```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

Double-click (or enter) to edit

```
#loading the data from csv file to a pandas dataframe
car_dataset = pd.read_csv('/content/car data.csv')
car_dataset
```

	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
...	...	...	...	...	...	...	...	...	...
296	city	2016	9.50	11.60	33988	Diesel	Dealer	Manual	0
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	Manual	0
298	city	2009	3.35	11.00	87934	Petrol	Dealer	Manual	0
299	city	2017	11.50	12.50	9000	Diesel	Dealer	Manual	0
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	Manual	0

301 rows × 9 columns

```
# First 5 rows of the dataset
car_dataset.head()
```

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

```
# FInding no.of rows and columns
car_dataset.shape
```

```
(301, 9)
```

```
# Information about the dataset
car_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
```

```
#Checking the missing values
car_dataset.isnull().sum()
```

```
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
```

```
# Statsitical measures of the dataset
car_dataset.describe()
```

	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

```
car_dataset['Fuel_Type'].value_counts()
```

```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
```

```
car_dataset['Transmission'].value_counts()
```

```
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

```
 car_dataset['Seller_Type'].value_counts()
```

```
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
```

Encoding the Categorical Data

```
 # Econding the Fuel_type Column
car_dataset.replace({'Fuel_Type':{'Petrol': 0, 'Diesel' : 1, 'CNG' : 2}}, inplace = True)

#Encoding the Transmission Column
car_dataset.replace({'Transmission':{'Manual':0, 'Automatic':1}}, inplace=True)

#Encoding the Seller Type
car_dataset.replace({'Seller_Type':{'Dealer':0, 'Individual':1}}, inplace=True)
```

Splitting the data into Training Data And Test Data

```
 X = car_dataset.drop(['Car_Name','Selling_Price'], axis = 1)
Y = car_dataset['Selling_Price']
```

```
X_train, X_test, Y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(301, 7) (240, 7) (61, 7)
```

Model Training

```
# Calling Linear Regression Model
model = LinearRegression()
```

```
model.fit(X_train, Y_train)
```

```
LinearRegression
LinearRegression()
```

```
training_data_prediction = model.predict(X_train)
```

```
training_data_score = metrics.r2_score(Y_train, training_data_prediction)
print('Training data Score:',training_data_score)
```

```
Training data Score: 0.8838169193709796
```

```
testing_data_prediction = model.predict(X_test)
```

```
testing_data_score = metrics.r2_score(y_test, testing_data_prediction)
print('Testing data Score:',testing_data_score)
```

```
Testing data Score: 0.8401532365377697
```

Building a Predictive System

```
input_data = (2014,5.59,27000,0,0,0,0)

input_data_as_numpy_array = np.asarray(input_data)

input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
```

```
prediction = model.predict(input_data_reshaped)
print("Car price according to the prediction is:",prediction)
```

```
Car price according to the prediction is: [3.82765933]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was
  warnings.warn(
```

```
prediction = model.predict(input_data_reshaped)
print("swift,2014,4.95,7.49,39000,Diesel,Dealer,Manual,o:",prediction)
```

```
swift,2014,4.95,7.49,39000,Diesel,Dealer,Manual,o: [3.82765933]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was
  warnings.warn(
```

```
prediction = model.predict(input_data_reshaped)
print("Bajaj Avenger 220 stsi,2010.0.45,0.95,27000,Petrol,Individual,Manual,0:",prediction)
```

```
Bajaj Avenger 220 stsi,2010.0.45,0.95,27000,Petrol,Individual,Manual,0: [3.82765933]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was
  warnings.warn(
```

```
prediction = model.predict(input_data_reshaped)
print(":",prediction)
```