**ASSIGNMENT 3**

**Assigned September 28, 2021**

**Due October 8, 2021**

**On Canvas, 11:59PM**

**50 points**

**In a comment section at the top of your program list all the team members.  There are three opportunities to submit your program.  However, only the last submitted version will be graded.**

This assignment is asking you to implement ID3 and Naïve Bayes classifiers on the Iris Data Set (<https://en.wikipedia.org/wiki/Iris_flower_data_set>).

Iris Data Set contains 150 data points (observations) of three types of the iris flower, classified into three categories, versicolor, virginica, and setosa. Each observation is a vector with 4 components, with the following respective meanings – petal length, petal width, sepal length, sepal width.  There is some overlap between virginica and versicolor, but no overlap between these and setosa (i.e. setosa is linearly separable from each viriginica and versicolor; see the plots on the pairs of dimensions C(4, 2) = 6 plots).

To make things easier, you are asked to recognize setosa vs the combined virginica and versicolor.  This way, there is a linear separation between the classes.

**NOTE:**

* **You are to write your own implementation.  Please do NOT USE MATLAB, or any other language package or toolbox**
* **If you use python, you can use numpy, matplotlib, seaborn, pandas, train test split from sklearn. Do not use any prebuilt classifiers**
* **Do not use the f1\_score and accuracy score or any other metrics from sklearn. Try to implement them by yourselves**
* **For full points, all plots should be properly labelled (with axis titles, legend, plot title)**
* **DO NOT share your work. You can discuss with your peers, but do not share your final submission or your code. No points will be awarded if we find concrete evidence of cheating**

You are to implement the following.

1) **(10 points)** The basic ID3 algorithm.

2)**(10 points)** The Naïve Bayes classifier

For computing probabilities, needed in each, the ID3 and Naive Bayes algorithms, you can use a function that computes histograms, to output the frequency of values in a histogram bin.

Discretize by using the bin center (rounded to the nearest integer) as the discrete value of an attribute.

Run your programs for bin numbers varying from 5 to 20 incrementing by 5.

For each algorithm:

1. **(10 points)**Compute the accuracy for each discretization. Compute Max\_accuracy, min\_accuracy, and average accuracy over the different number of bins.  Plot on the same figure (with different colors) these accuracy values as a function of the number of bins.

Some pointers:

* Make 5 train-test splits of the same data (with test size 33% and random split). For each split, compute the TEST accuracies for each bin size ([5:20:5]) and print them out
* Output format:

Text

Description automatically generated

* For the plots: Simply plot the accuracies for each bin size (you will get 4-line plots, plot them on the same figure
* You will have to do this for decision trees and naïve bayes
* Note: Since the dataset is binary classification of setosa and non-setosa (which is linearly classifiable), you should expect almost perfect accuracies.

2. **(5 points)**Compute the F measures (refer to my ROC ppt slides) for each discretization. Plot the F-measures over the different number of bins.

Some Pointers:

* Repeat similar steps, but this time, calculate TEST F1 score
* You do not need to print out the F1 scores
* Plot the F1 scores for all 4 bin sizes on the same plot

3. **(5 points)**Compute the ROC points for each discretization. Plot the ROC curves over the different number of bins.

Some pointers:

* Repeat similar steps, but this time, calculate TEST FPR and TPR to construct the ROC Plot.
* You do not need to print out the FPR and TPR scores
* Plot the ROC curves for all 4 bin sizes on the same plot
* Note: Your plot should look like the ‘Almost Perfect Classifier’ case (ROC ppt slides)

4. To further compare the two algorithms, assume one is the ground truth

Some pointers:

* By this we mean that use the TEST predictions from one model as the y\_true (test) for the other model
* For instance, y\_test\_true (DT) = NB(X\_test) and vice versa

4.1 **(5 points)**Compute the F measures (refer to my ROC ppt slides) for each discretization, when the ID3 is assumed to be the ground truth. Plot the F-measures over the different number of bins.

Some pointers:

* Feed X\_test to ID3, get y\_preds (this will be y\_test\_true for NB)
* Like Q2 (Get F1 score plot for NB)

4.2 **(5 points)**Compute the F measures (refer to my ROC ppt slides) for each discretization, when the Naive Bayes is assumed to be the ground truth. Plot the F-measures over the different number of bins.

Some pointers:

* Feed X\_test to NB, get y\_preds (this will be y\_test\_true for ID3)
* Similar to Q2 (Get F1 score plot for ID3)

**LIST OF DELIVERABLES:**

1. **Decision Trees:**

* **Accuracy scores for each discretization and each bin size (check sample output)**
* **1x Accuracy Score plot**
* **1x F1 Score plot**
* **1x ROC Curve**
* **1x F1 Score plot with Naïve Bayes as Ground Truth**

1. **Naïve Bayes:**

* **Accuracy scores for each discretization and each bin size (check sample output)**
* **1x Accuracy Score plot**
* **1x F1 Score plot**
* **1x ROC Curve**
* **1x F1 Score plot with Decision Tree as Ground Truth**

**SUBMISSION:**

1. **Report (.pdf):**
   * **Include group member names**
   * **Include proper titles and sections**
   * **Attach screenshots of output/plots**
2. **Source code (.zip)**

**NOTE: Please turn in the report and source code as two separate attachments on canvas**

See below ideas for ID3 (in MATLAB): how to compute the entropy

First load the iris data set using the following MATLAB instruction.  See example below:

>> load iris.dat

>> who

Your variables are:

iris

**>> size(iris)**

ans =

   150     5

**>> iris(1,:)**

ans =

    51    35    14     2     1

**>> iris(10,:)**

ans =

    49    31    15     1     1

**>> iris(100,:)**

ans =

    57    28    41    13     2

**>> iris(140,:)**

ans =

    69    31    54    21     3

**[iris(1:5,:); iris(51:55,:); iris(101:105,:)]**

ans =

    51    35    14     2     1

    49    30    14     2     1

    47    32    13     2     1

    46    31    15     2     1

    50    36    14     2     1

    70    32    47    14     2

    64    32    45    15     2

    69    31    49    15     2

    55    23    40    13     2

    65    28    46    15     2

    63    33    60    25     3

    58    27    51    19     3

    71    30    59    21     3

    63    29    56    18     3

    65    30    58    22     3

The attributes are **continuously valued**.  To discretize the attribute values, you can use hist Matlab function. For the iris data set, invoke hist on the first column as follows.

**>> [n,x]=hist(iris(:,1))  % default number of bins is 10**

n =

     9    23    14    27    22    20    18     6     5     6

% n is the frequency of values in each bin

x =

  Columns 1 through 7

    4.4800    4.8400    5.2000    5.5600    5.9200    6.2800    6.6400

  Columns 8 through 10

    7.0000    7.3600    7.7200           % x holds the bin centers

**>> sum(n)**          % check that all the data are assigned to a bin.

ans =

   150

% number of bins n=12

**>> [n,x]=hist(iris(:,1), 12)**

n =

     9    13    23    14    21    15    20    15     8     5     2     5

x =

  Columns 1 through 7

    4.4500    4.7500    5.0500    5.3500    5.6500    5.9500    6.2500

  Columns 8 through 12

    6.5500    6.8500    7.1500    7.4500    7.7500

+++++++++++++++++++++++++++++++++++

% number of bins n=15

**>> [n, x] = hist(iris(:,1), 15)**

n =

  Columns 1 through 12

     5     6    21    13    14    14    10    16    16    15     7     2

  Columns 13 through 15

     5     1     5

x =

  Columns 1 through 7

    4.4200    4.6600    4.9000    5.1400    5.3800    5.6200    5.8600

  Columns 8 through 14

    6.1000    6.3400    6.5800    6.8200    7.0600    7.3000    7.5400

  Column 15

    7.7800

+++++++++++++++++++++++++++++++++++++++++++++++++++

% n = 5 bins

**[n,x]=hist(iris(:,1), 5)**

n =

    32    41    42    24    11

x =  4.6600    5.3800    6.1000    6.8200    7.5400

++++++++++++++++++++++++++++++++++++++++++++++++++

>> % how to compute entropy

>> % P = [p1, ..., pn] is prob distribution

>> % log2P is a vector of log2 values: log2p1, ..., log2pn

>> % p1 log2 p1 + ... + pn log2 pn is the dot product of these

>> % two vectors

>> % example:

>> n

n = 32    41    42    24    11

>> % form p:

>> p = n/sum(n)

p =

    0.2133    0.2733    0.2800    0.1600    0.0733

>> %check that it is a prob

>> sum(p)

ans = 1

>> % form logs

>> logp = log2(p)

logp =  -2.2288   -1.8713   -1.8365   -2.6439   -3.7694

>> % Compute the entropy: - dot product

**>> Entropy= -logp\*p’, or Entropy = -sum(logp .\* p)**

Entropy =  2.2006