CPSC 340: Assignment1

Yiwei Hou(84435156)
Qiong Zhang(85896158)

1 Data Exploration

1.1 Summary Statistics

1. The minimum, maximum, mean, median and mode of all value across the dataset is presented in Table 1.

Table 1: Summary statistics

Minimum	Maximum	Mean	Median	Mode
0.3520	4.8620	1.3246	1.1590	0.7700

2. The 10%, 25%, 50%, 75%, and 90% quantiles across the dataset are presented in Table 2.

Table 2: Quantiles across the dataset

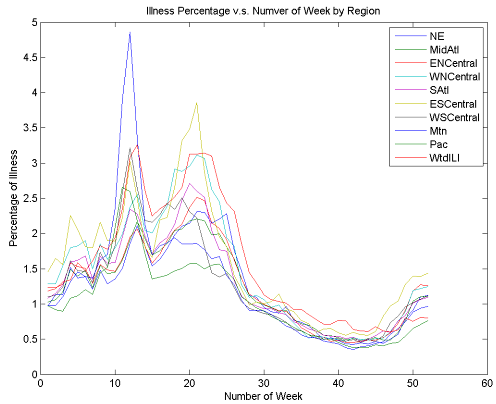
10%	25%	50%	75%	90%
0.3520	4.8620	1.3246	1.1590	0.7700

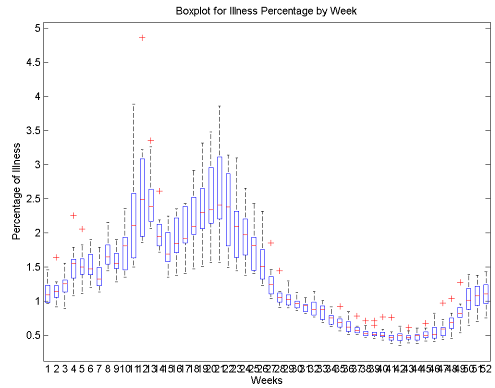
3. The regions with the highest mean is 'WtdILI' and the region with the lowest mean is 'Pac'. The region with the highest variance is 'Mtn' and the one with the lowest variance is 'Pac'.
4. The pair of regions with the highest correlation is 'MidAtl' and 'ENCentral'. The pair of regions with the lowest correlation is 'NE' and 'Mtn'.

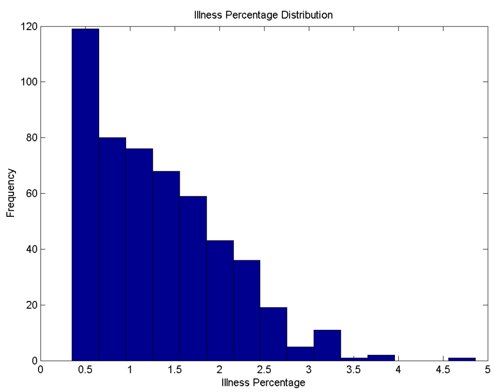
Mean is a meaning estimate of the most common value for continuous data. **Explanation**: Mode is a reliable estimate of the most common value for categorical data. For continuous data, as we can take the expectation of a variable as the most common value and the expectation for a continuous data can be estimated by the mean.

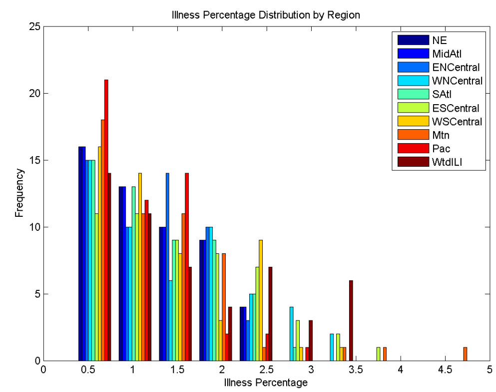
1.2 Data Visualization

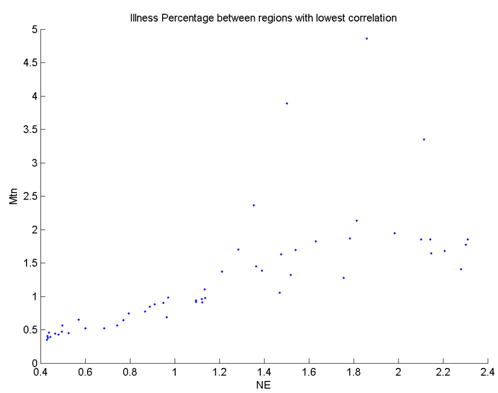
The six plots for this question are given as follows(the order is from left to right, top to bottom):

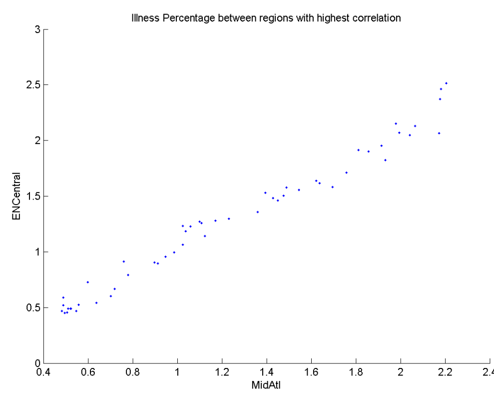

Illness Percentage v.s. Numver of Week by Region
Percentage of Illness
Number of Week
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI


Boxplot for Illness Percentage by Week
Percentage of Illness
Weeks


Illness Percentage Distribution
Frequency
Illness Percentage


Illness Percentage Distribution by Region
Frequency
Illness Percentage
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI


Illness Percentage between regions with lowest correlation
Mtn
NE


Illness Percentage between regions with highest correlation
ENCentral
MidAtl

2 Decision Trees

2.1 Decision Stump Implementation

```
function [ model ] = decisionStump( X,y )
% [model] = decisionStumpEquality(X,y)
%
% Fits a decision stump that simply test for an inequality

[n,d] = size(X);

% Address the case where we do not split
y_mode = mode(y);
minError = sum(y ~= y_mode);
splitVariable = [];
splitValue = [];
splitSat = y_mode;
splitNot = [];

if any(y ~= y(1)) % First check that all labels are not equal

    for j = 1:d
        for i = 1:n
            % Choose value to equate to
            value = X(i,j);

            % Find most likely class when rule is satisfied
            %   and not satisfied
            y_sat = mode(y(X(:,j)>value));
            y_not = mode(y(X(:,j)<=value));

            % Make predictions
            yhat = y_sat*ones(n,1);
            yhat(X(:,j)<=value) = y_not;

            % Compute error
            error = sum(yhat ~= y);

            % Update best rule
            if error < minError
                minError = error;
                splitVariable = j;
                splitValue = value;
                splitSat = y_sat;
```

```
                    splitNot = y_not;
                end
            end
        end
end

model.splitVariable = splitVariable;
model.splitValue = splitValue;
model.splitSat = splitSat;
model.splitNot = splitNot;
model.predict = @predict;
end

function [y] = predict(model,X)
[t,d] = size(X);

if isempty(model.splitVariable)
    y = model.splitSat*ones(t,1);
else
    y = zeros(t,1);
    for i = 1:t
        if X(i,model.splitVariable)>model.splitValue
            y(i,1) = model.splitSat;
        else
            y(i,1) = model.splitNot;
        end
    end
end
end
```

The updated error you by using inequalities is 0.25.

2.2 Construction Decision Trees

The if/else statement for one training example:

```
% Since we only get the predicted value for only one training
    example
yhat=0;

% Get the feature and threshold value for decision stump at depth
    1
j=model.splitModel.splitVariable
value=model.splitModel.splitValue

```

```
% Get the feature and threshold value for decision stump for first split
j1=model.subModel1.splitVariable
value1=model.subModle1.splitValue

% Get the feature and threshold value for decision stump for second split
j0=model.subModel0.splitVariable
value0=model.subModle0.splitValue


if X(:,j)>value
     if X(:,j1)>value1
     yhat=model.subModel1.splitSat;
     else
     yhat=model.subModel1.splitNot;
     end
else
     if X(:,j0)>value0
     yhat=model.subModel0.splitSat;
     else
     yhat=model.subModel0.splitNot;
     end
end
```

When the depth reaches 4, the error is 0.11 and keeps the same as the depth increases. The decision tree stops making more splits after depth 5. One possible reason why the training error stops decreasing is when a certain branch has observations all with the same label, there is no split to increase the classification accuracy. But if we use information gain, the algorithm will continue split the node into two leaf nodes until all data are classified correctly.

2.3 Costing of Fitting Decision Tress

$O(n^2 d \log n)$. We have to at most fit $\sum_{i=1}^{m} 2^{i-1} = 2^m - 1$ decision stumps in a depth m decision tree. When the number of leaves after mth step equal to the number of observations, i.e. $2^m = n$, then the decision tree actually stop growing, thus the cost would $O(nd \log n(2^m - 1)) = O(n^2 d \log n)$.

3 Training and Testing

3.1 Training and Testing Error Curves

Both training error and test error decreases as the depth increases and training error goes to zero when depth is greater than 10. And test error also stabilize after depth reaches 10.

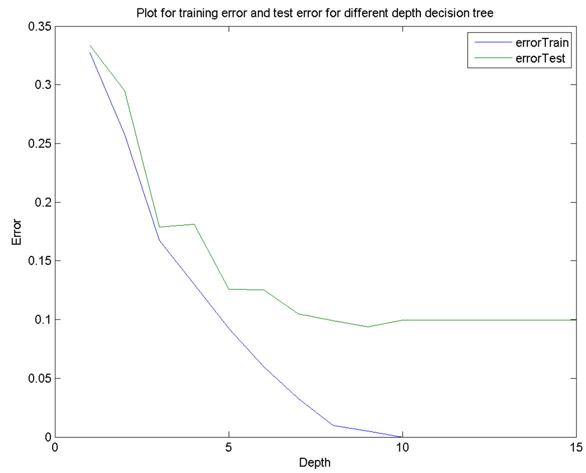


Figure 1: Training and Testing Error Curves

3.2 Validation Set

We could pick depth to be 3,6 or 9 with validation error 0.1450 if we minimize the validation set error. If we switch the training and validation set, the depths we pick change to 5,6 or 8 with validation error 0.1250. Note that when fit a deep decision tree, e.g. with depth 8 and depth 9, we may come across the problem of over-fitting. And the results show if we use validation set to choose the depth of the tree, it is sensitive to the validation set.

A better choice is to use Cross Validation to estimate the depth of decision tree. Since CV gives an unbiased estimate of the test error, we would select the depth of tree as the depth minimizes CV.

4 Naive Bayes

4.1 Bayes Rule for Drug Testing

$$
\begin{aligned}
P(D=1|T=1) &= \frac{P(D=1,T=1)}{P(T=1)} = \frac{P(D=1|T=1)P(D=1)}{1-P(T=0)} \\
&= \frac{P(D=1|T=1)P(D=1)}{1-P(T=0|D=1)P(D=1)-P(T=0|D=0)P(D=0)} \\
&= \frac{0.99*0.001}{1-0.01*0.001-0.99*0.999} = 0.0902
\end{aligned}
$$

4.2 Naive Bayes by Hand

(a) $P(y=1) = 6/10 = 0.6$; $P(y=0) = 4/10 = 0.4$
(b) $P(x_1 = 1|y=1) = 3/6 = 0.5$; $P(x_2 = 1|y=1) = 4/6$;
$P(x_1 = 1|y=0) = 4/4 = 1$; $P(x_2 = 1|y=0) = 1/4 = 0.25$
(c) Based on the bayes formula, we have

$$\begin{aligned} P(y=1|\hat{x}=[1,1]) &= c \times P(\hat{x}_1 = 1|y=1) \times P(\hat{x}_2 = 1|y=1)P(y=1) \\ &= c * 3/6 * 4/6 * 6/10] \\ &= c\frac{1}{5} \end{aligned}$$

and

$$\begin{aligned} P(y=0|\hat{x}=[1,1]) &= c \times P(\hat{x}_1 = 1|y=0) \times P(\hat{x}_2 = 1|y=1)P(y=0) \\ &= c * 1 * 1/4 * 4/10] \\ &= c\frac{1}{10} \end{aligned}$$

Since $c > 1$, we have $P(y=1|\hat{x}=[1,1]) > P(y=0|\hat{x}=[1,1])$. Thus by naive Bayes model, the most likely label for the test example would be 1.

4.3 Naive Bayes Implementation

Befor implementaion, if we run the code, we get:

1. Validation error with decision tree: 0.36
2. Validation error with naive Bayes: 0.66

The code for calculate *p_xy*

```
p_xy = (1/2)*ones(d,2,k);
for j=1:d
    for c=1:k
        p_xy(j,1,c)=full(sum(X(y==c,j)==1)/counts(c));
        p_xy(j,2,c)=1-p_xy(j,1,c);
    end
end
```

After modify *p_xy* correctly, the updated validation error is 0.19.

4.4 Runtime of Naive Bayes for Discrete Data

For classifying one example with a naive bayes model, we have to maximize

$$\prod_{j=1}^{d} P(X_{ij}|y_i = b)P(y_i = b)$$

where $b \in \{1 : k\}$. We have to get the probability for k class labels, and the cost of calculating each probability is $O(d)$, thus the complexity of classifying one example is $O(kd)$. Since we have t examples, the total cost of the classifying would be $O(kdt)$.