NAME: HRISHAV KARMAKAR

COLLEGE: TECHNO INTERNATIONAL NEW TOWN

BRANCH: ELECTRONICS AND COMMUNICATION ENGINEERING

YEAR: 3RD

GITHUB ACC.LINK: https://github.com/HRISHAV18

https://github.com/HRISHAV18/RinexProject

Heroku deployment link: https://majorproject-rinex.herokuapp.com/

1.MAJOR PROJECT 1
Machine learning-supervised learning-linear regression
Linear Regression
Dataset- Price Prediction
Heroku deployment

2.MAJOR PROJECT 2
Image Processing Projects using NumPy and OpenCV.

```
#30th AUGUST , 2022
```

```
#MACHINE LEARNING - SUPERVISED LEARNING -REGRESSION - LINEAR REGRESSION
# Linear Regression
#Dataset - Price Prediction
#Dataset - https://raw.githubusercontent.com/HRISHAV18/RinexProject/main/scrap%20price.csv
```

```
#1.Take the data and create dataframe
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/HRISHAV18/RinexProject/main/scrap%20pr
df
```

	ID	symboling	name	fueltypes	aspiration	doornumbers	carbody	drive
0	1	3	alfa-romero giulia	gas	std	two	convertible	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	
3	4	2	audi 100 ls	gas	std	four	sedan	
4	5	2	audi 100ls	gas	std	four	sedan	
...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	
202	203	-1	volvo 244dl	gas	std	four	sedan	
203	204	-1	volvo 246	diesel	turbo	four	sedan	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	

205 rows × 26 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                205 non-null    int64
 1   symboling         205 non-null    int64
 2   name              205 non-null    object
 3   fueltypes         205 non-null    object
```

```
 4   aspiration        205 non-null    object
 5   doornumbers       205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheels       205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

```
df.shape
```

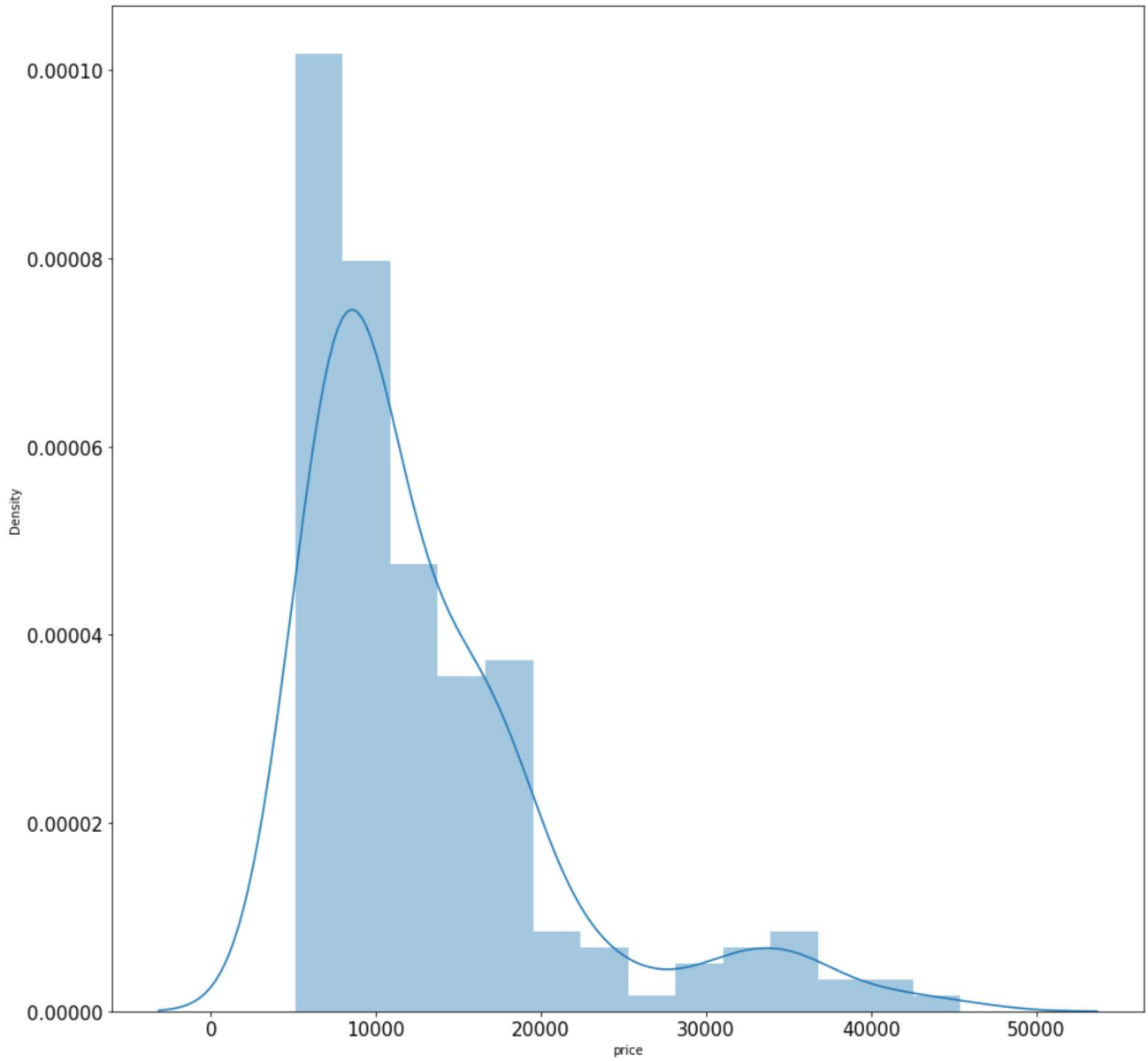
```
(205, 26)
```

```
df.size
```

```
5330
```

```
#VISUALISATION
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize = (15,15))
plt.xticks( fontsize=15)
plt.yticks( fontsize=15)
sns.distplot(df['price']) #distribution plot
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b0d3b2190>
```




```
#Now we have to remove or drop the ID and Symbling column
df = df.drop(columns=['ID','symboling'])
```

```
df.isna().sum()
```

```
name              0
fueltypes         0
aspiration        0
doornumbers       0
carbody           0
drivewheels       0
enginelocation    0
wheelbase         0
carlength         0
carwidth          0
carheight         0
curbweight        0
```

```
enginetype         0
cylindernumber     0
enginesize         0
fuelsystem         0
boreratio          0
stroke             0
compressionratio   0
horsepower         0
peakrpm            0
citympg            0
highwaympg         0
price              0
dtype: int64
```

```
df
```

