```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

car_dataset =pd.read_csv("/content/cardataset.csv")
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owr
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	F Owı
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	F Owı
2	Hyundai Verna 1.6	2012	600000	100000	Diesel	Individual	Manual	F

```
# *1. EDA (Exploratory Data Analysis)*
car_dataset.shape
```

```
(4340, 8)
```

```
car_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Car_Name       4340 non-null   object
 1   Year           4340 non-null   int64 
 2   Selling_Price  4340 non-null   int64 
 3   Kms_Driven     4340 non-null   int64 
 4   Fuel_Type      4340 non-null   object
 5   Seller_Type    4340 non-null   object
 6   Transmission   4340 non-null   object
 7   Owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

```
categorical=car_dataset.select_dtypes(include=[object])
numerical=car_dataset.select_dtypes(include=[np.int32, np.int64, np.float64, np.float32])
print("Categorical features:",categorical.shape[1])
print("Numerical features:",numerical.shape[1])
```

```
Categorical features: 5
Numerical features: 3
```

```
car_dataset.describe()
```

	Year	Selling_Price	Kms_Driven
count	4340.000000	4.340000e+03	4340.000000
mean	2013.090783	5.041273e+05	66215.777419
std	4.215344	5.785487e+05	46644.102194
min	1992.000000	2.000000e+04	1.000000
25%	2011.000000	2.087498e+05	35000.000000
50%	2014.000000	3.500000e+05	60000.000000
75%	2016.000000	6.000000e+05	90000.000000
max	2020.000000	8.900000e+06	806599.000000

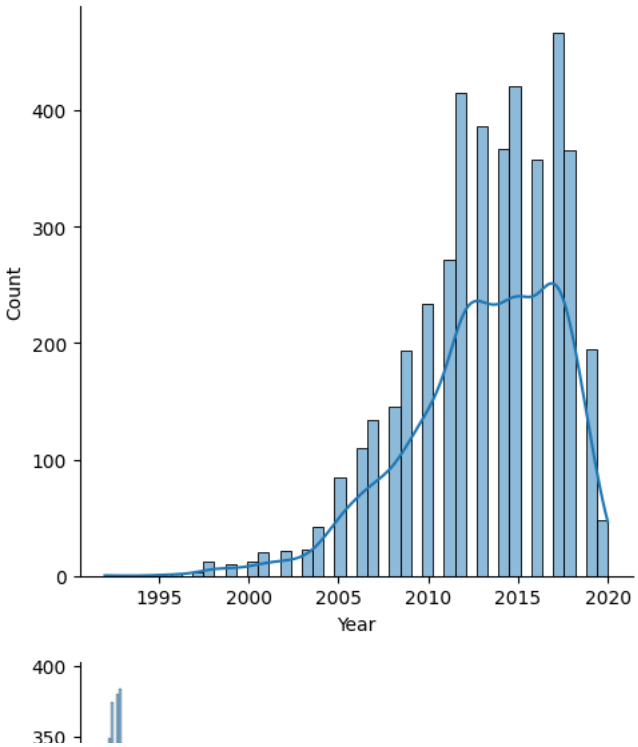
```
car_dataset.isnull().sum()
```

```
Car_Name         0
Year             0
Selling_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owr
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	F Owı
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	F Owı
2	Hyundai Verna 1.6	2012	600000	100000	Diesel	Individual	Manual	F

```
for i in numerical:
    sns.displot(x=car_dataset[i],kde=True)
plt.show()
```




```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owr
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	F Owı
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	F Owı
2	Hyundai Verna 1.6	2012	600000	100000	Diesel	Individual	Manual	F

```
seller=car_dataset['Seller_Type'].value_counts()
print(seller,'\n')
plt.bar(seller.index, seller.values,color=['green','orange'])
plt.title('Owner Type')
plt.show()
```

```
Individual          3244

Name: Seller_Type, dtype: int64
```

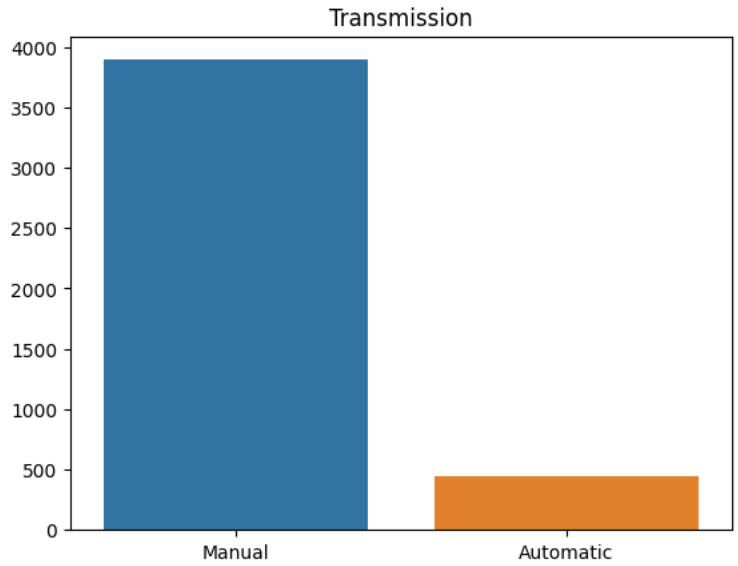
```
car_dataset.head()
```

1 to 5 of 5 entries Filter

index	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
1500	Datsun							

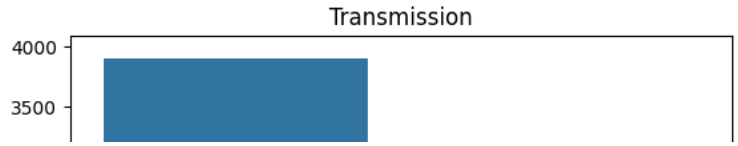
```
transmission=car_dataset['Transmission'].value_counts()
print(transmission,'\n')
sns.barplot(x=transmission.index, y=transmission.values)
plt.title('Transmission')
```

```
Manual       3892
Automatic     448
Name: Transmission, dtype: int64

Text(0.5, 1.0, 'Transmission')
```




```
transmission=car_dataset['Transmission'].value_counts()
print(transmission,'\n')
sns.barplot(x=transmission.index, y=transmission.values)
plt.title('Transmission')
```

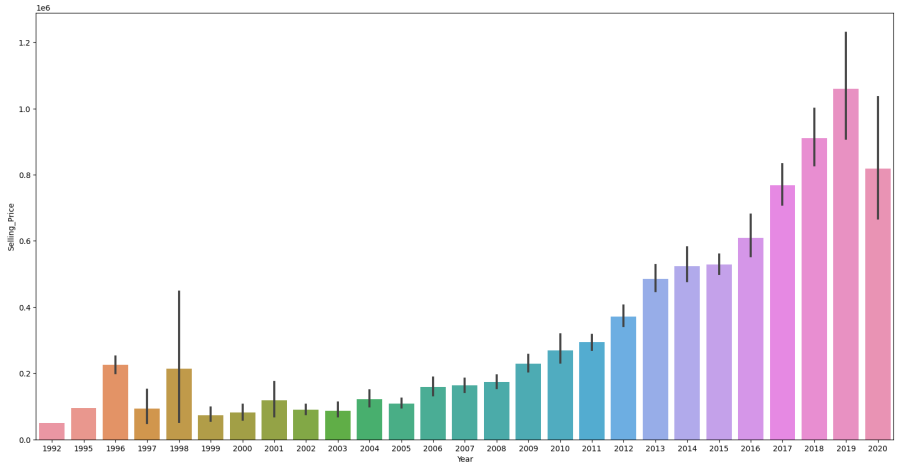
```
Manual       3892
Automatic     448
Name: Transmission, dtype: int64

Text(0.5, 1.0, 'Transmission')
```




```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owr
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	F Owı
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	F Owı
2	Hyundai Verna 1.6	2012	600000	100000	Diesel	Individual	Manual	F

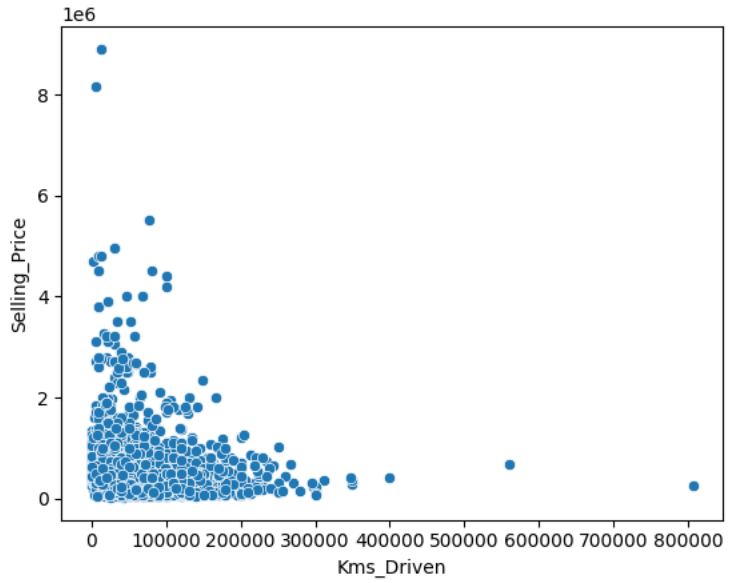
```
plt.figure(figsize=(20,10))
sns.barplot(y=car_dataset['Selling_Price'],x=car_dataset['Year'])
plt.show()
```




```
car_dataset.head()
```

Car Name	Year	Selling Price	Kms Driven	Fuel Type	Seller Type	Transmission	Owr

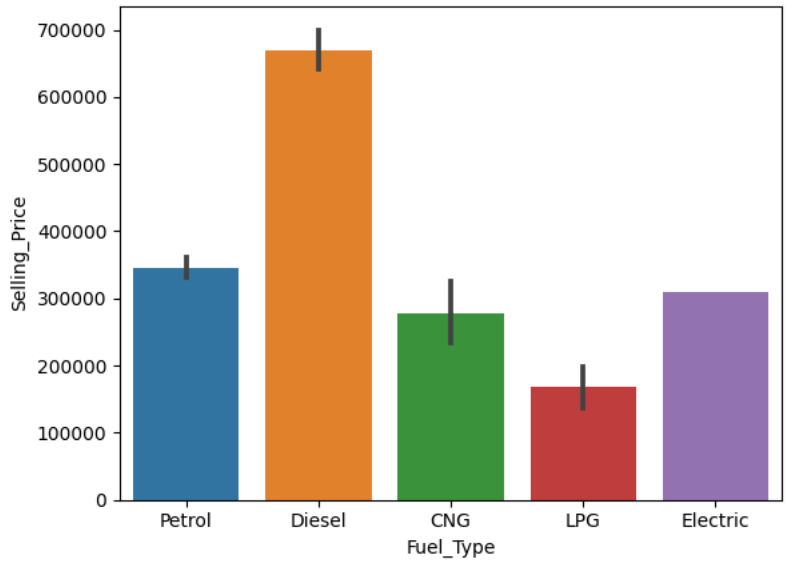
```
sns.scatterplot(y=car_dataset['Selling_Price'],x=car_dataset['Kms_Driven'])
plt.show()
```




```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owr
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	F Owı
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	F Owı
2	Hyundai Verna 1.6	2012	600000	100000	Diesel	Individual	Manual	F

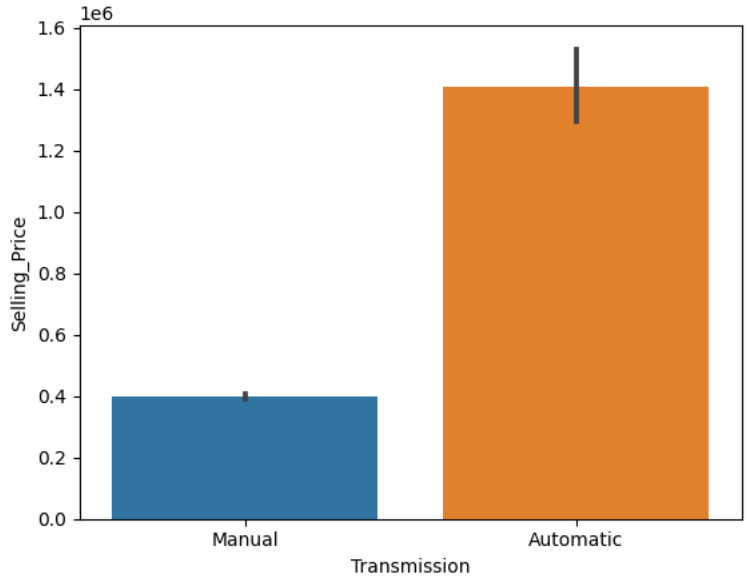
```
sns.barplot(y=car_dataset['Selling_Price'],x=car_dataset['Fuel_Type'])
plt.show()
```



```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

```
sns.barplot(y=car_dataset['Selling_Price'],x=car_dataset['Transmission'])
plt.show()
```




```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

```
sns.barplot(y=car_dataset['Selling_Price'],x=car_dataset['Owner'])
plt.show()
```


```
# *2. Preprocessing*
# (i) Encoding
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()

car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner

```
for i in ['Fuel_Type','Seller_Type','Transmission']:
    car_dataset[i]=encoder.fit_transform(car_dataset[i])
    print(car_dataset[i])
```

```
0       4
1       4
2       1
3       4
4       1
       ..
4335    1
4336    1
4337    4
4338    1
4339    4
Name: Fuel_Type, Length: 4340, dtype: int64
0       1
1       1
2       1
3       1
4       1
       ..
4335    1
4336    1
4337    1
4338    1
4339    1
Name: Seller_Type, Length: 4340, dtype: int64
0       1
1       1
2       1
3       1
4       1
       ..
4335    1
4336    1
4337    1
4338    1
4339    1
Name: Transmission, Length: 4340, dtype: int64
```

```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	2007	60000	70000	4	1	1	First Owner

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

```
1    2153
4    2123
0      40
3      23
2       1
Name: Fuel_Type, dtype: int64
```

```
car_dataset.head()
```

	Car_Name	Year	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	2007	60000	70000	4	1	1	First Owner
1	Maruti Wagon R LXI Minor	2007	135000	50000	4	1	1	First Owner
2	Hyundai Verna 1.6 SX	2012	600000	100000	1	1	1	First Owner
3	Datsun RediGO T Option	2017	250000	46000	4	1	1	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	1	1	1	Second Owner

```
# (ii) Outlier Detection and Handling
car_dataset.head()
```

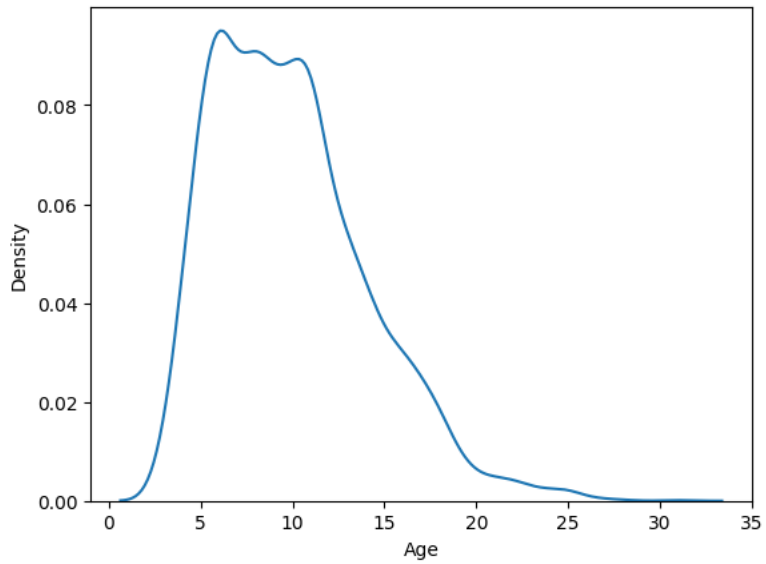
	Car_Name	Age	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	16	60000	70000	4	1	1	First Owner
1	Maruti Wagon R LXI Minor	16	135000	50000	4	1	1	First Owner
2	Hyundai Verna 1.6 SX	11	600000	100000	1	1	1	First Owner
3	Datsun RediGO T Option	6	250000	46000	4	1	1	First Owner
4	Honda Amaze VX i-DTEC	9	450000	141000	1	1	1	Second Owner

```
print(car_dataset.Age.value_counts(),'\n')
plt.boxplot(car_dataset.Age)
```

```
6     466
8     421
11    415
10    386
9     367
5     366
7     357
12    271
13    234
4     195
14    193
15    145
16    134
17    110
18     85
3      48
19     42
20     23
21     21
22     20
25     12
23     12
24     10
26      3
27      2
28      1
31      1
```

```
sns.kdeplot(car_dataset.Age)
```

```
<Axes: xlabel='Age', ylabel='Density'>
```




```
#Detecing Outliers using IQR Method (for skewed distribution)
age=car_dataset.Age
q1=age.quantile(0.25)
q3=age.quantile(0.75)
iqr=q3-q1
lower=q1-1.5*iqr
upper=q3+1.5*iqr

age[(age<lower)|(age>upper)]

print(lower)
print(upper)
```

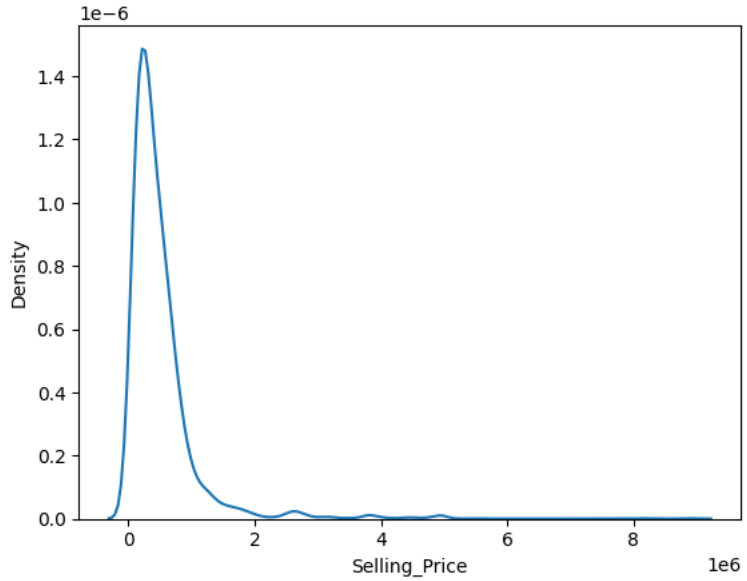
```
-0.5
19.5
```

```
age[(age<lower)|(age>upper)]

print(lower)
print(upper)
```

```
-0.5
19.5
```

```
#Handling Outliers for IQR Method
car_dataset.Age[car_dataset.Age<lower]=lower
car_dataset.Age[car_dataset.Age>upper]=upper

sns.kdeplot(car_dataset.Selling_Price)
```

```
<Axes: xlabel='Selling_Price', ylabel='Density'>
```




```
#Detecing Outliers using IQR Method (for skewed distribution)
sp=car_dataset.Selling_Price
q1=sp.quantile(0.25)
q3=sp.quantile(0.75)
iqr=q3-q1
lower=q1-1.5*iqr
upper=q3+1.5*iqr

sp[(sp<lower)|(sp>upper)]

print(lower)
print(upper)
```

```
-378125.625
1186875.375
```

```
# ### *3. Train and Test*

car_dataset.head()
```

	Car_Name	Age	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	16.0	60000.0	70000	4	1	1	First Owner
1	Maruti Wagon R LXI Minor	16.0	135000.0	50000	4	1	1	First Owner
2	Hyundai Verna 1.6 SX	11.0	600000.0	100000	1	1	1	First Owner
3	Datsun RediGO T Option	6.0	250000.0	46000	4	1	1	First Owner
4	Honda Amaze VX i-DTEC	9.0	450000.0	141000	1	1	1	Second Owner

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

X.head()
```

	Age	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	16.0	70000	4	1	1	First Owner
1	16.0	50000	4	1	1	First Owner

```
Y.head()
```

```
0     60000.0
1    135000.0
2    600000.0
3    250000.0
4    450000.0
Name: Selling_Price, dtype: float64
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1, random_state=2)
X_train.shape
```

```
(3906, 6)
```

```
X_train.head()
```

	Age	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
1987	5.0	52000	1	1	1	First Owner
676	12.0	140000	1	1	1	First Owner
110	4.0	15000	4	1	1	First Owner
1398	16.0	90000	4	1	1	Second Owner
122	10.0	60000	1	1	1	Second Owner

```
Y_train.shape
```

```
(3906,)
```

```
Y_train.head()
```

```
1987    800000.0
676     311000.0
110     750000.0
1398     75000.0
122     165000.0
Name: Selling_Price, dtype: float64
```

```
X_test.shape
```

```
(434, 6)
```

```
X_test.head()
```

	Age	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
1149	7.0	30000	4	0	1	First Owner
2245	6.0	10510	4	0	0	First Owner
4261	17.0	100000	1	1	1	Second Owner
2865	9.0	130000	1	1	0	First Owner
3110	5.0	60000	1	1	1	First Owner

```
Y_test.shape
```

```
(434,)
```

```
Y_test.shape
```

```
(434,)
```

```
car_dataset.head()
```

	Car_Name	Age	Selling_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	Maruti 800 AC	16.0	60000.0	70000	4	1	1	First Owner
1	Maruti Wagon R LXI Minor	16.0	135000.0	50000	4	1	1	First Owner
2	Hyundai Verna 1.6 SX	11.0	600000.0	100000	1	1	1	First Owner
3	Datsun RediGO T Option	6.0	250000.0	46000	4	1	1	First Owner
4	Honda Amaze VX i-DTEC	9.0	450000.0	141000	1	1	1	Second Owner

```
# *4. Feature Selection*

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_regression
bestfeatures=SelectKBest(mutual_info_regression,k=5)
bestfeatures.fit(X_train.values,Y_train.values)
X_train_selected=X_train[X_train.columns[bestfeatures.get_support()]]

X_train_selected.columns

X_test_selected=X_test[X_test.columns[bestfeatures.get_support()]]

X_train_selected.shape

X_test_selected.shape
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-59-953c0ef6ba3c> in <cell line: 6>()
      4 from sklearn.feature_selection import mutual_info_regression
      5 bestfeatures=SelectKBest(mutual_info_regression,k=5)
----> 6 bestfeatures.fit(X_train.values,Y_train.values)
      7 X_train_selected=X_train[X_train.columns[bestfeatures.get_support()]]
      8

                                 4 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/_array_api.py in _asarray_with_order(array, dtype, order, copy, xp)
    183     if xp.__name__ in {"numpy", "numpy.array_api"}:
    184         # Use NumPy API to support order
--> 185         array = numpy.asarray(array, order=order, dtype=dtype)
    186         return xp.asarray(array, copy=copy)
    187     else:

ValueError: could not convert string to float: 'First Owner'
```

SEARCH STACK OVERFLOW

```
# ### *5. Model Selection*
# #### (i) Linear Regression

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
scaler1=MinMaxScaler()
scaler1.fit(X[['Present_Price']])
X['Present_Price']=scaler1.transform(X[['Present_Price']])
sns.displot(X.Present_Price,kde=True)
plt.show()

from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train_selected,Y_train)
Y_pred=lr.predict(X_test_selected)

from sklearn import metrics
from sklearn.metrics import r2_score
r2=r2_score(Y_test,Y_pred)
print(r2)

mae=metrics.mean_absolute_error(Y_test,Y_pred)
mse=metrics.mean_squared_error(Y_test,Y_pred)
print("MAE is:",mae)
print("MSE is:",mse)
```

```
import math
from math import sqrt
rsme=sqrt(mse)
print("RSME is:",rsme)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-61-0e756ddee8f7> in <cell line: 7>()
      5 from sklearn.preprocessing import MinMaxScaler
      6 scaler1=MinMaxScaler()
----> 7 scaler1.fit(X[['Present_Price']])
      8 X['Present_Price']=scaler1.transform(X[['Present_Price']])
      9 sns.displot(X.Present_Price,kde=True)

                                 2 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in _raise_if_missing(self, key, indexer, axis_name)
   6128                 if use_interval_msg:
   6129                     key = list(key)
-> 6130                 raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6131
   6132             not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())

KeyError: "None of [Index(['Present_Price'], dtype='object')] are in the [columns]"

SEARCH STACK OVERFLOW
```

```
# ### *6. Prediction*

for i in X_train_selected.columns:
    print(X_train_selected[i].value_counts())

print('Min.Preset_Price is:',X_train_selected.Present_Price.min())
print('Max.Preset_Price is:',X_train_selected.Present_Price.max())

# #### 1. Age: 5 - 17
# #### 2. Present_Price: varies
# #### 3. Kms_Driven: varies
# #### 4. Fuel_Type: 0 - CNG, 1 - Petrol, 2 - Diesel
# #### 5. Seller_Type: 0 - Dealer, 1 - Individual

features=X_train_selected.columns

X_train_selected.head()

inputs=[]
for f in features:
    f=float(input(f'Enter {f}:'))
    inputs.append(f)

i=np.array(inputs)
i=i.reshape(1,-1)
ans=xg.predict(i)
print('The selling price of this car is predicted as',ans[0])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-63-45124a5e8f22> in <cell line: 3>()
      1 # ### *6. Prediction*
      2
----> 3 for i in X_train_selected.columns:
      4     print(X_train_selected[i].value_counts())
      5

NameError: name 'X_train_selected' is not defined

SEARCH STACK OVERFLOW
```

0s completed at 3:54 PM