Applied Machine Learning
Homework 8

Daniel Gonzalez
Colton Piper
31th of October, 2018

1 Results

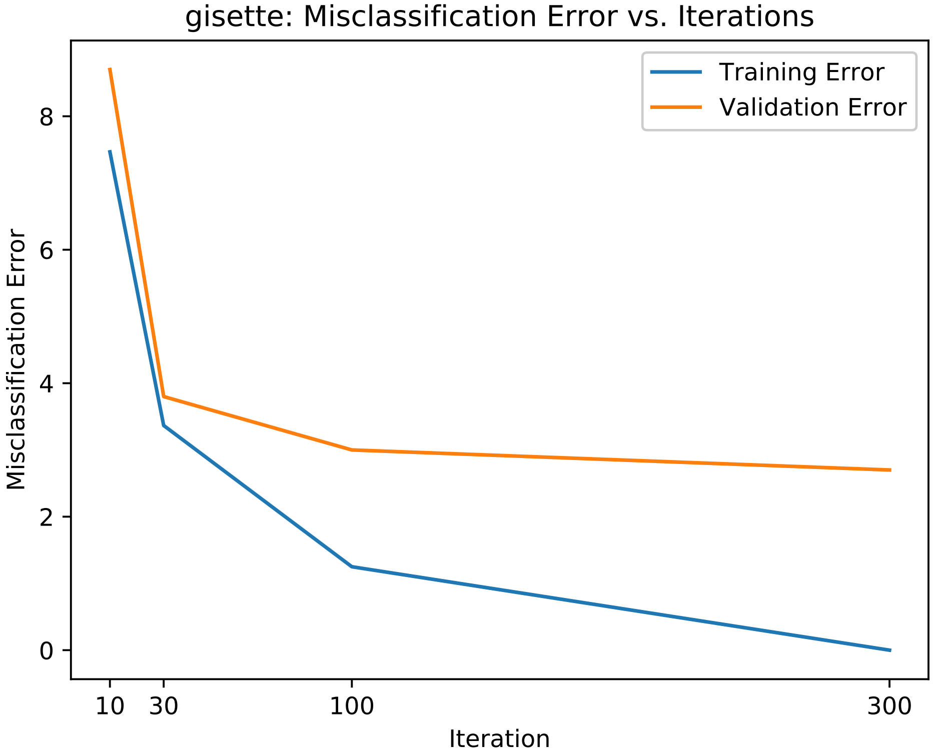
1.1 Table

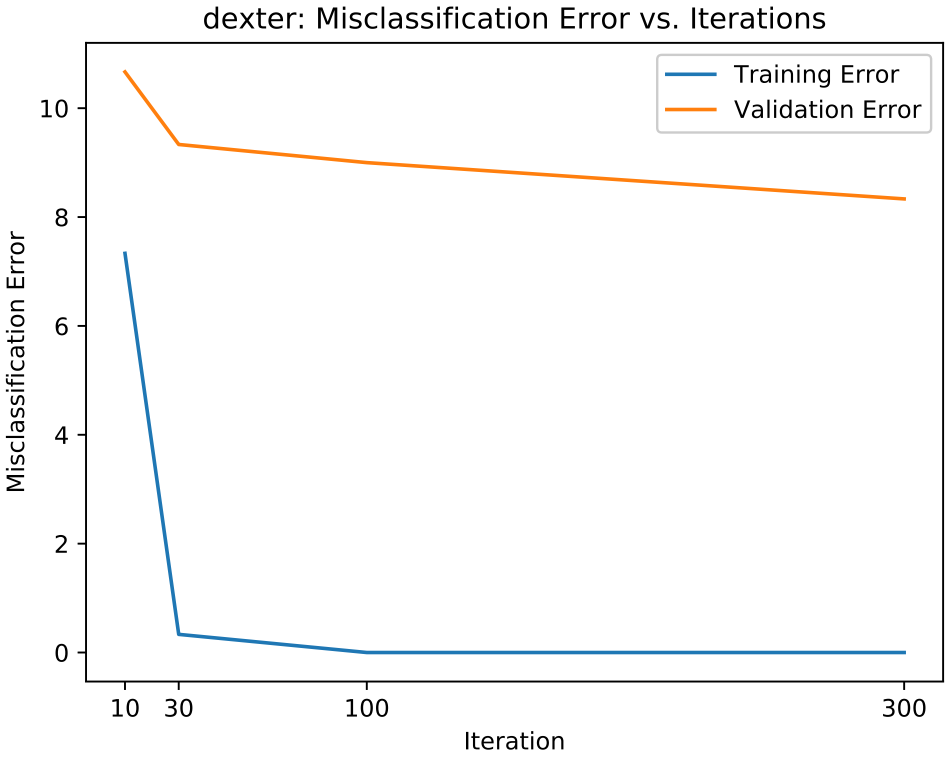
Table 1: Summary of Results: Original Learning Rates

