CAR PRICE PREDICTION

In [ ]:
```
#car price prediction using ml
```

importing libraries

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

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

In [3]:
```
dataset=pd.read_csv(r"E:\resume projects\car data.xls")
```

In [4]:
```
dataset
```

Out[4]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	N
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	N
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	N
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	N
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	N
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	Diesel	Dealer	N
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	N
298	city	2009	3.35	11.00	87934	Petrol	Dealer	N
299	city	2017	11.50	12.50	9000	Diesel	Dealer	N
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	N

301 rows × 9 columns

print columns name

In [5]:
```
dataset.columns
```

Out[5]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

print top 5 rows

In [6]:
```
dataset.head()
```

Out[6]:

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

display last 5 rows of the dataset

In [7]:
```
dataset.tail()
```

Out[7]:

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

find shape of our dataset(number of rows and number of columns)

In [8]:
```
dataset.shape
```

Out[8]:
```
(301, 9)
```

In [9]:
```
print("Number of rows" ,dataset.shape[0])
print("Number of columns" , dataset.shape[1])
```

```
Number of rows 301
Number of columns 9
```

get information about our dataset like the total number of rows ,total number of columns ,datatypes of each columns and memory requirement

In [10]:
```
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
```

check null values in the dataset

In [11]:
```
dataset.isnull()
```

Out[11]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
296	False	False	False	False	False	False	False	
297	False	False	False	False	False	False	False	
298	False	False	False	False	False	False	False	
299	False	False	False	False	False	False	False	
300	False	False	False	False	False	False	False	

301 rows × 9 columns

In [12]:
```
dataset.isnull().sum()
```

Out[12]:
```
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 [13]:

```
dataset.describe()
```

Out[13]:

	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 [14]:

```
dataset.head(1)
```

Out[14]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmiss
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Mai

In [15]:

```
import datetime
date_time=datetime.datetime.now()

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

In [16]:

```
dataset.head()
```

Out[16]:

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

In [17]:

```
dataset.drop('Year',axis=1,inplace=True)
```

In [18]:

```
dataset
```

Out[18]:

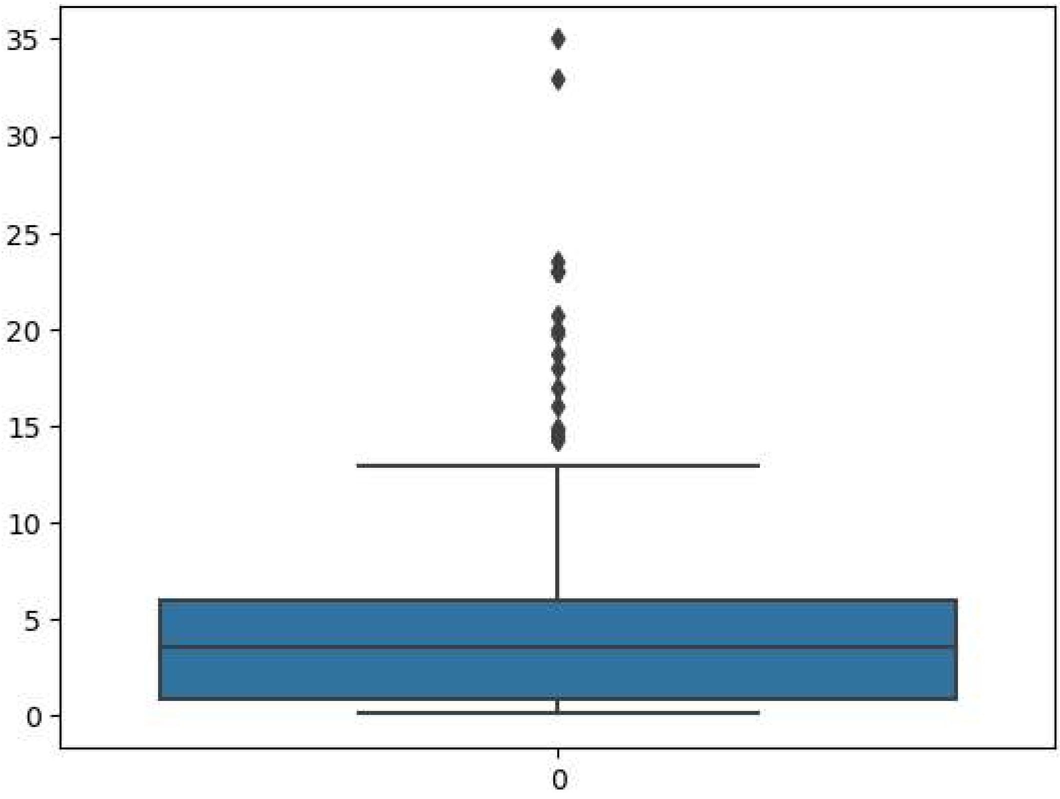
	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission
0	ritz	3.35	5.59	27000	Petrol	Dealer	Manual
1	sx4	4.75	9.54	43000	Diesel	Dealer	Manual
2	ciaz	7.25	9.85	6900	Petrol	Dealer	Manual
3	wagon r	2.85	4.15	5200	Petrol	Dealer	Manual
4	swift	4.60	6.87	42450	Diesel	Dealer	Manual
...	...	...	...	...	...	...	...
296	city	9.50	11.60	33988	Diesel	Dealer	Manual
297	brio	4.00	5.90	60000	Petrol	Dealer	Manual
298	city	3.35	11.00	87934	Petrol	Dealer	Manual
299	city	11.50	12.50	9000	Diesel	Dealer	Manual
300	brio	5.30	5.90	5464	Petrol	Dealer	Manual

301 rows × 9 columns

outlier removal

```
In [19]: sns.boxplot(dataset['Selling_Price'])
```

```
Out[19]: <Axes: >
```



two datapoints are very far away from other datapoints so it will be consider as outlier

In [20]:
```
sorted(dataset['Selling_Price'] ,reverse=True)
```

Out[20]:
```
[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 [21]:
```
(dataset['Selling_Price']>=33.0) & (dataset['Selling_Price']<=35.0)
```

Out[21]:
```
0      False
1      False
2      False
3      False
4      False
       ...
296    False
297    False
298    False
299    False
300    False
Name: Selling_Price, Length: 301, dtype: bool
```

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

Out[22]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission
64	fortuner	33.0	36.23	6000	Diesel	Dealer	Automatic
86	land cruiser	35.0	92.60	78000	Diesel	Dealer	Manual

this two are outlier

In [23]:

```
dataset[~(dataset['Selling_Price'] >=33.0)  & (dataset['Selling_Price']<=35.0)
```

Out[23]:

	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
...	...	...	...	...	...	...	...
296	city	9.50	11.60	33988	Diesel	Dealer	Manual
297	brio	4.00	5.90	60000	Petrol	Dealer	Manual
298	city	3.35	11.00	87934	Petrol	Dealer	Manual
299	city	11.50	12.50	9000	Diesel	Dealer	Manual
300	brio	5.30	5.90	5464	Petrol	Dealer	Manual

299 rows × 9 columns

In [24]:

```
data=dataset[~(dataset['Selling_Price'] >=33.0)  & (dataset['Selling_Price']<=
```

In [25]:

```
data.shape
```

Out[25]:

```
(299, 9)
```

Encoding the categorical columns

In [26]:

```
dataset.head(1)
```

Out[26]:

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

In [27]:

```
dataset['Fuel_Type'].unique()
```

Out[27]:

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

In [28]:

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

In [29]:

```
dataset['Fuel_Type'].unique()
```

Out[29]:

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

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

Out[30]: array(['Dealer', 'Individual'], dtype=object)
```

```
In [31]: dataset['Seller_Type']=dataset['Seller_Type'].map({'Dealer':0,'Individual':1})
```

```
In [32]: dataset['Seller_Type'].unique()

Out[32]: array([0, 1], dtype=int64)
```

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

Out[33]: array(['Manual', 'Automatic'], dtype=object)
```

```
In [34]: dataset['Transmission']=dataset['Transmission'].map({'Manual':0,'Automatic':1}
```

```
In [35]: dataset['Transmission'].unique()

Out[35]: array([0, 1], dtype=int64)
```

```
In [36]: dataset.head()
```

Out[36]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	C
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	

```
In [37]: dataset.tail()
```

Out[37]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission
296	city	9.50	11.6	33988	1	0	0
297	brio	4.00	5.9	60000	0	0	0
298	city	3.35	11.0	87934	0	0	0
299	city	11.50	12.5	9000	1	0	0
300	brio	5.30	5.9	5464	0	0	0

In [38]:

```
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   Selling_Price  301 non-null    float64
 2   Present_Price  301 non-null    float64
 3   Kms_Driven     301 non-null    int64  
 4   Fuel_Type      301 non-null    int64  
 5   Seller_Type    301 non-null    int64  
 6   Transmission   301 non-null    int64  
 7   Owner          301 non-null    int64  
 8   Age            301 non-null    int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 21.3+ KB
```

store feature matrix in X and response (target) variable in Y

In [39]:

```
x=dataset.drop(['Car_Name' , 'Selling_Price'], axis=1)
y=dataset['Selling_Price']
```

In [40]:

```
# x is our independent variable
```

In [41]:

```
x
```

Out[41]:

	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Age
0	5.59	27000	0	0	0	0	10
1	9.54	43000	1	0	0	0	11
2	9.85	6900	0	0	0	0	7
3	4.15	5200	0	0	0	0	13
4	6.87	42450	1	0	0	0	10
...	...	...	...	...	...	...	...
296	11.60	33988	1	0	0	0	8
297	5.90	60000	0	0	0	0	9
298	11.00	87934	0	0	0	0	15
299	12.50	9000	1	0	0	0	7
300	5.90	5464	0	0	0	0	8

301 rows × 7 columns

In [42]:
```
# y is our target variable
```

In [43]:
```
y
```

Out[43]:
```
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: 301, dtype: float64
```

splitting the datset into tarining set and testing set

In [44]:
```
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
```

import the models

In [45]:
```
dataset.head()
```

Out[45]:

	Car_Name	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	C
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	

In [46]:
```
pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: xgboost in c:\users\achal raghorte\appdata\roa
ming\python\python311\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pac
kages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-pac
kages (from xgboost) (1.11.1)
Note: you may need to restart the kernel to use updated packages.
```

In [47]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

model training

In [48]:
```
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[48]:
```
▾                                   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,
```

prediction of the test data

In [49]:
```
y_pred1=lr.predict(x_test)
y_pred2=rf.predict(x_test)
y_pred3=xgb.predict(x_test)
y_pred4=xg.predict(x_test)
```

evaluating the algorithm

In [50]:
```
from sklearn import metrics
```

In [51]:

```
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 [52]:

```
print(score1,score2,score3,score4)
```

```
0.8468053957657442 0.9608961457488217 0.9722866094451921 0.9550781240593306
```

In [53]:

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

In [54]:

```
final_data
```

Out[54]:

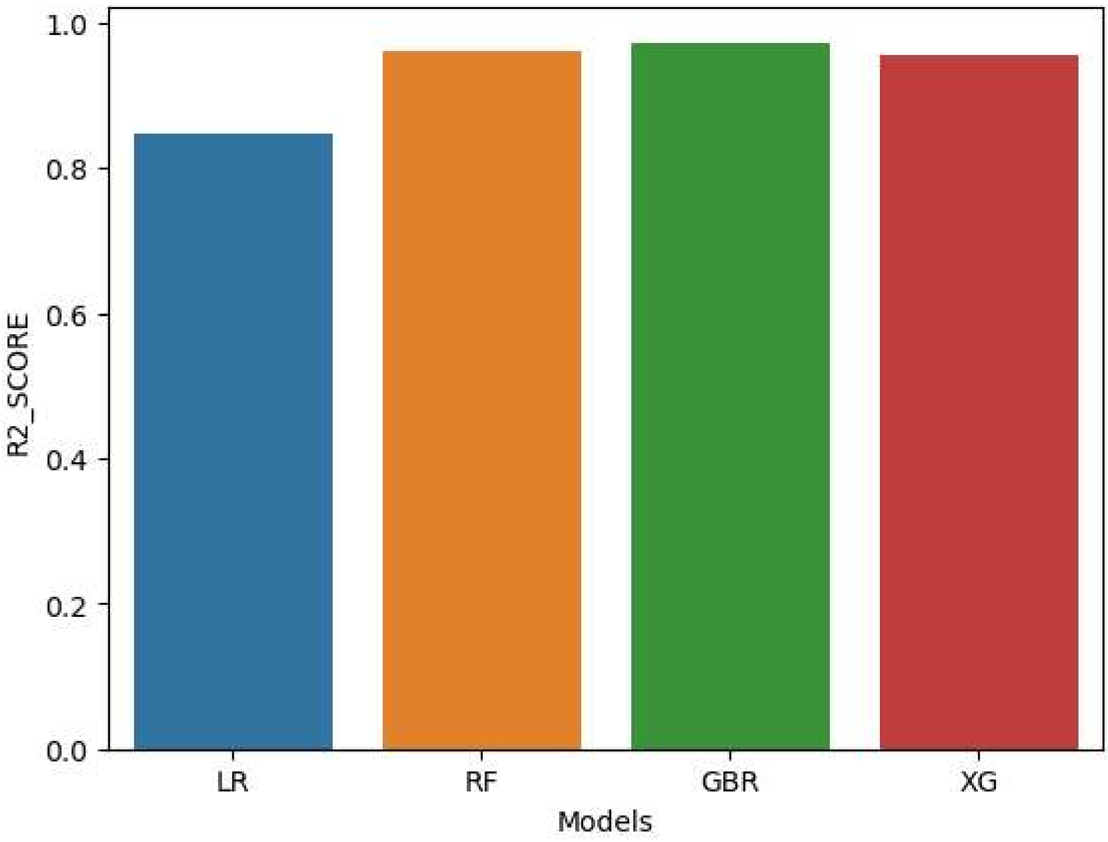
	Models	R2_SCORE
0	LR	0.846805
1	RF	0.960896
2	GBR	0.972287
3	XG	0.955078

In [55]:

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

Out[55]:

```
<Axes: xlabel='Models', ylabel='R2_SCORE'>
```



save the model

```
In [56]: xg=XGBRegressor()
         xg_final= xg.fit(x,y)
```

```
In [57]: import joblib
```

```
In [58]: joblib.dump(xg_final,'car_price_predictor')
```

```
Out[58]: ['car_price_predictor']
```

```
In [59]: model=joblib.load('car_price_predictor')
```

```
In [60]: model
```

```
Out[60]:
▾                         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,
```

prediction on new data

```
In [61]: 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 [62]: model.predict(data_new)
```

```
Out[62]: array([3.844488], dtype=float32)
```

GUI

In [*]:

```
from tkinter import *
import joblib
import pandas as pd

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 ml")
label = Label(master, text="car price prediction using ml", bg="black", fg="wh

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 3.352563
```

In [ ]:

```
```