# **Assignment 1**
## **Data Preparation**

Data retrieving

- Load the pandas library in to python script and call it "pd" by using this syntax : `import pandas as pd`
  "import" tell python to bring the library in, "as" tells python that pandas will be called "pd" from now on.
- Load the dataset using the `pandas.read_csv` function
  - Assign the filename of the dataset "Automobile.csv" in the first argumant
  - Assign the "`sep`" argument with "#" because the default is "," but the separator in this file is "#"
  - Assign the "`dec`" argument with "," because the default is "." but the decinmal placeholder in this file is ",".
  - Assign the "`names`" argument with the column names provided in the source information.
- Using the `head()` function to check the first 5 rows of the dataset to see if the DataFrame that was downloaded makes sense in terms of the values that are matched with each column.
- Check the dimension of the DataFrame using the `shape` attribute to make sure that the whole dataset has been captured. In this case, there were 238 rows of data which is more than the source information and 26 columns which matches the source information.

Check data types

- Check the type of all the variables using the `dtypes` attribute. The result shows that
  - The 'Symboling' was an "int64" type, which was not the right type. The 'Symboling' is a rating scale of insurance risk which should be an "object" type.
  - 'Wheel-base', 'Length', 'Width', 'Height','Bore', 'Stroke' and 'Compression-ratio' were "object" types which was the wrong type of data.
- Convert the data into the appropriate type by using the `astype()` function.
  - Use `{col: dtype, …}` where the "col" is a column name and "dtype" is the type to convert to.

Typo, Extra-whitespaces and upper/lower case and impossible value

- Check the unique values of the particular column and count them by using the `value_counts()` function and sort the result by the index column to make it easier to check the values by using the `sort_index()` function. (The unique value can be checked with the `unique()` function as well but unique() function will not return the count for each unique value)
  - Typos are very easy to spot. For example "vol00112ov", it is clearly a typo for "volvo".
  - For the whitespace, either it is obviously seen (there is a space at the start of the value shown) or you will see more than 1 as exactly the same value in the `value_counts()` function's result. For example, there are 2 "volvo" values in the "Make" column after checking with the `unique value()` function.
  - The upper/lower case are very easy to spot. For example, in the result of the `value_counts()` function for the column "Fuel-type", there is "Diesel" and "diesel".
  - For the impossible values, compare the unique values with the column information from the source. For example, the column "Symboling" according to the information source, "Symboling" column should have the value from -3 to 3 but there is "4" in the column as well. The "4" is an impossible value that needs to be dropped as it has no meaning in the dataset.
- Remove the whitespaces by using the `str()`function to convert the value into a string (Refsnes Data, 2021) and then use the `strip()` function to remove the space at the start and the end of the string.
- Replace the typos with the correct spelling and fix the upper/lower case problem by replacing every upper case with the lower case letter by using the `replace()` function. Use `{col: { old:new,…}` where "col" is the column name, "old" is the string that needs to be replaced and "new" is the string to replace into. Assign the argument "inplace" to "True" to put the result in place.
- Drop the impossible values by using the `loc` attribute to select the specific column with condition. For example, the column "Symboling" using the `loc` attribute to select every record except the records that have the value 4 in the "Symboling" column. Assign the condition of the selection with `"!= 4"`which means NOT equal 4]

Missing values

- Check for the missing value(NaN) by using the `isnull()` function. Use the `values` attribute and the `any()` function to return "True" if there is any NaN value in the dataset. In this case, the result is "True".
  - Check the number of the NaN value in each column by using the `isnull()` function followed by the `sum()` function. The result shown that the "Normalised-losses" has got 43 NaN values, "Num-of-doors" "Horsepower" and "Peak-rpm" column have got 2 NaN values, "Bore" "Stroke" and "Price" column have got 4 NaN values.
  - Form plot number 6 in this report, I have found that the **"Horsepower"** is related to the "City-mpg". The plot shows that the more "Horsepower" value, the less "City-mpg" value. As a result, I inspect the NaN values of the "horsepower" by subseting the rows that have the NaN value of "Horsepower" into object "Horsepowernull" to

have a closer look. I have found that 2 of them have got the same "City-mpg" value (Their City-mpg are 23). Then I created DataFrame called "City23" which store all the rows that have the "City-mpg" value equal to 23. Then replace the NaN values of the two rows with the mean of the "Horsepower" value of all the record in the "City23" by using the `fillna()` and `mean(axis=0)` function. Instead of replacing them with the mean of the whole column, replace them with the mean of the Horsepower column in "City23" which will give a more accurate value.

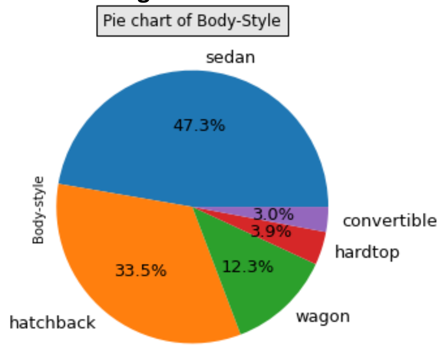
- For the **"Num-of-doors"** column, after I inspect the records with NaN value, I found that both of them have got the same "Body-style" which is sedan. As a result, I subset all the records with a sedan body style and use the `.value_counts()` to find out the ratio between the sedan car with two doors and the sedan car with four doors. The result shown that 85.11% of sedan cars have got four doors which mean potentially these two missing values are "four". I replace the NaN values with a fixed value which is "four" by using the `fillna()` function and assign the first document with "four".
- For the **"Price"** column, I choose to drop both rows with NaN value because the price is the important key of the dataset so it should not be replaced by an estimate value. To drop the rows I use the `dropna()` function and assign the argument "subset" with "Price".
- For the rest of the NaN value, replace them with the mean of its own column using the `mean()` function.

Sanity check

- Check for the whitespaces, typos, upper/lower case problem and the impossible values.
  - After removing the whitespaces, correcting the typo and correcting the upper/lower case problem. Check the unique values of each column by using the `value_counts()` function again. For example, the "Aspiration" column had 2 "Std" values, 172 "std" values, 1 "std " value, 57 "turbo" values and 3 "turrrrbo" values before the correction but after the correction it has 175 std values and 60 turbo values.
- Check for the duplicate records by using the `duplicated()` function and the `any()` function to return the result "True" if there is any duplicate value. In this case, the result is "True" which mean there is some duplicate record in the dataset
  - Drop duplicates except for the first occurrence by using the `drop_duplicates()` function and assign "inplace" argument with "True" so the duplicates get removed from the dataset. After that use the `duplicate.any()` function to make sure there are no duplicates left.
- Check for the NaN value again after dealing with them by using the `isnull.any()` function. The last result is "False" which mean there is no NaN value in the dataset anymore.

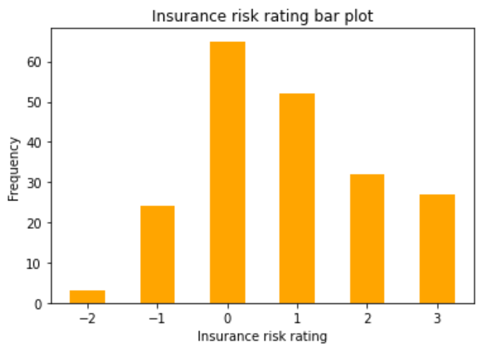
## Data Exploration

### 1. Choosing 3 columns



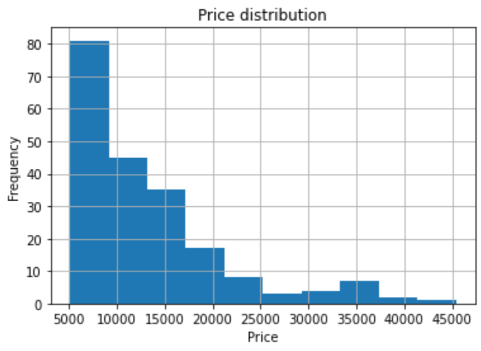
Plot No.1 - Pie chart of Body-style

- I chose a pie chart to represent the Body style data because the "Body-style" is a nominal variable and the pie chart is good at showing 'the relationships of parts to the whole for a variable' (SAS Institute Inc.,2021).
- This pie chart shows that the sedan is the most common "Body-style" of the cars at 47.3% followed by the hatch back at 33.5%. The convertible is the least popular body-style of the cars at 3%.



Plot No.2 - Bar plot of Symbolling

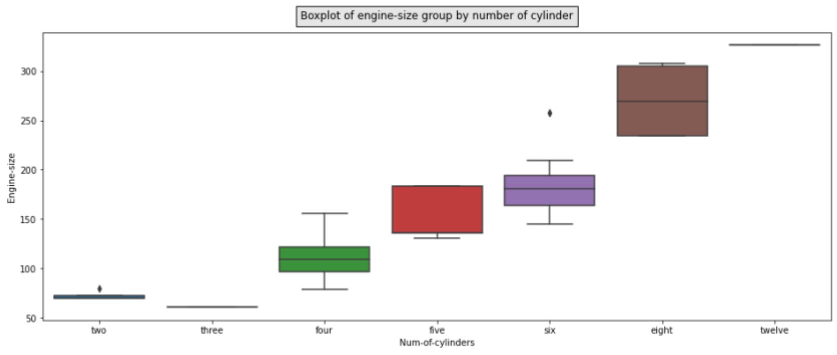
- I choose bar plot to represent the Symbolling column because a bar plot is good for ordinal variable and it is easy to read as well.
- The plot shows that the most common insurance risk rating is 0 which is in the middle of the scale and the less common insurance risk rating is -2. From the plot, it is obvious that the frequencies of the high risk ratings are significantly much more than the low risk rating.
-



Plot No.3 - The histogram of Price

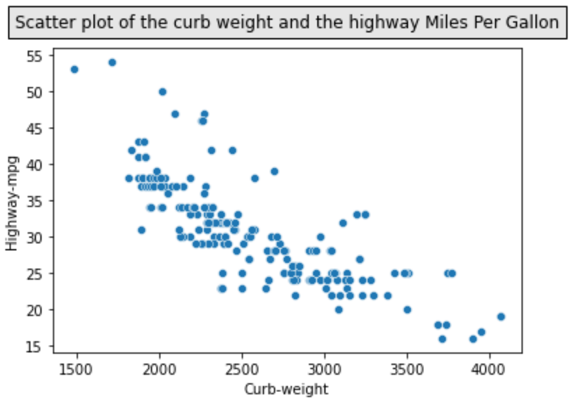
- I choose the histogram plot to represent the "Price" variable because histogram is suitable for numerical variable (continuous variable) and it can show the distribution of the data as well.
- From the histogram plot shown that the distribution of the "Price" column is right skewed. This means that the price of the car in the low range (cheap) is more common than the high range(expensive).The most common price of car is in the range between 5,000 to a bit under 10,000 and there is very low frequency for the price over 25,000.

**2. Relationships between columns**



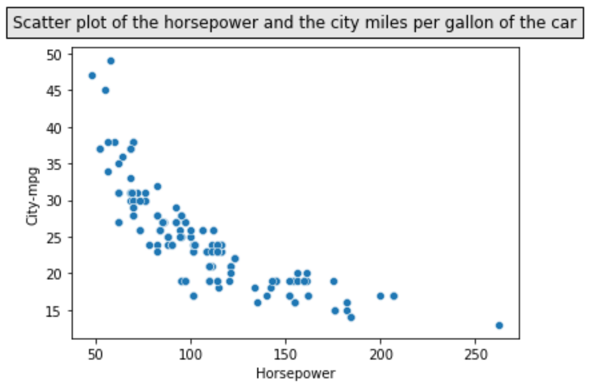
Plot No.4 - The boxplot of the "Engine-size" group by the "Num-of-cylinder"

- I choose the boxplot to represent the relationship between the engine-size and the number of cylinder because boxplot is good for continuous variable values for each category of categorical variable. (Wayne W., 2016)
- My hypothesis is the more cylinder the car has the bigger the engine-size will be.
- The plot shows that the three cylinders car has got the smallest engine-size and the twelve cylinders car has got the biggest engine-size. The boxplot shown that the more number of Cylinder the car have, the bigger the Engine-size of the car tend to be. Although, the two cylinders car that has got bigger engine-size than the three cylinders car. Nonetheless, in this data set, there is just 1 car that have got three cylinders which might not be a good representative of all the three cylinders car.



<u>Plot No.5 - The scatter plot of the“Highway-mpg” and the “Curb-weight”.</u>

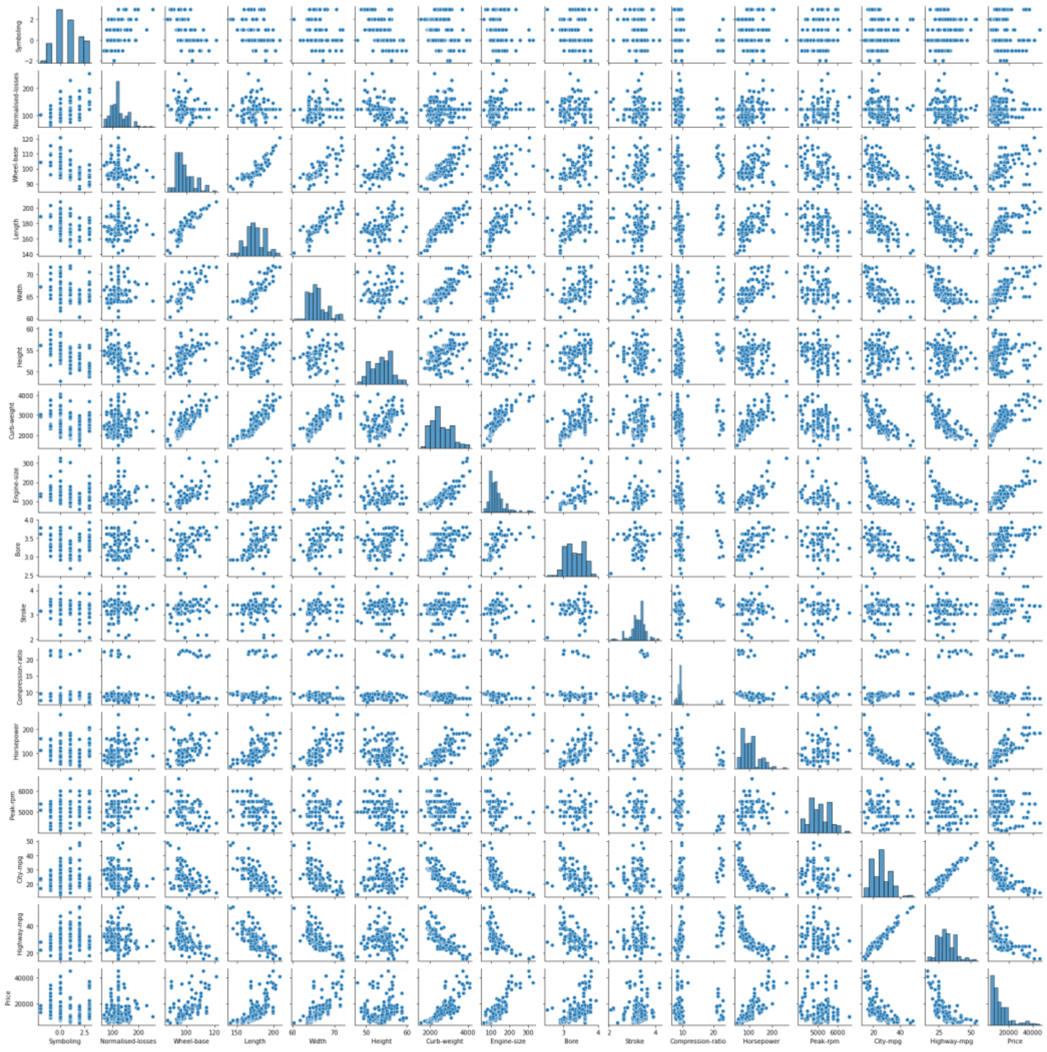
- I choose the scatter plot to represent the relationship between these two variables because the scatter plot is a good plot to compare two numeric variables.
- My hypothesis is a heavier car is consuming the fuel more than a lighter car.
- The plot shown the negative correlation between the two variables which mean the heaviler the car are the more they are consuming the fuel on the highway drive.



<u>Plot No.6 The scatter plot of “Horsepower” and “City-mpg”.</u>

- I choose the scatter plot to represent the relationship between these two variables because the scatter plot is a good plot to compare two numeric variables.
- My hypothesis is the more horsepower the car have got the more fuel they are consuming.
- The plot shown the negative correlation between the two variables which mean the more horsepower the car have the less miles per gallon of fuel the car can go or to make it easier is more horsepower the car have the more fuel they are consuming on the city drive.

**3. Scatter matrix**



<u>Plot No.7</u>

The scatter matrix of numeric variables

- The scatter matrix shown ‘bi-variate or pairwise relationship between different combinations of variables’. Diagonally from top left to right, the plots represent univariate distribution of data for the variable in that column’ (Ajitesh Kumar,2020).
- I choose to use seaborn library to plot the scatter matrix as it looks better and for me it is easier to see the shape of the plot.
- In the scatter matrix shown,
    - The strong positive linear like the scatter plot between “City-mpg” and “Highway-mpg” means the two variables have got a positive strong relationship. In the plot, The more higher value of the “City-mpg” the higher value of “Highway-mpg”. In this case, the strong linear of the relationship make a lot of sense as they both are reflection of fuel consumption and the same car should not be consuming lot different between the highway and the city drive. The example of the plots that have the strong positive linear other than the “City-mpg” and “Highway-mpg” are the plot between “Length” and “Curb-weight”, “Length” and “Wheel-base”.
    - The strong negative linear like the scatter plot between “City-mpg” and “Horsepower” and the plot between “Highway-mpg” and “Horsepower” means the two have hot a negative strong relationship. From the plot, The more horsepower of the car, the less miles per gallon of fuel the car can go. In this case, the correlation imply causation as the more horsepower mean the more fuel the car is consuming.
    - The moderate positive linear like the scatter plot between “Height” and “Length” or “Engine-size” and “Price” means that they might have some reasonable relationship but it might have some other condition in play to effect the correlation. For example, the bigger engine-size size tend to be more expensive but the brand of the car is also in play as well. Some of the car might have a bigger engine size but the price might be cheaper because of the brand price range might be lower.
    - The moderate negative like the scatter plot between “City-mpg” and “Length” or “City-mpg” and “Curb-weight” shown the opposite but same concept as the moderate positive linear above. It is make sense that the longer the car are the less miles per gallon of fuel the car can go or to make it easier to understand the longer the car are the more they are consuming the fuel. Although, there are many more factor in play to make the car consume more or less fuel such as the horsepower that I mentioned before or the “Curb-weight” that is going to effect the consumption rate as well.

- For the right skewed, histogram plot of “Price” and “Engine-size” are good examples. These right skewed histogram mean the most common car have got a low rang price and engine-size.

## Reference:

The pandas development team, 2021, *pandas.read_csv*. pandas development team, Version 1.3.4. viewed 9 November 2020. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html>.

Refsnes Data, 2021, *Python str() Function*, W3Schools, viewed 9 November 2021, <https://www.w3schools.com/python/ref_func_str.asp>.

James G., 2020, *Python String Strip: How To Use Python Strip*, Career Karma, viewed 9 November 2021, <https://careerkarma.com/blog/python-string-strip/>.

The pandas development team, 2021, *pandas.DataFrame.drop_duplicates* . pandas development team, Version 1.3.4. viewed 9 November 2021, <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html>.

CFI Education Inc.,2020, *Histogram*, CFI Education Inc., viewed 11 November 2021, <https://corporatefinanceinstitute.com/resources/excel/study/histogram/>.

Alison Smith, 2021, ENGINE SIZE CHART, CJ Pony Parts, viewed 11 November 2021, <https://www.cjponyparts.com/resources/engine-size-chart>.

SAS Institute Inc., *Pie Chart*, *SAS* Institute Inc., viewed 11 November 2021,<https://www.jmp.com/en_us/statistics-knowledge-portal/exploratory-data-analysis/pie-chart.html>.

Wayne W.,2016, *Two Variables*, Boston University School of Public Health, viewed 11 November 2021, <https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/datapresentation/DataPresentation7.html>.

Michael Waskom, 2021, *seaborn.pairplot*, seaborn.pydata.org, viewed 12 November 2021, <https://seaborn.pydata.org/generated/seaborn.pairplot.html>.

Ajitesh Kumar, 2020, *What, When, and How of Scatterplot Matrix in Python - Data Analytics*, Answer hub, viewed 12 November 2021, <https://dzone.com/articles/what-when-amp-how-of-scatterplot-matrix-in-python>.

Mike Yi, *What is a scatter plot?*, Chartio, viewed 12 November 2021,<https://chartio.com/learn/charts/what-is-a-scatter-plot/>.

Yan Holtz*, Controlling the order of distributions in a boxplot with Seaborn,* Yan Holtz, viewed 12 November 2021, <https://www.python-graph-gallery.com/35-control-order-of-boxplot>.