

```
In [1]: import warnings
```

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

```
In [3]: import pandas as pd
```

```
In [4]: data = pd.read_csv('car data.xls')
```

1. Display Top 5 Rows of The Dataset

```
In [5]: data.head()
```

```
Out[5]:
```

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

2. Check Last 5 Rows of The Dataset

```
In [6]: data.tail()
```

```
Out[6]:
```

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

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

```
In [7]: data.shape
```

```
Out[7]: (301, 9)
```

```
In [8]: print("Number of Rows",data.shape[0])  
        print("Number of Columns",data.shape[1])
```

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

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

```
In [9]: 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
```

5. Check Null Values In The Dataset

```
In [10]: data.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
```

6. Get Overall Statistics About The Dataset

```
In [11]: data.describe()
```

```
Out[11]:
```

| | 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 |

7. Data Preprocessing

```
In [12]: data.head(1)
```

```
Out[12]:
```

| | Car_Name | Year | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmis |
|---|----------|------|---------------|---------------|------------|-----------|-------------|----------|
| 0 | ritz | 2014 | 3.35 | 5.59 | 27000 | Petrol | Dealer | Ma |

```
In [13]: import datetime
```

```
In [14]: date_time = datetime.datetime.now()
```

```
In [15]: data['Age']=date_time.year - data['Year']
```

```
In [16]: data.head()
```

```
Out[16]:
```

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

```
In [17]: data.drop('Year',axis=1,inplace=True)
```

```
In [18]: data.head()
```

```
Out[18]:
```

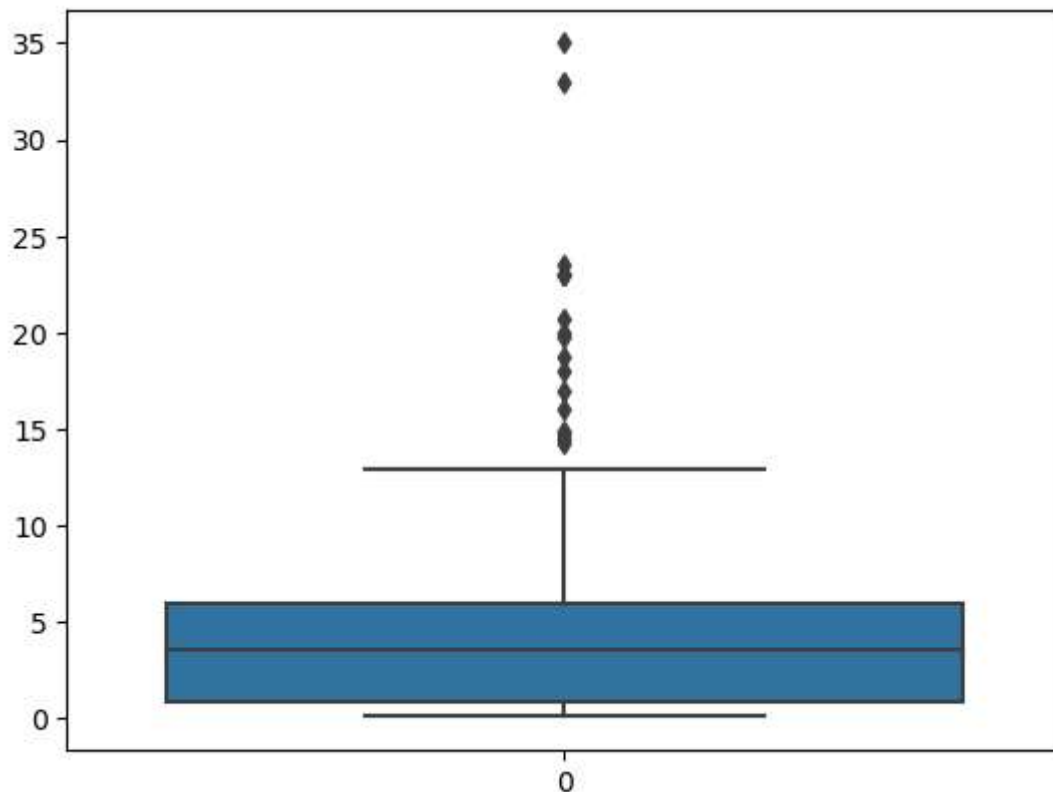
| | Car_Name | Selling_Price | Present_Price | Kms_Driven | Fuel_Type | Seller_Type | Transmission |
|---|----------|---------------|---------------|------------|-----------|-------------|--------------|
| 0 | ritz | 3.35 | 5.59 | 27000 | Petrol | Dealer | Manual |
| 1 | sx4 | 4.75 | 9.54 | 43000 | Diesel | 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 [19]: import seaborn as sns
```

