Car Pice Prediction using Hyperparameter Tuning

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

In [2]:
```
data=pd.read_csv("car.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Se

2.2 Display basic information about the training set using the info method.

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         6019 non-null   int64  
 1   Name               6019 non-null   object 
 2   Location           6019 non-null   object 
 3   Year               6019 non-null   int64  
 4   Kilometers_Driven  6019 non-null   int64  
 5   Fuel_Type          6019 non-null   object 
 6   Transmission       6019 non-null   object 
 7   Owner_Type         6019 non-null   object 
 8   Mileage            6017 non-null   object 
 9   Engine             5983 non-null   object 
 10  Power              5983 non-null   object 
 11  Seats              5977 non-null   float64
 12  New_Price          824 non-null    object 
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 658.5+ KB
```

In [5]:
```
data.drop("Unnamed: 0",axis=1,inplace=True)
```

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

Out[6]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26 km/
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19. kn
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18 kn
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20. kn
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15 kn

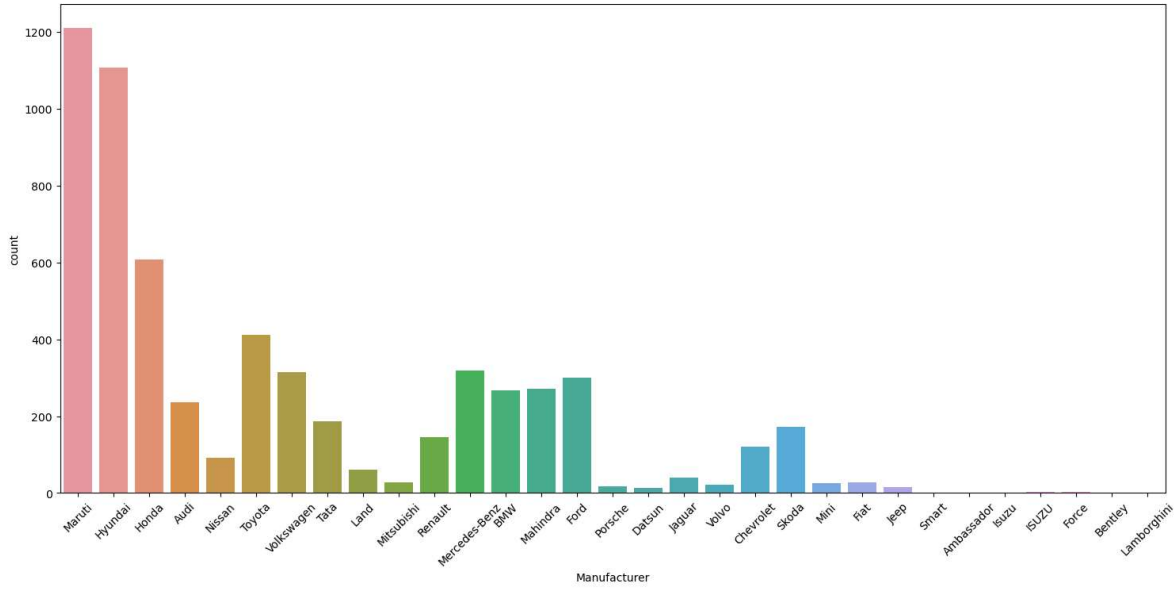
In [7]:
```
data['Manufacturer']=data.Name.str.split(expand=True)[0]
```

In [8]:
```
data.head()
```

Out[8]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Milea
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26 km/
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19. kn
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18 kn
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20. kn
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15 kn

In [9]:
```
plt.figure(figsize=(18,8))
sns.countplot(x=data.Manufacturer)
plt.xticks(rotation=45)
plt.show()
```



In [10]:
```
data.drop('Name',axis=1,inplace=True)
```

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

Out[11]:

	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine
0	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC
1	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC
2	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC
3	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC
4	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC

In [12]:
```
data.drop('Location',axis=1,inplace=True)
```

In [13]:
```
data.head()
```

Out[13]:

	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Se
0	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	
1	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	
2	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	
3	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	
4	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	

In [14]:
```
data=pd.get_dummies(data=data,columns=["Fuel_Type","Transmission","Owner_Type"
```

In [15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 46 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Year                         6019 non-null   int64  
 1   Kilometers_Driven            6019 non-null   int64  
 2   Mileage                      6017 non-null   object 
 3   Engine                       5983 non-null   object 
 4   Power                        5983 non-null   object 
 5   Seats                        5977 non-null   float64
 6   New_Price                    824 non-null    object 
 7   Price                        6019 non-null   float64
 8   Fuel_Type_Diesel             6019 non-null   uint8  
 9   Fuel_Type_Electric           6019 non-null   uint8  
 10  Fuel_Type_LPG                6019 non-null   uint8  
 11  Fuel_Type_Petrol             6019 non-null   uint8  
 12  Transmission_Manual          6019 non-null   uint8  
 13  Owner_Type_Fourth & Above    6019 non-null   uint8  
 14  Owner_Type_Second            6019 non-null   uint8  
 15  Owner_Type_Third             6019 non-null   uint8  
 16  Manufacturer_Audi            6019 non-null   uint8  
 17  Manufacturer_BMW             6019 non-null   uint8  
 18  Manufacturer_Bentley         6019 non-null   uint8  
 19  Manufacturer_Chevrolet       6019 non-null   uint8  
 20  Manufacturer_Datsun          6019 non-null   uint8  
 21  Manufacturer_Fiat            6019 non-null   uint8  
 22  Manufacturer_Force           6019 non-null   uint8  
 23  Manufacturer_Ford            6019 non-null   uint8  
 24  Manufacturer_Honda           6019 non-null   uint8  
 25  Manufacturer_Hyundai         6019 non-null   uint8  
 26  Manufacturer_ISUZU           6019 non-null   uint8  
 27  Manufacturer_Isuzu           6019 non-null   uint8  
 28  Manufacturer_Jaguar          6019 non-null   uint8  
 29  Manufacturer_Jeep            6019 non-null   uint8  
 30  Manufacturer_Lamborghini     6019 non-null   uint8  
 31  Manufacturer_Land            6019 non-null   uint8  
 32  Manufacturer_Mahindra        6019 non-null   uint8  
 33  Manufacturer_Maruti          6019 non-null   uint8  
 34  Manufacturer_Mercedes-Benz   6019 non-null   uint8  
 35  Manufacturer_Mini            6019 non-null   uint8  
 36  Manufacturer_Mitsubishi      6019 non-null   uint8  
 37  Manufacturer_Nissan          6019 non-null   uint8  
 38  Manufacturer_Porsche         6019 non-null   uint8  
 39  Manufacturer_Renault         6019 non-null   uint8  
 40  Manufacturer_Skoda           6019 non-null   uint8  
 41  Manufacturer_Smart           6019 non-null   uint8  
 42  Manufacturer_Tata            6019 non-null   uint8  
 43  Manufacturer_Toyota          6019 non-null   uint8  
 44  Manufacturer_Volkswagen      6019 non-null   uint8  
 45  Manufacturer_Volvo           6019 non-null   uint8  
dtypes: float64(2), int64(2), object(4), uint8(38)
memory usage: 599.7+ KB
```

In [16]:
```
data.drop("Manufacturer_Isuzu",axis=1,inplace=True)
```

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 45 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Year                         6019 non-null   int64  
 1   Kilometers_Driven            6019 non-null   int64  
 2   Mileage                      6017 non-null   object 
 3   Engine                       5983 non-null   object 
 4   Power                        5983 non-null   object 
 5   Seats                        5977 non-null   float64
 6   New_Price                    824 non-null    object 
 7   Price                        6019 non-null   float64
 8   Fuel_Type_Diesel             6019 non-null   uint8  
 9   Fuel_Type_Electric           6019 non-null   uint8  
 10  Fuel_Type_LPG                6019 non-null   uint8  
 11  Fuel_Type_Petrol             6019 non-null   uint8  
 12  Transmission_Manual          6019 non-null   uint8  
 13  Owner_Type_Fourth & Above    6019 non-null   uint8  
 14  Owner_Type_Second            6019 non-null   uint8  
 15  Owner_Type_Third             6019 non-null   uint8  
 16  Manufacturer_Audi            6019 non-null   uint8  
 17  Manufacturer_BMW             6019 non-null   uint8  
 18  Manufacturer_Bentley         6019 non-null   uint8  
 19  Manufacturer_Chevrolet       6019 non-null   uint8  
 20  Manufacturer_Datsun          6019 non-null   uint8  
 21  Manufacturer_Fiat            6019 non-null   uint8  
 22  Manufacturer_Force           6019 non-null   uint8  
 23  Manufacturer_Ford            6019 non-null   uint8  
 24  Manufacturer_Honda           6019 non-null   uint8  
 25  Manufacturer_Hyundai         6019 non-null   uint8  
 26  Manufacturer_ISUZU           6019 non-null   uint8  
 27  Manufacturer_Jaguar          6019 non-null   uint8  
 28  Manufacturer_Jeep            6019 non-null   uint8  
 29  Manufacturer_Lamborghini     6019 non-null   uint8  
 30  Manufacturer_Land            6019 non-null   uint8  
 31  Manufacturer_Mahindra        6019 non-null   uint8  
 32  Manufacturer_Maruti          6019 non-null   uint8  
 33  Manufacturer_Mercedes-Benz   6019 non-null   uint8  
 34  Manufacturer_Mini            6019 non-null   uint8  
 35  Manufacturer_Mitsubishi      6019 non-null   uint8  
 36  Manufacturer_Nissan          6019 non-null   uint8  
 37  Manufacturer_Porsche         6019 non-null   uint8  
 38  Manufacturer_Renault         6019 non-null   uint8  
 39  Manufacturer_Skoda           6019 non-null   uint8  
 40  Manufacturer_Smart           6019 non-null   uint8  
 41  Manufacturer_Tata            6019 non-null   uint8  
 42  Manufacturer_Toyota          6019 non-null   uint8  
 43  Manufacturer_Volkswagen      6019 non-null   uint8  
 44  Manufacturer_Volvo           6019 non-null   uint8  
dtypes: float64(2), int64(2), object(4), uint8(37)
memory usage: 593.8+ KB
```

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

Out[18]:

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	New_Price	Price	Fuel_Type_Diesel
0	2010	72000	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75	0
1	2015	41000	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50	1
2	2011	46000	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50	0
3	2012	87000	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00	1
4	2013	40670	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN	17.74	1

5 rows × 45 columns

In [19]:
```
data['Mileage']=data.Mileage.str.split(expand=True)[0]
```

In [20]:
```
data.head()
```

Out[20]:

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	New_Price	Price	Fuel_Type_Diesel
0	2010	72000	26.6	998 CC	58.16 bhp	5.0	NaN	1.75	0
1	2015	41000	19.67	1582 CC	126.2 bhp	5.0	NaN	12.50	1
2	2011	46000	18.2	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50	0
3	2012	87000	20.77	1248 CC	88.76 bhp	7.0	NaN	6.00	1
4	2013	40670	15.2	1968 CC	140.8 bhp	5.0	NaN	17.74	1

5 rows × 45 columns

In [21]:
```
data.Mileage.isnull().sum()
```

Out[21]: 2

In [22]:
```
data['Mileage']=pd.to_numeric(data.Mileage,errors="coerce")
```

In [23]:
```
data.Mileage.isnull().sum()
```

Out[23]: 2

In [24]:
```
data.Mileage.fillna(data.Mileage.mean(),inplace=True)
```

In [25]:
```
data.Mileage.isnull().sum()
```

Out[25]: 0

In [26]:
```
data.Mileage.dtype
```

Out[26]: dtype('float64')

In [27]:
```
data['Engine']=data.Engine.str.split(expand=True)[0]
```

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

Out[28]:

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	New_Price	Price	Fuel_Type_Diesel
0	2010	72000	26.60	998	58.16 bhp	5.0	NaN	1.75	0
1	2015	41000	19.67	1582	126.2 bhp	5.0	NaN	12.50	1
2	2011	46000	18.20	1199	88.7 bhp	5.0	8.61 Lakh	4.50	0
3	2012	87000	20.77	1248	88.76 bhp	7.0	NaN	6.00	1
4	2013	40670	15.20	1968	140.8 bhp	5.0	NaN	17.74	1

5 rows × 45 columns

In [29]:
```
data['Engine']=pd.to_numeric(data.Engine,errors="coerce")
```

In [30]:
```
data.Engine.isnull().sum()
```

Out[30]: 36

In [31]:
```
data.Engine.fillna(data.Engine.mean(),inplace=True)
```

In [32]:
```
data.Engine.isnull().sum()
```

Out[32]: 0

In [33]:
```
data['Power']=data.Power.str.split(expand=True)[0]
```

In [34]:
```
data['Power']=pd.to_numeric(data.Power,errors="coerce")
```

In [35]:
```
data.Power.isnull().sum()
```

Out[35]: 143

In [36]:
```
data.Power.fillna(data.Power.mean(),inplace=True)
```

In [37]:
```
data.Power.isnull().sum()
```

Out[37]: 0

In [38]:
```
data.Seats.isnull().sum()
```

Out[38]: 42

In [39]:
```
data.Seats.fillna(data.Seats.mean(),inplace=True)
```

In [40]:
```
data.Seats.isnull().sum()
```

Out[40]: 0

In [41]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 45 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Year                         6019 non-null   int64  
 1   Kilometers_Driven            6019 non-null   int64  
 2   Mileage                      6019 non-null   float64
 3   Engine                       6019 non-null   float64
 4   Power                        6019 non-null   float64
 5   Seats                        6019 non-null   float64
 6   New_Price                    824 non-null    object 
 7   Price                        6019 non-null   float64
 8   Fuel_Type_Diesel             6019 non-null   uint8  
 9   Fuel_Type_Electric           6019 non-null   uint8  
 10  Fuel_Type_LPG                6019 non-null   uint8  
 11  Fuel_Type_Petrol             6019 non-null   uint8  
 12  Transmission_Manual          6019 non-null   uint8  
 13  Owner_Type_Fourth & Above    6019 non-null   uint8  
 14  Owner_Type_Second            6019 non-null   uint8  
 15  Owner_Type_Third             6019 non-null   uint8  
 16  Manufacturer_Audi            6019 non-null   uint8  
 17  Manufacturer_BMW             6019 non-null   uint8  
 18  Manufacturer_Bentley         6019 non-null   uint8  
 19  Manufacturer_Chevrolet       6019 non-null   uint8  
 20  Manufacturer_Datsun          6019 non-null   uint8  
 21  Manufacturer_Fiat            6019 non-null   uint8  
 22  Manufacturer_Force           6019 non-null   uint8  
 23  Manufacturer_Ford            6019 non-null   uint8  
 24  Manufacturer_Honda           6019 non-null   uint8  
 25  Manufacturer_Hyundai         6019 non-null   uint8  
 26  Manufacturer_ISUZU           6019 non-null   uint8  
 27  Manufacturer_Jaguar          6019 non-null   uint8  
 28  Manufacturer_Jeep            6019 non-null   uint8  
 29  Manufacturer_Lamborghini     6019 non-null   uint8  
 30  Manufacturer_Land            6019 non-null   uint8  
 31  Manufacturer_Mahindra        6019 non-null   uint8  
 32  Manufacturer_Maruti          6019 non-null   uint8  
 33  Manufacturer_Mercedes-Benz   6019 non-null   uint8  
 34  Manufacturer_Mini            6019 non-null   uint8  
 35  Manufacturer_Mitsubishi      6019 non-null   uint8  
 36  Manufacturer_Nissan          6019 non-null   uint8  
 37  Manufacturer_Porsche         6019 non-null   uint8  
 38  Manufacturer_Renault         6019 non-null   uint8  
 39  Manufacturer_Skoda           6019 non-null   uint8  
 40  Manufacturer_Smart           6019 non-null   uint8  
 41  Manufacturer_Tata            6019 non-null   uint8  
 42  Manufacturer_Toyota          6019 non-null   uint8  
 43  Manufacturer_Volkswagen      6019 non-null   uint8  
 44  Manufacturer_Volvo           6019 non-null   uint8  
dtypes: float64(5), int64(2), object(1), uint8(37)
memory usage: 593.8+ KB
```

In [42]:
```
data.drop('New_Price',axis=1,inplace=True )
```

In [43]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 44 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Year                         6019 non-null   int64  
 1   Kilometers_Driven            6019 non-null   int64  
 2   Mileage                      6019 non-null   float64
 3   Engine                       6019 non-null   float64
 4   Power                        6019 non-null   float64
 5   Seats                        6019 non-null   float64
 6   Price                        6019 non-null   float64
 7   Fuel_Type_Diesel             6019 non-null   uint8  
 8   Fuel_Type_Electric           6019 non-null   uint8  
 9   Fuel_Type_LPG                6019 non-null   uint8  
 10  Fuel_Type_Petrol             6019 non-null   uint8  
 11  Transmission_Manual          6019 non-null   uint8  
 12  Owner_Type_Fourth & Above    6019 non-null   uint8  
 13  Owner_Type_Second            6019 non-null   uint8  
 14  Owner_Type_Third             6019 non-null   uint8  
 15  Manufacturer_Audi            6019 non-null   uint8  
 16  Manufacturer_BMW             6019 non-null   uint8  
 17  Manufacturer_Bentley         6019 non-null   uint8  
 18  Manufacturer_Chevrolet       6019 non-null   uint8  
 19  Manufacturer_Datsun          6019 non-null   uint8  
 20  Manufacturer_Fiat            6019 non-null   uint8  
 21  Manufacturer_Force           6019 non-null   uint8  
 22  Manufacturer_Ford            6019 non-null   uint8  
 23  Manufacturer_Honda           6019 non-null   uint8  
 24  Manufacturer_Hyundai         6019 non-null   uint8  
 25  Manufacturer_ISUZU           6019 non-null   uint8  
 26  Manufacturer_Jaguar          6019 non-null   uint8  
 27  Manufacturer_Jeep            6019 non-null   uint8  
 28  Manufacturer_Lamborghini     6019 non-null   uint8  
 29  Manufacturer_Land            6019 non-null   uint8  
 30  Manufacturer_Mahindra        6019 non-null   uint8  
 31  Manufacturer_Maruti          6019 non-null   uint8  
 32  Manufacturer_Mercedes-Benz   6019 non-null   uint8  
 33  Manufacturer_Mini            6019 non-null   uint8  
 34  Manufacturer_Mitsubishi      6019 non-null   uint8  
 35  Manufacturer_Nissan          6019 non-null   uint8  
 36  Manufacturer_Porsche         6019 non-null   uint8  
 37  Manufacturer_Renault         6019 non-null   uint8  
 38  Manufacturer_Skoda           6019 non-null   uint8  
 39  Manufacturer_Smart           6019 non-null   uint8  
 40  Manufacturer_Tata            6019 non-null   uint8  
 41  Manufacturer_Toyota          6019 non-null   uint8  
 42  Manufacturer_Volkswagen      6019 non-null   uint8  
 43  Manufacturer_Volvo           6019 non-null   uint8  
dtypes: float64(5), int64(2), uint8(37)
memory usage: 546.8 KB
```

In [44]:
```
data.head()
```

Out[44]:

	Year	Kilometers_Driven	Mileage	Engine	Power	Seats	Price	Fuel_Type_Diesel	Fuel_Type
0	2010	72000	26.60	998.0	58.16	5.0	1.75	0	
1	2015	41000	19.67	1582.0	126.20	5.0	12.50	1	
2	2011	46000	18.20	1199.0	88.70	5.0	4.50	0	
3	2012	87000	20.77	1248.0	88.76	7.0	6.00	1	
4	2013	40670	15.20	1968.0	140.80	5.0	17.74	1	

5 rows × 44 columns

In [45]:
```
import datetime
```

In [46]:
```
curr_year=datetime.datetime.now().year
```

In [47]:
```
data['Car_Age']=curr_year-data.Year
```

In [48]:
```
data.drop("Year",axis=1,inplace=True)
```

In [49]:
```
data.head()
```

Out[49]:

	Kilometers_Driven	Mileage	Engine	Power	Seats	Price	Fuel_Type_Diesel	Fuel_Type_Electr
0	72000	26.60	998.0	58.16	5.0	1.75	0	
1	41000	19.67	1582.0	126.20	5.0	12.50	1	
2	46000	18.20	1199.0	88.70	5.0	4.50	0	
3	87000	20.77	1248.0	88.76	7.0	6.00	1	
4	40670	15.20	1968.0	140.80	5.0	17.74	1	

5 rows × 44 columns

In [50]:
```
y=data['Price']
```

In [51]:
```
X=data.drop("Price",axis=1)
```

In [52]:
```
X.head()
```

Out[52]:

	Kilometers_Driven	Mileage	Engine	Power	Seats	Fuel_Type_Diesel	Fuel_Type_Electric	Fue
0	72000	26.60	998.0	58.16	5.0	0	0	
1	41000	19.67	1582.0	126.20	5.0	1	0	
2	46000	18.20	1199.0	88.70	5.0	0	0	
3	87000	20.77	1248.0	88.76	7.0	1	0	
4	40670	15.20	1968.0	140.80	5.0	1	0	

5 rows × 43 columns

In [53]:
```
from sklearn.preprocessing import StandardScaler
```

In [54]:
```
scaler=StandardScaler()
```

In [55]:
```
X=scaler.fit_transform(X)
```

In [56]:
```
X
```

Out[56]:
```
array([[ 0.14531489,  1.84779903, -1.03965343, ..., -0.23499873,
        -0.05917066,  1.02713851],
       [-0.19436922,  0.33507745, -0.0655149 , ..., -0.23499873,
        -0.05917066, -0.50216112],
       [-0.13958146,  0.01419711, -0.7043763 , ..., -0.23499873,
        -0.05917066,  0.72127858],
       ...,
       [-0.0409635 , -0.90260385,  1.46241471, ..., -0.23499873,
        -0.05917066,  0.41541866],
       [-0.13958146,  0.16699727, -1.03965343, ..., -0.23499873,
        -0.05917066,  0.10955873],
       [-0.12862391,  1.59458734, -1.14307225, ..., -0.23499873,
        -0.05917066,  0.72127858]])
```

In [57]:
```
from sklearn.model_selection import cross_val_score, GridSearchCV, RandomizedS
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error

# Define models
models = {
    'RandomForestRegressor': RandomForestRegressor(),
    'SVR': SVR(),
    'KNeighborsRegressor': KNeighborsRegressor()
}
```

2. Baseline Machine Learning Models:
Utilize the loaded dataset to train three baseline machine learning models: RandomForestRegressor, SVR (Support Vector Regressor), and KNeighborsRegressor.
Print the Mean Squared Error (MSE) for each model using 3-fold cross-validation.

In [58]:
```
# 2. Baseline Machine Learning Models
for name, model in models.items():
    mse_scores = cross_val_score(model, X, y, cv=3, scoring='neg_mean_squared_
    mse_mean = np.mean(-mse_scores)
    print(f'{name} MSE: {mse_mean}')
```

```
RandomForestRegressor MSE: 15.322193374547487
SVR MSE: 37.040319138333636
KNeighborsRegressor MSE: 20.04650964523172
```

3.Hyperparameter Optimization (HPO) - Grid Search:
For each of the three models (RandomForestRegressor, SVR, KNeighborsRegressor):
Implement Grid Search to find the optimal hyperparameters.
Print the best hyperparameters and the corresponding MSE.

In [59]:
```
# Define hyperparameter grids
param_grids = {
    'RandomForestRegressor': {'n_estimators': [10, 20, 30], 'max_depth': [15,
    'SVR': {'C': [1, 10, 100], 'kernel': ['poly', 'rbf', 'sigmoid'], 'epsilon
    'KNeighborsRegressor': {'n_neighbors': [2, 3, 5, 7, 10]}
}

# Hyperparameter Optimization - Grid Search
for name, model in models.items():
    param_grid = param_grids[name]
    grid_search = GridSearchCV(model, param_grid, cv=3, scoring='neg_mean_squa
    grid_search.fit(X, y)

    best_params = grid_search.best_params_
    best_mse = -grid_search.best_score_

    print(f'{name} - Grid Search:')
    print(f'Best Hyperparameters: {best_params}')
    print(f'Best MSE: {best_mse}')
```

```
RandomForestRegressor - Grid Search:
Best Hyperparameters: {'max_depth': 50, 'n_estimators': 20}
Best MSE: 15.161402845593623
SVR - Grid Search:
Best Hyperparameters: {'C': 100, 'epsilon': 1, 'kernel': 'rbf'}
Best MSE: 15.247477909847653
KNeighborsRegressor - Grid Search:
Best Hyperparameters: {'n_neighbors': 5}
Best MSE: 20.04650964523172
```

```
4. Hyperparameter Optimization (HPO) - Random Search:
For each of the three models (RandomForestRegressor, SVR,
KNeighborsRegressor):
Implement Random Search to explore hyperparameter space and find the
optimal hyperparameters.
Print the best hyperparameters and the corresponding MSE.
```

In [60]:
```
from scipy.stats import randint as sp_randint, uniform
# Define hyperparameter distributions for Random Search
param_dists = {
    'RandomForestRegressor': {'n_estimators': sp_randint(10, 100), 'max_depth
    'SVR': {'C': uniform(0, 50), 'kernel': ['poly', 'rbf', 'sigmoid'], 'epsil
    'KNeighborsRegressor': {'n_neighbors': sp_randint(1, 20)}
}

# Hyperparameter Optimization - Random Search
for name, model in models.items():
    param_dist = param_dists[name]
    random_search = RandomizedSearchCV(model, param_dist, n_iter=20, cv=3, sc
    random_search.fit(X, y)

    best_params = random_search.best_params_
    best_mse = -random_search.best_score_

    print(f'{name} - Random Search:')
    print(f'Best Hyperparameters: {best_params}')
    print(f'Best MSE: {best_mse}')
```

```
RandomForestRegressor - Random Search:
Best Hyperparameters: {'max_depth': 33, 'n_estimators': 24}
Best MSE: 15.233368541772947
SVR - Random Search:
Best Hyperparameters: {'C': 48.47923138822793, 'epsilon': 0.775132823361114
6, 'kernel': 'rbf'}
Best MSE: 16.06344983301277
KNeighborsRegressor - Random Search:
Best Hyperparameters: {'n_neighbors': 6}
Best MSE: 19.983116255428293
```

In [ ]: