CAR PRICE PREDICTION PROJECT

Submitted by:

CHETHANA M

ACKNOWLEDGMENT

I express.my sincere gratitude to **Flip Robo Technologies** for giving me this opportunity to carry out the project work.

A special thanks to my mentor **Mohd Kashif** for guiding me in completing this project and being available to resolve my doubts whenever I raise any tickets.

I also would love to take this moment to thank **DataTrained** for giving me all the knowledge about how to build an effective machine learning model and providing this opportunity to work as intern in Flip Robo Technologies.

INTRODUCTION

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 phase :

1. Data Collection phase
2. Model building phase

Data Collection Phase : Here I have scraped 5,231 used cars data. Data of used cars is scraped the from Cars24 website. Using Selinium the data is scraped from the website for different locations like **Bengaluru, Hyderabad, New Delhi, Noida, Gurgaon, Ahmedabad, Chennai** and **Mumbai**. The number of columns for data is limited to be 10, it includes columns like Brand, model, variant, manufacturing year, driven kilometers, Automatic/Manual, fuel, number of owners, location and at last target variable Price of the car. Few Brands that are included in the data are **Maruti, Hyundai, Nissan, Tata, Renault, Honda, Toyota, Kia, Mahindra, Skoda, Ford, Volkswagen, Mg, Datsun, Jeep, Mercedes Benz, Audi, Fiat, Bmw.**

Model Building Phase : After collecting the data, machine learning model is built. Before model building all the data pre-processing steps are done. Different models with different hyper parameters are built and then the best model is selected. In the model building phase following steps are included.

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building

5. Model Evaluation
6. Selecting the best model

Technical goals :

• Which variables are important to predict the price of the used car?

• How these variables are responsible to describe the price of the car?

Our main aim today is to make a model which can be used to predict the price of the used car based on other variables. We are going to use Linear Regression, Decision Tree Regressor, K Neighbors Regressor, SVR, Lasso and Ridge to build the different models for this dataset and see which model gives us a good accuracy.

Analytical Problem Framing

In this project, prediction will be made on what is the price of used cars using given explanatory variables that cover many aspects of the cars like model, brand, driven kilometers, number of owners, fuel etc. The goal of this project is to create a model that is able to accurately predict the price of a used car under the given features.

Here we can observe certain features of the cars in details and dataset includes 10 features such as,

Price	Price of the used car
Brand	Brand of the car
Model	Model of the car
Automatic	Whether the car is automatic or manually operated
Variant	Specific variant of the car under the given model
Km	Number of Kilometers driven by car by now
Location	From which city the car is
Man_year	Manufactured year of the car
Fuel	Whether the car uses Petrol or Diesel as fuel
Owner	Number of owners the car had by now

Data contains details of 5234 cars each having 10 variables.
Data contains Null values in Owner column.
Extensive EDA has to be performed to gain relationships of important variables and label.
Data contains numerical as well as categorical variable which needs to handled accordingly.
Machine Learning models is needed to be built to predict the price of the car.
We also need to find important features which affect the price positively or negatively.

In [2]:
```
data=pd.read_csv('Car_Pricing_final_data.csv')
```

In [3]:
```
df=pd.DataFrame(data)
df
```

Out[3]:

	Unnamed: 0	Unnamed: 0.1	Brand	Model	Automatic	Variant	Km	Location	Man_year	Fuel	Owner	Price
0	0	0	Maruti	Alto 800 LXI MANUAL	MANUAL	LXI MANUAL	15,999	Bengaluru	2013	Petro	1st	2,74,599
1	1	1	Maruti	Ritz VXI MANUAL	MANUAL	VXI MANUAL	28,022	Bengaluru	2011	Petro	1st	3,77,999
2	2	2	Hyundai	AURA SX (O) MT	MT	SX (O) MT	3,382	Bengaluru	2022	Petro	1st	8,07,099
3	3	3	Hyundai	i20 MAGNA O 1.2 MANUAL	MANUAL	MAGNA O 1.2 MANUAL	55,910	Bengaluru	2014	Petro	1st	4,42,299
4	4	4	Maruti	Swift ZXI MANUAL	MANUAL	ZXI MANUAL	47,003	Bengaluru	2012	Petro	1st	4,93,799
...	...	...	...	...	...	...	...	...	...	...	...	...
5229	5229	478	Hyundai	Xcent SX 1.2 MANUAL	MANUAL	SX 1.2 MANUAL	22,364	Mumbai	2016	Petro	1st	5,66,599
5230	5230	479	Ford	Ecosport 1.5 TREND TI VCT MANUAL	MANUAL	1.5 TREND TI VCT MANUAL	45,877	Mumbai	2016	Petro	2nd	5,18,999
5231	5231	480	Ford	Ecosport 1.5 TREND TI VCT MANUAL	MANUAL	1.5 TREND TI VCT MANUAL	45,877	Mumbai	2016	Petro	2nd	5,18,999
5232	5232	481	Maruti	Wagon R 1.0 LXI CNG MANUAL	MANUAL	LXI CNG MANUAL	44,517	Mumbai	2017	Petro	NaN	4,61,999
5233	5233	482	Ford	Ecosport 1.5TITANIUM TDCI MANUAL	MANUAL	1.5TITANIUM TDCI MANUAL	76,294	Mumbai	2014	Diese	2nd	4,99,599

5234 rows × 12 columns

Data Pre-processing Done :

Initially the data is scraped from Car24 website using selenium for the locations Bengaluru, Hyderabad, New Delhi, Noida, Gurgaon, Ahmedabad, Chennai and Mumbai. There is missing data in one variable, outliers were present and data was skewed, columns that doesn't create any impact on output variable. Treated the data for its skewness using 'Yeo-Johnson' method. The outliers were removed from continuous data using z-score method. Multicollinearity was checked and the column that is creating multicollinearity was dropped from the dataset.