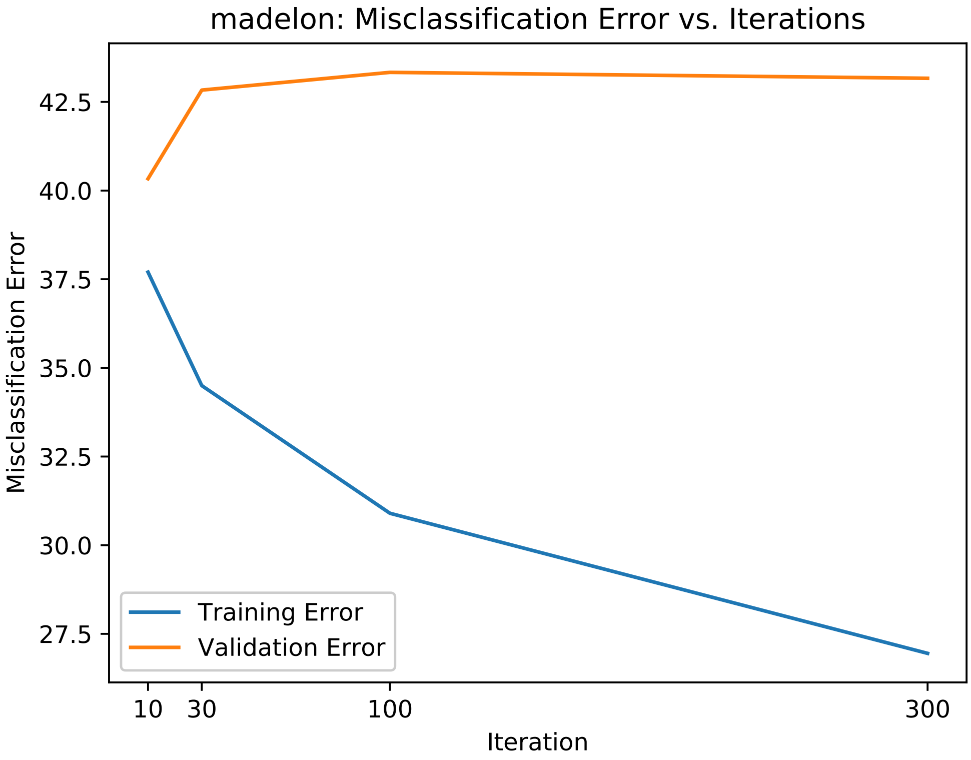
Data Set	Iteration k	Training Error %	Test Error %
`gisette`	10	7.467	8.7
	30	3.367	3.8
	100	1.25	3.0
	300	0.0	2.7
`dexter`	10	7.333	10.667
	30	0.333	9.333
	100	0.0	9.0
	300	0.0	8.333
`madelon`	10	37.7	40.333
	30	34.5	42.833
	100	30.9	43.333
	300	26.950	43.167

