



Graphic Era
HILL UNIVERSITY

Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)
University under section 2(f) of UGC Act, 1956



Car Price Prediction Using **DATA ANALYSIS**



BTECH- 2020-24

NAME OF THE STUDENTS-

1-Mukund Maheshwari (Sec-C)

2-Puru Mittal (Sec -D)

3Mohd Talha Nadeem (Sec-C)

INTRODUCTION

From a long time since being, a continuous paradigm of transactions of commodities has been into existence. Earlier these transactions were in the form of barter system which later was translated into a monetary system. And with consideration into these, all changes that were brought about the pattern of re-selling items was affected as well. There are two ways in which the re-selling of the item is carried out. One is offline and the other being online. In offline transactions, there is a mediator present in between who is very vulnerable to being corrupt and make overly profitable transactions. The second option is online wherein there is a certain platform which lets the user find the price he might get if he goes for selling

- Kilometers traveled – We know that the number of kilometers traveled by a vehicle has a huge role to play while putting the vehicle up for sale. The more the vehicle has traveled, the older it is.
- Fiscal power – It is the power output of the vehicle. More output yields better value out of a vehicle.
- Year of registration – It is the year when the vehicle was registered with the Road Transport Authority. The newer the vehicle is, the better value it will yield. By every passing year, the value will depreciate.
- Fuel Type – There were two types of fuel types present in the dataset that we had. Petrol and Diesel. It was relatively less dominant.

It's due to the above factors that we need a system that can develop a self-learning machine learning-based system. This was the basis on which a set of objectives was supposed to be formulated. One thing that was pre-determined was that this is going to be a real-time project.

S/W AND H/W REQUIREMENTS

Minimum Requirement:

Processor: Pentium 4/1.2 Ghz Celeron/Duron processor

CPU Speed: 1.4 GHz

RAM: 100 MB

Hard Disk: 80 GB

OS: Windows 98/ 2000/XP

Sound Card: NO

Language: Machine learning and Python



Data Analysis

Problems Statement

Develop an application to demonstrate how the Automotive Industry could harness data to take informed decisions.



Include Some Features Likes -----

- Demonstrate the use of data analytics in identifying:
- Customer segments, Most popular car specification combination (engine type, fuel, mileage, etc), Right time to launch a new car, etc.
- Any other queries you can think

Some steps followed by me to solve the problem—

1. *Identifying The Problem*
2. *Identify Available Data Sources And Includes Required Data*
3. *Using Python And his Libraries*
4. *Reading The concerned dataset*
5. *Data Understanding*
6. *Data Handling*
7. *Data Visulaisation*
8. *Statistical analysis*
9. *Data preparation*
10. *Data cleaning*
11. *Splitting the Data and Features Scaling*
12. *Buliding a Linear regression Model*
13. *Model Evaluation*
14. *Consists Reviews of Other sites*
15. *Analysing the Car Segments(A1,A2.....)*
16. *Implementation and development a Web App*
17. *Communicate Result*

Overview---

This dataset consists information about car listed on Given Sample data Sheet And Other Different Site . Each columns consists information about specific features like **Make, Model ,Varient** gives information about car company model and his Varient .. **Ex-Showroom prices , km_driven** (number of kilometre car has been driven), **fuel_Types** this feature the fuel type of car (CNG , petrol,diesel etc) ,

I used Some Excel file

1. Car data (all Details)
2. Selling Record Company Wise
3. Selling Record State Wise
4. Review (features,technical,best/worst)
5. Car Segments

Setting a virtual environment----

This should be initial step when you are building an end to end project. We need new virtual environment because each project required different set and different version of libraries so by making an individual environment for a specific project we can feed all the essential library to that environment.

follow these step to do so....

```
conda create -n carfare python=3.6
#some essential package for the environment will be installed
#automatically then you will get option
[y/n] ---> y #click y
```

It might be possible that we will get issue about absence of jupyter notebook for that we have to do one more step

```
>> pip install jupyter notebook # installing jupyter notebook on
env
>> jupyter notebook
```

our environment is created , now we will do our complete project on this environment.

System Requirement:-----

ode Type	Number of Servers (BM/VM)	CPU	RAMDisk partitions	IP addresses
Control Plane	3	4 cores	16 GB	Minimum 500 GB with XFS format for installer files partition.
Storage	3	8 cores	48 GB	Minimum 500 GB with XFS format for installer files partition + minimum 500 GB with XFS format for data storage partition.
Compute	3	16 cores	64 GB	Minimum 500 GB with XFS format for installer file partition. If you add additional cores, a total of 48-50 cores distributed across multiple nodes is recommended.

Minimum server specifications for a three-node configuration on Red Hat Enterprise Linux (x86, POWER and z)

Node Type	Number of Servers (BM/VM)	CPU	RAM	Disk partition	IP addresses
Control Plane/Storage/Compute	3	24 cores	64 GB	Minimum 500 GB with XFS format for installer files partition + minimum 500 GB with XFS format for data storage partition	

Server specifications for a three-node BM configuration on POWER8 (50-100 users)

Node Type	Number of Servers (BM)	CPU	RAM	HDD	Disk partition	Notes	IP addresses
Control Plane/Storage/Compute	3 Briggs S822 LC Big Data	20 cores	512 GB	10 to 12 HDDs	10% for (2 installer partition + 10 RAID 90% for data storage partition (replication RAID factor of 3) 5 for DSX)	GbitE switch forming private	

LIBRARY'S USED

1.PANDAS

2-NUMPY

3-PICKLE

4-MATPLOT

5-SEABORN

6-PLOTLY

7-STREAMLIT



1-PANDAS

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Advantages

- Fast and efficient for manipulating and analyzing data.
- Data from different file objects can be loaded.
- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Data set merging and joining.
- Flexible reshaping and pivoting of data sets
- Provides time-series functionality.
- Powerful group by functionality for performing split-apply-combine operations on data sets.

Why Pandas is used for Data Science

This is because pandas are used in conjunction with other libraries that are used for data science. It is built on the top of the **NumPy** library which means that a lot of structures of NumPy are used or replicated in Pandas. The data produced by Pandas are often used as input for plotting functions of **Matplotlib**, statistical analysis in **SciPy**, and machine learning algorithms in **Scikit-learn**.

Pandas program can be run from any text editor but it is recommended to use Jupyter Notebook for this as Jupyter given the ability to execute code in a particular cell rather than executing the entire file. Jupyter also provides an easy way to visualize pandas data frames and plots.

2-NUMPY

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called **ndarray**, it provides a lot of supporting functions that make working with **ndarray** very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

3-PICKLE

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

Advantages of using Pickle Module:

1. Recursive objects (objects containing references to themselves):

Pickle keeps track of the objects it has already serialized, so later references to the same object won't be serialized again. (The marshal module breaks for this.)

2. Object sharing (references to the same object in different places):

This is similar to self- referencing objects; pickle stores the object once, and ensures that all other references point to the master copy. Shared objects remain shared, which can be very important for mutable objects.

3. User-defined classes and their instances:

Marshal does not support these at all, but pickle can save and restore class instances transparently. The class definition must be importable and live in the same module as when the object was stored.


4-MATPLOTT

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Importing matplotlib :

```
from matplotlib import pyplot as plt  
or  
import matplotlib.pyplot as plt
```



Basic plots in Matplotlib :

Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.

FOR EG- HISTOGRAM, PIE CHART, BAR CHART etc

5-SEABORN-

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of [matplotlib](#) library and also closely integrated to the data structures from [pandas](#).

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset.

Different categories of plot in Seaborn

Plots are basically used for visualizing the relationship between variables. Those variables can be either be completely numerical or a category like a group, class or division. Seaborn divides plot into the below categories –

- **Relational plots:** This plot is used to understand the relation between two variables.
- **Categorical plots:** This plot deals with categorical variables and how they can be visualized.
- **Distribution plots:** This plot is used for examining univariate and bivariate distributions
- **Regression plots:** The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.
- **Matrix plots:** A matrix plot is an array of scatterplots.
- **Multi-plot grids:** It is an useful approach is to draw multiple instances of the same plot on different subsets of the dataset.

6-PLOTLY

The **Plotly Python** library is an interactive open-source library. This can be a very helpful tool for data visualization and understanding the data simply and easily. plotly graph objects are a high-level interface to plotly which are easy to use. It can plot various types of graphs and charts like scatter plots, line charts, bar charts, box plots, histograms, pie charts, etc. So you all must be wondering why plotly over other visualization tools or libraries? Here's the answer –

- Plotly has hover tool capabilities that allow us to detect any outliers or anomalies in a large number of data points.
- It is visually attractive that can be accepted by a wide range of audiences.
- It allows us for the endless customization of our graphs that makes our plot more meaningful and understandable for others.

Installation:

To install this module type the below command in the terminal.

```
pip install plotly
```

various plots using this module

- **Scatter Plot:** Scatter plot represent values for two different numeric variables. They are mainly used for the representation of the relationship between two variables.
- **Bar charts:** Bar charts are used when we want to compare different groups of data and make inferences of which groups are highest and which groups are common and compare how one group is performing compared to others.

7-STREAMLIT

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. Streamlit allows you to create a stunning-looking application with only a few lines of code.

Why should data scientists use Streamlit?

The best thing about Streamlit is that you don't even need to know the basics of web development to get started or to create your first web application. So if you're somebody who's into data science and you want to deploy your models easily, quickly, and with only a few lines of code, Streamlit is a good fit.

One of the important aspects of making an application successful is to deliver it with an effective and intuitive user interface. Many of the modern data-heavy apps face the challenge of building an effective user interface quickly, without taking complicated steps. Streamlit is a promising open-source Python library, which enables developers to build attractive user interfaces in no time.

Streamlit is the easiest way especially for people with no front-end knowledge to put their code into a web application:

- No front-end (html, js, css) experience or knowledge is required.
- You don't need to spend days or months to create a web app, you can create a really beautiful machine learning or data science app in only a few hours or even minutes.
- It is compatible with the majority of Python libraries (e.g. pandas, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy(latex)).
- Less code is needed to create amazing web apps.
- Data caching simplifies and speeds up computation pipelines.

Acquiring data set and importing all the essential library-

I have taken data set from here & The data set is in csv format. Now i will import all the essential library that will be needed for this project.

```
Importing the Dependencies

import pandas as pd
import numpy as np
import pickle

# for visualization thr datasheet
import matplotlib.pyplot as plt
import seaborn as sns

# for predaction linear regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn import metrics
import plotly
import plotly.express as px
```

Data Pre Processing—

Collection, Processing And red the concerned datasheet

Car datasheet(Ex-showroomPrice to all Features),Selling Car DAtasheet(state_wise & Company wise)

Code--

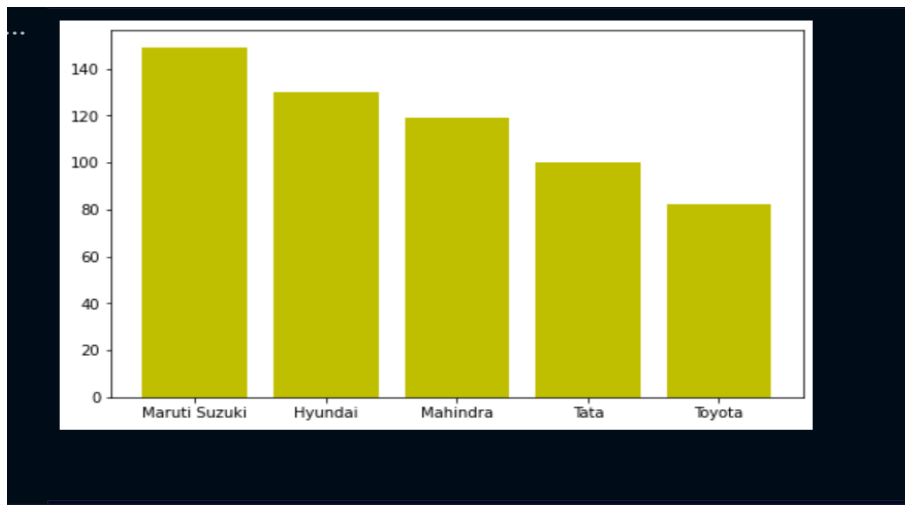
```
# Load and read the Data from csv file to Pandas Dataframe
df=pd.read_csv('datafile\cars_ds_final.csv',encoding='latin-1')
pf=pd.read_csv('datafile\sales-companywise.csv',encoding='latin-1')
cf=pd.read_csv('datafile\sale-Statewise.csv',encoding='latin-1')
```

Visualizing and analysng the Models and Companies----

```
# Bar_Chart of most cars companies
df['Make'].value_counts()[0:15]
plt.figure(figsize=(8,5))
```

```
plt.bar(list(df['Make'].value_counts()[0:5].keys()),list(df['Make'].value_counts()[0:5]),color="y")
plt.show()
```

Bar Chart ---



Data Handeling And cleaning the dataSheet:---

Here we have cars of 198 different companies and name of companies won't affect car's price ,price depends ,fuel type etc,.so i will drop the column replace and Now we will check the data type of each column if the data type is numerical then we have no such issue but if datatype is categorical then we need to convert all those categorical features into numerical values.and cleaning File

Like this---

```
# replace all null value and empty string
df=df.fillna('')
df=df.replace(' ', '')

#spilting the column
```

```

df['Displacement']=df['Displacement'].str.split(' ').str.get(0)
df['Fuel_Tank_Capacity']=df['Fuel_Tank_Capacity'].str.split(' ').str.get(0)
df['Height']=df['Height'].str.split(' ').str.get(0)
df['Length']=df['Length'].str.split(' ').str.get(0)

# repalcing Unused Data of column

df['Ex-Showroom_Price']=df['Ex-Showroom_Price'].str.replace('Rs.', '')
df['Ex-Showroom_Price']=df['Ex-Showroom_Price'].str.replace(', ', '')

df['Width']=df['Width'].str.split(' ').str.get(0)
df['Fuel_Tank_Capacity']=df['Fuel_Tank_Capacity'].str.split(' ').str.get(0)
df['Boot_Space']=df['Boot_Space'].str.split(' ').str.get(0)
df['City_Mileage']=df['City_Mileage'].str.split(' ').str.get(0)
df['City_Mileage']=df['City_Mileage'].str.split(' ').str.get(0).str.replace(', ', '')

# handling Mistake in Data Entry

df=df[df['Gears']!='7 Dual Clutch']
df=df[df['Gears']!='Single Speed Reduction Gear']

# replacing unused data
df['Ex-Showroom_Price']=df['Ex-Showroom_Price'].str.replace(', ', '')
df['Ex-Showroom_Price']=df['Ex-Showroom_Price'].str.replace('Rs.', '')
df['Displacement']=df['Displacement'].str.replace('cc', '')

# Data type correction / conversion
df[['Cylinders', 'Valves_Per_Cylinder', 'Doors', 'Seating_Capacity', 'Number_of_Airbags', 'Ex-Showroom_Price', 'Displacement']] = df[['Cylinders', 'Valves_Per_Cylinder', 'Doors', 'Seating_Capacity', 'Number_of_Airbags', 'Ex-Showroom_Price', 'Displacement']].apply(pd.to_numeric)

df['Basic_Warranty']=df['Basic_Warranty'].str.split(' ').str.get(0).str.replace('3rd', '3')

```

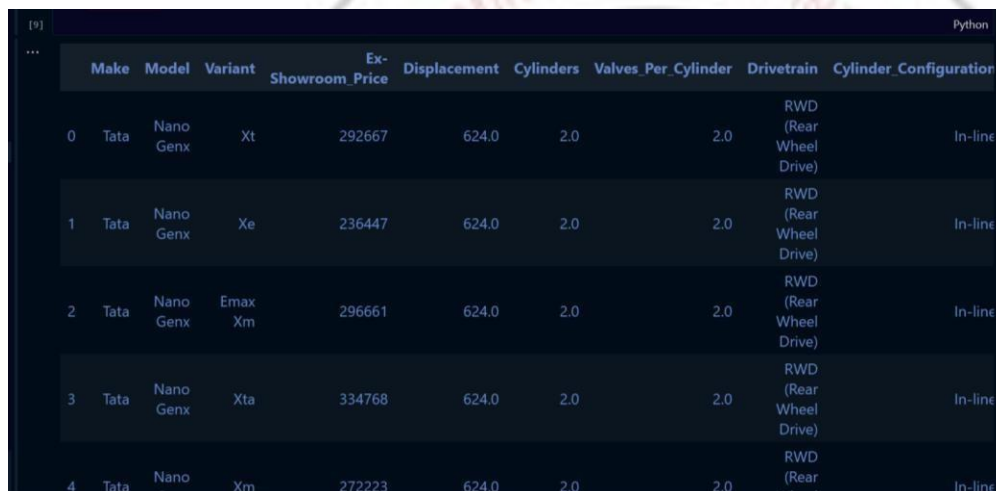
```
# checked the Data Are cleared Are Not
print(df['Ex-Showroom_Price'])
print(df['City_Mileage'])
```

Data Understanding:---

Print Some top Row and his columns code are here

```
df['Ex-Showroom_Price'].max()
df.head()
```

output—



	Make	Model	Variant	Ex-Showroom_Price	Displacement	Cylinders	Valves_Per_Cylinder	Drivetrain	Cylinder_Configuration
0	Tata	Nano Genx	Xt	292667	624.0	2.0	2.0	RWD (Rear Wheel Drive)	In-line
1	Tata	Nano Genx	Xe	236447	624.0	2.0	2.0	RWD (Rear Wheel Drive)	In-line
2	Tata	Nano Genx	Emax Xm	296661	624.0	2.0	2.0	RWD (Rear Wheel Drive)	In-line
3	Tata	Nano Genx	Xta	334768	624.0	2.0	2.0	RWD (Rear Wheel Drive)	In-line
4	Tata	Nano	Xm	272223	624.0	2.0	2.0	RWD (Rear	In-line

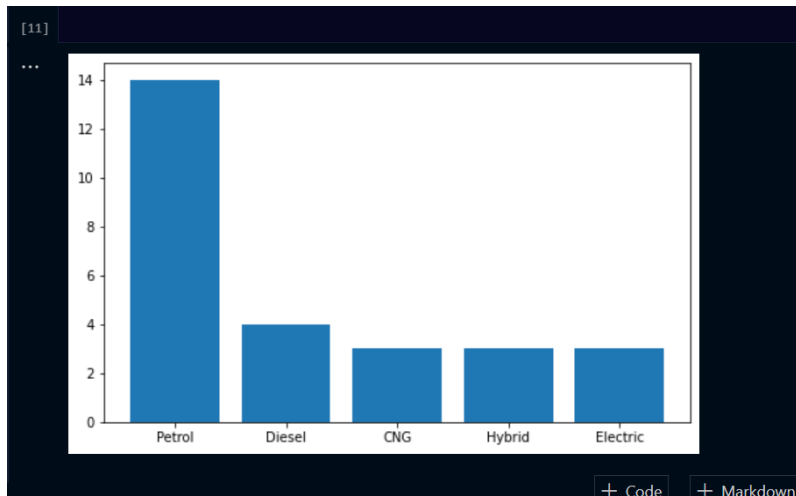
Data Visualisation:--

visualization Relation between fuel type and Showrrom price by Bar_Chart

code :--

```
df['Fuel_Type'].value_counts()
plt.figure(figsize=(8,5))
plt.bar(list(df['Fuel_Type'].value_counts()[0:5].keys()),list(df['Ex-Showroom_Price'].value_counts()[0:5]))
plt.show()
```

output----



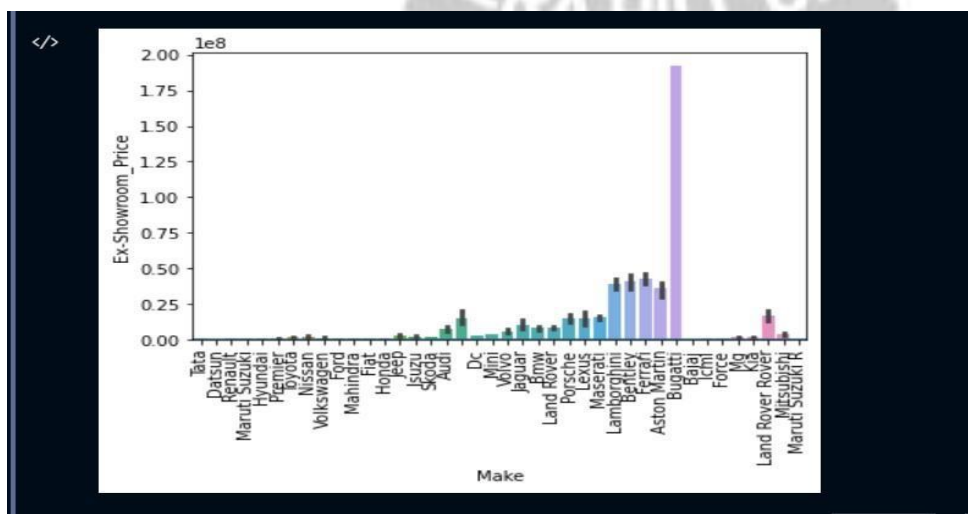
Pie Chart of Car Making Company:---

Barplot of Showroom price and company and Analysis

Code--

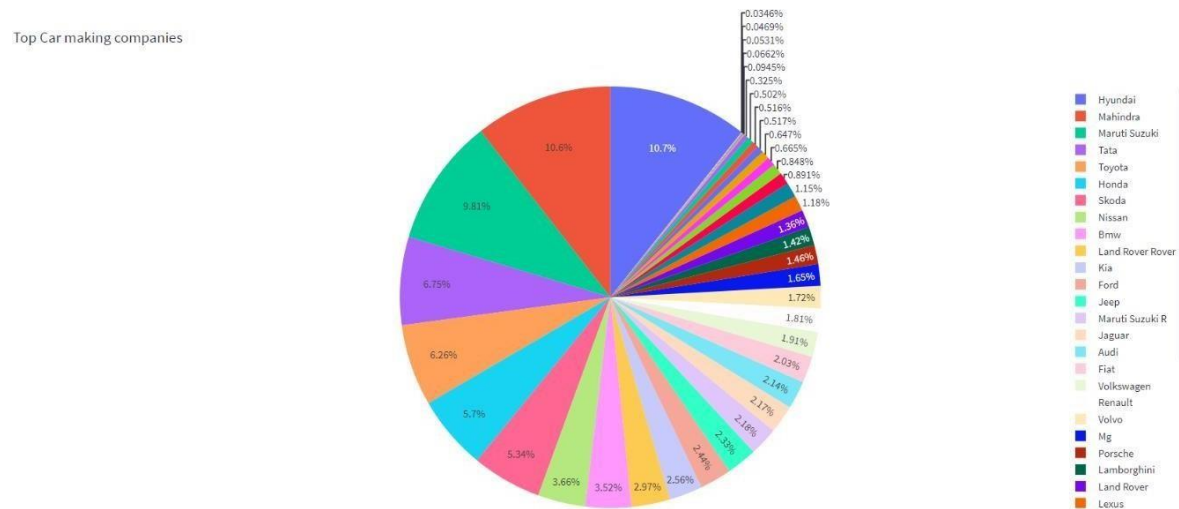
```
# barplot of Showroom price and company and Analysis
sns.distplot(df['Ex-Showroom_Price'])
sns.barplot(x=df['Make'], y=df['Ex-Showroom_Price'])
plt.xticks(rotation='vertical')
plt.show()
```

output—



And also pie chart of Top making company of car

The chart is---



Data Conversion object to int:--

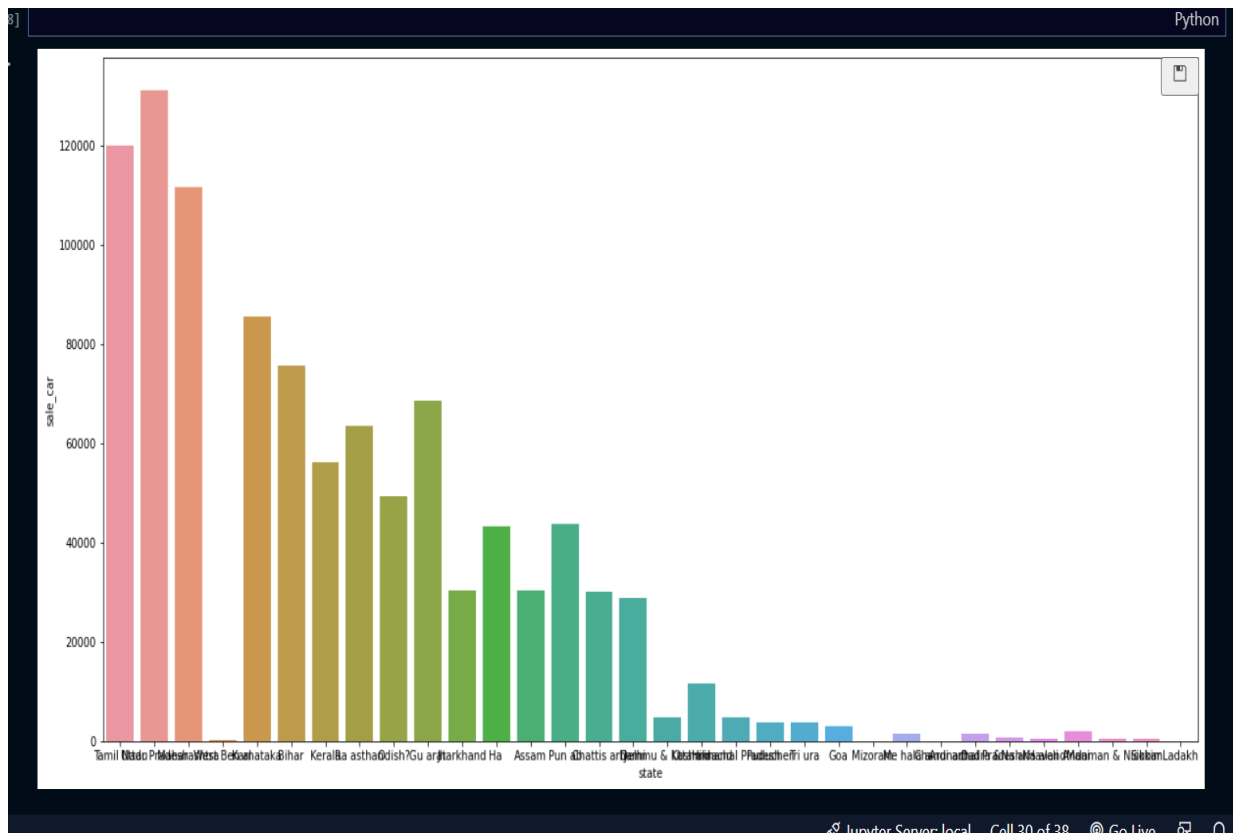
we will check the data type of each column if the data type is numerical then we have no such issue but if datatype is categorical then we need to convert all those categorical features into numerical values.

```
pf['ap22']=pf['ap22'].astype(int)
pf.info()
output---
```

```
#      Column      Non-Null Count  Dtype
---  -
0      Rank         25 non-null      int64
1      OEM           25 non-null      object
2      Bodystyle     25 non-null      object
3      Model         25 non-null      object
4      ap22          25 non-null      int32
5      ap21          25 non-null      object
6      Unnamed: 6     25 non-null      object
7      Unnamed: 7     25 non-null      object
dtypes: int32(1), int64(1), object(6)
memory usage: 1.6+ KB
```

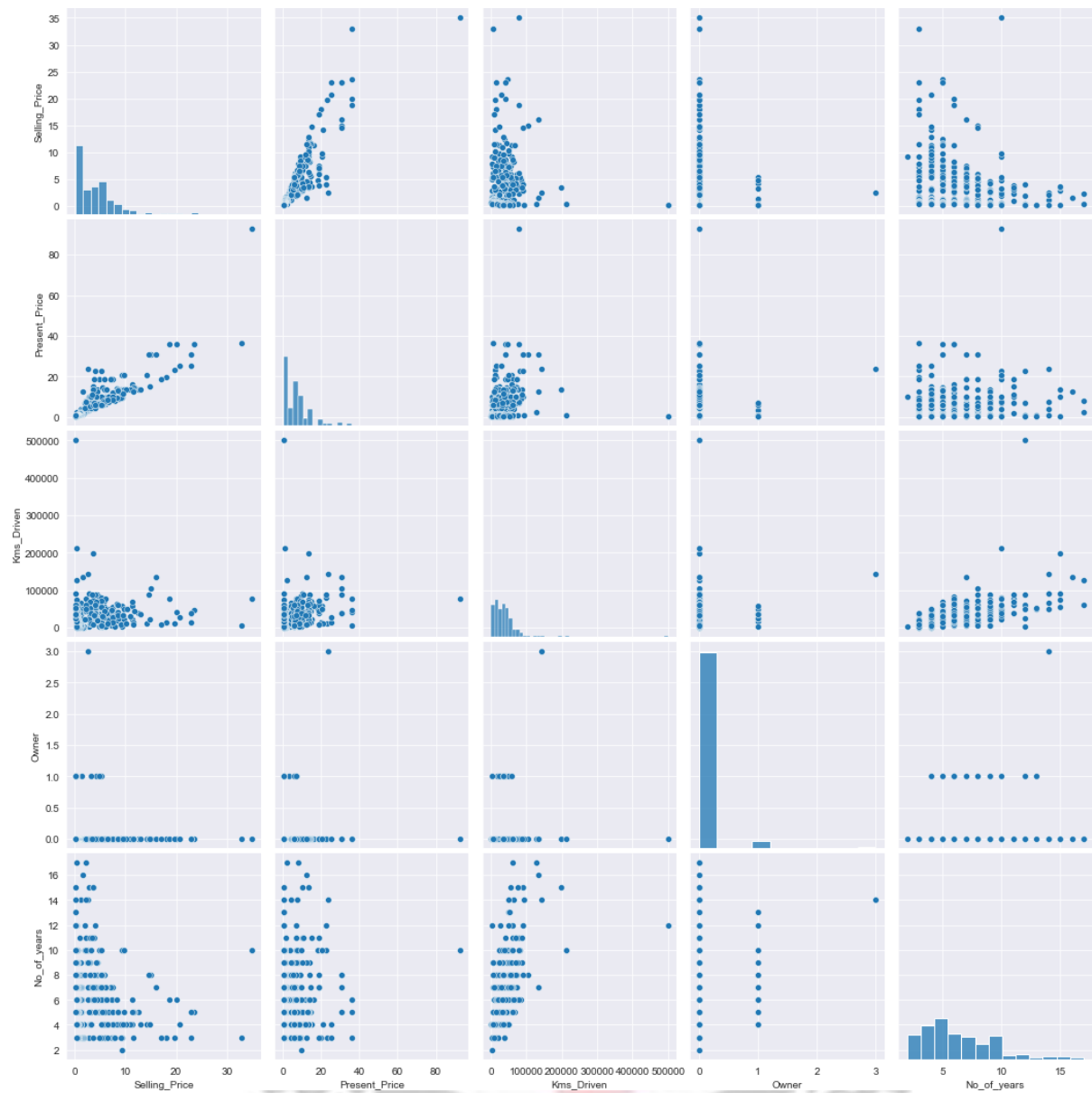
Bar Char of most selling car state wise:---

Few line code are generate a Bar chart code are like



Plotting pair plot---

We cannot visualize multi dimensional scatter plot hence by using pair plot we can visualize each and every dimension of (Dimension with numerical variable)multidimensional data precisely.

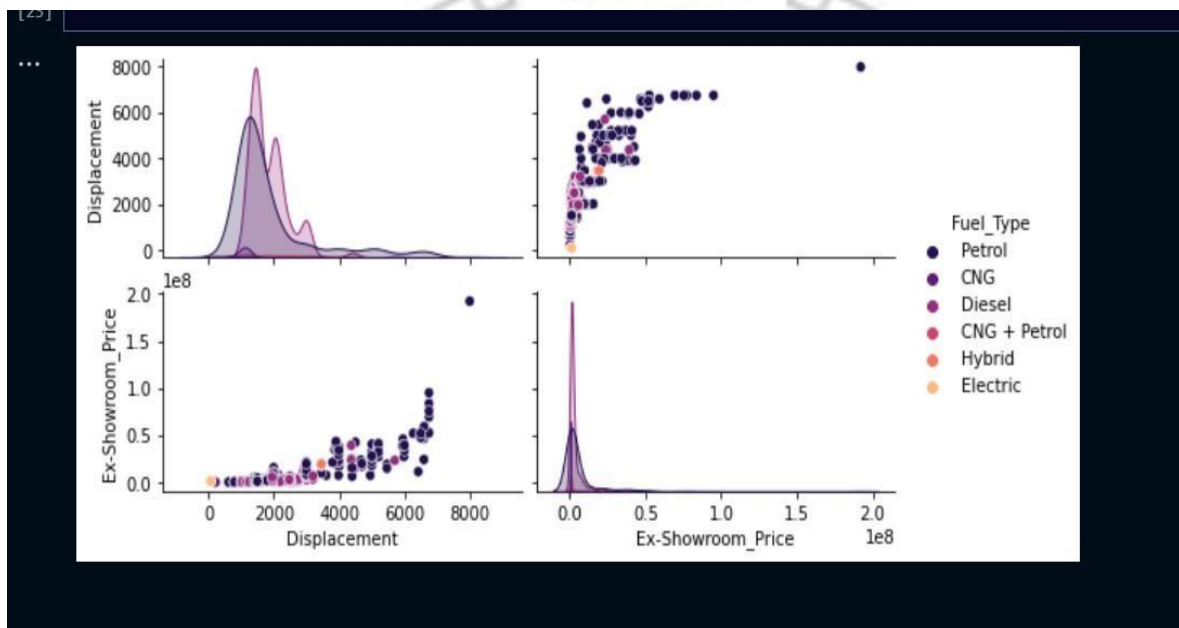


Fuel_Type are nominal feature and having small number of categorical variable we will use one hot encoding.

Visualiztion price effect of diaplacement fuel type :-

```
sns.pairplot(df,vars=[ 'Displacement', 'Ex-Showroom_Price'], hue='Fuel_Type',  
palette=sns.color_palette('magma'),diag_kind='kde',height=2,  
aspect=1.8);
```

output--

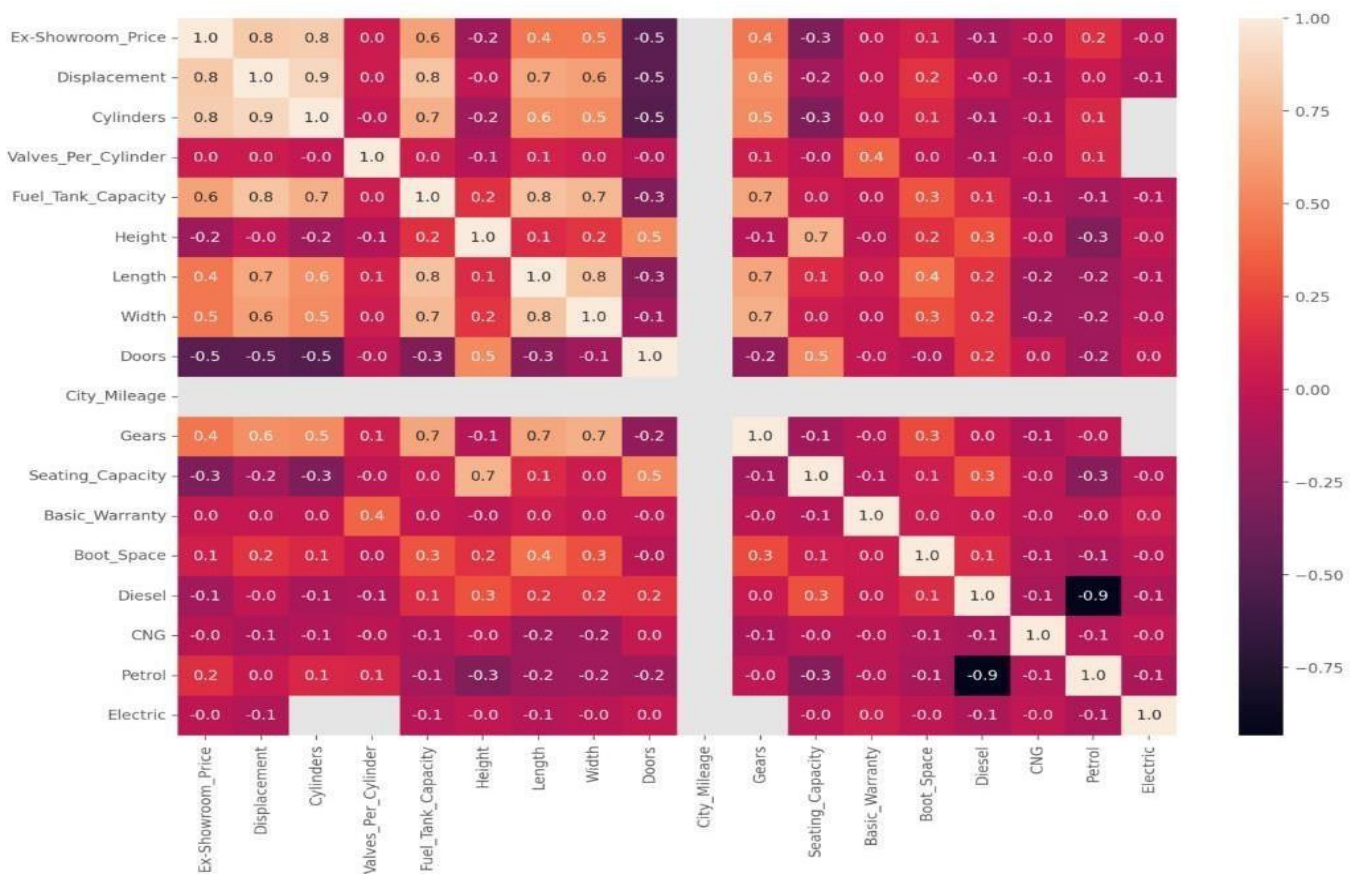


```
cat_df=pd.get_dummies(cat_df,drop_first=True)  
cat_df.head()
```

output—

	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	0	1	0	1
1	1	0	0	1
2	0	1	0	1
3	0	1	0	1
4	1	0	0	1

Now we have converted all the features into numerical variable. Here we will check correlation between feature but for this dataset we won't do feature selection, Because Feature selection is used when we have large features in a dataset but in this dataset we only check how features dataset are correlated with each other.



Pre Modelling Steps

Splitting dataset into dependent and independent variable.

```
X=df.iloc[:,1:]
Y=df.iloc[:,0]
print(X.shape)
print(Y.shape)[out] >> (301, 11)
(301,)
```

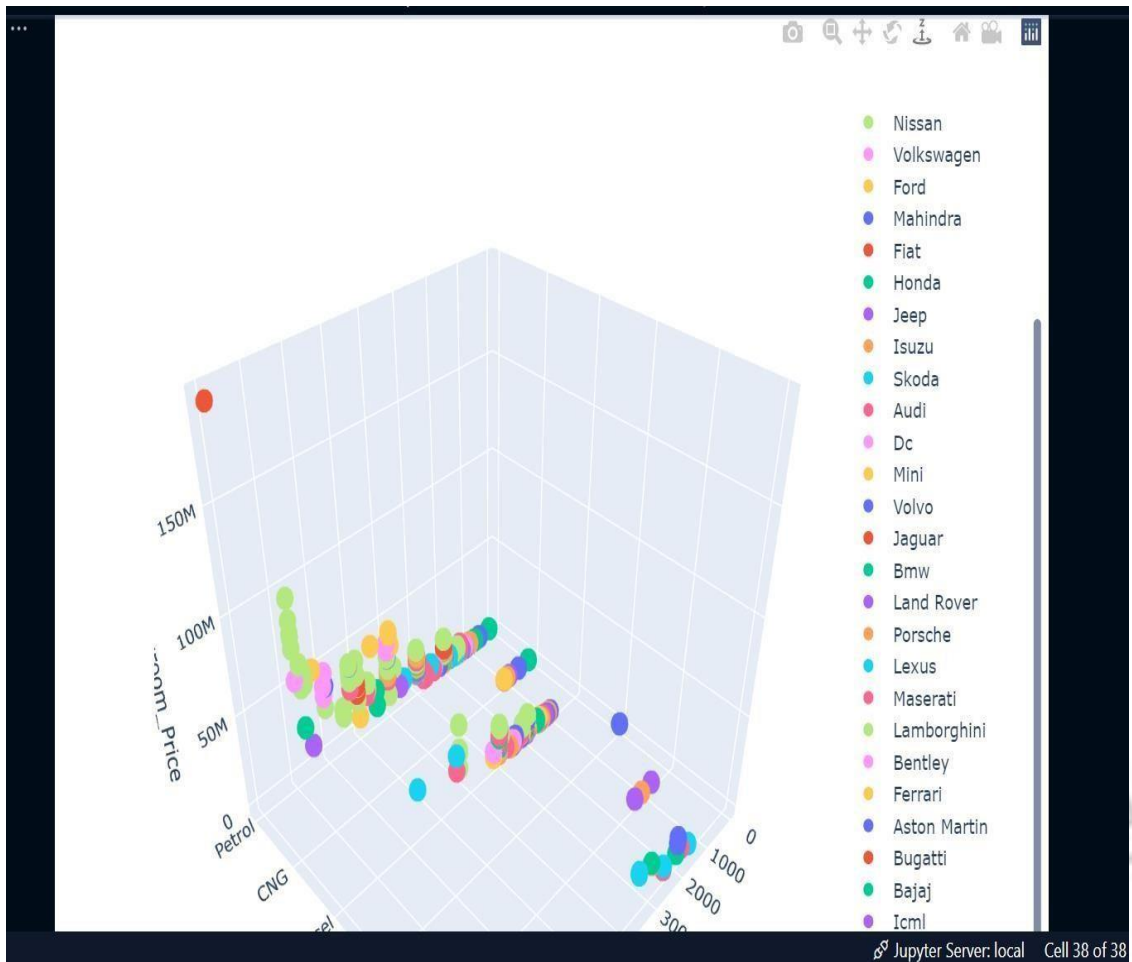
split the dataset into train and test set

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2
,random_state=1)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)[out]>> (240, 11)
(240,)
(61, 11)
(61,)
```

3D Graph Visualization:-

For 3d graph code are-

```
fig = px.scatter_3d(df, x='Displacement', z='Ex-Showroom_Price',
y='Fuel_Type',color='Make',width=800,height=750)
fig.update_layout(showlegend=True)
fig.show()
```



REFERENCES

1.YOUTUBE

2 CODE WITH HARRY

3-GITHUB

4-GOOGLE

5-UDEMY

6- SENIOR FACULTY

