Car Price Prediction using Linear Regression

Use Linear Regression (Ordinary Least Square) to Predict car price This tutorial explains the necessary steps in coding. To get a solution viewers must realise that OLS require the fullfillment of assumptions for effective modeling. To learn regression technique join our free courses at ybifoundation.org

Get understanding about data set

There are 9 vairable in the dataset

1. Brand-manaufacturing company
2. Model-model of cars
3. Year-year of manaufacturing
4. KM_Driven-total km driven
5. Selling_Price-selling price of car
6. Fuel-type of fuel used in car
7. Seller_Tpye-type of seller
8. Transmissin-type pf transmmion in car
9. Owner-whether current owner is first owner or repurchased

Import Library

```
import pandas as pd
```

```
import numpy as np
```

```
df = pd.read_csv(r'https://github.com/YBI-Foundation/Dataset/raw/main/Car%20Price.csv')
```

```
# df = pd.read_csv(r'C:\Users\YBI Foundation\Desktop\Car Price.csv')

# df = pd.read_csv(r'/content/Car Price.csv')
```

Get the Frist Five Rows of Dataframe

```
df.head()
```

	Brand	Model	Year	Selling_Price	KM_Driven	Fuel	Seller_Type	Transmission
0	Maruti	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual
1	Maruti	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual

Get Information of Dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Brand          4340 non-null   object
 1   Model          4340 non-null   object
 2   Year           4340 non-null   int64 
 3   Selling_Price  4340 non-null   int64 
 4   KM_Driven      4340 non-null   int64 
 5   Fuel           4340 non-null   object
 6   Seller_Type    4340 non-null   object
 7   Transmission   4340 non-null   object
 8   Owner          4340 non-null   object
dtypes: int64(3), object(6)
memory usage: 305.3+ KB
```

Get the Summary Statistics

```
df.describe()
```

	Year	Selling_Price	KM_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

Get Categories and Counts of Categorical Variables

```
df[['Brand']].value_counts()
```

```
Brand
Maruti          1280
Hyundai          821
Mahindra         365
Tata             361
Honda            252
Ford             238
Toyota           206
Chevrolet        188
Renault          146
Volkswagen       107
Skoda             68
Nissan            64
Audi              60
BMW               39
Fiat              37
Datsun            37
Mercedes-Benz     35
Mitsubishi         6
Jaguar             6
Land               5
Ambassador         4
Volvo              4
Jeep               3
OpelCorsa          2
MG                 2
Isuzu              1
Force              1
Daewoo             1
Kia                1
dtype: int64
```

```
df[['Model']].value_counts()
```

```
Model
Maruti Swift Dzire VDI                 69
Maruti Alto 800 LXI                    59
Maruti Alto LXi                        47
Hyundai EON Era Plus                   35
Maruti Alto LX                         35
                                       ..
Mahindra KUV 100 G80 K4 Plus            1
Mahindra KUV 100 mFALCON D75 K8         1
Mahindra KUV 100 mFALCON D75 K8 AW      1
Mahindra KUV 100 mFALCON G80 K2 Plus    1
Volvo XC60 D5 Inscription               1
Length: 1491, dtype: int64
```

```
df[['Fuel']].value_counts()
```

```
Fuel
Diesel      2153
Petrol      2123
CNG           40
LPG           23
Electric       1
dtype: int64
```

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

```
Seller_Type
Individual          3244
Dealer               994
Trustmark Dealer     102
dtype: int64
```

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

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

```
df[['Owner']].value_counts()
```

```
Owner
First Owner             2832
Second Owner            1106
Third Owner              304
Fourth & Above Owner      81
Test Drive Car            17
dtype: int64
```

```
#df[['Fuel','Seller_Tpype','Transmission','Owner']].value_counts()
```

Get Column Names

```
df.columns
```

```
Index(['Brand', 'Model', 'Year', 'Selling_Price', 'KM_Driven', 'Fuel',
       'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

Get Shape of Dataframe

```
df.shape
```

```
(4340, 9)
```

Get Encoding of Categorical Features

```
df.replace({'Fuel':{'Petrol':0,'Diesel':1,'CNG':2,'LPG':3,'Electric':4}},inplace=True)
```

```
df.replace({'Seller_Type':{'Individual':0,'Dealer':1,'Trustmark Dealer':2}},inplace=True)
```

```
df.replace({'Treansmission':{'Manual':0,'Automatic':1}},inplace=True)
```

```
df.replace({'Owner':{'First Owner':0,'Second Owner':1,'Third Owner':2,'Fourth & Above Owner':
```

```
#x =pd.get_dummies(x,columns=['Fuel','Seller_Type','Transmission','Owner'], drop_first=True)
```

Define y(dependent or label or target variable) and x(independent or features or attribute variable)

```
y = df['Selling_Price']
```

```
y.shape
```

```
(4340,)
```

```
y
```

```
0        60000
1       135000
2       600000
3       250000
4       450000
         ...
4335    409999
4336    409999
4337    110000
4338    865000
4339    225000
Name: Selling_Price, Length: 4340, dtype: int64
```

```
x = df[['Year','KM_Driven','Fuel','Seller_Type','Transmission','Owner']]
```

```
#x = df.drop(['Brand','Model,'Selling_Price'], axis=1)
```

```
x.shape
```

```
(4340, 6)
```

```
x
```

```
            Year  KM_Driven  Fuel  Seller_Type  Transmission  Owner
```

Get Train Test Split

```
from sklearn.model_selection import  train_test_split
```

```
      3   2017     40000     0          0        Manual      0
```

```
x_train, x_test, y_train, y_test= train_test_split(x,y, test_size= 0.3, random_state=2529 )
```

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((3038, 6), (1302, 6), (3038,), (1302,))
```

Get future predictions

Let select a random sample from existing dataset as new value Step to follow

1. extract a random row using sample function
2. separate X and y
3. predict

```
df_new = df.sample(1)
```

```
df_new
```

	Brand	Model	Year	Selling_Price	KM_Driven	Fuel	Seller_Type	Transmission	Ow
		BMW							

```
df_new.shape
```

```
(1, 9)
```

```
X_new = df_new.drop([ 'Brand','Selling_Price'],axis = 1)
```

0s completed at 4:51 AM