USED CAR PROJECT

SUBMITTED BY:

NAINCY JOSHI

DATASET DETAILS:

- We have scrapped the data using selenium with various car website such as cars24,cardekho,Olx,etc.
- The dataset contain following column such as Model_name,Location,year,Variant,EMI, Function,Owner and the Price.
- There are lot of missing value in the dataset
- The type of the data is “Object” datatype.
- The “Nan” values are present in various column.
- Various column has numeric as well as the character so we have to remove that character value.

In [77]: data

Out[77]:

	Model	Location	Year	Driven	Variant	Price	Owner	EMI	Function
0	Skoda Rapid 2011-2013 1.6 MPI Elegance, 2015, ...	HEBBAL, BENGALURU	2015	30,000 km	Petrol	₹ 5,95,000	NaN	NaN	NaN
1	MG Hector Sharp AT, 2019, Petrol	ERNAKULAM, KOCHI	2019	12,000 km	Petrol	₹ 17,60,000	NaN	NaN	NaN
2	BMW 5 Series 530d Highline Sedan, 2010, Diesel	MAHADEV VIHAR, DEHRADUN	2010	69,000 km	Diesel	₹ 7,51,000	NaN	NaN	NaN
3	Mahindra Bolero 2009 slx	VADAVALLI, COIMBATORE	2009	204,700 km	Mahindra Bolero 2009 slx	₹ 5,50,000	NaN	NaN	NaN
4	Maruti Suzuki Swift 2007 Petrol 92000 Km Driven	VAISHALI, GHAZIABAD	2007	93,000 km	Maruti Suzuki Swift 2007 Petrol 92000 Km Driven	₹ 1,15,000	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
1579	Hyundai Grand i10 Asta AT 2015	Mumbai	2015	34,028Km	Petrol	4,60,000	Second Owner	NaN	Automatic
1580	Hyundai Grand i10 1.2 Sportz (O) AT 2014	Mumbai	2014	47,000Km	Petrol	4,48,050	First Owner	NaN	Automatic
1581	Hyundai Grand i10 Magna CRDi 2014	Lucknow	2014	60,000Km	Petrol	4,65,000	First Owner	NaN	Manual
1582	Hyundai Grand i10 Magna 1.2 Kappa VTVT 2014	Guwahati	2014	44,000Km	Petrol	3,72,000	First Owner	NaN	Manual
1583	Hyundai Verna 1.6 VTVT SX 2008	Pune	2008	68,000Km	Petrol	2,32,000	Second Owner	NaN	Manual

5002 rows × 9 columns

DATA CLEANING:

- Data.isna().sum() will give us the missing value in the dataset.
- Data.describe() will gives the count,top and the freq of the dataset.
- First we have filled the missing data with the value “0”.
- We have used imputing technique to fill the “object” datatype in various column.

- In the column “Driven” along with Kilometer number there is shortcut kmis also mention,so we replace those km with the blank space. And then we have converted into Float type.
- In the “Price” column,we have removed the rupee sign,so as to make the column as a int type.

```
In [83]: data['Price'] = data['Price'].str.replace('₹', '')
         data['Price'] = data['Price'].str.replace(',', '')
         data['Price'] = data['Price'].str.replace('Lakh', '')

In [84]: data['Price'] = data['Price'].astype(float)

In [85]: data['EMI'] = data['EMI'].str.replace('₹', '')
         data['EMI'] = data['EMI'].str.replace(',', '')

In [86]: data['EMI'] = data['EMI'].astype(float)

In [87]: data['Year'] = data['Year'].str.replace('IV-', '')

In [88]: data['Year'] = data['Year'].astype(float)

In [94]: data['Driven'] = data['Driven'].str.replace('Km', '')
         data['Driven'] = data['Driven'].str.replace('km', '')
         data['Driven'] = data['Driven'].str.replace(',', '')

In [95]: data['Driven'] = data['Driven'].astype(float)
```

PREPROCESSING DATA:

- We have used LabelEncoder() to encode the features of the dataset.

- We have used mode to convert the “Object” datatype into Float.

Added new column:

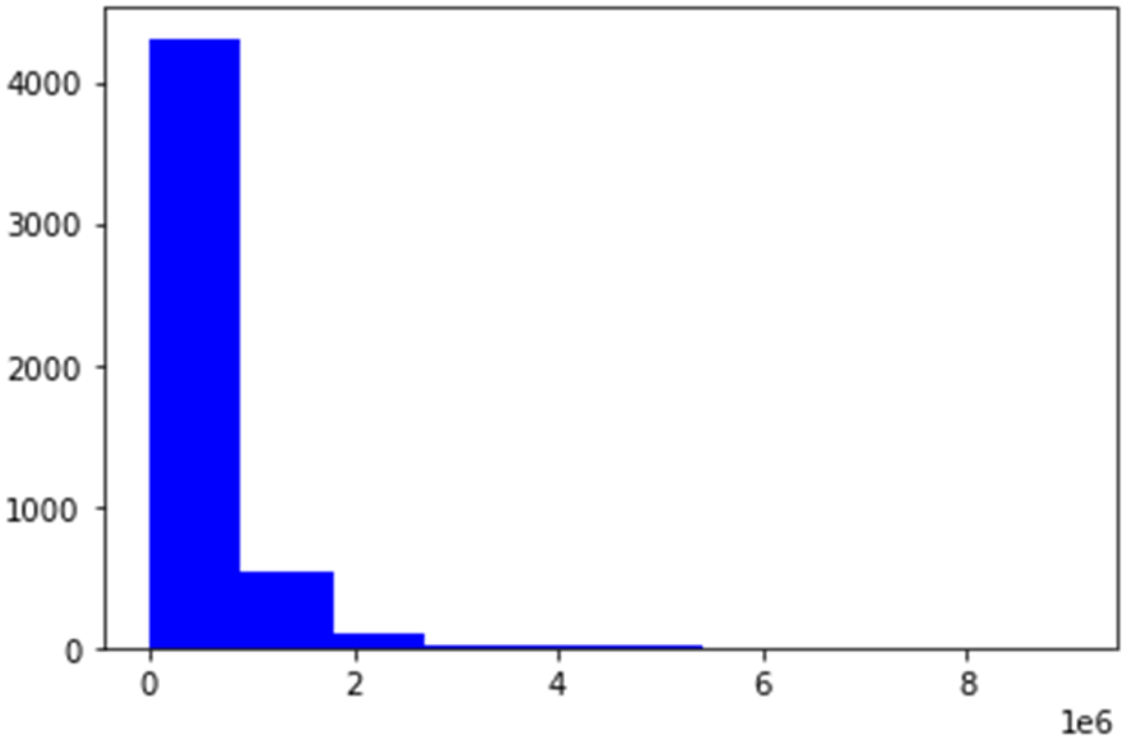
- We have added a new column named “No_of_year” .
- First we have added the current_year i.e 2021 and then we have subtracted the current_year with the year which has been given.
- From this we will get to know how old the car it is.

```
In [98]: data["Current_Year"]=2021

In [99]: data["No_of_years"]=data["Current_Year"]-data["Year"]
         data=data.drop(["Current_Year","Year"],axis=1)
         data.head()
```

