ECE580 Homework 1

Libo Zhang (NetID: lz200)

Data Preparation and Exploration

Question 1 (a) Solution:

I use pandas data frame to store the Automobile Data Set in Jupyter Notebook. Before data cleaning, the number of data points in the original data set is 205. I use 3 steps to clean my data set, and the results are shown below.

Step 1 – Remove the (non-continuous) features that are not of interest.

Number of data points in my data frame before Step 1 is 205.
Number of data points in my data frame after Step 1 is 205.
Number of data points removed at Step 1 is 0.

Step 2 – Remove any data points for which the target variable (price) is unknown.

Number of data points in my data frame before Step 2 is 205.
Number of data points in my data frame after Step 2 is 201.
Number of data points removed at Step 2 is 4.

Step 3 – Remove any data points for which the continuous predictor variables (features) are unknown.

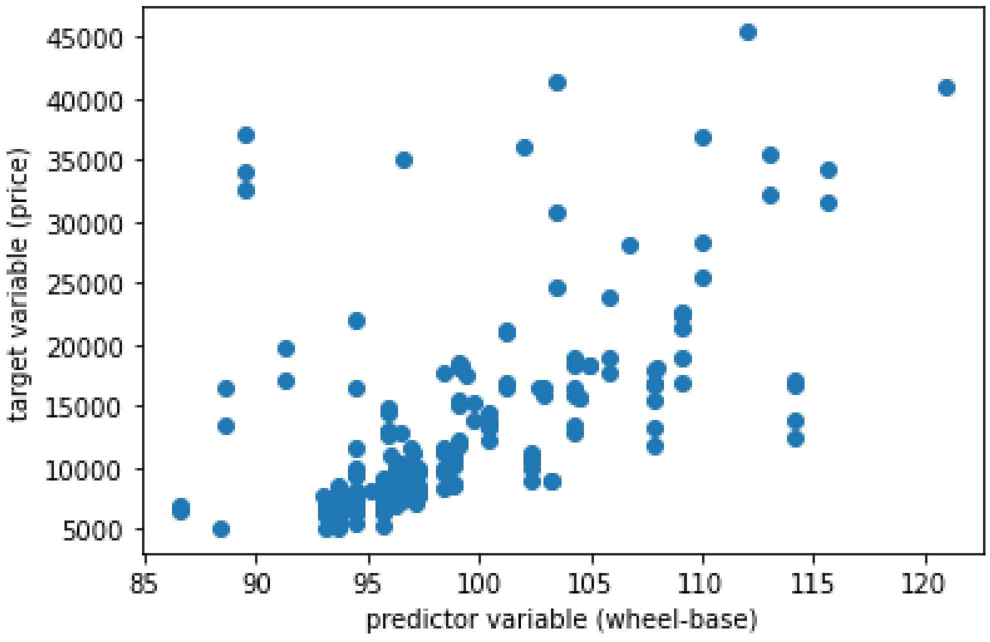
Number of data points in my data frame before Step 3 is 201.
Number of data points in my data frame after Step 3 is 195.
Number of data points removed at Step 3 is 6.

Conclusion: With the above 3 steps to clean the Automobile Data Set, I removed 10 data points. The number of data points that remain after cleaning the data set is 195.

Question 1 (b) & (c) Solutions:

I use matplotlib package to help plot 13 scatter plots. The required 13 scatter plots and their corresponding explanations are shown below.

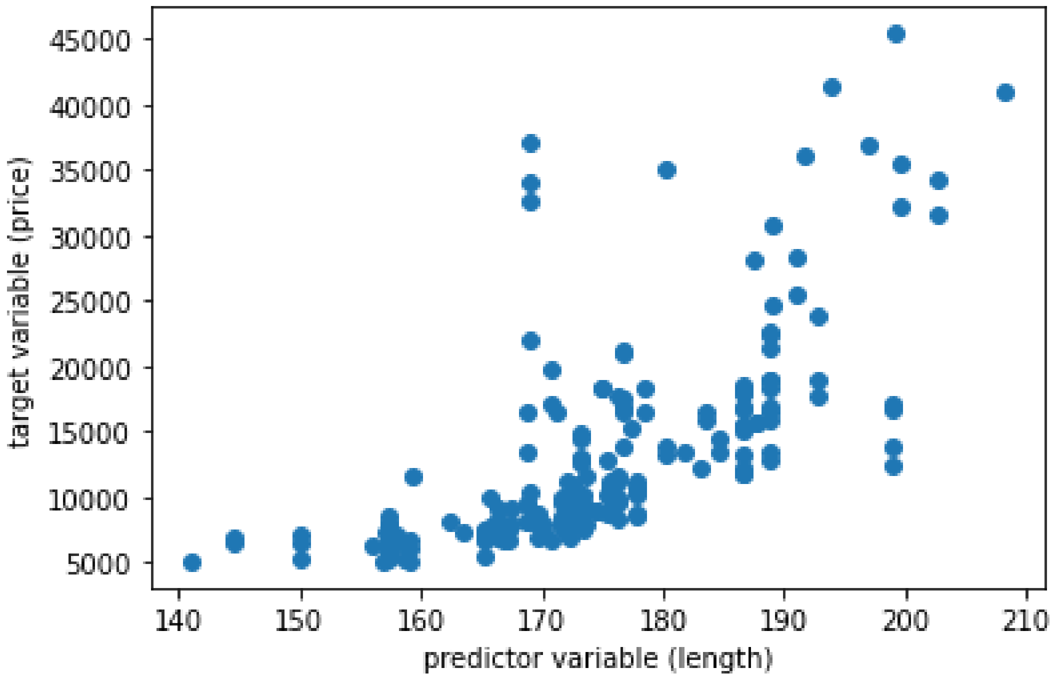
Scatter Plot 1



Explanation 1

Although there are several data points with very high target variable (price) values, all in all I think the automobile price has a linear relationship with feature wheel-base such that, when wheel-base becomes bigger, the automobile price generally becomes higher. Therefore, the feature wheel-base holds promise for predicting a car’s price.

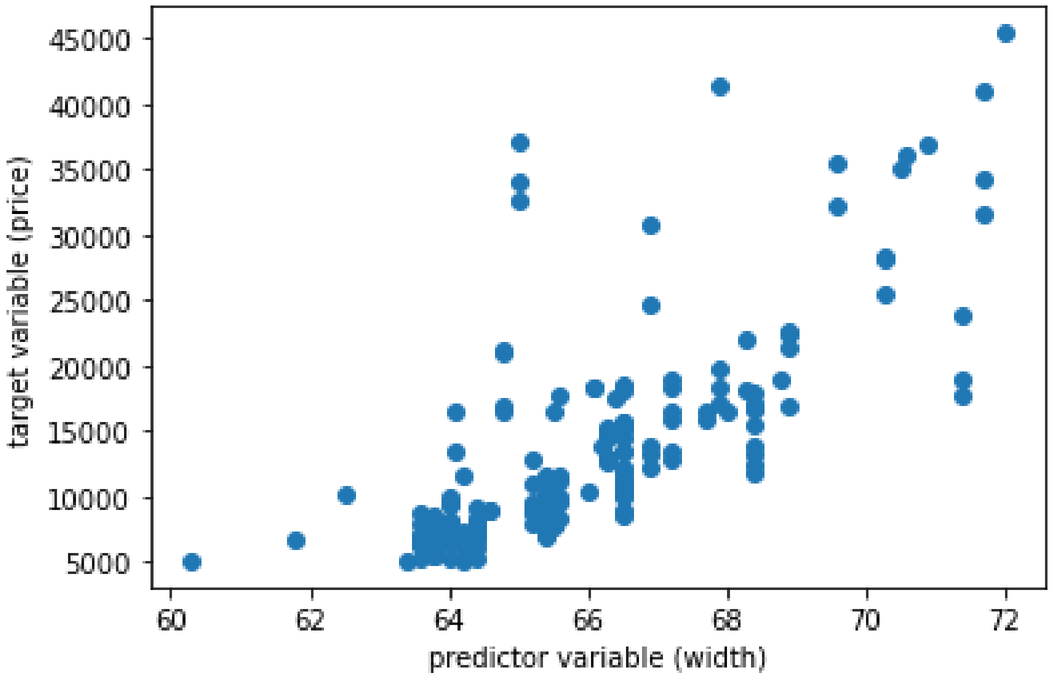
Scatter Plot 2



Explanation 2

The feature length holds promise for predicting a car’s price, because when the car’s length becomes longer, the car’s price generally becomes higher. There appears to be a nonlinear quadratic relationship between feature length and the car’s price, in such a way that $price = (length)^2$.

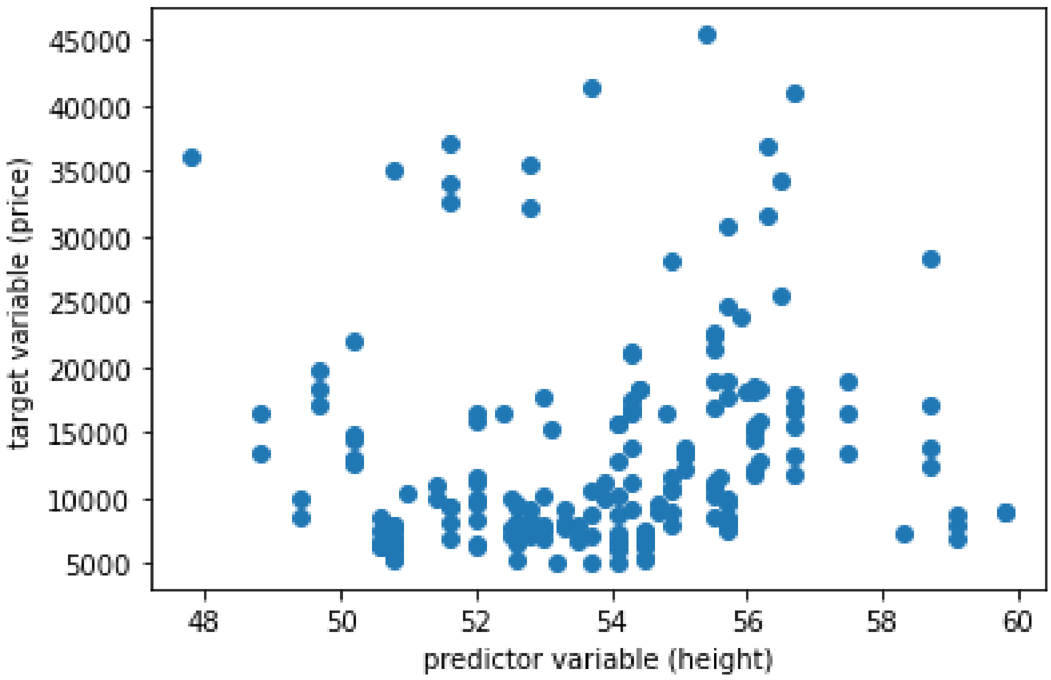
Scatter Plot 3



Explanation 3

The feature width holds promise for predicting a car's price, because when the car's width becomes wider, the car's price generally becomes higher. There appears to be a nonlinear quadratic relationship between feature width and the car's price, in such a way that $price = (width)^2$.

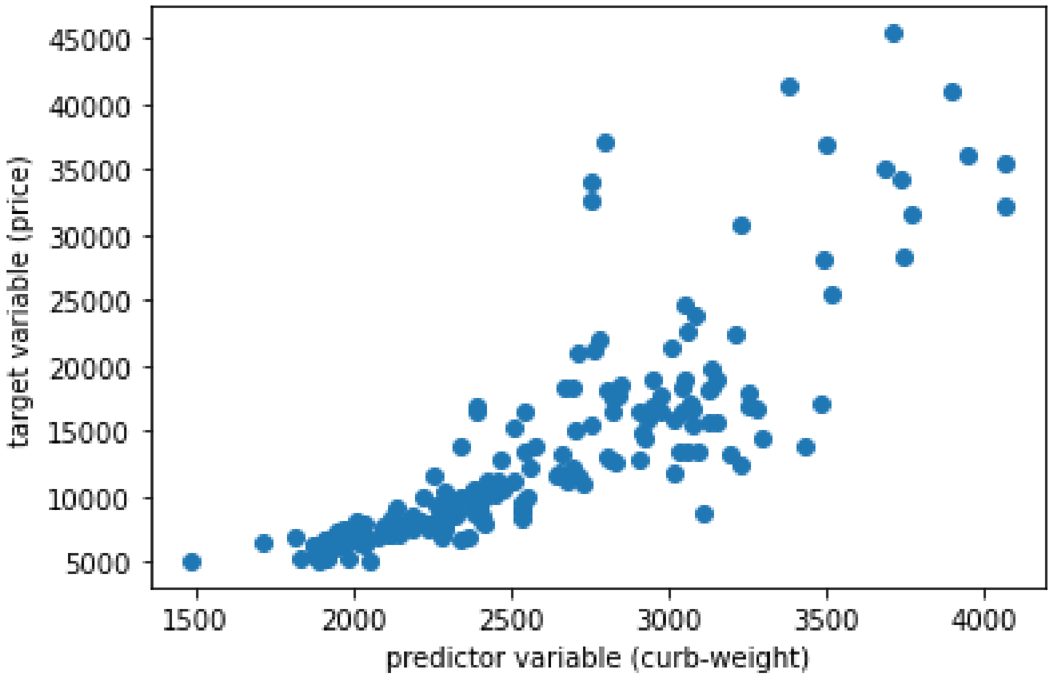
Scatter Plot 4



Explanation 4

The feature height does not hold promise for predicting a car's price, because although it seems to be a nonlinear quadratic relationship between feature height and the car's price, the data points are so dispersed and distorted that for the same height about 56, the cars' prices could range from around 10000 to even above 30000.

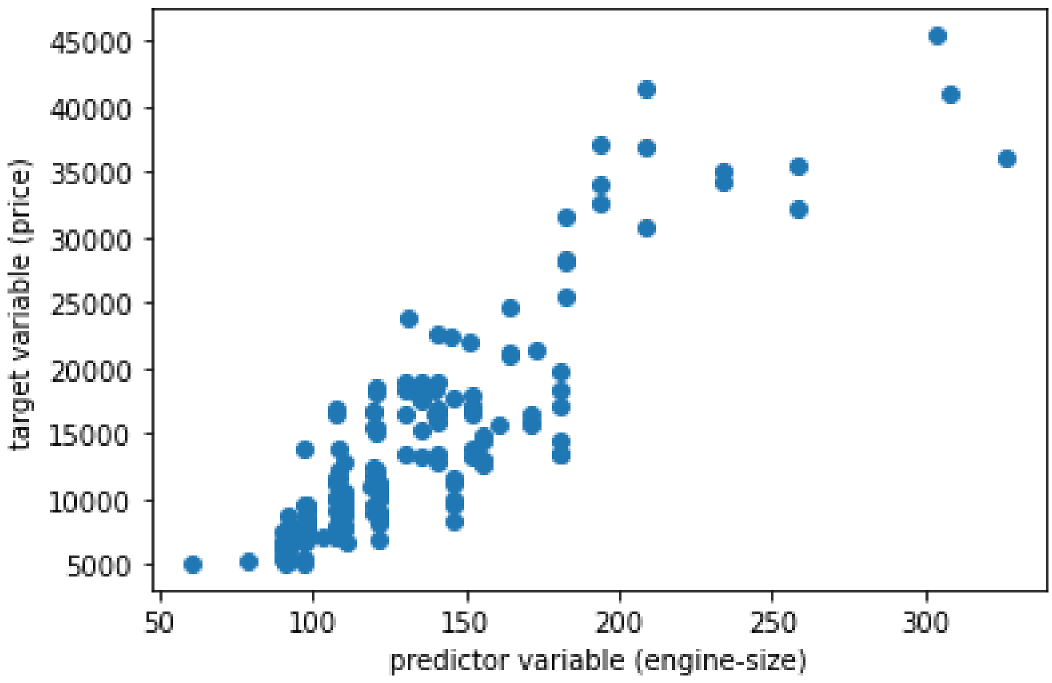
Scatter Plot 5



Explanation 5

The feature curb-weight holds promise for predicting a car's price, because when the car's curb-weight becomes bigger, the car's price generally becomes higher. There appears to be a nonlinear quadratic relationship between the feature curb-weight and the car's price, in such a way that $price = (curb\_weight)^2$.

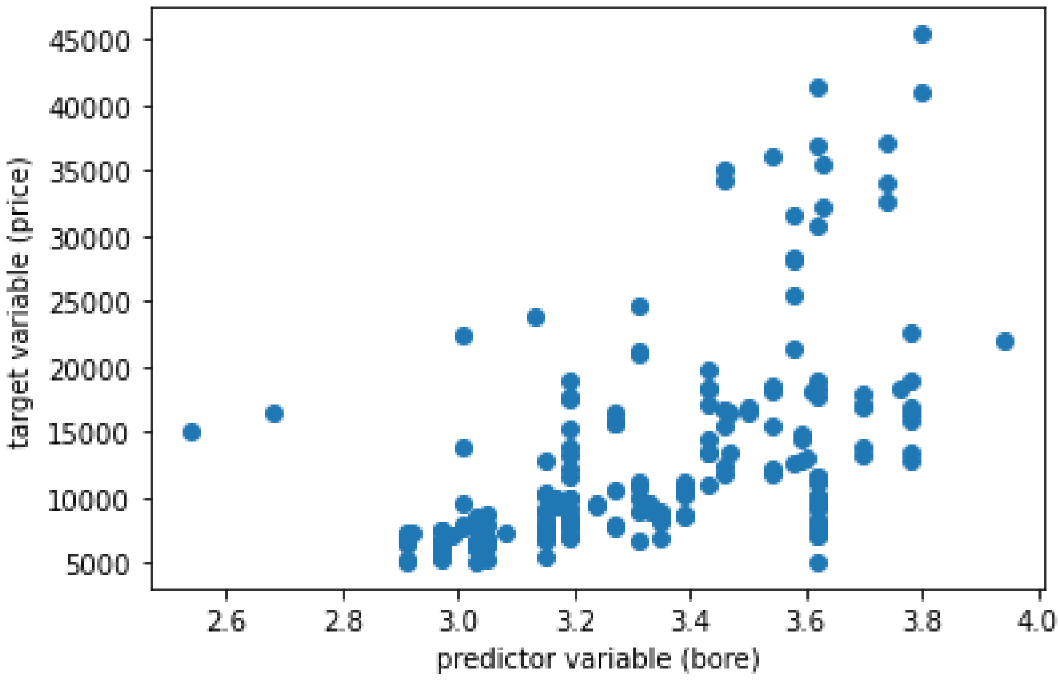
Scatter Plot 6



Explanation 6

The feature engine-size holds promise for predicting a car's price, because when the car's engine-size becomes bigger, the car's price generally becomes higher. Clearly, there appears to be a linear relationship between feature engine-size and the car's price.

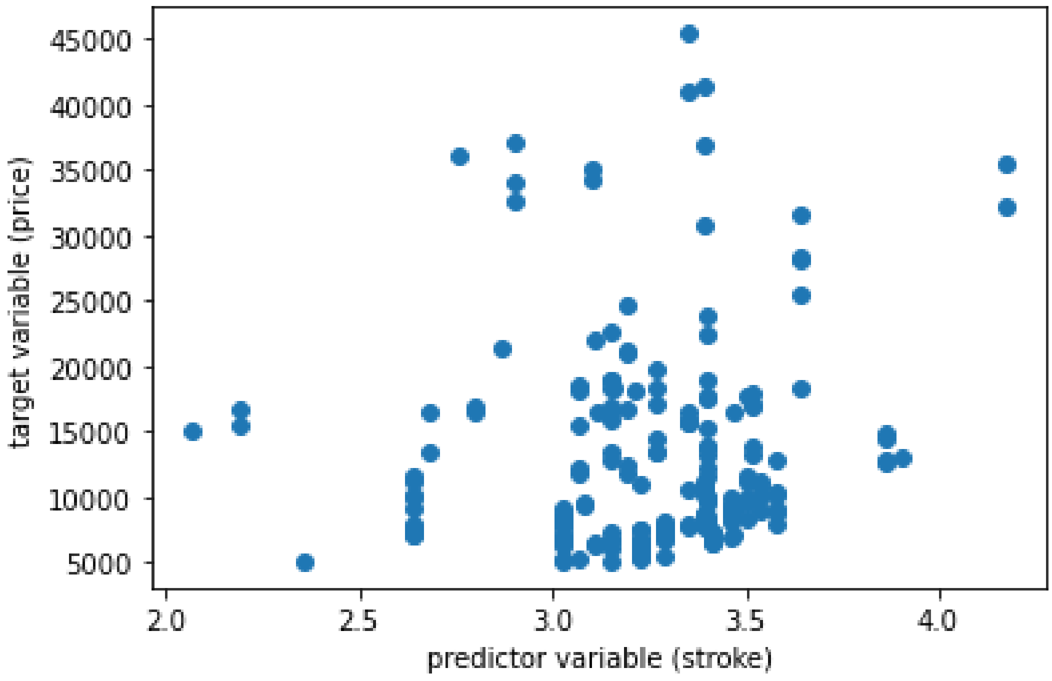
Scatter Plot 7



Explanation 7

The feature bore holds promise for predicting a car's price, because when the car's bore increases, the car's price generally increases. There appears to be a nonlinear quadratic relationship between the feature bore and the car's price, in such a way that $price = (bore)^2$.

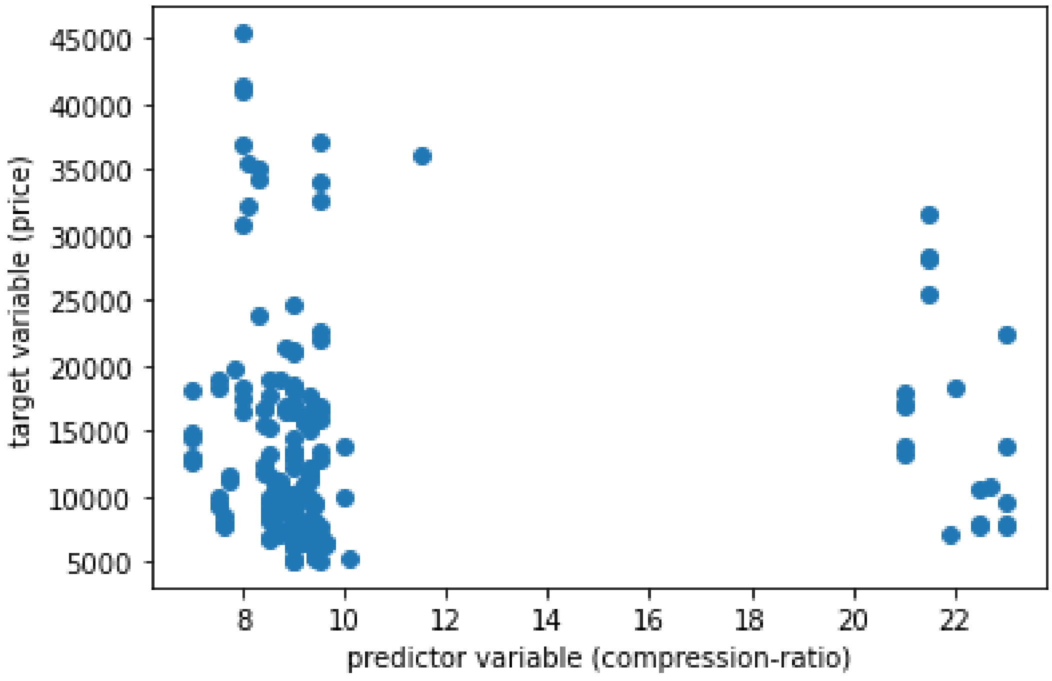
Scatter Plot 8



Explanation 8

The feature stroke does not hold promise for predicting a car's price, because most data points with various car prices are located between stroke value 3.0 and 4.0. Given Scatter Plot 8, neither linear nor nonlinear relationship between feature stroke and the car's price could be observed.

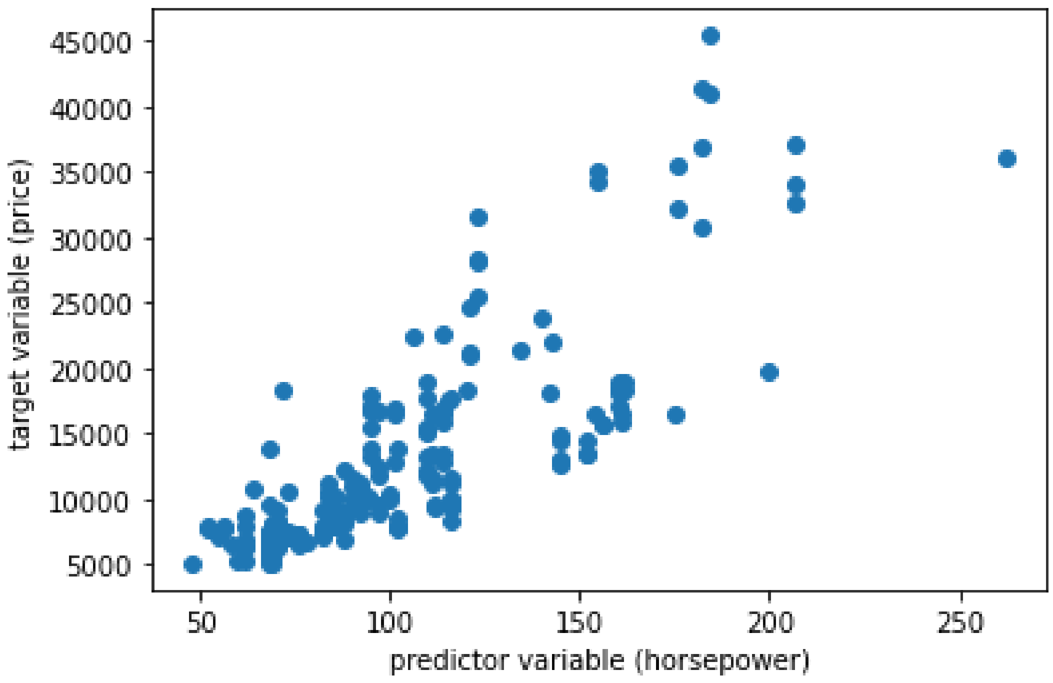
Scatter Plot 9



Explanation 9

The feature compression-ratio does not hold promise for predicting a car's price, because most data points with various car prices are located between compression-ratio value 8 and 10. Given Scatter Plot 9, neither linear nor nonlinear relationship between compression-ratio and the car's price could be found.

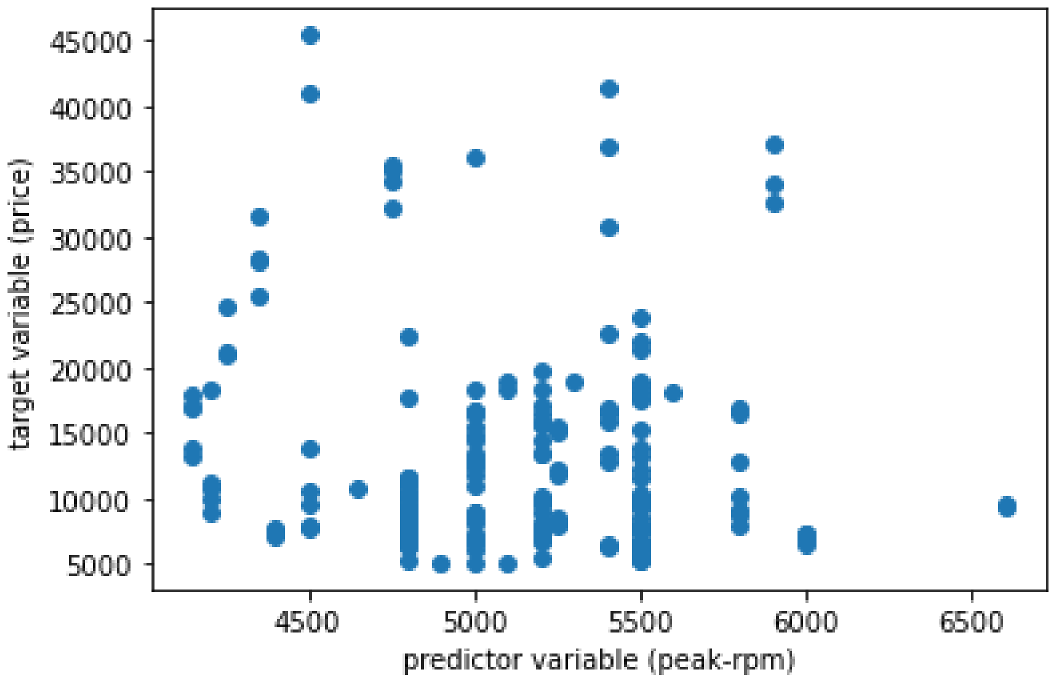
Scatter Plot 10



Explanation 10

The feature horsepower holds promise for predicting a car's price, because when the car's horsepower increases, the car's price generally increases, too. There appears to be a linear relationship between feature horsepower and the car's price.

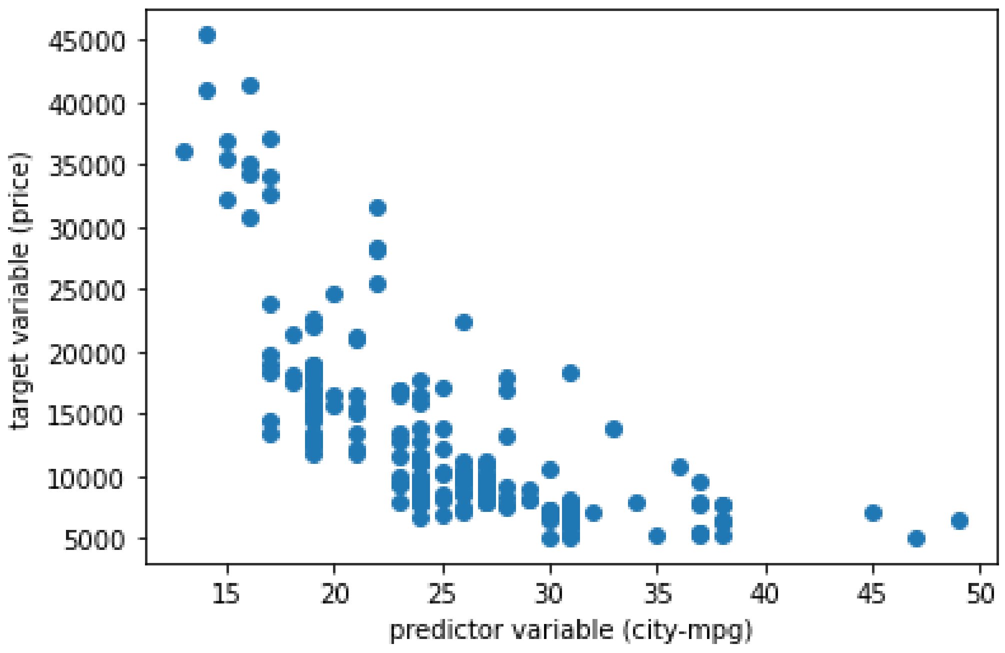
Scatter Plot 11



Explanation 11

The feature peak-rpm does not hold promise for predicting a car's price, because given Scatter Plot 11, the data points are so dispersed, diffuse, and distracted that neither linear nor nonlinear relationship between feature peak-rpm and the car's price could be observed.

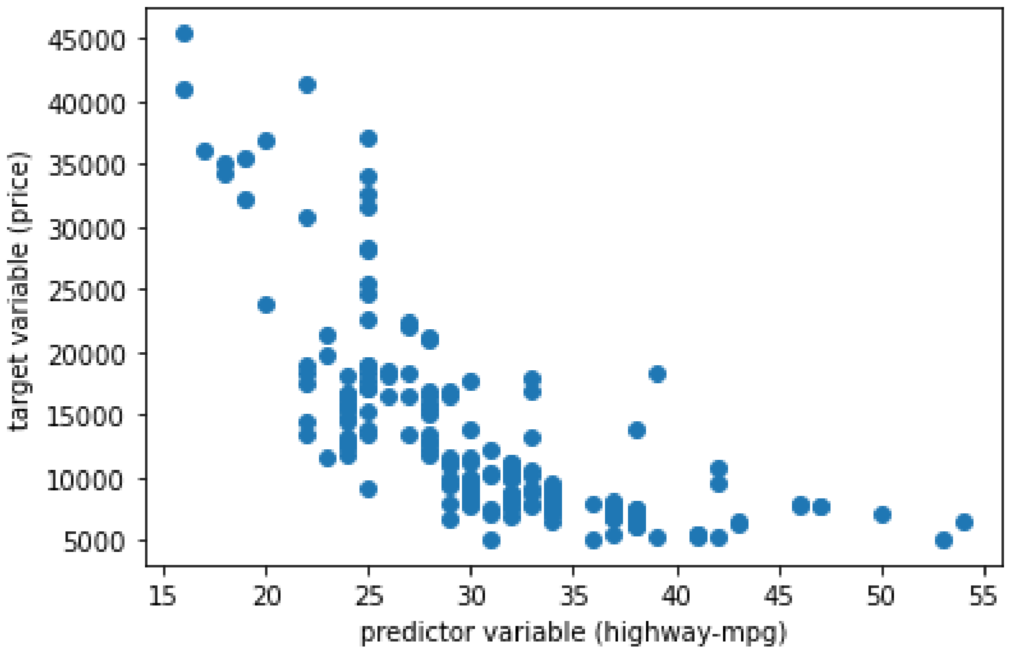
Scatter Plot 12



Explanation 12

The feature city-mpg holds promise for predicting a car's price, because when the car's city-mpg increases, the car's price generally decreases. There appears to be a nonlinear relationship between feature city-mpg and the car's price in such a way that $price = \frac{1}{(city\_mpg)}$.

Scatter Plot 13

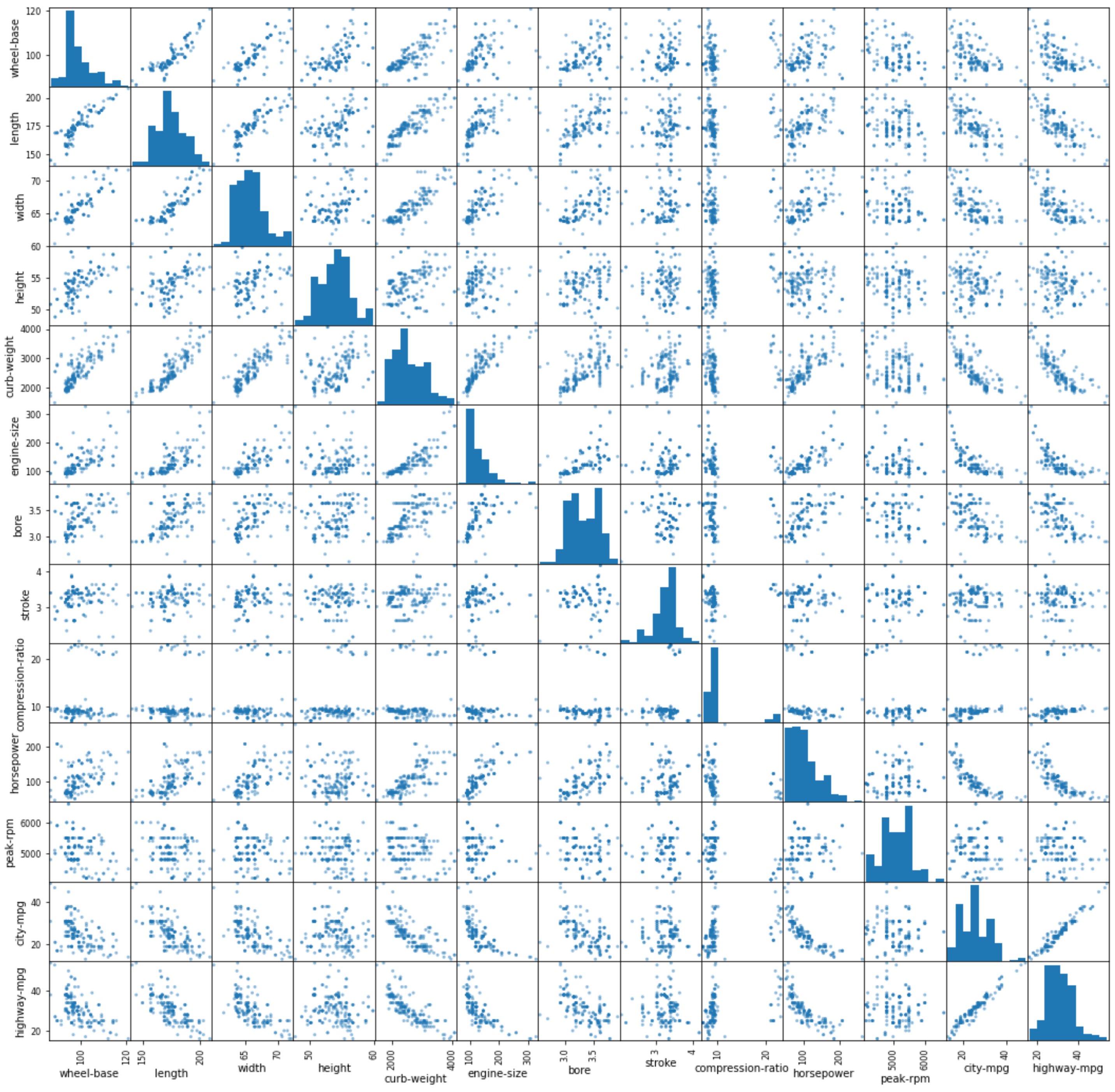


Explanation 13

The feature highway-mpg holds promise for predicting a car's price, because when the car's highway-mpg increases, the car's price generally decreases. There appears to be a nonlinear relationship between feature highway-mpg and the car's price in such a way that $price = \frac{1}{(highway\_mpg)}$.

Question 1 (d) Solution:

I use pandas "plotting.scatter_matrix()" function to help plot each pair-wise combination of features, and the scatter plots matrix is shown below. Only the upper half or the lower half of the matrix along the diagonal would be enough for us to examine the pair-wise combination of features since this scatter matrix is symmetric.



Question 1 (e) Solution:

I find that city-mpg and highway-mpg should preferably not both be used in the model simultaneously, because their pairwise scatter plot indicates a very clear linear relationship.

I find that wheel-base and length should preferably not both be used in the model simultaneously, because their pairwise scatter plot also indicates a very clear linear relationship.

I find that the feature curb-weight should not be used simultaneously with the following features: wheel-base, length, width, and engine-size. This is because the curb-weight's pairwise scatter plots with the other 4 features all indicate very clear linear relationship.

I find that the feature horsepower should not be simultaneously with the following features: city-mpg and highway-mpg. This is because the horsepower's pairwise scatter plots with the other 2 features both indicate very clear nonlinear relationship in such a way that $horsepower = \frac{1}{(city\_mpg)}$ or $horsepower = \frac{1}{(highway\_mpg)}$.

Question 1 (f) Solution:

My code for the "Data Preparation and Exploration" section is attached and shown below.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
%matplotlib inline
```

```
filename = "imports-85.data"
dataframe = pd.read_csv(filename, header = None)
```

```
print("The number of data points in the original data set is: " + str(len(dataframe)))
```

```
The number of data points in the original data set is: 205
```

```
# All 26 attributes in the data frame.
all_attributes = ["symboling", "normalized-losses", "make",
                  "fuel-type", "aspiration", "num-of-doors",
                  "body-style", "drive-wheels", "engine-location",
                  "wheel-base", "length", "width",
                  "height", "curb-weight", "engine-type",
                  "num-of-cylinders", "engine-size", "fuel-system",
                  "bore", "stroke", "compression-ratio",
                  "horsepower", "peak-rpm", "city-mpg",
                  "highway-mpg", "price"]
# Restrict ourselves to the 13 continuous predictor variables.
features = ["wheel-base", "length", "width",
            "height", "curb-weight", "engine-size",
            "bore", "stroke", "compression-ratio",
            "horsepower", "peak-rpm", "city-mpg", "highway-mpg"]
# Define the target variable.
target = "price"
# Assign each attribute to each column of the data frame correspondingly.
# This is because the original data set does not contain such information.
dataframe.columns = all_attributes
```

[5]:
```
# Question 1 (a)
# Step 1 - Remove the (non-continuous) features that are not of interest.
old_len = len(dataframe)
for attribute in all_attributes :
    if (attribute not in features) and (attribute != target) :
        dataframe.drop(attribute, axis = 1, inplace = True)
    else :
        continue
new_len = len(dataframe)
print("Number of data points in my data frame before Step 1: " + str(old_len))
print("Number of data points in my data frame after Step 1: " + str(new_len))
print("Number of data points removed at Step 1: " + str(old_len - new_len))
```

```
Number of data points in my data frame before Step 1: 205
Number of data points in my data frame after Step 1: 205
Number of data points removed at Step 1: 0
```

[6]:
```
# Question 1 (a)
# Step 2 - Remove any data points for which the target variable (price) is unknown.
old_len = len(dataframe)
dataframe = dataframe[dataframe[target] != "?"]
new_len = len(dataframe)
print("Number of data points in my data frame before Step 2: " + str(old_len))
print("Number of data points in my data frame after Step 2: " + str(new_len))
print("Number of data points removed at Step 2: " + str(old_len - new_len))
```

```
Number of data points in my data frame before Step 2: 205
Number of data points in my data frame after Step 2: 201
Number of data points removed at Step 2: 4
```

[7]:
```
# Question 1 (a)
# Step 3 - Remove any data points for which the continuous predictor variables are unknown.
old_len = len(dataframe)
for feature in features :
    dataframe = dataframe[dataframe[feature] != "?"]
new_len = len(dataframe)
print("Number of data points in my data frame before Step 3: " + str(old_len))
print("Number of data points in my data frame after Step 3: " + str(new_len))
print("Number of data points removed at Step 3: " + str(old_len - new_len))
```

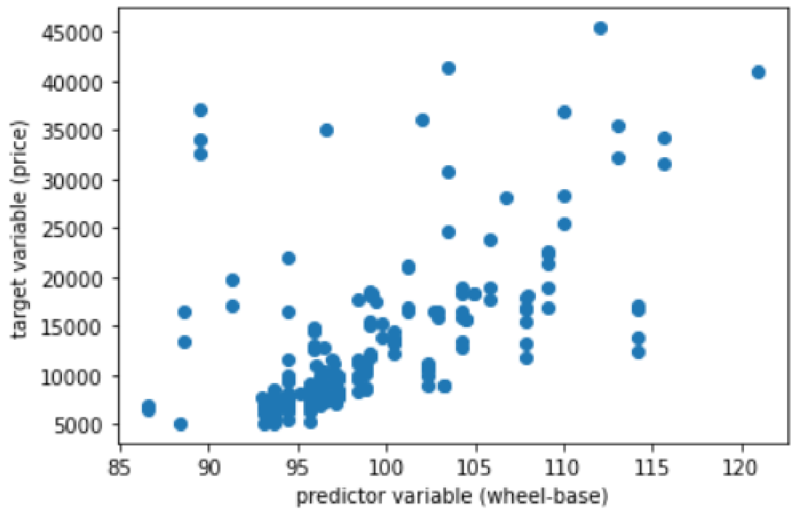
```
Number of data points in my data frame before Step 3: 201
Number of data points in my data frame after Step 3: 195
Number of data points removed at Step 3: 6
```

[8]:
```
# Do some housekeeping in this block.
df_numpy = dataframe.to_numpy()
for j in range(df_numpy.shape[1]) :
    for i in range(df_numpy.shape[0]) :
        df_numpy[i, j] = float(df_numpy[i, j])
features_matrix = df_numpy[:, 0:13]
target_vector = df_numpy[:, 13]
df_numpy = df_numpy.astype(np.float64)
print(features_matrix.shape)
print(target_vector.shape)
```

```
(195, 13)
(195,)
```

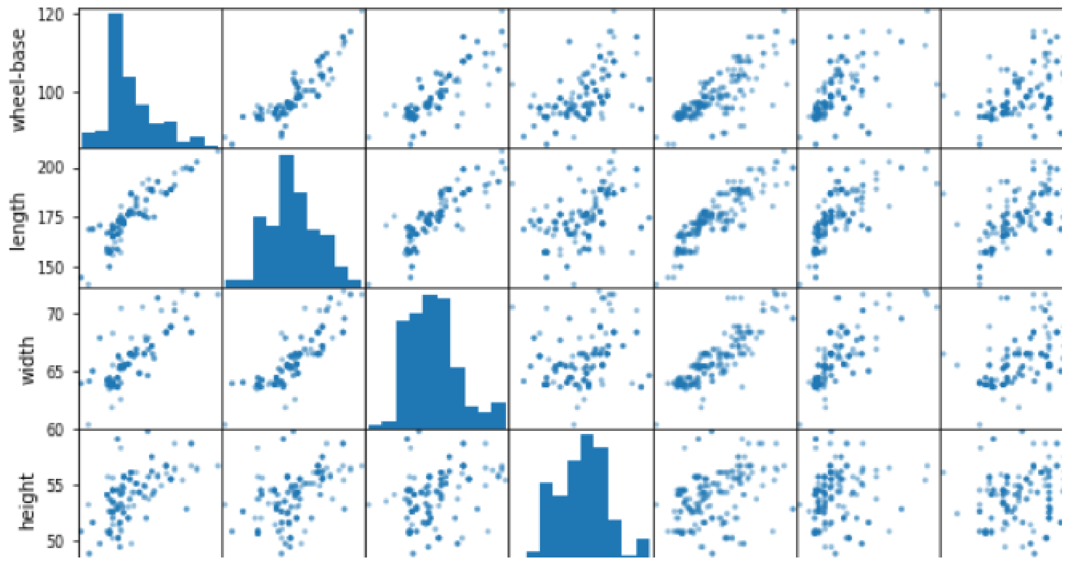
```
[9]: # Question 1 (b)
     def scatter_plot (features_matrix, target_vector, features) :
         for i in range(len(features)) :
             print("price vs. " + features[i])
             figure, axis = plt.subplots()
             plt.scatter(features_matrix[:, i], target_vector)
             plt.xlabel("predictor variable" + " " + "(" + features[i] + ")")
             plt.ylabel("target variable (price)")
             plt.show()

     # Make the 13 scatter plots.
     scatter_plot(features_matrix, target_vector, features)
```

```
price vs. wheel-base
```



```
price vs. length
```

```
[10]: # Question 1 (d)
      # Plot each pair-wise combination of features using scatter plots.
      # Make the scatter plots matrix.
      df_features = pd.DataFrame(df_numpy[:, 0:13], columns = features)
      pd.plotting.scatter_matrix(df_features, figsize = (18, 18));
```



Regression

Question 2 (a) Solution:

Model #1 – I decide to use only one feature, engine-size, in this linear model, and the model equation should be

$$\widehat{price} = \widehat{w_0} + \widehat{w_1}(engine\_size)$$

Explanation #1 – For the 3 unique linear models, I decide to use only one feature in Model 1, two features in Model 2, and three features in Model 3. Features used in Model 1 and 2 will also be used in Model 3, because individual features may be utilized in more than one model, and gradually increasing the number of features makes sure that the subset of features used in each model must be unique. In such way, I want to test if regression performance could gradually improve through adding more features in my model. In Model 1, I select engine-size as the single feature because according to the previous scatter plots, there appears to be a very clear linear relationship between feature engine-size and the car's price, and the data points are not dispersed or diffuse. Therefore, with this single feature in my model, I think relatively good performance could still be achieved.

Model #2 – I decide to use two features in this linear model. One is the previously used feature, engine-size, and the other feature is horsepower. The model equation should be

$$\widehat{price} = \widehat{w_0} + \widehat{w_1}(engine\_size) + \widehat{w_2}(horsepower)$$

Explanation #2 – I select horsepower as the second feature for two reasons. First, in the scatter plot of Question 1 (b) & (c), I find that there also appears to be a very good linear relationship between horsepower and the car's price. Second, in the pairwise features scatter plot, I find that engine-size and horsepower are not significantly correlated. Although we tend to think that if a car has a large engine-size, then its horsepower should also be large, I find that data points are still a bit dispersed and diffuse in the pairwise features scatter plot (at least compared with what I discussed in Question 1 (e)). Therefore, I expect that Model 2 could have a better regression performance (provide more addition information) by adding the second linear feature.

Model #3 – I decide to use three features in this linear model. The first two are still engine-size and horsepower. The third feature is width, with transformed quadratic nonlinear relationship. Then the model equation should be

$$\widehat{price} = \widehat{w_0} + \widehat{w_1}(engine\_size) + \widehat{w_2}(horsepower) + \widehat{w_3}(width) + \widehat{w_4}(width)^2$$

Explanation #3 – I select width as the third feature for two reasons. First, there appears to be a relatively good and clear quadratic nonlinear relationship between width and the car's price, so I choose the quadratically transformed feature to capture such nonlinear relationship I observed in Question 1. Second, in fact, there are other features having very good nonlinear relationship with the car's price, most of which, however, are strongly correlated with either engine-size or horsepower, such as curb-weight. Feature width is not strongly correlated with the first two features based on the pairwise features scatter plot, so I hope that adding this nonlinearly transformed feature could provide more additional information and improve my model's regression performance.

Note: Model #3 design methodology has been discussed with and approved by professor Tantum.

Question 2 (b) Solution:

I use the "LinearRegression()" function from "sklearn.linear_model" to help perform linear regression.

For Model #1, the model parameters and the model equation are shown below.

$$\widehat{price} = \widehat{w_0} + \widehat{w_1}(engine\_size)$$

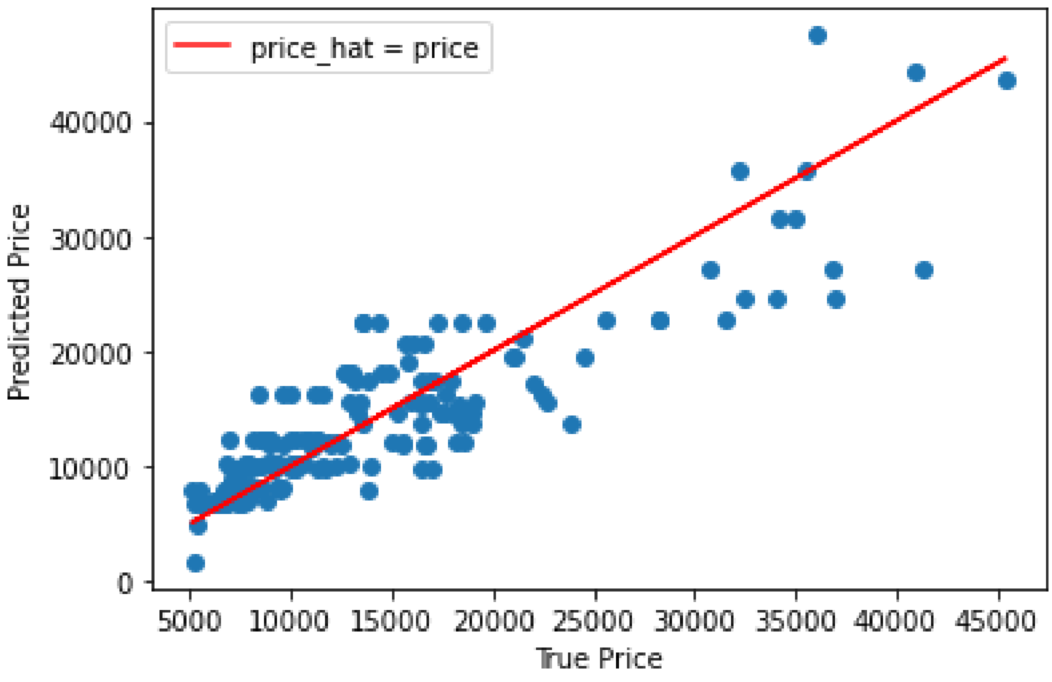
$$\widehat{price} = (-8865.4063) + 172.8442(engine\_size)$$

The Unadjusted R^2 for Model #1 is 0.7902.

The Adjusted R^2 for Model #1 is 0.7891.

Note – Although the homework does not ask to calculate the Adjusted R-squared value, here I decide to calculate and list both R-squared values because the Unadjusted R-squared would always tend to increase if we add more features into the model. If newly added features are strongly correlated with the old ones, as the homework handout mentions, this behavior could not provide more useful information and may result in overfitting or computational challenges. The Adjusted R-squared value could help us check the correlation between the newly added features and the old ones. In other words, if I add one feature in my model and the Adjusted R-squared value explicitly increases, this means that adding this new feature does provide additionally useful information and does improve the model's regression performance.

The scatter plot of the Model #1's predicted price as a function of the true price, along with the reference line representing perfect prediction, is shown below.



<u>Question: What is your impression of this model?</u>

Answer: My impression for Model #1 is "Not Bad".

<u>Question: How do the predicted prices compare to the true prices?</u>

I think for Model #1, some predicted prices do have a certain amount of distance to the reference line, and such distances represents clearly large differences from the true prices. However, there are still many predicted prices which are along the reference line, or very close to the reference line. This means many predicted prices are close to the true prices (some lower and some higher though). Therefore, I think all in all, Model #1's regression performance is "Not Bad".

Question: Are there price ranges where the model is particularly good?

Answer: I think for prices ranging from 5000 to 20000, Model #1 has relatively good performance.

Question: Are there price ranges where the model is particularly bad?

Answer: I think for prices ranging from 20000 to 45000, Model #1 has very bad performance, because many predicted prices are much lower than the true prices.

Question 2 (c) Solution:

For Model #2, the model parameters and the model equation are shown below.

$$\widehat{price} = \widehat{w_0} + \widehat{w_1}(engine\_size) + \widehat{w_2}(horsepower)$$

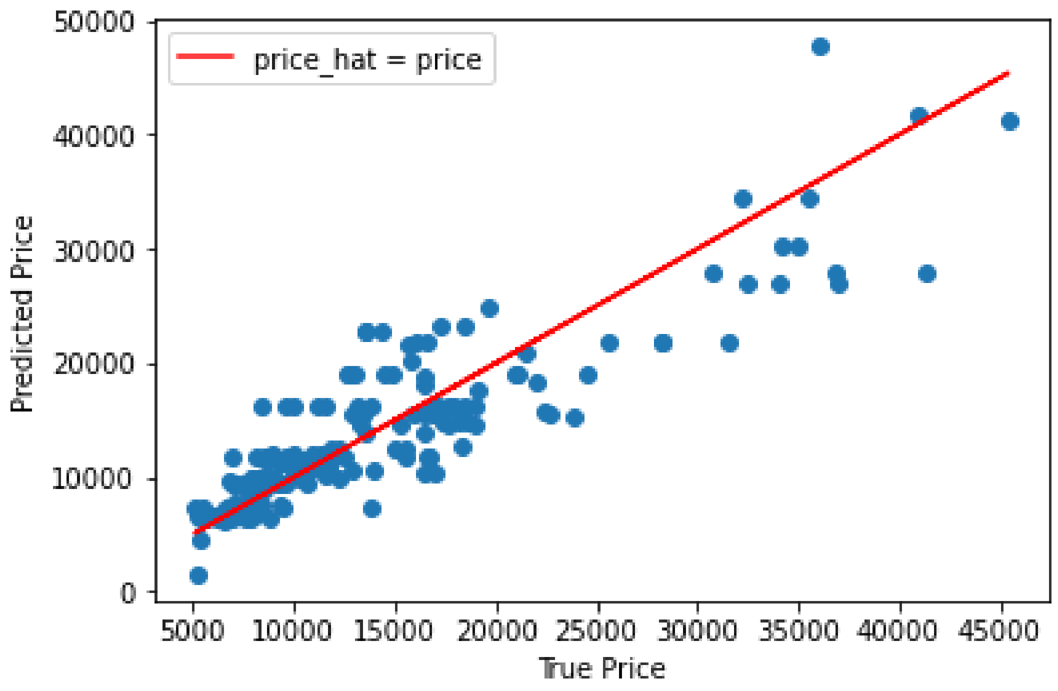
$$\widehat{price} = (-9080.5162) + 137.8419(engine\_size) + 45.4456(horsepower)$$

The Unadjusted R^2 for Model #2 is 0.8034.

The Adjusted R^2 for Model #2 is 0.8014.

We can see that both R-squared values of Model #2 are greater than those of Model #1.

The scatter plot of the Model #2's predicted price as a function of the true price, along with the reference line representing perfect prediction, is shown below.



Question: What is your impression of this model?

Answer: I think Model #2 is relatively good, at least better than Model #1.

Question: How do the predicted prices compare to the true prices?

Answer: Although there some data points far away from the reference line, I think many data points are very close to the reference line, which means that many predicted prices are actually very close to the true prices (some lower and some higher though). Therefore, I think Model #2 is relatively good.

Question: Are there price ranges where the model is particularly good?

Answer: I think for prices ranging from 5000 to 20000, Model #2 has relatively good performance.

Question: Are there price ranges where the model is particularly bad?

Answer: I think for prices ranging from 20000 to 45000, Model #2 has very bad performance, because some predicted prices are far too lower than the true prices.

Question 2 (d) Solution:

For Model #3, the model parameters and the model equation are shown below.

$$\widehat{price} = \widehat{w_0} + \widehat{w_1}(engine\_size) + \widehat{w_2}(horsepower) + \widehat{w_3}(width) + \widehat{w_4}(width)^2$$

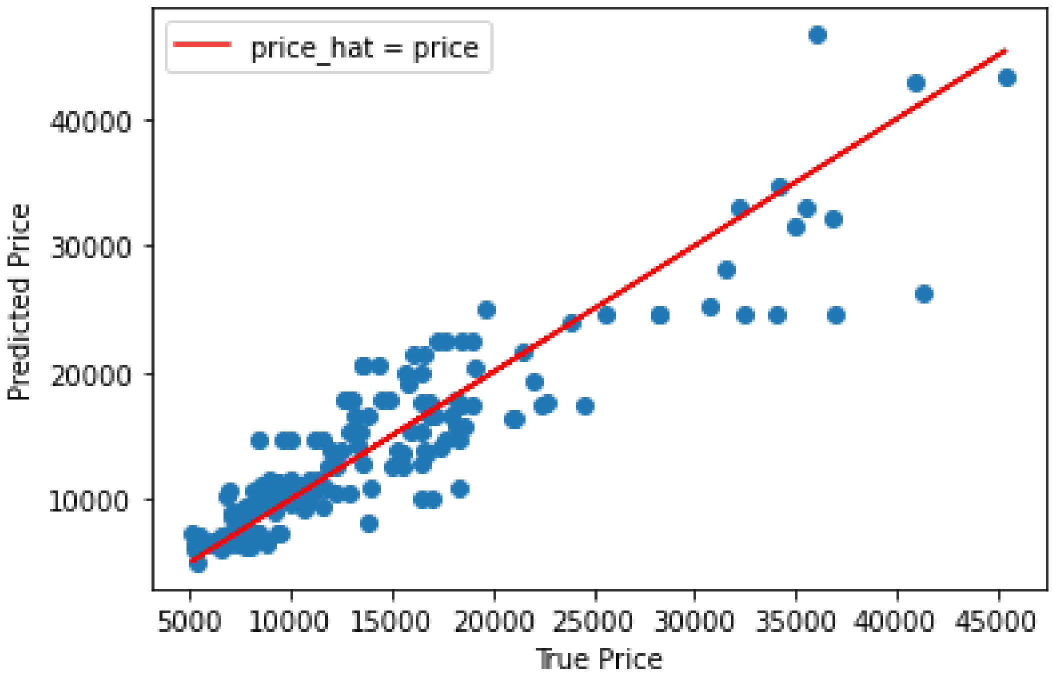
$$\widehat{price} = 809939.54 + 86.75(engine\_size) + 64.85(horsepower) + (-25283.71)(width) + 195.92(width)^2$$

The Unadjusted R^2 for Model #3 is 0.8440.

The Adjusted R^2 for Model #3 is 0.8407.

We can see that both Unadjusted and Adjusted R-squared values of Model #3 are greater than those of Model #1 and Model #2, which demonstrates my expectation when designing these three linear regression models.

The scatter plot of Model #3's predicted price as a function of the true price, along with the reference line representing perfect prediction, is shown below.



Question: What is your impression of this model?

Answer: I think Model #3's performance is very good, and I think Model #3 is the best model among all the three models that I designed, which exactly meets my expectation.

Question: How do the predicted prices compare to the true prices?

Answer: Many data points are fairly close to the reference line, and only a few points are far away from the line.

This means that many predicted prices are very close to the true prices (some lower and some higher though). Here I also need to mention that many predicted prices are actually much closer to the true prices, compared with Model #1 and Model #2.

Question: Are there price ranges where the model is particularly good?

Answer: I think for prices ranging from 5000 to 30000, Model #3 has very good performance.

Question: Are there price ranges where the model is particularly bad?

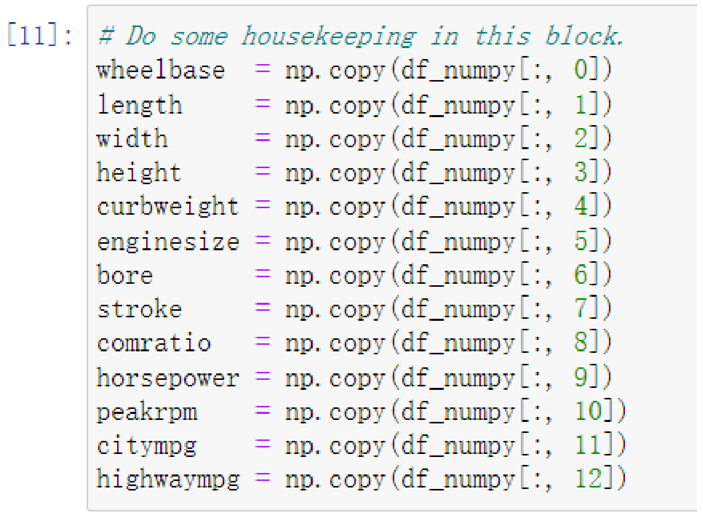
Answer: I think for prices ranging from 30000 to 45000, Model #3 has relatively bad performance. In this range, there are only a small amount of data points. Several predicted prices are relatively accurate, but there are a few predicted prices which are either much higher or much lower than the true prices.

Question 2 (e) Solution:

I would select Model #3. To explain my reasoning, firstly, Model #3 has the highest Unadjusted R-squared value and the highest Adjusted R-squared value, which means that Model #3 has the best regression performance among all three models. We can also verify this reasoning by examining three scatter plots of the predicted price as a function of the true price. Secondly, this decision actually meets my expectation when I initially designed these three models, as Model #1 having one feature, Model #2 having two features, and Model #3 having three features. If the feature I choose to add does provide additional information, then it is reasonable to have the fact that, Model #2 is better than Model #1, and Model #3 is better than Model #2. In addition, although Model #3 has the largest number of parameters, all three models still belong to linear regression models. Therefore, in conclusion, from both theoretical and experimental perspectives, I would certainly select Model #3 as the best model.

Question 2 (f) Solution:

My code for the "Regression" section is attached and shown below.



```
[11]: # Do some housekeeping in this block.
      wheelbase  = np.copy(df_numpy[:, 0])
      length     = np.copy(df_numpy[:, 1])
      width      = np.copy(df_numpy[:, 2])
      height     = np.copy(df_numpy[:, 3])
      curbweight = np.copy(df_numpy[:, 4])
      enginesize = np.copy(df_numpy[:, 5])
      bore       = np.copy(df_numpy[:, 6])
      stroke     = np.copy(df_numpy[:, 7])
      comratio   = np.copy(df_numpy[:, 8])
      horsepower = np.copy(df_numpy[:, 9])
      peakrpm    = np.copy(df_numpy[:, 10])
      citympg    = np.copy(df_numpy[:, 11])
      highwaympg = np.copy(df_numpy[:, 12])
```

Please continue to the next page.

[12]:
```
# Do some housekeeping in this block.
X_one = enginesize.reshape(-1, 1)

X_two = np.column_stack((enginesize, horsepower))

X_three = np.column_stack((enginesize, horsepower,
                           width, width * width))

y = df_numpy[:, -1]

print(X_one.shape)
print(X_two.shape)
print(X_three.shape)
print(y.shape)
```

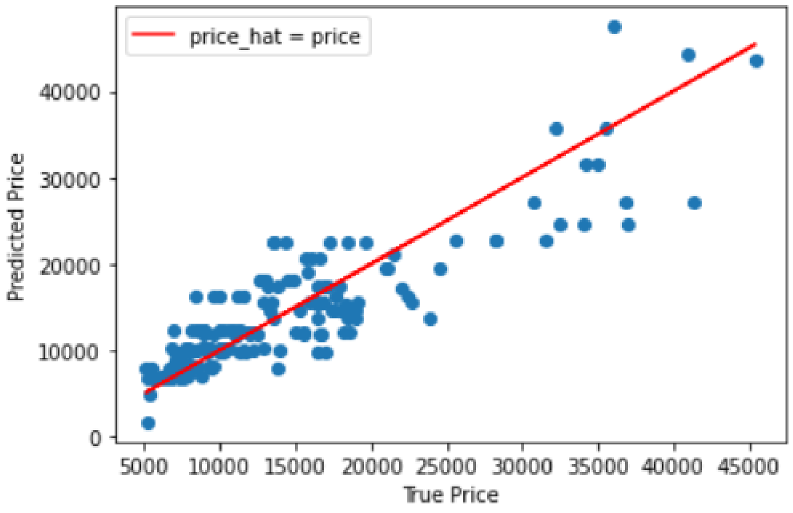
```
(195, 1)
(195, 2)
(195, 4)
(195,)
```

[13]:
```
# Define a function to calculate Adjusted R-squared.
def cal_adj_R_square (model, X, y) :
    # Extract the number of observatons
    n = X.shape[0]
    # Extract the number of predictor variables
    k = X.shape[1]
    # First find the unadjusted R-square score
    R_square = model.score(X, y)
    # Calculate the adjusted R-square score
    adj_R_square = 1 - (((1-R_square) * (n-1)) / (n-k-1))
    return adj_R_square
# Define a function to scatter plot the predicted price
# as a function of the true price, and plot the reference
# line representing perfect prediction.
def plot_pred_true_price (y_hat, y) :
    figure, axis = plt.subplots()
    axis.plot(y, y, label = "price_hat = price", color = "red")
    axis.scatter(y, y_hat)
    plt.xlabel("True Price")
    plt.ylabel("Predicted Price")
    plt.legend()
    plt.show()
```

[14]:
```
# Question 2 (b) i. + ii.
# Model #1
model_one = LinearRegression().fit(X_one, y)
yhat_one = model_one.predict(X_one)
print("w0 = %.4f (intercept)" % model_one.intercept_)
print("w1 = %.4f" % model_one.coef_[0])
print("Unadjusted R-squared = %.4f" % model_one.score(X_one, y))
print("Adjusted R-squared   = %.4f" % cal_adj_R_square(model_one, X_one, y))
```

```
w0 = -8865.4063 (intercept)
w1 = 172.8442
Unadjusted R-squared = 0.7902
Adjusted R-squared   = 0.7891
```

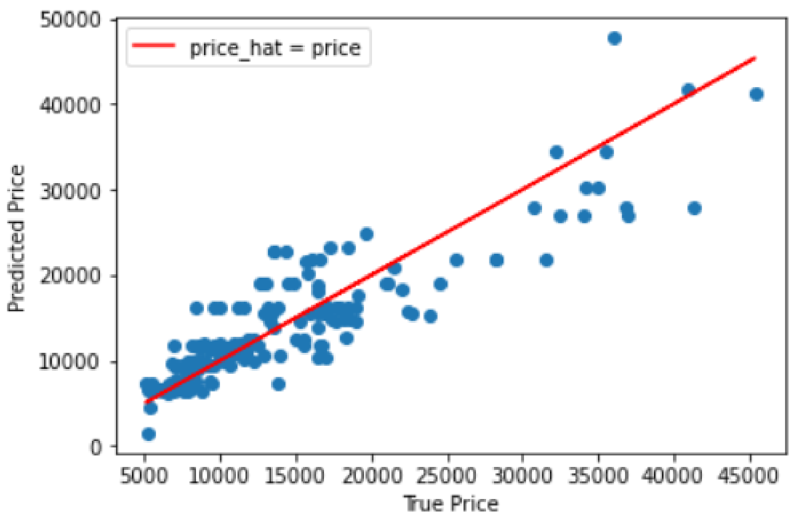
```
[15]: # Question 2 (b) iii.
      # Model #1
      plot_pred_true_price(yhat_one, y)
```



```
[16]: # Question 2 (c) i. + ii.
      # Model #2
      model_two = LinearRegression().fit(X_two, y)
      yhat_two = model_two.predict(X_two)
      print("w0 = %.4f (intercept)" % model_two.intercept_)
      print("w1 = %.4f" % model_two.coef_[0])
      print("w2 = %.4f" % model_two.coef_[1])
      print("Unadjusted R-squared = %.4f" % model_two.score(X_two, y))
      print("Adjusted R-squared   = %.4f" % cal_adj_R_square(model_two, X_two, y))
```

```
w0 = -9080.5162 (intercept)
w1 = 137.8419
w2 = 45.4456
Unadjusted R-squared = 0.8034
Adjusted R-squared   = 0.8014
```

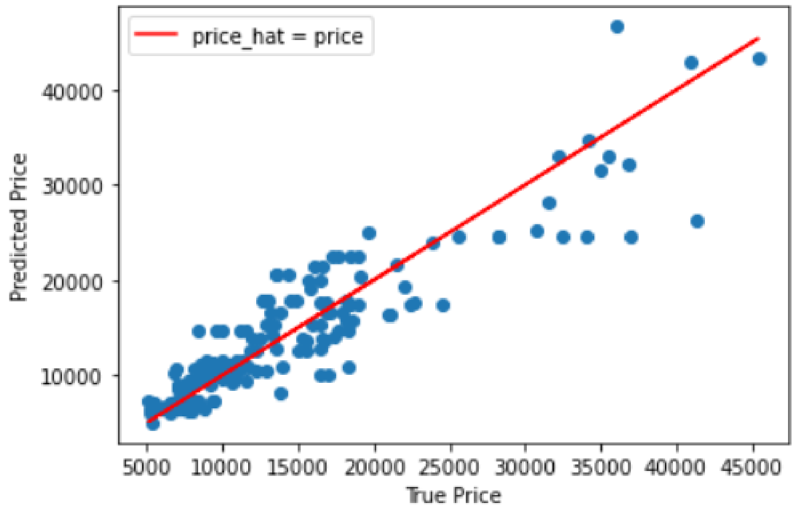
```
[17]: # Question 2 (c) iii.
      # Model #2
      plot_pred_true_price(yhat_two, y)
```



[18]:
```
# Question 2 (d) i. + ii.
# Model #3
model_three = LinearRegression().fit(X_three, y)
yhat_three = model_three.predict(X_three)
print("w0 = %.4f (intercept)" % model_three.intercept_)
print("w1 = %.4f" % model_three.coef_[0])
print("w2 = %.4f" % model_three.coef_[1])
print("w3 = %.4f" % model_three.coef_[2])
print("w4 = %.4f" % model_three.coef_[3])
print("Unadjusted R-squared = %.4f" % model_three.score(X_three, y))
print("Adjusted R-squared   = %.4f" % cal_adj_R_square(model_three, X_three, y))
```

```
w0 = 809939.5388 (intercept)
w1 = 86.7531
w2 = 64.8501
w3 = -25283.7140
w4 = 195.9169
Unadjusted R-squared = 0.8440
Adjusted R-squared   = 0.8407
```

[19]:
```
# Question 2 (d) iii.
# Model #3
plot_pred_true_price(yhat_three, y)
```



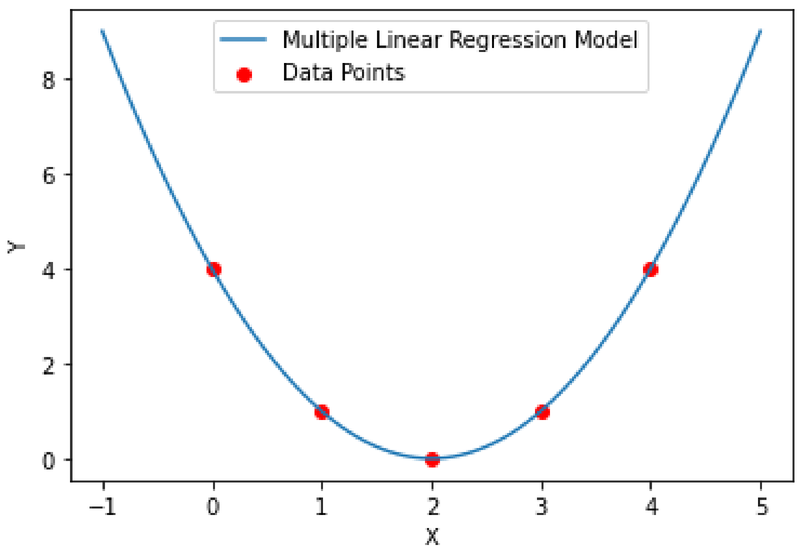
Please continue to the next page for Question 3 (a) and (b).

Question 3 (a) Solution:

No, the equivalence between the coefficient of determination (R^2) and the square of the sample correlation does not extend to multiple linear regression. Let's now discuss a simple example to explain this. Suppose I have one feature X and one target variable Y, and I have 5 data points with their (X, Y) coordinates as (0, 4), (1, 1), (2, 0), (3, 1), and (4, 4) respectively. I manually design a model to fit the data, and the model equation is shown below.

$$\hat{Y} = \widehat{w_0} + \widehat{w_1}X + \widehat{w_2}X^2 = 4 + (-4)X + 1X^2$$

This model has only 1 feature X, but 2 predictors X and X-squared, so this is a multiple linear regression model. The plot below shows how my multiple linear regression model performs given the 5 data points.



We can see that all 5 data points lie precisely on the quadratic curve (prediction) of my multiple linear regression model. Therefore, we will have the coefficient of determination equal to 1.0 ($R^2 = 1$ or 100%). However, the square of the sample correlation between X and Y is equal to 0 ($r^2 = 0$), because the relationship between X and Y is completely nonlinear (quadratic instead). Therefore, we can conclude that the equivalence between R^2 and r^2 does not extend to multiple linear regression.

Note: The method used to solve Question 3 (a) has been discussed with and approved by professor Tantum.

Question 3 (b) Solution:

From the theoretical perspective, R^2 represents the percentage of the variation in the response (target variable) that could be explained by the predictor variables in either the simple linear regression model or the multiple linear regression model. This means that both a linear relationship and a nonlinear (such as quadratic, cubic, or other curvilinear) relationship between features and the target could be regarded as being explainable.

As for r or r^2, however, it is used to measure the strength of a linear relationship between 2 variables, such as the feature X and the target Y. If the value of r or r^2 is very close to 0, it does not show that there is no relationship between X and Y, it only shows that there is no linear relationship between X and Y.

In conclusion, R^2 could provide more information about potentially nonlinear relationship between the features and the target, which is what r or r^2 could not provide.