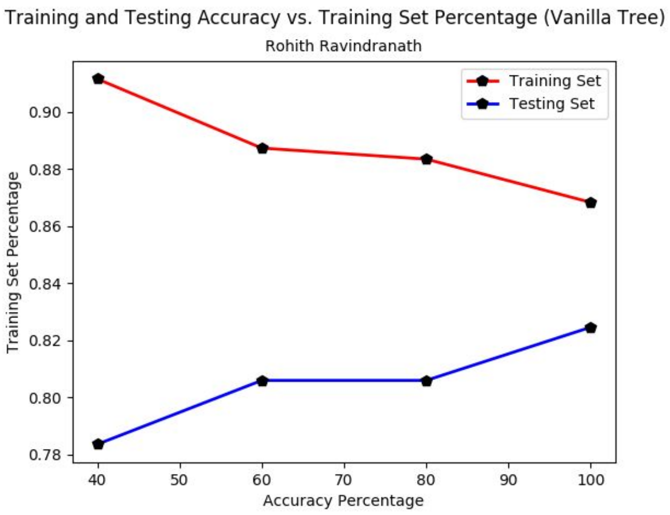
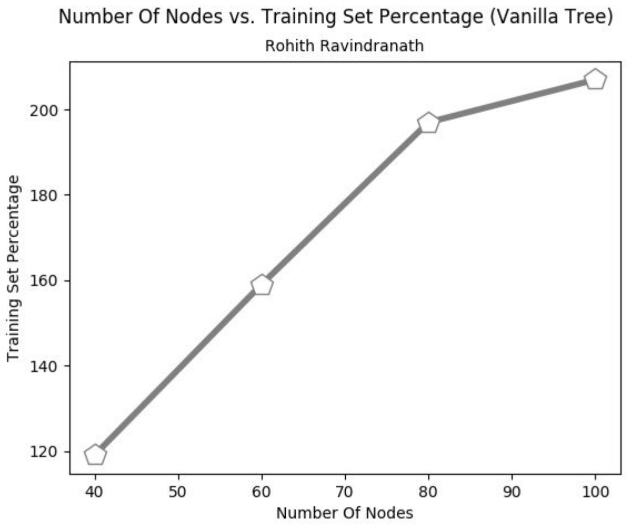
Rohith Ravindranath
PUID: 0028822977
Dan Goldwasser
CS 37300
3rd March 2018

Homework #2

I collaborated with Vishal Vasan, Meera Haridasa, Jullian Haresco, and Rohan Saxena. I affirm that I wrote the solutions in my own words and that I understand the solutions I am submitting.

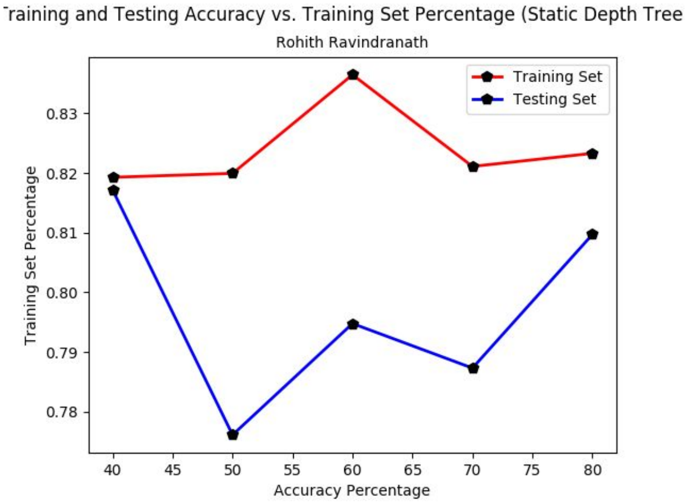
1. Vanilla Tree Plots



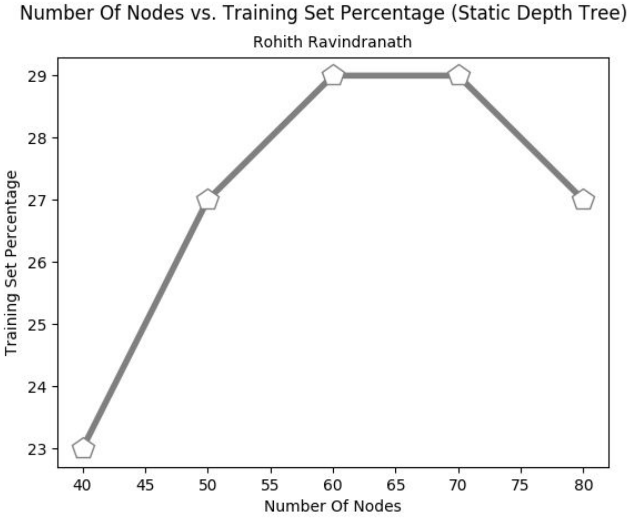


Question 1 Code Snippet

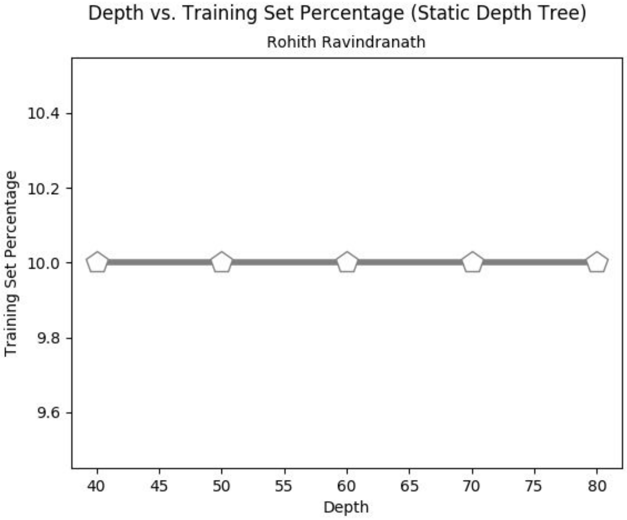
2. Depth Tree Plots


Training and Testing Accuracy vs. Training Set Percentage (Static Depth Tree

