In [1]:
```
import pandas as pd
import numpy as np
```

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

Selling_Price will be our Target variable

In [3]:
```
df.head()
```

Out[3]:

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

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

Out[4]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owne
296	city	2016	9.50	11.6	33988	Diesel	Dealer	Manual	
297	brio	2015	4.00	5.9	60000	Petrol	Dealer	Manual	
298	city	2009	3.35	11.0	87934	Petrol	Dealer	Manual	
299	city	2017	11.50	12.5	9000	Diesel	Dealer	Manual	
300	brio	2016	5.30	5.9	5464	Petrol	Dealer	Manual	

In [5]:
```
df.shape
```

Out[5]:
```
(301, 9)
```

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

Out[6]:

	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

In [7]:
```
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
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
df.head(1)
```

Out[9]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0

In [10]:
```
import datetime
```

In [11]:
```
date_time=datetime.datetime.now()
```

In [12]:
```
df['car_Age']=date_time.year-df['Year']
```

In [13]:
```
df.head()
```

Out[13]:

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

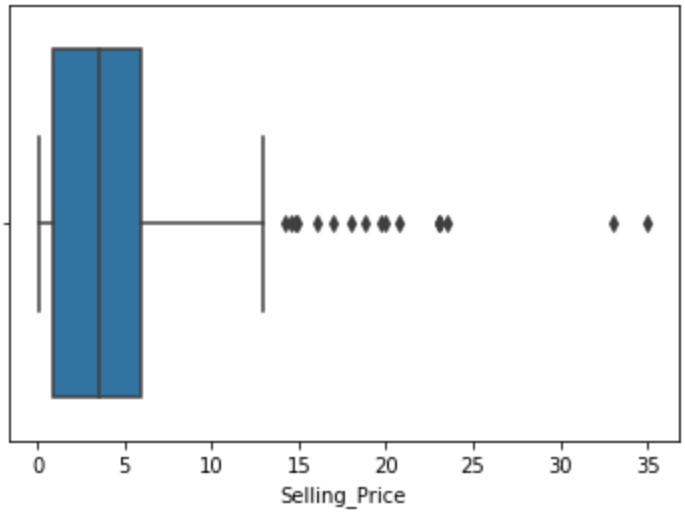
In [14]:
```
df.drop('Year',axis=1,inplace=True)
```

In [15]:
```
import seaborn as sns
```

```
D:\Users\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning: pand
as.util.testing is deprecated. Use the functions in the public API at pandas.tes
ting instead.
  import pandas.util.testing as tm
```

In [16]:
```
sns.boxplot(df['Selling_Price'])
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a89cfbeac8>
```



In [17]:
```
sorted(df['Selling_Price'],reverse=True)
```

Out[17]:
```
[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,
 11.75,
 11.5,
 11.45,
 11.25,
 11.25,
 11.25,
 10.9,
 10.25,
 10.11,
 9.7,
 9.65,
 9.5,
 9.25,
 9.25,
 9.25,
 9.15,
 9.1,
 8.99,
 8.75,
 8.65,
 8.55,
 8.5,
 8.4,
 8.4,
 8.35,
 8.25,
 8.25,
 7.9,
 7.75,
 7.75,
 7.75,
 7.5,
 7.5,
 7.5,
 7.45,
 7.45,
 7.45,
 7.4,
 7.25,
 7.25,
 7.2,
 7.05,
```

6.95,
6.85,
6.75,
6.7,
6.6,
6.5,
6.5,
6.45,
6.4,
6.25,
6.25,
6.15,
6.1,
6.0,
6.0,
6.0,
6.0,
5.95,
5.95,
5.9,
5.85,
5.85,
5.8,
5.75,
5.75,
5.65,
5.5,
5.5,
5.5,
5.5,
5.5,
5.4,
5.4,
5.35,
5.3,
5.3,
5.25,
5.25,
5.25,
5.25,
5.25,
5.25,
5.25,
5.2,
5.15,
5.11,
5.0,
4.95,
4.95,
4.9,
4.9,
4.85,
4.8,
4.8,
4.75,
4.75,
4.75,
4.75,
4.75,
4.75,
4.65,
4.6,

4.5,
4.5,
4.5,
4.5,
4.5,
4.5,
4.5,
4.4,
4.4,
4.4,
4.35,
4.15,
4.1,
4.1,
4.0,
4.0,
4.0,
4.0,
4.0,
3.95,
3.95,
3.9,
3.9,
3.8,
3.75,
3.75,
3.65,
3.6,
3.51,
3.5,
3.5,
3.49,
3.45,
3.35,
3.35,
3.25,
3.25,
3.25,
3.15,
3.1,
3.1,
3.1,
3.1,
3.0,
3.0,
3.0,
3.0,
2.95,
2.95,
2.9,
2.9,
2.9,
2.85,
2.85,
2.85,
2.75,
2.75,
2.7,
2.65,
2.65,
2.65,
2.55,

2.55,
2.5,
2.5,
2.35,
2.25,
2.25,
2.25,
2.1,
2.0,
1.95,
1.95,
1.75,
1.7,
1.65,
1.5,
1.45,
1.35,
1.35,
1.35,
1.25,
1.25,
1.2,
1.2,
1.2,
1.15,
1.15,
1.15,
1.15,
1.11,
1.1,
1.1,
1.1,
1.05,
1.05,
1.05,
1.05,
1.05,
1.0,
0.95,
0.9,
0.9,
0.8,
0.78,
0.75,
0.75,
0.75,
0.75,
0.72,
0.65,
0.65,
0.65,
0.65,
0.6,
0.6,
0.6,
0.6,
0.6,
0.6,
0.6,
0.6,
0.55,
0.55,

```
0.52,
0.51,
0.5,
0.5,
0.5,
0.5,
0.5,
0.48,
0.48,
0.48,
0.48,
0.45,
0.45,
0.45,
0.45,
0.45,
0.45,
0.45,
0.45,
0.45,
0.42,
0.42,
0.4,
0.4,
0.4,
0.4,
0.4,
0.38,
0.38,
0.35,
0.35,
0.35,
0.35,
0.31,
0.3,
0.3,
0.3,
0.27,
0.25,
0.25,
0.25,
0.25,
0.25,
0.25,
0.2,
0.2,
0.2,
0.2,
0.2,
0.2,
0.18,
0.17,
0.16,
0.15,
0.12,
0.1]
```

In [18]:
```
df = df[~(df['Selling_Price']>=33.0) & (df['Selling_Price']<=35.0)]
```

In [19]:
```
df.shape
```

Out[19]:
```
(299, 9)
```

In [20]:
```
df.head(1)
```

Out[20]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	car_A
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual	0	

In [21]:
```
df['Car_Name'].unique()
```

Out[21]:
```
array(['ritz', 'sx4', 'ciaz', 'wagon r', 'swift', 'vitara brezza',
       's cross', 'alto 800', 'ertiga', 'dzire', 'alto k10', 'ignis',
       '800', 'baleno', 'omni', 'fortuner', 'innova', 'corolla altis',
       'etios cross', 'etios g', 'etios liva', 'corolla', 'etios gd',
       'camry', 'Royal Enfield Thunder 500', 'UM Renegade Mojave',
       'KTM RC200', 'Bajaj Dominar 400', 'Royal Enfield Classic 350',
       'KTM RC390', 'Hyosung GT250R', 'Royal Enfield Thunder 350',
       'KTM 390 Duke ', 'Mahindra Mojo XT300', 'Bajaj Pulsar RS200',
       'Royal Enfield Bullet 350', 'Royal Enfield Classic 500',
       'Bajaj Avenger 220', 'Bajaj Avenger 150', 'Honda CB Hornet 160R',
       'Yamaha FZ S V 2.0', 'Yamaha FZ 16', 'TVS Apache RTR 160',
       'Bajaj Pulsar 150', 'Honda CBR 150', 'Hero Extreme',
       'Bajaj Avenger 220 dtsi', 'Bajaj Avenger 150 street',
       'Yamaha FZ  v 2.0', 'Bajaj Pulsar  NS 200', 'Bajaj Pulsar 220 F',
       'TVS Apache RTR 180', 'Hero Passion X pro', 'Bajaj Pulsar NS 200',
       'Yamaha Fazer ', 'Honda Activa 4G', 'TVS Sport ',
       'Honda Dream Yuga ', 'Bajaj Avenger Street 220',
       'Hero Splender iSmart', 'Activa 3g', 'Hero Passion Pro',
       'Honda CB Trigger', 'Yamaha FZ S ', 'Bajaj Pulsar 135 LS',
       'Activa 4g', 'Honda CB Unicorn', 'Hero Honda CBZ extreme',
       'Honda Karizma', 'Honda Activa 125', 'TVS Jupyter',
       'Hero Honda Passion Pro', 'Hero Splender Plus', 'Honda CB Shine',
       'Bajaj Discover 100', 'Suzuki Access 125', 'TVS Wego',
       'Honda CB twister', 'Hero Glamour', 'Hero Super Splendor',
       'Bajaj Discover 125', 'Hero Hunk', 'Hero  Ignitor Disc',
       'Hero  CBZ Xtreme', 'Bajaj  ct 100', 'i20', 'grand i10', 'i10',
       'eon', 'xcent', 'elantra', 'creta', 'verna', 'city', 'brio',
       'amaze', 'jazz'], dtype=object)
```

In [22]:
```
df['Fuel_Type'].unique()
```

Out[22]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [23]:
```
df['Fuel_Type']=df['Fuel_Type'].map({'Petrol':1,'Diesel':2,'CNG':3}).astype(in
t)
```

In [24]:
```
df['Seller_Type'].unique()
```

Out[24]:
```
array(['Dealer', 'Individual'], dtype=object)
```

In [25]:
```
df['Seller_Type']=df['Seller_Type'].map({'Dealer':1,'Individual':2}).astype(in
t)
```

In [26]:
```
df['Transmission'].unique()
```

Out[26]:
```
array(['Manual', 'Automatic'], dtype=object)
```

In [27]:
```
df['Transmission']=df['Transmission'].map({'Manual':1,'Automatic':2}).astype(in
t)
```

In [28]:
```
df.head()
```

Out[28]:

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

In [29]:
```
df.drop('Car_Name',axis=1,inplace=True)
```

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

Out[30]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	car_Age
0	3.35	5.59	27000	1	1	1	0	10
1	4.75	9.54	43000	2	1	1	0	11
2	7.25	9.85	6900	1	1	1	0	7
3	2.85	4.15	5200	1	1	1	0	13
4	4.60	6.87	42450	2	1	1	0	10

In [31]:
```
X = df.drop(['Selling_Price'],axis=1)
y = df['Selling_Price']
```

In [32]:
```
print(X)
print(y)
```

```
     Present_Price  Kms_Driven  Fuel_Type  Seller_Type  Transmission  Owner  \
0             5.59       27000          1            1             1      0
1             9.54       43000          2            1             1      0
2             9.85        6900          1            1             1      0
3             4.15        5200          1            1             1      0
4             6.87       42450          2            1             1      0
..             ...         ...        ...          ...           ...    ...
296          11.60       33988          2            1             1      0
297           5.90       60000          1            1             1      0
298          11.00       87934          1            1             1      0
299          12.50        9000          2            1             1      0
300           5.90        5464          1            1             1      0

     car_Age
0         10
1         11
2          7
3         13
4         10
..       ...
296        8
297        9
298       15
299        7
300        8

[299 rows x 7 columns]
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 299, dtype: float64
```

In [33]:
```
#from sklearn.model_selection import train_test_split
```

In [34]:
```
#X_test,y_test,X_train,y_train=train_test_split(X,y,test_size=0.20,random_state
=42)
```

In [35]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

In [36]:
```
#X_train = X_train.values.reshape(-1, 1) if len(X_train.shape) == 1 else X_trai
n.values
```

In [37]:
```
lr = LinearRegression()
lr.fit(X, y)

rf = RandomForestRegressor()
rf.fit(X, y)

xgb = GradientBoostingRegressor()
xgb.fit(X, y)

xg = XGBRegressor()
xg.fit(X, y)
```

Out[37]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In [38]:
```
y_pred_lr=lr.predict(X)
y_pred_rf=rf.predict(X)
y_pred_xbg=xgb.predict(X)
y_pred_xg=xg.predict(X)
```

In [39]:
```
from sklearn import metrics
```

In [40]:
```
score1 = metrics.r2_score(y,y_pred_lr)
score2 = metrics.r2_score(y,y_pred_rf)
score3 = metrics.r2_score(y,y_pred_xbg)
score4 = metrics.r2_score(y,y_pred_xg)
```

In [41]:
```
print(score1,score2,score3,score4)
```

```
0.878075829051472 0.991700509938964 0.9948207860112457 0.9999878414621973
```

In [42]:
```
final_data = pd.DataFrame({'Models':['LR','RF','GBR','XG'],
             "R2_SCORE":[score1,score2,score3,score4]})
```

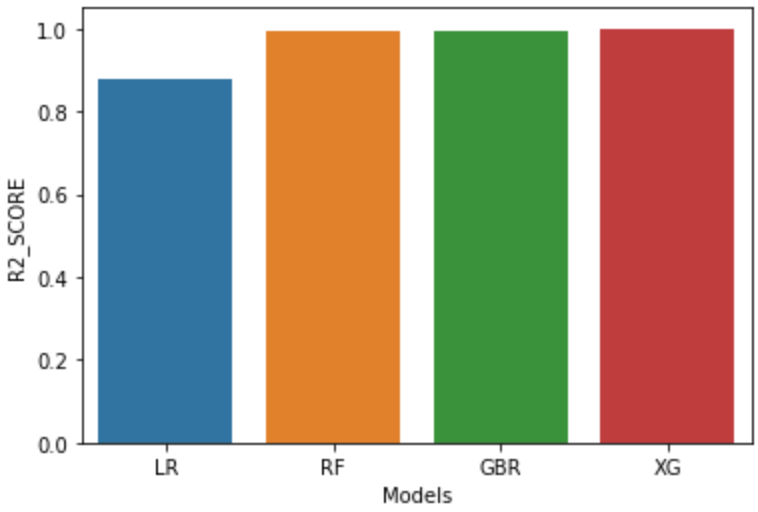
In [43]:
```
final_data
```

Out[43]:

	Models	R2_SCORE
0	LR	0.878076
1	RF	0.991701
2	GBR	0.994821
3	XG	0.999988

In [44]:
```
sns.barplot(final_data['Models'],final_data['R2_SCORE'])
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a89f9dff48>
```



In [45]:
```
xg = XGBRegressor()
xg= xg.fit(X,y)
```

In [46]:
```
data_input = pd.DataFrame({
    'Present_Price':5.59,
    'Kms_Driven':27000,
    'Fuel_Type':0,
    'Seller_Type':0,
    'Transmission':0,
    'Owner':0,
    'Age':8
},index=[0])
```

In [47]:
```
data_input
```

Out[47]:

	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	5.59	27000	0	0	0	0	8

In [48]:
```
xg.predict(data_input)
```

Out[48]:
```
array([3.7697704], dtype=float32)
```

In [49]:
```
data_input2 = pd.DataFrame({
    'Present_Price':4.59,
    'Kms_Driven':57000,
    'Fuel_Type':1,
    'Seller_Type':1,
    'Transmission':0,
    'Owner':0,
    'Age':10
},index=[0])
```

In [50]:
```
xg.predict(data_input2)
```

Out[50]:
```
array([3.133159], dtype=float32)
```

In [55]:
```
import joblib
```

In [57]:
```
joblib.dump(xg,'car_price_predictor')
```

Out[57]:
```
['car_price_predictor']
```

In [58]:
```
xg = joblib.load('car_price_predictor')
```

In [62]:

```
from tkinter import *

def show_entry_fields():
    try:
        p1=float(e1.get())
        p2=float(e2.get())
        p3=float(e3.get())
        p4=float(e4.get())
        p5=float(e5.get())
        p6=float(e6.get())
        p7=float(e7.get())

        # Load the model
        model = joblib.load('car_price_predictor')

        # Create a DataFrame with the input data
        data_new = pd.DataFrame({
            'Present_Price': [p1],
            'Kms_Driven': [p2],
            'Fuel_Type': [p3],
            'Seller_Type': [p4],
            'Transmission': [p5],
            'Owner': [p6],
            'Age': [p7]
        })

        # Predict the result
        result = model.predict(data_new)

        # Display the result
        Label(master, text="Car Purchase amount").grid(row=8)
        Label(master, text=result[0]).grid(row=10)
        print("Car Purchase amount", result[0])
    except Exception as e:
        print("An error occurred:", e)

master = Tk()
master.title("Car Price Prediction Using Machine Learning")
label = Label(master, text="Car Price Prediction Using Machine Learning", bg="b
lack", fg="white")
label.grid(row=0, columnspan=2)

Label(master, text="Present_Price").grid(row=1)
Label(master, text="Kms_Driven").grid(row=2)
Label(master, text="Fuel_Type").grid(row=3)
Label(master, text="Seller_Type").grid(row=4)
Label(master, text="Transmission").grid(row=5)
Label(master, text="Owner").grid(row=6)
Label(master, text="Age").grid(row=7)

e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)

e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
```

```
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
Button(master, text='Predict', command=show_entry_fields).grid()
mainloop()
```

In [ ]:

In [ ]:

In [ ]: