LAB CYCLE 5

Experiment No:5
Date:09/02/22

Aim:

You are given two files train.csv and test.csv containing the training dataand 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 [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 9.0)
```

In [27]:
```
train=pd.read_csv('train.csv')
train
```

Out[27]:

	Feature	Label
0	0.963585	-0.229634
1	0.715377	-0.979414
2	0.896298	-0.608057
3	0.049025	0.306430
4	0.299481	0.952607
...	...	...
995	0.196550	0.943661
996	0.068852	0.417877
997	0.786102	-0.971872
998	0.050846	0.310817
999	0.429965	0.428704

1000 rows × 2 columns

In [28]:
```
train.shape
```

Out[28]:
```
(1000, 2)
```

In [29]:
```
train.head()
```

Out[29]:

	Feature	Label
0	0.963585	-0.229634
1	0.715377	-0.979414
2	0.896298	-0.608057
3	0.049025	0.306430
4	0.299481	0.952607

In [30]:
```
train.describe()
```

Out[30]:

	Feature	Label
count	1000.000000	1000.000000

	Feature	Label
mean	0.521191	-0.050581
std	0.298140	0.712278
min	0.001970	-1.006216
25%	0.261184	-0.762565
50%	0.545637	-0.123868
75%	0.780254	0.664140
max	0.999975	1.005695

In [31]:
```
X=train.iloc[:, 0]
Y=train.iloc[:, 1]
```

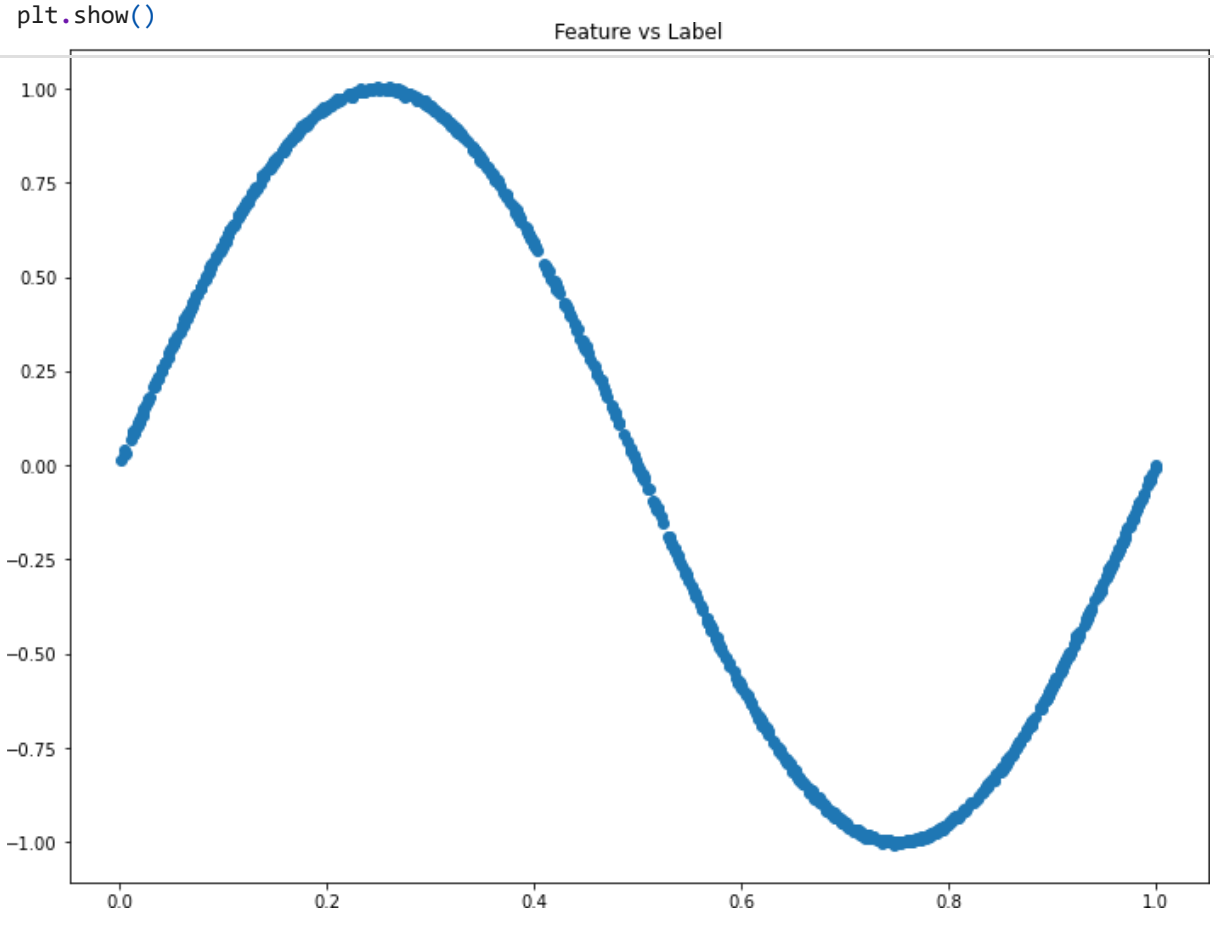
In [32]:
```
X.shape
```

Out[32]:
```
(1000,)
```

In [33]:
```
Y.shape
```

Out[33]:
```
(1000,)
```

In [34]:
```
plt.scatter(X,Y)
plt.title('Feature vs Label')
plt.show()
```



In [35]:
```
test=pd.read_csv('test.csv')
 test
```

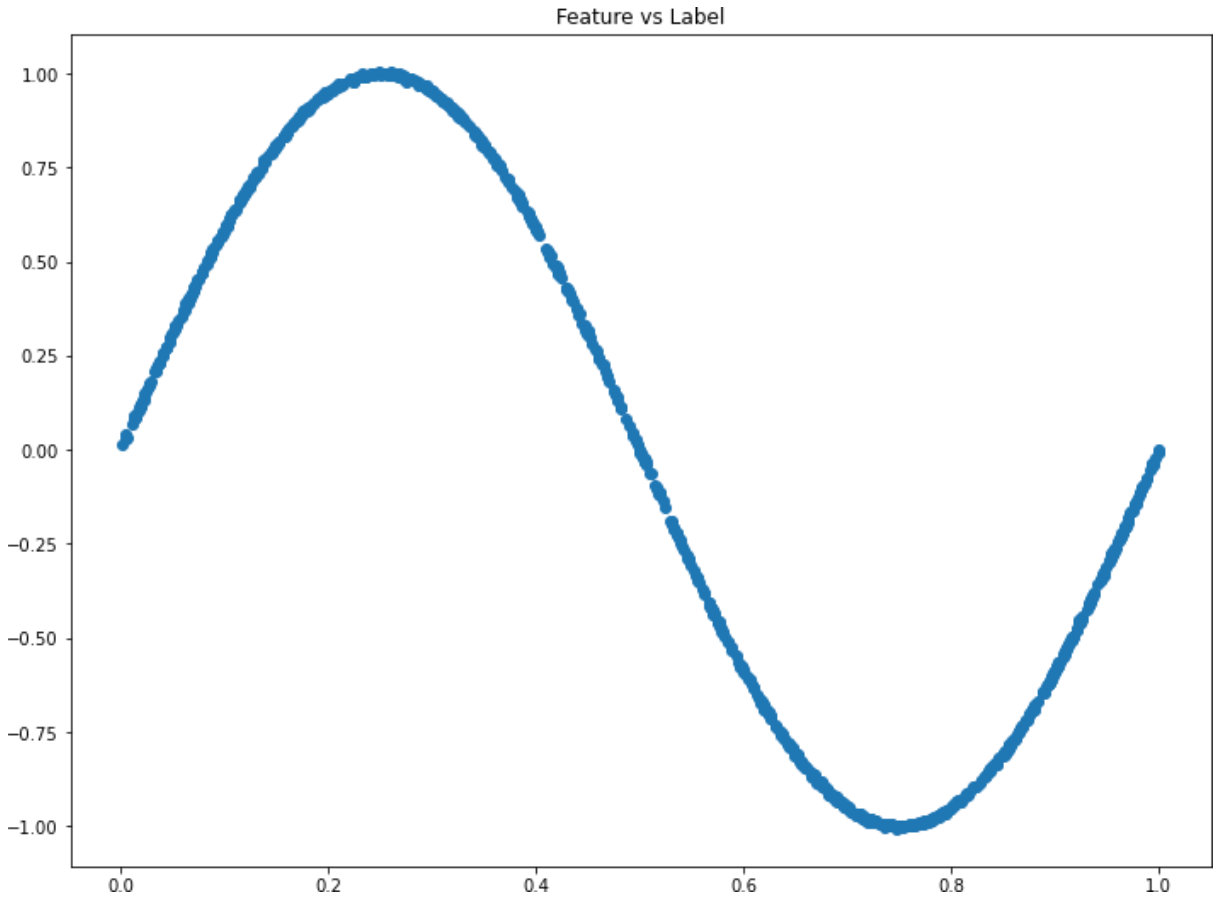
Out[35]:

	Feature	Label
0	0.355414	0.785368
1	0.451334	0.302000
2	0.127785	0.718821
3	0.734916	-0.990279
4	0.445669	0.333141
...	...	...
195	0.181268	0.906174
196	0.881642	-0.675052
197	0.621949	-0.697610
198	0.109905	0.632176
199	0.192497	0.934545

200 rows × 2 columns

In [36]:
```
Xtest=train.iloc[:, 0]
Ytest=train.iloc[:, 1]
```

In [37]:
```
plt.scatter(Xtest,Ytest)
plt.title('Feature vs Label')
plt.show()
```



In [38]:
```
# Building the model
theta1 = 0
theta0 = 0

