Assignment8

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
```

How Much is Your Car Worth?

Data about the retail price of 2005 General Motors cars can be found in `car_data.csv`.

The columns are:

1. Price: suggested retail price of the used 2005 GM car in excellent condition.
2. Mileage: number of miles the car has been driven
3. Make: manufacturer of the car such as Saturn, Pontiac, and Chevrolet
4. Model: specific models for each car manufacturer such as Ion, Vibe, Cavalier
5. Trim (of car): specific type of car model such as SE Sedan 4D, Quad Coupe 2D
6. Type: body type such as sedan, coupe, etc.
7. Cylinder: number of cylinders in the engine
8. Liter: a more specific measure of engine size
9. Doors: number of doors
10. Cruise: indicator variable representing whether the car has cruise control (1 = cruise)
11. Sound: indicator variable representing whether the car has upgraded speakers (1 = upgraded)
12. Leather: indicator variable representing whether the car has leather seats (1 = leather)

Tasks, Part 1

1. Find the linear regression equation for mileage vs price.
2. Chart the original data and the equation on the chart.
3. Find the equation's R^2 score (use the `.score` method) to determine whether the

equation is a good fit for this data. (0.8 and greater is considered a strong correlation.)

Tasks, Part 2

1. Use mileage, cylinders, liters, doors, cruise, sound, and leather to find the linear regression equation.
2. Find the equation's R^2 score (use the `.score` method) to determine whether the

equation is a good fit for this data. (0.8 and greater is considered a strong correlation.) 3. Find the combination of the factors that is the best predictor for price.

Tasks, Hard Mode

1. Research dummy variables in scikit-learn to see how to use the make, model, and body type.
2. Find the best combination of factors to predict price.

In [2]:
```
df = pd.read_csv("C:/Users/Karthi/Downloads/car_data.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Price	Mileage	Make	Model	Trim	Type	Cylinder	Liter	Doors	Cruise	Sound	Leather
0	17314.103129	8221	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	1	1
1	17542.036083	9135	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	1	0
2	16218.847862	13196	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	1	0
3	16336.913140	16342	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	0	0
4	16339.170324	19832	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	0	1

In [4]:
```
df.shape
```

Out[4]:
```
(804, 12)
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Price       0
Mileage     0
Make        0
Model       0
Trim        0
Type        0
Cylinder    0
Liter       0
Doors       0
Cruise      0
Sound       0
Leather     0
dtype: int64
```

In [6]:
```
df = df.drop_duplicates()
df.shape
```

Out[6]:
```
(804, 12)
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Price       float64
Mileage       int64
Make         object
Model        object
Trim         object
Type         object
Cylinder      int64
Liter       float64
Doors         int64
Cruise        int64
Sound         int64
Leather       int64
dtype: object
```

In [8]:
```
df.describe()
```

Out[8]:

	Price	Mileage	Cylinder	Liter	Doors	Cruise	Sound	Leather
count	804.000000	804.000000	804.000000	804.000000	804.000000	804.000000	804.000000	804.000000
mean	21343.143767	19831.934080	5.268657	3.037313	3.527363	0.752488	0.679104	0.723881
std	9884.852801	8196.319707	1.387531	1.105562	0.850169	0.431836	0.467111	0.447355
min	8638.930895	266.000000	4.000000	1.600000	2.000000	0.000000	0.000000	0.000000
25%	14273.073870	14623.500000	4.000000	2.200000	4.000000	1.000000	0.000000	0.000000
50%	18024.995019	20913.500000	6.000000	2.800000	4.000000	1.000000	1.000000	1.000000
75%	26717.316636	25213.000000	6.000000	3.800000	4.000000	1.000000	1.000000	1.000000
max	70755.466717	50387.000000	8.000000	6.000000	4.000000	1.000000	1.000000	1.000000

In [9]:
```
iqr = df['Mileage'].quantile(0.75) - df['Mileage'].quantile(0.25)
upper_threshold = df['Mileage'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Mileage'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[9]:
```
(41097.25, -1260.75)
```

In [10]:
```
df.Mileage = df.Mileage.clip(-1260.75,41097.25)
```

In [11]:
```
df.shape
```

Out[11]:
```
(804, 12)
```

In [12]:
```
iqr = df['Cylinder'].quantile(0.75) - df['Cylinder'].quantile(0.25)
upper_threshold = df['Cylinder'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Cylinder'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[12]:
```
(9.0, 1.0)
```

In [13]:
```
iqr = df['Liter'].quantile(0.75) - df['Liter'].quantile(0.25)
upper_threshold = df['Liter'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Liter'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[13]:
```
(6.199999999999999, -0.1999999999999993)
```

In [14]:
```
iqr = df['Doors'].quantile(0.75) - df['Doors'].quantile(0.25)
upper_threshold = df['Doors'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Doors'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[14]:
```
(4.0, 4.0)
```

In [15]:
```
df.Doors = df.Doors.clip(4.0,4.0)
df.shape
```

Out[15]:
```
(804, 12)
```

In [16]:
```
iqr = df['Cruise'].quantile(0.75) - df['Cruise'].quantile(0.25)
upper_threshold = df['Cruise'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Cruise'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[16]: (1.0, 1.0)

In [17]:
```
df.Cruise = df.Cruise.clip(1.0,1.0)
df.shape
```

Out[17]: (804, 12)

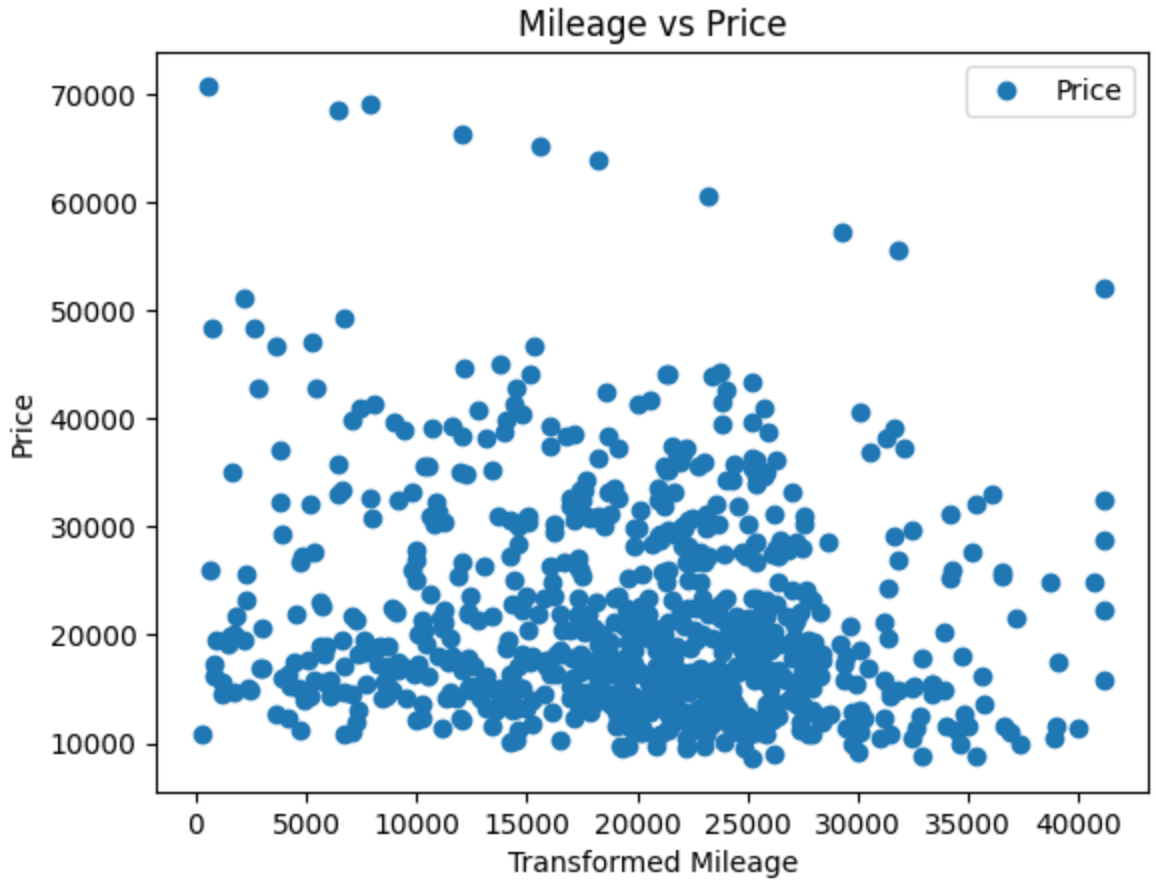
In [18]:
```
iqr = df['Sound'].quantile(0.75) - df['Sound'].quantile(0.25)
upper_threshold = df['Sound'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Sound'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[18]: (2.5, -1.5)

In [19]:
```
iqr = df['Leather'].quantile(0.75) - df['Leather'].quantile(0.25)
upper_threshold = df['Leather'].quantile(0.75) + (1.5 * iqr)
lower_threshold = df['Leather'].quantile(0.25) - (1.5 * iqr)
upper_threshold, lower_threshold
```

Out[19]: (2.5, -1.5)

In [20]:
```
import numpy as np
df['transformed'] = (df['Mileage']) # transformation
#df.groupby('Mileage')['Price'].mean().plot()
df.plot(x='transformed', y='Price', style='o')
plt.title('Mileage vs Price')
plt.xlabel('Transformed Mileage')
plt.ylabel('Price')
plt.show()
df[['transformed','Price']].corr()
```

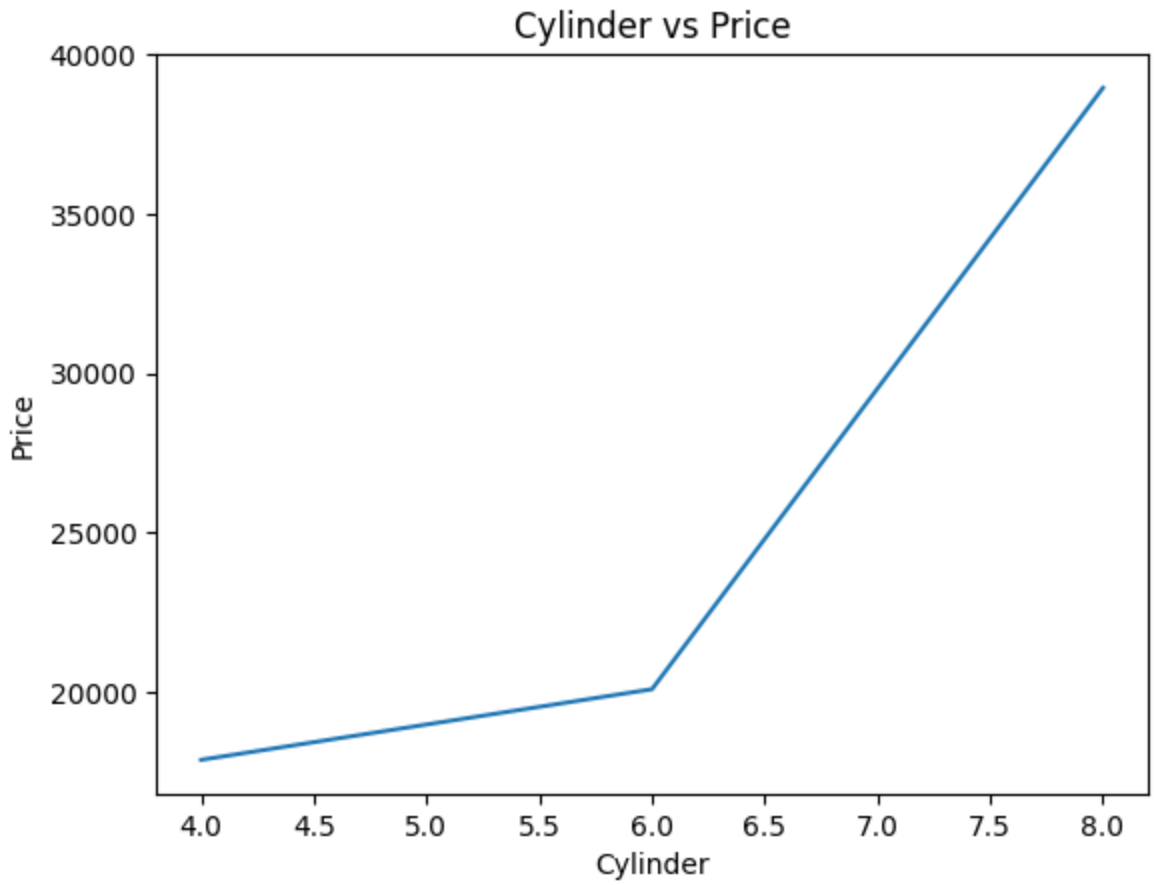


Out[20]:

	transformed	Price
transformed	1.000000	-0.146283
Price	-0.146283	1.000000

1. There is no linear relationship between Mileage and Price 2) Have tried all trasnformations - still not able to see a good linear relationship 3) Have decided to drop the feature

