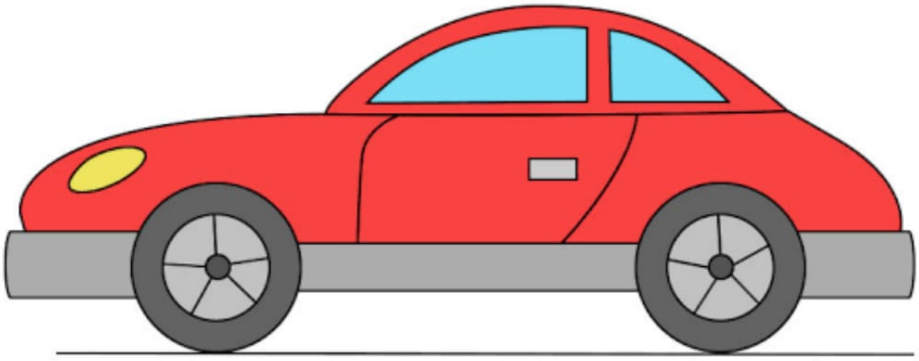


In [1]:
```
import pandas as pd
import datetime

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

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

from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from sklearn import metrics

import pickle
import joblib
```

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

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

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

In [4]:
```
dataset.head()
```

Out[4]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Ow
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]:
```
dataset.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]:
```
print('Number of Rows:',dataset.shape[0])
print('Number of Columns:',dataset.shape[1])
```

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

In [7]:
```
dataset.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]:
```
dataset.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

In [9]:
```
dataset.corr()
```

Out[9]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
Year	1.000000	0.236141	-0.047584	-0.524342	-0.182104
Selling_Price	0.236141	1.000000	0.878983	0.029187	-0.088344
Present_Price	-0.047584	0.878983	1.000000	0.203647	0.008057
Kms_Driven	-0.524342	0.029187	0.203647	1.000000	0.089216
Owner	-0.182104	-0.088344	0.008057	0.089216	1.000000

In [10]:
```
dataset.isnull().sum()
```

Out[10]:
```
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 [11]:
```
dataset.isna().sum()
```

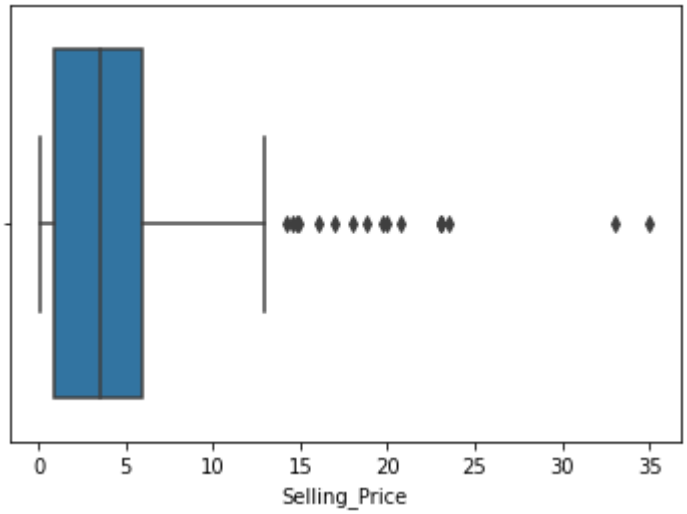
Out[11]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [12]:
```
date_time = datetime.datetime.now()
```

In [13]:
```
dataset['Age'] = date_time.year - dataset['Year']
```

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

In [15]:
```
sns.boxplot(dataset['Selling_Price'])
plt.show()
```



In [16]:
```
sorted(dataset['Selling_Price'],reverse=True)
```

Out[16]:
```
[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,
0.42,
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 [17]:
```
dataset = dataset[~(dataset['Selling_Price']>= 33.0) & (dataset['Selling_Price
```

In [18]:
```
dataset.shape
```

Out[18]:
```
(299, 9)
```

In [19]:
```
dataset['Fuel_Type'].unique()
```

Out[19]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [20]:
```
dataset['Fuel_Type'] = dataset['Fuel_Type'].map({'Petrol':0,'Diesel':1,'CNG':2
```

In [21]:
```
dataset['Fuel_Type'].unique()
```

Out[21]:
```
array([0, 1, 2], dtype=int64)
```

In [22]:
```
dataset['Seller_Type'].unique()
```

Out[22]:
```
array(['Dealer', 'Individual'], dtype=object)
```

In [23]:
```
dataset['Seller_Type'] = dataset['Seller_Type'].map({'Dealer':0,'Individual':1
```

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

Out[24]:
```
array([0, 1], dtype=int64)
```

In [25]:
```
dataset['Transmission'].unique()
```

Out[25]:
```
array(['Manual', 'Automatic'], dtype=object)
```

In [26]:
```
dataset['Transmission'] = dataset['Transmission'].map({'Manual':0,'Automatic':
```

In [27]:
```
dataset['Transmission'].unique()
```

Out[27]:
```
array([0, 1], dtype=int64)
```

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

Out[28]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	A
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	

In [29]:
```
X = dataset.drop(['Car_Name','Selling_Price'],axis=1)
y = dataset['Selling_Price']
```

In [30]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_sta
```

In [31]:
```
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[31]:

XGBRegressor

```
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='depthw
ise',
             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_we
ight=1,
```

In [32]:
```
y_pred_1 = lr.predict(X_test)
y_pred_2 = rf.predict(X_test)
y_pred_3 = xgb.predict(X_test)
y_pred_4 = xg.predict(X_test)
```

In [33]:
```
score_1 = metrics.r2_score(y_test,y_pred_1)
score_2 = metrics.r2_score(y_test,y_pred_2)
score_3 = metrics.r2_score(y_test,y_pred_3)
score_4 = metrics.r2_score(y_test,y_pred_4)
```

In [34]:
```
print(score_1,score_2,score_3,score_4)
```

```
0.67908849831294 0.7256280060484965 0.8667824604887008 0.8864839405756888
```

In [35]:
```
final_data = pd.DataFrame({'Models':['LR','RF','GBR','XG'],
              'R2_Score':[score_1,score_2,score_3,score_4]})
```

In [36]:
```
final_data
```

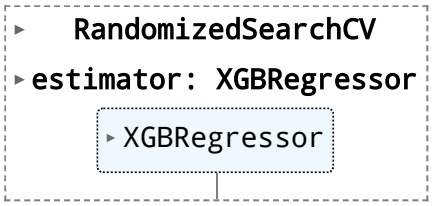
Out[36]:

	Models	R2_Score
0	LR	0.679088
1	RF	0.725628
2	GBR	0.866782
3	XG	0.886484

In [37]:
```
param_grid = {'gamma': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102.4, 20(
              'learning_rate': [0.01, 0.03, 0.06, 0.1, 0.15, 0.2, 0.25, 0.3000(
              'max_depth': [5,6,7,8,9,10,11,12,13,14],
              'n_estimators': [50,65,80,100,115,130,150],
              'reg_alpha': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102.4
              'reg_lambda': [0,0.1,0.2,0.4,0.8,1.6,3.2,6.4,12.8,25.6,51.2,102.4

xg_1 = XGBRegressor(random_state=15,verbosity=0,silent=0)
rcv = RandomizedSearchCV(estimator=xg_1,param_distributions=param_grid, n_iter=
                                verbose=1, random_state=15, n_jobs=-1)

rcv.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

Out[37]:



In [38]:
```
y_pred_5 = rcv.predict(X_test)
```

In [39]:
```
score_5 = metrics.r2_score(y_test,y_pred_5)
```

In [40]:
```
print(score_5)
```

```
0.913487881151229
```

In [72]:
```
xg = XGBRegressor()
xg_final = xg.fit(X,y)
```

In [42]:
```
xg = XGBRegressor()
xg_1.fit(X_train,y_train)
```

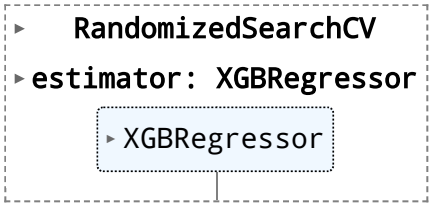
Out[42]:

```
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='depthw
ise',
             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_we
ight=1,
```

