REPORT

1.SELF INTRO:

- **Name : Madhuri.O**
- **Email id : madhuriodalla@gmail.com**
- **College : Kasireddy Narayan Reddy College of Engineering and Research**
- **Specification : Computer Science and Engineering**
- **Year : 2023**

2.My AI/ML model - Python code with comments:

1.MAJOR PROJECT 1

- **Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR and if possible deploy it on Heroku.**

DATASET LINK - https://www.kaggle.com/datasets/ammaraahmad/top-10-machine-learning-datasets?select=CO2_emission.csv

```
#1.Take a dataset and create dataframe

import pandas as pd

df = pd.read_csv("/content/CO2_emission.csv")

df
```

	Model_Year	Make	Model	Vehicle_Class	Engine_Size	Cylinders	Transmission	Fuel_Consumption_in_City(L/100 km)	Fuel_Consumption_in_City_Hwy(L/100 km)	Fuel_Consumption_comb(L/100km)	CO2_Emissions	Smog_Level
0	2021	Acura	ILX	Compact	2.4	4	AM8	9.9	7.0	8.6	199	3
1	2021	Acura	NSX	Two-seater	3.5	6	AM9	11.1	10.8	11.0	256	3
2	2021	Acura	RDX SH-AWD	SUV: Small	2.0	4	AS10	11.0	8.6	9.9	232	6
3	2021	Acura	RDX SH-AWD A-SPEC	SUV: Small	2.0	4	AS10	11.3	9.1	10.3	242	6
4	2021	Acura	TLX SH-AWD	Compact	2.0	4	AS10	11.2	8.0	9.8	230	7
...	...	...	...	...	...	...	...	...	...	...	...	...
930	2021	Volvo	XC40 T5 AWD	SUV: Small	2.0	4	AS8	10.7	7.7	9.4	219	5
931	2021	Volvo	XC60 T5 AWD	SUV: Small	2.0	4	AS8	11.1	8.3	9.9	230	5
932	2021	Volvo	XC60 T6 AWD	SUV: Small	2.0	4	AS8	11.7	8.6	10.3	240	7
933	2021	Volvo	XC90 T5 AWD	SUV: Standard	2.0	4	AS8	11.5	8.4	10.1	236	5
934	2021	Volvo	XC90 T6 AWD	SUV: Standard	2.0	4	AS8	12.1	8.5	10.5	245	7

935 rows × 12 columns

```
#to display the information present in the table

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 935 entries, 0 to 934
Data columns (total 12 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Model_Year                              935 non-null    int64
 1   Make                                    935 non-null    object
 2   Model                                   935 non-null    object
 3   Vehicle_Class                           935 non-null    object
 4   Engine_Size                             935 non-null    float64
 5   Cylinders                               935 non-null    int64
 6   Transmission                            935 non-null    object
 7   Fuel_Consumption_in_City(L/100 km)      935 non-null    float64
 8   Fuel_Consumption_in_City_Hwy(L/100 km)  935 non-null    float64
 9   Fuel_Consumption_comb(L/100km)          935 non-null    float64
 10  CO2_Emissions                           935 non-null    int64
 11  Smog_Level                              935 non-null    int64
dtypes: float64(4), int64(4), object(4)
memory usage: 87.8+ KB
```

df.shape # 935 rows and 13 columns

```
(935, 12)
```

df.size # total number of elements

```
11220
```

#to check the number to null values present

df.isnull()

	Model_Year	Make	Model	Vehicle_Class	Engine_Size	Cylinders	Transmission	Fuel_Consumption_in_City(L/100 km)	Fuel_Consumption_in_City_Hwy(L/100 km)	Fuel_Consumption_comb(L/100km)	CO2_Emissions	Smog_Level
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
930	False	False	False	False	False	False	False	False	False	False	False	False
931	False	False	False	False	False	False	False	False	False	False	False	False
932	False	False	False	False	False	False	False	False	False	False	False	False
933	False	False	False	False	False	False	False	False	False	False	False	False
934	False	False	False	False	False	False	False	False	False	False	False	False