1.2 Figures

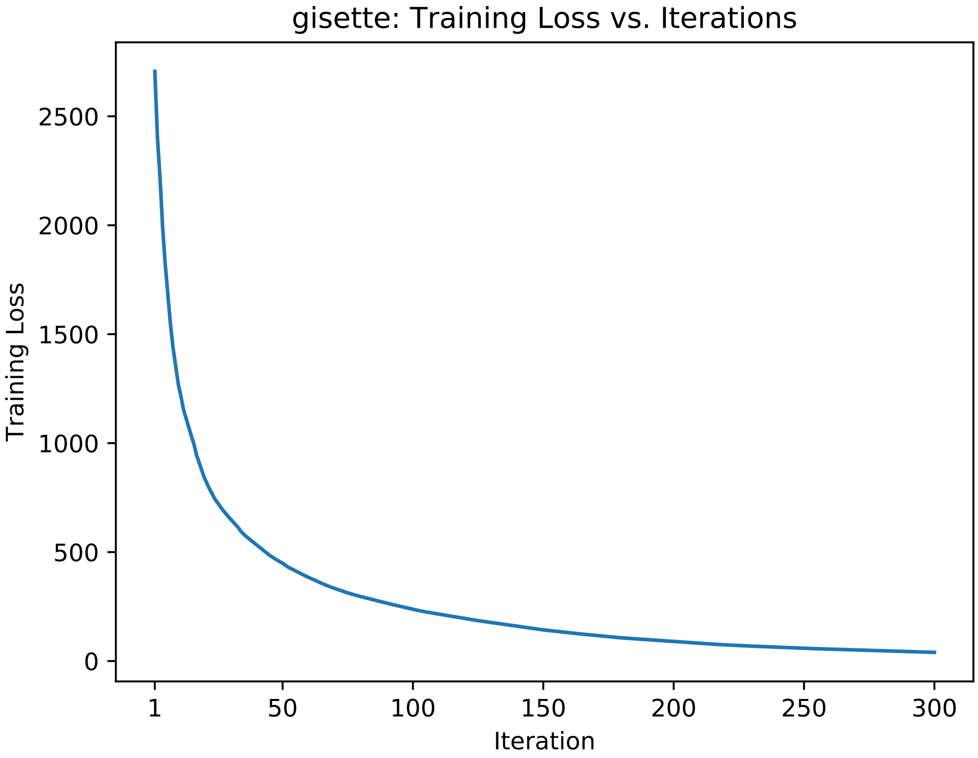
1.2.1 Misclassification Errors

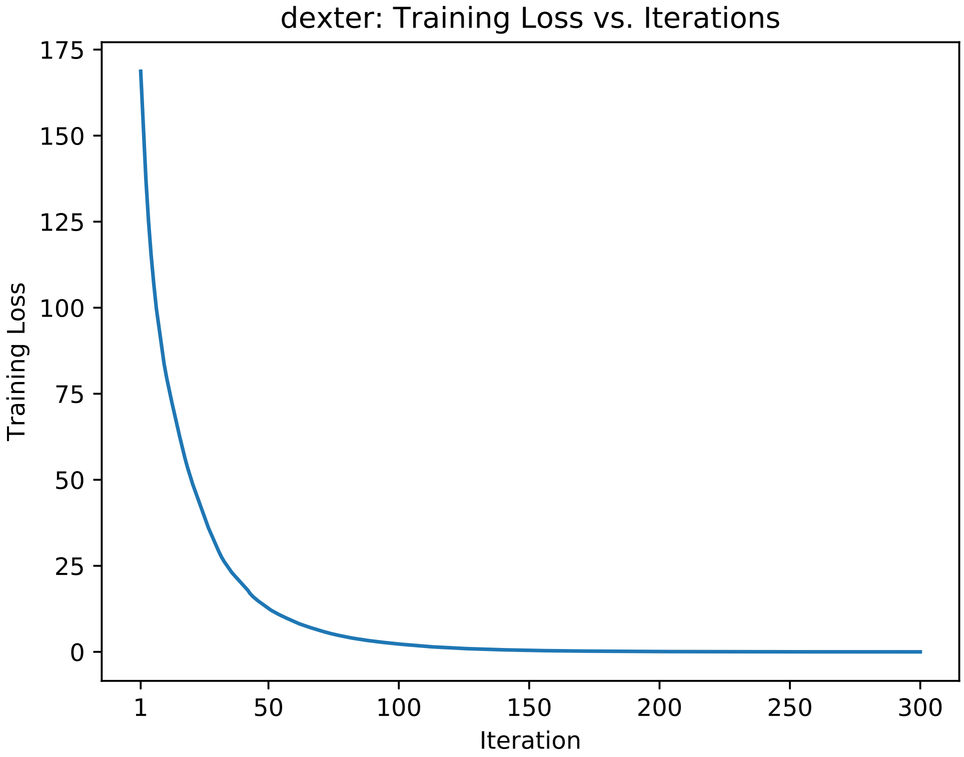


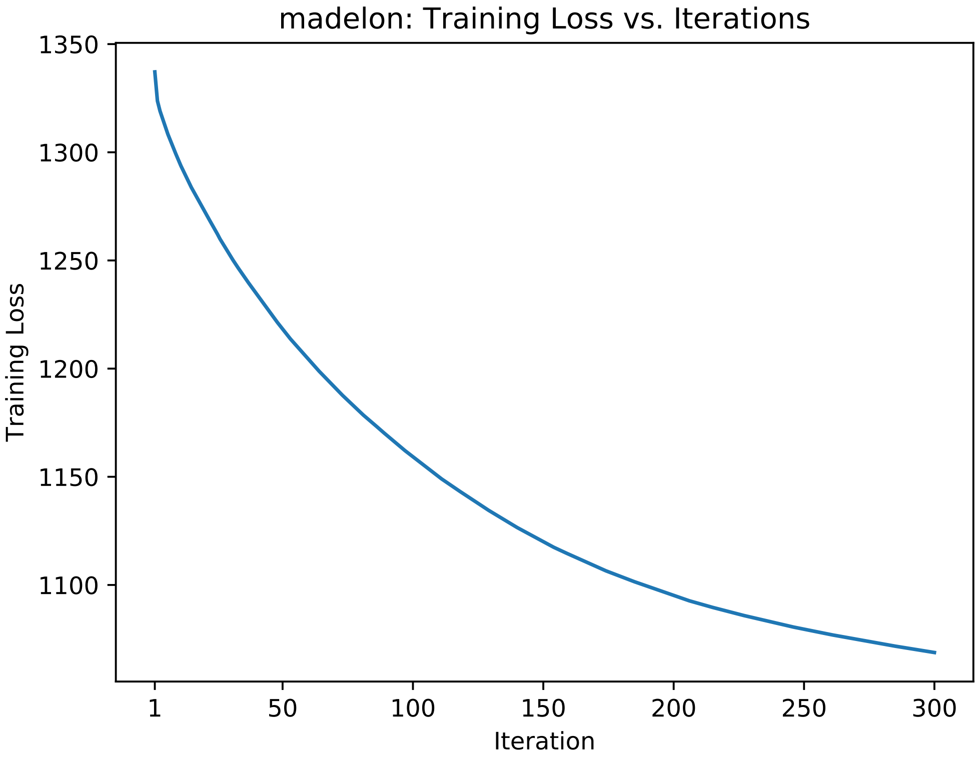

dexter: Misclassification Error vs. Iterations
Training Error
Validation Error
Misclassification Error
0
2
4
6
8
10
10
30
100
300
Iteration


madelon: Misclassification Error vs. Iterations
Training Error
Validation Error
Misclassification Error
27.5
30.0
32.5
35.0
37.5
40.0
42.5
10
30
100
300
Iteration

