Car Price Prediction With Machine Learning

Import important Libraries and Dataset

In [1]:

```
import warnings
```

In [2]:

```
warnings.filterwarnings('ignore')
```

In [3]:

```
import pandas as pd
```

In [4]:

```
data = pd.read_csv("C:\\Users\\91989\\Downloads\\raw.githubusercontent.com_amankharwal_Website-data_master_CarPrice.csv"
```

In [5]:

```
data
```

Out[5]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	engines
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	

205 rows × 26 columns

Top 5 Rows of the Dataset

In [6]:

```
data.head()
```

Out[6]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesiz
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	13
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	13
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	15
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	10
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	13

5 rows × 26 columns

Last 5 Rows of the Dataset

In [7]:

```
data.tail()
```

Out[7]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141

5 rows × 26 columns

In [8]:

```
data.shape
```

Out[8]:

```
(205, 26)
```

In [9]:

```
print("Number of rows",data.shape[0])
print("Number of columns",data.shape[1])
```

```
Number of rows 205
Number of columns 26
```

In [10]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [11]:

```
data.isnull().sum()
```

Out[11]:

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

Get Overall Statistics of the Dataset

In [12]:

```
data.describe()
```

Out[12]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	co
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	

Outlier Removal

In [13]:

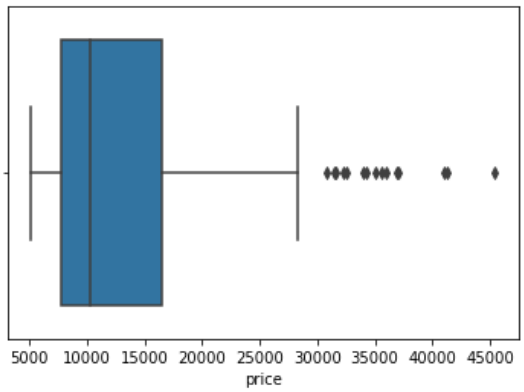
```
import seaborn as sns
```

In [14]:

```
sns.boxplot(data['price'])
```

Out[14]:

```
<AxesSubplot:xlabel='price'>
```



In [15]:

```
sorted(data['price'],reverse=True)
```

Out[15]:

```
[45400.0,
 41315.0,
 40960.0,
 37028.0,
 36880.0,
 36000.0,
 35550.0,
 35056.0,
 34184.0,
 34028.0,
 32528.0,
 32250.0,
 31600.0,
 31400.5,
 30760.0,
 28248.0,
 28176.0,
 25552.0,
```

In [16]:

```
newdata=data[~(data['price']>40000)&(data['price']<50000)]
newdata
```

Out[16]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	engines
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	
...	...	...	...	...	...	...	...	...	...	...	...	
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	

202 rows × 26 columns

In [17]:

```
newdata=newdata.drop(['car_ID','symboling','aspiration','drivewheel','wheelbase','boreratio','compressionratio','stroke'
```

In [18]:

```
newdata.head()
```

Out[18]:

	CarName	fueltype	doornumber	carbody	enginelocation	enginesize	horsepower	peakrpm	citympg	highwaympg	price
0	alfa-romero giulia	gas	two	convertible	front	130	111	5000	21	27	13495.0
1	alfa-romero stelvio	gas	two	convertible	front	130	111	5000	21	27	16500.0
2	alfa-romero Quadrifoglio	gas	two	hatchback	front	152	154	5000	19	26	16500.0
3	audi 100 ls	gas	four	sedan	front	109	102	5500	24	30	13950.0
4	audi 100ls	gas	four	sedan	front	136	115	5500	18	22	17450.0

Encoding the Categorical Values

In [19]:

```
newdata['fueltype'].unique()
```

Out[19]:

```
array(['gas', 'diesel'], dtype=object)
```

In [20]:

```
newdata['fueltype']=newdata['fueltype'].map({'gas':0,'diesel':1})
```

In [21]:

```
newdata['fueltype'].unique()
```

Out[21]:

```
array([0, 1], dtype=int64)
```

In [22]:

```
newdata['doornumber'].unique()
```

Out[22]:

```
array(['two', 'four'], dtype=object)
```

In [23]:

```
newdata['doornumber']=newdata['doornumber'].map({'two':0,'four':1})
```

In [24]:

```
newdata['doornumber'].unique()
```

Out[24]:

```
array([0, 1], dtype=int64)
```

In [25]:

```
newdata['carbody'].unique()
```

Out[25]:

```
array(['convertible', 'hatchback', 'sedan', 'wagon', 'hardtop'],
      dtype=object)
```

In [26]:

```
newdata['carbody']=newdata['carbody'].map({'convertible':0,'hatchback':1,'sedan':2,'wagon':3,'hardtop':4})
```

In [27]:

```
newdata['carbody'].unique()
```

Out[27]:

```
array([0, 1, 2, 3, 4], dtype=int64)
```

In [28]:

```
newdata['enginelocation'].unique()
```

Out[28]:

```
array(['front', 'rear'], dtype=object)
```

In [29]:

```
newdata['enginelocation']=newdata['enginelocation'].map({'front':0,'rear':1})
```

In [30]:

```
newdata['enginelocation'].unique()
```

Out[30]:

```
array([0, 1], dtype=int64)
```

In [31]:

```
newdata.head()
```

Out[31]:

	CarName	fueltype	doornumber	carbody	enginelocation	enginesize	horsepower	peakrpm	citympg	highwaympg	price
0	alfa-romero giulia	0	0	0	0	130	111	5000	21	27	13495.0
1	alfa-romero stelvio	0	0	0	0	130	111	5000	21	27	16500.0
2	alfa-romero Quadrifoglio	0	0	1	0	152	154	5000	19	26	16500.0
3	audi 100 ls	0	1	2	0	109	102	5500	24	30	13950.0
4	audi 100ls	0	1	2	0	136	115	5500	18	22	17450.0

Store feature matrix in X and reponse (target) in vector Y

In [32]:

```
x=newdata.drop(['CarName','price'],axis=1)
y=newdata['price']
```

In [33]:

```
x
```

Out[33]:

	fueltype	doornumber	carbody	enginelocation	enginesize	horsepower	peakrpm	citympg	highwaympg
0	0	0	0	0	130	111	5000	21	27
1	0	0	0	0	130	111	5000	21	27
2	0	0	1	0	152	154	5000	19	26
3	0	1	2	0	109	102	5500	24	30
4	0	1	2	0	136	115	5500	18	22
...	...	...	...	...	...	...	...	...	...
200	0	1	2	0	141	114	5400	23	28
201	0	1	2	0	141	160	5300	19	25
202	0	1	2	0	173	134	5500	18	23
203	1	1	2	0	145	106	4800	26	27
204	0	1	2	0	141	114	5400	19	25

202 rows × 9 columns

In [34]:

```
y
```

Out[34]:

```
0      13495.0
1      16500.0
2      16500.0
3      13950.0
4      17450.0
        ...
200    16845.0
201    19045.0
202    21485.0
203    22470.0
204    22625.0
Name: price, Length: 202, dtype: float64
```

Splitting Dataset into the Training set and Test set

In [35]:

```
from sklearn.model_selection import train_test_split
```

In [36]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=42)
```

In [37]:

```
newdata.head()
```

Out[37]:

	CarName	fueltype	doornumber	carbody	enginelocation	enginesize	horsepower	peakrpm	citympg	highwaympg	price
0	alfa-romero giulia	0	0	0	0	130	111	5000	21	27	13495.0
1	alfa-romero stelvio	0	0	0	0	130	111	5000	21	27	16500.0
2	alfa-romero Quadrifoglio	0	0	1	0	152	154	5000	19	26	16500.0
3	audi 100 ls	0	1	2	0	109	102	5500	24	30	13950.0
4	audi 100ls	0	1	2	0	136	115	5500	18	22	17450.0

Import the Models

In [38]:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

Model Training

In [39]:

```
lr = LinearRegression()
lr.fit(x_train,y_train)

rf = RandomForestRegressor()
rf.fit(x_train,y_train)

xgb = GradientBoostingRegressor()
xgb.fit(x_train,y_train)

xg = XGBRegressor()
xg.fit(x_train,y_train)
```

Out[39]:

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

Prediction on Test data

In [40]:

```
y_pred1=lr.predict(x_test)
y_pred2=rf.predict(x_test)
y_pred3=xgb.predict(x_test)
y_pred4=xg.predict(x_test)
```

Evaluating the model

In [41]:

```
from sklearn import metrics
```

In [42]:

```
score1 = metrics.r2_score(y_test,y_pred1)
score2 = metrics.r2_score(y_test,y_pred2)
score3 = metrics.r2_score(y_test,y_pred3)
score4 = metrics.r2_score(y_test,y_pred4)
```

In [43]:

```
print(score1,score2,score3,score4)
```

```
0.8060703572204628 0.870806567576798 0.8642957337787818 0.8077269219176914
```

In [44]:

```
final_data=pd.DataFrame({'Models':['LR','RF','GBR','XG'],'R2_Score':[score1,score2,score3,score4]})
```

In [45]:

```
final_data
```

Out[45]:

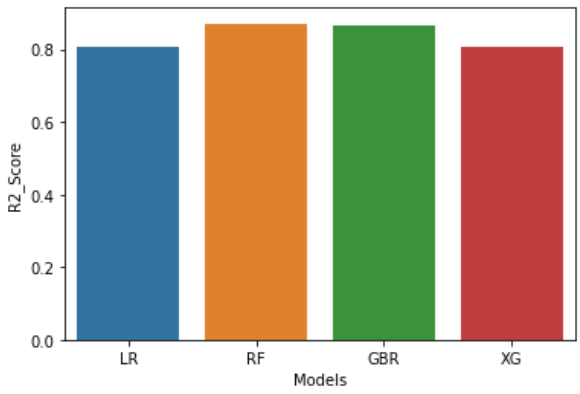
	Models	R2_Score
0	LR	0.806070
1	RF	0.870807
2	GBR	0.864296
3	XG	0.807727

In [46]:

```
sns.barplot(final_data['Models'],final_data['R2_Score'])
```

Out[46]:

```
<AxesSubplot:xlabel='Models', ylabel='R2_Score'>
```



Save the Model

In [47]:

```
xgb = GradientBoostingRegressor()
XGB_Final=xgb.fit(x,y)
```

In [48]:

```
import joblib
```

In [49]:

```
joblib.dump(XGB_Final,'car_price_predictor')
```

Out[49]:

```
['car_price_predictor']
```

In [50]:

```
Model = joblib.load('car_price_predictor')
```

Prediction on new data

In [51]:

```
import pandas as pd
data_new = pd.DataFrame({
    'fueltype':1,
    'doornumber':1,
    'carbody':3,
    'enginelocation':0,
    'enginesize':130,
    'horsepower':200,
    'peakrpm':5000,
    'citympg':20,
    'highwaympg':27,
},index=[0])
```

In [52]:

```
Model.predict(data_new)
```

Out[52]:

```
array([19452.31017606])
```

In [ ]:

```

```