CS 537: Artificial Intelligence
HW#5 Report
ML-Classifiers

Kumar Anupam (110614410)
Muhammad Ali Ejaz (110559131)

This report is on HW#5 and has two parts: Naïve Bayes and Decision-Tree.

Part 1: Naïve Bayes

Here we built a Naïve Bayes Classifier for classifying handwritten digits. The data given to us contained labeled training and test set. We used the training set for building the classifier. While building the classifier we first calculated the probability for white background and dark background. Here our design takes '+' and '#' to be dark background and ' ' to be white background for each pixel in the training image. We used a data structure called defaultdict (dictionary) in python so that two probabilities (white probability and dark probability) for every instance of each digit (from 0 to 9) can be stored and updated. We create dictionary out of dictionaries to store the training data.
After calculating the prior probability, we use log likelihood and Laplace smoothing to identify the test data images. The predicted digits are stored in a file called "output.txt". The test labels and the output labels are compared and the overall and per digit precision measurements along with other data are printed out.

How to run the code:

```
python nbayes.py
```

<The filenames traininglabels.txt, trainingimages.txt, testlabels.txt, testimages.txt, output.txt are hardcoded into the code>

Output:

```
Number of each digit in test data and its corresponding match number:
Digit 0 : Test data occurrence: 90 ; Matched successfully: 86
Digit 1 : Test data occurrence: 108 ; Matched successfully: 102
Digit 2 : Test data occurrence: 103 ; Matched successfully: 69
Digit 3 : Test data occurrence: 100 ; Matched successfully: 87
Digit 4 : Test data occurrence: 107 ; Matched successfully: 65
Digit 5 : Test data occurrence: 92 ; Matched successfully: 21
Digit 6 : Test data occurrence: 91 ; Matched successfully: 72
```

```
Digit 7 : Test data occurrence: 106 ; Matched successfully: 72
Digit 8 : Test data occurrence: 103 ; Matched successfully: 85
Digit 9 : Test data occurrence: 100 ; Matched successfully: 84

Percentage precision for each digit:
Digit 0 : 95.5555555556 %
Digit 1 : 94.4444444444 %
Digit 2 : 66.9902912621 %
Digit 3 : 87.0 %
Digit 4 : 60.7476635514 %
Digit 5 : 22.8260869565 %
Digit 6 : 79.1208791209 %
Digit 7 : 67.9245283019 %
Digit 8 : 82.5242718447 %
Digit 9 : 84.0 %

Total number of digits in test data: 1000
Total number of matched digits: 743
Overall percentage of precision: 74.3 %
```

Part 2: Decision-Tree

Here we built a Decision Tree to predict the credit worthiness of applicants. The data set had 3 types of attribute values: categorical, integers and reals. Some attribute values were missing. For missing values of continuous attribute, we took the average of present values and assigned to the missing values, since that would be the best estimate of these types of missing values. For missing values of discrete attributes, we take the value with maximum frequency to fill in.
To calculate the root node, we calculate the Information Gain for each column and take that column as the root node which gives us the maximum information gain. The attributes are retrieved and edges are constructed out of it. For each edge we check if it ends in a leaf node or not. Other edges end in attribute nodes and tree is further built recursively.
For better code clarity and ease we have used panda data frames to store and update the data set.
We have taken a set of 550 rows from the data set as the training data and the remaining 50 rows as the test data.
The accuracy that we get is 88%.

How to run the code:

```
python dtree.py
```

Output:

```
Accuracy of test data obtained: 88.0 %
```