

# **EXPLORING AUTOMOBILE DATA USING PYTHON**

**Project Report Submitted to National Skill Training Institute for Women,  
Trivandrum in the Partial Fulfillment and Requirements for IBM Advanced  
Diploma(Vocational) in IT, Networking and Cloud**

**By**

**Alfiya Fazil**

**ADIT/TVM/19/001**

**ADIT (2019-2021)**

**Under the supervision of**

**Mr. Poovaragavan Velumani (Master Trainer, Edunet Foundation)**



**GOVERNMENT OF INDIA**

**MINISTRY OF SKILL DEVELOPMENT & ENTREPRENEURSHIP**

**DIRECTORATE GENERAL OF TRAINING**

**July 2021**

# **CONTENTS**

## **ABSTRACT**

### **1. INTRODUCTION**

1.1. THE PROBLEM

1.2. UNDERSTANDING THE DATA

1.3. SOFTWARES AND LIBRARIES USED

### **2. DATA EXPLORATION**

2.1. DATA ACQUISITION

2.2. BASIC INSIGHTS FROM DATA

### **3. DATA WRANGLING**

3.1. DEALING WITH MISSING DATA

3.2. DATA FORMATTING

3.3. DATA NORMALIZATION

3.4. DATA BINNING

### **4. EXPLORATORY DATA ANALYSIS**

4.1. DESCRIPTIVE STATISTICS

4.2. BOXPLOT

4.3. SCATTERPLOT

4.4. GROUPBY

4.5. ANALYSIS OF VARIANCE

4.6. CORRELATION

4.7. PEARSON CORRELATION

**5. CONCLUSION**

**6. REFERENCE**

## **ABSTRACT**

The project titled 'EXPLORING AUTOMOBILE DATA' is an attempt at analyzing and visualizing the different aspects of a *Used Car Prices Dataset* obtained from *UCI Machine Learning Repository*. The insights gained from the dataset by applying various data analysis techniques can be further utilized in the future for predicting the most probable price of a car.

# 1. INTRODUCTION

## 1.1 THE PROBLEM

The value of a car drops right from the moment it is bought and the depreciation continues with each passing year. In fact, in the first year itself, the value of a car decreases by 20 percent of its initial value. The make and body-style of a car, engine size, overall condition of the vehicle and various other factors further affect the car's resale value. Hence, the purpose of this exploration of automobile data is to discover useful information from the data and answer the question *'What determines the price of used cars?'*.

## 1.2 UNDERSTANDING THE DATA

The dataset used in this project is taken from UCI Machine Learning Repository, a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms. The Automobile Dataset is in CSV (comma separated value) format with 205 rows and 26 columns. The header starts with the first attribute "symboling", corresponding to the insurance risk level of a car.

Cars are initially assigned a risk factor symbol associated with their price. Then, if an automobile is riskier, this symbol is adjusted by moving it up the scale. A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe. The second attribute "normalized-losses" is the relative average loss payment per insured vehicle year and the values range from 65 to 256. The other attributes are easy to understand. The 26th attribute is "price". This is our target value. This means "price" is the value that we are going to predict from the data-set, and the predictors should be all the other variables listed, like "symboling", "normalized-losses", "make" and so on.

### 1.3 SOFTWARES AND LIBRARIES USED

*Anaconda* is the data science platform used in this project for exploring and visualizing data. The reason why it's chosen is that *Anaconda* conveniently installs Jupyter Notebook, Python and other commonly used packages for data analytics. The *Jupyter Notebook* is an open source web application that can be used to create and share documents that contain live code, equations, visualizations, and text. *Python* is a free and open-source high-level programming language that has hundreds of different libraries and frameworks focused on Data Analytics and Machine Learning.

Python libraries, Numpy, Scipy, Pandas, Matplotlib and Seaborn are used here to carry out different data exploration tasks. *NumPy*, *Scipy* and *Pandas* are great for exploring and playing with data. *Matplotlib* is a data visualization library that makes graphs like you'd find in Excel or Google Sheets and *Seaborn* is also a visualization library based on matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.

# 1. DATA EXPLORATION

## 2.1 DATA ACQUISITION

The first step towards acquiring data is to collect the needed dataset from the [source](#). Once the dataset is downloaded and saved in the project directory, the dataset will be ready for getting probed using python libraries in the Jupyter Notebook.

```
In [1]: import pandas as pd

filename = "automobile.data"
df = pd.read_csv(filename)
print("The first 10 rows of the dataframe")
df.head(10)
```

The first 10 rows of the dataframe

```
Out[1]:
```

	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
4	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	15250
5	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	17710
6	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	18920
7	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3	140	5500	17	20	23875
8	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160	5500	16	22	?
9	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.8	101	5800	23	29	16430

10 rows x 26 columns

Using python pandas library the dataset values can be examined and the column names can be updated.

```
In [5]: headers = ["symboling", "normalized-losses", "make", "fuel-type", "aspiration", "num-of-doors", "body-style",
                  "drive-wheels", "engine-location", "wheel-base", "length", "width", "height", "curb-weight", "engine-type",
                  "num-of-cylinders", "engine-size", "fuel-system", "bore", "stroke", "compression-ratio", "horsepower",
                  "peak-rpm", "city-mpg", "highway-mpg", "price"]
df.columns = headers
df.head(10)
```

```
Out[5]:
```

	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
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115
4	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110
5	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110
6	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110
7	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3	140
8	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160
9	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.8	101

10 rows x 26 columns

## 2.2 BASIC INSIGHTS FROM DATA

Before conducting any analysis, it is necessary to have a thorough understanding of the data. There are many different forms of data. An object, float, int, and DateTime are the most common types held in Pandas objects. It's important to know what data types are used and how data is distributed.

To check the data type `df.dtypes` can be used as given below.

```
In [12]: df.dtypes
```

```
Out[12]: symboling          int64
normalized-losses  object
make              object
fuel-type         object
aspiration        object
num-of-doors      object
body-style        object
drive-wheels      object
engine-location   object
wheel-base       float64
length           float64
width            float64
height           float64
curb-weight       int64
engine-type       object
num-of-cylinders  object
engine-size       int64
fuel-system       object
bore             object
stroke           object
compression-ratio float64
horsepower        object
peak-rpm          object
city-mpg          int64
highway-mpg       int64
price            object
dtype: object
```

To understand the distribution of data in each column, we must examine the statistical summary of each column. The method `df.describe()` can be utilized to acquire the quick statistics.



```
In [8]: df.describe()
```

```
Out[8]:
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

It returns the number of terms in the column as “count”, average column value as “mean”, column standard deviation as “std”, the maximum and minimum values, as well as the boundary of each of the quartiles. By default, the `dataframe.describe()` function skips rows and columns that do not contain numbers. The `describe()` method returns a summary for numbers, to get a summary of all the columns with the object data-type we could add an argument “include = ‘all’” inside the describe function bracket.

```
In [14]: df.describe(include = "all")
```

```
Out[14]:
```

	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
count	204.000000	204	204	204	204	204	204	204	204	204.000000	...	204.000000	204	204	204	204.000000	204
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37	NaN	NaN
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN	NaN
freq	NaN	40	32	184	167	114	96	120	201	NaN	...	NaN	93	23	20	NaN	NaN
mean	0.823529	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.806373	...	126.892157	NaN	NaN	NaN	10.148137	NaN
std	1.239035	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.994144	...	41.744569	NaN	NaN	NaN	3.981000	NaN
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN	7.000000	NaN
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000	NaN	NaN	NaN	8.575000	NaN
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	119.500000	NaN	NaN	NaN	9.000000	NaN
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	142.000000	NaN	NaN	NaN	9.400000	NaN
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN	23.000000	NaN

11 rows x 26 columns

“Unique” is the number of distinct objects in the column, “top” is the most frequently occurring object, and “freq” is the number of times the top object appears in the column. Some values in the table are shown here as “NaN”, which stands for “not a number”. This is because that particular statistical metric cannot be calculated for that specific column data type.

## 2. DATA WRANGLING

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis. This process typically includes manually converting and mapping data from one raw form into another format to allow for more convenient consumption and organization of the data.

### 3.1 DEALING WITH MISSING DATA

A feature is said to have a "missing value" when no data is available for it. A missing value in a dataset is usually displayed as "?", "NaN," 0, or a blank cell. Before dealing with missing values, it's important to remember that each situation is different and should be assessed accordingly. The first step is to check whether the person or group who gathered the data can go back and figure out what the true value is. Another alternative is to remove the data containing the missing value. When dropping data, either the entire variable can be dropped or only the single data entry with the missing value.

In the car dataset, missing data comes with the question mark "?". It is replaced with NaN (Not a Number) using `df.replace('?', np.nan)` after importing the numpy python library.

```
df.head(10)
```

The first 10 rows of the dataframe

Out[1]:

	symboling	normalized-losses	make	fuel-type	aspi
0	3	?	alfa-romero	gas	
1	1	?	alfa-romero	gas	
2	2	164	audi	gas	
3	2	164	audi	gas	

```
In [2]: import numpy as np
df.replace("?", np.nan, inplace = True)
df.head(5)
```

Out[2]:

	symboling	normalized-losses	make	fuel-type	aspiration
0	3	NaN	alfa-romero	gas	std
1	1	NaN	alfa-romero	gas	std
2	2	164	audi	gas	std
3	2	164	audi	gas	std

To remove data that contains missing values, pandas library has a built-in method called "dropna". There are some missing values in "Price". Since the price of used cars is what we're

trying to predict in our upcoming analysis, we have to remove the cars(the rows) that don't have a listed price.

```
In [10]: df[['price']].head(11)
```

```
Out[10]:
```

	price
0	16500
1	16500
2	13950
3	17450
4	15250
5	17710
6	18920
7	23875
8	NaN
9	16430
10	16925



```
In [11]: df.dropna(subset=["price"],axis=0,inplace=True)
```

```
In [13]: df[['price']].head(10)
```

```
Out[13]:
```

	price
0	16500
1	16500
2	13950
3	17450
4	15250
5	17710
6	18920
7	23875
9	16430
10	16925



In numerical data, missing values can be replaced with the “average” value, while missing values in categorical data can be handled using the “mode” (often occurring) value. Because the column "bore" has some missing values, it can be replaced by the average of the entries in that column.

```
In [39]: df[['bore']].loc[53:61]
```

```
Out[39]:
```

	bore
53	3.08
54	NaN
55	NaN
56	NaN
57	NaN
58	3.39
59	3.39
60	3.39
61	3.39

```
In [40]: mean = df[['bore']].astype("float").mean(axis=0)  
mean
```

```
Out[40]: bore    3.33  
dtype: float64
```

```
In [41]: df[['bore']] = df[['bore']].replace(np.nan,mean)
```

```
In [42]: df[['bore']].loc[53:61]
```

```
In [42]: df[['bore']].loc[53:61]
```

```
Out[42]:
```

	bore
53	3.08
54	3.33
55	3.33
56	3.33
57	3.33
58	3.39
59	3.39
60	3.39
61	3.39



## 3.2 DATA FORMATTING

Data formatting is the process of transforming data into a common format, which helps users to perform comparisons.

Although the price feature's data type is "object," the desired data type should be an integer or float type. The “astype” method can be used to change data types.

```
In [61]: df[['price']].dtypes
```

```
Out[61]: price    object
         dtype: object
```


```
In [62]: df[["price"]] = df[["price"]].astype("float")
```

```
In [63]: df[['price']].dtypes
```

```
Out[63]: price    float64
         dtype: object
```

In the data-set, there are features named “city-mpg” and “highway-mpg”, which refers to a car's fuel usage in miles per gallon units. Since, mpg units are not used very often, we'll convert the values to L/100km, the metric equivalent.  $L / 100km = 235.21 / x \text{ mpg}$

```
In [84]: df["city-mpg"] = 235 / df["city-mpg"]
         df.rename(columns={'city-mpg': 'city-L/100km'}, inplace=True)
         df["highway-mpg"] = 235 / df["highway-mpg"]
         df.rename(columns={'highway-mpg': 'highway-L/100km'}, inplace=True)
         df.head()
```

city-mpg	highway-mpg		city-L/100km	highway-L/100km
21	27		11.190476	8.703704
19	26		12.368421	9.038462
24	30		9.791667	7.833333
18	22		13.055556	10.681818
19	25		12.368421	9.400000

### 3.3 DATA NORMALIZATION

Normalization is the process of transforming values of several variables into a similar range. Typical normalizations include scaling the variable so that the values range from 0 to 1.

```
In [14]: df["length"] = df["length"]/df["length"].max()  
df["width"] = df["width"]/df["width"].max()  
df['height'] = df['height']/df['height'].max()  
df[["length","width","height"]].head()
```

Out[14]:

	length	width	height
0	0.811148	0.890278	0.816054
1	0.822681	0.909722	0.876254
2	0.848630	0.919444	0.908027
3	0.848630	0.922222	0.908027
4	0.851994	0.920833	0.887960

Here, "length", "width" and "height" have been normalized in the range of [0,1].

### 3.4 DATA BINNING

Data binning, bucketing is a data pre-processing method used to minimize the effects of small observation errors. The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin.

In the data-set, "price" is a numerical variable ranging from 5,188 to 45,400, it has 201 unique values. We can categorize them into 3 bins: low, medium, and high-priced cars.

```

In [15]: binwidth = int((max(df['price'])-min(df['price']))/3)

In [16]: bins = range(int(min(df['price'])),int(max(df['price'])),binwidth)

In [17]: df['price-binned'] = pd.cut(df['price'],bins, labels=["Low","Medium","High"])

In [18]: df.loc[15:20,['price','price-binned']]
Out[18]:

```

	price	price-binned
15	41315.0	High
16	36880.0	High
17	5151.0	Low
18	6295.0	Low
19	6575.0	Low
20	5572.0	Low

```

In [19]: df['price-binned'].dtypes
Out[19]: CategoricalDtype(categories=['Low', 'Medium', 'High'], ordered=True)

```

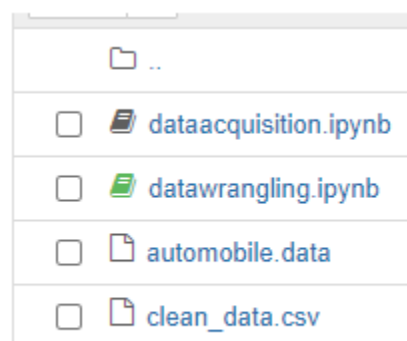
The pricing values have been separated into three equal-sized bins, and a bin array has been formed using the binwidth value, ranging from the minimum to the maximum value. Finally, a new column named 'price-binned' was added, which is similar to price.

After being cleaned, the dataset is now ready for further exploration and can be saved into another csv file.

```

In [ ]: df.to_csv('clean_df.csv')

```



### 3. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights from it.

#### 4.1 DESCRIPTIVE STATISTICS

Descriptive statistics is the type of statistics which is used to summarize and describe the dataset. It is used to describe the characteristics of data and to determine if the sample is normally distributed. It is displayed through tables, charts, frequency distributions and is generally reported as a measure of central tendency.

```
In [15]: df.describe()
```

```
Out[15]:
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-L/100km	highway-L/100km	price
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	0.830000	98.848000	0.837232	0.915250	0.899523	2555.705000	126.860000	10.170100	9.937914	8.041663	13205.690000
std	1.248557	6.038261	0.059333	0.029207	0.040610	518.594552	41.650501	4.014163	2.539415	1.844764	7966.982558
min	-2.000000	86.600000	0.678039	0.837500	0.799331	1488.000000	61.000000	7.000000	4.795918	4.351852	5118.000000
25%	0.000000	94.500000	0.800937	0.891319	0.869565	2163.000000	97.750000	8.575000	7.833333	6.911765	7775.000000
50%	1.000000	97.000000	0.832292	0.909722	0.904682	2414.000000	119.500000	9.000000	9.791667	7.833333	10270.000000
75%	2.000000	102.400000	0.881788	0.926042	0.928512	2928.250000	142.000000	9.400000	12.368421	9.400000	16500.750000
max	3.000000	120.900000	1.000000	1.000000	1.000000	4066.000000	326.000000	23.000000	18.076923	14.687500	45400.000000

For all numerical variables, the describe() method in pandas computes basic statistics. The mean, total number of data points, standard deviation, quartiles, and extreme values are all displayed. In these statistics, any NaN values are automatically skipped.

Categorical variables can be handled because they have discrete values and are divided into categories or groups. The drive system is a categorical variable in our dataset, and it is divided into three categories: forward-wheel drive, rear-wheel drive, and four-wheel drive. Using the `value_counts()` is one way to summarise categorical data.

```
In [19]: drive_wheel_count = df['drive-wheels'].value_counts()
drive_wheel_count

Out[19]: fwd      118
        rwd       74
        4wd        8
        Name: drive-wheels, dtype: int64
```

Totally, there are 118 cars in the fwd (forward-wheel drive) category, 75 cars in the rwd (rear-wheel drive) category, and 8 cars in the 4wd (four-wheel drive) category.

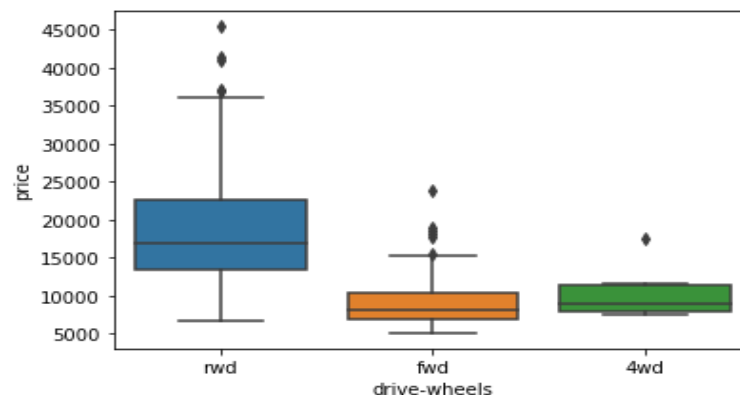
## 4.2 BOXPLOT

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”).

```
In [25]: import seaborn as sns

In [26]: sns.boxplot(x="drive-wheels",y="price",data=df)

Out[26]: <AxesSubplot:xlabel='drive-wheels', ylabel='price'>
```



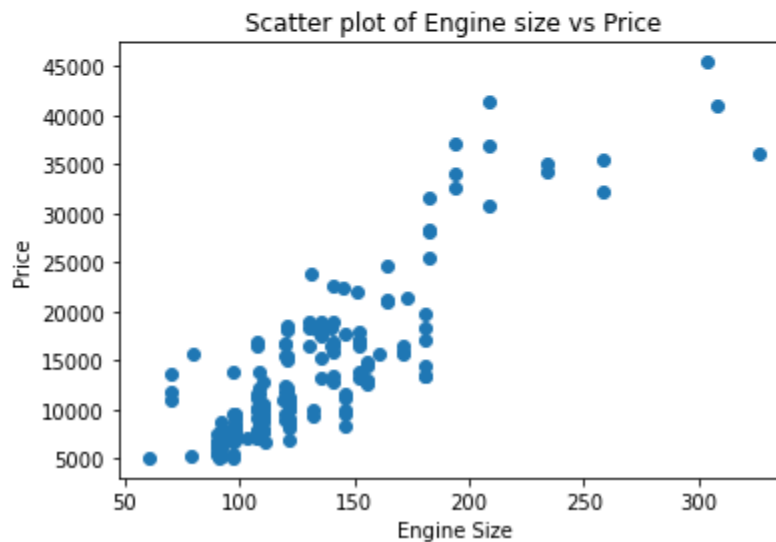


## 4.3 SCATTERPLOT

A scatterplot is a type of data display that shows the relationship between two numerical variables. Each member of the dataset gets plotted as a point whose x-y coordinates relate to its values for the two variables.

```
In [28]: import matplotlib.pyplot as plt
```

```
In [29]: x = df['engine-size']  
y = df['price']  
plt.scatter(x, y)  
plt.title('Scatter plot of Engine size vs Price')  
plt.xlabel('Engine Size')  
plt.ylabel('Price')  
plt.show()
```



The relationship between engine size and price is illustrated in the scatterplot above. The price of the car increases as the engine size increases, indicating a positive linear relationship between the two variables.

## 4.4 GROUPBY

The “groupby” method groups data by different categories. The data is grouped based on one or several variables and analysis is performed on the individual groups.

We can group "drive-wheels" and then average them to find which type of drive wheel is most valuable on average.

```
In [30]: df_group=df[['drive-wheels','body-style','price']]
drive_wheel_avg=df_group.groupby(['drive-wheels'],as_index=False).mean()
drive_wheel_avg
```

Out[30]:

	drive-wheels	price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19842.243243

According to our findings, rear-wheel drive automobiles are the most expensive on average, while 4-wheel and front-wheel drive vehicles are nearly equivalent in price.

## 4.5 ANALYSIS OF VARIANCE

Analysis of variance (ANOVA) is an analysis tool used in statistics that splits an observed aggregate variability found inside a data set into two parts: systematic factors and random factors. The systematic factors have a statistical influence on the given data set, while the random factors do not.

The `f_oneway` built-in function of the Scipy package can be used to perform the ANOVA test in Python. The data on the make and price is extracted first, and then it is grouped with other makes.

```

In [31]: group = df[['make', 'price']]
         group_anova = group.groupby(['make'], as_index=False)

In [32]: from scipy import stats

In [36]: anova_result = stats.f_oneway(group_anova.get_group('honda')['price'], group_anova.get_group('jaguar')['price'])
         anova_result
Out[36]: F_onewayResult(statistic=400.925870564337, pvalue=1.0586193512077862e-11)

In [37]: anova_result = stats.f_oneway(group_anova.get_group('honda')['price'], group_anova.get_group('subaru')['price'])
         anova_result
Out[37]: F_onewayResult(statistic=0.19744030127462606, pvalue=0.6609478240622193)

```

The ANOVA test result indicates that pricing between Hondas and Jaguars are significantly different, since the statistic score is very large (nearly 401) and the p-value is larger than 0.05. Also, because the statistic score is less than 1 and p-value is greater than 0.05, the prices of Hondas and Subarus are not significantly different.

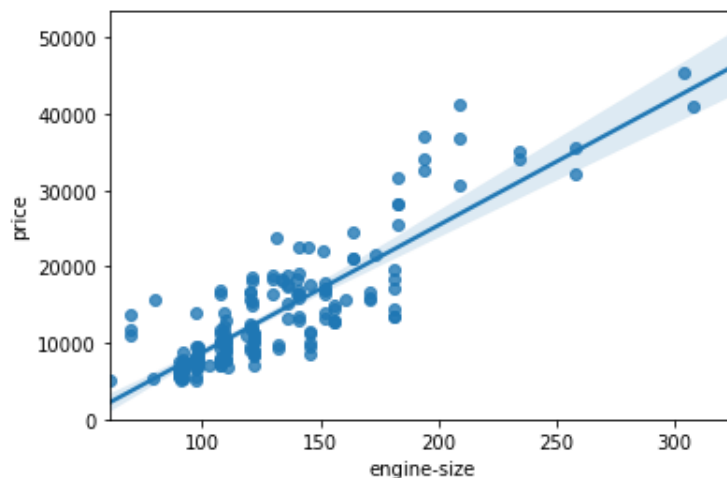
## 4.6 CORRELATION

Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect.

```

In [38]: sns.regplot(x="engine-size", y="price", data=df)
         plt.ylim(0,)
         plt.show()

```

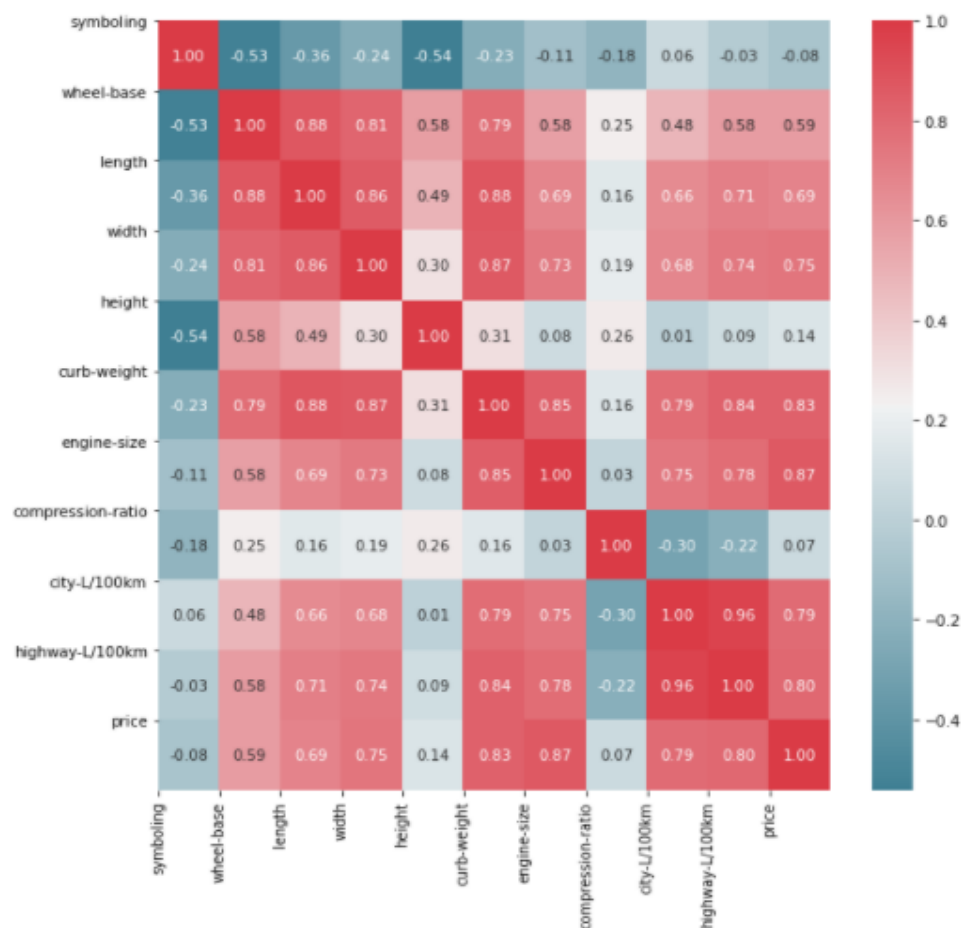


The correlation between "Engine-size" and "Price" is shown in the scatterplot above, which includes an additional linear line called a "regression line." The steepness of the straight line connecting the data points indicates a positive linear relationship or positive correlation between the two variables. With an increase in engine size, the price increases as well.

## 4.7 PEARSON CORRELATION

A Pearson correlation is a number between -1 and +1 that indicates to which extent 2 variables are linearly related. A heat map can be used to explain the correlation between the variables with the color intensity.

```
In [39]: corr = df.corr()
fig, ax = plt.subplots(figsize=(10,10))
colormap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, cmap=colormap, annot=True, fmt=".2f")
plt.xticks(range(len(corr.columns)),corr.columns)
plt.yticks(range(len(corr.columns)),corr.columns)
plt.show()
```



## **4. CONCLUSION**

At the end of this data exploration journey, the different aspects of the dataset are evident and so are the important variables to consider when predicting the car price. The variables are divided into two groups: continuous numerical variables such as length, width, curb weight, engine size, horsepower, city-L/100km, highway-L/100km, wheel-base, and bore, and categorical numerical variables such as drive-wheels. These variables will be fed into future machine learning models that will predict the cost of a used car automatically.

## 5. REFERENCE

1. <https://medium.com/>
2. <https://pythonspot.com/matplotlib-scatterplot/>