EXPERIMENT NO:5

DATE: 03-02-2022

Aim :

You are given two files train.csv and test.csv containing the training data and testing data respectively. Each file contains two columns: a feature and a label.

Problem Description

1. Understanding the data and simple curve fitting

a. Plot a feature vs label graph for both the training data and the test data.

b. Write a code to fit a curve that minimizes squared error cost function using gradient descent (with learning rate 0.05), on the training set while the model takes the following polynomials of degree 0, 1, 2, ...,9. In other words, fit 9 different curves to the training data, and hence estimate the parameters. Compare the estimated values with test data and measure squared error on the test set, name it as test error.

2. Visualization of the fitted curves

Draw separate plots of all 9 different curves that you have fit for the training dataset. Report squared error on both train and test data for each value of n in the form of a plot where along x-axis, vary n from 1 to 9 and along y-axis, plot both training error and test error. Explain which value of n is suitable for the dataset that you have, and why?

3. Regularization

Perform the following regularizations on the curves for which you obtain the minimum and maximum training error from above. Plot both training and test error after the regularization (use $\lambda = 0.25, 0.5, 0.75, 1$)

In [85]:
```
#You are given two files train.csv and test.csv containing the training data and testing data respectively. Each file contains
#two columns: a feature and a label.
#1. Plot a feature vs label graph for both the training data and the test data.
#2. Write a code to fit a curve that minimizes squared error cost function using gradient descent (with learning rate 0.05),
#on the training set by using linear regression model.
```

In [86]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 9.0)
```

In [87]:
```
# Preprocessing Input data
df = pd.read_csv('train.csv')
X = df.iloc[:, 0]
Y = df.iloc[:, 1]
dft = pd.read_csv('test.csv')
Xtest = dft.iloc[:, 0]
Ytest = dft.iloc[:, 1]
#plt.scatter(X, Y)
#plt.show()
```

In [88]:
```
# Building the model
theta1 = 0
theta0 = 0

alpha = 0.0001  # The learning Rate
epochs = 10000  # The number of iterations to perform gradient descent
```

In [89]:
```
m = float(len(X)) # Number of training examples
m
```

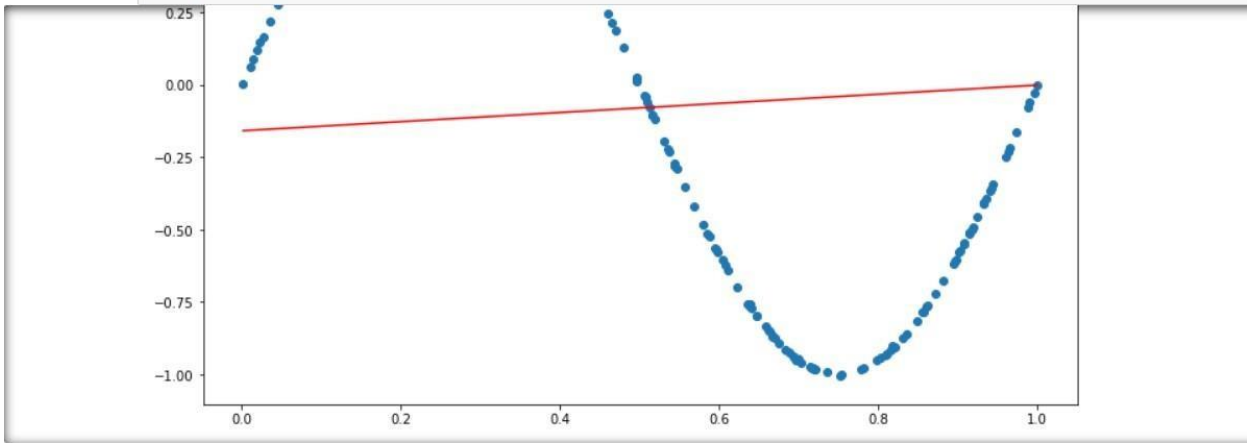
Out[89]:
```
1000.0
```

In [90]:
```
# Performing Gradient Descent
for i in range(epochs):
    h_x = theta0 + theta1*X   # The current predicted value of Y
    #h_x - Y
    print(h_x.iloc[0]-Y.iloc[0])
    d_theta1 = (1/m) * sum(X * (h_x -Y))  # Derivative wrt theta1
    d_theta0 = (1/m) * sum(h_x -Y)  # Derivative wrt theta0
    theta1 = theta1 - alpha * d_theta1  # Update theta1
    theta0 = theta0 - alpha * d_theta0  # Update theta0
#print (theta1, theta0)
```

```
0.2293287885420844
0.2293529961834792
0.22928181311449022
0.22925832903020296
0.2292348473651778
0.22921136811910647
0.22918789129168077
0.22916441688259248
0.22914094489153347
0.2291174753181956
0.22909400816227082
0.22907054342345107
0.22904708110142838
0.22902362119589478
0.2290001637065424
0.22897670863306327
0.22895325597514965
0.2289298057324937
0.22890635790478767
0.22888291249172388
```

In [91]:
```
# Making predictions
Y_pred = theta0 + theta1*Xtest

plt.scatter(Xtest, Ytest)
plt.plot([min(Xtest), max(Xtest)], [min(Y_pred), max(Y_pred)], color='red') # predicted
plt.show()
```



In [92]:
```
np.square(Y_pred - Ytest)
```

Out[92]:
```
0      0.708484
1      0.139499
2      0.546446
3      0.763873
4      0.163007
         ...
195    0.874260
196    0.286867
197    0.359005
198    0.422175
199    0.931540
Length: 200, dtype: float64
```