```
In [20]: sns.boxplot(data['Selling_Price'])
```

```
Out[20]: <Axes: >
```



```
In [21]: sorted(data['Selling_Price'],reverse=True)
```

```
Out[21]: [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]
```

```
In [22]: data = data[~(data['Selling_Price']>=33.0) & (data['Selling_Price']<=35.0)]
```

```
In [23]: data.shape
```

```
Out[23]: (299, 9)
```

Encoding the Categorical Columns

```
In [24]: data.head(1)
```

```
Out[24]:
```

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

```
In [25]: data['Fuel_Type'].unique()
```

```
Out[25]: array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

```
In [26]: data['Fuel_Type'] = data['Fuel_Type'].map({'Petrol':0, 'Diesel':1, 'CNG':2})
```

```
In [27]: data['Fuel_Type'].unique()
```

```
Out[27]: array([0, 1, 2], dtype=int64)
```

```
In [28]: data['Seller_Type'].unique()
```

```
Out[28]: array(['Dealer', 'Individual'], dtype=object)
```

```
In [29]: data['Seller_Type'] = data['Seller_Type'].map({'Dealer':0,'Individual':1})
```

```
In [30]: data['Seller_Type'].unique()
```

```
Out[30]: array([0, 1], dtype=int64)
```

```
In [31]: data['Transmission'].unique()
```

```
Out[31]: array(['Manual', 'Automatic'], dtype=object)
```

```
In [32]: data['Transmission'] = data['Transmission'].map({'Manual':0,'Automatic':1})
```

```
In [33]: data['Transmission'].unique()
```

```
Out[33]: array([0, 1], dtype=int64)
```

```
In [34]: data.head()
```

```
Out[34]:
```

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

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

```
In [35]: X = data.drop(['Car_Name', 'Selling_Price'], axis=1)
y = data['Selling_Price']
```

```
In [36]: y
```

```
Out[36]: 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
```

9. Splitting The Dataset Into The Training Set And Test Set

```
In [37]: from sklearn.model_selection import train_test_split
```

```
In [38]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state
```

10. Import The models

```
In [39]: ! pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in c:\users\r\appdata\roaming\python\python310\site-packages (1.7.6)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.10.0)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.23.5)
```

```
In [40]: from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

11. Model Training

```
In [41]: 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[41]: 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, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

12. Prediction on Test Data

```
In [42]: y_pred1 = lr.predict(X_test)
y_pred2 = rf.predict(X_test)
y_pred3 = xgb.predict(X_test)
y_pred4 = xg.predict(X_test)
```

13. Evaluating the Algorithm

```
In [43]: from sklearn import metrics
```

```
In [44]: 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 [45]: print(score1,score2,score3,score4)
```

0.6790884983129396 0.7508612295607724 0.8943559626238841 0.8864839405756888


```
In [46]: final_data = pd.DataFrame({'Models': ['LR', 'RF', 'GBR', 'XG'],  
                                   "R2_SCORE": [score1, score2, score3, score4]})
```

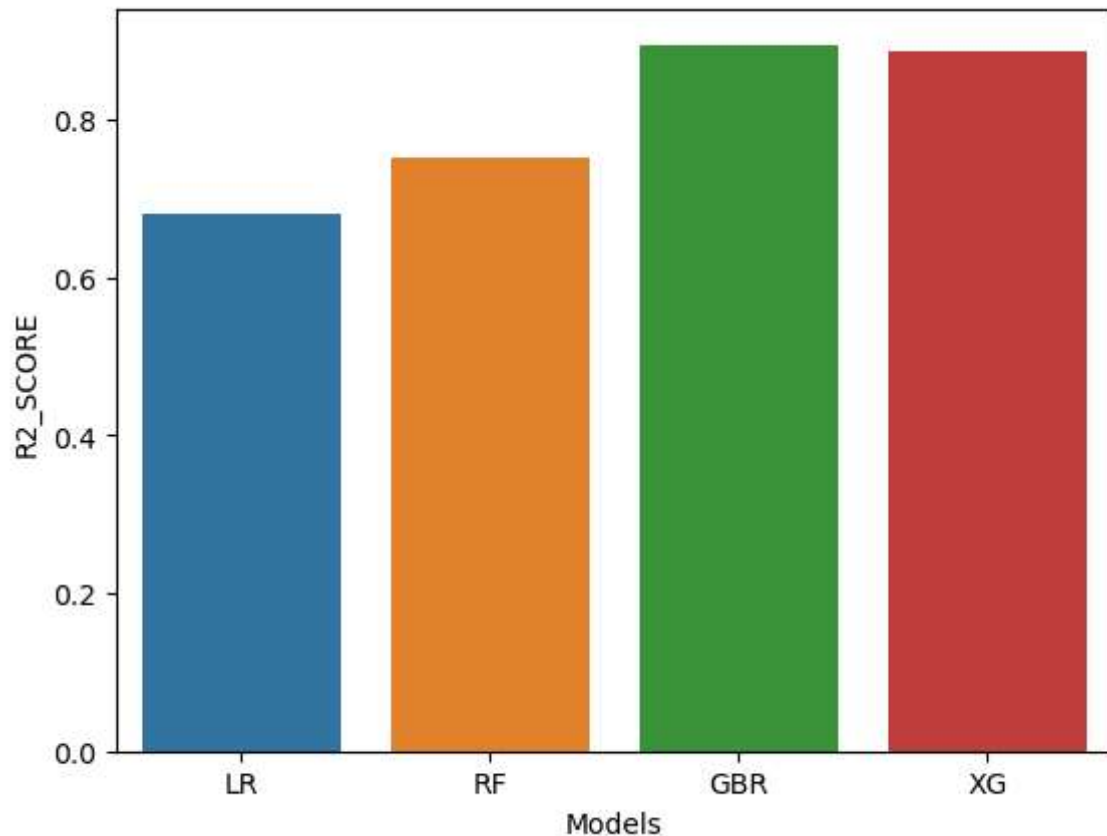
```
In [47]: final_data
```

```
Out[47]:
```

| | Models | R2_SCORE |
|---|--------|----------|
| 0 | LR | 0.679088 |
| 1 | RF | 0.750861 |
| 2 | GBR | 0.894356 |
| 3 | XG | 0.886484 |

```
In [48]: sns.barplot(x=final_data['Models'], y=final_data['R2_SCORE'])
```

```
Out[48]: <Axes: xlabel='Models', ylabel='R2_SCORE'>
```



14. Save The Model

```
In [49]: xg = XGBRegressor()  
xg_final = xg.fit(X,y)
```

```
In [50]: import joblib
```

```
In [51]: joblib.dump(xg_final, 'car_price_predictor')
```

```
Out[51]: ['car_price_predictor']
```

```
In [52]: model = joblib.load('car_price_predictor')
```

15. Prediction on New Data

```
In [53]: 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])
```

```
In [54]: model.predict(data_new)
```

```
Out[54]: array([3.7360353], dtype=float32)
```



```

In [55]: from tkinter import *
import joblib

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')
    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()
```

In []: