HW2

October 5, 2025

1 HW 2

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

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

1.1 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 the sci-kit package. * Follow `README.md` for homework submission instructions * In this notebook, we assume '../data/' location of all data files to be read and written

1.2 Related sklearn material and online tutorials

sklearn User Guide

1.2.1 sklearn data pre-processing

- train_test_split
- common_pittfalls
- train test split tutorial

1.2.2 sklearn multiple linear regression

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

1.2.3 sklearn polynomial regression

- generate polynomial features
- polinomial regression tutorial

1.2.4 correlation

- correlation

2 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.

[1]:
```
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

[7]:
```
# Replace ??? with code in the code cell below

df = pd.read_csv('../data/Car_data.csv')
```

[12]:
```
# View head of the data to confirm the correctness of your answer
df.head()
```

[12]:
```
   car_ID  symboling                   CarName fueltype aspiration doornumber  \
0       1          3        alfa-romero giulia      gas        std        two
1       2          3       alfa-romero stelvio      gas        std        two
2       3          1  alfa-romero Quadrifoglio      gas        std        two
3       4          2               audi 100 ls      gas        std       four
4       5          2                audi 100ls      gas        std       four

       carbody drivewheel enginelocation  wheelbase  …  enginesize  \
0  convertible        rwd          front       88.6  …         130
1  convertible        rwd          front       88.6  …         130
2    hatchback        rwd          front       94.5  …         152
3        sedan        fwd          front       99.8  …         109
4        sedan        4wd          front       99.4  …         136

  fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm citympg  \
0       mpfi       3.47    2.68              9.0        111     5000      21
1       mpfi       3.47    2.68              9.0        111     5000      21
2       mpfi       2.68    3.47              9.0        154     5000      19
3       mpfi       3.19    3.40             10.0        102     5500      24
4       mpfi       3.19    3.40              8.0        115     5500      18
```

```
   highwaympg    price
0          27  13495.0
1          27  16500.0
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

2.1 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),else make comment with # in front of the cell 3. Check if there are still NaN values in the dataframe using `isna()` method

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

[13]:
```
# Is there any missing data here on this dataset :
df.isnull()
```

[13]:
```
     car_ID  symboling  CarName  fueltype  aspiration  doornumber  carbody  \
0     False      False    False     False       False       False    False
1     False      False    False     False       False       False    False
2     False      False    False     False       False       False    False
3     False      False    False     False       False       False    False
4     False      False    False     False       False       False    False
..      …          …        …        …           …           …        …
200   False      False    False     False       False       False    False
201   False      False    False     False       False       False    False
202   False      False    False     False       False       False    False
203   False      False    False     False       False       False    False
204   False      False    False     False       False       False    False

     drivewheel  enginelocation  wheelbase  …  enginesize  fuelsystem  \
0         False           False      False  …       False       False
1         False           False      False  …       False       False
2         False           False      False  …       False       False
3         False           False      False  …       False       False
4         False           False      False  …       False       False
..          …               …        …   …          …           …
200       False           False      False  …       False       False
201       False           False      False  …       False       False
202       False           False      False  …       False       False
203       False           False      False  …       False       False
204       False           False      False  …       False       False

     boreratio  stroke  compressionratio  horsepower  peakrpm  citympg  \
```

```
0        False  False              False       False    False    False
1        False  False              False       False    False    False
2        False  False              False       False    False    False
3        False  False              False       False    False    False
4        False  False              False       False    False    False
..         …      …             …         …      …      …
200      False  False              False       False    False    False
201      False  False              False       False    False    False
202      False  False              False       False    False    False
203      False  False              False       False    False    False
204      False  False              False       False    False    False

     highwaympg  price
0         False  False
1         False  False
2         False  False
3         False  False
4         False  False
..          …      …
200       False  False
201       False  False
202       False  False
203       False  False
204       False  False

[205 rows x 26 columns]
```

[14]:
```python
#Remove the column with majority NaN values (if any)
df.dropna(axis=1, thresh=len(df)//2+1)
```

