HW 4

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.
- Please 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 (README.md) for homework submission instructions

Tutorials

- scikit-learn linear model (https://scikit-learn.org/stable/modules/linear_model.html)
- train-test-split (https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6)
- Multiple Linear Regression (https://www.investopedia.com/terms/m/mlr.asp#:~:text=Key%20Takeaways-,Multiple%20linear%20regression%20(MLR)%2C%20also%20known%20simply%20as%20multiple,uses%20just%20one%20explanatory%
- Polinomial Regression (https://towardsdatascience.com/polynomial-regression-bbe8b9d97491)
- Correlation (https://medium.com/analytics-vidhya/what-is-correlation-4fe0c6fbed47)

REGRESSION TASK USING SKLEARN

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 [135]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Q1 Read the `car_data.csv` data using pandas, and replace the ??? in the code cell below to

accomplish this taks.

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

In [136]:
```
# Replace ??? with code in the code cell below
```

In [137]:
```
# View head of the data to confirm the correctness of your answer
```

Out[137]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	en
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 Data cleaning and manipulation:

1. use isnull() to figure the number of NaN values per column
2. remove the column with 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 [138]:
```
# There is no missing data here on this dataset :
df.isnull()
```

Out[138]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
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
...	...	...	...	...	...	...	...	...

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd

In [139]:
```
# lets get some statistical information :
```

Out[139]:
```
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
```

Q3: In the dataset some of the columns are categorical, but we will only use the `fueltype` in our training.

1. Use label coder from sklearn and covert the categorical values to numerical values.
2. Now create a new dataframe with all the columns which holds only numeric values.

