```
#importing all the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Importing dataset
ds=pd.read_csv("car data.csv")
```

```
#Displaying Dataset
ds.head()
```

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

```
#Gives the no of rows and columns as output
ds.shape
```

```
(301, 9)
```

```
#displays all the columns
ds.columns
```

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

```
#Gives the info of the datatypes of attributes
ds.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
```

```
#Displaying all the attributes with object (categorical columns)
ds.select_dtypes(include="object").columns
```

```
Index(['Car_Name', 'Fuel_Type', 'Seller_Type', 'Transmission'],
dtype='object')
```

```
ds.select_dtypes(include=["float64","int64"]).columns #all the attributes of
```

```
Index(['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner'],
dtype='object')
```

```
ds.describe() #gives count ,mean,standard deviation of values
```

	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

```
ds.isna().sum() #gives no of null values for each attribute in dataset
```

```
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
```

```
ds=ds.drop(columns='Car_Name') #no need of car_name
```

```
ds.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Tr
0	2014	3.35	5.59	27000	Petrol	Dealer	
1	2013	4.75	9.54	43000	Diesel	Dealer	
2	2017	7.25	9.85	6900	Petrol	Dealer	
3	2011	2.85	4.15	5200	Petrol	Dealer	
4	2014	4.60	6.87	42450	Diesel	Dealer	

```
ds['Current Year']=2023 #Setting the current_year(new attribute) to 2023 so
```

```
ds.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Tr
0	2014	3.35	5.59	27000	Petrol	Dealer	
1	2013	4.75	9.54	43000	Diesel	Dealer	
2	2017	7.25	9.85	6900	Petrol	Dealer	
3	2011	2.85	4.15	5200	Petrol	Dealer	

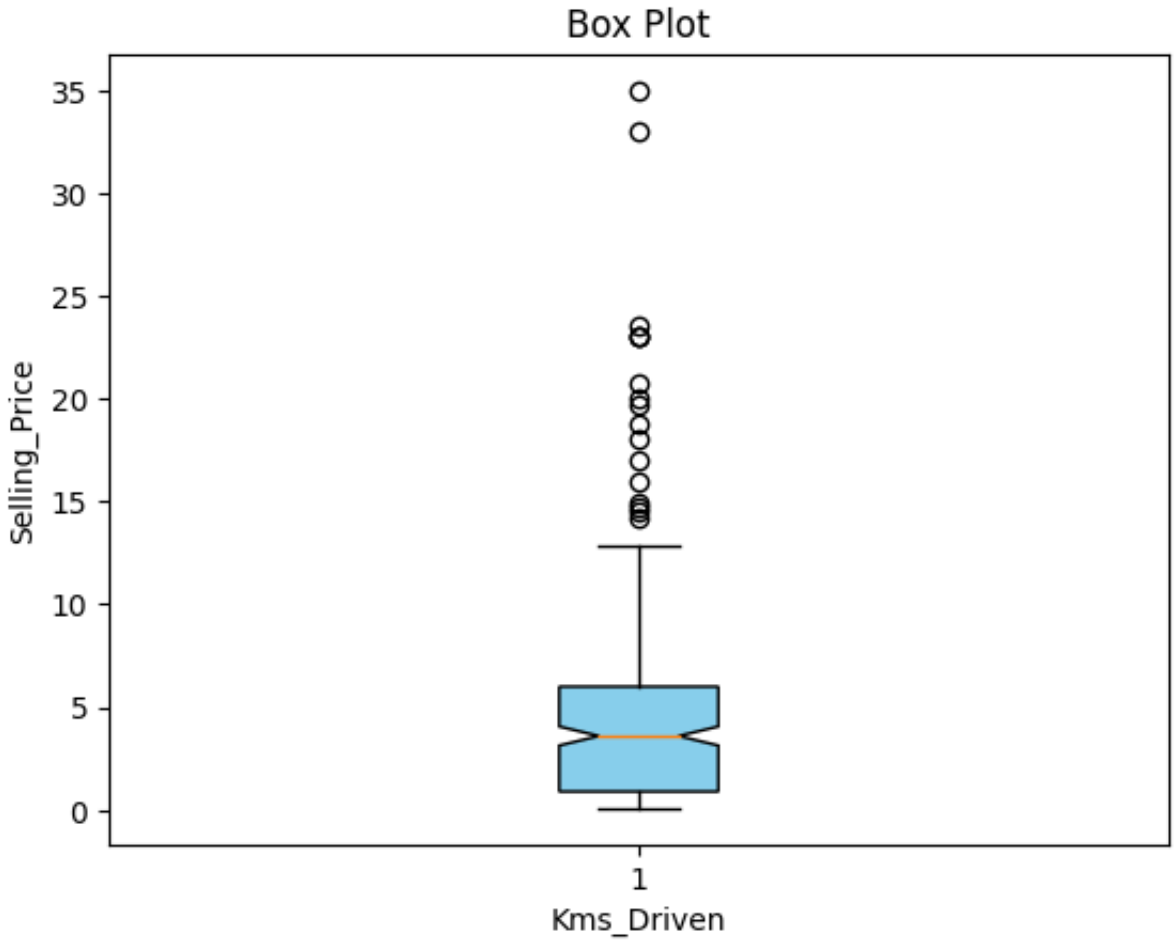
```
ds['Years Old']=ds['Current Year']-ds['Year'] #Calculating no of years the c
ds.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Tr
0	2014	3.35	5.59	27000	Petrol	Dealer	
1	2013	4.75	9.54	43000	Diesel	Dealer	
2	2017	7.25	9.85	6900	Petrol	Dealer	
3	2011	2.85	4.15	5200	Petrol	Dealer	