Rohith Ravindranath

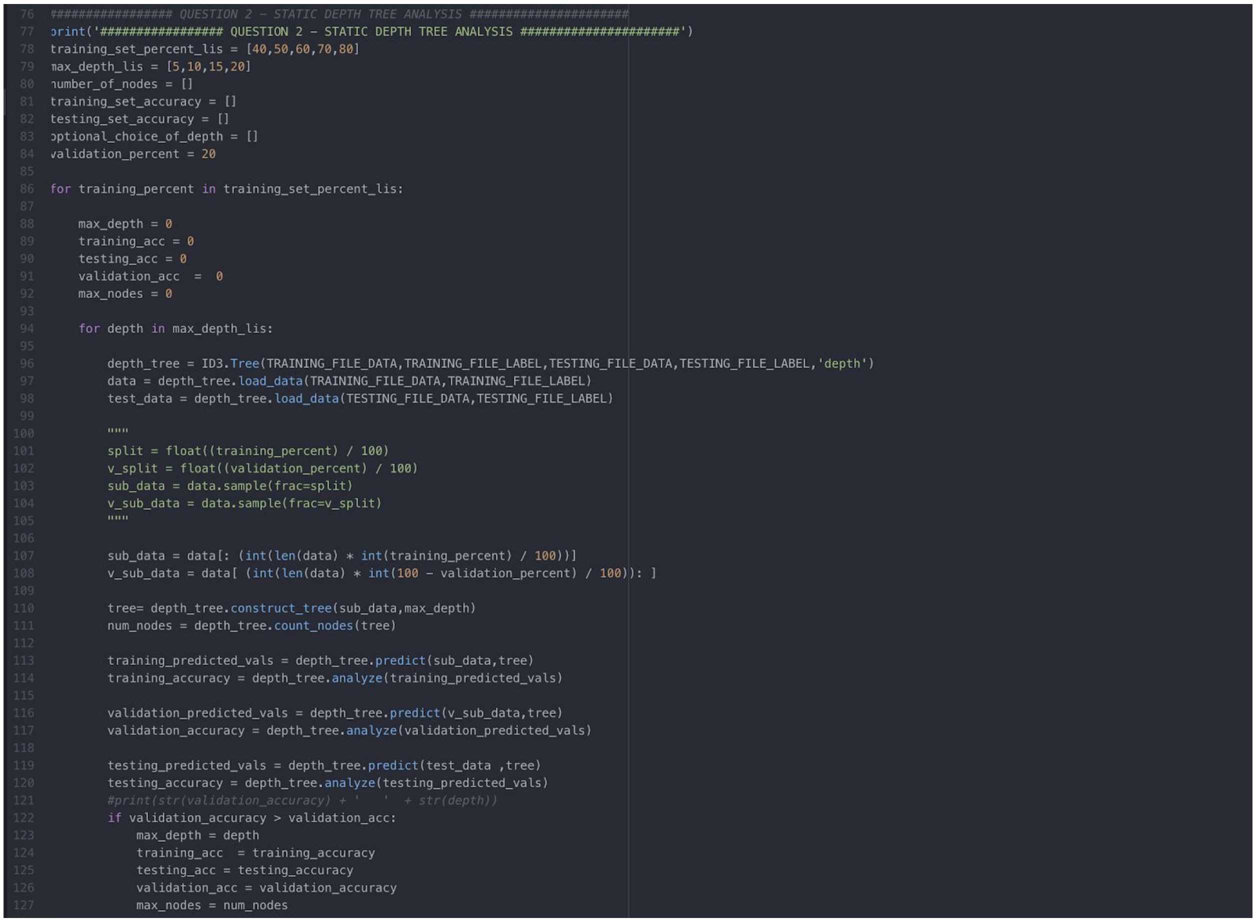

Number Of Nodes vs. Training Set Percentage (Static Depth Tree)

Rohith Ravindranath

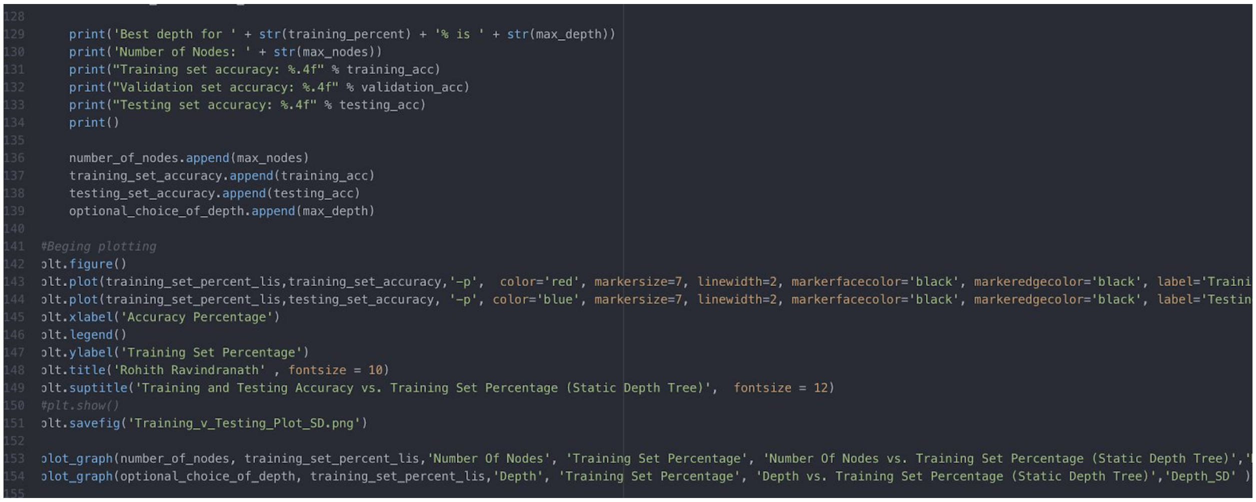

Depth vs. Training Set Percentage (Static Depth Tree)

Rohith Ravindranath

Question 2 Code Snippet

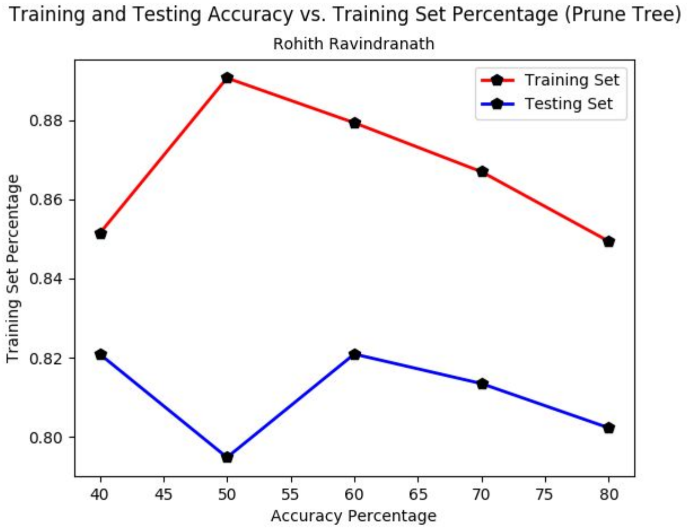


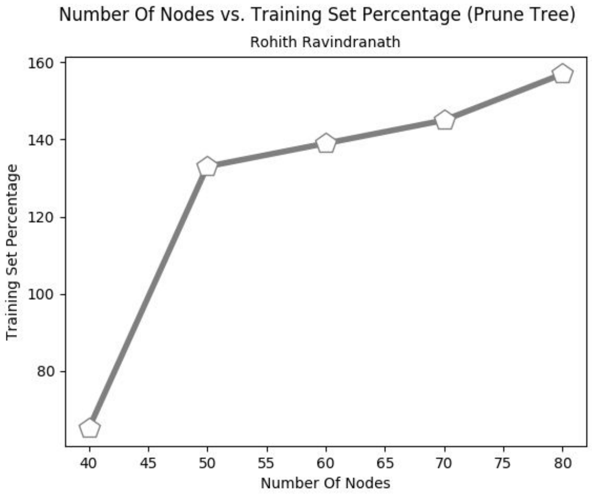
```python
################# QUESTION 2 - STATIC DEPTH TREE ANALYSIS ######################
print('################# QUESTION 2 - STATIC DEPTH TREE ANALYSIS ######################')
training_set_percent_lis = [40,50,60,70,80]
max_depth_lis = [5,10,15,20]
number_of_nodes = []
training_set_accuracy = []
testing_set_accuracy = []
optional_choice_of_depth = []
validation_percent = 20

for training_percent in training_set_percent_lis:

    max_depth = 0
    training_acc = 0
    testing_acc = 0
    validation_acc  =  0
    max_nodes = 0

    for depth in max_depth_lis:

        depth_tree = ID3.Tree(TRAINING_FILE_DATA,TRAINING_FILE_LABEL,TESTING_FILE_DATA,TESTING_FILE_LABEL,'depth')
        data = depth_tree.load_data(TRAINING_FILE_DATA,TRAINING_FILE_LABEL)
        test_data = depth_tree.load_data(TESTING_FILE_DATA,TESTING_FILE_LABEL)

        """
        split = float((training_percent) / 100)
        v_split = float((validation_percent) / 100)
        sub_data = data.sample(frac=split)
        v_sub_data = data.sample(frac=v_split)
        """

        sub_data = data[: (int(len(data) * int(training_percent) / 100))]
        v_sub_data = data[ (int(len(data) * int(100 - validation_percent) / 100)): ]

        tree= depth_tree.construct_tree(sub_data,max_depth)
        num_nodes = depth_tree.count_nodes(tree)

        training_predicted_vals = depth_tree.predict(sub_data,tree)
        training_accuracy = depth_tree.analyze(training_predicted_vals)

        validation_predicted_vals = depth_tree.predict(v_sub_data,tree)
        validation_accuracy = depth_tree.analyze(validation_predicted_vals)

        testing_predicted_vals = depth_tree.predict(test_data ,tree)
        testing_accuracy = depth_tree.analyze(testing_predicted_vals)
        #print(str(validation_accuracy) + '    '  + str(depth))
        if validation_accuracy > validation_acc:
            max_depth = depth
            training_acc  = training_accuracy
            testing_acc = testing_accuracy
            validation_acc = validation_accuracy
            max_nodes = num_nodes
```



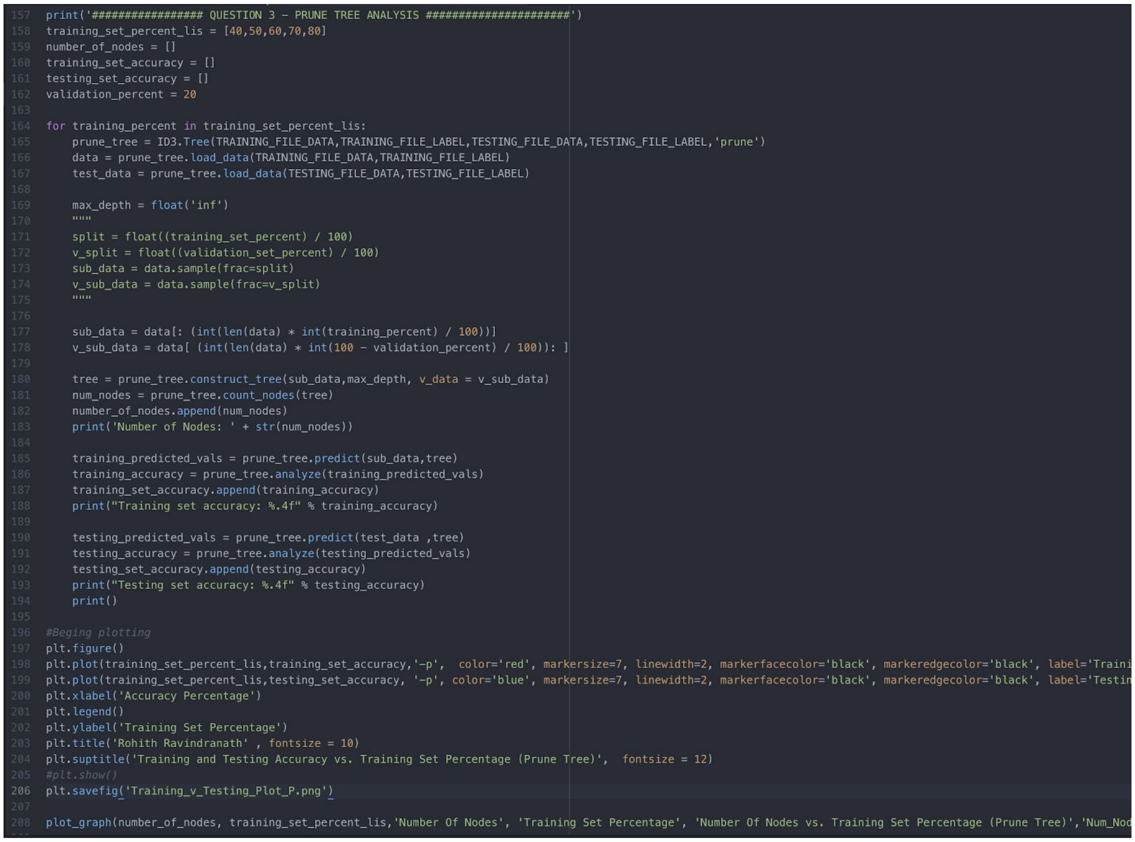
```python

    print('Best depth for ' + str(training_percent) + '% is ' + str(max_depth))
    print('Number of Nodes: ' + str(max_nodes))
    print("Training set accuracy: %.4f" % training_acc)
    print("Validation set accuracy: %.4f" % validation_acc)
    print("Testing set accuracy: %.4f" % testing_acc)
    print()

    number_of_nodes.append(max_nodes)
    training_set_accuracy.append(training_acc)
    testing_set_accuracy.append(testing_acc)
    optional_choice_of_depth.append(max_depth)

#Beging plotting
plt.figure()
plt.plot(training_set_percent_lis,training_set_accuracy,'-p',  color='red', markersize=7, linewidth=2, markerfacecolor='black', markeredgecolor='black', label='Traini
plt.plot(training_set_percent_lis,testing_set_accuracy, '-p', color='blue', markersize=7, linewidth=2, markerfacecolor='black', markeredgecolor='black', label='Testin
plt.xlabel('Accuracy Percentage')
plt.legend()
plt.ylabel('Training Set Percentage')
plt.title('Rohith Ravindranath' , fontsize = 10)
plt.suptitle('Training and Testing Accuracy vs. Training Set Percentage (Static Depth Tree)',  fontsize = 12)
#plt.show()
plt.savefig('Training_v_Testing_Plot_SD.png')

plot_graph(number_of_nodes, training_set_percent_lis,'Number Of Nodes', 'Training Set Percentage', 'Number Of Nodes vs. Training Set Percentage (Static Depth Tree)','
plot_graph(optional_choice_of_depth, training_set_percent_lis,'Depth', 'Training Set Percentage', 'Depth vs. Training Set Percentage (Static Depth Tree)','Depth_SD' )

```

3. Prune Tree Plots





Question 3 Code Snippet



```
print('################ QUESTION 3 - PRUNE TREE ANALYSIS #####################')
training_set_percent_lis = [40,50,60,70,80]
number_of_nodes = []
training_set_accuracy = []
testing_set_accuracy = []
validation_percent = 20

for training_percent in training_set_percent_lis:
    prune_tree = ID3.Tree(TRAINING_FILE_DATA,TRAINING_FILE_LABEL,TESTING_FILE_DATA,TESTING_FILE_LABEL,'prune')
    data = prune_tree.load_data(TRAINING_FILE_DATA,TRAINING_FILE_LABEL)
    test_data = prune_tree.load_data(TESTING_FILE_DATA,TESTING_FILE_LABEL)

    max_depth = float('inf')
    """
    split = float((training_set_percent) / 100)
    v_split = float((validation_set_percent) / 100)
    sub_data = data.sample(frac=split)
    v_sub_data = data.sample(frac=v_split)
    """

    sub_data = data[: (int(len(data) * int(training_percent) / 100))]
    v_sub_data = data[ (int(len(data) * int(100 - validation_percent) / 100)): ]

    tree = prune_tree.construct_tree(sub_data,max_depth, v_data = v_sub_data)
    num_nodes = prune_tree.count_nodes(tree)
    number_of_nodes.append(num_nodes)
    print('Number of Nodes: ' + str(num_nodes))

    training_predicted_vals = prune_tree.predict(sub_data,tree)
    training_accuracy = prune_tree.analyze(training_predicted_vals)
    training_set_accuracy.append(training_accuracy)
    print("Training set accuracy: %.4f" % training_accuracy)

    testing_predicted_vals = prune_tree.predict(test_data ,tree)
    testing_accuracy = prune_tree.analyze(testing_predicted_vals)
    testing_set_accuracy.append(testing_accuracy)
    print("Testing set accuracy: %.4f" % testing_accuracy)
    print()

#Beging plotting
plt.figure()
plt.plot(training_set_percent_lis,training_set_accuracy,'-p',  color='red', markersize=7, linewidth=2, markerfacecolor='black', markeredgecolor='black', label='Traini
plt.plot(training_set_percent_lis,testing_set_accuracy, '-p', color='blue', markersize=7, linewidth=2, markerfacecolor='black', markeredgecolor='black', label='Testin
plt.xlabel('Accuracy Percentage')
plt.legend()
plt.ylabel('Training Set Percentage')
plt.title('Rohith Ravindranath' , fontsize = 10)
plt.suptitle('Training and Testing Accuracy vs. Training Set Percentage (Prune Tree)',  fontsize = 12)
#plt.show()
plt.savefig('Training_v_Testing_Plot_P.png')

plot_graph(number_of_nodes, training_set_percent_lis,'Number Of Nodes', 'Training Set Percentage', 'Number Of Nodes vs. Training Set Percentage (Prune Tree)','Num_Nod
```

Question 4

We don't want to prune directly on the test set as this will make the decision optimized for that specific data set. In other words, pruning using the test data set will create a biased towards that specific data set and give a higher accuracy percentage than what it should be. Another reason that, we the testing set is too big and will not be efficient to use it as pruning mechanism.

Question 5

For a normal decision tree (vanilla tree) instead of classifying the terminal node with only one class label, you would calculate the probability distribution of all the class labels from the sub-set of data that corresponds to this terminal node. When we are using the tree to predict for a given test data set, we will get a ranking model based on a probability distribution of all class labels rather than only one class labe when we reach a terminal node. With respect to a depth decision tree, when we reach the max depth we will simply convert that node into a terminal node and calculate the probability distribution of all the class labels from the sub-set of data that corresponds to this node. When we are pruning the tree, if we decide to prune the child nodes of a certain node, we would simple calculate the probability distribution for the node based on the sub-set of data of both the child nodes.