Northeastern

College of Professional Studies
Northeastern University San Jose

MPS Analytics

Course: ALY6020

Assignment:

Module 2 –Project

Submitted on:

October 6, 2023

Submitted to:

Professor: Ahmadi Behzad

Submitted by:

Heejae Roh

Introduction

The car data describes the car's specifications and mpg. In this project, I will find out what factors affect mpg and find ways to improve mpg to save fuel and improve car performance. I will use Regression on numerical and binary variables. While using Regression to find out how to predict mpg through other numerical and binary variables I will check whether the assumption is satisfied.

Dataset Understanding

The dataset is about car mpg and specifications. There are 8 columns and 398 entries without null values. ‘cylinders’, ‘displacement’, and ‘horsepower’ are about engine specification. The acceleration of cars in the United States is stated as a "zero to sixty" time, where vi is zero and vf is 60 miles per hour or 27 meters per second (Meredith, 2001). ‘year’ indicates the year of production, and since 76 (1976) is the average, the acceleration is also estimated to be the time it takes to reach 60 miles per hour. The 'us_made' indicates whether it was produced in the United States, and 1 means it was produced in the United States.

Description of the variables/features in the dataset.

#	column name	Description
1	**Mpg**	Fuel efficiency in miles per gallon
2	**Cylinders**	Number of cylinders in the car
3	**Displacement**	the measure of the cylinder volume
4	**Horsepower**	calculate how quickly the force is produced, and related to maximum speed
5	**Weight**	overall weight of the car
6	**Acceleration**	Number of seconds to reach 60miles
7	**Model_Year**	Year of production
8	**Us_Made**	Whether car it made in the USA or not

Headtail of Dataset 1

	mpg	cylinders	displacement	horsepower	weight
0	18.0	8	307.0	130	3504
1	15.0	8	350.0	165	3693
2	18.0	8	318.0	150	3436
…				…	
395	32.0	4	135.0	84	2295
396	28.0	4	120.0	79	2625
397	31.0	4	119.0	82	2720

Headtail of Dataset 2

	acceleration	model_year	us_mode
0	12.0	70	1
1	11.5	70	1
2	11.0	70	1
…			
395	11.6	82	1
203	18.6	82	1
204	19.4	82	1

Data Cleansing

1. All column names were changed to lower case, and if there was a space, each words in column_names were connected with '_'.
2. There are six '?' in 'horsepower'. Ways to deal with missing values include replacing them with the median, mean, or the value immediately before or after.
3. In this case, I used regression to estimate horsepower as a dependent variable using other values and decide predicted missing horsepower values.
4. The predicted values based on all other variables are shown in the table below, and these values are distributed in various ways, with three values in the range lower than 25%, one value between 25% and 50%, and the other one value between 50% and 75%. I thought that predicting a value matches the data better than inserting the mean, median, or other values as is. Since the data was not that abundant, I decided to use the data included in '?'.

Predicted horsepower value using regression with other variables

horsepower	32	126	330	336	354	374
Predicted_value	58.35	94.01	59.33	98.35	78.75	74.07

Exploratory Data Analysis

Descriptive Analysis of Dataset 1

	mpg	cylinders	displacement	horsepower	weight
count	398	398	398	398	398
mean	23.51	5.45	193.43	104.06	2970.42
std	7.81	1.70	104.27	38.39	846.84
min	9.00	3.00	68.00	46.00	1613.00
25%	17.5	4.00	104.25	75.00	2223.75
50%	23.00	4.00	148.50	92.50	2803.50
75%	29.00	8.00	262.00	125.00	3608.00
max	46.60	8.00	455.00	230.00	5140.00

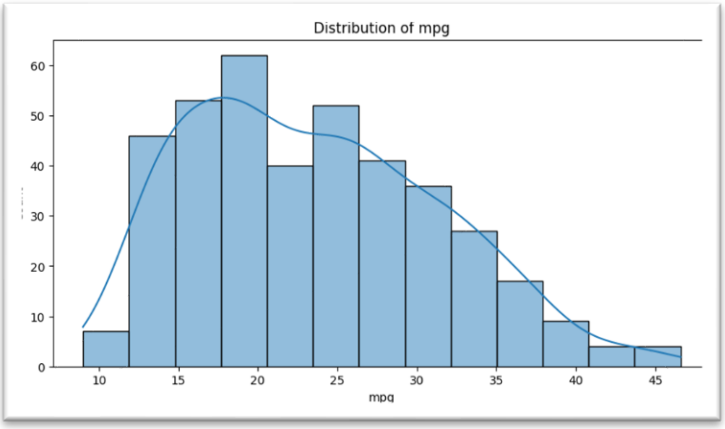
Descriptive Analysis of Dataset 2

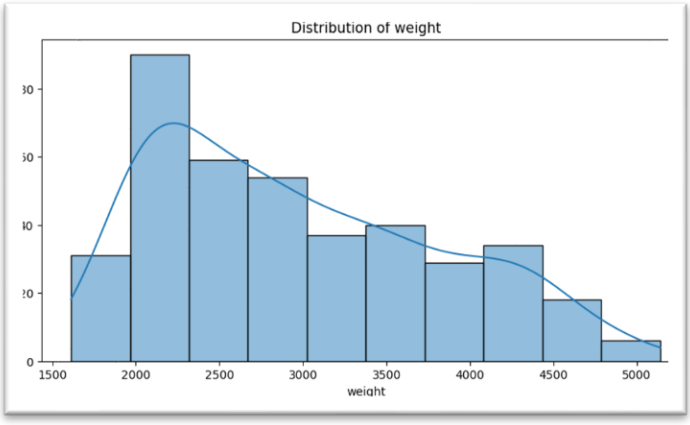
	acceleration	model_year	us_made
count	398	398	398
mean	15.56	76.01	0.63
std	2.76	3.70	0.48
min	8.00	70.00	0.00
25%	13.83	73.00	0.00
50%	15.50	76.00	1.00
75%	17.18	79.00	1.00
max	24.80	82.00	1.00

1. The mean of mpg is 23.5, the minimum is 9.00, and the maximum is 46.6, showing a distribution range between 9 and 46.6.
2. Cylinders are expressed as integer numbers, with a minimum of 3 and a maximum of 8, with a mean of 5.45. displacement has a mean of 193 and is skewed to the right. Horsepower has a mean of 104.06 and is also skewed to the right.
3. The weight is also skewed to the right and has the largest absolute value among the variables included in this data. Acceleration is gathered in a relatively narrow range, from a minimum value of 8 to a maximum value of 24. mean is 15.56.
4. The ‘model_year’ is 76 (1976) and is car data from 1970 to 1982. The ‘us_made’ has a mean of 0.63, which includes more cars made in the United States.

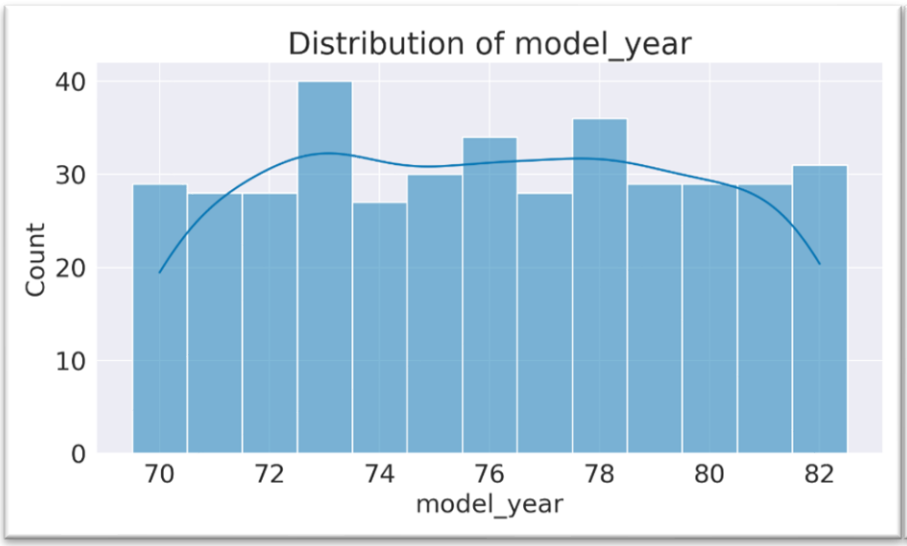
Data Visualizations

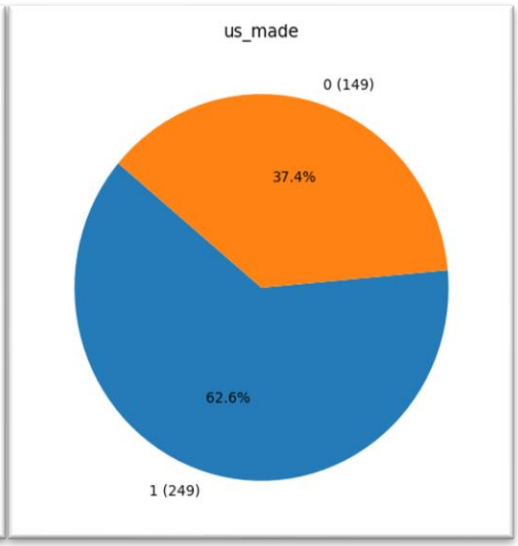
Histograms of primary attribute for linear regression



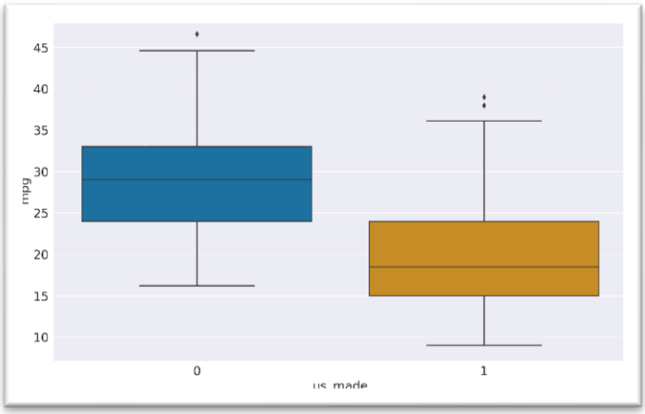


MPG is right skewed, and mode is around 20. MPG ranges from 10 to 45, with most cars ranging from 15 to 30. The weight is also right skewed, and the mode is located between 2000 and 2500. The weight is relatively evenly distributed between 1600 and 5140. We can assume that both graphs are normally distributed.



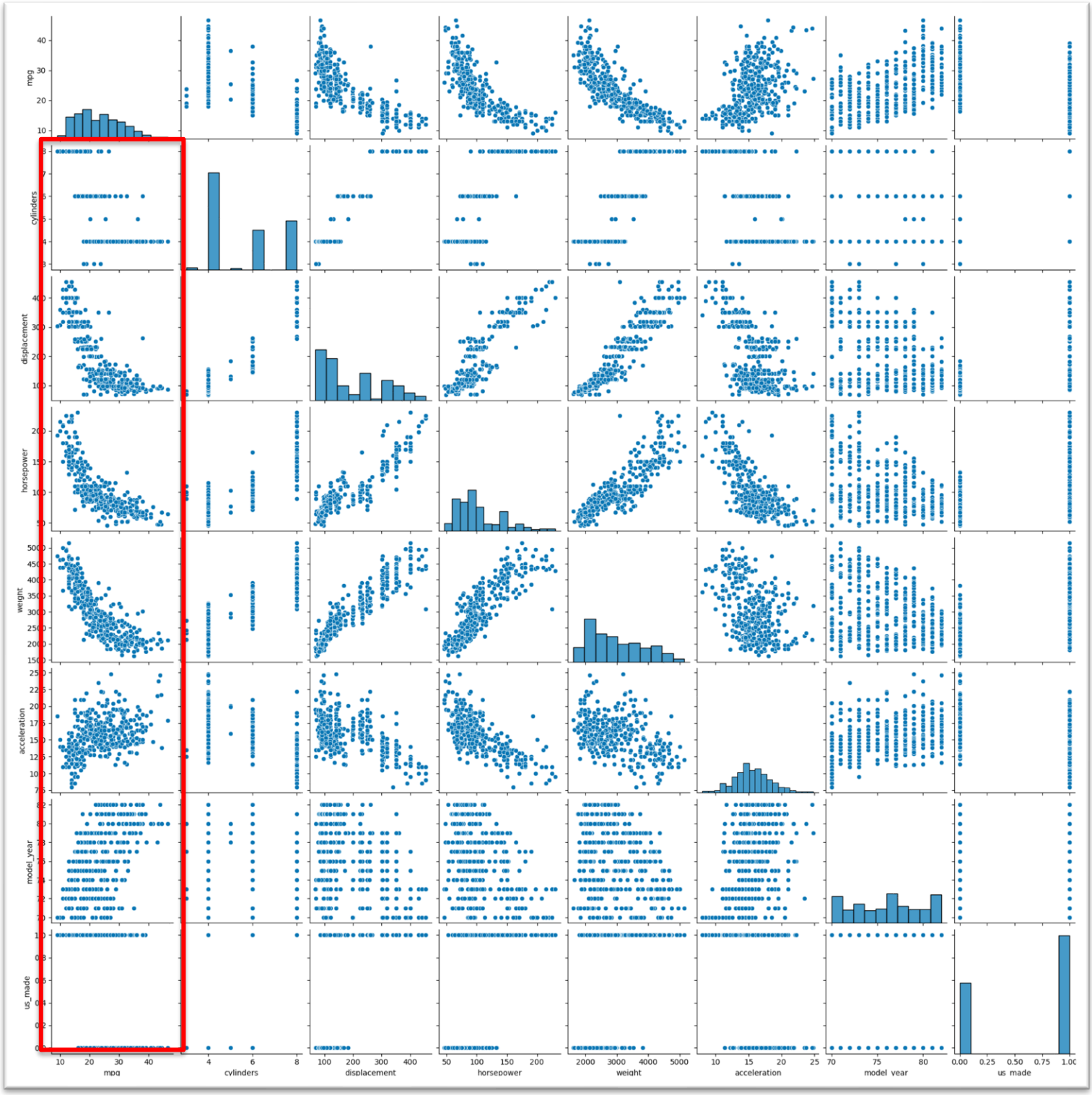


The ‘model_year’ is evenly distributed from 1970 to 1982. 1973, 1978, and 1976 are the top 3 most popular. The ‘us_made’ means 62.6% of cars are made in the United States, and 37.4% are cars made in regions other than the United States.



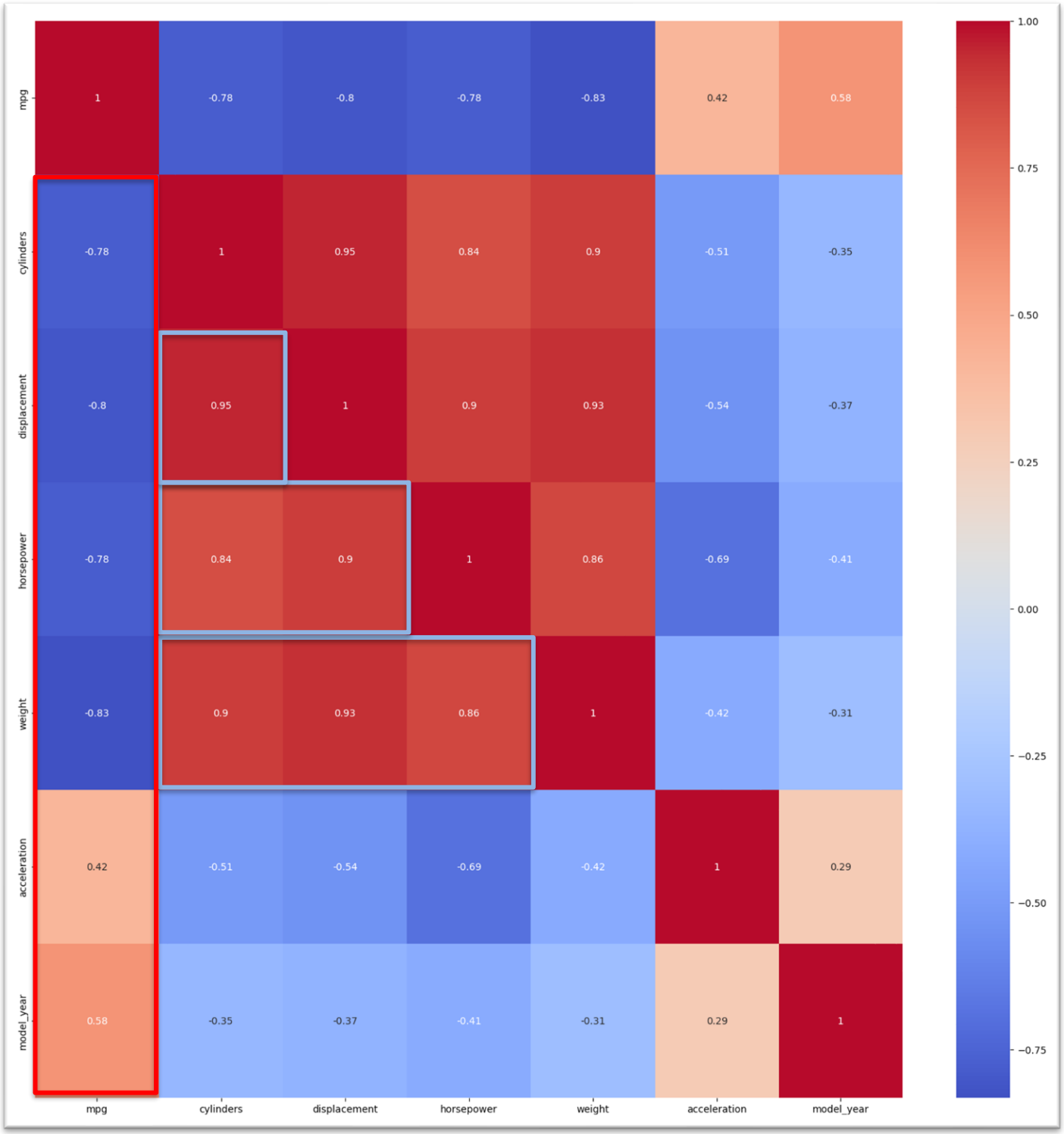
The 'us_made' showed higher mpg when not equal to 1. This can also be confirmed in the box plot. The ‘us_made’ equals to 0 on the left has mpg with a median near 30. The ‘us_made’ equals to 1 on the right has a boxplot with a median near 20.

Scatter plots between variables



The ‘cylinders’, ‘displacement’, ‘horsepower’, ‘weight’, and ‘us_made’ show a negative correlation with mpg, and acceleration and ‘mode_year’ show a positive correlation with mpg. Since correlations are also observed between other data, this will be verified through a correlation matrix or VIF.

Correlation Matrix



In the correlation matrix, weight has a strong correlation with cylinders, displacement, and horsepower. The ‘hoserepower’ also has a large correlation with cylinders and displacement, close to 0.9. Displacement also has a strong correlation with cylinders. The above results show that the numbers related to the engine have a high correlation. As confirmed earlier in the scatterplot, mpg has a positive correlation with two data, and a negative correlation with the other four data. weight has the smallest negative correlation, and ‘model_year’ has the largest positive correlation.

Result of OLS with numerical variables & Interpretation

Result 1: with all numerical attributes

OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.809
Model:	OLS	Adj. R-squared:	0.806
Method:	Least Squares	F-statistic:	275.5
Date:	Thu, 05 Oct 2023	Prob (F-statistic):	4.84e-137
Time:	04:06:09	Log-Likelihood:	-1053.5
No. Observations:	398	AIC:	2121.
Df Residuals:	391	BIC:	2149.
Df Model:	6		
Covariance Type:	nonrobust		

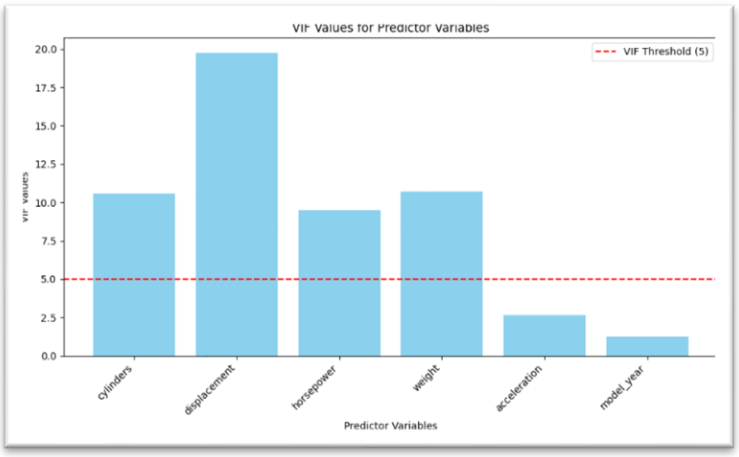
	coef	std err	t	P>|t|	[0.025	0.975]
const	-14.6322	4.743	-3.085	0.002	-23.958	-5.307
cylinders	-0.2577	0.331	-0.779	0.436	-0.908	0.393
displacement	0.0072	0.007	0.980	0.328	-0.007	0.022
horsepower	0.0004	0.014	0.027	0.978	-0.027	0.028
weight	-0.0069	0.001	-10.379	0.000	-0.008	-0.006
acceleration	0.0821	0.102	0.806	0.421	-0.118	0.282
model_year	0.7556	0.052	14.475	0.000	0.653	0.858

Omnibus:	37.131	Durbin-Watson:	1.215
Prob(Omnibus):	0.000	Jarque-Bera (JB):	57.428
Skew:	0.624	Prob(JB):	3.39e-13
Kurtosis:	4.380	Cond. No.	8.51e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.51e+04. This might indicate that there are strong multicollinearity or other numerical problems.

The above table is the result of regression predicting mpg using all numerical variables except 'us_made'. cylinders, displacement, horsepower, and acceleration did not contribute much to the prediction, with p values higher than 0.05. Because the correlation between independent variables is large, it can be assumed that even if regression is constructed with a small number of attributes, it will produce almost similar results as when multiple attributes are constructed together.

VIF histogram with all attributes



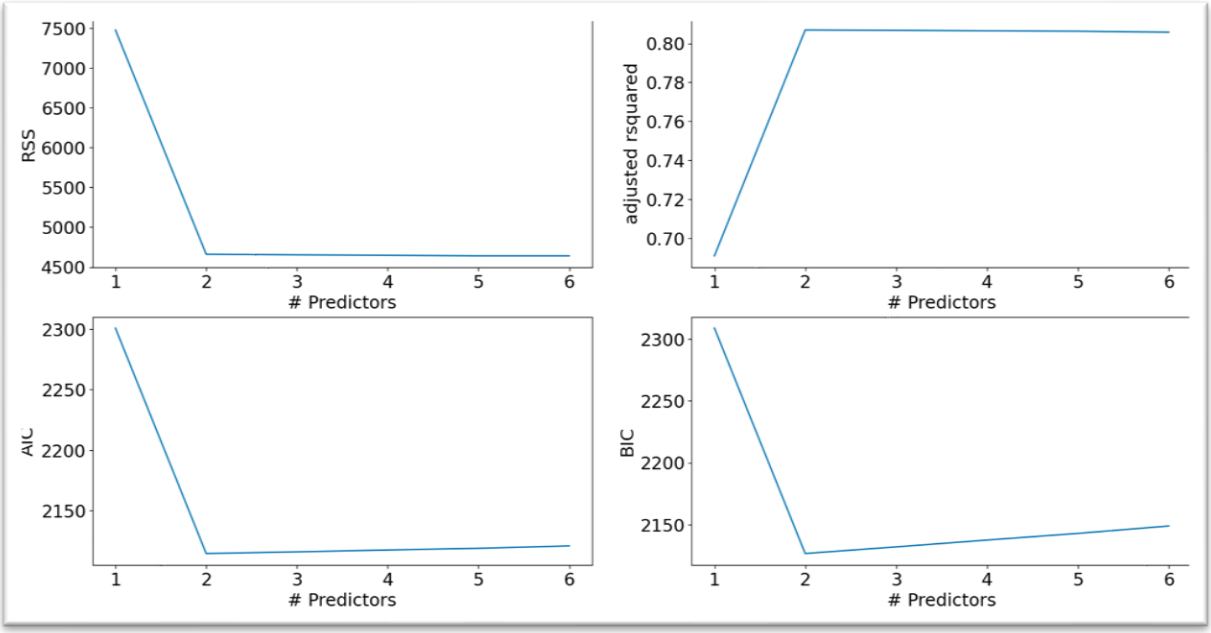
In the VIF results, cylinders, displacement, horsepower, and weight recorded a high multicollinearity of 5 or more. This is consistent with what was previously confirmed through the correlation matrix, and you can choose to remove all attributes or leave only one of the four. The four attributes with high VIF are all attributes that have a negative correlation with mpg.

Best Selection results by attributes number

Attribute number	Best R-squared	Attribute number	Best R-squared
1	0.6917	4	0.8083
2	0.8078	5	0.8086
3	0.8081	6	0.8086

As explained earlier, because many attributes are correlated with each other, all best-selections from 2 to 6 have similar R-squared values, except that the attribute number is 1. To prevent overfitting, we plan to construct regression with only a small number of attributes.

How to choose number of variables



This is a visualization based on the best selection method seen in the table above. RSS, AIC, BIC, and adj R-squared all show that choosing two attributes is most reasonable.

Result 2: After best subset method, choosing 2 attributes

OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.808
Model:	OLS	Adj. R-squared:	0.807
Method:	Least Squares	F-statistic:	830.4
Date:	Thu, 05 Oct 2023	Prob (F-statistic):	3.26e-142
Time:	04:08:20	Log-Likelihood:	-1054.3
No. Observations:	398	AIC:	2115.
Df Residuals:	395	BIC:	2127.
Df Model:	2		
Covariance Type:	nonrobust		

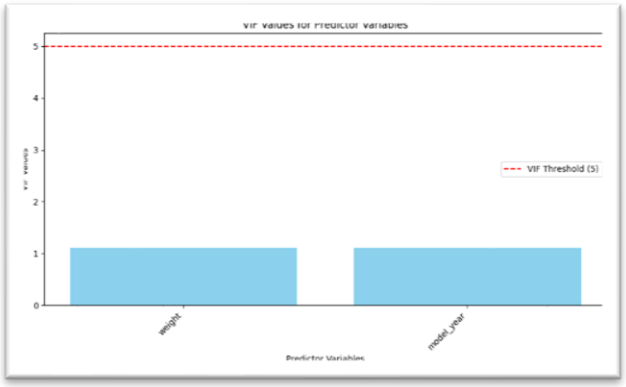
	coef	std err	t	P>|t|	[0.025	0.975]
const	-14.1980	3.968	-3.578	0.000	-21.998	-6.398
weight	-0.0067	0.000	-31.161	0.000	-0.007	-0.006
model_year	0.7566	0.049	15.447	0.000	0.660	0.853

Omnibus:	41.827	Durbin-Watson:	1.216
Prob(Omnibus):	0.000	Jarque-Bera (JB):	68.734
Skew:	0.665	Prob(JB):	1.19e-15
Kurtosis:	4.541	Cond. No.	7.12e+04

```
Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
 [2] The condition number is large, 7.12e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

When two attributes are selected, the highest r-squared value is when weight and model_year are used. Weight is one of the attributes with a negative correlation, and model_year is one of the data with a positive correlation. The R-squared value is 0.808, and the Adj. R-squared value is 0.807.

VIF histogram with all attributes



In the visualization that calculates the VIF value with these two attributes, you can see that it is lower than 5. These can be assumed to be close to 1, showing very low multicollinearity.

Anova test with categorical values and 'price'

Attribute	Anova p-value
us_made	2.10e-35

Now, if you do an anova test for the remaining categorical (binary) variable, us_made, you can see that the p-value is significantly lower than 0.05. Since us_made is already binary data, it can be applied to regression as is without the need to create dummy variables.

Result of OLS with dummy variables & Normalization and Interpretation

OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.819
Model:	OLS	Adj. R-squared:	0.818
Method:	Least Squares	F-statistic:	596.0
Date:	Thu, 05 Oct 2023	Prob (F-statistic):	5.15e-146
Time:	04:20:03	Log-Likelihood:	401.58
No. Observations:	398	AIC:	-795.2
Df Residuals:	394	BIC:	-779.2
Df Model:	3		
Covariance Type:	nonrobust		

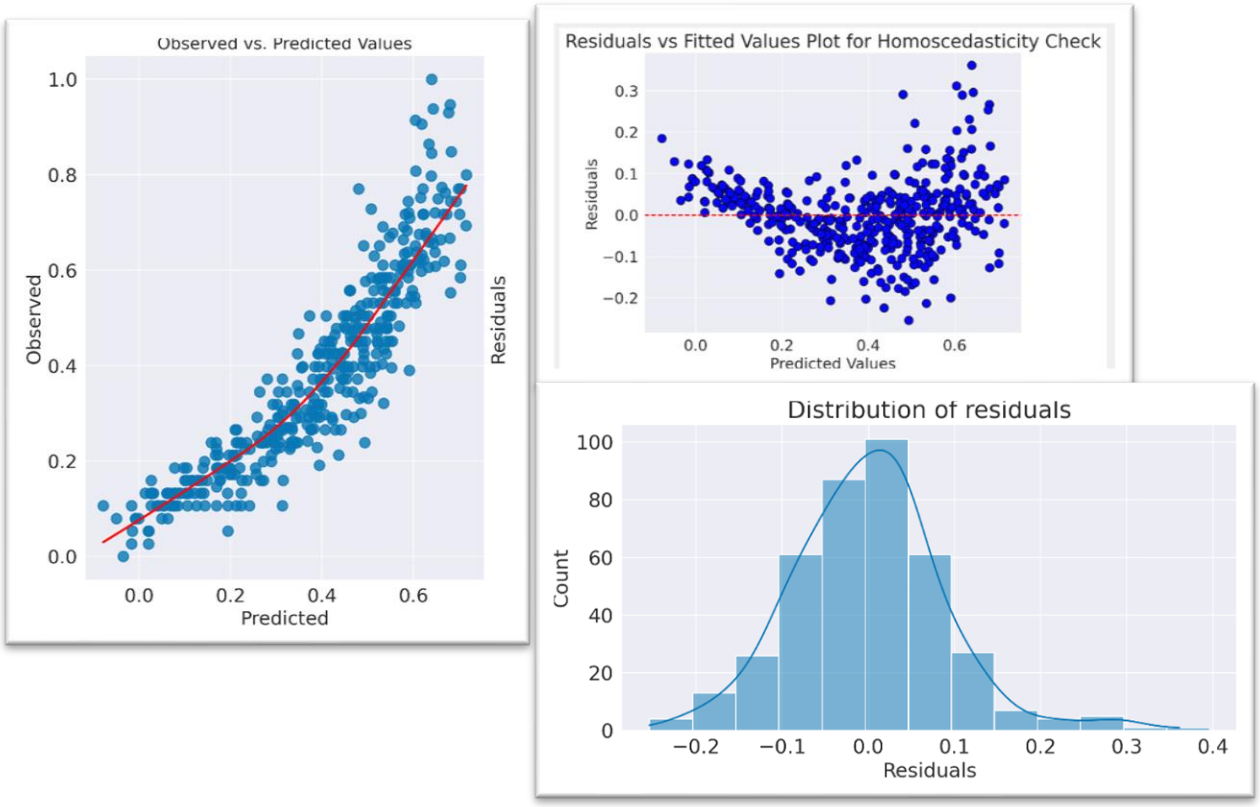
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5121	0.013	39.253	0.000	0.486	0.538
weight	-0.5537	0.024	-22.972	0.000	-0.601	-0.506
model_year	0.2458	0.015	16.176	0.000	0.216	0.276
us_made	-0.0577	0.011	-5.027	0.000	-0.080	-0.035

Omnibus:	30.279	Durbin-Watson:	1.230
Prob(Omnibus):	0.000	Jarque-Bera (JB):	52.555
Skew:	0.485	Prob(JB):	3.87e-12
Kurtosis:	4.492	Cond. No.	8.30

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

1. By applying us_made to regression that included only existing numerical variables, the final r-squared obtained was 0.819 and the adj r-squared was 0.818, showing that there is no significant difference. This is because the adjusted penalty is small because many attributes are not used.
2. All of the above data were applied after normalization. Looking at the coefficient, you can see that weight has the highest absolute value in negative correlation, -0.5537, and us_made also has -0.0577, so as the value increases, mpg decreases. In other words, the heavier the car, the greater the horsepower, acceleration, and displacement, and the higher the likelihood of lower mpg. The fact that us_made is a negative number shows that the above regression may result in lower mpg compared to cars not made in the United States.
3. model_year has a positive correlation with mpg. You can see that the larger model_year, in other words, the newer the model of the car, the better the mpg.

Checking regression Assumptions



1. In the observed vs predicted graph, you can check linearity. The residuals vs fitted values plot does not show any pattern and shows homoscedasticity.
2. Distribution of residuals appears to follow normality, and the Shakiro.test result of residuals is 0.977, confirming that it follows normality.
3. In the independence part, we confirmed that there is no multicollinearity through the VIF value.

Answering Questions

Part 1. Use proper data cleansing techniques to ensure that you have the highest quality data to model this problem. Detail your process and discuss the decisions you made to clean the data.

In this car dataset, the most important part of Data Cleansing was handling data processed with '?' in horsepower. There were six '?'. Rather than deleting the value or choosing to fill it in using mean, median, since this project aims to make predictions through regression with other variables. I predicted 'horsepower' missing values using 7 variables other than horsepower and filled in these values. I thought that I could fill in other missing values using regression as well as other predication techniques if needed.

To reduce confusion in coding, all column_names were converted to lowercase. The data was relatively clean, so not much work was needed.

Predicted horsepower value using regression with other variables

horsepower	32	126	330	336	354	374
Predicted_value	58.35	94.01	59.33	98.35	78.75	74.07

Part 2. Build a linear regression model to accurately predict miles per gallon (MPG) based on the attributes of a vehicle. Discuss the significant attributes and how they can help you build the proper car.

OLS Regression Results

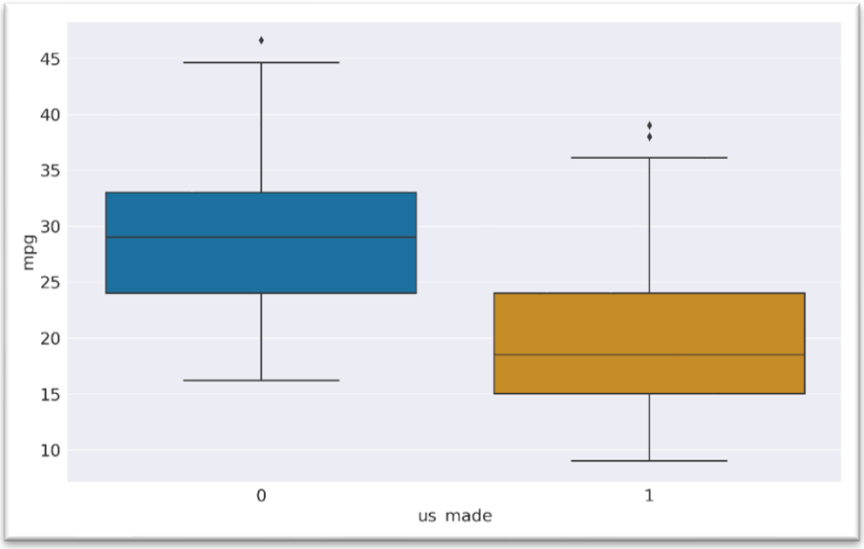
Dep. Variable:	mpg	R-squared:	0.819
Model:	OLS	Adj. R-squared:	0.818
Method:	Least Squares	F-statistic:	596.0
Date:	Thu, 05 Oct 2023	Prob (F-statistic):	5.15e-146
Time:	04:20:03	Log-Likelihood:	401.58
No. Observations:	398	AIC:	-795.2
Df Residuals:	394	BIC:	-779.2
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.5121	0.013	39.253	0.000	0.486	0.538
weight	-0.5537	0.024	-22.972	0.000	-0.601	-0.506
model_year	0.2458	0.015	16.176	0.000	0.216	0.276
us_made	-0.0577	0.011	-5.027	0.000	-0.080	-0.035

Omnibus:	30.279	Durbin-Watson:	1.230
Prob(Omnibus):	0.000	Jarque-Bera (JB):	52.555
Skew:	0.485	Prob(JB):	3.87e-12
Kurtosis:	4.492	Cond. No.	8.30

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

I used the best subset method and ultimately created a regression model with R-squared 0.819 and adj R-squared 0.818 using three attributes: weight, model_year, and us_made. To select these attributes, we checked multicollinearity and compared p-values from regressions including all attributes. Because the 'us_made' column was a binary value, significance was determined through an anova test. This model can be said to explain 81.9% of mpg with three attributes.



3. Optimize the model using selection techniques, explain whether the model can achieve the specified goals, and describe which attributes contribute to higher MPG over others.

I used best subset model. The best subsets regression is a model selection approach that consists of testing all possible combination of the predictor variables, and then selecting the best model according to some statistical criteria (kassambara, 2018). The reason I used the best subset is because there were relatively few attributes and it didn't take much time to check every regression with subset method. The goal was to predict mpg using as few attributes as possible.

As the 'weight' increases, the mpg decreases, which has a big impact on mpg. Therefore, we can think of developing a lightweight car for more efficiency. As 'model_year' increases, mpg decreases. If a car is a new model, the mpg can be increased. This is thought to be due to repeated improvement over the years, and because it is for temporal reasons rather than physical reasons, it may not be a factor that has a direct influence in adjusting mpg. The 'us_made' showed higher mpg when not equal to 1. This can also be confirmed in the box plot.

Conclusion

Through two projects on regression, I came up with my own process of performing regression. In the process, I found and applied code for assumptions that must be checked. From car data, I figured out which attributes affect mpg most. I constructed a regression with three attributes: weight, model_year, and us_made, and created a model with an r-squared of 0.819. Adj r-squared was 0.818, constructing a model with a high r-squared with a relatively small number of attributes. I have found the best regression within the data I have, but I know this is not the end. In the case of 'model_year', although this was the attribute affect a lot to decide mpg, it is a time element, and considering that it was the 70s and 80s, I think model_year would have had a big impact depending on the speed of technological development, especially at that time. But things may be different now, in 2023. I understand that the meaning of data can vary depending on when and in what context it is analyzed.

Reference

kassambara. (2018). Best subsets regression essentials in R. sthda. Retrieved from http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/155-best-subsets-regression-essentials-in-r/#:~:text=The%20best%20subsets%20regression%20is,according%20to%20some%20statistical%20criteria.

seaborn. (2023). seaborn.boxplot. Retrieved from https://seaborn.pydata.org/generated/seaborn.boxplot.html

scipy. (2023). scipy.stats.shapiro. Retrieved from https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html

Saturn Cloud. (2023, July 10). Filling NA using linear regression in R. Retrieved from https://saturncloud.io/blog/filling-na-using-linear-regression-in-r/

Navin Nishanth. (2020, June 29). Pandas — Bfill and Ffill. Medium. Retrieved from https://navinniish001.medium.com/pandas-bfill-and-ffill-b79e46ab87ae

Meredith, Barricella. (2001). Acceleration of a Car. hypertextbook. Retrieved from https://hypertextbook.com/facts/2001/MeredithBarricella.shtml#:~:text=The%20acceleration%20of%20cars%20in,and%204%20m%2Fs2.

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

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/

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/

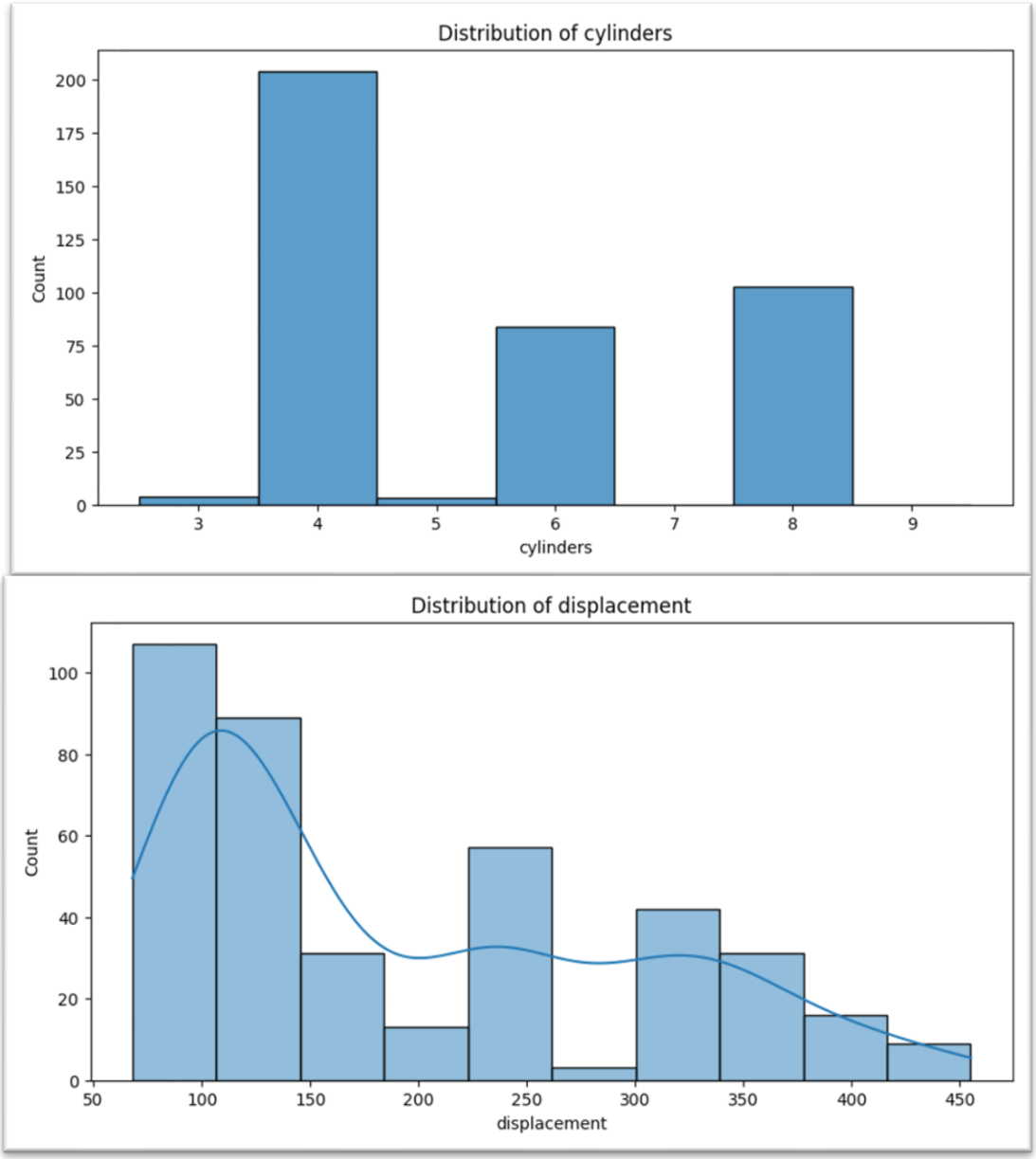
Prashant, Sahu. (2023, January 27). A Comprehensive guide to OLS regression: part-1. Analytics Vidhya.

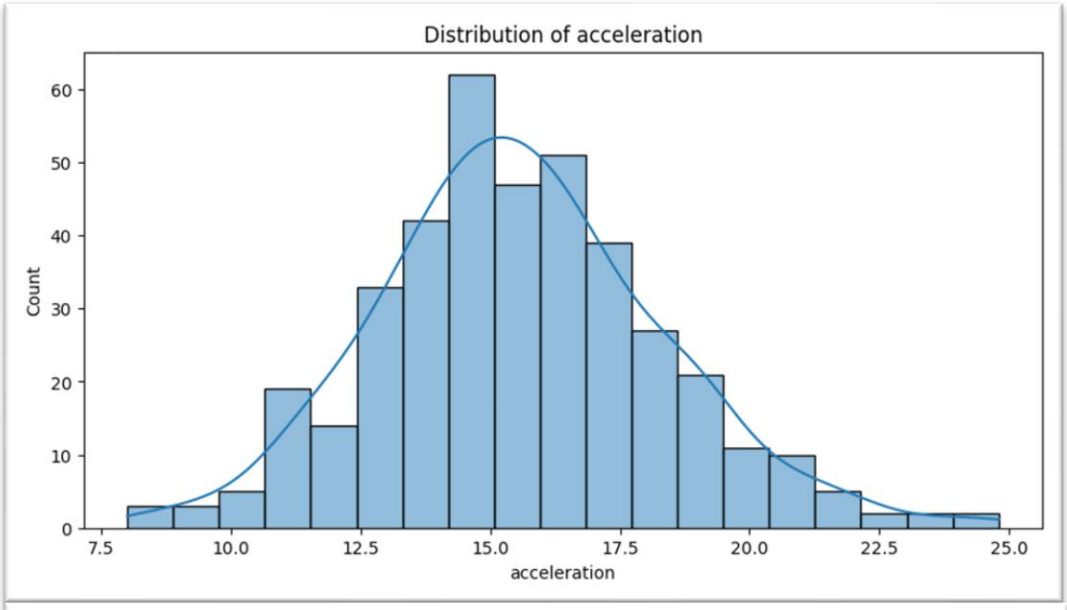
Retrieved from https://www.analyticsvidhya.com/blog/2023/01/a-comprehensive-guide-to-ols-regression-part-1/#:~:text=The%20ordinary%20least%20squares%20(OLS,sum%20of%20the%20squared%20residuals.

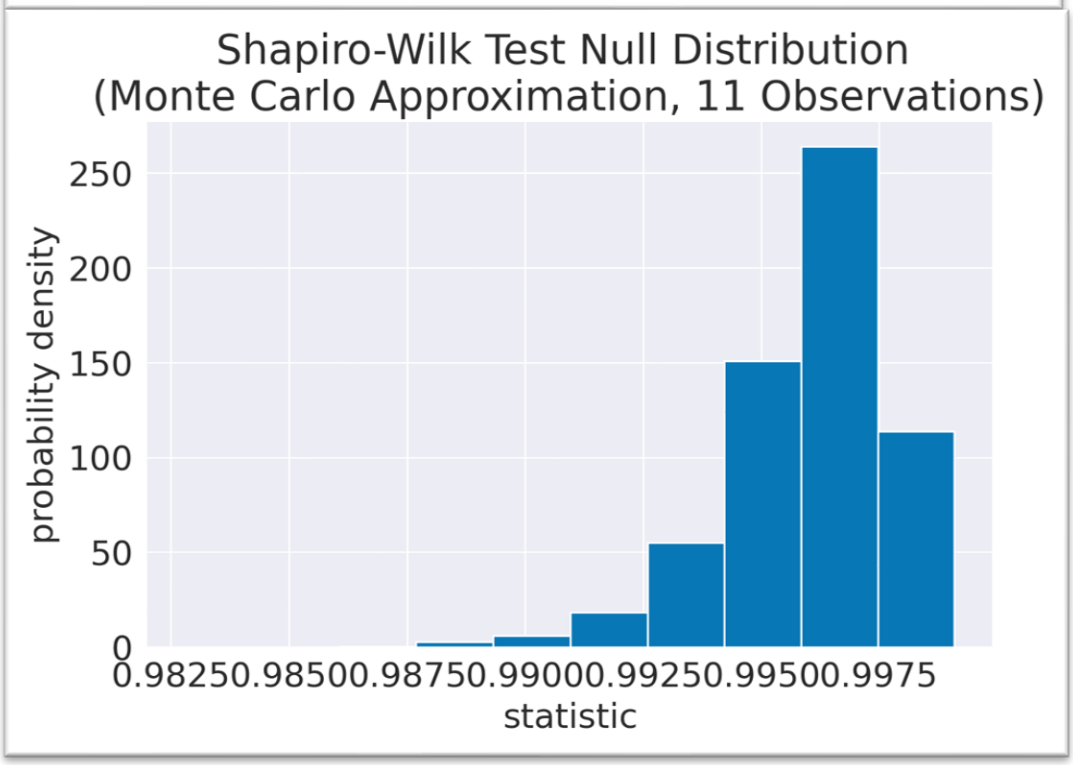
ubc. (2023). Regression with dummy variable. Retrieved from https://blogs.ubc.ca/datawithstata/home-page/regression/ordinary-least-square/#:~:text=Dummy%20Explanatory%20Variable%3A%20When%20one,OLS%20framework%20is%20still%20valid.

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):

```
Original file is located at
    https://colab.research.google.com/drive/1ho7ggwkZAeS59HKY1rzEm6lsl61gUynZ

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"""
```

```
df = pd.read_csv('car.csv')
df.head()

df.tail()

"""## Visualization & Understanding Dataset"""

pip install pandas_profiling

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='')

pip install colorama

pip install termcolor

import colorama
from colorama import Fore, Style  # maakes strings colored
from termcolor import colored

missing_values(df)

first_looking(df)

df.describe()

import pandas_profiling

df.profile_report()

"""### Data Cleansing"""

df['horsepower'].unique()

df.replace({"?":None}, inplace=True)

df['horsepower'].unique()

df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')

df.describe()
```

```
print(df['horsepower'].median())
print(df['horsepower'].mean())
print(df['horsepower'].mode().iloc[0])

missing_data = df[df['horsepower'].isna()]
complete_data = df[~df['horsepower'].isna()]

import statsmodels.api as sm

y = complete_data['horsepower']
X = complete_data.drop('horsepower', axis=1)

X.reset_index(drop=True, inplace=True)
y.reset_index(drop=True, inplace=True)

X = sm.add_constant(X)
y = pd.to_numeric(y, errors='coerce')

y

model = sm.OLS(y, X).fit()

X2 = missing_data.drop('horsepower', axis=1)
missing_data_with_const = sm.add_constant(X2)

predicted_data = model.predict(missing_data_with_const)
predicted_data

df.loc[df['horsepower'].isna(), 'horsepower'] = predicted_data

df['horsepower']

df.describe()

"""## Data Visualization"""

import seaborn as sns

df1 = df.copy()

# mpg histogram
#distribution of capital_gain
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['mpg'], kde=True)

plt.title("Distribution of mpg")

# cylinderso histogram
#distribution of cylinders
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['cylinders'], binwidth=1, binrange=(2.5, 9.5))

plt.title("Distribution of cylinders")

# displacement histogram
#distribution of displacement
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['displacement'], kde=True)

plt.title("Distribution of displacement")

# weight histogram
```

```
#distribution of weight
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['weight'], kde=True)

plt.title("Distribution of weight")

# acceleration histogram
#distribution of acceleration
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['acceleration'], kde=True)

plt.title("Distribution of acceleration")

# model_year histogram
#distribution of model_year
plt.figure(figsize=(10, 5))
sns.histplot(x=df1['model_year'], binwidth=1, binrange=(69.5, 82.5), kde=True)

plt.title("Distribution of model_year")

counts = df1['us_made'].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('us_made')  # Optional: Add a title to the chart
plt.show()



num_vars = df.drop(['us_made'], axis=1)
num_vars

sns.set_palette('colorblind')
sns.pairplot(data=df)

"""#### Categorical value ANOVA test"""

df_anova=df[['us_made', 'mpg']]
df_anova

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 != 'mpg':
        CategoryGroupLists = df_anova.groupby(i)['mpg'].apply(list).values
        AnovaResults = f_oneway(*CategoryGroupLists)
        print(f'P-Value for Anova with {i} is:', AnovaResults.pvalue)

"""### Data visualization"""

import seaborn as sns
```

```
car_corr_matrix=num_vars.corr()

plt.figure(figsize=(20, 20))
sns.heatmap(car_corr_matrix, cmap='coolwarm', annot=True)

"""### Best Subset method"""

import time
import itertools

y=df.mpg
y

X=df.drop(['mpg', 'us_made'], axis=1).astype('float64')
X

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}

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

# Could take quite awhile to complete...

models_best = pd.DataFrame(columns=["RSS", "model"])

tic = time.time()
for i in range(1,7):
    models_best.loc[i] = getBest(i)

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")

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

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

"""### Checking multicollinearity"""
```

```
df2=df.copy()

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('mpg ~ cylinders + displacement + horsepower + weight + acceleration + model_year',
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)

vif_df = vif_df[vif_df['variable'] != 'Intercept']
vif_df

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"""

print(models_best.loc[2, "model"].summary())

#find design matrix for regression model using 'rating' as response variable
y, X = dmatrices('mpg ~ weight + model_year', 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)

vif_df = vif_df[vif_df['variable'] != 'Intercept']
vif_df

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()

"""#### Checking proper number of attributes"""

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.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.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.xlabel('# Predictors')
plt.ylabel('BIC')

"""### with dummy varialbes"""

df1

y=df1[['mpg']]
x=df1.iloc[:, 1:9]
x=df1.drop(['mpg', 'cylinders', 'displacement', 'horsepower', 'acceleration'], axis=1)
x

y=y.mpg
y

x = sm.add_constant(x)

"""### Final Regression"""

y=df1[['mpg']]
x=df1.iloc[:, 1:9]
```

```
x=df1.drop(['mpg', 'cylinders', 'displacement', 'horsepower', 'acceleration'], axis=1)
x


result=sm.OLS(y, x).fit()

print(result.summary())

"""#### with normalization"""

x.drop(['us_made'], axis=1)

from sklearn.preprocessing import MinMaxScaler

min_max_scaler = MinMaxScaler().fit(x)
X_norm = min_max_scaler.transform(x)
X_norm

min_max_scaler = MinMaxScaler().fit(y)
y_norm = min_max_scaler.transform(y)
y_norm

y_norm_df = pd.DataFrame(y_norm, columns=y.columns)
y=y_norm_df.mpg
y


X_norm_df = pd.DataFrame(X_norm, columns=x.columns)
X_norm_df

x = X_norm_df
x = pd.concat([x, df1[['us_made']]], axis=1)

x
x = sm.add_constant(x)

result=sm.OLS(y, x).fit()

print(result.summary())

"""### checking assumptions"""

# %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)

predicted_values = result.fittedvalues
residuals = result.resid

predicted_values

residuals

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

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()

"""### Shapiro Test"""

from scipy import stats

x = data['Residuals']
x1 = residuals
x
res = stats.shapiro(x)

res.statistic

plt.figure(figsize=(10, 5))
sns.histplot(x=data['Residuals'], kde=True, binwidth=0.05)

plt.title("Distribution of residuals")

def statistic(x):
    # Get only the `shapiro` statistic; ignore its p-value
    return stats.shapiro(x).statistic
ref = stats.monte_carlo_test(x, stats.norm.rvs, statistic,
                             alternative='less')
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(8, 5))
def plot(ax):  # we'll re-use this
    ax.hist(ref.null_distribution, density=True)
    ax.set_title("Shapiro-Wilk Test Null Distribution \n"
                 "(Monte Carlo Approximation, 11 Observations)")
    ax.set_xlabel("statistic")
    ax.set_ylabel("probability density")
plot(ax)
plt.show()

sns.boxplot(data=df1, x="us_made", y="mpg")
```