```
ds=ds.drop(columns=['Year','Current Year']) #after thus no need of year boug
```

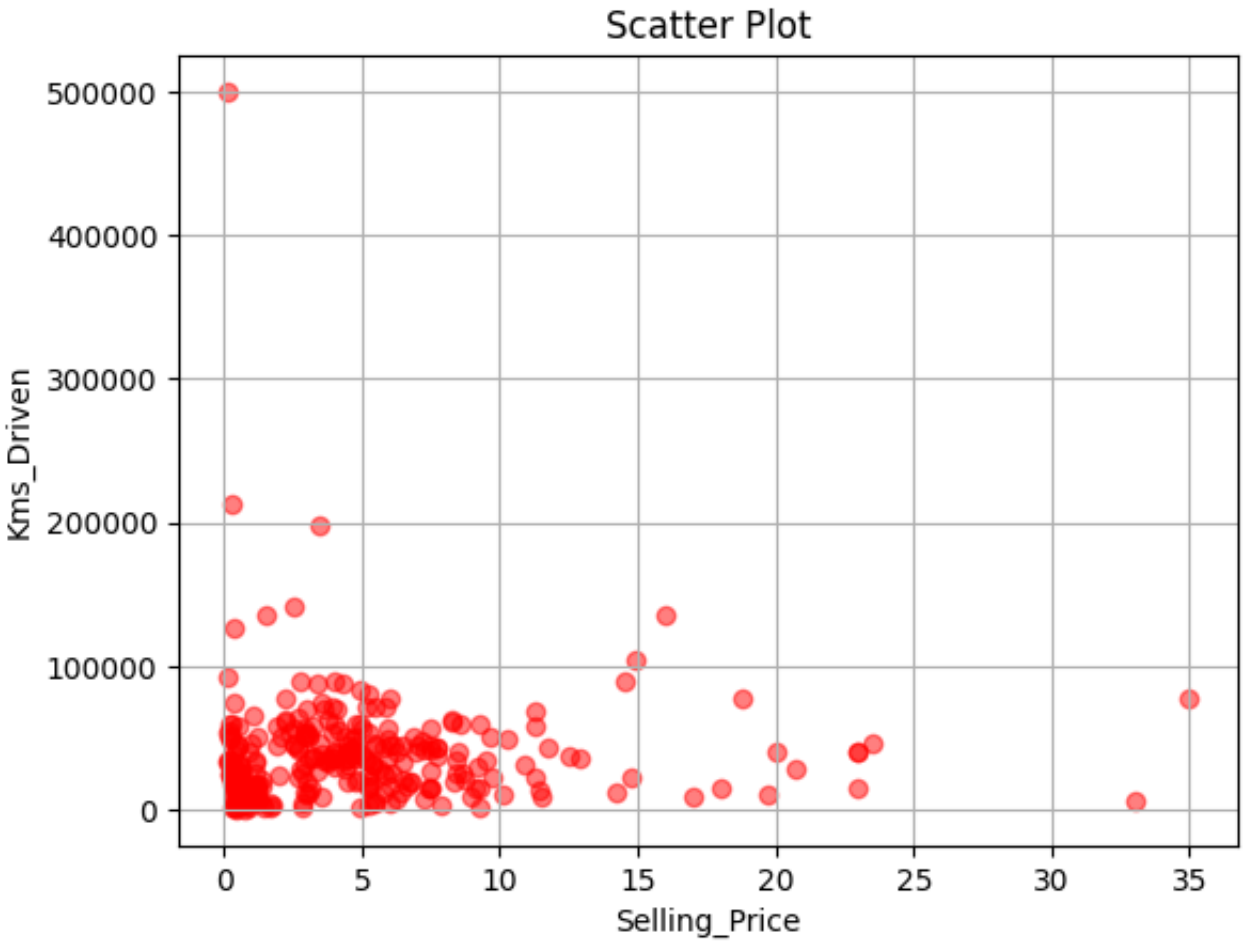
```
plt.boxplot(ds['Selling_Price'], notch=True, patch_artist=True, boxprops=dic
plt.xlabel('Kms_Driven')
plt.ylabel('Selling_Price')
plt.title('Box Plot')
plt.show()

```



```
plt.scatter(ds['Selling_Price'], ds['Kms_Driven'], marker='o', color='red',
plt.xlabel('Selling_Price')
plt.ylabel('Kms_Driven')
plt.title('Scatter Plot')
plt.grid(True)
plt.show()

```




```
ds.head()
```

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmis
0	3.35	5.59	27000	Petrol	Dealer	M
1	4.75	9.54	43000	Diesel	Dealer	M
2	7.25	9.85	6900	Petrol	Dealer	M
3	2.85	4.15	5200	Petrol	Dealer	M

```
ds=pd.get_dummies(data=ds,drop_first=True) #one hot coding,label encoding et
ds.head() #Labeling all object datatypes with numbers
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Years Old	Fuel_Type_Diesel
0	3.35	5.59	27000	0	9	0
1	4.75	9.54	43000	0	10	1
2	7.25	9.85	6900	0	6	0
3	2.85	4.15	5200	0	12	0

```
x=ds.drop(columns="Selling_Price") #Matrix of Features
```

```
y=ds['Selling_Price'] #Target varriable
```

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, ran
```

```
x_train.shape
```

```
(240, 8)
```

```
x_test.shape
```

```
(61, 8)
```

```
import warnings #to remove warnings during the prediction
```

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(criterion='squared_error') #object
warnings.filterwarnings("ignore", category=UserWarning, module="sklearn") # d
regressor.fit(x_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
y_pred=regressor.predict(x_test)
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.9556730772390396
```

HyperParameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV #the default value tak
```

```
parameters = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'criterion': ['squared_error', 'absolute_error', 'poisson', 'friedman_mse
}
```

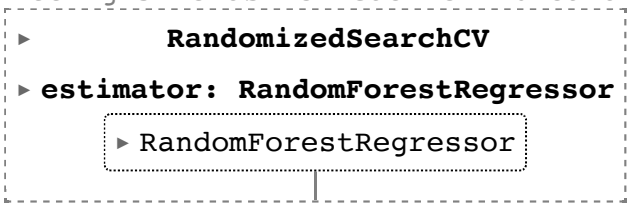
```
parameters
```

```
{'n_estimators': [100, 200, 300],
 'max_depth': [None, 5, 10],
 'min_samples_split': [2, 5, 10],
 'min_samples_leaf': [1, 2, 4],
 'criterion': ['squared_error', 'absolute_error', 'poisson',
'friedman_mse']}
```

```
random_cv = RandomizedSearchCV(estimator=regressor, param_distributions=param
```

```
random_cv.fit(x_train, y_train) #Training the data
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```



```
1 random_cv.best_estimator_
```

```
RandomForestRegressor(max_depth=10, min_samples_leaf=2, n_estimators=300)
```

```
1 random_cv.best_params_
```

```
{'n_estimators': 300,
 'min_samples_split': 2,
 'min_samples_leaf': 2,
 'max_depth': 10,
 'criterion': 'squared_error'}
```

```
1 ds.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Years Old	Fuel_Type_Diesel
0	3.35	5.59	27000	0	9	0
1	4.75	9.54	43000	0	10	1
2	7.25	9.85	6900	0	6	0
3	2.85	4.15	5200	0	12	0

Testing

```
1 single_obs1=[[8.50,3500,0,5,1,0,0,1]]
```

Double-click (or enter) to edit

```
1 regressor.predict(single_obs1)
```

```
array([7.3015])
```

```
1 single_obs2=[[6.50,3300,1,5,1,0,0,1]]
```

```
1 regressor.predict(single_obs2)
```

```
array([5.6525])
```

```
1 single_obs3=[[9.50,3700,1,5,1,0,0,1]]
```

```
1 regressor.predict(single_obs3)
```

```
array([8.2888])
```

```
1 single_obs4=[[9.10,3000,1,6,2,0,0,1]]
```

```
1 regressor.predict(single_obs4)
```

```
array([8.1623])
```

```
1 single_obs5=[[9.10,3600,1,6,2,0,0,1]]
```

```
1 regressor.predict(single_obs5)
```

```
array([8.1623])
```