Unit 2 - EDA in Python

For this unit for our portfolio, we were tasked with going through the tutorial of EDA located on Google colab, once the tutorial was completed, we had to do the below with the Auto-mpg dataset which could be found online.

Now, undertake similar EDA with Auto-mpg dataset:

- Identify missing values.
- Estimate Skewness and Kurtosis.
- Correlation Heat Map.
- Scatter plot for different parameters.
- Replace categorical values with numerical values (i.e., America 1, Europe 2 etc.).

Learning Outcomes

- Articulate the legal, social, ethical and professional issues faced by machine learning professionals.
- Understand the applicability and challenges associated with different datasets for the use of machine learning algorithms.
- Apply and critically appraise machine learning techniques to real-world problems, particularly where technical risk and uncertainty is involved.
- Systematically develop and implement the skills required to be effective member of a development team in a virtual professional environment, adopting real-life perspectives on team roles and organisation.

Tutorial

Exploratory Data Analysis or (EDA) is understanding the data sets by summarizing their main characteristics often plotting them visually. This step is very important especially when we arrive at modelling the data in order to apply Machine learning. Plotting in EDA consists of Histograms, Box plot, Scatter plot and many more. We can ask to define the problem statement or definition on our data set which is very important.

It all depends on the dataset that is being worked on, there is no method or common methods to perform EDA but some methods will be explored in the tutorial.

These are the libraries that will be used within the tutorial.

```
import pandas as pd
import numpy as np
import seaborn as sns                     #visualisation
import matplotlib.pyplot as plt           #visualisation
%matplotlib inline
sns.set(color_codes=True)
```

Loading the data into the pandas data frame is certainly one of the most important steps in EDA, as we can see that the value from the data set is comma-separated. So all we have to do is to just read the CSV into a data frame and pandas data frame does the job for us.

```
df = pd.read_csv("data.csv")
# To display the top 5 rows
df.head(5)
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

The code above helps display the top 5 rows which helps gives us a preview of the data

```
df.tail(5)                    # To display the botton 5 rows
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	46120
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	56670
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	50620
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury	Midsize	4dr Hatchback	23	16	204	50920
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury	Midsize	Sedan	26	17	61	28995

The code above displays the bottom 5 rows which also gives a preview of what is needed.

We can then **check the types of data** Here we check for the datatypes because sometimes the MSRP or the price of the car would be stored as a string, if in that case, we have to convert that string to the integer data only then we can plot the data via a graph. Here, in this case, the data is already in integer format so nothing to worry.

```
df.dtypes
```

```
Make                  object
Model                 object
Year                   int64
Engine Fuel Type      object
Engine HP            float64
Engine Cylinders     float64
Transmission Type     object
Driven_Wheels         object
Number of Doors      float64
Market Category       object
Vehicle Size          object
Vehicle Style         object
highway MPG            int64
city mpg               int64
Popularity             int64
MSRP                   int64
dtype: object
```

We then remove the irrelevant columns. In this case, the columns such as Engine Fuel Type, Market Category, Vehicle style, Popularity, Number of doors, Vehicle Size doesn't make any sense to me so I just dropped for this instance.

```
df = df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style',
'Popularity', 'Number of Doors', 'Vehicle Size'], axis=1)
df.head(5)
```

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

The columns will then be renamed as some of the columns can be confusing to read as seen below in the code it helps improve the readability of the dataset

```
df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders":
"Cylinders", "Transmission Type": "Transmission", "Driven_Wheels": "Drive
Mode","highway MPG": "MPG-H", "city mpg": "MPG-C", "MSRP": "Price" })
df.head(5)
```

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

The duplicate rows will now be removed as this can affect results produced but this also helps make the data more manipulative as there are less rows to handle.

```
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (989, 10)
```

```
df.count()      # Used to count the number of rows
```

```
Make            11914
Model           11914
Year            11914
HP              11845
Cylinders       11884
Transmission    11914
Drive Mode      11914
MPG-H           11914
MPG-C           11914
Price           11914
dtype: int64
```

```
df = df.drop_duplicates()
df.head(5)
```

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

As seen above the duplicates have been removed and we are now looking at the first 5 rows to ensure data integrity

```
df.count()
```

```
Make            10925
Model           10925
Year            10925
HP              10856
Cylinders       10895
Transmission    10925
Drive Mode      10925
MPG-H           10925
MPG-C           10925
Price           10925
dtype: int64
```

We then have to remove the null values within the dataset

```
print(df.isnull().sum())
```

```
Make             0
Model            0
Year             0
HP              69
Cylinders       30
Transmission     0
Drive Mode       0
MPG-H            0
MPG-C            0
Price            0
dtype: int64
```

```
df = df.dropna()     # Dropping the missing values.
df.count()
```

```
Make            10827
Model           10827
Year            10827
HP              10827
Cylinders       10827
Transmission    10827
Drive Mode      10827
MPG-H           10827
MPG-C           10827
Price           10827
dtype: int64
```

```
print(df.isnull().sum())    # After dropping the values
```

```
Make            0
Model           0
Year            0
HP              0
Cylinders       0
Transmission    0
Drive Mode      0
MPG-H           0
MPG-C           0
Price           0
dtype: int64
```

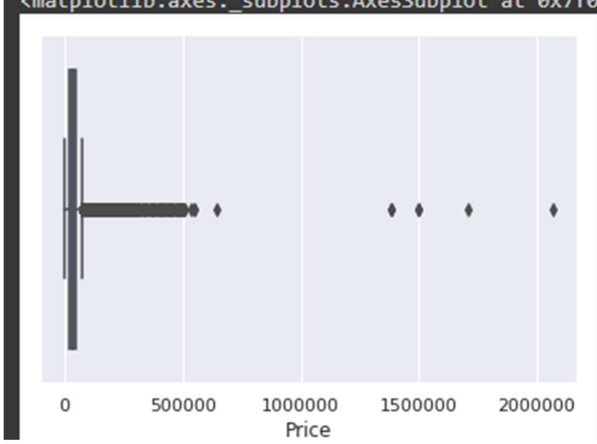
The above shows the amount of null values which is now 0 because they have been removed

The next step involves identifying outliers, it's a good idea to remove them because they are one of the primary reasons for resulting in a less accurate model.

Outliers can be seen with visualisers using a boxplot. Below is a boxplot of MSRP, Cylinders, Horsepower and engine size

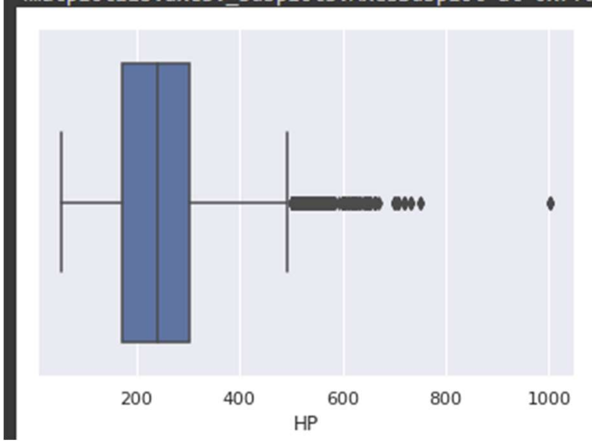
```
sns.boxplot(x=df['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0d36a38be0>
```



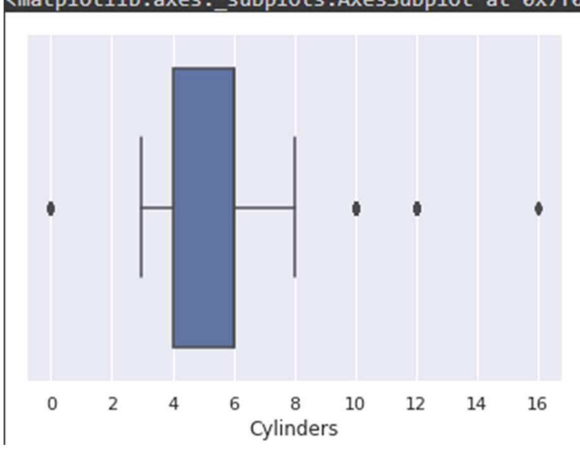
```
sns.boxplot(x=df['HP'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0d369b3ba8>
```




```
sns.boxplot(x=df['Cylinders'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0d3413ff28>
```




```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Year             9.0
HP             130.0
Cylinders        2.0
MPG-H            8.0
MPG-C            6.0
Price        21327.5
dtype: float64
```

Don't worry about the above values because it's not important to know each and every one of them because it's just important to know how to use this technique in order to remove the outliers.

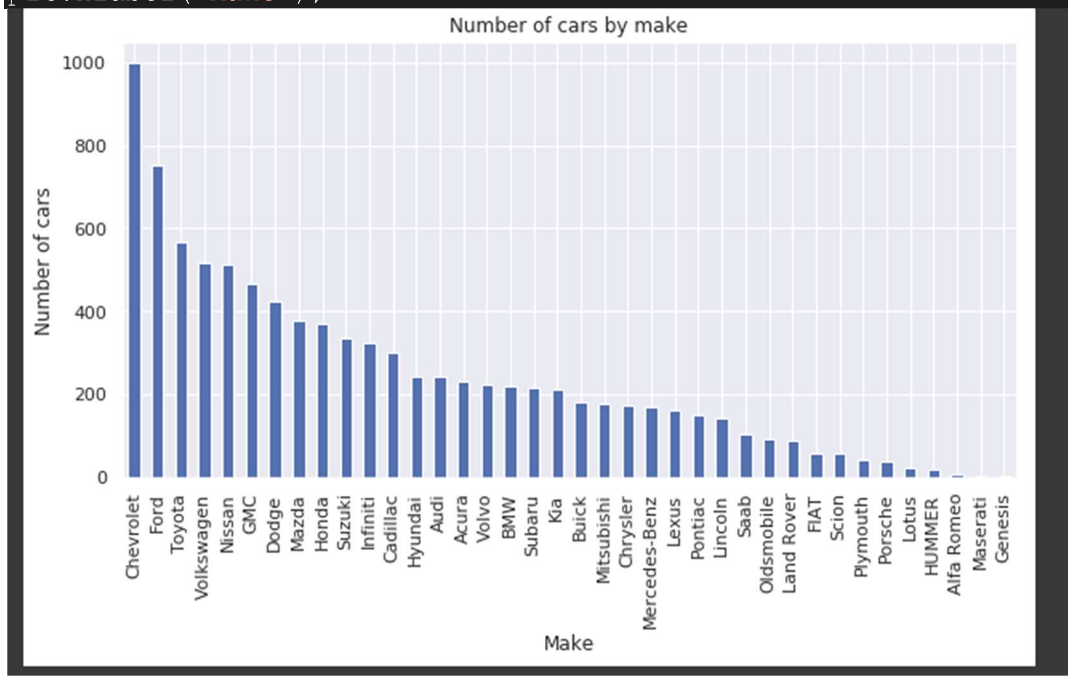
```
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df.shape
```

```
(9191, 10)
```

We can plot different features against one another – scatter against histogram

Histogram – the frequency of occurrence of variables in an interval

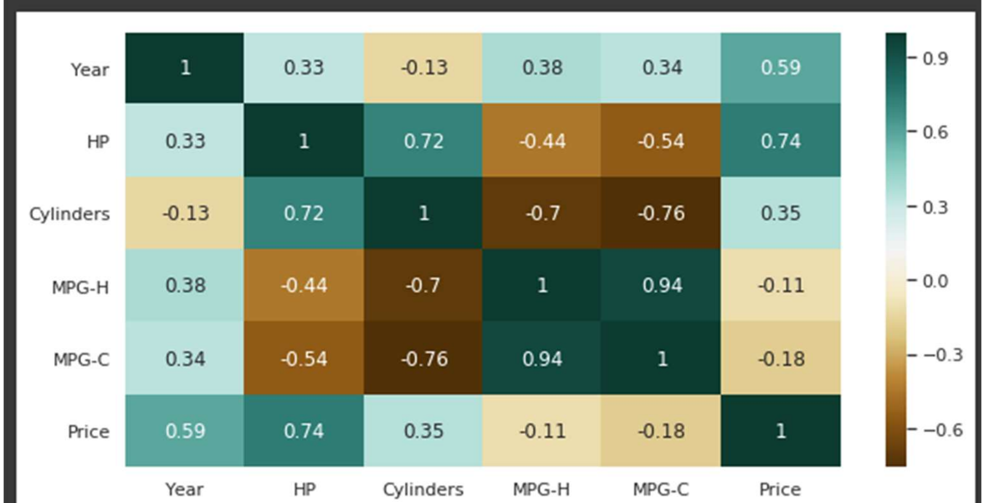
```
df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```



Heat maps - Heat Maps is a type of plot which is necessary when we need to find the dependent variables. One of the best way to find the relationship between the features can be done using heat maps.

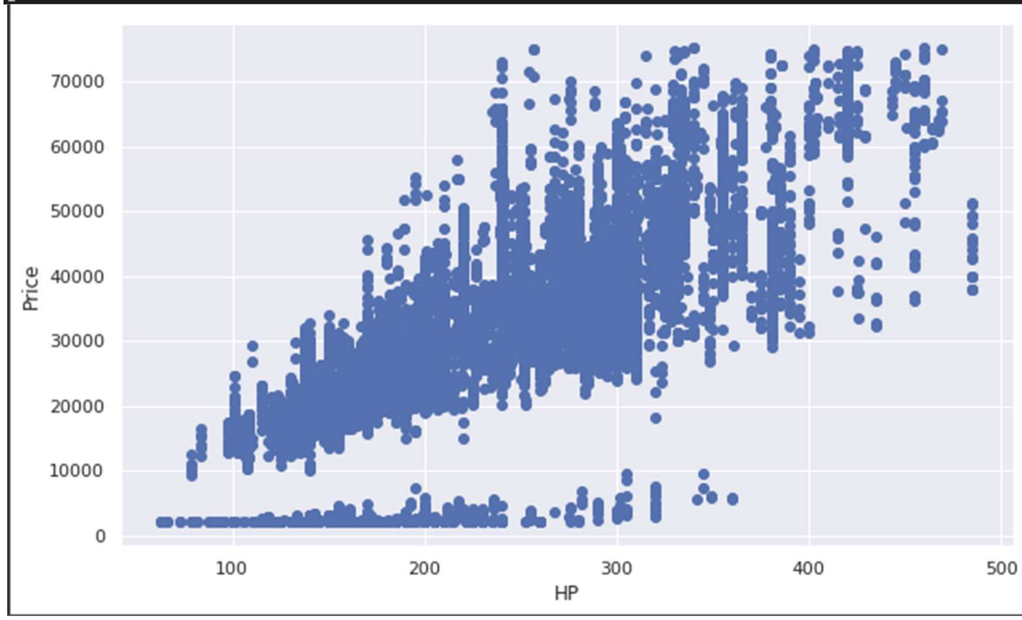
```
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
```

	Year	HP	Cylinders	MPG-H	MPG-C	Price
Year	1.000000	0.326726	-0.133920	0.378479	0.338145	0.592983
HP	0.326726	1.000000	0.715237	-0.443807	-0.544551	0.739042
Cylinders	-0.133920	0.715237	1.000000	-0.703856	-0.755540	0.354013
MPG-H	0.378479	-0.443807	-0.703856	1.000000	0.939141	-0.106320
MPG-C	0.338145	-0.544551	-0.755540	0.939141	1.000000	-0.180515
Price	0.592983	0.739042	0.354013	-0.106320	-0.180515	1.000000



Scatterplot - We generally use scatter plots to find the correlation between two variables. With the plot given below, we can easily draw a trend line. These features provide a good scattering of points.

```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['HP'], df['Price'])
ax.set_xlabel('HP')
ax.set_ylabel('Price')
plt.show()
```



End of Tutorial

Now, undertake similar EDA with Auto-mpg dataset:

- Identify missing values.
- Estimate Skewness and Kurtosis.
- Correlation Heat Map.
- Scatter plot for different parameters.
- Replace categorical values with numerical values (i.e., America 1, Europe 2 etc.).

Step 1 involves loading all of the libraries needed to undertake EDA with the Auto-mpg dataset

```
import pandas as pd
import numpy as np
import seaborn as sns                          #visualisation
import matplotlib.pyplot as plt                #visualisation
%matplotlib inline
sns.set(color_codes=True)
```

We the read the csv using pandas and we also check the first 5 rows of the data to ensure thatg it is correct.

```
df = pd.read_csv("auto-mpg.csv")
# to display the first 5 rows
df.head(5)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

The below is count code to identify the amount of rows we have of data

```
df.count()
```

```
mpg             398
cylinders       398
displacement    398
horsepower      398
weight          398
acceleration    398
model year      398
origin          398
car name        398
dtype: int64
```

The below is code that helps identify the type of data

```
df.dtypes
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

We now want to identify and remove the missing values as show with the below code

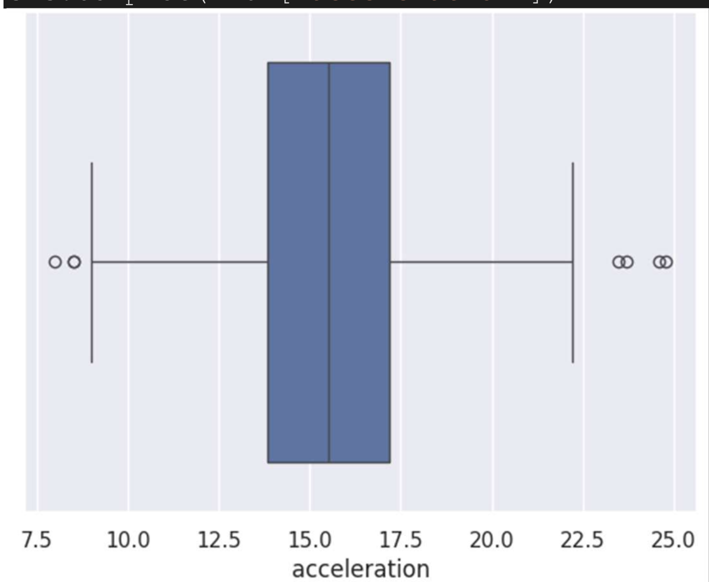
```
df = df.dropna() # removing the missing values
df.count()
```

```
mpg             398
cylinders       398
displacement    398
horsepower      398
weight          398
acceleration    398
model year      398
origin          398
car name        398
dtype: int64
```

As seen above there are no missing values within the dataset as the count remains the same

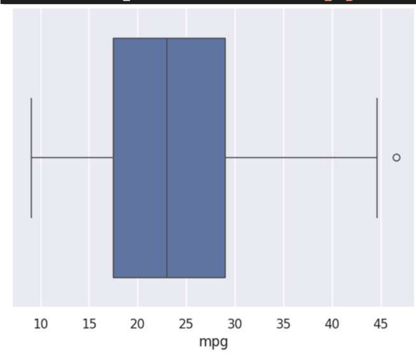
We then need to calculate the skewdness which can be done through a **boxplot** of each category a few examples can be seen below

```
sns.boxplot(x=df['acceleration'])
```



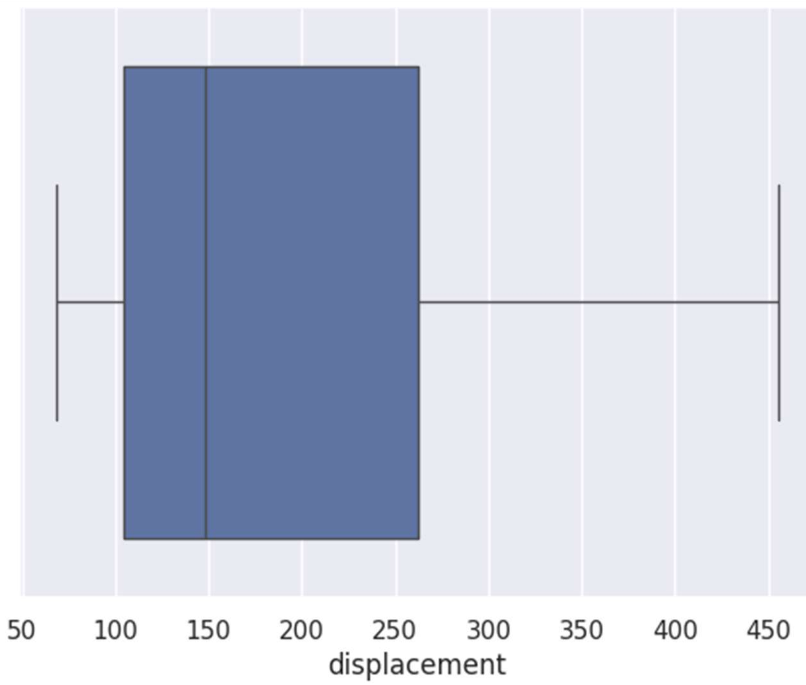
This shows normal distribution within the acceleration category with outliers on each side of the minimum and maximum whiskers of the boxplot.

```
sns.boxplot(x=df['mpg'])
```



This shows more of a positive skew as the box plot is slightly shifted to the left. Outliers remain beyond the maximum whisker'

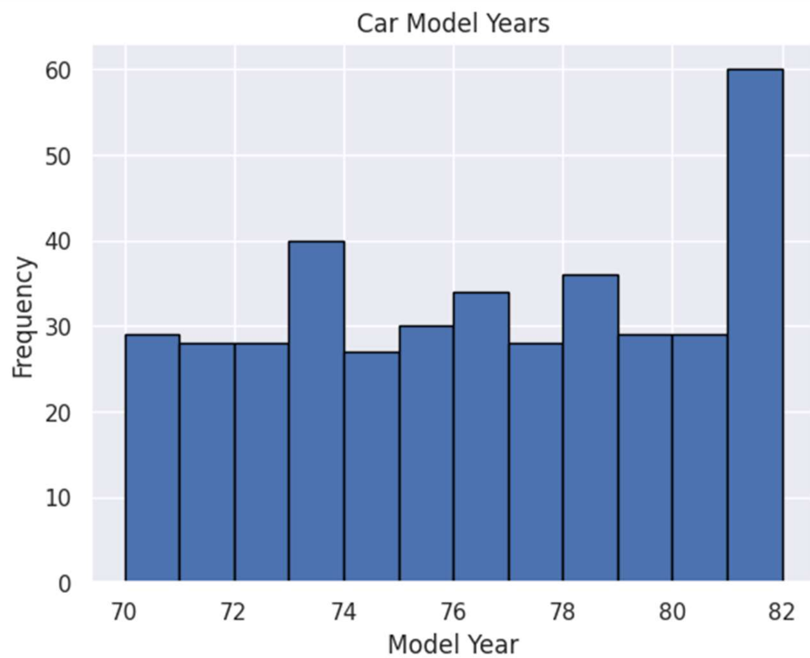
```
sns.boxplot(x=df['displacement'])
```



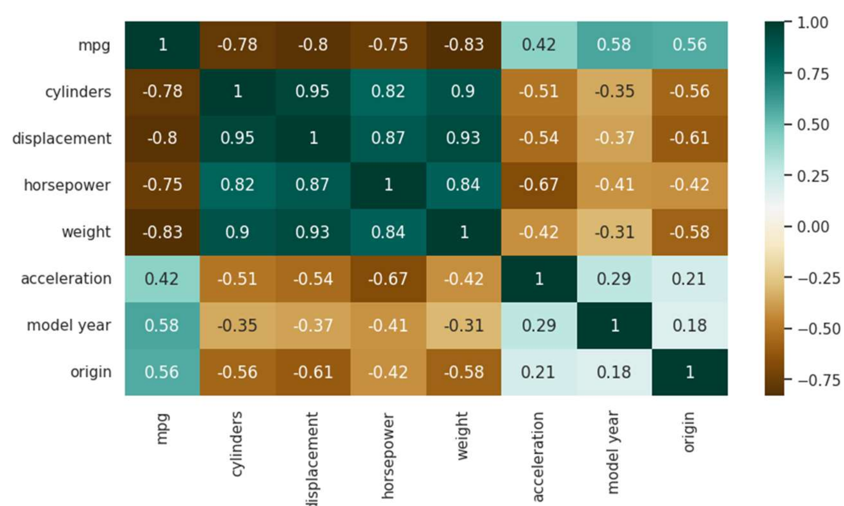
This also shows a positive skew as the boxplot has a shift to the left.

The kurtosis can be done through a **histogram** which shows that there is a higher frequency of cars in between the years 1981 - 1982

```
df['model year'].hist(bins=range(int(df['model year'].min()),
int(df['model year'].max()) + 1), edgecolor='black')
plt.title('Car Model Years')
plt.xlabel('Model Year')
plt.ylabel('Frequency')
```

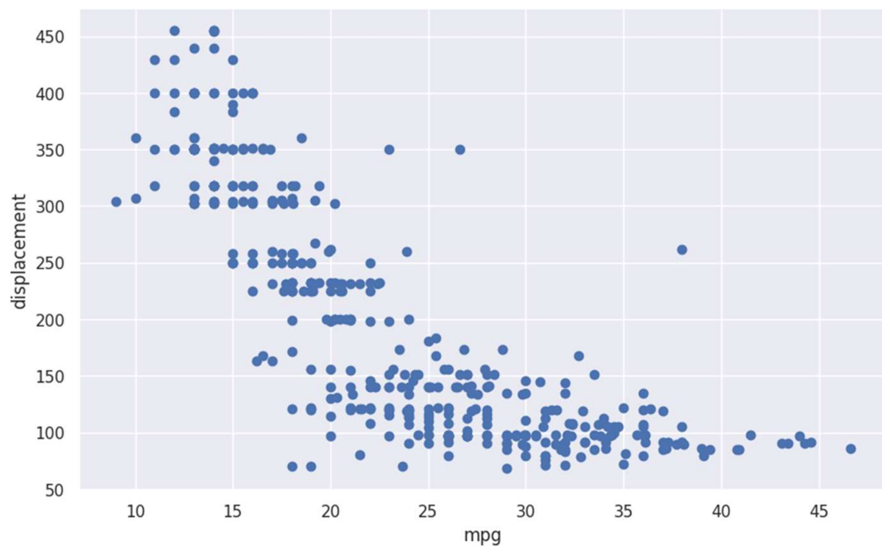


To determine the correlation a correlation matirix can be produced with the added feature of a heatmap to determine the skewedness of the correlation with darker blue cells showing a higher positive correlation and darker brown cells showing a stronger negative correlation.



A scatter plot can help find the correlation between two variables an example of one can be seen below between displacement within the vehicle and mpg. We can see that there is a negative correlation. Another example where there isn't a correlation can be seen when we have acceleration on the x axis and model year on the y axis.

```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['mpg'], df['displacement'])
ax.set_xlabel('mpg')
ax.set_ylabel('displacement')
plt.show()
```



```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['acceleration'], df['model year'])
ax.set_xlabel('acceleration')
ax.set_ylabel('model year')
plt.show()
```