# # Part 2 : Machine Learning Model Building

In [1]:
```
import pandas as pd # for data wrangling purpose
import numpy as np # Basic computation library
import seaborn as sns # For Visualization
import matplotlib.pyplot as plt # ploting package
%matplotlib inline
import warnings # Filtering warnings
warnings.filterwarnings('ignore')
```

**Importing excel featured engineered Data from Part 1**

In [2]:
```
df= pd.read_excel('Final ML Data.xlsx')
```

In [9]:
```
df.head()
```

Out[9]:

| | Fuel Type | KMs driven | Engine Displacement(CC) | Transmission | Milage(kmpl) | Max Power(bhp) | Torque(Nm) | Seatir Capaci |
|---|---|---|---|---|---|---|---|---|
| **0** | 4 | 18600 | 1497 | 1 | 17.40 | 117.3 | 145.0 | |
| **1** | 1 | 15000 | 1956 | 1 | 17.10 | 170.0 | 350.0 | |
| **2** | 1 | 115000 | 2499 | 1 | 14.80 | 80.0 | 19.0 | |
| **3** | 4 | 80000 | 1497 | 1 | 16.80 | 116.4 | 146.0 | |
| **4** | 4 | 35000 | 1197 | 1 | 20.36 | 78.9 | 111.8 | |

5 rows × 25 columns

In [10]:
```
# Splitting data in target and dependent feature
X = df.drop(['Price (Rs.)'], axis = 1)
Y = df['Price (Rs.)']
```

In [11]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scale = scaler.fit_transform(X)
```

**Importing require Machine Learning Library**

In [14]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import ExtraTreesRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import  GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
```

In [15]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error,  r2_score
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42, tes
print('Training feature matrix size:',X_train.shape)
print('Training target vector size:',Y_train.shape)
print('Test feature matrix size:',X_test.shape)
print('Test target vector size:',Y_test.shape)
```

```
Training feature matrix size: (6218, 24)
Training target vector size: (6218,)
Test feature matrix size: (2666, 24)
Test target vector size: (2666,)
```

### Finding Best Random State

In [101]:
```
from sklearn.ensemble import RandomForestRegressor
maxAccu=0
maxRS=0
for i in range(1,200):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=i,
    mod = RandomForestRegressor()
    mod.fit(X_train, Y_train)
    pred = mod.predict(X_test)
    acc=r2_score(Y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.9655888552211401  on Random_state  114
```

## Random Forest Regressor Algorithim

In [104]:
```
RFR=RandomForestRegressor()
RFR.fit(X_train,Y_train)
pred=RFR.predict(X_test)
R2_score = r2_score(Y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(Y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(Y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(Y_test,pre

# Cross Validation Score
scores = cross_val_score(RFR, X, Y, cv = 5).mean()*100
print("\nCross validation score :", scores)

# Difference between Accuracy and CV Score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 96.46493782044703
mean_squared_error: 9226345022.16905
mean_absolute_error: 51153.49883627064
root_mean_squared_error: 96053.86521202076

Cross validation score : 93.03122692981853

R2_Score - Cross Validation Score : 3.433710890628504
```

## XGBRegressor ML Model

In [105]:
```
XGB=XGBRegressor()
XGB.fit(X_train,Y_train)
pred=XGB.predict(X_test)
R2_score = r2_score(Y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(Y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(Y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(Y_test,pre

# Cross Validation on XGB Model
scores = cross_val_score(XGB, X, Y, cv = 5).mean()*100
print("\nCross validation score :", scores)

# Difference between Accuracy and CV Score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 96.8578288022264
mean_squared_error: 8200918149.917081
mean_absolute_error: 50118.93210268505
root_mean_squared_error: 90558.92087429643

Cross validation score : 93.2469040953667

R2_Score - Cross Validation Score : 3.6109247068596915
```

## Gradient Boosting Regressor ML Model

In [107]:

```
GBR=GradientBoostingRegressor()
GBR.fit(X_train,y_train)
pred=GBR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(Y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(Y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(Y_test,pre

# Cross Validation on Gradient Boosting
scores = cross_val_score(GBR, X, Y, cv = 5).mean()*100
print("\nCross validation score :", scores)

# Difference between Accuracy and CV Score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 94.9328623763045
mean_squared_error: 13224989439.06563
mean_absolute_error: 71240.04884627696
root_mean_squared_error: 114999.95408288487

Cross validation score : 90.1937305617025

R2_Score - Cross Validation Score : 4.739131814602004
```

## Decision Tree Regressor ML Model

In [108]:
```
DTR=DecisionTreeRegressor()
DTR.fit(X_train,y_train)
pred=DTR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(Y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(Y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(Y_test,pre
# Cross Validation Score
scores = cross_val_score(DTR, X, Y, cv = 5).mean()*100
print("\nCross validation score :", scores)

# Difference between Accuracy and CV Score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 91.79408304824462
mean_squared_error: 21417054969.521046
mean_absolute_error: 64770.82728592162
root_mean_squared_error: 146345.6694594037

Cross validation score : 88.83907795864332

R2_Score - Cross Validation Score : 2.9550050896013005
```

### Bagging Regressor ML Model

In [109]:
```
BR=BaggingRegressor()
BR.fit(X_train,Y_train)
pred=BR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(Y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(Y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(Y_test,pre

# Cross Validation Score
scores = cross_val_score(BR, X, Y, cv = 5).mean()*100
print("\nCross validation score :", scores)

# Difference between Accuracy and CV Score
diff = R2_score - scores
print("\nR2_Score - Cross Validation Score :", diff)
```

```
R2_score: 95.67561886178403
mean_squared_error: 11286430156.537165
mean_absolute_error: 55118.9760619255
root_mean_squared_error: 106237.61177914894

Cross validation score : 92.60347410600774

R2_Score - Cross Validation Score : 3.072144755776293
```

### Final model Selection

**All model are giving us R2 score & Cross validation Score more than 90%, So we will select model which has less difference between these score.**

**<u>On Basis of difference between R2 Score and Cross Validation Score Decision Tree Regressor</u> is selected as best model with 91.79% r2_score.**

**We will perform Hyper Parameter tuning over this model**

# Hyper Parameter Tunning

In [110]:
```
#importing necessary libraries
from sklearn.model_selection import GridSearchCV
```

In [111]:
```
parameter = {'criterion':['squared_error', 'friedman_mse', 'absolute_error', '
             'splitter':['best','random'],
             'max_features':['auto','sqrt','log2'],
             'min_samples_split':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
             'max_depth':[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]}
```

Giving DecisionTreeRegressor parameters.

In [112]:
```
GCV=GridSearchCV(DecisionTreeRegressor(),parameter,cv=5)
```

Running grid search CV for decisionTreesRegressor.

In [113]:
```
DecisionTreeRegressorGCV.fit(X_train,y_train)
```

Out[113]:
```
GridSearchCV(cv=10, estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['squared_error', 'friedman_mse',
                                       'absolute_error', 'poisson'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                       13, 14, 15],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                                               11, 12, 13, 14, 15],
                         'splitter': ['best', 'random']})
```

Tunning the model using GCV.

In [114]:
```
GCV.best_params_
```

Out[114]:
```
{'criterion': 'friedman_mse',
 'max_depth': 13,
 'max_features': 'auto',
 'min_samples_split': 4,
 'splitter': 'random'}
```

# Final Model

In [115]:
```
Final_mod=DecisionTreeRegressor(criterion='friedman_mse', max_depth=15, max_fe
                                min_samples_split=4, splitter='random')
Final_mod.fit(X_train,Y_train)
pred=Final_mod.predict(X_test)
print('R2_Score:',r2_score(Y_test,pred)*100)
print('mean_squared_error:',metrics.mean_squared_error(Y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(Y_test,pred))
print("RMSE value:",np.sqrt(metrics.mean_squared_error(Y_test, pred)))
```

```
R2_Score: 92.28906442588051
mean_squared_error: 20125176994.634956
mean_absolute_error: 70467.88619495885
RMSE value: 141863.2334138587
```

**Final Model is giving us R2 Score of 92.29% which is slightly improved compare to earlier R2 score of 91.79%.**

# Saving the model

In [116]:
```
# Saving the model using .pkl
import joblib
joblib.dump(Final_mod,"Car_Price.pkl")
```

Out[116]:
```
['Car_Price.pkl']
```

# Predictions Using Final Model

In [117]:
```
# Loading the saved model
model=joblib.load("Car_Price.pkl")

#Prediction
prediction = model.predict(X_test)
prediction
```

Out[117]:
```
array([ 379000.        , 1650000.        ,  355000.        , ...,
        556333.33333333,  199923.07692308,  364800.        ])
```

In [118]:
```
pd.DataFrame([model.predict(X_test)[:],y_test[:]],index=["Predicted","Actual"]
```

Out[118]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| **Predicted** | 379000.0 | 1650000.0 | 355000.0 | 332666.666667 | 490333.333333 | 470382.352941 | 590000.0 |
| **Actual** | 379000.0 | 1650000.0 | 465000.0 | 435000.000000 | 550000.000000 | 450000.000000 | 550000.0 |