Data Inputs- Logic- Output :

As discussed before there are 10 variables in the dataset, 1 output (Price) variable and 9 input variables.

INPUT VARIABLES : Brand, Model, variant, Man_year, Km, Automatic, Fuel, Owners, Location

OUTPUT VARIABLE : Price

About the Algorithms used :

The major aim in this project is to predict if the applicant pays back the loan or not in the given period of time based on the features using some of the regression and classifier algorithms.

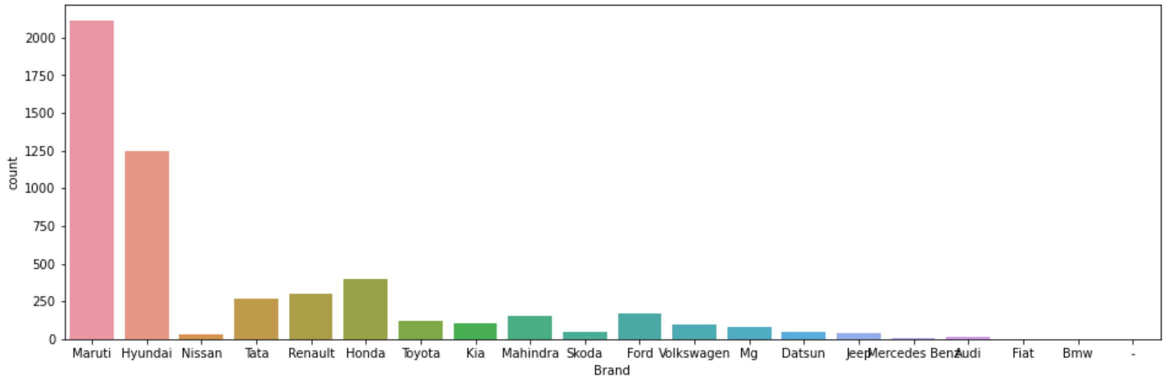
1. Linear Regression
2. Decision Tree Regressor
3. KNeighbors Regressor
4. SVR
5. Lasso
6. Ridge

Machine Learning Packages are used for in this Project

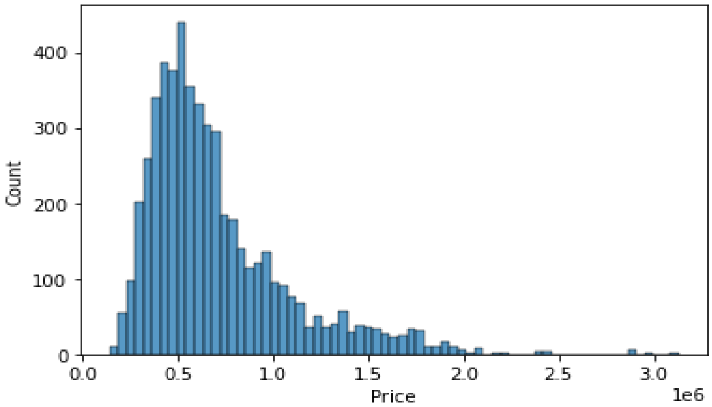
The data was scraped from Cars24 website which has 5234 instances and 10 variables, it is stored in the csv file, this csv file was uploaded to Jupyter and read using pandas library. Once the dataset is read DataFrame is created and further EDA process is done on the dataset using different functions available in pandas.

The Seaborn and Matplotlib are used to plot the different graphs and understand the relationship between each variable and output variable.

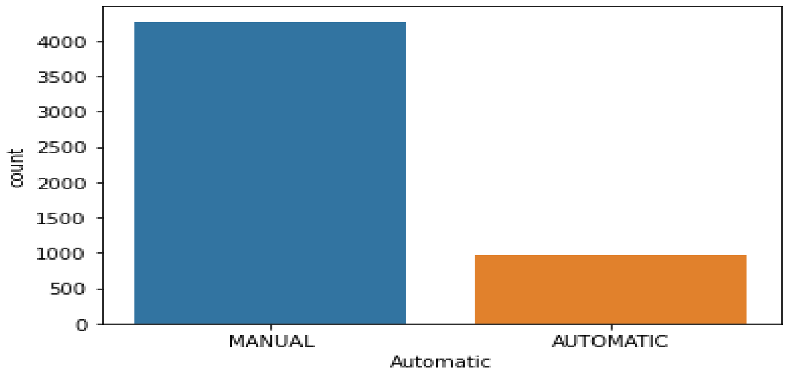
EDA and Visualization



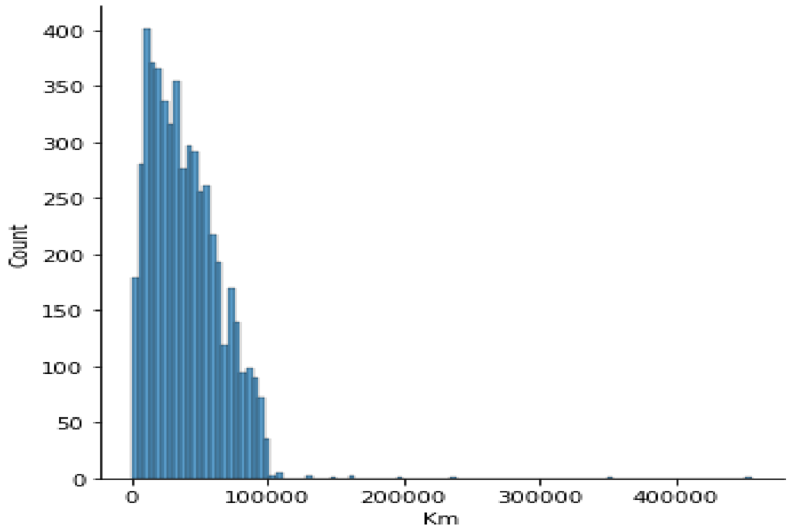
From the plot above we can observe that Highest number of cars are from Maruthi and Hyundai brands.



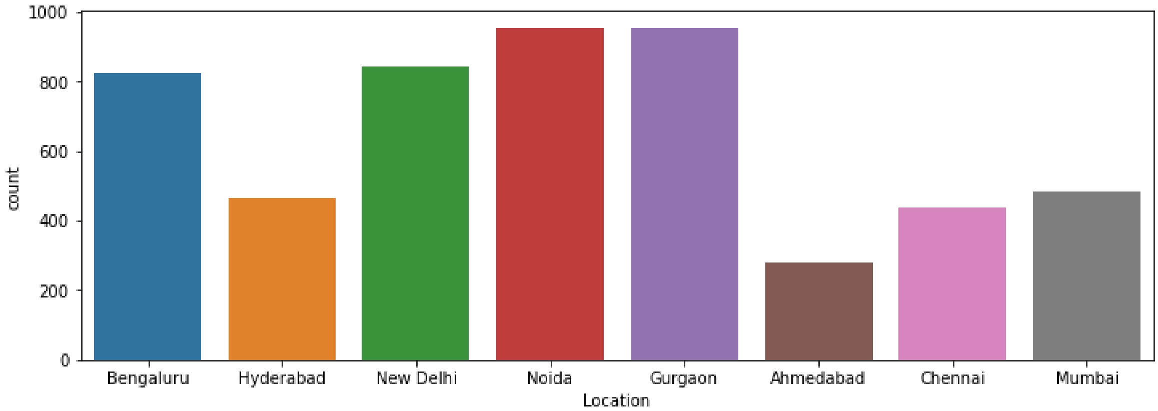
Above plot shows us the distribution of Price, it is understandable that most of the cars price ranges from 3 to 7 lakhs.



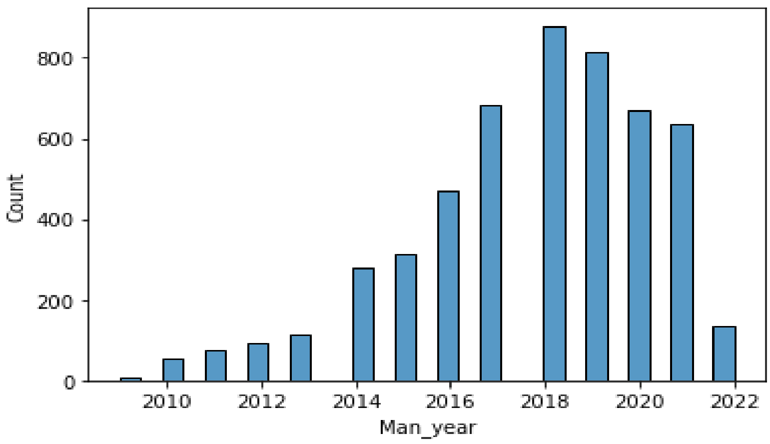
We can see from the above data that 4271 cars are of Manual operation and only 963 cars are Automatic.



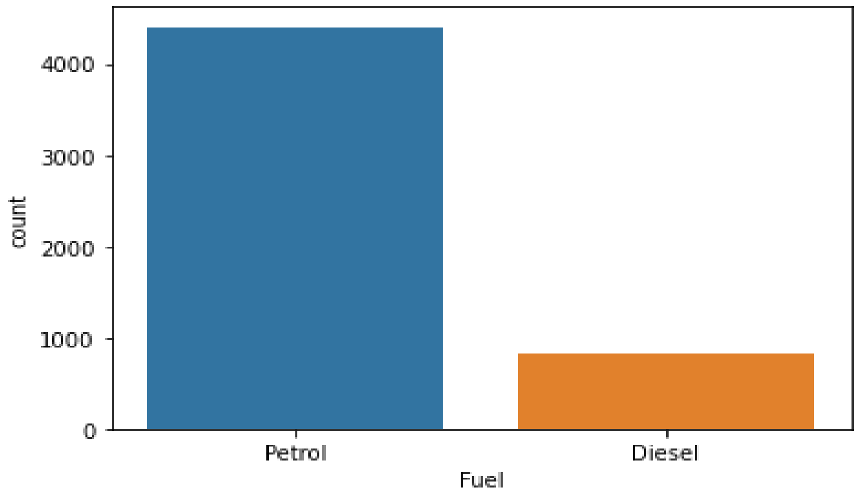
From the above plot it is clear that maximum number of cars driven Kms falls between 0 to 100000.