[14]:
```
     car_ID  symboling                   CarName fueltype aspiration  \
0         1          3        alfa-romero giulia      gas        std
1         2          3       alfa-romero stelvio      gas        std
2         3          1  alfa-romero Quadrifoglio      gas        std
3         4          2               audi 100 ls      gas        std
4         5          2                audi 100ls      gas        std
..      …        …                       …      …        …
200     201         -1           volvo 145e (sw)      gas        std
201     202         -1               volvo 144ea      gas      turbo
202     203         -1               volvo 244dl      gas        std
203     204         -1                 volvo 246   diesel      turbo
204     205         -1               volvo 264gl      gas      turbo

    doornumber      carbody drivewheel enginelocation  wheelbase  …  \
0          two  convertible        rwd          front       88.6  …
1          two  convertible        rwd          front       88.6  …
2          two    hatchback        rwd          front       94.5  …
```

```
3          four          sedan          fwd          front        99.8   …
4          four          sedan          4wd          front        99.4   …
..          …              …         …             …          …   …
200        four          sedan          rwd          front       109.1   …
201        four          sedan          rwd          front       109.1   …
202        four          sedan          rwd          front       109.1   …
203        four          sedan          rwd          front       109.1   …
204        four          sedan          rwd          front       109.1   …

     enginesize  fuelsystem  boreratio  stroke compressionratio horsepower  \
0           130        mpfi       3.47    2.68               9.0        111
1           130        mpfi       3.47    2.68               9.0        111
2           152        mpfi       2.68    3.47               9.0        154
3           109        mpfi       3.19    3.40              10.0        102
4           136        mpfi       3.19    3.40               8.0        115
..           …           …         …       …               …          …
200         141        mpfi       3.78    3.15               9.5        114
201         141        mpfi       3.78    3.15               8.7        160
202         173        mpfi       3.58    2.87               8.8        134
203         145         idi       3.01    3.40              23.0        106
204         141        mpfi       3.78    3.15               9.5        114

     peakrpm citympg  highwaympg    price
0       5000      21          27  13495.0
1       5000      21          27  16500.0
2       5000      19          26  16500.0
3       5500      24          30  13950.0
4       5500      18          22  17450.0
..        …       …           …       …
200     5400      23          28  16845.0
201     5300      19          25  19045.0
202     5500      18          23  21485.0
203     4800      26          27  22470.0
204     5400      19          25  22625.0

[205 rows x 26 columns]
```

[15]:
```python
# lets get some statistical information :
df.describe()
```

[15]:
```
           car_ID   symboling   wheelbase   carlength    carwidth   carheight  \
count  205.000000  205.000000  205.000000  205.000000  205.000000  205.000000
mean   103.000000    0.834146   98.756585  174.049268   65.907805   53.724878
std     59.322565    1.245307    6.021776   12.337289    2.145204    2.443522
min      1.000000   -2.000000   86.600000  141.100000   60.300000   47.800000
25%     52.000000    0.000000   94.500000  166.300000   64.100000   52.000000
50%    103.000000    1.000000   97.000000  173.200000   65.500000   54.100000
```

```
75%     154.000000    2.000000  102.400000  183.100000   66.900000   55.500000
max     205.000000    3.000000  120.900000  208.100000   72.300000   59.800000

        curbweight  enginesize   boreratio      stroke  compressionratio  \
count   205.000000  205.000000  205.000000  205.000000        205.000000
mean   2555.565854  126.907317    3.329756    3.255415         10.142537
std     520.680204   41.642693    0.270844    0.313597          3.972040
min    1488.000000   61.000000    2.540000    2.070000          7.000000
25%    2145.000000   97.000000    3.150000    3.110000          8.600000
50%    2414.000000  120.000000    3.310000    3.290000          9.000000
75%    2935.000000  141.000000    3.580000    3.410000          9.400000
max    4066.000000  326.000000    3.940000    4.170000         23.000000

       horsepower      peakrpm     citympg  highwaympg         price
count  205.000000   205.000000  205.000000  205.000000    205.000000
mean   104.117073  5125.121951   25.219512   30.751220  13276.710571
std     39.544167   476.985643    6.542142    6.886443   7988.852332
min     48.000000  4150.000000   13.000000   16.000000   5118.000000
25%     70.000000  4800.000000   19.000000   25.000000   7788.000000
50%     95.000000  5200.000000   24.000000   30.000000  10295.000000
75%    116.000000  5500.000000   30.000000   34.000000  16503.000000
max    288.000000  6600.000000   49.000000   54.000000  45400.000000
```

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.

[17]:
```python
# Label Encoding for 2-class columns:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['fueltype'] = le.fit_transform(df['fueltype'])
```

[18]:
```python
# Create new dataframe with selected columns. Careful, don't rename ex:
↪df_numeric
df=df.select_dtypes(include='number')
```

[19]:
```python
df.head()
```

[19]:
```
   car_ID  symboling  fueltype  wheelbase  carlength  carwidth  carheight  \
0       1          3         1       88.6      168.8      64.1       48.8
1       2          3         1       88.6      168.8      64.1       48.8
2       3          1         1       94.5      171.2      65.5       52.4
3       4          2         1       99.8      176.6      66.2       54.3
4       5          2         1       99.4      176.6      66.4       54.3
```

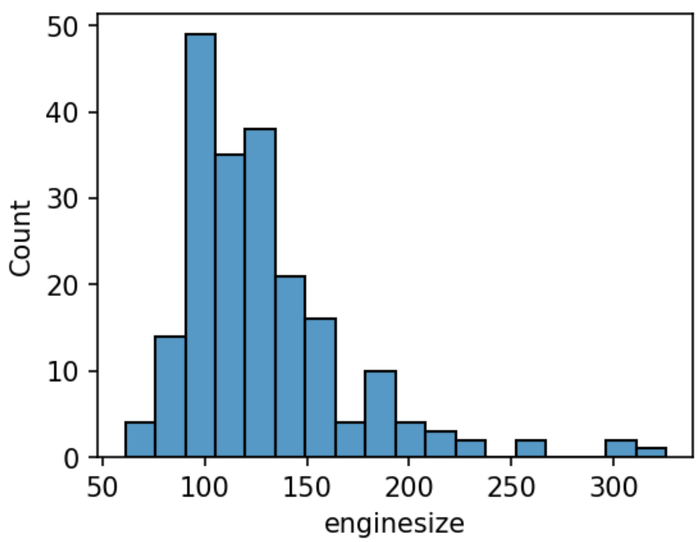
```
   curbweight  enginesize  boreratio  stroke  compressionratio  horsepower  \
0        2548         130       3.47    2.68               9.0         111
1        2548         130       3.47    2.68               9.0         111
2        2823         152       2.68    3.47               9.0         154
3        2337         109       3.19    3.40              10.0         102
4        2824         136       3.19    3.40               8.0         115

   peakrpm  citympg  highwaympg    price
0     5000       21          27  13495.0
1     5000       21          27  16500.0
2     5000       19          26  16500.0
3     5500       24          30  13950.0
4     5500       18          22  17450.0
```

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

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

[21]:
```
plt.figure(figsize=(4,3),dpi=150)
sns.histplot(df['enginesize'])
```

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

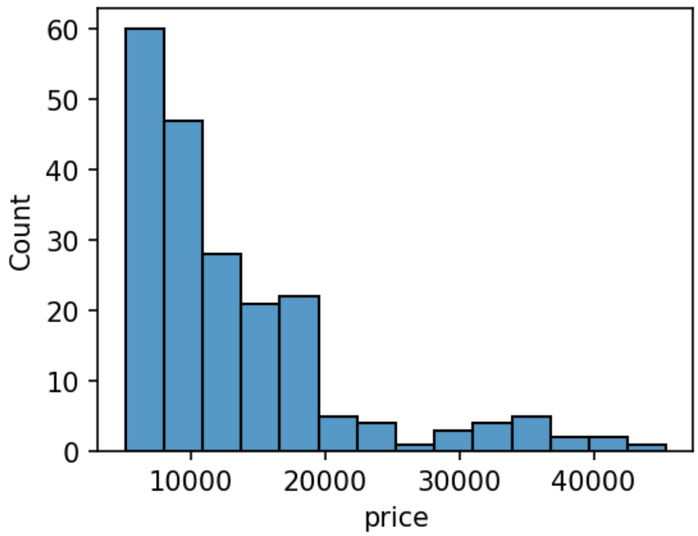


Q5: Use seaborn histplot to plot a distribution graph for the car prices

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

[24]:
```
plt.figure(figsize=(4,3),dpi=150)
sns.histplot(df['price'])
```

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

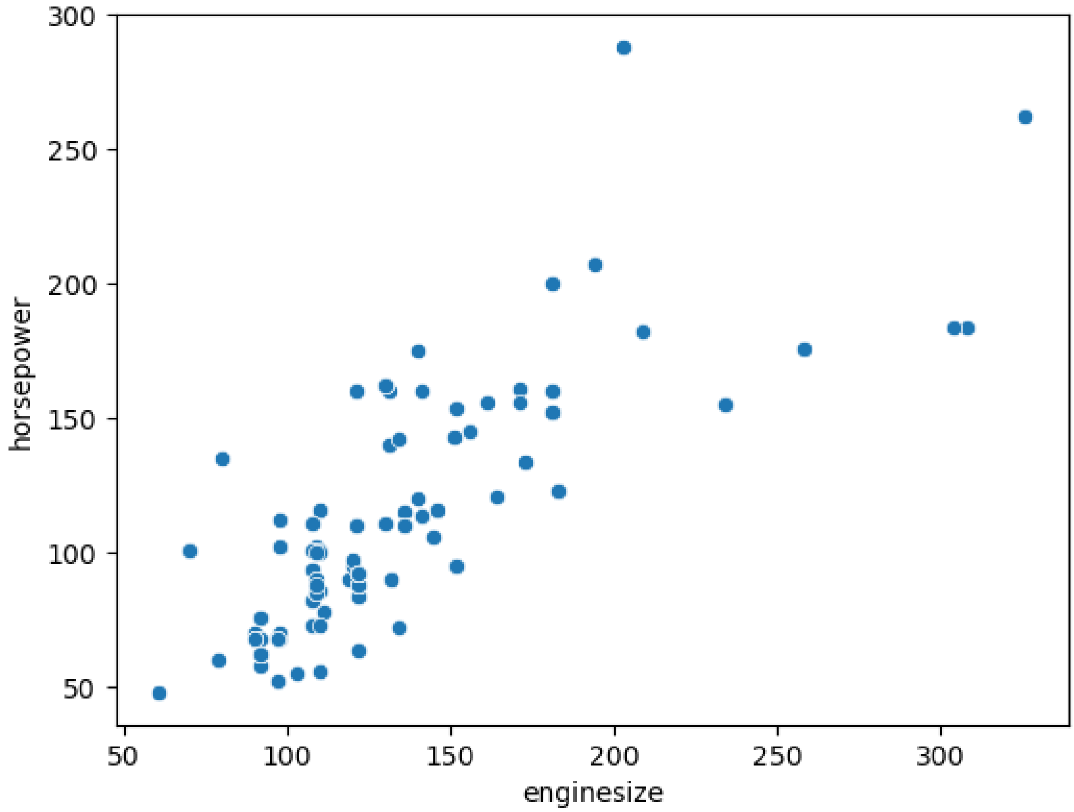


Q6: Use seaborn scatterplot to present the relation between enginesize and the horsepower of a car

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

[25]:
```
sns.scatterplot(x='enginesize', y='horsepower', data=df)
```

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



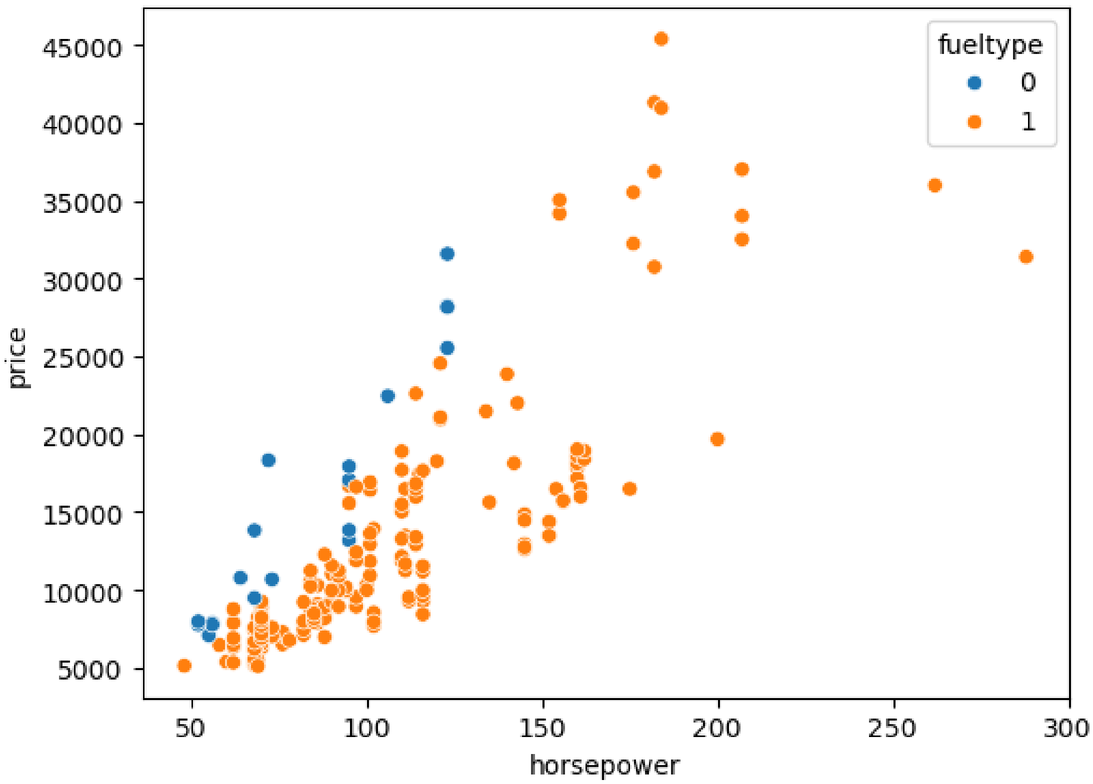
Q7: There is a correlation between the car price and the horsepower of a car. If horsepower of a car increase, the price of the car also increases most of the time, and in this question you will use the seaborn scatterplot to present the relation between price and horsepower.

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

[26]:
```
sns.scatterplot(x='horsepower', y='price', hue='fueltype', data=df)
```

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

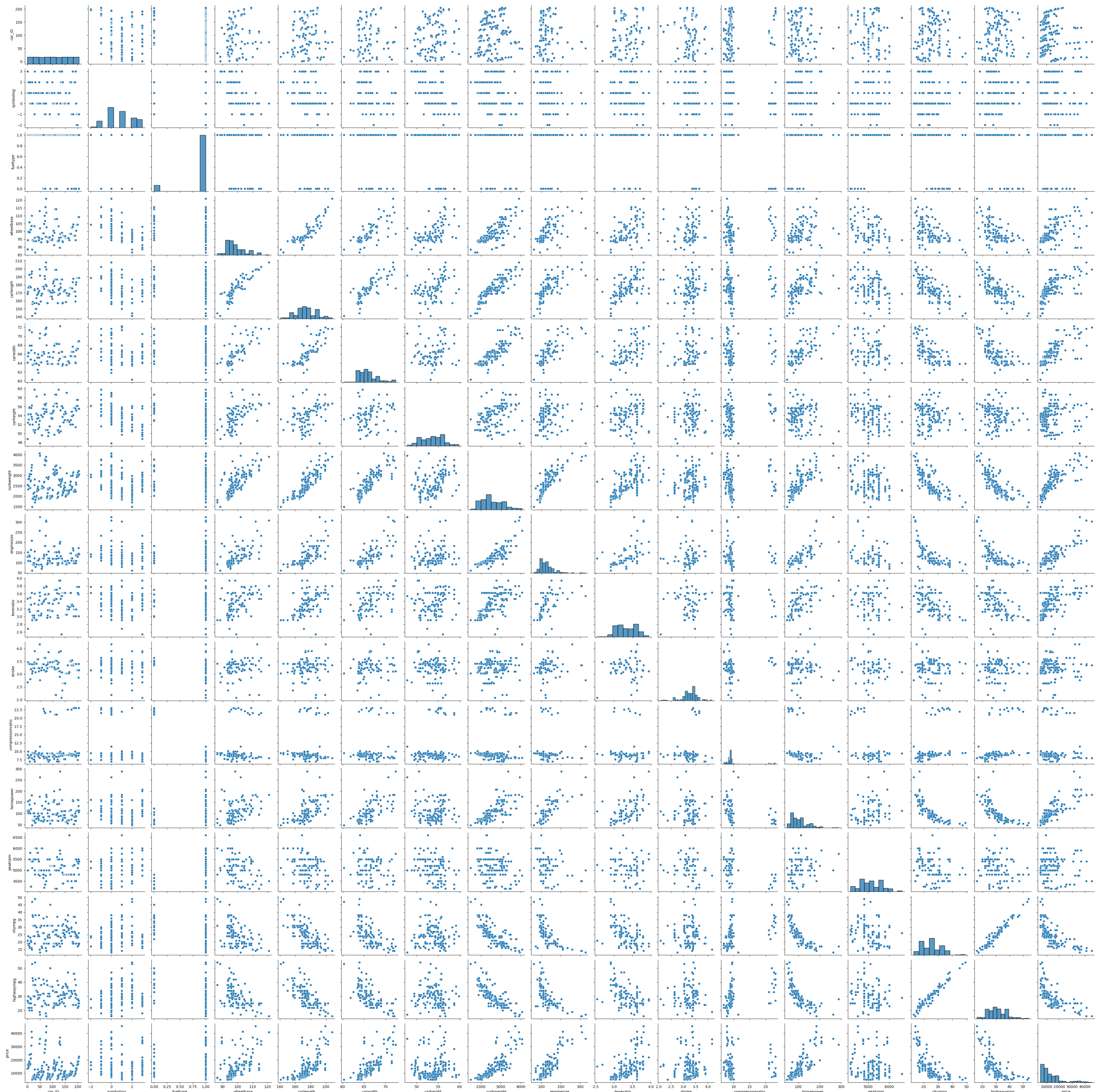


Q8: Use pairplot from sns to plot the data frame `df` and justify your feature selection.

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

[27]:
```python
# 2. Use pairplot from sns to plot our data frame df
sns.pairplot(df)
```

[27]: <seaborn.axisgrid.PairGrid at 0x32c23d400>



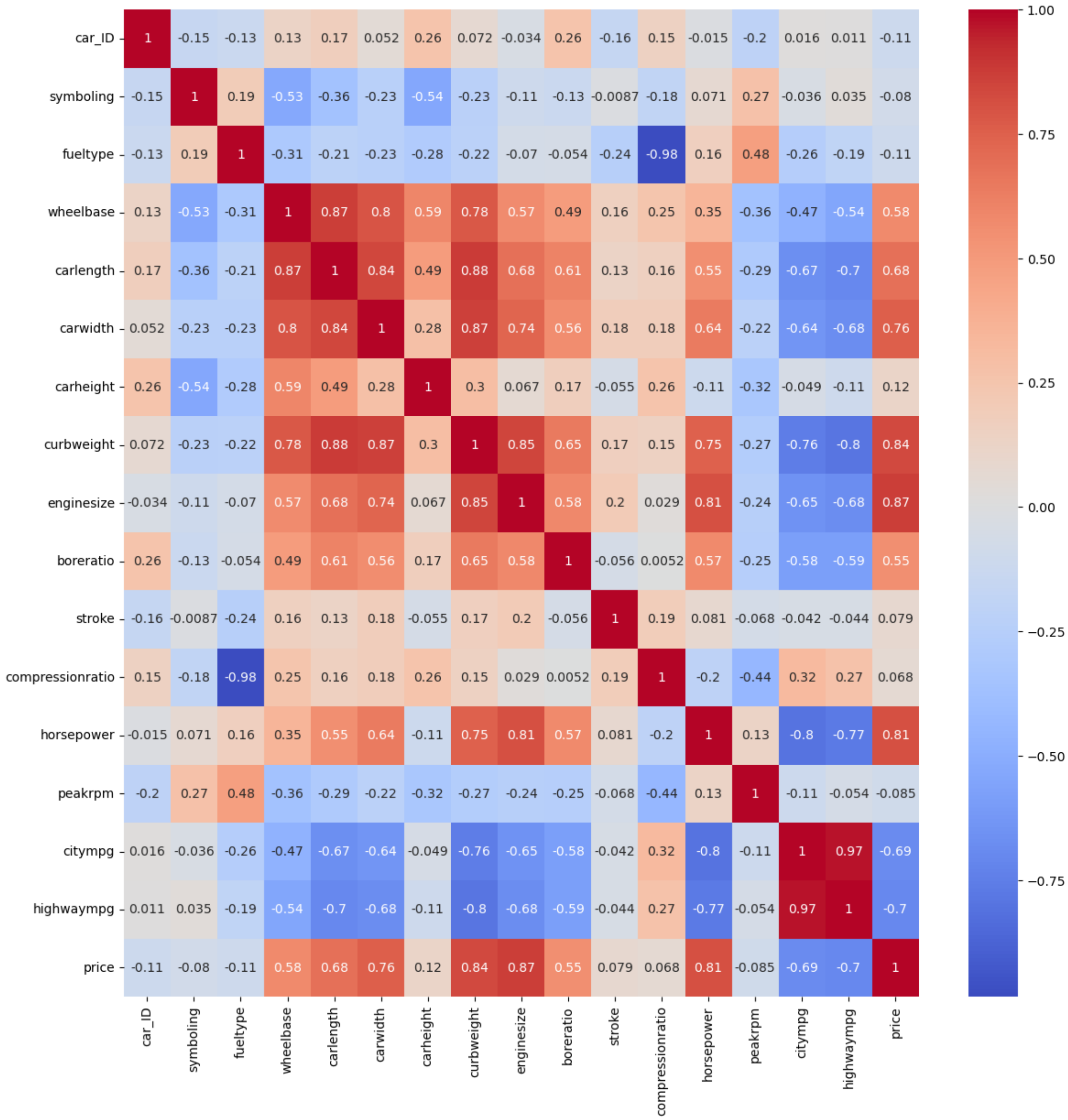
Q9 Data Visualization:

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

[28]:
```
corr_matrix = df.corr()
plt.figure(figsize=(14,14))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
```

[28]: <Axes: >



```
[29]: # Task 2: Update data frame 'df' to contain 5 columns from existing 'df' with
      ↪the highest correlation to column "price" and the price column itself.
      top_corr_cols = corr_matrix['price'].abs().sort_values(ascending=False)[1:6].
      ↪index.tolist()
      df=df[['price'] + top_corr_cols]
```

2.2 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

[30]:
```
y =df['price']
X =df.drop('price', axis=1)
X
```

[30]:

	enginesize	curbweight	horsepower	carwidth	highwaympg
0	130	2548	111	64.1	27
1	130	2548	111	64.1	27
2	152	2823	154	65.5	26
3	109	2337	102	66.2	30
4	136	2824	115	66.4	22
..	…	…	…	…	…
200	141	2952	114	68.9	28
201	141	3049	160	68.8	25
202	173	3012	134	68.9	23
203	145	3217	106	68.9	27
204	141	3062	114	68.9	25

[205 rows x 5 columns]

Q11 Use train_test_split to split the data set as train:test=(75%:25%) ratio.

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

[32]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=42)
# View the shape of your data set
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[32]: ((153, 5), (52, 5), (153,), (52,))

2.3 Regression Task

2.3.1 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 = _0 + _1*x_1 + _2*x_2 + … _p*x_p$

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

[34]:
```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

[34]: 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

[35]:
```
# 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(y_pred)
```

```
Test mean squared error (MSE): 14041064.92
[26090.89300501 19041.96831161 11282.01510357 13768.7747931
 23417.84670716  6488.8582937   6731.59600438  7375.70031489
 10692.93716198  6236.67774343 15565.73401052  7191.84067183
 15673.55244404 12304.60929575 38563.28129637  5491.50104535
 -1681.84856428 18842.11863788 11376.68555448 10481.56097807
 11879.06955665 21827.73930583  6465.95776328  3852.90908061
  5743.67538258 26905.50033411 15236.15410078 16483.23560903
  6499.16353239 16343.54237348 23148.76547475  5718.48479912
  6284.57214268 21599.31136587  8998.34450504 23125.86494434
 11809.84910882  8721.27893361  5230.2518289  18948.60610432
  9792.87020249 11738.73172882 14915.9927592   5874.6312074
  6423.59178201  9956.28155646  5718.48479912  8047.19820906
 16998.9051383  18816.92805442  5190.17590067 21965.36164919]
```

2.3.2 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

[36]:
```
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

[38]:
```
poly_reg_model = LinearRegression()
poly_reg_model.fit(poly_features, y_train)
```

```python
poly_reg_model.score(poly_features, y_train)
```

[38]: 0.8928327396853508