	name	fueltypes	aspiration	doornumbers	carbody	drivewheels	enginelc
0	alfa-romero giulia	gas	std	two	convertible	rwd	
1	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	audi 100 ls	gas	std	four	sedan	fwd	
4	audi 100ls	gas	std	four	sedan	4wd	
...	...	...	...	...	...	...	
200	volvo 145e (sw)	gas	std	four	sedan	rwd	
201	volvo 144ea	gas	turbo	four	sedan	rwd	
202	volvo 244dl	gas	std	four	sedan	rwd	
203	volvo 246	diesel	turbo	four	sedan	rwd	
204	volvo 264gl	gas	turbo	four	sedan	rwd	

205 rows × 24 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 24 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   name              205 non-null    object
 1   fueltypes         205 non-null    object
 2   aspiration        205 non-null    object
```

```
 3   doornumbers       205 non-null    object
 4   carbody           205 non-null    object
 5   drivewheels       205 non-null    object
 6   enginelocation    205 non-null    object
 7   wheelbase         205 non-null    float64
 8   carlength         205 non-null    float64
 9   carwidth          205 non-null    float64
 10  carheight         205 non-null    float64
 11  curbweight        205 non-null    int64
 12  enginetype        205 non-null    object
 13  cylindernumber    205 non-null    object
 14  enginesize        205 non-null    int64
 15  fuelsystem        205 non-null    object
 16  boreratio         205 non-null    float64
 17  stroke            205 non-null    float64
 18  compressionratio  205 non-null    float64
 19  horsepower        205 non-null    int64
 20  peakrpm           205 non-null    int64
 21  citympg           205 non-null    int64
 22  highwaympg        205 non-null    int64
 23  price             205 non-null    float64
dtypes: float64(8), int64(6), object(10)
memory usage: 38.6+ KB
```

```
#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
```

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	s
0	88.6	168.8	64.1	48.8	2548	130	3.47	
1	88.6	168.8	64.1	48.8	2548	130	3.47	
2	94.5	171.2	65.5	52.4	2823	152	2.68	
3	99.8	176.6	66.2	54.3	2337	109	3.19	
4	99.4	176.6	66.4	54.3	2824	136	3.19	
...	...	...	...	...	...	...	...	
200	109.1	188.8	68.9	55.5	2952	141	3.78	
201	109.1	188.8	68.8	55.5	3049	141	3.78	
202	109.1	188.8	68.9	55.5	3012	173	3.58	
203	109.1	188.8	68.9	55.5	3217	145	3.01	
204	109.1	188.8	68.9	55.5	3062	141	3.78	

205 rows × 14 columns

```
df_numeric.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
```

```
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   wheelbase         205 non-null    float64
 1   carlength         205 non-null    float64
 2   carwidth          205 non-null    float64
 3   carheight         205 non-null    float64
 4   curbweight        205 non-null    int64  
 5   enginesize        205 non-null    int64  
 6   boreratio         205 non-null    float64
 7   stroke            205 non-null    float64
 8   compressionratio  205 non-null    float64
 9   horsepower        205 non-null    int64  
 10  peakrpm           205 non-null    int64  
 11  citympg           205 non-null    int64  
 12  highwaympg        205 non-null    int64  
 13  price             205 non-null    float64
dtypes: float64(8), int64(6)
memory usage: 22.5 KB
```

```
#divide the data into i/p and o/p
#output - price
#input - All the columns except the price column
```

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

```
array([[  88.6,  168.8,   64.1, ..., 5000. ,   21. ,   27. ],
       [  88.6,  168.8,   64.1, ..., 5000. ,   21. ,   27. ],
       [  94.5,  171.2,   65.5, ..., 5000. ,   19. ,   26. ],
       ...,
       [ 109.1,  188.8,   68.9, ..., 5500. ,   18. ,   23. ],
       [ 109.1,  188.8,   68.9, ..., 4800. ,   26. ,   27. ],
       [ 109.1,  188.8,   68.9, ..., 5400. ,   19. ,   25. ]])
```

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

