Hw 5 Report

Submitted by :
Varun Kumar Vala(109929644)
Venkateswara Prasad Pandeti(110396994)

Naïve Bayes

Naïve Bayes Problem Design:

- When the testing images & labels are the given, what we want to predict is Probability of a Label given the 784 features in the image i.e. P(Label/f1,f2,f3,…,f784)
- So, we need to take argMax over all the Labels to give out the LABEL that suits that image in consideration i.e. argMax[P(Label/f1,f2,f3,…,f784)]
- For this we will need
    - Prior Probability of given Label – This is calculated using the training label data
    - Probability of a feature given Label i.e. P(f/Label)
- Now, it can be written as argMax[PriorProbability(Label)*P(f1/Label)*P(f2/Label)*,…,P(f784/Label)]
- Taking Log over this, makes this calculation easy.
- Also to ensure any of the terms don't give out zero, as that gives out an exception, we do smoothing with a factor of k=2 before we apply Log on the equation.

Calculating Probability of a feature given Label i.e. P(f/Label)

- For this, with the training data we created a List for each feature category i.e. ' ','+','#'
- This list is list of locations which is again of dictionary of number of times this feature occurred against each label, so at each location we will call a tuple "Label" : Count
- Please refer to <logCalculateForFeaturesGivenLabel>method for understanding the logic better.
- Everything is straight-forward once this list for each feature is computed from the training data.

Results :

- **Accuracy is coming out to be 77% roughly.**
- **Please refer README.txt to check the results. I'm printing out the results towards the end through <calculateAccuracy> method**

Decision Trees

Decision Trees Design :

- We are calculating the best attribute to split upon using entropy & information gain concept
- Once that is done, we recursively create Child Nodes each of them will again decide on the attribute to split upon and we will add these childNodes to the Root Node. Here we will reduce one feature for the childNode to build its tree.
- This process will terminate when the features are exhausted or when the data is exhausted. Here, we take the majorityLabel as the Leaf i.e., the Label that occurred more in the training set.
- Termination also happens when all the Labels are the same for the samples.Here we return that Label itself as the Leaf.

Handling Continuous Variables:

- We first check if the feature is float type/integer type/string type and do the required entropy&informationGain calculations
- Here, we either partition the sample into intervals.
- Another approach was to split the data into subsets over the mid((min+max)/2) of the sample
- For the 2nd approach gave better results

Handling Missing Data:

- Here, we first the Label against this particular sample. Then, we pick that 'featureValue' which has the highest frequency for the Label across the training samples.
- **Though, this approach was suggested by some professor in one of his online lectures, it didn't give us good results and we scraped them off in the program.**
- You can find the related code in one of the backup copies that are submitted.

Results:

- I have split the data into 75:25 ratio
- 75% i.e **training data** is kept in 'crx.data_backup_Training.txt'
- 25% i.e **testing data is** kept in 'crx.data_backup_Test.txt'
- **Accuracy is coming out to be 49% roughly**
- **Please refer README.txt to check the results. I'm printing out the results towards the end through < testAccuracyDecisionTree> method**