* 1. Write a paragraph describing how your code works.

The code for this assignment is written in Python using python 2.7.12 and can be found under https://github.com/nivetha27/Machine-Learning-Assignments/tree/master/Assignment%201. All the logic is present in PythonApplication1.py file. Please run the following command to install all dependencies :

*pip install -r requirements.txt*

To run for different confidences, please update the confidence array in line 606 of PythonApplication1.py. Also the training data file and test data file have to be in the same path as PythonApplication1.py

Here are the key functions in the code:

* Decision Tree Builder
  + def buildDTUsingID3(examples, attributes, targetAttribute, confidence, replaceMissingValuesByIgnoringClass = True, applyChiSquareBeforeChoosingBestAttr = True)
  + This function build the decision tree for a given training data set and set of attributes.
  + The confidence level is used for chi square testing to determine if a given node should be selected or not.
  + replaceMissingValuesByIgnoringClass determines how missing values are to be replaced. When true replaces unknown value the most occurring attribute value in the instances at that node irrespective of the class. When false, replaces unknown value with the most occurring attribute value matching the respective classes in the instances at that node
  + applyChiSquareBeforeChoosingBestAttr determines how to choose best attribute for building the tree. When true it applies chi square first to get the list of possible best nodes and then runs gain ratio for each attribute to determine the best attribute. When false, it first obtains the best attribute using gain ratio and then applies chiSquare to determine if the attribute should be considered
* Choosing Best Attribute
  + def chooseBestAttribute(examples, attributes, targetAttIndex, sampleSize)
  + This function computes gain ratio for each attribute and determines the best attribute as the one with the highest gain ratio.
* Pre-Prune using ChiSquare
  + def PrePruneUsingChiSquare(examples, attributes, targetAttrIndex, chosenBestAttr, confidence)
  + This function computes the chi square for a given attribute and determines if it’s a useful node or not.
* Get Classification on data
  + def getClassificationOnData(tree, testDataSet, attributes)
  + This function gets the classification on a given datasest
* Determine accuracy, precision, recall
  + def determineAccuracy(tree, testDataSet, attributes, targetAttribute)
  + This function computes accuracy, precision, recall for the given ID3 tree and testData

**1.2** On the *training* set, what is the accuracy of the tree trained at each confidence level? What are the precision and recall?

|  |  |  |  |
| --- | --- | --- | --- |
| Confidence | Accuracy | Precision | Recall |
| 0% | 94.885% | 98.63% | 74.83% |
| 50% | 91.445% | 94.159% | 59.882% |
| 80% | 87.235% | 41.987% | 85.0324% |
| 95% | 82.98% | 77.35% | 18.11% |
| 99% | 81.745% | 62.98 | 15.69% |

**1.3** On the *test* set, what is the accuracy of the tree trained at each confidence level? What are the precision and recall?

|  |  |  |  |
| --- | --- | --- | --- |
| Confidence | Accuracy | Precision | Recall |
| 0% | 68.248% | 33.49% | 26.54% |
| 50% | 68.484% | 33.177% | 24.872% |
| 80% | 70.608% | 36.039% | 21.66% |
| 95% | 73.96% | 42.86% | 10.36% |
| 99% | 75.044% | 51.91% | 11.42% |

**1.4** How many decision nodes are there in the tree trained at each confidence level?

|  |  |
| --- | --- |
| Confidence | #DecisionNodes |
| 0% | 38073 |
| 50% | 14488 |
| 80% | 6502 |
| 95% | 2139 |
| 99% | 847 |

**1.5** Which level of confidence produces the best results and why? What is your criterion for "the best results"?

The 99% confidence produces the best results. This is based on the criteria that the tree has highest accuracy, precision and recall values as compared to the other confidences.

**1.6** Try to interpret the tree with the highest accuracy (on the test set). What are the first 4 decisions (or less if the tree has less than 4 levels) applied to the largest fraction of the positively-labeled training examples? Negatively-labeled? Express these paths as English sentences. Do they make sense?

First 4 decision applied to largest fraction of training set :

1. positively-labeled training examples

**\\HowDidYouFindUs (Friend/Co-worker) \\WhoMakesPurchasesForYou (spouse) \\Registration Gender (Male) \\Login Failure Count (\\'(-inf-0.5]\\)**

1. best negative path

**\\HowDidYouFindUs (Friend/Co-worker) \\WhoMakesPurchasesForYou (spouse) \\Registration Gender (Male) \\Login Failure Count (\\'(-inf-0.5]\\)**

If how you found the website was from a friend/co-worker and who makes purchase for you is your spouse and the registration gender is Male and the Login Failure Count is less that 0.5, then the user can either view the page on the site or not. The fifth decision node or the nodes further down the tree help to determine class

**1.7** If all your accuracies are low, how have you tried to improve them and what do you suspect is failing?

My accuracies are improving as the confidence increases and tends towards 75.044% for 99% confidence

In order to reach the current accuracy values, I did the following w.r.t:

1. handling missing values:
   * I assigned attribute value that occurs maximum number of times for a given attribute for the instances at that node irrespective of the target class of the instance rather than assigning attribute value that occurs maximum number of times in the instances at that node for the same class.
   * This improved precision and recall value to be 75% and 0.19% whereas before it was 0%, however my accuracies remained the same
2. Choosing best attribute:
   * Instead of applying choosing best attribute using gain ratio and then running chi square test to decide whether to choose the node or not, I ran chi square test first
   * This improved my accuracy for my best tree by 0.224% and also my precision and recall metrics by 51.91% and 11.42% where it was 0% before

**2.1** Consider a sequential covering algorithm such as CN2 and a simultaneous covering algorithm such as ID3. Both algorithms are to be used to learn a target concept defined over instances represented by conjunctions of *n* boolean attributes. If ID3 learns a balanced decision tree of depth d, it will contain 2d−1 distinct decisions nodes, and therefore will have made 2d−1 distinct choices while constructing its output hypothesis.

1. How many rules will be formed if this tree is re-expressed as a disjunctive set of rules?

There would be one rule per path, therefore 2d−1 rules would be formed.

1. How many preconditions will each rule possess?

For a given depth d of balanced tree, the number non-leaf nodes along a path would be d-1.

Since, each attribute test along the path from the root to the leaf becomes a rule antecedent (precondition) and the classification at the leaf node becomes the rule consequent (postcondition), **the number of preconditions could be** **(d-1) .**

1. How many distinct choices would a *sequential* covering algorithm have to make to learn this same set of rules?

There are 2d−1 rules and (d-1) preconditions, the number of distinctive choices would be (d-1)\* 2d−1

1. Which system do you suspect would be more prone to overfitting if both were given the same training data? Why?

The sequential covering algorithm is more prone to overfitting for the same training data. As seen in 2.1(c) sequential covering algorithm makes (d-1)\* 2d−1 choices where as decision tree makes only (2d−1) choices

**2.2** Suppose FOIL is considering adding a literal to a clause using a binary predicate P.

1. If the clause currently contains n different variables, how many distinct literals can be generated? Two literals are considered identical if they differ only in the names of the new variables that they contain.

Let the binary predicate be represented as either P(x,y) or P(y,x) with the knowledge that P(x,y) is different from P(y,x). x can take one of the n existing variables or a new variable, therefore has (n+1) choices. y also has (n+1) choices. However, two new variables cannot be taken together.

There are two cases here

1. #Literals with no new variable : nC1 \* nC1 = n2
2. Literals with one new variable : (n).1 + 1.(n) = 2n

Total literals = 2n + n2

With negated literals included, total literals would be 2(2n + n2)

1. Why do you think FOIL doesn’t allow literals that contain only new variables, i.e. no previously used variables?

If a literal contains only new variables, then either a subsequent literal in the clause body connects one or more of those variables to one or more of the “old” variables, or it doesn’t. If it does, then the same clause will be generated with those two literals reversed, such that the restriction is not violated. If it doesn’t, then the literal is either always true (if the predicate is satisfiable) or always false (if it is unsatisfiable), independent of the “input” variables in the head. Thus, the literal would either be redundant or would render the clause body equivalent to False.