```
array([13495.   , 16500.   , 16500.   , 13950.   , 17450.   , 15250.   ,
       17710.   , 18920.   , 23875.   , 17859.167, 16430.   , 16925.   ,
       20970.   , 21105.   , 24565.   , 30760.   , 41315.   , 36880.   ,
        5151.   ,  6295.   ,  6575.   ,  5572.   ,  6377.   ,  7957.   ,
        6229.   ,  6692.   ,  7609.   ,  8558.   ,  8921.   , 12964.   ,
        6479.   ,  6855.   ,  5399.   ,  6529.   ,  7129.   ,  7295.   ,
        7295.   ,  7895.   ,  9095.   ,  8845.   , 10295.   , 12945.   ,
       10345.   ,  6785.   ,  8916.5  ,  8916.5  , 11048.   , 32250.   ,
       35550.   , 36000.   ,  5195.   ,  6095.   ,  6795.   ,  6695.   ,
        7395.   , 10945.   , 11845.   , 13645.   , 15645.   ,  8845.   ,
        8495.   , 10595.   , 10245.   , 10795.   , 11245.   , 18280.   ,
       18344.   , 25552.   , 28248.   , 28176.   , 31600.   , 34184.   ,
       35056.   , 40960.   , 45400.   , 16503.   ,  5389.   ,  6189.   ,
        6669.   ,  7689.   ,  9959.   ,  8499.   , 12629.   , 14869.   ,
       14489.   ,  6989.   ,  8189.   ,  9279.   ,  9279.   ,  5499.   ,
        7099.   ,  6649.   ,  6849.   ,  7349.   ,  7299.   ,  7799.   ,
        7499.   ,  7999.   ,  8249.   ,  8949.   ,  9549.   , 13499.   ,
       14399.   , 13499.   , 17199.   , 19699.   , 18399.   , 11900.   ,
       13200.   , 12440.   , 13860.   , 15580.   , 16900.   , 16695.   ,
```

```
       17075.   , 16630.   , 17950.   , 18150.   ,  5572.   ,  7957.   ,
        6229.   ,  6692.   ,  7609.   ,  8921.   , 12764.   , 22018.   ,
       32528.   , 34028.   , 37028.   , 31400.5  ,  9295.   ,  9895.   ,
       11850.   , 12170.   , 15040.   , 15510.   , 18150.   , 18620.   ,
        5118.   ,  7053.   ,  7603.   ,  7126.   ,  7775.   ,  9960.   ,
        9233.   , 11259.   ,  7463.   , 10198.   ,  8013.   , 11694.   ,
        5348.   ,  6338.   ,  6488.   ,  6918.   ,  7898.   ,  8778.   ,
        6938.   ,  7198.   ,  7898.   ,  7788.   ,  7738.   ,  8358.   ,
        9258.   ,  8058.   ,  8238.   ,  9298.   ,  9538.   ,  8449.   ,
        9639.   ,  9989.   , 11199.   , 11549.   , 17669.   ,  8948.   ,
       10698.   ,  9988.   , 10898.   , 11248.   , 16558.   , 15998.   ,
       15690.   , 15750.   ,  7775.   ,  7975.   ,  7995.   ,  8195.   ,
        8495.   ,  9495.   ,  9995.   , 11595.   ,  9980.   , 13295.   ,
       13845.   , 12290.   , 12940.   , 13415.   , 15985.   , 16515.   ,
       18420.   , 18950.   , 16845.   , 19045.   , 21485.   , 22470.   ,
       22625.   ])
```

```
#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)
```

```
print(x.shape) # 205 rows and 13 cols
print(x_train.shape) #153 rows and 13 cols (75%)
print(x_test.shape) # 52 rows and 13 cols (25%)
```

```
(205, 13)
(153, 13)
(52, 13)
```

```
print(y.shape) # 205 rows and 1 col
print(y_train.shape) # 153 rows and 1 cols(75 %)
print(y_test.shape) #52rows and 1 col(25%)
```

```
(205,)
(153,)
(52,)
```

```
#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([ 7228.83749589, 21490.68704221, 16843.92578075,  -408.88573459,
       12806.88877797, 14864.06930231,  6815.21828468,  7477.72764958,
       24370.76083649, 10122.57200019, 17227.20066046, 38255.87797552,
       11256.84858797, 15515.8531891 ,  6862.33962216, 14495.22658006,
       12908.29759634, 20089.7360149 , 12112.9445916 ,  7245.99470315,
       12180.32751173, 18941.52014488, 12446.19081036, 14964.92473417,
       22958.69376844,  8925.63443014,  7732.359849  , 18883.87938555,
        7815.43702522,  7137.47508384, 11136.70116598, 12703.41505146,
       22485.14049779,  8920.58002031,  7278.67268845, 28586.72492631,
       12234.13615858, 18290.11431538,  8085.5497993 , 38649.45073682,
        6761.48419029, 17525.36347714, 31858.57496193, 18698.70364197,
       12166.49554998,  8716.73070853,  8381.49914616, 17049.80445173,
       11020.16015803, 11326.88569464, 20247.44266236,  7818.27319638])
```

