MACHINE LEARNING INTERNSHIP

Under the guidance of Ravi kumar sir

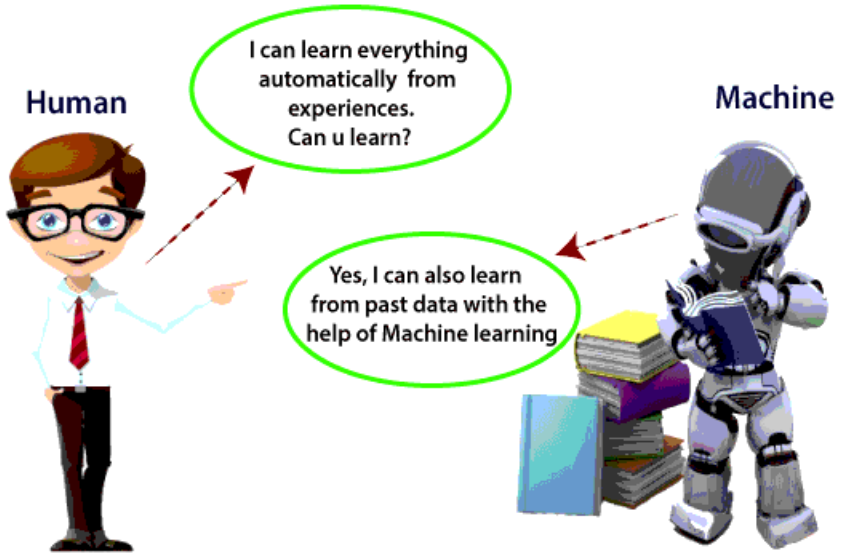
NIT WARANGAL

TABLE OF CONTENTS:

INTRODUCTION:

MACHINE LEARNING:

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system.



Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959.

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without

being explicitly programmed.

Classification of Machine Learning:

At a broad level, machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

1) Supervised Learning

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

The goal of supervised learning is to map input data with the output data. The supervised learning is based on supervision, and it is the same as when a student learns things in the supervision of the teacher. The example of supervised learning is spam filtering.

Supervised learning can be grouped further in two categories of algorithms:

* **Classification**
* **Regression**

2) Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning

is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:

- **Clustering**
- **Association**

Regression Analysis in Machine learning

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as temperature, age, salary, price, etc.

Types of Regression:

There are various types of regressions which are used in data science and machine learning. Each type has its own importance on different scenarios, but at the core, all the regression methods analyze the effect of the independent variable on dependent variables.

- Linear Regression
- Logistic Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression

* Random Forest Regression
* Ridge Regression
* Lasso Regression:

Linear Regression:

Linear regression is a statistical regression method which is used for predictive analysis.

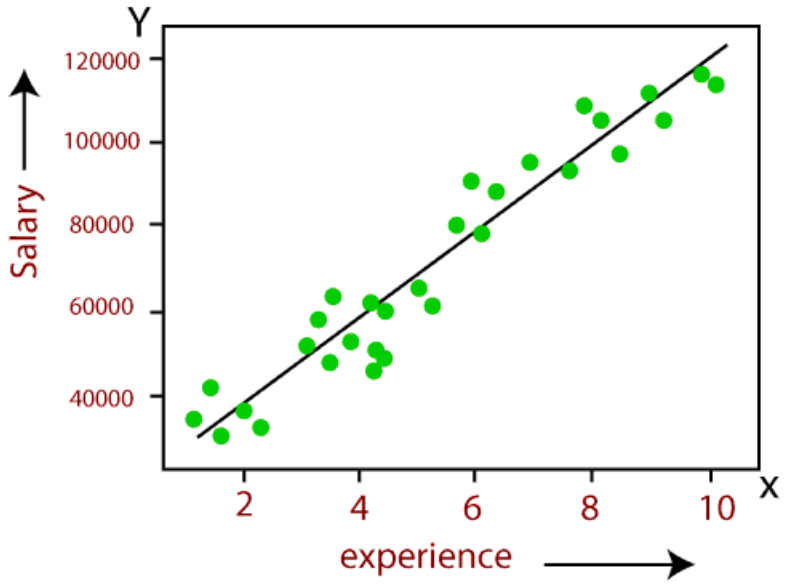
It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.

It is used for solving the regression problem in machine learning.

Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.

If there is only one input variable (x), then such linear regression is called simple linear regression. And if there is more than one input variable, then such linear regression is called multiple linear regression.

The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of the year of experience.



Mathematical equation for Linear regression:

Y= aX+b

Here, Y = dependent variables (target variables),

X= Independent variables (predictor variables),

a and b are the linear coefficients

Multiple Linear Regression:

Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable.

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

Polynomial Regression:

Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth

degree polynomial.

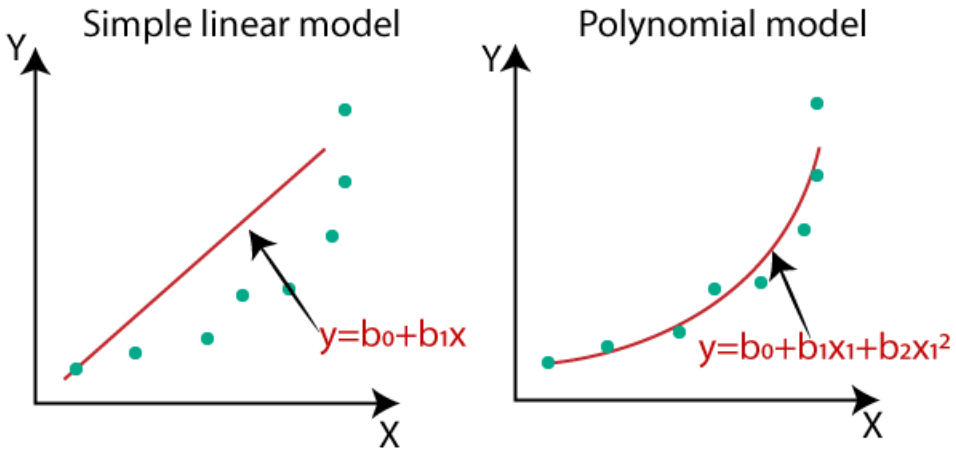
The Polynomial Regression equation

y= b0+b1x1+ b2x12+ b2x13+..... bnx1n

* It is also called the special case of Multiple Linear Regression in ML. Because we add some polynomial terms to the Multiple Linear regression equation to convert it into Polynomial Regression.
* It is a linear model with some modification in order to increase the accuracy.
* The dataset used in Polynomial regression for training is of non-linear nature.
* It makes use of a linear regression model to fit the complicated and non-linear functions and datasets.
* Hence, "In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modeled using a linear model."

Need for Polynomial Regression:

* If we apply a linear model on a linear dataset, then it provides us a good result as we have seen in Simple Linear Regression, but if we apply the same model without any modification on a non-linear dataset, then it will produce a drastic output. Due to which loss function will increase, the error rate will be high, and accuracy will be decreased.
* So for such cases, where data points are arranged in a non-linear fashion, we need the Polynomial Regression model. We can understand it in a better way using the below comparison diagram of the linear dataset and non-linear dataset.



Here we can attach the tasks are taught:

- **Multiple regression**
- **Polynomial regression**
- **Linear regression**

Linear regression:

Code:

 # -*- coding: utf-8 -*-

"""

Created on Mon May 11 12:43:35 2020

@author: manisha

#importing the libraries

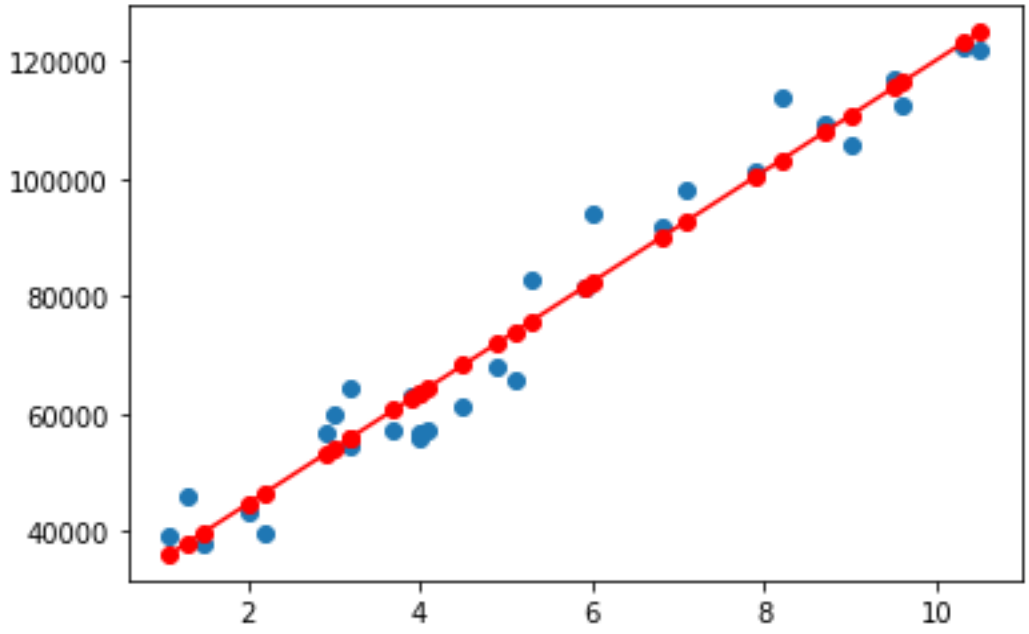
import numpy as np

import matplotlib.pyplot as plt

```
import pandas as pd

dataset=pd.read_csv('C:/Users/manisha/Desktop/nit/Salary_Data.csv')

x=dataset.iloc[:,0].values

y=dataset.iloc[:,1].values

n=np.size(x)

m_x,m_y=np.mean(x),np.mean(y)

nr=np.sum((x-m_x)*(y-m_x))

dr=np.sum((x-m_x)**2)

b0_1=nr/dr

b0_0=m_y-b0_1*m_x

y_pred=b0_0+b0_1*x

print("intercept-",b0_0)

print("slope-",b0_1)

#ploting the line

plt.scatter(x,y)

plt.plot(x,y_pred,color='r',marker='o')

from sklearn.metrics import r2_score

score=r2_score(y,y_pred)

print("R_Square-",score)
```

plt.show()

output:



Polynomial regression;

code:

 # -*- coding: utf-8 -*-

"""

Created on Wed May 13 12:19:01 2020

@author: mammu

"""import numpy as np

import matplotlib.pyplot as plt

```
import pandas as pd
#import the data set
dataset=pd.read_csv('C:/Users/manisha/Desktop/nit/Position_Salaries.csv')
x=dataset.iloc[:, 1:2].values
y=dataset.iloc[:, 2:].values
x=x.reshape(10,1)
y=y.reshape(10,1)
#fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(x,y)
#fitting polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=2)
x_poly=poly_reg.fit_transform(x)
poly_reg.fit(x_poly,y)
lin_reg_2=LinearRegression()
lin_reg_2.fit(x_poly,y)
```

```
#visualising the linear regression results
plt.scatter(x,y,color='red')
plt.plot(x, lin_reg.predict(x),color='blue')

plt.title('Linear Regression')
plt.xlabel('position level')
plt.ylabel('salary')
plt.show()

plt.scatter(x,y,color='red')
plt.plot(x,lin_reg_2.predict(poly_reg.fit_transform(x)),color='blue
')
plt.title('poltnomial level')
plt.xlabel('salary')
plt.show()

x_grid=np.arange(min(x),max(x),0.1)
x_grid=x_grid.reshape(len(x_grid),1)
plt.scatter(x,y,color='red')
plt.plot(x_grid,lin_reg_2.predict(poly_reg.fit_transform(x_grid)))
```

plt.show()

sample=[11,4.36]

sample=np.array(sample)

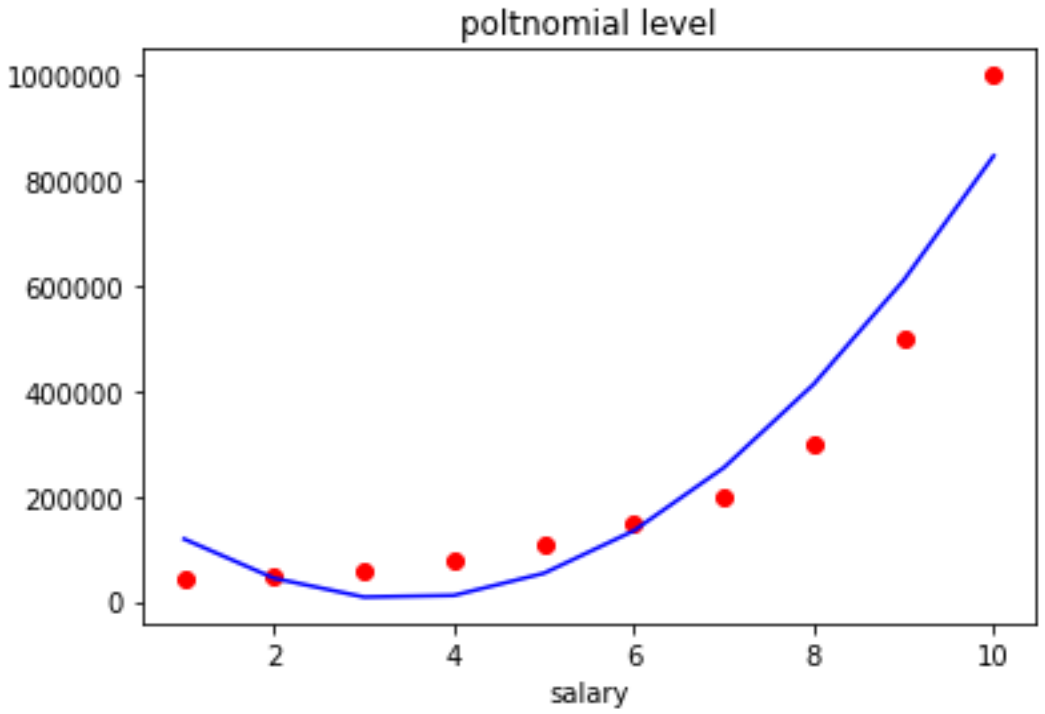
sample=sample.reshape(2,1)

