# Week 4

July 2, 2023

Load the data as a Pandas data frame and ensure that it imported correctly.

[233]:
```
import pandas as pd
```

[234]:
```
# Read Data
data = pd.read_csv('auto-mpg.csv')
data.head()
```

[234]:
```
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

Begin by prepping the data for modeling:

Remove the car name column.

[236]:
```
# Look at name of columns
data.columns
```

[236]:
```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model year', 'origin', 'car name'],
      dtype='object')
```

[237]:
```
# Remove column
data = data.drop(columns="car name")
```

[238]:
```
data.head()
```

[238]:

```
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70

   origin
0       1
1       1
2       1
3       1
4       1
```

The horsepower column values likely imported as a string data type. Figure out why and replace any strings with the column mean.

[239]:

```python
# Display data types
data.dtypes
```

[239]:

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
dtype: object
```

[241]:

```python
# Changes horsepower to numeric and converts errors to NaN
data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
```

[243]:

```python
# Removes any NaN
data = data.dropna(axis= 0, how='any')
```

[244]:

```python
data.dtypes
```

[244]:

```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model year        int64
origin            int64
dtype: object
```

Create dummy variables for the origin column.

[245]:
```python
# Creates dummies for origin column
dummy_origin = pd.get_dummies(data["origin"])
```

[246]:
```python
dummy_origin
```

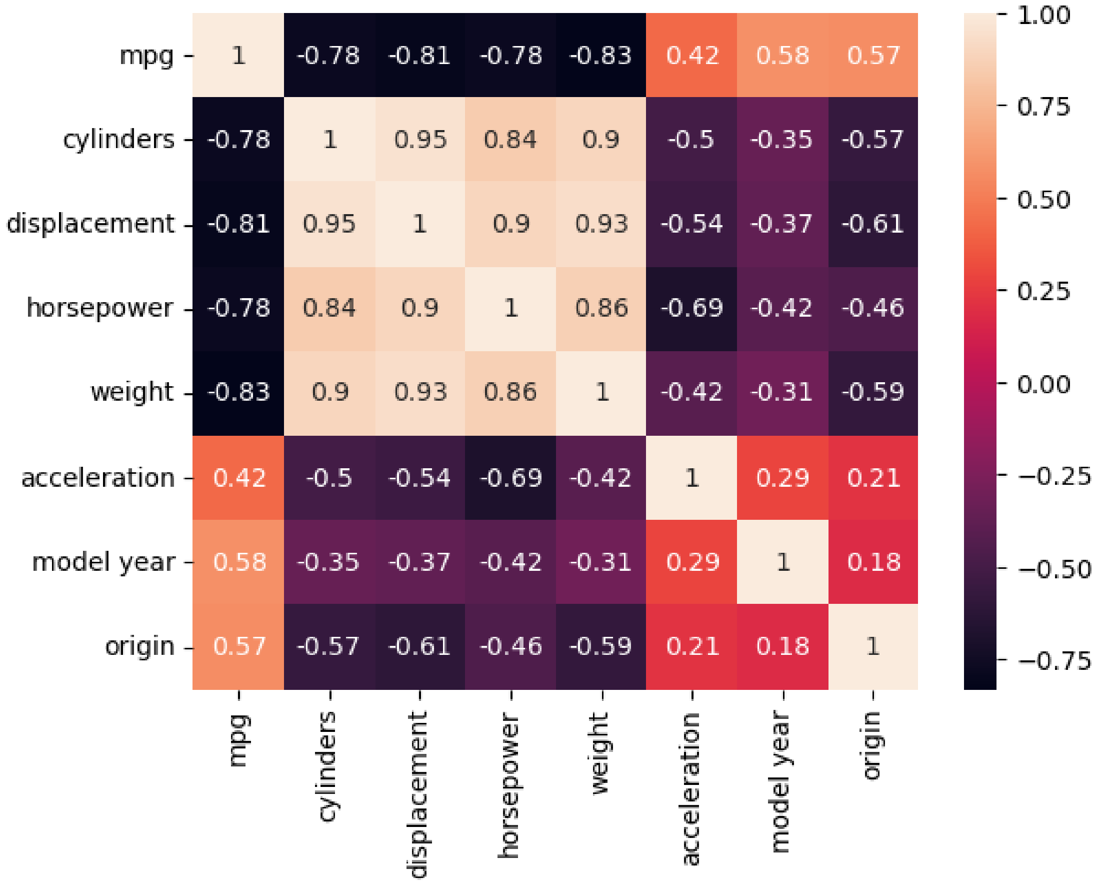
[246]:
```
     1  2  3
0    1  0  0
1    1  0  0
2    1  0  0
3    1  0  0
4    1  0  0
..  .. .. ..
393  1  0  0
394  0  1  0
395  1  0  0
396  1  0  0
397  1  0  0

[392 rows x 3 columns]
```

Create a correlation coefficient matrix and/or visualization. Are there features highly correlated with mpg?

[247]:
```python
# Creates correlation matrix
corr_matrix = data.corr()
```

[248]:
```python
import seaborn as sns
```

[249]:
```python
# Diplays correlation matrix
cm_plot = sns.heatmap(corr_matrix, annot=True)
```




The image above created from the correlation matrix of the Auto Data, shows us that MPG is highly correlated with the following features: Displacement and Weight.

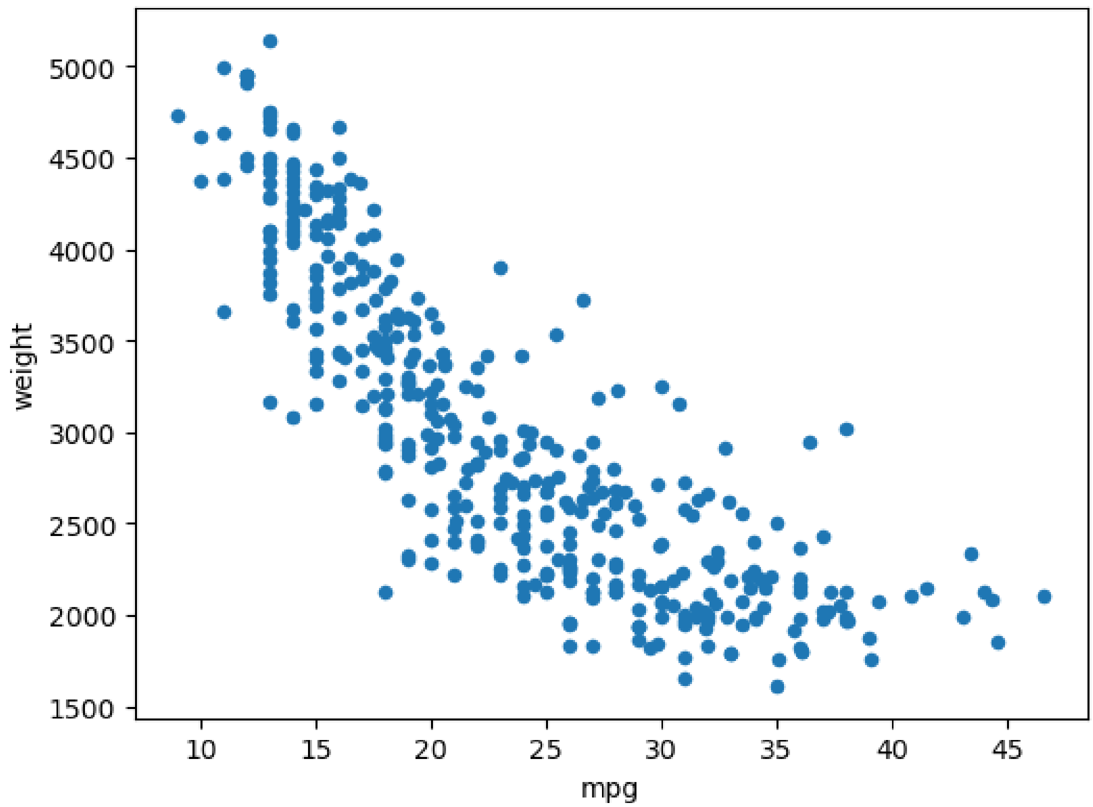
Plot mpg versus weight. Analyze this graph and explain how it relates to the corresponding correlation coefficient.

[250]:
```python
# Import Necessary Libraries
import matplotlib as plot
```

[251]:
```python
# Converts weight to numeric to avoid errors
data['weight'] = pd.to_numeric(data['weight'], errors='coerce')
```

[252]:
```python
# Creates scatter plot of age versus weight
data.plot.scatter(x = 'mpg', y = 'weight')
```

[252]: <AxesSubplot:xlabel='mpg', ylabel='weight'>




MPG versus weight has a correlation coefficient of - 0.83. Since this is so close to - 1, it indicates that there is a negative correlation between the two. As weight goes down, MPG goes up. The scatter plot shows this as the points are in a negative slope as MPG increases

Randomly split the data into 80% training data and 20% test data, where your target is mpg.

[253]:
```
from sklearn.model_selection import train_test_split
```

[254]:
```
# Separate the target from the features
feature = data.drop('mpg', axis=1)
target = data['mpg']

#Split the data into 80% training and 20% test
feature_train, feature_test, target_train, target_test =␣
↪train_test_split(feature, target, test_size=0.2, random_state=42)
```

Train an ordinary linear regression on the training data.

[255]:
```
# Import necessary Libraries
from sklearn.linear_model import LinearRegression
```

[256]:
```
# Create Linear Regression Model
regression = LinearRegression()
```

[257]:
```
# Drops any NaN to avoid errors
data = data.dropna(axis= 0, how='any')
```

[258]:
```
# Creates model
model = regression.fit(feature_train, target_train)
```

Calculate R2, RMSE, and MAE on both the training and test sets and interpret your results.

[259]:
```
# R2 for train data
regression.score(feature_train, target_train)
```

[259]:
```
0.826001578671067
```

[260]:
```
# R2 for test data
# R2 for train data
regression.score(feature_test, target_test)
```

[260]:
```
0.7901500386760352
```

[261]:
```
# Import necessary libraries
from sklearn.metrics import mean_squared_error
```

[262]:
```
model.fit(feature_test,target_test)
```

[262]:
```
LinearRegression()
```

[263]:
```
# Creates preditions of test features
y_pred_test = model.predict(feature_test)
```

[264]:
```
# RMSE of test
rmse_test = mean_squared_error(target_test, y_pred_test)**0.5
rmse_test
```

[264]:
```
2.9816057261316895
```

[265]:
```
model.fit(feature_train,target_train)
```

[265]:
```
LinearRegression()
```

[266]:
```
# Creates predictions of training features
y_pred_train = model.predict(feature_train)
```

[267]:
```
# RSEM of train
rmse_train = mean_squared_error(target_train, y_pred_train)**0.5
```

```
rmse_train
```

[267]: 3.3134960151437447

[268]:
```
# Import necessary Libraries
from sklearn.metrics import mean_absolute_error as mae
```

[269]:
```
# MAE of Test
print("MAE Test:", mae(target_test,y_pred_test))
```

```
MAE Test: 2.1864311671060714
```

[270]:
```
# MAE of Train
print("MAE Train:", mae(target_train,y_pred_train))
```

```
MAE Train: 2.548168196215135
```

Pick another regression model and repeat the previous two steps. Note: Do NOT choose logistic regression as it is more like a classification model.

[271]:
```
# Import necessary Libraries
from sklearn.linear_model import Ridge
```

[272]:
```
# Create Ridge Regression
ridgeReg = Ridge(alpha=10)

ridgeReg.fit(feature_train,target_train)
```

[272]: Ridge(alpha=10)

[273]:
```
# Runs ridge regression
train_score_ridge = ridgeReg.score(feature_train, target_train)
test_score_ridge = ridgeReg.score(feature_test, target_test)
```

[274]:
```
print("The train score for ridge model is {}".format(train_score_ridge))

print("The test score for ridge model is {}".format(test_score_ridge))
```

```
The train score for ridge model is 0.8258936780257804
The test score for ridge model is 0.7915211170468783
```