```
y_test #ACTUAL VALUES
```

```
array([ 6795. , 15750. , 15250. ,  5151. ,  9995. , 11199. ,  5389. ,
        7898. , 17199. ,  6529. , 20970. , 31400.5, 10945. , 18344. ,
        8916.5,  9989. ,  9295. , 18920. ,  7895. ,  6488. ,  9959. ,
       15580. ,  9895. , 11549. , 15998. ,  5118. ,  6938. , 16695. ,
        8358. ,  5499. ,  7975. , 12290. , 22018. ,  8948. ,  6849. ,
       41315. , 11595. , 18150. ,  6377. , 45400. ,  8916.5, 17450. ,
       34184. , 15040. , 11259. ,  7609. ,  7609. , 14869. , 11694. ,
        8495. , 23875. ,  7099. ])
```

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

```
[0.28862974 0.48503937 0.07070707 0.525      0.280068   0.15625
 0.41964286 0.61437908 0.125      0.16190476 0.67346939 0.38888889
 0.44736842]
```

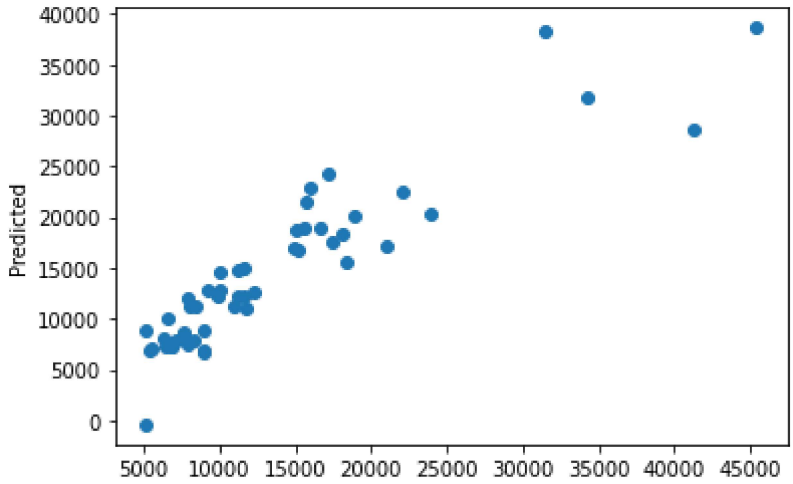
```
#INDIVIDUAL PREDICTION
model.predict([x_train[10]])# price of the car
```

```
array([8070.06474751])
```

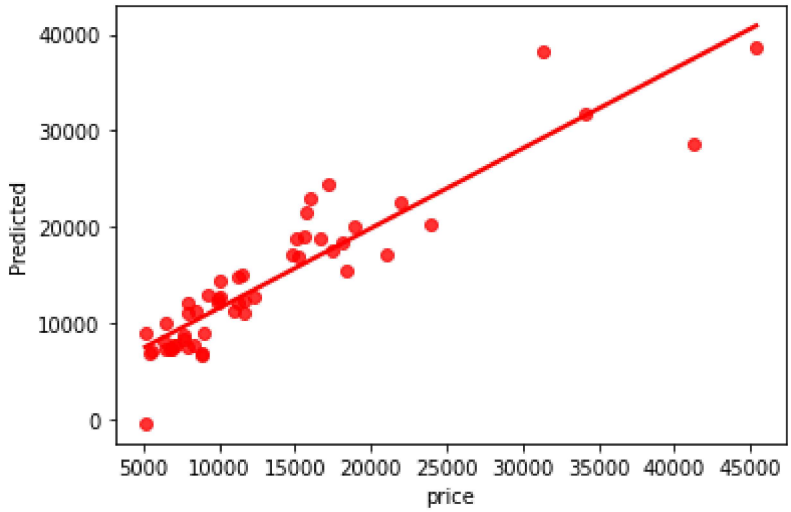
```
#INDIVIDUAL PREDICTION
model.predict([x_train[5]])
```

```
array([17024.72437501])
```

```
plt.scatter(y_test,y_pred);
plt.xlabel('Actual');
plt.ylabel('Predicted');
```




```
sns.regplot(x=y_test,y=y_pred,ci=None,color ='red');
plt.xlabel('price');
plt.ylabel('Predicted');
```




```
pred_df_numeric=pd.DataFrame({'Actual Value':y_test,'Predicted Value':y_pred,'Difference':
```

```
pred_df_numeric
```

	Actual Value	Predicted Value	Difference
0	6795.0	7228.837496	-433.837496
1	15750.0	21490.687042	-5740.687042
2	15250.0	16843.925781	-1593.925781
3	5151.0	-408.885735	5559.885735
4	9995.0	12806.888778	-2811.888778
5	11199.0	14864.069302	-3665.069302
6	5389.0	6815.218285	-1426.218285
7	7898.0	7477.727650	420.272350
8	17199.0	24370.760836	-7171.760836
9	6529.0	10122.572000	-3593.572000
10	20970.0	17227.200660	3742.799340
11	31400.5	38255.877976	-6855.377976
12	10945.0	11256.848588	-311.848588
13	18344.0	15515.853189	2828.146811
14	8916.5	6862.339622	2054.160378
15	9989.0	14495.226580	-4506.226580
16	9295.0	12908.297596	-3613.297596
17	18920.0	20089.736015	-1169.736015
18	7895.0	12112.944592	-4217.944592
19	6488.0	7245.994703	-757.994703
20	9959.0	12180.327512	-2221.327512
21	15580.0	18941.520145	-3361.520145
22	9895.0	12446.190810	-2551.190810
23	11549.0	14964.924734	-3415.924734
24	15998.0	22958.693768	-6960.693768
25	5118.0	8925.634430	-3807.634430
26	6938.0	7732.359849	-794.359849
27	16695.0	18883.879386	-2188.879386
28	8358.0	7815.437025	542.562975
29	5499.0	7137.475084	-1638.475084
30	7975.0	11136.701166	-3161.701166
31	12290.0	12703.415051	-413.415051
32	22018.0	22485.140498	-467.140498

32	22018.0	22485.140498	-467.140498
33	8948.0	8920.580020	27.419980
34	6849.0	7278.672688	-429.672688
35	41315.0	28586.724926	12728.275074
36	11595.0	12234.136159	-639.136159
37	18150.0	18290.114315	-140.114315
38	6377.0	8085.549799	-1708.549799
39	45400.0	38649.450737	6750.549263
40	8916.5	6761.484190	2155.015810
41	17450.0	17525.363477	-75.363477
42	34184.0	31858.574962	2325.425038

```
# FOR HEROKU APP DEPLOY
# INPUT IS CAR'S NAME
# OUTPUT IS FUELTYPES
```

45	7609.0	8716.730709	-1107.730709				

```
df_object = df.select_dtypes(include = ['object'])
df_object
```

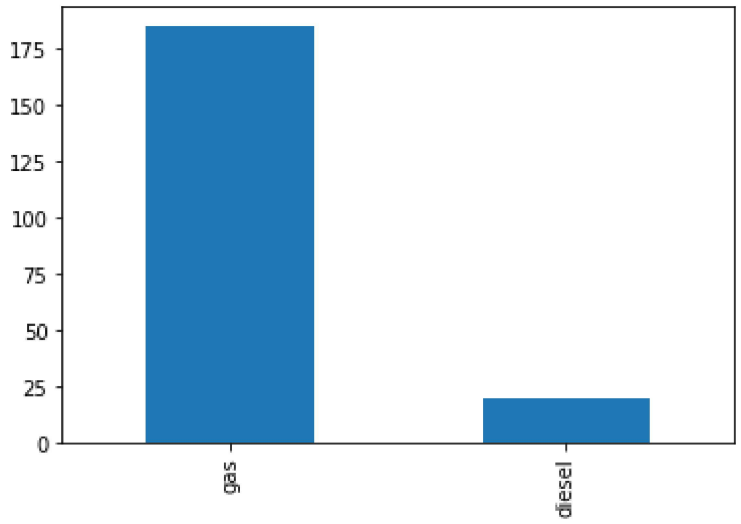
	name	fueltypes	aspiration	doornumbers	carbody	drivewheels	enginelc
0	alfa-romero giulia	gas	std	two	convertible	rwd	
1	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	audi 100 ls	gas	std	four	sedan	fwd	
4	audi 100ls	gas	std	four	sedan	4wd	
...	...	...	...	...	...	...	
200	volvo 145e (sw)	gas	std	four	sedan	rwd	
201	volvo 144ea	gas	turbo	four	sedan	rwd	
202	volvo 244dl	gas	std	four	sedan	rwd	
203	volvo 246	diesel	turbo	four	sedan	rwd	

```
#I just want to know how many fueltypes  are there
df['fueltypes'].value_counts()
```

```
gas       185
diesel     20
Name: fueltypes, dtype: int64
```

```
#VISUALISATION
df['fueltypes'].value_counts().plot(kind = 'bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b0ba34310>
```




```
#4.divide the data into input and output
x = df.iloc[:,0].values
y = df.iloc[:,1].values
print(x)
print(y)
```

```
'mazda glc' 'mazda rx-7 gs' 'buick electra 225 custom'
'buick century luxus (sw)' 'buick century' 'buick skyhawk'
'buick opel isuzu deluxe' 'buick skylark' 'buick century special'
'buick regal sport coupe (turbo)' 'mercury cougar' 'mitsubishi mirage'
'mitsubishi lancer' 'mitsubishi outlander' 'mitsubishi g4'
'mitsubishi mirage g4' 'mitsubishi g4' 'mitsubishi outlander'
'mitsubishi g4' 'mitsubishi mirage g4' 'mitsubishi montero'
'mitsubishi pajero' 'mitsubishi outlander' 'mitsubishi mirage g4'
'Nissan versa' 'nissan gt-r' 'nissan rogue' 'nissan latio' 'nissan titan'
'nissan leaf' 'nissan juke' 'nissan latio' 'nissan note' 'nissan clipper'
'nissan rogue' 'nissan nv200' 'nissan dayz' 'nissan fuga' 'nissan otti'
'nissan teana' 'nissan kicks' 'nissan clipper' 'peugeot 504'
'peugeot 304' 'peugeot 504 (sw)' 'peugeot 504' 'peugeot 504'
'peugeot 604sl' 'peugeot 504' 'peugeot 505s turbo diesel' 'peugeot 504'
'peugeot 504' 'peugeot 604sl' 'plymouth fury iii' 'plymouth cricket'

'plymouth fury iii' 'plymouth satellite custom (sw)'
'plymouth fury gran sedan' 'plymouth valiant' 'plymouth duster'
'porsche macan' 'porcshce panamera' 'porsche cayenne' 'porsche boxter'
'porsche cayenne' 'renault 12tl' 'renault 5 gtl' 'saab 99e' 'saab 99le'
'saab 99le' 'saab 99gle' 'saab 99gle' 'saab 99e' 'subaru' 'subaru dl'
'subaru dl' 'subaru' 'subaru brz' 'subaru baja' 'subaru r1' 'subaru r2'
'subaru trezia' 'subaru tribeca' 'subaru dl' 'subaru dl'
'toyota corona mark ii' 'toyota corona' 'toyota corolla 1200'
'toyota corona hardtop' 'toyota corolla 1600 (sw)' 'toyota carina'
'toyota mark ii' 'toyota corolla 1200' 'toyota corona' 'toyota corolla'
'toyota corona' 'toyota corolla' 'toyota mark ii'
'toyota corolla liftback' 'toyota corona' 'toyota celica gt liftback'
'toyota corolla tercel' 'toyota corona liftback' 'toyota corolla'
'toyota starlet' 'toyota tercel' 'toyota corolla' 'toyota cressida'
'toyota corolla' 'toyota celica gt' 'toyota corona' 'toyota corolla'
'toyota mark ii' 'toyota corolla liftback' 'toyota corona'
```

```
 'toyota starlet' 'toyouta tercel' 'vokswagen rabbit'
 'volkswagen 1131 deluxe sedan' 'volkswagen model 111' 'volkswagen type 3'
 'volkswagen 411 (sw)' 'volkswagen super beetle' 'volkswagen dasher'
 'vw dasher' 'vw rabbit' 'volkswagen rabbit' 'volkswagen rabbit custom'
 'volkswagen dasher' 'volvo 145e (sw)' 'volvo 144ea' 'volvo 244dl'
 'volvo 245' 'volvo 264gl' 'volvo diesel' 'volvo 145e (sw)' 'volvo 144ea'
 'volvo 244dl' 'volvo 246' 'volvo 264gl']
['gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'diesel' 'gas' 'gas' 'diesel' 'diesel' 'diesel'
 'diesel' 'diesel' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'diesel'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'diesel' 'gas' 'diesel' 'gas' 'diesel'
 'gas' 'diesel' 'gas' 'diesel' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'diesel'
 'diesel' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'gas' 'gas' 'gas' 'diesel' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'diesel' 'gas' 'diesel' 'gas' 'gas' 'diesel' 'gas' 'gas' 'gas'
 'gas' 'diesel' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas' 'gas'
 'gas' 'diesel' 'gas']
```

```
#5.train_test_split
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.Apply TF-IDF vectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
vect = TfidfVectorizer()
x_train_v = vect.fit_transform(x_train)
x_test_v = vect.transform(x_test)
```

```
#7.Apply CLASSIFIER
from sklearn.svm import SVC
model = SVC()
```

```
#8.model fitting
model.fit(x_train_v,y_train)
```

```
SVC()
```

```
#9.Predictor variable/predict the output
y_pred = model.predict(x_test_v)
y_pred #Predicted values
```

```
array(['gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
```

```
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas'], dtype=object)
```

```
y_test # actual values
```

```
array(['gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'diesel', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'diesel'], dtype=object)
```

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

```
96.15384615384616
```

```
#Evaluating a specific name (1st)
a = df['name'][10]
a
```

```
'bmw 320i'
```

```
a = vect.transform([a])
model.predict(a)
```

```
array(['gas'], dtype=object)
```

```
#Evaluating a specific name (2nd)
b = df['name'][12] #12th index from the message column
b
```

```
'bmw x1'
```

```
b = vect.transform([b])
model.predict(b)
```

```
array(['gas'], dtype=object)
```

```
#Evaluating by taking custom texts
c = 'audi'
c
```

```
'audi'
```

```
c = vect.transform([c])
model.predict(c)
```

```
array(['gas'], dtype=object)
```

```
#If ever I have to deploy my model,I will have to perform pipelining
```

```
#Pipelining
from sklearn.pipeline import make_pipeline
text_model = make_pipeline(TfidfVectorizer(),SVC())
text_model.fit(x_train,y_train)
```

```
Pipeline(steps=[('tfidfvectorizer', TfidfVectorizer()), ('svc', SVC())])
```

```
#predictor varibale
y_pred1 = text_model.predict(x_test)
y_pred1 # these are predicted outputs for pipelined model
```

```
array(['gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas'], dtype=object)
```

```
y_test #Actual output
```

```
array(['gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'diesel', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'gas',
       'gas', 'gas', 'gas', 'gas', 'gas', 'gas', 'diesel'], dtype=object)
```

```
#To check the accuracy of the pipelined model
accuracy_score(y_pred1,y_test)*100
```

```
96.15384615384616
```

```
#Individual Prediction/Evaluation of a specific name
a1 = df['name'][2]
a1
```

```
'alfa-romero Quadrifoglio'
```

```
text_model.predict([a1])
```

```
array(['gas'], dtype=object)
```

```
#JOBLIB  - 2 different types - 1.Dump and 2.Load
import joblib
joblib.dump(text_model,'gas-diesel')
#We are creating a newfile called gas-diesel and we are dumping our pipelined model
#inside it.
```

```
['gas-diesel']
```

```
#We are creating a STREAMLIT WEB APPLICATION
```

```
#Temporary deployment - Local host
```

```
!pip install streamlit --quiet #Installing the streamlit library
```

```
     |████████████████████████████████| 9.1 MB 7.3 MB/s
     |████████████████████████████████| 164 kB 56.5 MB/s
     |████████████████████████████████| 4.7 MB 43.4 MB/s
     |████████████████████████████████| 235 kB 37.7 MB/s
     |████████████████████████████████| 181 kB 4.7 MB/s
     |████████████████████████████████| 78 kB 4.8 MB/s
     |████████████████████████████████| 63 kB 1.4 MB/s
     |████████████████████████████████| 51 kB 5.3 MB/s
  Building wheel for validators (setup.py) ... done
```

```
%%writefile app.py
#%%writefile is amagic command to create app.py file
import streamlit as st
import joblib
model = joblib.load('gas-diesel')
st.title('GAS-DIESEL CLASSIFIER') #creates a title in web app
ip = st.text_input('Enter the message') #creates a text box in web app
op = model.predict([ip])
if st.button('Predict'):
  st.title(op[0]) # st.button will create a button with name Predict
  #st.title(op[0]) # the output will be displayed as a title
```

```
Writing app.py
```

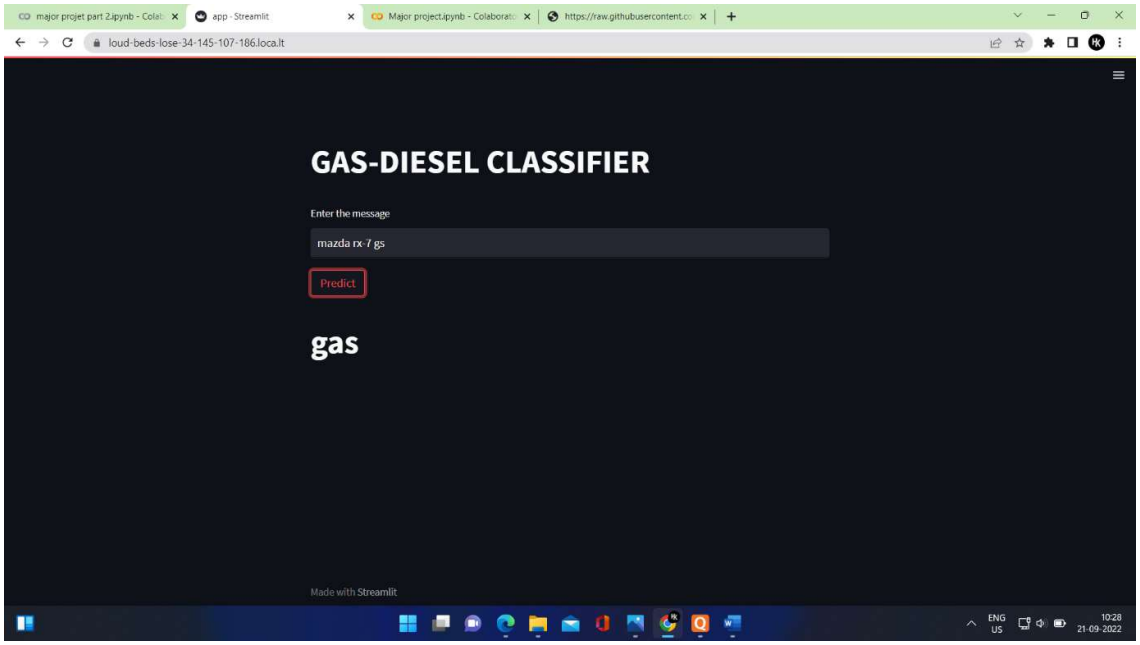
```
#TEMPORARY DEPLOYMENT PART OF WEB APPLICATION
!streamlit run app.py & npx localtunnel --port 8501
#8501 is the default port number for local tunnel
```

```
2022-09-21 04:52:04.219 INFO    numexpr.utils: NumExpr defaulting to 2 threads.

  You can now view your Streamlit app in your browser.

  Network URL: http://172.28.0.2:8501
  External URL: http://34.145.107.186:8501

npx: installed 22 in 5.583s
your url is: https://loud-beds-lose-34-145-107-186.loca.lt
```

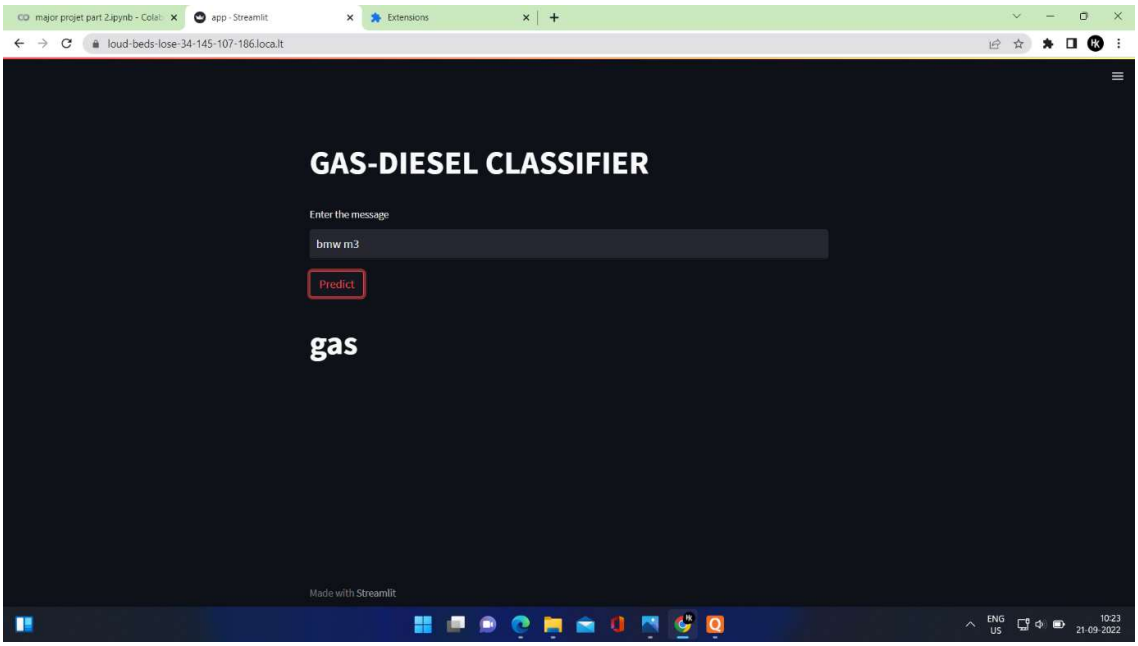
Colab paid products - Cancel contracts here

LOCAL APP DEPLOY

screenshots of local app 1:

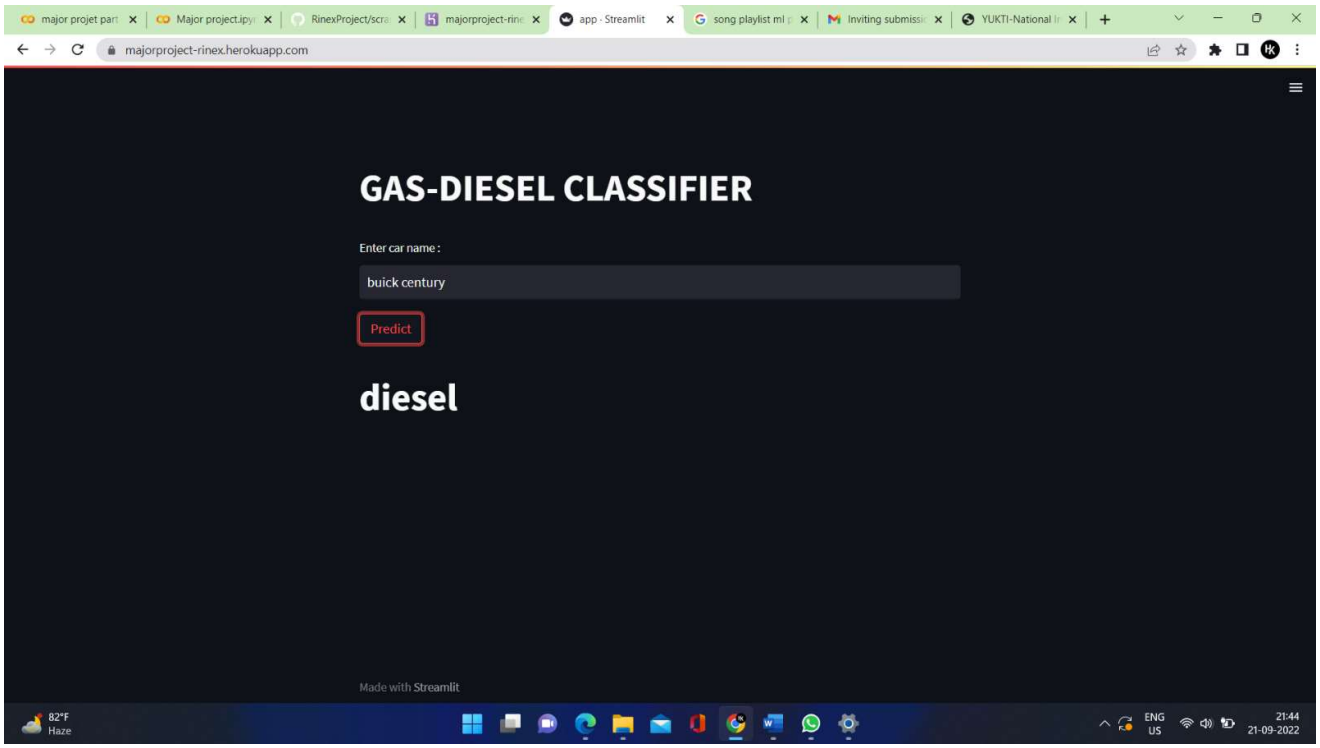


screenshots of local app 2:



Heroku deployment link: https://majorproject-rinex.herokuapp.com/

screenshots of web app 1:



screenshots of web app 2: