Report

Task 1: Data Acquisition and Preparation

Data Acquisition

In the above task first, we must load the three datasets into different data frames using pandas. Since the data1 and data2 were related to each other which made it easy to merge both the dataset into a single data frame based on unique value 'ID'. The next task was to merge the derived data frame with the data3 file. After performing the final merging, the dataset has 199 entries and a total of 27 columns. In this process the first two datasets were merged using the merge() function and the data3 were merged using the Concat() function.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 27 columns):
id                   199 non-null int64
symboling            199 non-null int64
normalised-losses    161 non-null float64
make                 199 non-null object
fuel-type            199 non-null object
aspiration           199 non-null object
num-of-doors         199 non-null object
body-style           199 non-null object
drive-wheels         199 non-null object
engine-location      199 non-null object
wheel-base           199 non-null float64
length               199 non-null float64
width                199 non-null float64
height               199 non-null float64
curb-weight          199 non-null int64
engine-type          199 non-null object
num-of-cylinders     199 non-null object
engine-size          199 non-null int64
fuel-system          199 non-null object
bore                 199 non-null float64
stroke               199 non-null float64
compression-ratio    199 non-null float64
horsepower           199 non-null int64
peak-rpm             199 non-null int64
city-mpg             199 non-null int64
highway-mpg          199 non-null int64
price                195 non-null float64
dtypes: float64(9), int64(8), object(10)
memory usage: 42.0+ KB
```

Information of Merged data frame

Data Preparation

In this process, we are supposed to check the data and perform certain steps to clean the given data. The process involved in the cleaning of the merged dataset is as follows: -

Processing of Duplicate Value: -

The first step was to find the duplicate value in the merged dataset 'df_final'. It was observed that Id: - 10180 was repeated thrice and which acted as a duplicate value.

Out[12]:

	id	symboling	normalised-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
116	10180	-1	90.0	toyota	gas	std	four	sedan	rwd	front	...	171	mpfi	3.27	3.35	9.2	156
154	10180	-1	90.0	toyota	gas	std	four	sedan	rwd	front	...	171	mpfi	3.27	3.35	9.2	156
180	10180	-1	90.0	toyota	gas	std	four	sedan	rwd	front	...	171	mpfi	3.27	3.35	9.2	156

3 rows × 27 columns

Duplicate value ID:- 10180

So, to remove the duplicate value there is a function called drop_duplicates which drops the duplicate values from the data frame. The drop_duplicates function where we specified a condition to drop duplicates value based on the id.

df_final.drop_duplicates(subset = 'id' , inplace = True)

Out[14]:

	id	symboling	normalised-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	pe
116	10180	-1	90.0	toyota	gas	std	four	sedan	rwd	front	...	171	mpfi	3.27	3.35	9.2	156	5

1 rows × 27 columns

Record after dropping duplicates

Dealing with the White Space: -

There were certain whitespaces observed inside the column fuel type. The first step was to identify the unique values inside the categorical column fuel type and then we used replace() function replace the 'diesel ' with 'diesel'.

```
In [19]: 1 df_final['fuel_type'].unique()
Out[19]: array(['gas', 'diesel', 'diesel '], dtype=object)

In [20]: 1 df_final['fuel_type'].replace('diesel ','diesel' ,inplace = True)

In [21]: 1 df_final['fuel_type'].unique()
Out[21]: array(['gas', 'diesel'], dtype=object)
```

Handling the Typos: -

In a certain column, the data was observed to have typing error. The column fuel system had typos with two similar values represented in different formats like 'Mpfi' and 'mpfi'. So, replace function was used to replace these kinds of values.

```
In [37]: 1 df_final['fuel_system'].unique()
Out[37]: array(['mpfi', '2bbl', 'mfi', '1bbl', 'Mpfi', 'idi', 'spdi', 'spfi'],
               dtype=object)

In [38]: 1 df_final['fuel_system'].replace('Mpfi','mpfi', inplace = True)

In [39]: 1 print df_final['fuel_system'].unique()
['mpfi' '2bbl' 'mfi' '1bbl' 'idi' 'spdi' 'spfi']
```

Handling Typos

In certain data, we also observed the combination of numerical as well as object like in a column named num_of_doors had values 2 and 4 instead of two and four. So, the same process was carried out to replace the numerical value into objects.

```
In [31]: 1 df_final['num_of_doors'].unique()
Out[31]: array([4L, 2L, 'two', 'four'], dtype=object)

In [32]: 1 df_final['num_of_doors'].replace(2,'two', inplace =True)

In [33]: 1 df_final['num_of_doors'].unique()
Out[33]: array([4L, 'two', 'four'], dtype=object)

In [34]: 1 df_final['num_of_doors'].replace(4,'four',inplace= True)

In [35]: 1 df_final['num_of_doors'].unique()
Out[35]: array(['four', 'two'], dtype=object)
```

Dealing the Null values: -

The data was observed to have null values in columns like normalised losses and price. It was observed that normalised losses had 38 null values and the price column had 4 null values. Since the value associated with the normalised losses were closely related to certain categorical data like make, body style, aspiration. The normalised_losses were more closely related to the individual make. So, first the mean was taken by grouping the value of normalised losses on the basis of make and then those values were filled instead of null values.

```
In [25]: df_final['normalised_losses'] = df_final.groupby(['make'])['normalised_losses'].transform(lambda x:x.fillna(x.mean()))
         print df_final
```

169	17912	1	NaN	isuzu	gas	std	
170	15023	0	190.000000	bmw	gas	std	
171	19840	3	197.000000	toyota	gas	std	
172	29057	2	134.000000	toyota	gas	std	
173	31313	2	122.000000	volkswagen	gas	std	
174	28028	3	186.000000	porsche	gas	std	
175	31143	3	146.200000	mitsubishi	gas	turbo	
176	35593	-1	65.000000	toyota	gas	std	
177	10749	1	190.000000	bmw	gas	std	
178	28520	1	NaN	mercury	gas	turbo	
179	34603	0	91.000000	toyota	diesel	std	
181	29693	-2	103.000000	volvo	gas	turbo	
182	18229	2	94.000000	volkswagen	diesel	turbo	
183	13205	-1	65.000000	toyota	gas	std	
184	13669	0	77.000000	toyota	gas	std	
185	35136	-1	110.290323	toyota	gas	std	
186	38153	1	168.000000	toyota	gas	std	
187	32794	0	102.000000	subaru	gas	turbo	
188	35224	0	102.000000	subaru	gas	std	
189	33955	0	121.250000	volkswagen	diesel	turbo	
190	16229	0	161.000000	audi	gas	turbo	

Filling the normalised losses by a mean value

Since there were certain data in ‘make’ column which never had any ‘normalised losses’ value. Data like Isuzu, mercury never had any value for normalised losses. So, the mean of normalised losses was taken to fill the null values.

The price column to had certain null values and the same method was implemented were first the mean was taken by grouping the value of price based on make and then those values were filled instead of null values.

```
In [26]: df_final['price'] =df_final.groupby(['make'])['price'].transform(lambda x:x.fillna(x.mean()))
         print df_final
```

197	mpfi	3.19	3.40	9.00	85	5250	
198	idi	3.01	3.40	23.00	106	4800	

	city_mpg	highway_mpg	price
0	24	30	13950.000000
1	18	22	17450.000000
2	19	25	17710.000000
3	17	20	23875.000000
4	23	29	16430.000000
5	23	29	16925.000000
6	21	28	20970.000000
7	21	28	21105.000000
8	47	53	5151.000000
9	38	43	6295.000000
10	38	43	6575.000000
11	37	41	5572.000000
12	31	38	6377.000000
13	24	30	7957.000000
14	31	38	6229.000000
15	31	38	6692.000000

Filling the price by a mean value

Dealing the impossible values

The data eventually had one impossible value which was out of a specified range in the dataset requirement. One kind of data was horsepower which was having a value of 1100 which was out of specified range of 48 to 288. The model with 1100 horsepower was falling in the category of Audi and the body style was a sedan.

Out[44]:

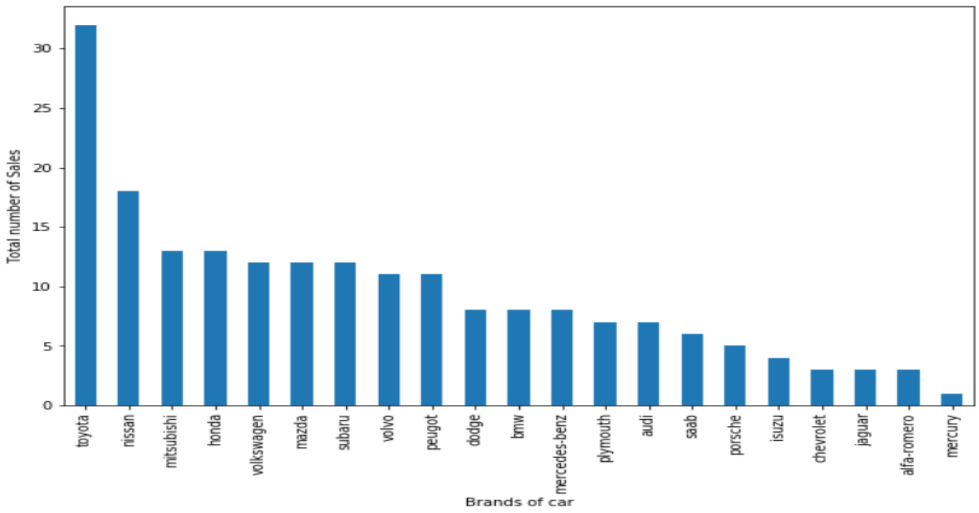
dy_style	drive_wheels	engine_location	...	engine_size	fuel_system	bore	stroke	compression_ratio	horsepower	peak_rpm	city_mpg	highway_mpg	price
sedan	fwd	front	...	109	mpfi	3.19	3.4	10.0	102	5500	24	30	13950.0
sedan	4wd	front	...	136	mpfi	3.19	3.4	8.0	115	5500	18	22	17450.0
sedan	fwd	front	...	136	mpfi	3.19	3.4	8.5	1100	5500	19	25	17710.0
sedan	fwd	front	...	131	mpfi	3.13	3.4	8.3	140	5500	17	20	23875.0
sedan	fwd	front	...	136	mpfi	3.19	3.4	8.5	110	5500	19	25	15250.0

The data with horsepower 1100 was satisfying certain match with the car having horsepower 110 If we compare the horsepower of id: -28588 with 16528 which has fallen under the categories of sedan and certain other similar properties like the fuel system, bore, stroke, compression ratio. We can also consider parameter of price which falls under same range. So, the value of horsepower 1100 was replaced with 110 considering it as a data entry error.

TASK 2: Data Exploration

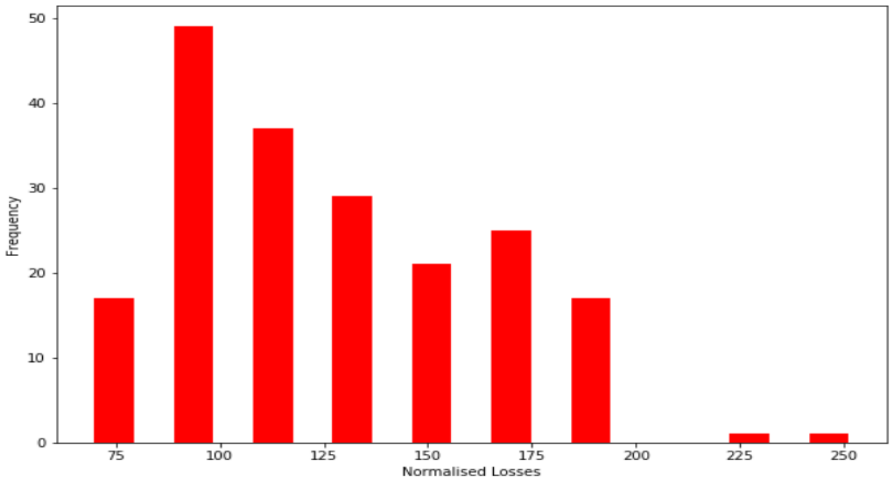
Task 2.1

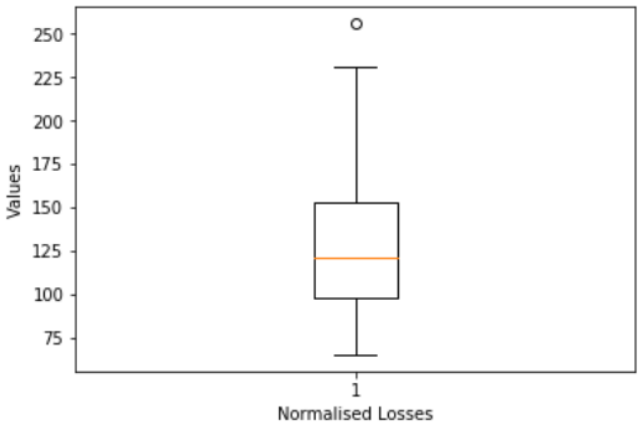
Graphical representation of a categorical column



The above graph describes the total number of cars sold in each category of “make”. In this graph, we used the categorical column of ‘make’ and we counted the sales of each make to plot the above bar graph. The reason for using the bar graph is to compare the result with the different categories of data inside the make column. So, from the graph we can conclude that Toyota has most sold cars and mercury has the least car sold in the market.

Graphical representation of a numerical column





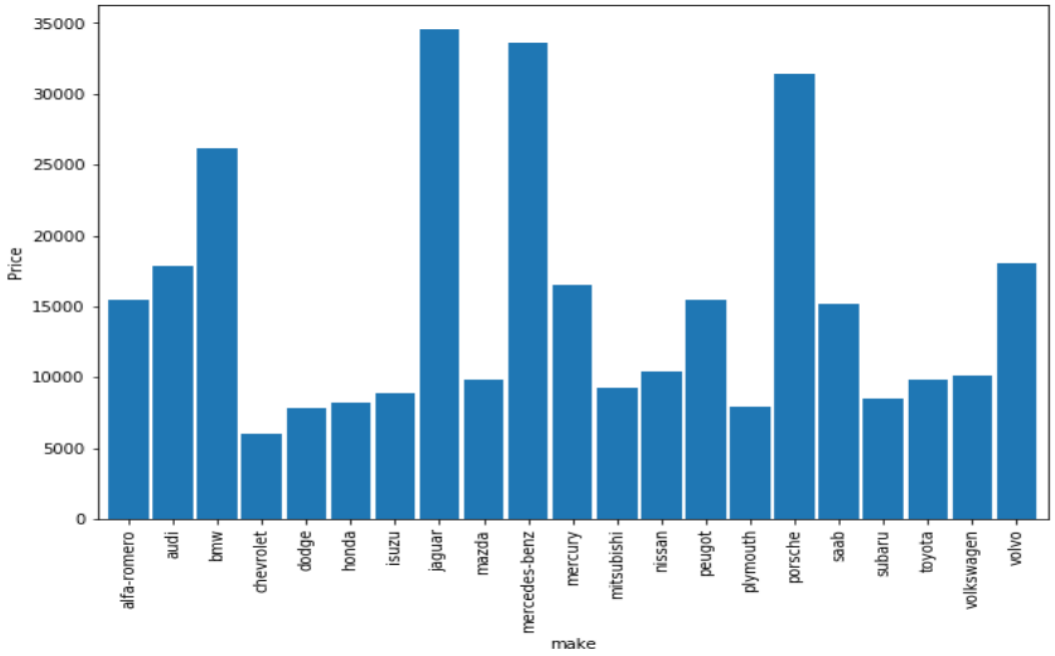
The histogram represents the frequency of losses observed in the 'normalised_losses' column. The boxplot represents the values of descriptive statistics like max, min, median and interquartile range of column 'normalised_losses'. The histogram and boxplot are used to represent the continuous data from a dataset. The boxplot also shows the outlier inside the specified column from a dataset. In this graph, we can observe a normalised loss with value greater than 250 which is termed to be an outlier.

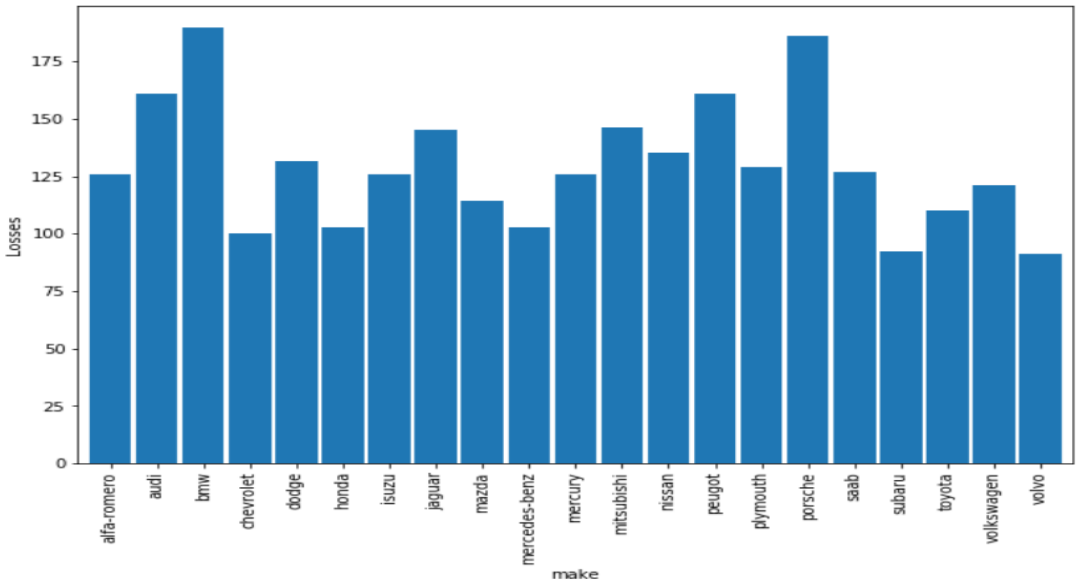
Task 2.2

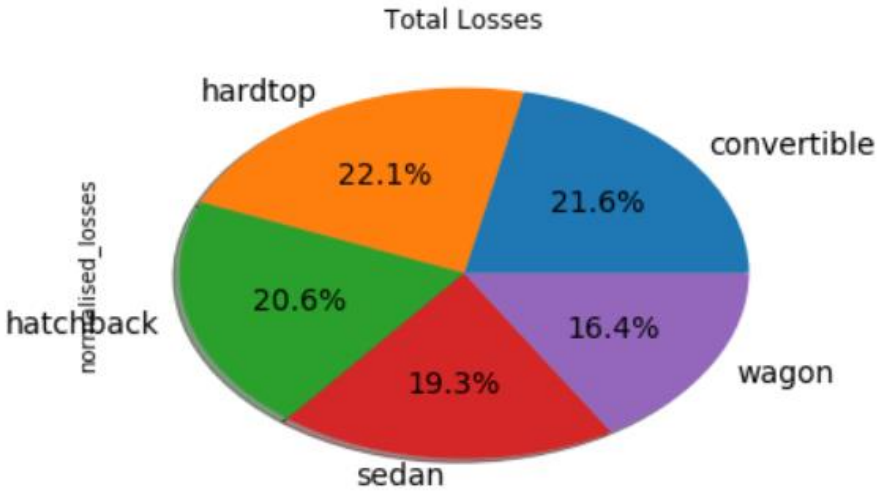
The three pairs of columns choose in this task are: -

- Make and Price
- Make and Normalised Losses
- Body style and Normalised Losses

The reason behind choosing these columns is to determine which make is most affordable and has less normalised losses observed. Which kind of body style has less percentage of losses?


Price
35000
30000
25000
20000
15000
10000
5000
0
alfa-romero
audi
bmw
chevrolet
dodge
honda
isuzu
jaguar
mazda
mercedes-benz
mercury
mitsubishi
nissan
peugot
plymouth
porsche
saab
subaru
toyota
volkswagen
volvo
make

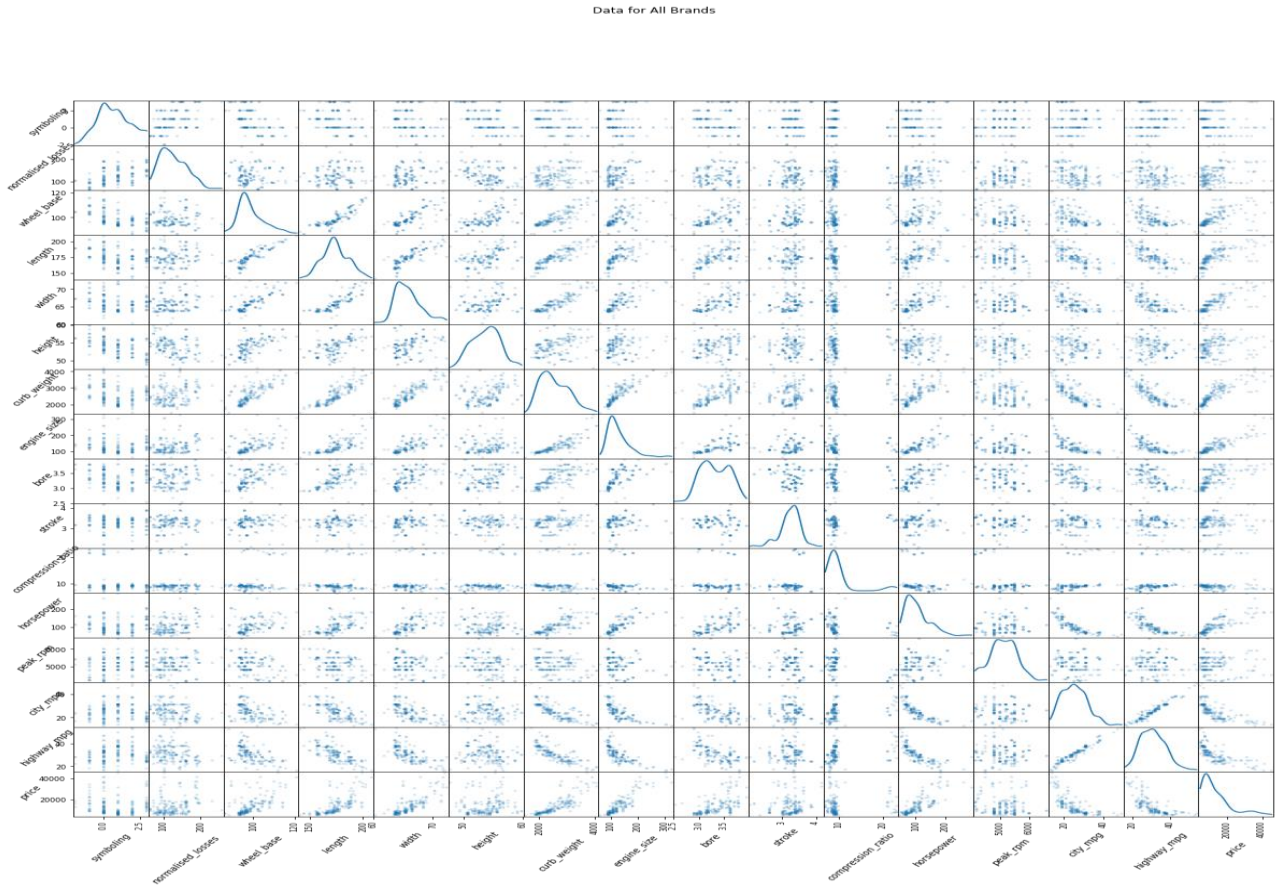

Losses
175
150
125
100
75
50
25
0
alfa-romero
audi
bmw
chevrolet
dodge
honda
isuzu
jaguar
mazda
mercedes-benz
mercury
mitsubishi
nissan
peugot
plymouth
porsche
saab
subaru
toyota
volkswagen
volvo
make


Total Losses
normalised_losses
hardtop
22.1%
convertible
21.6%
hatchback
20.6%
16.4%
wagon
19.3%
sedan

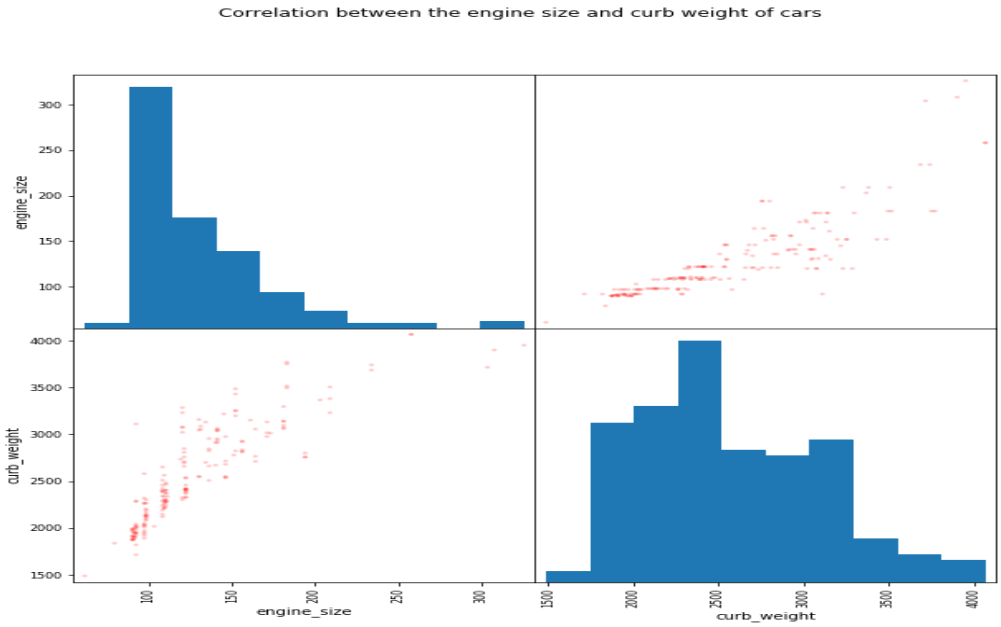
Observation: -

The first graph where we can observe that Chevrolet can be termed as an affordable brand but while observing the second graph, we can make a conclusion that Chevrolet cars had encountered about 50% of losses in insurance. It is observed that the best and affordable brand is one which has fewer losses. So, brand Subaru has fewer losses as compared to other major brands and the average price of the Subaru brand is almost like Chevrolet. When we observe the pie chart, we can conclude that wagon body style has lesser losses. It is observed that Subaru has wagon style cars. So, finally it is advisable to look for Subaru with wagon body style which is affordable and has fewer losses.

Task 2.3



Scatter Matrix



The above part of a scatter matrix plot shows the relationship between the curb weight and engine size. There exists a correlation between the two parameters because we can observe that when the engine size increases automatically there is an increase in the curb weight of a car is observed. Since the curb weight of a car depends on various other factors and engine size plays a major role in the increase in cars curb weight. So, we can conclude that there exists a positive correlation between engine size and curb weight.