

REPORT

1.SELF INTRO:

- ▶ Name : Susmitha Bathini
- ▶ Email id : susmitha1822@gmail.com
- ▶ College : Brilliant Institute Of Engineering & Technology
- ▶ Specification : Computer Science and Engineering
- ▶ Year : 2023

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

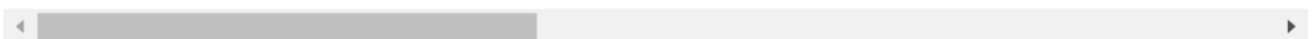
[MAJOR PROJECT -1](#)

```
# MAJOR PROJECT 1 - Choose any dataset of your choice and apply suitable REGRESSOR/CLASSIFIER  
#Dataset - '/content/TikTok_songs_2020.csv'
```

```
#1.Take a dataset and create dataframe  
import pandas as pd  
df = pd.read_csv("/content/TikTok_songs_2020.csv")  
df
```

	track_name	artist_name	artist_pop	album	track_pop	danceability	energy
0	Say So	Doja Cat	88	Hot Pink	80	0.787	0.673
1	Blinding Lights	The Weeknd	93	After Hours	90	0.514	0.730
2	Supalonely (feat. Gus Dapperton)	BENEE	67	Hey u x	63	0.862	0.631
3	Savage	Megan Thee Stallion	82	Suga	70	0.843	0.741
4	Moral of the Story	Ashe	68	Moral of the Story	76	0.572	0.406
...
287	Buttons	The Pussycat Dolls	68	PCD	65	0.570	0.821
288	Get Busy	Sean Paul	79	Dutty Rock	74	0.735	0.824
289	ROCKSTAR (feat. Roddy Ricch)	DaBaby	82	BLAME IT ON BABY	80	0.746	0.690
290	Who Says	Selena Gomez & The Scene	67	When The Sun Goes Down	76	0.682	0.927
291	Crystal Dolphin	Engelwood	50	Crust FM	60	0.558	0.776

292 rows × 18 columns



```
#to display the information present in the table
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292 entries, 0 to 291
Data columns (total 18 columns):
```

#	Column	Non-Null Count	Dtype
0	track_name	292 non-null	object
1	artist_name	292 non-null	object
2	artist_pop	292 non-null	int64
3	album	292 non-null	object
4	track_pop	292 non-null	int64
5	danceability	292 non-null	float64
6	energy	292 non-null	float64
7	loudness	292 non-null	float64
8	mode	292 non-null	int64
9	key	292 non-null	int64
10	speechiness	292 non-null	float64
11	acousticness	292 non-null	float64
12	instrumentalness	292 non-null	float64
13	liveness	292 non-null	float64
14	valence	292 non-null	float64
15	tempo	292 non-null	float64
16	time_signature	292 non-null	int64
17	duration_ms	292 non-null	int64

dtypes: float64(9), int64(6), object(3)

memory usage: 41.2+ KB

df.shape

(292, 18)

df.size

5256

```
track_name artist_name artist_pop album track_pop danceability energy
```

```
#To display 1st 5 rows indexes
df.head()
```

	track_name	artist_name	artist_pop	album	track_pop	danceability	energy
0	Say So	Doja Cat	88	Hot Pink	80	0.787	0.673
1	Blinding Lights	The Weeknd	93	After Hours	90	0.514	0.730
2	Supalonely (feat. Gus Dapperton)	BENEE	67	Hey u x	63	0.862	0.631
3	Savage	Megan Thee Stallion	82	Suga	70	0.843	0.741
4	Moral of the Story	Ashe	68	Moral of the Story	76	0.572	0.406

```
#To display last 5 row indexes
df.tail
```

<bound method NDFrame.tail of				track_name			
artist_name artist_pop \							
0		Say So	Doja Cat	88			
1		Blinding Lights	The Weeknd	93			
2	Supalonely (feat. Gus Dapperton)	BENEE	67				
3	Savage	Megan Thee Stallion	82				
4	Moral of the Story	Ashe	68				
..				
287	Buttons	The Pussycat Dolls	68				
288	Get Busy	Sean Paul	79				
289	ROCKSTAR (feat. Roddy Ricch)	DaBaby	82				
290	Who Says	Selena Gomez & The Scene	67				
291	Crystal Dolphin	Engelwood	50				
				album	track_pop	danceability	energy
0		Hot Pink	80	0.787	0.673	-4.583	0
1		After Hours	90	0.514	0.730	-5.934	1
2		Hey u x	63	0.862	0.631	-4.746	1
3		Suga	70	0.843	0.741	-5.609	1
4		Moral of the Story	76	0.572	0.406	-8.624	1
..	
287		PCD	65	0.570	0.821	-4.380	1
288		Dutty Rock	74	0.735	0.824	-4.143	0
289		BLAME IT ON BABY	80	0.746	0.690	-7.956	1
290		When The Sun Goes Down	76	0.682	0.927	-2.915	1
291		Crust FM	60	0.558	0.776	-6.868	1
				key	speechiness	acousticness	instrumentalness
0	11	0.1590	0.26400	0.000003	0.0904	0.779	

```
#To check the number to null values present
df.isnull()
```

1	1	0.0598	0.00146	0.000095	0.0897	0.334
2	7	0.0515	0.29100	0.000209	0.1230	0.841
3	11	0.3340	0.02520	0.000000	0.0960	0.680
4	10	0.0427	0.58700	0.000004	0.1020	0.265
...
287	2	0.2670	0.17800	0.000000	0.2890	0.408
288	10	0.0360	0.61500	0.000000	0.1580	0.726
289	11	0.1640	0.24700	0.000000	0.1010	0.497
290	4	0.0479	0.08430	0.000000	0.1490	0.744
291	9	0.1790	0.33000	0.000445	0.4100	0.247

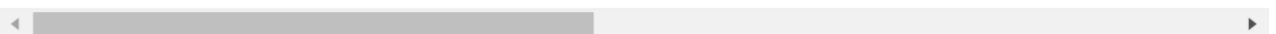
	tempo	time_signature	duration_ms
0	110.962	4	237893
1	171.005	4	200040
2	128.978	4	223488
3	168.983	4	155497
4	119.812	4	201084
...
287	210.857	4	225560
288	100.202	4	211666
289	89.977	4	181733
290	101.019	4	195613
291	128.064	4	114660

```
[292 rows x 18 columns]>
```

```
#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
```

	artist_pop	track_pop	danceability	energy	loudness	mode	key	speechiness
0	88	80	0.787	0.673	-4.583	0	11	0.15
1	93	90	0.514	0.730	-5.934	1	1	0.05
2	67	63	0.862	0.631	-4.746	1	7	0.05
3	82	70	0.843	0.741	-5.609	1	11	0.33
4	68	76	0.572	0.406	-8.624	1	10	0.04
...
287	68	65	0.570	0.821	-4.380	1	2	0.26
288	79	74	0.735	0.824	-4.143	0	10	0.03
289	82	80	0.746	0.690	-7.956	1	11	0.16
290	67	76	0.682	0.927	-2.915	1	4	0.04
291	50	60	0.558	0.776	-6.868	1	9	0.17

```
292 rows x 15 columns
```



```
#Now we have to drop or remove the artist_pop and Symbling column
```

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

	track_pop	danceability	energy	loudness	mode	key	speechiness	acoustic
0	80	0.787	0.673	-4.583	0	11	0.1590	0.2
1	90	0.514	0.730	-5.934	1	1	0.0598	0.0
2	63	0.862	0.631	-4.746	1	7	0.0515	0.2
3	70	0.843	0.741	-5.609	1	11	0.3340	0.0
4	76	0.572	0.406	-8.624	1	10	0.0427	0.5
...
287	65	0.570	0.821	-4.380	1	2	0.2670	0.1
288	74	0.735	0.824	-4.143	0	10	0.0360	0.6
289	80	0.746	0.690	-7.956	1	11	0.1640	0.2
290	76	0.682	0.927	-2.915	1	4	0.0479	0.0
291	60	0.558	0.776	-6.868	1	9	0.1790	0.3

292 rows × 14 columns



```
df_numeric.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 292 entries, 0 to 291
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   track_pop             292 non-null    int64
1   danceability          292 non-null    float64
2   energy                292 non-null    float64
3   loudness              292 non-null    float64
4   mode                  292 non-null    int64
5   key                   292 non-null    int64
6   speechiness           292 non-null    float64
7   acousticness          292 non-null    float64
8   instrumentalness      292 non-null    float64
9   liveness              292 non-null    float64
10  valence               292 non-null    float64
11  tempo                 292 non-null    float64
12  time_signature        292 non-null    int64
13  duration_ms           292 non-null    int64
dtypes: float64(9), int64(5)
memory usage: 32.1 KB
```

```
#divide the data into i/p and o/p
```

```
#output - track_name
```

```
#input - All the columns except the track_name column
```