1.2.2 Training Losses

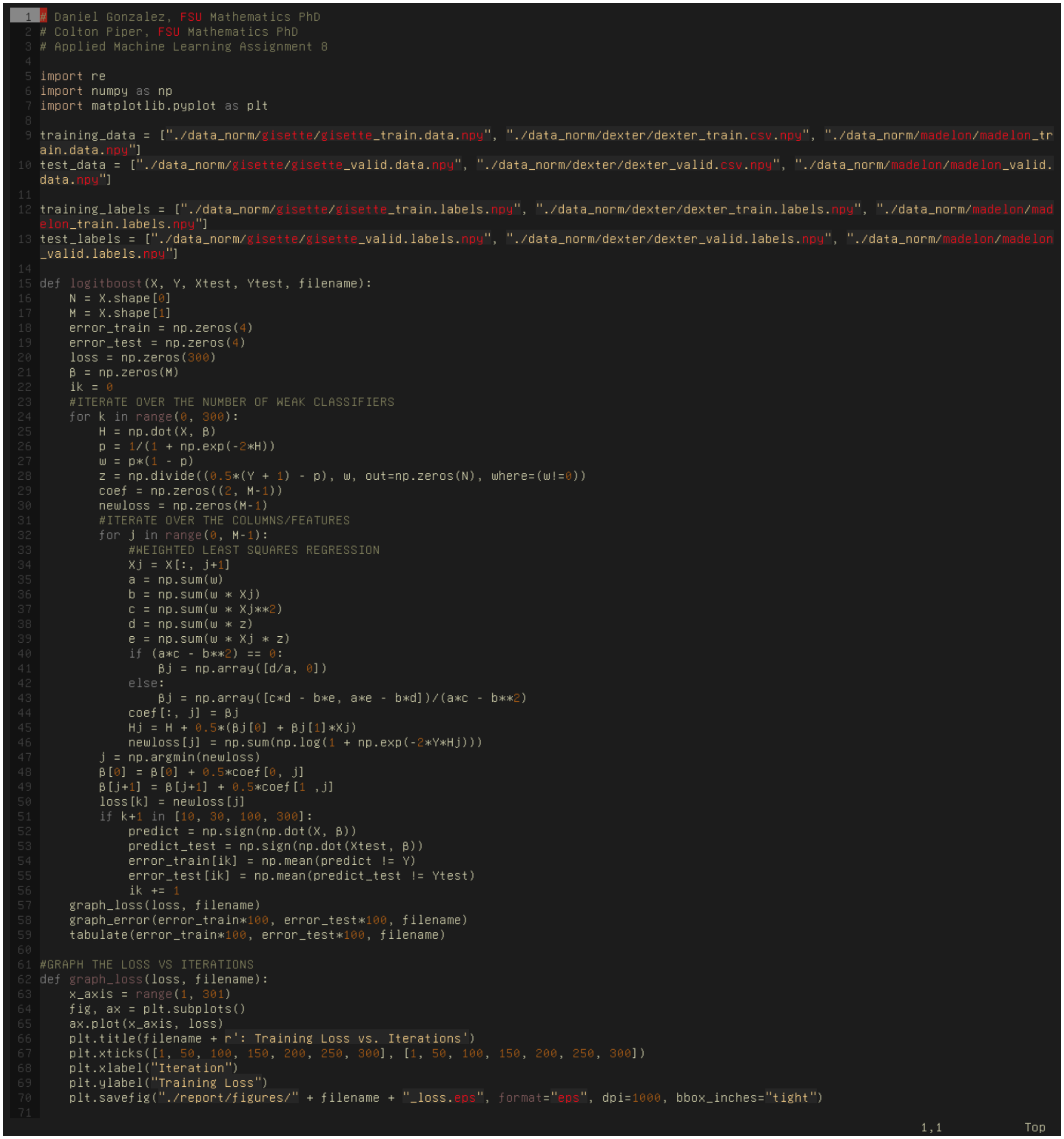




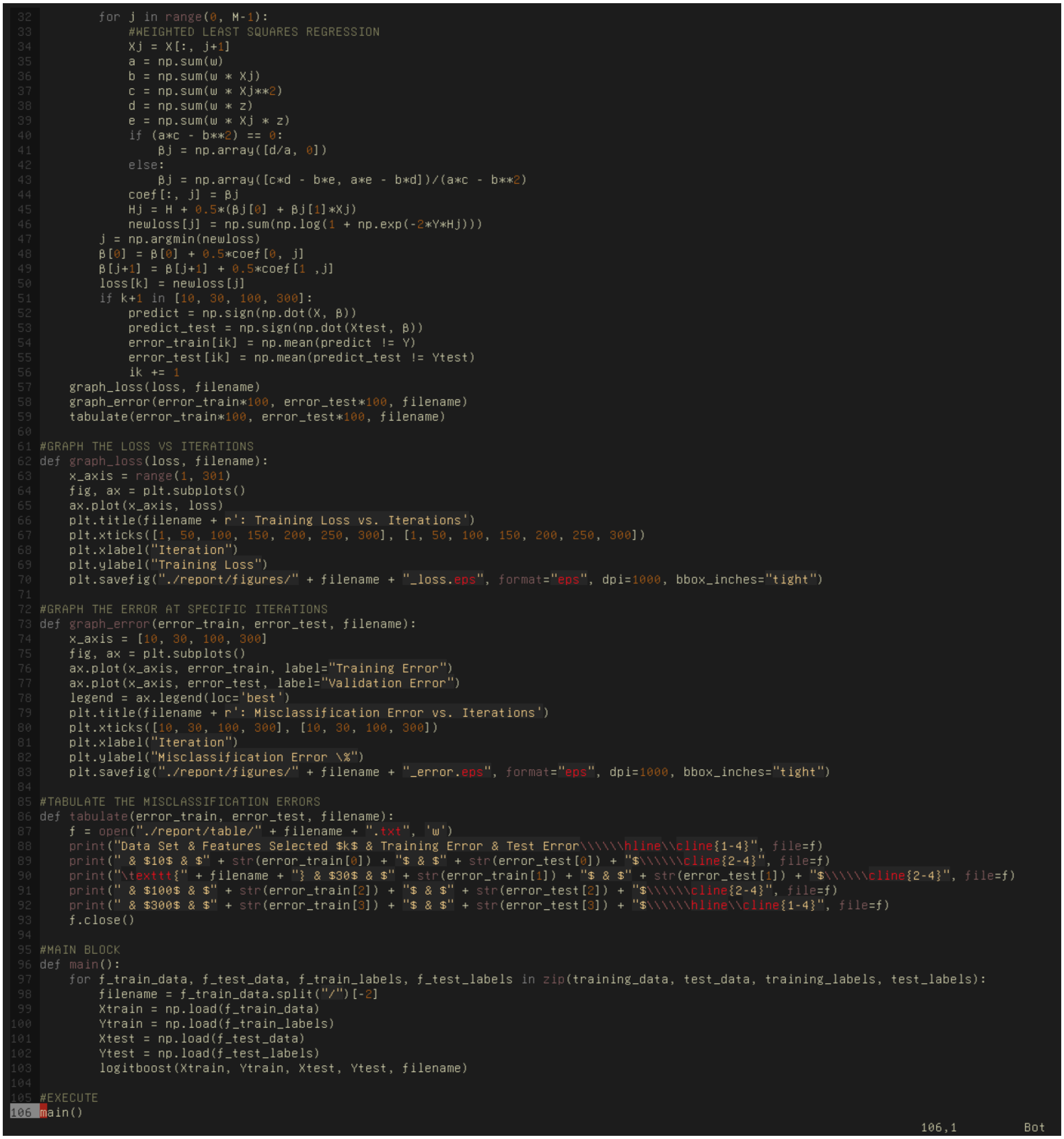

madelon: Training Loss vs. Iterations
Training Loss
1350
1300
1250
1200
1150
1100
Iteration
1
50
100
150
200
250
300

2 Appendix: Code

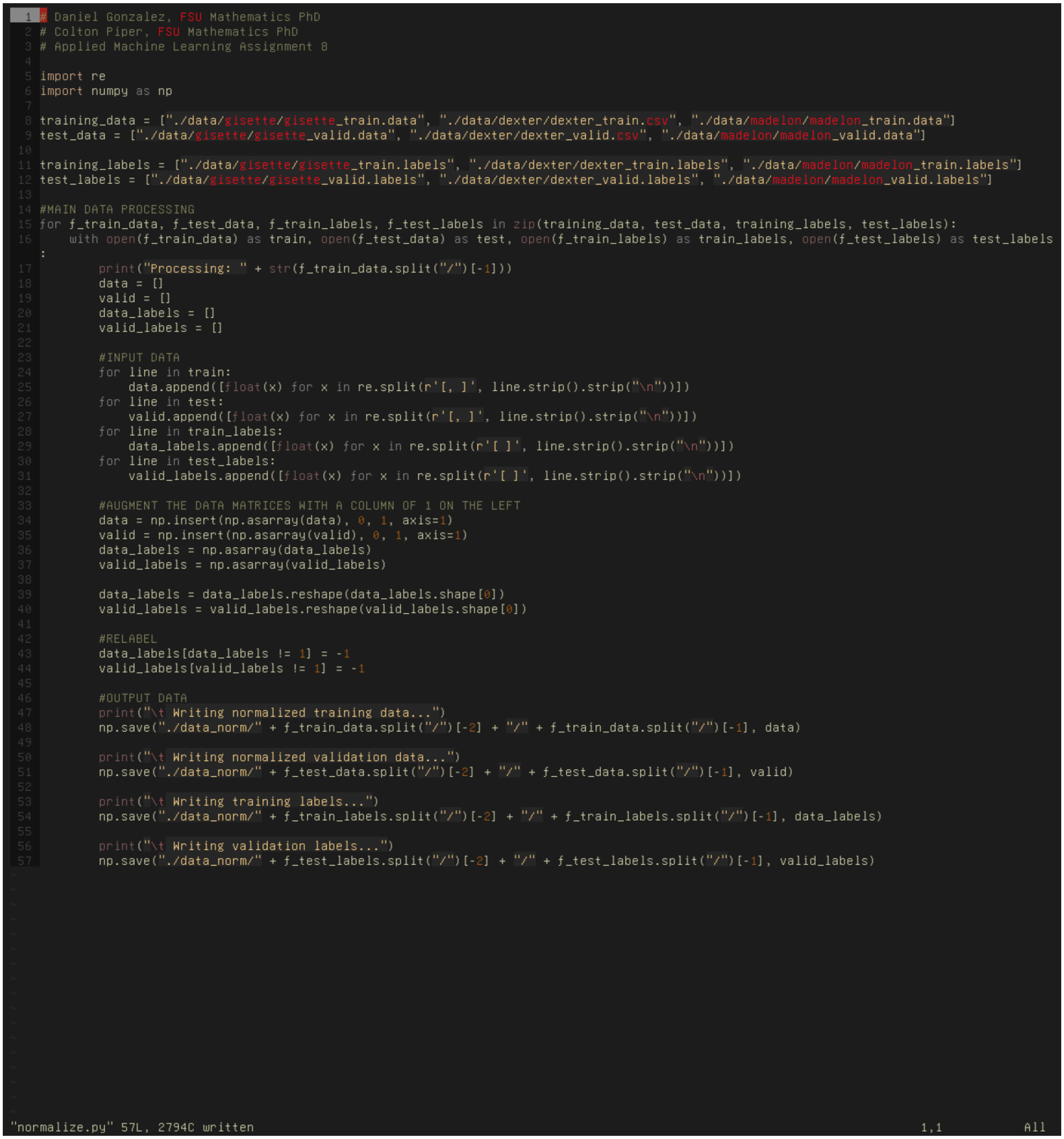
If the code looks too small, please zoom in on the pdf. The screenshots are `.png` images, so you should be able to zoom in and read at whatever is a comfortable size for you. The first two screenshots are of the actual FSA code, while the last screenshot is the python code for normalizing and pre-processing the data.



```python
# Daniel Gonzalez, FSU Mathematics PhD
# Colton Piper, FSU Mathematics PhD
# Applied Machine Learning Assignment 8

import re
import numpy as np
import matplotlib.pyplot as plt

training_data = ["./data_norm/gisette/gisette_train.data.npy", "./data_norm/dexter/dexter_train.csv.npy", "./data_norm/madelon/madelon_train.data.npy"]
test_data = ["./data_norm/gisette/gisette_valid.data.npy", "./data_norm/dexter/dexter_valid.csv.npy", "./data_norm/madelon/madelon_valid.data.npy"]

training_labels = ["./data_norm/gisette/gisette_train.labels.npy", "./data_norm/dexter/dexter_train.labels.npy", "./data_norm/madelon/madelon_train.labels.npy"]
test_labels = ["./data_norm/gisette/gisette_valid.labels.npy", "./data_norm/dexter/dexter_valid.labels.npy", "./data_norm/madelon/madelon_valid.labels.npy"]

def logitboost(X, Y, Xtest, Ytest, filename):
    N = X.shape[0]
    M = X.shape[1]
    error_train = np.zeros(4)
    error_test = np.zeros(4)
    loss = np.zeros(300)
    β = np.zeros(M)
    ik = 0
    #ITERATE OVER THE NUMBER OF WEAK CLASSIFIERS
    for k in range(0, 300):
        H = np.dot(X, β)
        p = 1/(1 + np.exp(-2*H))
        w = p*(1 - p)
        z = np.divide((0.5*(Y + 1) - p), w, out=np.zeros(N), where=(w!=0))
        coef = np.zeros((2, M-1))
        newloss = np.zeros(M-1)
        #ITERATE OVER THE COLUMNS/FEATURES
        for j in range(0, M-1):
            #WEIGHTED LEAST SQUARES REGRESSION
            Xj = X[:, j+1]
            a = np.sum(w)
            b = np.sum(w * Xj)
            c = np.sum(w * Xj**2)
            d = np.sum(w * z)
            e = np.sum(w * Xj * z)
            if (a*c - b**2) == 0:
                βj = np.array([d/a, 0])
            else:
                βj = np.array([c*d - b*e, a*e - b*d])/(a*c - b**2)
            coef[:, j] = βj
            Hj = H + 0.5*(βj[0] + βj[1]*Xj)
            newloss[j] = np.sum(np.log(1 + np.exp(-2*Y*Hj)))
        j = np.argmin(newloss)
        β[0] = β[0] + 0.5*coef[0, j]
        β[j+1] = β[j+1] + 0.5*coef[1 ,j]
        loss[k] = newloss[j]
        if k+1 in [10, 30, 100, 300]:
            predict = np.sign(np.dot(X, β))
            predict_test = np.sign(np.dot(Xtest, β))
            error_train[ik] = np.mean(predict != Y)
            error_test[ik] = np.mean(predict_test != Ytest)
            ik += 1
    graph_loss(loss, filename)
    graph_error(error_train*100, error_test*100, filename)
    tabulate(error_train*100, error_test*100, filename)

#GRAPH THE LOSS VS ITERATIONS
def graph_loss(loss, filename):
    x_axis = range(1, 301)
    fig, ax = plt.subplots()
    ax.plot(x_axis, loss)
    plt.title(filename + r': Training Loss vs. Iterations')
    plt.xticks([1, 50, 100, 150, 200, 250, 300], [1, 50, 100, 150, 200, 250, 300])
    plt.xlabel("Iteration")
    plt.ylabel("Training Loss")
    plt.savefig("./report/figures/" + filename + "_loss.eps", format="eps", dpi=1000, bbox_inches="tight")
```



```
        for j in range(0, M-1):
            #WEIGHTED LEAST SQUARES REGRESSION
            Xj = X[:, j+1]
            a = np.sum(w)
            b = np.sum(w * Xj)
            c = np.sum(w * Xj**2)
            d = np.sum(w * z)
            e = np.sum(w * Xj * z)
            if (a*c - b**2) == 0:
                βj = np.array([d/a, 0])
            else:
                βj = np.array([c*d - b*e, a*e - b*d])/(a*c - b**2)
            coef[:, j] = βj
            Hj = H + 0.5*(βj[0] + βj[1]*Xj)
            newloss[j] = np.sum(np.log(1 + np.exp(-2*Y*Hj)))
        j = np.argmin(newloss)
        β[0] = β[0] + 0.5*coef[0, j]
        β[j+1] = β[j+1] + 0.5*coef[1 ,j]
        loss[k] = newloss[j]
        if k+1 in [10, 30, 100, 300]:
            predict = np.sign(np.dot(X, β))
            predict_test = np.sign(np.dot(Xtest, β))
            error_train[ik] = np.mean(predict != Y)
            error_test[ik] = np.mean(predict_test != Ytest)
            ik += 1
    graph_loss(loss, filename)
    graph_error(error_train*100, error_test*100, filename)
    tabulate(error_train*100, error_test*100, filename)

#GRAPH THE LOSS VS ITERATIONS
def graph_loss(loss, filename):
    x_axis = range(1, 301)
    fig, ax = plt.subplots()
    ax.plot(x_axis, loss)
    plt.title(filename + r': Training Loss vs. Iterations')
    plt.xticks([1, 50, 100, 150, 200, 250, 300], [1, 50, 100, 150, 200, 250, 300])
    plt.xlabel("Iteration")
    plt.ylabel("Training Loss")
    plt.savefig("./report/figures/" + filename + "_loss.eps", format="eps", dpi=1000, bbox_inches="tight")

#GRAPH THE ERROR AT SPECIFIC ITERATIONS
def graph_error(error_train, error_test, filename):
    x_axis = [10, 30, 100, 300]
    fig, ax = plt.subplots()
    ax.plot(x_axis, error_train, label="Training Error")
    ax.plot(x_axis, error_test, label="Validation Error")
    legend = ax.legend(loc='best')
    plt.title(filename + r': Misclassification Error vs. Iterations')
    plt.xticks([10, 30, 100, 300], [10, 30, 100, 300])
    plt.xlabel("Iteration")
    plt.ylabel("Misclassification Error \%")
    plt.savefig("./report/figures/" + filename + "_error.eps", format="eps", dpi=1000, bbox_inches="tight")

#TABULATE THE MISCLASSIFICATION ERRORS
def tabulate(error_train, error_test, filename):
    f = open("./report/table/" + filename + ".txt", 'w')
    print("Data Set & Features Selected $k$ & Training Error & Test Error\\\\\\hline\\cline{1-4}", file=f)
    print(" & $10$ & $" + str(error_train[0]) + "$ & $" + str(error_test[0]) + "$\\\\\\cline{2-4}", file=f)
    print("\texttt{" + filename + "} & $30$ & $" + str(error_train[1]) + "$ & $" + str(error_test[1]) + "$\\\\\\cline{2-4}", file=f)
    print(" & $100$ & $" + str(error_train[2]) + "$ & $" + str(error_test[2]) + "$\\\\\\cline{2-4}", file=f)
    print(" & $300$ & $" + str(error_train[3]) + "$ & $" + str(error_test[3]) + "$\\\\\\hline\\cline{1-4}", file=f)
    f.close()

#MAIN BLOCK
def main():
    for f_train_data, f_test_data, f_train_labels, f_test_labels in zip(training_data, test_data, training_labels, test_labels):
