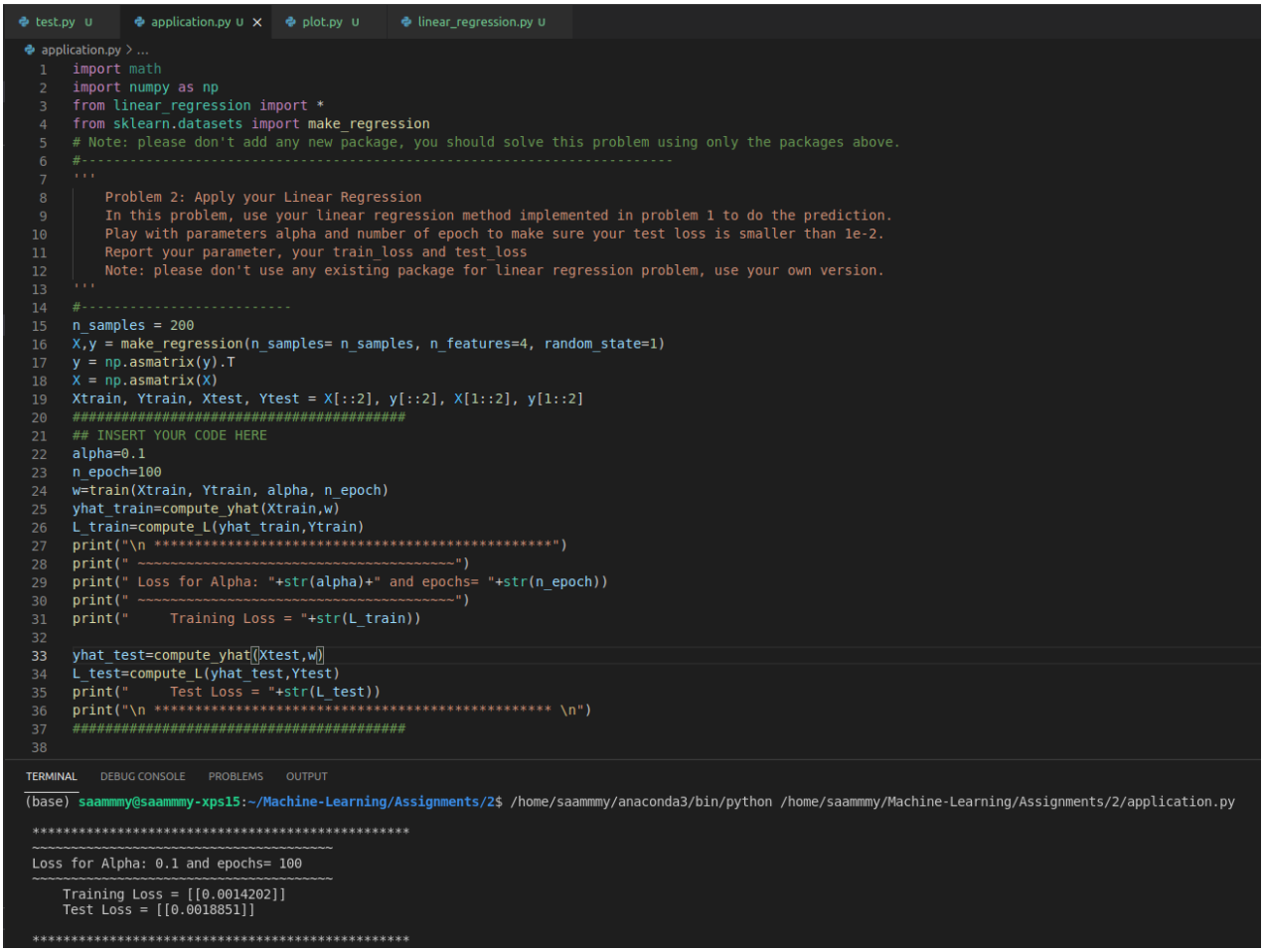
Name: Samarth Shah
WPI ID: 590146168

Date: 09/06/2021
Course: CS539-Machine Learning

Homework #2

Part 2: Make Predictions by using your implementation.

- First, I inserted my code into the application.py script.
- Below is my code where I trained the data and used the test data to find training loss.
- The o/p below shows both the training and test loss.



```
test.py U   application.py U   plot.py U   linear_regression.py U
application.py > ...
import math
import numpy as np
from linear_regression import *
from sklearn.datasets import make_regression
# Note: please don't add any new package, you should solve this problem using only the packages above.
#------------------------------------------------------------------------
'''
    Problem 2: Apply your Linear Regression
    In this problem, use your linear regression method implemented in problem 1 to do the prediction.
    Play with parameters alpha and number of epoch to make sure your test loss is smaller than 1e-2.
    Report your parameter, your train_loss and test_loss
    Note: please don't use any existing package for linear regression problem, use your own version.
'''
#--------------------------
n_samples = 200
X,y = make_regression(n_samples= n_samples, n_features=4, random_state=1)
y = np.asmatrix(y).T
X = np.asmatrix(X)
Xtrain, Ytrain, Xtest, Ytest = X[::2], y[::2], X[1::2], y[1::2]
########################################
## INSERT YOUR CODE HERE
alpha=0.1
n_epoch=100
w=train(Xtrain, Ytrain, alpha, n_epoch)
yhat_train=compute_yhat(Xtrain,w)
L_train=compute_L(yhat_train,Ytrain)
print("\n ***********************************************")
print(" ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
print(" Loss for Alpha: "+str(alpha)+" and epochs= "+str(n_epoch))
print(" ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
print("     Training Loss = "+str(L_train))

yhat_test=compute_yhat(Xtest,w)
L_test=compute_L(yhat_test,Ytest)
print("     Test Loss = "+str(L_test))
print("\n *********************************************** \n")
########################################

TERMINAL  DEBUG CONSOLE  PROBLEMS  OUTPUT
(base) saammmy@saammmy-xps15:~/Machine-Learning/Assignments/2$ /home/saammmy/anaconda3/bin/python /home/saammmy/Machine-Learning/Assignments/2/application.py

 ***********************************************
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Loss for Alpha: 0.1 and epochs= 100
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     Training Loss = [[0.0014202]]
     Test Loss = [[0.0018851]]

 ***********************************************
```

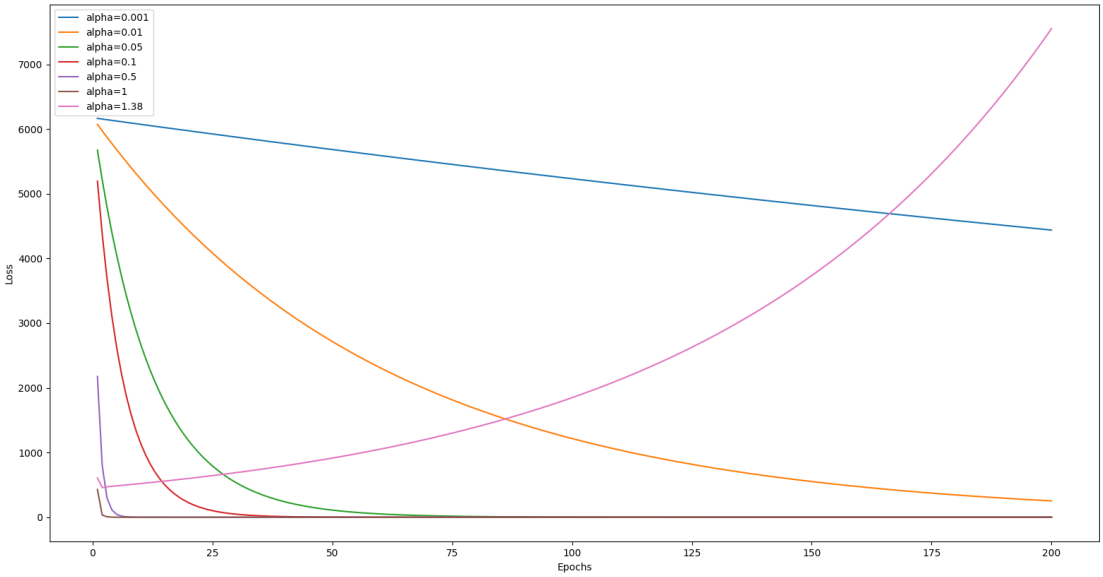
- After playing with values of epoch and alpha, I found that at:
  - **alpha=0.1 & epochs=100:** Training Loss= 0.0014202 Test Loss = 0.0018851
  - Remind you this is not the best answer. In fact, the graph plotted below shows we can achieve almost zero loss (in 10^(-29)) for various alpha and epoch values.

- Now we want to find the relationship between alpha and Number of epochs.
- The best way to do this is to plot the value of loss for various alpha and no of epochs.
- So, I developed a function named plot.py. The code for plot is as shown below.

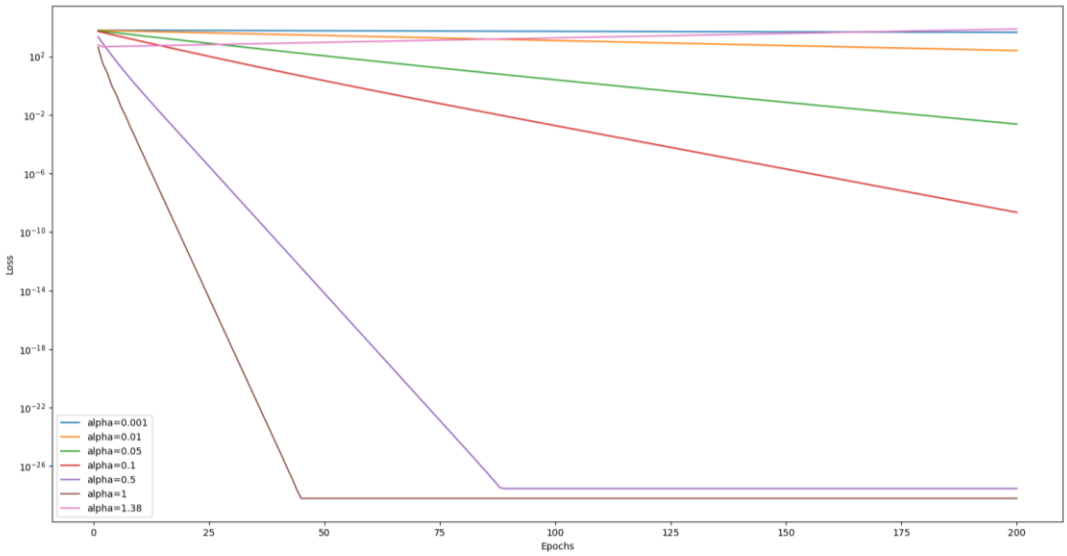
plot.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.core.function_base import linspace
from linear_regression import *
from sklearn.datasets import make_regression

n_samples = 200
X,y = make_regression(n_samples= n_samples, n_features=4, random_state=1)
y = np.asmatrix(y).T
X = np.asmatrix(X)
Xtrain, Ytrain, Xtest, Ytest = X[::2], y[::2], X[1::2], y[1::2]
alphaa=[0.005,0.01,0.05,0.1,0.5,1,1.38]
size=np.size(alphaa)
epoch=linspace(1,200,200)
for i in range(size):
    loss=np.zeros(np.size(epoch))
    for j in range(np.size(epoch)):
        n_samples = 200
        X,y = make_regression(n_samples= n_samples, n_features=4, random_state=1)
        y = np.asmatrix(y).T
        X = np.asmatrix(X)
        Xtrain, Ytrain, Xtest, Ytest = X[::2], y[::2], X[1::2], y[1::2]
        alpha=alphaa[i]
        n_epoch=epoch[j]
        w=train(Xtrain, Ytrain, alpha, int(epoch[j]))
        yhat_test=compute_yhat(Xtest,w)
        loss[j]=compute_L(yhat_test,Ytest)
    plt.plot(epoch,loss, label= 'alpha='+str(alphaa[i]))
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

```

- This code helps me to iterate through various alpha's and plot a graph between Loss and Epochs. Where x-axis is the no of epochs and y axis is the testing loss. Below is the graph and the interpretation:
- As you can see each color line represents the values of loss at no of epoch for a particular alpha.



- We can observe the following:
  - On increasing the alpha (Learning rate) the loss lowers down till alpha=1. Post that the model starts overshooting, this is shown for alpha =1.38 (pink line)
  - Now for a particular alpha, we can observe that on increasing epoch we see the loss decreasing and then becomes constant over no of epochs. Except for alpha>1, at this point the model keeps overshooting on increase of epochs.
  - As we can see from the graph, most have approached a loss of 0 by 100. But if we scale the Y- axis to "log" we get a clearer picture. The graph is shown below:
  - So below you can see at alpha=1 and epoch>=45 the loss is <=6.45x10^(-29). This can be confirmed by putting these values in application.py as well.



Hence, we understand on increasing alpha(alpha<=1) and no of epochs we get a lower loss.