<https://colab.research.google.com/drive/1g8P4qMwg6pb88at2l3KcYl91oE5y-aj#printMode=true>

```
x = df_numeric.iloc[:,0:13].values
```

```
X
array([[237893, 200040, 223488, 155497, 201084, 182200, 161733, 186653,
        170746, 193829, 180147, 210463, 228482, 221820, 223270, 193208,
        216320, 208026, 224661, 158571, 164113, 160000, 188293, 151018,
        164318, 154455, 128926, 106016, 218707, 177856, 179867, 157606,
        154424, 165726, 194253, 170947, 187571, 182693, 173333, 129241,
        239560, 147692, 197217, 147800, 218107, 141673, 176547, 210000,
        222961, 171375, 144013, 141224, 284480, 116630, 386907, 196520,
        154550, 140533, 136839, 134240, 159382, 390639, 242133, 196893,
        205733, 198040, 203093, 162547, 284733, 145920, 163173, 214558,
        133995, 165853, 220779, 163636, 177493, 140527, 195493, 215307,
        237800, 179125, 183290, 194000, 190920, 160097, 215320, 219813,
        205733, 221075, 251013, 110063, 153800, 80842, 232907, 192080,
        238507, 137744, 206221, 263173, 221253, 156443, 216267, 303373,
        224955, 281313, 319467, 159955, 178000, 184060, 205200, 147697,
        157712, 233228, 223258, 177479, 52867, 351467, 154787, 112493,
        116757, 149145, 144935, 229671, 179871, 168387, 137595, 208867,
        242327, 265263, 202667, 177866, 163173, 235988, 235320, 196653,
        230657, 160627, 61466, 100000, 258453, 143613, 180231, 115227,
        281080, 177323, 311867, 167916, 229670, 215200, 166857, 196800,
        195631, 171980, 208729, 248133, 115352, 163902, 200947, 247059,
        176840, 154424, 176960, 260640, 148040, 261880, 272507, 83940,
        200774, 164640, 174000, 278719, 165733, 142044, 232187, 266840,
        210285, 346436, 146523, 197213, 195213, 383547, 219427, 155867,
        204533, 154456, 117363, 215627, 148640, 132303, 174933, 229147,
        216013, 232368, 197933, 144000, 220487, 37632, 160907, 165391,
        279240, 229640, 242001, 180139, 220587, 118075, 177773, 249280,
        230895, 107404, 195429, 181852, 186467, 139413, 237493, 200307,
        115200, 215387, 227267, 157507, 209848, 165938, 68586, 158436,
        209709, 176787, 135016, 180672, 235767, 228173, 253107, 154567,
        60000, 118776, 151579, 166627, 248056, 234093, 153294, 214227,
        205333, 234945, 129371, 207747, 176640, 179133, 163019, 205296,
        276333, 109595, 210329, 235988, 77049, 187709, 223080, 164880,
        467587, 195429, 192177, 209274, 175467, 201990, 250337, 202907,
        223747, 191340, 110253, 337640, 215507, 173938, 188735, 292799,
        163574, 139741, 169020, 320680, 211150, 180161, 158707, 151520,
        246765, 177280, 244760, 279307, 204760, 205347, 206227, 225560,
        211666, 181733, 195613, 114660])
```

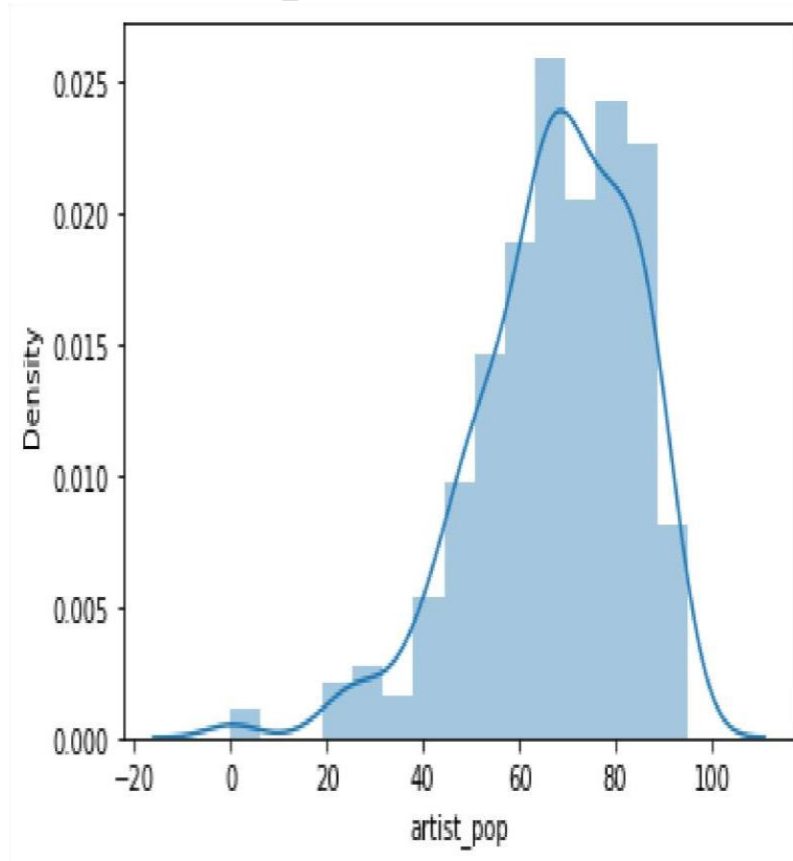
```
array([[ 80. ,  0.787,  0.673, ...,  0.779, 110.962,  4.  ],
       [ 90. ,  0.514,  0.73 , ...,  0.334, 171.005,  4.  ],
       [ 63. ,  0.862,  0.631, ...,  0.841, 128.978,  4.  ],
       ...,
       [ 80. ,  0.746,  0.69 , ...,  0.497,  89.977,  4.  ],
       [ 76. ,  0.682,  0.927, ...,  0.744, 101.019,  4.  ],
       [ 60. ,  0.558,  0.776, ...,  0.247, 128.064,  4.  ]])
```

```
y = df_numeric.iloc[:,13].values
y
```

