ASSIGNMENT 2

HOUSEPRICEPREDICTION USING LINEAR REGRESSION

Name-:Sidhant satapathy
Regd no-:21-27-06
Branch-:DATASCIENCE

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
data=pd.read_csv('Housepricepredict ion.csv')
data=data.drop(['Id'],axis=1)
data.head()
```

Out[2]:

	LotArea	OverallQual	1stFlrSF	GrLivArea	TotRmsAbvGrd	GarageCars	GarageArea	SalePrice
0	8450	7	856	1710	8	2	548	208500
1	9600	6	1262	1262	6	2	460	181500
2	11250	7	920	1786	6	2	608	223500
3	9550	7	961	1717	7	3	642	140000
4	14260	8	1145	2198	9	3	836	250000

In [3]:
```
def normalize(df):
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        result[feature_name] = (df[feature_name] - min_value) / (max_value - min_value)
    return result
```

In [4]:
```
data_normalized=normalize(data)
data_normalized.head()
```

Out[4]:

	LotArea	OverallQual	1stFlrSF	GrLivArea	TotRmsAbvGrd	GarageCars	GarageArea	SalePrice
0	0.033420	0.666667	0.119780	0.259231	0.500000	0.50	0.386460	0.241078
1	0.038795	0.555556	0.212942	0.174830	0.333333	0.50	0.324401	0.203583
2	0.046507	0.666667	0.134465	0.273549	0.333333	0.50	0.428773	0.261908
3	0.038561	0.666667	0.143873	0.260550	0.416667	0.75	0.452750	0.145952
4	0.060576	0.777778	0.186095	0.351168	0.583333	0.75	0.589563	0.298709

In [5]:
```
y=data_normalized['SalePrice'].values
x=data_normalized.drop(['SalePrice'],axis=1)
y.shape
```

Out[5]:
```
(1460,)
```

In [6]:
```
X_train=x[:1000]
Y_train=y[:1000]
X_test=x[1000:]
Y_test=y[1000:]
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(1000, 7) (1000,)
(460, 7) (460,)
```

In [7]:
```
init_weights=np.array([0,0,0,0,0,0,0,0])
```

```
bias=np.array([1 for i in range(1000)]).reshape(1000,1)
bias1=np.array([1 for i in range(460)]).reshape(460,1)
print(bias.shape)
X_train=np.array(X_train)
Y_train=np.array(Y_train)
X_train=np.hstack((bias,X_train))
X_test=np.hstack((bias1,X_test))
learningrate=0.01
```

```
(1000, 1)
```

In [14]:
```
def gradient_descent(init_weights,X_train,Y_train,learningrate):
    for i in range(len(X_train)):
        h=np.dot(init_weights,np.transpose(X_train[i]))
        a=np.dot(float(Y_train[i]-h),X_train[i])
        init_weights=init_weights+learningrate*a

    return init_weights
```

In [15]:
```
epochs=1000
for i in range(epochs):
    updated=gradient_descent(init_weights,X_train,Y_train,learningrate)
    init_weights=updated
print(init_weights)
```

```
[-0.14287045  0.18615064  0.32324598  0.2139313   0.31194918  0.00577725
  0.01997427  0.0949817 ]
```

In [10]:
```
y_predicted=[]
for i in range(len(X_test)):
    a=np.dot(init_weights,np.transpose(X_test[i]))
    y_predicted.append(a)
```

In [11]:
```
mean=np.mean(Y_test)
Y_actual=Y_test.tolist()
error=0
for i in range(len(Y_actual)):
    error=error+(Y_actual[i]-y_predicted[i])**2
SS_res=error
MSE=error/len(Y_actual)
MSE
```

Out[11]:
```
0.003656030698170533
```

In [17]:
```
sum=0
for i in range(len(Y_actual)):
    t=(Y_actual[i]-mean)**2
    sum=sum+t
SS_total=sum

R_two=1-(SS_res/SS_total)
R_two
```

Out[17]:
```
0.6842348618935512
```

HOUSEPRICE PREDICTION USING NORMAL EQUATIONS

In [459…
```
Normaleq_weights=np.dot(np.linalg.pinv(np.dot(X_train.T,X_train)),np.dot(X_train.T,Y_train))
Normaleq_weights
```

Out[459…
```
array([-0.14014159,  0.1843313 ,  0.32197978,  0.21630905,  0.30926841,
        0.00620942,  0.02180226,  0.09595618])
```

In [460…
```
y_predicted1=[]
for i in range(len(X_test)):
    a=np.dot(Normaleq_weights,np.transpose(X_test[i]))
    y_predicted1.append(a)
```

In [461…

```
mean=np.mean(Y_test)
Y_actual=Y_test.tolist()
error=0
for i in range(len(Y_actual)):
    error=error+(Y_actual[i]-y_predicted1[i])**2
SS_res=error
MSE=error/len(Y_actual)
MSE
```

Out[461…

```
0.0036749790321171398
```

In [462…

```
sum=0
for i in range(len(Y_actual)):
    t=(Y_actual[i]-mean)**2
    sum=sum+t
SS_total=sum
R_two=1-(SS_res/SS_total)
R_two
```

Out[462…

```
0.6825983265973184
```

Comparission with Sklearn LinearRegression

In [465…

```
from sklearn.linear_model import LinearRegression

linear_regression = LinearRegression()
linear_regression.fit(X_train,Y_train )
linear_regression.score(X_test,Y_test)
```

Out[465…

```
0.6825983265973219
```