HW 3

This assignment covers several aspects of Linear Regresstion. **DO NOT ERASE MARKDOWN CELLS AND INSTRUCTIONS IN YOUR HW submission**

- **Q** - QUESTION
- **A** - Where to input your answer

Instructions

Keep the following in mind for all notebooks you develop:

- Structure your notebook.
- Use headings with meaningful levels in Markdown cells, and explain the questions each piece of code is to answer or the reason it is there.
- Make sure your notebook can always be rerun from top to bottom (Kernel Tab -> Restart and Run All)
- Start working on this assignment as soon as possible. If you are a beginner in Python this might take a long time. One of the objectives of this assignment is to help you learn python and scikit-learn package.
- Follow README.md for homework submission instructions
- In this notebook we assume '../data/' location of all data files to be read and written

Related sklearn material and online tutorials

sklearn User Guide

sklearn data pre-processing

- train_test_split
- common_pittfalls
- train test split tutorial

sklearn multiple linear regression

- tutorial
- API documentation
- Linear Regression
- multiple linear regression tutorial

sklearn polynomial regression

- generate polynomial features
- polinomial regression tutorial

correlation

- correlation

Linear Regression

In jupyter notebook environment, commands starting with the symbol % are magic commands or magic functions. `%%timeit` is one of such function. It basically gives you the speed of execution of certain statement or blocks of codes.

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Q1 Read the `car_data.csv` data (we assume ../data/ location of all data files to be read and written) from **data** folder using pandas. Replace the ??? in the code cell below to accomplish this taks.

A1 Replace ??? with code in the code cell below

In [ ]:
```
# Replace ??? with code in the code cell below

# I saw no need to pass other parameters aside from the separator
df = pd.read_csv('../data/Car_data.csv', sep=',')
```

In [ ]:
```
# View head of the data to confirm the correctness of your answer
df.head()
```

Out[ ]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engi
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

5 rows × 26 columns

Data cleaning and manipulation

Q2 Here, you will practice the usage of common data cleaning and manipulation functions in 3 steps.

1. Use isnull() to figure out the number of NaN values per column
2. Remove the column with majority NaN values (if any)
3. Check if there are still NaN values in the dataframe using `isna()` method

A2 Replace ??? with code in the code cell below

In [ ]:
```
# There is no missing data here on this dataset :

# I add sum() in order to see a total of NaN values per column
df.isnull().sum()
```

Out[ ]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

In [ ]:
```
# If there were NaN values, I would use this code
# df.drop(columns=['column with NaN values'], inplace=True)

# Using isna() to check for NaN values, as it functions much like isnull()
df.isna().sum()
```

Out[ ]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

In [ ]:
```
# lets get some statistical information :

# Using describe() to collect some info about the dataframe
df.describe()
```

Out[ ]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	engin
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0(
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.9(
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.6
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.0(
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.0(
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.0(
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.0(
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.0(

Q3: In this task, out of all categorical columns, we focus only on the `fueltype` column processing in 2 steps.

1. Use label encoder from sklearn and convert the `fueltype` categorical values to numerical values.
2. Create a new dataframe that contains only the numerical columns.

A3 Replace ??? with code in the code cell below.

In [ ]:
```
# Label Encoding for 2-class columns:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

# use fit_transform() to fit and convert the fueltype category to numerical
df['fueltype'] = le.fit_transform(df['fueltype'])
```

In [ ]:
```
# Create new dataframe with selected columns

# Enclose the bracketed column with brackets in order to also copy the column name
df2 = df[['fueltype']].copy()
```

In [ ]:
```
df2.head()
```

Out[ ]:

	fueltype
0	1
1	1
2	1
3	1
4	1

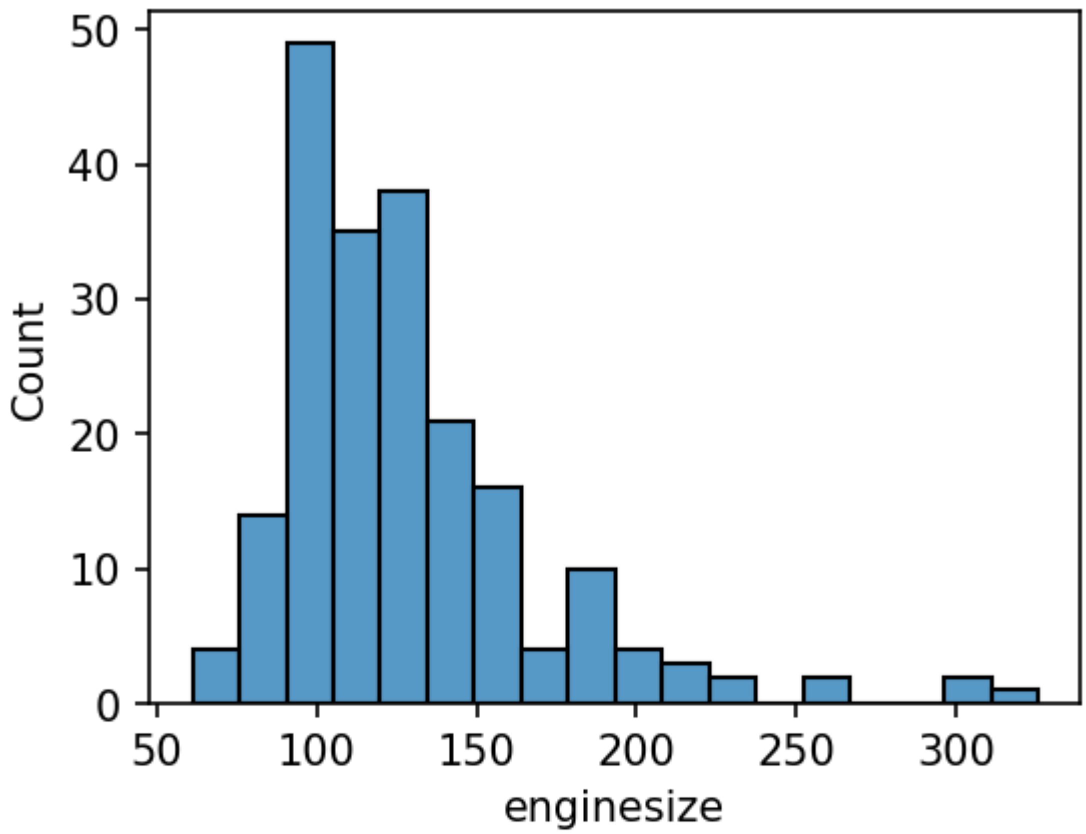
Q4: Use seaborn histplot to plot a distribution graph for the engine sizes

A4 Replace ??? with code in the code cell below

In [ ]:
```
plt.figure(figsize=(4,3),dpi=150)

# Specify column of focus for histogram
sns.histplot(data=df['enginesize'])
```

Out[ ]:
```
<Axes: xlabel='enginesize', ylabel='Count'>
```



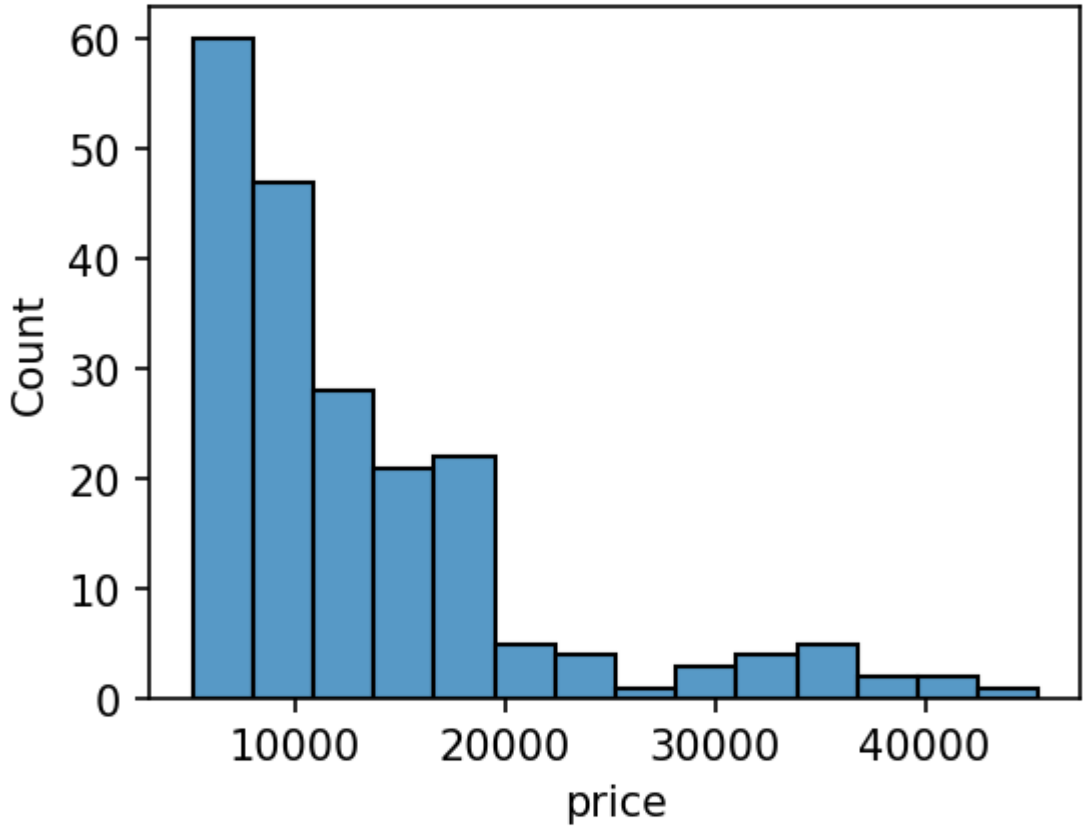
Q5: Use seaborn histplot to plot a distribution graph for the car prices

A5 Replace ??? with code in the code cell below

In [ ]:
```
plt.figure(figsize=(4,3),dpi=150)

# Specify column of focus for histogram
sns.histplot(data=df['price'])
```

Out[ ]:
```
<Axes: xlabel='price', ylabel='Count'>
```

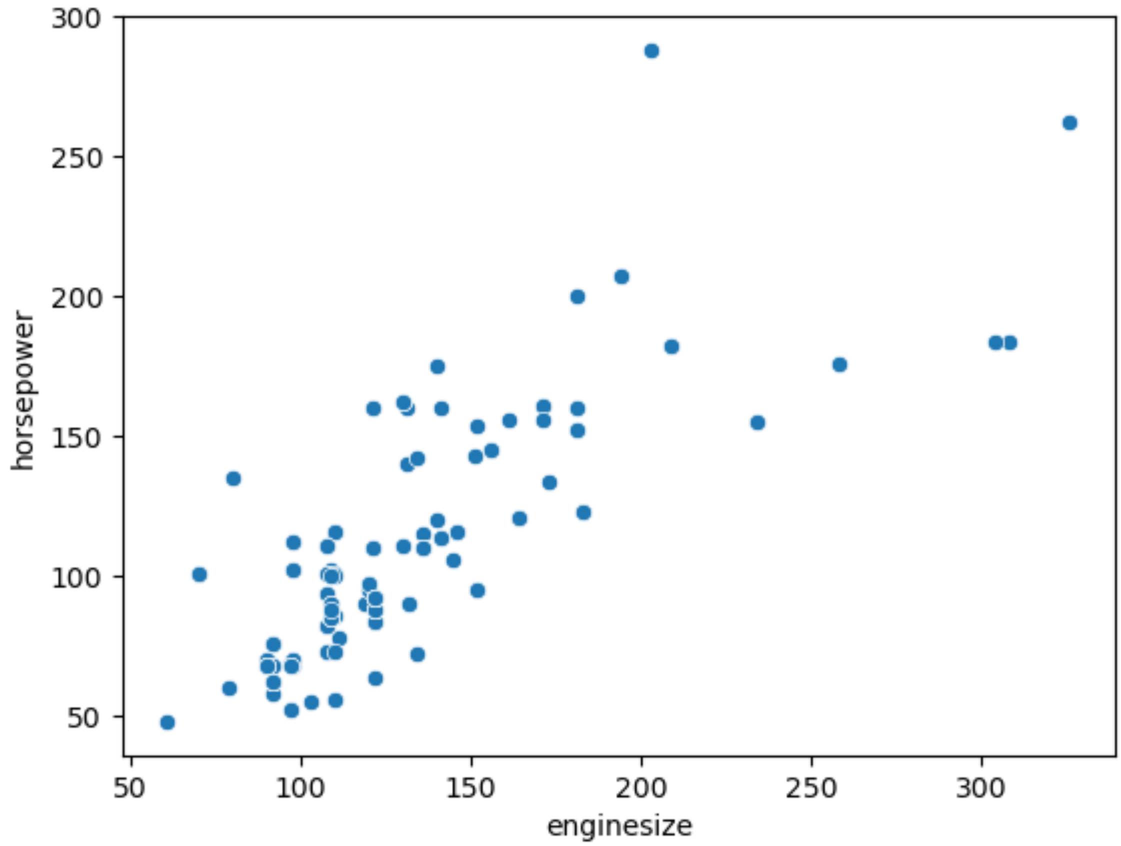


Q6: Use seaborn scatterplot to present the relation between enginesize and the horsepower of a car

A6 Replace ??? with code in the code cell below

In [ ]:
```
# Use whole dataframe but specify which two columns to compare in x & y parameters
sns.scatterplot(data=df, x='enginesize', y='horsepower')
```

Out[ ]:
```
<Axes: xlabel='enginesize', ylabel='horsepower'>
```



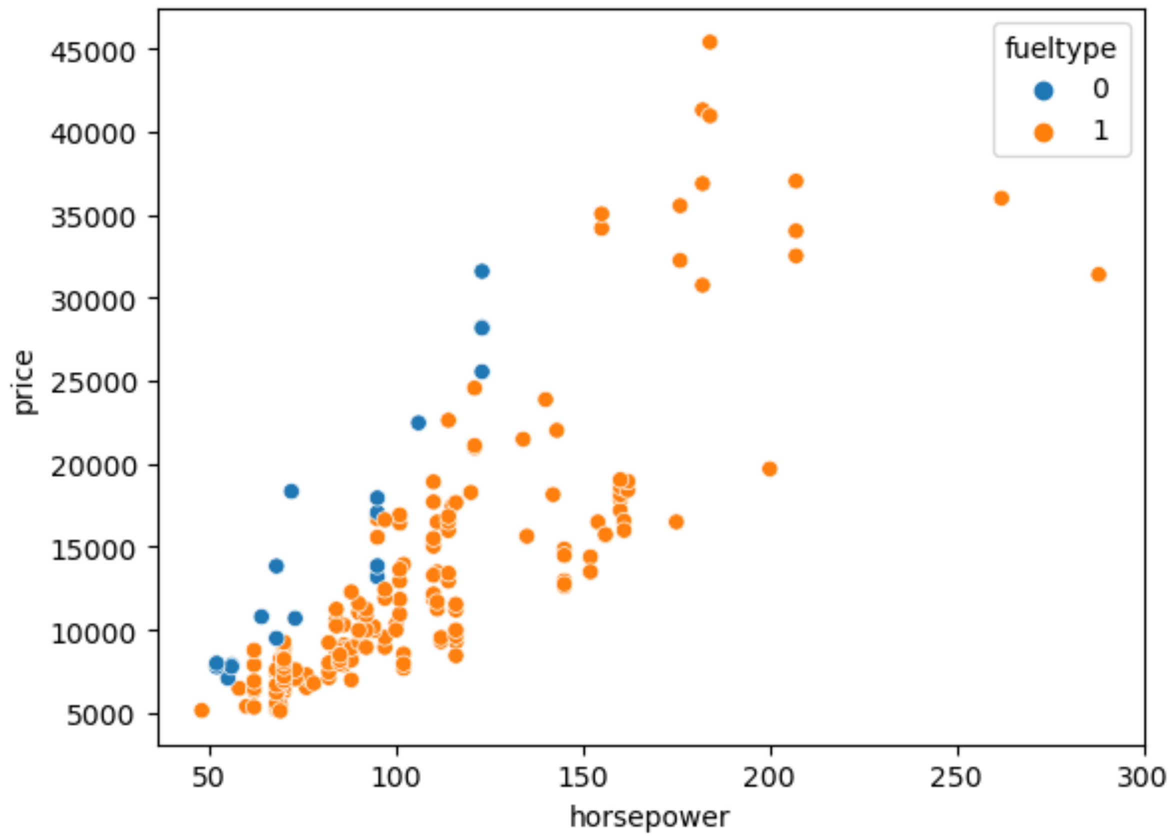
Q7: There is a correlation between the car price and the horsepower of a car. If horsepower of a car increase, the price of the car also increases most of the time, and in this question you will use the seaborn scatterplot to present the relation between price and horsepower.

Next, use `hue` parameter of scatterplot function to illustrate datapoints that relate to specific fueltype category.

A7 Replace ??? with code in the code cell below

In [ ]:
```
sns.scatterplot(data=df, x='horsepower', y='price', hue='fueltype')
```

Out[ ]:
```
<Axes: xlabel='horsepower', ylabel='price'>
```

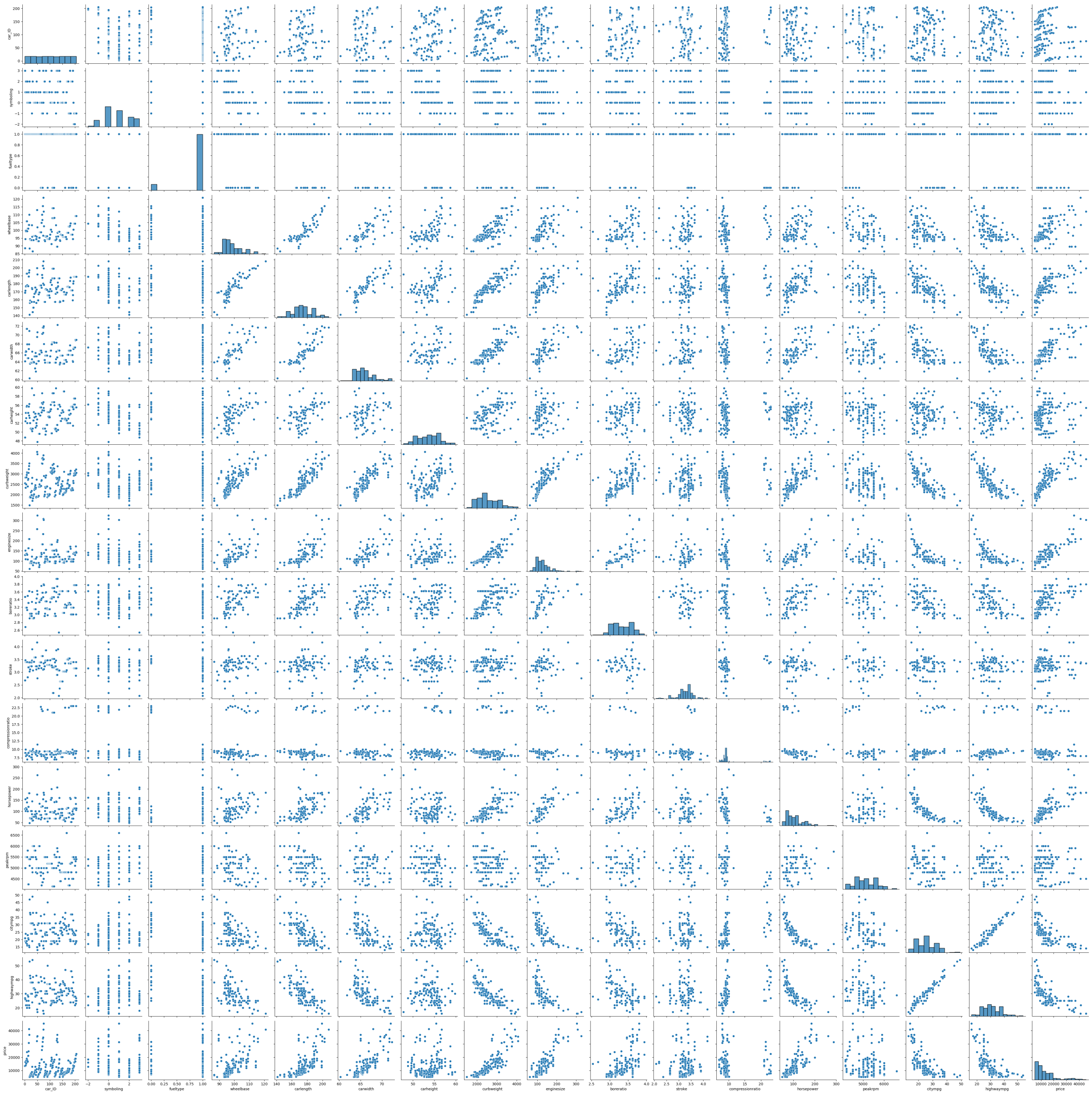


Q8: Use pairplot from sns to plot the data frame `df` and justify your feature selection.

A8: replace ??? with code in the code cell below.

In [ ]:
```
# 2. Use pairplot from sns to plot our data frame df
sns.pairplot(df)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x16a11dad550>
```



Q9 Data Visualization:

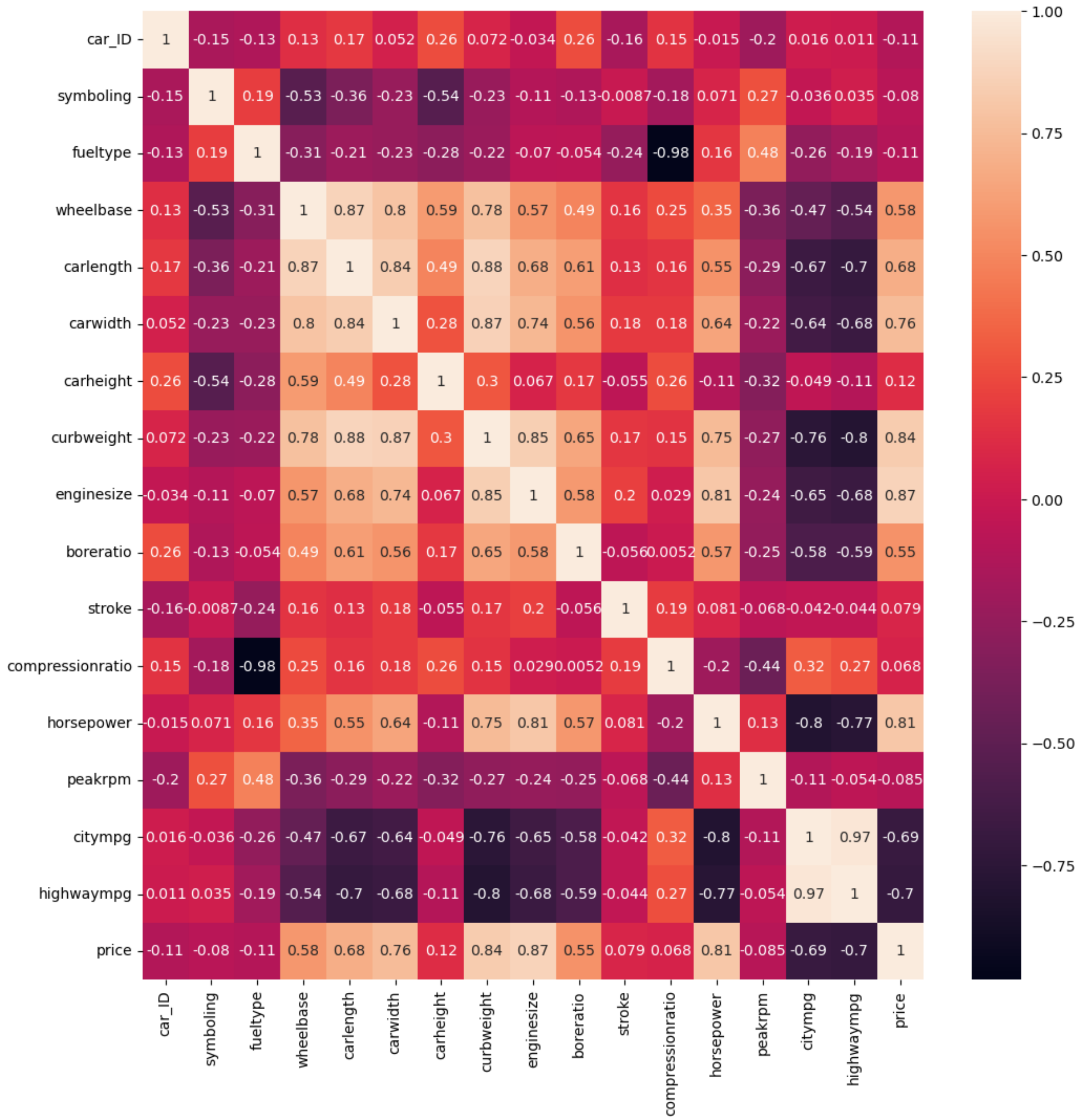
1. Use heatmap chart from seaborn library to findout the correlation between the columns in our dataset.
2. Update data frame 'df' to contain 5 columns from existing 'df' with the highest correlation to column "price". Also include price column in the updated data frame.

A9 Replace ??? with code in the code cell below

In [ ]:
```
corr_matrix = df.corr()
plt.figure(figsize=(12,12))
sns.heatmap(corr_matrix, annot=True)
```

```
C:\Users\jltx1\AppData\Local\Temp\ipykernel_15716\821394828.py:1: FutureWarning: Th
e default value of numeric_only in DataFrame.corr is deprecated. In a future versio
n, it will default to False. Select only valid columns or specify the value of nume
ric_only to silence this warning.
  corr_matrix = df.corr()
```

Out[ ]: <Axes: >



In [ ]:
```
# Task 2: Update data frame 'df' to contain 5 columns from existing 'df' with the h

# I chose the 5 columns with the highest absolute value (highest correlation to pri
df = df[['carwidth', 'curbweight', 'enginesize', 'horsepower', 'highwaympg', 'price
df.head()
```

Out[ ]:

	carwidth	curbweight	enginesize	horsepower	highwaympg	price
0	64.1	2548	130	111	27	13495.0
1	64.1	2548	130	111	27	16500.0
2	65.5	2823	152	154	26	16500.0
3	66.2	2337	109	102	30	13950.0
4	66.4	2824	136	115	22	17450.0

Data Preparation

Q10 Pre-processing

1. Assign 'price' column value to y and rest of the columns to x

A10 Replace ??? with code in the code cell below

In [ ]:
```
y = df.price.values
df.drop(columns=['price'], inplace=True)
X = df.values
```

```
C:\Users\jltx1\AppData\Local\Temp\ipykernel_15716\975429631.py:2: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/
user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(columns=['price'], inplace=True)
```

Q11 Use train_test_split to split the data set as train:test=(80%:20%) ratio.

A11 Replace ??? with code in the code cell below

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
# View the shape of your data set
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[ ]:
```
((164, 5), (41, 5), (164,), (41,))
```

Regression Task

Multiple Linear Regression

Q12 Fit multiple linear regression model on training data using all predictors, see (i) Linear Regression Example; (ii) scikit-learn linear model

$$Y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \ldots \beta_p * x_p$$

A12: Replace ??? with code in the code cell below

In [ ]:
```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

Out[ ]:
```
▾ LinearRegression

LinearRegression()
```

Q13: Model Scoring

1. Calculate the test MSE
2. Print the score from the model using test data

A13 Replace ??? with code in the code cell below

In [ ]:
```
# Calculate the score on train and test sets
# Your code goes below
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
y_pred = linear_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)   # Calculate the test MSE
print("Test mean squared error (MSE): {:.2f}".format(mse))

print(linear_model.score(X_test, y_test))
```

```
Test mean squared error (MSE): 18762701.22
0.805810951024779
```

Polinomial Regression

Q14: Polynomial extension of the feature set captures the non-linear dependencies in the data

1. Create a polinomial feature transformer with degree **TWO** using sklearn library PolynomialFeatures
2. Transform the training dataset using the polinomial feature transformer

A14 Replace ??? with code in the code cell below

In [ ]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
poly_features = poly.fit_transform(X_train)
```

Q15: Train the new model

1. Create a LinearRegression model using sklearn
2. Train the model using the transformed Train data(X_train)/ or Polinomial train data
3. Print the score for the Polinomial Regression for the Train data.

See (i) Linear Regression Example; (ii) Use the transformed X_train features inside the score() function for the correct model scores.

A15 Replace ??? with code in the code cell below

In [ ]:
```
poly_reg_model = LinearRegression()
poly_reg_model.fit(poly_features, y_train)

poly_reg_model.score(poly_features, y_train)
```

Out[ ]:
```
0.8495782236760439
```