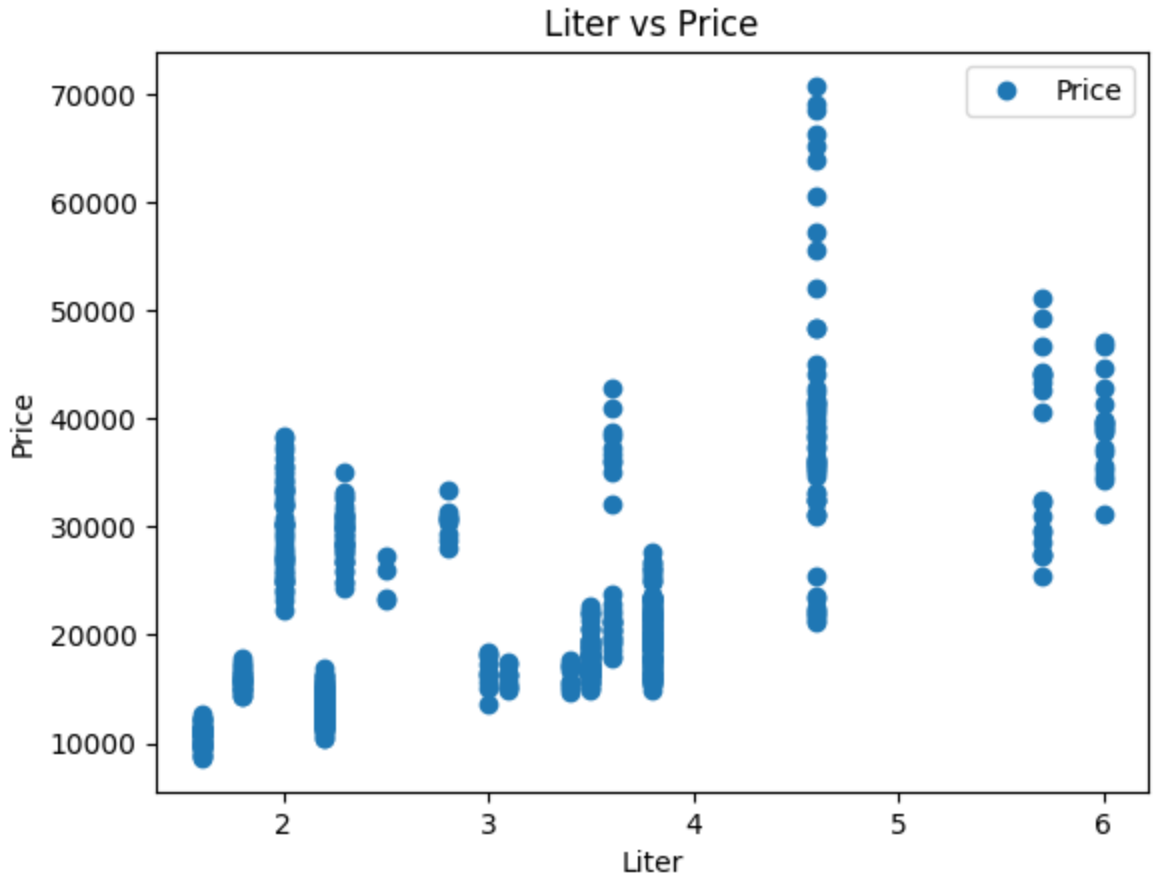
In [21]:
```
df.groupby('Cylinder')['Price'].mean().plot()
#df.plot(x='Cylinder', y='Price', style='o')
plt.title('Cylinder vs Price')
plt.xlabel('Cylinder')
plt.ylabel('Price')
plt.show()
df[['Cylinder','Price']].corr()
```



Out[21]:

	Cylinder	Price
Cylinder	1.000000	0.569086
Price	0.569086	1.000000

In [22]:
```
df.plot(x='Liter', y='Price', style='o')
plt.title('Liter vs Price')
plt.xlabel('Liter')
plt.ylabel('Price')
plt.show()
df[['Liter','Price']].corr()
```