#3.VISUALIZATION

```
import seaborn as sns
sns.distplot(df['artist_pop']) #distribution plot
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning
warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7faf9f5c5a10>
```



#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)
```

#7.RUN a CLASSIFIER/REGRESSOR/CLUSTERER

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
#Whatever data splitting /data allocation happens to the xtrain,x_test,ytrain,ytest var
#By default the training variables get 75% and testing variables get 25%
```

```
print(x.shape) # 266 rows and 64 columns
print(x_train.shape) # 266 rows and 64 columns (75%)
print(x_test.shape) #266 rows and 32 columns (25%)
```



```
(292, 13)  
(219, 13)  
(73, 13)
```

```
print(y.shape) #266 rows and 64 columns  
print(y_train.shape) #266 rows and 32 columns (75%)  
print(y_test.shape) #266 rows and 32 columns (25%)
```

```
(292,)  
(219,)  
(73,)
```

```
#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)
```

```
#8.MODEL FITTING
```

```
model.fit(x_train,y_train)
```

```
LinearRegression()
```

```
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([204420.76687172, 216621.10522868, 208073.36797116, 246354.63809667,
       220146.06293435, 221457.776425 , 202256.96397327, 180060.96331067,
       222773.44765844, 203117.43492293, 205305.97415318, 214444.17428759,
       165198.66443648, 190373.63953747, 211258.7871104 , 184373.3828675 ,
       211299.93010078, 215797.39363239, 224538.70327434, 220099.80843492,
       187545.67710525, 181169.92456849, 202798.5106638 , 214565.92476261,
       214321.50733066, 198679.51941993, 241930.55317749, 228819.43613548,
       187715.79352519, 254831.15407891, 304728.76458028, 237635.67552424,
       223054.16452454, 218854.79674282, 138766.96731257, 199836.03011388,
       222507.34597188, 186447.29361802, 256415.93793059, 215079.95861074,
       248529.3187128 , 206731.81982134, 243189.24434733, 176504.7022658 ,
       210605.28735211, 203868.54740158, 219477.61410327, 179044.4868585 ,
       225459.81743229, 203068.40545753, 207234.80714253, 242456.44731681,
       240337.10435598, 226810.86969058, 228362.27868767, 194909.00790559,
       208440.3316457 , 208717.5513641 , 223399.11637441, 242725.35693852,
       167886.82690319, 218129.9774878 , 169912.06323051, 229476.85836141,
       236688.03628033, 189692.67646151, 213919.36892938, 208164.26354881,
       233804.34862707, 222909.42740046, 215731.32444599, 211330.09283443,
       179548.42355043])
```

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

```
[0.83333333 0.25062035 0.43451708 0.81342339 0.          0.
 0.00760586 0.09266437 0.          0.07275181 0.09708006 0.04088615
 0.5          ]
```

```
#INDIVIDUAL PREDICTION
```

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

```
array([252781.42545176])
```

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 - Convert Image into Gray Scale

#Step - 3 - Inverted Gray Scale Image [For Shifting toward selected channel]

#Step - 4 - Apply Image Smoothing For Shading effect

#Step - 5 - Invert Blur Image and Apply division between gray and invert_blur.

#-----

#Step-1 import numpy

as np import cv2

#Read Image-----img =

cv2.imread("hulk.jpg") img =

cv2.resize(img,(800,600))

#Create Trackbar----

def nothing(x): pass

#window name

cv2.namedWindow("Color Adjustments",cv2.WINDOW_NORMAL) cv2.resizeWindow("Color Adjustments", (300, 300))

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,(21,21),0)

#Step -5

inverted_blur = clr - blur_img #inverted blurred image fltr =

cv2.divide(gray,inverted_blur,scale = scale)

```
#Output-----  
cv2.imshow("opt",fltr)    k = cv2.waitKey(1)  
    if k == ord("q"):  
break    if k == ord("s"):  
        cv2.imwrite("res.jpg",fltr)  
        cv2.destroyAllWindows()
```

GithubAccountLink: <https://github.com/42abcdefg/MajorProject>