"Temperature":45}

{"Humidity":74,"Temperature":33}

{"Humidity":32,"Temperature":21}

{"Humidity":34,"Temperature":35}

{"Humidity":65,"Temperature":20}

{"Humidity":28,"Temperature":23}

{"Humidity":64,"Temperature":26}

{"Humidity":54,"Temperature":42}

{"Humidity":33,"Temperature":40}

{"Humidity":23,"Temperature":39}

{"Humidity":93,"Temperature":27}

{"Humidity":45,"Temperature":35}

{"Humidity":26,"Temperature":36}

{"Humidity":24,"Temperature":40}

{"Humidity":76,"Temperature":28}

{"Humidity":55,"Temperature":28}

{"Humidity":14,"Temperature":42}

{"Humidity":51,"Temperature":48}

{"Humidity":24,"Temperature":38}

{"Humidity":83,"Temperature":43}

{"Humidity":95,"Temperature":35}

{"Humidity":57,"Temperature":30}

{"Humidity":62,"Temperature":47}

{"Humidity":94,"Temperature":33}

{"Humidity":64,"Temperature":20}

{"Humidity":28,"Temperature":41}

{"Humidity":69,"Temperature":50}

PROJECT-3

Create a dataset for IoT Sensory Feed

Objective: The objective is to create csv file for data samples broadcasted by publisher from VS code on to google colaboratory subscriber.

Here, VS code is publisher.

Google colaboratory is subscriber.

Publisher.py on VS code

```
import paho.mqtt.client as mqtt
import random
import time
broker='broker.hivemq.com'
port=1883
pub=mqtt.Client()
pub.connect(broker,port)
print('Broker Connect')
while -5:
    humidity=random.randint(10,100)
    temp=random.randint(20,50)
    k='{"Humidity":'+str(humidity)+',"Temperature":'+str(temp)+'}'
    pub.publish('codemanica',k)
    print(k)
    time.sleep(4)
```

Output

Windows PowerShell

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Lenovo\OneDrive\Desktop\virtual iot.L> & C:/Users/Lenovo/AppData/Local/Microsoft/WindowsApps/python3.10.exe "c:/Users/Lenovo/OneDrive/Desktop/virtual iot.L/publisher.py"

Broker Connect

{"Humidity":56,"Temperature":40}

{"Humidity":17,"Temperature":35}

{"Humidity":90,"Temperature":23}

{"Humidity":92,"Temperature":49}

{"Humidity":21,"Temperature":22}

{"Humidity":87,"Temperature":34}

{"Humidity":21,"Temperature":34}

{"Humidity":14,"Temperature":42}

{"Humidity":25,"Temperature":37}

{"Humidity":28,"Temperature":33}

{"Humidity":58,"Temperature":38}

{"Humidity":77,"Temperature":46}

{"Humidity":48,"Temperature":48}

{"Humidity":53,"Temperature":30}

{"Humidity":16,"Temperature":41}

{"Humidity":86,"Temperature":39}

{"Humidity":47,"Temperature":20}

{"Humidity":32,"Temperature":39}

{"Humidity":62,"Temperature":47}

{"Humidity":96,"Temperature":37}

{"Humidity":23,"Temperature":37}

{"Humidity":74,"Temperature":21}

{"Humidity":76,"Temperature":39}

{"Humidity":12,"Temperature":24}

{"Humidity":59,"Temperature":29}

{"Humidity":73,"Temperature":47}

Subscriber.py in Googlecolab

```
pip install paho-mqtt
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting paho-mqtt
  Downloading paho-mqtt-1.6.1.tar.gz (99 kB)
     |████████████████████████████████| 99 kB 4.2 MB/s
Building wheels for collected packages: paho-mqtt
  Building wheel for paho-mqtt (setup.py) ... done
  Created wheel for paho-mqtt: filename=paho_mqtt-1.6.1-py3-none-any.whl size=62133 sha256=8c5652b27126097aae0ce40a78e98e88b48570760743bd61c7ac83611a668194
  Stored in directory: /root/.cache/pip/wheels/d0/bf/ac/2b3f43f8c6fcd0f4ba5395397458c521eb0b52d33b574a5a40
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.6.1
```

```python
import paho.mqtt.client as mqtt
import json
import pandas as pd

sub=mqtt.Client()
sub.connect('broker.hivemq.com',1883)
print('Broker Conntected')
sub.subscribe('codemanica')
dataset=[]
i=0

def notification(sub,userdata,msg):
  global i
  data=(msg.payload).decode('utf-8')
  data=json.loads(data)
  h=data['Humidity']
  t=data['Temperature']
  dummy=[]
  dummy.append(h)
  dummy.append(t)
  dataset.append(dummy)
  print(dataset)
  i+=1
  if i==10:
    df=pd.DataFrame(dataset)
    df.to_csv('iot.csv')
    i=0

sub.on_message=notification
sub.loop_forever()
```

<u>Output</u>

```
Broker Conntected
[[90, 23]]
[[90, 23], [92, 49]]
[[90, 23], [92, 49], [21, 22]]
[[90, 23], [92, 49], [21, 22], [87, 34]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47], [96, 37]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47], [96, 37], [23, 37]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47], [96, 37], [23, 37], [74, 21]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47], [96, 37], [23, 37], [74, 21], [76, 39]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47], [96, 37], [23, 37], [74, 21], [76, 39],
[12, 24]]
[[90, 23], [92, 49], [21, 22], [87, 34], [21, 34], [14, 42], [25, 37],
[28, 33], [58, 38], [77, 46], [48, 48], [53, 30], [16, 41], [86, 39],
[47, 20], [32, 39], [62, 47], [96, 37], [23, 37], [74, 21], [76, 39],
[12, 24], [59, 29]]
```

PROJECT-4

Apply Data Wrangling on Data (CSV)

Step-1: Import data into google colab.

```
import pandas as pd
data=pd.read_csv('Data.csv')
data
```

Output:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
X=df.iloc[:,:-1]
X
```

	Country	Age	Salary
0	France	44.0	72000.0
1	Spain	27.0	48000.0
2	Germany	30.0	54000.0
3	Spain	38.0	61000.0
4	Germany	40.0	NaN

	Country	Age	Salary
5	France	35.0	58000.0
6	Spain	NaN	52000.0
7	France	48.0	79000.0
8	Germany	50.0	83000.0
9	France	37.0	67000.0

```
Y=df.iloc[:,-1]
Y
0 No
1 Yes
 2 No
 3 No
 4 Yes
 5 Yes
 6 No
 7 Yes
 8 No
 9 Yes
 Name: Purchased, dtype: object
X=X.values
X
```

```
array([['France', 44.0, 72000.0],
    ['Spain', 27.0, 48000.0],
    ['Germany', 30.0, 54000.0],
    ['Spain', 38.0, 61000.0],
    ['Germany', 40.0, nan],
    ['France', 35.0, 58000.0],
    ['Spain', nan, 52000.0],
    ['France', 48.0, 79000.0],
    ['Germany', 50.0, 83000.0],
    ['France', 37.0, 67000.0]], dtype=object)
```

```
Y=Y.values
Y
array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],
dtype=object)
from sklearn.impute import SimpleImputer
import numpy as np

si=SimpleImputer(missing_values=np.nan,strategy='mean')
X[:,1:3]=si.fit_transform(X[:,1:3])
x
```

```
array([['France', 44.0, 72000.0],
    ['Spain', 27.0, 48000.0],
    ['Germany', 30.0, 54000.0],
    ['Spain', 38.0, 61000.0],
    ['Germany', 40.0, 63777.77777777778],
    ['France', 35.0, 58000.0],
```

```
['Spain', 38.77777777777778, 52000.0],
['France', 48.0, 79000.0],
['Germany', 50.0, 83000.0],
['France', 37.0, 67000.0]], dtype=object)
```

Encoding: Encoding is a process of converting categorical column into a numerical column is called encoding.

OneHot Encoding: It is a process of representing categorical values in binary states. It is implied as inputs.

Label encoding: It is a process of assigning 0,1,2,3,… based on the number of labels. It is implied as Outputs.

Feature scaling: Feature scaling is the process of converting all the columns into standard scaler.

[ ]

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])],remainder='passthrough')
X=ct.fit_transform(X)
```

[ ]

```
X
```

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [0.0, 1.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

[ ]

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
Y=le.fit_transform(Y)
```

[ ]

```
Y
```

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1])
```

[ ]

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X=ss.fit_transform(X)
```

[ ]

```
X
```

```
array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

[ ]

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
```

[ ]

```
X_train.shape
```

(8, 5)

[ ]

```
X_test.shape
```

(2, 5)

[ ]

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

[ ]

```
ct=ColumnTransformer(transformers=[('enconder',OneHotEncoder(),[0])],remainder='passthrough')
X=ct.fit_transform(X)
X
```

```
array([[ 0.00000000e+00,  1.00000000e+00, -6.54653671e-01,
        -6.54653671e-01,  7.58874362e-01,  7.49473254e-01],
       [ 1.00000000e+00,  0.00000000e+00, -6.54653671e-01,
         1.52752523e+00, -1.71150388e+00, -1.43817841e+00],
       [ 1.00000000e+00,  0.00000000e+00,  1.52752523e+00,
        -6.54653671e-01, -1.27555478e+00, -8.91265492e-01],
       [ 1.00000000e+00,  0.00000000e+00, -6.54653671e-01,
         1.52752523e+00, -1.13023841e-01, -2.53200424e-01],
       [ 1.00000000e+00,  0.00000000e+00,  1.52752523e+00,
        -6.54653671e-01,  1.77608893e-01,  6.63219199e-16],
       [ 0.00000000e+00,  1.00000000e+00, -6.54653671e-01,
```

```
       -6.54653671e-01, -5.48972942e-01, -5.26656882e-01],
      [ 1.00000000e+00,  0.00000000e+00, -6.54653671e-01,
        1.52752523e+00,  0.00000000e+00, -1.07356980e+00],
      [ 0.00000000e+00,  1.00000000e+00, -6.54653671e-01,
       -6.54653671e-01,  1.34013983e+00,  1.38753832e+00],
      [ 1.00000000e+00,  0.00000000e+00,  1.52752523e+00,
       -6.54653671e-01,  1.63077256e+00,  1.75214693e+00],
      [ 0.00000000e+00,  1.00000000e+00, -6.54653671e-01,
       -6.54653671e-01, -2.58340208e-01,  2.93712492e-01]])
```

[ ]

```
from sklearn.preprocessing import  LabelEncoder
le=LabelEncoder()
Y=le.fit_transform(Y)
Y
```

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1])
```

[ ]

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

[ ]

```
X=ss.fit_transform(X)
X
```

```
array([[-1.22474487e+00,  1.22474487e+00, -6.54653671e-01,
       -6.54653671e-01,  7.58874362e-01,  7.49473254e-01],
      [ 8.16496581e-01, -8.16496581e-01, -6.54653671e-01,
        1.52752523e+00, -1.71150388e+00, -1.43817841e+00],
      [ 8.16496581e-01, -8.16496581e-01,  1.52752523e+00,
       -6.54653671e-01, -1.27555478e+00, -8.91265492e-01],
      [ 8.16496581e-01, -8.16496581e-01, -6.54653671e-01,
        1.52752523e+00, -1.13023841e-01, -2.53200424e-01],
      [ 8.16496581e-01, -8.16496581e-01,  1.52752523e+00,
       -6.54653671e-01,  1.77608893e-01,  2.35783334e-16],
      [-1.22474487e+00,  1.22474487e+00, -6.54653671e-01,
       -6.54653671e-01, -5.48972942e-01, -5.26656882e-01],
      [ 8.16496581e-01, -8.16496581e-01, -6.54653671e-01,
        1.52752523e+00,  8.88178420e-17, -1.07356980e+00],
      [-1.22474487e+00,  1.22474487e+00, -6.54653671e-01,
       -6.54653671e-01,  1.34013983e+00,  1.38753832e+00],
      [ 8.16496581e-01, -8.16496581e-01,  1.52752523e+00,
       -6.54653671e-01,  1.63077256e+00,  1.75214693e+00],
      [-1.22474487e+00,  1.22474487e+00, -6.54653671e-01,
       -6.54653671e-01, -2.58340208e-01,  2.93712492e-01]])
```

[ ]

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.5)
X_test.shape
```

```
(5, 6)
```

[ ]

```
X_train.shape
```

```
(5, 6)
```

[ ]

Y_test.shape

(5,)

[ ]

```
Y_train.shape
```

```
(5,)
```

PROJECT-5

Salary Prediction System

Objective: The objective is to predict predict salary based on years of experience.

```
import pandas as pd
dataset=pd.read_csv('Salary_Data.csv')
dataset
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0

	YearsExperience	Salary
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

```
X=dataset.iloc[:,0].values
X
array([ 1.1, 1.3, 1.5, 2. , 2.2, 2.9, 3. , 3.2, 3.2, 3.7, 3.9, 4. , 4. ,
4.1, 4.5, 4.9, 5.1, 5.3, 5.9, 6. , 6.8, 7.1, 7.9, 8.2, 8.7, 9. , 9.5,
9.6, 10.3, 10.5])
X=X.reshape(-1,1)
X
array([[ 1.1],
    [ 1.3],
    [ 1.5],
    [ 2. ],
    [ 2.2],
    [ 2.9],
    [ 3. ],
    [ 3.2],
    [ 3.2],
    [ 3.7],
    [ 3.9],
    [ 4. ],
    [ 4. ],
    [ 4.1],
    [ 4.5],
    [ 4.9],
    [ 5.1],
    [ 5.3],
    [ 5.9],
    [ 6. ],
    [ 6.8],
    [ 7.1],
    [ 7.9],
    [ 8.2],
    [ 8.7],
    [ 9. ],
    [ 9.5],
    [ 9.6],
    [10.3],
    [10.5]])
```

[ ]

```
Y=dataset.iloc[:,1].values
Y
```

```
array([ 39343.,  46205.,  37731.,  43525.,  39891.,  56642.,  60150.,
        54445.,  64445.,  57189.,  63218.,  55794.,  56957.,  57081.,
        61111.,  67938.,  66029.,  83088.,  81363.,  93940.,  91738.,
        98273., 101302., 113812., 109431., 105582., 116969., 112635.,
       122391., 121872.])
```

[ ]

```
import matplotlib.pyplot as plt
plt.scatter(X,Y)
```

[ ]

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.5)
regressor.fit(X_train,Y_train)
```

```
LinearRegression()
```

[ ]

```
regressor.coef_
```

```
array([9895.34870667])
```

[ ]

```
regressor.intercept_
```

```
22059.5077225769
```

[ ]

```
Y_pred=regressor.predict(X_test)
Y_pred
```

```
array([ 66588.57690258,  51745.55384258, 116065.32043591,  41850.20513591,
        34923.46104124, 111117.64608258,  50756.01897191,  58672.29793724,
        92316.48353991, 123981.59940125,  62630.43741991,  60651.36767858,
        53724.62358391,  53724.62358391,  81431.59996258])
```

[ ]

```
from sklearn.metrics import r2_score
print(r2_score(Y_pred,Y_test))
```

```
0.9406296210997459
```

[ ]

```
print(regressor.predict([[10.5]]))
```

PROJECT-6
Profit Prediction System

```
import pandas as pd
dataset=pd.read_csv('50_Startups data.csv')
dataset
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25

	R&D Spend	Administration	Marketing Spend	State	Profit
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56
32	63408.86	129219.61	46085.25	California	97427.84
33	55493.95	103057.49	214634.81	Florida	96778.92
34	46426.07	157693.92	210797.67	California	96712.80
35	46014.02	85047.44	205517.64	New York	96479.51
36	28663.76	127056.21	201126.82	Florida	90708.19
37	44069.95	51283.14	197029.42	California	89949.14
38	20229.59	65947.93	185265.10	New York	81229.06
39	38558.51	82982.09	174999.30	California	81005.76
40	28754.33	118546.05	172795.67	California	78239.91
41	27892.92	84710.77	164470.71	Florida	77798.83
42	23640.93	96189.63	148001.11	California	71498.49
43	15505.73	127382.30	35534.17	New York	69758.98
44	22177.74	154806.14	28334.72	California	65200.33
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

```
X=dataset.iloc[:,0:4]
```

X

R&D Spend	Administration	Marketing Spend	State	
0	165349.20	136897.80	471784.10	New York
1	162597.70	151377.59	443898.53	California
2	153441.51	101145.55	407934.54	Florida
3	144372.41	118671.85	383199.62	New York
4	142107.34	91391.77	366168.42	Florida
5	131876.90	99814.71	362861.36	New York
6	134615.46	147198.87	127716.82	California
7	130298.13	145530.06	323876.68	Florida
8	120542.52	148718.95	311613.29	New York
9	123334.88	108679.17	304981.62	California
10	101913.08	110594.11	229160.95	Florida
11	100671.96	91790.61	249744.55	California
12	93863.75	127320.38	249839.44	Florida
13	91992.39	135495.07	252664.93	California
14	119943.24	156547.42	256512.92	Florida
15	114523.61	122616.84	261776.23	New York
16	78013.11	121597.55	264346.06	California
17	94657.16	145077.58	282574.31	New York
18	91749.16	114175.79	294919.57	Florida
19	86419.70	153514.11	0.00	New York
20	76253.86	113867.30	298664.47	California
21	78389.47	153773.43	299737.29	New York
22	73994.56	122782.75	303319.26	Florida
23	67532.53	105751.03	304768.73	Florida
24	77044.01	99281.34	140574.81	New York

R&D Spend	Administration	Marketing Spend	State	
25	64664.71	139553.16	137962.62	California
26	75328.87	144135.98	134050.07	Florida
27	72107.60	127864.55	353183.81	New York
28	66051.52	182645.56	118148.20	Florida
29	65605.48	153032.06	107138.38	New York
30	61994.48	115641.28	91131.24	Florida
31	61136.38	152701.92	88218.23	New York
32	63408.86	129219.61	46085.25	California

```
y=dataset.iloc[:,-1]
y
```

```
0    192261.83
1    191792.06
2    191050.39
3    182901.99
4    166187.94
5    156991.12
6    156122.51
7    155752.60
8    152211.77
9    149759.96
10   146121.95
11   144259.40
12   141585.52
13   134307.35
14   132602.65
15   129917.04
16   126992.93
17   125370.37
18   124266.90
19   122776.86
20   118474.03
21   111313.02
22   110352.25
23   108733.99
24   108552.04
25   107404.34
26   105733.54
27   105008.31
28   103282.38
29   101004.64
30    99937.59
31    97483.56
32    97427.84
33    96778.92
34    96712.80
35    96479.51
36    90708.19
```

```
37    89949.14
38    81229.06
39    81005.76
40    78239.91
41    77798.83
42    71498.49
43    69758.98
44    65200.33
45    64926.08
46    49490.75
47    42559.73
48    35673.41
49    14681.40
Name: Profit, dtype: float64
```

[ ]

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[3])],remainder='passthrough')
x=ct.fit_transform(X)
x
```

```
array([[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.6534920e+05,
        1.3689780e+05, 4.7178410e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.6259770e+05,
        1.5137759e+05, 4.4389853e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.5344151e+05,
        1.0114555e+05, 4.0793454e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.4437241e+05,
        1.1867185e+05, 3.8319962e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
        9.1391770e+04, 3.6616842e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.3187690e+05,
        9.9814710e+04, 3.6286136e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.3461546e+05,
        1.4719887e+05, 1.2771682e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.3029813e+05,
        1.4553006e+05, 3.2387668e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.2054252e+05,
        1.4871895e+05, 3.1161329e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.2333488e+05,
        1.0867917e+05, 3.0498162e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.0191308e+05,
        1.1059411e+05, 2.2916095e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0067196e+05,
        9.1790610e+04, 2.4974455e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.3863750e+04,
        1.2732038e+05, 2.4983944e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 9.1992390e+04,
        1.3549507e+05, 2.5266493e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.1994324e+05,
        1.5654742e+05, 2.5651292e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.1452361e+05,
        1.2261684e+05, 2.6177623e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.8013110e+04,
        1.2159755e+05, 2.6434606e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 9.4657160e+04,
```

1.4507758e+05, 2.8257431e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.1749160e+04,
1.1417579e+05, 2.9491957e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 8.6419700e+04,
1.5351411e+05, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.6253860e+04,
1.1386730e+05, 2.9866447e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.8389470e+04,
1.5377343e+05, 2.9973729e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.3994560e+04,
1.2278275e+05, 3.0331926e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.7532530e+04,
1.0575103e+05, 3.0476873e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.7044010e+04,
9.9281340e+04, 1.4057481e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 6.4664710e+04,
1.3955316e+05, 1.3796262e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.5328870e+04,
1.4413598e+05, 1.3405007e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.2107600e+04,
1.2786455e+05, 3.5318381e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.6051520e+04,
1.8264556e+05, 1.1814820e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 6.5605480e+04,
1.5303206e+05, 1.0713838e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.1994480e+04,
1.1564128e+05, 9.1131240e+04],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 6.1136380e+04,
1.5270192e+05, 8.8218230e+04],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 6.3408860e+04,
1.2921961e+05, 4.6085250e+04],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 5.5493950e+04,
1.0305749e+05, 2.1463481e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 4.6426070e+04,
1.5769392e+05, 2.1079767e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 4.6014020e+04,
8.5047440e+04, 2.0551764e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 2.8663760e+04,
1.2705621e+05, 2.0112682e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 4.4069950e+04,
5.1283140e+04, 1.9702942e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 2.0229590e+04,
6.5947930e+04, 1.8526510e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 3.8558510e+04,
8.2982090e+04, 1.7499930e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.8754330e+04,
1.1854605e+05, 1.7279567e+05],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 2.7892920e+04,
8.4710770e+04, 1.6447071e+05],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.3640930e+04,
9.6189630e+04, 1.4800111e+05],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.5505730e+04,
1.2738230e+05, 3.5534170e+04],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 2.2177740e+04,
1.5480614e+05, 2.8334720e+04],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.0002300e+03,
1.2415304e+05, 1.9039300e+03],
[0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.3154600e+03,
1.1581621e+05, 2.9711446e+05],

```
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 1.3542692e+05, 0.0000000e+00],
[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 5.4205000e+02,
 5.1743150e+04, 0.0000000e+00],
[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 0.0000000e+00,
 1.1698380e+05, 4.5173060e+04]])
```

[ ]

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(x)
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.2)
X_test.shape
```

(10, 6)

[ ]

```
X_train.shape
```

(40, 6)

[ ]

```
Y_test.shape
```

(10,)

[ ]

```
Y_train.shape
```

(40,)

[ ]

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,Y_train)
```

LinearRegression()

[ ]

```
Y_pred=regressor.predict(X_test)
from sklearn.metrics import r2_score
print(r2_score(Y_test,Y_pred))
```

0.9224858151384518

PROJECT-7

Salary Estimation System

Objective: Estimate salary based on the career level
Using non-linear regression.
→First degree polynomial is ax+b.
→Second degree polynomial is ax^2+bx+c.
→Third degree polynomial is ax^3+bx^2+cx+d.
→Fourth degree polynomial is ax^4+bx^3+cx^2+dx+e.
→Polynomial features are calculated for non-linear regression.
→Polynomial features are the coefficients of polynomial expression that is for fourth degree polynomial
a,b,c,d,e are called polynomial features.

```
import pandas as pd
dataset=pd.read_csv('Position_Salaries.csv')
```

```
dataset
```

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

```
X=dataset.iloc[:,1].values

X

array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

X=X.reshape(-1,1)
```

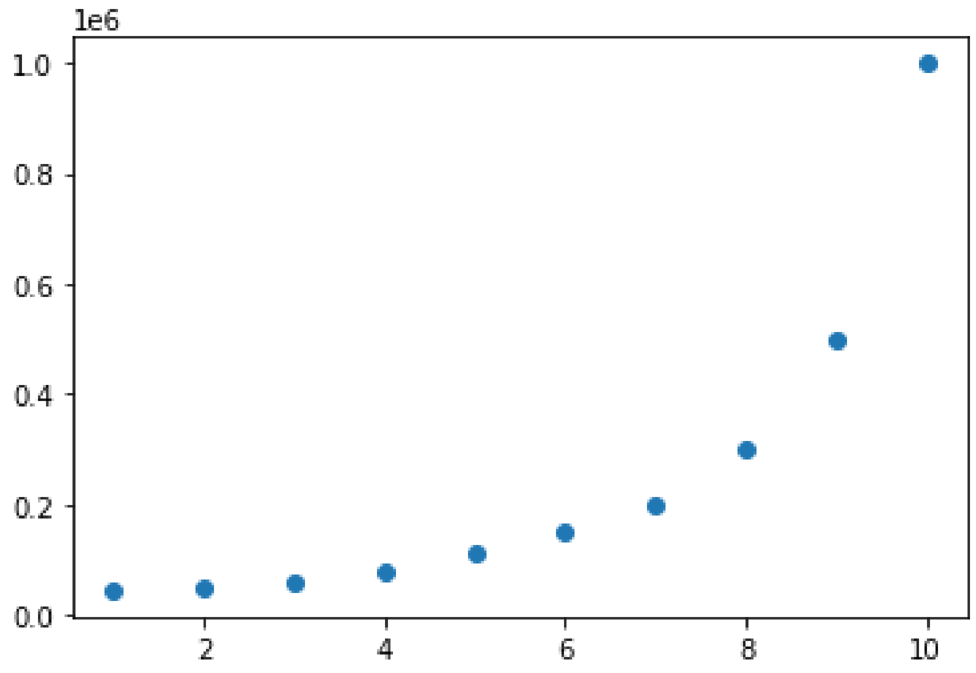
```
In [127]:
X
Out[127]:
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

```
Y=dataset.iloc[:,-1].values

Y

array([  45000,   50000,   60000,   80000,  110000,  150000,  200000,
        300000,  500000, 1000000])

import matplotlib.pyplot as plt
plt.scatter(X,Y)
```



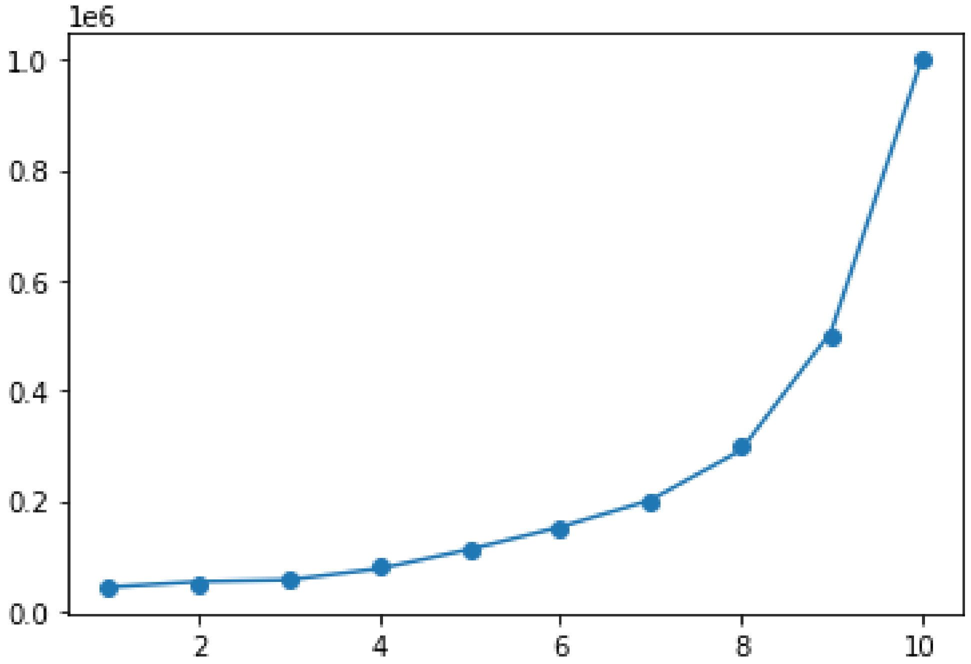
```
from sklearn.preprocessing import PolynomialFeatures
pf=PolynomialFeatures(degree=5)
X_poly=pf.fit_transform(X)
```

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
```

```
regressor.fit(X_poly,Y)
```

```
LinearRegression()
```

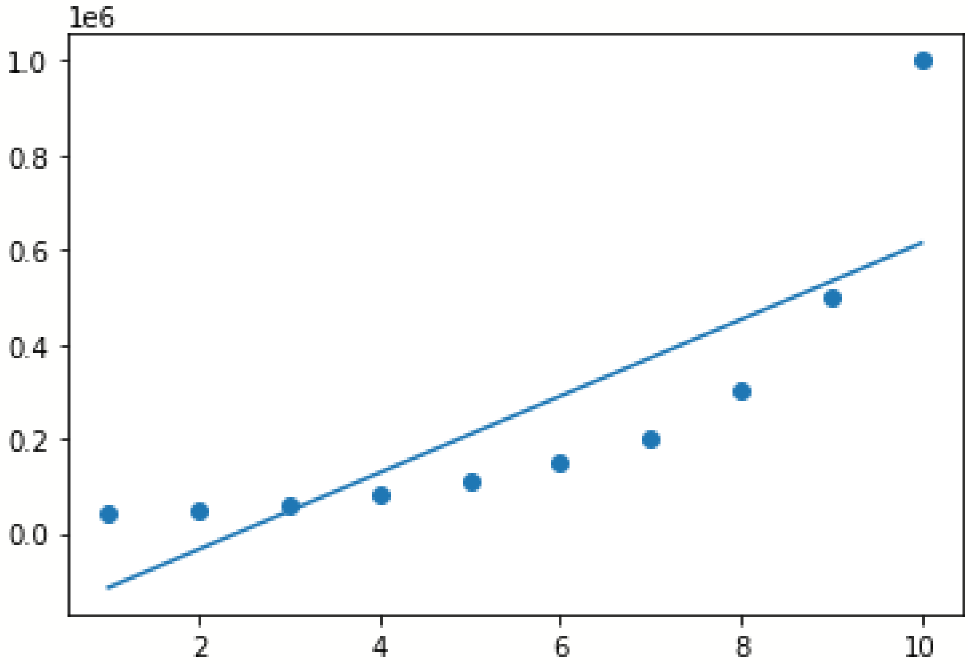
```
import matplotlib.pyplot as plt
plt.scatter(X,Y)
plt.plot(X,regressor.predict(X_poly))
```




```
regressor2=LinearRegression()
regressor2.fit(X,Y)
```

```
LinearRegression()
```

```
import matplotlib.pyplot as plt
plt.scatter(X,Y)
plt.plot(X,regressor2.predict(X))
```


1e6
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10

PROJECT-8
Product Sale Classification

Objective: The objective is to predict whether customer will be purchasing the product or not based on age and estimated salary.

```
import pandas as pd
dataset=pd.read_csv('product_sale.csv')
dataset
```

Age	EstimatedSalary	Purchased	
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

400 rows × 3 columns

```
X=dataset.iloc[:,:-1].values
X
```

```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000],
       [    26,  80000],
       [    26,  52000],
       [    20,  86000],
       [    32,  18000],
       [    18,  82000],
       [    29,  80000],
       [    47,  25000],
       [    45,  26000],
       [    46,  28000],
```

```
[   48,  29000],
[   45,  22000],
[   47,  49000],
[   48,  41000],
[   45,  22000],
[   46,  23000],
[   47,  20000],
[   49,  28000],
[   47,  30000],
[   29,  43000],
[   31,  18000],
[   31,  74000],
[   27, 137000],
[   21,  16000],
[   28,  44000],
[   27,  90000],
[   35,  27000],
[   33,  28000],
[   30,  49000],
[   26,  72000],
[   27,  31000],
[   27,  17000],
[   33,  51000],
[   35, 108000],
[   30,  15000],
[   28,  84000],
[   23,  20000],
[   25,  79000],
[   27,  54000],
[   30, 135000],
[   31,  89000],
[   24,  32000],
[   18,  44000],
[   29,  83000],
[   35,  23000],
[   27,  58000],
[   24,  55000],
[   23,  48000],
[   28,  79000],
[   22,  18000],
[   32, 117000],
[   27,  20000],
[   25,  87000],
[   23,  66000],
[   32, 120000],
[   59,  83000],
[   24,  58000],
[   24,  19000],
[   23,  82000],
[   22,  63000],
[   31,  68000],
[   25,  80000],
[   24,  27000],
[   20,  23000],
[   33, 113000],
[   32,  18000],
[   34, 112000],
[   18,  52000],
[   22,  27000],
```

```
[    28,  87000],
[    26,  17000],
[    30,  80000],
[    39,  42000],
[    20,  49000],
[    35,  88000],
[    30,  62000],
[    31, 118000],
[    24,  55000],
[    28,  85000],
[    26,  81000],
[    35,  50000],
[    22,  81000],
[    30, 116000],
[    26,  15000],
[    29,  28000],
[    29,  83000],
[    35,  44000],
[    35,  25000],
[    28, 123000],
[    35,  73000],
[    28,  37000],
[    27,  88000],
[    28,  59000],
[    32,  86000],
[    33, 149000],
[    19,  21000],
[    21,  72000],
[    26,  35000],
[    27,  89000],
[    26,  86000],
[    38,  80000],
[    39,  71000],
[    37,  71000],
[    38,  61000],
[    37,  55000],
[    42,  80000],
[    40,  57000],
[    35,  75000],
[    36,  52000],
[    40,  59000],
[    41,  59000],
[    36,  75000],
[    37,  72000],
[    40,  75000],
[    35,  53000],
[    41,  51000],
[    39,  61000],
[    42,  65000],
[    26,  32000],
[    30,  17000],
[    26,  84000],
[    31,  58000],
[    33,  31000],
[    30,  87000],
[    21,  68000],
[    28,  55000],
[    23,  63000],
[    20,  82000],
```

```
[    30, 107000],
[    28,  59000],
[    19,  25000],
[    19,  85000],
[    18,  68000],
[    35,  59000],
[    30,  89000],
[    34,  25000],
[    24,  89000],
[    27,  96000],
[    41,  30000],
[    29,  61000],
[    20,  74000],
[    26,  15000],
[    41,  45000],
[    31,  76000],
[    36,  50000],
[    40,  47000],
[    31,  15000],
[    46,  59000],
[    29,  75000],
[    26,  30000],
[    32, 135000],
[    32, 100000],
[    25,  90000],
[    37,  33000],
[    35,  38000],
[    33,  69000],
[    18,  86000],
[    22,  55000],
[    35,  71000],
[    29, 148000],
[    29,  47000],
[    21,  88000],
[    34, 115000],
[    26, 118000],
[    34,  43000],
[    34,  72000],
[    23,  28000],
[    35,  47000],
[    25,  22000],
[    24,  23000],
[    31,  34000],
[    26,  16000],
[    31,  71000],
[    32, 117000],
[    33,  43000],
[    33,  60000],
[    31,  66000],
[    20,  82000],
[    33,  41000],
[    35,  72000],
[    28,  32000],
[    24,  84000],
[    19,  26000],
[    29,  43000],
[    19,  70000],
[    28,  89000],
[    34,  43000],
```

```
[    30,  79000],
[    20,  36000],
[    26,  80000],
[    35,  22000],
[    35,  39000],
[    49,  74000],
[    39, 134000],
[    41,  71000],
[    58, 101000],
[    47,  47000],
[    55, 130000],
[    52, 114000],
[    40, 142000],
[    46,  22000],
[    48,  96000],
[    52, 150000],
[    59,  42000],
[    35,  58000],
[    47,  43000],
[    60, 108000],
[    49,  65000],
[    40,  78000],
[    46,  96000],
[    59, 143000],
[    41,  80000],
[    35,  91000],
[    37, 144000],
[    60, 102000],
[    35,  60000],
[    37,  53000],
[    36, 126000],
[    56, 133000],
[    40,  72000],
[    42,  80000],
[    35, 147000],
[    39,  42000],
[    40, 107000],
[    49,  86000],
[    38, 112000],
[    46,  79000],
[    40,  57000],
[    37,  80000],
[    46,  82000],
[    53, 143000],
[    42, 149000],
[    38,  59000],
[    50,  88000],
[    56, 104000],
[    41,  72000],
[    51, 146000],
[    35,  50000],
[    57, 122000],
[    41,  52000],
[    35,  97000],
[    44,  39000],
[    37,  52000],
[    48, 134000],
[    37, 146000],
[    50,  44000],
```

```
[    52,  90000],
[    41,  72000],
[    40,  57000],
[    58,  95000],
[    45, 131000],
[    35,  77000],
[    36, 144000],
[    55, 125000],
[    35,  72000],
[    48,  90000],
[    42, 108000],
[    40,  75000],
[    37,  74000],
[    47, 144000],
[    40,  61000],
[    43, 133000],
[    59,  76000],
[    60,  42000],
[    39, 106000],
[    57,  26000],
[    57,  74000],
[    38,  71000],
[    49,  88000],
[    52,  38000],
[    50,  36000],
[    59,  88000],
[    35,  61000],
[    37,  70000],
[    52,  21000],
[    48, 141000],
[    37,  93000],
[    37,  62000],
[    48, 138000],
[    41,  79000],
[    37,  78000],
[    39, 134000],
[    49,  89000],
[    55,  39000],
[    37,  77000],
[    35,  57000],
[    36,  63000],
[    42,  73000],
[    43, 112000],
[    45,  79000],
[    46, 117000],
[    58,  38000],
[    48,  74000],
[    37, 137000],
[    37,  79000],
[    40,  60000],
[    42,  54000],
[    51, 134000],
[    47, 113000],
[    36, 125000],
[    38,  50000],
[    42,  70000],
[    39,  96000],
[    38,  50000],
[    49, 141000],
```

```
[   39,  79000],
[   39,  75000],
[   54, 104000],
[   35,  55000],
[   45,  32000],
[   36,  60000],
[   52, 138000],
[   53,  82000],
[   41,  52000],
[   48,  30000],
[   48, 131000],
[   41,  60000],
[   41,  72000],
[   42,  75000],
[   36, 118000],
[   47, 107000],
[   38,  51000],
[   48, 119000],
[   42,  65000],
[   40,  65000],
[   57,  60000],
[   36,  54000],
[   58, 144000],
[   35,  79000],
[   38,  55000],
[   39, 122000],
[   53, 104000],
[   35,  75000],
[   38,  65000],
[   47,  51000],
[   47, 105000],
[   41,  63000],
[   53,  72000],
[   54, 108000],
[   39,  77000],
[   38,  61000],
[   38, 113000],
[   37,  75000],
[   42,  90000],
[   37,  57000],
[   36,  99000],
[   60,  34000],
[   54,  70000],
[   41,  72000],
[   40,  71000],
[   42,  54000],
[   43, 129000],
[   53,  34000],
[   47,  50000],
[   42,  79000],
[   42, 104000],
[   59,  29000],
[   58,  47000],
[   46,  88000],
[   38,  71000],
[   54,  26000],
[   60,  46000],
[   60,  83000],
[   39,  73000],
```

```
       [    59, 130000],
       [    37,  80000],
       [    46,  32000],
       [    46,  74000],
       [    42,  53000],
       [    41,  87000],
       [    58,  23000],
       [    42,  64000],
       [    48,  33000],
       [    44, 139000],
       [    49,  28000],
       [    57,  33000],
       [    56,  60000],
       [    49,  39000],
       [    39,  71000],
       [    47,  34000],
       [    48,  35000],
       [    48,  33000],
       [    47,  23000],
       [    45,  45000],
       [    60,  42000],
       [    39,  59000],
       [    46,  41000],
       [    51,  23000],
       [    50,  20000],
       [    36,  33000],
       [    49,  36000]])
```

[ ]

```
Y=dataset.iloc[:,-1].values
Y
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1])
```

[ ]

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X=ss.fit_transform(X)
X
```

```
array([[-1.78179743, -1.49004624],
```

```
[-0.25358736, -1.46068138],
[-1.11320552, -0.78528968],
[-1.01769239, -0.37418169],
[-1.78179743,  0.18375059],
[-1.01769239, -0.34481683],
[-1.01769239,  0.41866944],
[-0.54012675,  2.35674998],
[-1.20871865, -1.07893824],
[-0.25358736, -0.13926283],
[-1.11320552,  0.30121002],
[-1.11320552, -0.52100597],
[-1.6862843 ,  0.47739916],
[-0.54012675, -1.51941109],
[-1.87731056,  0.35993973],
[-0.82666613,  0.30121002],
[ 0.89257019, -1.3138571 ],
[ 0.70154394, -1.28449224],
[ 0.79705706, -1.22576253],
[ 0.98808332, -1.19639767],
[ 0.70154394, -1.40195167],
[ 0.89257019, -0.60910054],
[ 0.98808332, -0.84401939],
[ 0.70154394, -1.40195167],
[ 0.79705706, -1.37258681],
[ 0.89257019, -1.46068138],
[ 1.08359645, -1.22576253],
[ 0.89257019, -1.16703281],
[-0.82666613, -0.78528968],
[-0.63563988, -1.51941109],
[-0.63563988,  0.12502088],
[-1.01769239,  1.97500684],
[-1.59077117, -1.5781408 ],
[-0.92217926, -0.75592482],
[-1.01769239,  0.59485858],
[-0.25358736, -1.25512738],
[-0.44461362, -1.22576253],
[-0.73115301, -0.60910054],
[-1.11320552,  0.06629116],
[-1.01769239, -1.13766796],
[-1.01769239, -1.54877595],
[-0.44461362, -0.55037082],
[-0.25358736,  1.123426  ],
[-0.73115301, -1.60750566],
[-0.92217926,  0.41866944],
[-1.39974491, -1.46068138],
[-1.20871865,  0.27184516],
[-1.01769239, -0.46227625],
[-0.73115301,  1.91627713],
[-0.63563988,  0.56549373],
[-1.30423178, -1.1083031 ],
[-1.87731056, -0.75592482],
[-0.82666613,  0.38930459],
[-0.25358736, -1.37258681],
[-1.01769239, -0.34481683],
[-1.30423178, -0.4329114 ],
[-1.39974491, -0.63846539],
[-0.92217926,  0.27184516],
[-1.49525804, -1.51941109],
[-0.54012675,  1.38770971],
```

```
[-1.01769239, -1.46068138],
[-1.20871865,  0.50676401],
[-1.39974491, -0.10989798],
[-0.54012675,  1.47580428],
[ 2.03872775,  0.38930459],
[-1.30423178, -0.34481683],
[-1.30423178, -1.49004624],
[-1.39974491,  0.35993973],
[-1.49525804, -0.19799255],
[-0.63563988, -0.05116826],
[-1.20871865,  0.30121002],
[-1.30423178, -1.25512738],
[-1.6862843 , -1.37258681],
[-0.44461362,  1.27025028],
[-0.54012675, -1.51941109],
[-0.34910049,  1.24088543],
[-1.87731056, -0.52100597],
[-1.49525804, -1.25512738],
[-0.92217926,  0.50676401],
[-1.11320552, -1.54877595],
[-0.73115301,  0.30121002],
[ 0.12846516, -0.81465453],
[-1.6862843 , -0.60910054],
[-0.25358736,  0.53612887],
[-0.73115301, -0.2273574 ],
[-0.63563988,  1.41707457],
[-1.30423178, -0.4329114 ],
[-0.92217926,  0.4480343 ],
[-1.11320552,  0.33057487],
[-0.25358736, -0.57973568],
[-1.49525804,  0.33057487],
[-0.73115301,  1.35834485],
[-1.11320552, -1.60750566],
[-0.82666613, -1.22576253],
[-0.82666613,  0.38930459],
[-0.25358736, -0.75592482],
[-0.25358736, -1.3138571 ],
[-0.92217926,  1.56389885],
[-0.25358736,  0.09565602],
[-0.92217926, -0.96147882],
[-1.01769239,  0.53612887],
[-0.92217926, -0.31545197],
[-0.54012675,  0.47739916],
[-0.44461362,  2.32738512],
[-1.78179743, -1.43131652],
[-1.59077117,  0.06629116],
[-1.11320552, -1.02020853],
[-1.01769239,  0.56549373],
[-1.11320552,  0.47739916],
[ 0.03295203,  0.30121002],
[ 0.12846516,  0.03692631],
[-0.0625611 ,  0.03692631],
[ 0.03295203, -0.25672226],
[-0.0625611 , -0.4329114 ],
[ 0.41500455,  0.30121002],
[ 0.22397829, -0.37418169],
[-0.25358736,  0.15438573],
[-0.15807423, -0.52100597],
[ 0.22397829, -0.31545197],
```

```
[ 0.31949142, -0.31545197],
[-0.15807423,  0.15438573],
[-0.0625611 ,  0.06629116],
[ 0.22397829,  0.15438573],
[-0.25358736, -0.49164111],
[ 0.31949142, -0.55037082],
[ 0.12846516, -0.25672226],
[ 0.41500455, -0.13926283],
[-1.11320552, -1.1083031 ],
[-0.73115301, -1.54877595],
[-1.11320552,  0.41866944],
[-0.63563988, -0.34481683],
[-0.44461362, -1.13766796],
[-0.73115301,  0.50676401],
[-1.59077117, -0.05116826],
[-0.92217926, -0.4329114 ],
[-1.39974491, -0.19799255],
[-1.6862843 ,  0.35993973],
[-0.73115301,  1.09406114],
[-0.92217926, -0.31545197],
[-1.78179743, -1.3138571 ],
[-1.78179743,  0.4480343 ],
[-1.87731056, -0.05116826],
[-0.25358736, -0.31545197],
[-0.73115301,  0.56549373],
[-0.34910049, -1.3138571 ],
[-1.30423178,  0.56549373],
[-1.01769239,  0.77104772],
[ 0.31949142, -1.16703281],
[-0.82666613, -0.25672226],
[-1.6862843 ,  0.12502088],
[-1.11320552, -1.60750566],
[ 0.31949142, -0.72655996],
[-0.63563988,  0.18375059],
[-0.15807423, -0.57973568],
[ 0.22397829, -0.66783025],
[-0.63563988, -1.60750566],
[ 0.79705706, -0.31545197],
[-0.82666613,  0.15438573],
[-1.11320552, -1.16703281],
[-0.54012675,  1.91627713],
[-0.54012675,  0.88850715],
[-1.20871865,  0.59485858],
[-0.0625611 , -1.07893824],
[-0.25358736, -0.93211396],
[-0.44461362, -0.02180341],
[-1.87731056,  0.47739916],
[-1.49525804, -0.4329114 ],
[-0.25358736,  0.03692631],
[-0.82666613,  2.29802026],
[-0.82666613, -0.66783025],
[-1.59077117,  0.53612887],
[-0.34910049,  1.32898   ],
[-1.11320552,  1.41707457],
[-0.34910049, -0.78528968],
[-0.34910049,  0.06629116],
[-1.39974491, -1.22576253],
[-0.25358736, -0.66783025],
[-1.20871865, -1.40195167],
```

```
[-1.30423178, -1.37258681],
[-0.63563988, -1.04957339],
[-1.11320552, -1.5781408 ],
[-0.63563988,  0.03692631],
[-0.54012675,  1.38770971],
[-0.44461362, -0.78528968],
[-0.44461362, -0.28608712],
[-0.63563988, -0.10989798],
[-1.6862843 ,  0.35993973],
[-0.44461362, -0.84401939],
[-0.25358736,  0.06629116],
[-0.92217926, -1.1083031 ],
[-1.30423178,  0.41866944],
[-1.78179743, -1.28449224],
[-0.82666613, -0.78528968],
[-1.78179743,  0.00756145],
[-0.92217926,  0.56549373],
[-0.34910049, -0.78528968],
[-0.73115301,  0.27184516],
[-1.6862843 , -0.99084367],
[-1.11320552,  0.30121002],
[-0.25358736, -1.40195167],
[-0.25358736, -0.9027491 ],
[ 1.08359645,  0.12502088],
[ 0.12846516,  1.88691227],
[ 0.31949142,  0.03692631],
[ 1.94321462,  0.917872  ],
[ 0.89257019, -0.66783025],
[ 1.65667523,  1.76945285],
[ 1.37013584,  1.29961514],
[ 0.22397829,  2.12183112],
[ 0.79705706, -1.40195167],
[ 0.98808332,  0.77104772],
[ 1.37013584,  2.35674998],
[ 2.03872775, -0.81465453],
[-0.25358736, -0.34481683],
[ 0.89257019, -0.78528968],
[ 2.13424088,  1.123426  ],
[ 1.08359645, -0.13926283],
[ 0.22397829,  0.2424803 ],
[ 0.79705706,  0.77104772],
[ 2.03872775,  2.15119598],
[ 0.31949142,  0.30121002],
[-0.25358736,  0.62422344],
[-0.0625611 ,  2.18056084],
[ 2.13424088,  0.94723686],
[-0.25358736, -0.28608712],
[-0.0625611 , -0.49164111],
[-0.15807423,  1.65199342],
[ 1.75218836,  1.85754742],
[ 0.22397829,  0.06629116],
[ 0.41500455,  0.30121002],
[-0.25358736,  2.26865541],
[ 0.12846516, -0.81465453],
[ 0.22397829,  1.09406114],
[ 1.08359645,  0.47739916],
[ 0.03295203,  1.24088543],
[ 0.79705706,  0.27184516],
[ 0.22397829, -0.37418169],
```

```
[-0.0625611 ,  0.30121002],
[ 0.79705706,  0.35993973],
[ 1.46564897,  2.15119598],
[ 0.41500455,  2.32738512],
[ 0.03295203, -0.31545197],
[ 1.17910958,  0.53612887],
[ 1.75218836,  1.00596657],
[ 0.31949142,  0.06629116],
[ 1.27462271,  2.23929055],
[-0.25358736, -0.57973568],
[ 1.84770149,  1.53453399],
[ 0.31949142, -0.52100597],
[-0.25358736,  0.80041258],
[ 0.60603081, -0.9027491 ],
[-0.0625611 , -0.52100597],
[ 0.98808332,  1.88691227],
[-0.0625611 ,  2.23929055],
[ 1.17910958, -0.75592482],
[ 1.37013584,  0.59485858],
[ 0.31949142,  0.06629116],
[ 0.22397829, -0.37418169],
[ 1.94321462,  0.74168287],
[ 0.70154394,  1.7988177 ],
[-0.25358736,  0.21311545],
[-0.15807423,  2.18056084],
[ 1.65667523,  1.62262856],
[-0.25358736,  0.06629116],
[ 0.98808332,  0.59485858],
[ 0.41500455,  1.123426  ],
[ 0.22397829,  0.15438573],
[-0.0625611 ,  0.12502088],
[ 0.89257019,  2.18056084],
[ 0.22397829, -0.25672226],
[ 0.51051768,  1.85754742],
[ 2.03872775,  0.18375059],
[ 2.13424088, -0.81465453],
[ 0.12846516,  1.06469629],
[ 1.84770149, -1.28449224],
[ 1.84770149,  0.12502088],
[ 0.03295203,  0.03692631],
[ 1.08359645,  0.53612887],
[ 1.37013584, -0.93211396],
[ 1.17910958, -0.99084367],
[ 2.03872775,  0.53612887],
[-0.25358736, -0.25672226],
[-0.0625611 ,  0.00756145],
[ 1.37013584, -1.43131652],
[ 0.98808332,  2.09246627],
[-0.0625611 ,  0.68295315],
[-0.0625611 , -0.2273574 ],
[ 0.98808332,  2.0043717 ],
[ 0.31949142,  0.27184516],
[-0.0625611 ,  0.2424803 ],
[ 0.12846516,  1.88691227],
[ 1.08359645,  0.56549373],
[ 1.65667523, -0.9027491 ],
[-0.0625611 ,  0.21311545],
[-0.25358736, -0.37418169],
[-0.15807423, -0.19799255],
```

```
[ 0.41500455,  0.09565602],
[ 0.51051768,  1.24088543],
[ 0.70154394,  0.27184516],
[ 0.79705706,  1.38770971],
[ 1.94321462, -0.93211396],
[ 0.98808332,  0.12502088],
[-0.0625611 ,  1.97500684],
[-0.0625611 ,  0.27184516],
[ 0.22397829, -0.28608712],
[ 0.41500455, -0.46227625],
[ 1.27462271,  1.88691227],
[ 0.89257019,  1.27025028],
[-0.15807423,  1.62262856],
[ 0.03295203, -0.57973568],
[ 0.41500455,  0.00756145],
[ 0.12846516,  0.77104772],
[ 0.03295203, -0.57973568],
[ 1.08359645,  2.09246627],
[ 0.12846516,  0.27184516],
[ 0.12846516,  0.15438573],
[ 1.5611621 ,  1.00596657],
[-0.25358736, -0.4329114 ],
[ 0.70154394, -1.1083031 ],
[-0.15807423, -0.28608712],
[ 1.37013584,  2.0043717 ],
[ 1.46564897,  0.35993973],
[ 0.31949142, -0.52100597],
[ 0.98808332, -1.16703281],
[ 0.98808332,  1.7988177 ],
[ 0.31949142, -0.28608712],
[ 0.31949142,  0.06629116],
[ 0.41500455,  0.15438573],
[-0.15807423,  1.41707457],
[ 0.89257019,  1.09406114],
[ 0.03295203, -0.55037082],
[ 0.98808332,  1.44643942],
[ 0.41500455, -0.13926283],
[ 0.22397829, -0.13926283],
[ 1.84770149, -0.28608712],
[-0.15807423, -0.46227625],
[ 1.94321462,  2.18056084],
[-0.25358736,  0.27184516],
[ 0.03295203, -0.4329114 ],
[ 0.12846516,  1.53453399],
[ 1.46564897,  1.00596657],
[-0.25358736,  0.15438573],
[ 0.03295203, -0.13926283],
[ 0.89257019, -0.55037082],
[ 0.89257019,  1.03533143],
[ 0.31949142, -0.19799255],
[ 1.46564897,  0.06629116],
[ 1.5611621 ,  1.123426  ],
[ 0.12846516,  0.21311545],
[ 0.03295203, -0.25672226],
[ 0.03295203,  1.27025028],
[-0.0625611 ,  0.15438573],
[ 0.41500455,  0.59485858],
[-0.0625611 , -0.37418169],
[-0.15807423,  0.85914229],
```

```
 [ 2.13424088, -1.04957339],
 [ 1.5611621 ,  0.00756145],
 [ 0.31949142,  0.06629116],
 [ 0.22397829,  0.03692631],
 [ 0.41500455, -0.46227625],
 [ 0.51051768,  1.74008799],
 [ 1.46564897, -1.04957339],
 [ 0.89257019, -0.57973568],
 [ 0.41500455,  0.27184516],
 [ 0.41500455,  1.00596657],
 [ 2.03872775, -1.19639767],
 [ 1.94321462, -0.66783025],
 [ 0.79705706,  0.53612887],
 [ 0.03295203,  0.03692631],
 [ 1.5611621 , -1.28449224],
 [ 2.13424088, -0.69719511],
 [ 2.13424088,  0.38930459],
 [ 0.12846516,  0.09565602],
 [ 2.03872775,  1.76945285],
 [-0.0625611 ,  0.30121002],
 [ 0.79705706, -1.1083031 ],
 [ 0.79705706,  0.12502088],
 [ 0.41500455, -0.49164111],
 [ 0.31949142,  0.50676401],
 [ 1.94321462, -1.37258681],
 [ 0.41500455, -0.16862769],
 [ 0.98808332, -1.07893824],
 [ 0.60603081,  2.03373655],
 [ 1.08359645, -1.22576253],
 [ 1.84770149, -1.07893824],
 [ 1.75218836, -0.28608712],
 [ 1.08359645, -0.9027491 ],
 [ 0.12846516,  0.03692631],
 [ 0.89257019, -1.04957339],
 [ 0.98808332, -1.02020853],
 [ 0.98808332, -1.07893824],
 [ 0.89257019, -1.37258681],
 [ 0.70154394, -0.72655996],
 [ 2.13424088, -0.81465453],
 [ 0.12846516, -0.31545197],
 [ 0.79705706, -0.84401939],
 [ 1.27462271, -1.37258681],
 [ 1.17910958, -1.46068138],
 [-0.15807423, -1.07893824],
 [ 1.08359645, -0.99084367]])
```

[ ]

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
classifier.fit(X_train,Y_train)
```

LogisticRegression()

[ ]

```
Y_pred=classifier.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_pred,Y_test))
```

0.9

[ ]

```
classifier.predict([[15,40000]])
```

array([1])

PROJECT-9
Crop prediction system

Objective: In this project we are predicting crop by applying logistic regression, k-nearest neighbor classifier, decision tree and random forest.

```
import pandas as pd
dataset=pd.read_csv('crop.csv')
dataset
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

```
X=dataset.iloc[:,:-1]
X
```

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee

	N	P	K	temperature	humidity	ph	rainfall	label
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

```
X=dataset.iloc[:,:-1]
X
```

	N	P	K	temperature	humidity	ph	rainfall
0	90	42	43	20.879744	82.002744	6.502985	202.935536
1	85	58	41	21.770462	80.319644	7.038096	226.655537
2	60	55	44	23.004459	82.320763	7.840207	263.964248
3	74	35	40	26.491096	80.158363	6.980401	242.864034
4	78	42	42	20.130175	81.604873	7.628473	262.717340
...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507
2196	99	15	27	27.417112	56.636362	6.086922	127.924610
2197	118	33	30	24.131797	67.225123	6.362608	173.322839
2198	117	32	34	26.272418	52.127394	6.758793	127.175293
2199	104	18	30	23.603016	60.396475	6.779833	140.937041

2200 rows × 7 columns

```
Y=dataset.iloc[:,-1]
Y
```

```
0       rice
1       rice
2       rice
3       rice
4       rice
         ...
2195    coffee
2196    coffee
2197    coffee
2198    coffee
2199    coffee
Name: label, Length: 2200, dtype: object
```

[ ]

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X=ss.fit_transform(X)
```

[ ]
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)
```

[ ]
```
from sklearn.linear_model import LogisticRegression
lrClassifier=LogisticRegression()
lrClassifier.fit(X_train,Y_train)
```
LogisticRegression()

[ ]
```
from sklearn.neighbors import KNeighborsClassifier
knnClassifier=KNeighborsClassifier(n_neighbors=5)
knnClassifier.fit(X_train,Y_train)
```
KNeighborsClassifier()

[ ]

[ ]
```
from sklearn.tree import DecisionTreeClassifier
dtClassifier=DecisionTreeClassifier()
dtClassifier.fit(X_train,Y_train)
```
DecisionTreeClassifier()

[ ]
```
from sklearn.ensemble import RandomForestClassifier
rfClassifier=RandomForestClassifier(n_estimators=100)
rfClassifier.fit(X_train,Y_train)
```
RandomForestClassifier()

[ ]
```
Y_lr_pred=lrClassifier.predict(X_test)
Y_knn_pred=knnClassifier.predict(X_test)
Y_dt_pred=dtClassifier.predict(X_test)
Y_rf_pred=rfClassifier.predict(X_test)
```

[ ]
```
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_lr_pred,Y_test))
print(accuracy_score(Y_knn_pred,Y_test))
print(accuracy_score(Y_dt_pred,Y_test))
print(accuracy_score(Y_rf_pred,Y_test))
```
0.9590909090909091
0.9659090909090909
0.9863636363636363
0.990909090909091

POJECT-10

ML Flask Web App

Objective: The objective is to create web application for crop production system using flask.
Flask: Flask is a python web framework used to create web application.
→It is used to run web server locally.
→HTML templates will be working as front end for flask projects.

→Python will be working as the back end for flask projects.

→ML models will be pickled and transferred to the web server through python.

Pickling: It is a process of storing an object in string format in a pickle file.

→It is used in model deploying.

→It is done at training stage.

Unpickling: It is a process of loading an object in string format from pickle files(pkl) .

→Unpickling process will be done at deployment stage.

→In flask, HTML pages should be stored in templates folder only.

Templates

-Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Crop Prediction System
        </title>
    </head>
    <body>
        <form action="/predict" method="POST">
            <label>Enter Nitrogen:</label>
            <input type="text" name="N" /><br/>
            <label>Enter Phosphorous:</label>
            <input type="text" name="P" /><br/>
            <label>Enter Potassium:</label>
            <input type="text" name="K" /><br/>
            <label>Enter Temperature:</label>
            <input type="text" name="T" /><br/>
            <label>Enter Humidity:</label>
            <input type="text" name="H" /><br/>
            <label>Enter pH:</label>
            <input type="text" name="PH" /><br/>
            <label>Enter Rainfall:</label>
            <input type="text" name="R" /><br/>
            <input type="submit" />
            {{result}}
        </form>
```

```
    </body>
</html>
```

Procfile

web: gunicorn app:app

<u>Flask web app</u>

app.py

```
from flask import Flask,render_template,request
import pickle

model=pickle.load(open('model.pkl','rb'))

app=Flask(__name__)

@app.route('/')
def homePage():
    return (render_template('index.html'))

@app.route('/predict',methods=['POST'])
def collectData():
    N=float(request.form['N'])
    P=float(request.form['P'])
    K=float(request.form['K'])
    T=float(request.form['T'])
    H=float(request.form['H'])
    PH=float(request.form['PH'])
    R=float(request.form['R'])
    print(N,P,K,T,H,PH,R)
    result=model.predict([[N,P,K,T,H,PH,R]])
    return(result[0])

if __name__=="__main__":
    app.run(debug=True)
```

model.pkl (in google colab)

```
import pickle
f=open('model.pkl','wb')
pickle.dump(rfClassifier,f)
f.close()
```

Requirements

pandas

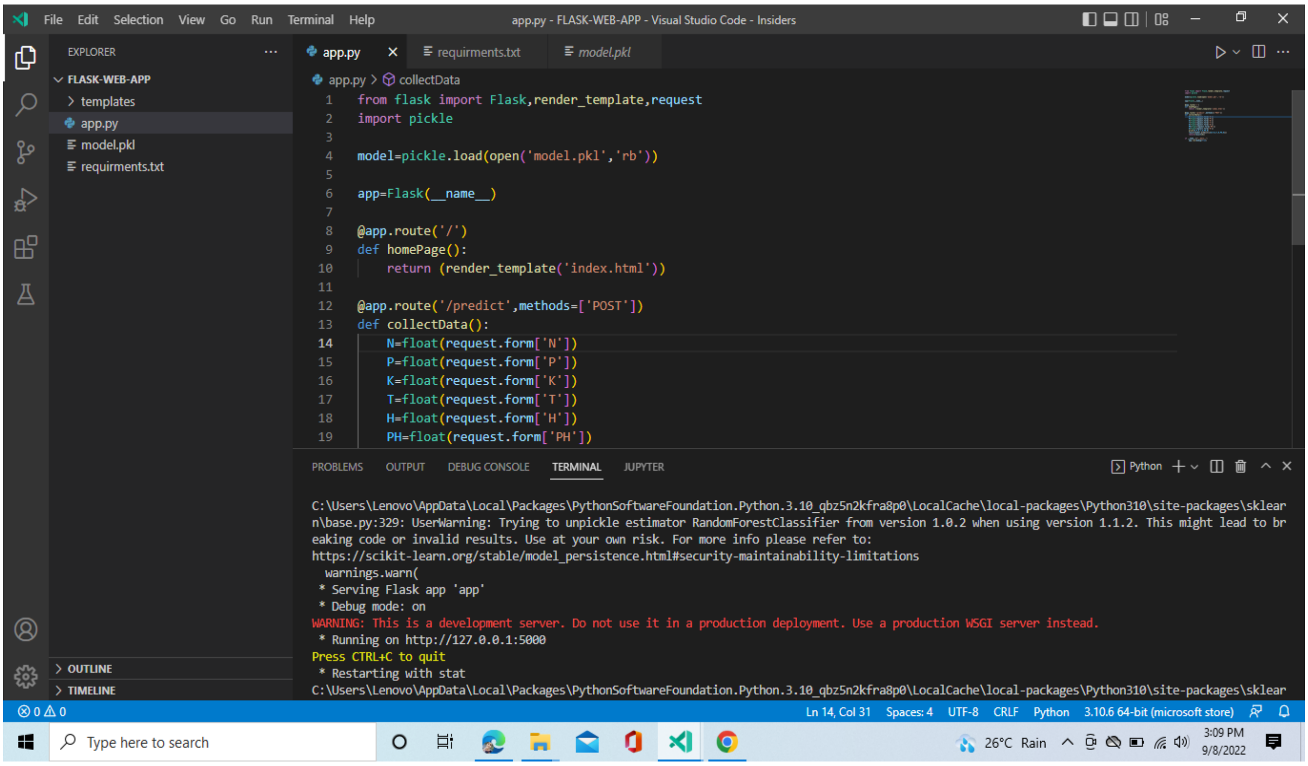
sklearn

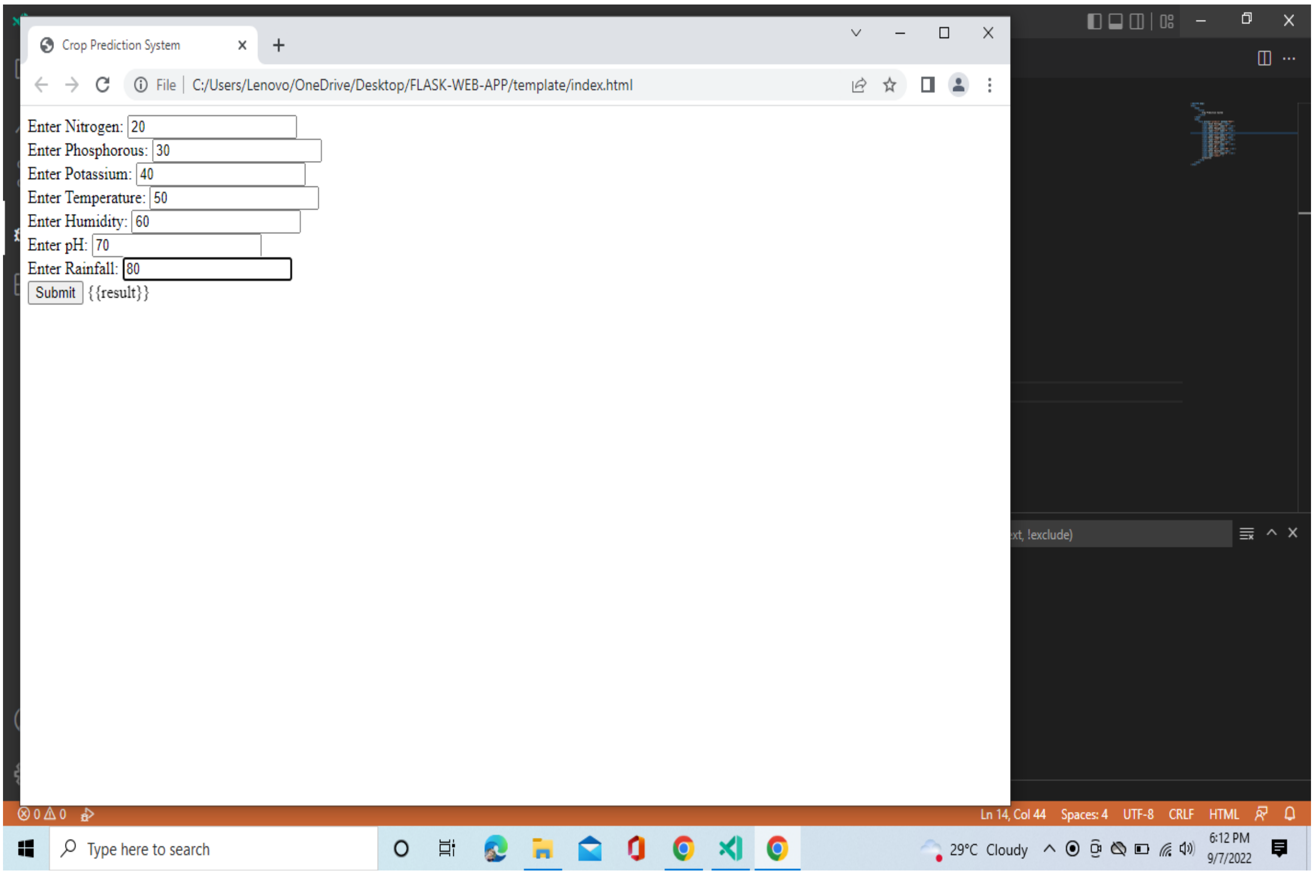
flask

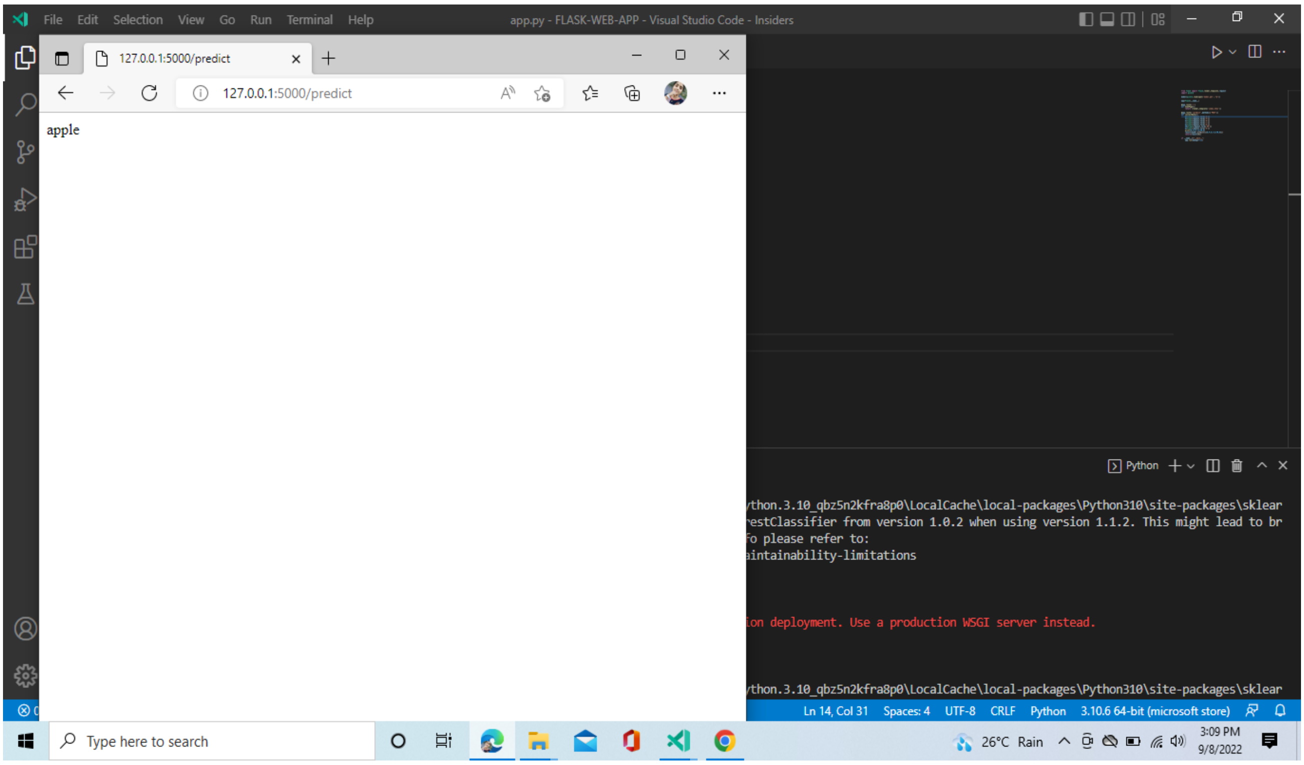
numpy

matplotlib

gunicorn


app.py - FLASK-WEB-APP - Visual Studio Code - Insiders
EXPLORER
FLASK-WEB-APP
templates
app.py
model.pkl
requirments.txt
from flask import Flask,render_template,request
import pickle
model=pickle.load(open('model.pkl','rb'))
app=Flask(__name__)
@app.route('/')
def homePage():
return (render_template('index.html'))
@app.route('/predict',methods=['POST'])
def collectData():
N=float(request.form['N'])
P=float(request.form['P'])
K=float(request.form['K'])
T=float(request.form['T'])
H=float(request.form['H'])
PH=float(request.form['PH'])
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
C:\Users\Lenovo\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 1.0.2 when using version 1.1.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
warnings.warn(
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
Ln 14, Col 31 Spaces: 4 UTF-8 CRLF Python 3.10.6 64-bit (microsoft store)
Type here to search
26°C Rain
3:09 PM
9/8/2022


Crop Prediction System
File | C:/Users/Lenovo/OneDrive/Desktop/FLASK-WEB-APP/template/index.html
Enter Nitrogen: 20
Enter Phosphorous: 30
Enter Potassium: 40
Enter Temperature: 50
Enter Humidity: 60
Enter pH: 70
Enter Rainfall: 80
Submit {{result}}
Ln 14, Col 44 Spaces: 4 UTF-8 CRLF HTML
Type here to search
29°C Cloudy
6:12 PM
9/7/2022


File Edit Selection View Go Run Terminal Help
app.py - FLASK-WEB-APP - Visual Studio Code - Insiders
127.0.0.1:5000/predict
apple
Python
ython.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklear
restClassifier from version 1.0.2 when using version 1.1.2. This might lead to br
fo please refer to:
aintainability-limitations
ion deployment. Use a production WSGI server instead.
ython.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\sklear
Ln 14, Col 31 Spaces: 4 UTF-8 CRLF Python 3.10.6 64-bit (microsoft store)
Type here to search
26°C Rain
3:09 PM
9/8/2022

PROJECT-11

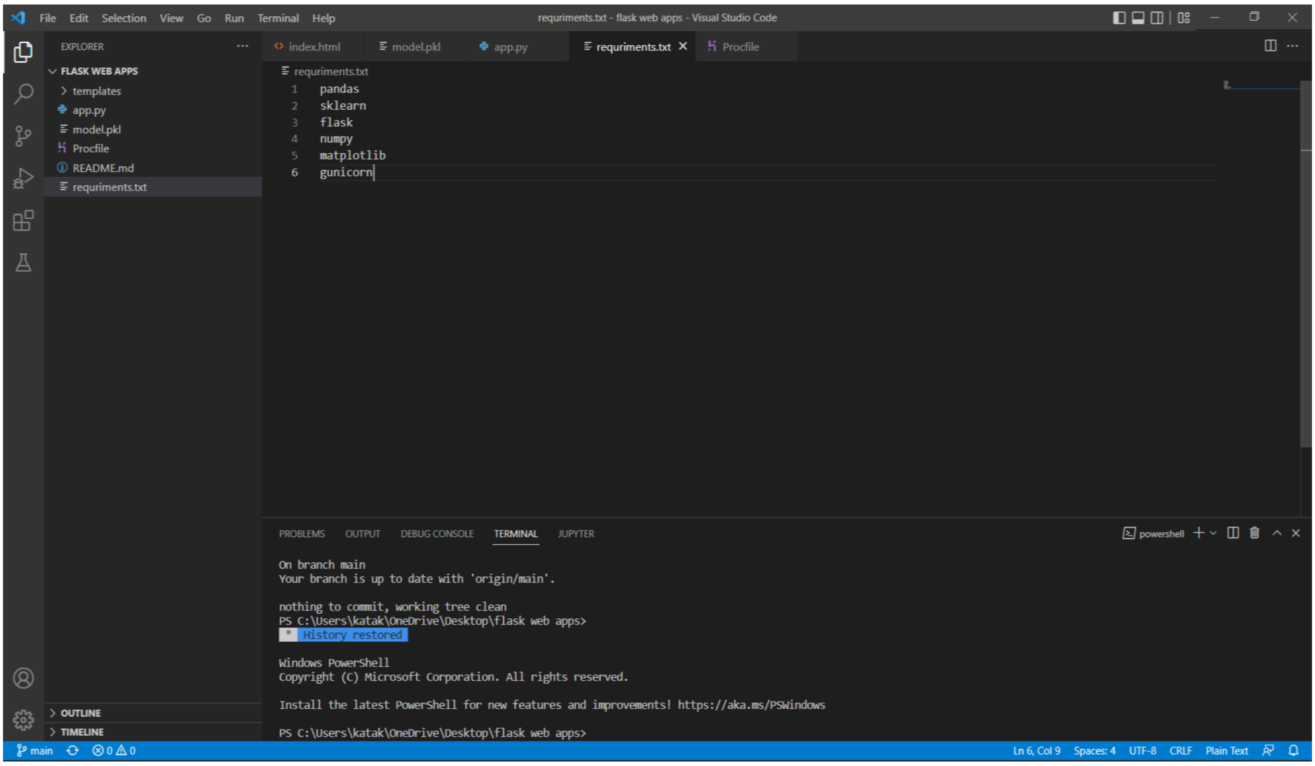
Heroku Deployment

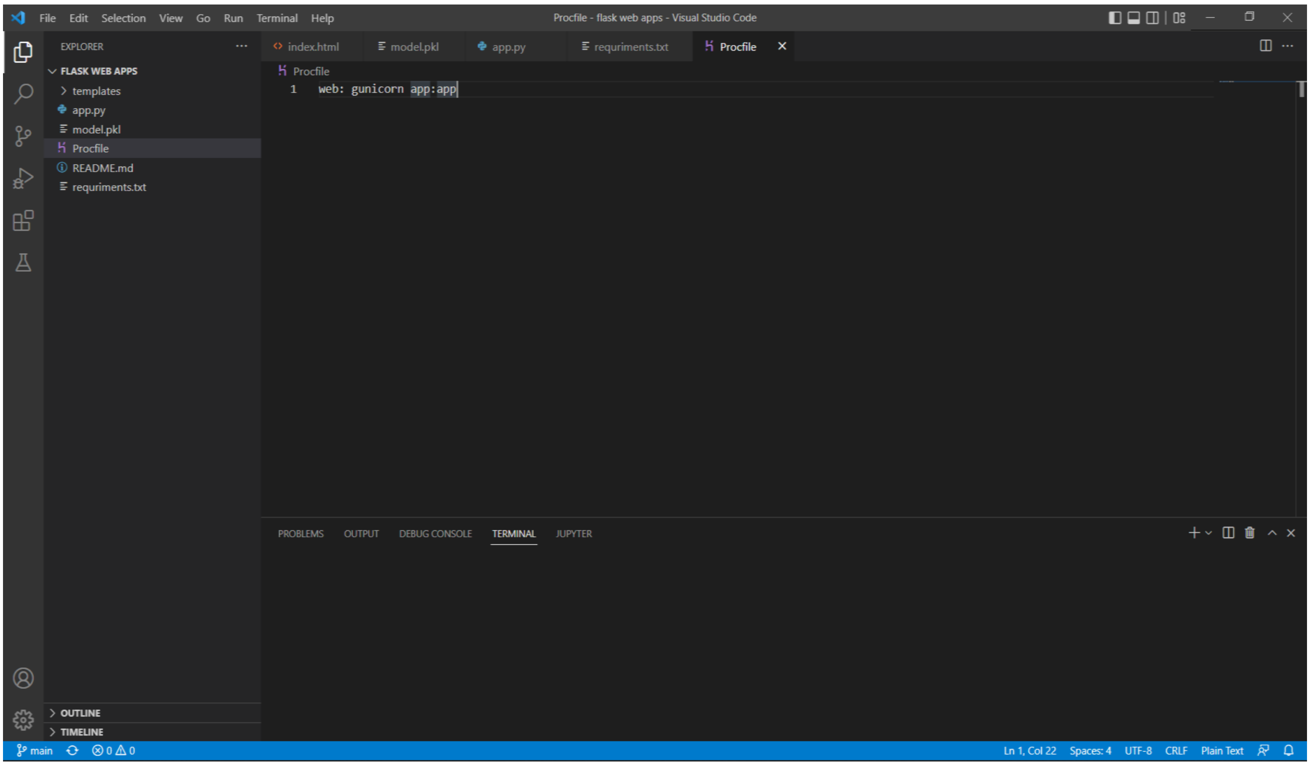
Objective: Deployment of crop production system Heroku platform.

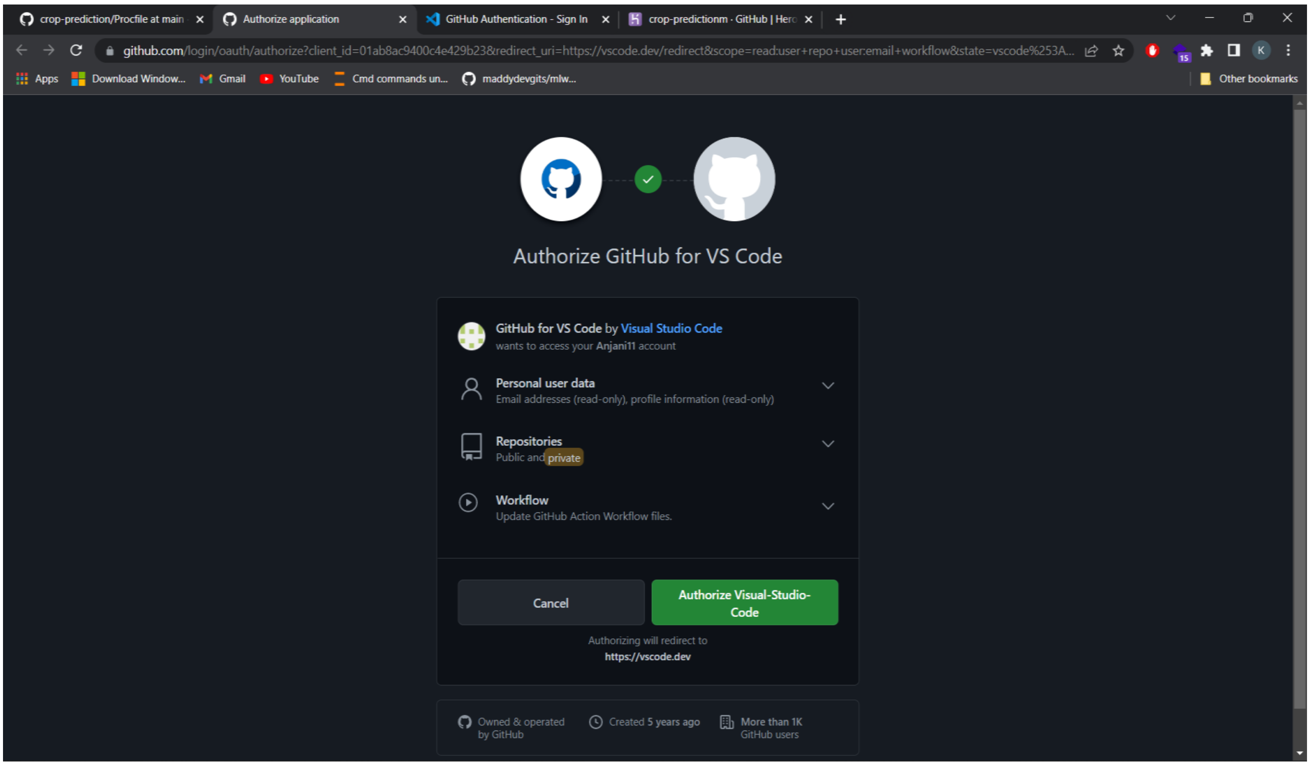
Whenever you are deploying ML model you have to create requirements text in which you have to provide list of packages related to project.

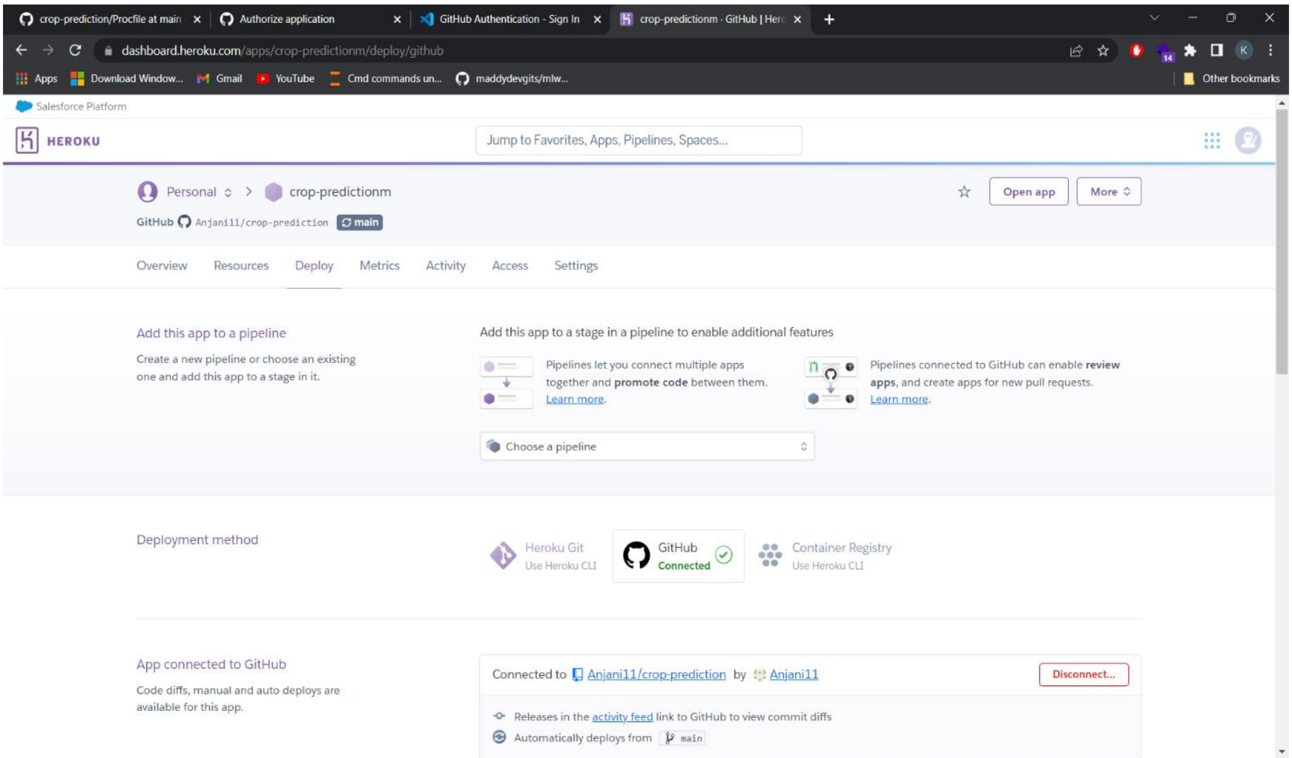
Procfile: procfile is a deployement file for Heroku platform and procfile will not have any extension.

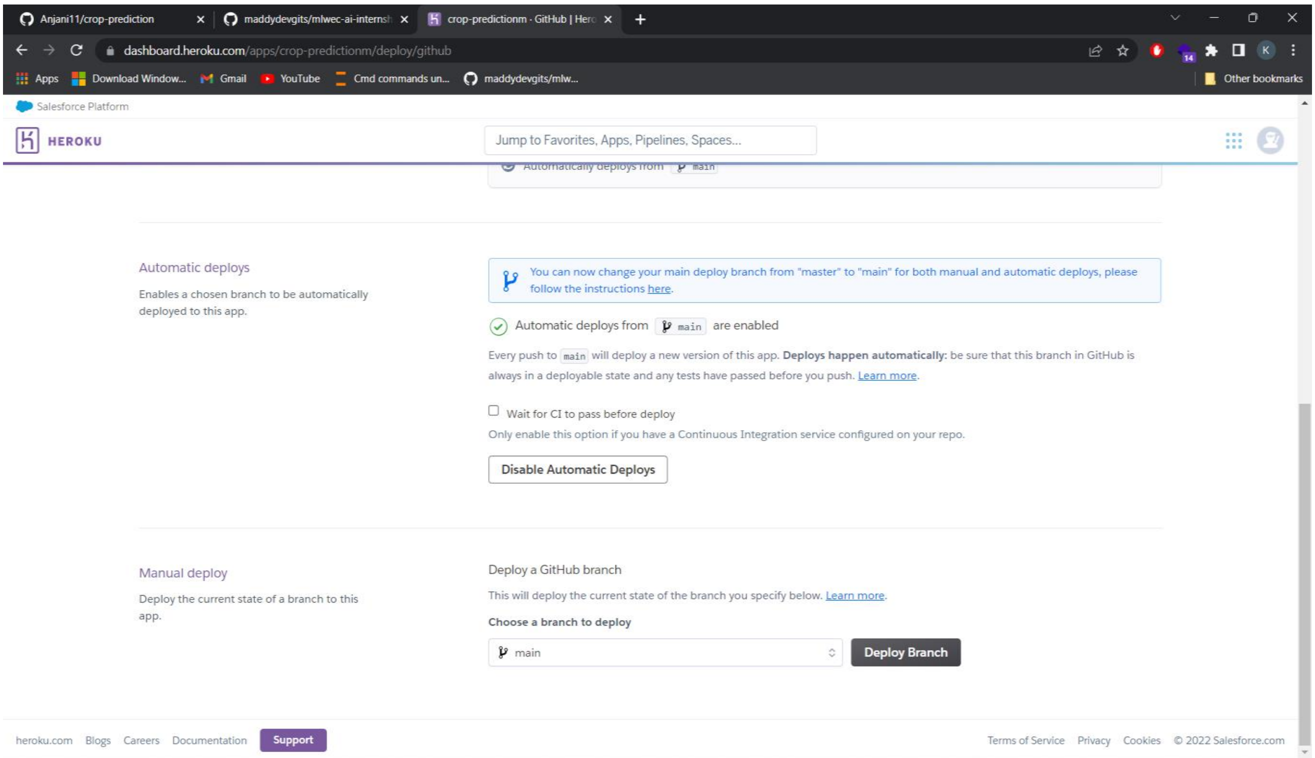
In the procfile we have to specific which applicant we have to run which script applicant we have to run specificly.

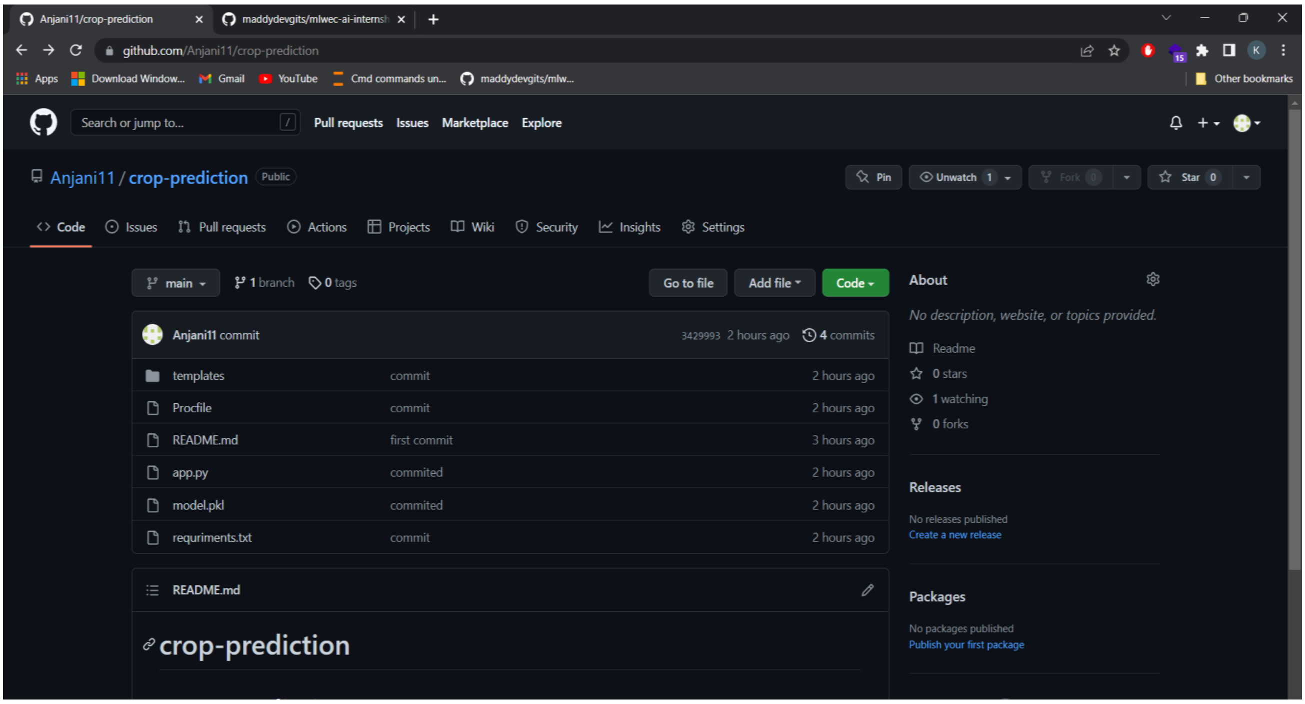



File Edit Selection View Go Run Terminal Help
Procfile - flask web apps - Visual Studio Code
EXPLORER
FLASK WEB APPS
templates
app.py
model.pkl
Procfile
README.md
requriments.txt
index.html
model.pkl
app.py
requriments.txt
Procfile
1 web: gunicorn app:app
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
OUTLINE
TIMELINE
main
Ln 1, Col 22 Spaces: 4 UTF-8 CRLF Plain Text

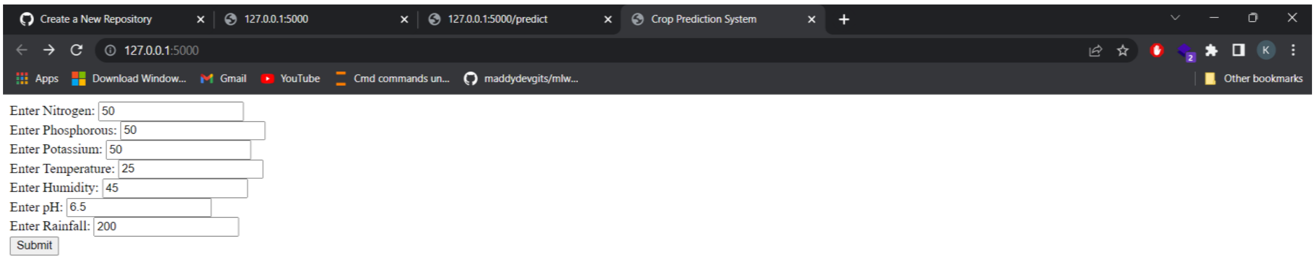

crop-prediction/Procfile at main
Authorize application
GitHub Authentication - Sign In
crop-predictionm - GitHub | Hero
Apps
Download Window...
Gmail
YouTube
Cmd commands un...
maddydevgits/mlw...
Other bookmarks
Authorize GitHub for VS Code
GitHub for VS Code by Visual Studio Code
wants to access your Anjani11 account
Personal user data
Email addresses (read-only), profile information (read-only)
Repositories
Public and private
Workflow
Update GitHub Action Workflow files.
Cancel
Authorize Visual-Studio-Code
Authorizing will redirect to
https://vscode.dev
Owned & operated by GitHub
Created 5 years ago
More than 1K GitHub users


crop-prediction/Procfile at main
Authorize application
GitHub Authentication - Sign In
crop-predictionm · GitHub | Hero
dashboard.heroku.com/apps/crop-predictionm/deploy/github
Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
crop-predictionm
GitHub Anjani11/crop-prediction main
Open app
More
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Add this app to a pipeline
Create a new pipeline or choose an existing one and add this app to a stage in it.
Add this app to a stage in a pipeline to enable additional features
Pipelines let you connect multiple apps together and promote code between them. Learn more.
Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. Learn more.
Choose a pipeline
Deployment method
Heroku Git
Use Heroku CLI
GitHub
Connected
Container Registry
Use Heroku CLI
App connected to GitHub
Code diffs, manual and auto deploys are available for this app.
Connected to Anjani11/crop-prediction by Anjani11
Disconnect...
Releases in the activity feed link to GitHub to view commit diffs
Automatically deploys from main

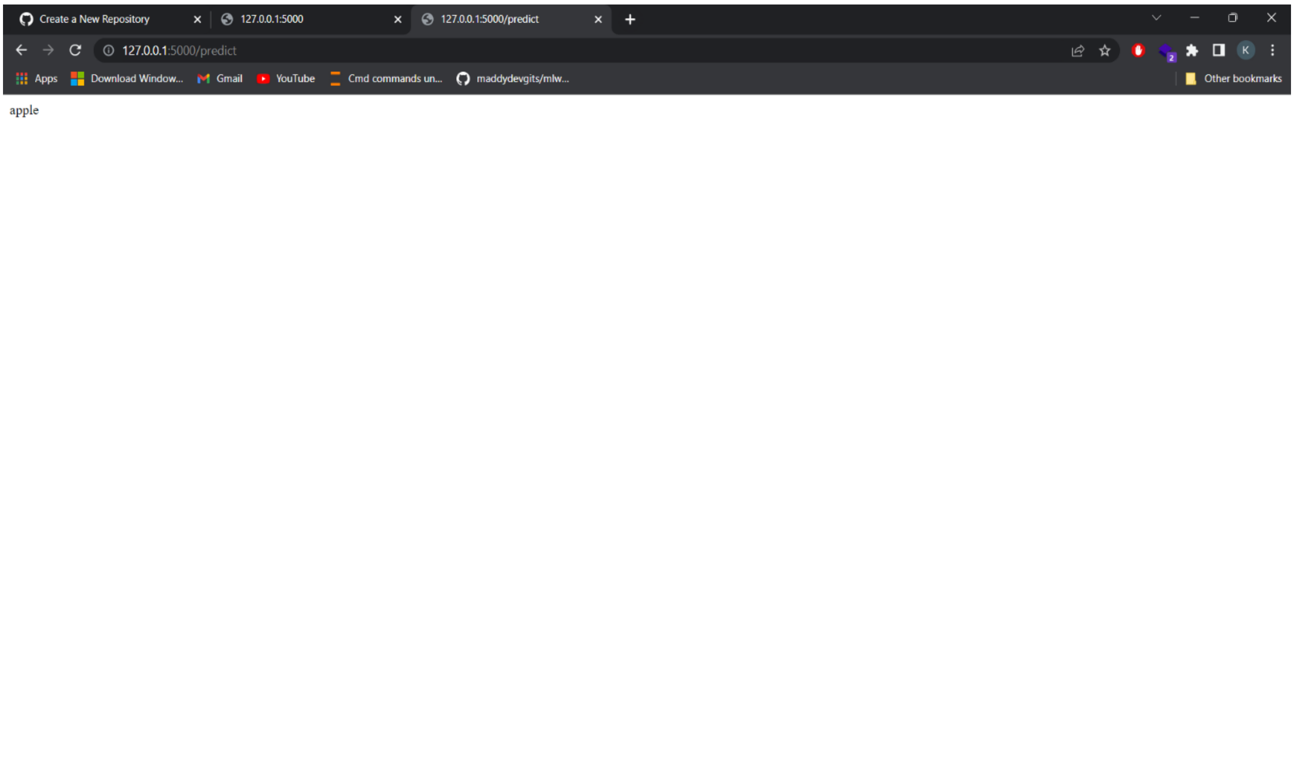

Anjani11/crop-prediction
maddydevgits/mlwec-ai-interns
crop-predictionm · GitHub | Hero
dashboard.heroku.com/apps/crop-predictionm/deploy/github
Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Automatic deploys
Enables a chosen branch to be automatically deployed to this app.
You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.
Automatic deploys from main are enabled
Every push to main will deploy a new version of this app. Deploys happen automatically: be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. Learn more.
Wait for CI to pass before deploy
Only enable this option if you have a Continuous Integration service configured on your repo.
Disable Automatic Deploys
Manual deploy
Deploy the current state of a branch to this app.
Deploy a GitHub branch
This will deploy the current state of the branch you specify below. Learn more.
Choose a branch to deploy
main
Deploy Branch
heroku.com
Blogs
Careers
Documentation
Support
Terms of Service
Privacy
Cookies
© 2022 Salesforce.com



Anjani11/crop-prediction
github.com/Anjani11/crop-prediction
Anjani11 / crop-prediction
Public
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
Anjani11 commit
3429993 2 hours ago
4 commits
templates
commit
2 hours ago
Procfile
commit
2 hours ago
README.md
first commit
3 hours ago
app.py
commited
2 hours ago
model.pkl
commited
2 hours ago
requriments.txt
commit
2 hours ago
About
No description, website, or topics provided.
Readme
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
crop-prediction



Crop Prediction System
127.0.0.1:5000
Enter Nitrogen: 50
Enter Phosphorous: 50
Enter Potassium: 50
Enter Temperature: 25
Enter Humidity: 45
Enter pH: 6.5
Enter Rainfall: 200
Submit


Create a New Repository
127.0.0.1:5000
127.0.0.1:5000/predict
127.0.0.1:5000/predict
Apps
Download Window...
Gmail
YouTube
Cmd commands un...
maddydevgits/mlw...
Other bookmarks
apple

MOVIE RECOMMENDATION SYSTEM

Introduction movie recommendation system :

- A movie recommendation system, or a movie recommender system, is an ML-based approach to filtering or predicting the users' film preferences based on their past choices and behavior.

- It's an advanced filtration mechanism that predicts the possible movie choices of the concerned user and their preferences towards a domain-specific item, aka movie.

- Reading the local TV guides, renting CDs and DVDs, watching tapes or filmstrip projectors... Today, this is all a relic of the past. The largest movie libraries in the world are all digitized and transferred to online streaming services, like Netflix, HBO, or YouTube.

- Enhanced with AI-powered tools, these platforms can now assist us with probably the most difficult chore of all — picking a movie.

- The basic concept behind a movie recommendation system is quite simple. In particular, there are two main elements in every recommender system: users and items.

- The basic concept behind a movie recommendation system is quite simple. In particular, there are two main elements in every recommender system: users and items.

- The system generates movie predictions for its users, while items are the movies themselves.

Tools :

- Google Co lab –It is one of the python idle for writing and execution of the code.

- CSV – Is used to upload the data set in the google co lab. At first the data set is saved in the from of zip file. After that we open it through the excel sheets and saved using through .csv

- KAGGLE-Is used to collect the data sets which we programming code. Here we collect two data sets as per our project need. Here we use movies and rating datasets.

Packages :

- Pandas:-pandas is used to get data frames and series. Which is also used for cleaning and analysis.

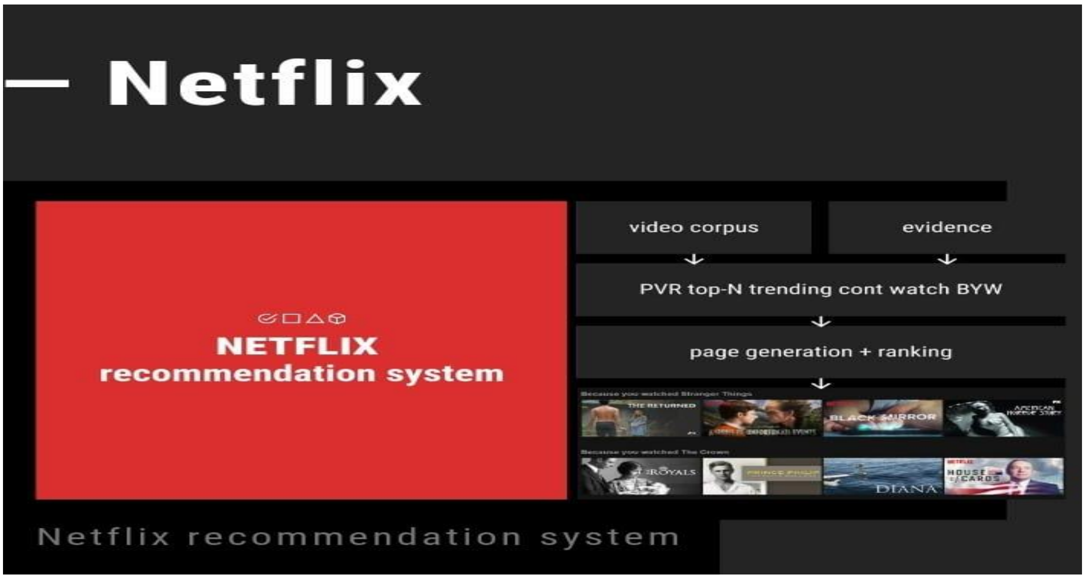
- NUMPY:-Numpy is the library of python programming language . Moreover numpy forms the foundation of the machine learning stack.

- Seaborn:-Seaborn is a data visualization library for python runs on top of the popular matplotlib data visualization library, although it provides a simple interface and aesthetically better-looking plots. In this tutorial , you will discover a gentle introduction to Seaborn data visulation.

- MATPLOT LIB:-Matplotlib is one of the ploting library in python which is however widely in use machine learning application with its numerical mathematics extension , numpy to create static , animator and interact to visualization.

Movie image :


MACHINE LEARNING PROJECT
MOVIE
Recommendation
System
Source Code Included

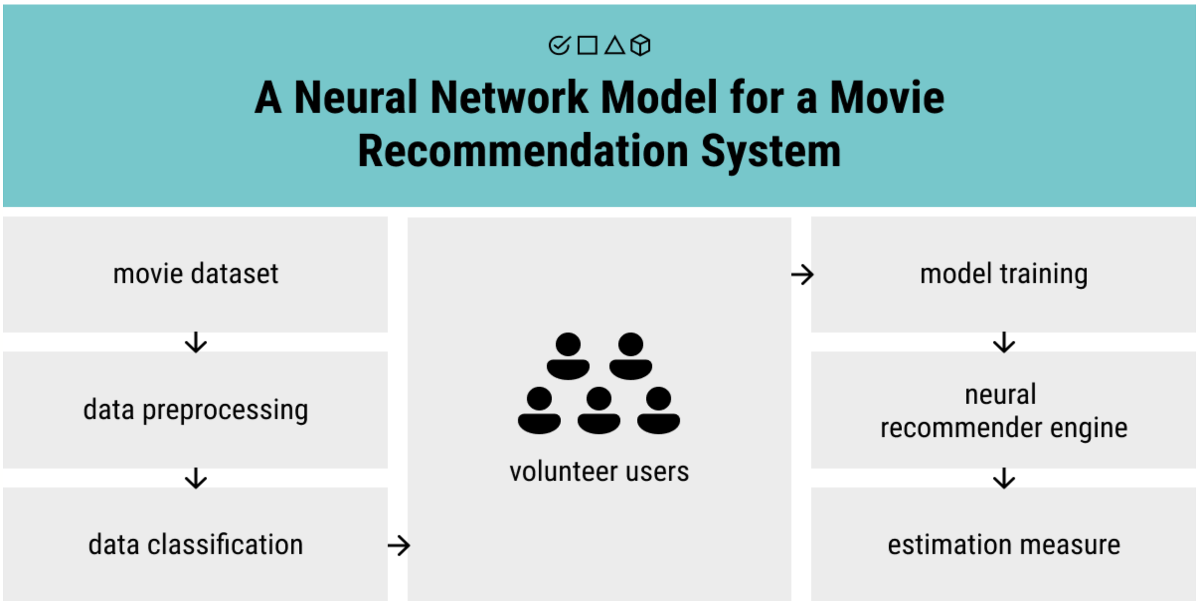

Netflix
NETFLIX
recommendation system
video corpus
evidence
PVR top-N trending cont watch BYW
page generation + ranking
Netflix recommendation system

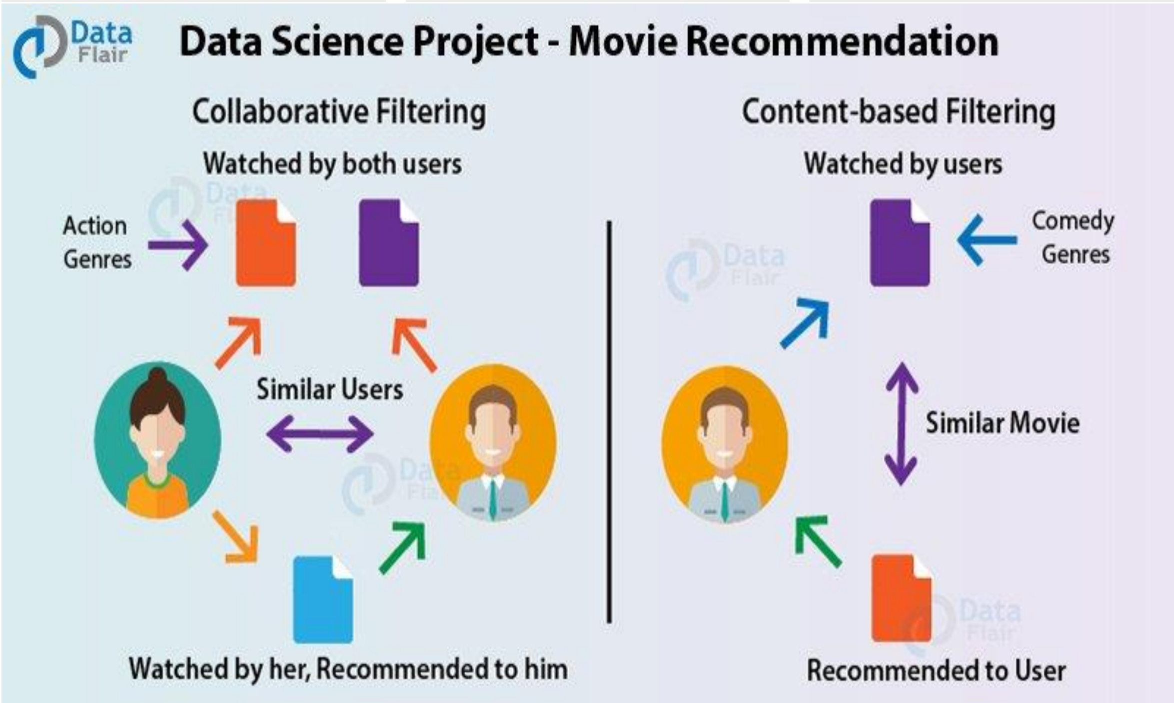
Project :

- In movie recommendation system or a movie recommender system , in a ml-based approach to filtering or predicting the uses 'film preference based on their paste choice and behaviour.

- We are leaving the age of facts coming into the age of recommendation.

- Content-based methods are based on the similarity of movie attributes. Using this type of recommender system, if a user watches one movie, similar movies are recommended. For example, if a user watches a comedy movie starring Adam Sandler, the system will recommend them movies in the same genre or starring the same actor, or both. With this in mind, the input for building a content-based recommender system is movie attributes.





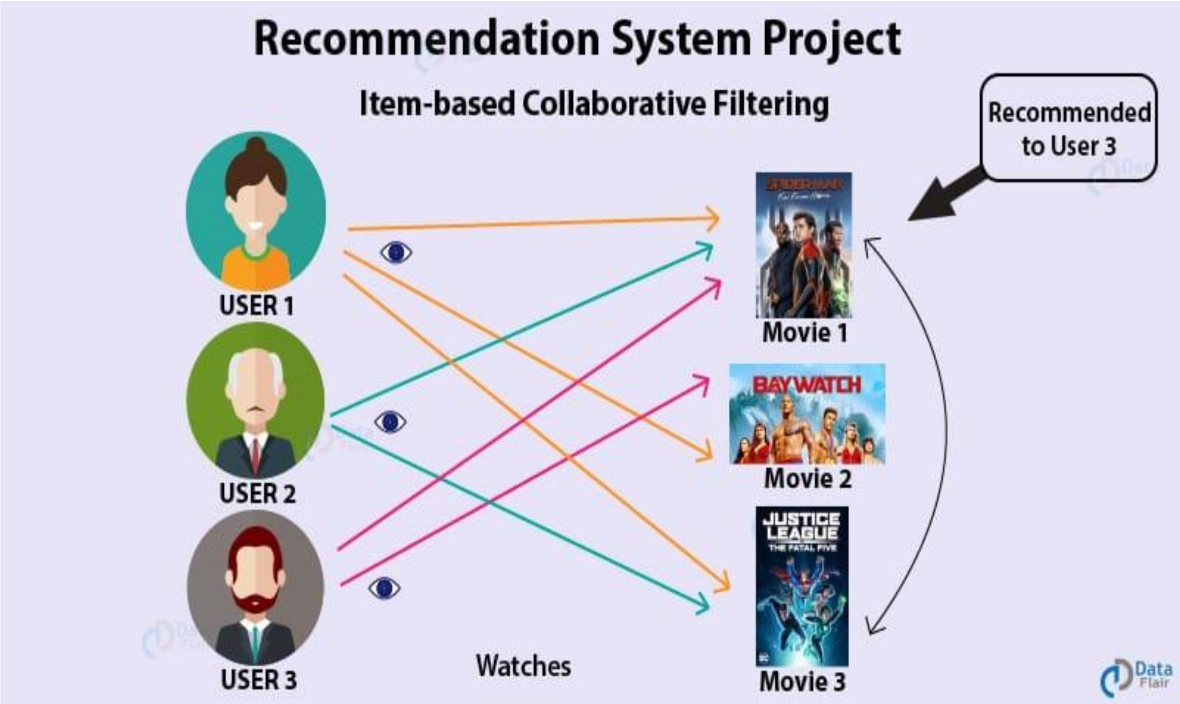
- B) Collaborative Filtering Movie Recommendation Systems

- With collaborative filtering, the system is based on past interactions between users and movies. With this in mind, the input for a collaborative filtering system is made up of past data of user interactions with the movies they watch.

- For example, if user A watches M1, M2, and M3, and user B watches M1, M3, M4, we recommend M1 and M3 to a similar user C. You can see how this looks in the figure below for clearer reference.

- This data is stored in a matrix called the user-movie interactions matrix, where the rows are the users and the columns are the movies.

- Now, let's implement our own movie recommendation system using the concepts discussed above.



Existing system :

- A movie recommendation is a system that provides suggestions to uses for certain resources books , movies, songs , etc .. based on some data set.
- Today , movie recommendation system are widely used by the most popular streaming services.
- Once again , ml proves to be a vital technological solution that makes our lives easier.
- The growth of the internet has resulted in an enormous amount of online data and information available to us.

Programming Languages :

- MACHINE LEARNING:-Machine learning is important because it gives enter prizes a view of trend in customer behaviour and business patterns , as well as support the development of new products. We use different labraries and packges.
- PYTHON:-since it’s relatively is to learn. Python is used for developed websites and software, task automation, data analysis , and data visualization. It is very easy to understand by every one.

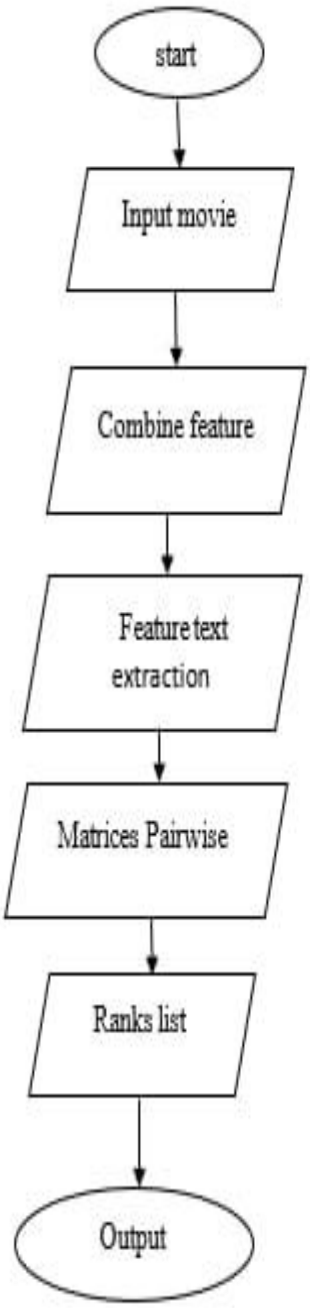
Pros of the project :

- Easy recommendations make less searches and some times end up un good deals.
- User views will give accurate information , these is also an advante if you purchase online as you can see other reviews too most of the time honest.
- Speed up the process of decision purchase based on the previous statistics.
- The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

cons of the project :

- Which the system recommends products with bias , then customer will be landing into wrong deals.
- Chances are that some websites may suggest products wrongly based on analysis of little information gathering.
- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
- The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

Block Diagram :



Working procedure :

The approach to build the movie recommendation engine consists of the following steps.

- Perform Exploratory Data Analysis (EDA) on the data
- Build the recommendation system
- Get recommendation.
- The dataset contains two CSV files, credits, and movies. The credits file contains all the metadata information about the movie and the movie file contains the information like name and id of the movie, budget, languages in the movie that has been released, etc.
- We only need the id, title, cast, and crew columns of the credits Data Frame. Let's merge the data frames into one on the column 'id'.
- The accuracy of predictions made by the recommendation system can be personalized using the "plot/description" of the movie.
- But the quality of suggestions can be further improved using the metadata of movie. Let's say the query to our movie recommendation engine is "The Dark Knight Rises". Then the predictions should also include movies directed by the director of the film. It should also include movies with the cast of the given query movie.
- For that, we utilize the following features to personalize the recommendation: cast, crew, keywords, genres.
- The movie data is present in the form of lists containing strings, we need to convert the data into a safe and usable structure. Let's apply the literal eval() function to the features.
- Get the index of the movie using the title
  .
- Get the list of similarity scores of the movies concerning all the movies.
- Enumerate them (create tuples) with the first element being the index and the second element is the cosine similarity score.
- Sort the list of tuples in descending order based on the similarity score.
- Get the list of the indices of the top 10 movies from the above sorted list. Exclude the first element because it is the title itself.
- Map those indices to their respective titles and return the movies

Program Code :

- First of all we will going to import all the required libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

- We will going to take two dataset first one is movie dataset in which we will have movie names and the second dataset will have ratings and userID

```
[ ] df = pd.read_csv('movies.csv')

    # first few rows of dataset
    df.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
df.shape
(27278, 3)
```

- So we have 27,278 movies. We don't need genres column so we are dropping that column

+ Code + Text

```
[ ] # drop genres column
    df.drop(['genres'],axis=1,inplace=True)
```

❖ We need to import our another dataset in which we have ratings for the movie

```
[ ] # import rating dataset
    rating = pd.read_csv("rating.csv")

    # columns
    rating.columns

    Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

```
[ ] # we need user id, movie id and rating
    rating = rating.loc[:,["userId","movieId","rating"]]
    rating.head()
```

	userId	movieId	rating
0	1	2	3.5
1	1	29	3.5
2	1	32	3.5
3	1	47	3.5
4	1	50	3.5

```
[ ] #then merge movie and rating data
    df = pd.merge(df,rating)
```

```
df.head()
```

```
[ ] df.head()
```

	movieId	title	userId	rating
0	1	Toy Story (1995)	3	4.0
1	1	Toy Story (1995)	6	5.0
2	1	Toy Story (1995)	8	4.0
3	1	Toy Story (1995)	10	4.0
4	1	Toy Story (1995)	11	4.5

- As noted here one user has rated one or more than one movie. This means that one movie has been rated by more than one user.

```
df.shape
```

```
(1048575, 4)
```

- In our kaggle kernel,we don't have much memory so we have to subset our dataset,we are going to take 1M rows

```
df = df.iloc[:1000000]
```

```
df.shape
```

```
(1000000, 4)
```

```
# basic stats
df.describe()
```

	movieId	userId	rating
count	1000000.000000	1000000.000000	1000000.000000
mean	5226.221078	3527.782357	3.528205
std	10891.738671	2020.573993	1.053749
min	1.000000	1.000000	0.500000
25%	799.000000	1811.000000	3.000000
50%	2012.000000	3540.000000	4.000000
75%	4019.000000	5240.000000	4.000000
max	59103.000000	7120.000000	5.000000

- **Data Visualization**
- Let's find the average rating of each movie
-

```
df.groupby("title").mean()['rating'].sort_values(ascending=False)
```

```
title
Jubilee (1977)                                                5.0
Rhyme & Reason (1997)                                         5.0
Swann in Love (Un amour de Swann) (1984)                      5.0
No End (Bez konca) (1985)                                     5.0
Bar Girls (1994)                                              5.0
                                                             ...
Bloody Angels (1732 Høtten: Marerittet har et postnummer) (1998)    0.5
Bloody Mama (1970)                                            0.5
Double Trouble (1967)                                         0.5
Dust Devil (1992)                                             0.5
Venom (1982)                                                  0.5
Name: rating, Length: 10359, dtype: float64
```

❖ Let's find the number of rating a particular movie has received

```
df.groupby("title").count()["rating"].sort_values(ascending=False)
```

```
title
Pulp Fiction (1994)                                          3498
Forrest Gump (1994)                                          3476
Silence of the Lambs, The (1991)                             3247
Shawshank Redemption, The (1994)                             3216
Jurassic Park (1993)                                         3129
                                                             ...
Full Moon in Paris (Les nuits de la pleine lune) (1984)        1
Funny About Love (1990)                                        1
Furies, The (1950)                                             1
Further Gesture, A (1996)                                      1
Krakatoa, East of Java (1969)                                  1
Name: rating, Length: 10359, dtype: int64
```

❖ Now we will going to make a datafame in which we will have rating and number of ratings column

```
ratings=pd.DataFrame(df.groupby("title").mean()['rating'])
ratings['number of ratings']=pd.DataFrame(df.groupby("title").count()["rating"])
print(ratings.head())
```

```
                            rating  number of ratings
title
'Round Midnight (1986)    3.785714                  7
'Salem's Lot (2004)       2.714286                  7
'Til There Was You (1997) 2.881579                 38
'burbs, The (1989)        3.042945                163
'night Mother (1986)      3.166667                 12
```

```
[ ] ratings.sort_values(by='rating', ascending=False)
```

title	rating	number of ratings
Jubilee (1977)	5.0	1
Rhyme & Reason (1997)	5.0	1
Swann in Love (Un amour de Swann) (1984)	5.0	1
No End (Bez konca) (1985)	5.0	1
Bar Girls (1994)	5.0	1
...	...	...
Bloody Angels (1732 Høtten: Marerittet har et postnummer) (1998)	0.5	1
Bloody Mama (1970)	0.5	1
Double Trouble (1967)	0.5	1
Dust Devil (1992)	0.5	1
Venom (1982)	0.5	1

10359 rows × 2 columns

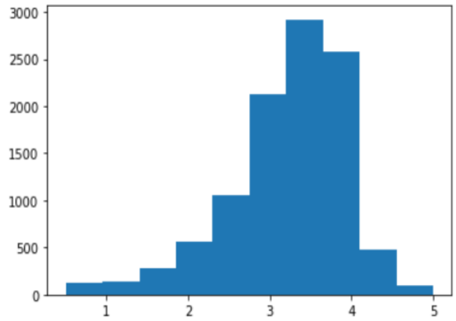
```
ratings.describe()
```

	rating	number of ratings
count	10359.000000	10359.000000
mean	3.223734	96.534415
std	0.740856	249.172573
min	0.500000	1.000000
25%	2.857143	4.000000
50%	3.363636	15.000000
75%	3.738095	67.000000
max	5.000000	3498.000000

- So from above we can see that we don't have any movie with rating 5

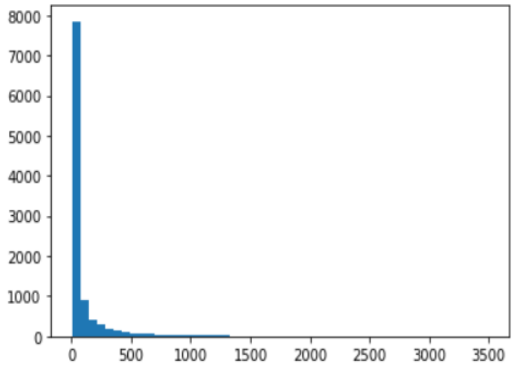
```
plt.hist(ratings['rating'])
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```




```
plt.hist(ratings['number of ratings'],bins=50)
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



❖ Recommender System

```
# lets make a pivot table in order to make rows are users and columns are movies. And values are rating
pivot_table = df.pivot_table(index = ["userId"],columns = ["title"],values = "rating")
pivot_table.head(5)
```

title	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'burbs, The (1989)	'night Mother (1986)	*batteries not included (1987)	...All the Marbles (California Dolls, The) (1981)	...And God Spoke (1993)	...And Justice for All (1979)	.45 (2006)	...	Zus & Zo (2001)	[REC] (2007)	\\"Great Performances\"\" Cats (1998)"	eXistenZ (1999)	loudQUIETloud: A Film About the Pixies (2006)	xXx (2002)	xXx: State of the Union (2005)	¡ Am (:
userId																			
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 10359 columns

```
pivot_table.shape
```

```
(7120, 10359)
```

❖ Now we will make a function which will recommend the movie with their correlation score.Note that higher the correlation more the movie related to each other

```
def recommend_movie(movie):
  movie_watched = pivot_table[movie]
  similarity_with_other_movies = pivot_table.corrwith(movie_watched)#find correlation between"TOYstory(1995)"and other movies
  similarity_with_other_movies = similarity_with_other_movies.sort_values(ascending=False)
  return similarity_with_other_movies.head()
```

```
recommend_movie('American President, The (1995)')
```

```
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2683: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.7/dist-packages/numpy/lib/function_base.py:2542: RuntimeWarning: divide by zero encountered in true_divide
  c *= np.true_divide(1, fact)
title
Maybe, Maybe Not (Bewegte Mann, Der) (1994)    1.0
Ringu 2 (Ring 2) (1999)                        1.0
Anything Else (2003)                           1.0
Man on a Ledge (2012)                          1.0
Inside Llewyn Davis (2013)                     1.0
dtype: float64
```

```
recommend_movie('Toy Story (1995)')
```

```
title
Dopamine (2003)                                     1.0
Twilight Saga: Breaking Dawn - Part 2, The (2012)   1.0
White Diamond, The (2004)                           1.0
Concert, Le (2009)                                  1.0
Mr. Popper's Penguins (2011)                        1.0
dtype: float64
```

Conclusion Remarks:

- Once again, ML proves to be a vital technological solution that makes our lives easier.
- And the more these systems evolve, the more advanced ML techniques we have at our disposal that generate the most accurate content for users and give them what they are looking for.
- Want to try creating a movie recommendation system on your own? Make sure you have the best quality labeled movie datasets with our professional team at **Label Your Data**.
- Today, movie recommendation systems are widely used by the most popular streaming services, enabling a more personalized experience and increased user satisfaction across the platforms.
- It's estimated that the world cinema has released more than 500,000 movies — a number beyond one person's control.
- With such an enormous number of motion pictures to choose from, developing and improving recommendation systems with ML was a crucial step to make this process easier and feasible.
- The bigger the choice, the harder it is to make the final decision.
- This is especially true for modern movie fans, who have thousands of movies to pick from.
- But thanks to machine learning, we now have recommendation systems based on its complex algorithms and techniques.
- By the nature of our system, it is not an easy task to evaluate the performance since there is no right or wrong recommendation; it is just a matter of opinions.
- Based on informal evaluations that we carried out over a small set of users we got a positive response from them.
- We would like to have a larger data set that will enable more meaningful results using our system.
- Additionally, we would like to incorporate different machine learning and clustering algorithms and study the comparative results.
- In this paper we have introduced Movie recommendation, a recommender system for movie recommendation.

- Eventually we would like to implement a web-based user interface that has a user database, and has the learning model tailored to each user.

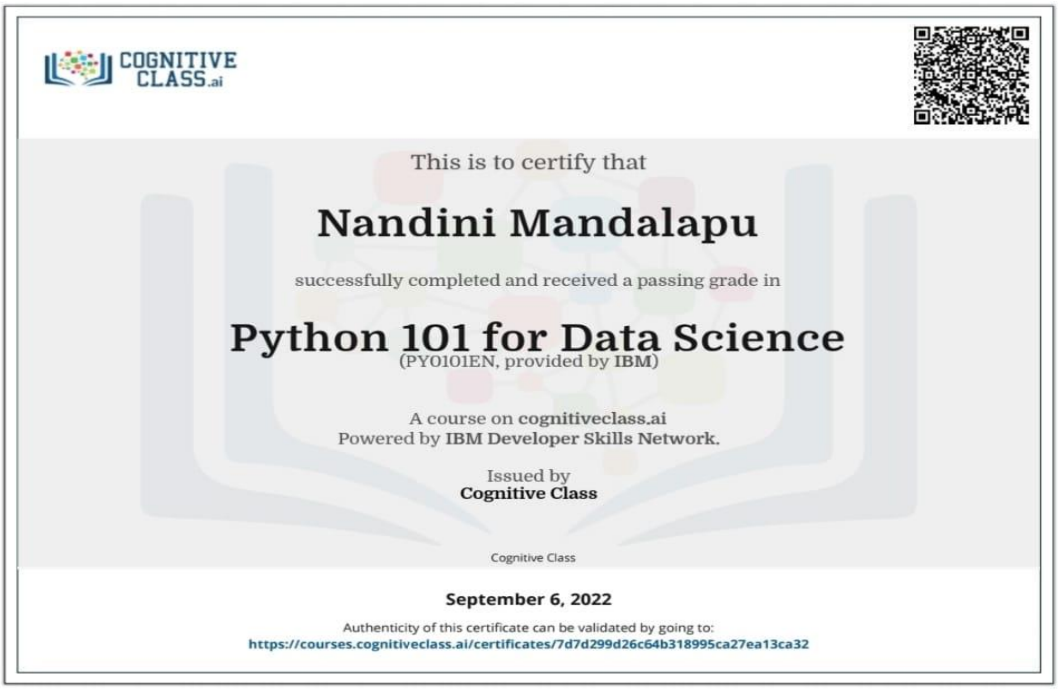
- We have implemented a recommendation system based on content-based filtering and collaborative filtering.

- Cold start problem in the dataset is addressed by adding 0 rating.

- The proposed system used 13 features consisting of user information, movie information and predicted top-10 movie that are similar to user interests using content based and collaborative based filtering.

- Our future work will be implementing the recommendation system using deep learning algorithm and analyse the improvement in the accuracy of the system.

- In this machine learning project, we build movie recommendation systems. We built a content-based recommendation engine that makes recommendations given the title of the movie as input.



COGNITIVE CLASS.ai

This is to certify that

Nandini Mandalapu

successfully completed and received a passing grade in

Python 101 for Data Science

(PY0101EN, provided by **IBM**)

A course on **cognitiveclass.ai**
Powered by **IBM Developer Skills Network.**

Issued by
Cognitive Class

Cognitive Class

September 6, 2022

Authenticity of this certificate can be validated by going to:
https://courses.cognitiveclass.ai/certificates/7d7d299d26c64b318995ca27ea13ca32

In recognition of the commitment to achieve professional excellence

PYTHON FOR DATA SCIENCE
IBM.
COGNITIVE CLASS

Nandini Mandalapu

Has successfully satisfied the requirements for:

Python for Data Science

Issued on: 12 SEP 2022
Issued by IBM

Verify: https://www.credly.com/go/NyT9foCz

IBM.

THE BEST

CERTIFICATE

OF INTERNSHIP

MAKE SKILLED

EARN THE SKILL WITH MAKING

PROUDLY PRESENTED TO

M Nandini

of Malineni Lakshmaiah Women's Engineering College for successful completion of Internship during the dates of **29 August - 14 September** 2022 and submitted the project with outstanding performance.

P. Mad

MADHU PARVATHANENI
CEO,Make Skilled