MAJOR PROJECT

12/31/2022

- APPLYING CLASSIFICATON/REGRESSION
- IMAGE PROCESSING USING OpenCV
- EXPLORATORY DATA ANALYSIS

SUBMITED BY:

NAME : CHELLAMALLA NITHIN
COLLEGE : SRI INDU COLLEGE OF ENGG. & TECH. HYDERABAD
YEAR : 3RD YEAR
DEPT. : COMPUTER SCIENCE AND ENGINEERING…

GUIDED BY:
AMEEN MANNA SIR
RINEX TECHNOLOGIES…

Problem Statement – 1:

Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR

Model Creation/Project

Classifier:

In machine learning, a classifier is **an algorithm that automatically assigns data points to a range of categories or classes**. Within the classifier category, there are two main models: supervised and unsupervised. In the supervised model, classifiers train to make distinctions between labelled and unlabelled data.

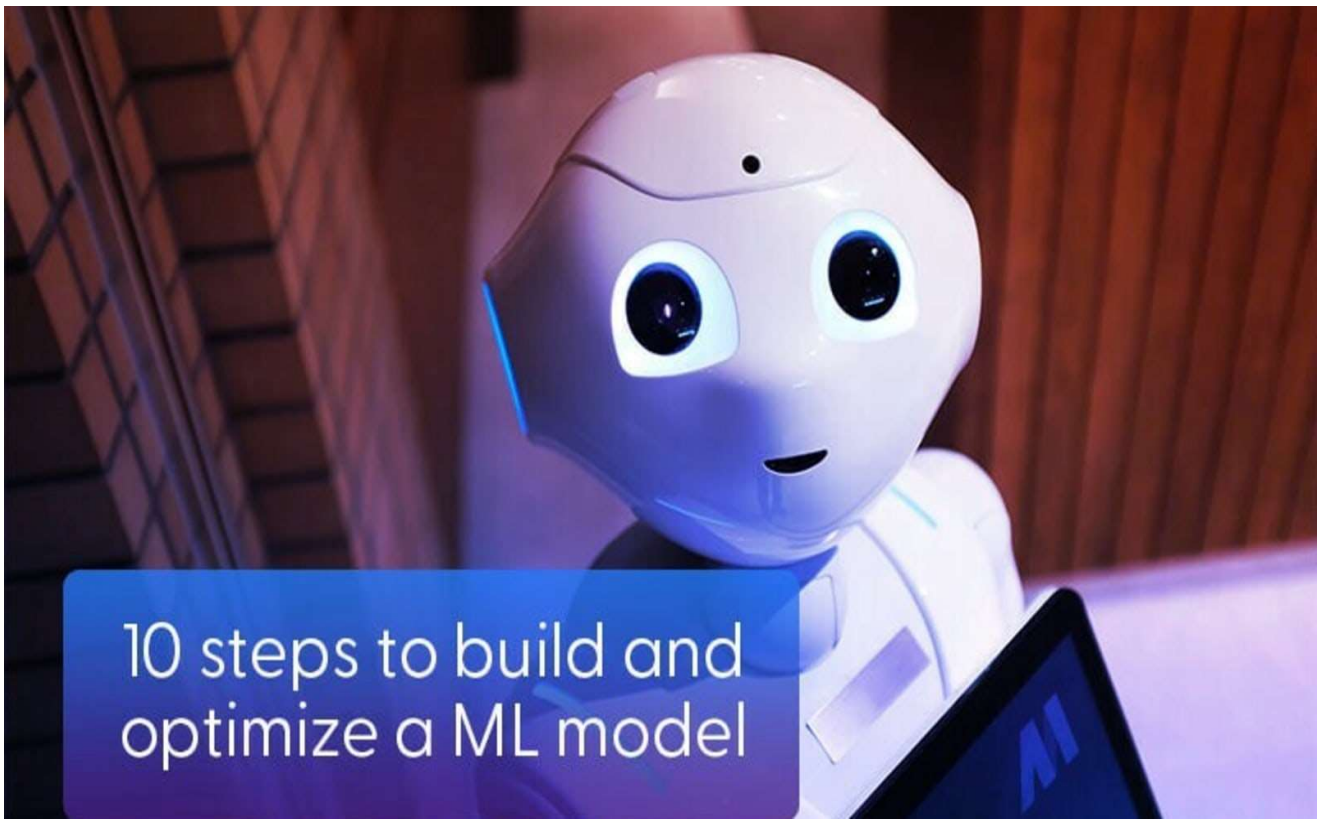
Regression:

Regression is **a technique for investigating the relationship between independent variables or features and a dependent variable or outcome**. It's used as a method for predictive modelling in machine learning, in which an algorithm is used to predict continuous outcomes.

#LOGISTIC REGRESSION

#Datset - https://github.com/chellamallanithin7/project_rinex

Drive link : https://drive.google.com/drive/folders/1uN43Udf0cX3xH15FaImYzdhRZDCJFOMD?usp=share_link



#1. Take the data and create dataframe

```
#1.Take the dataSet & Create Dataframe
import pandas as pd
df=pd.read_csv('/content/winequality-red.csv')
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

#2. Pre-processing – Data Filtering

```
[ ] df.isnull().sum()
```

```
fixed acidity          0
volatile acidity       0
citric acid            0
residual sugar         0
chlorides              0
free sulfur dioxide    0
total sulfur dioxide   0
density                0
pH                     0
sulphates              0
alcohol                0
quality                0
dtype: int64
```

```
[ ] #2. Preprocessing - Data Filtering
    #by the above result
    #Step 2 Not  Required
```

#3. Data Visualization

```
#3.Data Visualization
#Not Required
```

#4. Divide the data into input and output

```
[ ] #4. Divide into i/p & o/p
    x = df.iloc[:,0:11].values
    y = df.iloc[:,11].values
```

#5. Train and test variables

```
#5. Train and Test Variable
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=0)
```

#6. Not required because inputs are already scaled

```
[ ] #6. Normalize (Scaling) the data (inputs only)
    #Not Required
```

#7. Apply a Classifier, regressor or clusterer

```
[ ] #7. Apply the Classifiier/Regressor/clusterer
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
```

#8. Fitting the model

```
[ ] #8. fit the model
    model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression()
```

#9. PREDICT THE OUTPUT

```
#9. Predict the Output
y_pred = model.predict(x_test)
y_pred
```

```
array([6, 5, 6, 5, 6, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 6, 6, 6, 6, 5, 6, 5,
       6, 4, 5, 5, 5, 6, 5, 6, 6, 6, 6, 5, 6, 6, 5, 5, 6, 6, 5, 6, 6, 6,
       6, 5, 6, 6, 5, 6, 5, 5, 5, 6, 6, 5, 5, 5, 5, 6, 5, 5, 6, 6, 6, 5, 6,
       5, 6, 6, 6, 5, 6, 5, 6, 6, 6, 5, 5, 5, 6, 6, 5, 6, 6, 6, 6, 5, 6, 5,
       5, 5, 5, 5, 6, 5, 6, 5, 6, 5, 5, 6, 7, 6, 6, 6, 6, 5, 6, 5, 6, 5,
       6, 5, 6, 5, 6, 5, 6, 6, 6, 6, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 6, 5, 5,
       6, 6, 6, 5, 6, 5, 6, 5, 6, 5, 6, 5, 5, 6, 6, 6, 6, 5, 6, 6, 6, 5, 6,
       5, 6, 5, 5, 6, 6, 6, 5, 6, 5, 6, 5, 6, 6, 5, 6, 6, 5, 5, 6, 6, 6,
       7, 6, 6, 5, 5, 6, 5, 6, 7, 5, 6, 6, 5, 6, 6, 6, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 6, 5, 5, 6, 5, 5, 6, 6, 5, 6, 5, 6, 6, 5, 6, 5, 5, 6, 6, 5, 5,
       5, 6, 6, 6, 5, 6, 6, 6, 6, 5, 5, 6, 5, 5, 6, 6, 6, 6, 6, 6, 6, 5, 5,
       5, 5, 6, 5, 5, 5, 5, 5, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 5, 5,
       5, 5, 5, 5, 6, 6, 6, 5, 6, 6, 6, 6, 6, 5, 6, 6, 5, 6, 5, 6, 5, 6, 6, 5,
       6, 5, 6, 6, 6, 6, 6, 6, 5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 6, 6, 6, 5, 5,
       5, 5, 6, 6, 5, 5, 5, 6, 6, 6, 5, 6, 5, 5, 6, 6, 6, 5, 6, 5, 5, 5,
       6, 5, 5, 6, 5, 5, 6, 5, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6,
       5, 6, 6, 5, 6, 6, 6, 6, 5, 6, 5, 6, 5, 5, 5, 6, 5, 6, 6, 6, 5, 6, 6,
       5, 5, 6, 6, 6, 6, 6, 6, 5, 5, 5, 6, 6, 5, 6, 5, 6, 5, 5, 6, 5, 6,
       5, 6, 6, 5])
```

```
y_test
```

```
array([6, 5, 7, 6, 5, 6, 5, 6, 4, 5, 5, 5, 6, 5, 6, 6, 7, 5, 5, 4, 7, 6,
       6, 4, 6, 5, 5, 7, 5, 6, 5, 6, 5, 6, 7, 7, 5, 6, 6, 7, 5, 7, 6, 6,
       5, 5, 6, 6, 6, 5, 5, 5, 6, 6, 6, 5, 5, 5, 6, 5, 5, 6, 6, 6, 5, 6,
       5, 5, 6, 6, 6, 6, 4, 6, 5, 6, 5, 5, 5, 5, 6, 6, 5, 6, 6, 6, 5, 6, 5,
       5, 5, 5, 6, 4, 5, 7, 6, 6, 5, 6, 5, 8, 6, 6, 6, 5, 5, 5, 5, 7, 5,
       6, 5, 7, 5, 6, 6, 6, 7, 6, 6, 5, 7, 5, 5, 6, 6, 5, 5, 5, 6, 6, 6,
       6, 6, 6, 5, 6, 5, 8, 5, 6, 5, 6, 5, 4, 6, 7, 6, 5, 6, 6, 5, 5, 5,
       6, 6, 3, 6, 6, 6, 6, 6, 6, 6, 5, 5, 6, 6, 6, 6, 5, 5, 5, 8, 5, 6,
       6, 7, 7, 5, 5, 7, 5, 6, 6, 4, 5, 6, 5, 5, 6, 5, 6, 6, 5, 5, 5, 5,
       5, 5, 5, 6, 6, 5, 6, 6, 5, 6, 7, 6, 6, 6, 5, 5, 5, 6, 5, 6, 6, 5,
       5, 5, 6, 6, 5, 6, 6, 6, 3, 6, 5, 5, 7, 6, 7, 6, 6, 7, 7, 6, 5, 6,
       5, 5, 6, 5, 5, 5, 5, 6, 5, 5, 5, 6, 6, 5, 5, 5, 6, 7, 5, 6, 5, 6,
       5, 4, 5, 5, 6, 7, 6, 5, 5, 4, 5, 6, 7, 6, 6, 7, 5, 7, 5, 6, 6, 5,
       5, 5, 6, 6, 5, 6, 6, 6, 5, 6, 6, 5, 6, 5, 6, 5, 6, 6, 6, 6, 5, 5,
       6, 5, 5, 6, 5, 5, 5, 6, 4, 5, 4, 6, 5, 5, 5, 8, 6, 6, 7, 6, 5, 5,
       7, 5, 5, 5, 5, 6, 7, 5, 4, 5, 5, 5, 5, 8, 6, 5, 6, 5, 5, 7, 5, 6,
       5, 6, 7, 5, 6, 6, 5, 7, 5, 6, 5, 7, 5, 5, 6, 7, 6, 4, 6, 6, 4, 7,
       6, 5, 5, 7, 6, 6, 6, 6, 5, 6, 6, 6, 6, 5, 6, 7, 6, 6, 5, 7, 5, 6,
       5, 5, 7, 5])
```

#10. Accuracy

```
#10. Evaluation : Accuracy Score
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)*100
```

```
87.5
```

Accurate Score of the model is **<u>87.5</u>**

<u>#INDIVIDUAL PREDICTION & CUSTOM PREDICTION</u>

```
[ ] #Individual Prediction
    model.predict([[7.4,0.700,0.00,1.9,0.076,11.0,34.0,0.99780,3.51,0.56,9.4]])
```

```
array([5])
```

```
[ ] model.predict([[5.5,0.842,0.10,2.8,0.852,36.0,96.0,0.96632,8.41,8.56,10.4]])
```

```
array([5])
```

```
# Custom Prediction
model.predict([[5.9,0.550,0.10,2.2,0.062,39.0,51.0,0.99512,3.52,0.76,11.2]])
```

```
array([6])
```

<u>~~ END OF PROBLEM STATEMENT - 1</u>

Problem Statement – 2:

Create any of the Image Processing Projects using NumPy and/or OpenCV.

Image Processing:

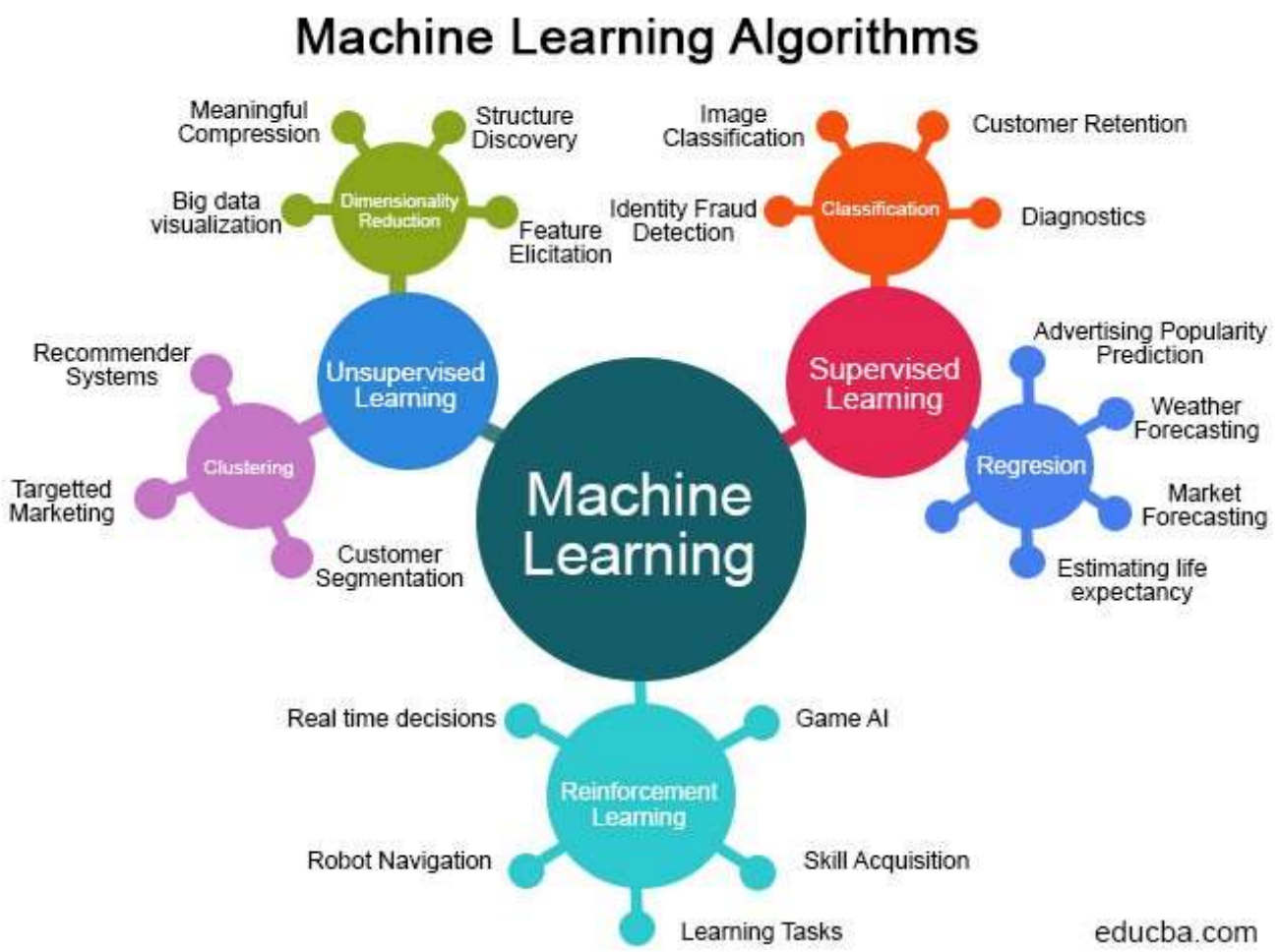
Image processing is done **to enhance an existing image or to sift out important information from it**. This is important in several Deep Learning-based Computer Vision applications, where such pre-processing can dramatically boost the performance of a model

Numpy:

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin and called Numeric (also variously known as the "**Numerical Python extensions**" or "NumPy").

OpenCV:

OpenCV (**Open Source Computer Vision Library**) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

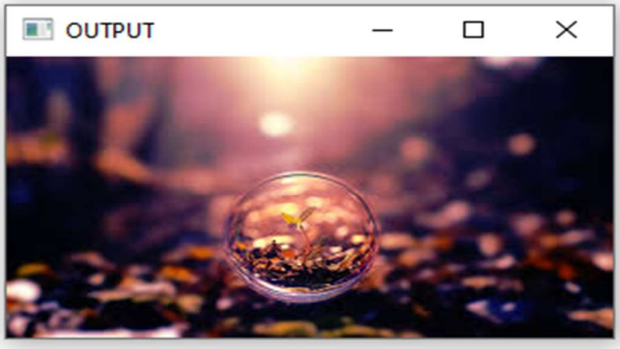


#1. READ an IMAGE

1.py - D:/python/1.py (3.10.9)

File Edit Format Run Options Window Help

```
#1.READ an IMAGE

import cv2 #importing the opencv/computer vision lib
img = cv2.imread('12345.jfif') #We are reading the image
cv2.imshow('OUTPUT',img) #We are displaying the image

cv2.waitKey(0)#3000 is 3000 milliseconds,0 will make sure , the image stays fore
cv2.destroyAllWindows()
```

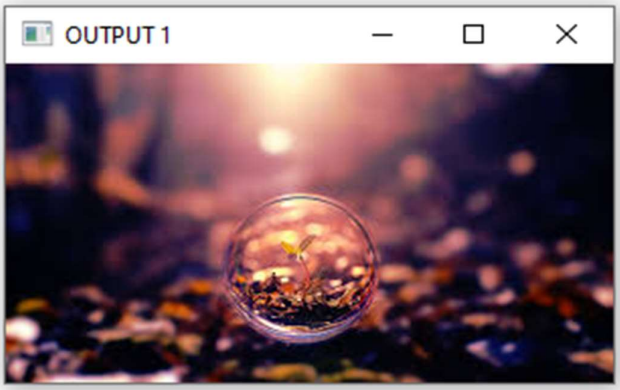


#2. Creating a Duplicate Image using Python

2.py - D:/python/2.py (3.10.9)

File Edit Format Run Options Window Help

```
#2.Creating a Duplicate Image using Python

import cv2
img = cv2.imread('12345.jfif')#reading the image
cv2.imshow('OUTPUT 1',img)#displaying the image

cv2.imwrite('nithin.png',img)#CReates a new image

cv2.waitKey(0)
cv2.destroyAllWindows()
```



#3. Read the Information about the image

3.py - D:/python/3.py (3.10.9)

File Edit Format Run Options Window Help

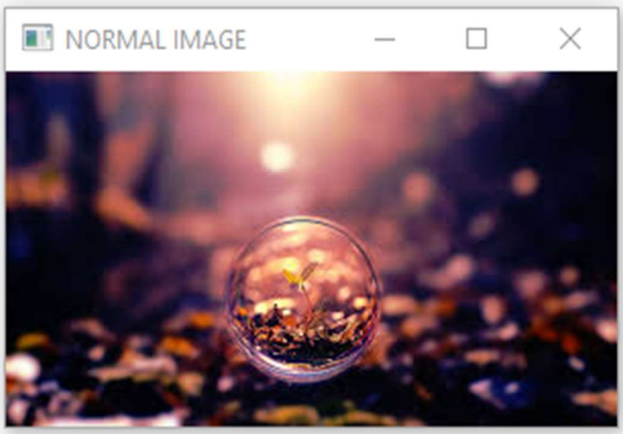
```
#3.Read the INFORMATION about the image

import cv2
img = cv2.imread('12345.jfif')
print(img.shape)


#OUTPUT: (168, 300, 3)
#344 is the Height of the Image in Pixels
#612 is the Width od the Image in Pixels
#3 is the depth/channel value - 3 by default(3 primary colors)
```

#4. GRAYSCALE IMAGE (BLACK and WHITE IMAGE)

4_method_1.py - D:/python/4_method_1.py (3.10.9)

File Edit Format Run Options Window Help

```
#4.GRAYSCALE IMAGE (BLACK and WHITE IMAGE)

import cv2
img = cv2.imread('12345.jfif')#reading the image
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)#conversion into grayscale
cv2.imshow('NORMAL IMAGE',img) #displays the original image
cv2.imshow('GRAY SCALE IMAGE',gray)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



Another Method for Scaling

4_method_2.py - D:/python/4_method_2.py (3.10.9)

File Edit Format Run Options Window Help

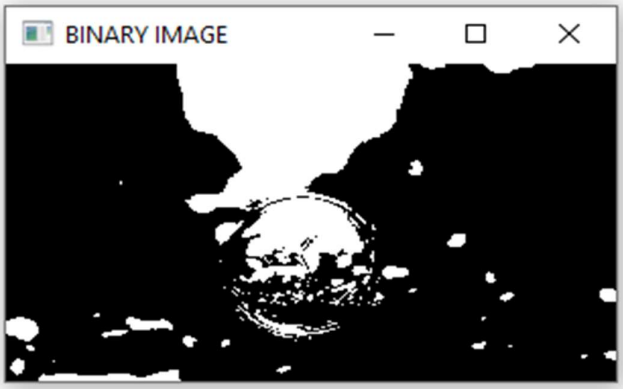
```
#4.GRAYSCALE SHORT PROGRAM

import cv2
img = cv2.imread('12345.jfif',0) #0 here converts the image into grayscale
cv2.imshow('GRAYSCALE IMAGE',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

#5. BINARY IMAGE CONVERSION (HIGH CONTRAST IMAGE)

5.py - D:/python/5.py (3.10.9)

File Edit Format Run Options Window Help

```
#5.BINARY IMAGE CONVERSION(HIGH CONTRAST IMAGE)

import cv2
img = cv2.imread('12345.jfif',0) #gray scale image
cv2.imshow('GRAY SCALE IMAGE',img)#displaying the grayscale image
cv2.waitKey(2000)

#============cv2.threshold(src,thresh,max value,conversion type)
ret,binary = cv2.threshold(img,127,255,cv2.THRESH_BINARY)#converts image from gr
#ret and binary are 2 varibales to be taken .So here ret is a dummy variable

cv2.imshow('BINARY IMAGE',binary)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

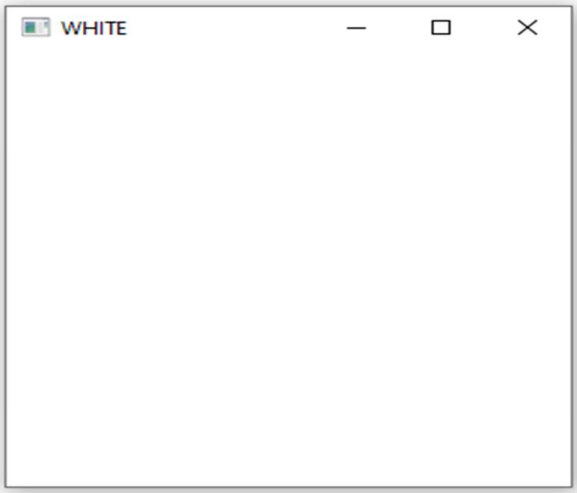


#6. SOLID BACKGROUND (BLACK or WHITE BACKGROUND)

6_white.py - D:/python/6_white.py (3.10.9)

File Edit Format Run Options Window Help

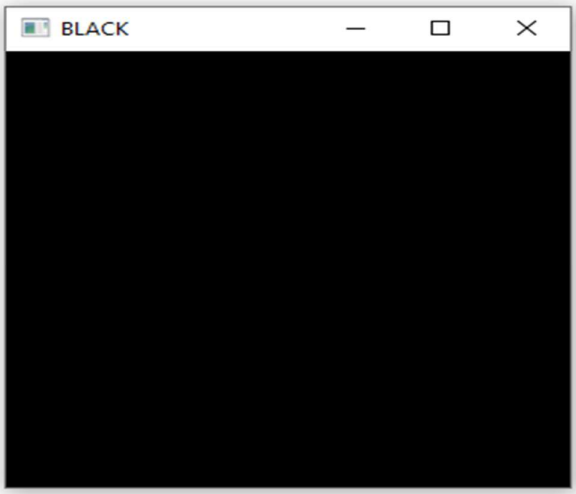
```
#6.SOLID BACKGROUND(BLACK or WHITE BACKGROUND)
#WHITE COLOR

import cv2
import numpy as np

img = np.ones((300,300,3))#500 is the length,500 is the width,3 is depth/channel
#In the above line , we are creating a white background

cv2.imshow('WHITE',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



6_balck.py - D:/python/6_balck.py (3.10.9)

File Edit Format Run Options Window Help

```
#6.SOLID BACKGROUND(BLACK or WHITE BACKGROUND)
#BLACK COLOR

import cv2
import numpy as np

img = np.zeros((300,300,3))#500 is the length,500 is the width,3 is depth/channe
#In the above line , we are creating a white background

cv2.imshow('BLACK',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

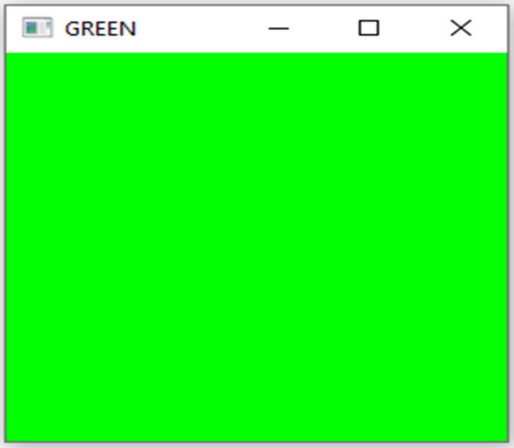


#7. SOLID COLORS (RED,GREEN and BLUE)

7_green.py - D:/python/7_green.py (3.10.9)

File Edit Format Run Options Window Help

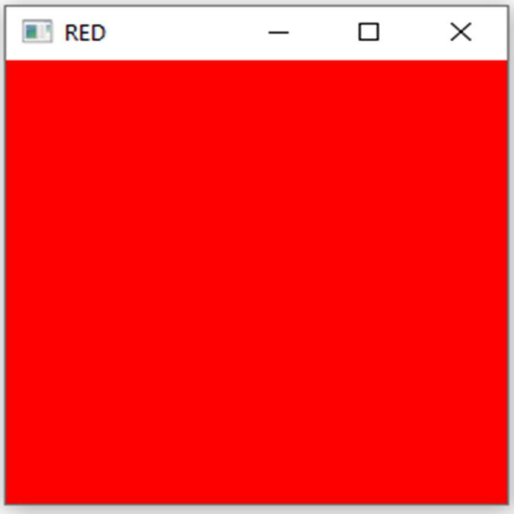
```
#7.solid colors
#GREEN
import cv2
import numpy as np
img = np.zeros((250,250,3)) #creates a black image

img[:] = 0,255,0 #Assigning the color(B,G,R)

cv2.imshow('GREEN',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



7_red.py - D:/python/7_red.py (3.10.9)

File Edit Format Run Options Window Help

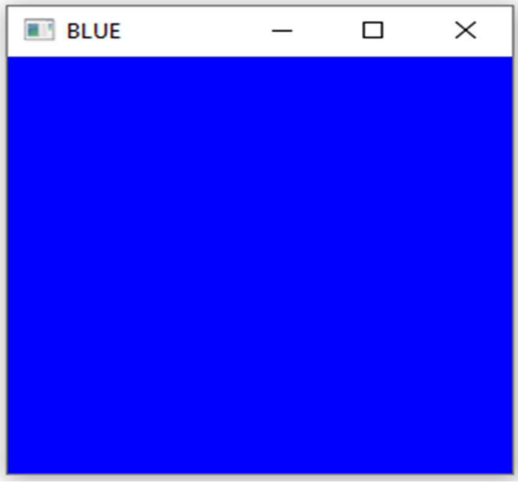
```
#7.SOLID COLORS (RED,GREEN and BLUE)
#RED
import cv2
import numpy as np
img = np.zeros((250,250,3)) #creates a black image

img[:] = 0,0,255 #Assigning the color(B,G,R)

cv2.imshow('RED',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



7_blue.py - D:/python/7_blue.py (3.10.9)

File Edit Format Run Options Window Help

```
#7. Solid Colors

#BLUE
import cv2
import numpy as np
img = np.zeros((250,250,3)) #creates a black image

img[:] = 255,0,0 #Assigning the color(B,G,R)

cv2.imshow('BLUE',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



#8. CHECKER BOARD

8.py - D:/python/8.py (3.10.9)

File Edit Format Run Options Window Help

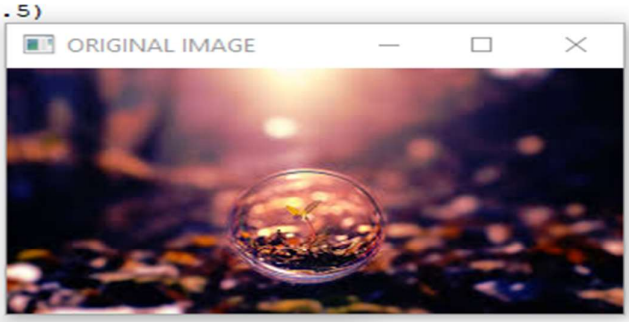
```
#8.CHECKER BOARD

import cv2
import numpy as np

img = np.zeros((200,200,3)) # BLACK BACKGROUND

img[0:100,0:100]= 255,255,255
img[100:200,100:200] = 255,255,255

cv2.imshow('CHECKER BOARD',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

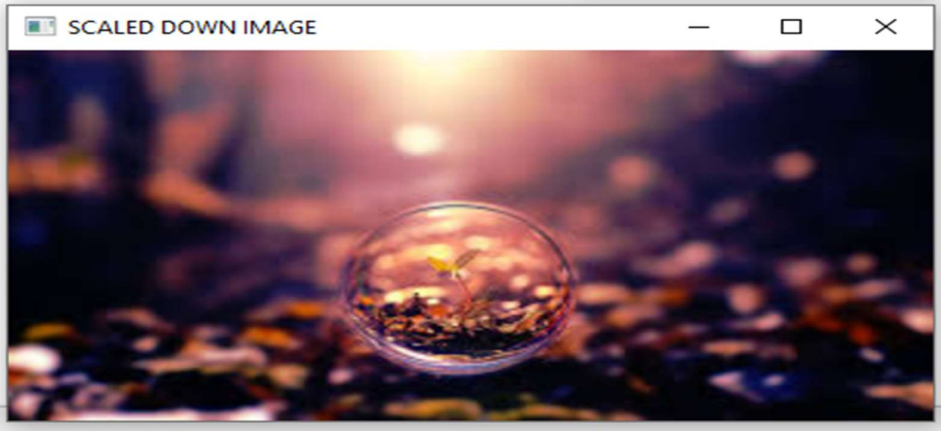
#9. RESIZING/SCALING the IMAGE

8_2.py - D:/python/8_2.py (3.10.9)

File Edit Format Run Options Window Help

```
#9.Resizing/Scaling
import cv2
import numpy as np

img = cv2.imread('12345.jfif')#reading the image
cv2.imshow('ORIGINAL IMAGE',img)
cv2.waitKey(500)

#1.Let us scale down the image to 150%
img1 = cv2.resize(img,None,fx = 1.5,fy = 1.5)
cv2.imshow('SCALED DOWN IMAGE',img1)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



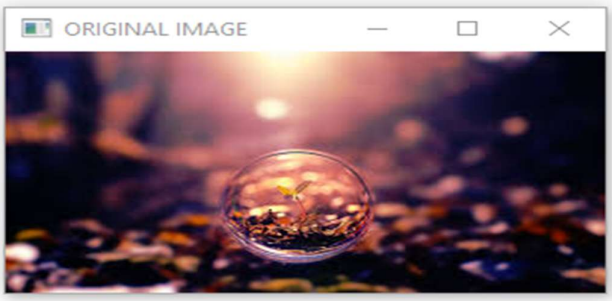


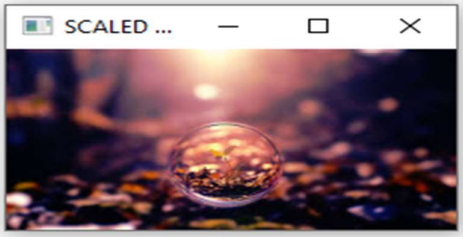
Ln: 15 Col

8_1.py - D:/python/8_1.py (3.10.9)

File Edit Format Run Options Window Help

```
#9.Resizing/Scaling
import cv2
import numpy as np

img = cv2.imread('12345.jfif')#reading the image
cv2.imshow('ORIGINAL IMAGE',img)
cv2.waitKey(500)



#1.Let us scale down the image to 75%
img1 = cv2.resize(img,None,fx = 0.75,fy = 0.75)
cv2.imshow('SCALED DOWN IMAGE',img1)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

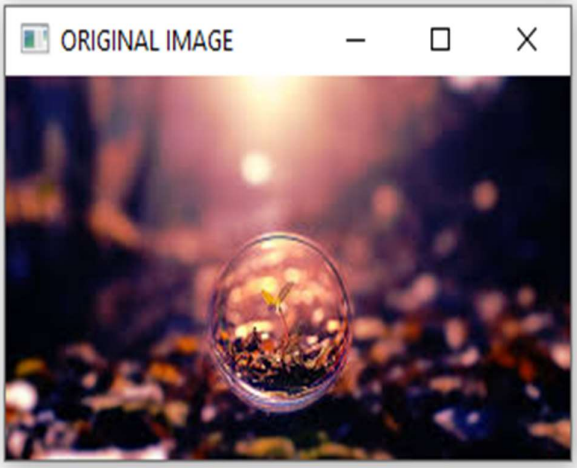


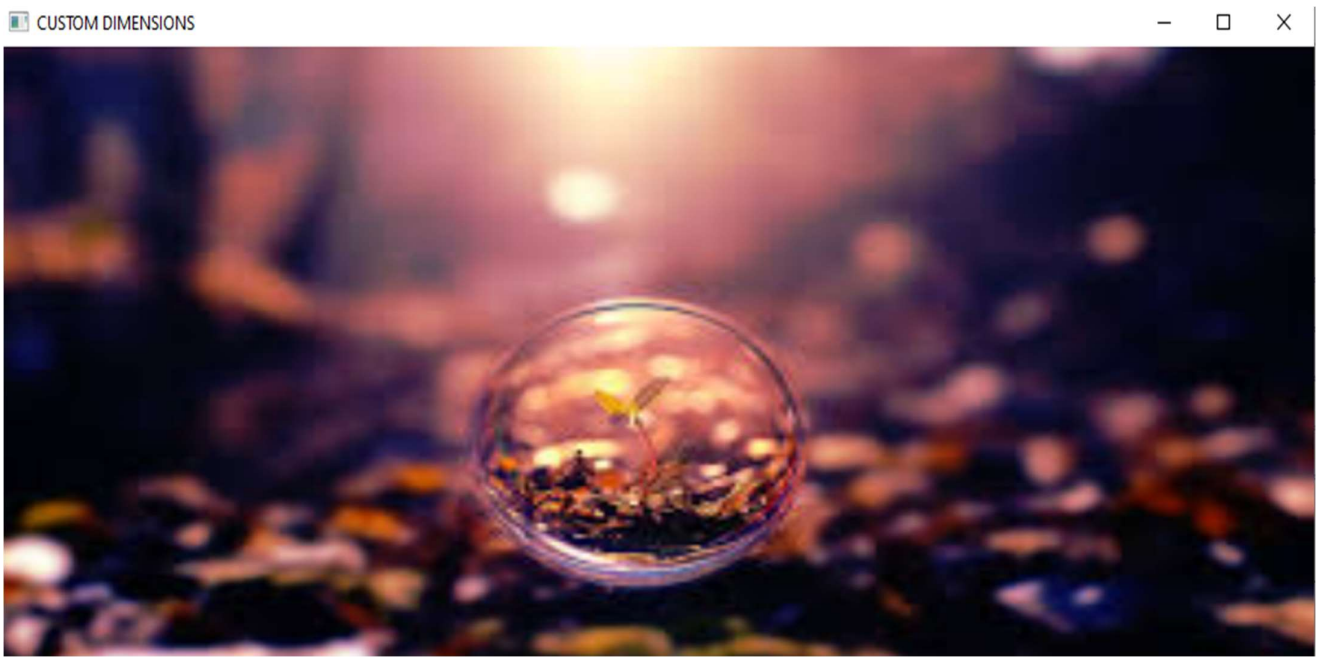


9_3.py - D:/python/9_3.py (3.10.9)

File Edit Format Run Options Window Help

```
#9.Resizing/Scaling
import cv2
import numpy as np

img = cv2.imread('12345.jfif')#reading the image
cv2.imshow('ORIGINAL IMAGE',img)
cv2.waitKey(500)

#3.Scaling using custom dimensions
img3 = cv2.resize(img,(1000,400))
cv2.imshow('CUSTOM DIMENSIONS',img3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```





#10. RECTANGLE/SQUARE

10.py - C:/Users/Dell/AppData/Local/Programs/Python/Python310/10.py (3.10.9)

File Edit Format Run Options Window Help

```
#10.RECTANGLE/SQUARE

import cv2
import numpy as np

img = np.ones((500,500,3)) #black background

#cv2.rectangle(src,point1,point2,color,thickness)
cv2.rectangle(img,(200,200),(400,400),(250,0,0),5)
cv2.imshow('RECTANGLE',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

RECTANGLE

#11. LIVE VIDEO from the WEBCAM

11.py - D:/python/11.py (3.10.9)

File Edit Format Run Options Window Help

```
#11.LIVE VIDEO from the WEBCAM

import cv2

cap = cv2.VideoCapture(0) #0 here considers the default webcam port

while True:
    ret,frame = cap.read()#from the cap variable , we read the video
    #ret and frame are 2 variables to be considered
    #We only use frame variable , ret is dummy

    cv2.imshow('My Live Sketch',frame)
    if cv2.waitKey(1) == 13:
        break

cap.release() #It release the default webcam port
cv2.destroyAllWindows()
```

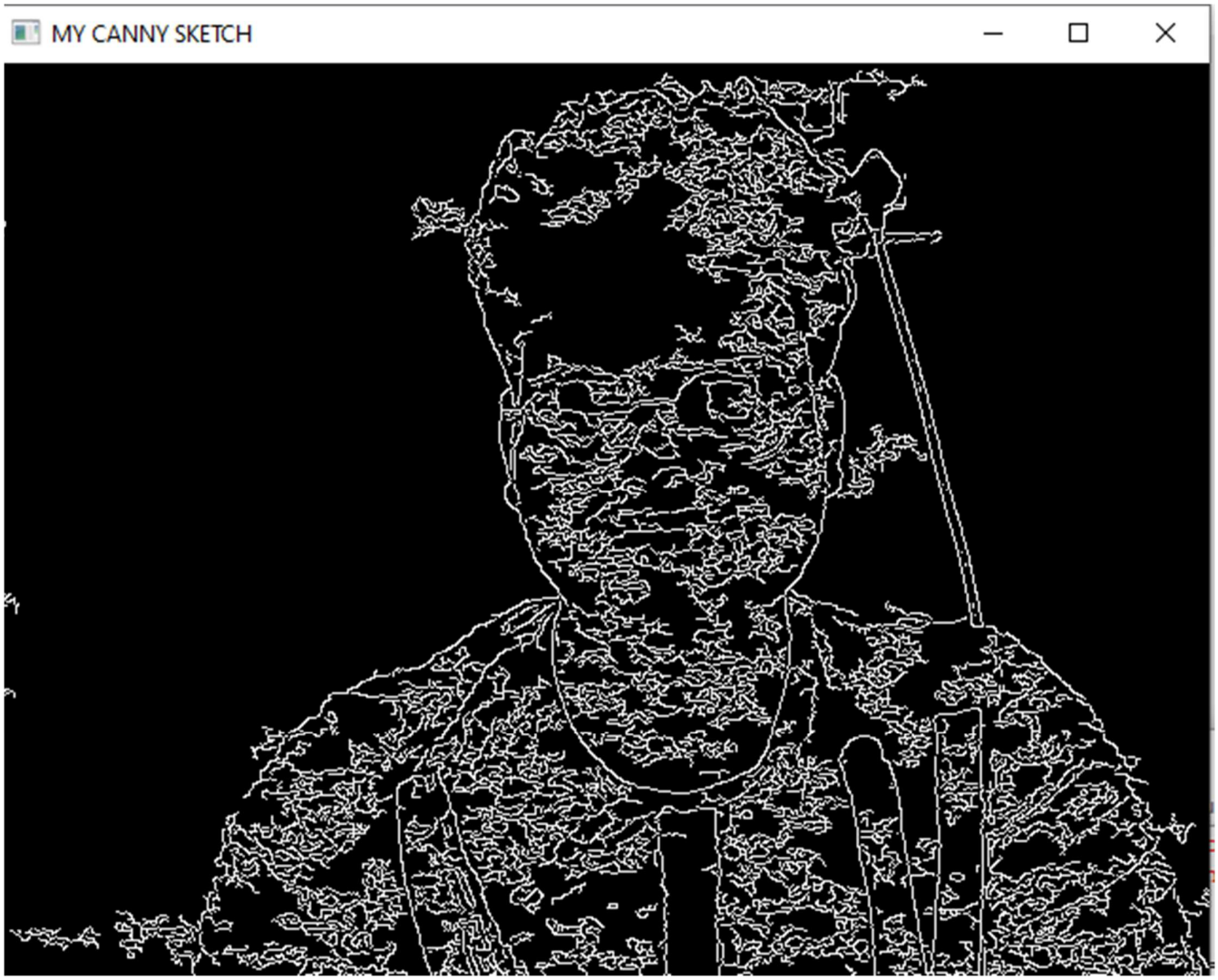
My Live Sketch

#12.MY CANNY (EDGE DETECTION TECHNIQUE) SKETCH

12.py - D:/python/12.py (3.10.9)

File Edit Format Run Options Window Help

```
#12.MY CANNY(EDGE DETECTION TECHNIQUE) SKETCH

import cv2
cap = cv2.VideoCapture(0)

while True:
    ret,frame = cap.read()
    canny = cv2.Canny(frame,20,150)#20,150 are the threshold values for best edg
    cv2.imshow('MY CANNY SKETCH',canny)

    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



Problem Statement – 2: (Optional)

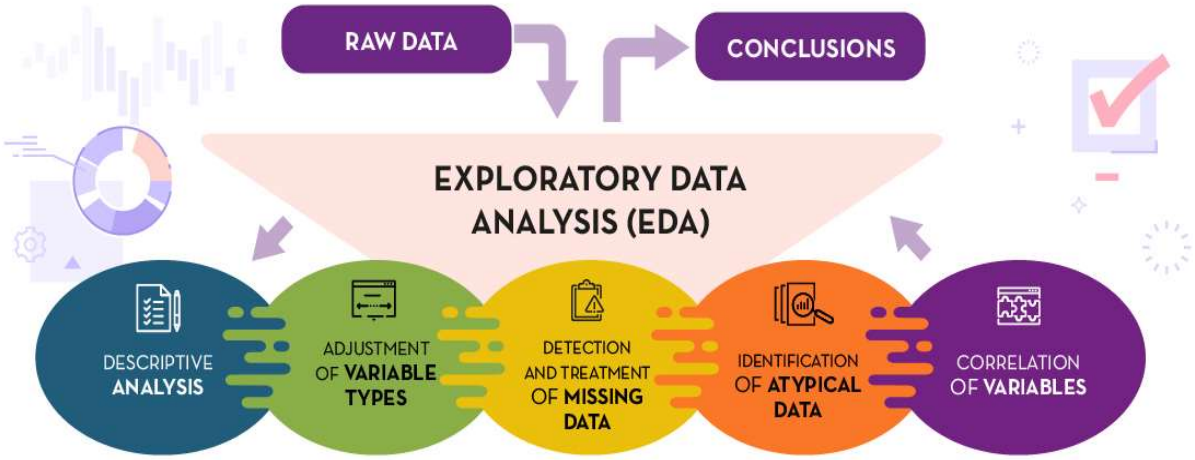
Choose any dataset of your choice and Perform Exploratory Data Analysis for Atleast 15 different facts/Conclusions.

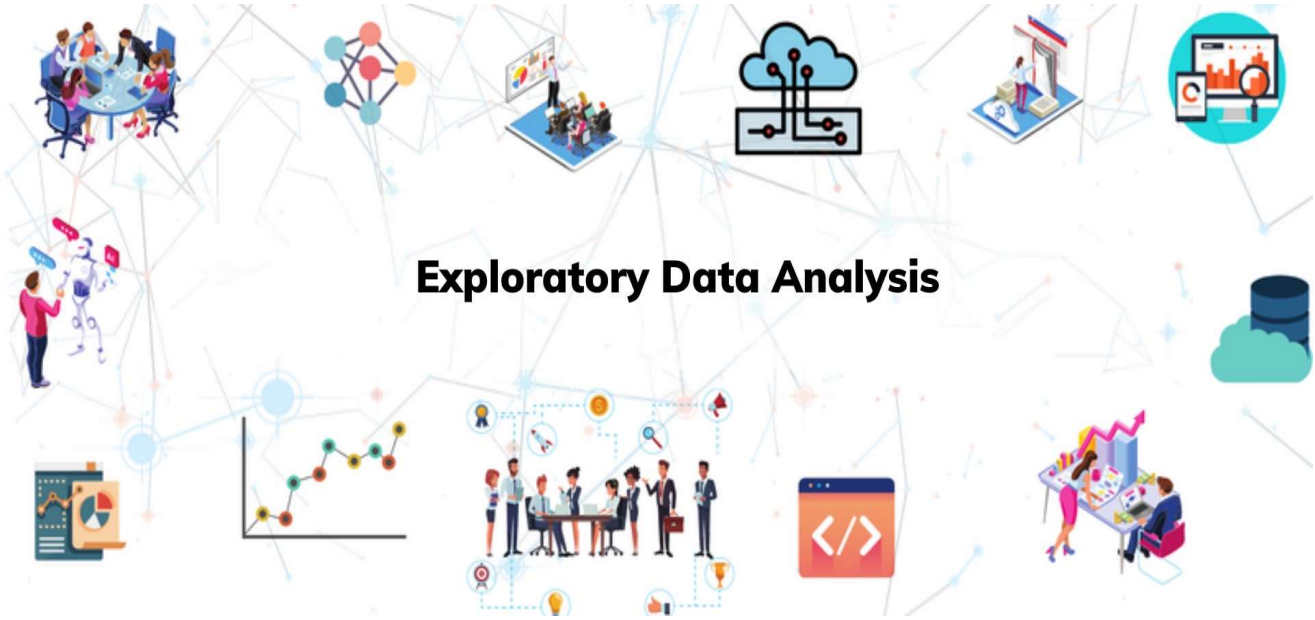
Exploratory Data Analysis:

Exploratory data analysis (EDA) involves **using graphics and visualizations to explore and analyse a data set**. The goal is to explore, investigate and learn, as opposed to confirming statistical hypotheses.

What are the types of exploratory data analysis?

The four types of EDA are **univariate non-graphical, multivariate non- graphical, univariate graphical, and multivariate graphical**.





FACTS/CONCLUSIONS:

I took dataset from www.Kaggle.com, the data is about Programming Language that Includes S.no, Year, Chief Developer, etc.

I have to Apply Exploratory Data Analysis on this Data Set Then I performed so many things On that data, They are:

- Import pandas to my data
- Dataset shape
- Dataset size
- Dataset info
- Dataset sum
- Dataset Unique
- Import Seaborn to my data
- Count plot to Chief Developer
- Count Plot to Predecessors
- Count value to Chief Developer
- Count Value to Predecessors
- Moreover, I done group by for Chief Developer

I Performed above Furnished Analysis on my Dataset Successfully Without Errors……

➔ Import pandas to my data:

```
import pandas as pd
df=pd.read_csv('/content/programminglanguage.csv')
df
```

	Year	Name	Chief developer, company	Predecessor(s)
0	1804	Jacquard machine	Joseph Marie Jacquard	none (unique language)
1	1879	Begriffsschrift	Gottlob Frege	none (unique language)
2	1944	ENIAC coding system	John von Neumann, John Mauchly, J. Presper Eck...	none (unique language)
3	1948	Plankalkül	Konrad Zuse	none (unique language)
4	1948	ARC/Birkbeck Assembler	Kathleen Booth	ENIAC Short Code
...	...	...	...	...
389	2018	Fortran 2018	ISO/IEC JTC1/SC22/WG5 N2150:2018	Fortran 2008
390	2019	Bosque	Mark Marron, Microsoft	JavaScript, TypeScript, ML
391	2020	C++20	C++ ISO/IEC 14882:2020	C++, Standard C, C
392	2021	Microsoft Power Fx	Vijay Mital, Robin Abraham, Shon Katzenberger,...	Excel formulas
393	2022	Carbon	Google	C++, Rust

394 rows × 4 columns

➔ Dataset Shape:

```
df.shape
```

```
(394, 4)
```

➔ Dataset Size:

```
df.size
```

```
1576
```

➔ Dataset info:

```
df.info
```

```
<bound method DataFrame.info of     Year                 Name  \
0    1804       Jacquard machine
1    1879        Begriffsschrift
2    1944    ENIAC coding system
3    1948             Plankalkül
4    1948  ARC/Birkbeck Assembler
..    ...                    ...
389  2018          Fortran 2018
390  2019                Bosque
391  2020                  C++20
392  2021     Microsoft Power Fx
393  2022                Carbon

                          Chief developer, company  \
0                            Joseph Marie Jacquard
1                                  Gottlob Frege
2    John von Neumann, John Mauchly, J. Presper Eck...
3                                    Konrad Zuse
4                                 Kathleen Booth
..                                              ...
389                ISO/IEC JTC1/SC22/WG5 N2150:2018
390                         Mark Marron, Microsoft
391                        C++ ISO/IEC 14882:2020
392  Vijay Mital, Robin Abraham, Shon Katzenberger,...
393                                          Google

              Predecessor(s)
0       none (unique language)
1       none (unique language)
2       none (unique language)
3       none (unique language)
4            ENIAC Short Code
..                        ...
389              Fortran 2008
390  JavaScript, TypeScript, ML
391        C++, Standard C, C
392           Excel formulas
393                C++, Rust

[394 rows x 4 columns]>
```

➔ Dataset Sum:

```
[ ] df.isnull().sum()
```

```
Year                        0
Name                        0
Chief developer, company    11
Predecessor(s)              37
dtype: int64
```

➔ Dataset Unique:

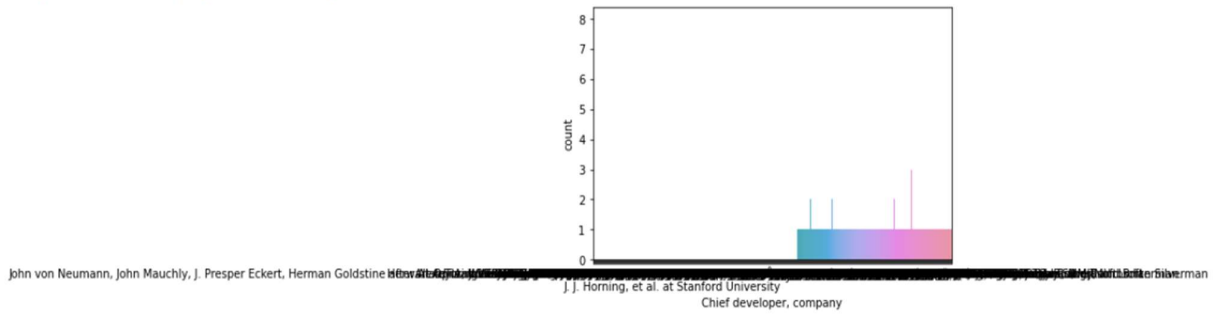
```
df.nunique()
```

```
Year                        78
Name                       393
Chief developer, company   335
Predecessor(s)             254
dtype: int64
```

➔ Count plot to Chief Developer:

```
import seaborn as sns
sns.countplot(x='Chief developer, company',data=df)
```

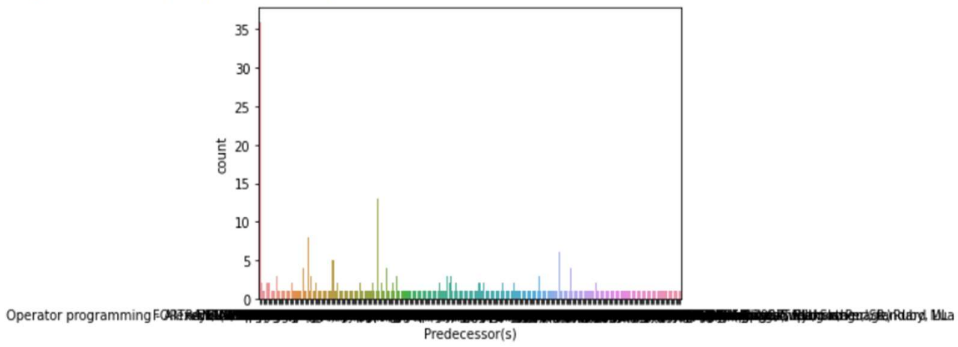
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f37ba1c62e0>
```



➔ Count plot to Predecessors:

```
import seaborn as sns
sns.countplot(x='Predecessor(s)',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f37b95f51f0>
```



➔ Count value to Chief Developer:

```
df['Chief developer, company'].value_counts()
```

```
IBM                                                                 8
Microsoft                                                           8
Apple Computer                                                      5
Google                                                              3
Allen Newell, Cliff Shaw, Herbert A. Simon                          3
                                                                   ..
Brian Kernighan                                                     1
Allan Ballard, Paul Whaley at the University of British Columbia    1
Arvind and Gostelow, University of California, Irvine               1
James S. Miller, Benjamin M. Brosgol et al. at Intermetrics         1
Vijay Mital, Robin Abraham, Shon Katzenberger, Darryl Rubin, Microsoft  1
Name: Chief developer, company, Length: 335, dtype: int64
```

➔ Count value to Predecessors:

```
df['Predecessor(s)'].value_counts()
```

```
none (unique language)    36
BASIC                     13
ALGOL 60                   8
C++, Standard C, C         6
LISP                       5
                          ..
InterPress                 1
Euclid                     1
Speakeasy-3                1
Pascal, C, ALGOL 68        1
C++, Rust                  1
Name: Predecessor(s), Length: 254, dtype: int64
```

➔ Group by for Chief Developer :

```
df.groupby('Chief developer, company').size()
```

```
Chief developer, company
ABB                                      1
ACM/GAMM                                 1
ANSI/ISO Standard C++                    1
ANSI/MIL-STD-1815A unchanged             1
ARA and Ada Europe (ISO/IEC 8652:2012)   1
                                        ..
Yukihiro Matsumoto                       1
Yves Caseau                              1
Zoltan Somogyi at University of Melbourne  1
designed by Intermetrics for NASA        1
many people at Apple Computer            1
Length: 335, dtype: int64
```

~~ END OF PROBLEM STATEMENT – 2

~~ END OF MAJOR PROJECT……