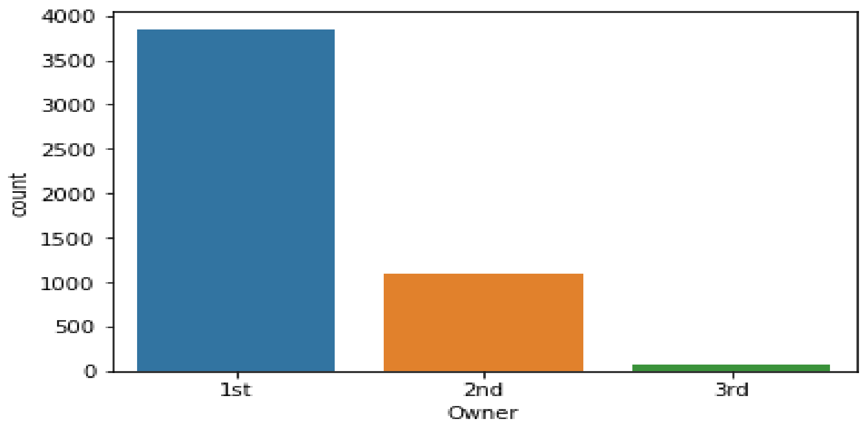
Data consists of 954 and 952 cars details which belongs to Gurgaon and Noida, 842 cars from New Delhi, 824 cars from Bengaluru, 483 cars from Mumbai, 465 cars from Hyderabad, 436 cars from Chennai and 278 cars from Ahmedabad.



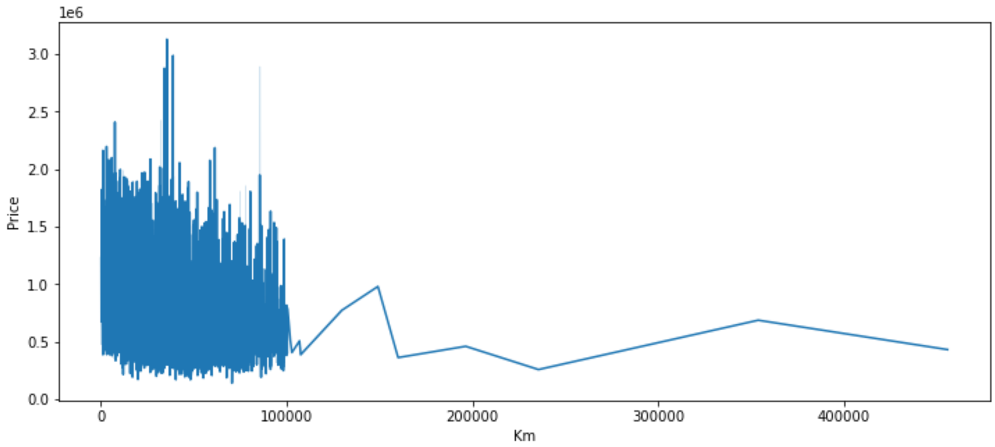
Most of the cars in the dataset are manufactured in the years 2017 to 2021 as we can observe in the above plot.



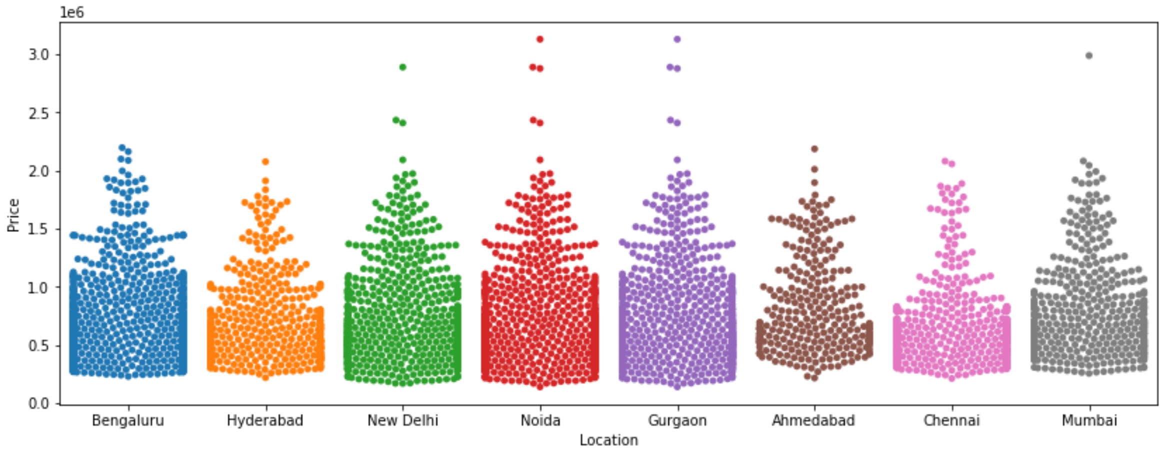
As we can observe from the above plot and data 4402 cars are Petrol cars and 832 are Diesel cars.



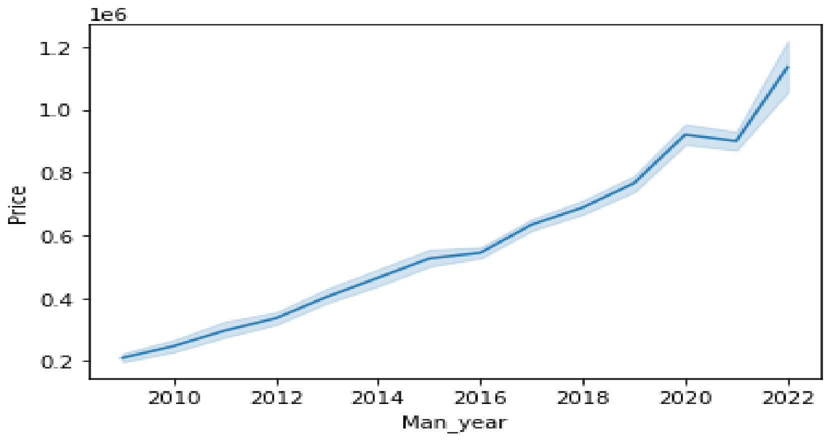
The data has 3851 cars from 1st owner, 1090 cars from 2nd and only 75 cars from 3rd owner.



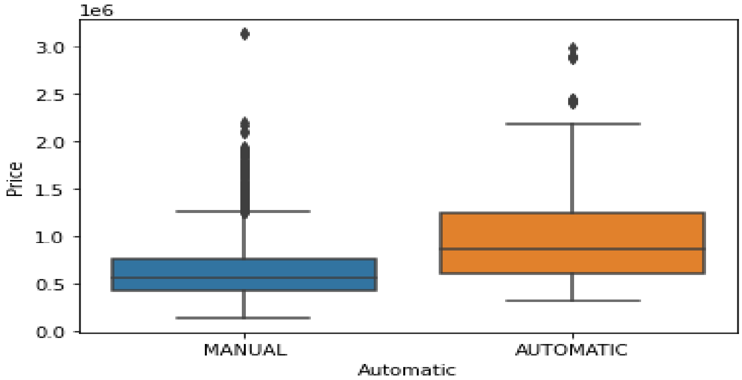
From the above plot we can observe that the price is high for those cars whose driven Km is less and as the driven Km increases the price decreases.



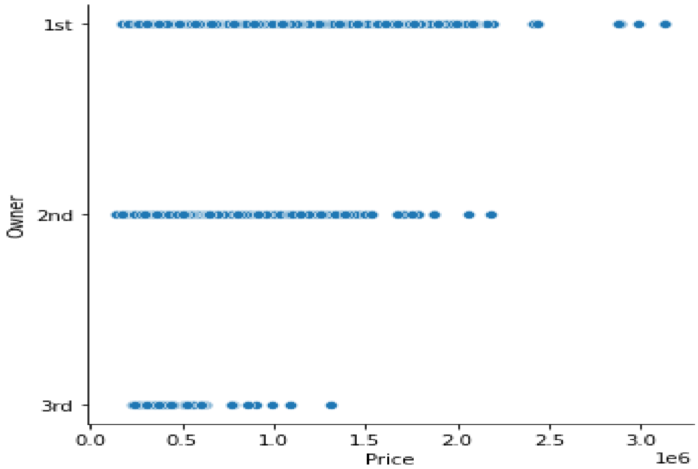
The price is reaching high for those cars in Noida and Gurgaon.



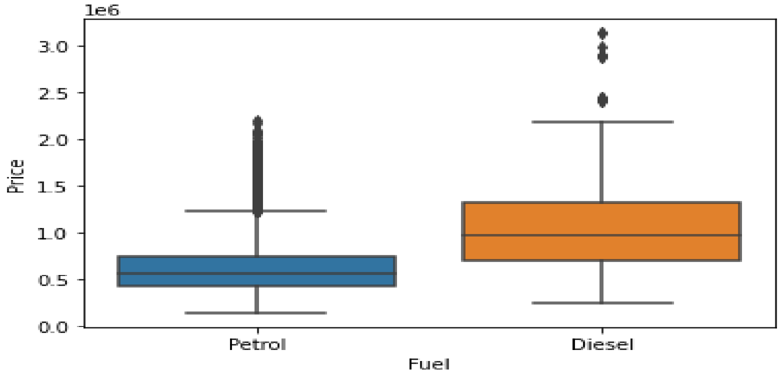
From the above plot we can observe that the price is increasing for the cars which are manufactured in the recent years, the older cars' price is lesser.



We can see that the min and max price is higher for the cars which have automatic operations compared to manual cars.



We can observe from the above plot that the price is higher for those cars which are sold from 1st owner than that of 2nd and owners.



The price is higher for the cars operate with Diesel than the Petrol cars.

As we discussed before there are some missing values in the Owner column, we can now handle these missing values by simple imputer as shown below.

```
In [100]: df.isnull().sum()
```

```
Out[100]: Brand          0
          Model          0
          Automatic      0
          Variant        0
          Km             0
          Location       0
          Man_year       0
          Fuel           0
          Owner        218
          Price          0
          dtype: int64
```

Only the column Owner has 218 mussing values

```
In [101]: from sklearn.impute import SimpleImputer
          imputer = SimpleImputer(missing_values=np.nan,strategy='most_frequent')
          df['Owner']=imputer.fit_transform(df['Owner'].values.reshape(-1,1))
```

Handling the Object type data using Label Encoder

```
In [103]: # Encoding the columns that has object dtype
          Object_columns=df.select_dtypes(include=[object])
          Object_columns.head(1)
```

Out[103]:

	Brand	Model	Automatic	Variant	Location	Fuel	Owner
0	Maruti	Alto 800 LXI	MANUAL	LXI	Bengaluru	Petrol	1st

```
In [104]: from sklearn.preprocessing import LabelEncoder
          le = LabelEncoder()
```

```
In [105]: for columns in Object_columns:
              df[columns] = le.fit_transform(df[columns])
```

```
In [106]: df.dtypes
```

```
Out[106]: Brand        int32
          Model        int32
          Automatic    int32
          Variant      int32
          Km           int64
          Location     int32
          Man_year     int64
          Fuel         int32
          Owner        int32
          Price        int64
          dtype: object
```

Hence all the data in dataset is encoded and we are left with integer type data in all the variables.

```
In [108]: df.describe()
```

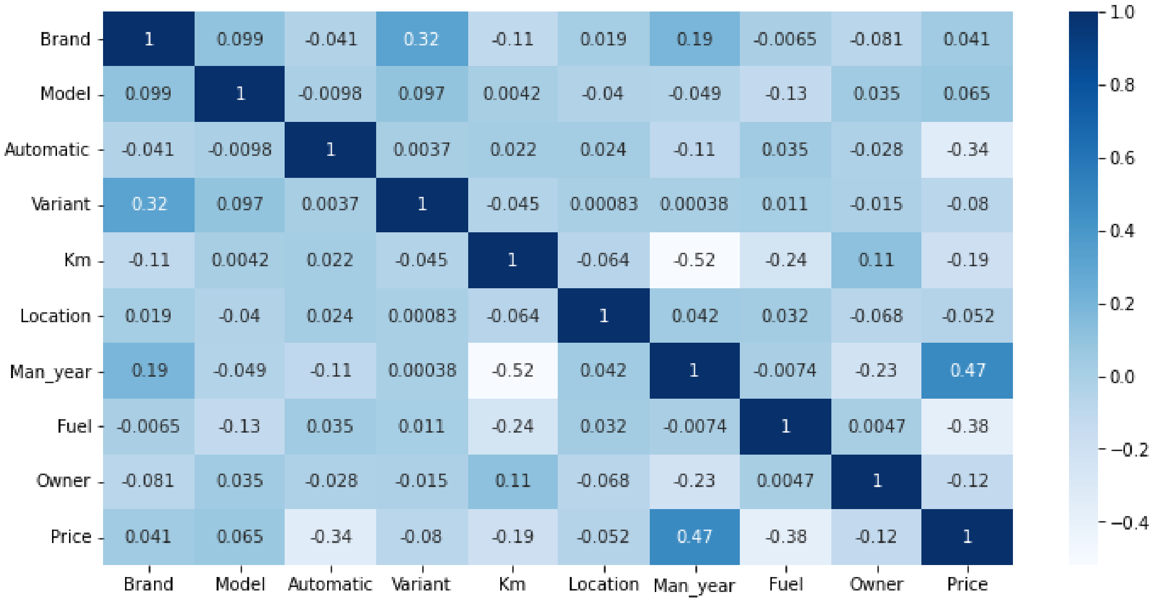
Out[108]:

	Brand	Model	Automatic	Variant	Km	Location	Man_year	Fuel	Owner	Price
count	5234.000000	5234.000000	5234.000000	5234.000000	5234.000000	5234.000000	5234.000000	5234.000000	5234.000000	5.234000e+03
mean	10.214941	379.017004	0.816011	399.334734	39770.272640	3.926060	2017.755063	0.841039	0.236912	7.022958e+05
std	3.561938	258.391964	0.387512	208.899875	26145.824706	2.269464	2.595399	0.365674	0.457694	3.756248e+05
min	0.000000	0.000000	0.000000	0.000000	188.000000	0.000000	2009.000000	0.000000	0.000000	1.390990e+05
25%	7.000000	134.000000	1.000000	246.000000	18704.500000	2.000000	2016.000000	1.000000	0.000000	4.464490e+05
50%	11.000000	346.500000	1.000000	410.000000	35799.500000	4.000000	2018.000000	1.000000	0.000000	5.998990e+05
75%	11.000000	634.000000	1.000000	581.000000	57162.750000	6.000000	2020.000000	1.000000	0.000000	8.393490e+05
max	19.000000	843.000000	1.000000	749.000000	455601.000000	7.000000	2022.000000	1.000000	2.000000	3.131049e+06

Key Observations :

- Mean > median (50th percentile) in the columns Model, Km, Price hence the data in these columns are skewed
- We can observe that there is a huge gap between 75th percentile and max in the columns Brand, Model, Variant, Km, Owner, Price and hence the data in these columns has outliers

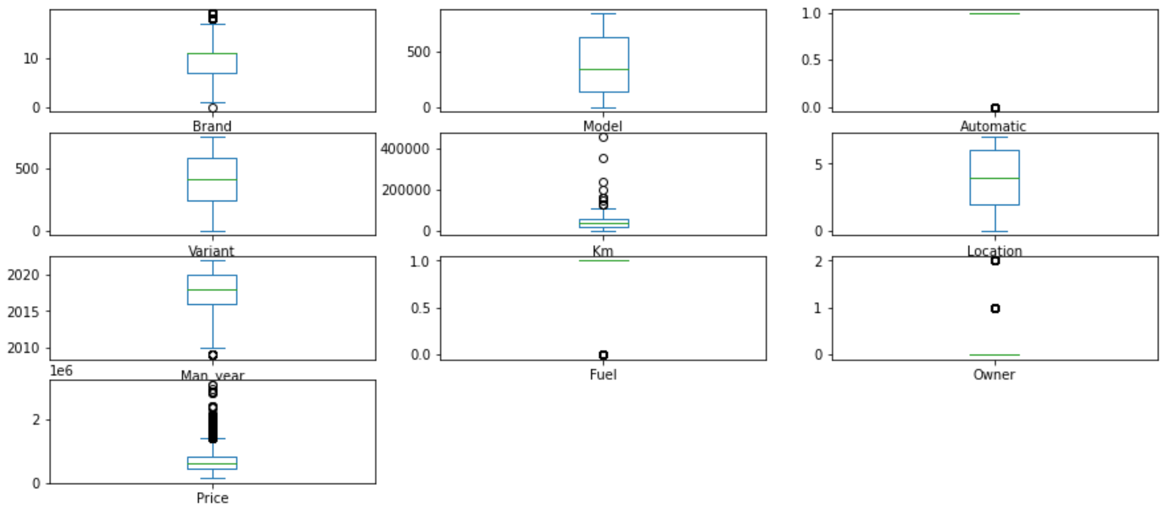
- In the columns Brand, Model, Variant, Km, Man_year, Price we can observe that there is a high gap between mean and std, hence the data is highly spreaded.



Key Observations :

- Price has a better correlation with Man_year, it have has a good correlation with Brand and Model
- Price has least correlation with Fuel
- Man_year has negative correlation with Km
- Brand has a better correlation with Variant compared to others

Removing Outliers from the data :



```
In [119]: outliers_col=['Km','Price']
          for col in outliers_col:
              from scipy.stats import zscore
              z=np.abs(zscore(df[col]))
```

```
In [120]: Threshold=3
          print(np.where(z>3))
```

```
(array([  17,   38,   76,  141,  188,  205,  225,  269,  445,  456,  538,
        552,  567,  600,  630,  887, 1216, 1232, 1297, 1370, 1461, 1525,
       1672, 1722, 1739, 1816, 1869, 1997, 2085, 2156, 2161, 2249, 2321,
       2367, 2405, 2453, 2498, 2622, 2863, 3009, 3010, 3027, 3084, 3090,
       3117, 3234, 3272, 3277, 3503, 3571, 3614, 3841, 3899, 3943, 3963,
       4095, 4131, 4304, 4329, 4334, 4336, 4373, 4532, 4712, 4970, 5037,
       5042, 5112, 5135, 5163, 5191, 5194, 5195, 5199], dtype=int64),)
```

```
In [121]: df_new=df[(z<3)]
          df_new
```

Out[121]:

	Brand	Model	Automatic	Variant	Km	Location	Man_year	Fuel	Owner	Price
0	11	21	1	365	15999	1	2013	1	0	274599
1	11	559	1	581	28022	1	2011	1	0	377999
2	7	19	1	523	3382	1	2022	1	0	807099
3	7	840	1	387	55910	1	2014	1	0	442299
4	11	646	1	725	47003	1	2012	1	0	493799

The outliers present in the data is removed using z-score method and framed a new data frame called df_new, now we can calculate the % of data lost in this process.

```
In [122]: df.shape
Out[122]: (5234, 10)
```

- There are 5234 rows and 10 columns in the old dataset

```
In [123]: df_new.shape
Out[123]: (5160, 10)
```

- There are 5160 rows and 10 columns in new dataset after removing outliers.

```
In [124]: # Now we can check for data Loss

          Dataloss = (((5234-5160)/5234)*100)
          Dataloss
Out[124]: 1.4138326327856323
```

We can observe that data loss in z-score method after removing outliers of 1.4% which is less than 10%, which tells that it is appropriate to remove outliers using this method.

Handing the skewed data :

```
In [125]: df_new.skew()
Out[125]: Brand        0.538537
          Model        0.162156
          Automatic   -1.697110
          Variant     -0.253692
          Km           1.706168
          Location    -0.091972
          Man_year    -0.749073
          Fuel        -1.921112
          Owner        1.662649
          Price        1.248219
          dtype: float64
```

```
In [126]: # we can observe that there is skewness present in the data in case of Km, Fuel, Owner and price

          from sklearn.preprocessing import PowerTransformer
          scaler = PowerTransformer(method='yeo-johnson')
```

```
In [127]: df_new[['Km','Fuel','Owner']]=scaler.fit_transform(df_new[['Km','Fuel','Owner']].values)
```

Hence the skewness is removed from the data using '**yeo-johnson**' method as shown in the above figure.

Splitting x and y data :

```
In [129]: #We can now check for multicolinearity

          x=df_new.drop(['Price'],axis=1)
          x.sample()
```

Out[129]:

	Brand	Model	Automatic	Variant	Km	Location	Man_year	Fuel	Owner
23	11	639	1	581	1.309691	1	2012	0.426133	1.854273

```
In [130]: y=df_new['Price']
          y

Out[130]: 0       274599
          1       377999
          2       807099
          3       442299
          4       493799
                   ...
          5229    566599
          5230    518999
          5231    518999
          5232    461999
          5233    499599
          Name: Price, Length: 5160, dtype: int64

In [131]: y.shape,x.shape

Out[131]: ((5160,), (5160, 9))
```

Once we split x and y data, we can see that the shape of y data is 5160 and shape of x data is 5160 * 9.

Checking for multicollinearity :

```
In [132]: from statsmodels.stats.outliers_influence import variance_inflation_factor

In [133]: def vif_calc():
              vif=pd.DataFrame()
              vif["VIF Factor"]=[variance_inflation_factor(x.values,i)for i in range(x.shape[1])]
              vif["Features"]=x.columns
              print(vif)

In [134]: vif_calc()

             VIF Factor   Features
          0   10.676823      Brand
          1    3.255841      Model
          2    5.693502  Automatic
          3    5.257862    Variant
          4    1.101183         Km
          5    4.037878   Location
          6   19.934223   Man_year
          7    1.086628       Fuel
          8    1.027014      Owner
```

As we can observe from the above table Brand and Man_year are creating multicollinearity in the data, from the correlation data we can observe that Brand is giving lesser contribution to Price than Man_year hence we can drop off Brand at this stage.

Splitting train and test data :

```
In [137]: # Splitting the data for training and testing the model

          x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25,random_state=42)

In [138]: x_train.shape,x_test.shape,y_train.shape,y_test.shape

Out[138]: ((3870, 8), (1290, 8), (3870,), (1290,))
```

As we discussed before we are going to use the below algorithms and build the model

- Linear Regression
- Decision Tree Regressor
- KNeighbors Regressor
- SVR
- Lasso
- Ridge

```
In [139]: from sklearn.linear_model import Ridge, Lasso
          lm=LinearRegression()
          DTR=DecisionTreeRegressor()
          KNR=KNeighborsRegressor()
          svr=SVR()
          La=Lasso()
          rd=Ridge()
```

```
In [140]: model=[lm,DTR,KNR,svr,La,rd]
          for m in model:
              m.fit(x_train,y_train)
              m.score(x_train,y_train)
              pred=m.predict(x_test)
              print("Score of ",m,"is :",m.score(x_train,y_train))
              print("r2 score :",r2_score(y_test,pred))
              print("Errors of ",m,"is")
              print("Mean absolute error :",mean_absolute_error(y_test,pred))
              print("Mean squared error :",mean_squared_error(y_test,pred))
              print("Root mean squared error :",np.sqrt(mean_squared_error(y_test,pred)))
              print('\n')
```

Scores of these models were as :

Model	Accuracy	R2-score
LinearRegression()	46.2%	46%
DecisionTreeRegressor()	100%	88.5%
KNeighborsRegressor()	93.1%	88.2%
SVR()	-6.2%	-6.3%
Lasso()	46.2%	46%
Ridge()	46.2%	46%

From the above table we can observe that Decision Tree Regressor and KNeighbors Regressor are the models which are giving best accuracy and r2 score, Hence we can consider these models to be best as of now.

Applying Cross Validation for the same models :

```
In [141]: #WE can now try with Cross validation for the models
          from sklearn.model_selection import cross_val_score
          model=[lm,DTR,KNR,svr,La,rd]
          for i in model:
              score=cross_val_score(i,x_train,y_train,cv=5)
              print("score of ",i,"is :",score)
              print("score mean of ",i,"is :",score.mean())
              print("score std of ",i,"is :",score.std())
              print('\n')
```

Even with CV we can observe that Decision Tree Regressor and KNeighbors Regressor is giving the best mean score of 83% and 85%.

Parameter tuning for Decision Tree Regressor :

```
In [142]: # Parameter tuning for Decision Tree regressor

          from sklearn.model_selection import GridSearchCV
          DTR=DecisionTreeRegressor()
          params={'criterion':['squared_error','friedman_mse','absolute_error','poisson']}
          grd=GridSearchCV(estimator=DTR,param_grid=params,cv=5)
          grd.fit(x_train,y_train)

Out[142]: GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
                       param_grid={'criterion': ['squared_error', 'friedman_mse',
                                                 'absolute_error', 'poisson']})
```

Best Score : 0.830567728580397
Best Param : 'criterion': 'friedman_mse'

Parameter tuning for KNeighbors Regressor :

```
In [145]: # parameter tuning for KNeighbors Regressor

          KNR=KNeighborsRegressor()
          params={'weights':['uniform','distance'],'algorithm':['auto','ball_tree','kd_tree','brute']}
          grd=GridSearchCV(estimator=KNR,param_grid=params,cv=5)
          grd.fit(x_train,y_train)

Out[145]: GridSearchCV(cv=5, estimator=KNeighborsRegressor(),
                       param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                                   'weights': ['uniform', 'distance']})
```

Best Score : 0.9112141564651894
Best Param : 'algorithm': 'brute', 'weights': 'distance'

As we can observe both the models after parameter tuning we see that KNeighbors Regressor is giving the best score of 91.12% with the parameters – algorithm = **brute** and weights = **distance,** hence we can finalize this as our final model.

Saving the final model with the above parameters :

In [148]:
```
# Saving the best model
Final_regressor=KNeighborsRegressor(algorithm='brute',weights='distance')
Final_regressor.fit(x_train,y_train)
pred=Final_regressor.predict(x_test)
print("Score: ",Final_regressor.score(x_train,y_train)*100)
print('R2_Score:',r2_score(y_test,pred)*100)
print('mean_squared_error:',mean_squared_error(y_test,pred))
print('mean_absolute_error:',mean_absolute_error(y_test,pred))
print("RMSE value:",np.sqrt(mean_squared_error(y_test,pred)))
```

```
Score:  99.99999994948516
R2_Score: 91.59486399543098
mean_squared_error: 9076222154.154293
mean_absolute_error: 57390.8022006161
RMSE value: 95269.2088460605
```

In [149]:
```
# We can save the model now
import joblib
joblib.dump(Final_regressor,'Car_Pricing.obj')
```

Out[149]:
```
['Car_Pricing.obj']
```

Now we can predict the values and compare it with the original values using the saves final model.

In [153]:
```
# Making a DataFrame of Predicted values and Original values
df1=pd.DataFrame({'Predicted values':pred.round(2),'Original values':y_test})
df1
```

Out[153]:

	Predicted values	Original values
2422	485223.44	491499
3575	716526.42	793299
669	470223.62	494599
1100	366397.59	366099
2553	963137.63	1056699
...	...	...
4609	880780.42	765699
3472	812558.46	876099
1371	814816.84	836949
3710	1287971.07	1139899
4741	964119.52	936399

1290 rows × 2 columns

We can observe that both predicted and original values to be almost near by with KNeighbors model with is giving 99.9% accuracy and r2 score as 91.5.

CONCLUSION

From the Exploratory Data Analysis, we could generate insight from the data. How each of the features relates to the target. Also, while training the model we could observe that KNeighbors Regressor was performing well with the accuracy 99.99%.

From the EDA we could also observe that target variable was highly dependent on Number or owners, fuel, manufactured year.

Few variables like Location, Km had poor correlation with target variable.

Cleaning of the data included few techniques like removing outliers, treating skewed data, dropping the variables which were creating multicollinearity.

While building the model we could observe that it is always good to build 4 to 5 models for the same train test data and compare the scores of the models then pick the best model instead of sticking onto single model.

As of now we could finalize **KNeighbors Regressor** with **algorithm = brute** and **weights = distance** as the best model which is giving **99.99%** accuracy score and **91.5** r2 score which was best compared to other models.