Used Car Price Analysis & Resale Price Analysis

Group - G8

Course Code - CAP776

Submitted To -

Lovely Professional University Phagwara, Punjab

Date of Sumiton - 17/11/22

SUBMITTED BY: -

Krishna Kumar (12108305)

Signature of the student:

SUBMITTED TO: -

Girish Kumar Sir

Professor of Programming languages

Signature of the Teacher's

<u>To whom so ever it may concern</u>

I, **Krishna Kumar**, **12108305**, hereby declare that the work done by me on **"Used Car price Analysis and Resale Price Analysis" form June 2022 To November 2022**, Lovely Professional University, Phagwara, Panjab, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **Master in Computer Applications**.

Krishna Kumar (12108305)
Signature of the student
Dated:

ACKNOWLEDGEMENT

I would want to offer my sincere gratitude to everyone who has supported and assisted me during the process. I am grateful to the mentor from Lovely Professional University for his continued assistance throughout the project, starting with his first counsel and encouragement that resulted in the project's final report. Additionally, I want to thank My Mentor Girish Kumar Sir To guided me and increased me to do make this project.

A special thanks goes out to my team member who assisted me in finishing the assignment by sharing their knowledge and unique suggestions.

I also want to express my gratitude to my parents for their unwavering interest in and inspiration from me. Without them, I would not have been able to finish my project.

At the end, I want to thank my friends who displayed appreciation to my work and motivated me to continue my work.

Finally, I want to thank my mentor for helping to push me to keep working and for their support.

Student name :- Krishna Kumar (12108305)
Date of Submission :- November 16, 2022

ABSTRACT

A car price analysis has been a high interest research area, as it requires noticeable effort and knowledge of the field expert. Considerable number of distinct attributes are examined for the reliable and accurate prediction.

- To build a model for predicting the price of used cars the applied three machine learning techniques are Artificial Neural Network and linear regression.
- Respective performances of different algorithms were then compared to find one that best suits the available data set. The final prediction model was integrated into Java application. Furthermore, the model was evaluated using test data and the accuracy of 82% was obtained.
- To build a reselling car price prediction to selling the car in best price because some is not able to find the right place to reselling our used car, so this is opportunity to make a best car reselling workspace.

TABLES OF CONTENTS

Contents Page numbers

Introduction of Project

Details about the project:

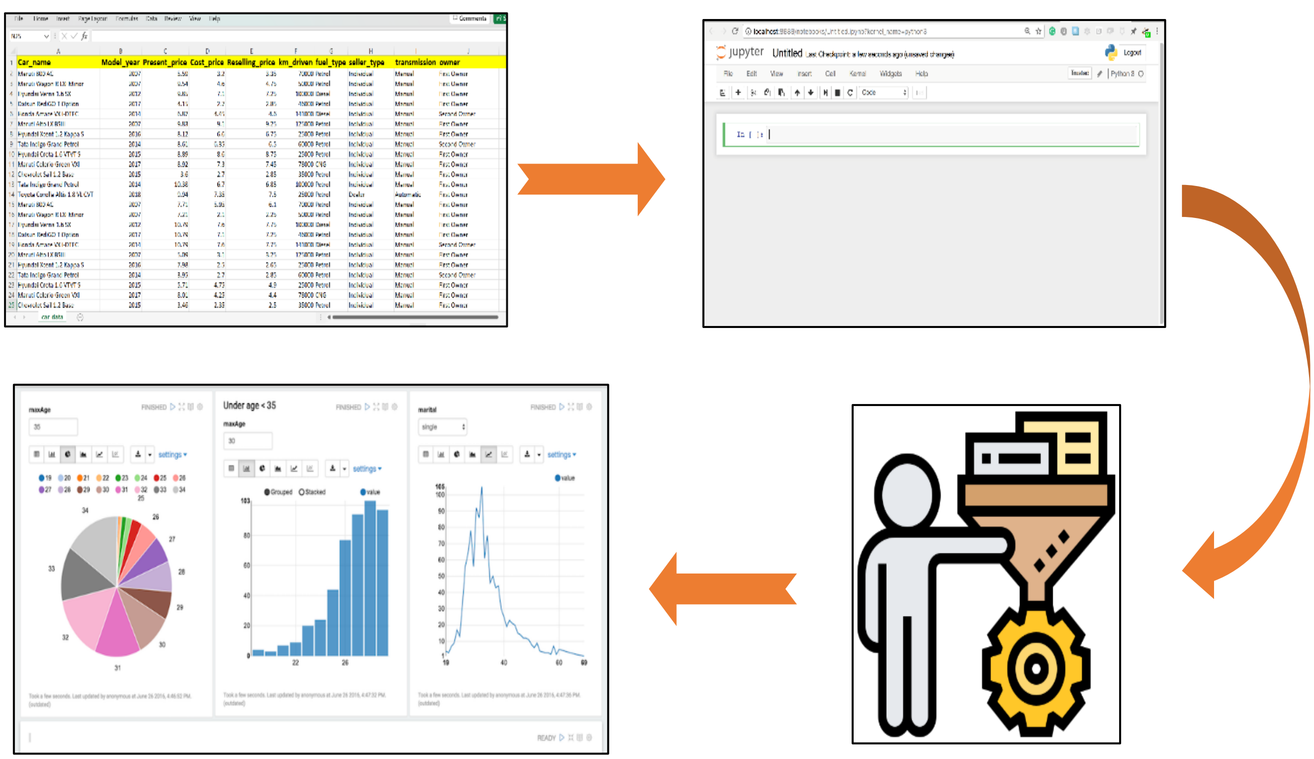
Given the variety of elements that influence a used car's market pricing, determining if the quoted price is accurate, is a difficult undertaking. The goal of this research is to create machine learning models that can precisely forecast a used car's price based on its attributes so that buyers can make educated decisions. On a dataset made up of the selling prices of various brands and models, we put several learning techniques into practice and evaluated their effectiveness. We will evaluate the effectiveness of various data sets, and the cost of the automobile will be decided by several factors. Some techniques are employed because they provide us value as an output rather than a classified value, which makes it feasible to forecast the real price and resale price of a car rather than the automobile's price range. A user interface that accepts input from any user and shows the price of a car based on their inputs has also been developed. The challenge of predicting car prices is crucial and significant, especially when the vehicle is old and not brand-new. As the market for second-hand automobiles grows, more and more potential customers are looking for alternatives to brand-new vehicles.

Objective of the project

- To develop an efficient and effective model which predicts the price of a used car according to user's inputs.
- To develop a model which visualized the data reselling price of car according to user's inputs.
- To achieve good accuracy.
- To develop a User Interface (UI) which is user-friendly and takes input from the user and analyse the price.
- Getting the data from data base using some technic.
- The objective of the project is to find the possible resale value of a car based on its model, brand, vehicle type, fuel type and whether repaired or not.

What we can do in this project

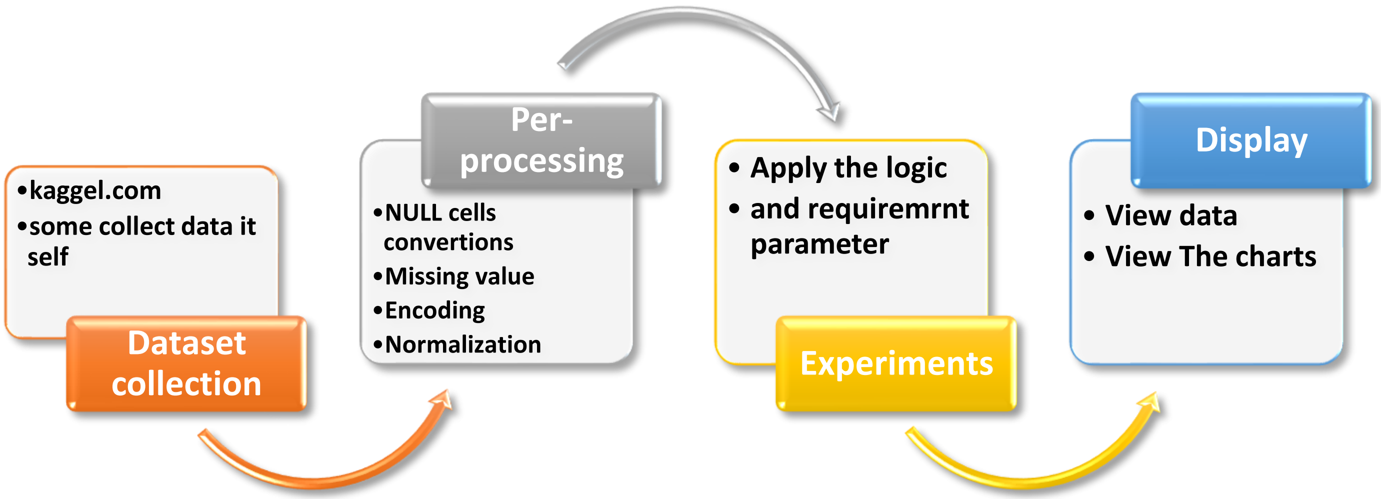
In this project we are main propose to be working on data analysis and manage those data with the help of programming language and using the set of data to predict the price to reselling car through different field like car_name , selling_cost, reselling_cost, model_year, fuel_type, present_price and oner. and through this filter field we can analysis the data and predict the car price, and all the data facing through using the csv data set file, and also showing those data through various graph like pie chart, scatter graph, bar chart, and so on.



Methodology

Using methodology

The benchmark dataset from kaggle.com and some other data were scraped for the study on Indian autos in order to build an efficient intelligent model. The following is the project's methodology:



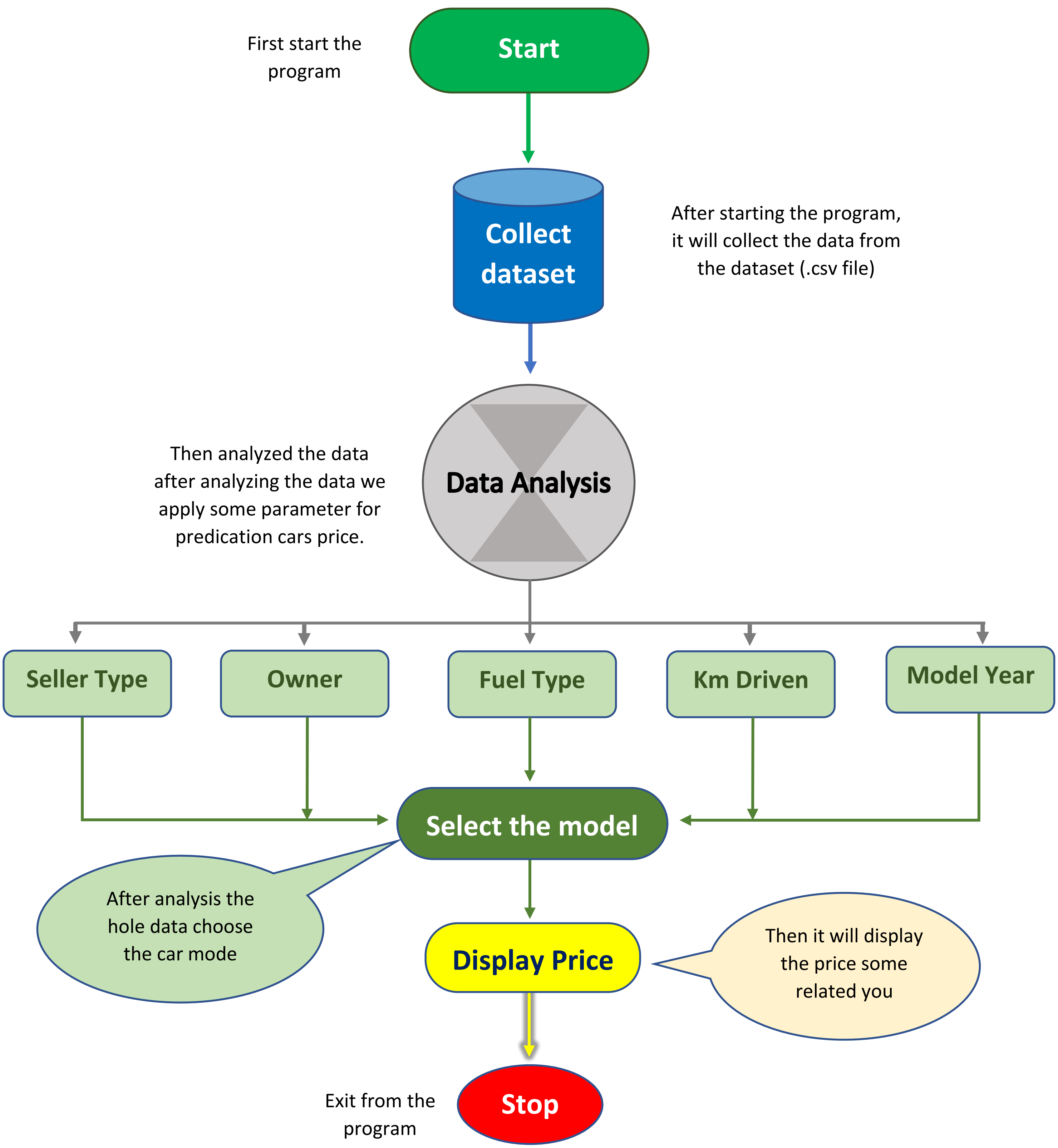
The dataset was pre-processed after data collection to remove samples with missing data, remove non-numerical components from numerical attributes, convert categorical values into numerical values (if necessary), fix any discrepancies in the units, and remove attributes that don't affect price evaluations if necessary to reduce the complexity of the model.

Data preparation and understanding is a crucial step in building a model because it provides insight into the data and identifies any corrections or modifications that need to be made before designing and implementing the model. To gain a deeper understanding of the data's quality, including outliers and skewedness of the figures, descriptive statistics of categorical and numerical variables were conducted. Additionally, it helps to be aware of the key factors that influence how prices are determined. This was accomplished by creating a correlation matrix for each characteristic to comprehend the relationships between the various components.

The data is then organized and translated into a format that the data mining technology can process. Various data mining techniques have been developed to forecast used automobile prices and values.

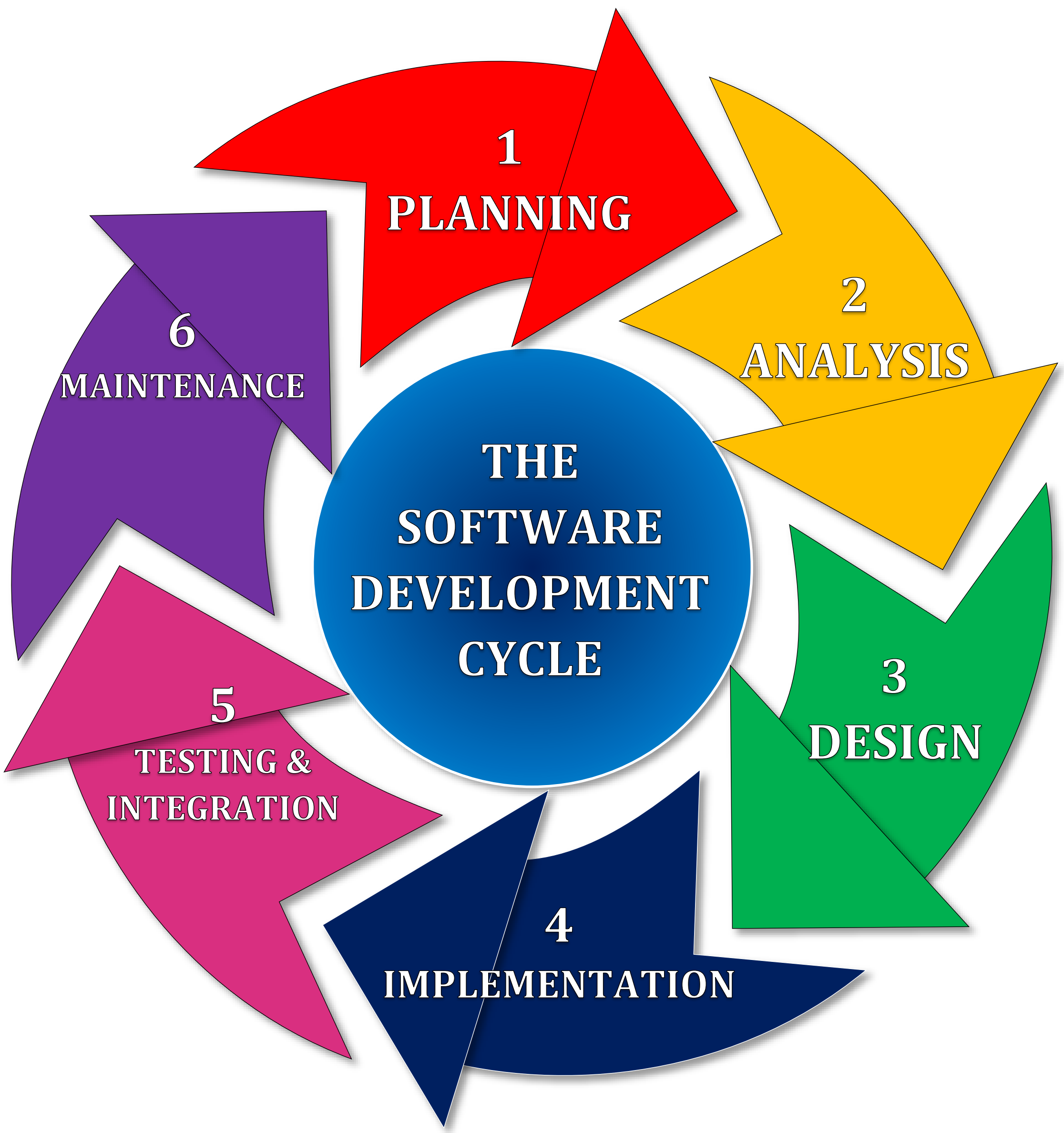
Three models are suggested to be constructed in this study using certain approach and logic.

Flowchart diagram



Software Development Life Cycle

The software business uses a wide range of procedures, such as those for software analysis, development, maintenance, and publication. Software services including training, documentation, and consultancy are also a part of this sector. Our emphasis is on the life cycle of software development (SDLC). As a result, various project kinds have distinct needs. Therefore, it could be necessary to select the SDLC stages based on the unique requirements of the project. We have a variety of software development methodologies to select from when implementing software because of these distinct demands and requirements.



Requirement Analysis

In systems engineering and software engineering, requirements analysis refers to the processes involved in identifying the requirements that must be satisfied for a new or modified project, taking into account the potentially conflicting requirements of the various stakeholders, as well as in analyzing, documenting, validating, and managing software or system requirements.
For a system or piece of software to succeed or fail, requirements analysis is essential. project. The specifications should be written down, implementable, measurable, and tested. Traceable, connected to recognized business opportunities or demands, and described with enough specificity for system design.

System Requirement

Our system can be used in windows 7, and windows 8 and windows 10-11 with 32 bit, and 64 bits
operating system and also supported for another platform such as Linux OS X.

For Windows 7 and Windows 8 based computers, higher processor with 4 GB ram

Software Requirements:

Compatible operating system: Windows, Mac.
Software: Python, PIP 2.7 or above, Jupyter notebook.
Library: NumPy, Pandas, Metplotlib, etc.
Web browser: chrome, Firefox, etc.

Hardware Requirements:

Hardware recommends by all the software needed.
RAM: 4GB or more
Hard Drive: 10 GB or more
Processor dual core 2.4GHz (i5 or i7 series Intel processor or equivalent AMD)

Motivation for the project

The motivation of this project comes with observing their difficulties in busy situation while I was there as I usually visit the place to purchase a secondhand car. Personally, I don't have much time to analysis of car. Without a system it is very difficult. Other than that, I value learning data analysis tool and technic and development because I have less experience in this area and it will be helpful in future for my carrier.

New expectation is there for this project due the current situation in the country with Covid-19 virus. This kind of solution will help to make the online data analysis.

Brief description of the work done

In this project I am creating a data set and then linking our program through using python programming, and using some library like NumPy, pandas, etc. And after that processing all data displaying data through some chart and graph.

Problem during working on this project

During working on this project, I am facing few problems like large dataset creation, and handle it, installing some library, coding in error etc.

But I can manage the all problem with thinking on it and checking line-by-line code so many times. And learn how to handle the large data set and how to work on any large project. I can solve the problem with help of my mentor and my colleagues, notes and through googling.

Code with Snipping :

Importing the all library which we need in this project:

```
import pandas as pd

import numpy as np

import plotly.figure_factory as ff
```

import plotly.express as px

import plotly.graph_objects as go

import matplotlib.pyplot as plt

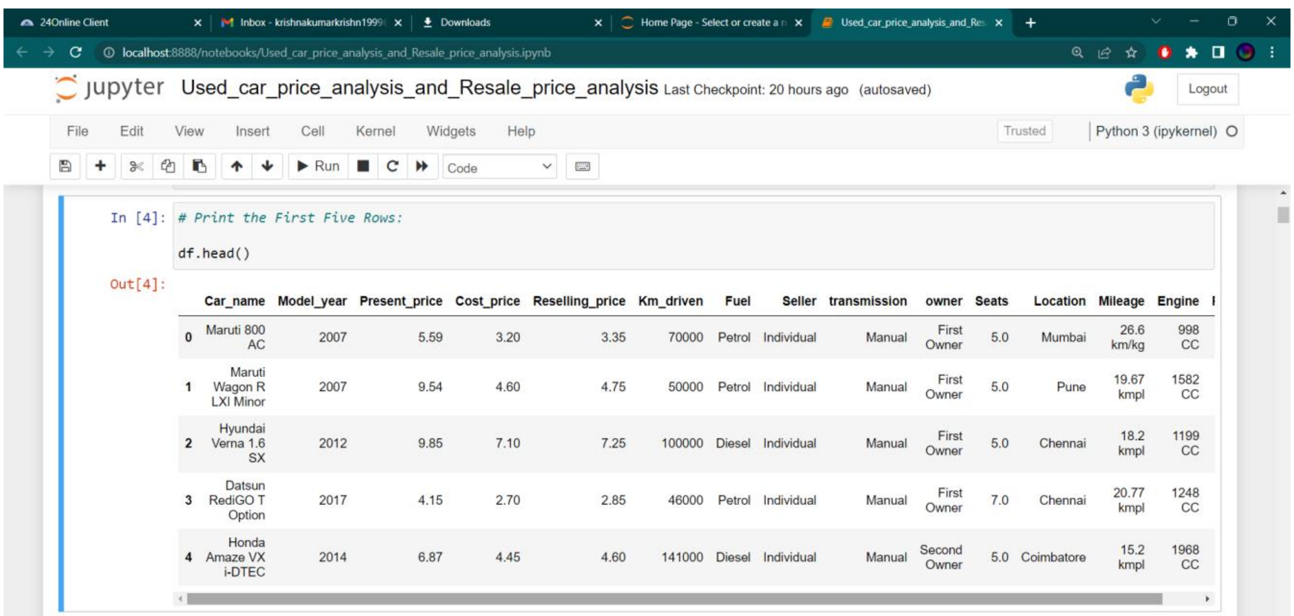
import seaborn as sns

Inporting the data set in read mode:

df=pd.read_csv(r"D:\Classes\Trem - 3\Projects\Python\car_data.csv")

Print the First Five Rows:

df.head()



```
In [4]: # Print the First Five Rows:

df.head()
```

Out[4]:

	Car_name	Model_year	Present_price	Cost_price	Reselling_price	Km_driven	Fuel	Seller	transmission	owner	Seats	Location	Mileage	Engine
0	Maruti 800 AC	2007	5.59	3.20	3.35	70000	Petrol	Individual	Manual	First Owner	5.0	Mumbai	26.6 km/kg	998 CC
1	Maruti Wagon R LXI Minor	2007	9.54	4.60	4.75	50000	Petrol	Individual	Manual	First Owner	5.0	Pune	19.67 kmpl	1582 CC
2	Hyundai Verna 1.6 SX	2012	9.85	7.10	7.25	100000	Diesel	Individual	Manual	First Owner	5.0	Chennai	18.2 kmpl	1199 CC
3	Datsun RediGO T Option	2017	4.15	2.70	2.85	46000	Petrol	Individual	Manual	First Owner	7.0	Chennai	20.77 kmpl	1248 CC
4	Honda Amaze VX i-DTEC	2014	6.87	4.45	4.60	141000	Diesel	Individual	Manual	Second Owner	5.0	Coimbatore	15.2 kmpl	1968 CC

Printing the hole number of rows and columns:

print("---------------------------------------")

print("Show the number of columns",df.shape[1])

print("Show the number of Rows",df.shape[0])

print("---------------------------------------")

```
---------------------------------------
Show the number of columns 15
Show the number of Rows 4340
---------------------------------------
```

Showing the hole information in this dataset:

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Car_name         4340 non-null   object
 1   Model_year       4340 non-null   int64
 2   Present_price    4340 non-null   float64
 3   Cost_price       4340 non-null   float64
 4   Reselling_price  4340 non-null   float64
 5   Km_driven        4340 non-null   int64
 6   Fuel             4340 non-null   object
 7   Seller           4340 non-null   object
 8   transmission     4340 non-null   object
 9   owner            4340 non-null   object
 10  Seats            4328 non-null   float64
 11  Location         4340 non-null   object
 12  Mileage          4340 non-null   object
 13  Engine           4331 non-null   object
 14  Power            4330 non-null   object
dtypes: float64(4), int64(2), object(9)
memory usage: 508.7+ KB
```

Now we Converting Object data type to Category data type:

```
df["Mileage"] = df["Mileage"].astype(str).str.rstrip(" kmpl")

df["Mileage"] = df["Mileage"].astype(str).str.rstrip(" km/g")

df["Engine"] = df["Engine"].astype(str).str.rstrip(" CC")

df["Power"] = df["Power"].astype(str).str.rstrip(" bhp")

df["Power"]= df["Power"].replace(regex="null", value = np.nan)


df["Fuel"]=df["Fuel"].astype("category")

df["transmission"]=df["transmission"].astype("category")
```

df["owner"]=df["owner"].astype("category")

df["Mileage"]=df["Mileage"].astype("float")

df["Power"]=df["Power"].astype("float")

df["Engine"]=df["Engine"].astype("float")

Again Showing the hole information in this dataset After the conversion:

df.head()

Out[82]:

ar_name	Model_year	Present_price	Cost_price	Reselling_price	Km_driven	Fuel	Seller	transmission	owner	Seats	Location	Mileage	Engine	Power
laruti 800 AC	2007	5.59	3.20	3.35	70000	Petrol	Individual	Manual	First Owner	5.0	Mumbai	26.60	998.0	58.16
Maruti Wagon R .XI Minor	2007	9.54	4.60	4.75	50000	Petrol	Individual	Manual	First Owner	5.0	Pune	19.67	1582.0	126.20
Hyundai /erna 1.6 SX	2012	9.85	7.10	7.25	100000	Diesel	Individual	Manual	First Owner	5.0	Chennai	18.20	1199.0	88.70
Datsun RediGO T Option	2017	4.15	2.70	2.85	46000	Petrol	Individual	Manual	First Owner	7.0	Chennai	20.77	1248.0	88.76
Honda maze VX i-DTEC	2014	6.87	4.45	4.60	141000	Diesel	Individual	Manual	Second Owner	5.0	Coimbatore	15.20	1968.0	140.80

Describe the data Show Statics:

df.describe(include="object").style.set_properties(**{"background-color":"red","color":"black"})

Out[83]:

	Car_name	Seller	Location
count	4340	4340	4340
unique	1491	3	11
top	Maruti Swift Dzire VDI	Individual	Mumbai
freq	69	3244	582

Check null missing values in this dataset:

df.isnull().sum()

```
Out[10]: Car_name           0
         Model_year         0
         Present_price      0
         Cost_price         0
         Reselling_price    0
         Km_driven          0
         Fuel               0
         Seller             0
         transmission       0
         owner              0
         Seats             12
         Location           0
         Mileage            0
         Engine             9
         Power             90
         dtype: int64
```

Drop Nan values Column Power:

df.drop(columns="Power",inplace=True)

checking the Seats missing values:

df[df.Seats.isnull()]

Out[14]:

	Car_name	Model_year	Present_price	Cost_price	Reselling_price	Km_driven	Fuel	Seller	transmission	owner	Seats	Location	Mileage	Engine
208	Renault Scala RxL	2013	8.10	7.75	7.90	55000	Petrol	Dealer	Manual	First Owner	NaN	Kolkata	16.10	1365.0
1385	Hyundai Elite i20 Asta Option CVT BSIV	2019	6.60	6.20	6.35	3000	Petrol	Individual	Automatic	First Owner	NaN	Pune	0.00	1198.0
1460	Tata Manza Aura Quadrajet BS IV	2012	4.90	4.50	4.65	35000	Diesel	Individual	Manual	Second Owner	NaN	Coimbatore	0.00	1249.0
2074	Tata Tiago 1.05 Revotorq XT Option	2017	8.93	8.53	8.68	35000	Diesel	Individual	Manual	First Owner	NaN	Pune	16.10	1497.0
2096	Maruti Omni MPI STD BSIV	2012	25.39	24.99	25.14	90000	Petrol	Individual	Manual	First Owner	NaN	Coimbatore	0.00	2997.0
2325	Toyota Corolla Altis 1.8 VL AT	2010	5.20	4.80	4.95	80000	Petrol	Individual	Automatic	Third Owner	NaN	Pune	16.10	1499.0
2335	Maruti Swift VXI	2013	5.00	4.60	4.75	90000	Petrol	Individual	Manual	First Owner	NaN	Mumbai	16.10	NaN

2335	Maruti Swift VXI	2013	5.00	4.60	4.75	90000	Petrol	Individual	Manual	First Owner	NaN	Mumbai	16.10	NaN
2369	Maruti Alto K10 LXI	2010	8.10	7.70	7.85	64000	Petrol	Dealer	Manual	First Owner	NaN	Chennai	19.50	1061.0
2780	Maruti Esteem Lxi - BSIII	2006	5.90	5.50	5.65	90000	Petrol	Individual	Manual	First Owner	NaN	Pune	0.00	1799.0
3272	Chevrolet Sail Hatchback 1.3 TCDi	2015	10.00	9.60	9.75	40000	Diesel	Individual	Manual	First Owner	NaN	Mumbai	18.48	NaN
3404	Hyundai Santro LE zipPlus	2003	5.60	5.20	5.35	50000	Petrol	Individual	Manual	Fourth & Above Owner	NaN	Jaipur	16.10	NaN
3810	Nissan Terrano XL Plus 85 PS	2015	6.90	6.50	6.65	55000	Diesel	Dealer	Manual	First Owner	NaN	Kolkata	14.00	NaN

mode=df.Seats.mode()

mode

```
Out[15]: 0    5.0
         Name: Seats, dtype: float64
```

 # Fill nan value in Seats Column with Mode:

df["Seats"].fillna(value=mode[0],inplace=True)

 # Checking the null data set :

df.isnull().sum()

```
In [18]: df.isnull().sum()

Out[18]: Car_name           0
         Model_year         0
         Present_price      0
         Cost_price         0
         Reselling_price    0
         Km_driven          0
         Fuel               0
         Seller             0
         transmission       0
         owner              0
         Seats              0
         Location           0
         Mileage            0
         Engine             9
         dtype: int64
```

 # Show Nan value in Engine Column:

df[df.Engine.isnull()]

Out[20]:

	Car_name	Model_year	Present_price	Cost_price	Reselling_price	Km_driven	Fuel	Seller	transmission	owner	Seats	Location	Mileage	Engine
2335	Maruti Swift VXI	2013	5.00	4.60	4.75	90000	Petrol	Individual	Manual	First Owner	5.0	Mumbai	16.10	NaN
3272	Chevrolet Sail Hatchback 1.3 TCDi	2015	10.00	9.60	9.75	40000	Diesel	Individual	Manual	First Owner	5.0	Mumbai	18.48	NaN
3404	Hyundai Santro LE zipPlus	2003	5.60	5.20	5.35	50000	Petrol	Individual	Manual	Fourth & Above Owner	5.0	Jaipur	16.10	NaN
3520	Tata Tiago 1.2 Revotron XT	2018	3.95	3.55	3.70	80000	Petrol	Individual	Manual	First Owner	5.0	Delhi	18.48	NaN
3522	Skoda Laura Ambiente 2.0 TDI CR MT	2010	8.01	7.61	7.76	100000	Diesel	Individual	Manual	Second Owner	5.0	Kochi	0.00	NaN
3810	Nissan Terrano XL Plus 85 PS	2015	6.90	6.50	6.65	55000	Diesel	Dealer	Manual	First Owner	5.0	Kolkata	14.00	NaN
4011	Maruti Alto LX BSIII	2008	10.38	9.98	10.13	120000	Petrol	Individual	Manual	Second Owner	5.0	Pune	20.30	NaN
4152	Hyundai Accent Executive CNG	2010	0.99	0.59	0.74	110000	CNG	Individual	Manual	First Owner	5.0	Mumbai	0.00	NaN
4229	Tata Indica Vista TDI LX	2015	6.79	6.39	6.54	50000	Diesel	Individual	Manual	Second Owner	7.0	Bangalore	17.00	NaN

#Drop nan Column:

new_data=df.dropna(axis=0)

#Show the number of rows per column:

new_data.count()

```
Out[22]: Car_name           4331
         Model_year         4331
         Present_price      4331
         Cost_price         4331
         Reselling_price    4331
         Km_driven          4331
         Fuel               4331
         Seller             4331
         transmission       4331
         owner              4331
         Seats              4331
         Location           4331
         Mileage            4331
         Engine             4331
         dtype: int64
```

Showing the expensive Cars

new_data[new_data["Reselling_price"]>90]

Out[21]:

	Car_name	Model_year	Present_price	Cost_price	Reselling_price	Km_driven	Fuel	Seller	transmission	owner	Seats	Location	Mileage	Engin
586	Hyundai Santro GS	2005	92.6	92.2	92.35	56580	Petrol	Dealer	Manual	First Owner	7.0	Kochi	11.33	4134.
1086	Renault Pulse RxZ	2017	92.6	92.2	92.35	22000	Diesel	Dealer	Manual	First Owner	5.0	Kochi	20.36	1197.
1586	Honda City i DTEC S	2014	92.6	92.2	92.35	90000	Diesel	Individual	Manual	Second Owner	5.0	Coimbatore	21.10	814.
2086	Mahindra Quanto C8	2013	92.6	92.2	92.35	82082	Diesel	Dealer	Manual	First Owner	5.0	Coimbatore	18.90	1197.
2586	Toyota Innova 2.5 G (Diesel) 7 Seater	2014	92.6	92.2	92.35	90000	Diesel	Individual	Manual	Second Owner	5.0	Coimbatore	17.11	1968.
3086	Maruti Ertiga VDI	2013	92.6	92.2	92.35	80000	Diesel	Individual	Manual	First Owner	8.0	Coimbatore	18.20	1248.
3586	Ford Figo Diesel Titanium	2012	92.6	92.2	92.35	63700	Diesel	Individual	Manual	First Owner	5.0	Delhi	15.10	1196.
4086	Mahindra Scorpio 2.6 Turbo 7 Str	2008	92.6	92.2	92.35	120000	Diesel	Individual	Manual	Second Owner	7.0	Pune	12.99	2494.

Only Eight cars in this dataset so expensive

Create a new variable “Age_of_Car”

```
new_data["Current_Year"]=2022

new_data["Age_of_car"]=new_data["Current_Year"]-new_data["Model_year"]

new_data.drop("Current_Year",axis=1,inplace=True)

new_data
```

Out[22]:

ent_price	Cost_price	Reselling_price	Km_driven	Fuel	Seller	transmission	owner	Seats	Location	Mileage	Engine	Age_of_car	Company	Model
5.59	3.20	3.35	70000	Petrol	Individual	Manual	First Owner	5.0	Mumbai	26.60	998.0	15	Maruti	800AC
9.54	4.60	4.75	50000	Petrol	Individual	Manual	First Owner	5.0	Pune	19.67	1582.0	15	Maruti	WagonR
9.85	7.10	7.25	100000	Diesel	Individual	Manual	First Owner	5.0	Chennai	18.20	1199.0	10	Hyundai	Verna1.6
4.15	2.70	2.85	46000	Petrol	Individual	Manual	First Owner	7.0	Chennai	20.77	1248.0	5	Datsun	RediGOT
6.87	4.45	4.60	141000	Diesel	Individual	Manual	Second Owner	5.0	Coimbatore	15.20	1968.0	8	Honda	AmazeVX
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5.00	4.60	4.75	80000	Diesel	Individual	Manual	Second Owner	5.0	Kochi	20.54	1598.0	8	Hyundai	i20Magna
6.00	5.60	5.75	80000	Diesel	Individual	Manual	Second Owner	5.0	Bangalore	28.09	1248.0	8	Hyundai	i20Magna
2.30	1.90	2.05	83000	Petrol	Individual	Manual	Second Owner	5.0	Kochi	14.40	1598.0	13	Maruti	800AC
4.40	4.00	4.15	90000	Diesel	Individual	Manual	First Owner	5.0	Bangalore	18.50	1197.0	6	Hyundai	Creta1.6
4.60	4.20	4.35	40000	Petrol	Individual	Manual	First Owner	5.0	Pune	16.50	1198.0	6	Renault	KWIDRXT

 # Again, Create a new columns Company and Model:

#column Company and Model:

new_data["Company"]=new_data["Car_name"].str.split(" ").str[0]

new_data["Model"]=new_data["Car_name"].str.split(" ").str[1]+new_data["Car_name"].str.split(" ").str[2]

 # Now Start Data Visualization:

Let's show the Company Name and Total cars:

```
company=pd.DataFrame(new_data["Company"].value_counts().sort_values(ascending=False).reset_index().rename(columns={"index":"Company","Company":"Total_Cars"}))

fig = ff.create_table(company, index=True)

for i in range(len(fig.layout.annotations)):
    fig.layout.annotations[i].font.size =15

fig.show()
```

	Company	Total_Cars
0	Maruti	1278
1	Hyundai	819
2	Mahindra	365
3	Tata	359
4	Honda	252
5	Ford	238
6	Toyota	206
7	Chevrolet	187
8	Renault	146
9	Volkswagen	107
10	Skoda	67
11	Nissan	63
12	Audi	60
13	BMW	39
14	Datsun	37
15	Fiat	37
16	Mercedes-Benz	35
17	Jaguar	6
18	Mitsubishi	6
19	Land	5
20	Volvo	4
21	Ambassador	4
22	Jeep	3
23	MG	2
24	OpelCorsa	2
25	Daewoo	1
26	Force	1
27	Isuzu	1
28	Kia	1

Most Expensive Cars Company:

maximum=new_data[["Company","Reselling_price"]][new_data.Reselling_price==new_data["Reselling_price"].max()]

maximum

Out[26]:

	Company	Reselling_price
586	Hyundai	92.35
1086	Renault	92.35
1586	Honda	92.35
2086	Mahindra	92.35
2586	Toyota	92.35
3086	Maruti	92.35
3586	Ford	92.35
4086	Mahindra	92.35

Show Top 20 Most Expensive Company:

n=new_data[["Reselling_price","Company"]].sort_values(by="Reselling_price",ascending=False).head(20)

con=n["Company"]

p=n["Reselling_price"]

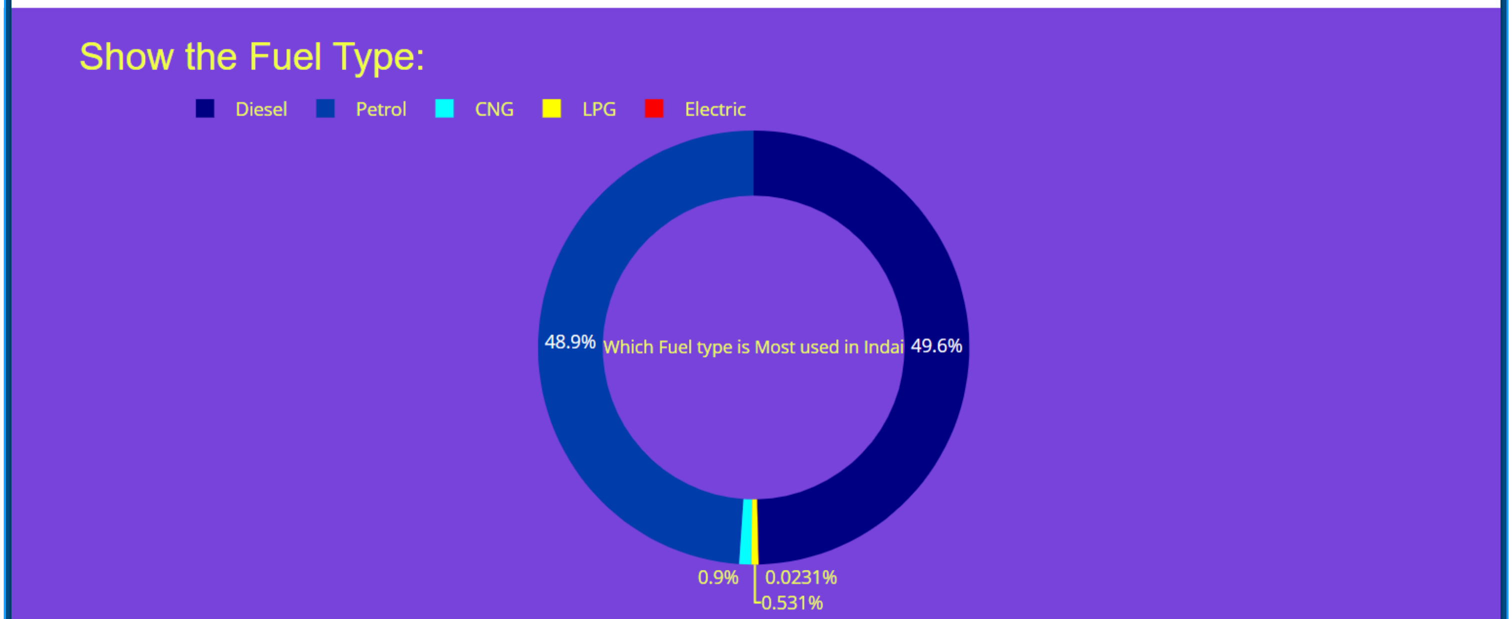
```
go_fig = go.Figure()

obj = go.Table(
    header = dict(values=["Company", "Reselling_price"],
                  fill_color = 'red',
                  align = 'left',
                  font=dict(color="white", size = 15)),
    cells = dict(values=[con, p],
                 fill_color = 'yellow',
                 align = 'left',
                 font=dict(color="#0D4C92", size = 15)))
go_fig.add_trace(obj)
go_fig.show()
```

Company	Reselling_price
Toyota	92.35
Mahindra	92.35
Hyundai	92.35
Honda	92.35
Ford	92.35
Mahindra	92.35
Maruti	92.35
Renault	92.35
Tata	35.98
Maruti	35.98
Volkswagen	35.98
Maruti	35.98
Honda	35.98
Renault	35.98
Honda	35.98
Ford	35.98
Maruti	35.71
Chevrolet	35.71
Hyundai	35.71
Tata	35.71

Fuel Type is Most Used:

```
fuel_type=pd.DataFrame(new_data["Fuel"].value_counts().reset_index().rename(columns={"index":"Fuel","Fuel":"Total"}))

fig=go.Figure(data=[go.Pie(labels=fuel_type["Fuel"],
                           values=fuel_type["Total"],
                           hole=.7,
                           title="Which Fuel type is Most used in Indai",
                           marker_colors=px.colors.sequential.Jet,)])
fig.update_layout(title="Show the Fuel Type:")

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total ascending', ticksuffix=' ', showline=False)

fig.update_traces(hovertemplate=None, marker=dict(line=dict(width=0)))

fig.update_layout(margin=dict(t=80, b=0, l=70, r=40),hovermode="y unified",
```

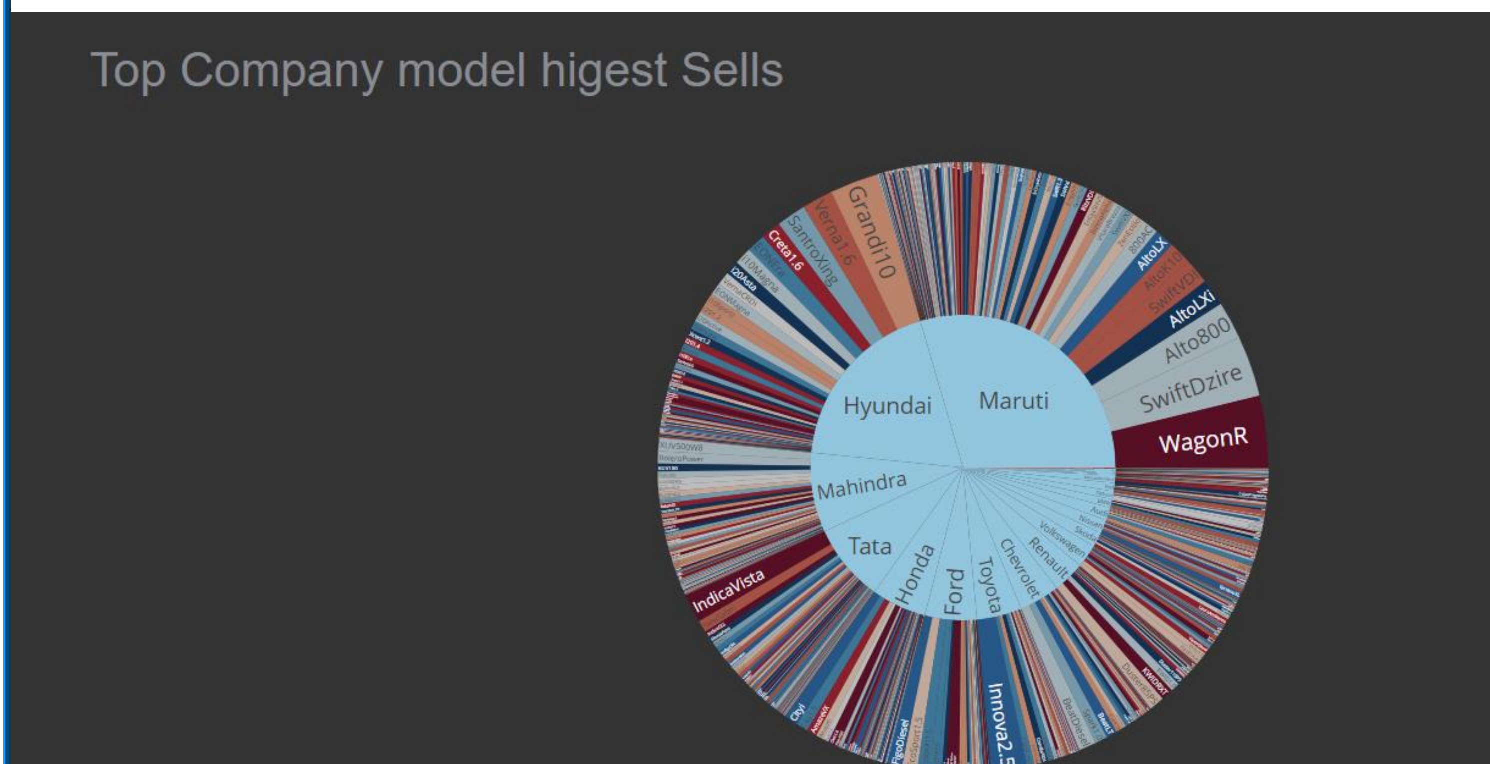
```
                xaxis_title=' ', yaxis_title=" ",
height=400,plot_bgcolor='#7743DB', paper_bgcolor='#7743DB',
title_font=dict(size=25, color='#F0FF42', family="Lato, sans-
serif"),
 font=dict(color='#F0FF42'),
                legend=dict(orientation="h", yanchor="bottom", y=1,
xanchor="right", x=0.5),
                hoverlabel=dict(bgcolor="black", font_size=13,
font_family="Lato, sans-serif"))
 fig.show()
```



Which Company Model is highest Sell in India:

```
company=pd.DataFrame(new_data[["Company","Model"]].value_counts().sort_values(ascending=False).reset_index().rename(columns={0:"Total"}))



fig=px.sunburst(company,path=["Company","Model"],values="Total",color="Model",

color_discrete_sequence=px.colors.sequential.RdBu,title="Top 10 Company model higest Sells")

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total ascending', ticksuffix=' ', showline=False)

fig.update_traces(hovertemplate=None, marker=dict(line=dict(width=0)))

fig.update_layout(margin=dict(t=80, b=0, l=70, r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ", height=400,plot_bgcolor='#333', paper_bgcolor='#333',
```

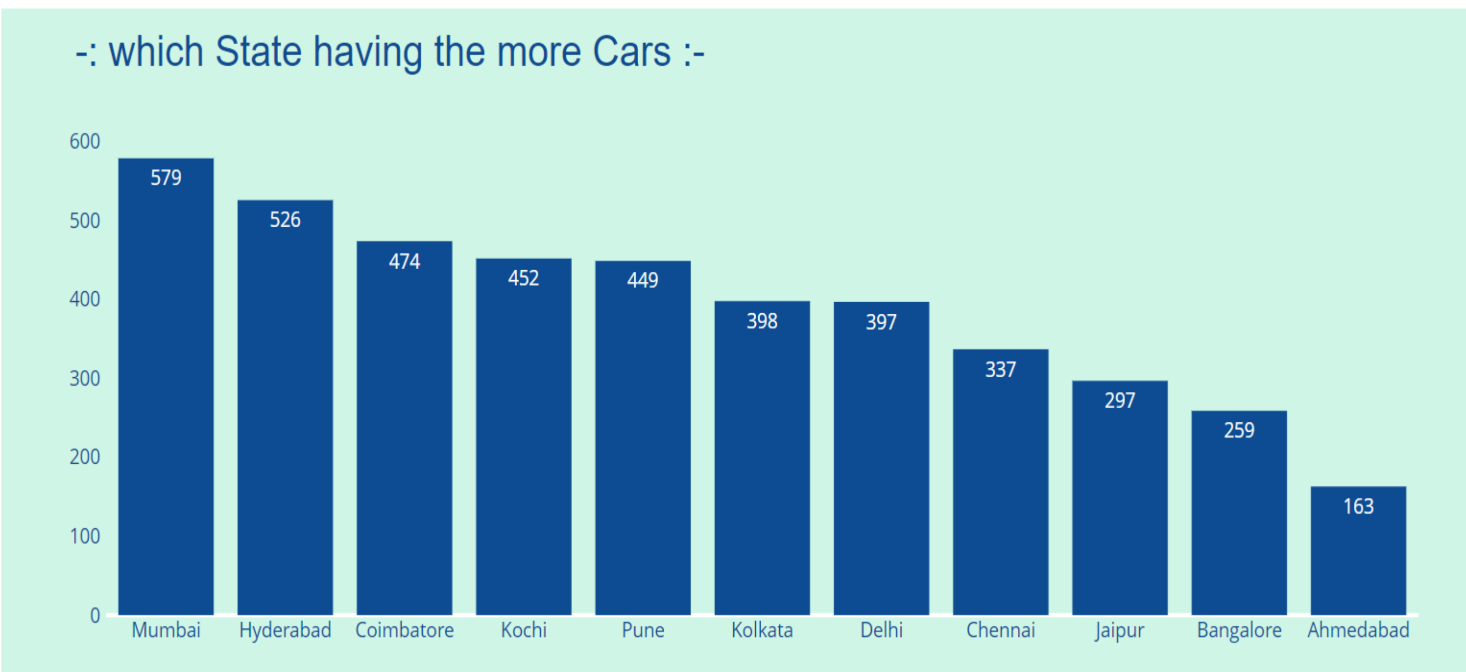
```
title_font=dict(size=25, color='#8a8d93', family="Lato, sans-serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom", y=1, xanchor="right", x=0.5),
                  hoverlabel=dict(bgcolor="black", font_size=13, font_family="Lato, sans-serif"))


fig.show()
```



Cars According to States:

```
location=pd.DataFrame(new_data["Location"].value_counts().sort_values(ascending=False).reset_index().rename(columns={"index":"Location","Location":"Count"}))

fig=px.bar(location,x="Location",y="Count",title="-: which State having the more Cars :-",
        color_discrete_sequence=['#0D4C92'],text="Count")

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total ascending', ticksuffix=' ', showline=False)

fig.update_traces(hovertemplate=None, marker=dict(line=dict(width=0)))

fig.update_layout(margin=dict(t=80, b=0, l=70, r=40),hovermode="y unified",
                xaxis_title=' ', yaxis_title=" ", height=400,plot_bgcolor='#CFF5E7', paper_bgcolor='#CFF5E7',
```

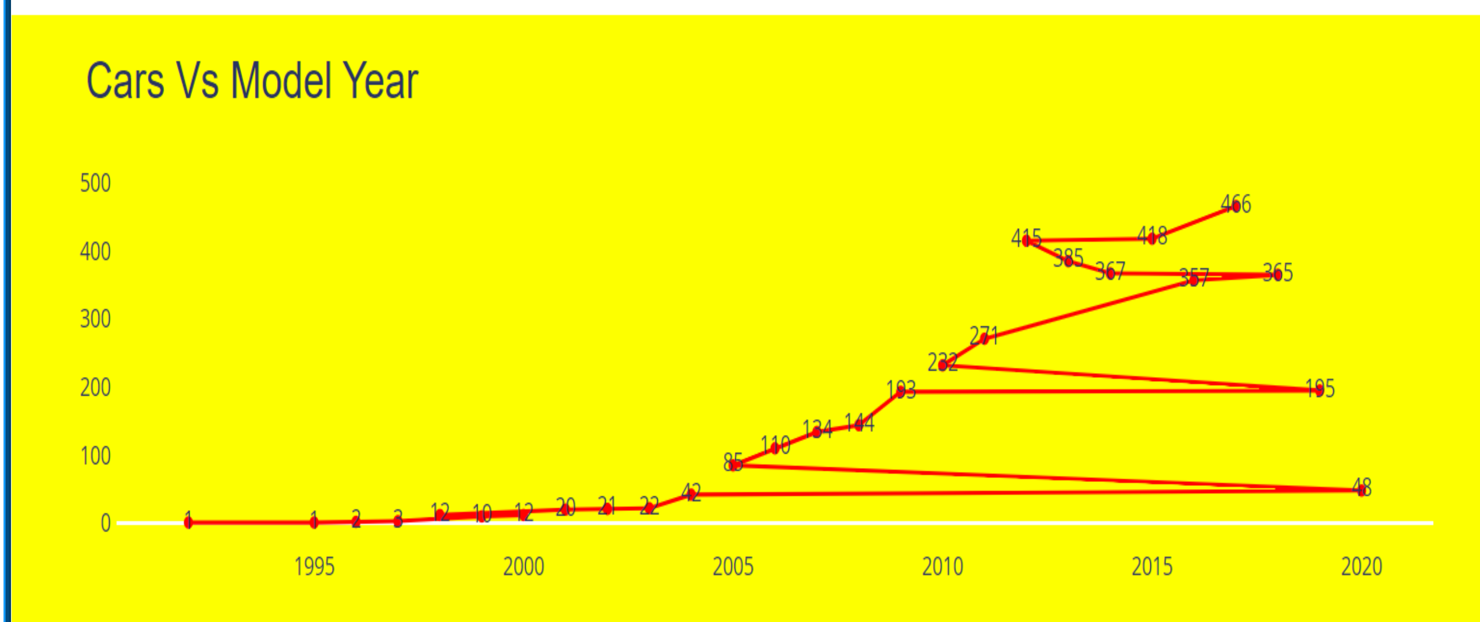
```
title_font=dict(size=25, color='#0D4C92', family="Lato, sans-serif"),
 font=dict(color='#0D4C92'),
                legend=dict(orientation="h", yanchor="bottom", y=1, xanchor="right", x=0.5),
                hoverlabel=dict(bgcolor="#FFE1E1", font_size=15, font_family="Lato, sans-serif"))


fig.show()
```



In which year the most cars have been used

```
year=pd.DataFrame(new_data["Model_year"].value_counts().sort_values(ascending=False).reset_index().rename(columns={"index":"Model_year","Model_year":"Count"}))



fig = px.line(year, x='Model_year', y="Count",markers=True,text="Count",title="Cars Vs Model Year",color_discrete_sequence=["red"])

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total ascending', ticksuffix=' ', showline=False)

fig.update_traces(hovertemplate=None, marker=dict(line=dict(width=0)))

fig.update_layout(margin=dict(t=80, b=0, l=70, r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ", height=300,plot_bgcolor='#FDFF00', paper_bgcolor='#FDFF00',
```

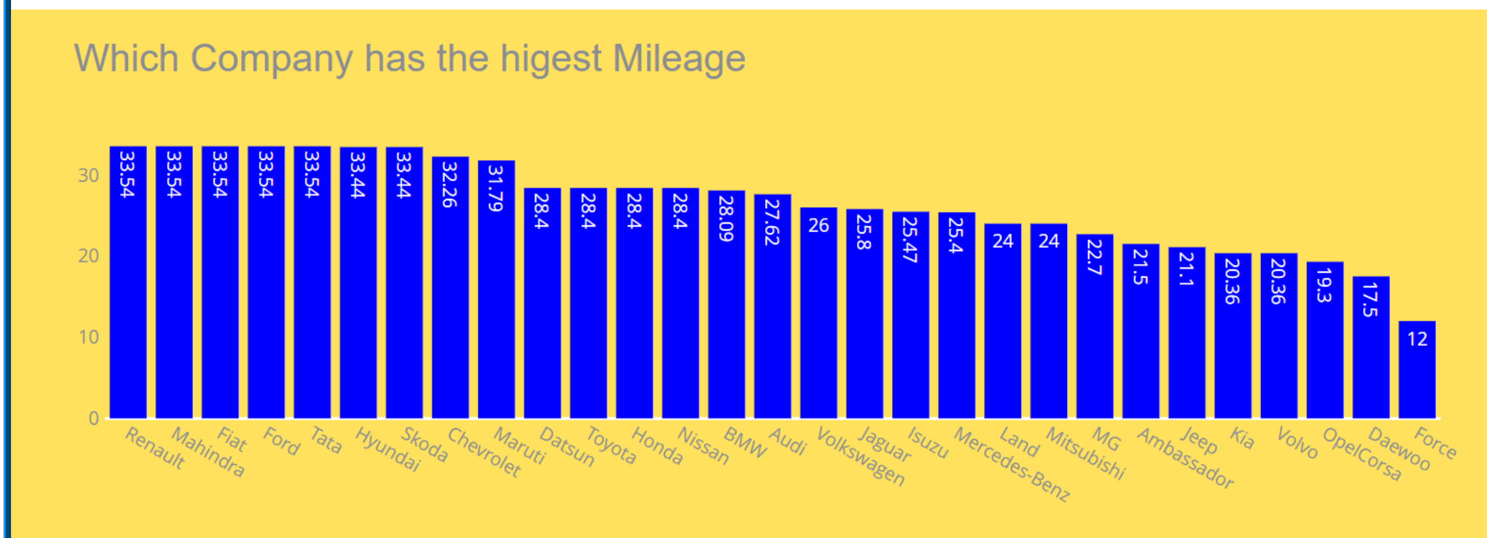
```
title_font=dict(size=25, color='#293462', family="Lato, sans-serif"),
 font=dict(color='#293462'),
                 legend=dict(orientation="h", yanchor="bottom", y=1, xanchor="right", x=0.5),
                 hoverlabel=dict(bgcolor="#1CD6CE", font_size=15, font_family="Lato, sans-serif"))

fig.show()
```



Show the higest Mileage of Each Company:

```
mileage=new_data.groupby(["Company"])["Mileage"].max().sort
_values(ascending=False).reset_index()


fig=px.bar(mileage,x="Company",y="Mileage",title="Which
Company has the higest
Mileage",text="Mileage",color_discrete_sequence=["blue"])
fig.update_xaxes(showgrid=False)


fig.update_yaxes(showgrid=False, categoryorder='total
ascending', ticksuffix=' ', showline=False)


fig.update_traces(hovertemplate=None,
marker=dict(line=dict(width=0)))


fig.update_layout(margin=dict(t=80, b=0, l=70,
r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ",
height=350,plot_bgcolor='#FFE15D', paper_bgcolor='#FFE15D',
title_font=dict(size=25, color='#8a8d93', family="Lato, sans-
serif"),
 font=dict(color='#8a8d93'),
```

```
                legend=dict(orientation="h", yanchor="bottom", y=1,
xanchor="right", x=0.5),
                hoverlabel=dict(bgcolor="#DEF5E5", font_size=13,
font_family="Lato, sans-serif"))

fig.show()
```

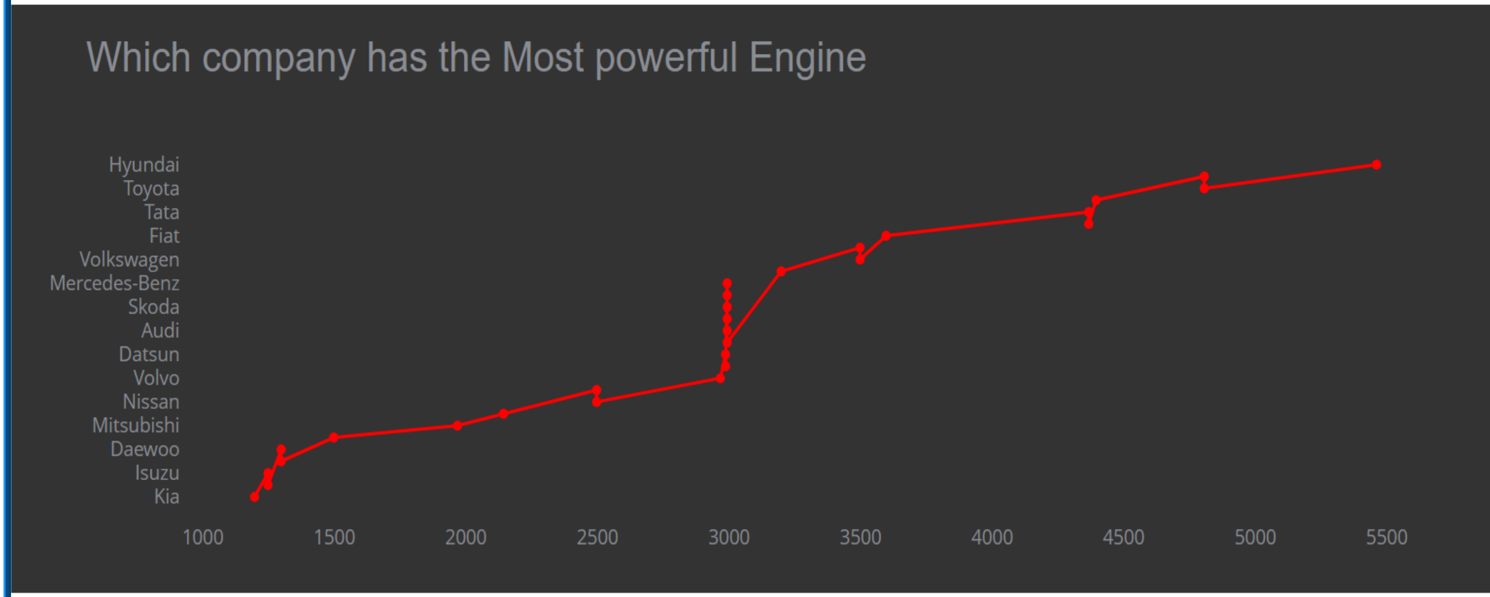


Which Company has the most Powerful Engine:

```
Engine=new_data.groupby(["Company"])["Engine"].max().sort_va
lues(ascending=False).reset_index()

fig=px.line(Engine,x="Engine",y="Company",title="Which
company has the Most powerful Engine",markers=True,
        color_discrete_sequence=["red"])
```

```
fig.update_xaxes(showgrid=False)


fig.update_yaxes(showgrid=False, categoryorder='total
ascending', ticksuffix=' ', showline=False)


fig.update_traces(hovertemplate=None,
marker=dict(line=dict(width=0)))


fig.update_layout(margin=dict(t=80, b=0, l=70,
r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ",
height=350,plot_bgcolor='#333', paper_bgcolor='#333',
title_font=dict(size=25, color='#8a8d93', family="Lato, sans-
serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom", y=1,
xanchor="right", x=0.5),
                  hoverlabel=dict(bgcolor="black", font_size=13,
font_family="Lato, sans-serif"))
```

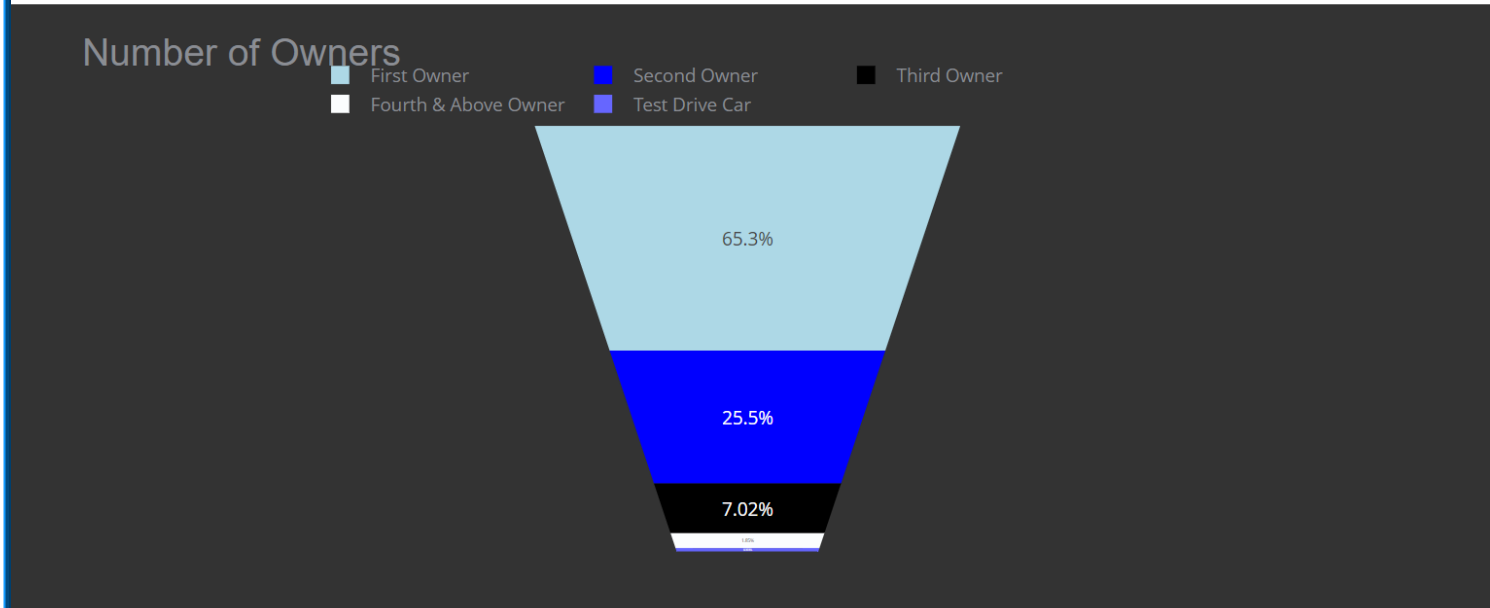
fig.show()



Let's check the Number of Owner's:

```
owner_type=pd.DataFrame(new_data["owner"].value_counts().sort_values(ascending=False).reset_index().rename(columns={"index":"Owner_type","owner":"Count"}))


fig = px.funnel_area(names=owner_type["Owner_type"],
                     values=owner_type["Count"],
                      title="Number of Owners",
                      color_discrete_sequence=["lightblue","blue",
"black"])
fig.update_xaxes(showgrid=False)
```

```
fig.update_yaxes(showgrid=False, categoryorder='total
ascending', ticksuffix=' ', showline=False)


fig.update_traces(hovertemplate=None,
marker=dict(line=dict(width=0)))


fig.update_layout(margin=dict(t=80, b=40, l=30,
r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ",
height=400,plot_bgcolor='#333', paper_bgcolor='#333',
title_font=dict(size=25, color='#8a8d93', family="Lato, sans-
serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom", y=1,
xanchor="right", x=0.5),
                  hoverlabel=dict(bgcolor="black", font_size=14,
font_family="Lato, sans-serif"))


 fig.show()
```

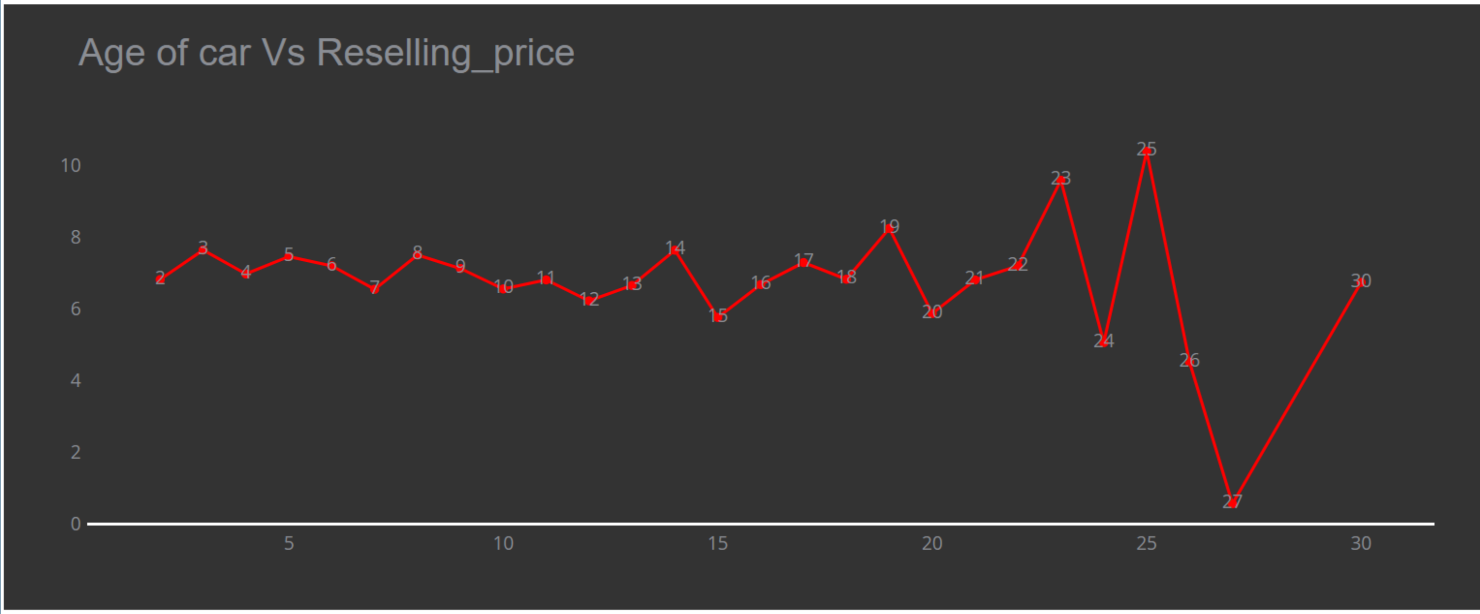


Show the Reselling According to Age of Car:

```
age=pd.pivot_table(new_data,index=["Age_of_car"],values=["Reselling_price"])

age=pd.DataFrame(age).sort_values(by="Age_of_car",ascending=False).reset_index()

fig=px.line(age,x="Age_of_car",y="Reselling_price",title="Age of car Vs Reselling_price",text="Age_of_car",
        color_discrete_sequence=["red"])

fig.update_xaxes(showgrid=False)
```

```
fig.update_yaxes(showgrid=False, categoryorder='total ascending', ticksuffix=' ', showline=False)


fig.update_traces(hovertemplate=None, marker=dict(line=dict(width=0)))


fig.update_layout(margin=dict(t=80, b=40, l=30, r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ", height=400,plot_bgcolor='#333', paper_bgcolor='#333',
title_font=dict(size=25, color='#8a8d93', family="Lato, sans-serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom", y=1, xanchor="right", x=0.5),
                  hoverlabel=dict(bgcolor="black", font_size=14, font_family="Lato, sans-serif"))
```

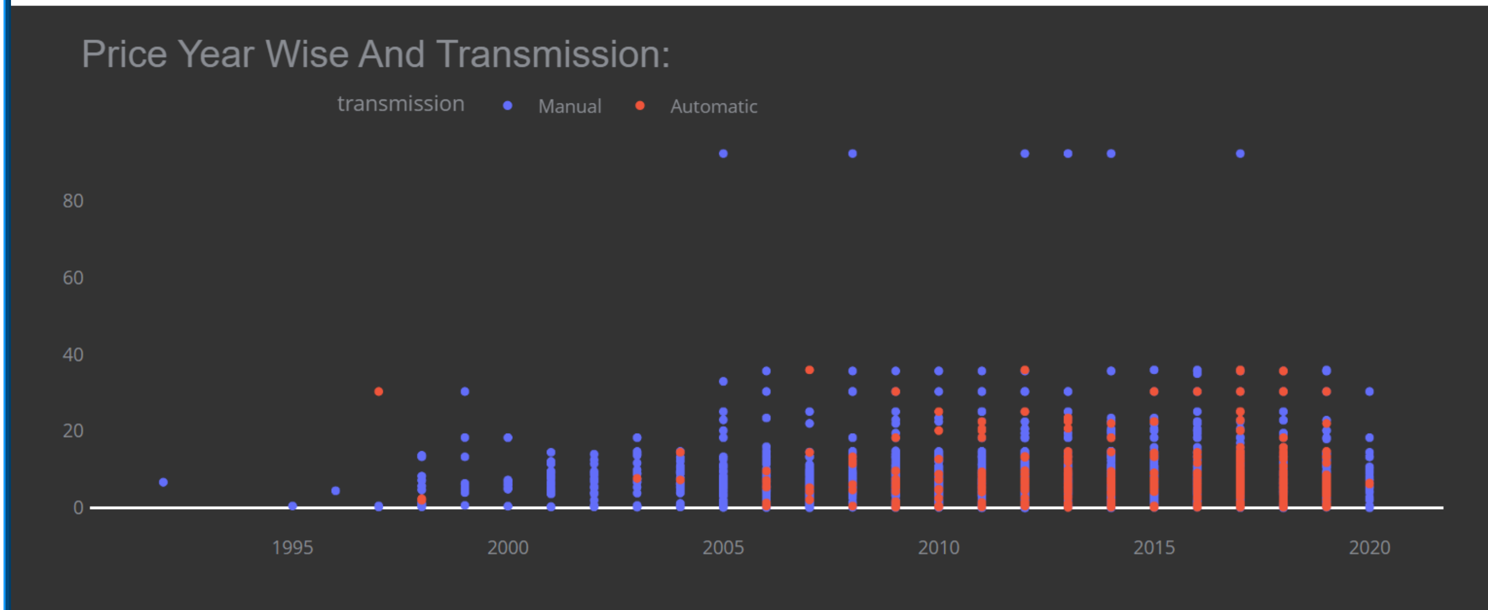
fig.show()



Let's show the Price according to Year and transmission Type:

```
fig=px.scatter(new_data,x="Model_year",y='Reselling_price'
,color="transmission",title="Price Year Wise And
Transmission:")

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total
ascending', ticksuffix=' ', showline=False)
```

```
fig.update_traces(hovertemplate=None,
marker=dict(line=dict(width=0)))


fig.update_layout(margin=dict(t=80, b=40, l=30,
r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ",
height=400,plot_bgcolor='#333', paper_bgcolor='#333',
title_font=dict(size=25, color='#8a8d93', family="Lato, sans-
serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom",
y=1, xanchor="right", x=0.5),
                  hoverlabel=dict(bgcolor="black", font_size=14,
font_family="Lato, sans-serif"))


fig.show()
```



Show the Price Location wise :

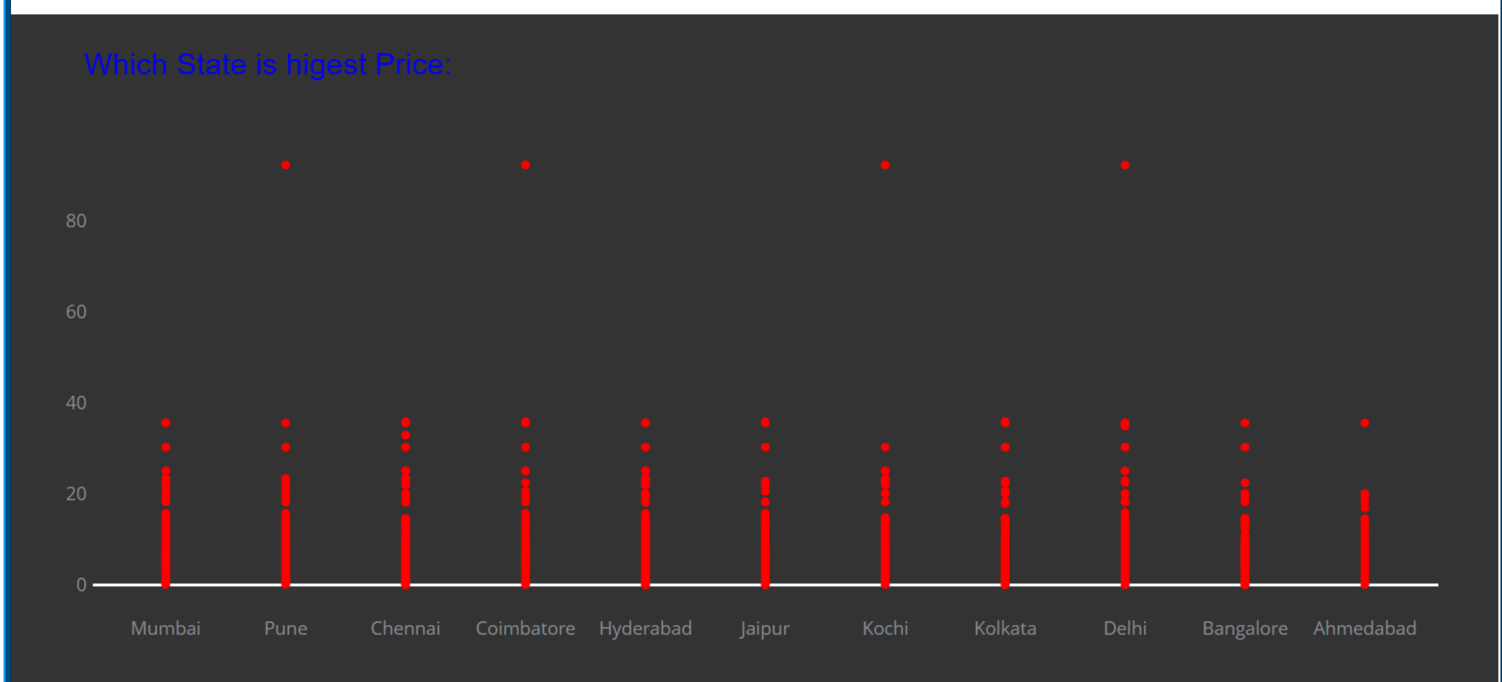
fig=px.scatter(new_data,x="Location",y="Reselling_price",title="Which State is higest Price:",color_discrete_sequence=["red"])

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total ascending', ticksuffix=' ', showline=False)

fig.update_traces(hovertemplate=None, marker=dict(line=dict(width=0)))

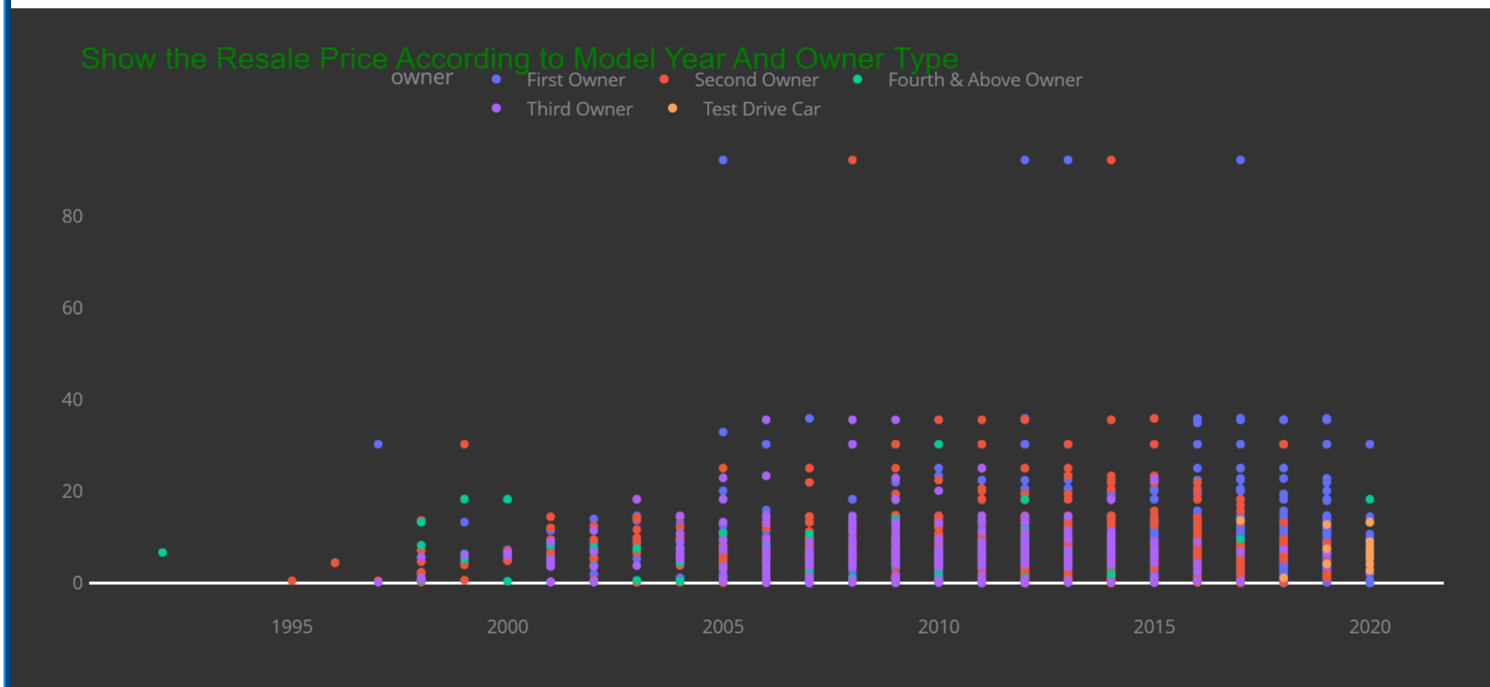
```
fig.update_layout(margin=dict(t=80, b=40, l=30, r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ", height=450,plot_bgcolor='#333', paper_bgcolor='#333',
title_font=dict(size=20, color='blue', family="Lato, sans-serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom", y=1, xanchor="right", x=0.5),
                  hoverlabel=dict(bgcolor="black", font_size=14, font_family="Lato, sans-serif"))



fig.show()
```



Price According to OwnerShip and Year:

```
fig=px.scatter(new_data,x="Model_year",y='Reselling_price'
,color="owner",title="Show the Resale Price According to
Model Year And Owner Type")

fig.update_xaxes(showgrid=False)

fig.update_yaxes(showgrid=False, categoryorder='total
ascending', ticksuffix=' ', showline=False)

fig.update_traces(hovertemplate=None,
marker=dict(line=dict(width=0)))

fig.update_layout(margin=dict(t=80, b=40, l=30,
r=40),hovermode="y unified",
                  xaxis_title=' ', yaxis_title=" ",
height=450,plot_bgcolor='#333', paper_bgcolor='#333',
title_font=dict(size=20, color='green', family="Lato, sans-
serif"),
 font=dict(color='#8a8d93'),
                  legend=dict(orientation="h", yanchor="bottom",
y=1, xanchor="right", x=0.5),
```

```
                hoverlabel=dict(bgcolor="black", font_size=14,
font_family="Lato, sans-serif"))


fig.show()
```



CONCLUSION

Used vehicle market involves many factors when it comes to predicting the fast-selling vehicles that maintain profit and reduce inventory cost for the retailers.

- The main aim of the project is to predict the price of second-hand reconditioned and second-hand used cars.
- The average residual value was reasonably low for all the approaches. Thus, we conclude that predicting the price of second-hand cars is a very risky enterprise, but which is feasible.
- This system will be very useful to car dealers and car owners who need to assess the value of their cars.
- In future research we can explore other factors that influence the sales period of a used vehicle. For example, the level of fuel efficiency, whether the vehicle is electric or hybrid, level of discount from the original price.
- Incorporating these factors in the analysis can improve the accuracy to choose non-overage vehicles and have a positive impact on profit.

Future Plans

- We will add more features to improve our project.
- There will be create mobile application.
- And also, creating a web application.
- We will add SSL security system.
- New product update newsletter will be added.
- SMS alert system is easier for the customer.
- We also work on online payment gateway integration.
- Additionally, it is just a beginning. Supplementary the system may be used in various other types of analysis process.
- Working on backend to connect the servers

REFERENCES

https://www.javatpoint.com/pandas-numpy

https://www.youtube.com/watch?v=QXeEoD0pB3E&list=PLsyeobzWxl7poL9JTVyndKe62ieoN-MZ3&ab_channel=Telusko

https://www.youtube.com/watch?v=ZyhVh-qRZPA&list=PL-osiE80TeTsWmV9i9c58mdDCSskIFdDS

https://www.youtube.com/watch?v=CmorAWRsCAw&list=PLeo1K3hjS3uuASpe-1LjfG5f14Bnozjwy

https://www.youtube.com/watch?v=JrIG6Qtb4eg&list=PL_1pt6K-CLoDMEbYy2PcZuITWEjqMfyoA