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

In [39]:
```
m = float(len(X)) # Number of training examples
m
```

Out[39]:
```
1000.0
```

In [40]:
```
    theta1 = theta1 - alpha * d_theta1  # Update theta1
# 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
```

```
    theta0 = theta0 - alpha * d_theta0  # Update theta0
#print (theta1, theta0)
```

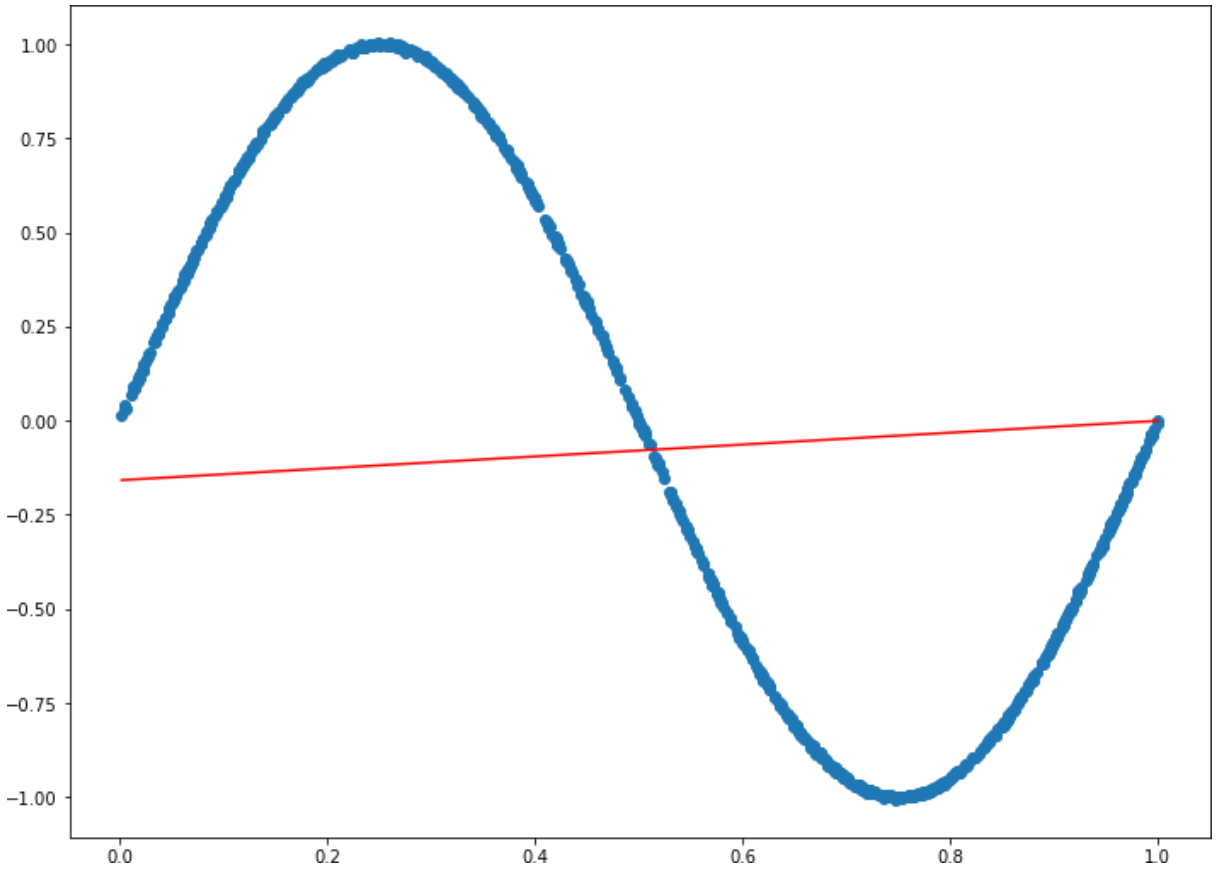
```
0.2296343649
0.22961084448976127
0.2295873265034135
0.22956381094064787
0.2295402978011555
0.22951678708462764
0.22949327879075548
0.22946977291923037
0.22944626946974359
0.22942276844198656
0.22939926983565065
0.22937577365042733
0.2293522798860081
0.22932878854208444
0.22930529961834795
0.22928181311449025
0.229258329030203
0.22923484736517782
0.2292113681191065
0.2291878912916808
0.2291644168825925
0.2291409448915335
0.22911747531819562
0.22909400816227085
0.2290705434234511
0.2290470811014284
0.2290236211958948
0.22900016370654241
0.2289767086330633
0.22895325597514968
0.22892980573249372
0.2289063579047877
0.2288829124917239
0.2288594694929946
0.22883602890829222
0.22881259073730914
0.2287891549797378
0.2287657216352707
0.22874229070360033
0.22871886218441928
0.22869543607742016
0.22867201238229556
0.22864859109873822
0.22862517222644083
0.22860175576509617
0.228578341714397
0.22855493007403624
0.22853152084370668
0.22850811402310128
0.22848470961191303
0.22846130760983485
0.22843790801655986
0.22841451083178108
0.22839111605519166
0.22836772368648475
0.22834433372535354
0.22832094617149126
0.22829756102459123
0.2282741782843467
0.22825079795045106
0.22822742002259772
```

```
0.07742649356792505
0.07741670875491685
0.077406924632315
0.07739714120003421
0.07738735845798933
0.07737757640609508
0.0773677950442663
0.0773580143724178
0.07734823439046434
0.07733845509832082
0.07732867649590203
0.07731889858312285
0.07730912135989815
0.07729934482614276
0.07728956898177161
0.07727979382669958
0.07727001936084155
0.07726024558411249
0.07725047249642725
```

In [16]:

```
# 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') # predic
plt.show()
```



In [17]:

```
np.square(Y_pred - Ytest)
```

Out[17]:

```
0      0.005966
1      0.750336
2      0.217426
3      0.098841
4      1.000242
```

```
          ...
995    0.950468
996    0.184013
997    0.718269
998    0.101802
999    0.246834
Length: 1000, dtype: float64
```