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

In [140]:
```
# Label Encoding for 2-class columns:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [141]:
```
# Create new dataframe with selected columns
df= df.select_dtypes(include = "number")
```

In [142]:

Out[142]:

	car_ID	symboling	fueltype	wheelbase	carlength	carwidth	carheight	curbweight	engine
0	1	3	1	88.6	168.8	64.1	48.8	2548	
1	2	3	1	88.6	168.8	64.1	48.8	2548	
2	3	1	1	94.5	171.2	65.5	52.4	2823	
3	4	2	1	99.8	176.6	66.2	54.3	2337	
4	5	2	1	99.4	176.6	66.4	54.3	2824	

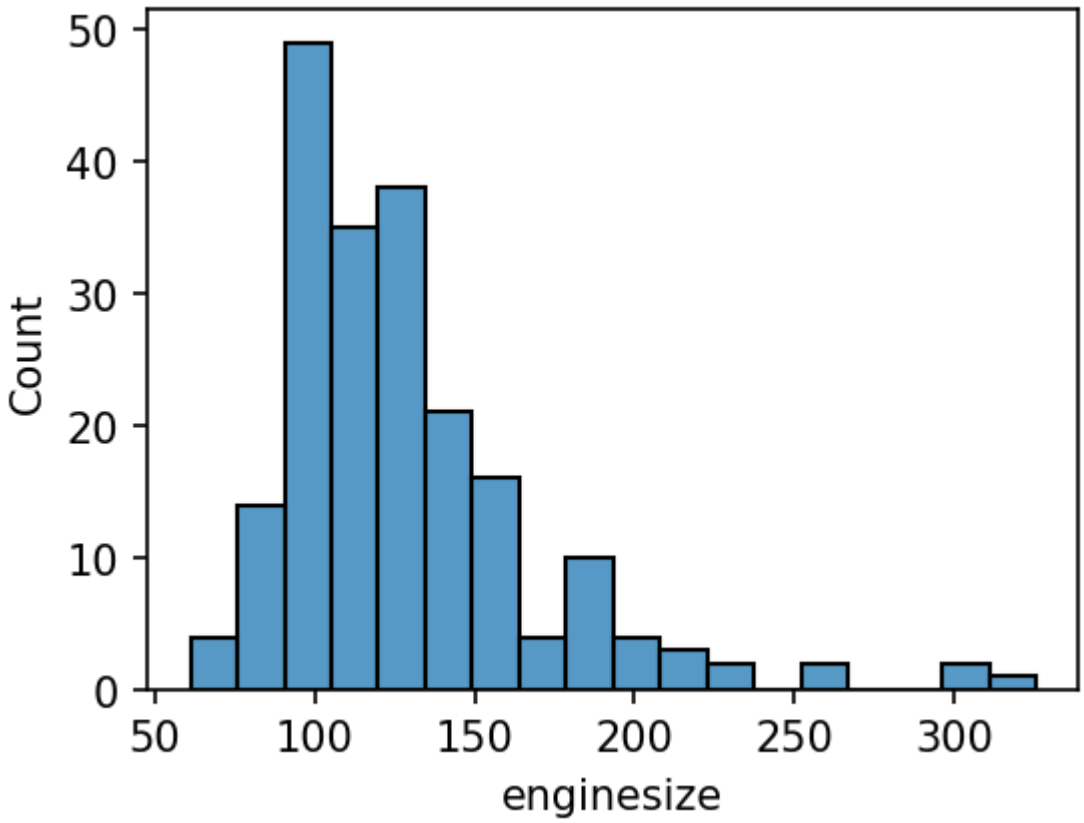
Q4: Use seaborn to plot a distribution graph for the engine sizes

Hint: use histplot from seaborn

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

In [123]:
```
plt.figure(figsize=(4,3),dpi=150)
```

Out[123]: <AxesSubplot:xlabel='enginesize', ylabel='Count'>

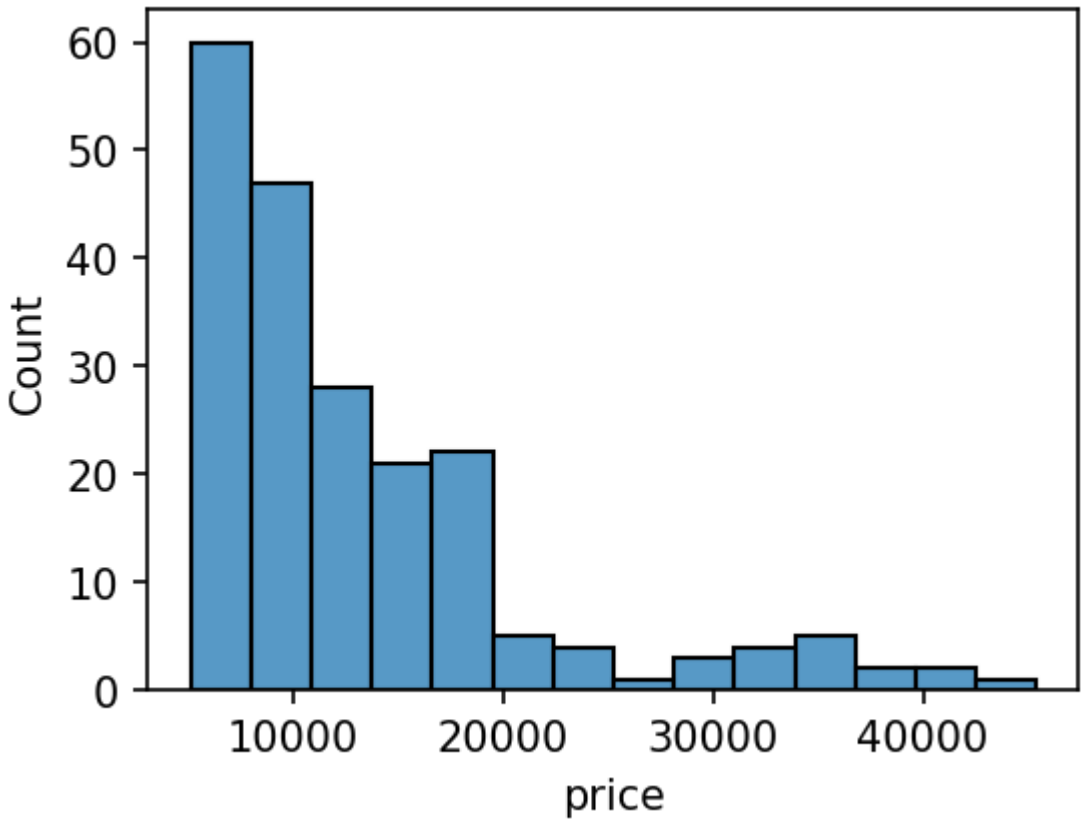


Q5: Use seaborn to plot a distribution graph for the car prices

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

In [85]:
```
plt.figure(figsize=(4,3),dpi=150)
```

Out[85]: <AxesSubplot:xlabel='price', ylabel='Count'>

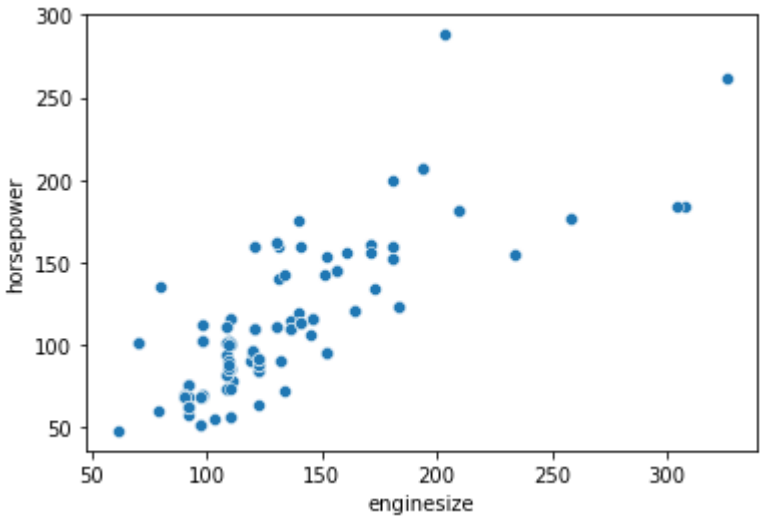


Q6: Do you think there is any relation between enginesize and the horsepower of a car? Use seaborn scatterplot to present the relation between them.

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

In [86]:

```
Out[86]: <AxesSubplot:xlabel='enginesize', ylabel='horsepower'>
```

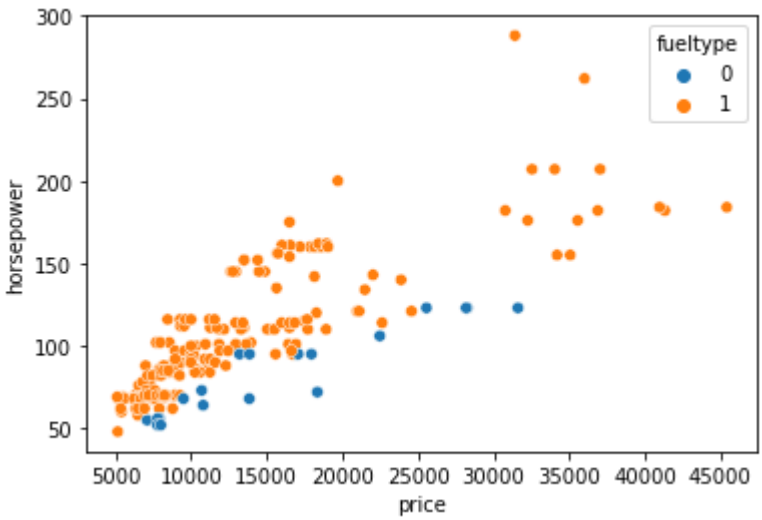


Q7: In Real-world there is a correlation between the car price and the horsepower of a car. If horsepower of a car increase, the price of the car also increases most of the type. Use seaborn scatterplot to present the relation between price and horsepower. Also try to show which dada point in the graph belows to which fueltype.
`Hint:` Use `hue` parameter of scatterplot for this.

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

In [87]:

```
Out[87]: <AxesSubplot:xlabel='price', ylabel='horsepower'>
```

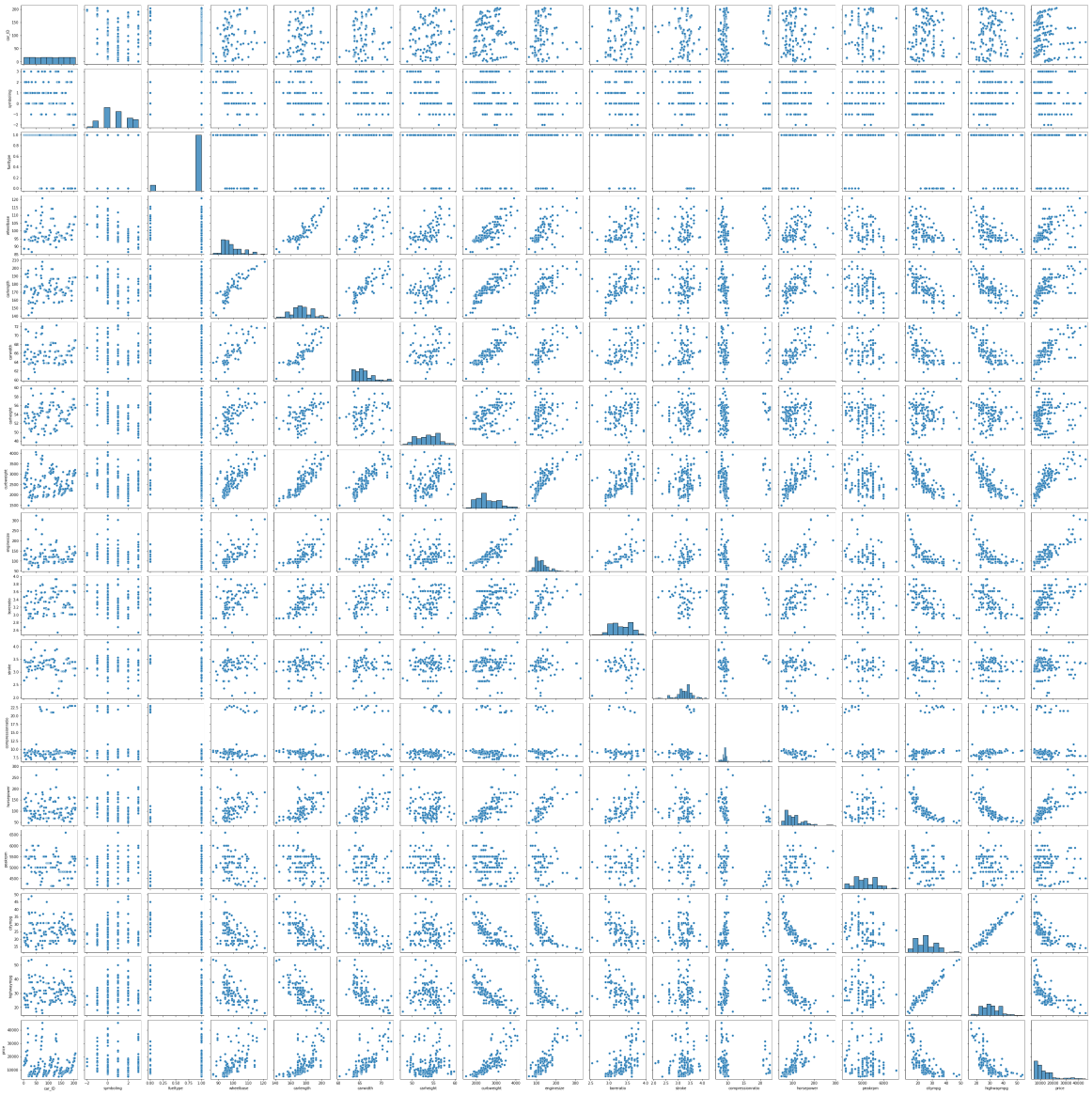


Q8: Use pairplot from sns to plot our data frame df for better understanding of your selection

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

In [66]:
```
# 2. Use pairplot from sns to plot our data frame df for better unders
```

Out[66]:
```
<seaborn.axisgrid.PairGrid at 0x1d6c3b7cfa0>
```



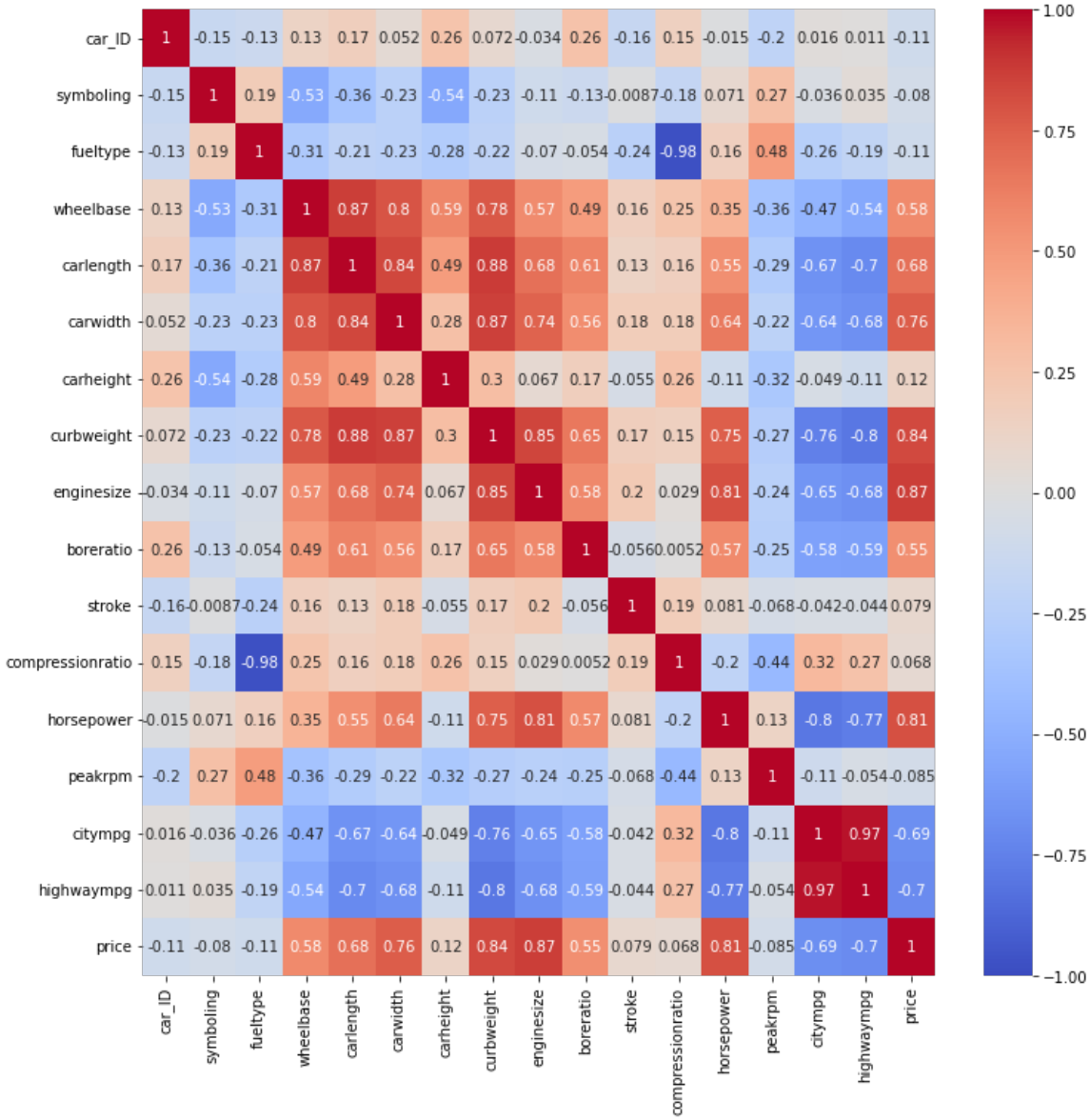
Q9 Data Visualization:

1. Use heatmap chart from seaborn library to findout the correlation between the columns in our dataset.
2. Choose a set of columns which are significantly related to our goal.

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

In [88]:
```
corr_matrix = df.corr()
plt.figure(figsize=(12,12))
```

Out[88]: <AxesSubplot:>



In [74]:
```
# Task 2: Choose columns of significant relations and create a new dat
#upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),k=1).ast
```

```
C:\Users\pedro\AppData\Local\Temp/ipykernel_1788/2127141390.py:2: Dep
recationWarning: `np.bool` is a deprecated alias for the builtin `boo
l`. To silence this warning, use `bool` by itself. Doing this will no
t modify any behavior and is safe. If you specifically wanted the num
py scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://nump
y.org/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.o
rg/devdocs/release/1.20.0-notes.html#deprecations)
  upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),k=1).a
stype(np.bool))
```

Data Preparation

Q10

1. Drop car_ID column
2. Assign price column value to y and rest columns to x

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

In [143]:
```
y = df.price.values
df.drop(columns = ['car_ID','price'], inplace = True)
x = df.values
```

Out[143]:
```
array([[ 3.000e+00,  1.000e+00,  8.860e+01, ...,  5.000e+03,  2.100e+
01,
         2.700e+01],
       [ 3.000e+00,  1.000e+00,  8.860e+01, ...,  5.000e+03,  2.100e+
01,
         2.700e+01],
       [ 1.000e+00,  1.000e+00,  9.450e+01, ...,  5.000e+03,  1.900e+
01,
         2.600e+01],
       ...,
       [-1.000e+00,  1.000e+00,  1.091e+02, ...,  5.500e+03,  1.800e+
01,
         2.300e+01],
       [-1.000e+00,  0.000e+00,  1.091e+02, ...,  4.800e+03,  2.600e+
01,
         2.700e+01],
       [-1.000e+00,  1.000e+00,  1.091e+02, ...,  5.400e+03,  1.900e+
01,
         2.500e+01]])
```

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 [144]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x,y, test_size = 0
# View the shape of your data set
```

Out[144]:
```
((164, 15), (41, 15), (164,), (41,))
```

Regression Task

Multiple Linear Regression

Q12 Fit multiple linear regression model on training data using all predictors. *Hints:* (i) Linear Regression Example (https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py); (ii) scikit-learn linear model (https://scikit-learn.org/stable/modules/linear_model.html)

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

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

In [145]:
```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
```

Out[145]:
```
LinearRegression()
```

Q13:

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 [146]:
```
# 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))
```

```
Test mean squared error (MSE): 11903274.60
0.7833819769679776
```

Polinomial Regression

Q14:

1. Create a polinomial feature transformer with degree **TWO**. _Hint:_use sklearn library PolynomialFeatures (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html)
2. Transform the training dataset using the polinomial feature transformer

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

In [147]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree = 2, include_bias = False)
```

Q15:

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.

Hints: (i) Linear Regression Example (https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py); (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 [148]:
```
poly_reg_model = linear_model
poly_reg_model.fit(poly_features, y_train)
```

Out[148]: 0.9966623059665934