Car Price Prediction Project

Submitted by:

Mekhala Misra

ACKNOWLEDGMENT

I took help from following websites:

1)Geek for geeks

2)Pandas documentation

3)researchgate.net

INTRODUCTION

- ## Business Problem Framing

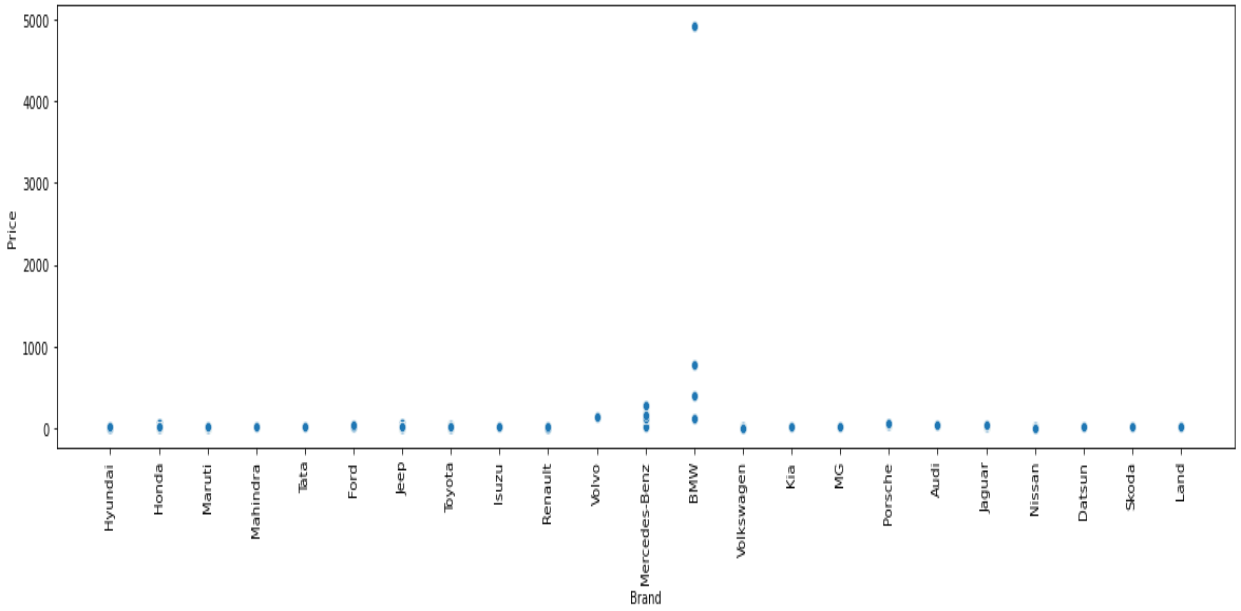
  Sudden out break of covid-19 has hit hard many businesses including used car sales market. Due to this pandemic, there are many changes in the used car prices so the old machine learning models used by companies are no longer valid for post pandemic data. So here we are working on this project in 2 phases:

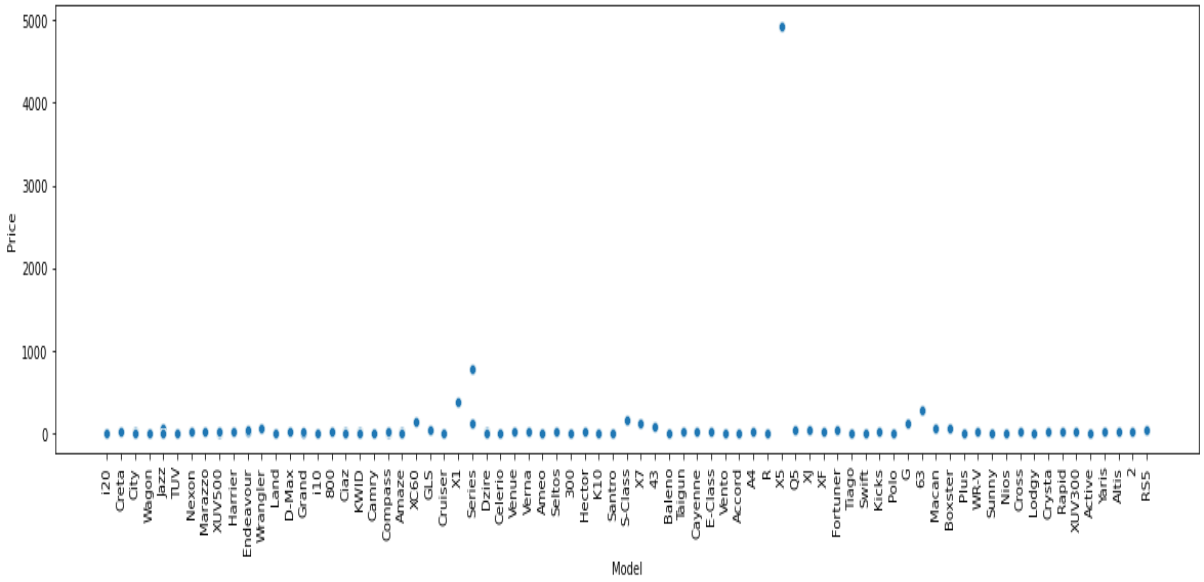
  1. Data Collection: I collected used car data from two sites, cardekho.com and cars24.com.
  2. Analysing the data and building the model.

- ## Review of Literature

  Using various scatter graph I came to certain conclusion:

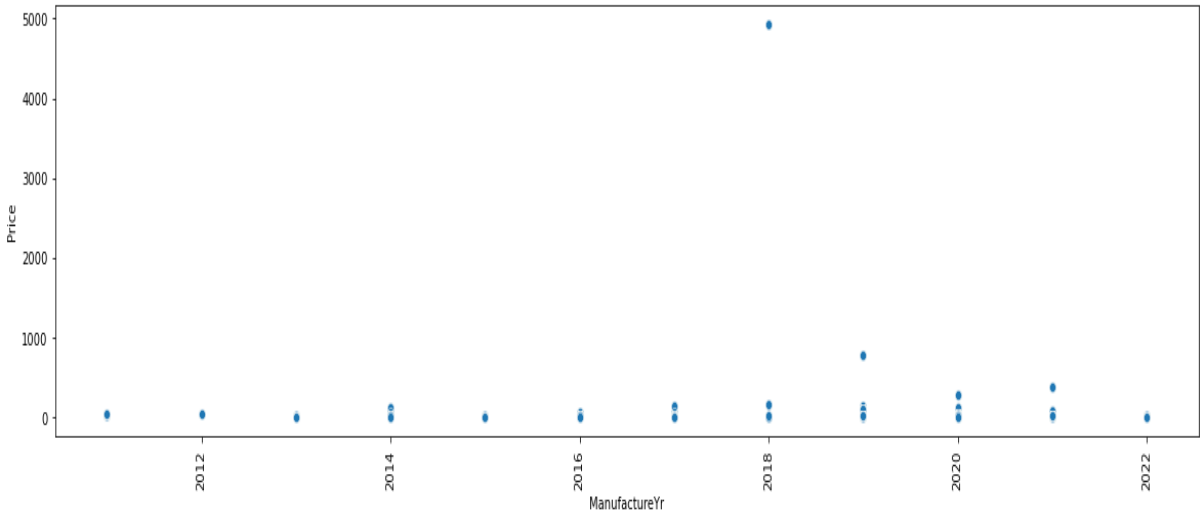
  1) Here we can see that how price varies with the car brand. BMW used cars are most expensive. The price is in lakhs.



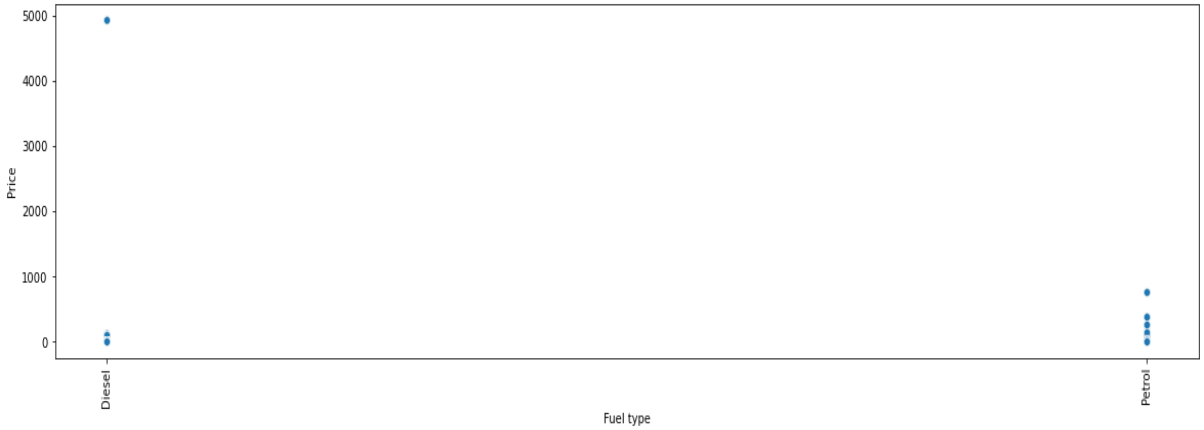


2) We can observe that x1, series and 63 model of BMW and Mercedes are the costliest even if the cars are used.

3) We can observe that newer the used car higher is the price.

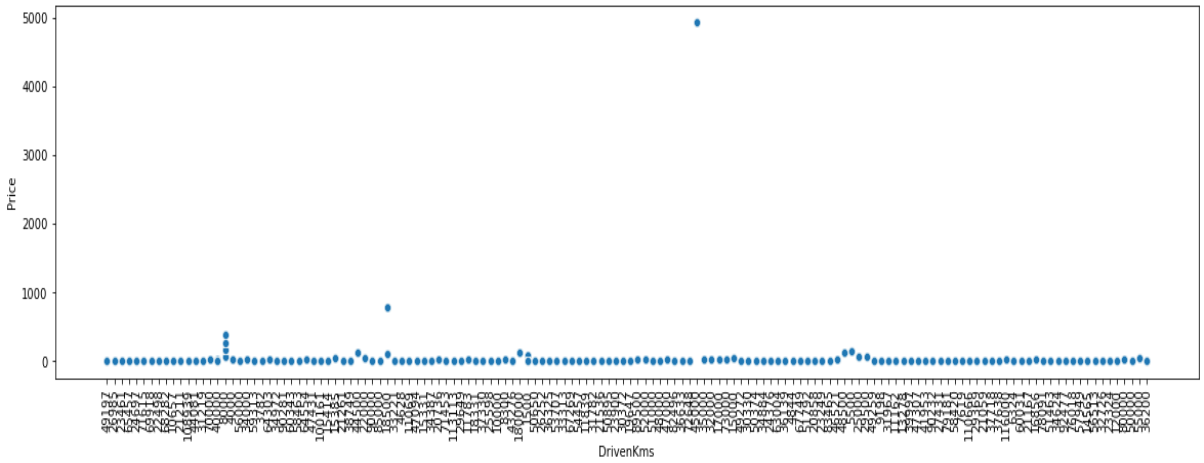


4) Petrol cars are more expensive than diesel cars.



5) Automatic cars are costlier than manual.

6) Lesser the driven kilometers higher is the cost. That means lesser the driven kilometers better the state of car so higher the price.



- ## Motivation for the Problem Undertaken

  Since the outbreak of covid-19 there were economic crisis throughout the world in such a situation all the business finances has changed so this was the biggest motivation to take up this problem and work on it and generate the models to see how this has impacted the cost of used cars.

