PROJECT 1

By Trey Gower

https://github.com/TreyGower7/COE_379L_Projects/tree/main/Project_1

Data Preparation and Insights

Preparing data is pertinent to any sort of analysis that will take place. The motivation behind it remains in the fact that data is flawed and human intervention can contribute to a much better outcome of results. For this reason, I explored and performed a few steps to make the Cars dataset more optimal for the staging of analysis. The dataset contained a few parameters including mpg, number of cylinders, displacement, horsepower, weight, acceleration, model_year, origin, and car_name. Of which, in the data exploration phase, some notable ones are these: horsepower, origin, and car_name.

Notes on origin and car_name:

The interest arose in that origin had one hot encoding already for 1.American 2.European and 3.Japanese. And Car_name needed to be converted to string type from object type. That is about all that had to be done for those.

Horsepower Exploration:

There are some question marks acting as null values that we would like to deal with:

```
#Problem 1 data missing?
cars_ds.isnull().sum()
#Nothing appears lets double check using unique
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

```
cars_ds['horsepower'].unique()
#We can see a ? representing null values
```

```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
```

```
ind = cars_ds[cars_ds['horsepower'] == '?'].index
cars_ds[cars_ds['horsepower'] == '?']
#Get the data for these null values
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	car_name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

```
#Replacing the question marks with null values now
import numpy as np
cars_ds.replace('?', np.nan, inplace = True)
```

```
#Check it worked and we see it does
cars_ds.isnull().sum()
```

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
car_name        0
dtype: int64
```

```
#replace null values with mean
cars_ds['horsepower'].fillna(cars_ds['horsepower'].mean() ,inplace = True)
```

PROJECT 1

By Trey Gower

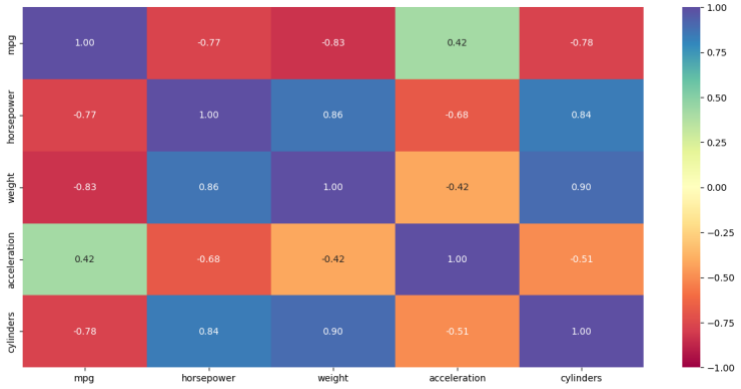
```
for i in ind:
    print(cars_ds[cars_ds.index == i]["horsepower"])
```

```
32    104.469388
Name: horsepower, dtype: float64
126    104.469388
Name: horsepower, dtype: float64
330    104.469388
Name: horsepower, dtype: float64
336    104.469388
Name: horsepower, dtype: float64
354    104.469388
Name: horsepower, dtype: float64
374    104.469388
Name: horsepower, dtype: float64
```

Linear Regression

The goal of this project was to train a model to predict the fuel efficiency of a car based on specific parameters that influence the Miles Per Gallon (MPG) of the vehicle. These parameters in the dataset include the number of cylinders, horsepower, weight, and acceleration. Since only these columns were necessary for accurately predicting fuel efficiency, other columns such as model_year, origin, car_name, and displacement were dropped.

After selecting the relevant data, I conducted an analysis on the correlation between each parameter and fuel efficiency. A negative correlation indicates that as the weight of the car decreases, for example, MPG increase. This plot provides a more detailed visualization of this correlation.



In training the model, a fairly standard procedure was followed. I utilized scikit-learn's train_test_split function to divide the data into training and testing datasets. The split ratio employed was 30% for testing data and 70% for training data. Subsequently, a linear regression model was fitted to the training data.

Model Performance

The model's performance aligned closely with expectations, as evaluated using an arbitrary accuracy score indicator. Notably, the 30% test dataset was predicted with 100% accuracy. Personally, I believe that utilizing error estimation metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-Squared (R2) would have provided more robust evaluation metrics. However, upon visual comparison, the accuracy score appears satisfactory in this instance. This is because each predicted MPG value, derived from the number of cylinders, horsepower, weight, and acceleration, precisely matched the actual MPG values in the test data.