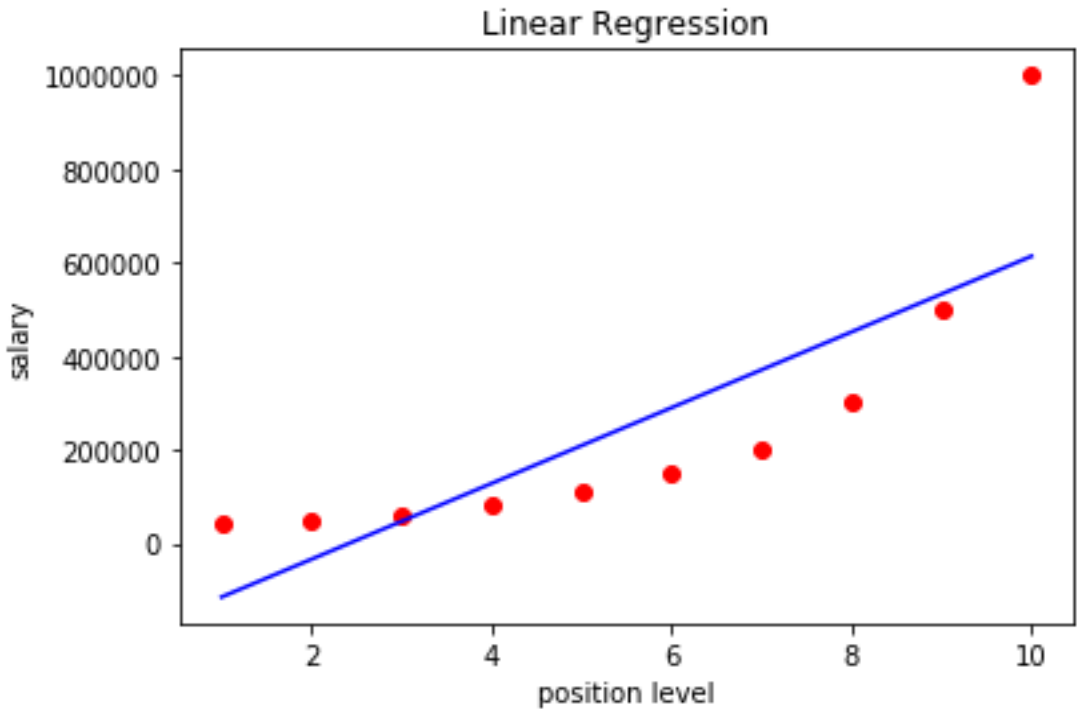
sample.reshape(1,2)

print("lin_reg results for level 6.5,4.36",lin_reg.predict(sample))

print("polynomial regression results for level6.5,4.36",lin_reg_2.predict(poly_reg.fit_transform(sample)))

output:





Multiple regression:

code:

```
# -*- coding: utf-8 -*-
"""
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd


dataset=pd.read_csv('C:/Users/manisha/Desktop/nit/Position_Salaries.csv')
```

```
x= dataset.iloc[:, :4].values

y= dataset.iloc[:, 4].values


from sklearn.preprocessing import
LabelEncoder,OneHotEncoder

LabelEncoder=LabelEncoder()

x[:, 1]=LabelEncoder.fit_transform(x[:, 3])


OneHotEncoder=OneHotEncoder(categories='auto')

x=OneHotEncoder.fit_transform(x).toarray()

x=x[:, 1:]

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,ran
dom_state=0)


#feature scaling

""" from sklearn.preprocessing import StandardScaler

sc_X=StandardScaler()

x_train=sc_X.fit_transform(x_test)

sc_y=StandardScaler()
```

y_train

"""

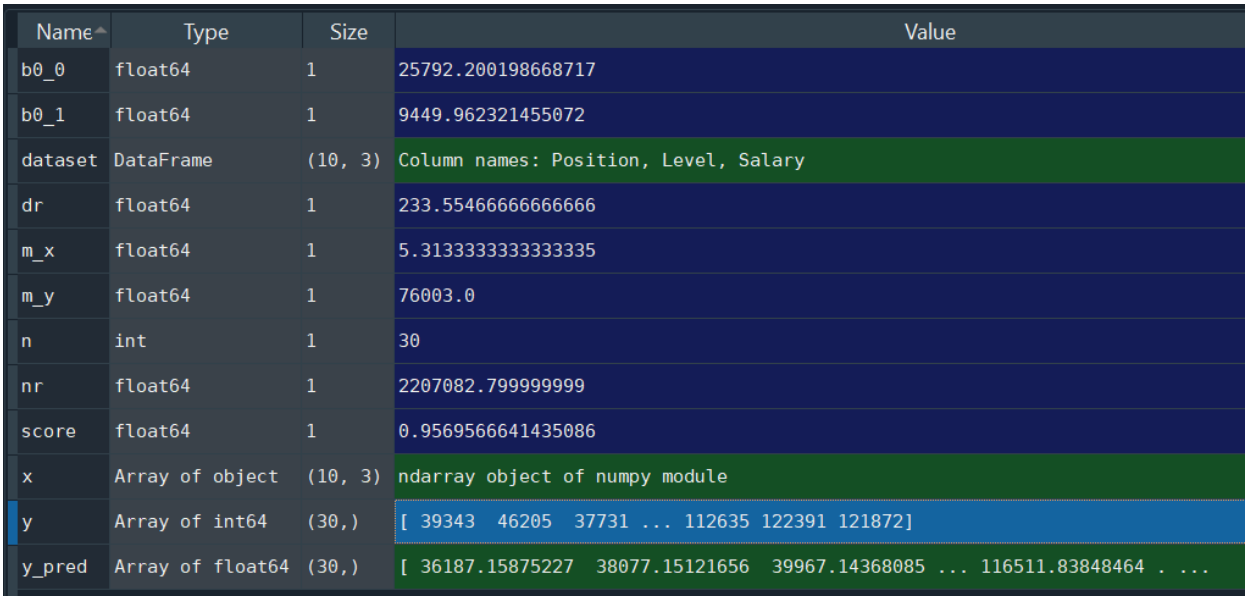
from sklearn.linear_model import LinearRegression

regressor=LinearRegression()

regressor.fit(x_train,y_train)

y_pred=regressor.predict(x_test)

output:



Name	Type	Size	Value
b0_0	float64	1	25792.200198668717
b0_1	float64	1	9449.962321455072
dataset	DataFrame	(10, 3)	Column names: Position, Level, Salary
dr	float64	1	233.55466666666666
m_x	float64	1	5.3133333333333335
m_y	float64	1	76003.0
n	int	1	30
nr	float64	1	2207082.799999999
score	float64	1	0.9569566641435086
x	Array of object	(10, 3)	ndarray object of numpy module
y	Array of int64	(30,)	[39343 46205 37731 ... 112635 122391 121872]
y_pred	Array of float64	(30,)	[36187.15875227 38077.15121656 39967.14368085 ... 116511.83848464 . ...

So here we can attach the task they're given

Real-time estimation and prediction of mortality caused by COVID-19 with patient information based algorithm.

Introduction:-

The mortality rate is the most important factor that determines whether a highly infectious disease becomes a public concern and carries risks causing a pandemic. Different virus epidemics take place throughout the world every year, but only a few rise to the level of public concern

COVID-19 caused by a coronavirus (2019-nCoV) brought world-wide attention and caused public panic because many deaths had been reported without being put in the context of the many mild infections and its potentially low case fatality rate

The high infection rate is certain, the mortality rate of COVID-19 has not been definitively determined. It is reasonable to suspect that the deaths of six of the first 41 patients (15%) in Wuhan (Huang et al., 2020) in the earliest reports by Chinese scholars were inaccurate. When the initial mortality rates were reported, only patients who were critically ill were included. Patients with mild symptoms, as well as those with asymptomatic infections, were not analyzed

Methodology:

We have to do this take the dataset and plot the values to the graph

- Import modules
- And plot the graph through dataset
- So this process will be done on 5 states like
- Andhra

Pradesh,Telangana,Maharastra,Karnataka,Tamilnadu

Algorithm:

Actually we used polynomial regression but its failed to implement the model

yes we would do the modle in Jupiter notebook .

We write code separately for the each state and predict how many cases per day by day We use bar graph and line plot and scatter plot to plot the alogorithm.

This is the process in making the model.

We take the csv file and use the data and plotted.no regression is used for now.

Code:-

```
Results:#  """

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

x = [67,200,37,25,58]

y = ['20may','21may','22may','23may','24may']

plt.title("CORANA CASES",fontsize=15)

plt.xlabel("ANDHRA PRADESH",fontsize=12)
```

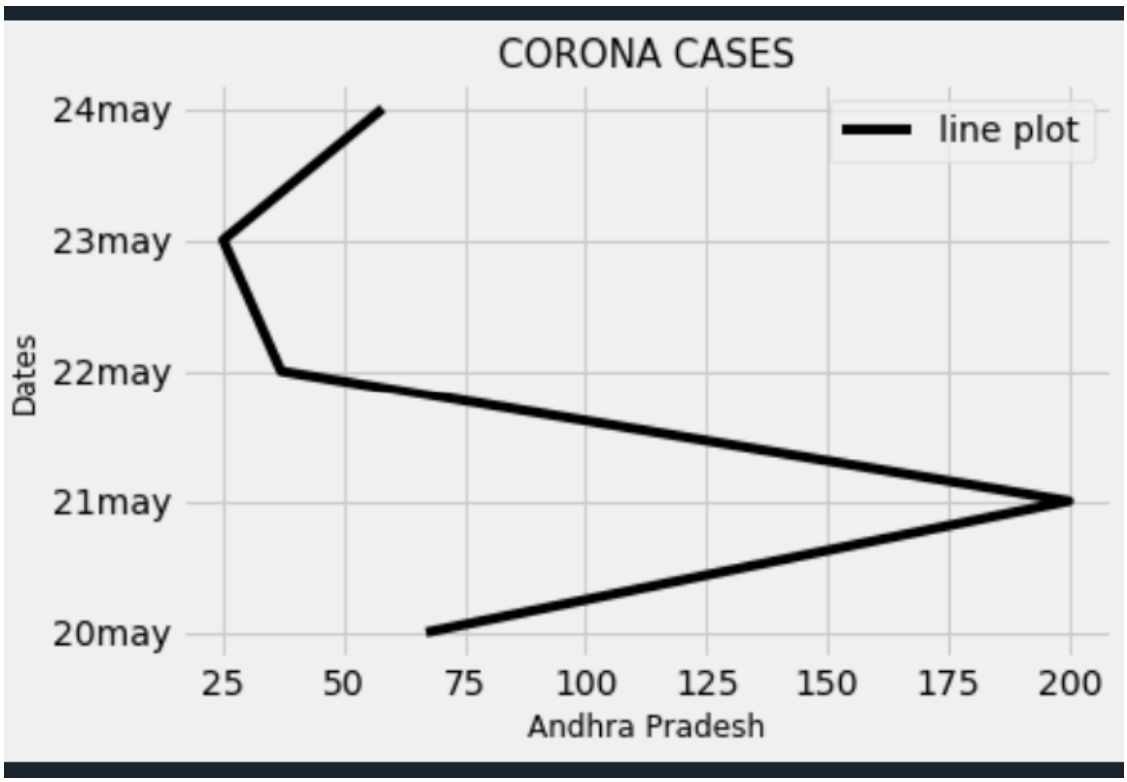
plt.ylabel("DATES",fontsize=12)

plt.plot(x,y,color="black",label="line plot")

plt.legend(loc=1)

plt.show()

output:



code:

```
x = [57,168,67,34,34]

y = ['20may','21may','22may','23may','24may']

plt.title("CORANA CASES",fontsize=25)

plt.xlabel("TELANGANA",fontsize=12)
```

