Used Car Price Prediction

Data Science With Python Lab Project Report

Bachelor

in

Computer Science

By

TANKALA ABISHEK AND KARUKOLA PUNEETH

S190430

S190901

Rajiv Gandhi University Of Knowledge And Technologies

S.M. Puram , Srikakulam -532410

Andhra Pradesh, India

Abstract

Used car price prediction

Now a days,the market of used cars is increasing more than the new cars.Therfore,the buyer must know the accurate selling price.We are working on the project of used cars and predicting the price based on the factors like mileage,fuel-type,no.of owners,year of manufacturing,horse power,no.of years used etc.We are aiming to get accurate results by using the machine learning and Data science.The Machine Learning models include Random Forest Model and Linear Regression.We are working on multiple datasets from the CAR DEKHO website and used cars from USA available in Kaggle.Our goal is to create a model that uses our dataset to accurately forecast the price of a used car.This project can benefit both sellers and buyers to get actual sale price of used car.

In this project,we use some of the python libraries such as pandas,numpy,matplotlib and scikit-learn etc.

Contents

Abstract **1**

1 Introduction **3**

1.1 Introduction to Your Project 3

1.2 Application 4

1.3 Motivation Towards Your Project 5

1.4 Problem Statement 5

2 Approach To Your Project **6**

2.1 Explain About Your Project 6

2.2 Data Set 6

2.3 Prediction technique 7

2.4 Graphs 7

2.5 Visualization 12

3 Code **21**

4 Conclusion and Future Work **33**

4.1 Conclusion 33

Chapter 1

Introduction

1.1 Introduction to Your Project

Our project focuses on Used Car Price Prediction, utilizing various parameters such as engine capacity, mileage, number of owners, and horse power. By employing data preprocessing techniques with pandas and visualizing insights through matplotlib and seaborn, we aim to develop an accurate model. To achieve this, we employ both Linear Regression and Random Forest Regression techniques. Throughout the project, we diligently worked on the dataset,removing outliers to enhance the accuracy of our predictions.

1.2 Application

Here are a few applications for a project based on used car price prediction:

Car Buyers and Sellers: Individuals looking to buy or sell used cars can benefit from price prediction models. It helps sellers set realistic prices based on the vehicle's characteristics, market trends, and historical data. Buyers, on the other hand, can utilize these predictions to evaluate whether the listed price is fair or negotiate effectively.

Online Platforms: Platforms that help in buying and selling of used cars, such as websites helps sellers set reasonable asking prices and assists buyers in evaluating the value of a particular car.

Automotive Industry Analysis: Used car price prediction models can provide valuable insights to analysts and researchers studying the automotive market. The data generated can be used to understand market trends, fluctuations in prices, and the impact of various factors on used car values.

1.3 Motivation Towards Your Project

Our motivation for this project came from the high market demand for used cars, surpassing that of new cars. Recognizing the potential issue of sellers increasing prices beyond reasonable levels, we aimed to address this concern by developing a solution that benefits both buyers and sellers. Our primary focus was to assist buyers in purchasing cars at fair and optimal prices while indirectly helping sellers in setting attractive prices that attract potential buyers.Driving factor behind this project was the opportunity to explore and understand the dependency between various parameters and the overall condition of the car, which significantly influence its price.

1.4 Problem Statement

Our Project is Used Car Price Prediction. The project aims to develop a machine learning model that can predict the price of a used car.The dataset for this project is taken from kaggle and it is webscraped from CAR DEKHO website.This project will utilize various features for predicting the price of a car such as Mileage,Kilometer driven,engine,Number of owners,Horsepower etc. By analyzing these features,the model should predict accurate price using better machine learning model.The dataset is splitted into training set and test set for model development.

Chapter 2

Approach To Your Project

2.1 Explain About Your Project

This Project is about to predict the price of a used car.This project can benefit both sellers and buyers to get actual price of used car.Buyers can relate actual price with selling price and can decide to buy it or not. This project will benefit sellers to put a optimum price to sell any particular car.Buyers can check upon features such as mileage,No. of Owners,Engine of a car for a better price.

2.2 Data Set

The Dataset for this Used car price prediction project is taken from Kaggle Website.Dataset contains various features such as year,kilometer driven,Number of owners,Horsepower,Engine,fuel,sellerType,transmission,Mileage etc.

year - year of selling

Kilometer driven - Number of kilometers driven

HorsePower - Horsepower of the car

Engine - Engine of the car

Mileage - Mileage of the car

Transmission - Transmission of the car either manual or automatic

fuel - fuel of the car either petrol,diesel,CNG etc

sellerType - seller type of the car either individual or Dealer

2.3 Prediction technique

Our prediction techinques are Linear Regression and RandomForestRegressor.Linear Regression is a model that shows the relation between dependent and independent variables.we select a suitable regression model for our prediction that is RandomForest Regressor.we use Random forest as it predicts output with high accuracy, even for the large dataset it runs efficiently.

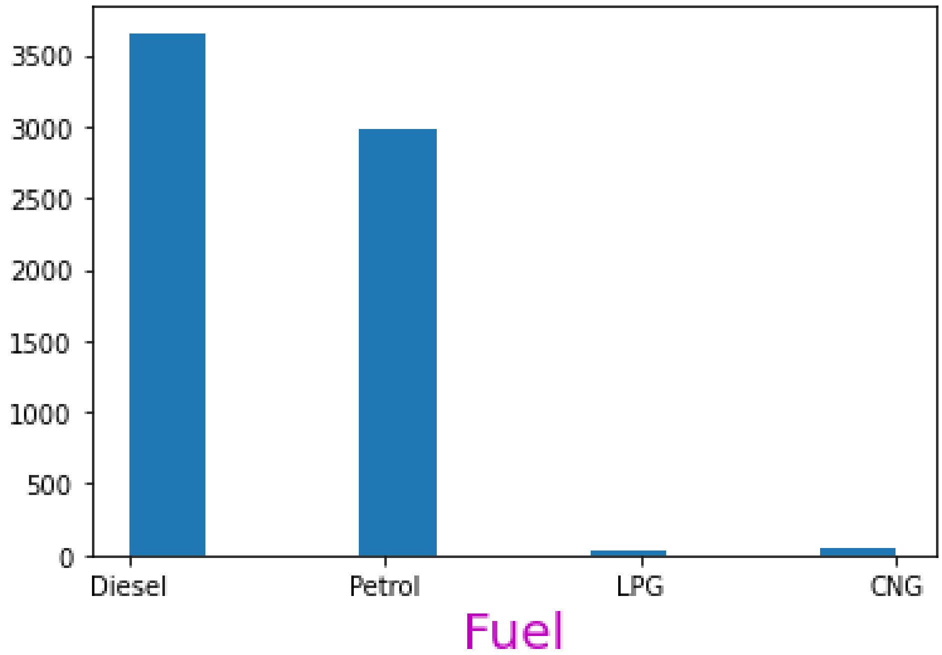
2.4 Graphs

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Histogram Plot For Fuel type

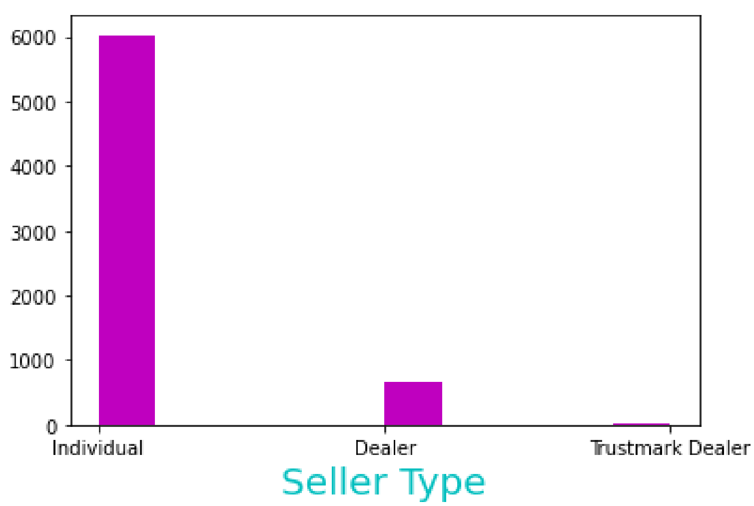
```
plt.hist(df['fuel'])
plt.xlabel('Fuel',color='m',fontsize=20)
```

```
plt.show()
```



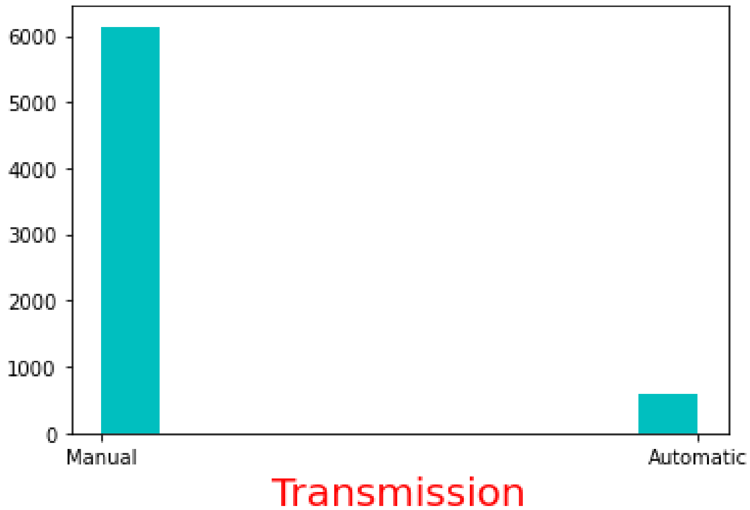
Histogram Plot For Seller type

```
plt.hist(df['seller_type'],color='m')

plt.xlabel('Seller Type',color='c',fontsize=20)

plt.show()
```



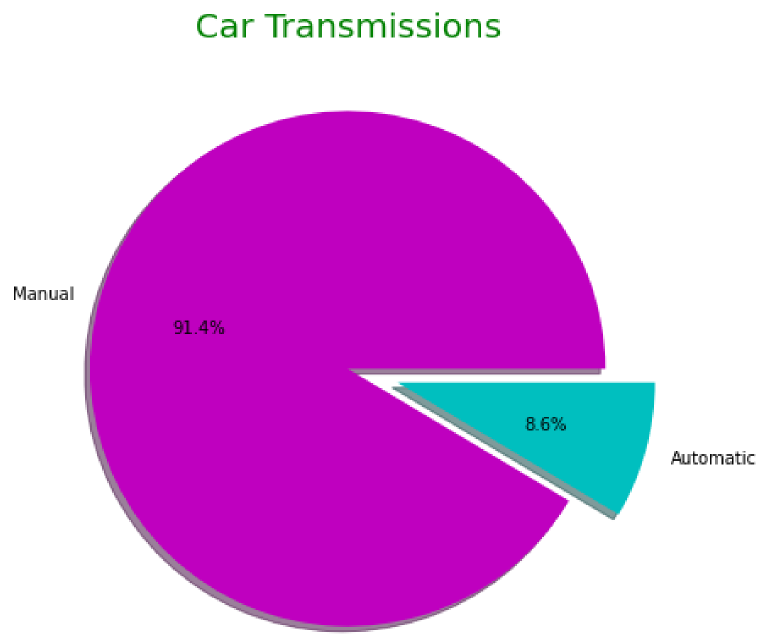
Histogram Plot For Transmission

```
plt.hist(df['transmission'],color='c')

plt.xlabel('Transmission',color='r',fontsize=20)

plt.show()
```



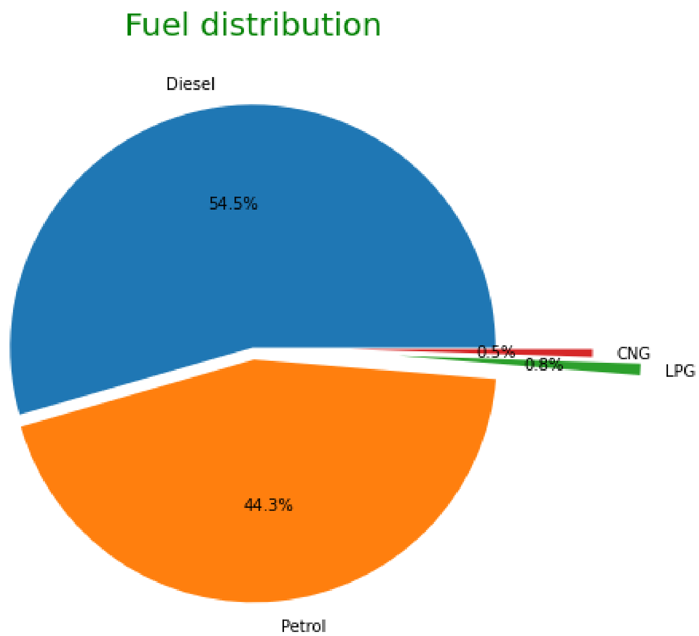
Pie Chart For Car Transmissions

```
data=df['transmission'].value_counts()

l=df['transmission'].unique()

plt.figure(figsize=(7,7))

c=['m','c']  #m-magenta c-cyan

e=[0,0.2]

#code

#code

plt.title('Car Transmissions',color='green',size=20)

plt.pie(data,labels=l,colors=c,explode=e,autopct='%1.1f%%',shadow=True)

plt.show()
```



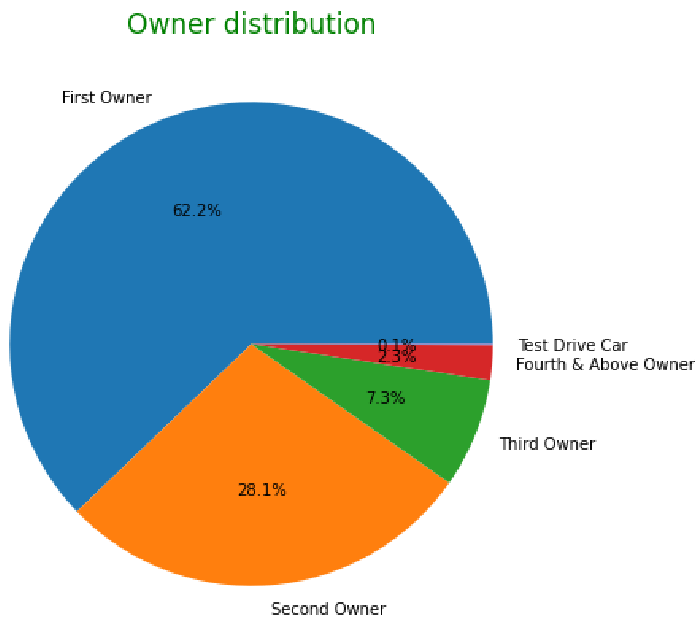
Pie Chart For Fuel Distribution

```
data=df['fuel'].value_counts()

l=df['fuel'].unique()


e=[0,0.05,0.6,0.4]

plt.figure(figsize=(7,7))


plt.title('Fuel distribution',size=20,color='green')


plt.pie(data,labels=l,shadow=False,autopct='%1.1f%%',explode=e)


plt.show()
```



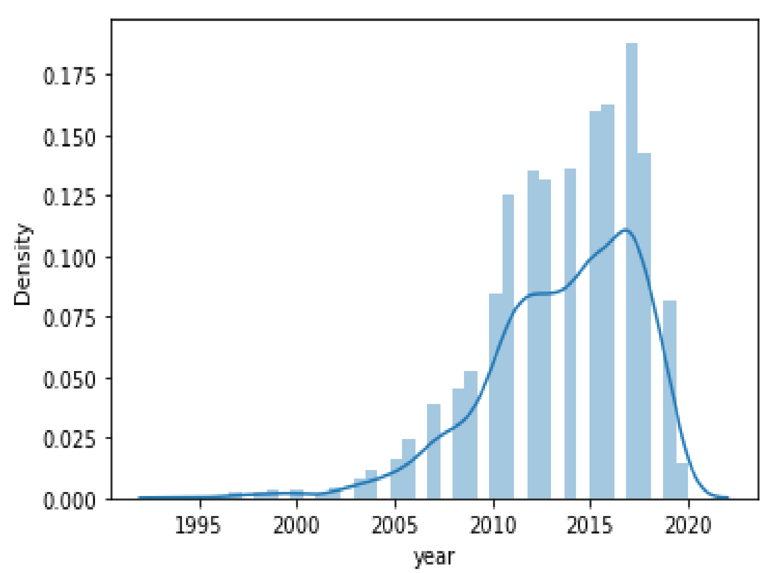
Pie Chart for Owner Distribution

```
data=df['owner'].value_counts()

l=df['owner'].unique()

plt.figure(figsize=(7,7))

plt.title('Owner distribution',size=17,color='g')

plt.pie(data,labels=l,autopct='%1.1f%%')

plt.show()
```



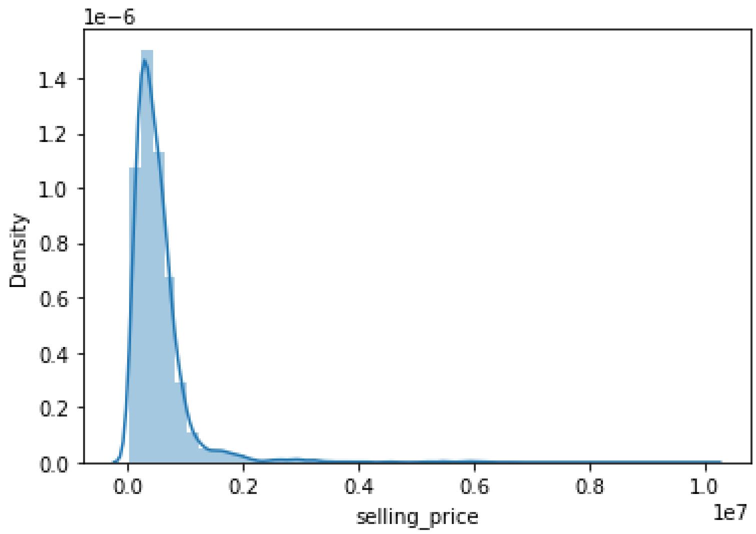
2.5 Visualization

```
sns.distplot(df['year'])
plt.show()
```



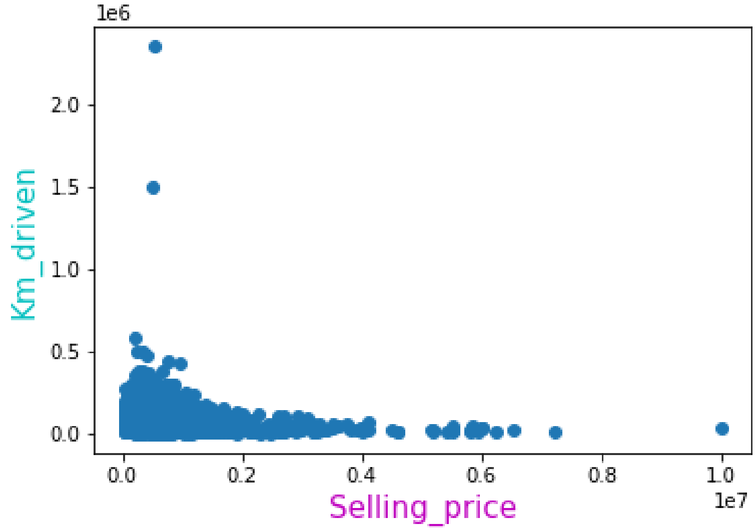
Distplot for the Selling Price

```
sns.distplot(df['selling_price'])

plt.show()
```



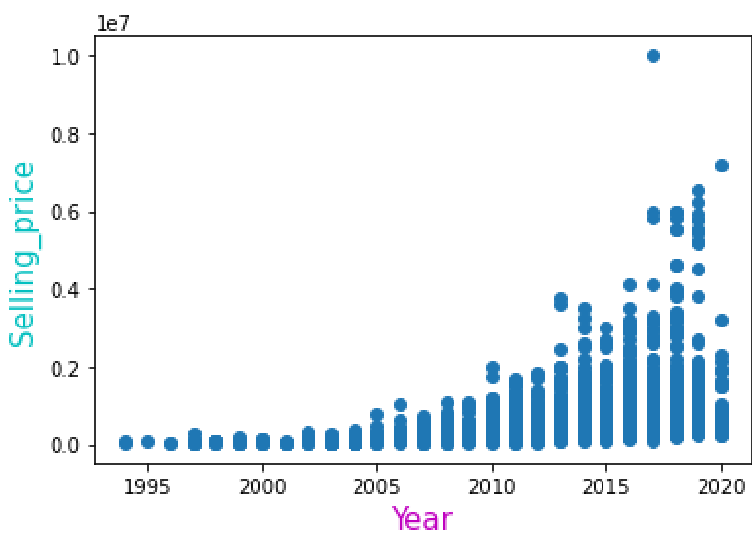
Scatter Plot between selling price and km driven

```
plt.scatter(df['selling_price'],df['km_driven'])

plt.xlabel('Selling_price',color='m',fontsize=15)

plt.ylabel('Km_driven',color='c',fontsize=15)

plt.show()
```



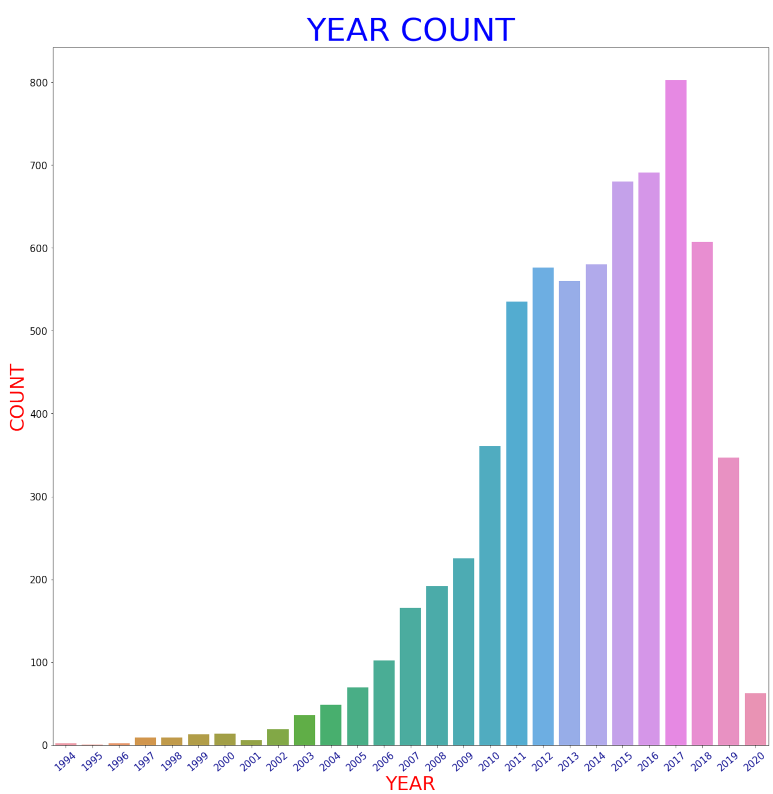
Scatter Plot between year and selling price

```
plt.scatter(df['year'],df['selling_price'])

plt.xlabel('Year',color='m',fontsize=15)

plt.ylabel('Selling_price',color='c',fontsize=15)

plt.show()
```



Countplot for the years

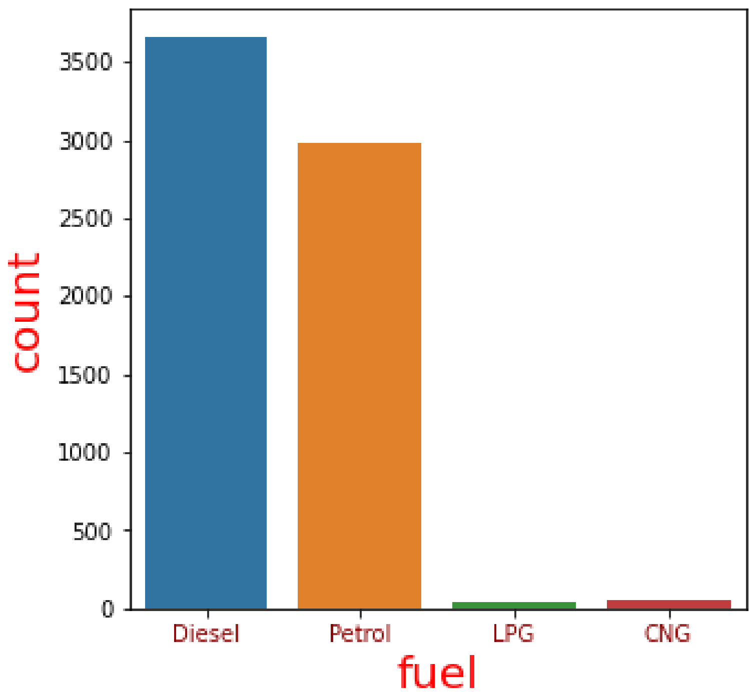
```
plt.figure(figsize=(20,20))
sns.countplot(x='year',data=df)  #countplot in seaborn
plt.xlabel('YEAR',fontsize=30,color='r')
plt.ylabel('COUNT',fontsize=30,color='r')
plt.title('YEAR COUNT',fontsize=50,color='b')
plt.yticks(fontsize=15)  #yticks refers to points in y axis
plt.xticks(fontsize=15,rotation=40,color='darkblue')
plt.show()
```



Countplot for the fuel

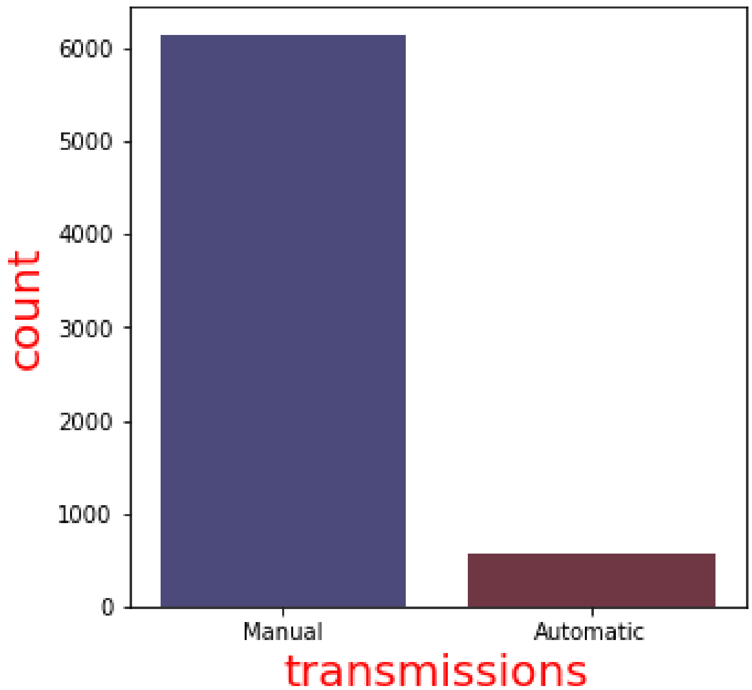
We are counting the cars which runs with petro,diesel etc

```
plt.figure(figsize=(5,5))

sns.countplot(x='fuel',data=df)

plt.xlabel('fuel',fontsize=20,color='r')

plt.ylabel('count',fontsize=20,color='r')

plt.xticks(color='darkred')

plt.show()
```



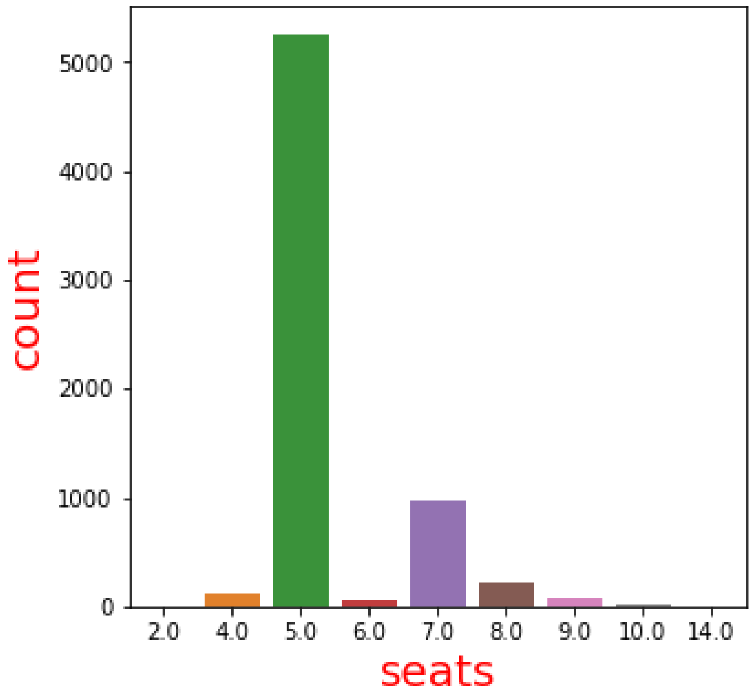
Countplot for the Transimissions

```
plt.figure(figsize=(5,5))

sns.countplot(x='transmission',data=df,palette='icefire')

plt.xlabel('transmissions',fontsize=20,color='r')

plt.ylabel('count',fontsize=20,color='r')

plt.show()
```



Countplot for the seats

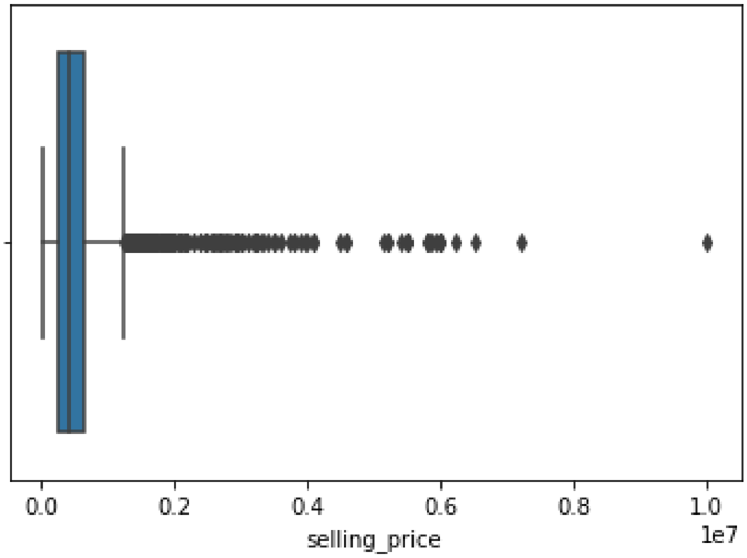
```
plt.figure(figsize=(5,5))

sns.countplot(df['seats'])

plt.xlabel('seats',fontsize=20,color='r')

plt.ylabel('count',fontsize=20,color='r')

plt.show()
```



Boxplot for the target variable selling price

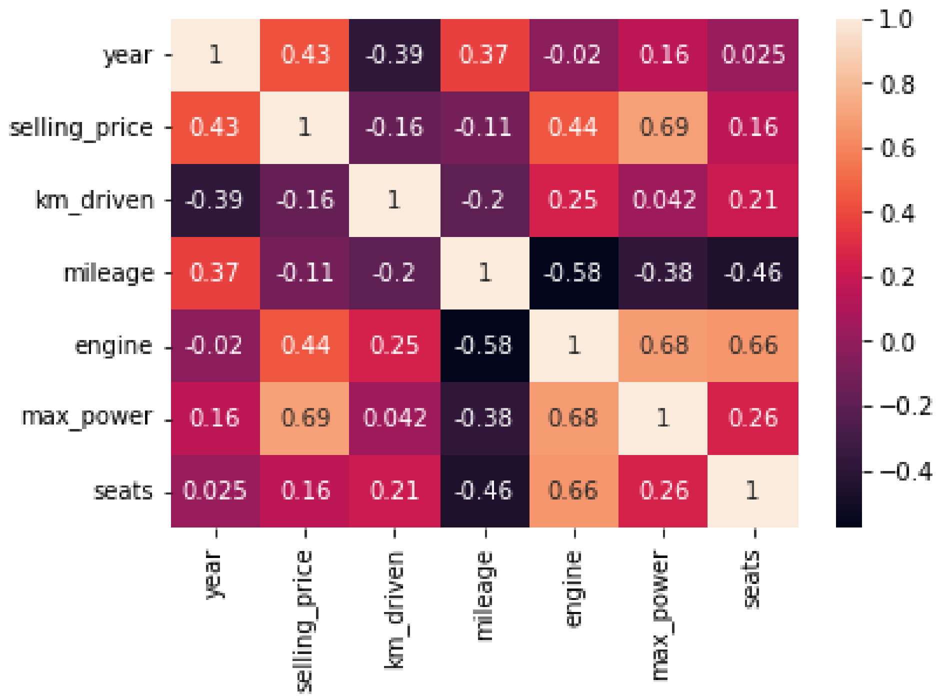
```
sns.boxplot(df['selling_price'])
```

```
plt.show()
```



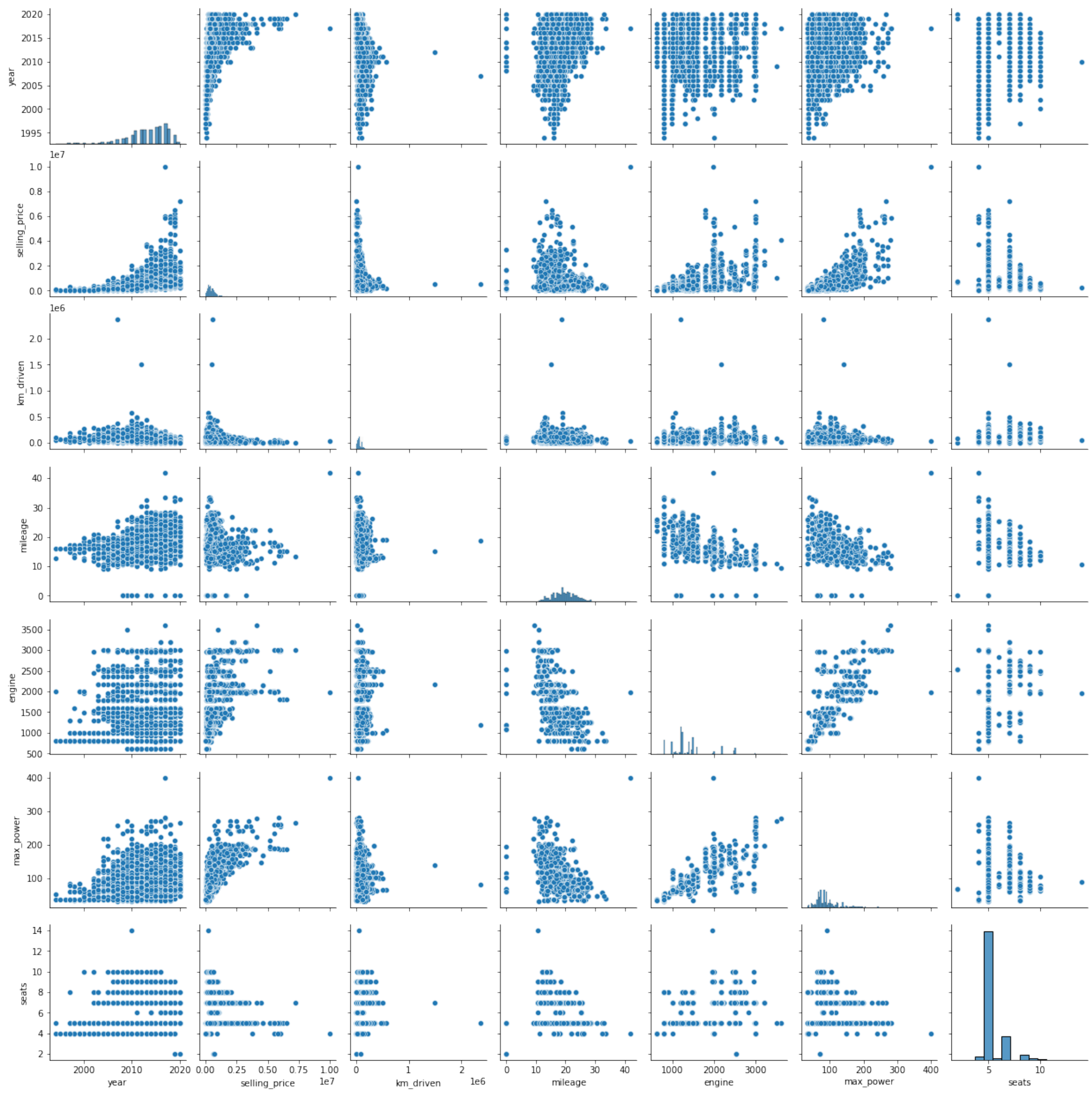
Heatmap

```
sns.heatmap(df.corr(),annot=True)

plt.show()
```



Pairplot

```
sns.pairplot(df)

plt.show()
```



Chapter 3

Code

Pandas

- Pandas is a popular open-source library in Python.
- It is used for data manipulation and analysis.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Importing Essential libraries

```python
import pandas as pd
```

```python
import numpy as np

import warnings

warnings.simplefilter("ignore")
```

Importing csv file to jupyter notebook using Pandas

```python
s=pd.read_csv('Car details v3.csv')

df=pd.DataFrame(s)

df
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	sea
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5
...	...	...	...	...	...	...	...	...	...	...	...	...	
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.8 kmpl	1493 CC	110 bhp	24@ 1,900-2,750(kgm@ rpm)	5
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.3 kmpl	1248 CC	73.9 bhp	190Nm@ 2000rpm	5
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5

8128 rows × 13 columns

Head and Tail

df.head(6) is used to print first 6 rows from the dataset

```
df.head(6)
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	74 bhp	190Nm@ 2000rpm	5.0
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	103.52 bhp	250Nm@ 1500-2500rpm	5.0
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.0
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	90 bhp	22.4 kgm at 1750-2750rpm	5.0
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.0
5	Hyundai Xcent 1.2 VTVT E Plus	2017	440000	45000	Petrol	Individual	Manual	First Owner	20.14 kmpl	1197 CC	81.86 bhp	113.75nm@ 4000rpm	5.0

df.tail(6) is used to print last 6 rows from the dataset

```
df.tail(6)
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	sea
8122	Hyundai i20 Magna 1.4 CRDi	2014	475000	80000	Diesel	Individual	Manual	Second Owner	22.54 kmpl	1396 CC	88.73 bhp	219.7Nm@ 1500-2750rpm	5
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.8 kmpl	1493 CC	110 bhp	24@ 1,900-2,750(kgm@ rpm)	5
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.3 kmpl	1248 CC	73.9 bhp	190Nm@ 2000rpm	5
8126	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5
8127	Tata Indigo CR4	2013	290000	25000	Diesel	Individual	Manual	First Owner	23.57 kmpl	1396 CC	70 bhp	140Nm@ 1800-3000rpm	5

```
df.shape would give the shape of the dataset
```

```
df.shape
```

```
Out[156]:
(8128, 13)
```

```
df.ndim
```

```
Out[157]:
2
```

df.info() would give the description about the dataset.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   object
 9   engine         7907 non-null   object
 10  max_power      7913 non-null   object
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

```
df[df['fuel']=='Petrol']   #getting rows which contain fuel as petrol
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	torque	seat
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	78 bhp	12.7@ 2,700(kgm@ rpm)	5.
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	88.2 bhp	11.5@ 4,500(kgm@ rpm)	5.
5	Hyundai Xcent 1.2 VTVT E Plus	2017	440000	45000	Petrol	Individual	Manual	First Owner	20.14 kmpl	1197 CC	81.86 bhp	113.75nm@ 4000rpm	5.
7	Maruti 800 DX BSII	2001	45000	5000	Petrol	Individual	Manual	Second Owner	16.1 kmpl	796 CC	37 bhp	59Nm@ 2500rpm	4.
11	Maruti Zen LX	2005	92000	100000	Petrol	Individual	Manual	Second Owner	17.3 kmpl	993 CC	60 bhp	78Nm@ 4500rpm	5.
...	...	...	...	...	...	...	...	...	...	...	...	...	.
8118	Hyundai i20 Magna	2013	380000	25000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5.
8119	Maruti Wagon R LXI Optional	2017	360000	80000	Petrol	Individual	Manual	First Owner	20.51 kmpl	998 CC	67.04 bhp	90Nm@ 3500rpm	5.
8120	Hyundai Santro Xing GLS	2008	120000	191000	Petrol	Individual	Manual	First Owner	17.92 kmpl	1086 CC	62.1 bhp	96.1Nm@ 3000rpm	5.
8121	Maruti Wagon R VXI BS IV with ABS	2013	260000	50000	Petrol	Individual	Manual	Second Owner	18.9 kmpl	998 CC	67.1 bhp	90Nm@ 3500rpm	5.
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.5 kmpl	1197 CC	82.85 bhp	113.7Nm@ 4000rpm	5.

3631 rows × 13 columns

```
df['seats'].mode()
```

```
Out[161]:

0    5.0
Name: seats, dtype: float64
```

```
df.columns  #returns columns in the dataset
```

```
Out[162]:

Index(['name', 'year', 'selling_price', 'km_driven', 'fuel', 'seller_type',
       'transmission', 'owner', 'mileage', 'engine', 'max_power', 'torque',
       'seats'],
      dtype='object')
```

```
df.year.value_counts()
```

```
2017    1018
2016     859
2018     807
2015     776
2013     670
2012     651
2014     621
2011     592
2019     583
2010     394
2009     246
2008     214
2007     183
2006     124
2005      97
2020      74
2004      62
2003      49
2002      27
2000      22
1999      18
1997      11
2001      10
1998      10
1996       3
1994       3
1995       2
1983       1
1991       1
Name: year, dtype: int64
```

Checking Null Values

```
df.isnull().sum()
```

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          221
engine           221
max_power        215
torque           222
seats            221
dtype: int64
```

```
df.dropna(inplace=True)
```

```
df.isnull().sum()
```

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
mileage          0
engine           0
max_power        0
torque           0
seats            0
dtype: int64
```

```python
df['mileage']=df['mileage'].apply(lambda x:x.split()[0])

df['engine']=df['engine'].apply(lambda x:x.split()[0])

df['max_power']=df['max_power'].apply(lambda x:x.split()[0])


df['mileage']=df['mileage'].astype('float')

df['engine']=df['engine'].astype('int')

df['max_power']=df['max_power'].astype('float')


df.describe()
```

	year	selling_price	km_driven	mileage	engine	max_power	seats
count	6717.000000	6.717000e+03	6.717000e+03	6717.000000	6717.000000	6717.000000	6717.000000
mean	2013.611136	5.263860e+05	7.339834e+04	19.466585	1430.985857	87.766100	5.434271
std	3.897402	5.235504e+05	5.870328e+04	4.048102	493.469198	31.724555	0.983805
min	1994.000000	2.999900e+04	1.000000e+00	0.000000	624.000000	32.800000	2.000000
25%	2011.000000	2.500000e+05	3.800000e+04	16.800000	1197.000000	67.100000	5.000000
50%	2014.000000	4.200000e+05	6.820300e+04	19.440000	1248.000000	81.830000	5.000000
75%	2017.000000	6.500000e+05	1.000000e+05	22.500000	1498.000000	100.000000	5.000000
max	2020.000000	1.000000e+07	2.360457e+06	42.000000	3604.000000	400.000000	14.000000

Binning

```
min1=df['selling_price'].min()

max1=df['selling_price'].max()

bins=np.linspace(min1,max1,4)

group_names=['budget_friendly','medium','premium']

df['car_range']=pd.cut(df['selling_price'],bins,labels=group_names)
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats	car_ra
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.40	1248	74.00	5.0	budget_frie
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14	1498	103.52	5.0	budget_frie
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.70	1497	78.00	5.0	budget_frie
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.00	1396	90.00	5.0	budget_frie
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.10	1298	88.20	5.0	budget_frie
...	...	...	...	...	...	...	...	...	...	...	...	...	
8121	Maruti Wagon R VXI BS IV with ABS	2013	260000	50000	Petrol	Individual	Manual	Second Owner	18.90	998	67.10	5.0	budget_frie
8122	Hyundai i20 Magna 1.4 CRDi	2014	475000	80000	Diesel	Individual	Manual	Second Owner	22.54	1396	88.73	5.0	budget_frie
8123	Hyundai i20 Magna	2013	320000	110000	Petrol	Individual	Manual	First Owner	18.50	1197	82.85	5.0	budget_frie
8124	Hyundai Verna CRDi SX	2007	135000	119000	Diesel	Individual	Manual	Fourth & Above Owner	16.80	1493	110.00	5.0	budget_frie
8125	Maruti Swift Dzire ZDi	2009	382000	120000	Diesel	Individual	Manual	First Owner	19.30	1248	73.90	5.0	budget_frie

6717 rows × 13 columns

```
df.car_range.value_counts()
```

```
Out[184]:
budget_friendly    6681
medium               33
premium               2
Name: car_range, dtype: int64
```

Dealing with Outliers

```python
print("Shape Before removing outliers",df.shape)
```

```
Shape Before removing outliers (6717, 13)
```

```python
for col in ['km_driven','selling_price','year','mileage','max_power','e
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5*IQR
    upper_bound = Q3 + 1.5*IQR
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

print("Shape After removing outliers",df.shape)
```

```
Shape After removing outliers (5329, 13)
```

Converting categorical columns to numerical columns

```
df['fuel'].replace(['Diesel','Petrol','LPG','CNG'],[0,1,2,3],inplace=Tr

df['seller_type'].replace(['Individual', 'Dealer', 'Trustmark Dealer'],

df['transmission'].replace(['Manual', 'Automatic'],[0,1],inplace=True)

df['owner'].replace(['First Owner', 'Second Owner', 'Third Owner','Four
```

```
df.head()
```

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max_power	seats	car_range
0	Maruti Swift Dzire VDI	2014	450000	145500	0	0	0	0	23.40	1248	74.00	5.0	budget_friendly
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	0	0	0	1	21.14	1498	103.52	5.0	budget_friendly
2	Honda City 2017-2020 EXi	2006	158000	140000	1	0	0	2	17.70	1497	78.00	5.0	budget_friendly
3	Hyundai i20 Sportz Diesel	2010	225000	127000	0	0	0	0	23.00	1396	90.00	5.0	budget_friendly
4	Maruti Swift VXI BSIII	2007	130000	120000	1	0	0	0	16.10	1298	88.20	5.0	budget_friendly

Building the Model

```
#importing required libraries

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn import metrics
```

```python
from sklearn.metrics import r2_score
```

Splitting the Dataset

Initializing the dependent and independent variables and splitting the dataset into train set and test set.

our Target variable is the selling price.

```python
y=df['selling_price']
x=df.drop(columns=['selling_price','name','car_range'])
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.1,rand
```

Linear Regression

```python
Lr = LinearRegression()
Lr.fit(X_train,y_train)
```

Predicting using Linear regression

Predicting using Linear regression model and store it in ypred variable

```python
y_pred = Lr.predict(X_test)
print("R2 score :",r2_score(y_test,y_pred))
```

Random Forest Regressor

```
R2 score: 0.7407950865219055
```

```
Rf=RandomForestRegressor()

Rf.fit(X_train,y_train)
```

Predicting using Random forest Regressor

Predicting using Random Forest regressor model and store it in ypred variable

```
y_pred=Rf.predict(X_test)

print("R2 score:",r2_score(y_pred,y_test))
```

```
R2 score: 0.8944272324589815
```

Here,We can observe that r2 score of our model is 0.89

Model Evaluation

Checking whether our model is predicting well or not.

Testing the Data with the model

```
Rf.predict([['2014','145500','0','0','0','0','23.40','1248','74.00','5.
```

```
Out[204]:
array([470952.44])
```

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In conclusion, our project aimed to predict car prices using specific machine learning models, namely Linear Regression and Random Forest Regressor. By using specific features like mileage, kilometer driven, engine capacity, and number of owners,transmissions, we developed model that successfully estimated car prices.This model can help both buyers and sellers in determining fair and reasonable prices based on relevant car attributes. This project demonstrates the potential of machine learning in the automotive industry, empowering individuals to make informed decisions regarding car pricing. Here,We select the Model with highest accuray i.e Random forest Regressor.