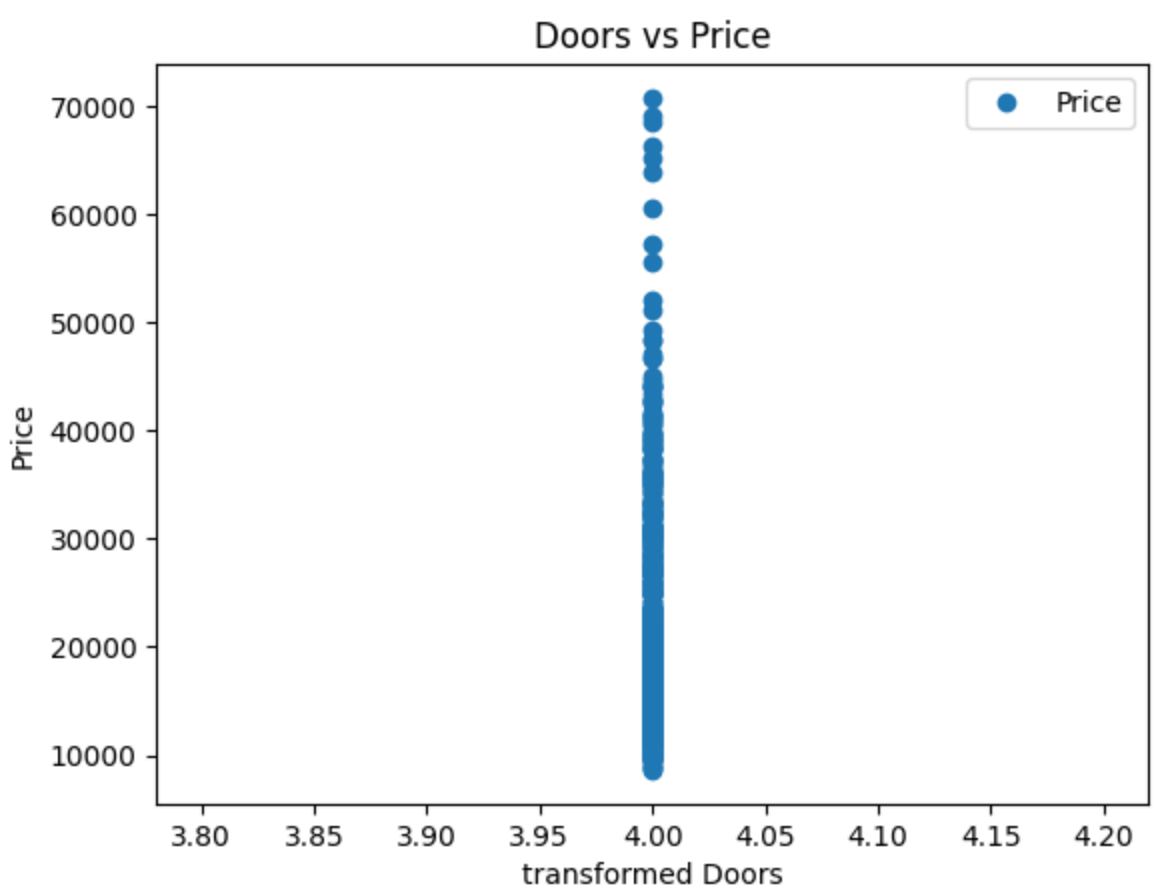


Out[22]:

	Liter	Price
Liter	1.000000	0.558146
Price	0.558146	1.000000

In [23]:

```
import numpy as np
df['transformed'] = (df['Doors']) # transformation
df.plot(x='Doors', y='Price', style='o')
plt.title('Doors vs Price')
plt.xlabel('transformed Doors')
plt.ylabel('Price')
plt.show()
df[['transformed','Price']].corr()
```

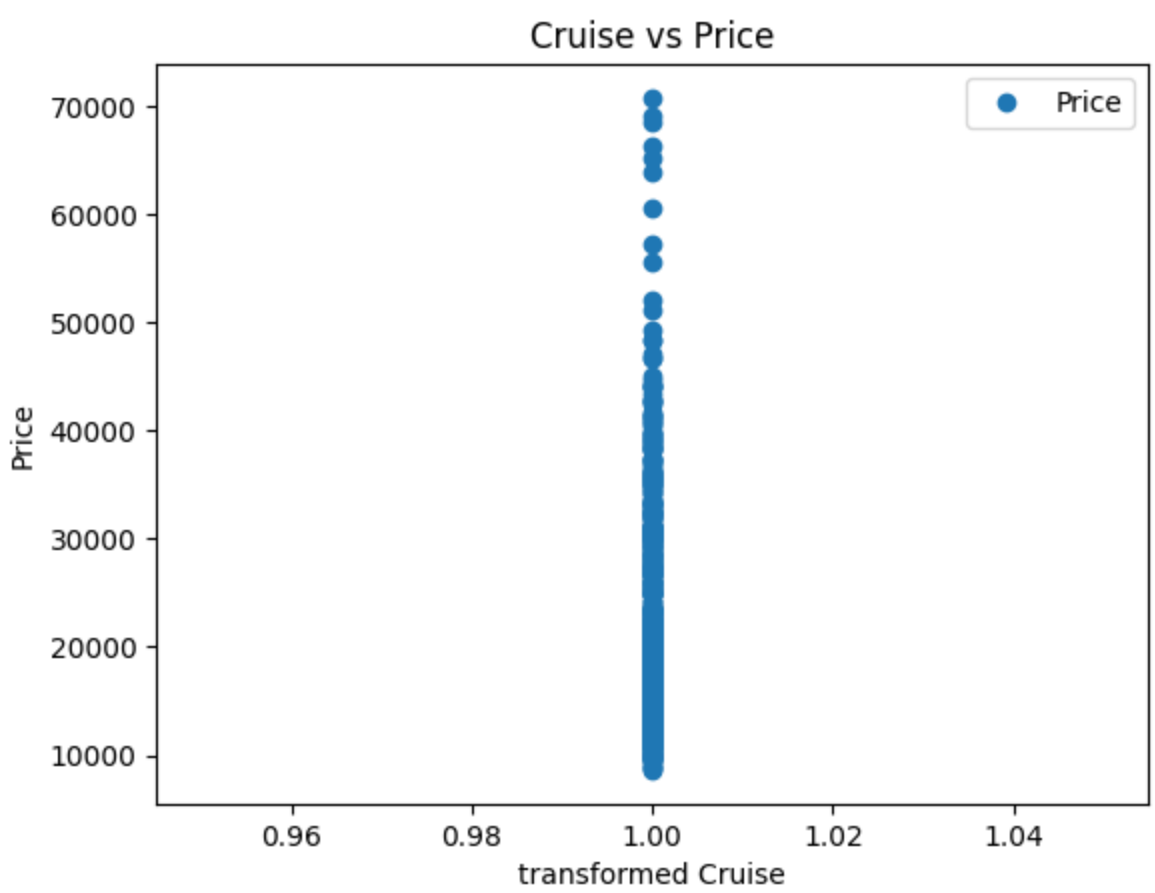


Out[23]:

	transformed	Price
transformed	NaN	NaN
Price	NaN	1.0

1. There is no linear relationship between Doors and Price 2) Have tried all trasnformations - still not able to see a good linear relationship 3) Have decided to drop the feature

In [24]:
```
import numpy as np
df['transformed'] = (df['Cruise']) # transformation
df.plot(x='Cruise', y='Price', style='o')
plt.title('Cruise vs Price')
plt.xlabel('transformed Cruise')
plt.ylabel('Price')
plt.show()
df[['transformed','Price']].corr()
```

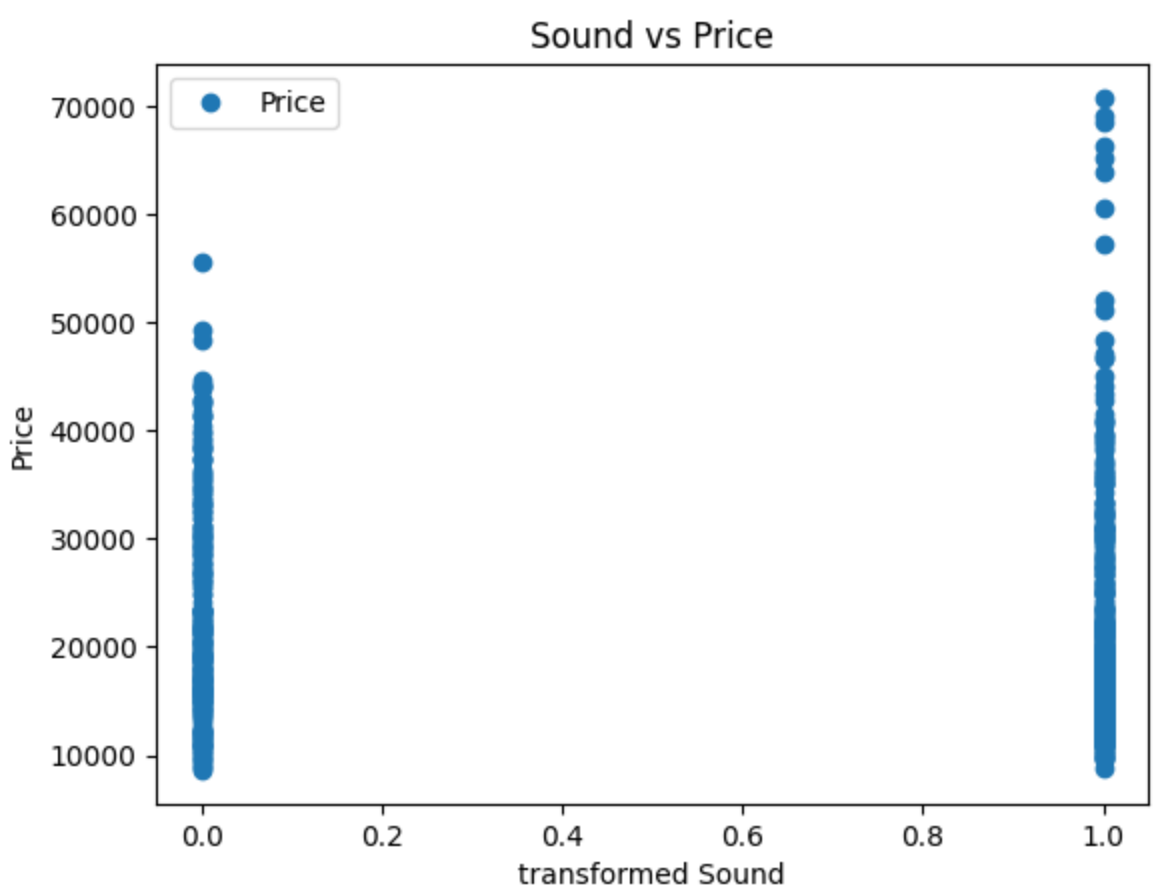


Out[24]:

	transformed	Price
transformed	NaN	NaN
Price	NaN	1.0

1. There is no linear relationship between Cruise and Price 2) Have tried all trasnformations - still not able to see a good linear relationship 3) Have decided to drop the feature

In [25]:
```
import numpy as np
df['transformed'] = (df['Sound']) # transformation
df.plot(x='Sound', y='Price', style='o')
plt.title('Sound vs Price')
plt.xlabel('transformed Sound')
plt.ylabel('Price')
plt.show()
df[['transformed','Price']].corr()
```

