DSC550 Week 4

April 7, 2024

[30]:
```
import pandas as pd
file_path = '/Users/nickblackford/Desktop/Python/auto-mpg.csv'
df = pd.read_csv(file_path)
df.head()
```

[30]:
```
    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
```

[31]:
```
# remove car name
df = df.drop(columns=['car name'])
```

[32]:
```
# check horsepower type
print(df['horsepower'].dtype)

# horsepower is likely type 'string' because it includes 'NA values'
```

```
object
```

[33]:
```
# Convert horsepower to numeric
df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')

# Calculate the mean of the horsepower
mean_horsepower = df['horsepower'].mean()

# Replace NA values with mean
df['horsepower'].fillna(mean_horsepower, inplace=True)
```

[34]:
```python
# Create dummy variables for the 'origin' column
df = pd.get_dummies(df, columns=['origin'], prefix='origin')
```

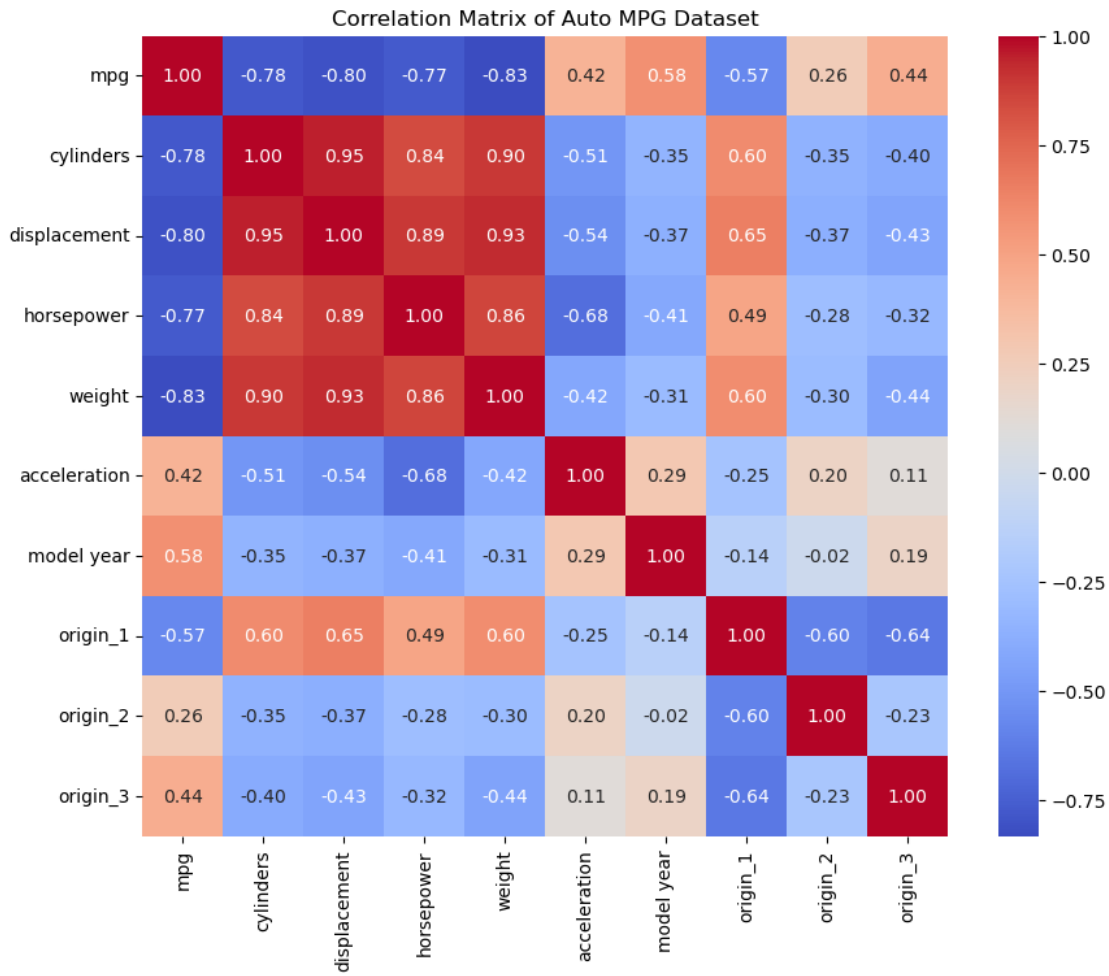
[35]:
```python
df.head()
```

[35]:
```
    mpg  cylinders  displacement  horsepower  weight  acceleration  \
0  18.0          8         307.0       130.0    3504          12.0
1  15.0          8         350.0       165.0    3693          11.5
2  18.0          8         318.0       150.0    3436          11.0
3  16.0          8         304.0       150.0    3433          12.0
4  17.0          8         302.0       140.0    3449          10.5

   model year  origin_1  origin_2  origin_3
0          70      True     False     False
1          70      True     False     False
2          70      True     False     False
3          70      True     False     False
4          70      True     False     False
```

[37]:
```python
# Calculate the correlation matrix
corr_matrix = df.corr()
corr_matrix
```

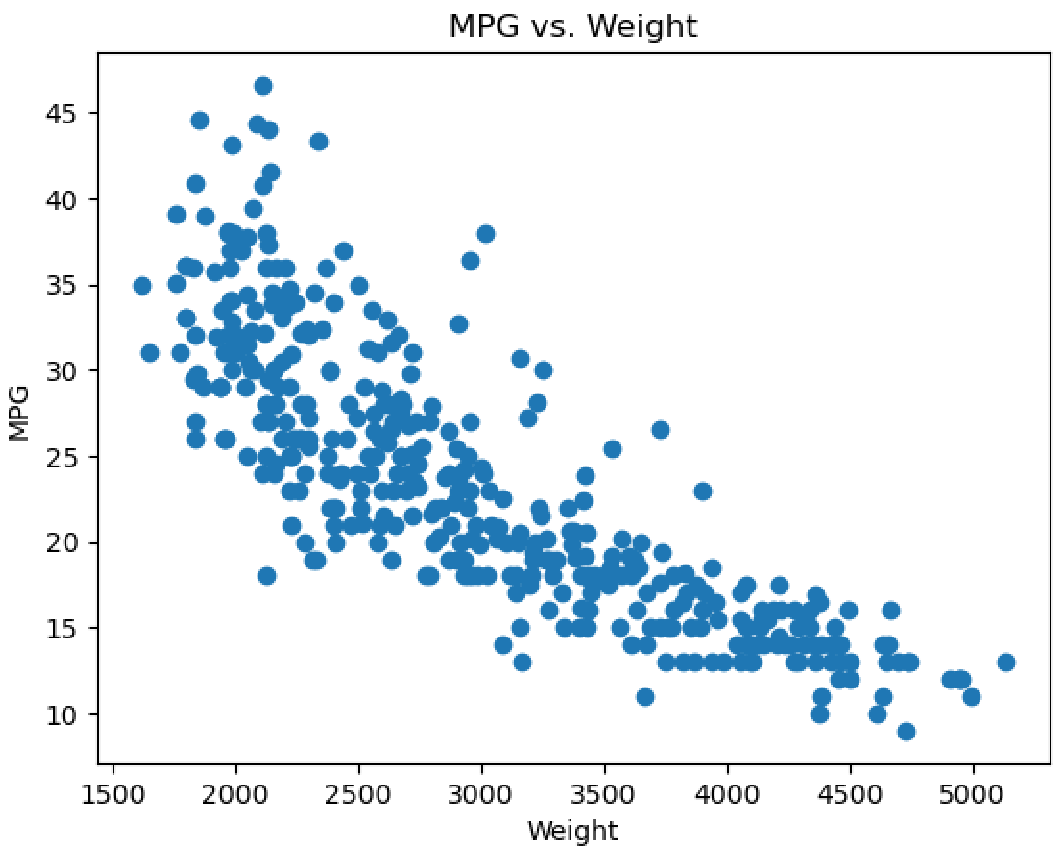
[37]:
```
                   mpg  cylinders  displacement  horsepower    weight  \
mpg           1.000000  -0.775396     -0.804203   -0.771437 -0.831741
cylinders    -0.775396   1.000000      0.950721    0.838939  0.896017
displacement -0.804203   0.950721      1.000000    0.893646  0.932824
horsepower   -0.771437   0.838939      0.893646    1.000000  0.860574
weight       -0.831741   0.896017      0.932824    0.860574  1.000000
acceleration  0.420289  -0.505419     -0.543684   -0.684259 -0.417457
model year    0.579267  -0.348746     -0.370164   -0.411651 -0.306564
origin_1     -0.568192   0.604351      0.651407    0.486083  0.598398
origin_2      0.259022  -0.352861     -0.373886   -0.281258 -0.298843
origin_3      0.442174  -0.396479     -0.433505   -0.321325 -0.440817

              acceleration  model year  origin_1  origin_2  origin_3
mpg               0.420289    0.579267 -0.568192  0.259022  0.442174
cylinders        -0.505419   -0.348746  0.604351 -0.352861 -0.396479
displacement     -0.543684   -0.370164  0.651407 -0.373886 -0.433505
horsepower       -0.684259   -0.411651  0.486083 -0.281258 -0.321325
weight           -0.417457   -0.306564  0.598398 -0.298843 -0.440817
acceleration      1.000000    0.288137 -0.250806  0.204473  0.109144
model year        0.288137    1.000000 -0.139883 -0.024489  0.193101
origin_1         -0.250806   -0.139883  1.000000 -0.597198 -0.643317
origin_2          0.204473   -0.024489 -0.597198  1.000000 -0.229895
origin_3          0.109144    0.193101 -0.643317 -0.229895  1.000000
```

[39]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Matrix of Auto MPG Dataset")
plt.show()
```



Based on the heatmap, cylinders, displacement, horsepower, and weight are all strongly, negatively correlated with mpg

[40]:
```python
# Plot mpg versus weight
plt.scatter(df['weight'], df['mpg'])
plt.xlabel('Weight')
plt.ylabel('MPG')
plt.title('MPG vs. Weight')
plt.show()
```



The scatter plot has a clear downward trend, representing the negative correlation between mpg and weight.

[44]:
```
from sklearn.model_selection import train_test_split

# Define the features and the target
X = df.drop(columns=['mpg'])
y = df['mpg']

# Split the data into 80% training data and 20% test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 ↪random_state=42)

# Check the size of the training and test sets
print(len(X_train))
print(len(X_test))
```

```
318
80
```

[45]:
```python
from sklearn.linear_model import LinearRegression

# Initialize the linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)
```

[45]: LinearRegression()

[46]:
```python
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import numpy as np

# Make predictions on the training set
y_train_pred = model.predict(X_train)

# Calculate R2, RMSE, and MAE for the training set
r2_train = r2_score(y_train, y_train_pred)
rmse_train = np.sqrt(mean_squared_error(y_train, y_train_pred))
mae_train = mean_absolute_error(y_train, y_train_pred)

# Make predictions on the test set
y_test_pred = model.predict(X_test)

# Calculate R2, RMSE, and MAE for the test set
r2_test = r2_score(y_test, y_test_pred)
rmse_test = np.sqrt(mean_squared_error(y_test, y_test_pred))
mae_test = mean_absolute_error(y_test, y_test_pred)

# Print the results
print("Training set metrics:")
print(f"R2: {r2_train:.2f}")
print(f"RMSE: {rmse_train:.2f}")
print(f"MAE: {mae_train:.2f}")

print("\nTest set metrics:")
print(f"R2: {r2_test:.2f}")
print(f"RMSE: {rmse_test:.2f}")
print(f"MAE: {mae_test:.2f}")
```

```
Training set metrics:
R2: 0.82
RMSE: 3.37
MAE: 2.61

Test set metrics:
R2: 0.84
```

```
RMSE: 2.89
MAE: 2.29
```

Interpretation:

R2: Our model explains 84% of the variablilty in mpg.

RMSE: The average error between the actual mpg and our model's predicted mpg is 2.89 mpg

MAE: The average magnitude between our model and actual mpg is 2.29 mpg.

[57]:
```python
# Try eslatic net regression to address multicollinearity concerns
from sklearn.linear_model import ElasticNet

elastic_net_model = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_net_model.fit(X_train, y_train)
```

[57]:
```
ElasticNet(alpha=0.1)
```

[58]:
```python
# Make predictions on the training set
y_train_pred_en = elastic_net_model.predict(X_train)

# Calculate R2, RMSE, and MAE for the training set
r2_train_en = r2_score(y_train, y_train_pred_en)
rmse_train_en = np.sqrt(mean_squared_error(y_train, y_train_pred_en))
mae_train_en = mean_absolute_error(y_train, y_train_pred_en)

# Make predictions on the test set
y_test_pred_en = elastic_net_model.predict(X_test)

# Calculate R2, RMSE, and MAE for the test set
r2_test_en = r2_score(y_test, y_test_pred_en)
rmse_test_en = np.sqrt(mean_squared_error(y_test, y_test_pred_en))
mae_test_en = mean_absolute_error(y_test, y_test_pred_en)

# Print the results
print("Elastic Net Regression Metrics:")
print("Training set:")
print(f"R2: {r2_train_en:.2f}")
print(f"RMSE: {rmse_train_en:.2f}")
print(f"MAE: {mae_train_en:.2f}")

print("\nTest set:")
print(f"R2: {r2_test_en:.2f}")
print(f"RMSE: {rmse_test_en:.2f}")
print(f"MAE: {mae_test_en:.2f}")
```

```
Elastic Net Regression Metrics:
Training set:
R2: 0.82
```

```
RMSE: 3.39
MAE: 2.61

Test set:
R2: 0.84
RMSE: 2.92
MAE: 2.32
```

Interpretation:

R2: Our model explains 84% of the variablilty in mpg.

RMSE: The average error between the actual mpg and our model's predicted mpg is 2.92 mpg

MAE: The average magnitude between our model and actual mpg is 2.32 mpg.

The eslatic net actually did slightly worse than our stand linear regression model.

[ ]:

```

```