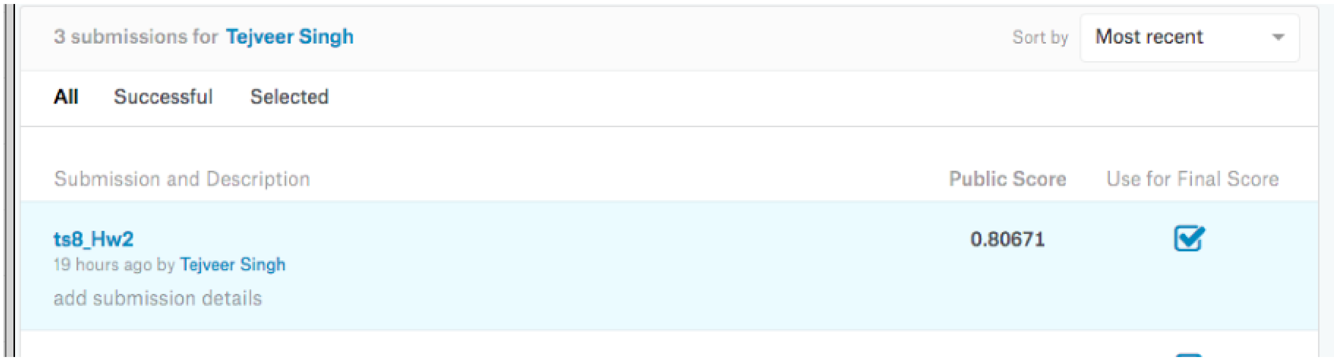
.

Page 1: screenshot of your leaderboard accuracy **and** mention your **best test dataset accuracy obtained on kaggle**.



3 submissions for Tejveer Singh

Sort by Most recent

Submission and Description	Public Score	Use for Final Score
ts8_Hw2 19 hours ago by Tejveer Singh add submission details	0.80671	☑

The accuracy is 0.80671

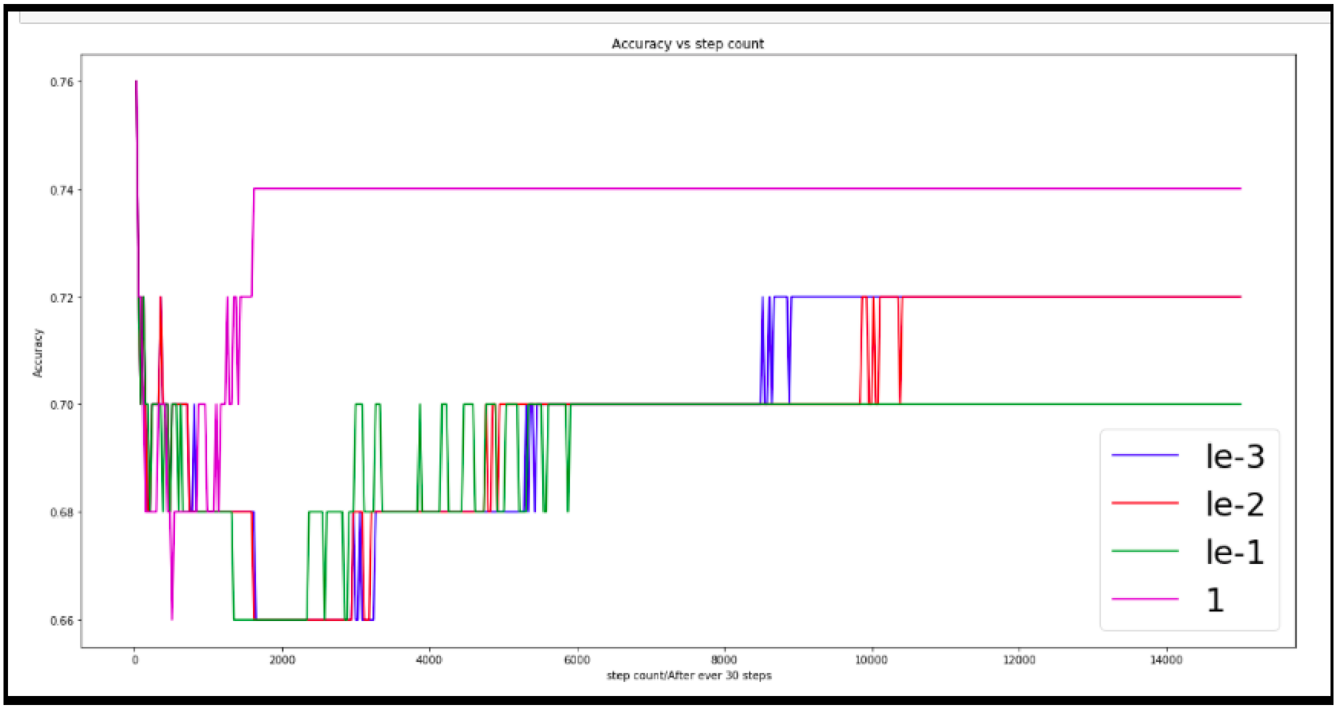
Page 2: A plot of the accuracy every 30 steps, for each value of the regularization constant. You should plot the curves for all regularization constants in the same plot using different colors with a label showing the corresponding values

Number of epochs=50

number of steps=300

X-axis – 15000 steps (calcuated accuracy after every 30 steps)

y-axis - Accuracy



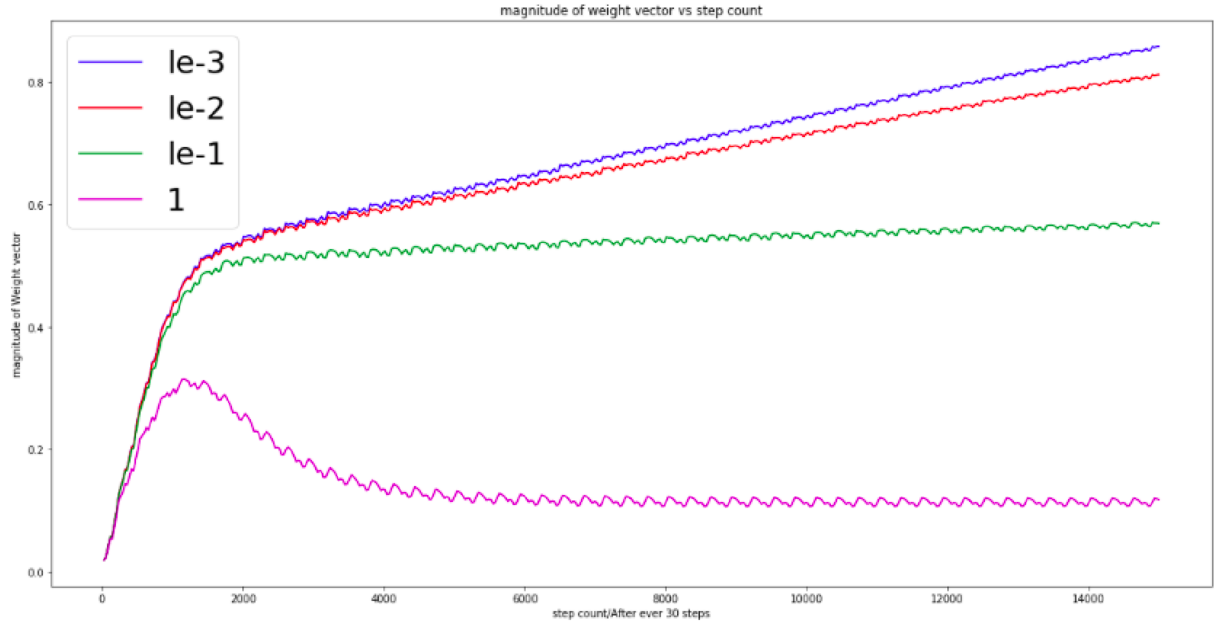
Page 3: A plot of the magnitude of the coefficient vector every 30 steps, for each value of the regularization constant. You should plot the curves for all regularization constants in the same plot using different colors with a label showing the corresponding values.

Number of epochs=50

number of steps=300

X-axis – 15000 steps (calcuated magnitude of weight after every 30 steps)

y-axis – Magnitude of weight vector



Page 4: Your estimate of the best value of the regularization constant, together with a brief description of why you believe that is a good value. What was your choice for the learning rate and why did you choose it ?

The best value of regularization constrant is (le-3) .This value is giving me best validation test accuracy .Then it is also giving me best test set accuracy on kaggle.

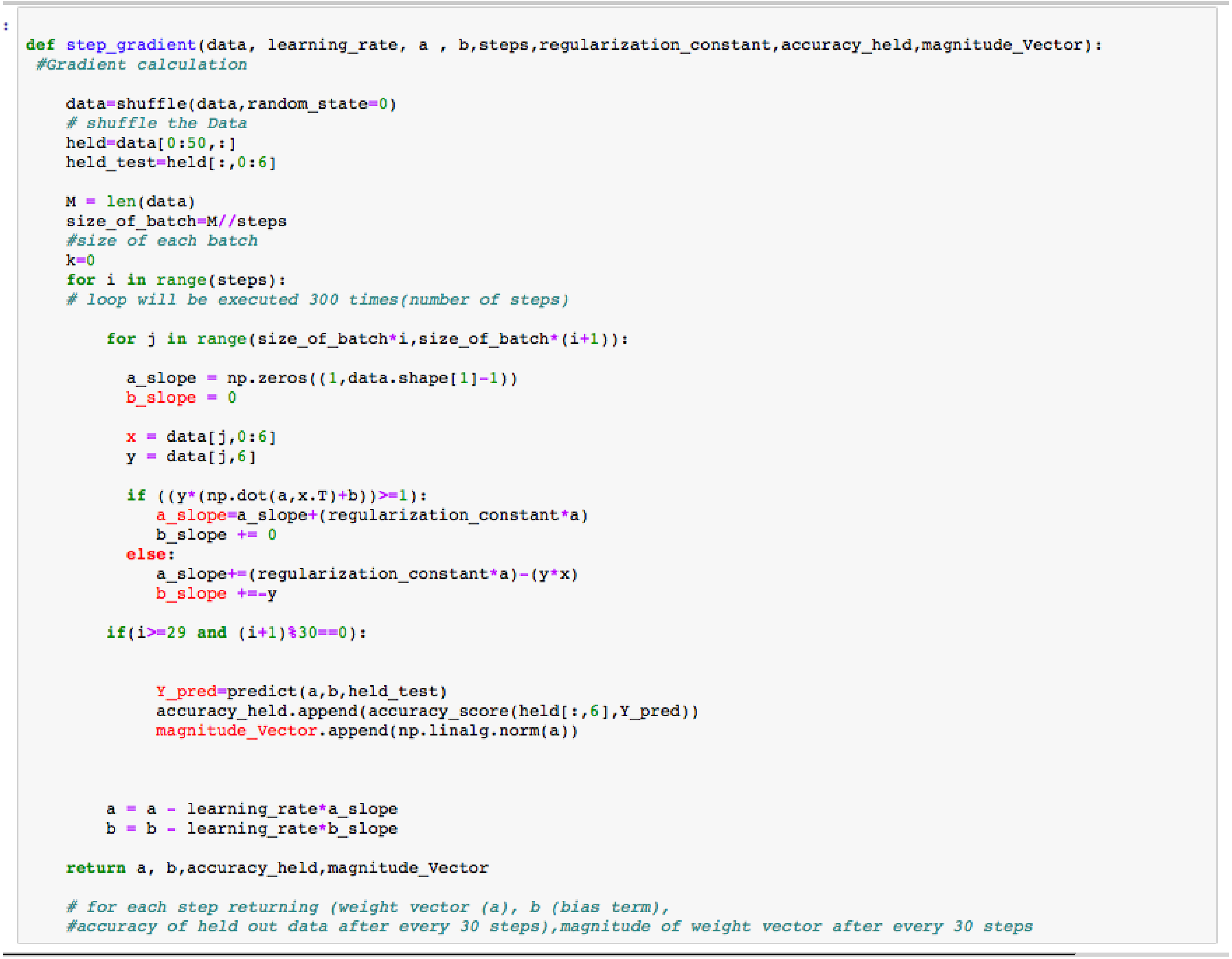
The choice of my learning rate was 0.01 in starting but then I reduced it to 0.001 and it converged to global minima quite well.

After each epoch I reduced the learning rate by 1%.After this I was able to achieve around 81% Accuracy on validation set and test set.

Page:5

Code Snippet:

The code calculates the gradient at every epoch and update the weight parameters.



```
:
def step_gradient(data, learning_rate, a , b,steps,regularization_constant,accuracy_held,magnitude_Vector):
 #Gradient calculation

    data=shuffle(data,random_state=0)
    # shuffle the Data
    held=data[0:50,:]
    held_test=held[:,0:6]

    M = len(data)
    size_of_batch=M//steps
    #size of each batch
    k=0
    for i in range(steps):
    # loop will be executed 300 times(number of steps)

        for j in range(size_of_batch*i,size_of_batch*(i+1)):

          a_slope = np.zeros((1,data.shape[1]-1))
          b_slope = 0

          x = data[j,0:6]
          y = data[j,6]

          if ((y*(np.dot(a,x.T)+b))>=1):
             a_slope=a_slope+(regularization_constant*a)
             b_slope += 0
          else:
             a_slope+=(regularization_constant*a)-(y*x)
             b_slope +=-y

        if(i>=29 and (i+1)%30==0):


             Y_pred=predict(a,b,held_test)
             accuracy_held.append(accuracy_score(held[:,6],Y_pred))
             magnitude_Vector.append(np.linalg.norm(a))



        a = a - learning_rate*a_slope
        b = b - learning_rate*b_slope

    return a, b,accuracy_held,magnitude_Vector

    # for each step returning (weight vector (a), b (bias term),
    #accuracy of held out data after every 30 steps),magnitude of weight vector after every 30 steps
```