Project 05: Machine Learning

COURSE: ARTIFICIAL INTELLIGENCE (CSE 537)

INSTRUCTOR: Professor I.V. Ramakrishnan

PROJECT PARTNERS

GULSHAN BHATIA (111491348)

gbhatia@cs.stonybrook.edu

Contribution = 25%

HIMANI SINGH (111491995)

hisingh@cs.stonybrook.edu

Contribution = 25%

AISHWARYA DANOJI(111493647)

adanoji@cs.stonybrook.edu

Contribution = 25%

UNS REHMAN(111743506)

urehman@cs.stonybrook.edu

Contribution = 25%

1. Clickstream Mining with Decision Trees [50 points]

A decision tree is represented as with each internal node for testing an attribute where branch corresponds to an attribute value and each leaf node assigns a classification.

We first take the input dataset as provided. For the given training examples, we first calculate the entropy to measure the impurity of the training samples.
The number of positive and negative examples in the training set is calculated. Then the entropy is calculated as below:

$$\text{Entropy} = -p_{pos}\log p_{pos} - p_{neg}\log p_{neg}$$

Once the entropy is calculated, information gain is computed. IG is the reduction in entropy achieved. It is the change in the information entropy from a prior state to a state that takes some information as:

$$\text{Gain}(S,a) = \text{Entropy}(S) - \text{Entropy}(S|a)$$

Steps followed in our program are as follows:

1-The first step in this process is getting the "next best" attribute from the set of available attributes. The call to split_attribute takes care of this step.

2- The next step is to create a new decision tree containing the chosen attribute as the root node. All that remains to do after this is to create the subtrees for each of the values in the best attribute.The values for the chosen attribute is stored in the var featureVal.

3-Next, the code loops through each of these unique values and creates a subtree for them by making a recursive call to the create_decision_tree function.As given in the problem statement there can be at most 5 children for each node.

4-A list is maintained to keep track of the attribute already selected for splitting. Again the entropy and information gain is calculated to determine on which attribute to split in the next subtree.

5-Once the next-best-attribute is found it is passed to the create_decision_tree function. The call to create_decision_tree will return the subtree for the remaining list of attributes and the subset of data passed into it.

6-Now all that's left is to add each of these subtrees to the current decision tree and return it.

7- We then use the chi-square stopping criterion to split when the attribute chosen is irrelevant. The value of p is computed which is the probability of the observed values.
8- If the value of p is smaller than the threshold, the test is passed. With the increase in the p-value, the size of the tree increases.

Screenshots:

1. When p = 0.01

```
C:\Python>python submit.py -p 0.01 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated

C:\Python>python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.7284
Output file prediction accuracy:  0.7284
Tree prediction matches output file
```

2. When p = 0.05

```
C:\Python>python submit.py -p 0.05 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated

C:\Python>python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.7284
Output file prediction accuracy:  0.7284
Tree prediction matches output file
```

3. When p = 1

```
C:\Python>python submit.py -p 1 -f1 train.csv -f2 test.csv -o output.csv -t tree.pkl
Data Loading: done
Training...
Testing...
Output files generated

C:\Python>python autograder_basic.py
Data Loading: done
Tree prediction accuracy:  0.7284
Output file prediction accuracy:  0.7284
Tree prediction matches output file
```

Accuracy = 72%

2. Spam Filter [50 points]

Various functions defined in our program are:

1) main() - To parse the training and testing data , that is given in command line by the user.
2) classifymails()- to determine whether the mail is spam or not and write the result into output.csv file.
3) NB() - To carry out naive bayes classification calculation, to find probability of a mail being spam or not spam.
4) getAccuracy ()- to find the accuracy of the implemented naive bayes classifier.

The equation that our program uses for calculation is as follows:
Wi= words[i] that has occured in the training dataset
S= spam
H= ham

$$P(S/Wi) = \frac{P(Wi/S)P(S)}{P(Wi/S)P(S)+P(Wi/H)P(H)}$$

Pr(S|Wi) is the probability that a message is a spam,given that the word “Wi” is in it;
Pr(S) is the overall probability for message is spam;
Pr(Wi|S) is the probability that the word "Wi" appears given that message is spam;
Pr(H) is the overall probability for message is not spam(ham)
Pr(Wi|H) is the probability that the word "Wi" appears given that message is ham.

Assuming that there is no *a priori* reason for any incoming message to be spam rather than ham or vice-versa, we can consider both cases to have equal probabilities of 50%

$P(S) = 0.5 \; ; \; P(H) = 0.5$

Thus we can rewrite the above equation as :

$$P(S/Wi) = \frac{P(Wi/S))}{P(Wi/S)+P(Wi/H)}$$

In our program in function NB() :

s= P(Wi/S)

p=P(Wi/H)

p1=P(S/Wi)

Without taking laplacian smoothing into consideration the Accuracy obtained is 86%.

We need to use smoothing parameter in case the word present in test sample is not present in train sample. Because for such words value of both “s” and “p” as defined above comes to 0.

In that case we have used the following equation to find the correct probability for such words:

$$P'(S/Wi) = \frac{aP(S)+nP(S/Wi)}{a+n}$$

Here ,

P’(S/Wi) is the correct probability obtained by applying laplacian smoothing , done for taking into consideration words that are not present in training set.

a- Strength we give to background information about incoming spam ;

n is the number of occurrences of this word “Wi” in the training dataset ;

We get the best accuracy when we take value of strength ie. “a” as 3. The corrected probability is stored in the variable “np” in function NB(). As we increase the value of “a” the accuracy tends to decrease .

These probability values np are stored in the array wordp[] for every word present in the sample test data.

Now to find the probability that the message is a spam we use the following equation:

$$p = \frac{p1*p2*.....*pn}{p1*p2*...pn+(1-p1)(1-p2)...(1-pn)}$$

The above value is calculated and stored in variable “prob” in function NB.

Once we have calculated the probability for a mail being spam and not spam, I compared these two probabilities .

In classifymails() we have checked which value is greater spam_prob or ham_prob and assigned the label spam or ham to that test sample accordingly.

Accuracy = 92%

Below is the screenshot on running the program:

References:

https://en.wikipedia.org/wiki/ID3_algorithm

https://en.wikipedia.org/wiki/Chi-squared_test

https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering