Part 1:

The code is run through the command: `python q1_classifier.py -p 1 -f1 clickstream-data/train.csv -f2 clickstream-data/test.csv -o clickstream-data/train_label.csv -to clickstream-data/test_label.csv` . Where clickstream-data/train.csv refers to the training data set, clickstream-data/test.csv refers to the test data set, clickstream-data/train_label.csv, refers to the labels of the training set, clickstream-data/test_label.csv refers to the labels of the test data set, and '1' refers to the p cutoff value.

The way the code works is by iterating over all possible attributes, and finding the min p-value produced from a particular split (by looking at all possible splits). Then, if this p-value is less than the threshold p-value, it is added to a list of acceptable attributes. From this list, the node with the most information gain is taken. This information gain is calculated by calculating the entropy of the parent, and the conditional entropy of the children. The children are then added to the tree, and the cycle is repeated until no nodes pass the p-value threshold.

Running the code with a p-value threshold of 1, we obtain the following:

Number Of Errors: 8046
Total Labels: 25000
    PERCENT ERROR: 32.184000000000005
    ACCURACY: 0.67816

Size of tree: 15373

Changing the p-value threshold to .05, we obtain:

Number Of Errors: 7294
Total Labels: 25000
    PERCENT ERROR: 29.176000000000002
    ACCURACY: 0.70824

Size of tree: 2259

Changing the p-value threshold to .01, we obtain:

Number Of Errors: 6496
Total Labels: 25000

PERCENT ERROR: 25.984
ACCURACY: 0.74016

Size of tree: 603

Accuracy is taken to be the value of (True Positives + True Negatives)/(True Positives + True Negatives + False Positives + False Negatives).

We understandably observe that as we lower the p-value threshold, the size of the tree is reduced; less nodes pass the Chi-square criteria. Additionally, we also see that the accuracy goes up. This is also expected. We started off by building the full tree, we essentially made it so that each leaf was an individual input value. The accuracy of this is expectedly low, because we would need to have the same exact input leaf in order to follow where it would go on the tree. This tree is extremely subject to the effects of noisy data. Then, as we limited the tree, we dealt more with probabilities and an input value could be placed in without having to exactly replicate previous values. This essentially helped limit the effects of noisy data.

Something else important to note, is that having our shortest tree, produces the most accuracy. This means that relatively quickly, we have gained enough information in order to classify attributes.

The accuracies are all relatively high. Due to such, I haven't done anything to try to improve performance, other than changing the p-value threshold.