Project 1. Machine Learning (STA 5365). Aditya Ranjan Bhattacharya (arb17b) and Matthew Laird (mrl14b)

For Project 1 we used DecisionTreeClassifier model from sklearn[1] as our decision tree model. In this case, we used 2 separate functions data_to_numpy() and label_to_numpy(), which loads the features and labels from the files to a numpy array.

In order to get all the datasets running at the same time, we put the filenames of the training features, training labels, testing features and testing labels in 4 separate lists. The list with the features for both training and testing also contains the number of features for the dataset as the 2nd entry.

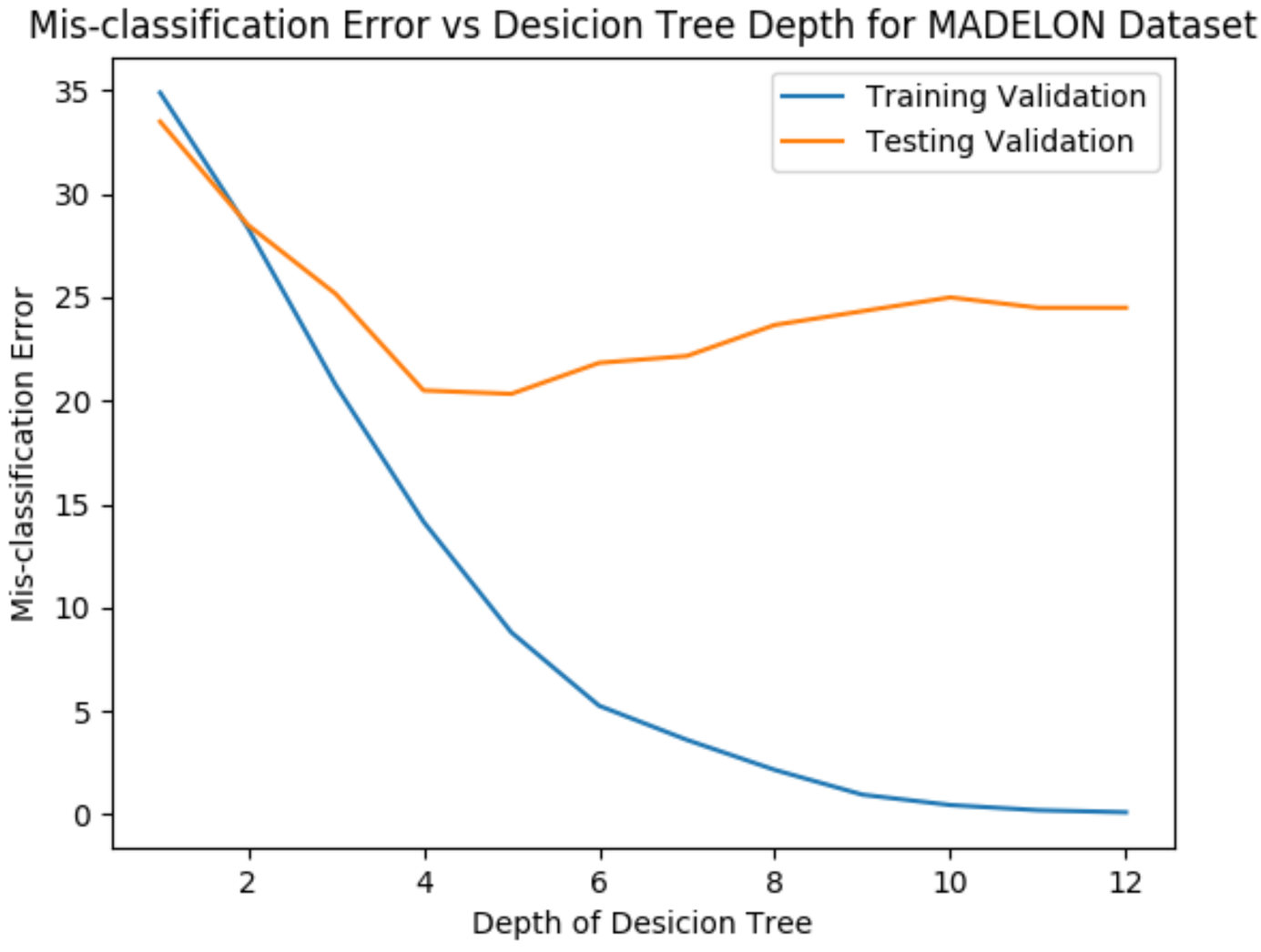
Once the datasets are loaded, the model is trained on the training set of the data, and then validation checks using both training and testing set are done. This happens iteratively, as the model goes from having depth = 1 to depth = 12. These are then saved in 2 separate numpy arrays, one for the training validation, and the other for testing validation.

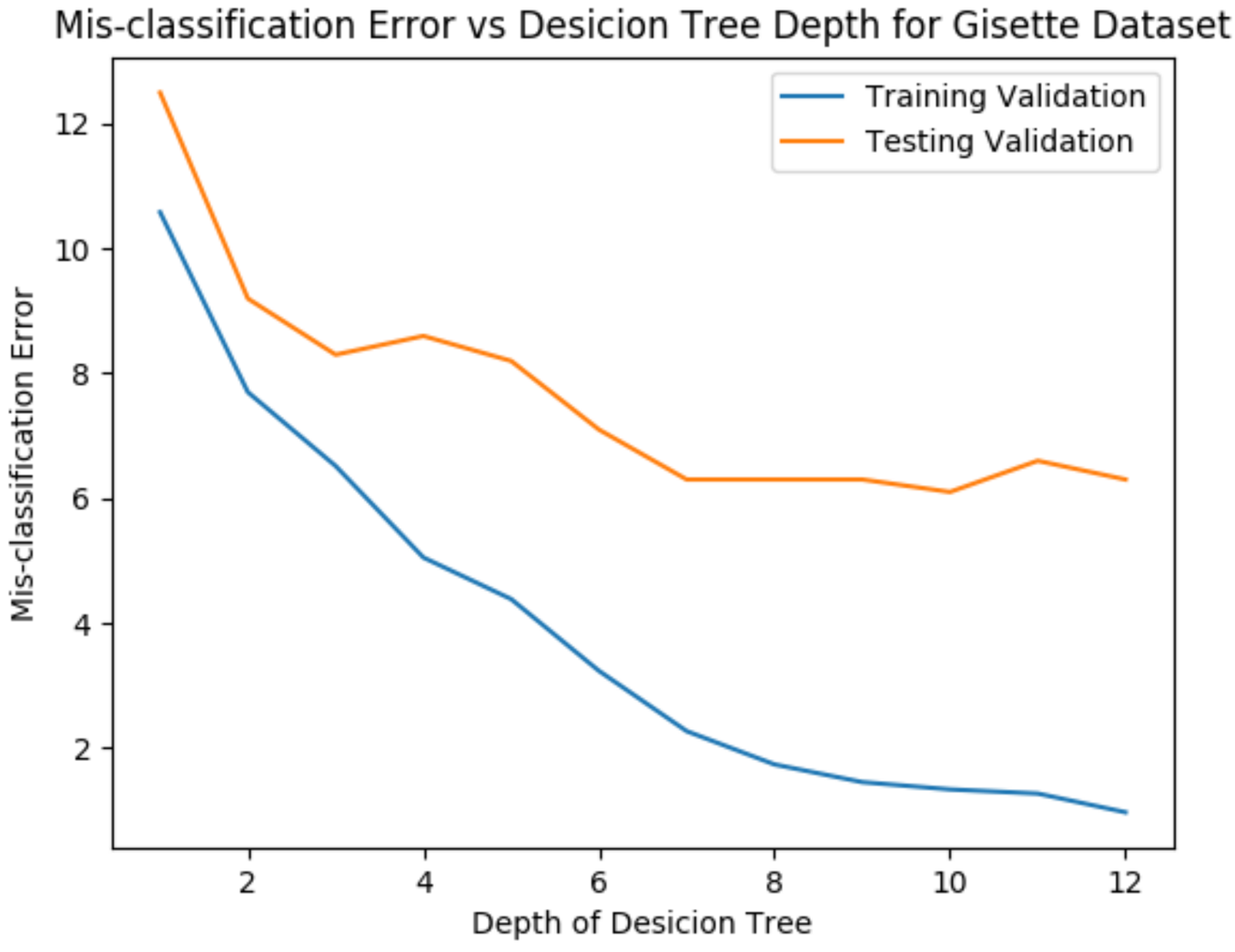
Once all 12 models' accuracy parameters are saved in the arrays, we plot them first, and then find the index of the testing validation with the lowest value of mis-classification error. Now as array indexing starts from 0, but the minimum depth of the decision tree model is 1. Thus decision tree model with depth = 1 has its mis-classification error saved in scores_test[0], and in general depth = x has its mis-classification error on testing set saved in scores_test[x-1].

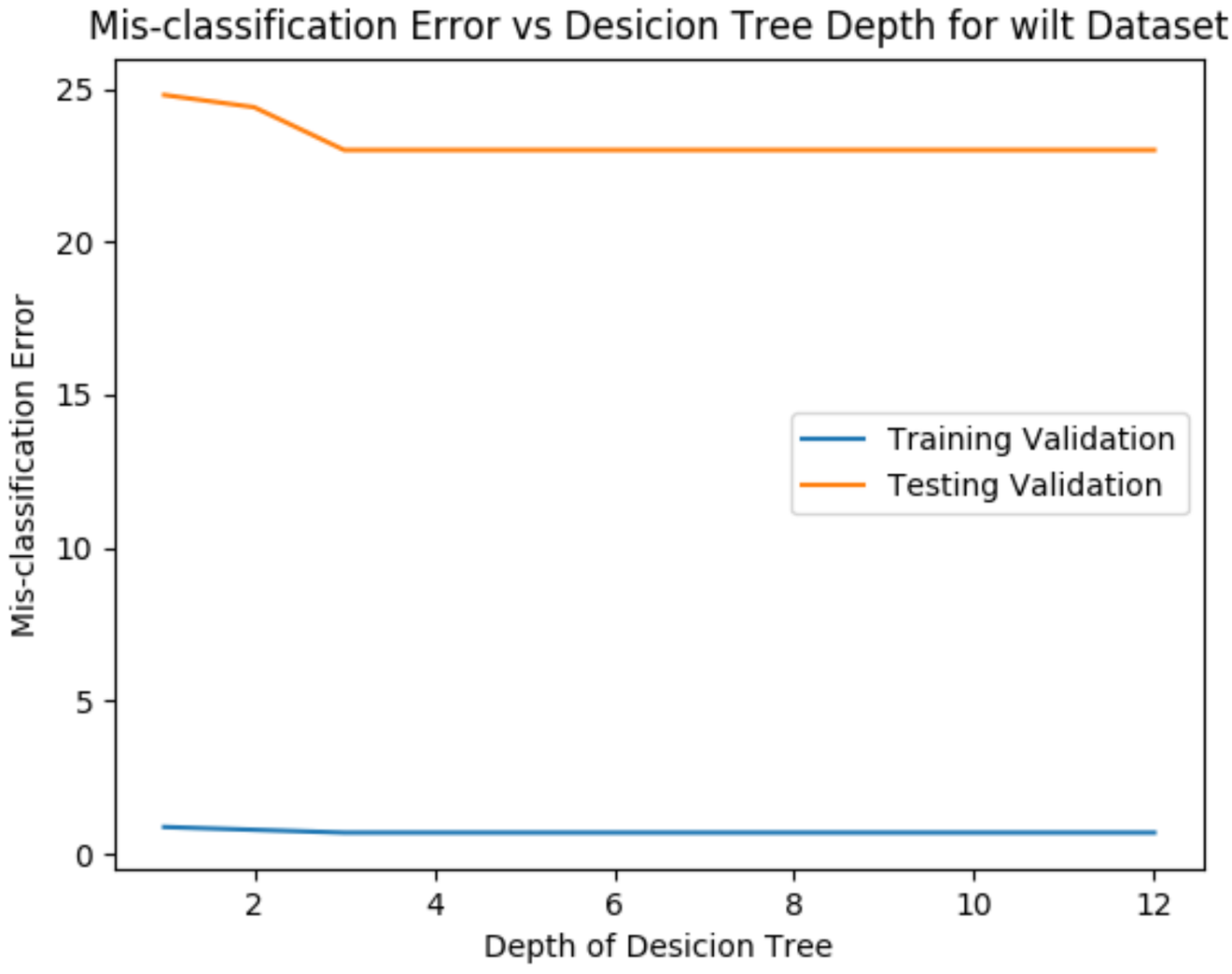
The results are given below.

Min. Test Classification Error and Related Tree Depth For Datasets

Dataset	Depth of Decision Tree with Minimum Test Classification Error	Value of Minimum Test Classification Error (%)
Madelon	5	20.33
Gisette	10	6.10
Wilt	3	23.0


Mis-classification Error vs Desicion Tree Depth for MADELON Dataset
Training Validation
Testing Validation
Mis-classification Error
Depth of Desicion Tree
35
30
25
20
15
10
5
0
2
4
6
8
10
12


Mis-classification Error vs Desicion Tree Depth for Gisette Dataset
Training Validation
Testing Validation
Mis-classification Error
Depth of Desicion Tree
2
4
6
8
10
12


Mis-classification Error vs Desicion Tree Depth for wilt Dataset
Mis-classification Error
Depth of Desicion Tree
Training Validation
Testing Validation
0
5
10
15
20
25
2
4
6
8
10
12

```
from sklearn.tree import DecisionTreeClassifier
import numpy
import matplotlib.pyplot as plt
from numpy import genfromtxt

#Features Extracted From .data and .csv files.
def data_to_numpy(file, num_features):

	if "wilt" in file:
		#wilt is in .csv, hence we use numpy.genfromtxt which loads csv to a numpy array
faster and easily.
		X = genfromtxt(file, delimiter=',')
		print(X.shape)
		return X

	else:

		X = numpy.empty([0,num_features])

		with open(file, 'r') as f:
			content = f.readlines()

			for line in content:
				row = numpy.empty(0)

				for x in line.strip("\n,\'").split():
					row = numpy.append(row, float(x))

				X = numpy.append(X, [row], 0)
			print(X.shape)

		return X
#Labels Extracted From .label files.
def label_to_numpy(file):

	y = numpy.empty(0)

	with open(file, 'r') as f:
		content = f.readlines()

		for line in content:
			y = numpy.append(y, int(line))

		print(y.shape)

	return y

datasets_training_features = [["MADELON/madelon_train.data", 500],
                              ["Gisette/gisette_train.data", 5000],
                              ["wilt/wilt_train.csv", 6]]


datasets_test_features = [["MADELON/madelon_valid.data", 500],
                          ["Gisette/gisette_valid.data", 5000],
                          ["wilt/wilt_test.csv", 6]]

datasets_training_labels = ["MADELON/madelon_train.labels",
                            "Gisette/gisette_train.labels",
                            "wilt/wilt_train.labels"]

datasets_test_labels = ["MADELON/madelon_valid.labels",
                        "Gisette/gisette_valid.labels",
                        "wilt/wilt_test.labels"]

for i in range(0,len(datasets_test_labels)):
#Get the Training and Validation Sets
	X_train = data_to_numpy(datasets_training_features[i][0], datasets_training_features[i][1])
	y_train = label_to_numpy(datasets_training_labels[i])

	X_test = data_to_numpy(datasets_test_features[i][0], datasets_test_features[i][1])
	y_test = label_to_numpy(datasets_test_labels[i])


	#Initialise empty arrays to store the mis-classification errors on training and testing sets
	scores_train = numpy.empty(0)
	scores_test = numpy.empty(0)

	#The depths of DT are all taken in a list of numbers as given below:
	depth_list = [1,2,3,4,5,6,7,8,9,10,11,12]

	for depth in depth_list:
		#We used DecisionTreeClassifier from sklearn.
		model = DecisionTreeClassifier(max_depth=depth, random_state=0,
min_samples_leaf=1, max_leaf_nodes=datasets_test_features[i][1], presort=True)
		model.fit(X_train, y_train)

		#Gives the accuracy of the model on the training/testing set (above/below). The
miscalculation error is computed as (1-accuracy)*100.00%
		training_scores = model.score(X_train, y_train)
		test_scores = model.score(X_test, y_test)

		#Error for DT with depth 1 will be at index 0. scores_train[0] gives efficiency of DT
with depth 1. Also, remember scores is accuracy
		scores_train = numpy.append(scores_train, (1 - training_scores)*100.00)
		scores_test = numpy.append(scores_test, (1 - test_scores)*100.00)

	print(“Scores of Each Depth: ”, scores_test)

	minimum_depth = numpy.argmin(scores_test)

	#Because of the comment above, we add 1 to the minimum_depth, as the latter is the
INDEX of the numpy array with lowest value (See numpy.argmin on google)
	print("Minimum test classification error was achieved on tree with depth: ",
minimum_depth+1, " and the minimum test classification error was: ", scores_test[minimum_depth])

	#Plotting the graph
	fig = plt.figure()
	plt.plot(depth_list, scores_train, label='Training Validation')
	plt.plot(depth_list, scores_test, label='Testing Validation')
	plt.xlabel('Depth of Desicion Tree')
	plt.ylabel('Mis-classification Error')
	plt.title("Mis-classification Error vs Desicion Tree Depth for " +
str(datasets_test_labels[i].split('/')[0]) + " Dataset")
	plt.legend()
	fig.savefig("hw1_"+str(datasets_test_labels[i].split('/')[0])+".png")
```