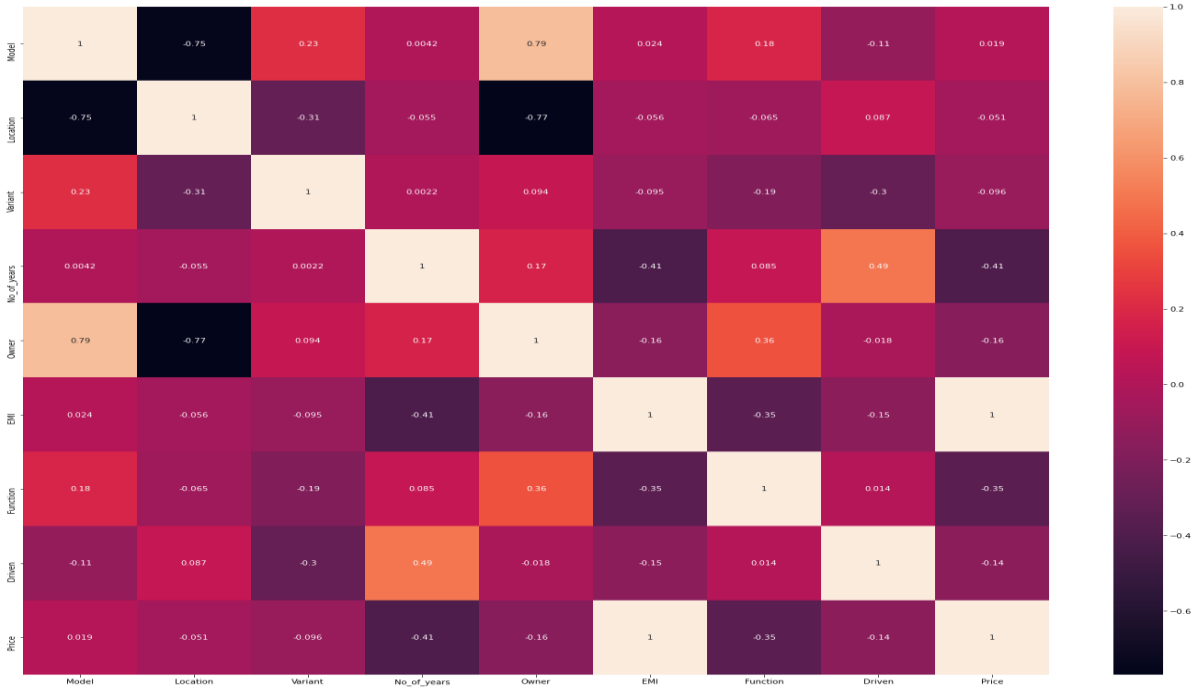
Out[99]:

	Model	Location	Driven	Variant	Price	Owner	EMI	Function	No_of_years
0	Skoda Rapid 2011-2013 1.6 MPI Elegance, 2015, ...	HEBBAL, BENGALURU	30000.0	Petrol	595000.0	NaN	NaN	NaN	6.0
1	MG Hector Sharp AT, 2019, Petrol	ERNAKULAM, KOCHI	12000.0	Petrol	1760000.0	NaN	NaN	NaN	2.0
2	BMW 5 Series 530d Highline Sedan, 2010, Diesel	MAHADEV VIHAR, DEHRADUN	69000.0	Diesel	751000.0	NaN	NaN	NaN	11.0
3	Mahindra Bolero 2009 slx	VADAVALLI, COIMBATORE	204700.0	Mahindra Bolero 2009 slx	550000.0	NaN	NaN	NaN	12.0
4	Maruti Suzuki Swift 2007 Petrol 92000 Km Driven	VAISHALI, GHAZIABAD	93000.0	Maruti Suzuki Swift 2007 Petrol 92000 Km Driven	115000.0	NaN	NaN	NaN	14.0

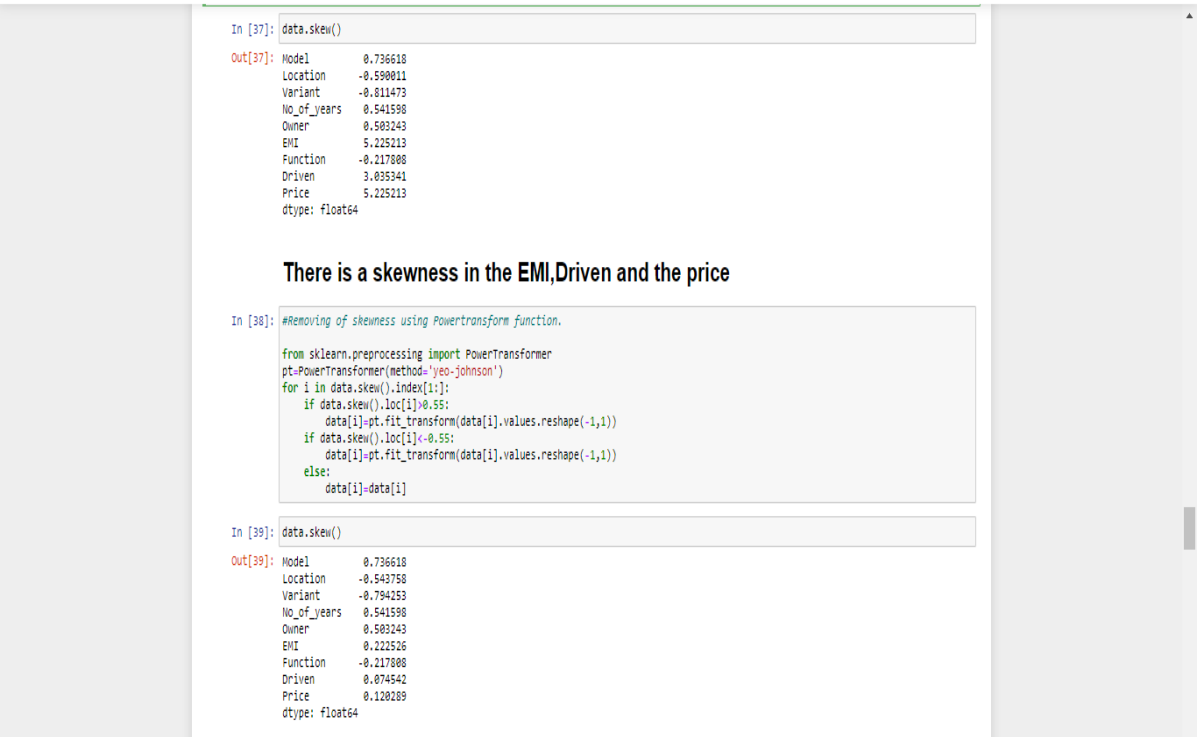
- We have imputing “EMI” value based on the previous calculation.every EMI which are present in the dataset has been for 44.95 years so we have divided the car Price with the year i.e 44.95 years.
- The skewness in the target column i.e “Price” is the 5.22



Data Correlation:



- Highest correlation of the target column “Price” is with the No_of_years that the car has been used.
- The “Driven” i.e for how many km it has been used.
- The “Model” i.e if we use sedan,Suv then the price is more.
- The function i.e if it is Automatic then the price is more and if it is Manual then the Price is less.



```
In [37]: data.skew()

Out[37]: Model          0.736618
         Location      -0.590011
         Variant       -0.811473
         No_of_years    0.541598
         Owner          0.503243
         EMI            5.225213
         Function      -0.217808
         Driven         3.035341
         Price          5.225213
         dtype: float64
```

There is a skewness in the EMI,Driven and the price

```
In [38]: #Removing of skewness using Powertransform function.

         from sklearn.preprocessing import PowerTransformer
         pt=PowerTransformer(method='yeo-johnson')
         for i in data.skew().index[1:]:
             if data.skew().loc[i]>0.55:
                 data[i]=pt.fit_transform(data[i].values.reshape(-1,1))
             if data.skew().loc[i]<-0.55:
                 data[i]=pt.fit_transform(data[i].values.reshape(-1,1))
             else:
                 data[i]=data[i]
```

```
In [39]: data.skew()

Out[39]: Model          0.736618
         Location      -0.543758
         Variant       -0.794253
         No_of_years    0.541598
         Owner          0.503243
         EMI            0.222526
         Function      -0.217808
         Driven         0.074542
         Price          0.120289
         dtype: float64
```

- There is a skewness in the various column such as EMI,Driven and the Price.
- To remove skewness,we have used PowerTransformer()
- Currently, PowerTransformer supports the Box-Cox transform and the Yeo-Johnson transform.
- The optimal parameter for stabilizing variance and minimizing skewness is estimated through maximum likelihood.
- After successfully removal of skewness,we will build the model.

BUILDING THE MODEL

- First we will split the data into “X” as the features and “Y” as the target feature or vector.
- Then we will train and test the data using train_test_split.We will take 20% as the test data and 80% as the train data and then we will predict the test data.

Splitting the data into "X" as a feature and "Y" as the vector or the target column

In [146]:
```
# Splitting the data into x and y for train test split.

X=data.drop(['Price'],axis=1)
Y=data['Price']
```

We will train and test the data using train_test_split ¶

In [147]:
```
from sklearn.model_selection import train_test_split,cross_val_score
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=1)
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(4001, 8)
(4001,)
(1001, 8)
(1001,)
```

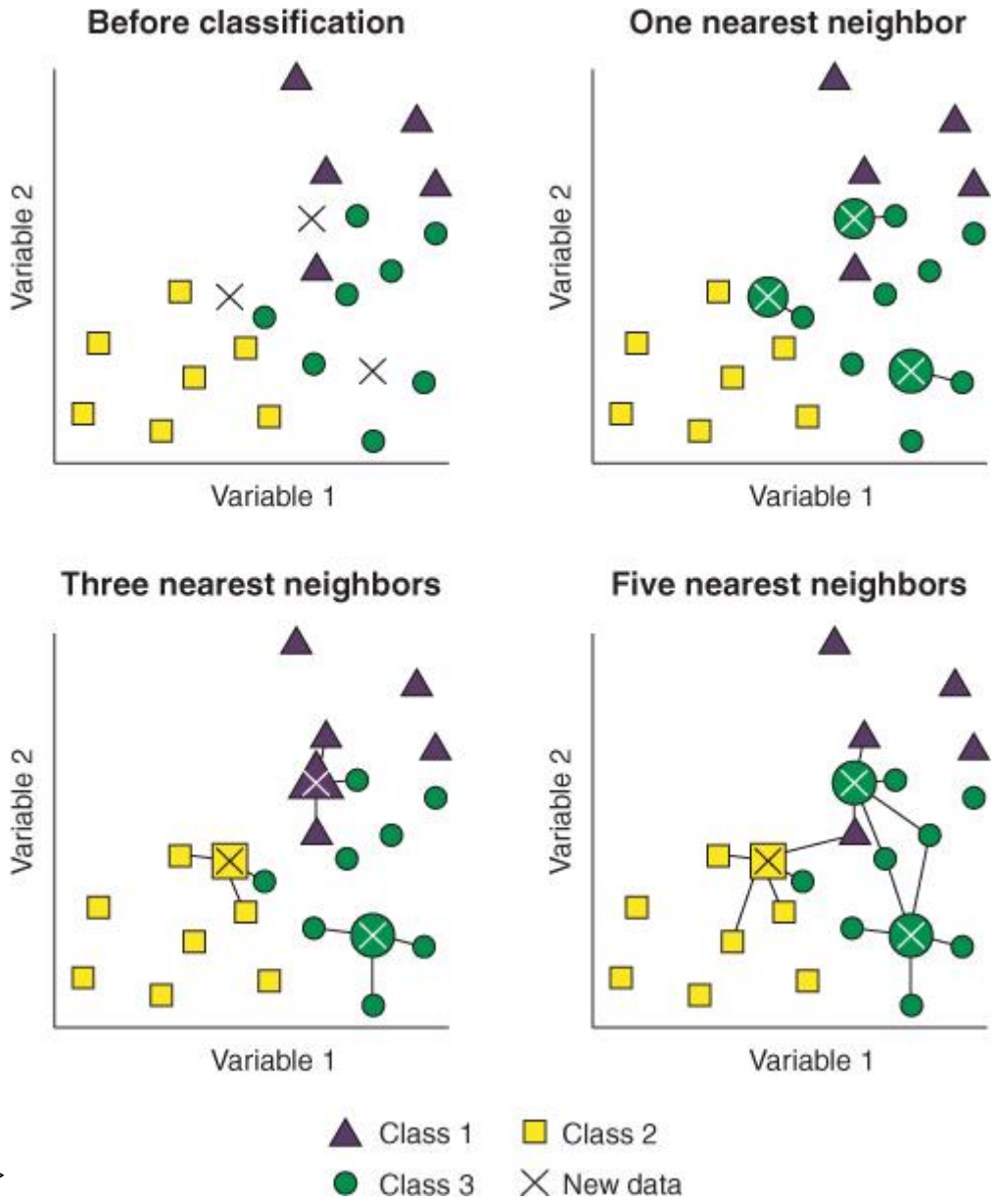
```
In [148]: from sklearn.tree import DecisionTreeRegressor
          from sklearn.neighbors import KNeighborsRegressor
          from sklearn.ensemble import RandomForestRegressor
          from xgboost import XGBRegressor
          from sklearn import model_selection
          models=[]
          models.append(('CART', DecisionTreeRegressor()))
          models.append(("KNN",KNeighborsRegressor()))
          models.append(("RF", RandomForestRegressor()))
          models.append(("XGBOOST", XGBRegressor()))
          names=[]
          result=[]
          for name,model in models:
              k_fold=model_selection.KFold(n_splits=10,shuffle=True,random_state=7)
              score=model_selection.cross_val_score(model,X_train,Y_train,cv=k_fold,scoring="r2")
              result.append(score)
              names.append(name)
              print(name,score.mean(),score.std())

          CART 0.9994081787592728 0.0009473854218247374
          KNN 0.6706865714083624 0.06211854331715496
          RF 0.9998922447476299 0.0001308740225973010 2
          XGBOOST 0.9998424092711369 9.728746871967122e-05
```

