Creating a model for 'Japan Used Cars Price Prediction' to check what should be the price of a car in resale market

In this post, I will go through the whole process of creating machine learning model on the '**Japan Used car price prediction'**. In this, we need to predict price of a used car listed on the website ' tc-v.com'.

We all know that there is a great demand of used car by middle and lower middle class people for whom affording a new car is quite difficult due to expansiveness. Reduced cash inflow due to the pandemic (Covid-19)has forced buyers to look for alternatives other than new cars, and the used car industry has high growth potential in these terms. As the sales and production of new vehicles have been hindered due to the pandemic, the used car market is gaining traction among buyers.

We will cover below points in the blog

1. PROBLEM STATEMENT
2. DATA ANALYSIS
3. EDA
4. PRE-PROCESSING PIPELINE
5. BUILDING MACHINE LEARNING MODELS
6. CONCLUDING REMARKS

Source of Dataset: The dataset for this project is received from below link

https://github.com/dsrscientist/dataset4/blob/main/Japan_used_cars_datasets.csv

1. PROBLEM STATEMENT

Cars' data was scraped from tc-v.com and it included Information about Japan's largest online used car marketplace. Ten features were assembled for each car in the dataset.

This dataset includes 10 features:

Feature	Type	Description
Price	Integer	The sale price of the vehicle in the ad
Mark	String	The brand of car

Feature	Type	Description
Model	String	model of the vehicle
Years	Integer	The vehicle registration year
Mileage	Integer	miles traveled by vehicle
Engine_capacity	Integer	The measurement of the total volume of the cylinders in the engine
Transmission	String	The type of gearbox used by the car
Drive	String	wheel drive(2wd, 4wd and awd)
Hand_drive	String	Left-hand traffic (LHT) and right-hand traffic (RHT)
Fuel	String	The type of fuel used by the car(gasoline, diesel, hybrid, lpg and cng)

2 DATA ANALYSIS:

For data analysis, we need to get the data and observe it properly.

Importing the libraries

```
#importing the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

We need to first import all the libraries that will be useful in reading a file, preprocessing, missing values handling, for splitting, plotting graphs, model creation and many more.

Reading the csv file

csv file contains the data, it is dataset, which will need to be used for processing. In short we can say, it’s the soul of all the process.

```
#reading the dataset
df=pd.read_csv('Japan_used_cars_datasets.csv')
print(df.shape)
df.head()
```

```
(2318, 11)
```

	id	price	mark	model	year	mileage	engine_capacity	transmission	drive	hand_drive	fuel
0	0	80	nissan	march	2003	80000	1240	at	2wd	rhd	gasoline
1	1	110	nissan	march	2010	53000	1200	at	2wd	rhd	gasoline
2	2	165	nissan	lafesta	2005	47690	2000	at	2wd	rhd	gasoline
3	3	190	toyota	avensis	2008	130661	1990	at	2wd	rhd	gasoline
4	4	190	daihatsu	mira	2006	66300	660	at	2wd	rhd	gasoline

Given data set have 2318 rows and 11 columns

3 EDA/EXPLORATORY DATA ANALYSIS

There is a need to remove the null values and duplicate rows as it will affect the predicted values. Sometimes instead of dropping the null values we can replace them with any appropriate value, we can say it will be mean, median, mode or 0. We can replace it with any value which will be appropriate as per the dataset.

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

```
id                 0
price              0
mark               0
model              0
year               0
mileage            0
engine_capacity    0
transmission       0
drive              0
hand_drive         0
fuel               0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

```
df.describe()
```

	id	price	year	mileage	engine_capacity
count	2318.000000	2318.000000	2318.000000	2318.000000	2318.000000
mean	1169.047023	971.522433	2005.972390	100013.194996	1507.010785
std	674.460724	288.673112	3.698863	52512.478883	549.585170
min	0.000000	80.000000	1979.000000	2000.000000	9.000000
25%	583.250000	776.000000	2004.000000	67000.000000	1300.000000
50%	1168.500000	1000.000000	2006.000000	94000.000000	1490.000000
75%	1753.750000	1213.000000	2009.000000	124000.000000	1800.000000
max	2335.000000	1400.000000	2015.000000	790000.000000	12340.000000

As we can see that there are no null values and no duplicate data are available in the given data set.

```
df.dtypes
```

```
id                   int64
price                int64
mark                object
model               object
year                 int64
mileage              int64
engine_capacity      int64
transmission        object
drive               object
hand_drive          object
fuel                object
dtype: object
```

It has '*object'*,and *'int64'* data types.

