CSE 537: Artificial Intelligence
I.V. Ramakrishnan
Fall 2022

Project 05 Report

Menachem Goldring
Nicole Niemiec

Part I: Clickstream Mining with Decision Trees

1) Results:

p_value=1:

```
Number of internal nodes: 185
Number of leaf nodes: 741
Model Accuracy: 0.7388
```

p_value=0.05:

```
Number of internal nodes: 22
Number of leaf nodes: 89
Model Accuracy: 0.75332
```

p_value=0.01:

```
Number of internal nodes: 19
Number of leaf nodes: 77
Model Accuracy: 0.75328
```

From these results, we noticed that the lower the p_value for the chi-square criterion, the smaller the size of the tree. There's also the fact that with a full tree, the accuracy was slightly worse than with a smaller tree obtained from a smaller p_value.

2) From the results we obtained. We can see that with a p_value of .05, we obtained the best accuracy. However if in a different case where memory usage/ tree size was more important, with a p_value of 0.01 resulting in the smallest size tree, that would be the best option instead of p_value of 0.05

Part II: Spam Filter

Implementation:

In order to build our Naive Bayesian Spam Filter, we can use the given bag of words model. For every email, we find the conditional probability of it being spam given the contents, and if this conditional probability is greater than the conditional probability of the email not being spam, we classify the email as spam. In order to do this, we compare our numerators given that the denominators are the same in both probabilities.

To calculate the probability of spam, or P(spam), we get the counts of the total spam and not spam emails from the given training dataset. We can use the same technique for P(not spam), the probability of not spam.

To calculate the conditional probabilities of P(word|not spam) and P(word|spam), we calculate the fraction of times a word occurs in both spam and non spam emails.

Testing:

In order to train our spam filter, we work in two phases. First, after computing the conditional probability of a collection of words that were included in a spam email, we then multiply them with the individual probability of an email being labeled as spam, resulting in the probability of an email being labeled as spam. We will repeat this process for emails labeled as not spam. We make a decision based on if the probability of an email being spam is greater than the probability of it not being spam.

In order to optimize our distribution, we can add a laplace smoothing function to our multinomial distribution by adding an alpha value in order to increase our accuracy. This alpha value represents the frequency of occurrence when we find new words not in our word dictionary.

Results:

With Laplace Smoothing, we are able to achieve an accuracy of 95.69% with our Naive Bayes Spam Classifier. However, we note that with a different test dataset than our training dataset, our implementation will not work; this is because any zero probability will upset our probability function as a whole.