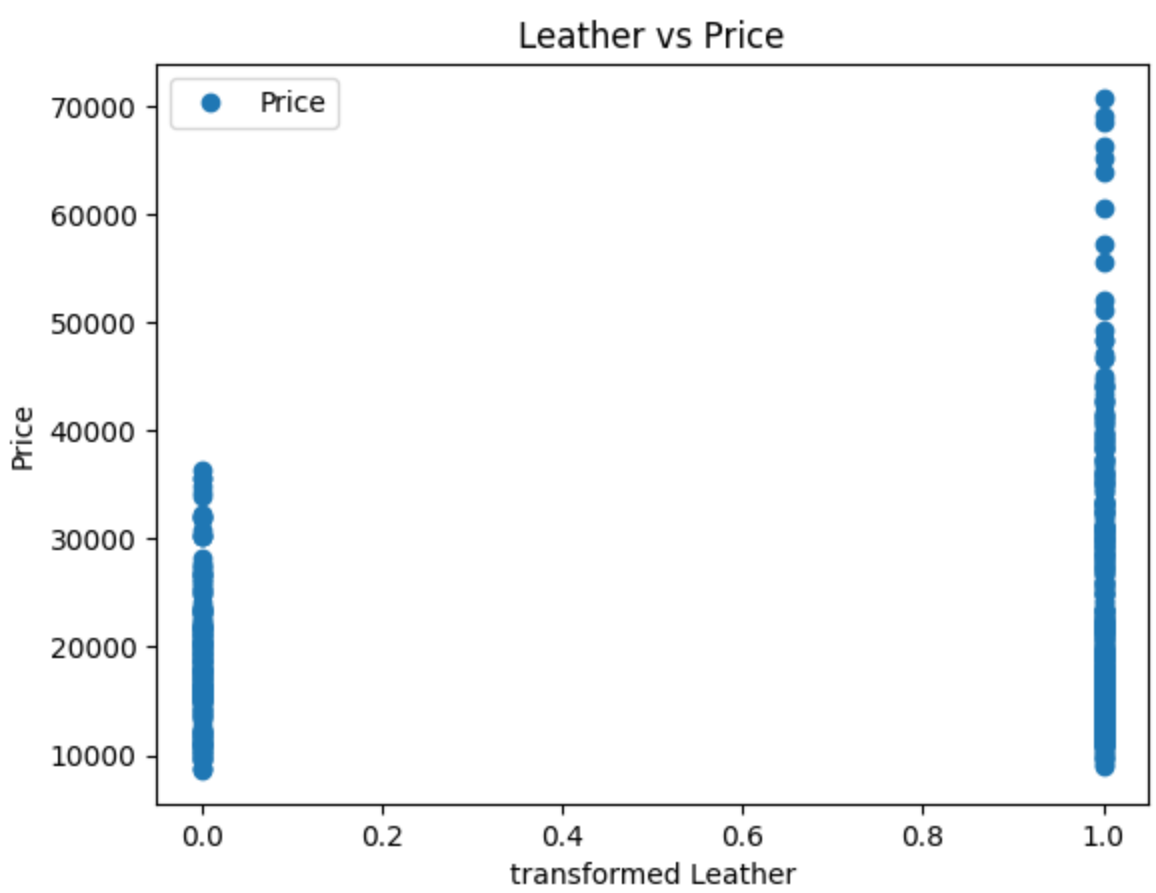


Out[25]:

	transformed	Price
transformed	1.000000	-0.124348
Price	-0.124348	1.000000

1. There is no linear relationship between Sound and Price 2) Have tried all trasnformations - still not able to see a good linear relationship 3) Have decided to drop the feature

In [26]:
```
import numpy as np
df['transformed'] = (df['Leather']) # transformation
df.plot(x='Leather', y='Price', style='o')
plt.title('Leather vs Price')
plt.xlabel('transformed Leather')
plt.ylabel('Price')
plt.show()
df[['transformed','Price']].corr()
```



Out[26]:

	transformed	Price
transformed	1.000000	0.157197
Price	0.157197	1.000000

1. There is no linear relationship between Leather and Price 2) Have tried all trasnformations - still not able to see a good linear relationship 3) Have decided to drop the feature

In [30]:
```
X = df[['Cylinder','Liter','Doors','Cruise','Sound','Leather']].values
Y = df['Price'].values
```

In [31]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

In [32]:
```
from sklearn.preprocessing import StandardScaler ## standrard scalig
scaler = StandardScaler() #initialise to a variable
scaler.fit(X_train) # we are finding the values of mean and sd from the td
X_train_scaled = scaler.transform(X_train) # fit (mean, sd) and then transform the training data
X_test_scaled = scaler.transform(X_test) # transform the test data
```