        filename = f_train_data.split("/")[-2]
        Xtrain = np.load(f_train_data)
        Ytrain = np.load(f_train_labels)
        Xtest = np.load(f_test_data)
        Ytest = np.load(f_test_labels)
        logitboost(Xtrain, Ytrain, Xtest, Ytest, filename)

#EXECUTE
main()
```



```
# Daniel Gonzalez, FSU Mathematics PhD
# Colton Piper, FSU Mathematics PhD
# Applied Machine Learning Assignment 8

import re
import numpy as np

training_data = ["./data/gisette/gisette_train.data", "./data/dexter/dexter_train.csv", "./data/madelon/madelon_train.data"]
test_data = ["./data/gisette/gisette_valid.data", "./data/dexter/dexter_valid.csv", "./data/madelon/madelon_valid.data"]

training_labels = ["./data/gisette/gisette_train.labels", "./data/dexter/dexter_train.labels", "./data/madelon/madelon_train.labels"]
test_labels = ["./data/gisette/gisette_valid.labels", "./data/dexter/dexter_valid.labels", "./data/madelon/madelon_valid.labels"]

#MAIN DATA PROCESSING
for f_train_data, f_test_data, f_train_labels, f_test_labels in zip(training_data, test_data, training_labels, test_labels):
    with open(f_train_data) as train, open(f_test_data) as test, open(f_train_labels) as train_labels, open(f_test_labels) as test_labels:
        print("Processing: " + str(f_train_data.split("/")[-1]))
        data = []
        valid = []
        data_labels = []
        valid_labels = []

        #INPUT DATA
        for line in train:
            data.append([float(x) for x in re.split(r'[, ]', line.strip().strip("\n"))])
        for line in test:
            valid.append([float(x) for x in re.split(r'[, ]', line.strip().strip("\n"))])
        for line in train_labels:
            data_labels.append([float(x) for x in re.split(r'[ ]', line.strip().strip("\n"))])
        for line in test_labels:
            valid_labels.append([float(x) for x in re.split(r'[ ]', line.strip().strip("\n"))])

        #AUGMENT THE DATA MATRICES WITH A COLUMN OF 1 ON THE LEFT
        data = np.insert(np.asarray(data), 0, 1, axis=1)
        valid = np.insert(np.asarray(valid), 0, 1, axis=1)
        data_labels = np.asarray(data_labels)
        valid_labels = np.asarray(valid_labels)

        data_labels = data_labels.reshape(data_labels.shape[0])
        valid_labels = valid_labels.reshape(valid_labels.shape[0])

        #RELABEL
        data_labels[data_labels != 1] = -1
        valid_labels[valid_labels != 1] = -1

        #OUTPUT DATA
        print("\t Writing normalized training data...")
        np.save("./data_norm/" + f_train_data.split("/")[-2] + "/" + f_train_data.split("/")[-1], data)

        print("\t Writing normalized validation data...")
        np.save("./data_norm/" + f_test_data.split("/")[-2] + "/" + f_test_data.split("/")[-1], valid)

        print("\t Writing training labels...")
        np.save("./data_norm/" + f_train_labels.split("/")[-2] + "/" + f_train_labels.split("/")[-1], data_labels)

        print("\t Writing validation labels...")
        np.save("./data_norm/" + f_test_labels.split("/")[-2] + "/" + f_test_labels.split("/")[-1], valid_labels)
```

```
"normalize.py" 57L, 2794C written                                              1,1           All
```