Separating categorical and numerical columns in a different list form

```
# finding categorical variables

categorical = [var for var in df.columns if df[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :', categorical)
```

```
There are 6 categorical variables

The categorical variables are : ['mark', 'model', 'transmission', 'drive', 'hand_drive', 'fuel']
```

```
numerical = [var for var in df.columns if df[var].dtype!='O']

print('There are {} numerical variables\n'.format(len(numerical)))

print('The numerical variables are :', numerical)
```

```
There are 5 numerical variables

The numerical variables are : ['id', 'price', 'year', 'mileage', 'engine_capacity']
```

```
# checking for categorical columns
categorical_columns=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        categorical_columns.append(i)
print(categorical_columns)
```

```
['mark', 'model', 'transmission', 'drive', 'hand_drive', 'fuel']
```

Making a data frame of categorical columns and their unique features.

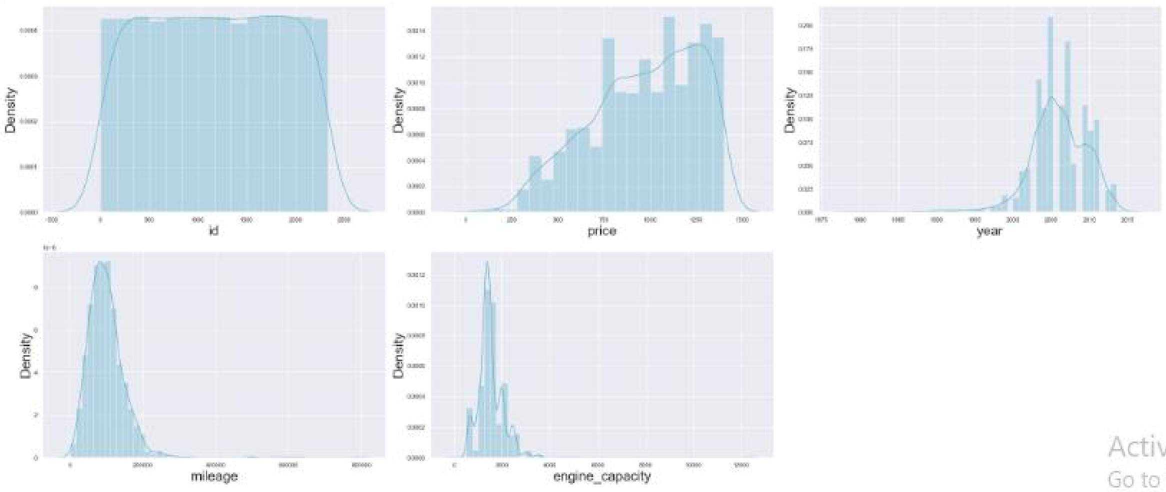
```python
pd.options.display.max_colwidth =None
col_name=[]
num=[]
feat=[]
for i in categorical_columns:
    col_name.append(i)
    num.append(len(df[i].unique()))
    feat.append(df[i].unique())
inf=pd.DataFrame({'FEATURE':col_name,"NUMBER OF UNIQUE FEATURES":num,"UNIQUE FEATURES":feat})
inf
```

	FEATURE	NUMBER OF UNIQUE FEATURES	UNIQUE FEATURES
0	mark	28	[nissan, toyota, daihatsu, volkswagen, mazda, honda, subaru, mercedes-benz, kubota, mitsubishi, suzuki, bmw, chrysler, smart, gm, opel, isuzu, land rover, peugeot, hyundai, kia motors, audi, citroen, mitsuoka, volvo, ford, hino, jaguar]
1	model	258	[march, lafesta, avensis, mira, passat, bongo van, step wgn, sambar, inspire, mercedes-benz others, note, passo, impreza, kubota others, life, progres, cube cubic, cube, a-class, coltplus, mr wagon, pajero mini, fit, legacy b4, vitz, mark ii blit, insight, move, tanto, terios kid, az-wagon, coo, verisa, swift, delica d2, tiida, sienta, mini, colt, platz, bluebird sylphy, eclipse, bb, moco, clipper truck, esse, ek wagon, polo, x-trail, atenza sport, ad van, pixis space, freed spike, stream, alto, raum, wish, every, demio, jimny, premacy, cr-v, lapin, delica, wagon r, lancer, naked, odyssey, ractis, porte, corolla rumion, tiida latio, voxy, carol, sunny, bongo truck, boon, s-mx, pt cruiser, golf, ad expert, fit hybrid, serena, edix, move custom, impreza anesis, estima, carry truck, solio, mpv, vanette van, accord, tribute, pajero io, noah, vanette truck, delica truck, lancer cargo, atenza wagon, atenza, ...]
2	transmission	3	[at, mt, cvt]
3	drive	3	[2wd, 4wd, awd]
4	hand_drive	3	[rhd, center, lhd]
5	fuel	5	[gasoline, diesel, hybrid, lpg, cng]

After data frame of categorical columns , I wanted to check the distribution of numerical columns

```python
#Distribution plot for all numerical columns
sns.set(style="darkgrid")

plt.figure(figsize = (30,30))
plotnumber = 1
for column in df[numerical_columns]:
    if plotnumber <=13:
        ax = plt.subplot(5,3,plotnumber)
        sns.distplot(df[column],color='c')
        plt.xlabel(column,fontsize = 25)
        plt.ylabel('Density',fontsize = 25)
    plotnumber+=1
plt.tight_layout()
```



Checking the correlation among the features

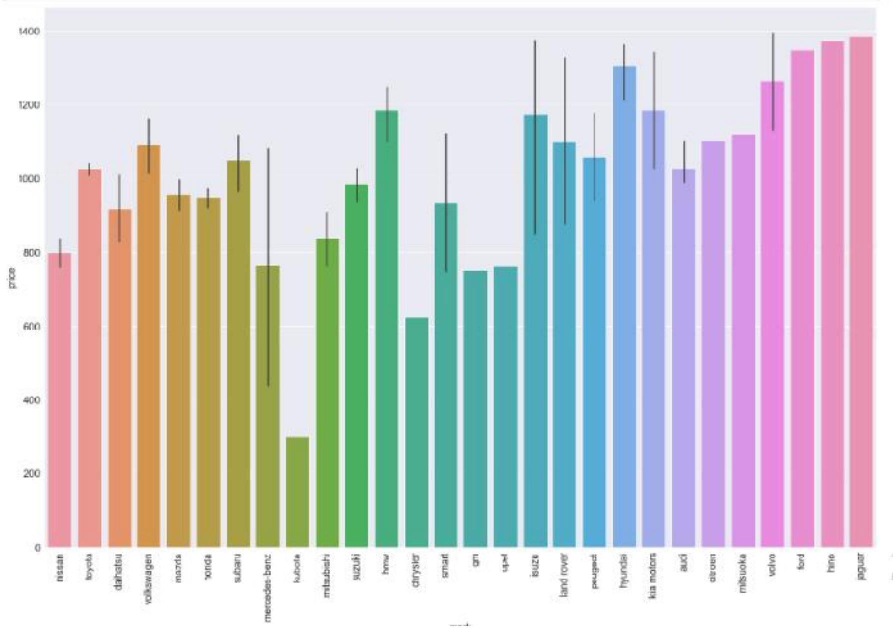
```
print(df.corr()['price'].drop(['price']).sort_values())
```

```
year              -0.211092
mileage            0.021581
engine_capacity    0.173748
id                 0.985100
Name: price, dtype: float64
```

```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True,annot_kws={'size':10})
plt.show()
```

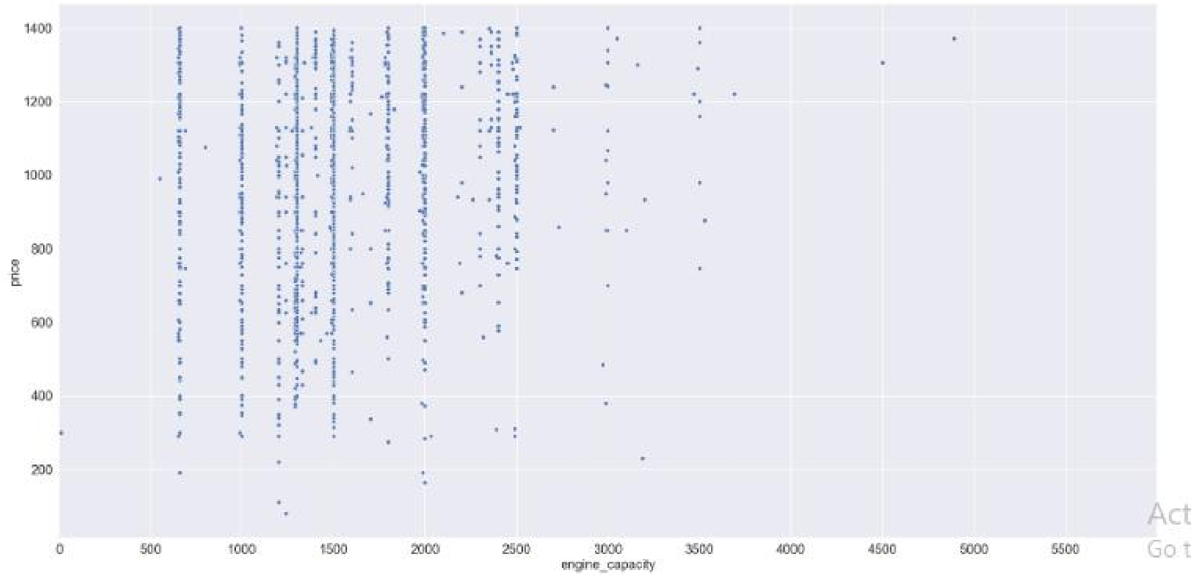
Checking the mark wise price of the cars according to given data

```
plt.figure(figsize=(30,20))
plt.xticks(size=20,rotation=90)
plt.yticks(size=20)
plt.xlabel('MARK',fontsize=20)
plt.ylabel('PRICE',fontsize=20)
sns.barplot(x=df['mark'],y=df['price'])
plt.show()
```



Checking the relation among the price and engine capacity through scattered plot

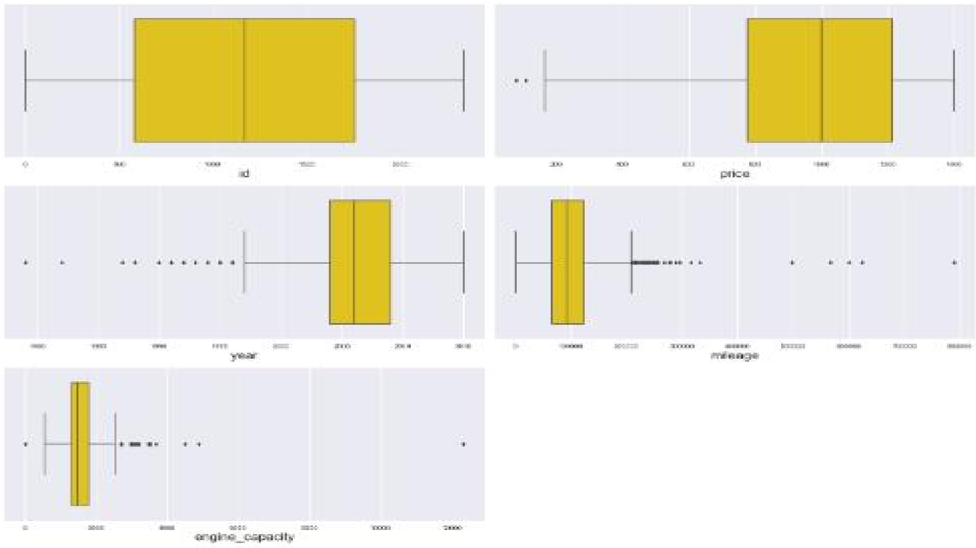
```python
plt.figure(figsize=(30,15))
sns.scatterplot(x=df['engine_capacity'],y=df['price'])
plt.xlabel('engine_capacity',fontsize=20)
plt.ylabel('price',fontsize=20)
plt.xticks(size=20)
plt.yticks(size=20)
plt.xticks(np.arange(0,6000,500))
plt.xlim(0,6000)
plt.show()
```



Checking the outliers of numerical data

```python
# Identifying the outliers using boxplot

plt.figure(figsize=(20,15),facecolor='white')
plotnumber=1
for column in numerical_columns:
    if plotnumber<=6:
        ax=plt.subplot(3,2,plotnumber)
        sns.boxplot(df[column],color='gold')
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.tight_layout()
```

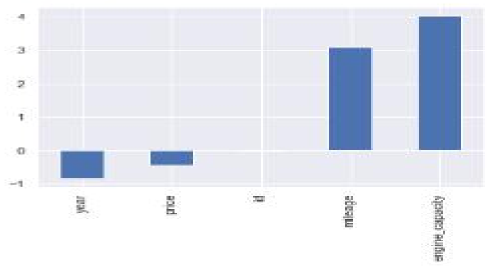


There are outliers present in the columns like engine_capacity , mileage and year

Checking the skewness of numerical columns

```python
print(df.skew().sort_values())
df.skew().sort_values().plot(kind='bar')
plt.show()
```

```
year               -0.845917
price              -0.466780
id                 -0.002276
mileage             3.089142
engine_capacity     4.038892
dtype: float64
```



Plot and data both showing that

Year, Price and id are having negative skew on the other hand mileage and engine_capacity are having positive skew

PRE PROCESSING

Encoding

Encoding the categorical columns with Binary encoder

```
import category_encoders as ce
ben=ce.BinaryEncoder(cols=categorical_columns)
```

```
df=ben.fit_transform(df)
df
```

	id	price	mark_0	mark_1	mark_2	mark_3	mark_4	model_0	model_1	model_2	...	engine_capacity	transmission_0	transmission_1	drive_0	drive_
0	0	80	0	0	0	0	1	0	0	0	...	1240	0	1	0	
1	1	110	0	0	0	0	1	0	0	0	...	1200	0	1	0	
2	2	165	0	0	0	0	1	0	0	0	...	2000	0	1	0	
3	3	190	0	0	0	1	0	0	0	0	...	1990	0	1	0	
4	4	190	0	0	0	1	1	0	0	0	...	660	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2313	2331	1400	0	0	0	1	0	0	0	0	...	996	0	1	0	
2314	2332	1400	0	0	0	1	0	0	0	1	...	3000	0	1	0	
2315	2333	1400	0	0	1	1	1	0	0	1	...	660	1	1	0	
2316	2334	1400	0	0	1	1	0	1	0	0	...	660	0	1	1	
2317	2335	1400	0	0	0	1	0	1	0	0	...	3000	0	1	0	

2318 rows × 28 columns

Encoding increases the number of columns from 11 to 28

After separating the feature and target column I am checking the skewness of features

```
x_scaled=df_new.drop(['price'],axis=1)
y=df_new['price']
```

```
x_scaled.skew()
```

```
id                  -0.282512
mark_0               9.416675
mark_1               2.708708
mark_2               0.897529
mark_3              -1.207557
mark_4               0.880819
model_0             33.120970
model_1              2.311707
model_2              0.868977
model_3              0.325800
model_4             -0.200498
model_5              0.024598
model_6             -0.013665
model_7              0.185718
model_8             -0.563060
year                -0.000962
mileage             -0.027962
engine_capacity      0.007618
transmission_0       4.885655
transmission_1      -4.885083
drive_0              3.226541
drive_1             -3.290072
hand_drive_0        12.415532
hand_drive_1         0.000000
fuel_0              23.388002
fuel_1              11.597563
fuel_2             -11.243473
dtype: float64
```

Removing the outliers of numerical columns

```
df_nums=df.copy()
features=df[['id', 'price', 'year', 'mileage', 'engine_capacity']]
```

```
# 1st quantile
Q1=features.quantile(0.25)

# 3rd quantile
Q3=features.quantile(0.75)

# IQR
IQR=Q3 - Q1

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

```
df_nums.shape
```

```
(2197, 28)
```

```
(2318-2197)/2318*100
```

```
5.220017256255392
```

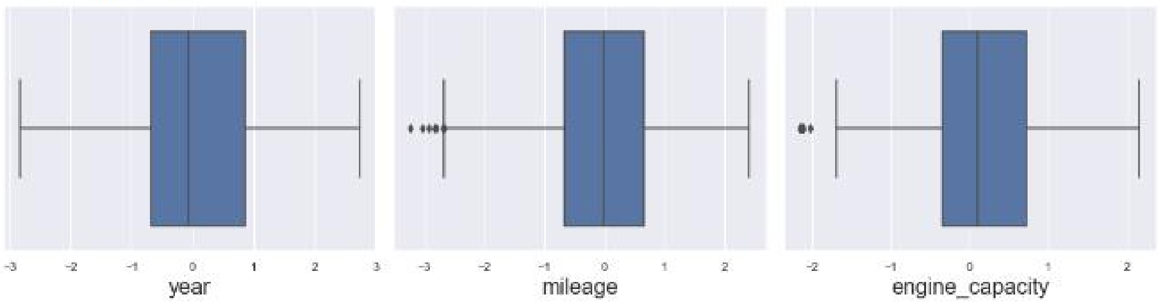
5.22% of data loss in removing the outliers

As number of rows reduced from 2318 to 2197

Checking the outliers through box plot

```
# Checking if the outliers is reduced or not
sns.set(style="darkgrid")

plt.figure(figsize=(20,15),facecolor='white')
plotnumber=1
for column in x_scaled[['year','mileage','engine_capacity']]:
    if plotnumber<=30:
        ax=plt.subplot(4,4,plotnumber)
        sns.boxplot(df_new[column],color='b')
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.tight_layout()
```



Selecting the best (20) features out of 28 features through K-best feature selection method

```
from sklearn.feature_selection import SelectKBest,f_classif
select=SelectKBest(score_func=f_classif,k=20)
k=20
fit=select.fit(x_scaled,y)
```

```
cols=fit.get_support(indices=True)
print(f' top {k} FEATURES INDEX  = {cols}')
```

```
top 20 FEATURES INDEX  = [ 0  1  3  4  5  6  7  8  9 10 11 13 15 16 17 18 19 22 25 26]
```

```
features=x_scaled.columns[cols]
list(features)
```

```
['id',
 'mark_0',
 'mark_2',
 'mark_3',
 'mark_4',
 'model_0',
 'model_1',
 'model_2',
 'model_3',
 'model_4',
 'model_5',
 'model_7',
 'year',
 'mileage',
 'engine_capacity',
 'transmission_0',
 'transmission_1',
 'hand_drive_0',
 'fuel_1',
 'fuel_2']
```

```
print(f' BEST {len(cols)} FEATURES ARE AS FOLLOWS :  \n\n{list(features)} ')
```

```
 BEST 20 FEATURES ARE AS FOLLOWS :

['id', 'mark_0', 'mark_2', 'mark_3', 'mark_4', 'model_0', 'model_1', 'model_2', 'model_3', 'model_4', 'model_5', 'model_7', 'ye
ar', 'mileage', 'engine_capacity', 'transmission_0', 'transmission_1', 'hand_drive_0', 'fuel_1', 'fuel_2']
```

BUILDING MACHINE LEARNING MODELS

Importing required libraries to select the random state
After splitting the data into train and test

```
In [66]:
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

```
In [68]:
for i in range (0,200):
    x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=i)
    lr=LinearRegression()
    lr.fit(x_train,y_train)
    lr_train_pred=lr.predict(x_train)
    lr_test_pred=lr.predict(x_test)
    lr_train_accuracy=r2_score(y_train,lr_train_pred)
    lr_test_accuracy=r2_score(y_test,lr_test_pred)
    if(round(lr_train_accuracy*100,1)==round(lr_test_accuracy*100,1)):
        print('\n\nAT RANDOM STATE--',i)
        print(f'\n\nTRAINING ACCURACY IS -  {round((lr_train_accuracy)*100,2)}--AND TESTING ACCURACY IS {round((lr_test_accu
```

```
TRAINING ACCURACY IS -  99.81--AND TESTING ACCURACY IS 99.81


AT RANDOM STATE-- 33


TRAINING ACCURACY IS -  99.81--AND TESTING ACCURACY IS 99.79


AT RANDOM STATE-- 34


TRAINING ACCURACY IS -  99.81--AND TESTING ACCURACY IS 99.81


AT RANDOM STATE-- 35


TRAINING ACCURACY IS -  99.8--AND TESTING ACCURACY IS 99.81
```

```
In [69]:
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=34)
```

Random state 34 is selected as at this point training and testing accuracies are equal

FOR LINEAR REGRESSION MODEL

```
1  score(lr,x_train,x_test,y_train,y_test,train=True)
2  score(lr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE LinearRegression() is 99.81
mean sqaured error is -- 0.001989616938398985
mean sqaured error is -- 0.001989616938398985
root mean sqaured error is -- 0.044605122333639086
mean aboslute error is -- 0.03722162436646706
difference between rmse and mae is  0.00738349796717238

Testing SCORE FOR THE LinearRegression() is 99.81

CROSS VAL SCORE IS -- 90.27
adjusted r2_score for LinearRegression() is 99.81
mean sqaured error is -- 0.001754992298106324
mean aboslute error is - 0.03517557933460611
root mean sqaured error is -- 0.0418926282071956
mean aboslute error is -- 0.03517557933460611
```

Training score =99.81

Teating score =99.81

Adjusted R2 score =99.81

CROSS VAL SCORE IS = 90.27

FOR DECISION TREE REGRESSON

MODEL 2)-DECISION TREE

```
1  from sklearn.tree import DecisionTreeRegressor
2  dt=DecisionTreeRegressor()
```

```
1  score(dt,x_train,x_test,y_train,y_test,train=True)
2  score(dt,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE DecisionTreeRegressor() is 100.0
mean sqaured error is -- 4.0629715598362593e-32
mean sqaured error is -- 4.0629715598362593e-32
root mean sqaured error is -- 2.0156814132784623e-16
mean aboslute error is -- 8.657291809830824e-17
difference between rmse and mae is  1.14995223229538e-16

Testing SCORE FOR THE DecisionTreeRegressor() is 100.0

CROSS VAL SCORE IS -- -187.06
adjusted r2_score for DecisionTreeRegressor() is 100.0
mean sqaured error is -- 3.012427072234439e-05
mean aboslute error is - 0.001934262777723109
root mean sqaured error is -- 0.005488558164249003
mean aboslute error is -- 0.001934262777723109
difference between rmse and mae is  0.003554295386525893
```

Training score =100

Teating score =100

Adjusted R2 score =100

Cross val score is = -187.06

FOR KNN

```
MODEL 3) KNN
```

```
from sklearn.neighbors import KNeighborsRegressor
knr=KNeighborsRegressor()
```

```
knr.fit(x_train,y_train)
```

```
KNeighborsRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
score(knr,x_train,x_test,y_train,y_test,train=True)
score(knr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE KNeighborsRegressor() is 89.79

mean sqaured error is -- 0.1049859146643431

mean sqaured error is -- 0.1049859146643431

root mean sqaured error is -- 0.32401530004668466

mean aboslute error is -- 0.22947656274550202

difference between rmse and mae is  0.09453873730118265

Testing SCORE FOR THE KNeighborsRegressor() is 83.12

CROSS VAL SCORE IS -- -617.51

adjusted r2_score for KNeighborsRegressor() is 83.06

mean sqaured error is -- 0.1542080057286002

mean aboslute error is - 0.27150768201868697

root mean sqaured error is -- 0.3926932718147845

mean aboslute error is -- 0.27150768201868697

difference between rmse and mae is  0.1211855897960975
```

Training score =89.79

Testing score =83.12

Adjusted R2 value=83.06
Cross validation score=-617.51

After TUNING THE PARAMETERS------------

Training score =100

Testing score =85.86

Cross validation score=-577.57

```
knr=KNeighborsRegressor()
```

```
pu={"n_neighbors":np.arange(2,5),
   "weights":['uniform', 'distance'],
   "algorithm":['auto', 'ball_tree', 'kd_tree', 'brute'],
    'leaf_size':np.arange(40,50)}
```

```
kgsv=GridSearchCV(knr,param_grid=pu)
```

```
kgsv.fit(x_train,y_train)
```

```
GridSearchCV(estimator=KNeighborsRegressor(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'leaf_size': array([40, 41, 42, 43, 44, 45, 46, 47, 48, 49]),
                         'n_neighbors': array([2, 3, 4]),
                         'weights': ['uniform', 'distance']})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
kgsv.best_params_
```

```
{'algorithm': 'auto', 'leaf_size': 40, 'n_neighbors': 4, 'weights': 'distance'}
```

```
knr=kgsv.best_estimator_
```

```
knr.fit(x_train,y_train)
```

```
KNeighborsRegressor(leaf_size=40, n_neighbors=4, weights='distance')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
score(knr,x_train,x_test,y_train,y_test,train=True)
print('\n\n------------------------\n\n')
score(knr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE KNeighborsRegressor(leaf_size=40, n_neighbors=4, weights='distance') is 100.0

mean squared error is -- 2.3081125377435987e-15

mean squared error is -- 2.3081125377435987e-15

root mean squared error is -- 4.804281983547167e-08

mean abosolute error is -- 2.0621465779773316e-08

difference between rmse and mae is  2.7421354055698356e-08

------------------------


Testing SCORE FOR THE KNeighborsRegressor(leaf_size=40, n_neighbors=4, weights='distance') is 85.86

CROSS VAL SCORE IS -- -577.51
```

FOR RANDOM FOREST REGRESSION

Training score =100

Testing score =100

Cross validation score=-187.84

Adjusted R2 value = 100

MODEL 4)-RANDOM FOREST REGRESSOR

```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
```

```
rfr.fit(x_train,y_train)
```

RandomForestRegressor()
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
score(rfr,x_train,x_test,y_train,y_test,train=True)
score(rfr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE RandomForestRegressor() is 100.0

mean sqaured error is -- 4.152728006232281e-06

mean sqaured error is -- 4.152728006232281e-06

root mean sqaured error is -- 0.002037824331543885

mean aboslute error is -- 0.0007872434926212501

difference between rmse and mae is  0.0012505808389226348


Testing SCORE FOR THE RandomForestRegressor() is 100.0


CROSS VAL SCORE IS -- -187.84

adjusted r2_score for RandomForestRegressor() is 100.0

mean sqaured error is -- 1.745119198008833e-05

mean aboslute error is - 0.0018611474608785361

root mean sqaured error is -- 0.004177462385239194

mean aboslute error is -- 0.0018611474608785361

difference between rmse and mae is  0.0023163149243606573
```

FOR SVR

MODEL 5) SVR

```python
from sklearn.svm import SVR
svr=SVR()
```

```python
svr.fit(x_train,y_train)
```

```
SVR()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```python
score(svr,x_train,x_test,y_train,y_test,train=True)
score(svr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE SVR() is 99.44

mean sqaured error is -- 0.00578144857100038

mean sqaured error is -- 0.00578144857100038

root mean sqaured error is -- 0.07603583741237009

mean aboslute error is -- 0.06114837404051081

difference between rmse and mae is  0.014887463371859283

Testing SCORE FOR THE SVR() is 97.74

CROSS VAL SCORE IS -- -19.08

adjusted r2_score for SVR() is 97.74

mean sqaured error is -- 0.020602994015964066

mean aboslute error is - 0.08078619410997623

root mean sqaured error is -- 0.14353743071395722

mean aboslute error is -- 0.08078619410997623

difference between rmse and mae is  0.06275123660398099
```

Training score =99.44

Testing score =97.74

Adjusted R2 value=97.74

Cross validation score=-19.04

ADA BOOST REGRESSOR

Training score =99.44

Testing score =97.74

Ajusted R2 value= 99.63

Cross validation score=-19.04

MODEL 6) ADA BOOST REGRESSOR

```
from sklearn.ensemble import AdaBoostRegressor
adr=AdaBoostRegressor()
```

```
adr.fit(x_train,y_train)
```

AdaBoostRegressor()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
score(adr,x_train,x_test,y_train,y_test,train=True)
score(adr,x_train,x_test,y_train,y_test,train=False)
```

```
Training SCORE FOR THE AdaBoostRegressor() is 99.56

mean sqaured error is -- 0.004489293635979762

mean sqaured error is -- 0.004489293635979762

root mean sqaured error is -- 0.06700219127744825

mean aboslute error is -- 0.04784736916875553

difference between rmse and mae is  0.019154822108692716


Testing SCORE FOR THE AdaBoostRegressor() is 99.63


CROSS VAL SCORE IS -- -290.65

adjusted r2_score for AdaBoostRegressor() is 99.63

mean sqaured error is -- 0.0033416736105010224

mean aboslute error is - 0.04484113148171567

root mean sqaured error is -- 0.057807210713725174

mean aboslute error is -- 0.04484113148171567

difference between rmse and mae is  0.012966079232009507
```

At each and every step I have done Hyperparametric tuning also

Conclusion

Here we can easily see that LINEAR REGRESSION gave a quite close value for Training and Testing data

ADJSUTED R2 SCORE and TESTING SCORES WERE CLOSER TO EACH OTHER as well as CROSS VAL SCORE is also close with them

Hense WE WILL PREFER LINEAR REGRESSION MODEL