# Importing required libraries

In [1]:
```
import warnings
import pandas as pd
```

In [2]:
```
warnings.filterwarnings('ignore')
```

# Load data

In [3]:
```
data = pd.read_csv('car data.csv')
```

# Preprocess Data

In [4]:
```
#Display Top 5 Rows of The Dataset
data.head()
```

Out[4]:

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

In [5]:
```
#Check Last 5 Rows of The Dataset
data.tail()
```

Out[5]:

| | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission |
|---|---|---|---|---|---|---|---|---|
| **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 [6]:
```
#Find Shape of Our Dataset (Number of Rows And Number of Columns)
data.shape
print("Number of Rows",data.shape[0])
print("Number of Columns",data.shape[1])
```

```
Number of Rows 301
Number of Columns 9
```

In [7]:
```
#Get Information About Our Dataset Like the Total Number of Rows, Total Number of Colu
data.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]:
```
#Check Null Values In The Dataset
data.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]:
```
#Get Overall Statistics About The Dataset
data.describe()
```

Out[9]:

| | 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 [10]:
```
#Data Preprocessing
data.head(1)
```

Out[10]:

| | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission | O |
|---|---|---|---|---|---|---|---|---|---|
| **0** | ritz | 2014 | 3.35 | 5.59 | 27000 | Petrol | Dealer | Manual | |

In [11]:
```
import datetime
date_time = datetime.datetime.now()
data['Age']=date_time.year - data['Year']
data.head()
```

Out[11]:

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

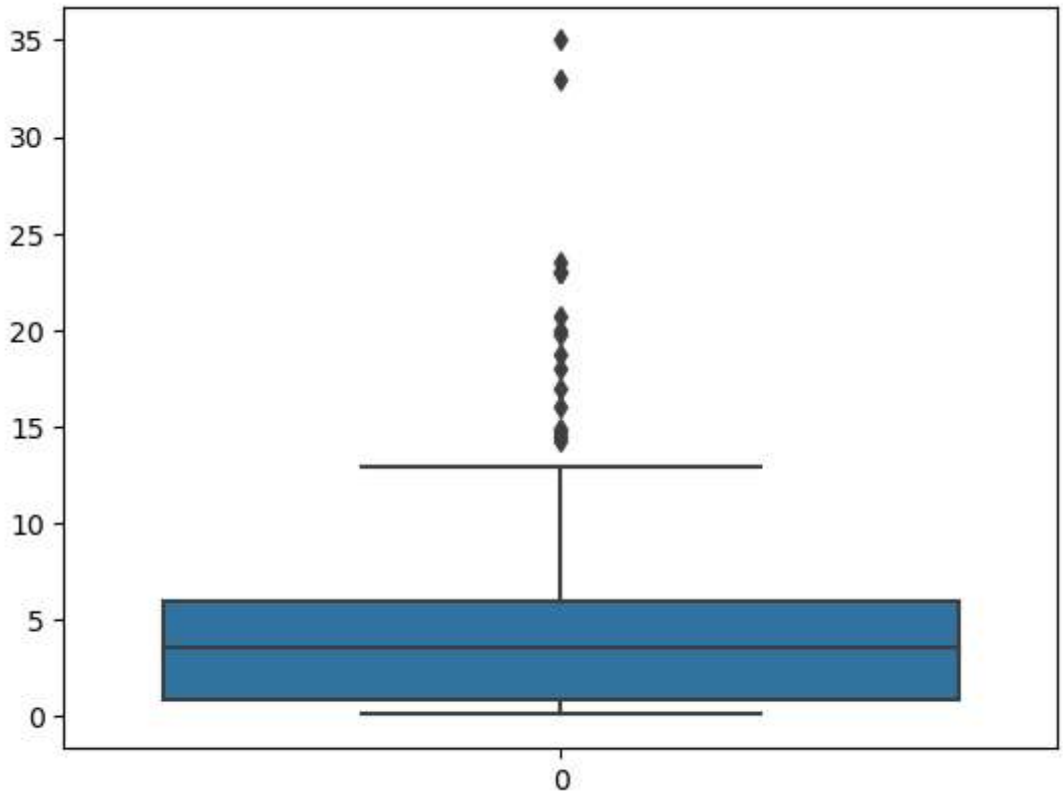
In [12]:
```
data.drop('Year',axis=1,inplace=True)
data.head()
```

Out[12]:

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

In [13]:
```
#Outlier Removal
import seaborn as sns
sns.boxplot(data['Selling_Price'])
```

Out[13]:
```
<AxesSubplot: >
```




In [14]:
```
sorted(data['Selling_Price'],reverse=True)
```

Out[14]:
```
[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,
4.5,
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,
0.2,
0.2,
0.2,
0.2,
0.2,
0.2,
0.18,
0.17,
0.16,
0.15,
```

```
0.12,
0.1]
```

In [15]:
```
data = data[~(data['Selling_Price']>=33.0) & (data['Selling_Price']<=35.0)]
data.shape
```

Out[15]:
```
(299, 9)
```

In [16]:
```
#Encoding the Categorical Columns
data.head(1)
```

Out[16]:

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

In [17]:
```
data['Fuel_Type'].unique()
```

Out[17]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [18]:
```
data['Fuel_Type'] = data['Fuel_Type'].map({'Petrol':0,'Diesel':1,'CNG':2})
data['Fuel_Type'].unique()
```

Out[18]:
```
array([0, 1, 2], dtype=int64)
```

In [19]:
```
data['Seller_Type'].unique()
```

Out[19]:
```
array(['Dealer', 'Individual'], dtype=object)
```

In [20]:
```
data['Seller_Type'] = data['Seller_Type'].map({'Dealer':0,'Individual':1})
```

In [21]:
```
data['Seller_Type'].unique()
```

Out[21]:
```
array([0, 1], dtype=int64)
```

In [22]:
```
data['Transmission'].unique()
```

Out[22]:
```
array(['Manual', 'Automatic'], dtype=object)
```

In [23]:
```
data['Transmission'] =data['Transmission'].map({'Manual':0,'Automatic':1})
data['Transmission'].unique()
```

Out[23]:
```
array([0, 1], dtype=int64)
```

In [24]:
```
data.head()
```

Out[24]:

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

# Splitting data

In [25]:
```
#Store Feature Matrix In X and Response(Target) In Vector y
X = data.drop(['Car_Name','Selling_Price'],axis=1)
y = data['Selling_Price']
y
```

Out[25]:
```
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 [26]:
```
#Splitting The Dataset Into The Training Set And Test Set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)
```

# Model Training

In [27]:
```
#Import the models
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

In [28]:
```
#Model Training
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[28]:

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=No
ne,
             enable_categorical=False, eval_metric=None, feature_types=No
ne,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=No
ne,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
```

In [29]:

```
#Prediction on Test Data
y_pred1 = lr.predict(X_test)
y_pred2 = rf.predict(X_test)
y_pred3 = xgb.predict(X_test)
y_pred4 = xg.predict(X_test)
```

In [30]:

```
#Evaluating the Algorithm
from sklearn import metrics
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)
print(score1,score2,score3,score4)
```

```
0.6790884983129406 0.7274609041329735 0.8835823354571616 0.8887471822279068
```

In [31]:

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

Out[31]:

| | Models | R2_SCORE |
|---|---|---|
| **0** | LR | 0.679088 |
| **1** | RF | 0.727461 |
| **2** | GBR | 0.883582 |
| **3** | XG | 0.888747 |

In [32]:

```
#Save The Model
xg = XGBRegressor()
xg_final = xg.fit(X,y)
import joblib
joblib.dump(xg_final,'car_price_predictor')
model = joblib.load('car_price_predictor')
```

In [33]:

```
xg_final.save_model('car_price_predictor.json')
```

In [34]:
```
#Prediction on New Data
import pandas as pd
data_new = pd.DataFrame({
    'Present_Price':5.59,
    'Kms_Driven':27000,
    'Fuel_Type':0,
    'Seller_Type':0,
    'Transmission':0,
    'Owner':0,
    'Age':8
},index=[0])
model.predict(data_new)
```

Out[34]:
```
array([3.45956], dtype=float32)
```

In [ ]:
```

```