College of Professional Studies
Northeastern University San Jose

MPS Analytics

Course: ALY6020

Assignment:

Module 2 – Midweek Project

Submitted on:

October 5, 2023

Submitted to:

Professor: Ahmadi Behzad

Submitted by:

Heejae Roh

Introduction

This dataset contains data about the price and specifications of cars. Specifications are largely divided into numerical variables and categorical variables. Looking at the types of columns, the main data are length, width, etc., which are data about the overall size of the car. Data about the engine consists of numerical data and categorical data, and provides detailed information about the engine.

Based on this data, I will conduct regression with price as the dependent variable. In the process, if necessary, we will perform data cleansing and apply selection methods that can be used in regression. I would also like to determine whether the assumption is satisfied.

Dataset Understanding

The dataset is about car price and specifications. There are 26 columns and 205 entries. This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) its assigned insurance risk rating, (c) its normalized losses in use as compared to other cars (UC Irvine, n.d.). However, the data we use is the data excluding (c) from the above data. Since the data was donated in 1987, it must be taken into account that it is over 30 years old. Data can be broadly divided into 'price', automobile specifications, and 'symboling' (insurance risk rating).

Description of the variables/features in the dataset.

#	column name	Description
1	**Car_ID**	Unique id of each observation (Interger)
2	**Symboling**	Its assigned insurance risk rating, A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.(Categorical)
3	**carCompany**	Name of car company (Categorical)
4	**fueltype**	Car fuel type i.e gas or diesel (Categorical)
5	**aspiration**	Aspiration used in a car (Categorical)
6	**doornumber**	Number of doors in a car (Categorical)
7	**carbody**	body of car (Categorical)
8	**drivewheel**	type of drive wheel (Categorical)
9	**enginelocation**	Location of car engine (Categorical)
10	**wheelbase**	Weelbase of car (Numeric)
11	**carlength**	Length of car (Numeric)
12	**carwidth**	Width of car (Numeric)
13	**carheight**	height of car (Numeric)
14	**curbweight**	The weight of a car without occupants or baggage. (Numeric)
15	**enginetype**	Type of engine. (Categorical)
16	**cylindernumber**	cylinder placed in the car (Categorical)
17	**enginesize**	Size of car (Numeric)
18	**fuelsystem**	Fuel system of car (Categorical)
19	**boreratio**	Boreratio of car (Numeric)
20	**stroke**	Stroke or volume inside the engine (Numeric)

21	**compressionratio**	compression ratio of car (Numeric)
22	**horsepower**	Horsepower (Numeric)
23	**peakrpm**	car peak rpm (Numeric)
24	**citympg**	Mileage in city (Numeric)
25	**highwaympg**	Mileage on highway (Numeric)
26	**price**	Dependent Variable. Price of car (Numeric)

Headtail of Dataset 1

	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
…				…	
202	203	-1	volvo 244dl	gas	std
203	204	-1	volvo 246	diesel	turbo
204	205	-1	volvo 264gl	gas	turbo

Headtail of Dataset 2

	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
…				…	
202	four	sedan	rwd	front	109.1
203	four	sedan	rwd	front	109.1
204	four	sedan	rwd	front	109.1

Headtail of Dataset 3

	…	enginesize	fuelsystem	boreratio	stroke	compressionratio
0		130	mpfi	3.47	2.68	9.0
1		130	mpfi	3.47	2.68	9.0
2		152	mpfi	2.68	3.47	9.0
…	…			…		
202		173	rwd	3.58	2.87	8.8
203		145	rwd	3.01	3.4	23.0
204	…	141	rwd	3.78	3.15	9.5

Headtail of Dataset 4

	horsepower	peakrpm	citympg	highwaympg	price
0	111	5000	21	27	13495
1	111	5000	21	27	16500
2	154	5000	19	26	16500
…	…			…	
202	134	5500	18	23	21485
203	106	4800	26	27	22470
204	114	5400	19	25	22625

Data Cleansing

1. Changing one of CarName from ‘audi 100 ls’ to ‘audi 100ls’ to clarify category of CarName. However, there were cases where similar car names had different prices. Therefore, I thought that the specifications could be different regardless of the car's name, so I did not change the CarName as much as possible.
2. It turns out that there are no missing values. Because the entity of data was not large, we tried to analyze using existing data as much as possible and as it is.
3. To facilitate analysis when modifying columns, the column_name was modified by applying lowercase to all columns.

Exploratory Data Analysis

Descriptive Analysis of Dataset 1

	car_id	symbolling	wheelbase	carlength	carwidth
count	205	205	205	205	205
mean	103.00	0.83	98.76	174.05	65.91
std	59.32	1.25	6.02	12.34	2.15
min	1.00	-2.0	86.6	141.1	60.3
25%	52.00	0.00	94.50	166.30	64.1
50%	103.00	1.00	97.00	173.20	65.5
75%	154.00	2.00	102.40	183.10	66.90
max	205.00	3.00	120.9	208.1	72.3

Descriptive Analysis of Dataset 2

	carheight	curbweight	enginesize	boreratio	stroke
count	205	205	205	205	205
mean	53.72	2555.57	126.91	3.33	3.26
std	2.44	520.68	41.64	0.27	0.31
min	47.80	1488.00	61.00	2.54	2.07
25%	52.00	2145.00	97.00	3.15	3.11
50%	54.10	2414.00	120.00	3.31	3.29
75%	55.5	2935.00	141.00	3.58	3.41
max	59.8	4066.00	326.00	3.94	4.17

1. All variables have 205 counts. Among them, 'car_id' is an automatically assigned number from 1 to 205. We can see that there are a total of 205 'car_id'.
2. The part about car specifications, from 'wheelbase' to 'carheight', can be understood in terms of length. Structurally, wheelbase refers to the distance between the two wheels when viewed from the side of the car. Therefore, it is a lower number than ‘carlength’ and a higher number than car width. 'Curbweight' is the weight of the car and has the largest number among the variables that numerically represent the car's appearance.
3. As explained earlier, ‘symboling’ indicates that the car is risky when it is 3, and when it is -3, it indicates that the car is safe. The minimum of ‘symboling’ is -2 and the max is 3. The mean is 0.83, which shows that there are more cars labeled as dangerous than safe cars.
4. 'Engine size' to 'peakrpm' are elements that indicate detailed specifications of the engine part, which is the most important part of the car. 'enginesize' shows a minimum of 61 and a maximum of 326. This range of engine sizes is relatively different by car. ‘Boreratio’ and stroke can directly affect horsepower. At this point, if you search 'boreratio' and ‘stroke’, you will find numbers around 1 which is lower than 3. This shows that bore and stroke were relatively large in the 1980s, before technological development.

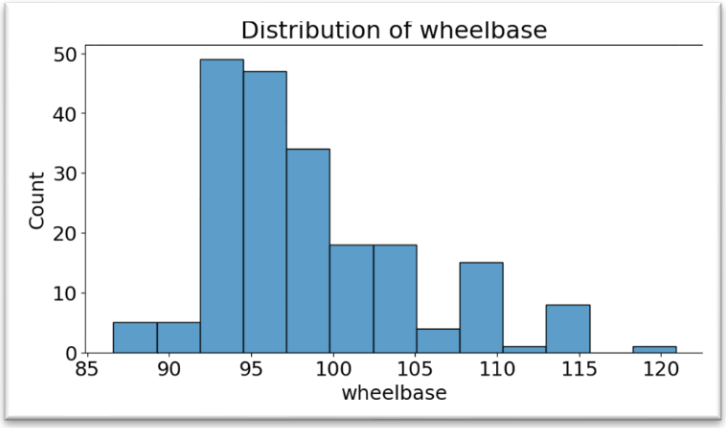
Descriptive Analysis of Dataset 3

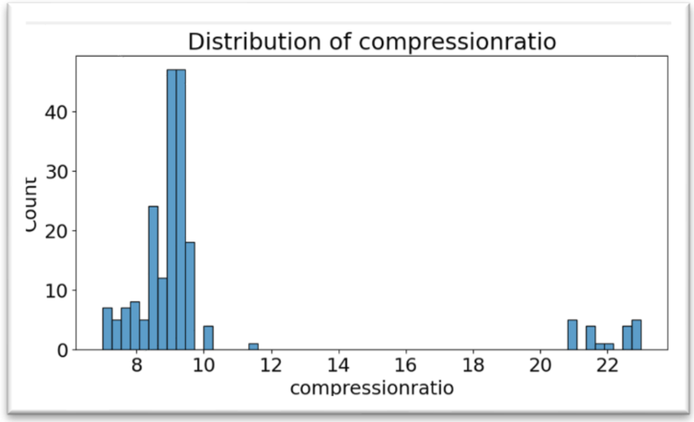
	compressionratio	horsepower	peakrpm	citympg/ highwaympg	price
count	205	205	205	205/ 205	205
mean	10.14	104.12	5125.12	25.22/ 30.75	13276
std	3.97	39.54	476.99	6.54/ 6.88	7988
min	7.00	48.00	4150.00	13.00/ 16.00	5118
25%	8.60	70.00	4800.00	19.00/ 25.00	7788
50%	9.00	95.00	5200.00	24.00/ 30.00	10295
75%	9.40	116.00	5500.00	30.00/ 34.00	16503
max	23.00	288.00	6600.00	49.00/ 54.00	45400

5. 'Compressionratio' refers to how much the engine compresses fuel. Horsepower is the specifications for engines seen so far and refers to the power that the engine has.
6. 'Peakrpm' is the maximum number of revolutions per minute, which is a numerical representation of the engine speed when the car is driven at maximum power.
7. MPG numbers tell you how many miles a car can go on a gallon of fuel (Cazoo, 2023). 'mpg' is divided into when driving at low speed in the city and when driving at high speed on the highway. Higher mpg is mainly seen when driving at high speeds.
8. Lastly, 'price' refers to the final price of the car. The range is 5,118 to 45,400 with a mean of 13,276.

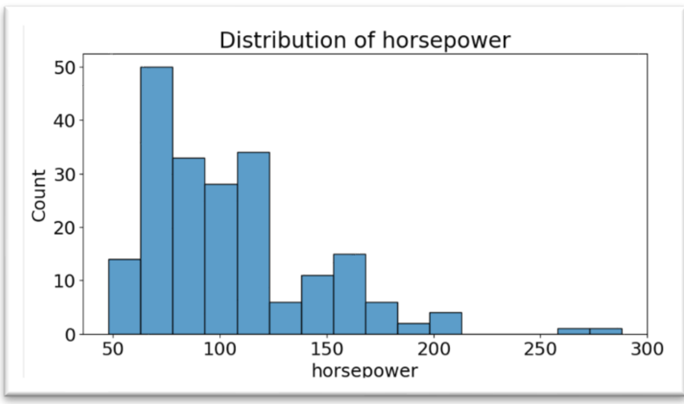
Data Visualizations

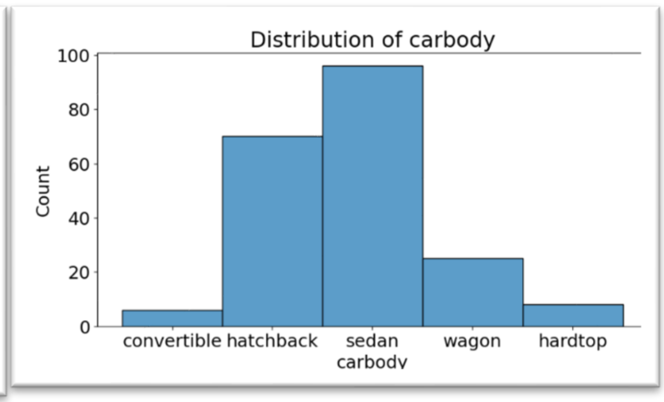
Histograms & Pie chart of primary attribute for linear regression



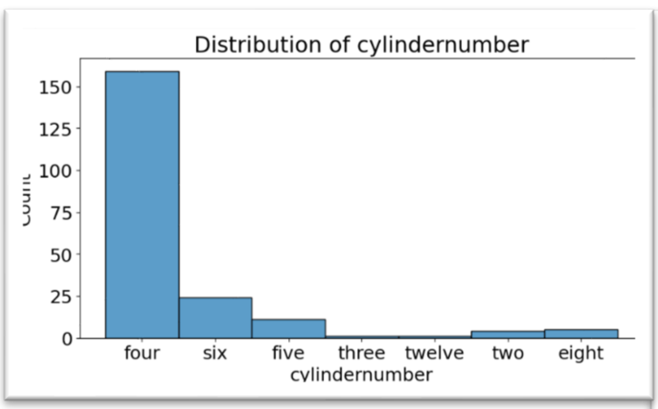


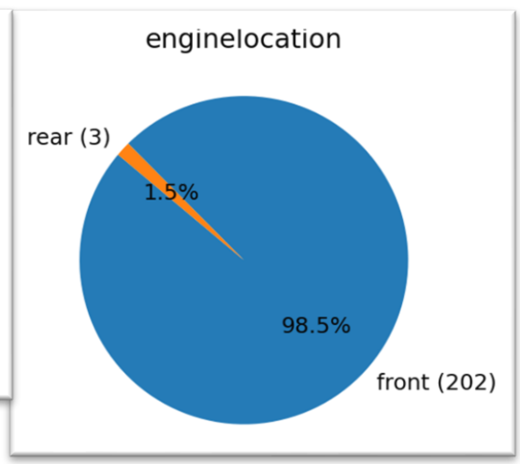
'wheelbase' shows a mode around 92.5 and shows a shape similar to normal distribution compared to other attributes. Most 'compressionratios' are between 0 and 10, but there are parts where values greater than 20 are gathered. It seems difficult to treat this as an outlier.





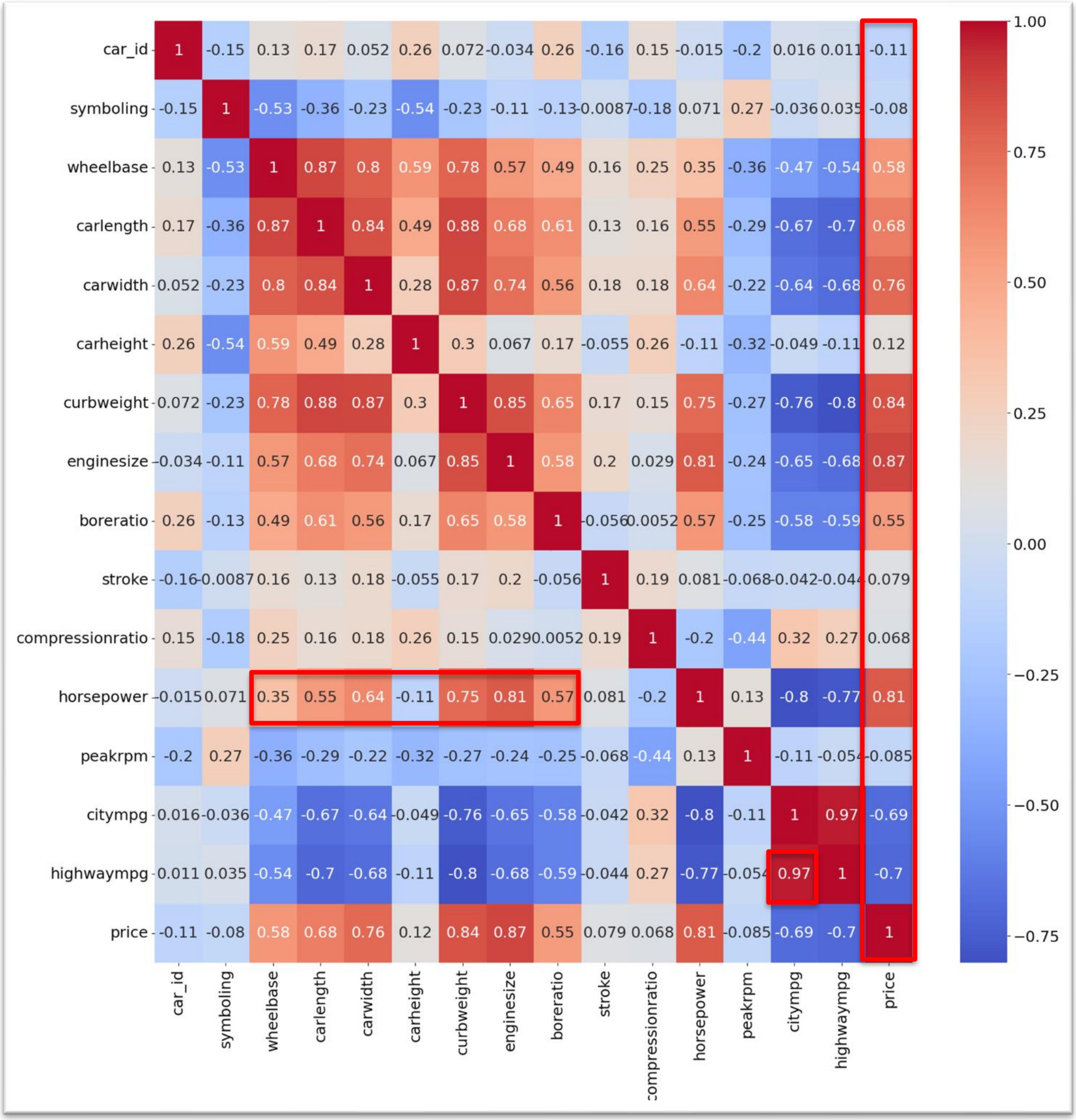
Horsepower appears relatively similar to a normal distribution, but is right skewed, with a small number of counts between 250 and 300. The mode is configured between 50 and 100. Carbody is a categorical variable, with sedan accounting for the largest number of five categories, followed by hatchback.





Although the 'cylindernumber' is in a number but a categorical variable, it is structured almost like a numerical variable. Four is the most, and except for six and five, the count appears to be very low. In this case, we will process it as categorical as shown in the data description. The engine location is imbalanced data, with rear having only 3 pieces of data and accounting for only 1.5%.

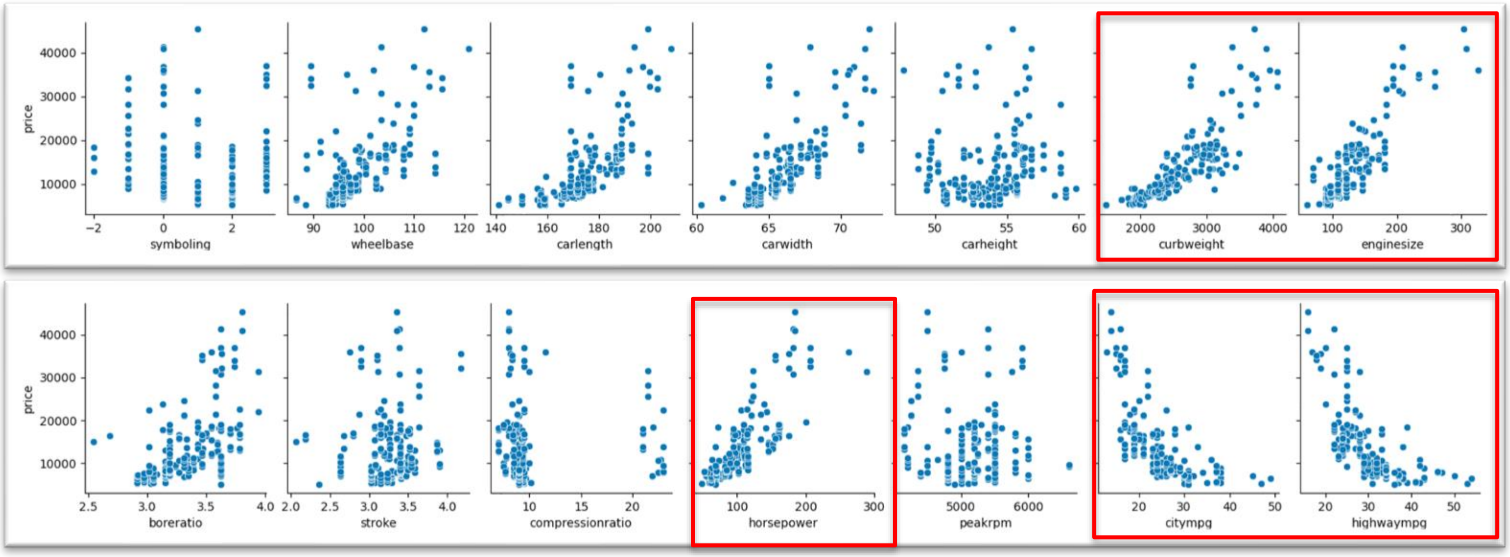
Correlation Matrix



The dependent variable is price, but I thought I should look more closely at multicollinearity in the correlation matrix. This is because the data originally had a data structure that allowed for a high correlation between the engine part, parts based on the length of the car body, and mpg.

I thought that I should focus on analyzing in the future to exclude columns with multicollinearity from the regression model through VIF analysis.

Scatter plots with price



In the scatter plot with price, you can see that ‘curbweight’, ‘enginesize’, and ‘horsepower’ show linearity in positive way, relatively clearly. In the future, I will apply regression directly and compare p-values. Although ‘citympg’ and ‘highwaympg’ show a negative correlation, the two graphs appear almost similar. This means that both attributes may have multicollinearity.

Result of OLS with numerical variables & Interpretation

The ordinary least squares (OLS) algorithm is a method for estimating the parameters of a linear regression model. The OLS algorithm aims to find the values of the linear regression model's parameters (i.e., the coefficients) that minimize the sum of the squared residuals (Prashant, Sahu, 2023). I decided to use the library included in ‘statsmodels.api’. I decided to use OLS here, before performing OLS let’s check the assumptions of OLS.

Assumptions of OLS (Steven, 2021)

A1. The linear regression model is “linear in parameters.”

A2. There is a random sampling of observations.

A3. The conditional mean should be zero.

A4. There is no multi-collinearity (or perfect collinearity).

A5. Spherical errors: There is homoscedasticity and no autocorrelation

A6: Optional Assumption: Error terms should be normally distributed.

During the analysis process, I tried to make sure that the above assumptions were satisfied. To check linearity, I drew a scatterplot. The dataset includes 205 cars’ data, and although its population is unknown, I will assume it was randomly sampled. 'A3. For 'The conditional mean should be zero.', after creating regression, I will check whether the error is independent of value x, or in other words, whether the error shows any pattern or not.

Result 1: with all numerical attributes

```
                     OLS Regression Results with all atributes
==============================================================================
Dep. Variable:                  price   R-squared:                       0.852
Model:                            OLS   Adj. R-squared:                  0.841
Method:                 Least Squares   F-statistic:                     78.05
Date:                Wed, 04 Oct 2023   Prob (F-statistic):           7.97e-71
Time:                        19:16:29   Log-Likelihood:                -1936.7
No. Observations:                 205   AIC:                             3903.
Df Residuals:                     190   BIC:                             3953.
Df Model:                          14
Covariance Type:            nonrobust
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
const            -5.165e+04   1.57e+04     -3.299      0.001   -8.25e+04   -2.08e+04
symboling          285.8829    243.335      1.175      0.242    -194.101     765.867
wheelbase          167.6990    107.450      1.561      0.120     -44.250     379.648
carlength          -94.8179     55.502     -1.708      0.089    -204.297      14.661
carwidth           466.6185    247.995      1.882      0.061     -22.559     955.796
carheight          194.7522    138.223      1.409      0.160     -77.897     467.402
curbweight           1.8776      1.736      1.082      0.281      -1.546       5.301
enginesize         116.7820     13.831      8.443      0.000      89.500     144.064
boreratio         -984.4276   1194.709     -0.824      0.411   -3341.025    1372.169
stroke           -3056.1620    778.046     -3.928      0.000   -4590.881   -1521.443
compressionratio   286.4752     83.425      3.434      0.001     121.918     451.033
horsepower          32.5014     16.264      1.998      0.047       0.420      64.583
peakrpm              2.3582      0.670      3.518      0.001       1.036       3.680
citympg           -286.9397    179.856     -1.595      0.112    -641.710      67.831
highwaympg         191.3036    159.902      1.196      0.233    -124.108     506.716
==============================================================================
Omnibus:                       24.845   Durbin-Watson:                   0.903
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               78.581
Skew:                           0.412   Prob(JB):                     8.64e-18
Kurtosis:                       5.919   Cond. No.                     4.05e+05
==============================================================================

Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
 [2] The condition number is large, 4.05e+05. This might indicate that there are
 strong multicollinearity or other numerical problems.
```

This is the result of regression including all numerical attributes. Based on these results, I will use the best subset method to determine which numerical variables are important factors in determining price based on the P-value. The notes mentioned multicollinearity, and I will check it with VIF (Variance Inflation Factor) values.

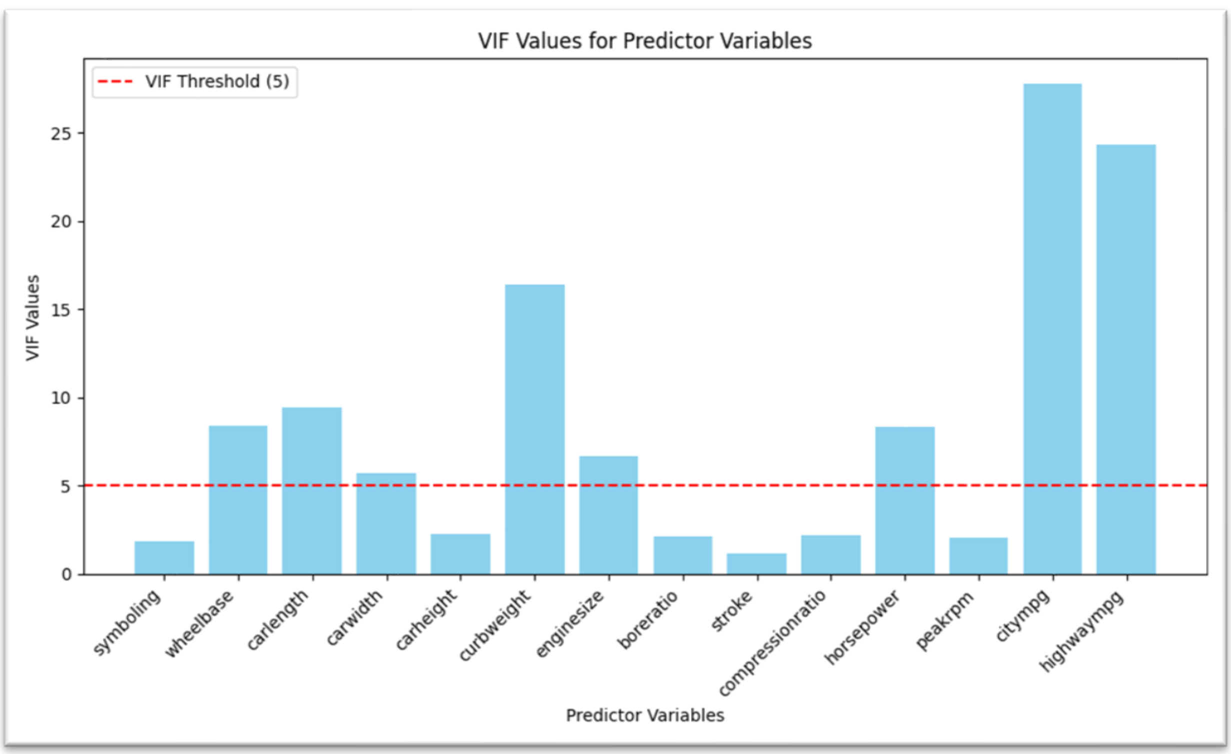
Result2. Best subset method

Attribute number	Best rsquared	Attribute number	Best rsquared
1	0.764129	8	0.846640
2	0.794584	9	0.847444
3	0.818156	10	0.848532
4	0.827328	11	0.849750
5	0.836399	12	0.850434
6	0.842301	13	0.851340
7	0.845324	14	0.851869

With this rough level of results, I first wanted to check the VIF using a model with 6 attributes and a model with 14 attributes. Initially, using all 14 attributes, I decided to proceed with the VIF check and proceed with the analysis by referring to the 6 attributes and removing those with a result of 5 or higher from the VIF.

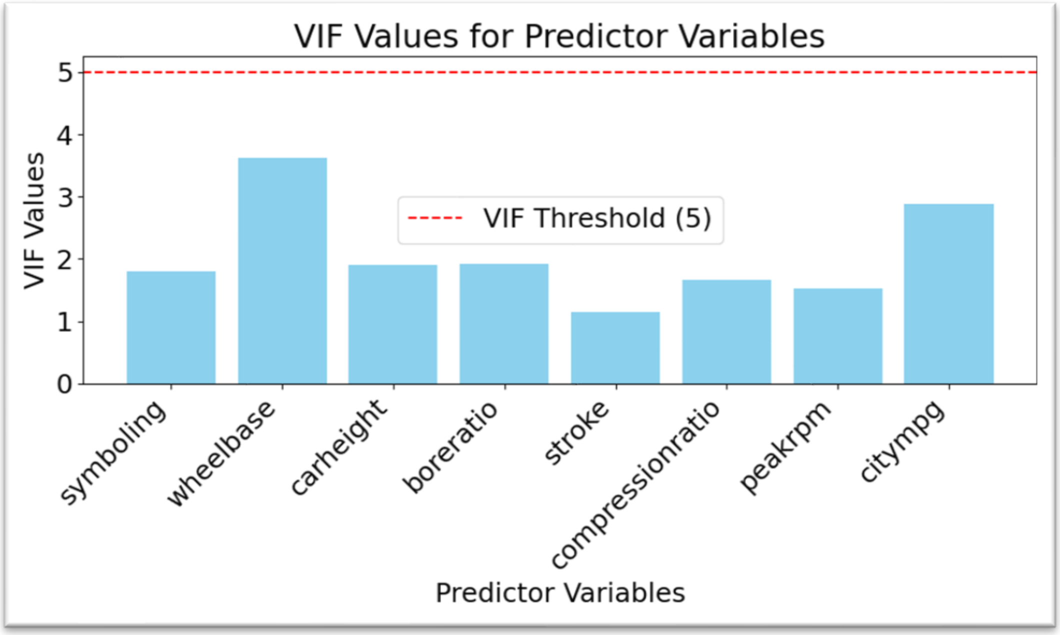
```
                    OLS Regression Results with 6 attributes
==============================================================================
Dep. Variable:                  price   R-squared:                       0.842
Model:                            OLS   Adj. R-squared:                  0.838
Method:                 Least Squares   F-statistic:                     176.3
Date:                Wed, 04 Oct 2023   Prob (F-statistic):           1.38e-76
Time:                        19:39:31   Log-Likelihood:                -1943.1
No. Observations:                 205   AIC:                             3900.
Df Residuals:                     198   BIC:                             3924.
Df Model:                           6
Covariance Type:            nonrobust
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
const             -6.189e+04   1.08e+04     -5.705      0.000   -8.33e+04   -4.05e+04
carwidth            777.2925    164.551      4.724      0.000     452.795    1101.790
enginesize          118.3319     12.495      9.470      0.000      93.691     142.972
stroke            -2932.6397    756.220     -3.878      0.000   -4423.919   -1441.360
compressionratio    266.6884     67.795      3.934      0.000     132.996     400.381
horsepower           39.0894     12.587      3.106      0.002      14.267      63.911
peakrpm               2.2804      0.632      3.611      0.000       1.035       3.526
==============================================================================
Omnibus:                       20.904   Durbin-Watson:                   0.907
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               62.065
Skew:                           0.330   Prob(JB):                     3.33e-14
Kurtosis:                       5.614   Cond. No.                     2.48e+05
==============================================================================
```

VIF histogram with all attributes



I re-checked the VIF by removing attributes that were not included in the 6 best subset modeling and had high VIF. At first, I removed 'carlength', 'carwidth', 'curbweight', 'enginesize', 'highwaympg', and 'price'. I checked VIF again without removed columns