Analytical Problem Framing

- ## Mathematical/ Analytical Modelling of the Problem

  This is a regression problem as we must find out the price of used cars using few independent variables. I used scatter plots for analysing relations between features and label price further I used heatmap and vif score to check the multicollinearity problem, checked and treated outliers using z-score technique and skewness using power transform.

- ## Data Sources and their formats

Data set is collected from cars24 and carDekho from various cities as Delhi, NCR, Mumbai, Hyderabad, Chennai, Banglore,Kolkata,Pune etc. Below is the shape of data:

```
In [60]:  1 df.shape

Out[60]: (7200, 8)
```

Following are the columns present and their respective data types:

```
In [30]:  1 df.dtypes

Out[30]: Brand           object
         Model           object
         Variant         object
         ManufactureYr    int64
         Fuel type       object
         Control         object
         DrivenKms       object
         Price          float64
         dtype: object
```

Here is the glimpse of data:

Out[2]:

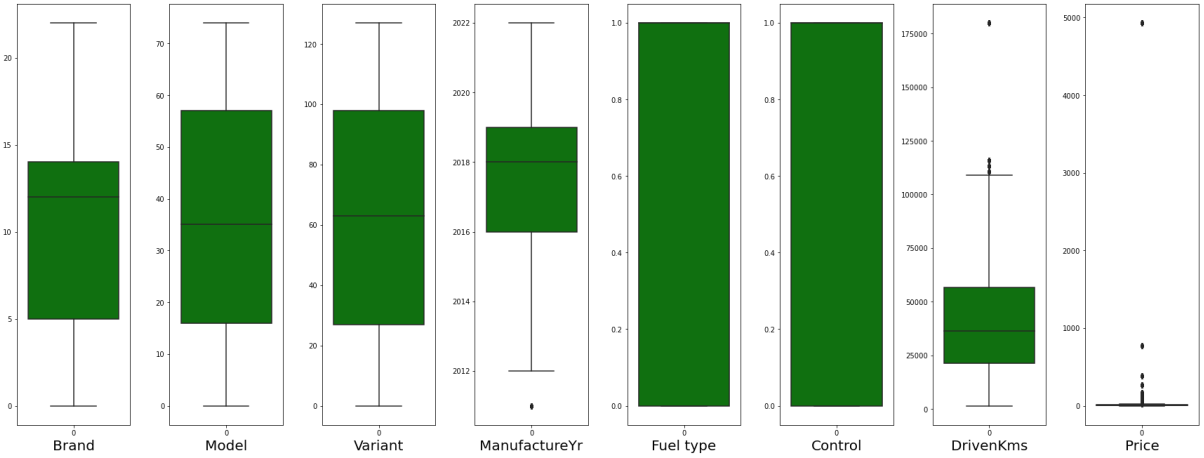
	Unnamed: 0	Brand	Model	Variant	ManufactureYr	Fuel type	Control	DrivenKms	Price
0	0	Hyundai	i20	1.4 Sportz	2017	Diesel	Manual	49,197	6.25
1	1	Hyundai	Creta	1.6 VTVT SX Plus	2018	Petrol	Manual	26,985	9.26
2	2	Honda	City	VX MT	2020	Petrol	Manual	23,461	11.23
3	3	Maruti	Wagon	LXI BS IV	2016	Petrol	Manual	65,457	3.41
4	4	Hyundai	Creta	1.6 SX Option	2019	Petrol	Manual	24,697	13.16
...	...	...	...	...	...	...	...	...	...
7195	7195	Land	2	SE	2014	Diesel	Automatic	80,000	18.50
7196	7196	Honda	City	i VTEC CVT VX	2014	Petrol	Automatic	50,000	6.88
7197	7197	Audi	RS5	Coupe	2011	Petrol	Automatic	55,000	35.00
7198	7198	Hyundai	Creta	1.6 VTVT AT SX Plus	2018	Petrol	Automatic	36,200	12.60
7199	7199	Tata	Nexon	XZ Plus DualTone Roof	2020	Petrol	Manual	40,000	9.75

7200 rows × 9 columns

• Data Pre-processing

- The dataset had no null values but the label price was as object data type so I converted it into float type.
- While analysing the data I came across that the prices of high-end cars like Mercedes and BMW were in crores but filled as lakhs so I converted such kind of prices to lakhs.
- DrivenKms feature was also object type so I converted it into float type as well.
- Few features such as Brand, Model, Variant, ManufactureYr, Fuel type and Contol are categorical in nature so applied ordinal encoding on them.
- As per heatmap multicollinearity doesnot exists but in order to be sure I calculated vif score and found out that the manufactureYr is greater than 10 but since its an important feature so could not drop it.

- Outliers are present in DrivenKms and Price but since Price is a label so we would consider outlier in only DrivenKms. So treated it using z-score technique.



Since the data loss was only 0.5% so we dropped rows having outliers.

In [61]:
```
#Calculating Data Loss
loss=((7200-40)/7200)*100
print("The data loss-",100-loss,"%")
```

```
The data loss- 0.5555555555555571 %
```

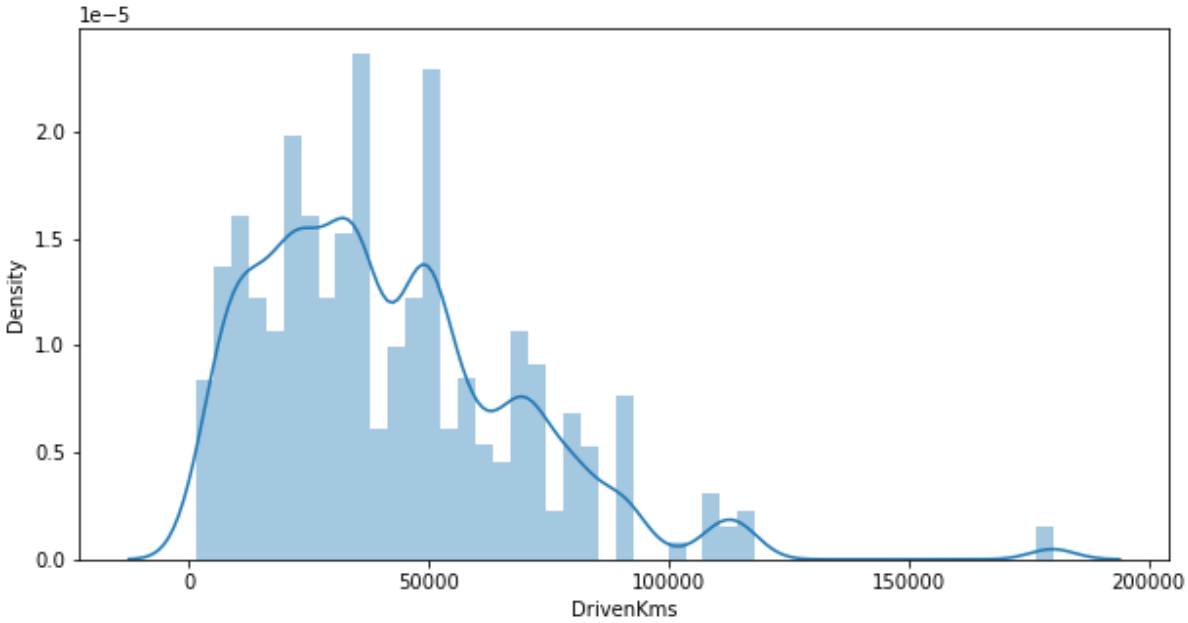
We can bear with 0.5% data loss so lets drop these outlier rows.

In [62]:
```
#removing rows with outliers
df=df.drop(df.index[i[0]])
df
```

We could observe skewness in DrivenKms. Considering the range of skewness as -0.5 to 0.5, we could observe that DrivenKms skewness value is 1.2 which is not in range (-0.5, 0.5) so we treated it using power transform method.

In [57]:
```
df.skew()
```

Out[57]:
```
Brand           0.154527
Model           0.196101
Variant        -0.008555
ManufactureYr  -0.457882
Fuel type      -1.104441
Control        -0.694034
DrivenKms       1.208040
Price          10.585203
dtype: float64
```



We fixed this by using yeo-Johnson transformation technique.

```
In [65]:
#using yeo-johnson transformation
df['DrivenKms']=power_transform(df[['DrivenKms']].to_numpy().reshape(-1,1),method='yeo-johnson')
df
```

Out[65]:

	Brand	Model	Variant	ManufactureYr	Fuel type	Control	DrivenKms	Price
0	5.0	74.0	22.0	2017	0.0	1.0	0.474803	6.25
1	5.0	19.0	28.0	2018	1.0	1.0	-0.399018	9.26
2	4.0	17.0	100.0	2020	1.0	1.0	-0.570195	11.23
3	13.0	62.0	65.0	2016	1.0	1.0	0.983113	3.41
4	5.0	19.0	26.0	2019	1.0	1.0	-0.508670	13.16
5	4.0	33.0	14.0	2016	1.0	1.0	1.157500	4.83
6	4.0	17.0	122.0	2019	1.0	0.0	1.110279	12.68
7	4.0	17.0	123.0	2019	1.0	1.0	-0.619352	11.30
8	12.0	55.0	124.0	2017	0.0	1.0	1.064165	6.58
9	19.0	41.0	113.0	2021	1.0	1.0	-1.353676	10.95
10	12.0	40.0	66.0	2018	0.0	1.0	-0.686856	9.89
11	12.0	71.0	106.0	2016	0.0	1.0	2.073742	9.10

- Data was then normally distributed using standardisation technique.

In [67]:
```
x=df.drop(['Price'],axis=1)
y=df['Price']
```

In [68]:
```
scaler=StandardScaler()
x=scaler.fit_transform(x)
x
```

Out[68]:
```
array([[-9.27043829e-01,  1.62144964e+00, -1.06094400e+00, ...,
        -1.68803506e+00,  7.05625928e-01,  4.74803186e-01],
       [-9.27043829e-01, -7.65325784e-01, -9.03110148e-01, ...,
         5.92404757e-01,  7.05625928e-01, -3.99017576e-01],
       [-1.09779561e+00, -8.52117617e-01,  9.90896042e-01, ...,
         5.92404757e-01,  7.05625928e-01, -5.70194846e-01],
       ...,
       [-1.78080271e+00,  4.49759889e-01, -2.19163469e-01, ...,
         5.92404757e-01, -1.41718148e+00,  6.65363395e-01],
       [-9.27043829e-01, -7.65325784e-01, -9.29415790e-01, ...,
         5.92404757e-01, -1.41718148e+00, -1.68391655e-03],
       [ 1.46348104e+00,  1.89384387e-01,  1.30656374e+00, ...,
         5.92404757e-01,  7.05625928e-01,  1.46032895e-01]])
```

- # Hardware and Software Requirements and Tools Used

We imported following packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy.stats import zscore
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import power_transform
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

- Since this is a regression problem so I will use all the regression algorithms such as:
  Decision Tree Regressor
  Random Forest Regressor
  Knn Regressor
  XGBoost Regressor

- Testing of Identified Approaches (Algorithms)

- We applied decision tree regressor, random forest regressor, knn regressor and XGBoost algorithms on the clean data and got 100%,100%,99.9% and 100% accuracy respectively.
- Depending on the model accuracy and various error parameters we opted for random forest regressor and didn't apply hyperparameter tuning as it was not needed.
- So we saved our best performing random forest regressor model.

- Run and Evaluate selected models

1) Decision Tree Regression

We applied this algorithm and found the train accuracy to be 100% and test accuracy to be 99.9%.

In [89]:
```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=100,test_size=0.20)
dt.fit(x_train,y_train)
dt_pred_train=dt.predict(x_train)
dt_pred_test=dt.predict(x_test)
dt_acc_train=r2_score(y_train,dt_pred_train)*100
dt_acc_test=r2_score(y_test,dt_pred_test)*100
print("acc train",dt_acc_train)
print("acc test",dt_acc_test)
```

```
acc train 99.9999993779874
acc test 100.0
```

The test accuracy for decision tree regression is 100% and its cv score is 99.9% thus making us sure that the model is not overfitted, although decision tree regressor is prone to overfitting.

In [93]:
```
cv_score_best_dt=cross_val_score(dt,x,y,cv=19).mean()*100
print("cross validation score is-",cv_score_best_dt)
print("accuracy score for decision tree regression model is-",dt_acc_test)
```

```
cross validation score is- 99.999995733702
accuracy score for decision tree regression model is- 100.0
```

We also tested this model on other metrics:
We calculated Root mean square error, Mean absolute error and Mean square errors and there values are shown in the snapshot below:

Calculating RMSE,MAE,MSE Errors

In [92]:
```
dt_rmse=np.sqrt(mean_squared_error(y_test, dt_pred_test))
dt_mae=mean_absolute_error(y_test, dt_pred_test)
dt_mse=mean_squared_error(y_test,dt_pred_test)
print("RMSE::",np.sqrt(mean_squared_error(y_test, dt_pred_test)))
print("MAE::",mean_absolute_error(y_test, dt_pred_test))
print("MSE::",mean_squared_error(y_test,dt_pred_test))
```

```
RMSE:: 5.494963336934237e-15
MAE:: 3.378272211942498e-15
MSE:: 3.019462207425144e-29
```

Looking at these errors we can say that model is performing really well.

2) Random Forest Regressor

We applied this algorithm and found the train accuracy to be 100% and test accuracy to be 100%.

In [72]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=12,test_size=0.20)
rf.fit(x_train,y_train)
pred_train=rf.predict(x_train)
pred_test=rf.predict(x_test)
rf_train_acc=round(r2_score(y_train,pred_train)*100,1)
rf_test_acc=round(r2_score(y_test,pred_test)*100,1)
print("\nTrain Accuracy- ",rf_train_acc)
print("\nTest Accuracy- ",rf_test_acc)
```

```
Train Accuracy-  100.0

Test Accuracy-  100.0
```

The test accuracy for random forest regression is 100% and its cv score is 99.9% thus making us sure that the model is not overfitted.

In [74]:
```
cv_score_best_rf=cross_val_score(rf,x,y,cv=22).mean()*100
print("cross validation score is-",cv_score_best_rf)
print("accuracy score for random forest regression model is-",rf_test_acc)
```

```
cross validation score is- 99.94368346009979
accuracy score for random forest regression model is- 100.0
```

We also tested this model on other metrics:
We calculated Root mean square error, Mean absolute error and Mean square errors and there values are shown in the snapshot below:

Calculating RMSE,MAE,MSE Errors

In [75]:
```
rf_rmse=np.sqrt(mean_squared_error(y_test, pred_test))
rf_mae=mean_absolute_error(y_test, pred_test)
rf_mse=mean_squared_error(y_test,pred_test)
print("RMSE::",np.sqrt(mean_squared_error(y_test, pred_test)))
print("MAE::",mean_absolute_error(y_test, pred_test))
print("MSE::",mean_squared_error(y_test,pred_test))
```

```
RMSE:: 1.0794830995527927e-14
MAE:: 7.244980531087404e-15
MSE:: 1.1652837622201047e-28
```

Looking at these errors we can say that model is performing really well.

3) Knn Regressor

We applied this algorithm and found the train accuracy to be 99.9% and test accuracy to be 99.9%.

In [76]:
```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=50,test_size=0.20)
knn.fit(x_train,y_train)
knn_pred_train=knn.predict(x_train)
knn_pred_test=knn.predict(x_test)
knn_acc_train=r2_score(y_train,knn_pred_train)
knn_acc_test=r2_score(y_test,knn_pred_test)
print("acc train",knn_acc_train*100)
print("acc test",knn_acc_test*100)
```

```
acc train 99.99999921879663
acc test 99.99998186420095
```

The test accuracy for Knn regression is 99.9% and its cv score is 99.9% thus making us sure that the model is not overfitted.

In [78]:
```
cv_score_best_knn=cross_val_score(knn,x,y,cv=11).mean()*100
print("cross validation score is-",cv_score_best_knn)
print("accuracy score for K Nearest classifier model is-",knn_acc_test*100)
```

```
cross validation score is- 99.932134774835
accuracy score for K Nearest classifier model is- 99.99998186420095
```

We also tested this model on other metrics:
We calculated Root mean square error, Mean absolute error and Mean square errors and there values are shown in the snapshot below:

Calculating RMSE,MAE,MSE Errors

In [79]:
```
knn_rmse=np.sqrt(mean_squared_error(y_test, knn_pred_test))
knn_mae=mean_absolute_error(y_test, knn_pred_test)
knn_mse=mean_squared_error(y_test,knn_pred_test)
print("RMSE::",np.sqrt(mean_squared_error(y_test, knn_pred_test)))
print("MAE::",mean_absolute_error(y_test, knn_pred_test))
print("MSE::",mean_squared_error(y_test,knn_pred_test))
```

```
RMSE:: 0.18418797326971895
MAE:: 0.004867318435754245
MSE:: 0.033925209497206704
```

Looking at these errors we can say that model is performing well but not as good as previous two models.

4) XGBoost Regressor

We applied this algorithm and found the train accuracy to be 100% and test accuracy to be 100%.

XGBoost Regressor

In [80]:
```
from xgboost import XGBRegressor
xgmod=XGBRegressor()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=100,test_size=0.20)
xgmod.fit(x_train,y_train)
pred_train=xgmod.predict(x_train)
pred_test=xgmod.predict(x_test)
xg_train_acc=round(r2_score(y_train,pred_train)*100,1)
xg_test_acc=round(r2_score(y_test,pred_test)*100,1)
print("\nTrain Accuracy- ",xg_train_acc)
print("\nTest Accuracy- ",xg_test_acc)
```

```
Train Accuracy-  100.0

Test Accuracy-  100.0
```

The test accuracy for XGBoost regression is 100% and its cv score is 99.9% thus making us sure that the model is not overfitted.

Cross Validation Score

In [81]:
```
cv_score_best_xg=cross_val_score(xgmod,x,y,cv=20).mean()*100
print("cross validation score is-",cv_score_best_xg)
print("accuracy score for Knn classifier model is-",xg_test_acc)
```

```
cross validation score is- 99.99988634635048
accuracy score for Knn classifier model is- 100.0
```

We also tested this model on other metrics:
We calculated Root mean square error, Mean absolute error and Mean square errors and there values are shown in the snapshot below:

Calculating RMSE,MAE,MSE Errors

In [82]:
```
xgb_rmse=np.sqrt(mean_squared_error(y_test, pred_test))
xgb_mae=mean_absolute_error(y_test, pred_test)
xgb_mse=mean_squared_error(y_test,pred_test)
print("RMSE::",np.sqrt(mean_squared_error(y_test, pred_test)))
print("MAE::",mean_absolute_error(y_test, pred_test))
print("MSE::",mean_squared_error(y_test,pred_test))
```

```
RMSE:: 0.0012174487505740208
MAE:: 0.0008198481198795449
MSE:: 1.4821814602742441e-06
```

Looking at these errors we can say that model is performing well but not as good as decision tree regressor and random forest regressor.

The best performing model is random forest Regressor as its test accuracy and CV score both are almost same. The Root mean square error, mean absolute error and mean square error is least for random forest regressor so we will finalize this model.
Since the accuracy is at its best so we don't need to perform hyper parameter tuning on it. so lets save our model.

CONCLUSION

From the above study we can conclude that if any used car is just 3-4 years old, is of petrol type, automatic and has driven less kilometres then their cost should be comparatively higher than other normal cars.