RINEX TECHNOLOGIES

MAJOR PROJECTS

NAME - ANURIMA GOSWAMI.

BRANCH - ELECTRONICS AND COMMUNICATION ENGINEERING.

YEAR - 4TH YEAR (7TH SEM).

COLLEGE - SWAMI VIVIKANANDA INSTITUTE OF SCIENCE AND TECHNOLOGY.

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to **Mr. AMEEN MANNA**. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

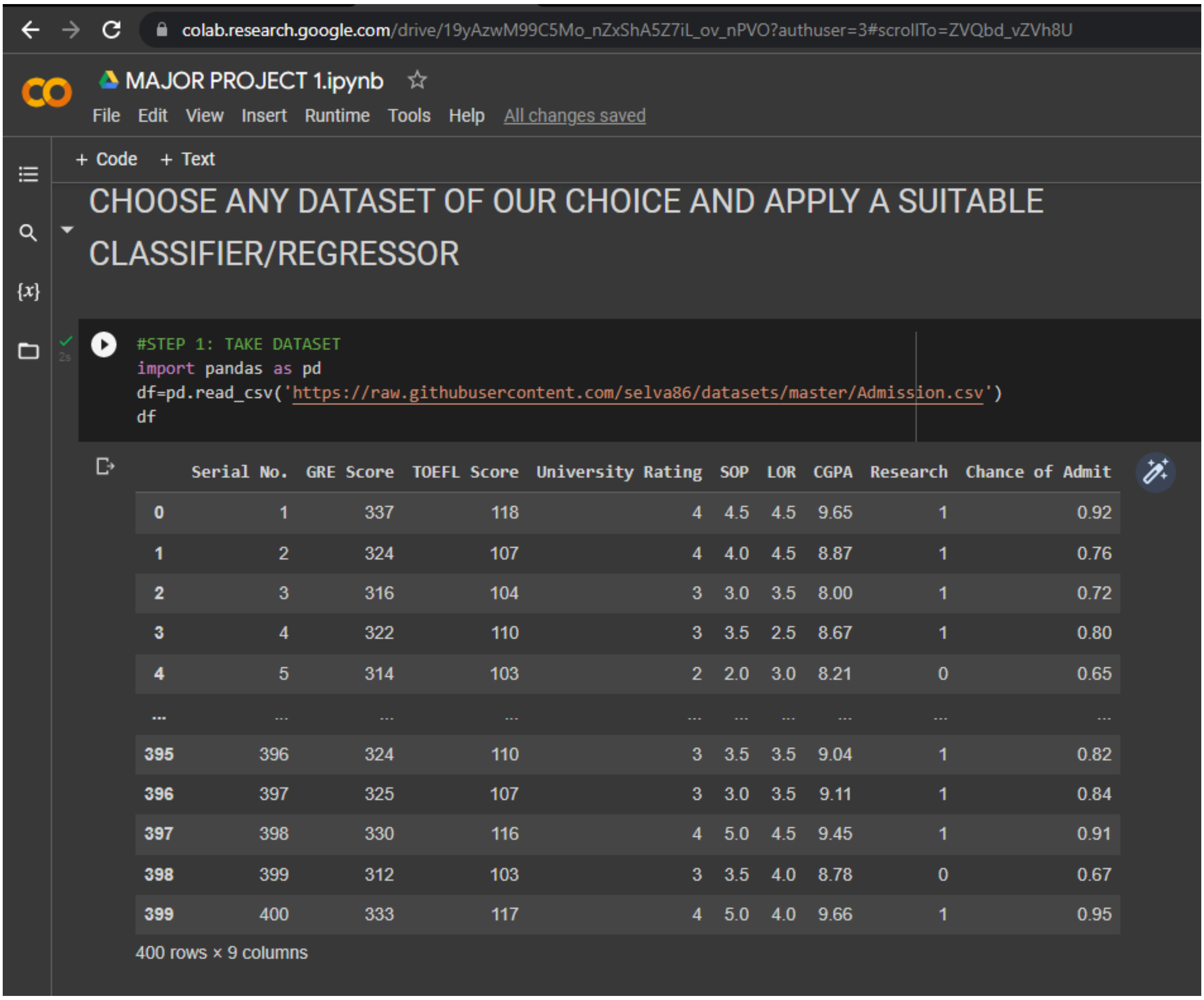
The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

MAJOR PROJECT 1:

APPLY A SUITABLE CLASSIFIER / REGRESSOR

Dataset:-

https://raw.githubusercontent.com/selva86/datasets/master/Admission.csv

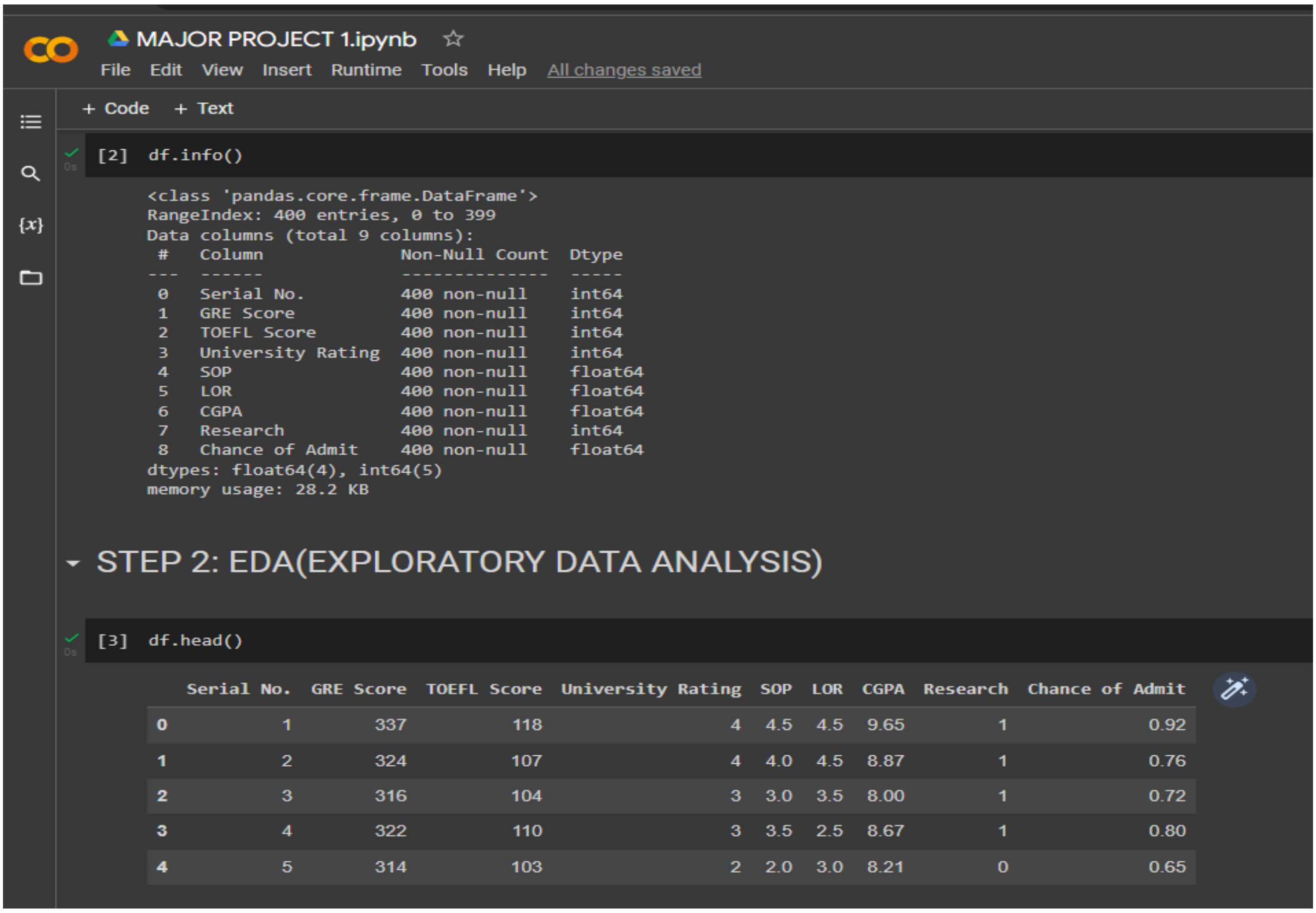

colab.research.google.com/drive/19yAzwM99C5Mo_nZxShA5Z7iL_ov_nPVO?authuser=3#scrollTo=ZVQbd_vZVh8U
MAJOR PROJECT 1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

CHOOSE ANY DATASET OF OUR CHOICE AND APPLY A SUITABLE CLASSIFIER/REGRESSOR

```
#STEP 1: TAKE DATASET
import pandas as pd
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Admission.csv')
df
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns



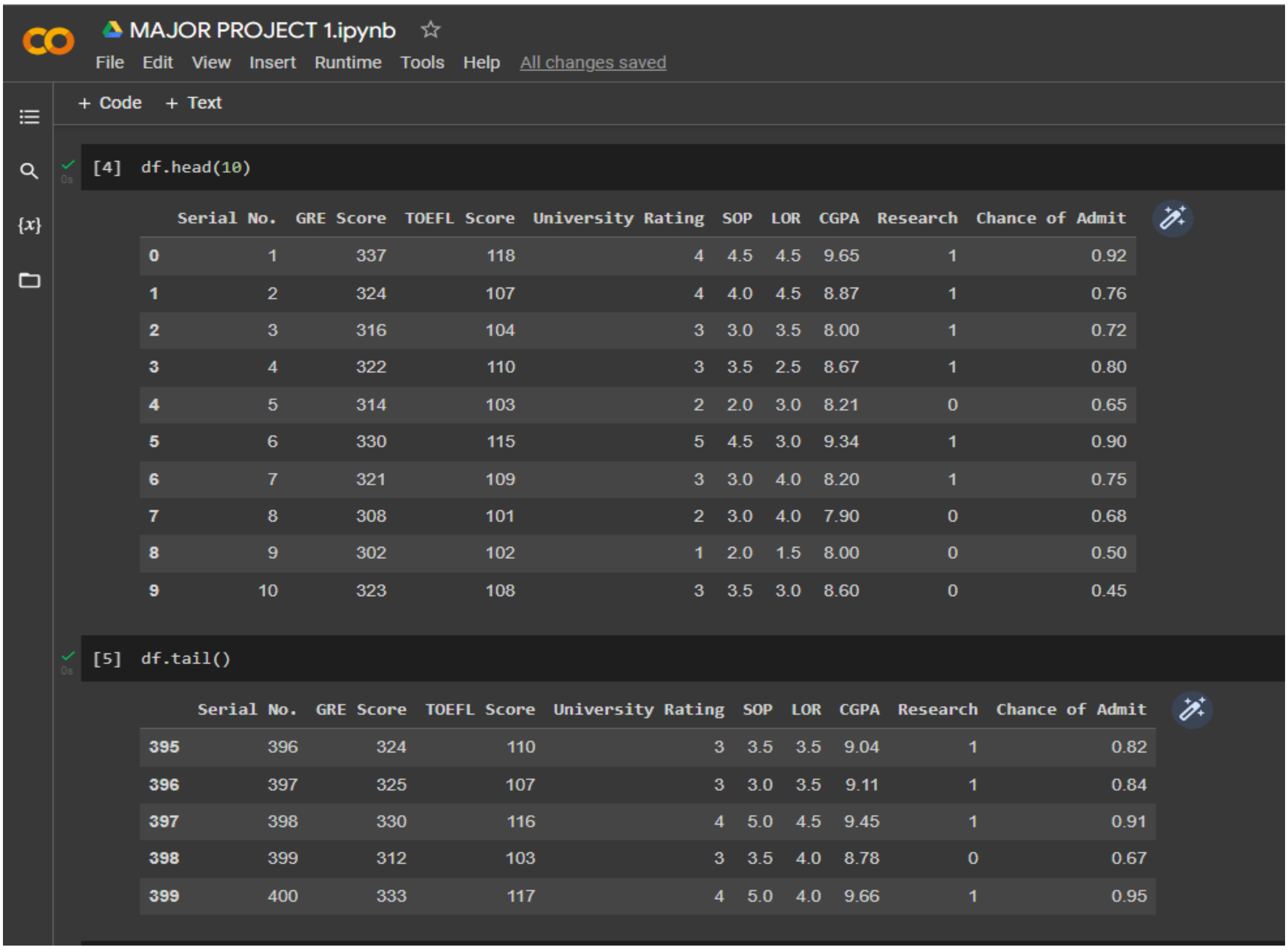
```
[2]  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          400 non-null    int64
 2   TOEFL Score        400 non-null    int64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

STEP 2: EDA(EXPLORATORY DATA ANALYSIS)

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

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

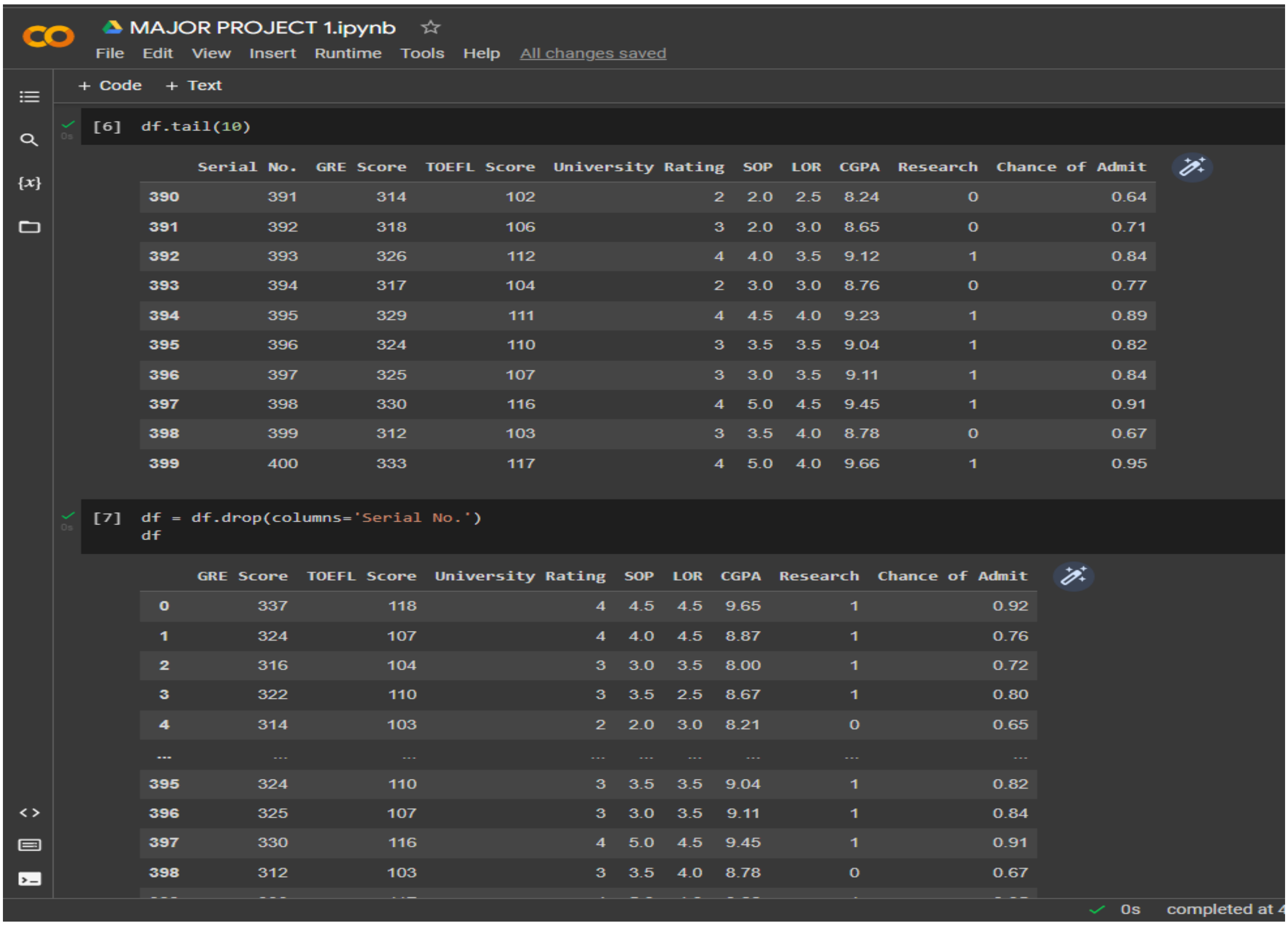


```
[4]  df.head(10)
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
5	6	330	115	5	4.5	3.0	9.34	1	0.90
6	7	321	109	3	3.0	4.0	8.20	1	0.75
7	8	308	101	2	3.0	4.0	7.90	0	0.68
8	9	302	102	1	2.0	1.5	8.00	0	0.50
9	10	323	108	3	3.5	3.0	8.60	0	0.45

```
[5]  df.tail()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

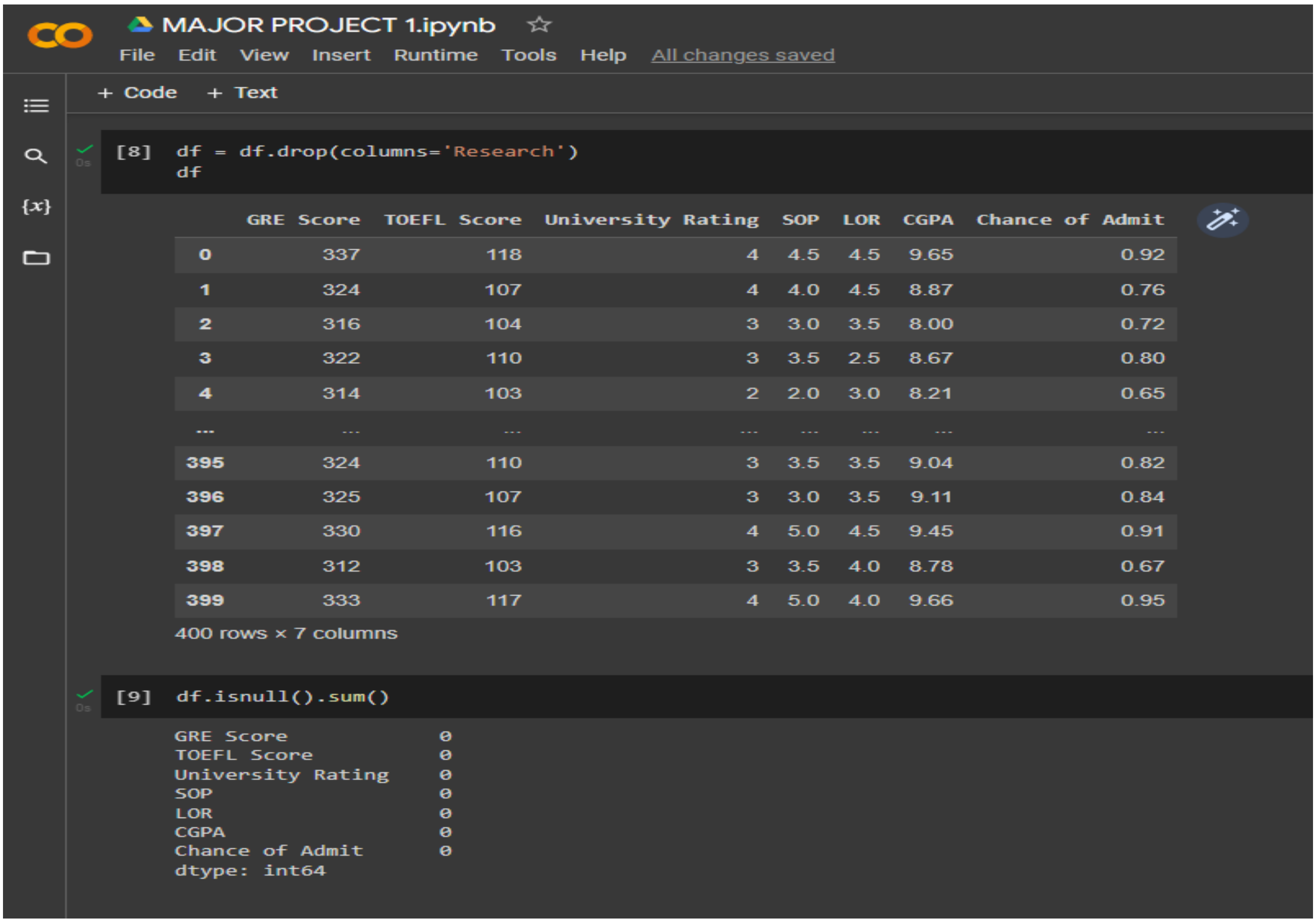


```
[6] df.tail(10)
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
390	391	314	102	2	2.0	2.5	8.24	0	0.64
391	392	318	106	3	2.0	3.0	8.65	0	0.71
392	393	326	112	4	4.0	3.5	9.12	1	0.84
393	394	317	104	2	3.0	3.0	8.76	0	0.77
394	395	329	111	4	4.5	4.0	9.23	1	0.89
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

```
[7] df = df.drop(columns='Serial No.')
    df
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0.82
396	325	107	3	3.0	3.5	9.11	1	0.84
397	330	116	4	5.0	4.5	9.45	1	0.91
398	312	103	3	3.5	4.0	8.78	0	0.67



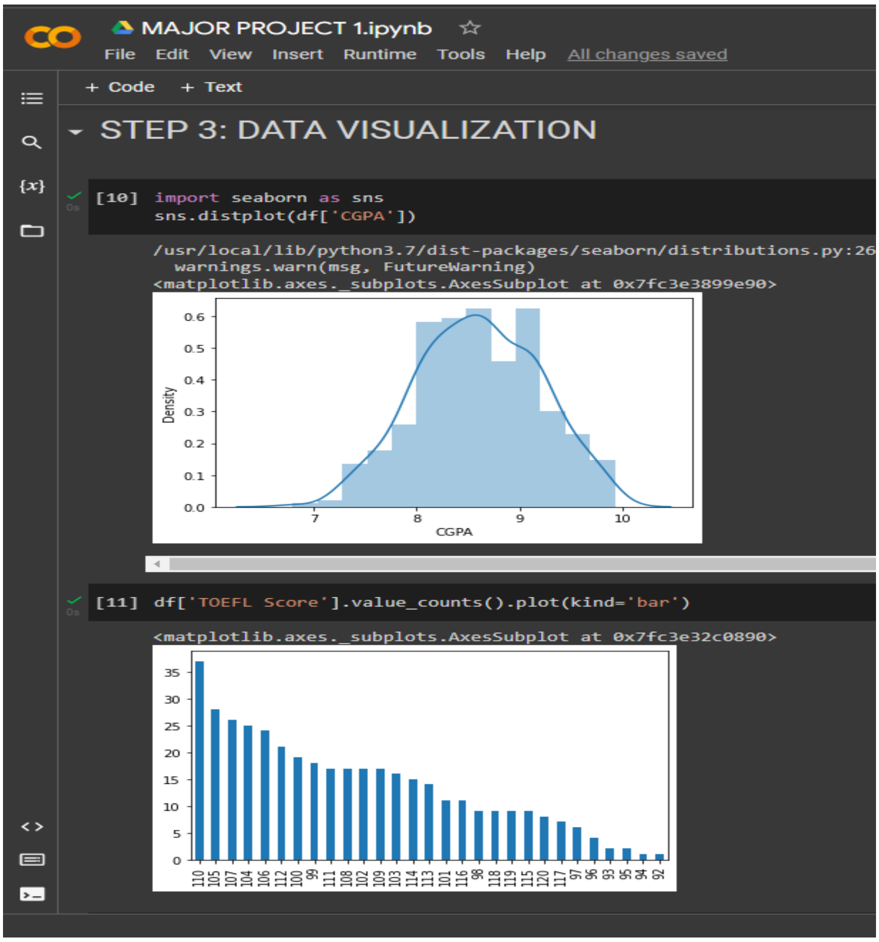
```
[8] df = df.drop(columns='Research')
    df
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Chance of Admit
0	337	118	4	4.5	4.5	9.65	0.92
1	324	107	4	4.0	4.5	8.87	0.76
2	316	104	3	3.0	3.5	8.00	0.72
3	322	110	3	3.5	2.5	8.67	0.80
4	314	103	2	2.0	3.0	8.21	0.65
...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	0.82
396	325	107	3	3.0	3.5	9.11	0.84
397	330	116	4	5.0	4.5	9.45	0.91
398	312	103	3	3.5	4.0	8.78	0.67
399	333	117	4	5.0	4.0	9.66	0.95

400 rows × 7 columns

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

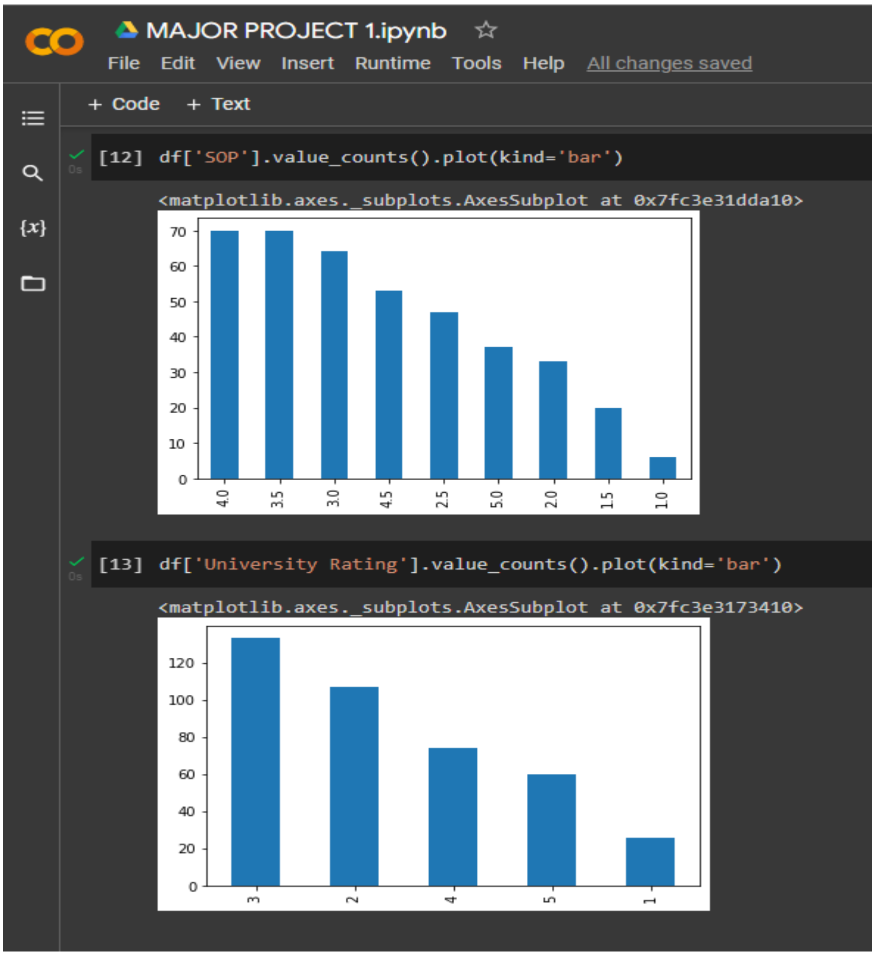
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Chance of Admit      0
dtype: int64
```



STEP 3: DATA VISUALIZATION

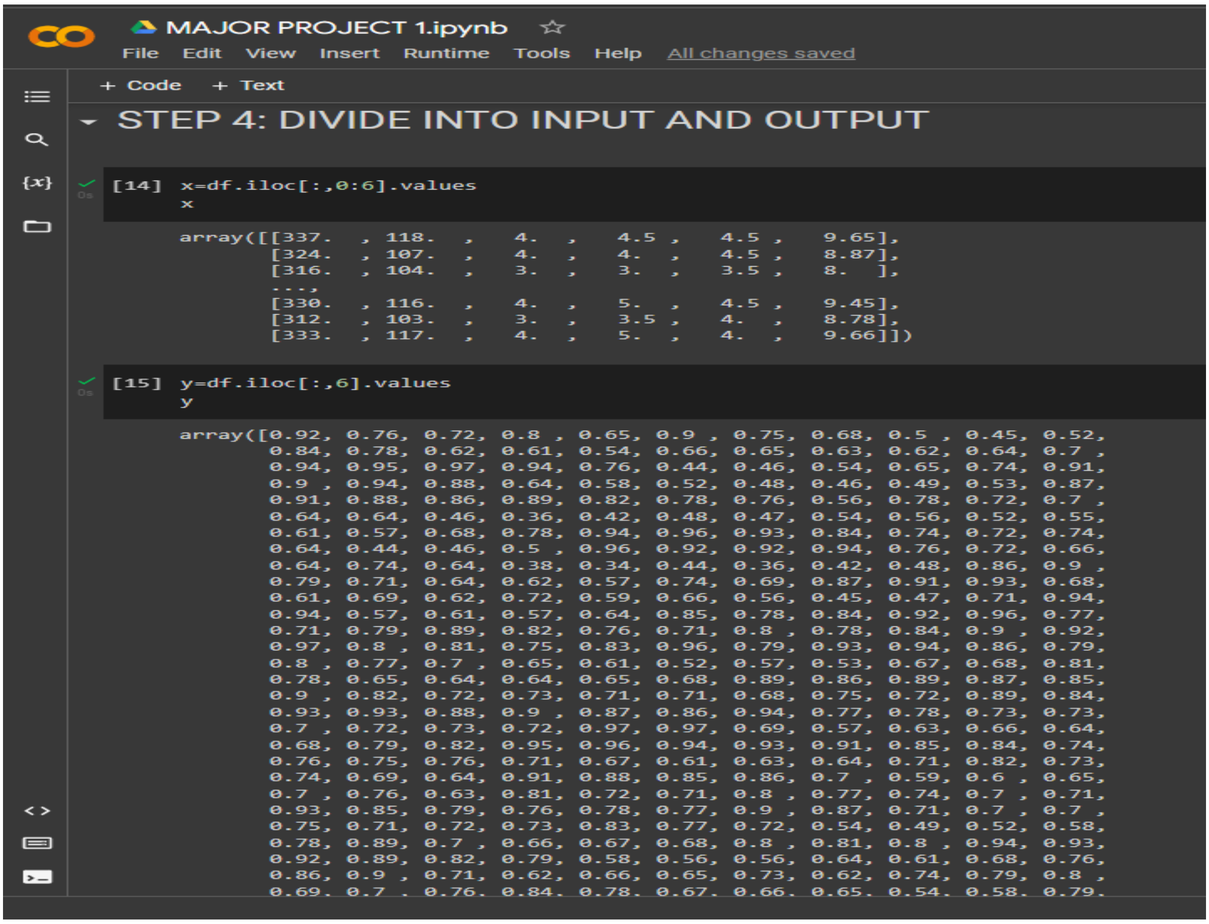
```
[10] import seaborn as sns
     sns.distplot(df['CGPA'])

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:26
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc3e3899e90>
```

```
[11] df['TOEFL Score'].value_counts().plot(kind='bar')

<matplotlib.axes._subplots.AxesSubplot at 0x7fc3e32c0890>
```



```
[12] df['SOP'].value_counts().plot(kind='bar')

<matplotlib.axes._subplots.AxesSubplot at 0x7fc3e31dda10>
```

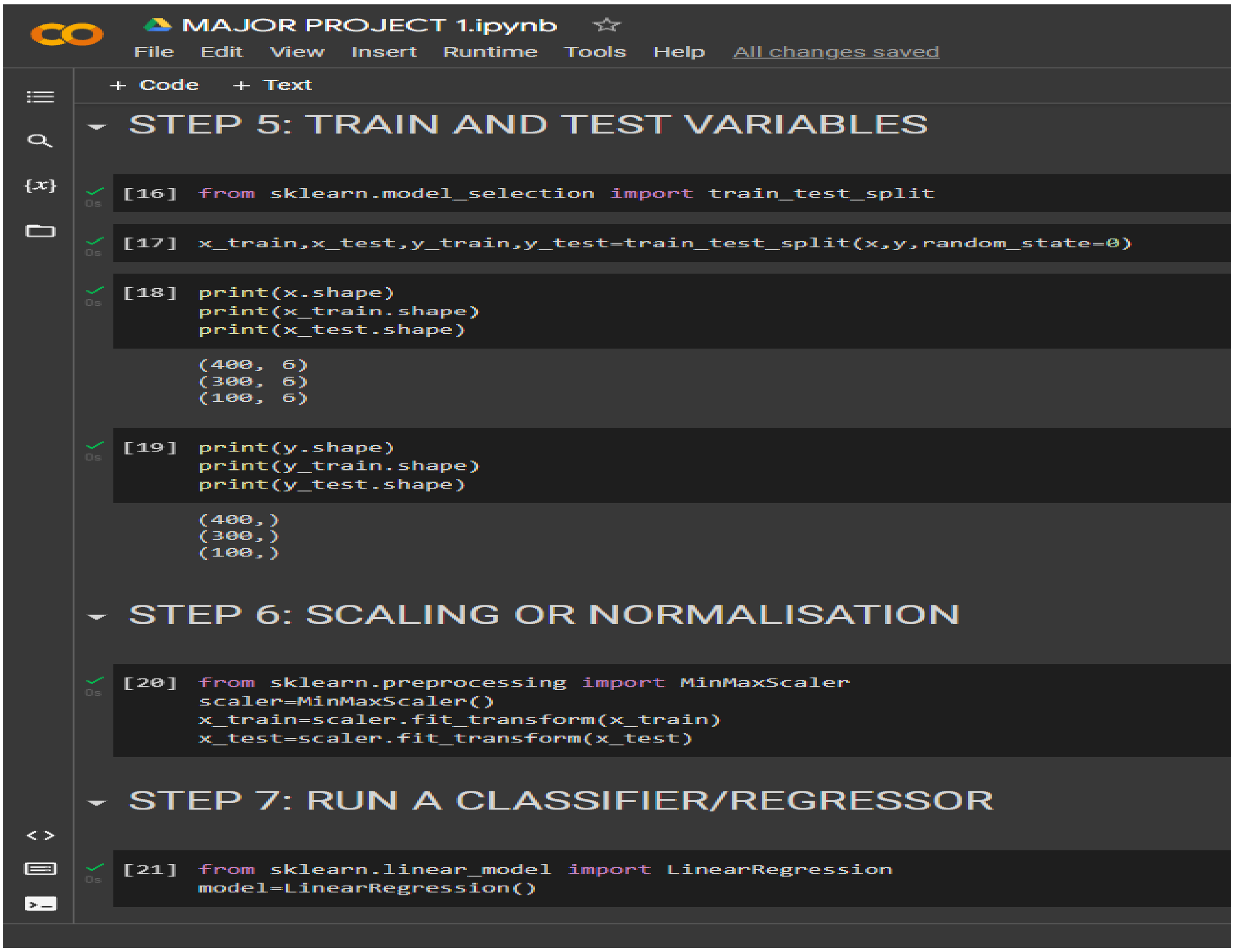
```
[13] df['University Rating'].value_counts().plot(kind='bar')

<matplotlib.axes._subplots.AxesSubplot at 0x7fc3e3173410>
```



STEP 4: DIVIDE INTO INPUT AND OUTPUT

```
[14] x=df.iloc[:,0:6].values
     x

     array([[337.  , 118.  ,   4.  ,   4.5 ,   4.5 ,   9.65],
            [324.  , 107.  ,   4.  ,   4.  ,   4.5 ,   8.87],
            [316.  , 104.  ,   3.  ,   3.  ,   3.5 ,   8.  ],
            ...,
            [330.  , 116.  ,   4.  ,   5.  ,   4.5 ,   9.45],
            [312.  , 103.  ,   3.  ,   3.5 ,   4.  ,   8.78],
            [333.  , 117.  ,   4.  ,   5.  ,   4.  ,   9.66]])
```

```
[15] y=df.iloc[:,6].values
     y

     array([0.92, 0.76, 0.72, 0.8 , 0.65, 0.9 , 0.75, 0.68, 0.5 , 0.45, 0.52,
            0.84, 0.78, 0.62, 0.61, 0.54, 0.66, 0.65, 0.63, 0.62, 0.64, 0.7 ,
            0.94, 0.95, 0.97, 0.94, 0.76, 0.44, 0.46, 0.54, 0.65, 0.74, 0.91,
            0.9 , 0.94, 0.88, 0.64, 0.58, 0.52, 0.48, 0.46, 0.49, 0.53, 0.87,
            0.91, 0.88, 0.86, 0.89, 0.82, 0.78, 0.76, 0.56, 0.78, 0.72, 0.7 ,
            0.64, 0.64, 0.46, 0.36, 0.42, 0.48, 0.47, 0.54, 0.56, 0.52, 0.55,
            0.61, 0.57, 0.68, 0.78, 0.94, 0.96, 0.93, 0.84, 0.74, 0.72, 0.74,
            0.64, 0.44, 0.46, 0.5 , 0.96, 0.92, 0.92, 0.94, 0.76, 0.72, 0.66,
            0.64, 0.74, 0.64, 0.38, 0.34, 0.44, 0.36, 0.42, 0.48, 0.86, 0.9 ,
            0.79, 0.71, 0.64, 0.62, 0.57, 0.74, 0.69, 0.87, 0.91, 0.93, 0.68,
            0.61, 0.69, 0.62, 0.72, 0.59, 0.66, 0.56, 0.45, 0.47, 0.71, 0.94,
            0.94, 0.57, 0.61, 0.57, 0.64, 0.85, 0.78, 0.84, 0.92, 0.96, 0.77,
            0.71, 0.79, 0.89, 0.82, 0.76, 0.71, 0.8 , 0.78, 0.84, 0.9 , 0.92,
            0.97, 0.8 , 0.81, 0.75, 0.83, 0.96, 0.79, 0.93, 0.94, 0.86, 0.79,
            0.8 , 0.77, 0.7 , 0.65, 0.61, 0.52, 0.57, 0.53, 0.67, 0.68, 0.81,
            0.78, 0.65, 0.64, 0.64, 0.65, 0.68, 0.89, 0.86, 0.89, 0.87, 0.85,
            0.9 , 0.82, 0.72, 0.73, 0.71, 0.71, 0.68, 0.75, 0.72, 0.89, 0.84,
            0.93, 0.93, 0.88, 0.9 , 0.87, 0.86, 0.94, 0.77, 0.78, 0.73, 0.73,
            0.7 , 0.72, 0.73, 0.72, 0.97, 0.97, 0.69, 0.57, 0.63, 0.66, 0.64,
            0.68, 0.79, 0.82, 0.95, 0.96, 0.94, 0.93, 0.91, 0.85, 0.84, 0.74,
            0.76, 0.75, 0.76, 0.71, 0.67, 0.61, 0.63, 0.64, 0.71, 0.82, 0.73,
            0.74, 0.69, 0.64, 0.91, 0.88, 0.85, 0.86, 0.7 , 0.59, 0.6 , 0.65,
            0.7 , 0.76, 0.63, 0.81, 0.72, 0.71, 0.8 , 0.77, 0.74, 0.7 , 0.71,
            0.93, 0.85, 0.79, 0.76, 0.78, 0.77, 0.9 , 0.87, 0.71, 0.7 , 0.7 ,
            0.75, 0.71, 0.72, 0.73, 0.83, 0.77, 0.72, 0.54, 0.49, 0.52, 0.58,
            0.78, 0.89, 0.7 , 0.66, 0.67, 0.68, 0.8 , 0.81, 0.8 , 0.94, 0.93,
            0.92, 0.89, 0.82, 0.79, 0.58, 0.56, 0.56, 0.64, 0.61, 0.68, 0.76,
            0.86, 0.9 , 0.71, 0.62, 0.66, 0.65, 0.73, 0.62, 0.74, 0.79, 0.8 ,
            0.69, 0.7 , 0.76, 0.84, 0.78, 0.67, 0.66, 0.65, 0.54, 0.58, 0.79,
```



STEP 5: TRAIN AND TEST VARIABLES

```
[16] from sklearn.model_selection import train_test_split
```

```
[17] x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

```
[18] print(x.shape)
     print(x_train.shape)
     print(x_test.shape)

     (400, 6)
     (300, 6)
     (100, 6)
```

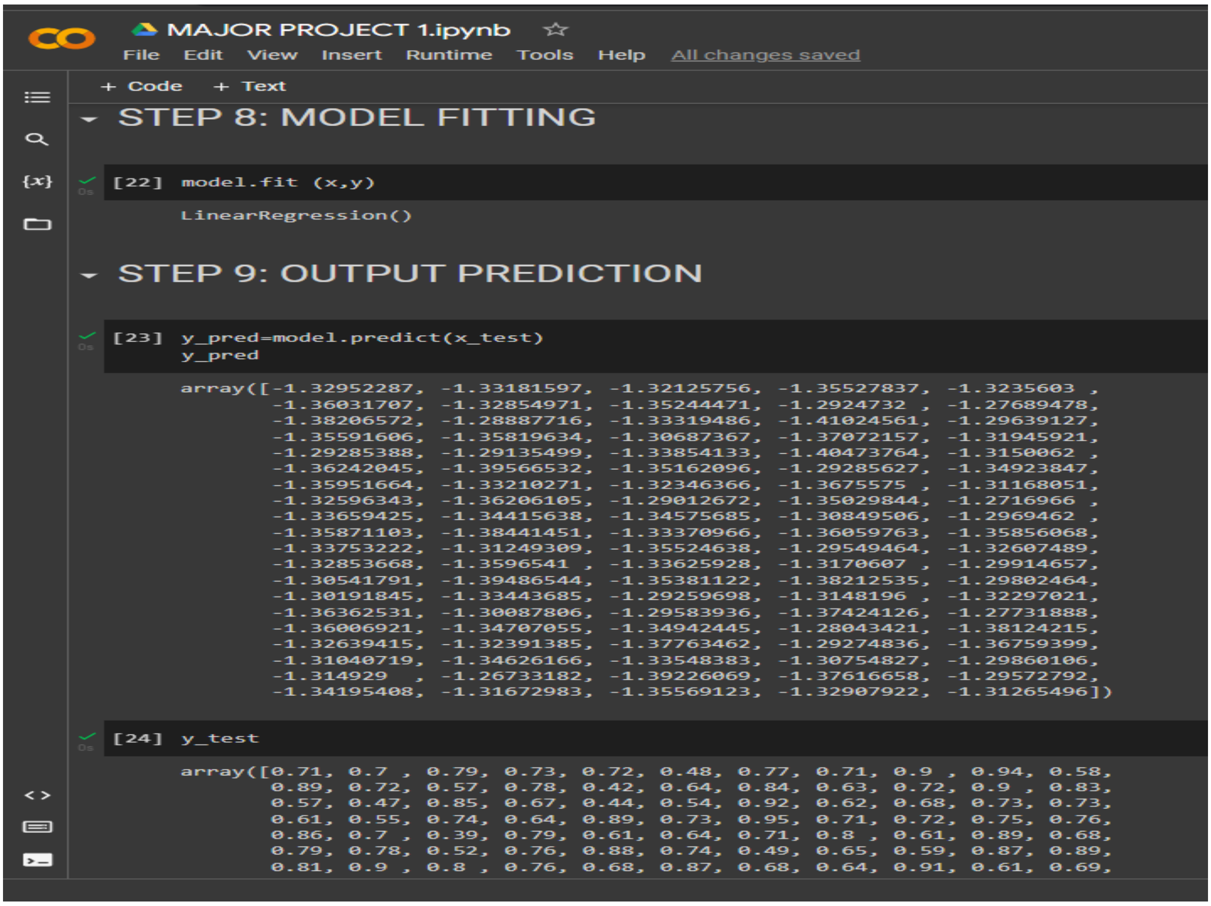
```
[19] print(y.shape)
     print(y_train.shape)
     print(y_test.shape)

     (400,)
     (300,)
     (100,)
```

STEP 6: SCALING OR NORMALISATION

```
[20] from sklearn.preprocessing import MinMaxScaler
     scaler=MinMaxScaler()
     x_train=scaler.fit_transform(x_train)
     x_test=scaler.fit_transform(x_test)
```

STEP 7: RUN A CLASSIFIER/REGRESSOR

```
[21] from sklearn.linear_model import LinearRegression
     model=LinearRegression()
```



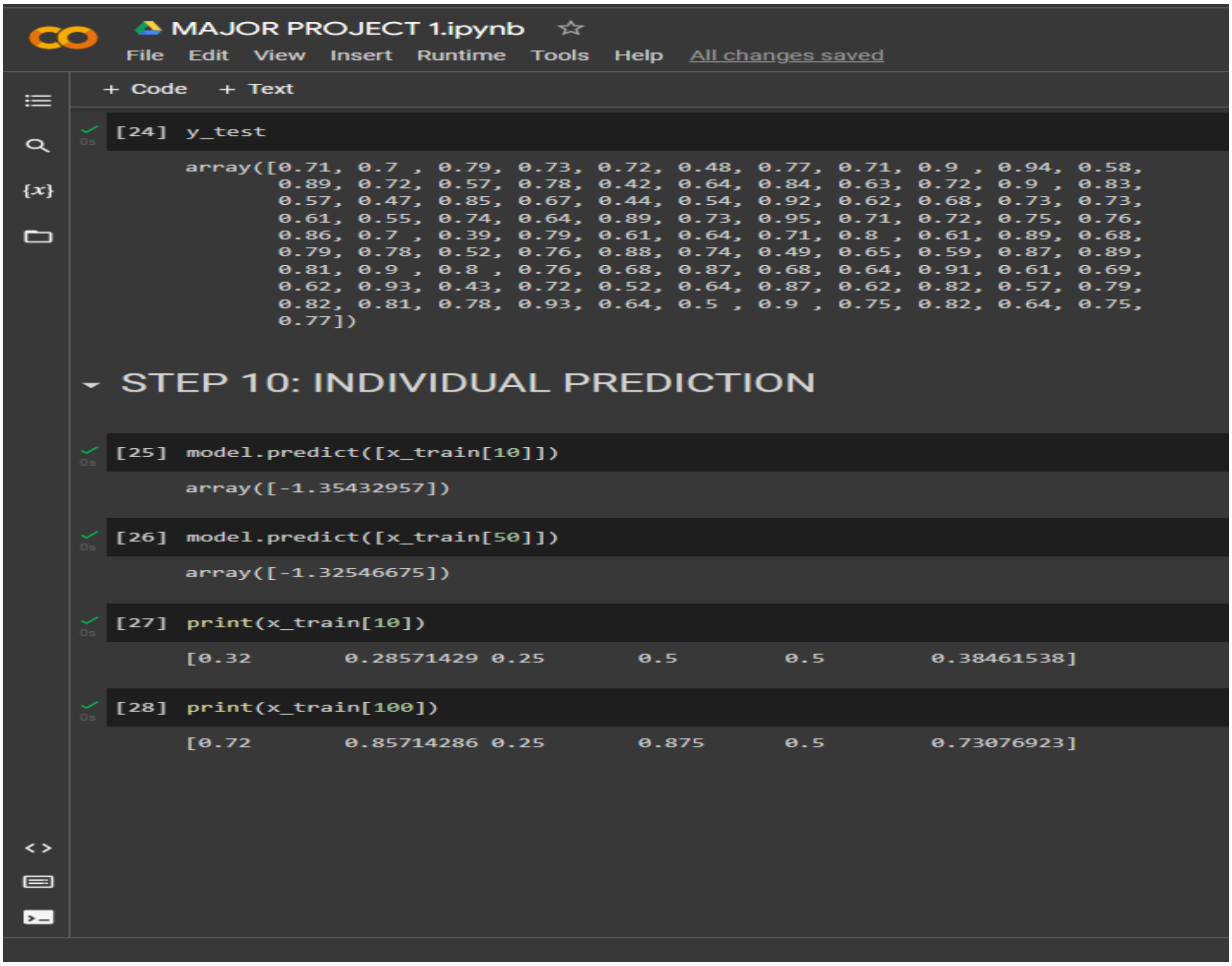
STEP 8: MODEL FITTING

```
[22] model.fit (x,y)

     LinearRegression()
```

STEP 9: OUTPUT PREDICTION

```
[23] y_pred=model.predict(x_test)
     y_pred

     array([-1.32952287, -1.33181597, -1.32125756, -1.35527837, -1.3235603 ,
            -1.36031707, -1.32854971, -1.35244471, -1.2924732 , -1.27689478,
            -1.38206572, -1.28887716, -1.33319486, -1.41024561, -1.29639127,
            -1.35591606, -1.35819634, -1.30687367, -1.37072157, -1.31945921,
            -1.29285388, -1.29135499, -1.33854133, -1.40473764, -1.3150062 ,
            -1.36242045, -1.39566532, -1.35162096, -1.29285627, -1.34923847,
            -1.35951664, -1.33210271, -1.32346366, -1.3675575 , -1.31168051,
            -1.32596343, -1.36206105, -1.29012672, -1.35029844, -1.2716966 ,
            -1.33659425, -1.34415638, -1.34575685, -1.30849506, -1.2969462 ,
            -1.35871103, -1.38441451, -1.33370966, -1.36059763, -1.35856068,
            -1.33753222, -1.31249309, -1.35524638, -1.29549464, -1.32607489,
            -1.32853668, -1.3596541 , -1.33625928, -1.3170607 , -1.29914657,
            -1.30541791, -1.39486544, -1.35381122, -1.38212535, -1.29802464,
            -1.30191845, -1.33443685, -1.29259698, -1.3148196 , -1.32297021,
            -1.36362531, -1.30087806, -1.29583936, -1.37424126, -1.27731888,
            -1.36006921, -1.34707055, -1.34942445, -1.28043421, -1.38124215,
            -1.32639415, -1.32391385, -1.37763462, -1.29274836, -1.36759399,
            -1.31040719, -1.34626166, -1.33548383, -1.30754827, -1.29860106,
            -1.314929  , -1.26733182, -1.39226069, -1.37616658, -1.29572792,
            -1.34195408, -1.31672983, -1.35569123, -1.32907922, -1.31265496])
```

```
[24] y_test

     array([0.71, 0.7 , 0.79, 0.73, 0.72, 0.48, 0.77, 0.71, 0.9 , 0.94, 0.58,
            0.89, 0.72, 0.57, 0.78, 0.42, 0.64, 0.84, 0.63, 0.72, 0.9 , 0.83,
            0.57, 0.47, 0.85, 0.67, 0.44, 0.54, 0.92, 0.62, 0.68, 0.73, 0.73,
            0.61, 0.55, 0.74, 0.64, 0.89, 0.73, 0.95, 0.71, 0.72, 0.75, 0.76,
            0.86, 0.7 , 0.39, 0.79, 0.61, 0.64, 0.71, 0.8 , 0.61, 0.89, 0.68,
            0.79, 0.78, 0.52, 0.76, 0.88, 0.74, 0.49, 0.65, 0.59, 0.87, 0.89,
            0.81, 0.9 , 0.8 , 0.76, 0.68, 0.87, 0.68, 0.64, 0.91, 0.61, 0.69,
```



```
[24] y_test

     array([0.71, 0.7 , 0.79, 0.73, 0.72, 0.48, 0.77, 0.71, 0.9 , 0.94, 0.58,
            0.89, 0.72, 0.57, 0.78, 0.42, 0.64, 0.84, 0.63, 0.72, 0.9 , 0.83,
            0.57, 0.47, 0.85, 0.67, 0.44, 0.54, 0.92, 0.62, 0.68, 0.73, 0.73,
            0.61, 0.55, 0.74, 0.64, 0.89, 0.73, 0.95, 0.71, 0.72, 0.75, 0.76,
            0.86, 0.7 , 0.39, 0.79, 0.61, 0.64, 0.71, 0.8 , 0.61, 0.89, 0.68,
            0.79, 0.78, 0.52, 0.76, 0.88, 0.74, 0.49, 0.65, 0.59, 0.87, 0.89,
            0.81, 0.9 , 0.8 , 0.76, 0.68, 0.87, 0.68, 0.64, 0.91, 0.61, 0.69,
            0.62, 0.93, 0.43, 0.72, 0.52, 0.64, 0.87, 0.62, 0.82, 0.57, 0.79,
            0.82, 0.81, 0.78, 0.93, 0.64, 0.5 , 0.9 , 0.75, 0.82, 0.64, 0.75,
            0.77])
```

STEP 10: INDIVIDUAL PREDICTION

```
[25] model.predict([x_train[10]])

     array([-1.35432957])
```

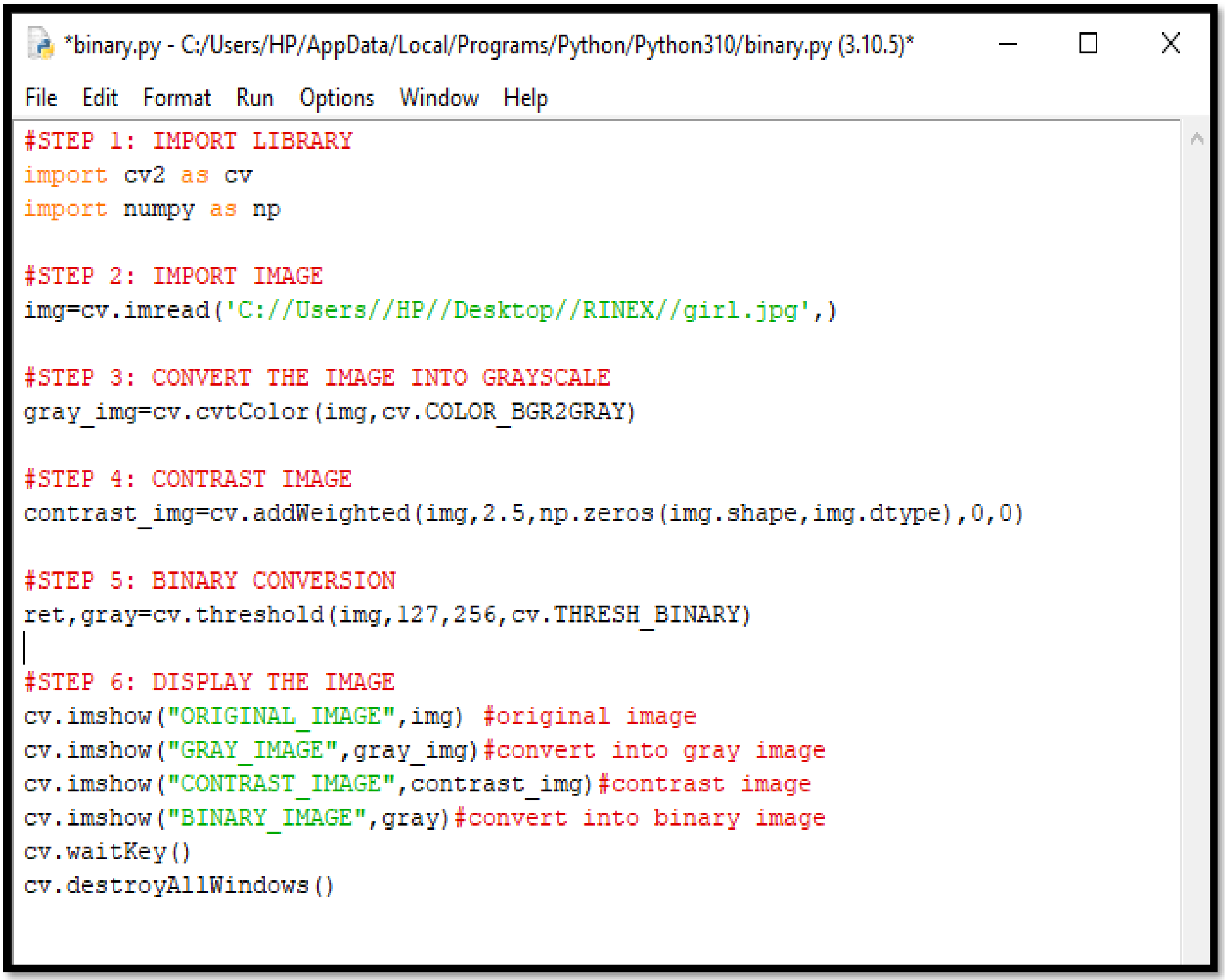
```
[26] model.predict([x_train[50]])

     array([-1.32546675])
```

```
[27] print(x_train[10])

     [0.32       0.28571429 0.25       0.5        0.5        0.38461538]
```

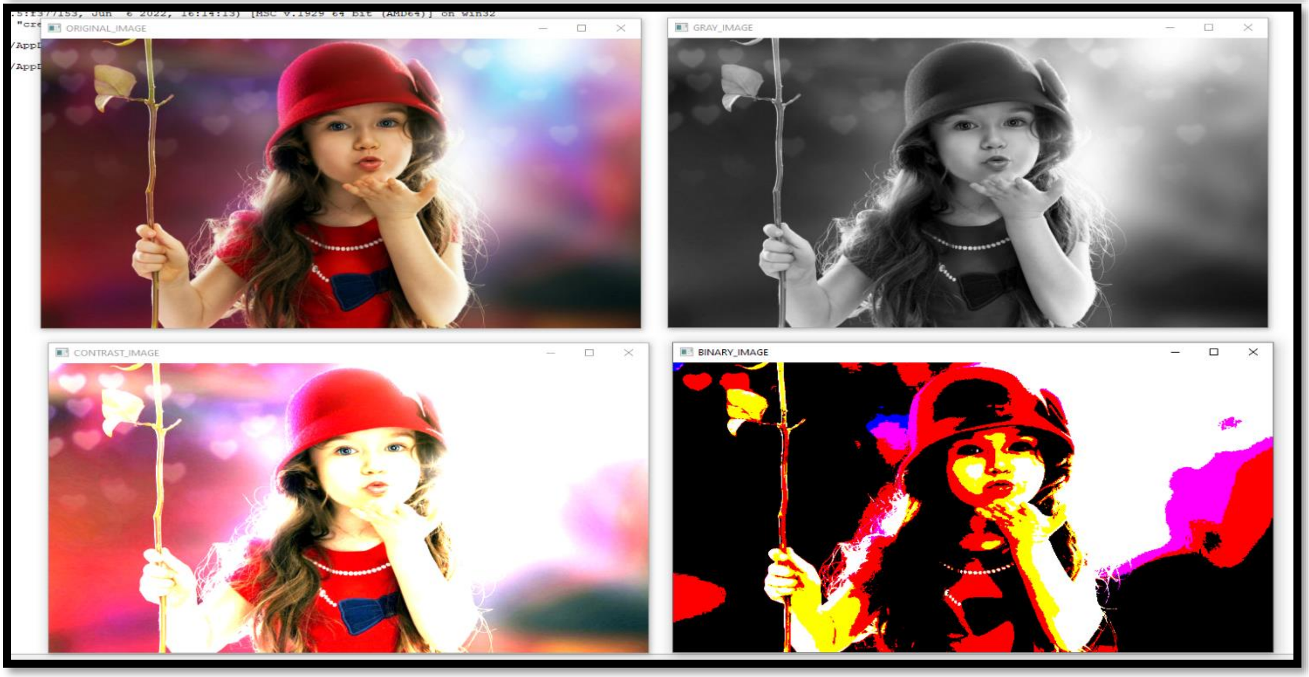
```
[28] print(x_train[100])

     [0.72       0.85714286 0.25       0.875      0.5        0.73076923]
```

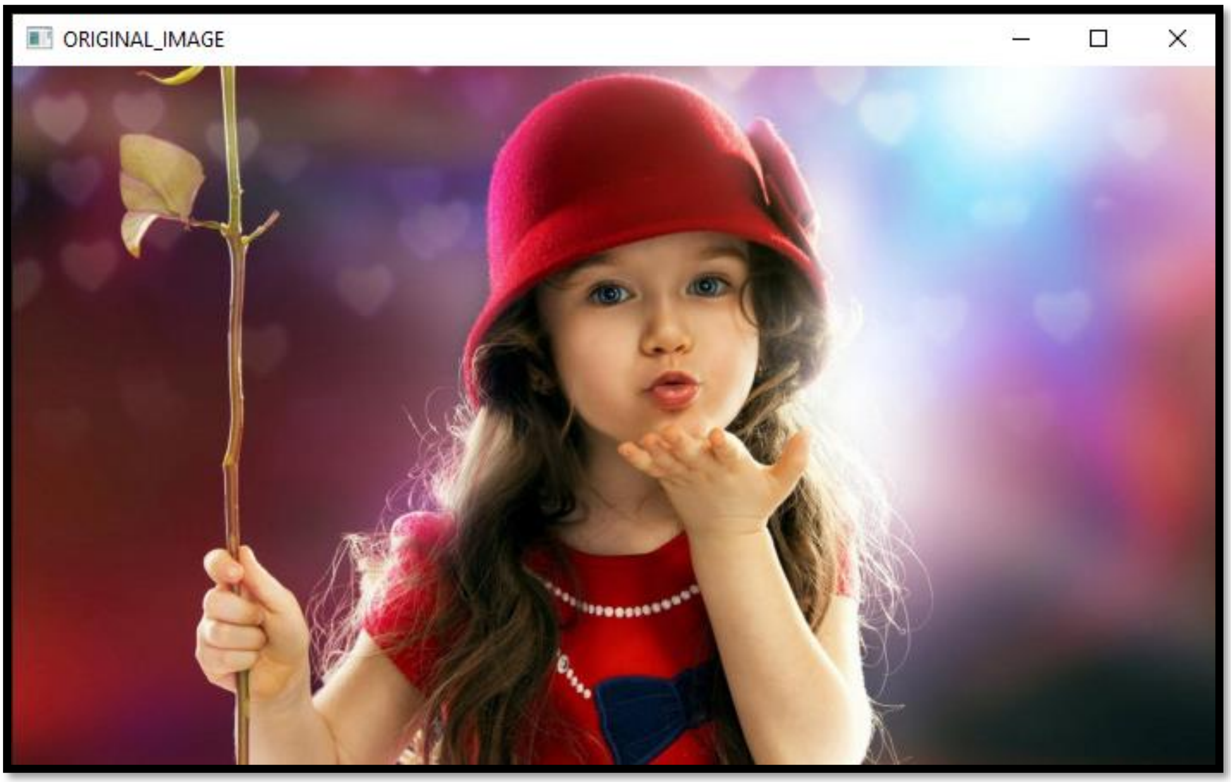
MAJOR PROJECT 2:

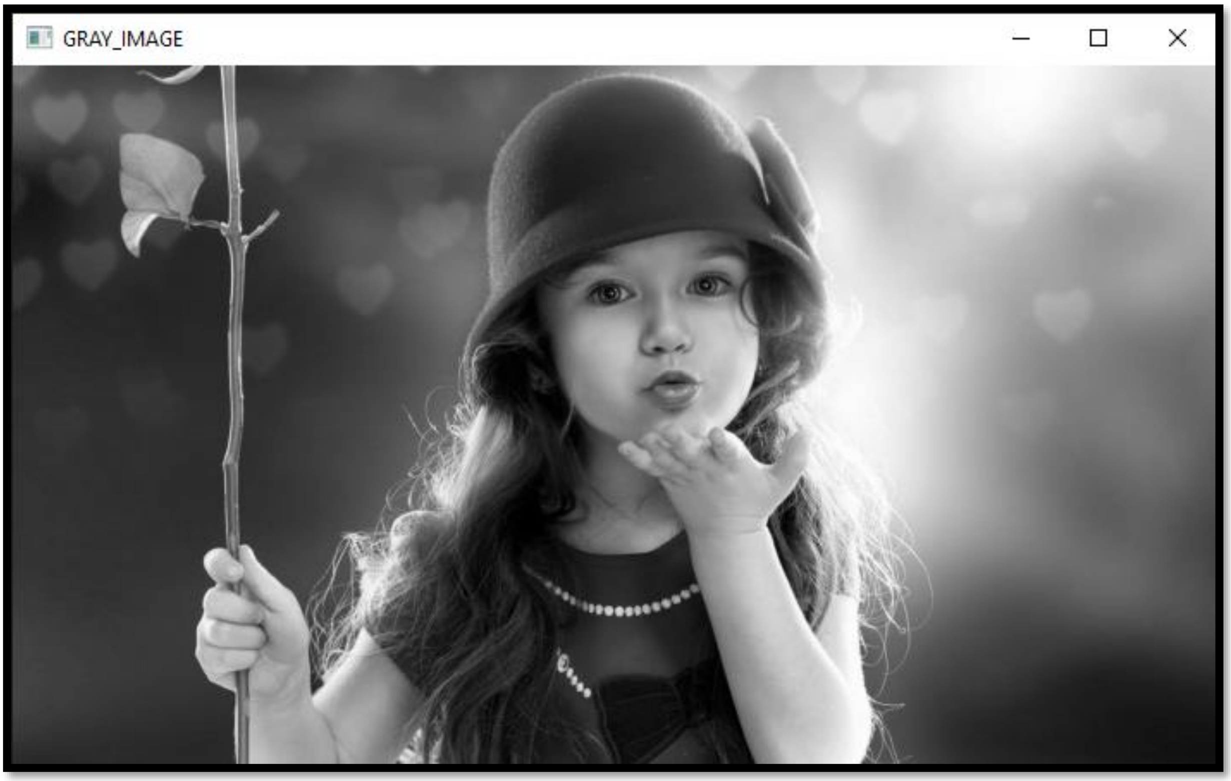
PROJECT ON IMAGE PROCESSING OF IMAGE TO GRAY SCALE , CONTRAST &BINARY CONVERSION

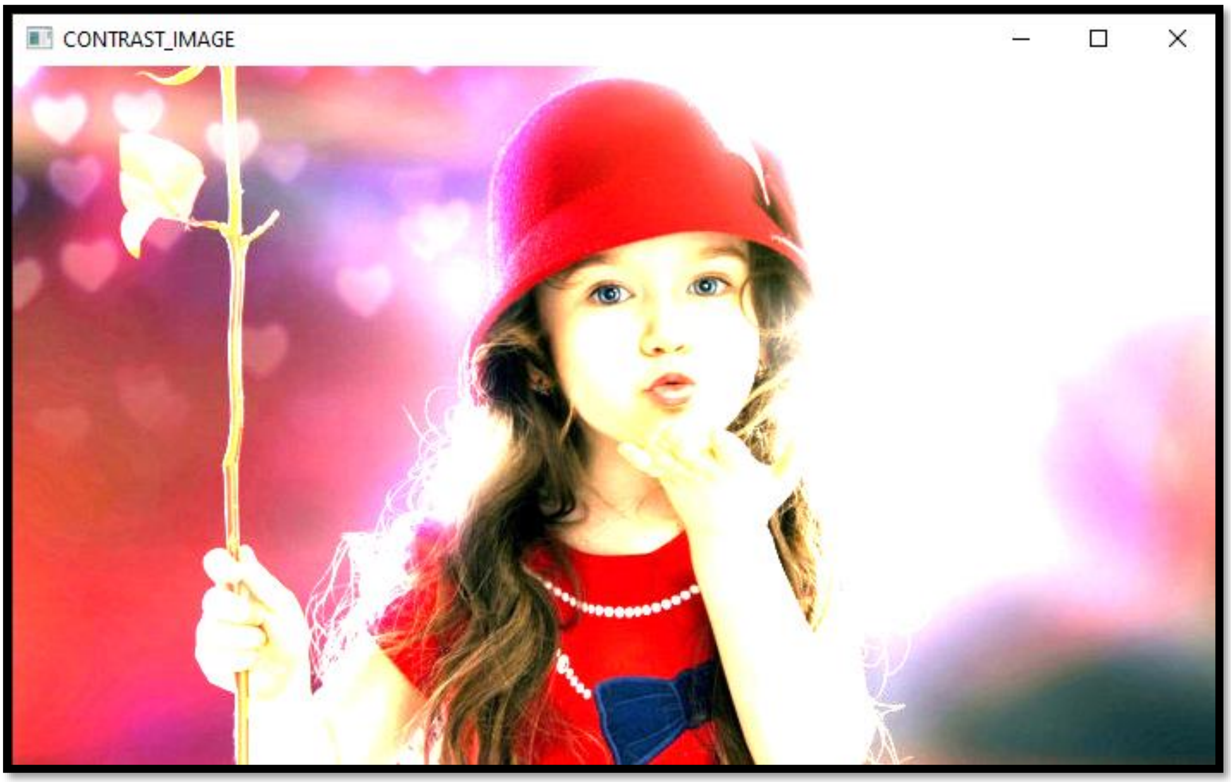

binary.py - C:/Users/HP/AppData/Local/Programs/Python/Python310/binary.py (3.10.5)
File Edit Format Run Options Window Help

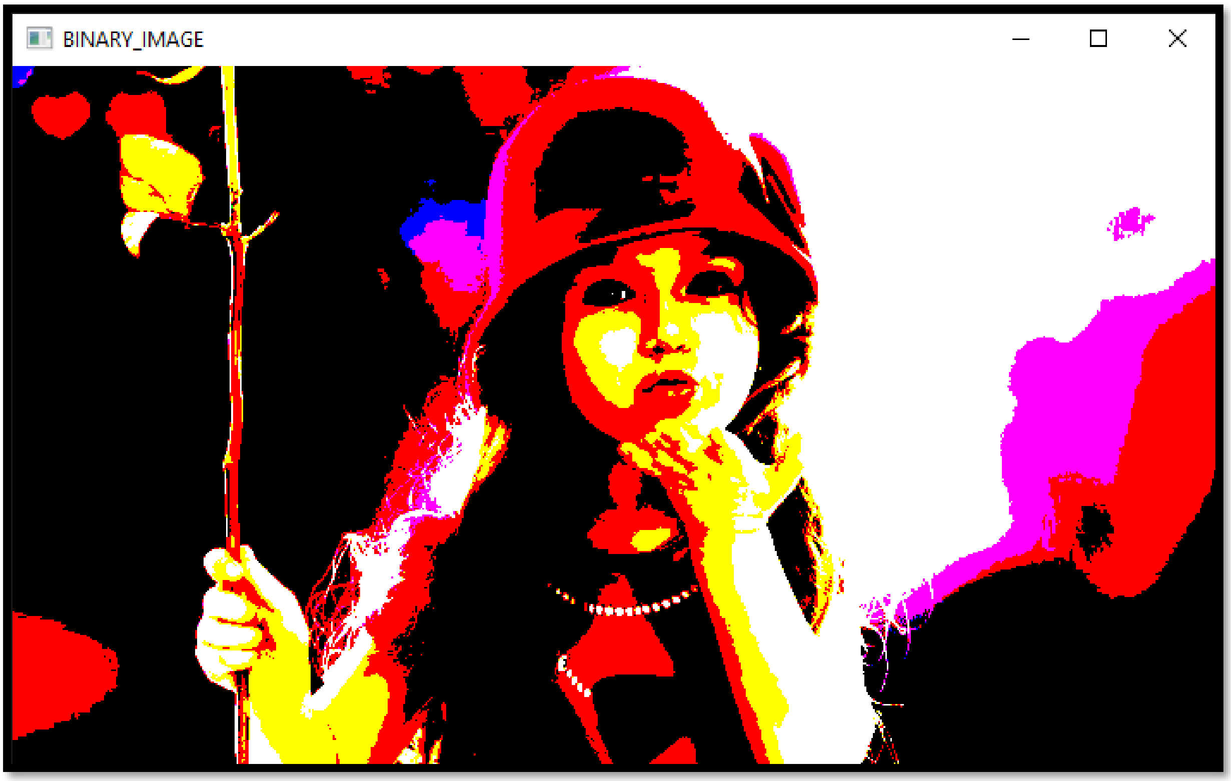
```
#STEP 1: IMPORT LIBRARY
import cv2 as cv
import numpy as np

#STEP 2: IMPORT IMAGE
img=cv.imread('C://Users//HP//Desktop//RINEX//girl.jpg',)

#STEP 3: CONVERT THE IMAGE INTO GRAYSCALE
gray_img=cv.cvtColor(img,cv.COLOR_BGR2GRAY)

#STEP 4: CONTRAST IMAGE
contrast_img=cv.addWeighted(img,2.5,np.zeros(img.shape,img.dtype),0,0)

#STEP 5: BINARY CONVERSION
ret,gray=cv.threshold(img,127,256,cv.THRESH_BINARY)

#STEP 6: DISPLAY THE IMAGE
cv.imshow("ORIGINAL_IMAGE",img) #original image
cv.imshow("GRAY_IMAGE",gray_img)#convert into gray image
cv.imshow("CONTRAST_IMAGE",contrast_img)#contrast image
cv.imshow("BINARY_IMAGE",gray)#convert into binary image
cv.waitKey()
cv.destroyAllWindows()
```




ORIGINAL_IMAGE


GRAY_IMAGE





THANK YOU