Old Car Price Prediction using Machine Learning in Python

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv('car data.xls')
df.head()
```

Out[2]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmissi
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Man
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Man
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Man
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Man
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Man

Find Shape of Our Dataset (Number of Rows And Number of Columns)

In [3]:
```
df.shape
```

Out[3]:
```
(301, 9)
```

In [4]:
```
df.nunique()
```

Out[4]:
```
Car_Name          98
Year              16
Selling_Price    156
Present_Price    147
Kms_Driven       206
Fuel_Type          3
Seller_Type        2
Transmission       2
Owner              3
dtype: int64
```

In [5]:
```
df["Transmission"].unique()
```

Out[5]:
```
array(['Manual', 'Automatic'], dtype=object)
```

Get Information About Our Dataset Like the Total Number of Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

Check Null Values In The Dataset

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

Get Overall Statistics About The Dataset

In [8]:
```
df.describe()
```

Out[8]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

Data Preprocessing

In [9]:
```
import datetime
```

In [27]:
```
date_time = datetime.datetime.now()
date_time
```

Out[27]:
```
datetime.datetime(2024, 4, 20, 15, 55, 6, 392768)
```

In [28]:
```
df['Age']=date_time.year - df['Year']
```

In [30]:
```
df.head()
```

Out[30]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmissi
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Man
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Man
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Man
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Man
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Man

In [31]:
```
df.drop(["Year"],axis=1,inplace=True)
```

In [32]:
```
df.head()
```

Out[32]:

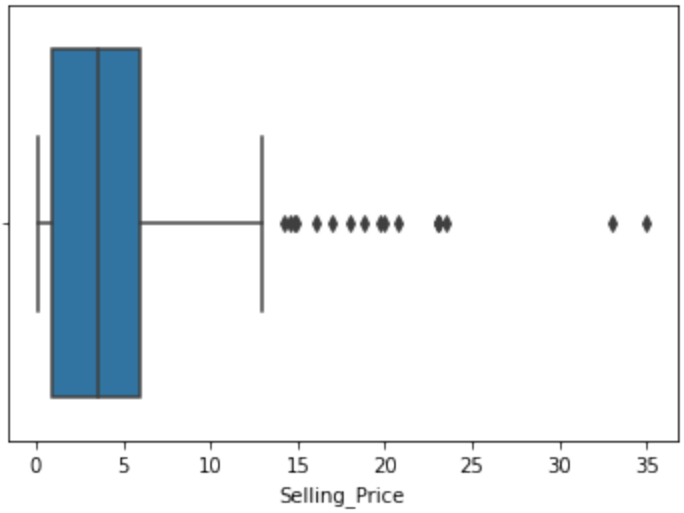
	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	O
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	
1	sx4	4.75	9.54	43000	Diesel	Dealer	Manual	
2	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual	
3	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual	
4	swift	4.60	6.87	42450	Diesel	Dealer	Manual	

Outlier Removal

In [33]:
```
sns.boxplot(df["Selling_Price"])
```

```
C:\Users\lax\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[33]: <AxesSubplot:xlabel='Selling_Price'>



In [34]:
```
sorted(df["Selling_Price"],reverse=True)
```

Out[34]:
```
[35.0,
 33.0,
 23.5,
 23.0,
 23.0,
 23.0,
 20.75,
 19.99,
 19.75,
 18.75,
 18.0,
 17.0,
 16.0,
 14.9,
 14.73,
 14.5,
 14.25,
 12.9,
 12.5,
```

In [27]:
```
df=df[~(df["Selling_Price"]>=33.0) & (df["Selling_Price"]<=35.0)]
df.head(5)
```

Out[27]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	O
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	
1	sx4	4.75	9.54	43000	Diesel	Dealer	Manual	
2	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual	
3	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual	
4	swift	4.60	6.87	42450	Diesel	Dealer	Manual	

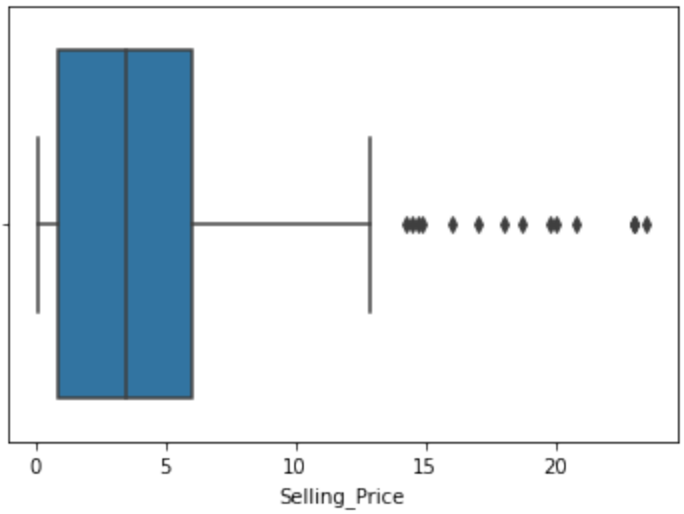
In [28]:
```
df.shape
```

Out[28]:
```
(299, 9)
```

In [29]:
```
sns.boxplot(df["Selling_Price"])
```

```
C:\Users\lax\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments wi
thout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[29]: <AxesSubplot:xlabel='Selling_Price'>



In [32]:
```
obj=(df.dtypes=="object")
cols=list(obj[obj].index)
print(cols)
print(len(cols))
```

```
['Car_Name', 'Fuel_Type', 'Seller_Type', 'Transmission']
4
```

In [37]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
for cols in list(obj[obj].index):
    df[cols]=label_encoder.fit_transform(df[cols])
```

In [38]:
```
df.head()
```

Out[38]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	O
0	89	3.35	5.59	27000	2	0	1	
1	92	4.75	9.54	43000	1	0	1	
2	68	7.25	9.85	6900	2	0	1	
3	95	2.85	4.15	5200	2	0	1	
4	91	4.60	6.87	42450	1	0	1	

Store Feature Matrix In X and Response(Target) In Vector y

In [41]:
```
from sklearn.model_selection import train_test_split

X = df.drop(['Car_Name','Selling_Price'],axis=1)
y = df['Selling_Price']
```

Splitting The Dataset Into The Training Set And Test Set

In [42]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

print('Shape of X_train = ', X_train.shape)
print('Shape of y_train = ', y_train.shape)
print('Shape of X_test = ', X_test.shape)
print('Shape of y_test = ', y_test.shape)
```

```
Shape of X_train =  (239, 7)
Shape of y_train =  (239,)
Shape of X_test =  (60, 7)
Shape of y_test =  (60,)
```

Import The models

In [43]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

Model Training

In [44]:
```
lr=LinearRegression()
lr.fit(X_train,y_train)

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

gbr=GradientBoostingRegressor()
gbr.fit(X_train,y_train)

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

Out[44]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=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,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

Prediction on Test Data

In [45]:
```
y_pred1=lr.predict(X_test)
y_pred2=rf.predict(X_test)
y_pred3=gbr.predict(X_test)
y_pred4=xg.predict(X_test)
```

Evaluating the Algorithm¶

In [47]:
```
from sklearn import metrics
```

In [48]:
```
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 [49]:
```
print(score1,score2,score3,score4)
```

```
0.8962148379693471 0.9678360967953072 0.9709615167440474 0.9574538744570458
```

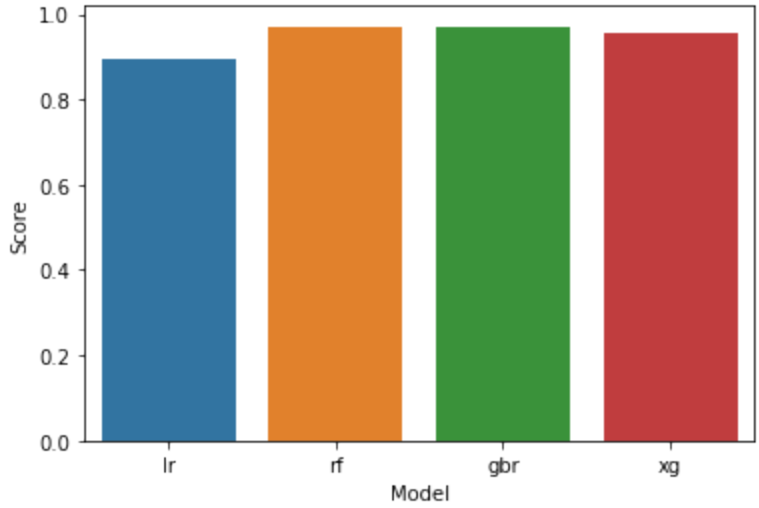
In [53]:
```
final_data=pd.DataFrame({"Model":["lr","rf","gbr","xg"],"Score":[score1,score2,
final_data
```

Out[53]:

	Model	Score
0	lr	0.896215
1	rf	0.967836
2	gbr	0.970962
3	xg	0.957454

In [54]:
```
sns.barplot(x=final_data["Model"],y=final_data["Score"])
```

Out[54]:
```
<AxesSubplot:xlabel='Model', ylabel='Score'>
```



Save The Model

In [55]:
```
xg = XGBRegressor()
xg_final = xg.fit(X,y)
```