Homework 5 Report

Spam Dataset

a) I did not use any additional features for spam.
b) My Decision Tree gave me 83% and Random Forests gave 84%. My Kaggle score was around 74%.
c) (29) <= 0.
(20) <= 0.
(30) <= 0.
(27) <= 0.
(4) <= 0.
(1) <= 0.
(10) <= 0.
(14) <= 0.
(13) <= 0.
(26) <= 4.
(18) <= 0.
(32) <= 1.
(17) <= 0.
(30) <= 4.
(25) <= 1.
(16) <= 0.
(3) <= 0.
(25) <= 0.
(28) <= 0.
(19) <= 0.
(30) <= 3.
(26) <= 0.
(30) <= 2.
(1) <= 0.
(1) <= 0.

d) (29) <= 0. (20 trees)
(20) <= 0. (20 trees)
(30) <= 0. (10 trees)
(17) <= 0. (7 trees)
(32) <= 0. (8 trees)
(7) <= 0. (15 trees)
(1) <= 0. (20 trees)
(26) <= 1. (2 trees)

(26) <= 0. (4 trees)
(26) <= 2. (3 trees)
(14) <= 0. (2 trees)
(11) <= 0. (1 trees)
(27) <= 2. (1 trees)
(31) <= 0. (1 trees)

Census Dataset:

a) I used some external code for the pre-processing step to handle the extra/missing features and their values. Otherwise, no extra features added.
b) My Decision tree gives me about 85% while my random forest did slightly better with 86.3%. My Kaggle score came to 82%.

c) (relationship) <= 1.
(education-num) <= 11.
(capital-gain) <= 5013.
(capital-loss) <= 1740.
(hours-per-week) <= 30.
(age) <= 33.
(education) <= 8.
(age) <= 27.
(native-country) <= 37.
(capital-loss) <= 0. =
(occupation) <= 4.
(capital-gain) <= 3103.
(capital-gain) <= 2407.
(age) <= 28.
(occupation) <= 5.
(hours-per-week) <= 40.
(age) <= 30.
(fnlwgt) <= 55291.
(fnlwgt) <= 105229.
(workclass) <= 5.
(fnlwgt) <= 167319.
(fnlwgt) <= 348152.
(fnlwgt) <= 185216.

d) (relationship) <= 1. (50 trees)
(education-num) <= 12. (24 trees)
(education-num) <= 13. (26 trees)
(fnlwgt) <= 210013. (1 tree)
(fnlwgt) <= 83064. (12 trees)
(age) <= 59. (6 trees)
(age) <= 39. (2 trees)
(occupation) <= 4. (20 trees)

Pruning/ Additional Implementation(s) (PART 5)

- My Decision Tree class is implemented in a way that each node of this class is either a list of the split arguments in the form [index to split on, threshold for split].
- I further have a isLeaf indicator that turns on/off based on where I am in the decision tree.
- Missing values are replaced with {} or 'NaN' in Matlab (external code implementation)
- For bagging I take out 30% of the overall data randomly.
- I speed my my segmentation process by only considering unique values of a column as appropriate thresholds. This sped up the process from an initial run-time of 3 minutes to a mere 2 seconds.
- Impurity criteria is just the information gain function implemented in lecture.
- Whenever I have a confusion, classifier predicts the optimal label to be the mode of the remaining label and makes the decision.
- I make sure that my tree wont classify until 2 criteria are met:
    - More 99% of the remaining labels are the same.
    - The tree will classify regardless of label at a certain depth (around 25 for the tree and 12 for a tree in the forest). It will find the mode of the remaining labels and make a guess.

Random forest techniques are described above. The only modification that I included was considering the depth of the tree as a hyper parameter and tuning to find both optimal depth as well optimal number of bags.

Table of Contents

Splitting my training data into train and validation sets

```
load('training_data.mat')
load('test_data.mat')
load('column_names.mat')

training_data = data(:,1:14);
training_labels = data(:,15);

pick = 0.9*32724;
pick = double(pick);
validation_index = randsample(32724,pick);

val_label = training_labels(validation_index);
val_matrix = training_data(validation_index,:);
rest = setdiff(1:5172,validation_index);
train_matrix = training_data(rest,:);
train_label = training_labels(rest)';
train_label = double(train_label);
```

```
Warning: Integer operands are required for colon operator when used as
 index
```

Making a tree

```
test_matrix = test_data;

%dt = DecisionTree();
%trained_tree = dt.train(train_matrix,train_label);
pred_label = zeros(1,length(test_data));
for i = 1:length(test_matrix)
testing_data = test_matrix(i,:);
pred_label(i) = trained_tree.predict(testing_data,column_names);
end
```

```
Undefined variable "trained_tree" or class "trained_tree.predict".

Error in HW05 (line 29)
pred_label(i) = trained_tree.predict(testing_data,column_names);
```

Validation Testing

```
%accuracy = sum(pred_label==val_label)/length(val_label);
%display(accuracy,'The Accuracy is');
```

Spam test

```
Submission_matrix = [(1:length(test_data))', pred_label'];
```

Random Forests

```
%training_labels = double(training_labels');
pred_label = zeros(length(val_label),1);
for index = 1:20
    [data, label] = bagging(train_matrix,train_label');
    df = DecisionTree();
    classifier = df.train(data,label);
    pred_label = zeros(1,length(val_label));
    for i = 1:length(val_matrix)
        testing_data = val_matrix(i,:);
        pred_label(i,index) =
 classifier.predict(testing_data,column_names);
    end
end

predictions = mode(pred_label');
```

Submission

```
Submission_matrix_f = [(1:length(test_data))', predictions'];
%accuracy = 100*sum(predictions==val_label)/length(val_label);
%display(accuracy,'The Accuracy is (in %)');
```

Published with MATLAB® R2015b

Making a DecisionTree Class

```
classdef DecisionTree
    properties
        node = NaN;
        left = NaN;
        right = NaN;
        isLeaf = false
        depth =0;
    end
    methods

        function obj = train(obj,data,labels)
            obj.depth = obj.depth+1;
            if sum(labels) >= 0.99*length(labels)
                obj.isLeaf = true;
                obj.node = 1;
                obj.left = NaN;
                obj.right = NaN;
                return
            else if sum(labels) <= 0.01*length(labels)
                    obj.isLeaf = true;
                    obj.node = 0;
                    obj.left = NaN;
                    obj.right = NaN;
                    return

                 else
            [index,threshold] = Segmentor(data,labels);
            obj.node = [index,threshold];
            if obj.depth <= 25
            obj.right = obj.train(data(data(:,index) >
 threshold,:),labels(data(:,index) > threshold));
            obj.left = obj.train(data(data(:,index) <=
 threshold,:),labels(data(:,index) <= threshold));
            else
                obj.isLeaf = true;
                obj.node = mode(labels);
                obj.left = NaN;
                obj.right = NaN;
            end
                end
            end
        end

        function label = predict(obj,data,column_names)
            if obj.isLeaf == true
                label = obj.node;
                display(label,'The assigned Label');
            else
                split_value = obj.node;
                index = split_value(1);
```

```
                threshold = split_value(2);
                formatSpec= '(%s) <= %d. \n';
                input  = column_names{index};
                dis = sprintf(formatSpec,input,threshold);
                disp(dis);
                if data(:,index) <= threshold
                    obj = obj.left;
                    label = obj.predict(data(data(:,index) <=
 threshold,:),column_names);
                else
                    dt = obj.right;
                    label = dt.predict(data(data(:,index) >
 threshold,:),column_names);
                end
            end
        end


    end
end
```

```
ans =

  DecisionTree with properties:

      node: NaN
      left: NaN
     right: NaN
    isLeaf: 0
     depth: 0
```

Published with MATLAB® R2015b

Segmentor

```
function [indices, threshold,tempo] = Segmentor(data,labels)
            [~, wid] = size(data);
            tempo = zeros(wid,1);
            return_threshold = zeros(wid,1);
            for index = 1:wid
                select_col = data(:,index);
                unique_col = unique(select_col);
                temp = ones(length(unique_col),1);
                for i = 1:length(unique_col)
                    left_label_hist = labels(select_col >
 unique_col(i));
                    right_label_hist = labels(select_col <=
 unique_col(i));
                    temp(i) =
 Impurity(left_label_hist,right_label_hist);
                end
                [tempo(index), index_val] = max(temp);
                return_threshold(index) = unique_col(index_val);
            end
            [~, indices] = max(tempo);
            threshold = return_threshold(indices);
        end
```

```
Not enough input arguments.

Error in Segmentor (line 3)
            [~, wid] = size(data);
```

Published with MATLAB® R2015b

Impurity using info-gain

```
function imba = Impurity(left_label_hist,right_label_hist)
            %Entropy for the left child
            len_l = length(left_label_hist);
            p_l = sum(left_label_hist)/len_l;
            if (p_l == 0) || (p_l ==1)
                e_l = 0;
            else
            q_l = 1-p_l;
            e_l = -1*(p_l*log2(p_l) + q_l*log2(q_l));
            end
            %Entropy for the right child
            len_r = length(right_label_hist);
            p_r = sum(right_label_hist)/len_r;
            if (p_r == 0) || (p_r ==1)
                e_r = 0;
            else
            q_r = 1-p_r;
            e_r = -1*(p_r*log2(p_r) + q_r*log2(q_r));
            end
            %Entropy for the parent
            len = len_l + len_r;
            p_p = (sum(left_label_hist) + sum(right_label_hist))/len;
            q_p = 1 - p_p;
            e_p = -10 * (p_p*log2(p_p) + q_p*log2(q_p));
            weighed_ent = (len_l*e_l + len_r*e_r)/len;
            imba = e_p - weighed_ent;
            %imba = abs(imba);

        end
```

```
Not enough input arguments.

Error in Impurity (line 4)
            len_l = length(left_label_hist);
```

Published with MATLAB® R2015b

Bagging implementation

```
function [r_data, r_labels] = bagging(data,labels)
    [len, wid] = size(data);
    sample_count = 0.3*len;
    sample_count = double(sample_count);
    collab = [data,labels];
    index = randsample((1:len),sample_count);
    collab = collab(index,:);
    r_data = collab(:,1:wid);
    r_labels = collab(:,wid+1);
```

```
Not enough input arguments.
```

```
Error in bagging (line 4)
    [len, wid] = size(data);
```

Published with MATLAB® R2015b

Table of Contents

EXTERNAL FUNCTION TO READ .cvs FILES AND CONVERT TO .mat

```
function [data,column_names,string_conversions] =
 string_CSV_read(filename)

% string_CSV_read This function reads the data contain in a .csv file
 and
%  transform it for its use. The function returns the next values:

%   data: contains the readed data. The columns that contains nominal
%   values are transformed into numerical ones.
%   column_names: if the file contains a header, this variable saves
%   the name of each column(*).
%   string_conversions: this variable contains the nominal values of
%   the columns that have been transformed. If the column has
%   numerical value, it contains {}. Otherwise, the nominal values
%   are saved in column position.
%   (*)NOTE: In case of a entire nominal dataset with no header, the
 first
%   example can be confused with header.
%
%
%   Example: the file contains the following data:
%           A,B,C,D
%    1,2,Sun,YES
%    3,1,Rain,YES
%    3,5,Sun,NO
%
%     Then, the function returns:
%         - data = [ 1 2 2 2
%                    3 1 1 2
%                    3 5 2 1]
%         - column_names = {'A','B','C', 'D'}
%         - string_conversions = {{}
%                                 {}
%                                 {'Rain' 'Sun'}
%                                 {'NO' 'YES'}}

% Created by:
% Pedro L#pez Garc#a (Phd. Student, University of Deusto,Bilbao)
```

```
% Enrique Onieva Caracuel (PostDoctoral Researcher & Project Manager,
 University of Deusto, Bilbao).
%   Twitter: @EnriqueOnieva
% Research Gate: https://www.researchgate.net/profile/Enrique_Onieva
%                https://www.researchgate.net/profile/
Pedro_Lopez_Garcia

    fid = fopen(filename,'r');
    filestrings = [];
```

```
Not enough input arguments.

Error in string_CSV_read (line 43)
    fid = fopen(filename,'r');
```

Read the file

```
    while 1
        line = fgetl(fid);
        if ~ischar(line),break,end
        tmp = regexp(line,'([^ ,:]*)','tokens');
        str = cat(2,tmp{:});
        filestrings = cat(1,filestrings,str);
    end
```

Take the data, number of rows and number of columns

```
    data = str2double(filestrings);
    nrows = size(data,1);
ncolumns = size(data,2);
```

Is There a header in the file?

```
column_names = {};
    if isnan(data(1,:)) == ones(1, ncolumns)
        column_names = filestrings(1,:);
        data = data(2:end, :);
 nrows=nrows-1;
    end
```

Dictionary creation

```
    string_conversions = {};
    for i = 1: ncolumns
        % if the columns contains nominal values
        if isnan(data(:,i)) == ones (nrows, 1)
            %Take these values and save it in string_conversions
            names = unique(filestrings(2:end,i));
            string_conversions{i} = names;
```

```
            % Transform the names into numerical value
            for j = 1: numel(names)
               index = find(strcmp(filestrings(:,i), names{j}));
               for k = 1: size(index, 1)
                   data(index(k)-1, i) = j;
               end
            end
        else
            string_conversions{i} = {};
        end
    end
```

Published with MATLAB® R2015b