Model training

In [33]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train_scaled, y_train)
```

Out[33]:
```
▾ LinearRegression
LinearRegression()
```

In [42]:
```
coeff_df = pd.DataFrame(regressor.coef_,['Mileage','Cylinder','Liter','Cruise','Sound','Leather'
y_pred = regressor.predict(X_test_scaled)
coeff_df
```

Out[42]:

	Coefficient
Mileage	3.968821e+03
Cylinder	1.781515e+03
Liter	6.821210e-13
Cruise	0.000000e+00
Sound	-7.859006e+02
Leather	1.395426e+03

In [36]:
```
regressor.intercept_ # c
```

Out[36]:
```
21554.11855241913
```

In [53]:
```
df1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df1
```

Out[53]:

	Actual	Predicted
0	30274.710575	13391.276836
1	15595.884133	24315.814121
2	18800.958899	24473.631062
3	18678.414123	21806.162483
4	42741.523666	26303.200690
...	...	...
156	16300.465240	24473.631062
157	11080.516378	16847.830122
158	20047.951361	21806.162483
159	16825.190882	24473.631062
160	21020.836777	24947.081886

161 rows × 2 columns

In [68]:
```
from sklearn import metrics
print('R2- SCORE:', metrics.r2_score(y_test,y_pred))
```

```
R2- SCORE: 0.2036939909955311
```

In [38]:
```
df.columns
```

Out[38]:
```
Index(['Price', 'Mileage', 'Make', 'Model', 'Trim', 'Type', 'Cylinder',
       'Liter', 'Doors', 'Cruise', 'Sound', 'Leather', 'transformed'],
      dtype='object')
```

In [72]:
```
Car_Features = ['Mileage', 'Cylinder','Liter','Cruise','Sound','Leather']
Car_Price = ['Price']
X = df[Car_Features]
Y = df[Car_Price]
```

In [74]:
```
from itertools import combinations

combos = []

for i in range(1,7):
    combos.append(combinations(Car_Features, i))

check_score = 0
for combin in combos:
    for groupx in combin:
        groupx = list(groupx)
        x = df[groupx]

        model = linear_model.LinearRegression(fit_intercept = False)

        model = model.fit(X,Y)
        model_score = model.score(X , Y)
        if check_score <model.score(X, Y):
            check_score = model.score(X, Y)
            maxgroup = groupx

print(maxgroup,'model_score=', check_score)
```

```
['Mileage'] model_score= 0.42881262611689586
```

In [55]:
```
df = pd.read_csv("C:/Users/Karthi/Downloads/car_data.csv")
```

In [56]:
```
df.head()
```

Out[56]:

	Price	Mileage	Make	Model	Trim	Type	Cylinder	Liter	Doors	Cruise	Sound	Leather
0	17314.103129	8221	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	1	1
1	17542.036083	9135	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	1	0
2	16218.847862	13196	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	1	0
3	16336.913140	16342	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	0	0
4	16339.170324	19832	Buick	Century	Sedan 4D	Sedan	6	3.1	4	1	0	1

In [58]:
```
dummy = pd.get_dummies(df[['Make','Model','Type']])
dummy.head()
```

Out[58]:

	Make_Buick	Make_Cadillac	Make_Chevrolet	Make_Pontiac	Make_SAAB	Make_Saturn	Model_9-2X AWD	Model_9_3
0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0

5 rows × 43 columns

In [59]:
```
df_dummies = pd.concat([df['Price'], dummy], axis = 1)
df_dummies.head()
```

Out[59]:

	Price	Make_Buick	Make_Cadillac	Make_Chevrolet	Make_Pontiac	Make_SAAB	Make_Saturn	Model_9-2X AWD
0	17314.103129	1	0	0	0	0	0	0
1	17542.036083	1	0	0	0	0	0	0
2	16218.847862	1	0	0	0	0	0	0
3	16336.913140	1	0	0	0	0	0	0
4	16339.170324	1	0	0	0	0	0	0

5 rows × 44 columns

In [60]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
for i in ['Make', 'Model', 'Trim', 'Type']:
    df[i] = label_encoder.fit_transform(df[i])
df.head()
```

Out[60]:

	Price	Mileage	Make	Model	Trim	Type	Cylinder	Liter	Doors	Cruise	Sound	Leather
0	17314.103129	8221	0	10	44	3	6	3.1	4	1	1	1
1	17542.036083	9135	0	10	44	3	6	3.1	4	1	1	0
2	16218.847862	13196	0	10	44	3	6	3.1	4	1	1	0
3	16336.913140	16342	0	10	44	3	6	3.1	4	1	0	0
4	16339.170324	19832	0	10	44	3	6	3.1	4	1	0	1

In [61]:
```
Car_Features = list(df.columns)[1:]
X = df.iloc[:, 1:]
Y = df.Price
```

In [69]:
```
from itertools import combinations

combos = []

for i in range(1,7):
    combos.append(combinations(Car_Features,i))

check_score = 0
for combin in combos:
    for groupx in combin:
        groupx = list(groupx)
        x =df[groupx]

        model = linear_model.LinearRegression(fit_intercept = False)

        model = model.fit(X,Y)
        model_score = model.score(X, Y)
        if check_score < model.score(X,Y):
                check_score = model.score(X,Y)
                maxgroup = groupx

print(maxgroup, 'model_score=', check_score)
```

```
['Mileage'] model_score= 0.4994922079608122
```

In [ ]: