NAME OF THE PROJECT

Used car price prediction

Submitted by:
REENA

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. Some of the reference sources are as follows:

- Coding Ninjas
- Medium.com
- StackOverflow

INTRODUCTION

Business Problem Framing

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model.
This project contains two phaseData Collection Phase You have to scrape at least 5000 used cars data. You can scrape more data as well, it's up to you. more the data better the model In this section You need to scrape the data of used cars from websites (Olx, cardekho, Cars24 etc.) You need web scraping for this. You have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometers, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends

on the website from which you are fetching the data. Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback. Note – The data which you are collecting is important to us. Kindly don't share it on any public platforms. Model Building Phase After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like. 1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best mode

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

1.Firstly,we will prepare our own dataset using web scraping.

2. After that we will check whether the project is a regression type or a classification type.

3. We will also check whether our dataset is balanced or imbalanced. If it is an imbalanced one, we will apply sampling techniques to balance the dataset.

4. Then we will do model building and check its accuracy.

5. Our main motto is to build a model with good accuracy .

Describe the domain related concepts that you think will be useful for better understanding of the project.

Review of Literature

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper

SOFTWARE: Jupyter Notebook (Anaconda 3) – Python 3.7.6

LIBRARIES: The tools, libraries, and packages we used for accomplishing this project are pandas, numpy, matplotlib, seaborn, scipy stats, sklearn.decomposition, sklearn standardscaler, GridSearchCV, joblib. from sklearn.preprocessing

import StandardScaler As these columns are different in scale, they are standardized to have common scale while building machine learning model. This is useful when you want to compare data that correspond to different units. from sklearn.preprocessing import Label Encoder Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand. from sklearn.model_selection import train_test_split,cross_val_score Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets. Through pandas library we loaded our csv file ‘Data file’ into dataframe and performed data manipulation and analysis. With the help of numpy we worked with arrays. With the help of matplotlib and seaborn we did plot various graphs and figures and done data visualization. With sklearn’s standardscaler package we scaled all the feature variables onto single scale

ANALYTICAL PROBLEM FRAMING MATHEM

ATICAL/ ANALYTICAL MODELING OF THE PROBLEM

If you look at data science, we are actually using mathematical models to model (and hopefully through the model to explain some of the things that we have seen) business circumstances, environment etc and through these model, we can get more insights such as the outcomes of our decision undertaken, what should we do next or how shall we do it to improve the odds. So mathematical models are important, selecting the right one to answer the business question can tremendous value to the organization. Here I am using Random Forest Regressor with accuracy .

DATA SOURCES AND THEIR FORMATS

Data Source:

The read_csv function of the pandas library is used to read the content of a CSV file into the python environment as a pandas DataFrame. The function can read the files from the OS by using proper path to the file. Data description: Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. DATA PREPROCESSING DONE

♣ I have checked for null values

♣ I have label encoded the object type columns in the dataset.

♣ I have checked the correlation between dependant and independent variables using heatmap. I have seen

most of the independent variables are correlated with each other and the target variable is positively correlated with a very few independent variables.

♣I have done some visualization.I have checked outliers using boxplots ,but no outliers are present.

♣ I also have checked for skewness in my data, but the skewness present is very negligible, so I don't consider it.

♣ I have splitted the dependant and independent variables into x and y.

♣ I have scaled the data using StandardScaler method and made my data ready for model building.

DATA DESCRIPTION

After loading all the required libraries we loaded the data into our jupyter notebook. The dataset contains4722 records (rows) and 12 features.

```
#Importing Libraries

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error

from sklearn.model_selection import cross_val_score

import warnings
warnings.filterwarnings('ignore')
```

```
data.shape
```

```
(4722, 12)
```

```
data.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 0.1', 'Model', 'Make_Year', 'Driven_Kilometers',
       'Fuel', 'Transmission', 'Owner(s)', 'Mileage', 'Engine', 'Price',
       'Location'],
      dtype='object')
```

Checking null values:

```
data.shape
```

```
(4722, 12)
```

```
data.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 0.1', 'Model', 'Make_Year', 'Driven_Kilometers',
       'Fuel', 'Transmission', 'Owner(s)', 'Mileage', 'Engine', 'Price',
       'Location'],
      dtype='object')
```

Dealing with null values:

```
data=data.dropna()
```

```
data
```

	Unnamed: 0	Unnamed: 0.1	Model	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Location
0	19	19	Kia Seltos Htx G	2021	13400 Kms	Petrol	Manual	1	16.80	1497	1425000.0	Ahmedabad
1	121	121	Tata Harrier Xz Dark Edition Bsiv	2019	36000 Kms	Diesel	Manual	1	17.00	1956	1700000.0	Ahmedabad
2	122	122	Tata Indica Ev Lx	2012	30645 Kms	Diesel	Manual	1	25.00	1396	131000.0	Ahmedabad
3	125	125	Jeep Compass . Sport Dct	2022	3000 Kms	Petrol	Automatic	1	14.10	1368	1850000.0	Ahmedabad
4	127	127	Volkswagen Tiguan Allspace motion	2020	20000 Kms	Petrol	Automatic	1	17.01	1984	2950000.0	Ahmedabad
...	...	...	...	...	...	...	...	...	...	...	...	...
4717	7754	833	Hyundai I . Spotz	2017	63851 Kms	Petrol	Manual	1	18.60	1197	625000.0	Pune
4718	7755	834	Hyundai I Sportz At .	2013	52507 Kms	Petrol	Automatic	2	15.00	1396	420000.0	Pune
4719	7756	835	Honda Jazz . Sv I Vtec	2017	32703 Kms	Petrol	Manual	2	18.70	1199	565000.0	Pune
4720	7757	836	Honda City . V Mt Exclusive	2010	33359 Kms	Petrol	Manual	1	17.00	1497	375000.0	Pune
4721	7758	837	Honda City Sv Mt	2014	62919 Kms	Petrol	Manual	1	17.40	1497	560000.0	Pune

4706 rows × 12 columns

Dropping unwanted columns:

```
#It contains numbers index that doesn't affect our car price, hence removing this column
data = data.drop('Unnamed: 0',axis=1)
data = data.drop('Unnamed: 0.1',axis=1)
```

```
data.columns
```

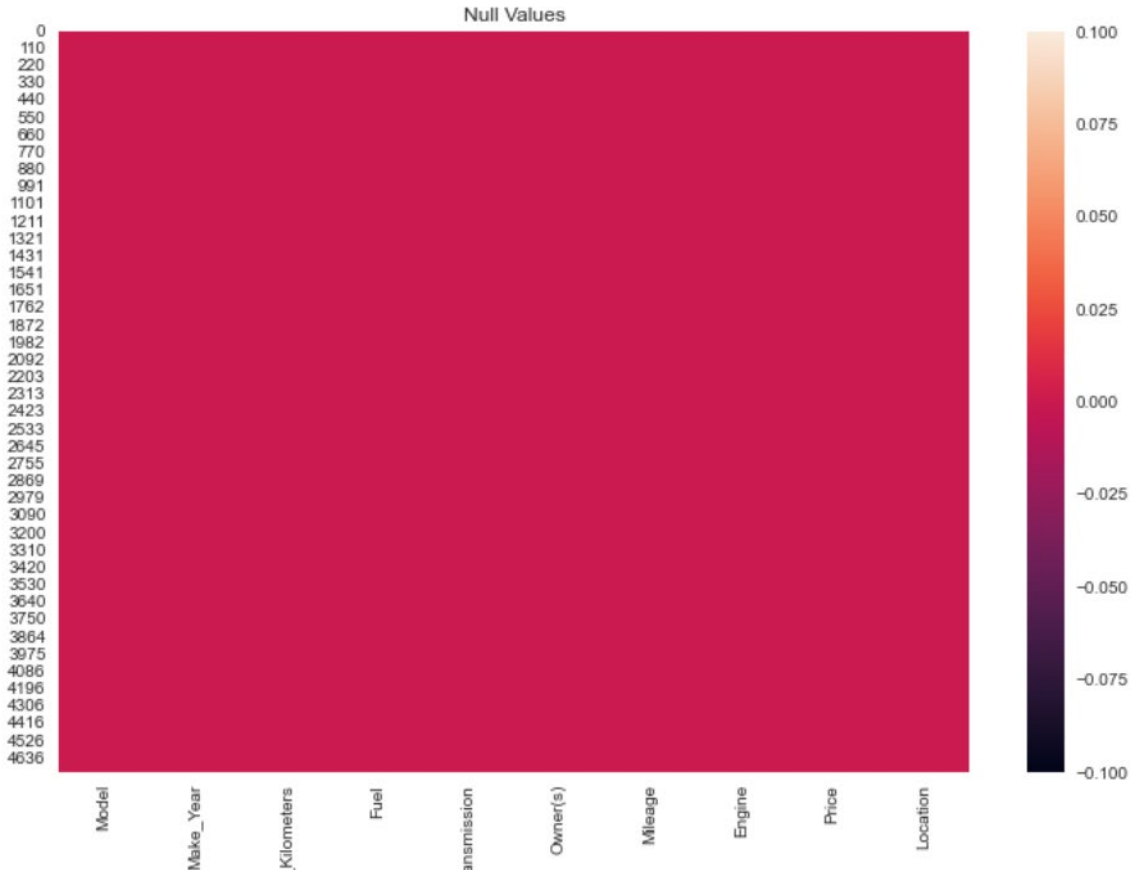
```
Index(['Model', 'Make_Year', 'Driven_Kilometers', 'Fuel', 'Transmission',
       'Owner(s)', 'Mileage', 'Engine', 'Price', 'Location'],
      dtype='object')
```

Data information:-

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4706 entries, 0 to 4721
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Model              4706 non-null   object 
 1   Make_Year          4706 non-null   int64  
 2   Driven_Kilometers  4706 non-null   object 
 3   Fuel               4706 non-null   object 
 4   Transmission       4706 non-null   object 
 5   Owner(s)           4706 non-null   int64  
 6   Mileage            4706 non-null   float64
 7   Engine             4706 non-null   object 
 8   Price              4706 non-null   float64
 9   Location           4706 non-null   object 
dtypes: float64(2), int64(2), object(6)
memory usage: 404.4+ KB
```

Checking null values using heatmap:



We can see that now there is no any null values present in given dataset.

Checking uniques values:

```
data.nunique().sort_values().to_frame("Unique Values")
```

	Unique Values
Transmission	2
Owner(s)	4
Fuel	5
Location	12
Make_Year	20
Engine	149
Mileage	450
Price	813
Model	1519
Driven_Kilometers	1754

Data preprocessing:

```
# Data pre processing

data["Driven_Kilometers"]=data["Driven_Kilometers"].apply(lambda x: x.replace(',','') if x!='-' else '-')
data["Driven_Kilometers"]=data["Driven_Kilometers"].apply(lambda x: int(x.split(' ')[0]) if x!='-' else 0)
data
```

	Model	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Engine	Price	Location
0	Kia Seltos Htx G	2021	13400	Petrol	Manual	1	16.80	1497	1425000.0	Ahmedabad
1	Tata Harrier Xz Dark Edition Bsiv	2019	36000	Diesel	Manual	1	17.00	1956	1700000.0	Ahmedabad
2	Tata Indica Ev Lx	2012	30645	Diesel	Manual	1	25.00	1396	131000.0	Ahmedabad
3	Jeep Compass . Sport Dct	2022	3000	Petrol	Automatic	1	14.10	1368	1850000.0	Ahmedabad
4	Volkswagen Tiguan Allspace motion	2020	20000	Petrol	Automatic	1	17.01	1984	2950000.0	Ahmedabad
...	...	...	...	...	...	...	...	...	...	...
4717	Hyundai I . Spotz	2017	63851	Petrol	Manual	1	18.60	1197	625000.0	Pune
4718	Hyundai I Sportz At .	2013	52507	Petrol	Automatic	2	15.00	1396	420000.0	Pune
4719	Honda Jazz . Sv I Vtec	2017	32703	Petrol	Manual	2	18.70	1199	565000.0	Pune
4720	Honda City . V Mt Exclusive	2010	33250	Petrol	Manual	1	17.00	1497	375000.0	Pune

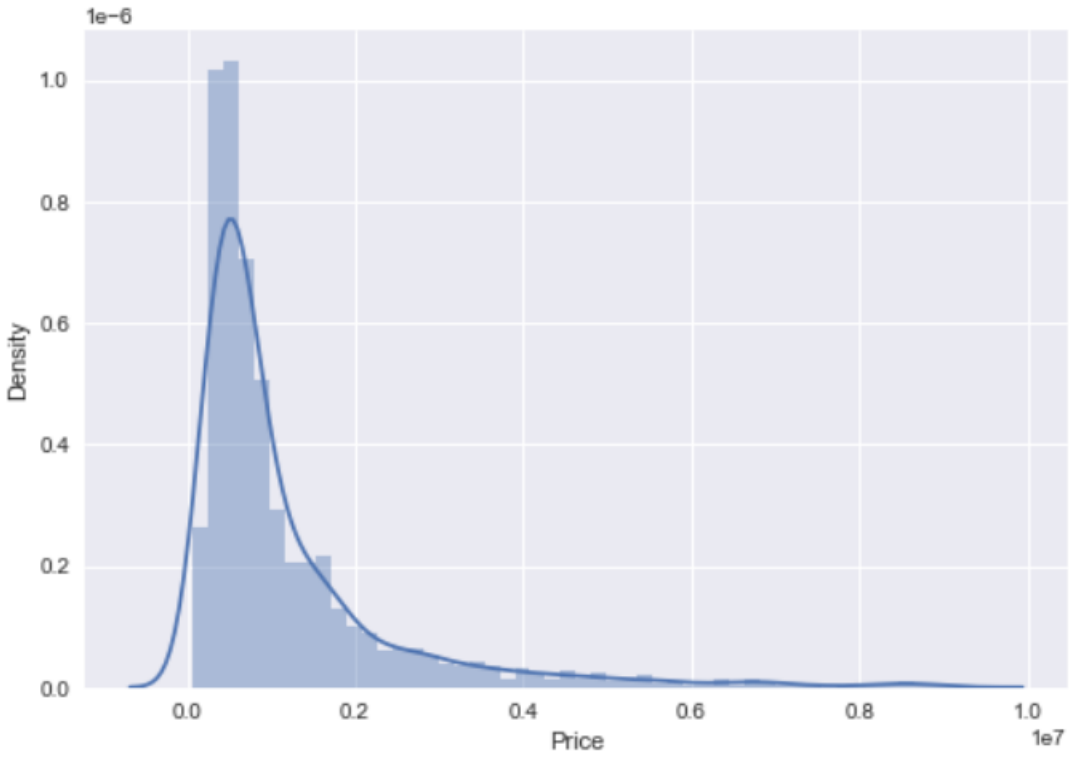
Data description:

```
#Let's check the overall metrics of each column


data.describe()
```

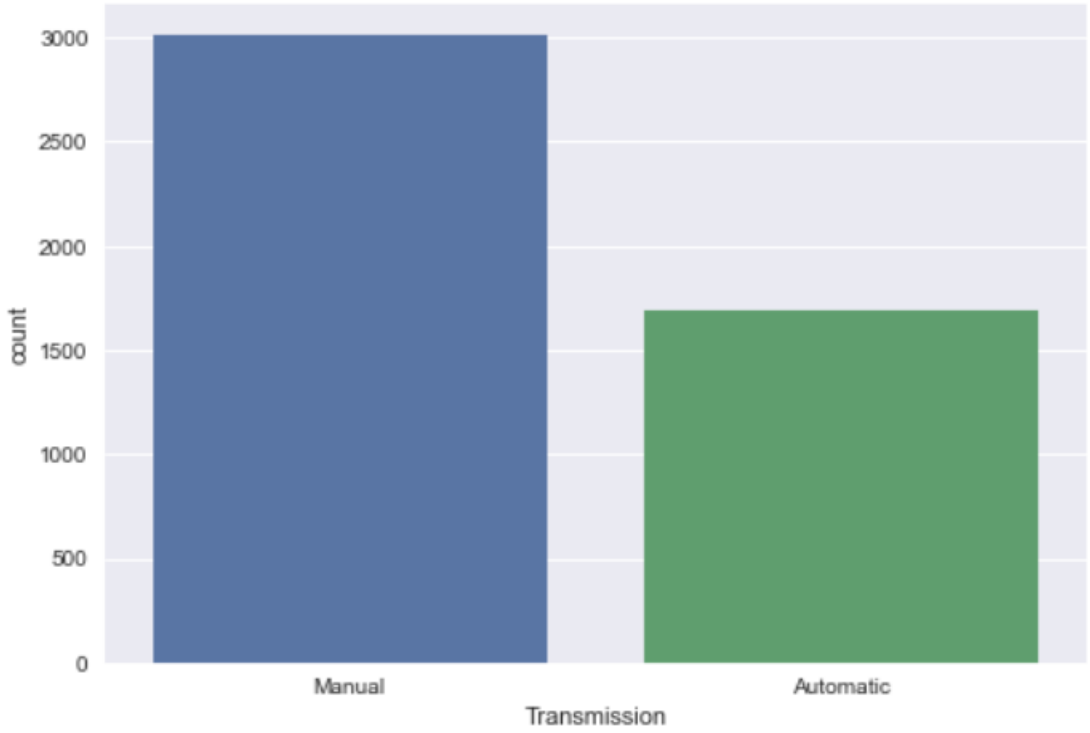
	Make_Year	Driven_Kilometers	Owner(s)	Mileage	Price
count	4706.000000	4706.000000	4706.000000	4706.000000	4.706000e+03
mean	2015.725669	55514.239907	1.232894	58.881549	1.197412e+06
std	3.356749	41296.767424	0.480158	280.495161	1.343059e+06
min	2003.000000	145.000000	1.000000	0.000000	5.645900e+04
25%	2014.000000	30000.250000	1.000000	15.960000	4.332500e+05
50%	2016.000000	52000.000000	1.000000	18.150000	7.100000e+05
75%	2018.000000	73000.000000	1.000000	20.890000	1.390000e+06
max	2022.000000	970000.000000	4.000000	2995.000000	9.200000e+06

Univariate Analysis

```
#Let's check the Target column : "Price"

plt.style.use('seaborn')
sns.distplot(data['Price'])
plt.show()
```



```
sns.countplot(x = 'Transmission', data = data)
plt.show()
```

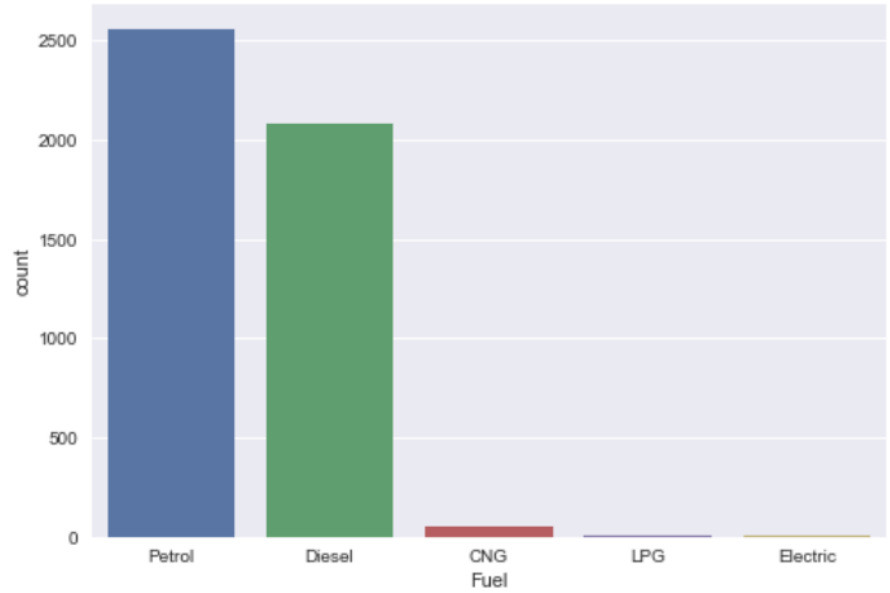


Manual tramission is mostly used.

Fuel type:

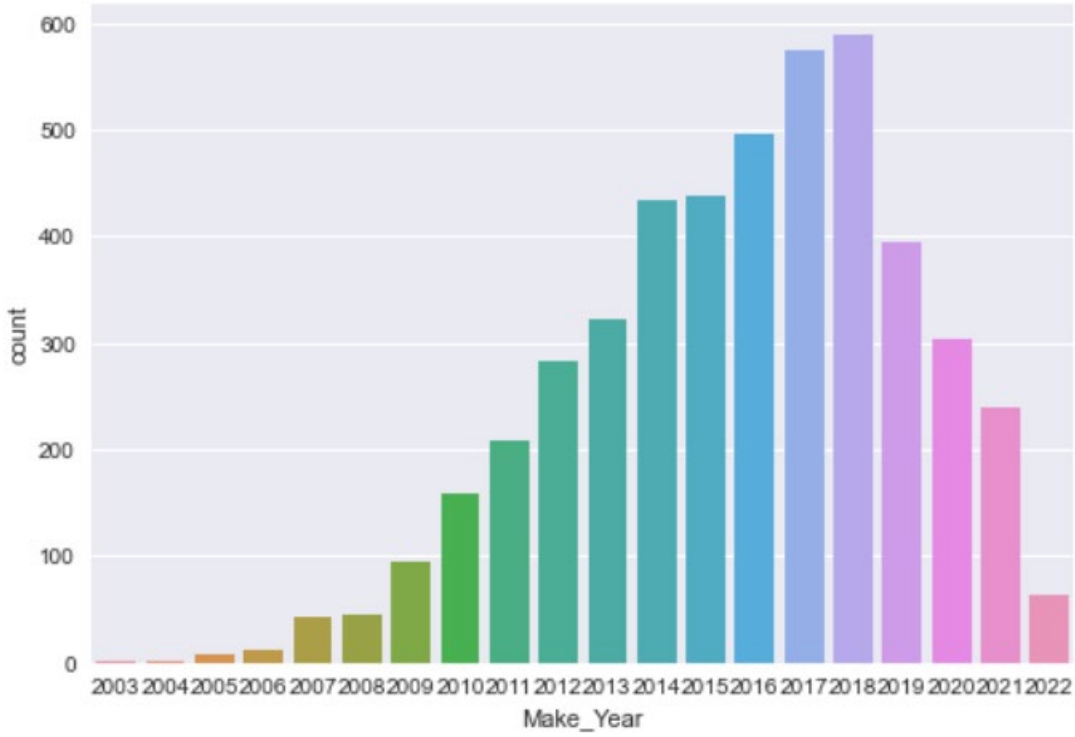
```
sns.countplot(x = 'Fuel', data = data)
```

```
<AxesSubplot:xlabel='Fuel', ylabel='count'>
```



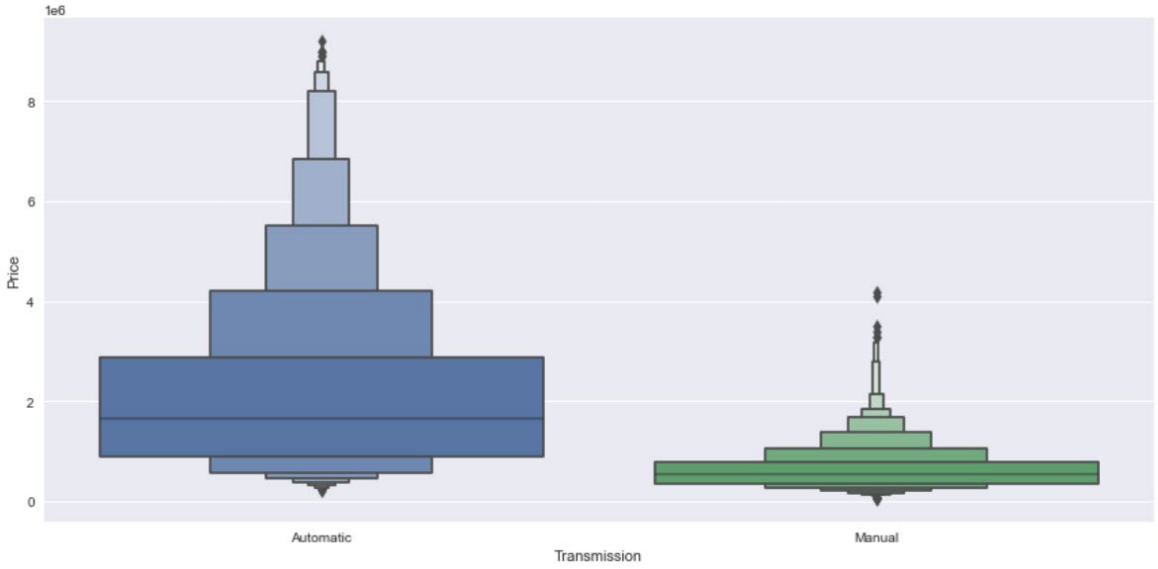
Make year:

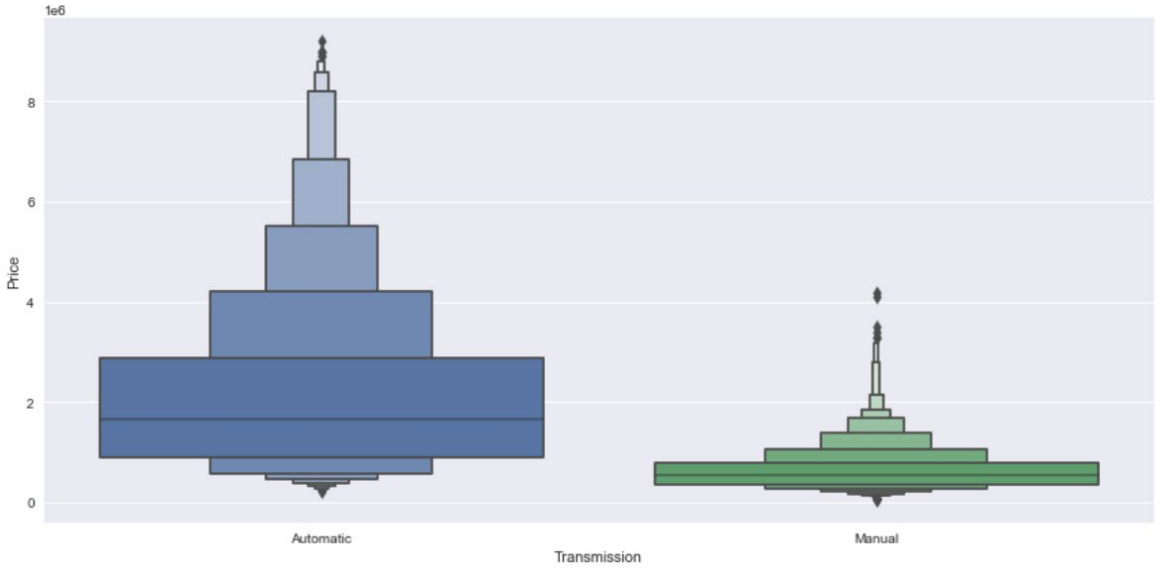
```
sns.countplot(x = 'Make_Year', data = data)
plt.show()
```



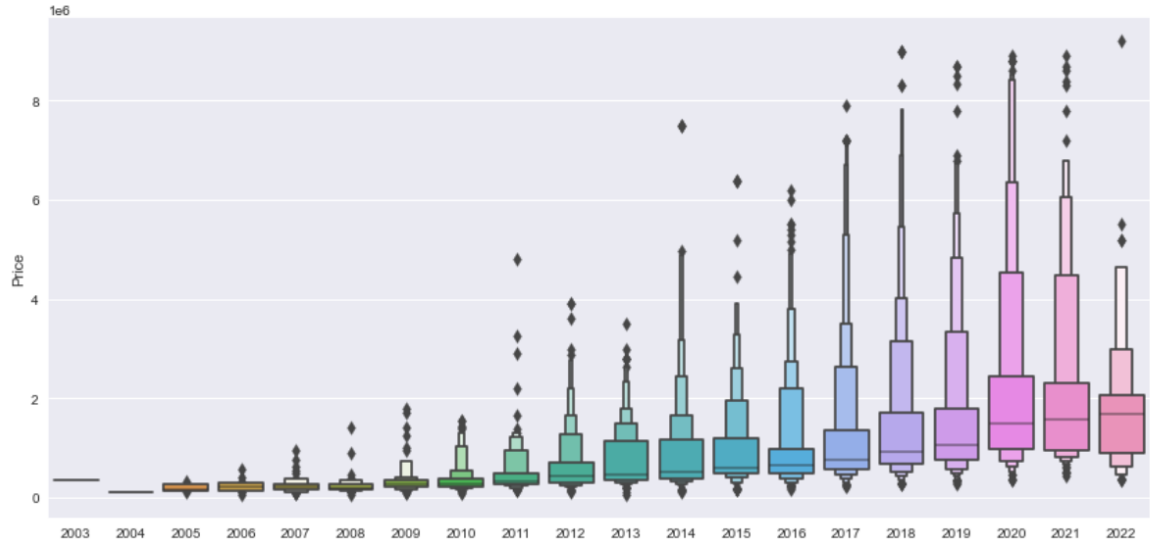
Bivariate Analysis

```
sns.catplot(y = 'Price', x = 'Transmission',data= data.sort_values("Price", ascending = False),
            kind = "boxen", height = 6, aspect = 2)

plt.tight_layout
plt.show()
```


1e6
8
6
4
2
0
Price
Automatic
Manual
Transmission


1e6
8
6
4
2
0
Price
Automatic
Manual
Transmission

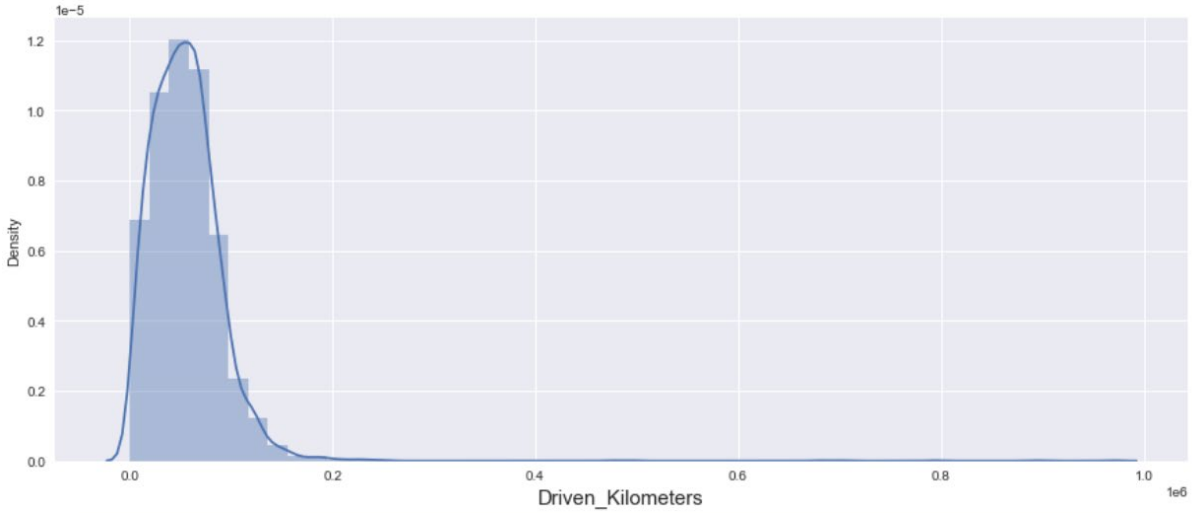
```
sns.catplot(y = 'Price', x = 'Make_Year',data= data.sort_values("Price", ascending = False),
            kind = "boxen", height = 6, aspect = 2)

plt.tight_layout
plt.show()
```

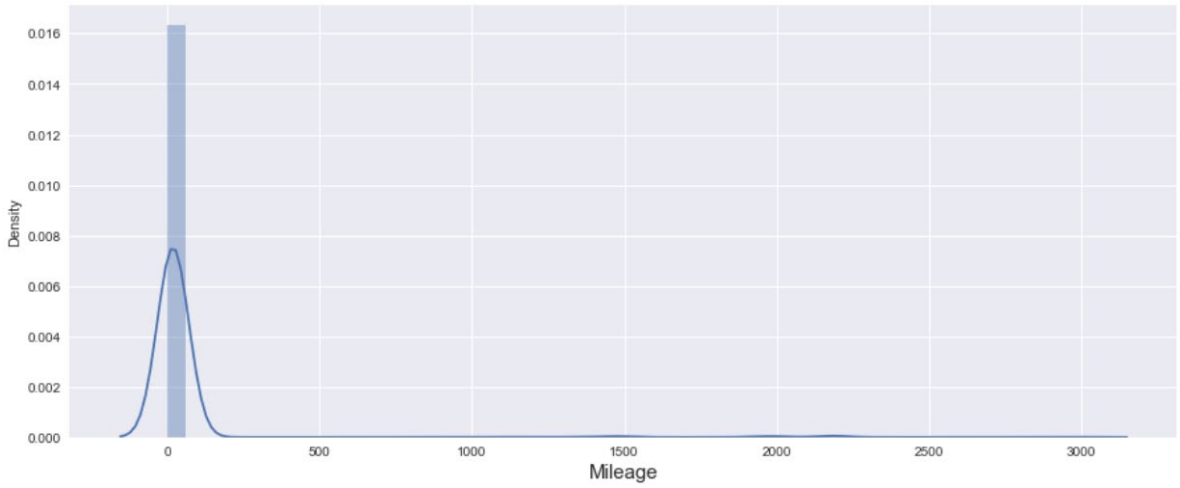


Multivariate Analysis

```
#Let's check the correlation by using the Heatmap (in order to check teh relation between features)

plt.figure(figsize=(15,8))
sns.heatmap(data.corr(),cmap='YlGnBu',annot = True)
plt.show()
```

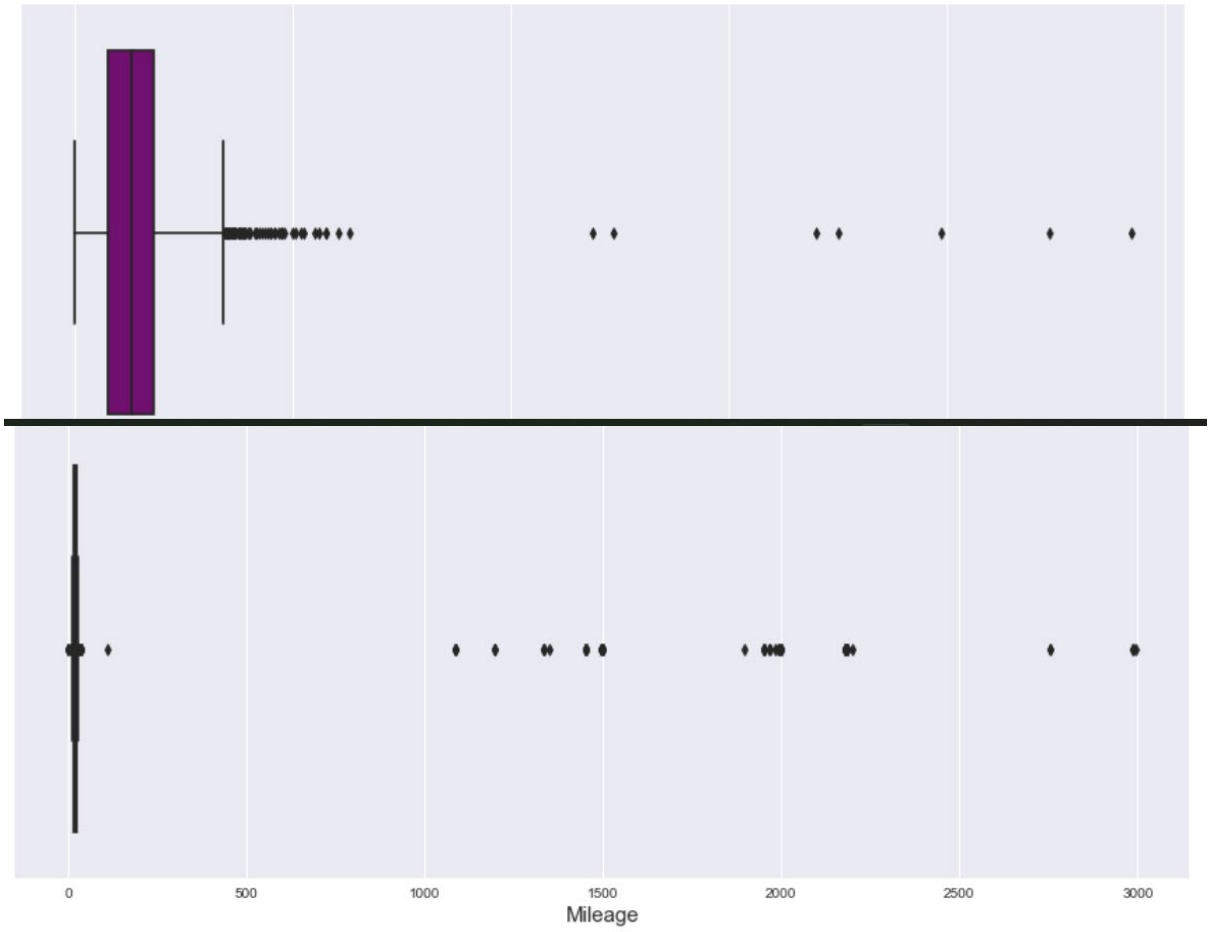
```
#Normal Distribution plotting

columns = ['Driven_Kilometers', 'Mileage'] #with numerical data

plt.tight_layout()
for i in columns:
    plt.figure(figsize=[15,6])
    sns.distplot(data[i])
    plt.xlabel(i,fontsize=15)
plt.show()
```





Checking outliers

```
# Using Boxplot to checkthe outliers

columns = ['Driven_Kilometers', 'Mileage']

plt.tight_layout()
for i in columns:
    plt.figure(figsize=[15,6])
    sns.boxplot(data[i],color = 'Purple')
    plt.xlabel(i,fontsize=15)
plt.show()
```



Applying IQR Method

```
Q1 = data[features].quantile(0.25)
Q3 = data[features].quantile(0.75)
IQR = Q3-Q1

data_new1 = data[~((data[features] < (Q1-1.5*IQR)) | (data[features]> (Q3 + 1.5*Q3))).any(axis = 1)]
```

```
print('Shape - Before and After:\n')
print('Shape Before'.ljust(20),":",data.shape)
print('Shape After'.ljust(20),":",data_new1.shape)
print('Percentage Loss'.ljust(20),":",((data.shape[0]-data_new1.shape[0])/data.shape[0])*100)
```

Applying z-score Method

```
from scipy.stats import zscore  #importing zscore from library

z=np.abs(zscore(data[features]))
threshold = 3
data_new2 = data[(z<3).all(axis=1)]
```

```
print('Shape - Before and After:\n')
print('Shape Before'.ljust(20),":",data.shape)
print('Shape After'.ljust(20),":",data_new2.shape)
print('Percentage Loss'.ljust(20),":",((data.shape[0]-data_new2.shape[0])/data.shape[0])*100)
```

Skewness

:
```
data.skew()
```

:
```
Make_Year            -0.446291
Driven_Kilometers     7.718390
Owner(s)              2.044945
Mileage               7.149449
Price                 2.779204
dtype: float64
```

:
```
#Skewness after applying the outliers technique

data_new.skew()
```

:
```
Make_Year            -0.470312
Driven_Kilometers     0.607592
Owner(s)              2.038422
Mileage               0.430599
Engine                1.191865
Price                 2.848018
dtype: float64
```

```
# Create a dataframe that will contain the Brands (by extracting the name from the title of the model column)

df1 = {}
df1 = pd.DataFrame(data_new['Model'].str.split().tolist(), columns="Brand Models A B C D E F G ".split())
df1 = df1.drop(columns=['Models','A','B','C','D','E','F','G'],axis=1)


df1['Index'] = range(4568)


df1.set_index('Index',inplace=True)#ssigning the index column
df1
```

Index	Brand
0	Kia
1	Tata

```
data_new['Index'] = range(4568)


data_new.set_index('Index',inplace=True) #assigning the index
data_new
```

Index	Model	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Price	Location
0	Kia Seltos Htx G	2021	3.280108	Petrol	Manual	1	16.80	1425000.0	Ahmedabad
1	Tata Harrier Xz Dark Edition Bsiv	2019	3.711400	Diesel	Manual	1	17.00	1700000.0	Ahmedabad
2	Tata Indica Ev Lx	2012	3.637433	Diesel	Manual	1	25.00	131000.0	Ahmedabad
3	Jeep Compass . Sport Dct	2022	2.720446	Petrol	Automatic	1	14.10	1850000.0	Ahmedabad
4	Volkswagen Tiguan Allspace motion	2020	3.448488	Petrol	Automatic	1	17.01	2950000.0	Ahmedabad
...	...	...	...	...	...	...	...	...	...
4563	Hyundai I . Spotz	2017	3.986997	Petrol	Manual	1	18.60	625000.0	Pune
4564	Hyundai I Sportz At .	2013	3.890695	Petrol	Automatic	2	15.00	420000.0	Pune
4565	Honda Jazz . Sv I Vtec	2017	3.667106	Petrol	Manual	2	18.70	565000.0	Pune
4566	Honda City . V Mt Exclusive	2010	3.676221	Petrol	Manual	1	17.00	375000.0	Pune
4567	Honda City Sv Mt	2014	3.979676	Petrol	Manual	1	17.40	560000.0	Pune

```
Used_Cars = pd.concat([data_new, df1], axis=1) #Combine both the datasets
```

```
Used_Cars  #Display the extracted dataset
```

Index	Model	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Price	Location	Brand
0	Kia Seltos Htx G	2021	3.280108	Petrol	Manual	1	16.80	1425000.0	Ahmedabad	Kia
1	Tata Harrier Xz Dark Edition Bsiv	2019	3.711400	Diesel	Manual	1	17.00	1700000.0	Ahmedabad	Tata
2	Tata Indica Ev Lx	2012	3.637433	Diesel	Manual	1	25.00	131000.0	Ahmedabad	Tata
3	Jeep Compass . Sport Dct	2022	2.720446	Petrol	Automatic	1	14.10	1850000.0	Ahmedabad	Jeep
4	Volkswagen Tiguan Allspace motion	2020	3.448488	Petrol	Automatic	1	17.01	2950000.0	Ahmedabad	Volkswagen
...	...	...	...	...	...	...	...	...	...	...
4563	Hyundai I . Spotz	2017	3.986997	Petrol	Manual	1	18.60	625000.0	Pune	Hyundai
4564	Hyundai I Sportz At .	2013	3.890695	Petrol	Automatic	2	15.00	420000.0	Pune	Hyundai
4565	Honda Jazz . Sv I Vtec	2017	3.667106	Petrol	Manual	2	18.70	565000.0	Pune	Honda
4566	Honda City . V Mt Exclusive	2010	3.676221	Petrol	Manual	1	17.00	375000.0	Pune	Honda
4567	Honda City Sv Mt	2014	3.979676	Petrol	Manual	1	17.40	560000.0	Pune	Honda

4568 rows × 10 columns

Label encoding:

```
#Let's use Label encoder for encoding some of the columns

l1 = ['Transmission','Fuel','Make_Year']

#Let's use Label Encoder method

from sklearn.preprocessing import LabelEncoder  #importing library

le = LabelEncoder()  #calling function

for i in l1:
    Used_Cars[i]= le.fit_transform(Used_Cars[i].values.reshape(-1,1))
Used_Cars.head()
```

	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Price	Location	Brand
0	18	3.280108	4	1	1	16.80	1425000.0	Ahmedabad	Kia
1	16	3.711400	1	1	1	17.00	1700000.0	Ahmedabad	Tata
2	9	3.637433	1	1	1	25.00	131000.0	Ahmedabad	Tata
3	19	2.720446	4	0	1	14.10	1850000.0	Ahmedabad	Jeep
4	17	3.448488	4	0	1	17.01	2950000.0	Ahmedabad	Volkswagen

Get dummies:

```
#Get dummies
l2=pd.get_dummies(Used_Cars['Brand'])

#Concat with main dataframe by dropping workclass dataframe
Used_Cars=pd.concat([Used_Cars.drop('Brand',axis=1),l2],axis=1)
#Get dummies
l3=pd.get_dummies(Used_Cars['Location'])

#Concat with main dataframe by dropping workclass dataframe
Used_Cars=pd.concat([Used_Cars.drop('Location',axis=1),l3],axis=1)
Used_Cars
```

	Make_Year	Driven_Kilometers	Fuel	Transmission	Owner(s)	Mileage	Price	Audi	Bmw	Chevrolet	...	Chennai	Gurgaon	Hyderabad	Jaipur	Kol
0	18	3.280108	4	1	1	16.80	1425000.0	0	0	0	...	0	0	0	0	
1	16	3.711400	1	1	1	17.00	1700000.0	0	0	0	...	0	0	0	0	
2	9	3.637433	1	1	1	25.00	131000.0	0	0	0	...	0	0	0	0	
3	19	2.720446	4	0	1	14.10	1850000.0	0	0	0	...	0	0	0	0	
4	17	3.448488	4	0	1	17.01	2950000.0	0	0	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4563	14	3.986997	4	1	1	18.60	625000.0	0	0	0	...	0	0	0	0	
4564	10	3.890695	4	0	2	15.00	420000.0	0	0	0	...	0	0	0	0	

Splitting features and labels

```
X = Used_Cars.drop(columns = 'Price') #Features
Y = Used_Cars['Price'] #Label
```

```
#let's check for our dimensions after splitting the data

print('Features dimension:\t',X.shape,'\nLabel Dimension:\t',Y.shape)
```

```
Features dimension:      (4568, 47)
Label Dimension:         (4568,)
```

Scaling data

```
#Scaling the data
from sklearn.preprocessing import StandardScaler
Scaler = StandardScaler()

X_scaled = Scaler.fit_transform(X)
```

Finding the Best Random State

```
from sklearn.linear_model import LinearRegression

maxR2_Score = 0
maxRS = 0

for i in range(200):
    x_train,x_test,y_train,y_test = train_test_split(X_scaled,Y,test_size = 0.20,random_state = i)
    LR = LinearRegression()
    LR.fit(x_train,y_train)
    predrf = LR.predict(x_test)
    Score = r2_score(y_test,predrf)
    if Score>maxR2_Score:
        maxR2_Score = Score
        maxRS = i

print('The best accuracy is ',maxR2_Score, ' with Random State ',maxRS)
```

```
The best accuracy is  0.7599311744665296  with Random State  88
```

```
#Let's split our dataset for training and testing purpose

x_train,x_test,y_train,y_test = train_test_split(X_scaled, Y, test_size =0.20, random_state = maxRS)
```

Model Building

```
#Importing all required Libraries that will be used for building a model

from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Lasso,Ridge
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeRegressor
```

```
rf=RandomForestRegressor()
kn=KNeighborsRegressor()
gb=GradientBoostingRegressor()
dt = DecisionTreeRegressor(max_features='auto')
ls=Lasso()
rd=Ridge()

model=[rf,kn,gb,dt,ls,rd]
kf = KFold(n_splits=5, random_state=43, shuffle=True)

train=[]
test=[]
cv=[]
```

```
for m in model:
    m = m.fit(x_train,y_train)
    pred_train=m.predict(x_train)
    pred_test=m.predict(x_test)
    train_score=r2_score(y_train,pred_train)
    train.append(train_score*100)
    test_score=r2_score(y_test,pred_test)
    test.append(test_score*100)
    print(m)
    print('R Squared (R2): ',test_score*100)
    print('Mean Squared Error (MSE): ',mean_squared_error(y_test,pred_test))
    print('Root Mean Squared Error (RMSE): ',np.sqrt(mean_squared_error(y_test, pred_test)))
    print('Mean Absolute Error (MAE): ',mean_absolute_error(y_test,pred_test))
    score=cross_val_score(m,X_scaled,Y,cv=kf)
    cv.append(score.mean()*100)
    plt.figure(figsize=[15,6])
    sns.scatterplot(x=pred_test, y=y_test)
    plt.xlabel('Predicted Values')
    plt.ylabel('Actual Values')
    plt.title('Predicted vs. Actual Values')
    plt.show()
```

```
RandomForestRegressor()
R Squared (R2):  89.88094131140285
Mean Squared Error (MSE):  160715539634.9257
Root Mean Squared Error (RMSE):  400893.42677939445
Mean Absolute Error (MAE):  183436.48027612796
```

```
----------------------------------------------------------
KNeighborsRegressor()
R Squared (R2):  76.12863381520887
Mean Squared Error (MSE):  379136006250.74506
Root Mean Squared Error (RMSE):  615740.2100324008
Mean Absolute Error (MAE):  320710.36214442016
```

Saving best model:

```
#Let's save our model for future predictions

import joblib

joblib.dump(rf,'Used_Car_Price_Prediction.obj')
```

```
['Used_Car_Price_Prediction.obj']
```

Conclusion:

- In this paper, we built several regression models to predict the selling price of cars by given some of the cars features. We evaluated and compared each model to determine the one with highest performance. We also looked at how some models rank the features according to their importance. In this paper, we followed the data science process starting with getting the data, then cleaning and pre-processing the data, followed by exploring the data and building models, then evaluating the results.
- As a recommendation, we advise to use this model (or a version of it trained with more recent data) by car market who want to get an idea about car price. The model can be used also with datasets that covered areas provided that they contain the same features. We also suggest that people take into consideration the features that were deemed as most important as seen in the previous section; this might help them estimate the car price is better.