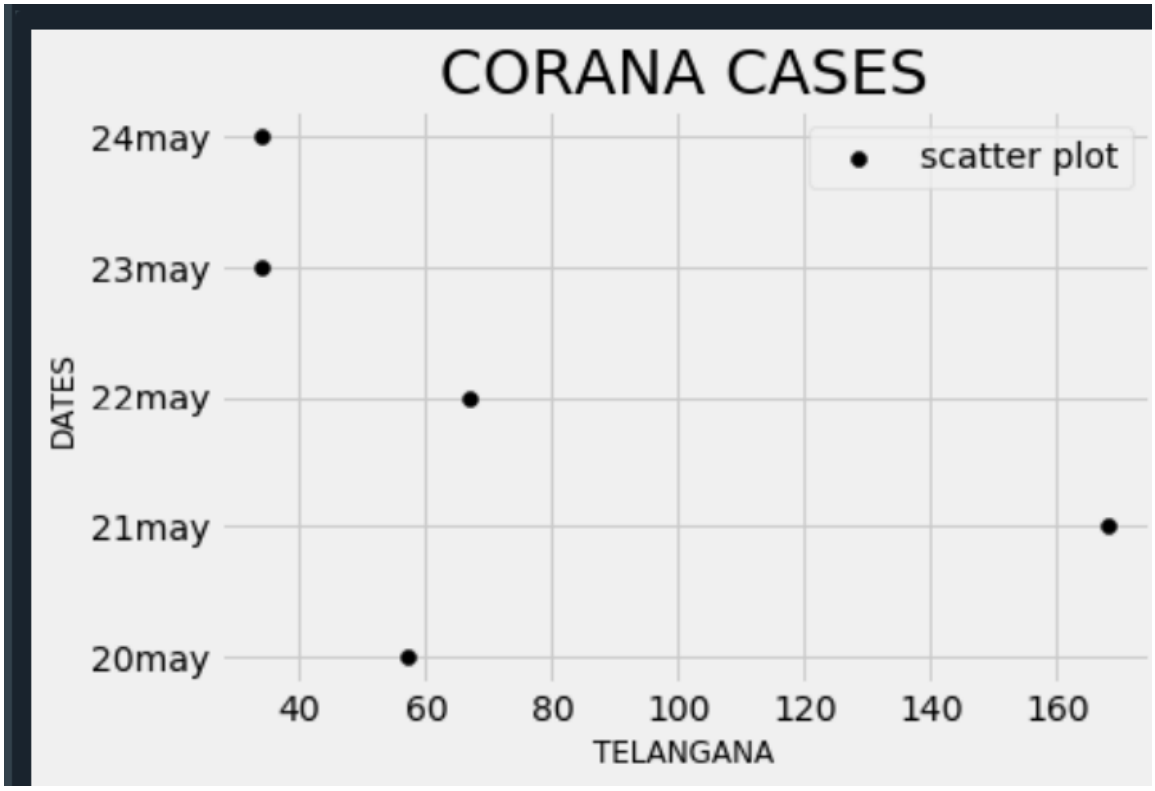
plt.ylabel("DATES",fontsize=12)

plt.scatter(x,y,color="black",label="scatter plot")

plt.legend(loc=1)

plt.show()

output:



code:

x = [57,168,67,34,34]

y = ['20may','21may','22may','23may','24may']

plt.title("CORANA CASES",fontsize=25)

plt.xlabel("MAHARASHTRA",fontsize=12)

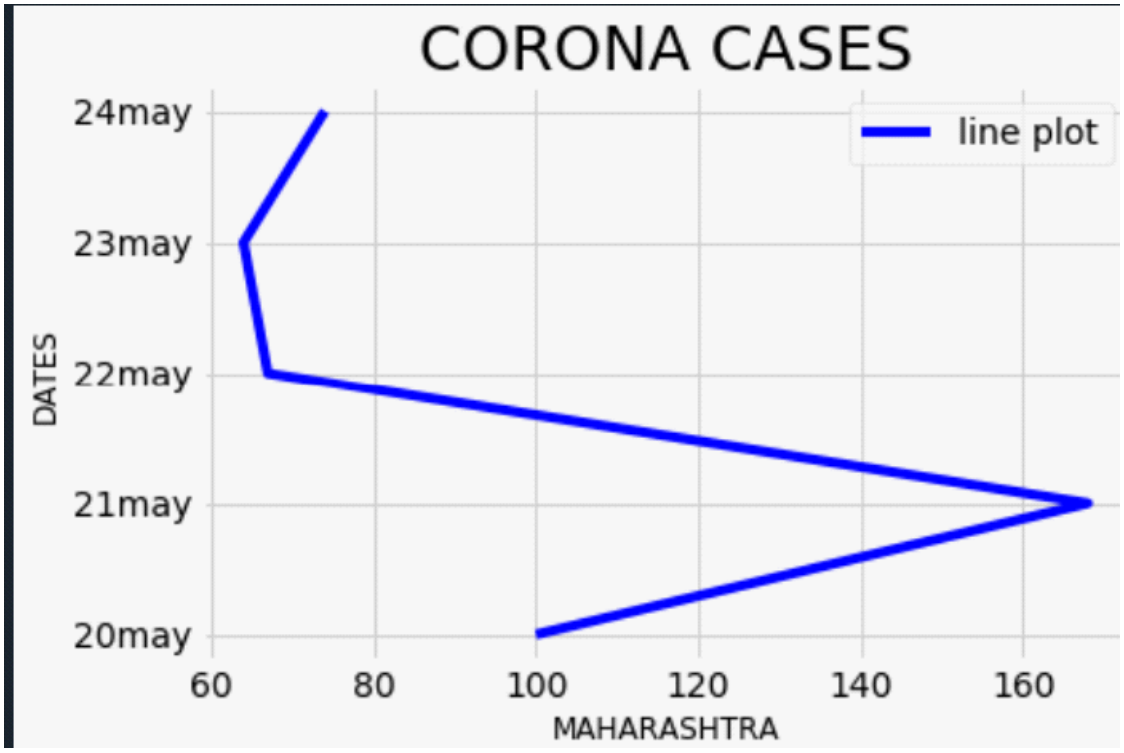
plt.ylabel("DATES",fontsize=12)

plt.scatter(x,y,color="black",label="scatter plot")

plt.legend(loc=1)

plt.show()

output;



CODE;

x = [57,168,67,34,34]

y = ['20may','21may','22may','23may','24may']

plt.title("CORANA CASES",fontsize=25)

plt.xlabel("KARNATAKA",fontsize=12)

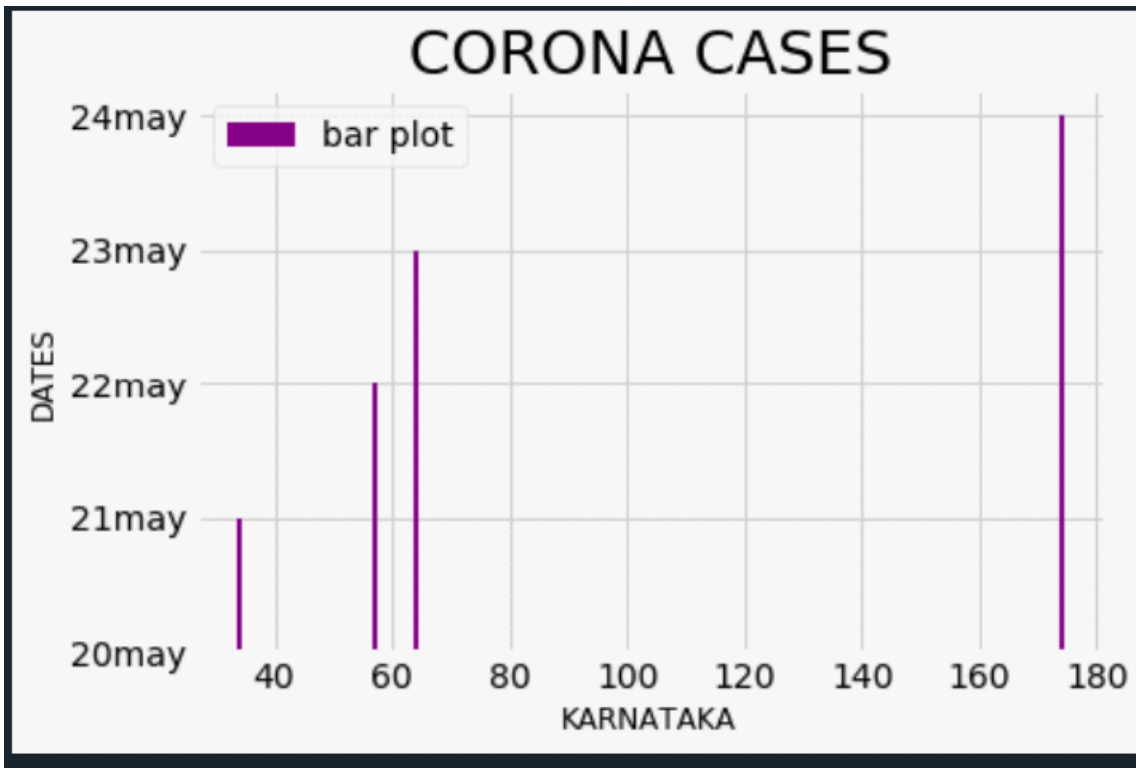
plt.ylabel("DATES",fontsize=12)

plt.scatter(x,y,color="black",label="scatter plot")

plt.legend(loc=1)

plt.show()

output:



CODE:

x = [150,100,87,124,200]

y = ['20may','21may','22may','23may','24may']

plt.title("CORANA CASES",fontsize=25)

plt.xlabel("TAMILNADU",fontsize=12)

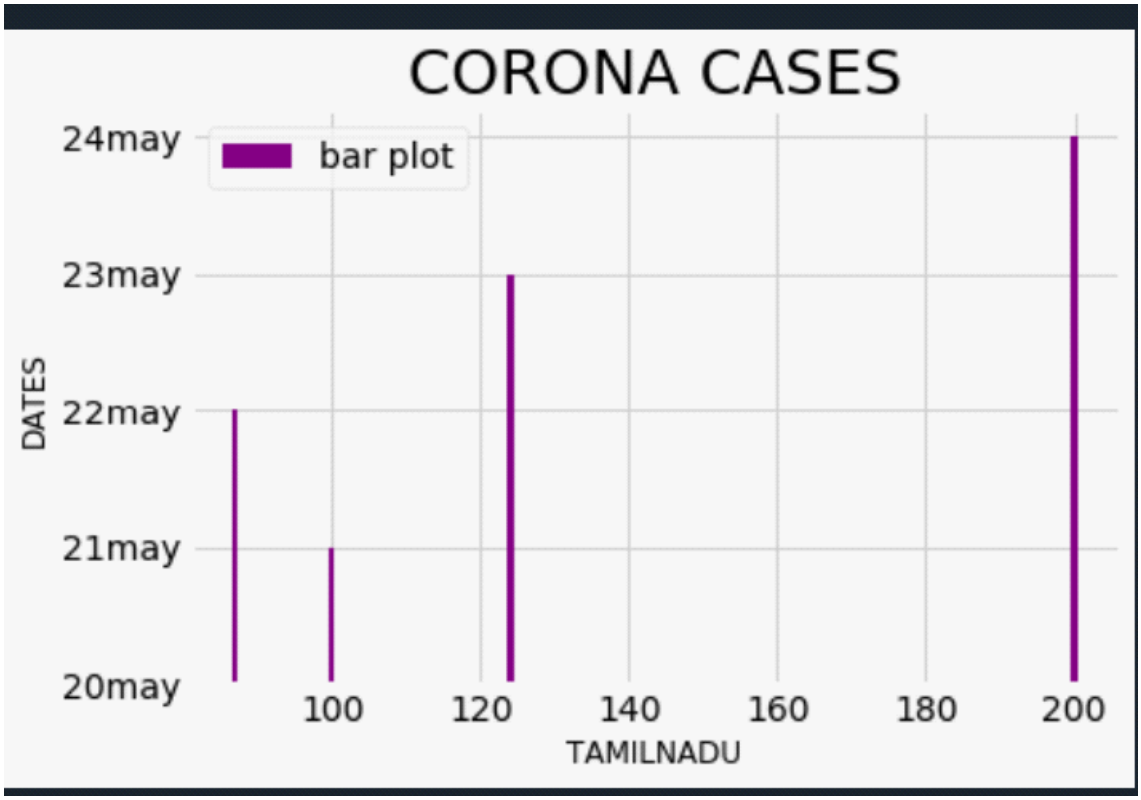
plt.ylabel("DATES",fontsize=12)

plt.bar(x,y,color="PURPLE",label="bar plot")

plt.legend(loc=2)

plt.show()

output:



CODE:

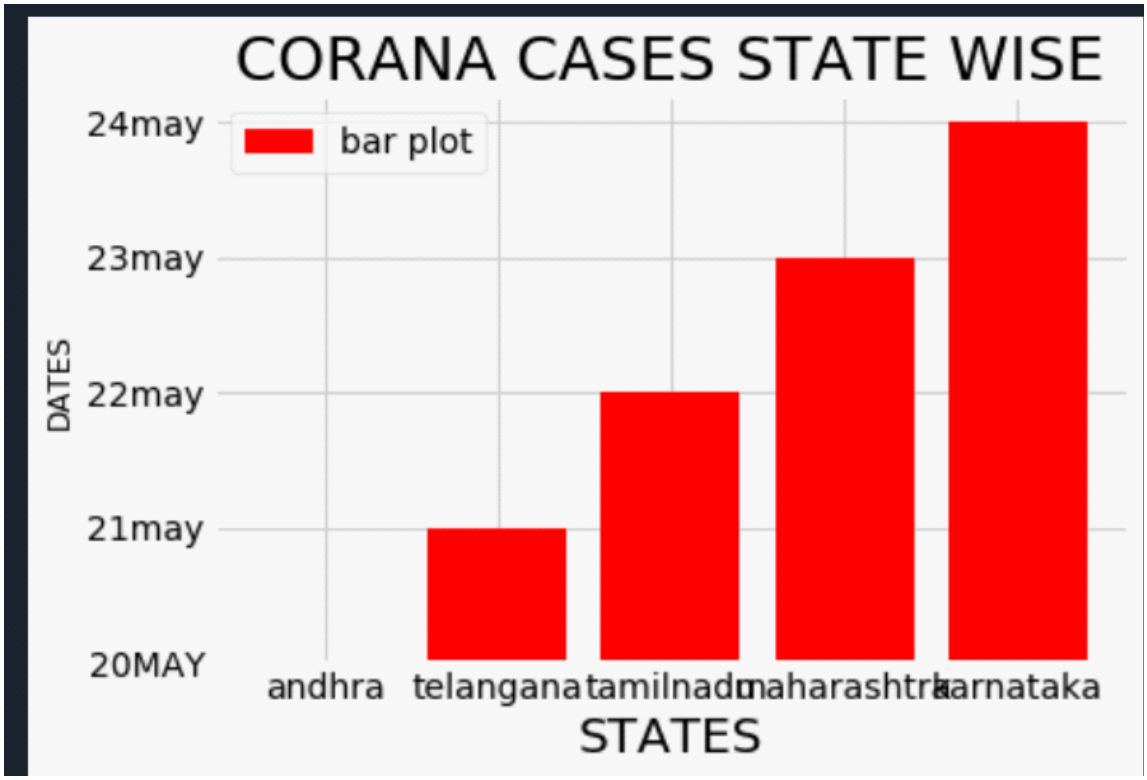
x = ['andhra','telangana','tamilnadu','maharashtra','karnataka']

y = ['20MAY','21may','22may','23may','24may']

plt.title("CORANA CASES MORTALITY STATE WISE",fontsize=25)

```
plt.rcParams['figure.figsize'] = (20,6)
plt.xlabel("STATES",fontsize=20)
plt.ylabel("DATES",fontsize=12)
plt.bar(x,y,color="red",label="bar plot")
plt.legend(loc=2)
plt.show()
```

output:



conclusion:

The model can be generalized to predict case fatality for any infection

(including

asymptomat), to predict the rate of severe disease, and to predict the death rate for

patients who develop severe disease. These early, accurate predictions inform the

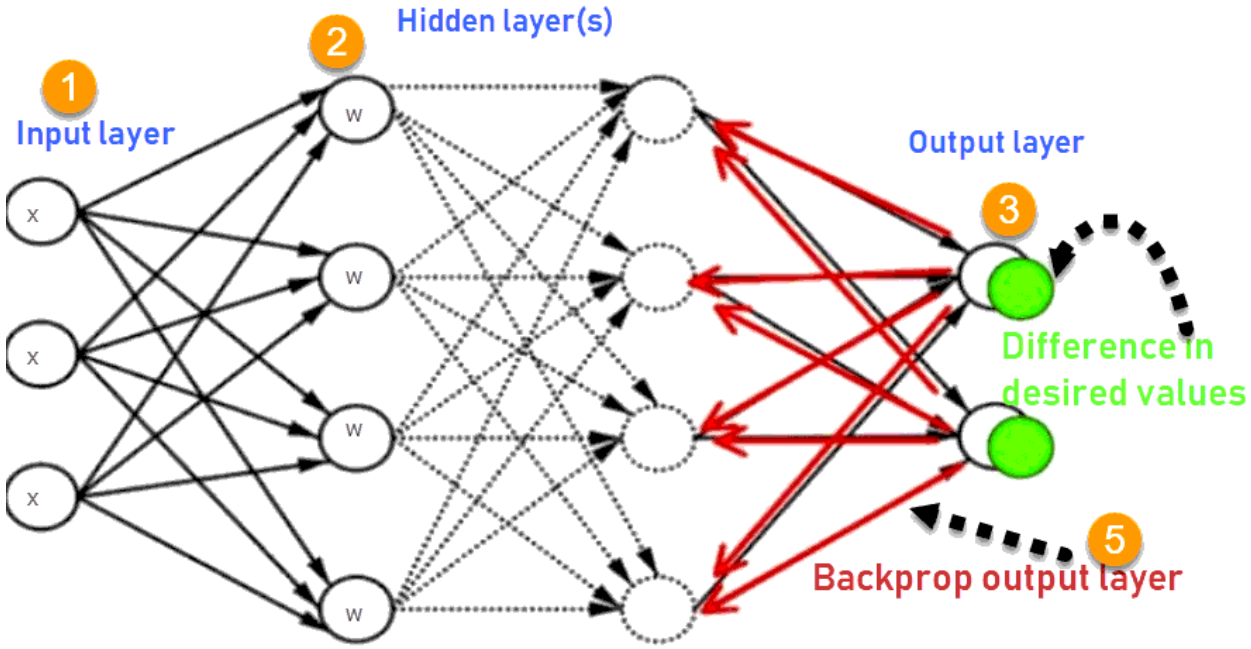
public, society, and governments to estimate the extent of the disease's harm and to

develop suitable strategies.

What is Backpropagation?

Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.



1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

   ErrorB= Actual Output – Desired Output

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Types of Backpropagation Networks:

Two Types of Backpropagation Networks are:

* **Static Back-propagation**
* **Recurrent Backpropagation**

Static back-propagation:

It is one kind of backpropagation network which produces a mapping of

a static input for static output. It is useful to solve static classification issues like optical character recognition.

Recurrent Backpropagation:

Recurrent backpropagation is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.

The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation.

Face Recognition: A **facial recognition system** is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape.

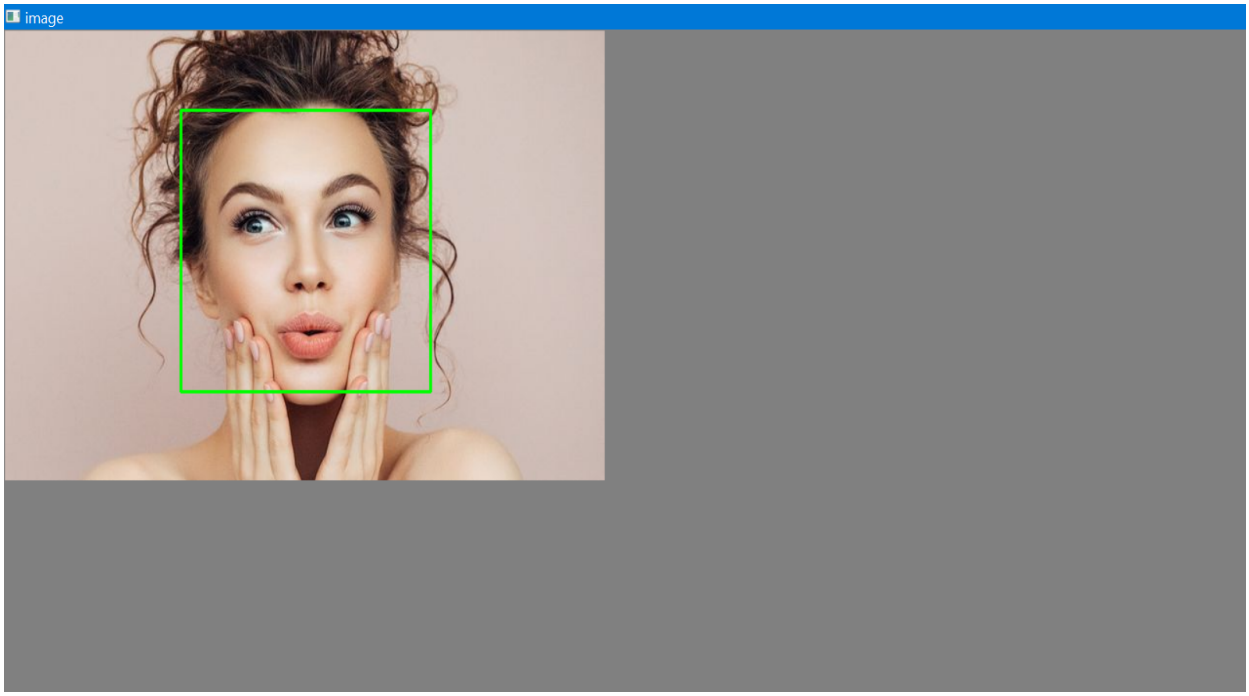
Face recognition: in this practical we're identify faces in the picture

Code:

```
import cv2

import numpy as np

# loading the source image
```

```
img = cv2.imread('girl.jpg')

# converting to grayscale

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

haar_cascade_face =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

faces_rects = haar_cascade_face.detectMultiScale(img_gray,
scaleFactor = 1.3, minNeighbors = 10);

# getting no. of faces detected

print('Faces Detected : ', len(faces_rects))

for (x,y,w,h) in faces_rects:

img = cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('image', img)

cv2.waitKey(0);

cv2.destroyAllWindows();

cv2.waitKey(1)
```

output:



Face Recognition with webcam:

Code: # Face Recognition

```
import cv2

face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml') # We load the cascade for the face.

eye_cascade=cv2.CascadeClassifier('haarcascade_eye.xml') # We load the cascade for the eyes.

def detect(gray, frame):

for (x, y, w, h) in faces: cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

```
 roi_gray = gray[y:y+h, x:x+w]
roi_color = frame[y:y+h, x:x+w]
eyes = eye_cascade.detectMultiScale(roi_gray, 1.1, 3)  for (ex, ey, ew, eh) in eyes: cv2.rectangle(roi_color,(ex, ey),(ex+ew, ey+eh), (0, 255, 0), 2)
return frame
video_capture = cv2.VideoCapture(0)
while True: # We repeat infinitely (until break):
    _, frame = video_capture.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
canvas = detect(gray, frame)
 cv2.imshow('Video', canvas)
 if cv2.waitKey(1) & 0xFF == ord('q'):
 break
video_capture.release()
cv2.destroyAllWindows()
```

output:

whenever execute the code the webcam opens and detect the face of yours .

Task:

Facemask Detection Based on Convolutional Neural Networks:

Abstract : Detecting faces mask with occlusions is a challenging task due to two main reasons: 1) the absence of large datasets of masked faces, and 2) the absence of facial cues from the masked regions.

1. Introduction:

Face detection has emerged as a very interesting problem in image processing and computer vision. It has a range of applications from facial motion capture to face recognition

which at the start needs the face to be detected with a very good accuracy. Face detection is more relevant today because it not only used on images but also in video applications like real time surveillance and face detection in videos. High accuracy image classification is possible now with the advancements
of Convolutional networks. Pixel level information is often required after face detection which most face detection methods fail to provide. Obtaining pixel level details has been a challenging part in semantic segmentation
Also most of the widely used face detection algorithms tend to focus on the detection of frontal faces.

Methodology:

- image datasets of the faces
- With mask
- With out mask
- And dump it on code with and with out datasets.
- We are play with the images and dump it on is in mask or with out mask

Code:

```
from imutils.video import VideoStream
import numpy as np
import imutils
import cv2
import tensorflow.keras
import numpy as np
import jovian
import keras
import matplotlib.pyplot as plt
import time

from keras.preprocessing.image import
ImageDataGenerator
image_gen=ImageDataGenerator(rotation_range
=30                          width_shift_range=0.1
height_shift_range=0.1,
  rescale=1/255, # Rescale the image by
normalzing it.
shear_range=0.2
 zoom_range=0.2
horizontal_flip=True,
 fill_mode='nearest'  )

img = cv2.imread('./face2.jpg')
```

```
plt.imshow(image_gen.random_transform(img)
  image_gen.flow_from_directory('./dataset/test')
image_gen.flow_from_directory('./dataset/train')
image_shape = (150,150,3)
from keras.model import sequential
from keras.layers import Activation, Dropout,
Flatten, Dense, Conv2D, MaxPooling2D
model = Sequential()

model.add(Conv2D(filters=32,
kernel_size=(3,3),input_shape=(150,150,3),
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64,
kernel_size=(3,3),input_shape=(150,150,3),
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64,
kernel_size=(3,3),input_shape=(150,150,3),
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))

# Dropouts help reduce overfitting by randomly
turning neurons off during training.
# Here we say randomly turn off 50% of
neurons.
model.add(Dropout(0.5))

# Last layer, remember its binary, 0=cat , 1=dog
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
optimizer='adam',
 metrics=['accuracy'])
model.summary()
batch_size = 16

train_image_gen =
image_gen.flow_from_directory('./dataset/train',
```

```
target_size=image_shape[:2],

batch_size=batch_size,

class_mode='binary')
test_image_gen=image_gen.flow_from_director
y('./dataset/test',
 target_size=image_shape[:2],
  batch_size=batch_size,
  class_mode='binary')
  train_image_gen.class_indices
 # with mask 1 #without mask 0
```

##testing on real world.

```
import numpy as np
from keras.preprocessing import image
from keras.models import load_model
model = load_model('./saved
models/keras_facemask_0906.h5')

    img = './face.jpg'
    img = image.load_img(img, target_size=(150,
150))
    img = image.img_to_array(img)
```

```
    img = img/255

    prediction =
model.predict_classes(img.reshape(1,150,150,3))
    for i in prediction:
        if(i==1):
            return('With Mask',(0,255,0))
        else:
            return('Without Mask',(0,0,255))
```

conclusion: The problem of face mask detection in the wild have been explored in many existing researches, and the corresponding face detectors have been tested on datasets of normal faces. On these datasets, some face detectors have achieved extremely high performances and it seems to be somehow difficult to further improve them.so this what we done might be this is correct or not we are done that's it we didn't got output the the process will be done will wind up this topic we are put much of effort to what is what in that includes.

---------- *END*----------