ASSIGNMENT 5 Report

MOHIT MARWARI(112027274)
SRUTI KUMARI(112026600)

PART I: Clickstream mining with Decision Trees

Threshold	Tree Prediction accuracy	Output file prediction accuracy	Nodes created
0.01	0.748	0.748	6
0.05	0.748	0.748	6
0.1	0.748	0.748	76
1.0	0.748	0.748	1371

1. Tree prediction accuracy and output file prediction values are taken from the output of autograder_basic.py. To find the number of nodes created, we have kept a global counter and incremented it whenever the constructor of TreeNode() was called.
   Also, on running the autograder we get the message that "Tree prediction matches output file".
2. As we saw, there was not much improvement in accuracy on increasing the threshold, so it would be good if we prune the tree using lower threshold. As this would lead to expanding less number of nodes with similar accuracy and thus would help in reducing computation time.

PART II: SPAM FILTER AND EXTRA CREDIT:

In this part of the assignment we need to make a spam filter that uses a Naives Bayes. We first use the training data to find out the number of occurrences of the spam words in this training data. We divide this by the number of words in the training data to get the probability of spam words. Now similarly we will calculate the number of ham words in the training data and calculate its probability.
To identify whether the mail is a *spam* or a *ham* we do the following: we keep a track of all the spam words and their number of occurrences in a dictionary called *spam_words_dict.* Now we analyse the test data set and check check the conditional probability of that word being a spam word. After calculating the spam word probability we will check the conditional probability of that word being a ham word. Now we will sum up the conditional probabilities for all the words(words being both a spam and ham) in the mail and compare the two probabilities compared, if the spam word probability is greater than the ham word probability, we will mark that mail as a spam otherwise we will mark it as a ham. While, calculating the probabilities we are taking the sum of the log of the probabilities instead of taking the products of the probabilities. We are doing this as probability is a number between 0 and 1 and multiplying consecutive probabilities together was causing the probabilities to become even smaller at each iteration. So we are taking the sum of log instead. The value of the prediction that we got after this was: **88.3%.**

While calculating the probabilities, it might happen that some words occur only in the test data and are not seen in training. To account for this factor, we are using the concept of Laplace Smoothing. We tried many values of Laplace smoothing: 0.01, 0.1, 1, 2, 5, 10. The effect of each parameter value is shown below:

Parameter Value	Accuracy (%)
0	88.3
0.01	90.8
0.1	90.8
1	90.8
2	90.8
5	91.0
10	91.1
100	91.6

We find that for our case, when the parameter value is 100, we are getting the most accuracy. So we will take this value as our smoothing parameter. This also shows that by adding the principle of Laplace Smoothing we are able to increase our model accuracy. Laplace smoothing would also help us the avoid NaN values sometimes while calculating the probabilities (log0 values) Other ways of improving to increase the accuracy could be to introduce domain specific features into our models. Some of them are that spam words have frequenct occurances of !. Other could be some spam mails like: “You have won 10000$. Submit your mail and bank account details to claim it”. For these sentences, we can give more probability to numbers and money and add this as a feature to our model to better improve our accuracy.