- We will build the model using Regressor like DecisionTreeRegressor,KNeighborRegressor, XGBRegressor and the RandomForestRegressor.
- Regressor is used to predict continuous variables.
- Random forest and XGBoost are two tree-based learners that create an ensemble of many trees to improve prediction accuracy.

Random forest trains many trees in parallel on different bootstrap samples from the data, and XGBoost trains sequential trees that prioritize misclassified cases.

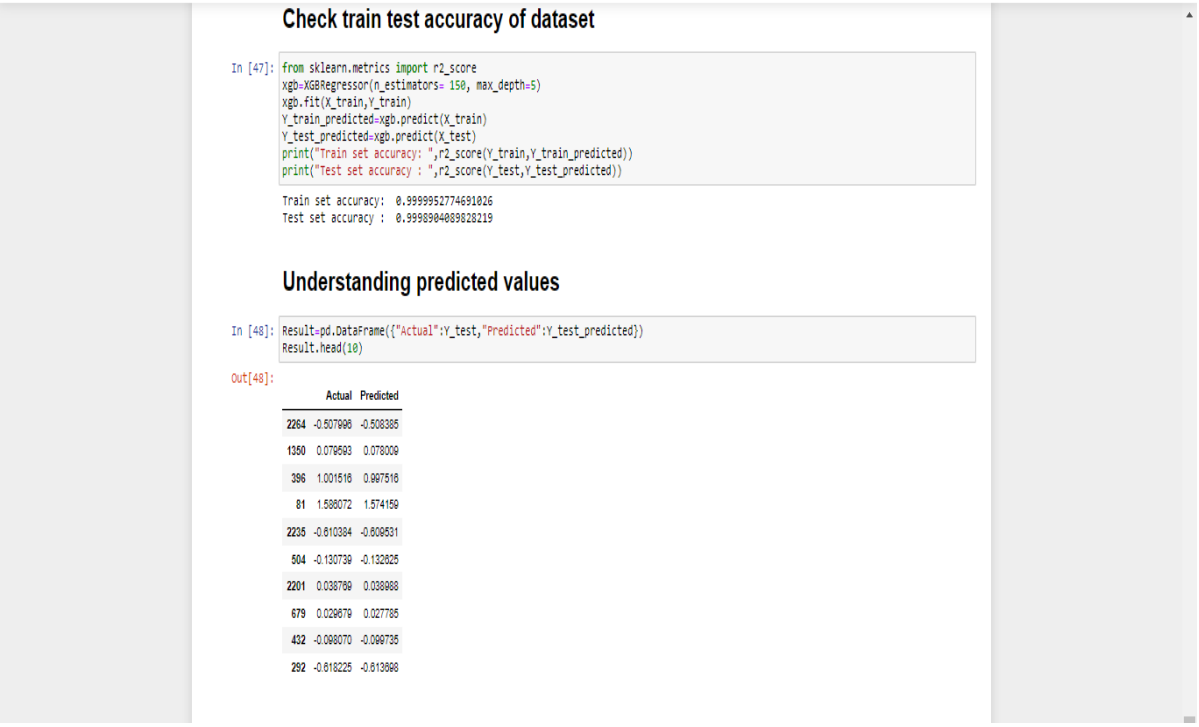
- The kNN algorithm is a lazy learner. In other words, it doesn't do any work during model training (instead, it just stores the training data); it does all of its work when it makes predictions. When making predictions, the kNN algorithm looks in the training set for the *k* cases most similar to each of the new, unlabeled data values. Each of those *k* most similar cases votes on the predicted value of the new data.
- We may want to evaluate a multioutput regression using k-fold cross-validation.
- This can be achieved in the same way as evaluating any other machine learning model.
- We will fit and evaluate a *DecisionTreeRegressor,KNNRegressor*. RandomForestRegressor,XGBRegressor model on the test problem using 10-fold cross-validation with seven repeats.

- We will use R2 scoring to evaluate the performance metric of the mean and the standard mean as the score.
-



-
- We will used RandomizedSearchCV for hyper-parameter tunning for XGBRegressor so that we can build the model with more accuracy.

- RandomizedSearchCV implements a “fit” and a “score” method. In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.
- The we will check the best parameter so we got n_estimator=150 and the max_depth=5



Check train test accuracy of dataset

In [47]:
```
from sklearn.metrics import r2_score
xgb=XGBRegressor(n_estimators= 150, max_depth=5)
xgb.fit(X_train,Y_train)
Y_train_predicted=xgb.predict(X_train)
Y_test_predicted=xgb.predict(X_test)
print("Train set accuracy: ",r2_score(Y_train,Y_train_predicted))
print("Test set accuracy : ",r2_score(Y_test,Y_test_predicted))
```

```
Train set accuracy:  0.9999952774691026
Test set accuracy :  0.9998904089828219
```

Understanding predicted values

In [48]:
```
Result=pd.DataFrame({"Actual":Y_test,"Predicted":Y_test_predicted})
Result.head(10)
```

Out[48]:

	Actual	Predicted
2264	-0.507996	-0.508385
1350	0.079593	0.078009
396	1.001516	0.997516
81	1.586072	1.574159
2235	-0.610384	-0.609531
504	-0.130739	-0.132625
2201	0.038769	0.038968
679	0.029679	0.027785
432	-0.098070	-0.099735
292	-0.618225	-0.613668

- Lastly we will save the model using joblib.
- We have successfully build our model.there is a scope for improvement if we have added some more columns in our dataset.