Report On

Data Cleaning and Summarizing (Automobile.csv)

By - Shubhdeep Singh (s3764546)

1. Data Preparation

1.1 Handling Null Values –

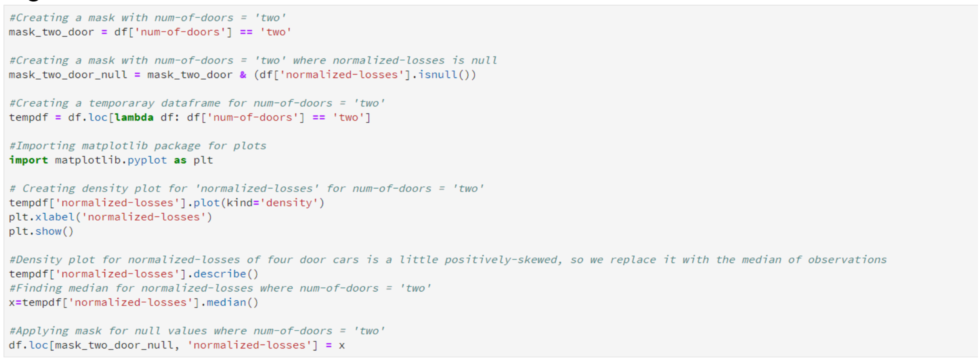
The automobile dataset (df) contained null values for certain features that required special attention. A simple ‘df.isna().sum()’ command provided with the sum of null values present within each column.

Normalized-losses – The column ‘normalized-losses’, in specific, contains a significant number (47) null values. Dropping all the null values could’ve been easy but it would’ve made us lose potential data. The way around was to replace the null values. Since most of the values are missing, the mean or median for the whole column wouldn’t have been the solution. ‘Normalized-losses’ indicates the average loss of value per insured vehicle year. The value is normalized over the range of vehicle sizes. This indicates that a vehicle belonging to a specific group defined by the size (two-door small, station wagons, sports/speciality, etc...) of the vehicle should have somewhat similar values for ‘normalized-losses’. The only column available for relating to the above description was ‘num-of-doors’. As such the value for null (NaN) for ‘normalised-losses’ was determined by grouping together the vehicles with ‘two’ and ‘four’ doors respectively.

Potential issues/errors –
1. ‘num-of-doors’ include null values.
2. Typos such as ‘fourR’ and upper case letters ‘Four’
3. Adjoining whitespace with the values.

Determined using - df['num-of-doors'].value_counts()
Solution – Dropped the tuples with null values. Replaced the typos with correct values and removed white space using ‘Series.str.strip()’.

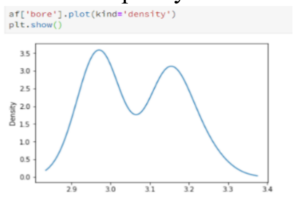
After cleaning the data for ‘num-of-doors’ column, we were able to successfully create a mask for ‘num-of-doors’ = ‘two’ and ‘num-of-doors’ = ‘four’. For each mask we created a temporary mask for null values followed by creating the density plots for ‘normalised-losses’ for both classes of doors. The plots were positively skewed. As a result the null values were replaced by the median of ‘normalised-losses’ for the corresponding ‘num-of-doors’ classes.



```python
#Creating a mask with num-of-doors = 'two'
mask_two_door = df['num-of-doors'] == 'two'

#Creating a mask with num-of-doors = 'two' where normalized-losses is null
mask_two_door_null = mask_two_door & (df['normalized-losses'].isnull())

#Creating a temporaray dataframe for num-of-doors = 'two'
tempdf = df.loc[lambda df: df['num-of-doors'] == 'two']

#Importing matplotlib package for plots
import matplotlib.pyplot as plt

# Creating density plot for 'normalized-losses' for num-of-doors = 'two'
tempdf['normalized-losses'].plot(kind='density')
plt.xlabel('normalized-losses')
plt.show()

#Density plot for normalized-losses of four door cars is a little positively-skewed, so we replace it with the median of observations
tempdf['normalized-losses'].describe()
#Finding median for normalized-losses where num-of-doors = 'two'
x=tempdf['normalized-losses'].median()

#Applying mask for null values where num-of-doors = 'two'
df.loc[mask_two_door_null, 'normalized-losses'] = x
```

Figure 1.1 Replacing null values for ‘normalized-losses’ with the median of value grouped by ‘num-of-doors’

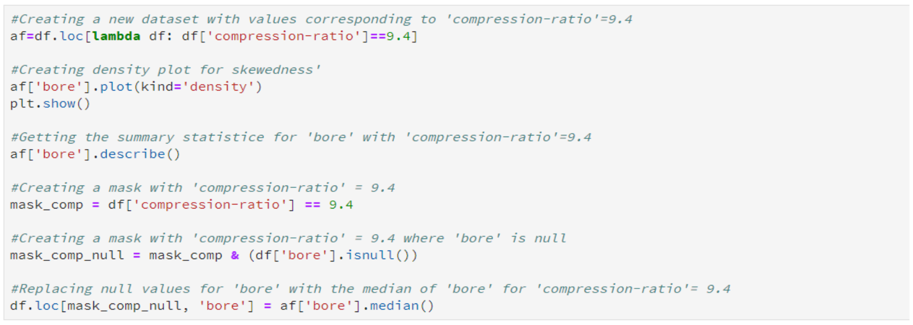
Bore – The ‘bore’ of an engine is the diameter of the cylinder in which the piston of an internal combustion engine travels. ‘Compression-ratio’, another feature available in the dataset is the ratio of largest capacity to the smallest capacity of the combustion chamber of an internal combustion engine. As such, the ‘compression-ratio’ and ‘bore’ must have a relation. Fortunately, all the null ‘bore’ values have the same ‘compression-ratio’. With limited domain knowledge, it should be safe to assume that the null ‘bore’ values can be determined by one of the summary-statistics on the dataset of vehicles with ‘compression-ratio’= 9.4. Also, there are no outliers present in the ‘bore’ as per the box-plot. So it will be valid to replace null values with some measure of central tendency. Furthermore, guessing the null values from a smaller distribution would provide us with a better estimate for the values. The density plot for the same is shown resembling a multi-modal plot.



```python
af['bore'].plot(kind='density')
plt.show()
```

Figure 1.2 Density plot for bore with compression-ratio=9.4

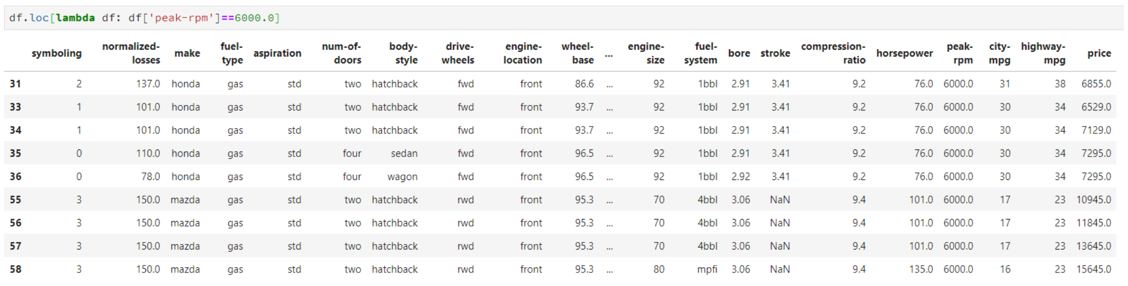
The Series.describe() function shows us the mean and median values close to each other. So we replace the null values for ‘bore’ with the median of rows with ‘compression-ratio’= 9.4.



```
#Creating a new dataset with values corresponding to 'compression-ratio'=9.4
af=df.loc[lambda df: df['compression-ratio']==9.4]

#Creating density plot for skewedness'
af['bore'].plot(kind='density')
plt.show()

#Getting the summary statistice for 'bore' with 'compression-ratio'=9.4
af['bore'].describe()

#Creating a mask with 'compression-ratio' = 9.4
mask_comp = df['compression-ratio'] == 9.4

#Creating a mask with 'compression-ratio' = 9.4 where 'bore' is null
mask_comp_null = mask_comp & (df['bore'].isnull())

#Replacing null values for 'bore' with the median of 'bore' for 'compression-ratio'= 9.4
df.loc[mask_comp_null, 'bore'] = af['bore'].median()
```

Figure 1.3 Replacing null values for ‘bore’ with median of ‘bore’ when ‘compression-ratio’=9.4

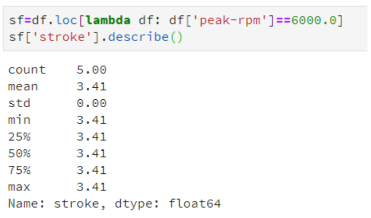
Stroke – Analogous to ‘bore’, the ‘stroke’ of an automobile is related to the ‘peak-rpm’. Stroke-length usually is the measure of the length travelled by the piston inside the combustion chamber of an internal combustion engine. As per the domain knowledge, smaller the stroke, larger would be the peak-rpm produced. Favourably enough, the ‘peak-rpm’ values corresponding to the null ‘stroke’ values are same. Grouping the values with ‘peak-rpm’=6000.0, we assume that the ‘null’ values for stroke can be determined by using one of the measures of central tendency.



```
df.loc[lambda df: df['peak-rpm']==6000.0]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
31	2	137.0	honda	gas	std	two	hatchback	fwd	front	86.6	...	92	1bbl	2.91	3.41	9.2	76.0	6000.0	31	38	6855.0
33	1	101.0	honda	gas	std	two	hatchback	fwd	front	93.7	...	92	1bbl	2.91	3.41	9.2	76.0	6000.0	30	34	6529.0
34	1	101.0	honda	gas	std	two	hatchback	fwd	front	93.7	...	92	1bbl	2.91	3.41	9.2	76.0	6000.0	30	34	7129.0
35	0	110.0	honda	gas	std	four	sedan	fwd	front	96.5	...	92	1bbl	2.91	3.41	9.2	76.0	6000.0	30	34	7295.0
36	0	78.0	honda	gas	std	four	wagon	fwd	front	96.5	...	92	1bbl	2.92	3.41	9.2	76.0	6000.0	30	34	7295.0
55	3	150.0	mazda	gas	std	two	hatchback	rwd	front	95.3	...	70	4bbl	3.06	NaN	9.4	101.0	6000.0	17	23	10945.0
56	3	150.0	mazda	gas	std	two	hatchback	rwd	front	95.3	...	70	4bbl	3.06	NaN	9.4	101.0	6000.0	17	23	11845.0
57	3	150.0	mazda	gas	std	two	hatchback	rwd	front	95.3	...	70	4bbl	3.06	NaN	9.4	101.0	6000.0	17	23	13645.0
58	3	150.0	mazda	gas	std	two	hatchback	rwd	front	95.3	...	80	mpfi	3.06	NaN	9.4	135.0	6000.0	16	23	15645.0

Figure 1.4 Values with ‘peak-rpm’=6000

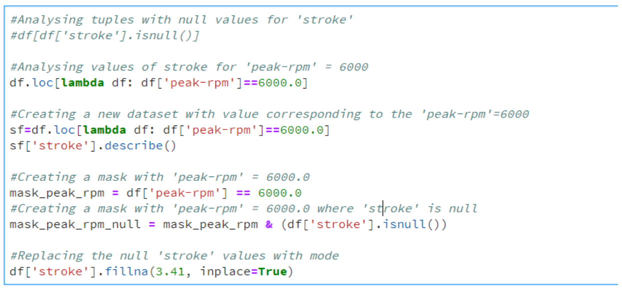


```
sf=df.loc[lambda df: df['peak-rpm']==6000.0]
sf['stroke'].describe()
```

```
count    5.00
mean     3.41
std      0.00
min      3.41
25%      3.41
50%      3.41
75%      3.41
max      3.41
Name: stroke, dtype: float64
```

Figure 1.5 Summary statistics for ‘stroke’ with ‘peak-rpm’=6000

It was quite evident from the values that the mode (3.41) of the ‘stroke’ value for ‘peak-rpm’=60000 could be used to replace the null values since it is the most frequently occurring value.



```
#Analysing tuples with null values for 'stroke'
#df[df['stroke'].isnull()]

#Analysing values of stroke for 'peak-rpm' = 6000
df.loc[lambda df: df['peak-rpm']==6000.0]

#Creating a new dataset with value corresponding to the 'peak-rpm'=6000
sf=df.loc[lambda df: df['peak-rpm']==6000.0]
sf['stroke'].describe()

#Creating a mask with 'peak-rpm' = 6000.0
mask_peak_rpm = df['peak-rpm'] == 6000.0
#Creating a mask with 'peak-rpm' = 6000.0 where 'stroke' is null
mask_peak_rpm_null = mask_peak_rpm & (df['stroke'].isnull())

#Replacing the null 'stroke' values with mode
df['stroke'].fillna(3.41, inplace=True)
```

Figure 1.6 Replacing the null ‘stroke’ values with the mode when ‘peak-rpm’=6000

Horsepower & peak-rpm – ‘Horsepower’ is the measure of the maximum power of an engine. The ‘peak-rpm’ is the measure of an engine that specifies the rpm with the maximum power. Horsepower depends on the size of an engine. As such we tried to analyse the engines with the same size as those with null (NaN) values for horsepower that is 132. In our dataset, however, there were no similar records for the engine-size = 132. Also the values for ‘normalized-losses’ for these rows were the null values that we replaced earlier with the median while cleaning the same. We assumed it would be better to drop these tuples since no other measure is relatable but this would incur additional cost of losing the records for the make ‘renault’.

```
df.loc[lambda df: df['engine-size']==132]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
130	0	95.0	renault	gas	std	four	wagon	fwd	front	96.1	...	132	mpfi	3.46	3.9	8.7	NaN	NaN	23	31	9295.0
131	2	134.0	renault	gas	std	two	hatchback	fwd	front	96.1	...	132	mpfi	3.46	3.9	8.7	NaN	NaN	23	31	9895.0

Figure 1.7 Analysing values with engine-size=132

Thinking statistically, however, there are just two null values. It wouldn’t make much difference if we replaced these values with some summary statistics. Since the density plots for both these features are positively skewed we assumed that the values must fall near the median and so replaced them with the same.

```
#Replacing null values for 'horsepower' with median that spans the rows and extends column-wise
df['horsepower'].fillna(df['horsepower'].median(axis=0), inplace=True)
#Replacing null values for 'peak-rpm' with median that spans the rows and extends column-wise
df['peak-rpm'].fillna(df['peak-rpm'].median(axis=0), inplace=True)
```

Figure 1.8 Replacing null values for ‘horsepower’ with column median

Price – Price is a measure for the symboling of a particular vehicle. Keeping this in mind, we safely assumed that the automobiles falling in the same group of symboling fall in same price range.

```
df.loc[lambda df: df['price'].isnull()]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
9	0	134.0	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160.0	5500.0	16	22	NaN
44	1	134.0	isuzu	gas	std	two	sedan	fwd	front	94.5	...	90	2bbl	3.03	3.11	9.6	70.0	5400.0	38	43	NaN
45	0	95.0	isuzu	gas	std	four	sedan	fwd	front	94.5	...	90	2bbl	3.03	3.11	9.6	70.0	5400.0	38	43	NaN
129	1	134.0	porsche	gas	std	two	hatchback	rwd	front	98.4	...	203	mpfi	3.94	3.11	10.0	288.0	5750.0	17	28	NaN

Figure 1.9 Analysing null values for ‘price’

Given the fact that replacing the null values with a measure of central tendency for a subset of values is better and since the null values belong to ‘symboling’ = 0 and ‘symboling’ = -1, we try replacing the values with summary statistics. Analysing further, we realised there is a small portion of cars that have a huge price tag which is a cause for positive skewedness in the density plots of ‘price’ for both ‘symboling’=0 and ‘symboling’=1 and thus we replace the null values with respective median values.

1.2 Removing typos and redundant white spaces –

Once all null (NaN) were taken care of, we went ahead with taking care of typos for various ‘qualitative’ features and the respective redundant white space.

```
#Converting all object column values to lower case
df = df.apply(lambda x: x.str.lower() if(x.dtype == 'object') else x)
#Removing all the redundant white space from the object columns
df = df.apply(lambda x: x.str.strip() if(x.dtype == 'object') else x)
```

Figure 1.10 Converting the values with dtype=’object’ to lowercase and removing redundant white space

```
df['drive-wheels'].value_counts()
```

```
fwd    127
rwd    109
Name: drive-wheels, dtype: int64
```

```
df['make'].replace('vol00112ov', 'volvo', inplace=True)
```

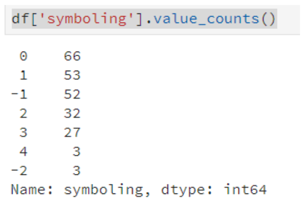
```
df['aspiration'].replace('turrrrbo', 'turbo', inplace=True)
```

```
df['drive-wheels'].replace('4wd', 'fwd', inplace=True)
```

Figure 1.11 Removing typos after column analysis for each object type feature

1.3 Handling Impossible Values –

Symboling – Symboling is the vehicle safety measure. It indicates the insurance risk rating. As per description, it is the degree to which the vehicle is more risky than its price indicates.



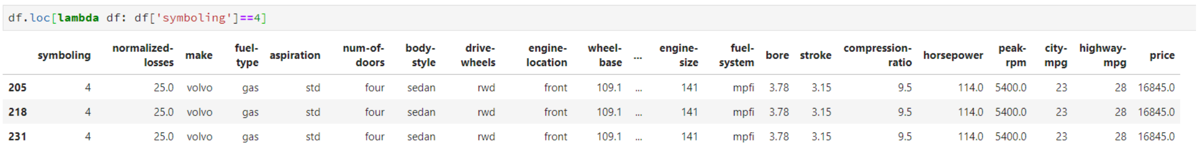
```
df['symboling'].value_counts()
```

```
 0     66
 1     53
-1     52
 2     32
 3     27
 4      3
-2      3
Name: symboling, dtype: int64
```

Figure 1.12 Analysing symboling values

We checked the values in the ‘symboling’ column using “df['symboling'].value_counts()”. We encountered an impossible value ‘4’. Cars are assigned a risk rating between (-3, -2, -1, 0, 1, 2, 3). 4 being an impossible value must be handled.

The following output shows that all the tuples with the symboling value ‘4’ have similar values for the rest of the features.

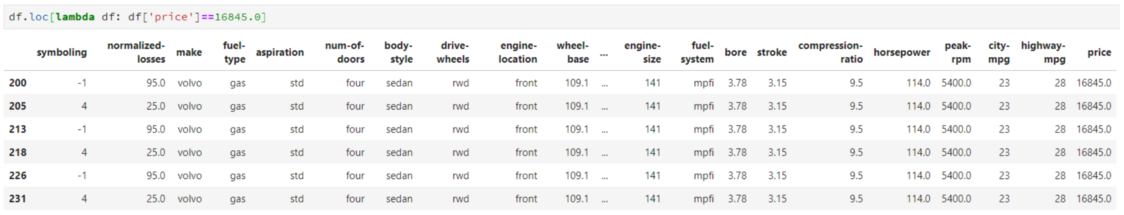


```
df.loc[lambda df: df['symboling']==4]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
205	4	25.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
218	4	25.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
231	4	25.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0

Figure 1.13 Analysing values with ‘symboling’=4

Also ‘symboling’, as per description of the dataset, is related to the price of the vehicle. Keeping this in view, we searched for records of the similar ‘make’ trying to find out if there were other records with similar features. We also searched for all the vehicles with price= 16845.0



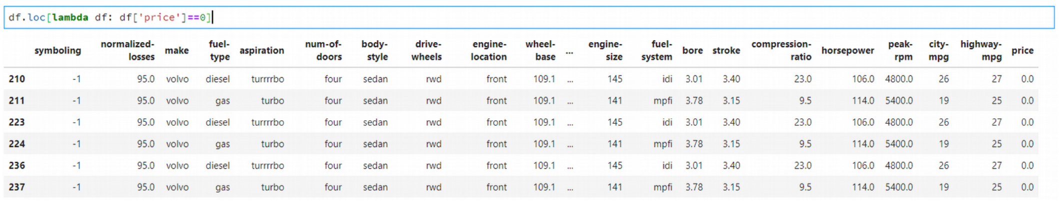
```
df.loc[lambda df: df['price']==16845.0]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
200	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
205	4	25.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
213	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
218	4	25.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
226	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0
231	4	25.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	23	28	16845.0

Figure 1.14 Analysing values with ‘price’=16845

In the above output every feature for all six records was similar except for ‘normalized-losses’ which was an impossible value given the description of the columns in dataset. Since all other features were similar and the vehicle had the same ‘make’, we replaced the symboling to ‘-1’ and ‘normalized-losses’ to ‘95.0’ for each record with price = ‘16845.0’. We also came across vehicles with similar ‘make’ having the same feature values.

Price – There were vehicles in dataset with ‘price’=0. All these values had the same ‘symboling’=-1 .Trying to analyse all vehicles with price = 0, we came across similar values grouped together according to the ‘fuel-type’.



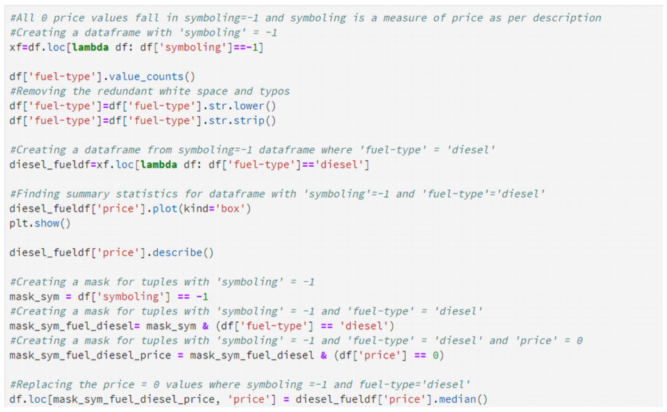
```
df.loc[lambda df: df['price']==0]
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	highway-mpg	price
210	-1	95.0	volvo	diesel	turrrrbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106.0	4800.0	26	27	0.0
211	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	19	25	0.0
223	-1	95.0	volvo	diesel	turrrrbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106.0	4800.0	26	27	0.0
224	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	19	25	0.0
236	-1	95.0	volvo	diesel	turrrrbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106.0	4800.0	26	27	0.0
237	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114.0	5400.0	19	25	0.0

Figure 1.15 Analysing tuples with ‘price’=0

Building upon this analysis we went one step further to analyse the prices of all vehicles with ‘symboling’=-1 and ‘fuel-type’ = ‘diesel’ and replaced the ‘price’ = 0 values for diesel vehicles having symboling = -1 with their respective medians.

Potential issues/errors –
1. Upper case letters in ‘fuel-type’
2. Adjoining whitespace with the values.



```
#All 0 price values fall in symboling=-1 and symboling is a measure of price as per description
#Creating a dataframe with 'symboling' = -1
xf=df.loc[lambda df: df['symboling']==-1]

df['fuel-type'].value_counts()
#Removing the redundant white space and typos
df['fuel-type']=df['fuel-type'].str.lower()
df['fuel-type']=df['fuel-type'].str.strip()

#Creating a dataframe from symboling=-1 dataframe where 'fuel-type' = 'diesel'
diesel_fueldf=xf.loc[lambda df: df['fuel-type']=='diesel']

#Finding summary statistics for dataframe with 'symboling'=-1 and 'fuel-type'='diesel'
diesel_fueldf['price'].plot(kind='box')
plt.show()

diesel_fueldf['price'].describe()

#Creating a mask for tuples with 'symboling' = -1
mask_sym = df['symboling'] == -1
#Creating a mask for tuples with 'symboling' = -1 and 'fuel-type' = 'diesel'
mask_sym_fuel_diesel= mask_sym & (df['fuel-type'] == 'diesel')
#Creating a mask for tuples with 'symboling' = -1 and 'fuel-type' = 'diesel' and 'price' = 0
mask_sym_fuel_diesel_price = mask_sym_fuel_diesel & (df['price'] == 0)

#Replacing the price = 0 values where symboling =-1 and fuel-type='diesel'
df.loc[mask_sym_fuel_diesel_price, 'price'] = diesel_fueldf['price'].median()
```

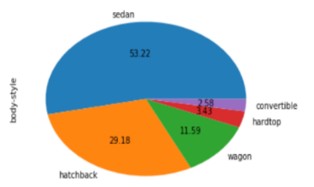
Figure 1.16 Sample code for replacing the ‘price’=0 with the median grouped by ‘fuel-type’ = ‘diesel

2. Data Exploration

2.1

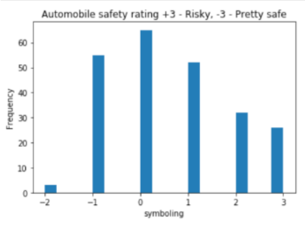
```
df['body-style'].value_counts().plot(kind='pie', autopct='%.2f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xc78e518>
```



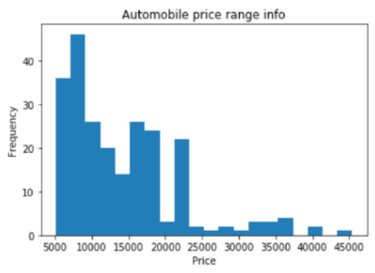
Body-Style (Nominal) – The body-style of the vehicle can provide insights about the preferred style of the automobile. Knowing the preferred body-style, a customer should be able to make better decision about which type of vehicle he should consider to buy.

```
df['symboling'].plot(kind='hist',bins=20)
plt.title('Automobile safety rating +3 - Risky, -3 - Pretty safe')
plt.xlabel('symboling')
plt.show()
```



Symboling (Ordinal) – The safety is of uttermost importance today to every automobile owner. 'symboling', the ordinal feature present in the dataset is the insurance risk rating of various vehicles. It is important to provide this knowledge to a customer. The graph depicts the frequency of the number of cars with a particular risk rating. Most of the cars shown have symboling=0. While there are some cars that are pretty safe with a rating of -2. This should ensure a customer that a car with 'symboling' 0 or more should be a trustworthy vehicle.

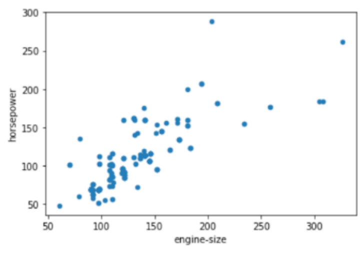
```
df['price'].plot(kind='hist',bins=20)
plt.title('Automobile price range info')
plt.xlabel('Price')
plt.show()
```



'Price' (Numerical) – Price is what everyone considers before buying a vehicle. Knowing the price of the majority of vehicles should provide the customer with better understanding about how much to spend for a vehicle.

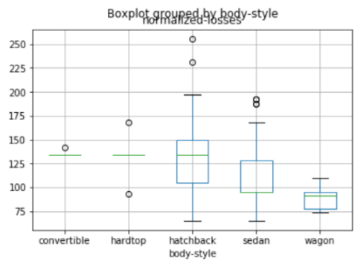
2.2

Hypothesis 1. 'Horsepower' should increase with increase in the 'engine-size'



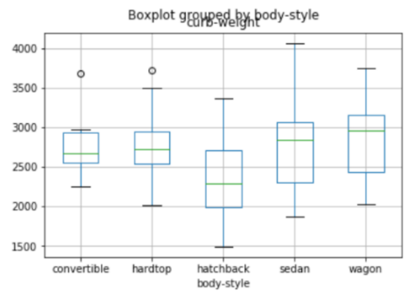
Horsepower of an engine is an important factor while buying a vehicle for competitive sports or certain household purpose. As a hypothesis we analyse whether the size of engine can be a measure of the horsepower. The adjacent figure is a scatter plot between 'engine-size' and 'horsepower' of automobiles. We chose a scatter plot because it depicts the trend of values better and because a customer should be able to understand the significance of horsepower with respect to engine-size without any undue effort. The hypothesis is correct since the horsepower is increasing with the increase in the size of an engine.

Hypothesis 2. 'Normalized – losses' should be influenced by the style of the vehicle.



'Normalized-losses' indicates the average loss of value of a vehicle per insured vehicle year. It would be interesting to explore the 'normalized-losses' based on the style of the vehicle. This will provide a customer with the insights about how much value their vehicle is going to lose over the years if they buy it. A boxplot provides better summary statistics and hence was ideal to represent the distribution of values. The hypothesis was correct since the loss in value for station wagons and sedans was comparatively less as compared to other types of vehicles.

Hypothesis 3. 'Curb-weight' of vehicles with bigger 'body-styles' should be more as compared to relatively smaller vehicles.



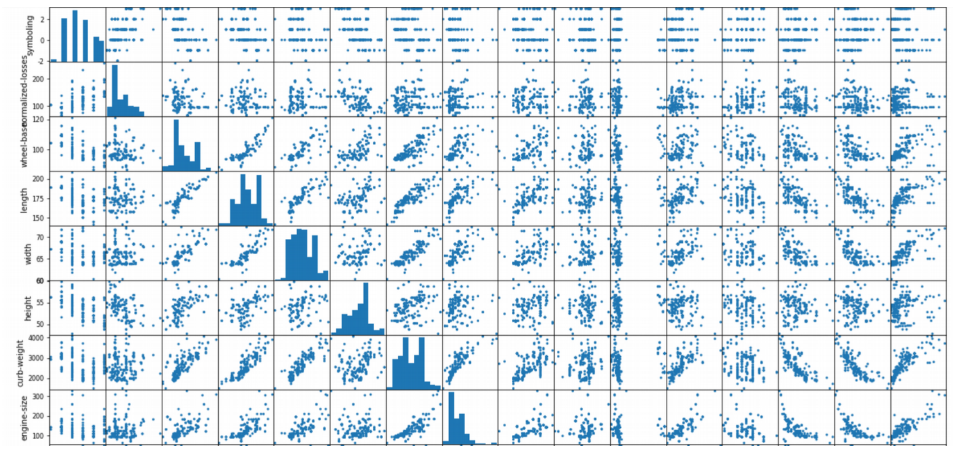
The curb-weight is the weight of a fully-loaded automobile. It includes all the necessary equipment supplied with the vehicle as well as the fuel and the additional fluids required by the vehicle. Curb-weight therefore depicts the weight of a fully-operational vehicle. Having insights about the curb-weight are important since it helps to determine whether or not a vehicle is safe to use on a certain bridge or road with a specified weight limit. A box-plot provides better summary statistics and measures of central tendency for vehicles based on their 'body-style'. The graph clearly depicts that wagon's of all vehicles have the greatest max curb-weight and median curb-weight. Hence wagons are the heaviest. It also makes it visible that a hatchback with its smallest size/style amongst the listed vehicles had the smallest value for curb-weight and that most hatchbacks are light as compared to other vehicles.

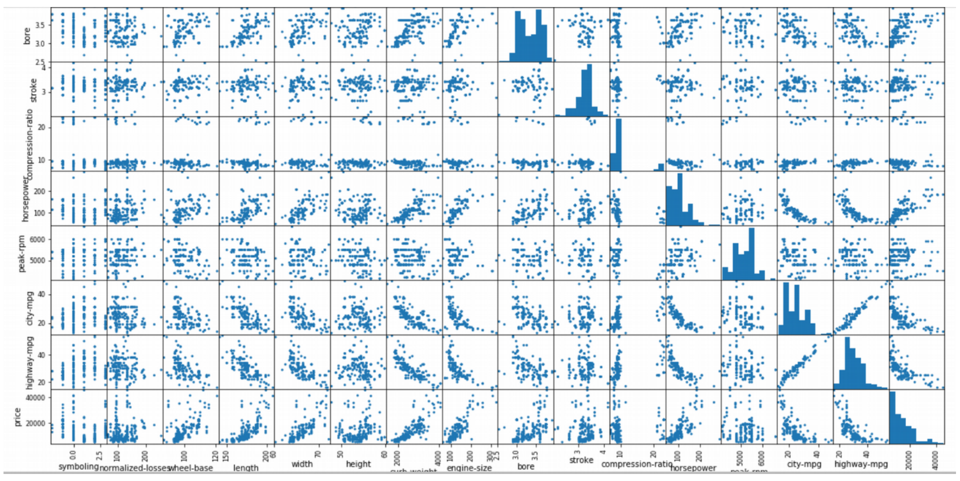
2.3 Scatter Matrix for all the numerical columns -

The scatter matrix for all the numerical columns depicts the relationship between all the quantitative features of an automobile.

Observations – 1. Vehicles with smaller engine-size have higher 'city-mpg' and 'highway-mpg'.

2. 'Curb-weight' of a vehicle increases with an increase in 'wheel-base', 'length', 'width', 'height' and 'engine-size'.
3. The 'price' of vehicles increases with the increase in 'wheel-base', 'length', 'width', 'height', 'engine-size' and 'horsepower'.





3. **References –**

https://www.physicsforums.com/threads/increase-rpm-by-decreasing-stroke.706297/
https://www.tripsavvy.com/what-is-curb-weight-4117850