In [43]:
```
xg_1 = XGBRegressor()
rcv.fit(X_train,y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

Out[43]:



In [44]:
```
pickle.dump(rcv,open('car_price_predictor.pkl','wb'))
```

In [45]:
```
model = pickle.load(open('car_price_predictor.pkl','rb'))
```

In [46]:
```
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])
```

In [47]:
```
model.predict(data_new)
```

Out[47]:
```
array([3.529566], dtype=float32)
```

In [73]:
```
joblib.dump(xg_final,'car_price_predictor.job')
```

Out[73]:
```
['car_price_predictor.job']
```

In [74]:
```
model_1 = joblib.load('car_price_predictor.job')
```

In [75]:
```
data_new_1 = pd.DataFrame({'Present_Price':5.59,
                         'Kms_Driven':27000,
                         'Fuel_Type':0,
                         'Seller_Type':0,
                         'Transmission':0,
                         'Owner':0,
                         'Age':8
                          },index=[0])
```

In [76]:
```
model_1.predict(data_new_1)
```

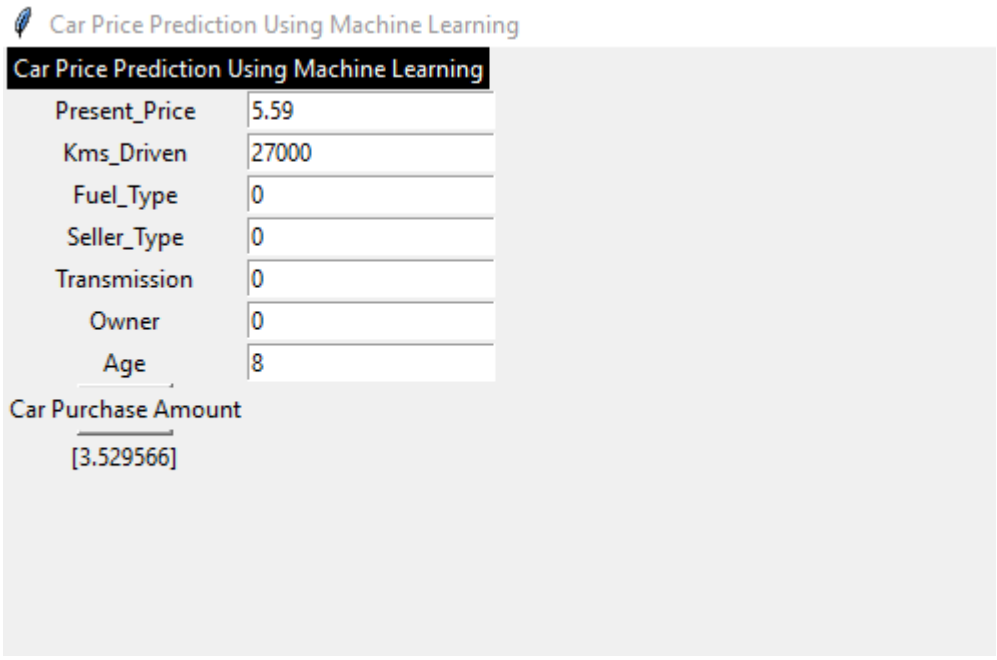
Out[76]:
```
array([3.7360353], dtype=float32)
```

In [87]:
```
xg_final.save_model('xgb_model.json')
```

In [89]:
```
from tkinter import *

def show_entry_fields():
    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())
```

```
    model = joblib.load('car_price_predictor.pkl')
    data_new = pd.DataFrame({
    'Present_Price':p1,
    'Kms_Driven':p2,
    'Fuel_Type':p3,
    'Seller_Type':p4,
    'Transmission':p5,
    'Owner':p6,
    'Age':p7
},index=[0])
    result = model.predict(data_new)
    Label(master, text='Car Purchase Amount').grid(row=8)
    Label(master, text=result).grid(row=10)
    print('Car Purchase Amount', result[0])

master = Tk()
master.title('Car Price Prediction Using Machine Learning')
label = Label(master, text = 'Car Price Prediction Using Machine Learning'
                          , bg = 'black', fg = 'white'). \
                               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()
```

```
Car Purchase Amount 3.529566
```



CONNECT WITH ME :

LinkedIn GitHub kaggle Medium

PRASADMJADHAV2