Artificial Intelligence
Report For Project 05: Machine Learning

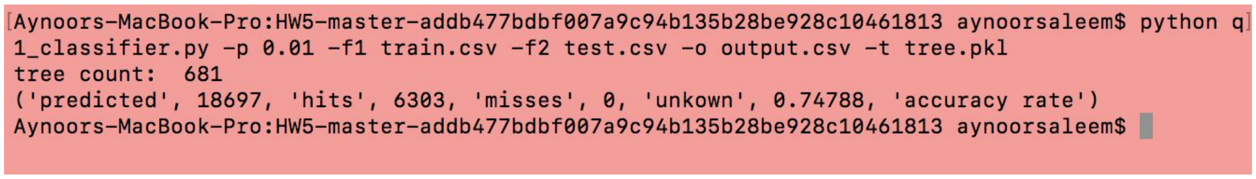
Question 1) (50 Points) : Clickstream Mining with Decision Trees

Function ID3Ply(trainingData, labelList, testPValue, skipList):
The training data is parsed into the data objects (the 'feature' class), and the hit/miss count of each feature is updated. In the case where the node passes the relevance test, i.e. the its p-value is less than the test p-value, a new node is added to the decision tree. Next the data is split based on the the selected feature's values, and the algorithm runs recursively on each subset of the training data. For cases where A) no feature of the node meets the p-value significance test, B) no training cases remain or C) no attributes remain, the algorithm creates a LeafNode returns.

Classes and Functions Implemented:

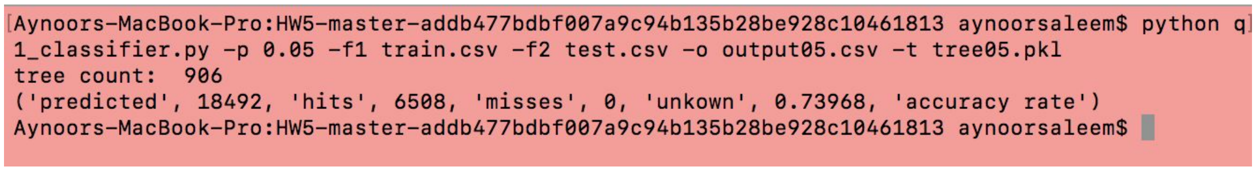
ID3Ply: the recursive algorithm that sorts the data and checks features for relevance
Feature: a data-class to represent the features, their values and occurrence counts
DecisionNode: a decision node in the decision tree (supports non-normalized data)
LeafNode: a leaf nodes in the decision tree
DecisionTreeToTestingTree: transforms the decision tree to the grading format
TreeNode: (from TA) the grading format of a decision tree

<u>Results:</u>
For each value of threshold, what is your tree's accuracy and size (size equals number of internal nodes and leaves)?
python q1_classifier.py -p 0.01 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl



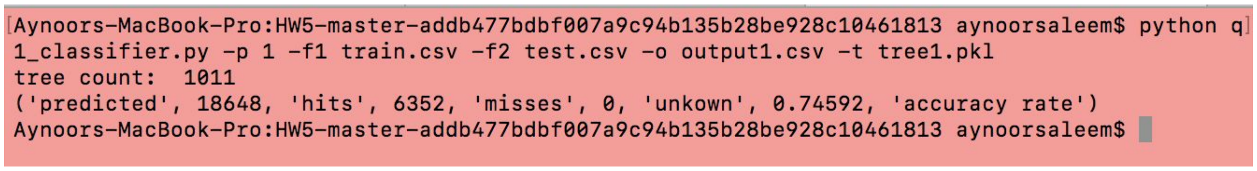
```
Aynoors-MacBook-Pro:HW5-master-addb477bdbf007a9c94b135b28be928c10461813 aynoorsaleem$ python q1_classifier.py -p 0.01 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
tree count:  681
('predicted', 18697, 'hits', 6303, 'misses', 0, 'unkown', 0.74788, 'accuracy rate')
Aynoors-MacBook-Pro:HW5-master-addb477bdbf007a9c94b135b28be928c10461813 aynoorsaleem$
```

python q1_classifier.py -p 0.05 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl



```
Aynoors-MacBook-Pro:HW5-master-addb477bdbf007a9c94b135b28be928c10461813 aynoorsaleem$ python q1_classifier.py -p 0.05 -f1 train.csv -f2 test.csv -o output05.csv -t tree05.pkl
tree count:  906
('predicted', 18492, 'hits', 6508, 'misses', 0, 'unkown', 0.73968, 'accuracy rate')
Aynoors-MacBook-Pro:HW5-master-addb477bdbf007a9c94b135b28be928c10461813 aynoorsaleem$
```

python q1_classifier.py -p 1 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl



```
[Aynoors-MacBook-Pro:HW5-master-addb477bdbf007a9c94b135b28be928c10461813 aynoorsaleem$ python q]
1_classifier.py -p 1 -f1 train.csv -f2 test.csv -o output1.csv -t tree1.pkl
tree count:  1011
('predicted', 18648, 'hits', 6352, 'misses', 0, 'unkown', 0.74592, 'accuracy rate')
Aynoors-MacBook-Pro:HW5-master-addb477bdbf007a9c94b135b28be928c10461813 aynoorsaleem$ 
```

What do you observe?
As the threshold increases the accuracy improves little; however, the count of nodes in the tree increase as the threshold p-value increases. This is because we are accepting a wider distribution on the relevancy curve, so even semi-relevant features get included.

Explain which options work well and why?
The option that works best is the one with the lowest threshold, i.e. 0.01. Oddly though, a threshold value of 1 also has good accuracy. From these initial results, it appears that only a very few features have good predictive accuracy over a user's click behavior.

Question 2(50 Points) : Spam Filter

Implement the Naive Bayes algorithm to classify spam.

Analysis:
We classified the mail in spam/ham classes using Naive Bayes Classifier. The Bayes Theorem used to classify the mail vector is :

$$p(C_i|X) = \frac{p(C_i)p(X|C_i)}{p(X)}$$

Where,

P(Ci|X)	: Probability of being in Class Ci given data X also called as posterior.
P(Ci)	: Prior probability of class Ci which is updated into posterior after evidence X
P(X|Ci)	: It is likelihood of observing evidence given class Ci.
P(X)	: It is the normalizing factor. Generally constant over classes.

We extracted the features in each email. Then we iterated over all features to update the posterior weights using the likelihood of features given ham/spam. We tried using the log of the probabilities as a smoothing parameter, but there was no noticeable performance increase.

Statistics:

Without looking at email domains:

Category	Total	Predicted Correctly	Accuracy
Ham	420	411	97.85
Spam	580	446	76.89
Total	1000	857	85.7

When we looked at the email domains as well, the accuracy increased significantly. See below:

With Full Access to Emails(Extra Credit Part):

Category	Total	Predicted Correctly	Accuracy
Ham	420	417	99.28
Spam	580	505	87
Total	1000	922	92.2

Contributions:

Michael Anderson: Started part 1 wrote the skeleton code and started off with implementing the algorithm
Aynoor Saleem: Continued part 1 with Michael and finished it
Jatin Garg: Started off with implementation of part 2
Urmi : Continued implementing part two and finished it.
Extra Credit was mutually done.

For all the parts all group members consulted any issues with each other and helped in resolving them.