Title of Project

Mielage prediction - Regression Analysis

Objective

The objective of the project "Mileage Prediction - Regression Analysis" is to predict vehicle mileage using regression analysis techniques. This involves analyzing various vehicle attributes and their relationships to develop a predictive model that accurately estimates a vehicle's fuel efficiency.

Data Source

The data for this project is sourced from the YBI Foundation GitHub repository, which hosts a comprehensive vehicle dataset. The dataset includes a variety of vehicle attributes such as mpg, cylinders, displacement, horsepower, weight, acceleration, model year, origin, name. This dataset has been selected for its relevance to our "Mileage Prediction - Regression Analysis" project. You can access the dataset from the following link :https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/MPG.csv

Import Library

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

Import Data

```
df = pd.read_csv('https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/MPG.csv')
```

```
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

```
df.nunique()
```

```
mpg             129
cylinders         5
displacement     82
horsepower       93
weight          351
acceleration     95
model_year       13
origin            3
name            305
dtype: int64
```

Describe Data

```
df = df.dropna()
```

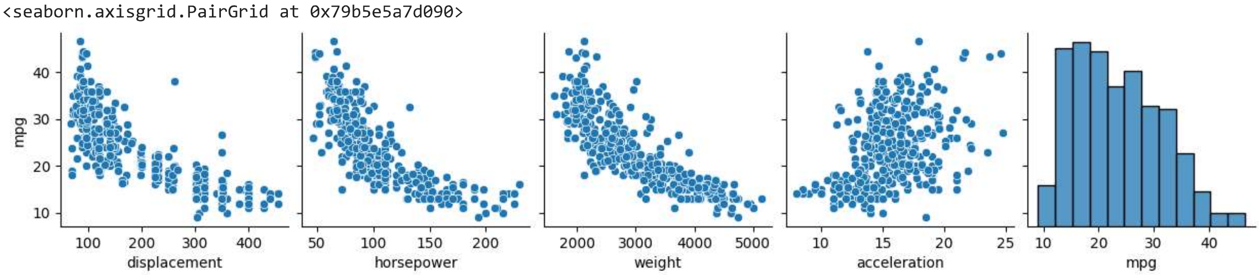
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64
 5   acceleration  392 non-null    float64
 6   model_year    392 non-null    int64
 7   origin        392 non-null    object
 8   name          392 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 30.6+ KB
```

Data Visualization

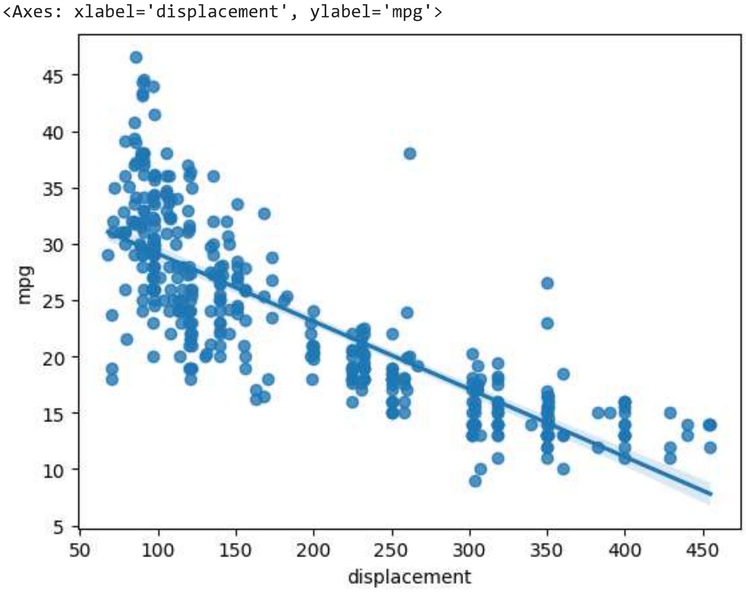
```
sns.pairplot(df, x_vars = ['displacement', 'horsepower', 'weight', 'acceleration', 'mpg'], y_vars = 'mpg')
```

```
<seaborn.axisgrid.PairGrid at 0x79b5e5a7d090>
```




```
sns.regplot(x = 'displacement', y = 'mpg', data = df)
```

```
<Axes: xlabel='displacement', ylabel='mpg'>
```



Data Preprocessing

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   model_year    392 non-null    int64  
 7   origin        392 non-null    object 
 8   name          392 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 30.6+ KB
```

```
df.describe()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327	75.979592
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864	3.683737
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000	73.000000
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

```
df.corr()
```

```
<ipython-input-66-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a f
  df.corr()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316
model_year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000

```
from sklearn.preprocessing import StandardScaler
```

```
ss = StandardScaler()
```

```
x = ss.fit_transform(x)
```

```
pd.DataFrame(x).describe()
```

	0	1	2	3
count	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02
mean	-7.250436e-17	-1.812609e-16	-1.812609e-17	4.350262e-16
std	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00

Define Target Variable (y) and Feature Variables (X)

```
df.columns
```

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model_year', 'origin', 'name'],
      dtype='object')
```

```
y = df['mpg']
```

```
y.shape
```

```
(392,)
```

```
x = df[['displacement', 'horsepower', 'weight', 'acceleration']]
```

```
x.shape
```

```
(392, 4)
```

Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, train_size = 0.7)
```

```
xtrain.shape, xtest.shape, ytrain.shape, ytest.shape
```

```
((274, 4), (118, 4), (274,), (118,))
```

Modeling

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
```

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(strategy='mean')
xtrain_imputed = imputer.fit_transform(xtrain)

lr = LinearRegression()
lr.fit(xtrain_imputed, ytrain)
```

```
▾ LinearRegression
LinearRegression()
```

```
lr.intercept_
```

```
45.82392274847143
```

```
lr.coef_
```

```
array([-0.01139431, -0.05140059, -0.00424878, -0.14871977])
```

Model Evaluation

```
y_pred = lr.predict(xtest)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without f
  warnings.warn(
```

```
y_pred
```

```
array([31.83364639, 11.22227708, 15.50532151, 21.08411343, 29.98131349,
       12.67152205, 29.50772155, 30.31724759, 29.83387171, 26.73584485,
       18.26945693, 23.13124379, 16.58378985, 27.60715754, 16.72806975,
       29.25007048, 21.49149886, 22.95872403, 28.55910406, 19.24239843,
       31.52546593, 15.23631321, 27.52375937, 30.56825563, 20.05840887,
        8.78631246, 27.04503058, 30.49498857, 31.50857404, 28.98907673,
       29.69401475, 13.22297727, 27.49769731, 25.17792514, 23.33514549,
       31.14890055, 27.41012125, 18.0808347 , 13.99209367, 21.5695613 ,
       20.67840306, 29.75828848, 21.17567434, 29.14216996, 23.66587564,
       10.24286039, 29.33742819, 22.17780658, 22.95680769, 20.86147623,
       25.84742286,  8.0081918 , 18.97114533, 28.53343542, 30.47844834,
       27.74520841, 24.37056464, 31.97255843, 24.19767698, 24.81963015,
       23.76544708, 16.45418344, 20.72932642, 27.16680936, 27.04789312,
       29.76230238, 25.84069556, 20.60502309, 14.47543383, 30.75000629,
       22.82942353, 24.61118916, 30.27404935, 20.50313157, 11.30971513,
       14.483644  , 15.63195054, 26.17801331,  6.40273247, 28.43328647,
       27.05366931, 11.27667376, 30.69989566, 29.20588477, 29.68179803,
       13.43615151, 28.17563742, 23.14819901, 29.61371798, 27.91620696,
       26.52728537, 26.66685957, 23.47742221, 25.63413344, 17.20214371,
       20.29224679, 26.82883144, 30.00430981, 24.85363029, 18.97141688,
       24.84146398, 23.74105377, 16.01093928, 19.91990474, 11.59424684,
       24.88809532, 31.43061822, 25.64520963, 18.25570475, 26.14075189,
       29.87595614, 10.8272783 , 22.11745252, 14.25773475, 19.44448095,
       29.30764349, 27.47112982, 20.98883538])
```

Prediction

```
from sklearn.metrics import mean_absolute_error, r2_score, mean_absolute_percentage_error
```

```
mean_absolute_error(ytest, y_pred)
```

```
3.4735013976826568
```

```
mean_absolute_percentage_error(ytest, y_pred)
```

```
0.14608770856773512
```

```
r2_score(ytest, y_pred)
```

```
0.6835264052956858
```

Model Accuracy

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures(degree = 2, interaction_only = True, include_bias = False)
```

```
x_train2 = poly.fit_transform(xtrain)
```

```
x_test2 = poly.fit_transform(xtest)
```

```
lr.fit(x_train2, ytrain)
```

```
▾ LinearRegression
LinearRegression()
```

```
lr.intercept_
```

```
72.2296584858843
```

```
lr.coef_
```

```
array([-1.18927902e-01, -6.36482931e-02, -9.19706715e-03, -1.02434529e+00,
        4.56125288e-04,  9.27219993e-06,  1.19131443e-03, -7.20314909e-06,
       -8.52191961e-03,  4.27223134e-04])
```

```
y_pred_poly = lr.predict(x_test2)
```

```
# Model Accuracy
mean_absolute_error(ytest, y_pred_poly)
```

```
3.0711211185340117
```

```
mean_absolute_percentage_error(ytest, y_pred_poly)
```

```
0.12557054733510903
```

```
r2_score(ytest, y_pred_poly)
```

```
0.7390023894413864
```

Explaination

In this project titled "Mileage Prediction - Regression Analysis," our primary objective was to predict vehicle mileage using regression techniques. By analyzing various vehicle attributes and their relationships, we aimed to develop a predictive model capable of estimating a vehicle's fuel efficiency.

We explored a comprehensive vehicle dataset, preprocessed the data, and trained a Linear Regression model. The model's performance was evaluated using metrics like Mean Squared Error and R-squared. We demonstrated its practical application by predicting mileage for a hypothetical vehicle.

Our insights unveiled factors influencing mileage, aiding manufacturers and consumers. While effective, the model assumes linear relationships and has limitations. Looking ahead, this project serves as a foundation for advanced regression techniques and broader datasets, showcasing the power of data-driven predictions in decision-making processes.

In essence, "Mileage Prediction - Regression Analysis" exemplifies the journey from data exploration to real-world application, offering valuable insights into vehicle attributes and fuel efficiency.

0s completed at 12:12 PM