

AUTOMOBILE PRICE PREDICTION USING MACHINE LEARNING TECHNIQUES

Submitted by

Milton Rodrigues 4SO17CS068 rodriguesmilton21@gmail.com

Anson Dsouza 4SO17CS016 anson5@gmail.com



INSTRUCTOR

ABHISHEK C

Machine Learning Tutor

2020-2021

Contents

Acknowledgement	2
Abstract.....	3
About the company	4
Introduction.....	5
Problem Statement	6
Objective.....	6
1 Requirement Specification	6
1.1 Hardware Requirements.....	6
1.2 Software Requirements	6
2. Exploratory Data Analysis	7
2.1 Importing Required Libraries:	7
2.3 Importing Data Set.....	7
2.4 Data Cleaning.....	7
2.5 Data Plotting	12
2.5.1 Count of Engine Type vs Engine-Type.....	12
2.5.2 Number of vehicles vs Number of door	13
2.5.3 Number of Vehicles vs Fuel Type	14
2.5.4 Number of Vehicles vs Body Style.....	15
2.5.5 Vehicle Price vs Make	16
2.5.6 Vehicle Price vs Body Style.....	17
2.5.7 Vehicle Price vs Drive-Wheels	18
2.5.8 Pair Plot of 6 Automobile Characteristics.....	19
3. Preparing Machine Learning Model	21
3.1 Importing Required Libraries.....	21
3.1 Preparing Training and Test Data	21
3.1.1 Removing unnecessary data:.....	21
3.1.2 Creating Dummies	21
3.1.3 Splitting Data into Train and Test.....	22
3.2 Creating Machine Learning Models	24
3.2.1 Linear Regression	24
3.2.2 Support Vector Machine Regression	27
3.2.3 Random Forest Regression	30
4.ML Model Chart	32
5.Hurdles.....	32
6.Conclusion	33
7.References:.....	33

Acknowledgement

The internship opportunity I had with IC Solutions was a great chance for learning Machine Learning Techniques and Exploratory Data Analysis with different ML libraries. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it.

I would like to express my deepest gratitude and special thanks to Mr Abhishek C who is ML tutor of IC Solutions for guiding and giving us necessary advices

Finally, I would like to thank my teammate for his active involvement in the project.

Abstract

Automobile price prediction is a high-interest research area. Where predicting the price of an automobile requires expertise in that field. Considering distinct attributes of the automobile models, ML-models are trained and are used to predict accurate price of it. In this we have used three different models for making predictions. However, the dataset consists of data from 1985 “Ward’s Automotive Yearbook”. Respective performance of different algorithms where used to find one of the best suits the data sets available. The final prediction model was evaluated from the test data and the accuracy of 92.45 % was obtained.

About the company

ICS is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At ICS, we believe that service and quality is the key to success.

We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

Development - We develop responsive, functional and super fast websites. We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small view-port and slow internet connection.

Mobile Application - We offer a wide range of professional Android, iOS & Hybrid app development services for our global clients, from a start-up to a large enterprise.

Design - We offer professional Graphic design, Brochure design & Logo design. We are experts in crafting visual content to convey the right message to the customers.

Consultancy - We are here to provide you with expert advice on your design and development requirement.

Videos - We create a polished professional video that impresses your audience.

Introduction

Automobile price prediction is one of the interesting and popular problem, In this Internship we analysed the Automobile Data set using different python libraries. We used the different attributes of the automobile which is in the dataset for training different types models to find the most affective model which gave better results.

In this project mainly there where three stages Create the model, train the model, and score and test the model. Creating the model includes three sub-activities such as Get the data, Prepare the data, and define features. Train the model includes, choose and apply an algorithm. Score and test the model includes, predict new automobile prices.

Get the data: The first thing we need in Machine Learning is data. We are using Automobile price data (Raw) from 1985 The data set mainly includes information such as make, model, technical specification and price.

Prepare the data: A dataset usually requires some pre-processing before it can be used. Handling of missing values in the data set is done in this part. Either we remove the rows with missing values or we replace the value with respect to average or frequency of other elements of the same column.

Define features: In our data set, each row represents one automobile, and each column is a feature of that automobile. Let's build a model that uses a subset of the features in out dataset. You can come back later and select different features, run the experiment again, and see if you get better results.

Now, the data is ready we are using it to train models and predict the price using test data and measure its accuracy. We are doing the same process for different algorithms/models to fetch the better results.

The Algorithmic technique we are using in this project is **Regression**. There are different many regression algorithms. In this project we are mainly using three different algorithms for prediction. Namely, **Linear Regression**, **Support Vector Machine Regression** and **Decision Tress Regression**.

Problem Statement

A company wants to enter into the market by setting up their manufacturing unit and produce cars locally to give competitions to other car manufacturing companies.

They have contracted a automobile consulting company to understand the factors on which the pricing of the car depends. They would like to know the factors affecting the price of car.

The Company now has the raw data set they managed to get it from an automobile company, which contains the price and specifications of each models. Using which we are required to find the factors that affect the price.

The Company wants to know:

1. Which variables are significant in predicting the price of car.
2. Predict the price of an automobile given multiple attributes of it.

Objective

Our main objective is to do the required exploratory data analysis on the data set provided. We use different ML libraries like pandas, matplotlib, seaborn for visualizing the data to interpret things like on which factor affect the pricing of the automobile.

We also train the different ML models like Linear Regression, Decision Trees and SVM Regression using dataset we have now to predict the automobile prices.

1 Requirement Specification

1.1 Hardware Requirements

- Device: Laptop/Personal Computer
- Processor: Intel Core i3-7020U CPU @ 2.30GHz 2.30Hz
- Installed RAM: 4GM (Minimum), 8GB (Recommended)

1.2 Software Requirements

- Operating System: Windows 10
- IDE: Jupyter notebook or Google Collab
- Programming Language: Python 3

2. Exploratory Data Analysis

2.1 Importing Required Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Pandas for storing .csv data values into the data frame for further processing.
Numpy is used for Vector multiplication.
Matplotlib is used for plotting graphs.

2.3 Importing Data Set

```
df = pd.read_csv('/content/Automobile price data _Raw_.csv')
```

As you can see here, we are taking the .csv data into the data frame df.

2.4 Data Cleaning

```
df.head()
```

	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	pe
0	3	?	alfa- romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111	5
1	3	?	alfa- romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	dohc	four	130	mpfi	3.47	2.68	9.0	111	5
2	1	?	alfa- romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	9.0	154	5
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc	four	109	mpfi	3.19	3.40	10.0	102	5
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc	five	136	mpfi	3.19	3.40	8.0	115	5

Here are some of the missing values ? which may effect the analysis so we are replacing ‘?’ with NaN

Code:

```
df.replace("?", np.nan, inplace = True)
df.head()
```


Automobile Price Prediction Using Machine Learning Techniques

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3

Lets replace “NaN” by mean value in columns

```
avg_norm_loss = df["normalized-losses"].astype("float").mean(axis=0)
```

```
avg_bore = df['bore'].astype('float').mean(axis=0)
```

```
avg_stroke = df["stroke"].astype("float").mean(axis = 0)
```

```
avg_horsepower = df['horsepower'].astype('float').mean(axis=0)
```

```
avg_peakrpm = df['peak-rpm'].astype('float').mean(axis=0)
```

To see which values are present in a column and their frequency we are using the following code

```
df['num-of-doors'].value_counts()
```

Output:

```
four      114
two        89
Name: num-of-doors, dtype: int64
```

From above we can notice that the “four” has more frequency
So we replacing NaN with four

```
df["num-of-doors"].replace(np.nan, "four", inplace=True)
```

Drop all rows that have price as “NaN”

```
# simply drop whole row with NaN in "price" column
df.dropna(subset=["price"], axis=0, inplace=True)
```

```
# reset index, because we dropped two rows
df.reset_index(drop=True, inplace=True)
df.head()
```

Correcting the data format

The last step in data cleaning is to make the data in a correct format for its use.

Let us see the columns data format

```
#correct data format
```

```
df.dtypes
```

```
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
```

Here you can see that the data are of not same type, so for plotting graphs we need data of same types.

Let us convert the data types to required format using the below code

```
#changing its type
```

```
df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")
df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
df[["price"]] = df[["price"]].astype("float")
df[["peak-rpm"]] = df[["peak-rpm"]].astype("float")
df.dtypes
```

```
symboling          int64
normalized-losses  int64
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             float64
stroke           float64
compression-ratio float64
horsepower        object
peak-rpm         float64
city-mpg          int64
highway-mpg       int64
price            float64
dtype: object
```

Now you can see the data we require for plotting graphs and training are converted into float64 type

Data Standardization

It is a process of transferring data into common format which allows researcher to make the meaningful comparison

Example: converting L/100km unit of “city-mpg” and “highway-mpg” to miles per gallon using the formula

$L/100km = 235 / mpg$

```
df['city-L/100km'] = 235/df["city-mpg"]
```

```
df["highway-mpg"] = 235/df["highway-mpg"]
```

power	peak-rpm	city-mpg	highway-mpg	price	city-L/100km
111	5000.0	21	8.703704	13495.0	11.190476
111	5000.0	21	8.703704	16500.0	11.190476
154	5000.0	19	9.038462	16500.0	12.368421
102	5500.0	24	7.833333	13950.0	9.791667
115	5500.0	18	10.681818	17450.0	13.055556

Data Normalization

It is a process of transforming values of several variables into a similar range.

```
df['length'] = df['length'] / df['length'].max()
df['width'] = df['width'] / df['width'].max()
df['height'] = df['height'] / df['height'].max()

# show the scaled columns
df[["length", "width", "height"]].head()
```

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

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

Now we have our data ready for plotting graphs.

2.5 Data Plotting

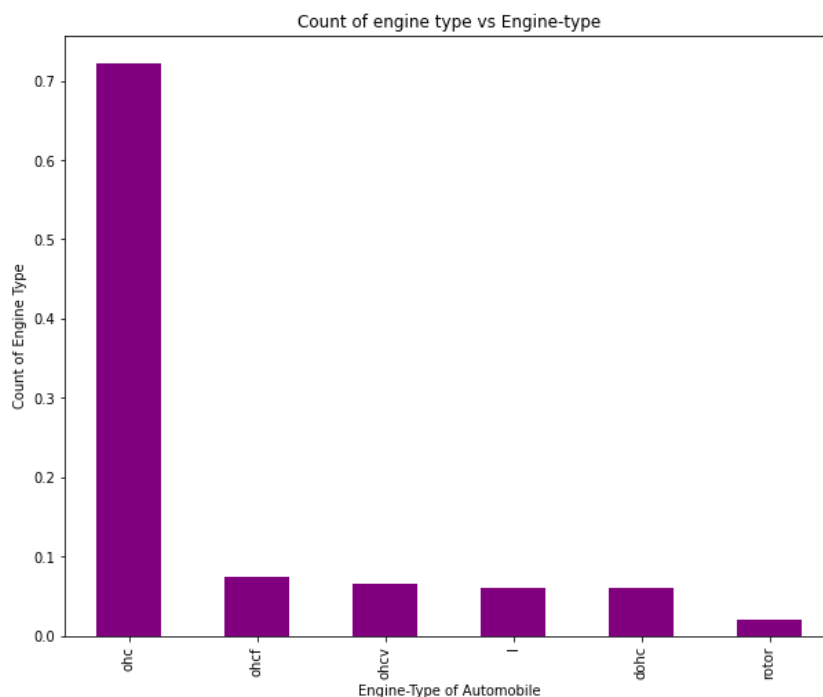
2.5.1 Count of Engine Type vs Engine-Type

Let us now plot a graph against Count of Engine types and number of engines of that type using matplotlib libraries

The below code is used to plot graph:

```
plt.figure(1)
plt.subplot()
df['engine-
type'].value_counts(normalize=True).plot(figsize=(10,8),kind='bar',color='purple')
plt.title("Count of engine type vs Engine-type")
plt.ylabel('Count of Engine Type')
plt.xlabel('Engine-Type of Automobile');
```

Output:



Inference:

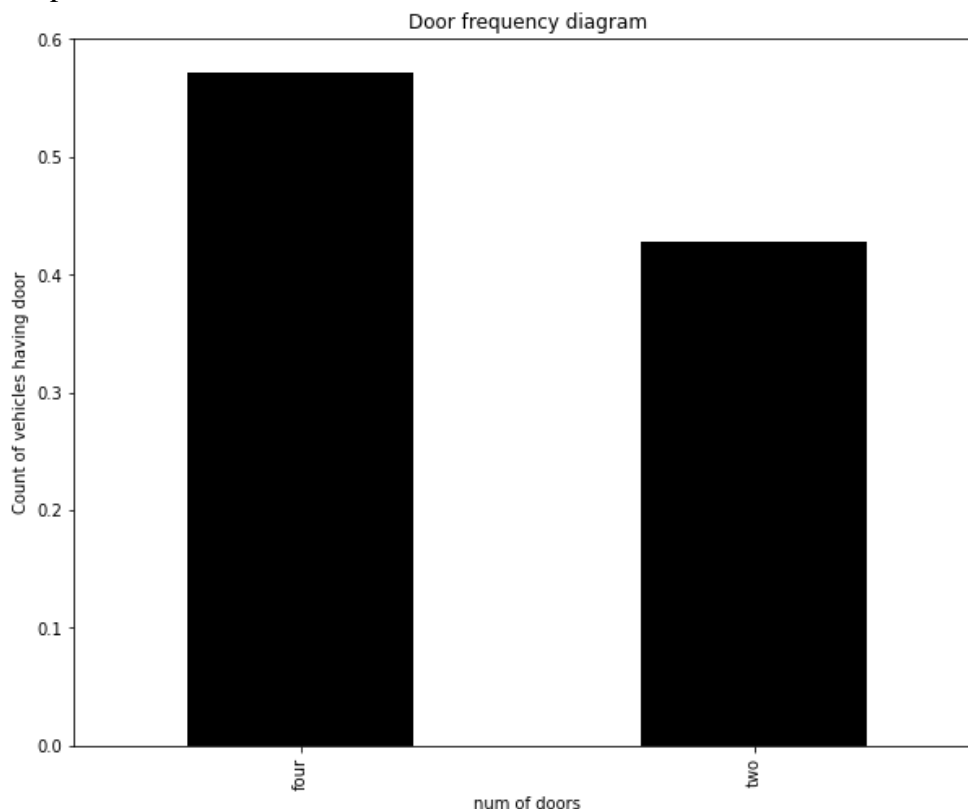
- The different types of engines available where OHC, OHCF, OHCV, DOHC, MOTOR and some engines types where not specified in the data set represented as “_”
- We could see from the graph that the number of OHC engines where used more, followed by OHCF engines
- That is 70% of the cars in 1985 used OHC engines.

2.5.2 Number of vehicles vs Number of door

The plot against the count of vehicles having the two different number of doors is plotted using the below code

```
plt.subplot()
df['num-of-
doors'].value_counts(normalize=True).plot(figsize=(10,8),kind='bar',c
olor='black')
plt.title("Door frequency diagram")
plt.ylabel('Count of vehicles having door')
plt.xlabel('num of doors');
```

Output:



Inference:

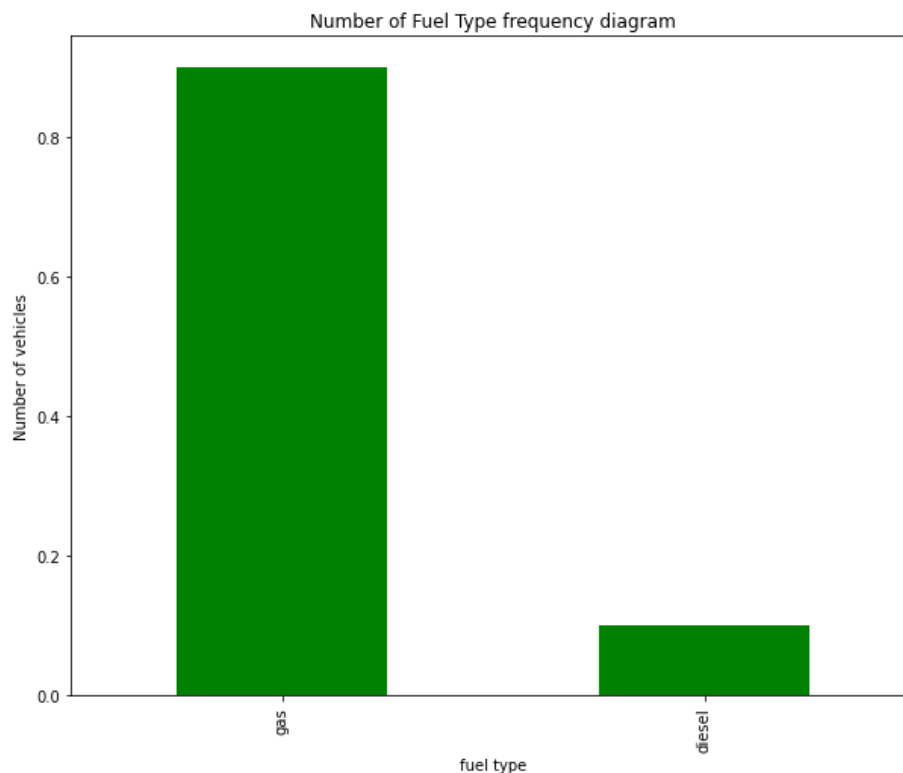
- The dataset contains cars of 4 or 2 doors as per the dataset.
- The numbers of cars with four doors where more common than the two doored vehicles
- Almost 57% of the cars has 4 doors and 43% where of 2 doors

2.5.3 Number of Vehicles vs Fuel Type

The graph Number of vehicle vs fuel type is plotted using the below given code

```
plt.subplot()  
df['fuel-  
type'].value_counts(normalize= True).plot(figsize=(10,8),kind='bar',c  
olor='green')  
plt.title("Number of Fuel Type frequency diagram")  
plt.ylabel('Number of vehicles')  
plt.xlabel('fuel type');
```

Output:



Inference:

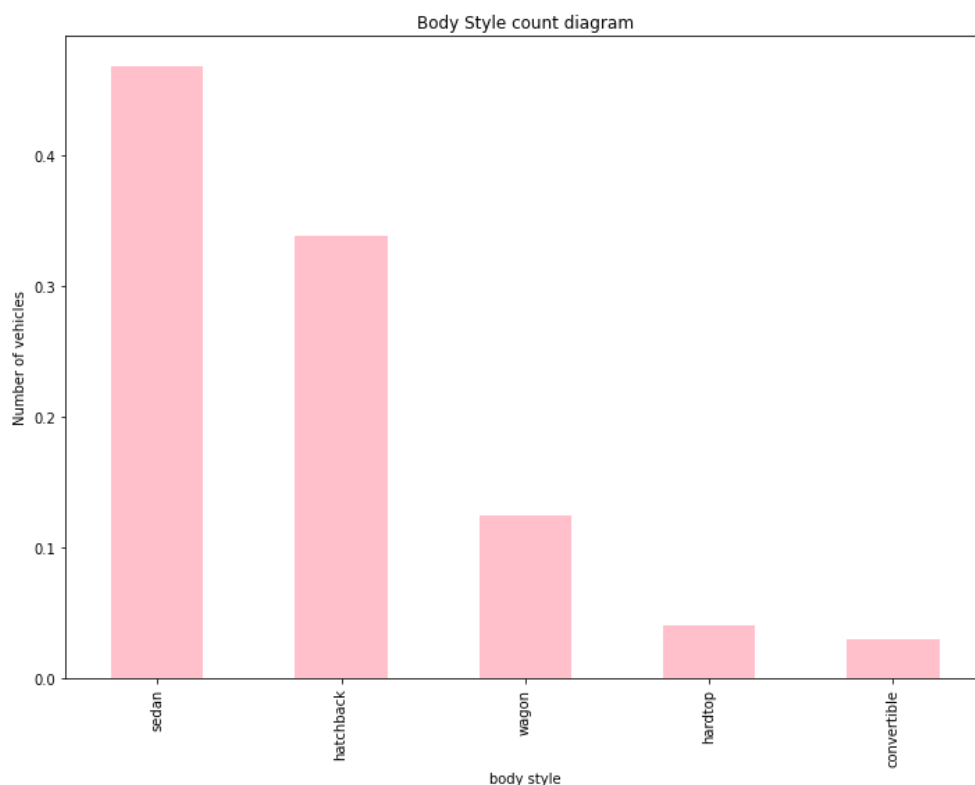
- From the plot we could infer that the vehicles used two different types of fuel that is Gas and Diesel
- The number of vehicles using Gas were more common than the vehicles which used Diesel as a fuel
- Gas is preferred by almost 85% of the vehicles

2.5.4 Number of Vehicles vs Body Style

The plot against body style and the number of vehicles having different body style is plotted using the below code

```
plt.subplot()  
df['body-  
style'].value_counts(normalize=True).plot(figsize=(10,8),kind='bar',c  
olor='pink')  
plt.title("Body Style count diagram")  
plt.ylabel('Number of vehicles')  
plt.xlabel('body style');  
plt.tight_layout()  
plt.show()
```

Output:



Inference:

- From the above graph we could infer that the different type of Body style available where sedan, hatchback, wagon, hardtop, convertible
- Most of the vehicles where of sedan body style which is around 48% followed by hatchback which is 32% and wagon
- Hardtop and convertible where rarely used body styles

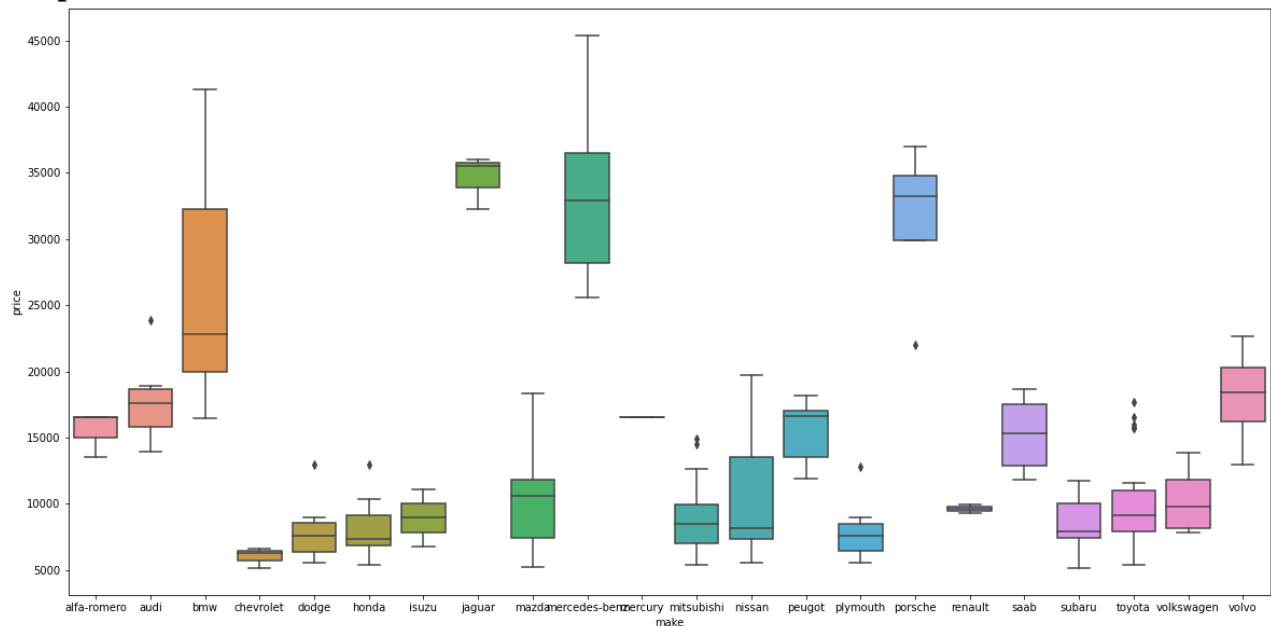
2.5.5 Vehicle Price vs Make

The plot against vehicle price and its make is plotted using the following code:

```
import seaborn as sns
plt.rcParams['figure.figsize']=(20,10)

ax = sns.boxplot(x="make", y="price", data=df)
```

Output:



Inference:

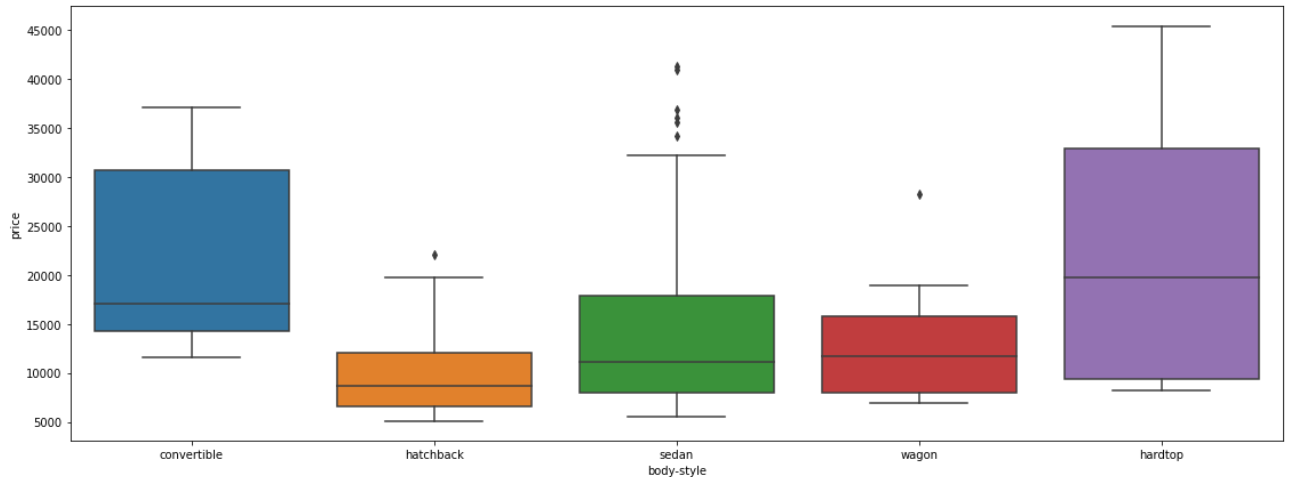
- From the above figure we could infer that there were 22 different automobile companies as you could see in the x axis
- Mercedes-Benz, BMW, Jaguar, Porsche produces most expensive cars more than \$25000,
- Chevrolet, Dodge, Honda, Mitsubishi, Nissan, Plymouth, Subaru, Toyota where the companies which produced low budget cars in range below \$25000
- Mercedes-Benz where the most expensive cars and would cost up to \$37000
- Chevrolet where the low budget vehicles

2.5.6 Vehicle Price vs Body Style

Plot against the vehicle price and body style is done using the following python code

```
plt.rcParams['figure.figsize']=(19,7)
ax = sns.boxplot(x="body-style", y="price", data=df)
```

Output:



Inference:

- We could infer that the cars with hardtop body style were more expensive and have a price more than \$30,000
- The Hatchback body styled vehicles were the low budget ones
- Hatchback and sedan turbo models are available below \$20,000
- The Hardtop styled vehicles were the most expensive but their vehicles range started from \$7,000 whereas the starting price of convertible vehicles is \$15,000

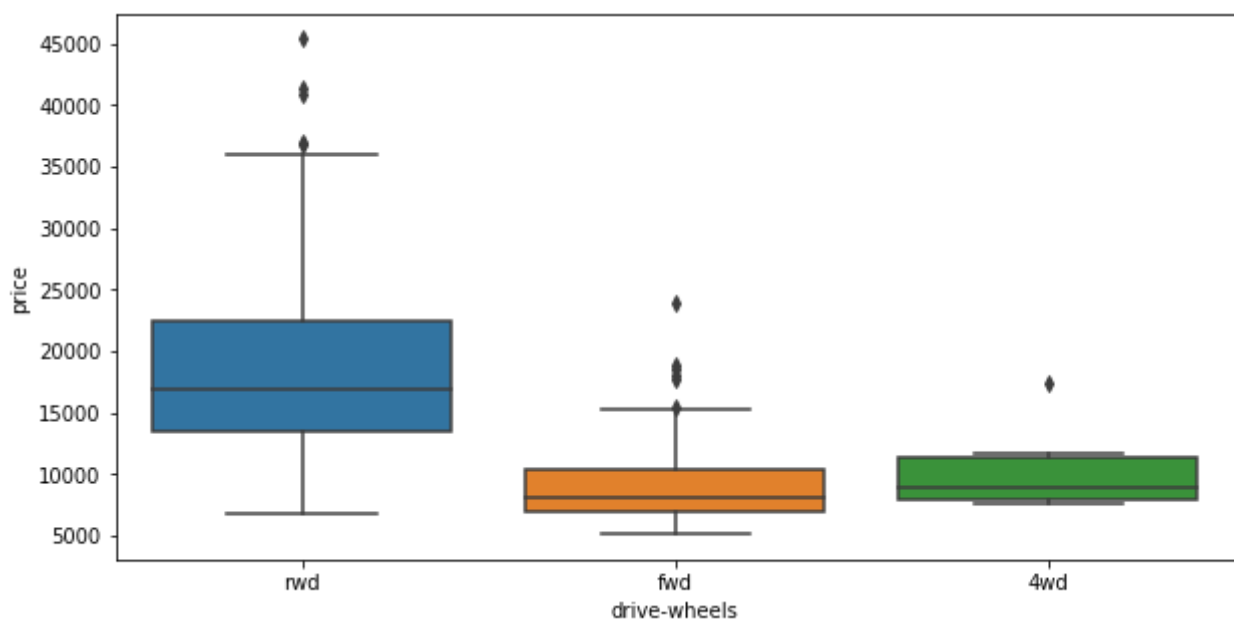
2.5.7 Vehicle Price vs Drive-Wheels

The plot against Vehicle Price and Drive Wheels is produced using the following python code.

Code:

```
plt.rcParams['figure.figsize']=(10,5)
ax = sns.boxplot(x="drive-wheels", y="price", data=df)
```

Output:



Inference:

- From the graph we could infer that the RWD drive-wheel vehicles were the most expensive ones followed by 4WD wheeled vehicles
- The FWD wheeled vehicles were the least priced vehicles
- RWD price ranges from \$12000 to \$22000
- FWD price ranges from \$700 to \$11000
- 4WD price ranges from \$800 to \$12000

2.5.8 Pair Plot of 6 Automobile Characteristics

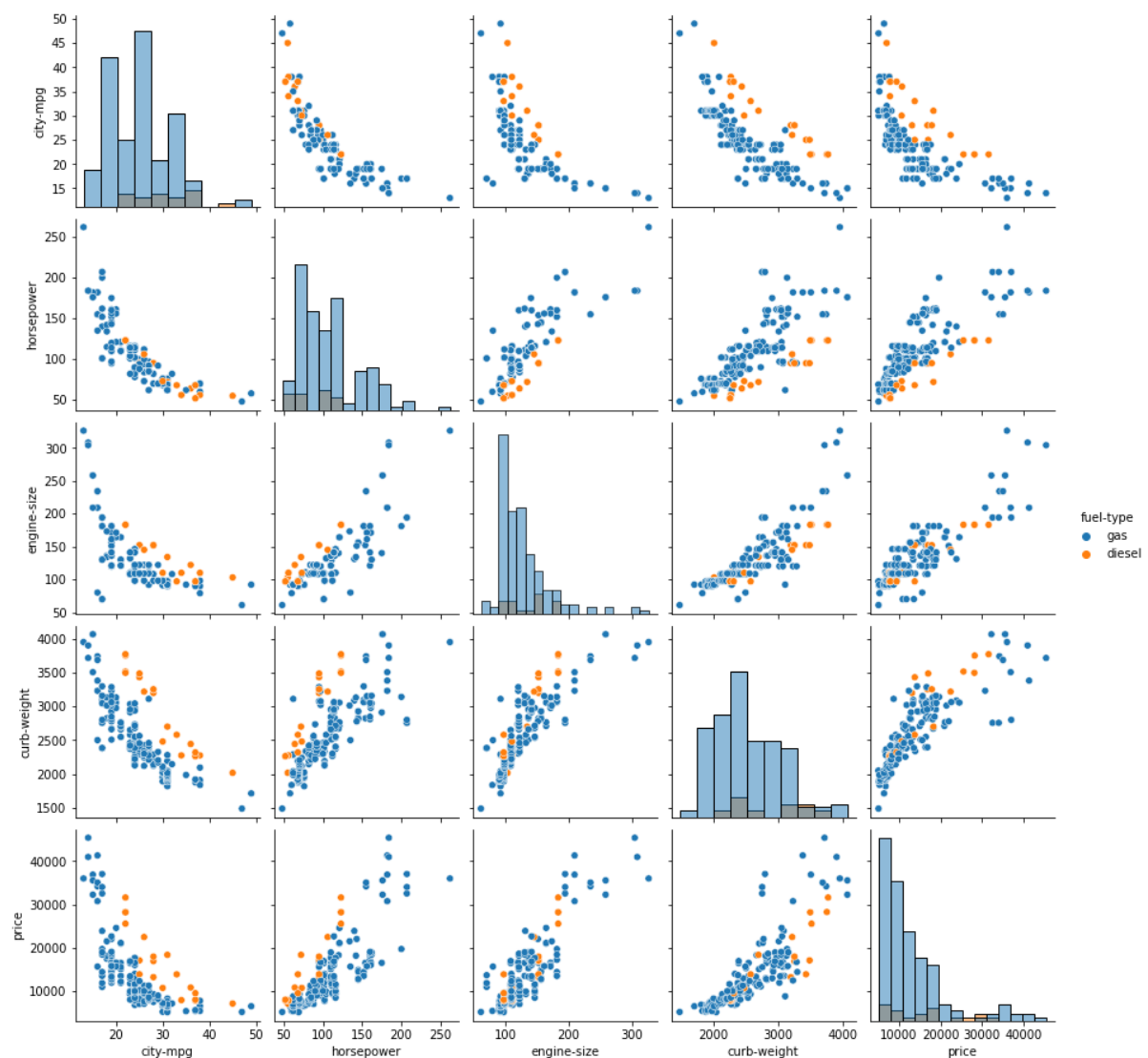
The following python code is used to plot the different pair of 6 characteristics of automobile

Using which we could use to infer a lot of things in a single python plot

Code:

```
graph = sns.pairplot(df[["city-mpg", "horsepower", "engine-size", "curb-weight", "price", "fuel-type"]], hue="fuel-type", diag_kind="hist")
```

Output:



Price Analysis:

- With increase in the price of automobile the city-mpg decreases, since most of the expensive vehicles does not provide good milage. (price vs city-mpg)
- Price of the vehicles increase with increase in the Horse Power of the vehicle (price vs Horsepower)
- Price increases with increase in engine size of the automobile .(price vs engine-size)
- Vehicle with low horse power is likely to provide more milage (horse-power vs city mpg)
- Vehicles price increases with increase in the curb-weight

From the above observation the plot against price and other characteristics of the automobile the graphs obtained where **Linear curves**. That is the data have linear relation. Since have linear relationship we could use regression algorithms for predicting the approximate price of the vehicles when other characteristics are given. Let us predict the price of vehicles in the next section using different algorithmic techniques and see which algorithm best suits the problem.

3. Preparing Machine Learning Model

3.1 Importing Required Libraries

The flowing python code import all required libraries requires for the process

Code:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score    # for mean
, vairance and diff from pred and actual
from sklearn.metrics import accuracy_score
```

3.1 Preparing Training and Test Data

3.1.1 Removing unnecessary data:

The following code is used to remove not required columns from the data frame. Because during the model training we are using only Attributes which are numbers and attributes which are related linearly.

Code:

```
df.drop(['make', 'symboling', 'normalized-losses', 'horsepower-
binned'], axis=1, inplace=True)    #drop un used values in data
df.head()
```

3.1.2 Creating Dummies

Since there are different types of vehicles represented using different strings for example there are makes likes BMW, Mercedes, to represent this in number we use the concept of representing using extra column for each type and representing it with 1 or 0 depending on the rows make. Similarly, do the same for all other attributes represented using class of strings This is code using the following python code:

Code:

```
ol=['body-style', 'drive-wheels', 'engine-location', 'engine-
type', 'fuel-system', 'num-of-doors', 'aspiration',
    'fuel-type', 'num-of-cylinders']
df=pd.get_dummies(df, columns=col, drop_first=True)
```

3.1.3 Splitting Data into Train and Test

Now let us divide this data we have into two sets namely

1. Train
2. Test

Train Data set is used for training the data model we using

Test Data set is used to measure the performance of the Model, Test Dataset are the data which is not seen by the model. It is used to check the performance of the model using unseen data.

The following code is used to divide the data set into two parts

Code:

```
train,test=train_test_split(df,test_size=0.2,random_state=0)
```

There the train and test variables are data frame used to represent train and test data respectively.

Now lets us prepare the Y(output) of the Train and Test data using the following code

Here the X of training and test data frame should not contain price. Price is put into Y variable since it is what being predicted.

Code:

```
y_train=train.price  
y_test=test.price  
train.drop('price',axis=1,inplace=True) #dropping the price  
test.drop('price',axis=1,inplace=True)
```

The data is ready to train different Regression Models!

Lets us see how the training data looks like

Automobile Price Prediction Using Machine Learning Techniques

train

	wheel- base	length	width	height	curb- weight	engine- size	bore	stroke	compression- ratio	horsepower	peak- rpm	city- mpg	highway- mpg	city- L/100km	body- style_hardtop	body- style_hatchback	st
66	106.7	0.901009	0.976389	0.918060	3495	183	3.58	3.64	21.5	123	4350.0	22	9.400000	10.681818	1	0	
26	93.7	0.755887	0.886111	0.846154	2191	98	3.03	3.39	7.6	102	5500.0	24	7.833333	9.791667	0	0	
113	107.9	0.897165	0.950000	0.948161	3252	152	3.70	3.52	21.0	95	4150.0	28	7.121212	8.392857	0	0	
168	98.4	0.846708	0.911111	0.886288	2975	146	3.62	3.50	9.3	116	4800.0	24	7.833333	9.791667	0	0	
63	104.9	0.840942	0.918056	0.909699	2700	134	3.43	3.64	22.0	72	4200.0	31	6.025641	7.580645	0	0	
...
67	115.6	0.973570	0.995833	0.941472	3770	183	3.58	3.64	21.5	123	4350.0	22	9.400000	10.681818	0	0	
192	104.3	0.907256	0.933333	0.939799	2935	141	3.78	3.15	9.5	114	5400.0	24	8.392857	9.791667	0	0	
117	93.7	0.755887	0.886111	0.846154	1967	90	2.97	3.23	9.4	68	5500.0	31	6.184211	7.580645	0	1	
47	93.1	0.764536	0.891667	0.904682	1890	91	3.03	3.15	9.0	68	5000.0	30	7.580645	7.833333	0	1	
172	102.4	0.843825	0.923611	0.918060	2414	122	3.31	3.54	8.7	92	4200.0	27	7.343750	8.703704	0	0	

160 rows × 42 columns

y_train

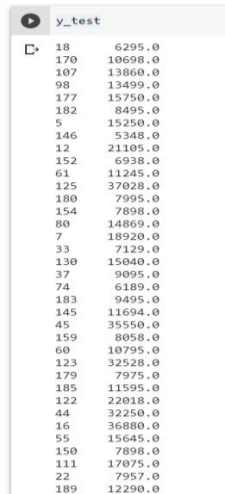
[121] y_train

```
66    28176.0
26     8558.0
113    17950.0
168    17669.0
63    18344.0
...
67    31600.0
192    15985.0
117     6229.0
47     5195.0
172    10898.0
Name: price, Length: 160, dtype: float64
```

test

	wheel- base	length	width	height	curb- weight	engine- size	bore	stroke	compression- ratio	horsepower	peak- rpm	city- mpg	highway- mpg	city- L/100km	body- style_hardtop	body- style_hatchback	st
18	94.5	0.749159	0.883333	0.869565	1874	90	3.030000	3.110000	9.6	70	5400.0	38	5.465116	6.184211	0	0	
170	102.4	0.843825	0.923611	0.918060	2480	110	3.270000	3.350000	22.5	73	4500.0	30	7.121212	7.833333	0	0	
107	114.2	0.955790	0.950000	0.981605	3430	152	3.700000	3.520000	21.0	95	4150.0	25	9.400000	9.400000	0	0	
98	100.4	0.873138	0.923611	0.921405	3095	181	3.430000	3.270000	9.0	152	5200.0	17	10.681818	13.823529	0	0	
177	104.5	0.902451	0.923611	0.904682	3151	161	3.270000	3.350000	9.2	156	5200.0	19	9.791667	12.368421	0	0	
182	97.3	0.825084	0.909722	0.931438	2275	109	3.190000	3.400000	9.0	85	5250.0	27	6.911765	8.703704	0	0	
5	99.8	0.851994	0.920833	0.887960	2507	136	3.190000	3.400000	8.5	110	5500.0	19	9.400000	12.368421	0	0	
146	95.7	0.762614	0.883333	0.911371	1985	92	3.050000	3.030000	9.0	62	4800.0	35	6.025641	6.714286	0	0	
12	101.2	0.849592	0.900000	0.908027	2765	164	3.310000	3.190000	9.0	121	4250.0	21	8.392857	11.190476	0	0	
152	95.7	0.799135	0.894444	0.886288	2081	98	3.190000	3.030000	9.0	70	4800.0	30	6.351351	7.833333	0	0	
61	98.8	0.854397	0.923611	0.928094	2425	122	3.390000	3.390000	8.6	84	4800.0	26	7.343750	9.038462	0	0	
125	89.5	0.811629	0.902778	0.862876	2800	194	3.740000	2.900000	9.5	207	5900.0	17	9.400000	13.823529	0	0	
180	97.3	0.825084	0.909722	0.931438	2264	97	3.010000	3.400000	23.0	52	4800.0	37	5.108696	6.351351	0	0	
154	95.7	0.799135	0.894444	0.886288	2275	110	3.270000	3.350000	22.5	56	4500.0	34	6.527778	6.911765	0	0	
80	95.9	0.832292	0.920833	0.839465	2921	156	3.590000	3.860000	7.0	145	5000.0	19	9.791667	12.368421	0	0	
7	105.8	0.925997	0.991667	0.931438	2954	136	3.190000	3.400000	8.5	110	5500.0	19	9.400000	12.368421	0	0	
33	93.7	0.720807	0.888889	0.879599	1956	92	2.910000	3.410000	9.2	76	6000.0	30	6.911765	7.833333	0	0	
130	99.1	0.896684	0.923611	0.938127	2707	121	2.540000	2.070000	9.3	110	5250.0	21	8.392857	11.190476	0	0	
37	96.5	0.804901	0.905556	0.891304	2289	110	3.150000	3.580000	9.0	86	5800.0	27	7.121212	8.703704	0	0	

y_test



A screenshot of a Jupyter Notebook cell showing the variable `y_test`. The variable is a list of 30 numerical values representing predicted car prices. The values are displayed in a two-column format, with the first column showing the index of each element and the second column showing the value itself. The values range from approximately 6,295.0 to 122,900.0.

Index	Value
18	6295.0
170	10698.0
107	13860.0
98	13499.0
177	15750.0
182	8495.0
5	15250.0
146	5348.0
12	21105.0
152	6938.0
61	11245.0
125	37028.0
180	7995.0
154	7898.0
80	14869.0
7	18920.0
33	7129.0
130	15040.0
37	9095.0
74	6189.0
183	9495.0
145	11694.0
45	35550.0
159	8058.0
60	10795.0
123	32528.0
179	7975.0
185	11595.0
122	22018.0
44	32250.0
16	36880.0
55	15645.0
150	7898.0
111	17075.0
22	7957.0
189	12290.0

3.2 Creating Machine Learning Models

3.2.1 Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a **linear** equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

We are using the following code to create model and fit the training data to the model and finally predict the price using test data and store in `y_pred` variable

Code:

```
regressor=LinearRegression()  
regressor.fit(train,y_train)  
  
y_pred=regressor.predict(test)
```

Here regressor is the object of the model, `y_pred` contains the predicting values by the model.

Code:

```
actual_data=np.array(y_test)  
for i in range(len(y_pred)):  
    expl=((actual_data[i]-y_pred[i])/actual_data[i])*100.0  
    print('Actual Value ${:,.2f}-----  
    Predicted value ${:,.2f}'.format(actual_data[i],y_pred[i],expl))
```

Output:

```
Actual Value $6,295.00-----Predicted value $5,976.98  
Actual Value $10,698.00-----Predicted value $13,467.57  
Actual Value $13,860.00-----Predicted value $14,097.19
```

Automobile Price Prediction Using Machine Learning Techniques

```
Actual Value $13,499.00-----Predicted value $17,388.86
Actual Value $15,750.00-----Predicted value $18,025.65
Actual Value $8,495.00-----Predicted value $9,897.61
Actual Value $15,250.00-----Predicted value $17,460.22
Actual Value $5,348.00-----Predicted value $6,223.77
Actual Value $21,105.00-----Predicted value $20,113.31
Actual Value $6,938.00-----Predicted value $7,621.49
Actual Value $11,245.00-----Predicted value $9,815.35
Actual Value $37,028.00-----Predicted value $37,224.84
Actual Value $7,995.00-----Predicted value $11,061.73
Actual Value $7,898.00-----Predicted value $9,085.20
Actual Value $14,869.00-----Predicted value $13,578.92
Actual Value $18,920.00-----Predicted value $19,614.23
Actual Value $7,129.00-----Predicted value $7,124.37
Actual Value $15,040.00-----Predicted value $24,569.23
Actual Value $9,095.00-----Predicted value $9,504.15
Actual Value $6,189.00-----Predicted value $7,100.10
Actual Value $9,495.00-----Predicted value $12,105.88
Actual Value $11,694.00-----Predicted value $9,292.51
Actual Value $35,550.00-----Predicted value $32,452.79
Actual Value $8,058.00-----Predicted value $8,802.13
Actual Value $10,795.00-----Predicted value $12,544.62
Actual Value $32,528.00-----Predicted value $34,028.00
Actual Value $7,975.00-----Predicted value $10,245.24
Actual Value $11,595.00-----Predicted value $12,829.31
Actual Value $22,018.00-----Predicted value $18,267.29
Actual Value $32,250.00-----Predicted value $32,452.79
Actual Value $36,880.00-----Predicted value $34,079.16
Actual Value $15,645.00-----Predicted value $15,388.53
Actual Value $7,898.00-----Predicted value $6,420.82
Actual Value $17,075.00-----Predicted value $14,284.06
Actual Value $7,957.00-----Predicted value $10,559.88
Actual Value $12,290.00-----Predicted value $10,584.68
Actual Value $12,170.00-----Predicted value $14,496.91
Actual Value $17,450.00-----Predicted value $16,952.14
Actual Value $8,189.00-----Predicted value $10,255.65
Actual Value $12,440.00-----Predicted value $12,570.54
Actual Value $5,118.00-----Predicted value $7,157.44
```

Here we could see the actual data present in `y_test` and the predicted data in the `y_pred`.

Let us now calculate the performance of the Data in Train and Test set

The following code block is used to calculate the accuracy of the model using **r2 score**

Code:

```
r_square=r2_score(y_test,y_pred)*100.0
r_train=r2_score(y_train,regressor.predict(train))*100.0
print('Accuracy of Test,Predict Data is {:.2f} %'.format(r_square)
)
print('Accuracy of Train Data is {:.2f} %'.format(r_train))
```

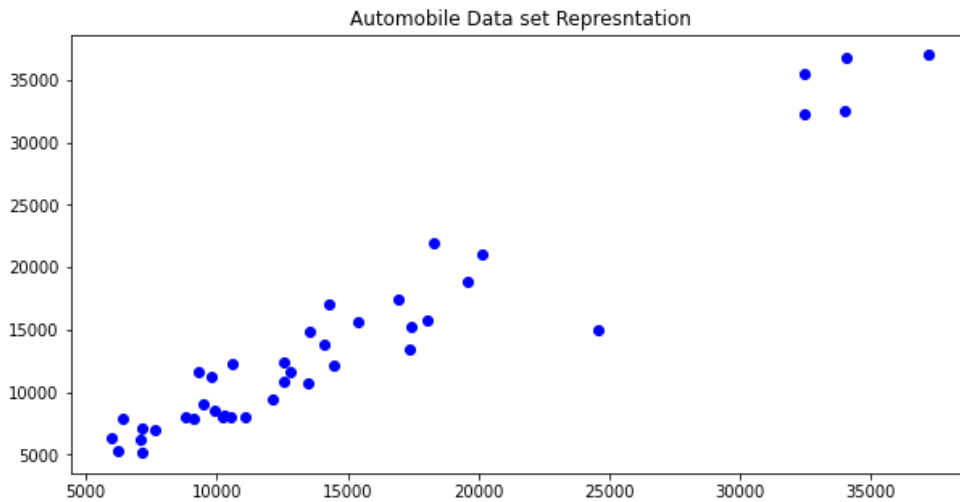
Output:

```
Accuracy of Test, Predict Data is 92.45 %
Accuracy of Train Data is 93.98 %
```

Plot of predicted result vs actual values

Code:

```
plt.scatter(y_pred,y_test,color='blue')  
plt.title('Automobile Data set Representation')  
plt.show()
```



Using Linear Regression we could get Train accuracy of 92.45% and Test accuracy of 93.98%. The graph obtained is linear

3.2.2 Support Vector Machine Regression

The following code create SVM model and the model is trained using the same we already have.

Code:

```
from sklearn import svm
SVM = svm.SVC()
SVM.fit(train,y_train)
```

Output:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None,
coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale',
kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Now the SVM model is trained, let us predict the data using test data set

```
y_predict = SVM.predict(test)
```

Now let us see the actual value and predicted value

Code:

```
actual_data=np.array(y_test)
for i in range(len(y_predict)):
    expl=((actual_data[i]-y_predict[i])/actual_data[i])*100.0
    print('Actual Value ${:,.2f}-----
Predicted value ${:,.2f}'.format(actual_data[i],y_predict[i],expl))
```

Output:

```
Actual Value $6,295.00-----Predicted value $5,572.00
Actual Value $10,698.00-----Predicted value $7,775.00
Actual Value $13,860.00-----Predicted value $16,500.00
Actual Value $13,499.00-----Predicted value $18,150.00
Actual Value $15,750.00-----Predicted value $18,150.00
Actual Value $8,495.00-----Predicted value $9,279.00
Actual Value $15,250.00-----Predicted value $9,279.00
Actual Value $5,348.00-----Predicted value $7,775.00
Actual Value $21,105.00-----Predicted value $7,775.00
Actual Value $6,938.00-----Predicted value $7,775.00
Actual Value $11,245.00-----Predicted value $8,921.00
Actual Value $37,028.00-----Predicted value $18,150.00
Actual Value $7,995.00-----Predicted value $8,921.00
Actual Value $7,898.00-----Predicted value $7,775.00
Actual Value $14,869.00-----Predicted value $16,500.00
Actual Value $18,920.00-----Predicted value $18,150.00
Actual Value $7,129.00-----Predicted value $7,295.00
Actual Value $15,040.00-----Predicted value $16,500.00
Actual Value $9,095.00-----Predicted value $9,279.00
Actual Value $6,189.00-----Predicted value $6,229.00
Actual Value $9,495.00-----Predicted value $7,775.00
Actual Value $11,694.00-----Predicted value $8,921.00
Actual Value $35,550.00-----Predicted value $18,150.00
```

Automobile Price Prediction Using Machine Learning Techniques

```
Actual Value $8,058.00-----Predicted value $7,775.00
Actual Value $10,795.00-----Predicted value $8,921.00
Actual Value $32,528.00-----Predicted value $18,150.00
Actual Value $7,975.00-----Predicted value $9,279.00
Actual Value $11,595.00-----Predicted value $9,279.00
Actual Value $22,018.00-----Predicted value $18,150.00
Actual Value $32,250.00-----Predicted value $18,150.00
Actual Value $36,880.00-----Predicted value $18,150.00
Actual Value $15,645.00-----Predicted value $7,295.00
Actual Value $7,898.00-----Predicted value $8,921.00
Actual Value $17,075.00-----Predicted value $16,500.00
Actual Value $7,957.00-----Predicted value $7,609.00
Actual Value $12,290.00-----Predicted value $9,279.00
Actual Value $12,170.00-----Predicted value $16,500.00
Actual Value $17,450.00-----Predicted value $18,150.00
Actual Value $8,189.00-----Predicted value $8,921.00
Actual Value $12,440.00-----Predicted value $16,500.00
Actual Value $5,118.00-----Predicted value $7,775.00
```

Here we could see the actual data present in `y_test` and the predicted data in the `y_predict`.

Let us now calculate the performance of the Data in Train and Test set

The following code block is used to calculate the accuracy of the model using **r2 score**

```
r_square=r2_score(y_test,y_predict)*100.0 #in LinearRegression not e
xist accuracy exist the r2_square to calc diff**2 between predict and
actual
r_train=r2_score(y_train,SVM.predict(train))*100.0
print('Accuracy of Test,Predict Data is {:.2f} %'.format(r_square)
)
print('Accuracy of Train Data is {:.2f} %'.format(r_train))
```

Output:

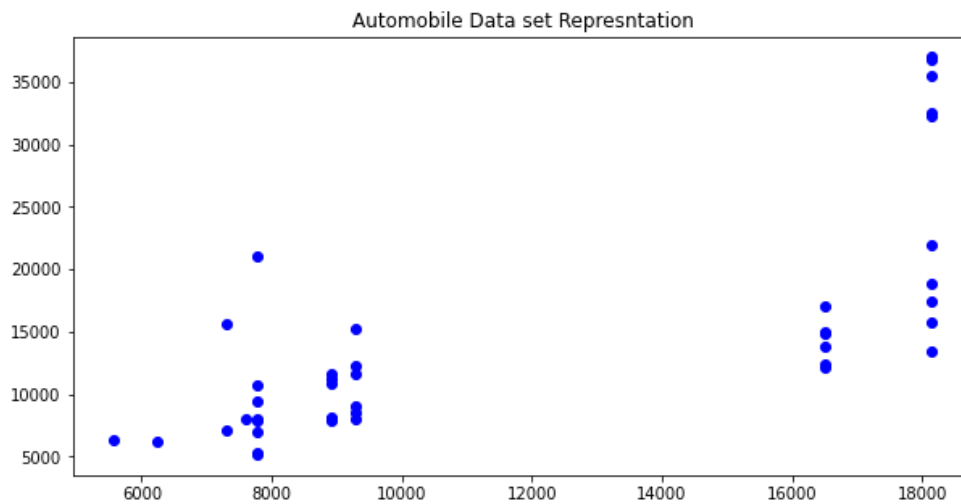
```
Accuracy of Test Predict Data is 40.42 %
Accuracy of Train Data is 44.40 %
```

Plot of predicted result vs actual values

Code:

```
plt.scatter(y_predict,y_test,color='blue')
plt.title('Automobile Data set Represntation')
plt.show()
```

Automobile Price Prediction Using Machine Learning Techniques



Using SVM Regression we could get Train accuracy of 44.40% and Test accuracy of 40.42%.

3.2.3 Random Forest Regression

We are using the following code to create model and fit the training data to the model and predict the price using test data and store in y_prediction variable

Code:

```
from sklearn.ensemble import RandomForestRegressor
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(train, y_train)
```

Output:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=2, max_features='auto',
                      max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2,
                      min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

Now the RandomForestRegressor model is trained, let us predict the data using test data set

Code:

```
y_prediction=regr.predict(test)
```

Now let us see the actual value and predicted value

```
actual_data=np.array(y_test)
for i in range(len(y_prediction)):
    expl=((actual_data[i]-y_prediction[i])/actual_data[i])*100.0
    print('Actual Value ${:,.2f}-----
Predicted value ${:,.2f}'.format(actual_data[i],y_prediction[i],expl)
)
```

```
Actual Value $6,295.00-----Predicted value $8,243.37
Actual Value $10,698.00-----Predicted value $9,043.25
Actual Value $13,860.00-----Predicted value $17,253.64
Actual Value $13,499.00-----Predicted value $16,182.00
Actual Value $15,750.00-----Predicted value $16,317.08
Actual Value $8,495.00-----Predicted value $8,339.00
Actual Value $15,250.00-----Predicted value $9,661.11
Actual Value $5,348.00-----Predicted value $8,243.37
Actual Value $21,105.00-----Predicted value $16,013.29
Actual Value $6,938.00-----Predicted value $8,243.37
Actual Value $11,245.00-----Predicted value $8,916.71
Actual Value $37,028.00-----Predicted value $32,058.44
Actual Value $7,995.00-----Predicted value $8,310.98
Actual Value $7,898.00-----Predicted value $8,310.98
Actual Value $14,869.00-----Predicted value $16,101.58
Actual Value $18,920.00-----Predicted value $16,615.43
Actual Value $7,129.00-----Predicted value $8,243.37
```

Automobile Price Prediction Using Machine Learning Techniques

```
Actual Value $15,040.00-----Predicted value $15,863.39
Actual Value $9,095.00-----Predicted value $8,402.95
Actual Value $6,189.00-----Predicted value $8,243.37
Actual Value $9,495.00-----Predicted value $8,476.37
Actual Value $11,694.00-----Predicted value $13,524.02
Actual Value $35,550.00-----Predicted value $36,161.85
Actual Value $8,058.00-----Predicted value $8,243.37
Actual Value $10,795.00-----Predicted value $8,888.69
Actual Value $32,528.00-----Predicted value $32,058.44
Actual Value $7,975.00-----Predicted value $8,271.39
Actual Value $11,595.00-----Predicted value $8,567.19
Actual Value $22,018.00-----Predicted value $16,194.16
Actual Value $32,250.00-----Predicted value $36,161.85
Actual Value $36,880.00-----Predicted value $35,575.47
Actual Value $15,645.00-----Predicted value $9,580.69
Actual Value $7,898.00-----Predicted value $8,374.93
Actual Value $17,075.00-----Predicted value $17,253.64
Actual Value $7,957.00-----Predicted value $8,355.21
Actual Value $12,290.00-----Predicted value $12,339.60
Actual Value $12,170.00-----Predicted value $15,863.39
Actual Value $17,450.00-----Predicted value $16,182.00
Actual Value $8,189.00-----Predicted value $8,916.71
Actual Value $12,440.00-----Predicted value $16,644.45
Actual Value $5,118.00-----Predicted value $8,243.37
```

Here we could see the actual data present in `y_test` and the predicted data in the `y_predict`.

Let us now calculate the performance of the Data in Train and Test set

Code:

```
r_square=r2_score(y_test,y_prediction)*100.0
r_train=r2_score(y_train,regr.predict(train))*100.0
print('Accuracy of Test,Predict Data is {:.2f} %'.format(r_square)
)
print('Accuracy of Train Data is {:.2f} %'.format(r_train))
```

Output:

```
Accuracy of Test Predict Data is 90.80 %
Accuracy of Train Data is 89.26 %
```

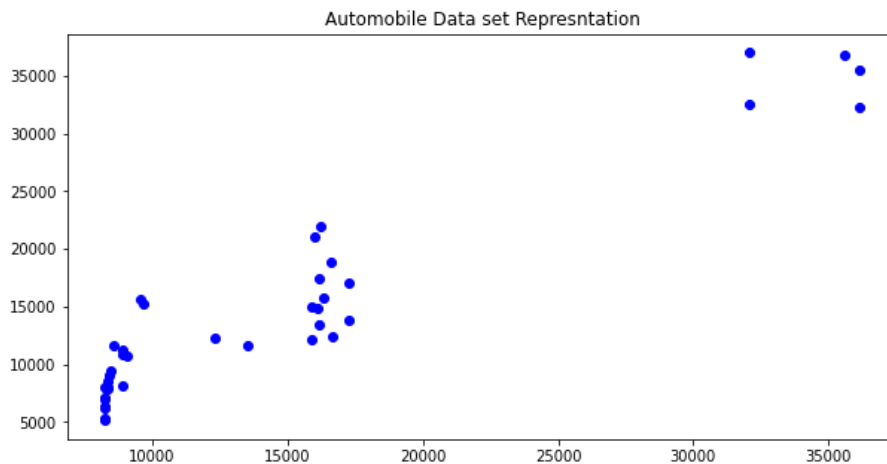
Plot of predicted result vs actual values

Code:

```
plt.scatter(y_prediction,y_test,color='blue')
plt.title('Automobile Data set Represntation')
plt.show()

plt.scatter(y_train,regr.predict(train),color='blue')
plt.title('Automobile Data set Represntation')
plt.show()
```


Automobile Price Prediction Using Machine Learning Techniques



Using Random Forest Regression we could get Train accuracy of 89.26% and Test accuracy of 90.80%.

4.ML Model Chart

R2 Score for test data set

Serial Number	ML Model	Accuracy (%)
1	Linear Regression	92.45
2	Random Forest Regression	90.80
3	Support Vector Machines	40.42

R2 Score for train data set

Serial Number	ML Model	Accuracy (%)
1	Linear Regression	93.48
2	Random Forest Regression	89.26
3	Support Vector Machines	44.40

5.Hurdles

- Less amount of training data, Since there where only 205 data entries the predictions are not so precise.We used multiple automobile characteristics to train the data rather than using one data frame column
- The raw data that is .csv file had many missing entries.We could overcome this using data cleaning techniques as described in this report's previous section

6.Conclusion

In this project three different machine learning techniques were used to predict the price of the automobile vehicle. We got different train/ test accuracies for different algorithms. Using Linear Regression, we could get the test accuracy of 92.45 which is the best accuracy which we could get out of 3 used algorithms. Followed by Random Forest Regression, we were able to get the test accuracy of 90.80. Finally, the least accurate algorithm was support vector machine which could fetch us the test accuracy of 40.42.

The main limitation of this study is very less number of records have been used. As a future we would like to collect more data and to use advanced techniques like Artificial Neural Networks, Convolutional neural networks etc

7.References:

Books:

Deep Learning by Yann Lecun

Sites:

1.Sea Born: <https://seaborn.pydata.org/>

2.Scikit Learn: <https://scikit-learn.org/stable/>

3.Pandas: <https://pandas.pydata.org/>

Kaggle: <https://www.kaggle.com/>

GoogleCollab link:

<https://colab.research.google.com/drive/1UKXCKcZ8cDje9IB7hYwx1bjylZiXhRS3?usp=sharing>

(upload csv file before running the file)