VIF histogram after removing attributes based on previous VIF histogram

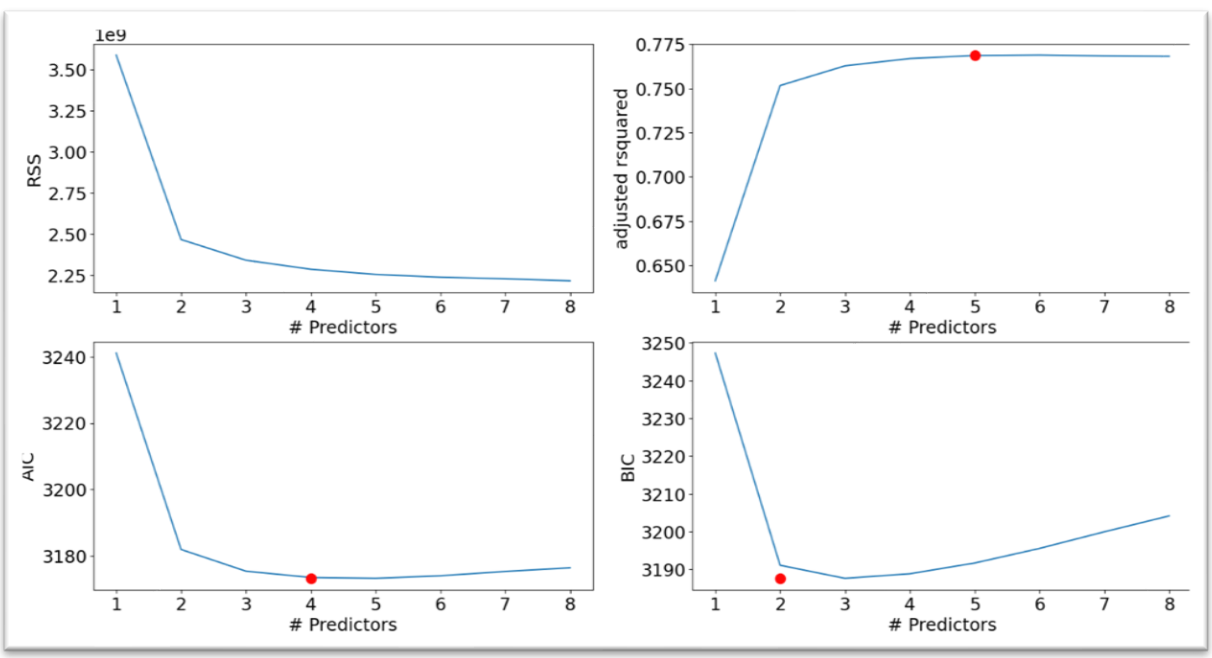


'Wheelbase' appears to have a moderately high VIF, and the above attributes appear to have relatively little multicollinearity. I decided to construct the result once again using these attributes as numerical value candidates for the final regression.

Best subset method with 8 numerical attributes without VIF

Attribute number	Best rsquared	Attribute number	Best rsquared
1	0.643412	5	0.775694
2	0.754648	6	0.777372
3	0.767114	7	0.778316
4	0.772582	8	0.779529

How to choose number of variables



I finally decided on 4 as the number of attributes based on this graph and R-squared values. The attributes are 'symbolling', 'wheelbase', 'compressionratio', and 'horsepower'.

Result of OLS with dummy variables & Interpretation

Dummy Explanatory Variable: When one or more of the explanatory variables is a dummy variable but the dependent variable is not a dummy, the OLS framework is still valid (ubc, 2023). I added categorical variables as dummy variables I selected earlier. It is applied to regression using a dummy variable.

```
    OLS Regression Results with a few numeric and all categorical attributes
     ==============================================================================
     Dep. Variable:                  price   R-squared:                       0.913
     Model:                            OLS   Adj. R-squared:                  0.898
     Method:                 Least Squares   F-statistic:                     60.55
     Date:                Tue, 03 Oct 2023   Prob (F-statistic):           1.19e-76
     Time:                        23:41:43   Log-Likelihood:                -1882.7
     No. Observations:                 205   AIC:                             3827.
     Df Residuals:                     174   BIC:                             3930.
     Df Model:                          30
     Covariance Type:            nonrobust
==========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   -1.117e+04   8509.342     -1.313      0.191    -2.8e+04    5621.793
symboling                 436.9001    239.595      1.823      0.070     -35.986     909.786
wheelbase                 368.2513     72.850      5.055      0.000     224.468     512.035
compressionratio         -447.9250    513.594     -0.872      0.384   -1461.601     565.751
horsepower                114.1498     17.645      6.469      0.000      79.325     148.975
aspiration_turbo         -664.2139    811.928     -0.818      0.414   -2266.710     938.282
carbody_hardtop         -5251.6017   1483.797     -3.539      0.001   -8180.160   -2323.044
carbody_hatchback       -5091.1784   1319.446     -3.859      0.000   -7695.358   -2486.999
carbody_sedan           -4097.1392   1367.687     -2.996      0.003   -6796.532   -1397.747
carbody_wagon           -4955.1510   1506.576     -3.289      0.001   -7928.668   -1981.634
drivewheel_fwd           -555.3629   1062.952     -0.522      0.602   -2653.302    1542.577
drivewheel_rwd           1124.8300   1234.218      0.911      0.363   -1311.135    3560.795
enginelocation_rear      6387.6779   2670.549      2.392      0.018    1116.838    1.17e+04
enginetype_dohcv        -1.705e+04   4605.650     -3.701      0.000   -2.61e+04   -7956.125
enginetype_l              747.4585   1723.324      0.434      0.665   -2653.852    4148.769
enginetype_ohc           3708.5169    985.669      3.762      0.000    1763.109    5653.924
enginetype_ohcf          3212.0360   1299.458      2.472      0.014     647.308    5776.764
enginetype_ohcv         -3210.3829   1336.586     -2.402      0.017   -5848.391    -572.374
enginetype_rotor        -7668.9047   1752.295     -4.376      0.000   -1.11e+04   -4210.414
cylindernumber_five      -1.54e+04   2235.547     -6.890      0.000   -1.98e+04    -1.1e+04
cylindernumber_four     -1.908e+04   2291.915     -8.326      0.000   -2.36e+04   -1.46e+04
cylindernumber_six      -1.368e+04   1936.665     -7.065      0.000   -1.75e+04   -9860.391
cylindernumber_three    -1.285e+04   4142.378     -3.102      0.002    -2.1e+04   -4674.727
cylindernumber_twelve   -7479.4343   4282.928     -1.746      0.083   -1.59e+04     973.743
cylindernumber_two      -7668.9047   1752.295     -4.376      0.000   -1.11e+04   -4210.414
fuelsystem_2bbl          -577.5154    872.767     -0.662      0.509   -2300.089    1145.058
fuelsystem_4bbl         -1101.5106   3142.414     -0.351      0.726   -7303.666    5100.645
fuelsystem_idi           8247.4901   6843.181      1.205      0.230   -5258.838    2.18e+04
fuelsystem_mfi          -4219.5180   2919.924     -1.445      0.150   -9982.547    1543.512
fuelsystem_mpfi         -1482.6046    990.359     -1.497      0.136   -3437.268     472.059
fuelsystem_spdi         -3700.8498   1480.817     -2.499      0.013   -6623.526    -778.173
fuelsystem_spfi          -816.1745   2765.423     -0.295      0.768   -6274.267    4641.918
     ==============================================================================
     Omnibus:                       27.721   Durbin-Watson:                   1.417
     Prob(Omnibus):                  0.000   Jarque-Bera (JB):               95.838
     Skew:                           0.449   Prob(JB):                     1.55e-21
     Kurtosis:                       6.227   Cond. No.                     1.02e+16
     ==============================================================================
```

The table above depicts regression through dummy, including dummy variables with categories limited to 10 or less. The reason we limited the categories to 10 is because the pile can become too large and the number of data is not large enough. Here, only dummies with a p-value of 0.05 or less were selected and all dummies were removed, leaving only carbody, engine location, wheelbase, and cylindernumber.

Anova test with categorical values and ‘price’

Attribute	Anova p-value	Attribute	Best rsquared
carname	3.19e-16	fueltype	0.13
aspiration	0.01	doornumber	0.65
carbody	5.03e-06	drivewheel	6.63e-24
enginelocation	1.99e-06	enginetype	4.69e-09
cylindernumber	8.06e-41	fuelsystem	2.99e-16

Based on the above regression results and anova test results, I decided to leave only carbody, engine location, and cylinder number among the categorical variables and exclude other categorical variables.

Final results with numerical and categorical variables after normalization

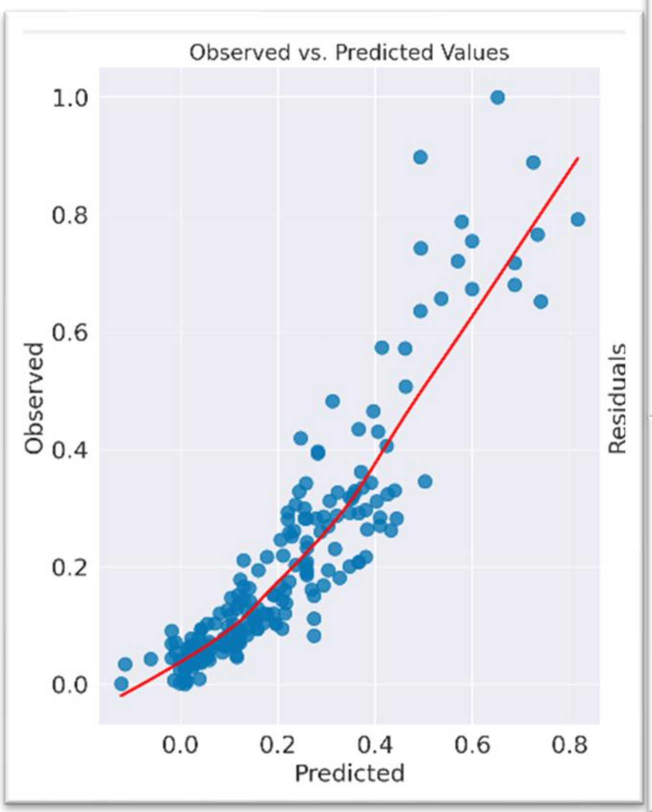
```
                          Final OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.877
Model:                            OLS   Adj. R-squared:                  0.867
Method:                 Least Squares   F-statistic:                     96.40
Date:                Wed, 04 Oct 2023   Prob (F-statistic):           2.77e-78
Time:                        20:56:15   Log-Likelihood:                 255.73
No. Observations:                 205   AIC:                            -481.5
Df Residuals:                     190   BIC:                            -431.6
Df Model:                          14
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                     0.3623      0.052      6.974      0.000       0.260       0.465
wheelbase                 0.3963      0.043      9.288      0.000       0.312       0.480
compressionratio          0.0537      0.023      2.290      0.023       0.007       0.100
horsepower                0.4831      0.055      8.765      0.000       0.374       0.592
carbody_hardtop          -0.1214      0.040     -3.042      0.003      -0.200      -0.043
carbody_hatchback        -0.1607      0.033     -4.912      0.000      -0.225      -0.096
carbody_sedan            -0.1429      0.033     -4.268      0.000      -0.209      -0.077
carbody_wagon            -0.1821      0.037     -4.983      0.000      -0.254      -0.110
enginelocation_rear       0.2872      0.054      5.297      0.000       0.180       0.394
cylindernumber_five      -0.1894      0.042     -4.492      0.000      -0.273      -0.106
cylindernumber_four      -0.3087      0.040     -7.715      0.000      -0.388      -0.230
cylindernumber_six       -0.2004      0.038     -5.232      0.000      -0.276      -0.125
cylindernumber_three     -0.2300      0.086     -2.675      0.008      -0.400      -0.060
cylindernumber_twelve    -0.0765      0.084     -0.913      0.362      -0.242       0.089
cylindernumber_two       -0.2378      0.053     -4.465      0.000      -0.343      -0.133
==============================================================================
Omnibus:                       67.108   Durbin-Watson:                   1.067
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              420.237
Skew:                           1.075   Prob(JB):                     5.58e-92
Kurtosis:                       9.677   Cond. No.                         33.1
==============================================================================
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

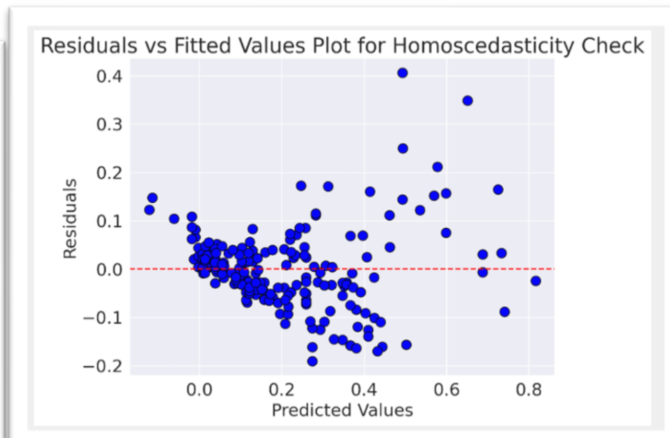
‘Symboling’ showed a p-value of 0.309 as a result of configuring regression with a dummy variable. I ultimately constructed a regression including the three selected dummy variables excluding ‘symboling’. To compare coefficients between variables, normalization and regression were performed on numerical variables and price (dependent variable). Normalization is preferred over standardization when our data doesn't follow a normal distribution. It can be useful in those machine learning algorithms that do not assume any distribution of data like the k-nearest neighbor and neural networks (Harshal, 2022).

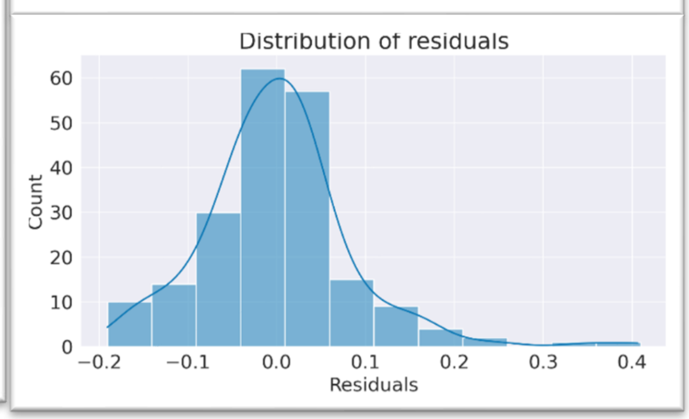
Interpretation of final regression

1. In numerical variables, horsepower, which can be said to be the power of the engine, recorded the highest coefficient at 0.4831.
2. The wheelbase was 0.3963 and the compression ratio was 0.0537, the lowest among the three, and the p-value was also 0.023. All three values have a positive correlation with price.
3. Most categorical values are negative, which can be assumed to mean that the values removed from dummy variables are positive. A negative value means that these factors worked to lower the price of the car, and in the case of enginelocation_rear, it is shown as positive because it had a positive effect on the price.
4. Engine location, which has only two factors, is understood to have the greatest pricing power.
5. Finally, horsepower (0.4831) > wheelbase (0.3963) > cylindernumber_four (-0.3087) > enginelocation_rear (0.2872). It has the above impact on price.
6. The final R-squared value is 0.877 and adjusted R-squared is 0.867 which is higher than 0.852 when all numerical variables are used and lower than 0.913 when selected numerical variables and all categorical variables are reflected together, but the AIC and BIC is much lower than other regression.

Checking regression Assumptions







1. In the Observed vs Predicted value graph on the far left, you can see that the observed value and predicted value are gathered around y=x. I can confirm that the observed value is lower than the predicted value up to about 0.4. After 0.4, the observed value decreases, but you can see that the observed value becomes larger than the predicted value.
2. In regression analysis, homoscedasticity means a situation in which the variance of the dependent variable is the same for all the data. Homoscedasticity facilitates analysis because most methods are based on the assumption of equal variance (Statistics.com, n.d.).
3. Residuals vs Fitted graph is a graph for checking homoscedasticity. Here, price must have the same variance for all data. In other words, residuals vs fitted should appear without showing any pattern.
4. From the distribution of residuals, we can see that the regression model has normality. The value of the shapiro test is 0.937, so it is difficult to say that there is sufficient grounds to reject H0.

Answering Questions

1. What were the three most significant variables?

Numerical variables: horsepower > wheelbase > compression ratio

Categorical variables: cylindernumber_four > enginelocation_rear > cylindernumber_three

After normalizing both the dependent variable (price) and the independent variable, the coefficient of horsepower was the highest among the numerical variables at 0.4831. Next, the coefficient of wheelbase was 0.3963. The compression ratio was the lowest at 0.0537 and the p-value was the highest at 0.023.

Among categorical variables, car body, engine location, and cylinder number showed the highest coef. Among them, the order of cylindernumber_four, enginelocation_rear, and cylindernumber_three showed the greatest impact on price.

2. Of those three, which had the greatest positive influence on car prices?

Numerical variable: horsepower/ Categorical variable: enginelocation_rear

The higher the horsepower, which represents the maximum driving ability of the vehicle, the higher the price tended to be. Horsepower is a measurement used to calculate how quickly the force is produced from a vehicle's engine. It is a key component used to establish the vehicle's total number of miles during its lifetime. It is also used to inform the driver of the vehicle's maximum running capacity (Kia, 2023).

Engine location was divided into rear and front in the data we had, and in visualization, we could see that the front accounted for 98.5%. Engine location at the rear had the greatest positive effect on price. This also meant that the car could move quickly and with greater power. Unless you’re on the racetrack, rear engine vehicles are kind of hard to come by. Reason being, these engine applications have a higher learning curve than other configurations. But for race cars under the control of a professional driver, rear engines are great. They provide a lot of power and traction to the back wheels, which makes them quick to accelerate. Although, that same power to the back wheels can come back to bite them (leithcars, n.d.).

3. How accurate was the model?

The model finally uses three numerical variables and three categorical columns to achieve an R-squared of 0.877 and Adj. R-squared was recorded at 0.867. The model finally uses three numerical variables and three categorical columns to achieve an R-squared of 0.877 and Adj. R-squared was recorded at 0.867. The difference between the two indicators is relatively small and that it shows a fairly high explanatory result by using only 6 variables.

Conclusion

While working on this midweek project, I created a framework for how to perform regression. Select numerical values to include through best_selection. Multicollinearity identified through visualization is confirmed through a VIF graph. Select categorical variables to include along with numerical variables and construct dummy variables. After normalization of numerical variables, the resulting values are compared. Check the assumptions during each process and whether the assumptions are satisfied in the final process. Review the process again to see if there is a way to increase R-squared.

I will keep in mind what mechanisms are needed when applying other machine learning models like this framework. I will create my own template Python code that can be applied to each machine learning model.

Reference

UC Irvine. (n.d.). Automobile. Retrieved from https://archive.ics.uci.edu/ml/datasets/Automobile

Eryk, Lewinson. (2019, June 3). Verifying the assumptions of linear regression in Python and R. Medium. Retrieved from https://towardsdatascience.com/verifying-the-assumptions-of-linear-regression-in-python-and-r-f4cd2907d4c0

smith. (2016). Best subset selection. Retrieved from http://www.science.smith.edu/~jcrouser/SDS293/labs/lab8-py.html

Naveen. (2023, January 21). Pandas convert column to float in dataframe. SparkBy. Retrieved from https://sparkbyexamples.com/pandas/pandas-convert-string-to-float-type-dataframe/#:~:text=Use%20pandas%20DataFrame.,float%2C%20you%20can%20use%20numpy.

Cazoo. (2023). What is mpg?. Retrieved from https://www.cazoo.co.uk/the-view/buying/what-is-mpg/

Brandan, Gillogly. (2020, January 8). How bore vs. stroke can affect horsepower. HAGERTY. Retrieved from https://www.hagerty.com/media/maintenance-and-tech/how-bore-vs-stroke-can-affect-horsepower/

sefidian. (n.d.). Measure the correlation between numerical and categorical variables and the correlation between two categorical variables in Python: Chi-Square and ANOVA. Retrieved from http://www.sefidian.com/2020/08/02/measure-the-correlation-between-numerical-and-categorical-variables-and-the-correlation-between-two-categorical-variables-in-python-chi-square-and-anova/#:~:text=The%20ANOVA%20test%20is%20used,variables%20for%20each%20categorical%20value.

Zach. (2022, October 12). How to Test for Multicollinearity in Python. STATOLOGY. Retrieved from https://www.statology.org/multicollinearity-in-python/#:~:text=The%20most%20straightforward%20way%20to,between%201%20and%20positive%20infinity.

Harshal, Patil. (2022, September 27). Which is better normalization or standardization?. LinkedIn. Retrieved from https://www.linkedin.com/pulse/which-better-normalization-standardization-harshal-patil/

SHRUTI_IYYER. (2020). Step by step assumptions - linear regression. kaggle. Retrieved from https://www.kaggle.com/code/shrutimechlearn/step-by-step-assumptions-linear-regression

Steven Odhiambo. (2021, July 13). Key assumptions of OLS: econometrics review. ALBERT. Retrieved from https://www.albert.io/blog/key-assumptions-of-ols-econometrics-review/

Rafi, Atha. (2020, November 20). Multi-Linear regression using python. Medium. Retrieved from https://medium.com/swlh/multi-linear-regression-using-python-44bd0d10082d

Prashant, Sahu. (2023, January 27). A Comprehensive guide to OLS regression: part-1. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2023/01/a-comprehensive-guide-to-ols-regression-part-1/#:~:text=The%20ordinary%20least%20squares%20(OLS,sum%20of%20the%20squared%20residuals.

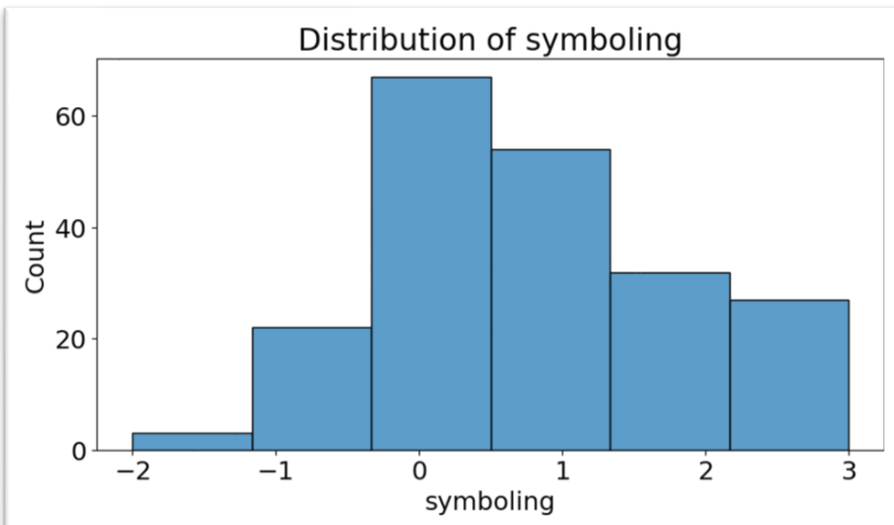
ubc. (2023). Regression with dummy variable. Retrieved from https://blogs.ubc.ca/datawithstata/homepage/regression/ordinary-least-square/#:~:text=Dummy%20Explanatory%20Variable%3A%20When%20one,OLS%20framework%20is%20still%20valid.

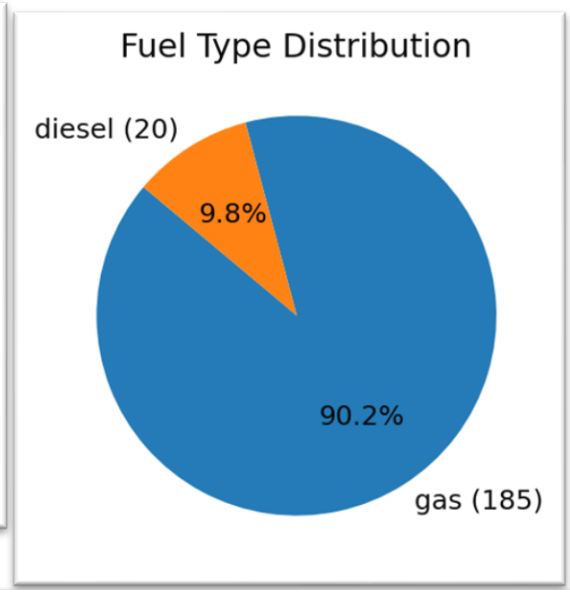
kia. (2023). What Is the horsepower in a car?. Retrieved from https://www.kia.com/mu/discover-kia/ask/what-is-the-horsepower-in-a-car.html#:~:text=Horsepower%20is%20a%20measurement%20used%20to%20calculate%20how%20quickly%20the,the%20vehicle's%20maximum%20running%20capacity.

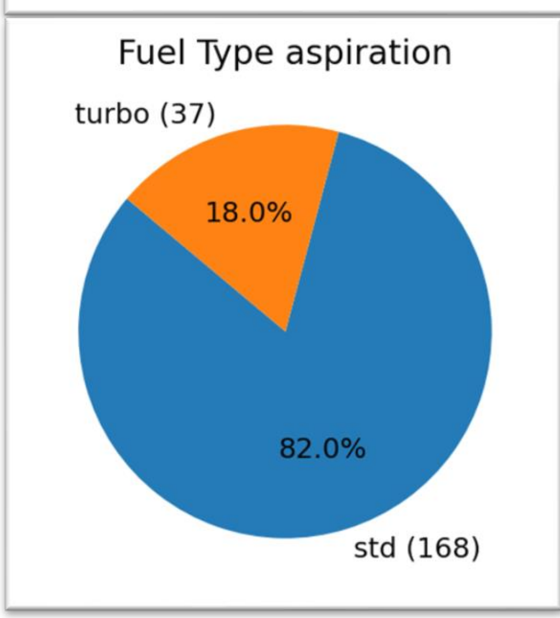
leithcars. (n.d.). Front vs did vs rear engines – which is best?. Retrieved from https://www.leithcars.com/blogs/1421/lifestyle/front-vs-mid-vs-rear-engines/

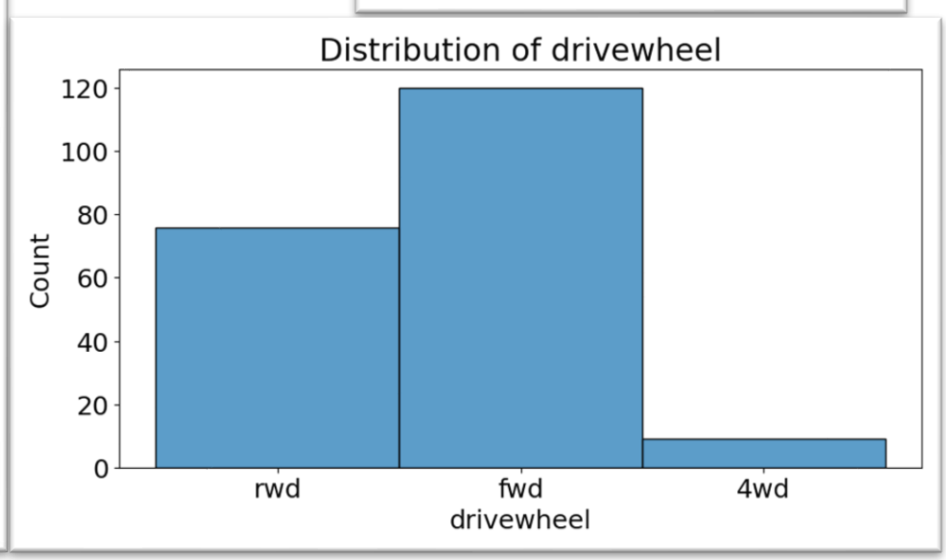
Statistics.com. (n.d.). Homoscedasticity in regression. Retrieved from https://www.statistics.com/glossary/homoscedasticity-in-regression/#:~:text=In%20regression%20analysis%20%2C%20homoscedasticity%20means,the%20assumption%20of%20equal%20variance.

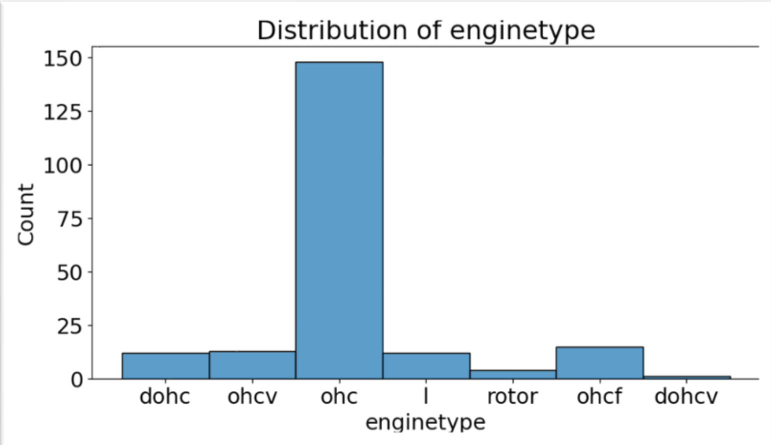
Appendix (graphs):

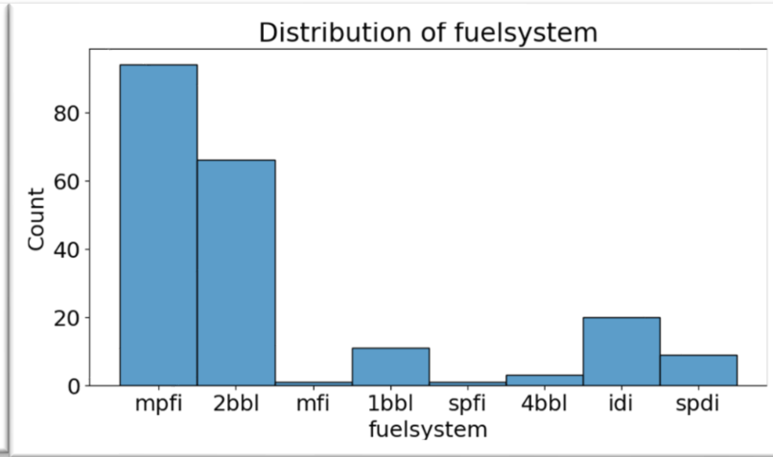


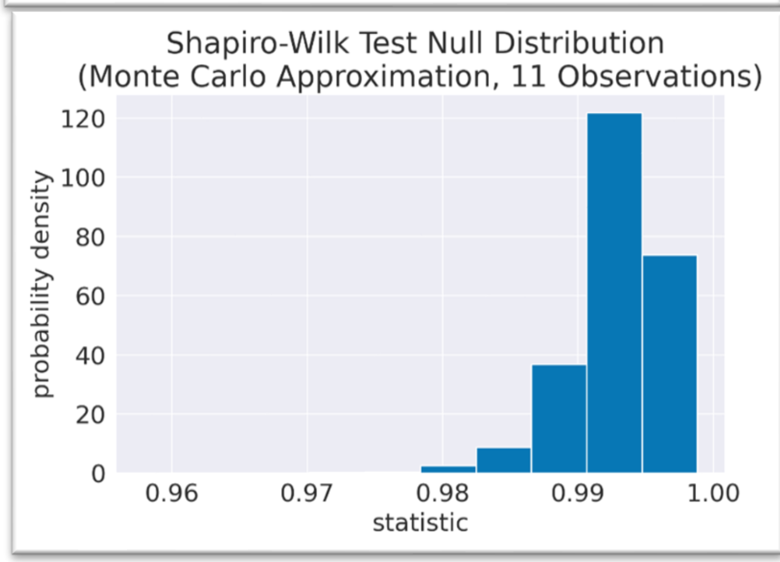












Appendix (Python code):

```
 # In[ ]:
import pandas as pd
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from collections import Counter

# ### DATA Import
# In[ ]:
df = pd.read_csv('CarPrice_Assignment2.csv')
df.head()

# In[ ]:
df.head()

# In[ ]:
df.tail()

# ## Visualization & Understanding Dataset
# In[ ]:
pip install pandas_profiling

# In[ ]:
def missing_values(df):
    missing_number = df.isnull().sum().sort_values(ascending=False)
```

```
    missing_percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)
    missing_values = pd.concat([missing_number, missing_percent], axis=1, keys=['Missing_Number',
'Missing_Percent'])
    return missing_values[missing_values['Missing_Number']>0]
def first_looking(df):
    print(colored("Shape:", attrs=['bold']), df.shape,'\n',
          colored('-'*79, 'red', attrs=['bold']),
          colored("\nInfo:\n", attrs=['bold']), sep='')
    print(df.info(), '\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    print(colored("Number of Uniques:\n", attrs=['bold']), df.nunique(),'\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    print(colored("Missing Values:\n", attrs=['bold']), missing_values(df),'\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    print(colored("All Columns:", attrs=['bold']), list(df.columns),'\n',
          colored('-'*79, 'red', attrs=['bold']), sep='')
    df.columns= df.columns.str.lower().str.replace('&', '_').str.replace(' ', '_')
    print(colored("Columns after rename:", attrs=['bold']), list(df.columns),'\n',
              colored('-'*79, 'red', attrs=['bold']), sep='')

# In[ ]:
pip install colorama

# In[ ]:
pip install termcolor

# In[ ]:
import colorama
from colorama import Fore, Style  # maakes strings colored
from termcolor import colored

# In[ ]:
missing_values(df)

# In[ ]:
first_looking(df)

# In[ ]:
df.describe()

# In[ ]:
import pandas_profiling

# In[ ]:
df.profile_report()

# ## Data Visualization
# In[ ]:
import seaborn as sns

# In[ ]:
df1 = df.copy()

# In[ ]:
# wheelbase histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['wheelbase'])
plt.title("Distribution of wheelbase")

# In[ ]:
# compressionratio histogram
#distribution of compressionratio
```

```
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['compressionratio'])
plt.title("Distribution of compressionratio")

# In[ ]:
# horsepower histogram
#distribution of horsepower
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['horsepower'])
plt.title("Distribution of horsepower")

# In[ ]:
# Symboling histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['symboling'], bins= 6)
plt.title("Distribution of symboling")

# In[ ]:
counts = df1['fueltype'].value_counts()
# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values
# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)
# Display the pie chart
plt.title('Fuel Type Distribution')  # Optional: Add a title to the chart
plt.show()

# In[ ]:
counts = df1['aspiration'].value_counts()
# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values
# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)
# Display the pie chart
plt.title('Fuel Type aspiration')  # Optional: Add a title to the chart
plt.show()

# In[ ]:

# In[ ]:
counts = df1['aspiration'].value_counts()
# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values
# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)
# Display the pie chart
plt.title('Fuel Type aspiration')
plt.show()

# In[ ]:
# Symboling histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
```

```
sns.histplot(x=df1['carbody'], bins= 5)
plt.title("Distribution of carbody")

# In[ ]:
# Symboling histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['drivewheel'], bins= 3)
plt.title("Distribution of drivewheel")

# In[ ]:
counts = df1['enginelocation'].value_counts()
# Extract labels and sizes for the pie chart
labels = counts.index.tolist()
sizes = counts.values
# Create a pie chart
plt.figure(figsize=(6, 6))  # Optional: Set the figure size
plt.pie(sizes, labels=[f'{label} ({count})' for label, count in zip(labels, sizes)], autopct='%.1f%%',
startangle=140)
# Display the pie chart
plt.title('enginelocation')
plt.show()

# In[ ]:
# Symboling histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['enginetype'], bins= 7)
plt.title("Distribution of enginetype")

# In[ ]:
# Symboling histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['cylindernumber'], bins= 7)
plt.title("Distribution of cylindernumber")

# In[ ]:
# Symboling histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['fuelsystem'], bins= 8)
plt.title("Distribution of fuelsystem")

# ### Saving before changing df as df1
# In[ ]:
df1 = df.copy()

# In[ ]:
cat_vars = [x for x in df.columns if df[x].dtype =="object"]
cat_vars

# In[ ]:
num_vars = [x for x in df.columns if x not in cat_vars]
num_vars

# In[ ]:
df_cat = df[cat_vars]
df_cat.head()

# In[ ]:
df_num=df[num_vars]
df_num.head()
```

```python
df_num

# In[ ]:
df_num2=df_num.copy()
df_num2.drop(['car_id'], axis=1)
df_num2_1=df_num2.iloc[:,1:8]
df_num2_1
df_num2_1=pd.concat([df_num2_1, df_num2['price']],axis=1)
df_num2_1

# In[ ]:
sns.set_palette('colorblind')
sns.pairplot(data=df_num2_1)

# In[ ]:
df_num2=df_num.copy()
df_num2.drop(['car_id'], axis=1)
df_num2_2=df_num2.iloc[:,8:17]

# In[ ]:
sns.set_palette('colorblind')
sns.pairplot(data=df_num2_2)

# #### Categorical value ANOVA test
# In[ ]:
df_cat.head()

# In[ ]:
df_anova=pd.concat([df_cat, df_num['price']], axis=1)
df_anova

# In[ ]:
from scipy.stats import f_oneway
# Running the one-way anova test between CarPrice and FuelTypes
# Assumption(H0) is that FuelType and CarPrices are NOT correlated
# Finds out the Prices data for each FuelType as a list
for i in df_anova.columns:
    if i != 'price':
        CategoryGroupLists = df_anova.groupby(i)['price'].apply(list).values
        AnovaResults = f_oneway(*CategoryGroupLists)
        print(f'P-Value for Anova with {i} is:', AnovaResults.pvalue)

# ### Regression right away
# In[ ]:
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
import statsmodels.api as sm
from scipy import stats

# In[ ]:
y=df_num[['price']]
x=df_num.iloc[:, 1:15]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

# In[ ]:
X2 = sm.add_constant(x)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
```

```python
# ### Data visualization
# In[ ]:
import seaborn as sns

# In[ ]:
car_corr_matrix=df_num.corr()

# In[ ]:
plt.figure(figsize=(20, 20))
sns.heatmap(car_corr_matrix, cmap='coolwarm', annot=True)

# ### Best Subset method
# In[ ]:
y=df.price
y

# In[ ]:
X=df_num.drop(['car_id', 'price'], axis=1).astype('float64')
X

# In[ ]:
def processSubset(feature_set):
    # Fit model on feature_set and calculate RSS
    X_subset = sm.add_constant(X[list(feature_set)])
    model = sm.OLS(y,X_subset)
    regr = model.fit()
    RSS = ((regr.predict(X_subset) - y) ** 2).sum()
    return {"model":regr, "RSS":RSS}

# In[ ]:
def getBest(k):
    tic = time.time()
    results = []
    for combo in itertools.combinations(X.columns, k):
        results.append(processSubset(combo))
    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)
    # Choose the model with the highest RSS
    best_model = models.loc[models['RSS'].argmin()]
    toc = time.time()
    print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "seconds.")
    # Return the best model, along with some other useful information about the model
    return best_model

# In[ ]:
# Could take quite awhile to complete...
models_best = pd.DataFrame(columns=["RSS", "model"])
tic = time.time()
for i in range(1,15):
    models_best.loc[i] = getBest(i)
toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")

# In[ ]:
# Gets the second element from each row ('model') and pulls out its rsquared attribute
models_best.apply(lambda row: row[1].rsquared, axis=1)

# In[ ]:
print(models_best.loc[6, "model"].summary())

# In[ ]:
print(models_best.loc[14, "model"].summary())
```

```python
# ### Checking multicollinearity
# In[ ]:
df2=df_num.drop(['car_id'], axis=1).astype('float64')

# In[ ]:
from patsy import dmatrices
from statsmodels.stats.outliers_influence import variance_inflation_factor
#find design matrix for regression model using 'rating' as response variable
y, X = dmatrices('price ~
symboling+wheelbase+carlength+carwidth+carheight+curbweight+enginesize+boreratio+stroke+compressionrat
io+horsepower+peakrpm+citympg+highwaympg', data=df2, return_type='dataframe')
#create DataFrame to hold VIF values
vif_df = pd.DataFrame()
vif_df['variable'] = X.columns
#calculate VIF for each predictor variable
vif_df['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
#view VIF for each predictor variable
print(vif_df)

# In[ ]:
vif_df = vif_df[vif_df['variable'] != 'Intercept']
vif_df

# In[ ]:
plt.figure(figsize=(10, 6))
plt.bar(vif_df['variable'], vif_df['VIF'], color='skyblue')
plt.axhline(y=5, color='red', linestyle='--', label='VIF Threshold (5)')
plt.xlabel('Predictor Variables')
plt.ylabel('VIF Values')
plt.title('VIF Values for Predictor Variables')
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.tight_layout()
plt.show()

# ### Multicollinearity check 2
# In[ ]:
#find design matrix for regression model using 'rating' as response variable
y, X = dmatrices('price ~
symboling+wheelbase+carheight+boreratio+stroke+compressionratio+peakrpm+citympg', data=df2,
return_type='dataframe')
#create DataFrame to hold VIF values
vif_df = pd.DataFrame()
vif_df['variable'] = X.columns
#calculate VIF for each predictor variable
vif_df['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
#view VIF for each predictor variable
print(vif_df)

# In[ ]:
vif_df = vif_df[vif_df['variable'] != 'Intercept']
vif_df

# In[ ]:
plt.figure(figsize=(10, 6))
plt.bar(vif_df['variable'], vif_df['VIF'], color='skyblue')
plt.axhline(y=5, color='red', linestyle='--', label='VIF Threshold (5)')
plt.xlabel('Predictor Variables')
plt.ylabel('VIF Values')
plt.title('VIF Values for Predictor Variables')
plt.xticks(rotation=45, ha='right')
plt.legend()
plt.tight_layout()
```

```
plt.show()

# ### to check with constant
# In[ ]:
y=df_num[['price']]
X=df_num.iloc[:, 1:15]
X=df_num.drop(['car_id', 'carlength', 'carwidth', 'curbweight', 'enginesize', 'highwaympg', 'price'],
axis=1).astype('float64')

# In[ ]:
X

# In[ ]:
y=y.price
y

# In[ ]:
def processSubset(feature_set):
    # Fit model on feature_set and calculate RSS
    X_subset = sm.add_constant(X[list(feature_set)])
    model = sm.OLS(y,X_subset)
    regr = model.fit()
    RSS = ((regr.predict(X_subset) - y) ** 2).sum()
    return {"model":regr, "RSS":RSS}

# In[ ]:
def getBest(k):
    tic = time.time()
    results = []
    for combo in itertools.combinations(X.columns, k):
        results.append(processSubset(combo))
    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)
    # Choose the model with the highest RSS
    best_model = models.loc[models['RSS'].argmin()]
    toc = time.time()
    print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "seconds.")
    # Return the best model, along with some other useful information about the model
    return best_model

# In[ ]:
# Could take quite awhile to complete...
models_best = pd.DataFrame(columns=["RSS", "model"])
tic = time.time()
for i in range(1,9):
    models_best.loc[i] = getBest(i)
toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")

# In[ ]:
models_best

# In[ ]:
# Gets the second element from each row ('model') and pulls out its rsquared attribute
models_best.apply(lambda row: row[1].rsquared, axis=1)

# In[ ]:
print(models_best.loc[5, "model"].summary())

# In[ ]:
plt.figure(figsize=(20,10))
plt.rcParams.update({'font.size': 18, 'lines.markersize': 10})
# Set up a 2x2 grid so we can look at 4 plots at once
```

```
plt.subplot(2, 2, 1)
# We will now plot a red dot to indicate the model with the largest adjusted R^2 statistic.
# The argmax() function can be used to identify the location of the maximum point of a vector
plt.plot(models_best["RSS"])
plt.xlabel('# Predictors')
plt.ylabel('RSS')
# We will now plot a red dot to indicate the model with the largest adjusted R^2 statistic.
# The argmax() function can be used to identify the location of the maximum point of a vector
rsquared_adj = models_best.apply(lambda row: row[1].rsquared_adj, axis=1)
plt.subplot(2, 2, 2)
plt.plot(rsquared_adj)
plt.plot(rsquared_adj.argmax(), rsquared_adj.max(), "or")
plt.xlabel('# Predictors')
plt.ylabel('adjusted rsquared')
# We'll do the same for AIC and BIC, this time looking for the models with the SMALLEST statistic
aic = models_best.apply(lambda row: row[1].aic, axis=1)
plt.subplot(2, 2, 3)
plt.plot(aic)
plt.plot(aic.argmin(), aic.min(), "or")
plt.xlabel('# Predictors')
plt.ylabel('AIC')
bic = models_best.apply(lambda row: row[1].bic, axis=1)
plt.subplot(2, 2, 4)
plt.plot(bic)
plt.plot(bic.argmin(), bic.min(), "or")
plt.xlabel('# Predictors')
plt.ylabel('BIC')

# In[ ]:
print(models_best.loc[4, "model"].summary())

# ### with dummy varialbes
# In[ ]:
df1

# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['car_id', 'carname', 'fueltype', 'doornumber', 'carheight', 'boreratio', 'peakrpm',
'citympg', 'stroke', 'carlength', 'carwidth', 'curbweight', 'enginesize', 'highwaympg', 'price'],
axis=1)
x

# In[ ]:
#### 'aspiration'
asp = pd.get_dummies(x[["aspiration"]],drop_first = True)
x = pd.concat([x,asp],axis=1)
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = True)
x = pd.concat([x,cbd],axis=1)
### 'drivewheel'
dw = pd.get_dummies(x[["drivewheel"]],drop_first = True)
x = pd.concat([x,dw],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = True)
x = pd.concat([x,engl],axis=1)
### 'enginetype'
engt =pd.get_dummies(x[["enginetype"]],drop_first = True)
x = pd.concat([x,engt],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = True)
x = pd.concat([x,cyln],axis=1)
### 'fuelsystem'
```

```
fs = pd.get_dummies(x[["fuelsystem"]],drop_first = True)
x = pd.concat([x,fs],axis=1)

# In[ ]:
X

# In[ ]:
y=y.price
y

# In[ ]:
def processSubset(feature_set):
    # Fit model on feature_set and calculate RSS
    X_subset = sm.add_constant(X[list(feature_set)])
    model = sm.OLS(y,X_subset)
    regr = model.fit()
    RSS = ((regr.predict(X_subset) - y) ** 2).sum()
    return {"model":regr, "RSS":RSS}

# In[ ]:
def getBest(k):
    tic = time.time()
    results = []
    for combo in itertools.combinations(X.columns, k):
        results.append(processSubset(combo))
    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)
    # Choose the model with the highest RSS
    best_model = models.loc[models['RSS'].argmin()]
    toc = time.time()
    print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "seconds.")
    # Return the best model, along with some other useful information about the model
    return best_model

# In[ ]:
models_best

# In[ ]:
# Gets the second element from each row ('model') and pulls out its rsquared attribute
models_best.apply(lambda row: row[1].rsquared, axis=1)

# In[ ]:
print(models_best.loc[4, "model"].summary())

# In[ ]:
x

# In[ ]:
y=df.price
y

# In[ ]:
x = sm.add_constant(x)

# ### again with p-value < 0.05 in dummy
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['car_id', 'carname', 'fueltype', 'fuelsystem', 'drivewheel', 'aspiration', 'enginetype',
'doornumber', 'carheight', 'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength', 'carwidth',
'curbweight', 'enginesize', 'highwaympg', 'price'], axis=1)
x
```

```
# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = True)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = True)
x = pd.concat([x,engl],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = True)
x = pd.concat([x,cyln],axis=1)

# In[ ]:
x.drop(['carbody', 'enginelocation', 'cylindernumber'],axis=1,inplace=True)

# In[ ]:
x

# In[ ]:
y=df.price
y

# In[ ]:
x = sm.add_constant(x)

# ### Backward selection
# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# In[ ]:

# ### Drop symboling
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['car_id', 'symboling', 'carname', 'fueltype', 'fuelsystem', 'drivewheel', 'aspiration',
'enginetype', 'doornumber', 'carheight', 'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength',
'carwidth', 'curbweight', 'enginesize', 'highwaympg', 'price'], axis=1)
x

# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = True)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = True)
x = pd.concat([x,engl],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = True)
x = pd.concat([x,cyln],axis=1)

# In[ ]:
x.drop(['carbody', 'enginelocation', 'cylindernumber'],axis=1,inplace=True)

# In[ ]:
x

# In[ ]:
y=df.price
y

# In[ ]:
x = sm.add_constant(x)
```

```
# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# #### with normalization
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['carbody', 'enginelocation', 'cylindernumber', 'car_id', 'symboling', 'carname',
'fueltype', 'fuelsystem', 'drivewheel', 'aspiration', 'enginetype', 'doornumber', 'carheight',
'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength', 'carwidth', 'curbweight', 'enginesize',
'highwaympg', 'price'], axis=1)
x

# In[ ]:
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler().fit(x)
X_norm = min_max_scaler.transform(x)
X_norm

# In[ ]:
min_max_scaler = MinMaxScaler().fit(y)
y_norm = min_max_scaler.transform(y)
y_norm

# In[ ]:
y_norm_df = pd.DataFrame(y_norm, columns=y.columns)
y=y_norm_df.price
y

# In[ ]:

# In[ ]:
X_norm_df = pd.DataFrame(X_norm, columns=x.columns)
X_norm_df

# In[ ]:
x = X_norm_df
x = pd.concat([x, df1[['carbody', 'enginelocation', 'cylindernumber']]], axis=1)

# In[ ]:
x

# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = True)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = True)
x = pd.concat([x,engl],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = True)
x = pd.concat([x,cyln],axis=1)

# In[ ]:
x.drop(['carbody', 'enginelocation', 'cylindernumber'],axis=1,inplace=True)

# In[ ]:
x

# In[ ]:
x = sm.add_constant(x)
```

```
# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# In[ ]:

# ### Drop symboling
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['car_id', 'symboling', 'carname', 'fueltype', 'fuelsystem', 'drivewheel', 'aspiration',
'enginetype', 'doornumber', 'carheight', 'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength',
'carwidth', 'curbweight', 'enginesize', 'highwaympg', 'price'], axis=1)
x

# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = True)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = True)
x = pd.concat([x,engl],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = True)
x = pd.concat([x,cyln],axis=1)

# In[ ]:
x.drop(['carbody', 'enginelocation', 'cylindernumber'],axis=1,inplace=True)

# In[ ]:
x

# In[ ]:
y=df.price
y

# In[ ]:
x = sm.add_constant(x)

# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# #### without dummy drop=False
#
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['carbody', 'enginelocation', 'cylindernumber', 'car_id', 'symboling', 'carname',
'fueltype', 'fuelsystem', 'drivewheel', 'aspiration', 'enginetype', 'doornumber', 'carheight',
'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength', 'carwidth', 'curbweight', 'enginesize',
'highwaympg', 'price'], axis=1)
x

# In[ ]:
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler().fit(x)
X_norm = min_max_scaler.transform(x)
X_norm

# In[ ]:
min_max_scaler = MinMaxScaler().fit(y)
```

```
y_norm = min_max_scaler.transform(y)
y_norm

# In[ ]:
y_norm_df = pd.DataFrame(y_norm, columns=y.columns)
y=y_norm_df.price
y


# In[ ]:
X_norm_df = pd.DataFrame(X_norm, columns=x.columns)
X_norm_df

# In[ ]:
x = X_norm_df
x = pd.concat([x, df1[['carbody', 'enginelocation', 'cylindernumber']]], axis=1)

# In[ ]:
x

# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = False)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = False)
x = pd.concat([x,engl],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = False)
x = pd.concat([x,cyln],axis=1)

# In[ ]:
x.drop(['carbody', 'enginelocation', 'cylindernumber'],axis=1,inplace=True)

# In[ ]:
x

# In[ ]:
x = sm.add_constant(x)

# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# ### without dummy drop
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['car_id', 'symboling', 'carname', 'fueltype', 'fuelsystem', 'drivewheel', 'aspiration',
'enginetype', 'doornumber', 'carheight', 'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength',
'carwidth', 'curbweight', 'enginesize', 'highwaympg', 'price'], axis=1)
x

# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = True)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = True)
x = pd.concat([x,engl],axis=1)
### 'cylindernumber'
cyln = pd.get_dummies(x[["cylindernumber"]],drop_first = True)
x = pd.concat([x,cyln],axis=1)
```

```
# In[ ]:
x.drop(['carbody', 'enginelocation', 'cylindernumber'],axis=1,inplace=True)

# In[ ]:
x

# In[ ]:
y=df.price
y

# In[ ]:
x = sm.add_constant(x)

# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# #### without cylindernumber
# In[ ]:
y=df1[['price']]
x=df1.iloc[:, 1:26]
x=df1.drop(['carbody', 'enginelocation', 'cylindernumber', 'car_id', 'symboling', 'carname',
'fueltype', 'fuelsystem', 'drivewheel', 'aspiration', 'enginetype', 'doornumber', 'carheight',
'boreratio', 'peakrpm', 'citympg', 'stroke', 'carlength', 'carwidth', 'curbweight', 'enginesize',
'highwaympg', 'price'], axis=1)
x

# In[ ]:
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler().fit(x)
X_norm = min_max_scaler.transform(x)
X_norm

# In[ ]:
min_max_scaler = MinMaxScaler().fit(y)
y_norm = min_max_scaler.transform(y)
y_norm

# In[ ]:
y_norm_df = pd.DataFrame(y_norm, columns=y.columns)
y=y_norm_df.price
y

# In[ ]:
X_norm_df = pd.DataFrame(X_norm, columns=x.columns)
X_norm_df

# In[ ]:
x = X_norm_df
x = pd.concat([x, df1[['carbody', 'enginelocation']]], axis=1)

# In[ ]:
x

# In[ ]:
### 'carbody'
cbd = pd.get_dummies(x[["carbody"]],drop_first = False)
x = pd.concat([x,cbd],axis=1)
### 'enginelocation'
engl =pd.get_dummies(x[["enginelocation"]],drop_first = False)
x = pd.concat([x,engl],axis=1)
```

```python
# In[ ]:
x.drop(['carbody', 'enginelocation'],axis=1,inplace=True)

# In[ ]:
x
# In[ ]:
x = sm.add_constant(x)

# In[ ]:
result=sm.OLS(y, x).fit()
print(result.summary())

# ### checking assumptions
# In[ ]:
get_ipython().run_line_magic('matplotlib', 'inline')
get_ipython().run_line_magic('config', "InlineBackend.figure_format ='retina'")
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.stats.api as sms
sns.set_style('darkgrid')
sns.mpl.rcParams['figure.figsize'] = (15.0, 9.0)
def linearity_test(model, y):
    '''
    Function for visually inspecting the assumption of linearity in a linear regression model.
    It plots observed vs. predicted values and residuals vs. predicted values.
    Args:
    * model - fitted OLS model from statsmodels
    * y - observed values
    '''
    fitted_vals = model.predict()
    resids = model.resid
    fig, ax = plt.subplots(1,2)
    sns.regplot(x=fitted_vals, y=y, lowess=True, ax=ax[0], line_kws={'color': 'red'})
    ax[0].set_title('Observed vs. Predicted Values', fontsize=16)
    ax[0].set(xlabel='Predicted', ylabel='Observed')
    sns.regplot(x=fitted_vals, y=resids, lowess=True, ax=ax[1], line_kws={'color': 'red'})
    ax[1].set_title('Residuals vs. Predicted Values', fontsize=16)
    ax[1].set(xlabel='Predicted', ylabel='Residuals')
linearity_test(result, y)

# In[ ]:
predicted_values = result.fittedvalues
residuals = result.resid

# In[ ]:
predicted_values

# In[ ]:
residuals

# In[ ]:
data = pd.DataFrame({'Predicted Values': predicted_values, 'Residuals': residuals})
data

# In[ ]:
plt.figure(figsize=(8, 6))
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values Plot for Homoscedasticity Check')
sns.scatterplot(x='Predicted Values', y='Residuals', data=data, color='blue', edgecolor='k')
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```