935 rows × 12 columns

To display 1st 5 row indexes

df.head()

	Model_Year	Make	Model	Vehicle_Class	Engine_Size	Cylinders	Transmission	Fuel_Consumption_in_City(L/100 km)	Fuel_Consumption_in_City_Hwy(L/100 km)	Fuel_Consumption_comb(L/100km)	CO2_Emissions	Smog_Level
0	2021	Acura	ILX	Compact	2.4	4	AM8	9.9	7.0	8.6	199	3
1	2021	Acura	NSX	Two-seater	3.5	6	AM9	11.1	10.8	11.0	256	3
2	2021	Acura	RDX SH-AWD	SUV: Small	2.0	4	AS10	11.0	8.6	9.9	232	6
3	2021	Acura	RDX SH-AWD A-SPEC	SUV: Small	2.0	4	AS10	11.3	9.1	10.3	242	6
4	2021	Acura	TLX SH-AWD	Compact	2.0	4	AS10	11.2	8.0	9.8	230	7

#To display last 5 row indexes

df.tail()

	Model_Year	Make	Model	Vehicle_Class	Engine_Size	Cylinders	Transmission	Fuel_Consumption_in_City(L/100 km)	Fuel_Consumption_in_City_Hwy(L/100 km)	Fuel_Consumption_comb(L/100km)	CO2_Emissions	Smog_Level
930	2021	Volvo	XC40 T5 AWD	SUV: Small	2.0	4	AS8	10.7	7.7	9.4	219	5
931	2021	Volvo	XC60 T5 AWD	SUV: Small	2.0	4	AS8	11.1	8.3	9.9	230	5
932	2021	Volvo	XC60 T6 AWD	SUV: Small	2.0	4	AS8	11.7	8.6	10.3	240	7
933	2021	Volvo	XC90 T5 AWD	SUV: Standard	2.0	4	AS8	11.5	8.4	10.1	236	5
934	2021	Volvo	XC90 T6 AWD	SUV: Standard	2.0	4	AS8	12.1	8.5	10.5	245	7

#2.Preprocessing — Filtering of Data

#We want to consider only the numeric data

#So we will create a new dataframe with only numeric data

df_numeric = df.select_dtypes(include = ['float64','int64'])

df_numeric

	Model_Year	Engine_Size	Cylinders	Fuel_Consumption_in_City(L/100 km)	Fuel_Consumption_in_City_Hwy(L/100 km)	Fuel_Consumption_comb(L/100km)	CO2_Emissions	Smog_Level
0	2021	2.4	4	9.9	7.0	8.6	199	3
1	2021	3.5	6	11.1	10.8	11.0	256	3
2	2021	2.0	4	11.0	8.6	9.9	232	6
3	2021	2.0	4	11.3	9.1	10.3	242	6
4	2021	2.0	4	11.2	8.0	9.8	230	7
...	...	...	...	...	...	...	...	...
930	2021	2.0	4	10.7	7.7	9.4	219	5
931	2021	2.0	4	11.1	8.3	9.9	230	5
932	2021	2.0	4	11.7	8.6	10.3	240	7
933	2021	2.0	4	11.5	8.4	10.1	236	5
934	2021	2.0	4	12.1	8.5	10.5	245	7

935 rows × 8 columns

#to display the table information which contains only numeric data

df_numeric.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 935 entries, 0 to 934
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Model_Year                              935 non-null    int64
 1   Engine_Size                             935 non-null    float64
 2   Cylinders                               935 non-null    int64
 3   Fuel_Consumption_in_City(L/100 km)      935 non-null    float64
 4   Fuel_Consumption_in_City_Hwy(L/100 km)  935 non-null    float64
 5   Fuel_Consumption_comb(L/100km)          935 non-null    float64
 6   CO2_Emissions                           935 non-null    int64
 7   Smog_Level                              935 non-null    int64
dtypes: float64(4), int64(4)
memory usage: 58.6 KB
```

#2.To remove model_year column

df_numeric = df_numeric.drop(['Model_Year'],axis = 1)#axis = 1 -column,axis = 0 - row

df_numeric

	Engine_Size	Cylinders	Fuel_Consumption_in_City(L/100 km)	Fuel_Consumption_in_City_Hwy(L/100 km)	Fuel_Consumption_comb(L/100km)	CO2_Emissions	Smog_Level
0	2.4	4	9.9	7.0	8.6	199	3
1	3.5	6	11.1	10.8	11.0	256	3
2	2.0	4	11.0	8.6	9.9	232	6
3	2.0	4	11.3	9.1	10.3	242	6
4	2.0	4	11.2	8.0	9.8	230	7
...	...	...	...	...	...	...	...
930	2.0	4	10.7	7.7	9.4	219	5
931	2.0	4	11.1	8.3	9.9	230	5
932	2.0	4	11.7	8.6	10.3	240	7
933	2.0	4	11.5	8.4	10.1	236	5
934	2.0	4	12.1	8.5	10.5	245	7

935 rows × 7 columns

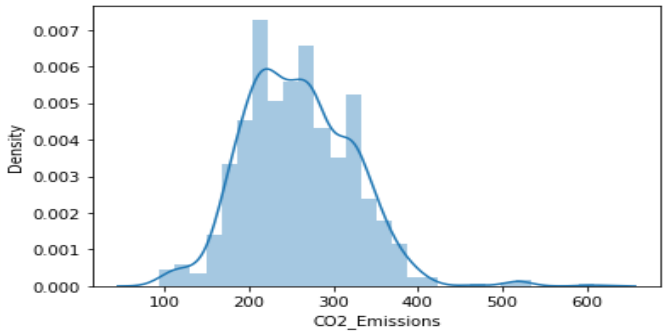
df_numeric.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 935 entries, 0 to 934
Data columns (total 7 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Engine_Size                             935 non-null    float64
 1   Cylinders                               935 non-null    int64
 2   Fuel_Consumption_in_City(L/100 km)      935 non-null    float64
 3   Fuel_Consumption_in_City_Hwy(L/100 km)  935 non-null    float64
 4   Fuel_Consumption_comb(L/100km)          935 non-null    float64
 5   CO2_Emissions                           935 non-null    int64
 6   Smog_Level                              935 non-null    int64
dtypes: float64(4), int64(3)
memory usage: 51.3 KB
```

#3.VISUALIZATION

import seaborn as sns

sns.distplot(df['CO2_Emissions']) # distribution plot



#4.divide the data into i/p and o/p

#output - Smog_Level

#input - All the columns except the Smog_Level column

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

```
array([[2.021e+03, 2.400e+00, 4.000e+00, 9.900e+00, 7.000e+00, 8.600e+00],
       [2.021e+03, 3.500e+00, 6.000e+00, 1.110e+01, 1.080e+01, 1.100e+01],
       [2.021e+03, 2.000e+00, 4.000e+00, 1.100e+01, 8.600e+00, 9.900e+00],
       ...,
       [2.021e+03, 2.000e+00, 4.000e+00, 1.170e+01, 8.600e+00, 1.030e+01],
       [2.021e+03, 2.000e+00, 4.000e+00, 1.150e+01, 8.400e+00, 1.010e+01],
       [2.021e+03, 2.000e+00, 4.000e+00, 1.210e+01, 8.500e+00, 1.050e+01]])
```

```
y = df_numeric.iloc[:,6]
y
```

```
0       199
1       256
2       232
3       242
4       230
      ...
930     219
931     230
932     240
933     236
934     245
Name: CO2_Emissions, Length: 935, dtype: int64
```

#5.TRAIN and TEST VARIABLES

#sklearn.model_selection - package , train_test_split - library

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state = 0)
```

#Whatever data splitting /data allocation happens to the xtrain,x_test,ytrain,ytest variables , we want those

#allocated values to remain constant.By default the training variables get 75 % and testing variables get 25%

```
print(x.shape) # 935 rows and 16 cols
print(x_train.shape) # 935 rows and 16 cols (75%)
print(x_test.shape) # 935 rows and 6 cols (25%)
```

```
(935, 6)
(701, 6)
(234, 6)
```

```
print(y.shape) # 935 rows and 6 col s
print(y_train.shape) # 935 rows and 6 cols (75%)
print(y_test.shape) # 935 rows and 6 cols (25%)
```

```
(935,)
(701,)
(234,)
```

#6.SCALING or NORMALISATION -DONE ONLY FOR INPUTS

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

```
#7.RUN a CLASSIFIER/REGRESSOR/CLUSTERER
from sklearn.linear_model import LinearRegression
model = LinearRegression()
#8.MODEL FITTING
model.fit(x_train,y_train)
```

```
LinearRegression()
```

```
#9.PREDICT THE OUTPUT
y_pred = model.predict(x_test)#By taking the input testing data , we predict the output
y_pred #PREDICTED VALUES
```

```
array([244.24954433, 213.00517032, 236.7668758 , 279.63865564,
       419.42898381, 387.2095768 , 292.1292442 , 206.34087855,
       368.35154999, 315.67397036, 455.55907417, 173.35669567,
       450.75719255, 260.90011622, 226.73431455, 249.96385422,
       318.52378947, 245.39802308, 503.74864613, 326.69302564,
       387.83818422, 233.66797356, 279.61159514, 279.78211659,
       248.95573539, 422.21074841, 289.40734077, 280.37758396,
       297.01203625, 189.09817326, 378.54844471, 232.79872539,
       248.7791344 , 120.94132618, 362.53269547, 298.36240208,
       323.5624867 , 255.59164005, 188.27795178, 127.40522014,
       338.24946676, 244.88346614, 369.71672289, 337.02465378,
       408.12770536, 293.61996994, 212.39284515, 407.14058157,
       228.05629326, 360.8149231 , 259.23587643, 211.40027158,
       318.52378947, 219.27783294, 261.61112453, 459.94830432,
       298.14734665, 407.14058157, 286.91308648, 254.14024232,
       219.45443393, 317.77111846, 399.5854309 , 238.03471942,
       237.22419662, 233.83213031, 220.14249083, 121.04544503,
       254.49493355, 291.22525829, 206.83444044, 239.68371318,
       301.19301463, 187.85685795, 240.03691516, 186.82345303,
       323.5624867 , 282.11332508, 295.4717657 , 282.38199307,
       255.75448427, 268.01433675, 270.12065133, 605.8595324 ,
       305.356099  , 288.7858632 , 304.18778394, 302.57962153,
       295.00984051, 182.29386295, 396.40530581, 226.1990618 ,
       202.03415651, 249.62589827, 212.76394115, 281.16563654,
       334.06848838, 211.40572137, 354.12425359, 263.63118613,
       221.37595219, 209.12280579, 258.88267446, 196.0872799 ,
       240.00067409, 295.76711588, 383.69671058, 306.22620667,
       315.84449181, 365.75167351, 139.50240512, 240.31763499,
       302.38926377, 205.05053654, 157.69869735, 387.14335093,
       270.26709079, 308.06098198, 309.18846568, 244.91970721,
       271.59676081, 241.36199489, 257.78137768, 187.00005402,
       209.65260876, 245.90947899, 351.11721677, 303.72501333,
       223.84366409, 343.91746826, 440.38148346, 256.21639538,
       235.43484294, 393.30272821, 281.72707621, 318.52378947,
       114.40954037, 379.58952687, 463.23137634, 380.79602006,
       230.92314688, 249.69838041, 240.24975721, 298.36240208,
       235.04539989, 487.23665974, 362.53269547, 267.00621791,
       368.91626741, 412.15130338, 239.25802906, 454.71690058,
       284.48830765, 269.28913349, 331.71309067, 412.5237119 ,
       246.5324599 , 393.30712769, 454.71690058, 228.76269721,
       278.92531265, 232.83496646, 410.93653046, 228.76269721,
       260.51067317, 382.79036184, 384.66222364, 477.68641622,
       287.45928013, 202.09200831, 277.48916095, 360.76015825,
       201.97085494, 305.41172334, 265.94426323, 272.08115622,
       448.19515483, 366.54843954, 284.48830765, 357.44297826,
       381.48081328, 263.63118613, 226.26693958, 353.54730999,
       254.53117462, 251.27489203, 351.24233066, 311.59489746,
       220.14249083, 219.96588984, 245.57462406, 360.76015825,
       245.18518102, 366.13978995, 268.01433675, 384.81412696,
       249.62589827, 286.13730141, 220.27042059, 283.51907788,
       293.26676797, 427.18734816, 369.04505667, 123.39166218,
       295.00984051, 189.09817326, 209.09574529, 233.32852836,
       372.21097718, 252.29825689, 222.24605986, 279.78211659,
       251.84816026, 282.88250808, 214.78448394, 456.67121749,
       337.96505748, 270.87854238, 336.94169262, 364.39584382,
       223.73954524, 362.53269547, 375.27145224, 319.09768746,
       199.00499086, 307.93305222, 322.08786648, 300.71779979,
       405.54836204, 273.93838768, 286.91308648, 282.62189563,
       265.37276942, 307.68855937])
```

```
y_test #ACTUAL VALUES
```

```
689    226
236    196
738    223
766    255
266    363
      ...
736    240
172    268
492    249
735    233
363    279
Name: CO2_Emissions, Length: 234, dtype: int64
```

```
print(x_train[10]) #these are scaled/normalised values
```

```
[0.         0.18571429 0.07692308 0.36121673 0.57647059 0.43438914]
```

```
#INDIVIDUAL PREDICTION
```

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

```
array([300.14122644])
```

MAJOR PROJECT 2

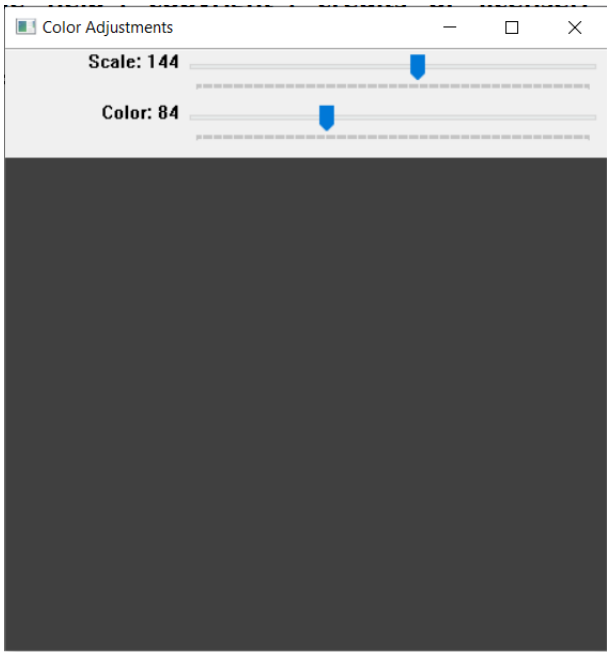
- **Create any of the Image Processing Projects using Numpy and OpenCV.**

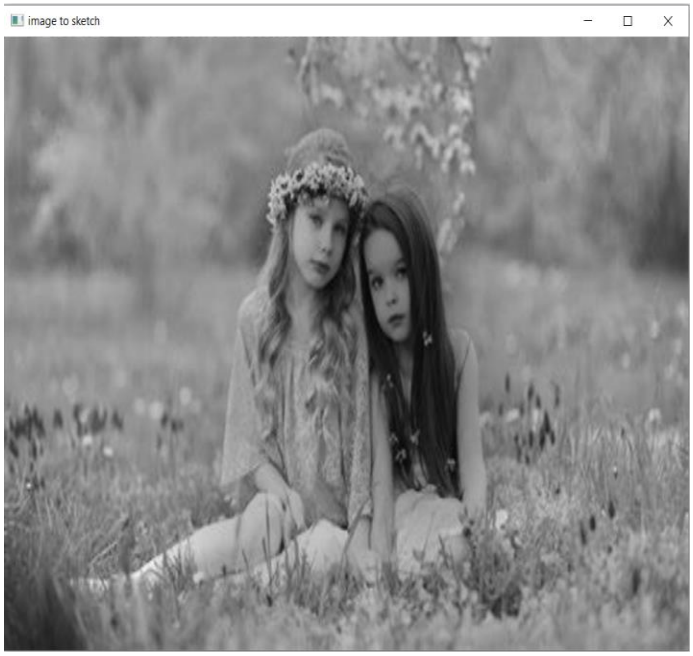
NAME OF THE PROJECT: Image to sketch

Code:

```
#Step - 1 - Load Libraries and Image
#Step - 2 - Converte Image into Gray Scale
#Step - 3 - Inveted Gary Scale Image [For Shifting toward selected channel]
#Step - 4 - Apply Image Smooting For Shading effect
#Step - 5 - Invert Blur Image and Apply division between gray and invert_blur.
#-----------------------------------------------------------------------------------------------------------------------
#Step-1-Importing numpy and cv2 packages
import numpy as np
import cv2

#Read Image----------
#imread( ) is used to read the image for the given  directory
img = cv2.imread('image to sketch.jpg')
#resize( ) is used to change the image size
img = cv2.resize(img,(450,450))

#Create Trackbar----

def nothing(x):#Define a function which can be used as call back function for the trackbar
   pass

#namedWindow( ) takes two arguments-1.window_name:Used to name window that
displayed,2.flag:Represents if window size is automatically set or adjustable
cv2.namedWindow("Color Adjustments",cv2.WINDOW_NORMAL)

#It takes 3 arguments-1.window_name, 2.width, 3.height
cv2.resizeWindow("Color Adjustments", (450, 450))

#createTrackbar()-Used to read the current poisition of the trackbar slider
cv2.createTrackbar("Scale", "Color Adjustments", 0, 255, nothing)
cv2.createTrackbar("Color", "Color Adjustments", 0, 255, nothing)

#Step -2
#Convert into gray--
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
while True:
   scale =  cv2.getTrackbarPos("Scale", "Color Adjustments")
   clr = cv2.getTrackbarPos("Color", "Color Adjustments") #getting track bar value
```

```
#Extracting Color Code --
#Step - 3
inverted_gray =  clr - gray  #inverted color image
#Step -4
blur_img = cv2.GaussianBlur(inverted_gray,(255,255),0) #Used to smoothing the input image
#Step -5
inverted_blur = clr - blur_img  #inverted blured image
fltr = cv2.divide(gray,inverted_blur,scale = scale)



#Output----------------------
cv2.imshow("image to sketch",fltr) #show the image to sketch image
k = cv2.waitKey(1)
#use waitkey to add delay and stop the function when the user presses esc key
if k == ord("q"):
    break
if k == ord("s"):
   cv2.imwrite("image to sketch.jpg",fltr) #Used to save an image to any storage device

cv2.destroyAllWindows() #destroy all widows after exiting the while loop
```





Github Account Link - https://github.com/Odalla-Madhuri/mlrinex