Predicting Fuel Efficiency (Decision Tree Regression and Linear Regression)

In [139…

```python
# Import packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.tree import DecisionTreeRegressor
```

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

In [63]:

```python
# Load data into dataframe
# Link
link = '/Users/Malloryh5/Downloads/auto-mpg.csv'
# read csv file
data = pd.read_csv(link)
# Print head
data.head()
```

Out[63]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Prep the data for modeling

In [64]:

```python
# Drop unneeded columns
data = data.drop('car name', axis=1)
```

Find data type for all data. If strings found replace any strings with the column mean.

In [65]:

```python
# Review data types
data.dtypes
```

Out[65]:

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

In [66]:

```python
# Review horsepower data
data['horsepower'].unique()
```

Out[66]:

```
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',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

In [67]:

```python
# Replace any data points that is not a number with nan
data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
```

In [68]:

```python
# Replace any nan in horsepower with mean
data['horsepower'] = data['horsepower'].fillna(data['horsepower'].mean())
```

In [69]:

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

In [70]:

```python
data.head()
```

Out[70]:

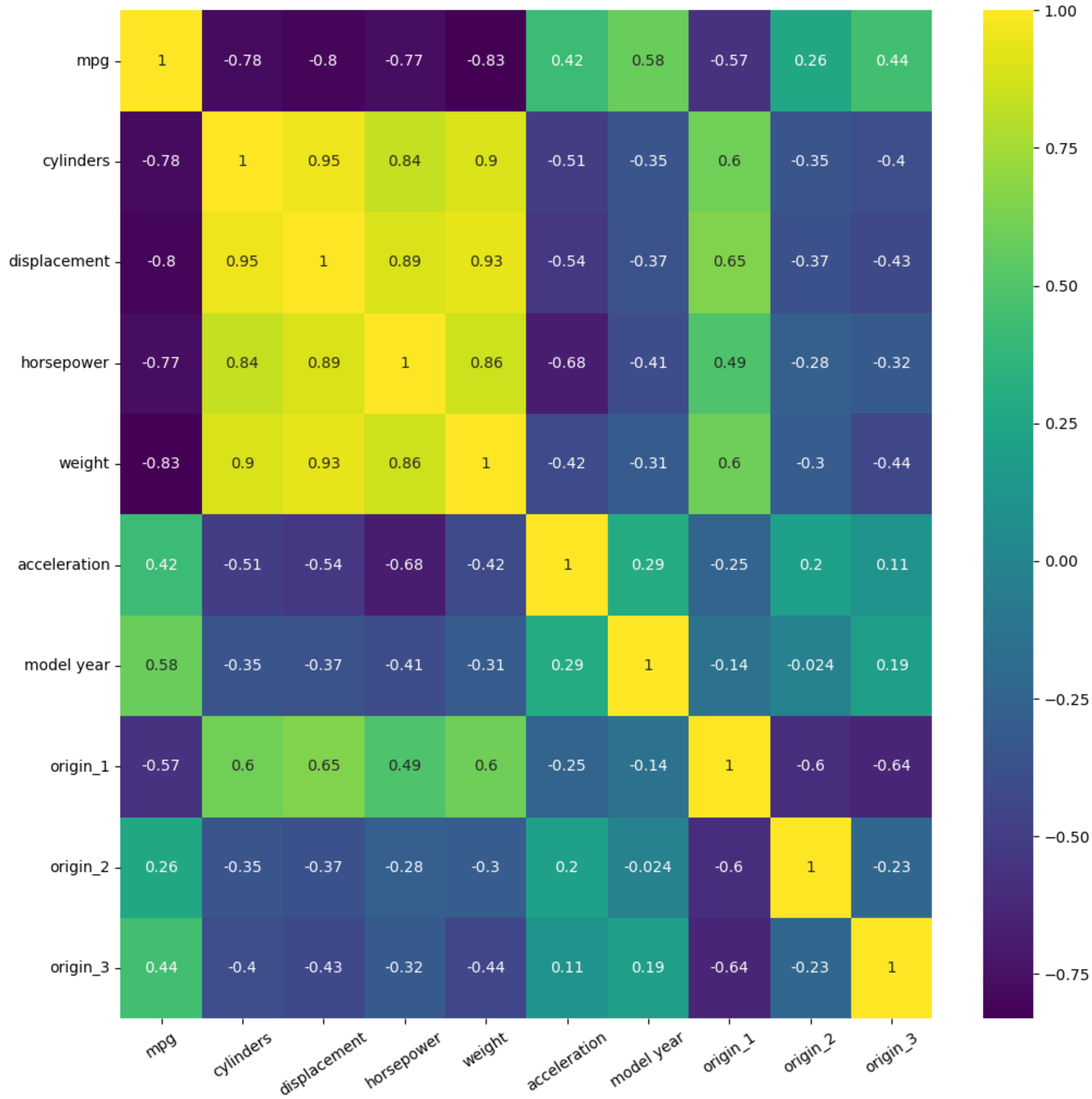
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin_1	origin_2	origin_3
0	18.0	8	307.0	130.0	3504	12.0	70	True	False	False
1	15.0	8	350.0	165.0	3693	11.5	70	True	False	False
2	18.0	8	318.0	150.0	3436	11.0	70	True	False	False
3	16.0	8	304.0	150.0	3433	12.0	70	True	False	False
4	17.0	8	302.0	140.0	3449	10.5	70	True	False	False

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

In [71]:

```python
# Make a correlation coefficient matrix
corr_matrix = data.corr()
```

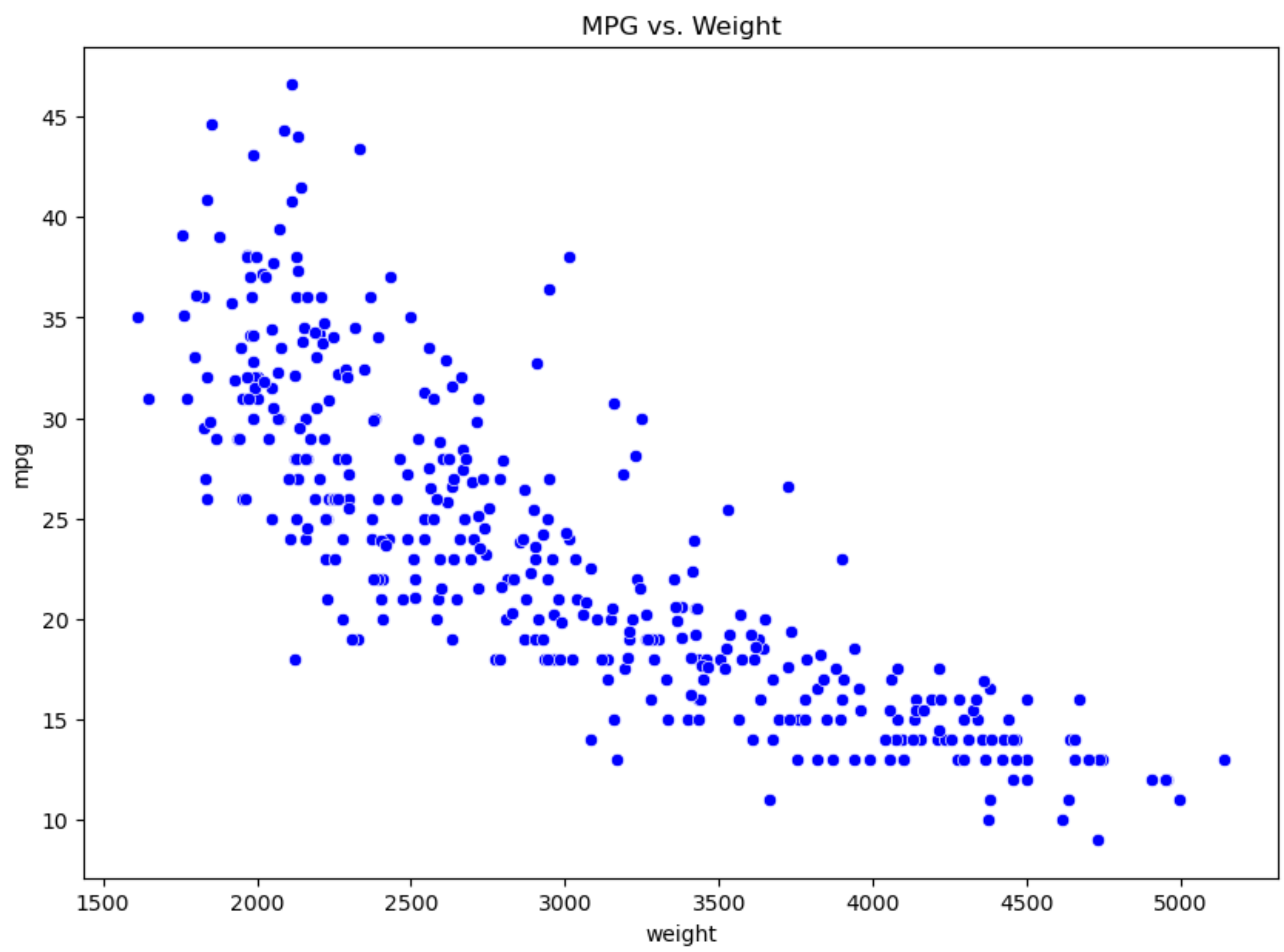
In [94]:
```python
# make a heat map to view the correlation matrix
plt.figure(figsize=(12,12))
# Make heatmap
sns.heatmap(corr_matrix, annot=True, cmap= 'viridis')
# Rotate x-axis labels
plt.xticks(rotation=34)
# Rotate y-axis labels
plt.yticks(rotation= 0)
# Show graph
plt.show()
```



	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin_1	origin_2	origin_3
mpg	1	-0.78	-0.8	-0.77	-0.83	0.42	0.58	-0.57	0.26	0.44
cylinders	-0.78	1	0.95	0.84	0.9	-0.51	-0.35	0.6	-0.35	-0.4
displacement	-0.8	0.95	1	0.89	0.93	-0.54	-0.37	0.65	-0.37	-0.43
horsepower	-0.77	0.84	0.89	1	0.86	-0.68	-0.41	0.49	-0.28	-0.32
weight	-0.83	0.9	0.93	0.86	1	-0.42	-0.31	0.6	-0.3	-0.44
acceleration	0.42	-0.51	-0.54	-0.68	-0.42	1	0.29	-0.25	0.2	0.11
model year	0.58	-0.35	-0.37	-0.41	-0.31	0.29	1	-0.14	-0.024	0.19
origin_1	-0.57	0.6	0.65	0.49	0.6	-0.25	-0.14	1	-0.6	-0.64
origin_2	0.26	-0.35	-0.37	-0.28	-0.3	0.2	-0.024	-0.6	1	-0.23
origin_3	0.44	-0.4	-0.43	-0.32	-0.44	0.11	0.19	-0.64	-0.23	1

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

In [103…
```python
# Make scatterplot of mpg vs weigt
## Figure size
plt.figure(figsize=(10,7))
## Scatter plot
sns.scatterplot(data=data, x='weight', y='mpg', color='blue')
## Title
plt.title('MPG vs. Weight')
## Show results
plt.show()
```



In [104…
```
# Analyze te relationship
print(corr_matrix['mpg']['weight'])
```

```
-0.8317409332443344
```

There is a negative relationship between the mpg and weight of a car. The more a car weighes the higher the gas milage.

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

In [113…
```
# Variable that revomes mpg
X = data.drop('mpg', axis = 1)
# Variable that is only mpg
y = data['mpg']
```

In [123…
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 0)
```

Train an ordinary linear regression on the training data.

In [144…
```
# Save linear regression as variable. Allow linear regression
lin_reg = LinearRegression()
# Fit the train data to reduce RSS
lin_reg.fit(X_train, y_train);
```

Out[144]:
```
▾ LinearRegression
LinearRegression()
```

In [126…
```
# Make linear regression prediction using train data
y_train_pred = lin_reg.predict(X_train)
# Make linear regression prediction using test data
y_test_pred = lin_reg.predict(X_test)
```

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

In [129…
```
# Test Matrix

## Calculate R^2 test
r2_test = r2_score(y_test, y_test_pred)
## Calculate RMSE test
rmse_test = mean_squared_error(y_test, y_test_pred, squared=False)
## Calculate MAE test
mae_test = mean_absolute_error(y_test, y_test_pred)
```

In [131…
```
# Train Matrix

## Calculate R^2 test
r2_train = r2_score(y_train, y_train_pred)
## Calculate RMSE test
rmse_train = mean_squared_error(y_train, y_train_pred, squared=False)
## Calculate MAE test
mae_train = mean_absolute_error(y_train, y_train_pred)
```

In [136…
```
# Print test and training results
print(f'MPG training set: R2={round(r2_train,3)}, RMSE={round(rmse_train,3)}, MAE={round(mae_train,3)}')
print(f'MPG test set: R2={round(r2_test,3)}, RMSE={round(rmse_test,3)}, MAE={round(mae_test,3)}')
```

```
MPG training set: R2=0.821, RMSE=3.282, MAE=2.48
MPG test set: R2=0.827, RMSE=3.315, MAE=2.7
```

The model is a good predictor. The R^2 is high, while the MAE and RMSE are low.

1. **R^2:**
    - Test: .821 or 82.1%
    - Train: .827 or 82.7%
    - 82% of the dependent variable can be explained by the independent variable. This means that the prediction matches 82% of the data.
2. **MAE:**
    - Test: 3.282
    - Train: 3.315
    - On average, the difference of the predictions sum is only 3.3 (test) and 3.2 (train).
3. **RMSE:**
    - Test: 2.48
    - Train: 2.7
    - The predictions differ from the values of only about 2.5.

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

Train an ordinary Decision Tree Regression on the training data.

In [145…

```
# Allow decision tree regressor
tree_regression = DecisionTreeRegressor()
# Fit to data
tree_regression.fit(X_train, y_train);
```

Out[145]:

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [148…

```
# Make decision tree regressor for test data
y_test_pred_2 = tree_regression.predict(X_test)
# Make decision tree regressor for train data
y_train_pred_2 = tree_regression.predict(X_train)
```

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

In [151…

```
## Calculate R^2 test
r2_test_2 = r2_score(y_test, y_test_pred_2)
## Calculate RMSE test
rmse_test_2 = mean_squared_error(y_test, y_test_pred_2)
## Calculate MAE test
mae_test_2 = mean_absolute_error(y_test, y_test_pred_2)
```

In [153…

```
## Calculate R^2 train
r2_train_2 = r2_score(y_train, y_train_pred_2)
## Calculate RMSE train
rmse_train_2 = mean_squared_error(y_train, y_train_pred_2)
## Calculate MAE train
mae_train_2 = mean_absolute_error(y_train, y_train_pred_2)
```

In [155…

```
# Print test and training results
print(f'MPG training set: R2={round(r2_train_2,3)}, RMSE={round(rmse_train_2,3)}, MAE={round(mae_train_2,3)}')
print(f'MPG test set: R2={round(r2_test_2,3)}, RMSE={round(rmse_test_2,3)}, MAE={round(mae_test_2,3)}')
```

```
MPG training set: R2=1.0, RMSE=0.0, MAE=0.0
MPG test set: R2=0.864, RMSE=8.611, MAE=2.146
```

The model is overfitted. The train data is higher then the test data. This is not a good regression model to use.

In [ ]: