Assignment 1

Zitong Wang 40883150
CPSC 340: Data Mining and Machine Learning

January 12, 2019

1 Linear Algebra Review

1.1 Basic Operation

1. 14.
2. 0.
3. $\begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}$, .
4. $\sqrt{5}$.
5. $\begin{bmatrix} 6 \\ 5 \\ 7 \end{bmatrix}$, .
6. 19.
7. $\begin{bmatrix} 11 & 10 & 10 \\ 10 & 14 & 10 \\ 10 & 10 & 14 \end{bmatrix}$.

1.2 Matrix Algebra Rules

1. *true*.
2. *true*.
3. *false*.
4. *true*.
5. *false*.
6. *true*.
7. *false*.
8. *true*.
9. *true*.
10. *false*.

2 Probability Review

2.1 Rules of Probability

1. 5.
2. 0.55.
3. $\frac{11}{12}$.

2.2 Bayes Rule and Conditional Probability

1. $0.97 * 0.0001 + (1 - 0.0001) * 0.0001 = 0.0010969$
2. False positive
3. $\frac{P(T=1|D=1)*P(D=1)}{P(T=1)} = 0.001$
4. No since the calculated probability is too small to conclude that a person given test positive is a drug user
5. Increase the probability of false positive

3 Calculus Review

3.1 One-variable Derivatives

1. $\frac{\partial f}{\partial x} = 6x - 2$
2. $\frac{\partial f}{\partial x} = 1 - 2x$
3. $\frac{\partial f}{\partial x} = 1 - \frac{1}{p(x)} * p'(x) = 1 - (1 + exp(-x)) * \frac{exp(-x)}{(1+exp(-x))^2} = p(x)$.

3.2 Multi-variable derivatives

1. $\nabla f(x_1, x_2) =< 2x_1 + exp(x_1 + 2x_2), 2 * exp(x_1 + 2x_2) >$.
2. $\nabla f(x_1, x_2, x_3) =< \frac{exp(x_1)}{z}, \frac{exp(x_2)}{z}, \frac{exp(x_3)}{z} >$.
3. $\nabla f = a^T$.
4. $\nabla f(x_1, x_2) =< 2x_1 - x_2, 2x_2 - x_1 >$.
5. $\nabla f(x_1, x_2, ..., x_d) =< x_1, x_2, ..., x_d >$

3.3 Optimization

1. $\frac{14}{3}$.
2. $\frac{1}{4}$.
3. 0.

4. 0.5.
5. 2.835.
6. $(1, 0.607)$

3.4 Deviation of Code

Please see python file grads.py

4 Algorithms and Data Structures Review

4.1 Trees

1. 6.
2. 6.

4.2 Common Runtimes

1. $O(n)$.
2. $O(log(n))$.
3. $O(1)$.
4. $O(d)$.
5. $O(d^2)$.

4.3 Running times of code

1. $O(N)$.
2. $O(N)$.
3. $O(1)$.
4. $O(N^2)$.

5 Data Exploration

5.1 Summary Statistics

5.1.1

1. 0.35200000000000004
2. 4.862
3. 1.3246250000000002

4. 1.1589999999999998
5. 0.77

5.1.2

1. 5%: 0.46495000000000003
2. 25%: 0.718
3. 50%: 1.1589999999999998
4. 75%: 1.81325
5. 95%: 2.624049999999999

5.1.3

Lowest Mean is Pac, highest mean is WtdILI and highest variance is Mtn lowest variance is Pac.

5.2 Data Visualization

1. Plot D: This is true because plot D is histogram in the first place and each 'big' rectangle were decomposed into several 'small' rectangles which corresponding to each column.
2. Plot C: Since there is no category for output of a matrix, therefore each value will be best performed by histogram.
3. Plot B: Since B is the only boxplot here while each data was differed by week, also it shows min, max, mean, upper bound as well as lower bound for distribution of each region,therefore it fits the description.
4. Plot A: Since it records illness data by week to week, we can see the trend of illness by percentage and weeks via this plot.
5. Plot F: Since this is a scatterplot and the trend of the data set in this plot can be perfectly described by a linear best fit line which meaning this data set has small variance and high correlation.
6. Plot E: Since this scatterplot has more outliers and less condensed comparing to E there for this plot has higher variation as well as lower correlation.

6 Decision Trees

6.1 Splitting rule

Categorical features.

6.2 Decision Stump Implementation

Decision Stump with inequality rule error: 0.253. Please refer to figure 1 for the related plot.

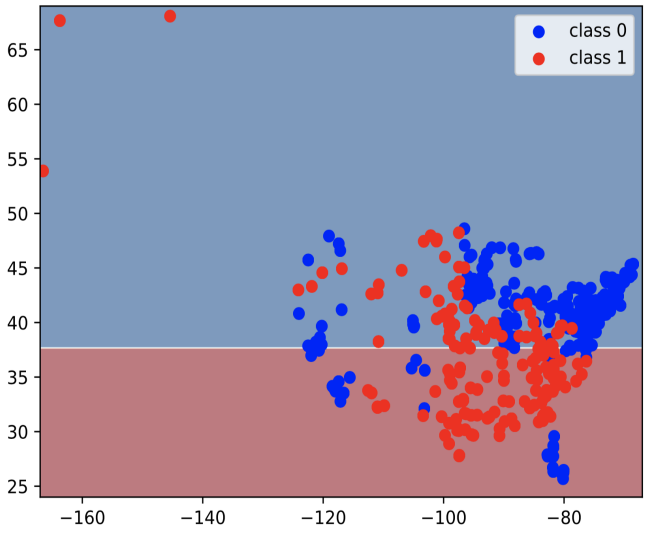


Figure 1: Classification boundary plot for DecisionStumpErrorRate

6.3 Decision Stump Info Gain Implementation

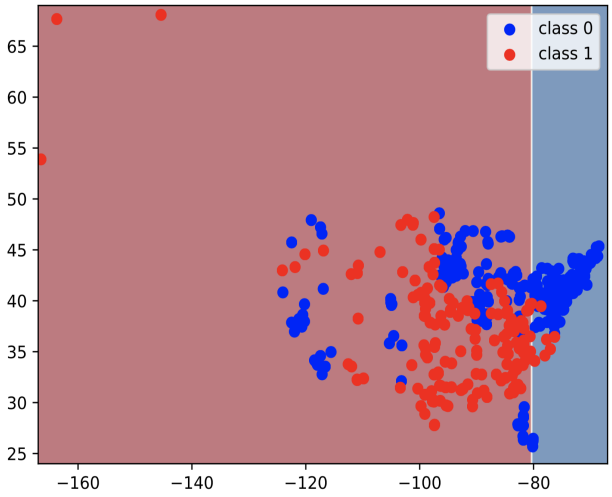


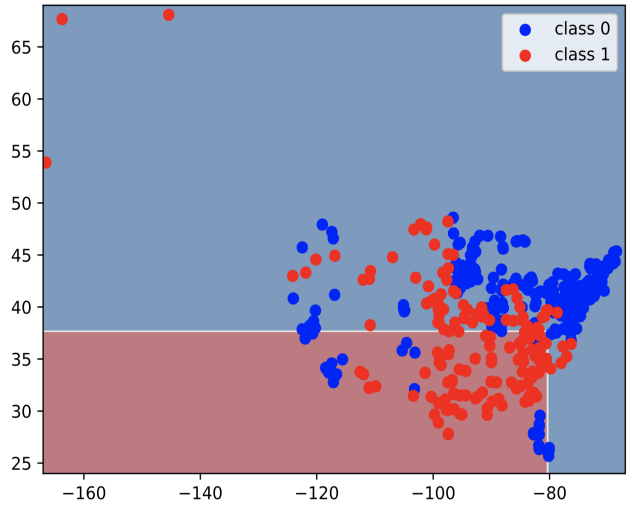
Figure 2: Classification Boundary Plot for DecisionStumpInfoGain

Decision Stump with info gain rule error: 0.325. Please refer to figure 2 for the related plot.

6.4 Constructing Decision Trees

6.4.1

This time, error = 0.242 smaller than all cases above.



Classification Boundary Plot for Constructing Decision Tree

6.4.2 6.4 code

```
def simple_predict(self, X):

        if (self, X[0, 0]) > -85.379384:
            if(self, X[0, 1]) > 29.658295:
                y = 1
            else:
                y = 0

            elif(self, X[0, 1]) > 39.661223:
                y = 1
            else:
                y = 0

        return y
```

6.5 Decision Tree Training Error

In this Classification error vs Depth of Tree diagram, Decision Stump error rate has the lowest initial classification error which is around 0.25 and it keeps decreasing as depth of tree goes down. It reaches depth 4 where classification error keep unchanging around 0.1 but depth of tree keeps still going down. The reason of this is because each threshold in decision stump algorithm only pay attention to single one component of the input vector, as tree go further (beyond 4) the classification error will therefore stay the same.

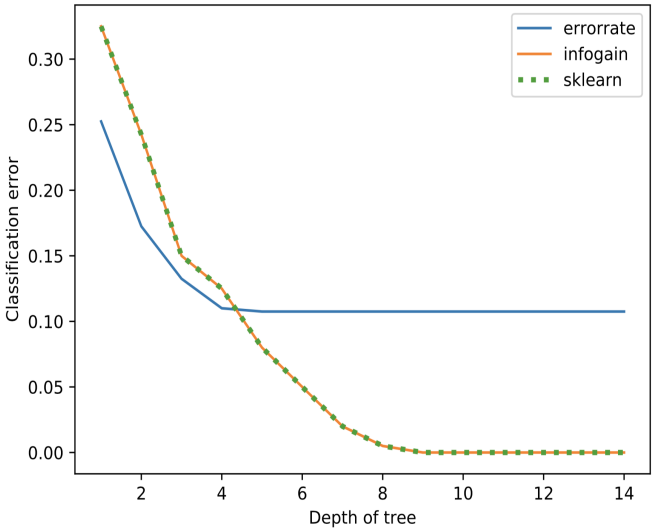


Figure 3: Classification Error Versus Depth of Tree

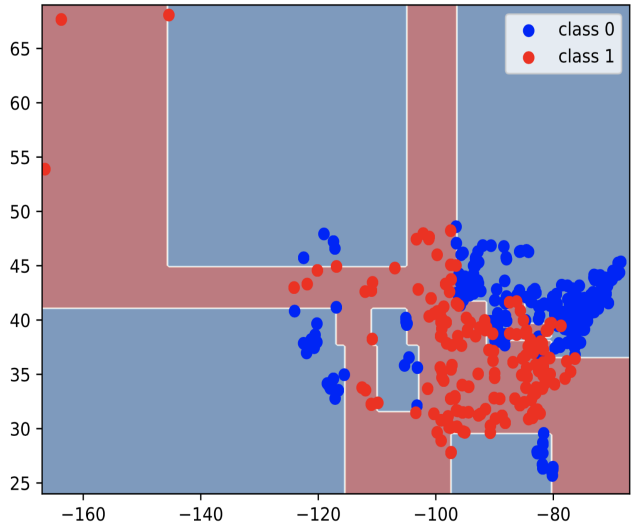


Figure 4: Classification Boundary Plot for max depth = 14

For DecisionStumpInfoGain, it initially has higher classification than the previous one but as tree goes further, its classification error would drop down to nearly zero since as tree goes down information for entropy goes up where accuracy of prediction will also go up causing classification error goes down.

Sklearn has the exact same trend as infoGain but it is discrete on the diagram, this is because decision tree classifier classifies information at each state using entropy function as infoGain, therefore it would have same performance as infoGain while its output is discrete and runtime is substantially faster.

6.6 Comparing implementation

No, they are probably not the same. If we sampled another set of data while those 2 algorithms still produce same curve of Classification error rate and tree depth, we should be more confident to state those 2 algorithms' implementation are probably equivalent.

6.7 Cost of Fitting Decision Trees

For the cost of fitting decision tree, initially there are indeed $2^m - 1$ number of stumps we need to go over, but as we've been mentioned in the question as well as the hint that provided: we should consider every step as greedy recursive and we don't have to go every node as we splitting the tree. If the tree depth is m, as we splitting data, every time we found satisfied data we can just move on to the next depth such that the tight bound for the cost should be $O(mndlog(n))$ instead of initially assumed $O(2^m ndlog(n))$.

Remark: All source code for questions in this chapter were implemented in the code folder along with my submission.