Apollo Institute of Engineering & Technology

Lab Manual

B.E. Semester: - VII

Subject:- Applied Machine Learning (3171617)

Name:	_____
Branch:	_____
Batch:	_____
Enrollment	_____

CERTIFICATE

This is to certify that Mr./Ms._____

Enrolment No._____Semester_____

Branch_____has satisfactorily completed his/her term work in Course of "**Pattern Recognition"** with subject code **3171613** for the academic term 2023-2024.

Staff In-charge **Date of Submission** **Head of Department**

APOLLO INSTITUTE OF ENGINEERING & TECHNOLOGY

B. E. Semester: - 7th
Academic Term: 2023 – 24

Subject: - Pattern Recognition (3171613)

INDEX

Practical: 1

Aim: Study About Introduction of Python and Implement Basic Python Program.

What is Python?

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

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

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

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

Program 1:

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

Output 1:

```
Hello, World!
```

Program 2:

```
num = int(input("Enter a number: "))

if num < 0:
   print("Enter a positive number")
else:
   sum = 0
   # use while loop to iterate until zero
   while(num > 0):
      sum += num
      num -= 1
   print("The sum is",sum)
```

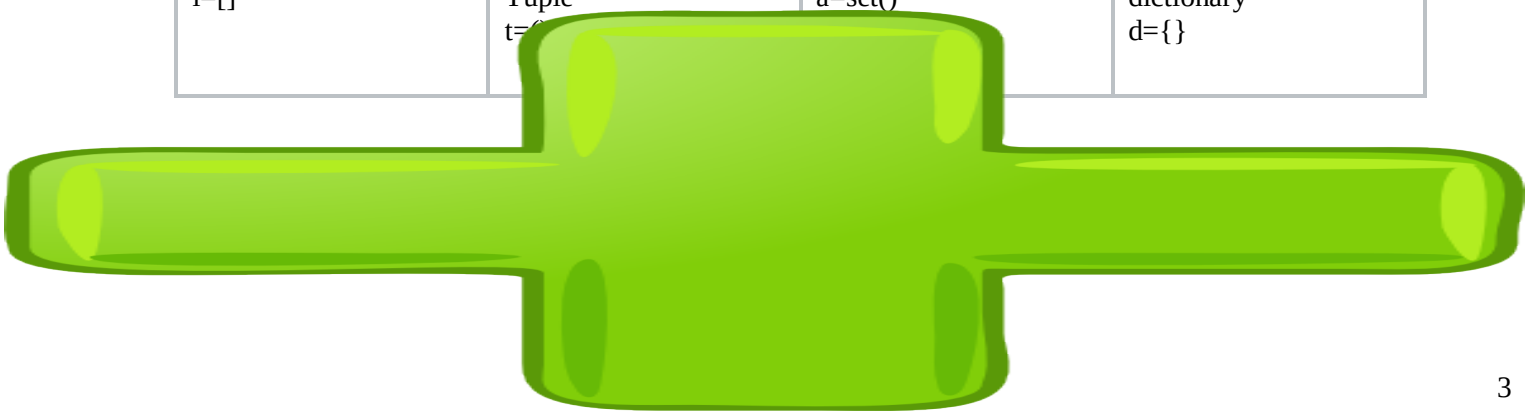
Output 2:

```
Enter a number: 5
The sum is 15
```

Practical: 2

Aim: Study About Basic Python Functionality and Implement List, Tuple, Dictionary, Set Related Program.

List	Tuple	Set	Dictionary
List is a non-homogeneous data structure that stores the elements in single row and multiple rows and columns	Tuple is also a non-homogeneous data structure that stores single row and multiple rows and columns	Set data structure is also non-homogeneous data structure but stores in single row	Dictionary is also a non-homogeneous data structure which stores key value pairs
List can be represented by [ ]	Tuple can be represented by ( )	Set can be represented by { }	Dictionary can be represented by { }
List allows duplicate elements	Tuple allows duplicate elements	Set will not allow duplicate elements	Set will not allow duplicate elements and dictionary doesn't allow duplicate keys.
List can use nested among all	Tuple can use nested among all	Set can use nested among all	Dictionary can use nested among all
Example: [1, 2, 3, 4, 5]	Example: (1, 2, 3, 4, 5)	Example: {1, 2, 3, 4, 5}	Example: {1, 2, 3, 4, 5}
List can be created using list() function	Tuple can be created using tuple() function.	Set can be created using set() function.	Dictionary can be created using dict() function.
List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple.	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)
Creating an empty list l=[]	Creating an empty Tuple t=	Creating a set a=set()	Creating an empty dictionary d={}



Program:

```
# use of list, tuple, set and dictionary

# Lists
l = []

# Adding Element into list
l.append(5)
l.append(10)
print("Adding 5 and 10 in list", l)

# Popping Elements from list
l.pop()
print("Popped one element from list", l)
print()

# Set
s = set()

# Adding element into set
s.add(5)
s.add(10)
print("Adding 5 and 10 in set", s)

# Removing element from set
s.remove(5)
print("Removing 5 from set", s)
print()

# Tuple
t = tuple(l)

# Tuples are immutable
print("Tuple", t)
print()

# Dictionary
d = {}

# Adding the key value pair
d[5] = "Five"
d[10] = "Ten"
print("Dictionary", d)

# Removing key-value pair
del d[10]
print("Dictionary", d)
```

Output:

```
Adding 5 and 10 in list [5, 10]
Popped one element from list [5]

Adding 5 and 10 in set {10, 5}
Removing 5 from set {10}

Tuple (5,)

Dictionary {5: 'Five', 10: 'Ten'}
Dictionary {5: 'Five'}
```

Practical: 3

Aim: Study about NumPy, SciPy libraries.

What is NumPy?

- NumPy is a Python library used for working with arrays.
- It also has functions for working in the domain of linear algebra, fourier transform, and matrices.
- NumPy stands for Numerical Python.

Why Use NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
- Arrays are very frequently used in data science, where speed and resources are very important.

Program:

```
# Creation of Arrays
import numpy as np

# Creating a rank 1 Array
arr = np.array([1, 2, 3])
print("Array with Rank 1: \n",arr)

# Creating a rank 2 Array
arr = np.array([[1, 2, 3], [4, 5, 6]])
print("Array with Rank 2: \n", arr)

# Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using passed tuple:\n", arr)
```

Output:

```
Array with Rank 1:
 [1 2 3]
Array with Rank 2:
 [[1 2 3]
 [4 5 6]]

Array created using passed tuple:
 [1 3 2]
```

What is SciPy?

- SciPy is a scientific computation library that uses NumPy underneath.
- SciPy stands for Scientific Python.
- It provides more utility functions for optimization, stats and signal processing.
- Like NumPy, SciPy is open source so we can use it freely.
- SciPy was created by NumPy's creator Travis Olliphant.

Why Use SciPy?

- If SciPy uses NumPy underneath, why can we not just use NumPy?
- SciPy has optimized and added functions that are frequently used in NumPy and Data Science.

Program:

```
import numpy as np
from scipy import linalg

# We are trying to solve a linear algebra system which can be given as
#       x + 3y +10z = 10
#       2x + 12y + 7z = 18
#       5x + 8y + 8z = 30
# Creating input array
a = np.array([[1, 3, 10], [2, 12, 7], [5, 8, 8]])

# Solution Array
b = np.array([[10], [18], [30]])

# Solve the linear algebra
x = linalg.solve(a, b)
```

```
# Checking Results
print("\n Checking results, Vectors must be zeros")
print(a.dot(x) - b)
```

Output:

```
[[4.55393586]
 [0.51311953]
 [0.39067055]]

 Checking results, Vectors must be zeros
[[0.]
 [0.]
 [0.]]
```

Numpy VS SciPy

Numpy:

- Numpy is written in C and used for mathematical or numeric calculation.
- It is faster than other Python Libraries
- Numpy is the most useful library for Data Science to perform basic calculations.
- Numpy contains nothing but array data type which performs the most basic operation like sorting, shaping, indexing, etc.

SciPy:

- SciPy is built in top of the NumPy
- SciPy module in Python is a fully-featured version of Linear Algebra while Numpy contains only a few features.
- Most new Data Science features are available in Scipy rather than Numpy.

Practical: 4

Aim: Study about Pandas, Matplotlib libraries.

What is Pandas?

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.

Why Use Pandas?

- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.
- Relevant data is very important in data science.

What Can Pandas Do?

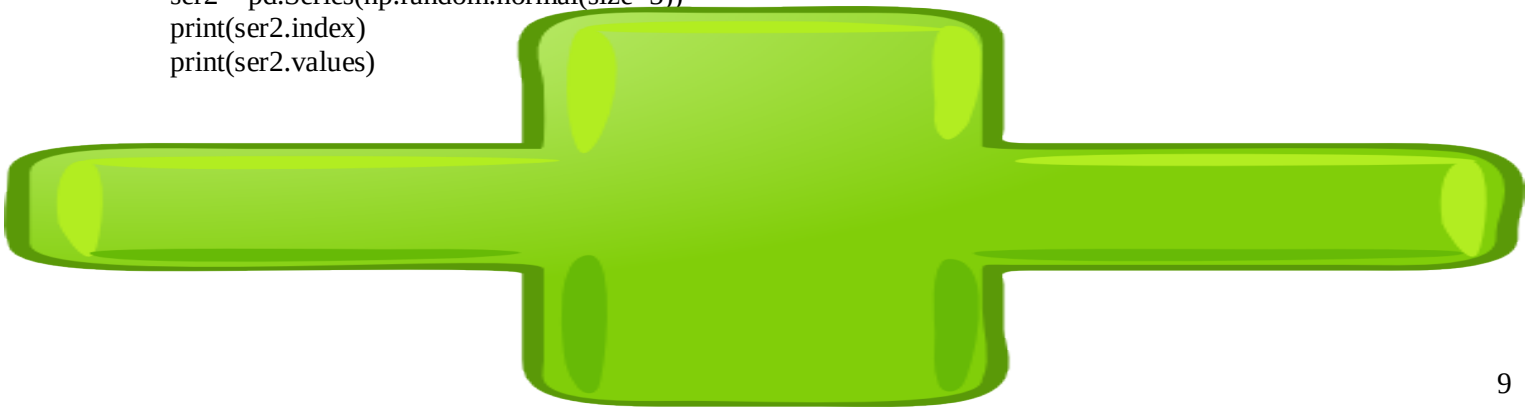
Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contain wrong values, like empty or NULL values. This is called cleaning the data.

Program:

```
import pandas as pd
import numpy as np

ser1 = pd.Series({"India": "New Delhi",
                  "Japan": "Tokyo",
                  "UK": "London"})

print(ser1.values)
print(ser1.index)

print("\n")

ser2 = pd.Series(np.random.normal(size=5))
print(ser2.index)
print(ser2.values)
```



Output:

```
['New Delhi' 'Tokyo' 'London']
Index(['India', 'Japan', 'UK'], dtype='object')


RangeIndex(start=0, stop=5, step=1)
[ 1.22139371 -0.10330068 -0.30804345 -1.07830258  1.00387381]
```

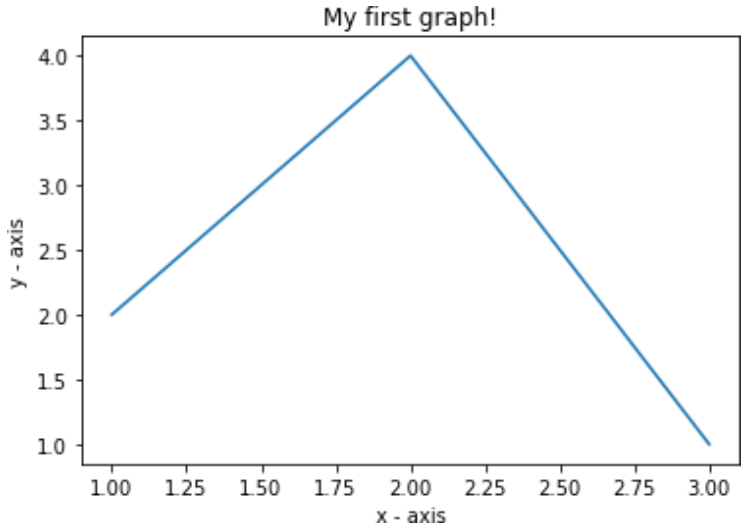
What is Matplotlib?

- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source and we can use it freely.
- Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Program:

```
# importing the required module
import matplotlib.pyplot as plt

# x axis values
x = [1,2,3]
# corresponding y axis values
y = [2,4,1]

# plotting the points
plt.plot(x, y)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('My first graph!')

# function to show the plot
plt.show()
```

Output:


My first graph!
4.0
3.5
3.0
2.5
2.0
1.5
1.0
y - axis
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
x - axis

Practical:5

Aim: Write a Python Program to Read and Write CSV files.

The writer class has following methods

writerow()
This function writes items in an iterable (list, tuple or string) ,separating them nby comma character.

writerows()
This function takes a list of iterables as parameter and writes each item as a comma separated line of items in the file.
Following example shows use of write() function. First a file is opened in 'w' mode. This file is used to obtain writer object. Each tuple in list of tuples is then written to file using writerow() method.

Program:

```
import csv
persons=[('Lata',22,45),('Anil',21,56),('John',20,60)]
csvfile=open('persons.csv','w', newline='')
obj=csv.writer(csvfile)
for person in persons:obj.writerow(person)
csvfile.close()
```

Output:

```
Lata,22,45
Anil,21,56
John,20,60
```

read()
This function returns a reader object which returns an iterator of lines in the csv file. Using the regular for loop, all lines in the file are displayed in the following example.

Program:

```
csvfile=open('persons.csv','r', newline='')
obj=csv.reader(csvfile)
for row in obj:
print (row)
```

Output:

Practical: 6

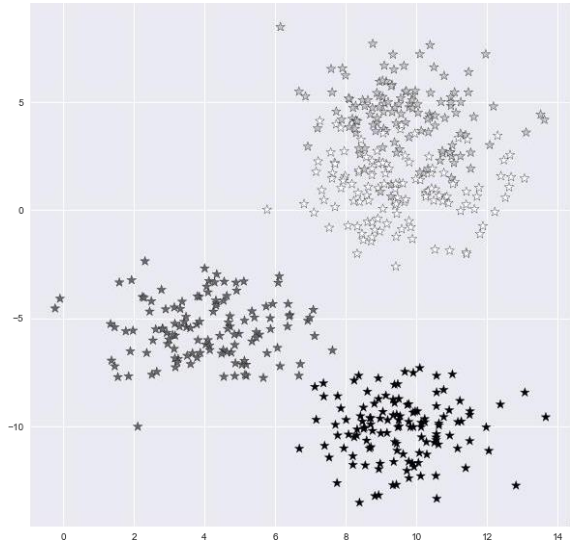
Aim: Implement KNN Algorithm on any Dataset.

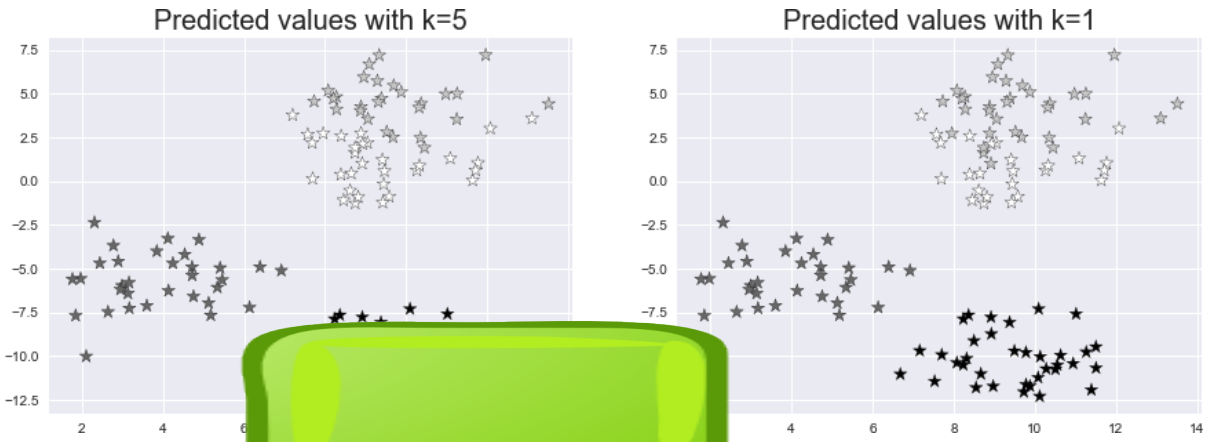
Program:

```
#Importing the modules

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

#Creating Dataset

X, y = make_blobs(n_samples = 500, n_features = 2, centers = 4,cluster_std = 1.5, random_state = 4)

#Visualize the Dataset

plt.style.use('seaborn')
plt.figure(figsize = (10,10))
plt.scatter(X[:,0], X[:,1], c=y, marker= '*',s=100,edgecolors='black')
plt.show()

#Splitting Data into Training and Testing Datasets

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)

#KNN Classifier Implementation

knn5 = KNeighborsClassifier(n_neighbors = 5)
knn1 = KNeighborsClassifier(n_neighbors=1)

#Predictions for the KNN Classifiers

knn5.fit(X_train, y_train)
knn1.fit(X_train, y_train)

y_pred_5 = knn5.predict(X_test)
y_pred_1 = knn1.predict(X_te

#Predict Accuracy for both k v
```

```
print("Accuracy with k=1", accuracy_score(y_test, y_pred_1)*100)

#Visualize Predictions

plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
plt.scatter(X_test[:,0], X_test[:,1], c=y_pred_5, marker= '*', s=100,edgecolors='black')
plt.title("Predicted values with k=5", fontsize=20)

plt.subplot(1,2,2)
plt.scatter(X_test[:,0], X_test[:,1], c=y_pred_1, marker= '*', s=100,edgecolors='black')
plt.title("Predicted values with k=1", fontsize=20)
plt.show()
```

Output:



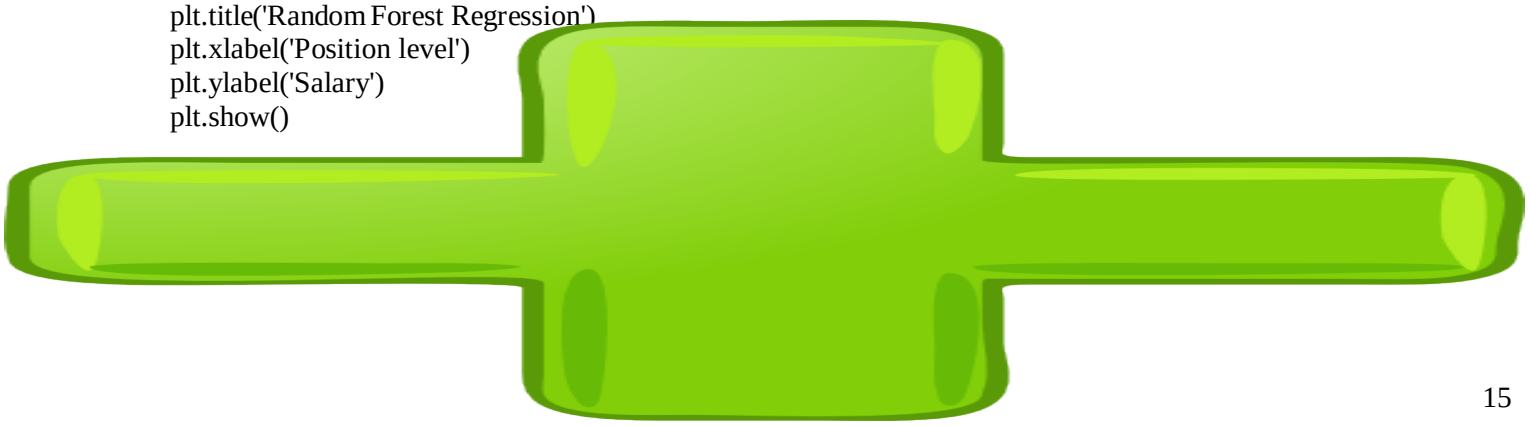


Practical: 7

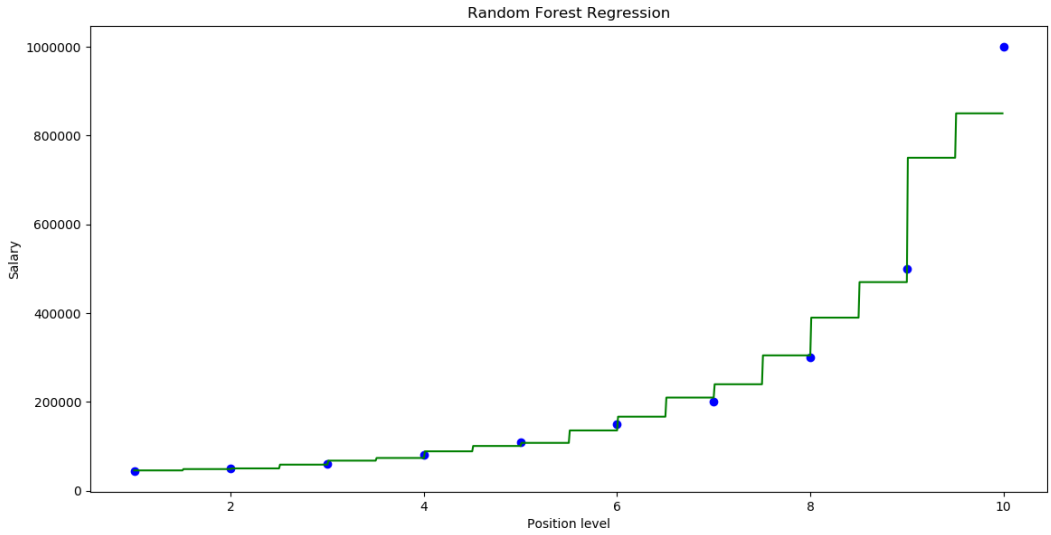
Aim: Implement Random Forest Algorithm on any Dataset.

Program:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv('Salaries.csv')

# Fitting Random Forest Regression to the dataset
# import the regressor
from sklearn.ensemble import RandomForestRegressor

# create regressor object
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)

# fit the regressor with x and y data
regressor.fit(x, y)

Y_pred = regressor.predict(np.array([6.5]).reshape(1, 1)) # test the output by changing values

# Visualising the Random Forest Regression results

# arrange for creating a range of values
# from min value of x to max
# value of x with a difference of 0.01
# between two consecutive values
X_grid = np.arrange(min(x), max(x), 0.01)

# reshape for reshaping the data into a len(X_grid)*1 array,
# i.e. to make a column out of the X_grid value
X_grid = X_grid.reshape((len(X_grid), 1))

# Scatter plot for original data
plt.scatter(x, y, color = 'blue')

# plot predicted data
plt.plot(X_grid, regressor.predict(X_grid),
                    color = 'green')
plt.title('Random Forest Regression')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Output:


Random Forest Regression
1000000
800000
600000
400000
200000
0
Salary
2
4
6
8
10
Position level

Practical: 8

Aim: Implement Naive Bayes Classifiers on any Dataset.

Program:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

gnb = GaussianNB()
mnb = MultinomialNB()
y_pred_gnb = gnb.fit(X_train, y_train).predict(X_test)
cnf_matrix_gnb = confusion_matrix(y_test, y_pred_gnb)
print(cnf_matrix_gnb)

y_pred_mnb = mnb.fit(X_train, y_train).predict(X_test)
cnf_matrix_mnb = confusion_matrix(y_test, y_pred_mnb)
print(cnf_matrix_mnb)
```

Output:

```
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]


[[16  0  0]
 [ 0  0 18]
 [ 0  0 11]]
```

Practical: 9

Aim: Implement Linear Regression Algorithm on any Dataset.

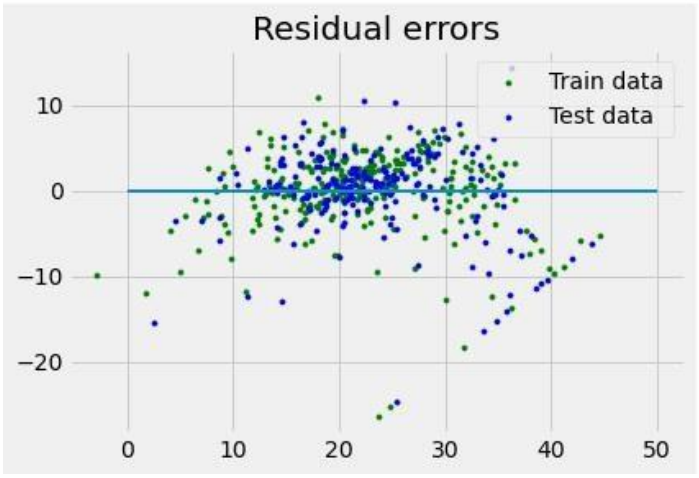
Program:

```
import matplotlib.pyplot as mtpplt
import numpy as nmp
from sklearn import datasets as DS
from sklearn import linear_model as LM
from sklearn import metrics as mts

# First, we will load the boston dataset
boston1 = DS.load_boston(return_X_y = False)

# Here, we will define the feature matrix(H) and response vector(f)
H = boston1.data
f = boston1.target

# Now, we will split X and y datasets into training and testing sets
from sklearn.model_selection import train_test_split as tts
H_train, H_test, f_train, f_test = tts(H, f, test_size = 0.4, random_state = 1)

# Here, we will create linear regression object
reg1 = LM.LinearRegression()

# Now, we will train the model by using the training sets
reg1.fit(H_train, f_train)

# here, we will print the regression coefficients
print('Regression Coefficients are: ', reg1.coef_)

# Here, we will print the variance score: 1 means perfect prediction
print('Variance score is: {}'.format(reg1.score(H_test, f_test)))

# Here, we will plot for residual error

# here, we will set the plot style
mtpplt.style.use('fivethirtyeight')

# here we will plot the residual errors in training data
mtpplt.scatter(reg1.predict(H_train), reg1.predict(H_train) - f_train, color = "green", s = 10, label = 'Train
data')

# Here, we will plot the residu
mtpplt.scatter(reg1.predict(H_                    = "blue", s = 10, label = 'Test data')

# Here, we will plot the line fo
```

```
# here, we will plot the legend
mtpplt.legend(loc = 'upper right')

# now, we will plot the title
mtpplt.title("Residual errors")

# here, we will define the method call for showing the plot
mtpplt.show()
```

Output:

```
Regression Coefficients are:  [-8.95714048e-02  6.73132853e-02  5.04649248e-02  2.18579583e+00
 -1.72053975e+01  3.63606995e+00  2.05579939e-03 -1.36602886e+00
  2.89576718e-01 -1.22700072e-02 -8.34881849e-01  9.40360790e-03
 -5.04008320e-01]
Variance score is: 0.720905667266178
```



Practical: 10

Aim: Implement Logistic Regression Algorithm on any Dataset.

Program:

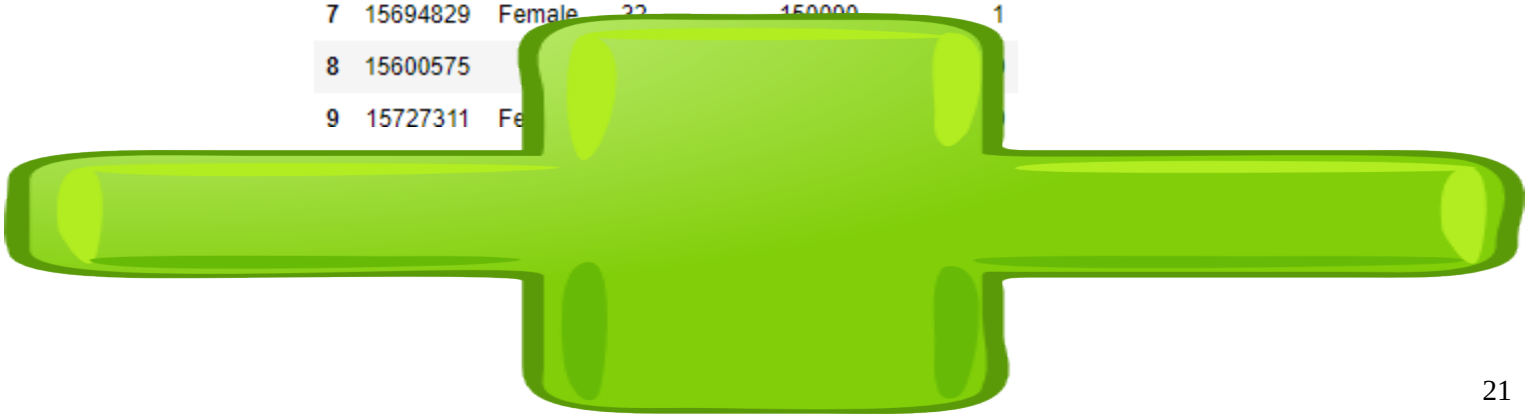
```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from math import exp
plt.rcParams["figure.figsize"] = (10, 6)

data = pd.read_csv("Social_Network_Ads.csv")
data.head(10)

# Visualizing the dataset by Age and Purchased
plt.scatter(data['Age'], data['Purchased'])
plt.show()

# Divide the Data into training set and test set
X_train, X_test, y_train, y_test = train_test_split(data['Age'], data['Purchased'], test_size=0.20)

# Building the Logistic Regression model

# Normalizing the data
def normalize(X):
  return X - X.mean()

# Make predictions
def predict(X, b0, b1):
  return np.array([1 / (1 + exp(-1*b0 + -1*b1*x)) for x in X])

# The model
def logistic_regression(X, Y):

   X = normalize(X)

   # Initializing variables
   b0 = 0
   b1 = 0
   L = 0.001
   epochs = 300

   for epoch in range(epochs)
```

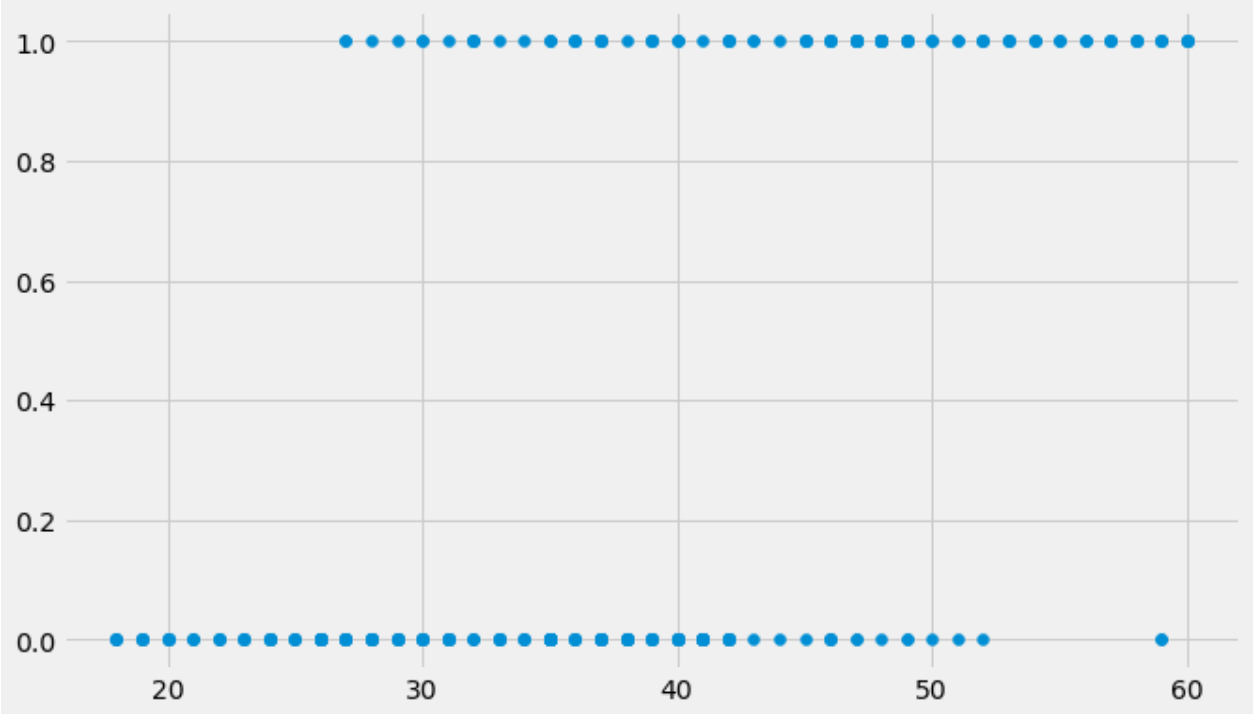
```
    D_b1 = -2 * sum(X * (Y - y_pred) * y_pred * (1 - y_pred)) # Loss wrt b1
    # Update b0 and b1
    b0 = b0 - L * D_b0
    b1 = b1 - L * D_b1

    return b0, b1

# Training the Model
b0, b1 = logistic_regression(X_train, y_train)

# Making predictions and setting a threshold
X_test_norm = normalize(X_test)
y_pred = predict(X_test_norm, b0, b1)
y_pred = [1 if p >= 0.5 else 0 for p in y_pred]

# Plotting the data
plt.scatter(X_test, y_test)
plt.scatter(X_test, y_pred, c="red")
plt.show()

# Calculating the accuracy
accuracy = 0
for i in range(len(y_pred)):
    if y_pred[i] == y_test.iloc[i]:
        accuracy += 1
        print(f"Accuracy = {accuracy / len(y_pred)}")
```

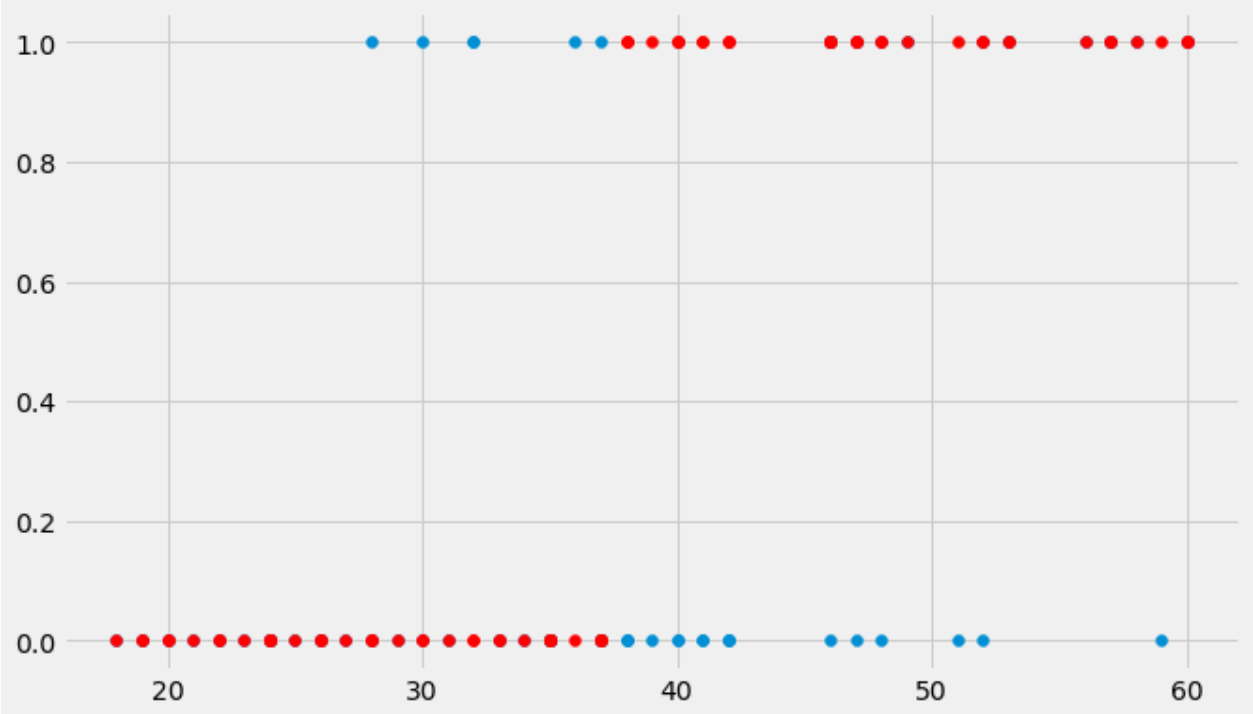
Output:

Out[8]:

	User ID	Gender	Age	Estimated Salary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575				
9	15727311	Fe			








```
Accuracy = 0.6625
Accuracy = 0.675
Accuracy = 0.6875
Accuracy = 0.7
Accuracy = 0.7125
```

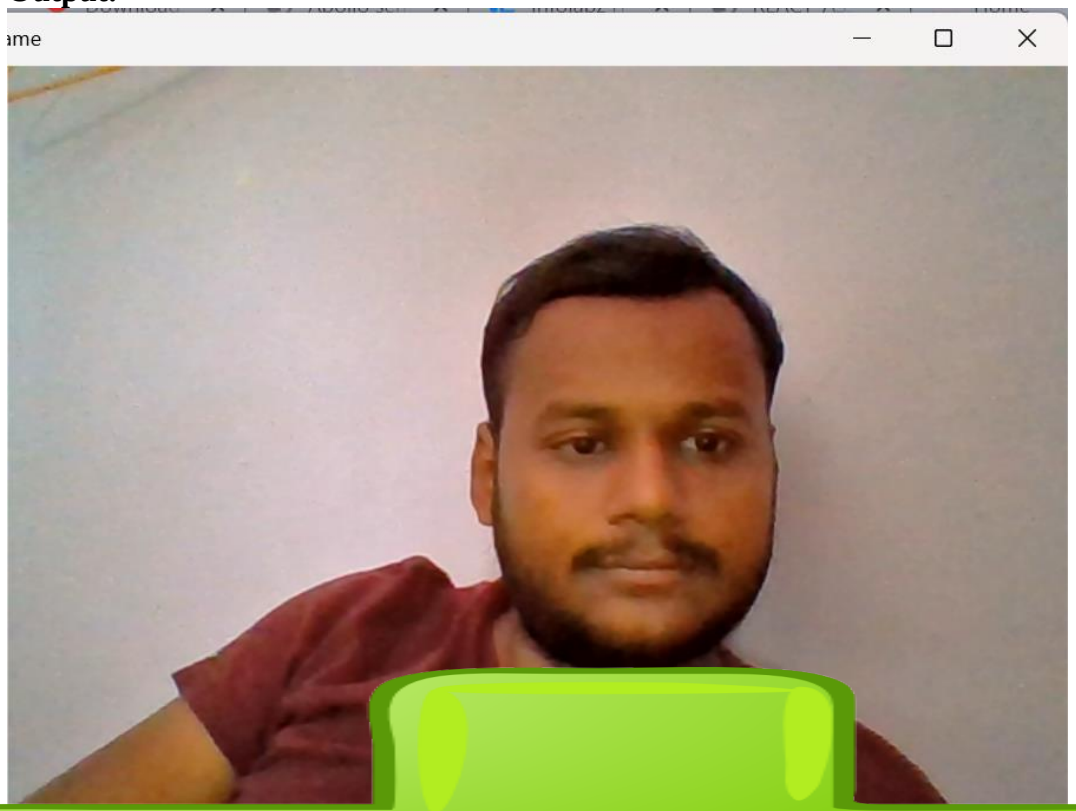
Practical: 11

Aim: Write A script open camera Using OpenCV and apply gray scale mode.

Open camera Using OpenCV:

```
cap =cv2.VideoCapture(0)
if (cap.isOpened()==False):
    print("Error opening vidio stream or file")
while(cap.isOpened()):
    ret,frame = cap.read()
    if ret == True:
        cv2.imshow('Frame',frame)
        if cv2.waitKey(25) & 0xFF == ord('a'):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

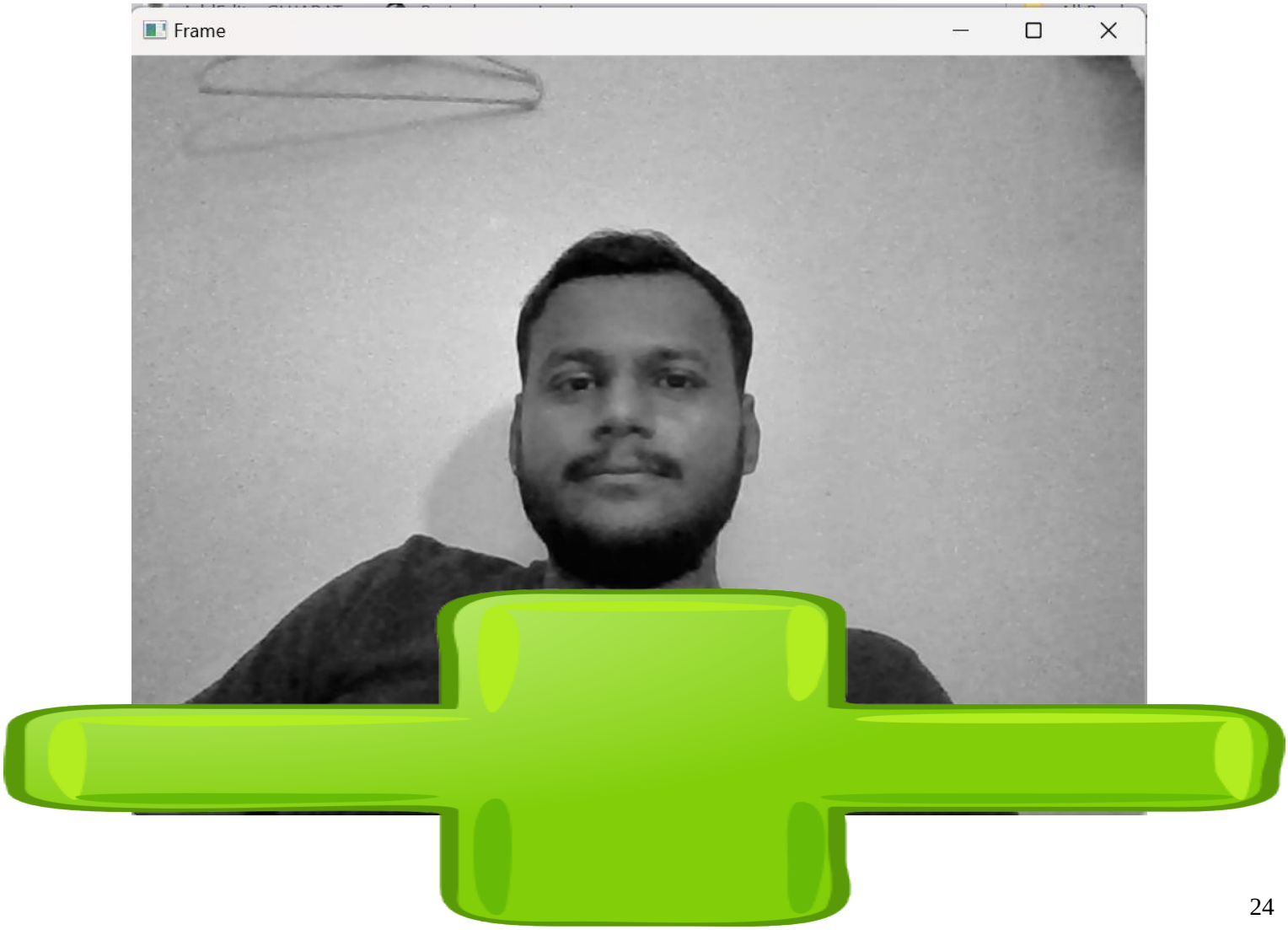
Output:



Open camera with apply gray scale mode:

```
import cv2
cap =cv2.VideoCapture(0)
if (cap.isOpened()==False):
    print("Error opening vidio stream or file")
while(cap.isOpened()):
    ret,frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    if ret == True:
        cv2.imshow('Frame',gray)
        if cv2.waitKey(25) & 0xFF == ord('a'):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

Output:



Practical: 10

Aim: Capture image click on key ”c” & save with ‘capture image.jpg’

```
import cv2

# Initialize the camera
cap = cv2.VideoCapture(0)  # 0 represents the default camera (usually the built-in webcam)

if not cap.isOpened():
    print("Error: Could not open camera.")
    exit()

while True:
    # Capture a frame from the camera
    ret, frame = cap.read()

    # Display the captured frame in a window
    cv2.imshow("Camera", frame)

    # Check if the "c" key is pressed
    key = cv2.waitKey(1) & 0xFF
    if key == ord("c"):
        # Save the captured frame as an image (you can customize the filename and format)
        cv2.imwrite("captured_image.jpg", frame)
        print("Image captured!")

    # Check if the "q" key is pressed to exit the loop
    if key == ord("a"):
        break

# Release the camera and close all OpenCV windows
cap.release()
cv2.destroyAllWindows()
```

Output:

Practical: 10

Aim: Face Recognition

```
import cv2

face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)
video_capture = cv2.VideoCapture(0)
def detect_bounding_box(vid):
    gray_image = cv2.cvtColor(vid, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5, minSize=(40, 40))
    for (x, y, w, h) in faces:
        cv2.rectangle(vid, (x, y), (x + w, y + h), (0, 255, 0), 4)
    return faces
while True:

    result, frame = video_capture.read()
    if result is False:
        break

    faces = detect_bounding_box(frame)

    cv2.imshow(
        "My Face Detection Project", frame
    )

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

video_capture.release()
cv2.destroyAllWindows()
```

Output: