CPSC 340 Assignment 1

1 Data Exploration

1.1 Summary Statistics

1. Minimum: 0.352
2. Maximum: 4.862
3. Mean: 1.325
4. Median: 1.159
5. Mode: 0.770
6. 5th percentile: 0.465
7. 25th percentile: 0.718
8. 50th percentile: 1.159
9. 75th percentile: 1.813
10. 95th percentile: 2.624
11. Highest mean is in WtdILI and lowest mean is in Pac.
12. Highest variance is in Mtn and lowest variance is in Pac.

The mode is not the most reliable estimate of the most common value, since the values have to match exactly for it to count. Since values are continuous, a better way could be to plot a histogram of the values, and observe the frequency of the data from there.

1.2 Data Visualization

The figure contains the following plots, in a shuffled order:

1. D; shows the frequency of illness percentages per region
2. C; it shows the frequency of each illness percentage in the dataset
3. B; it shows a boxplot for each week with the distribution of data
4. A; it shows the change of illness percentages of each region from 0 to 50 weeks
5. F; it is a scatterplot with points close to one another on a straight line
6. E; it is a scatterplot with points not grouped onto a line (points are scattered around more randomly)

2 Decision Trees

2.1 Splitting rule

Features which have discrete values, eg. 0, 1, 2, 3, etc

2.2 Decision Stump Implementation

Error from using inequalities: 0.255

2.3 Constructing Decision Trees

Work done in code

2.4 Decision Tree Training Error

Using the model from sklearn, the error percentage of prediction decreases as the depth of the tree increases, plateau-ing when the depth of the tree is approximately ¿9. Similarly, using my model, the error percentage of prediction decreases as the depth of the tree increases. However, the performance plateau occurs much earlier at a depth of approximately 5. At depths ¿5, the classification error does not seem to deviate.

This could be because the sklearn model is much more sophisticated than my model and thus provides better accuracy at bigger depths.

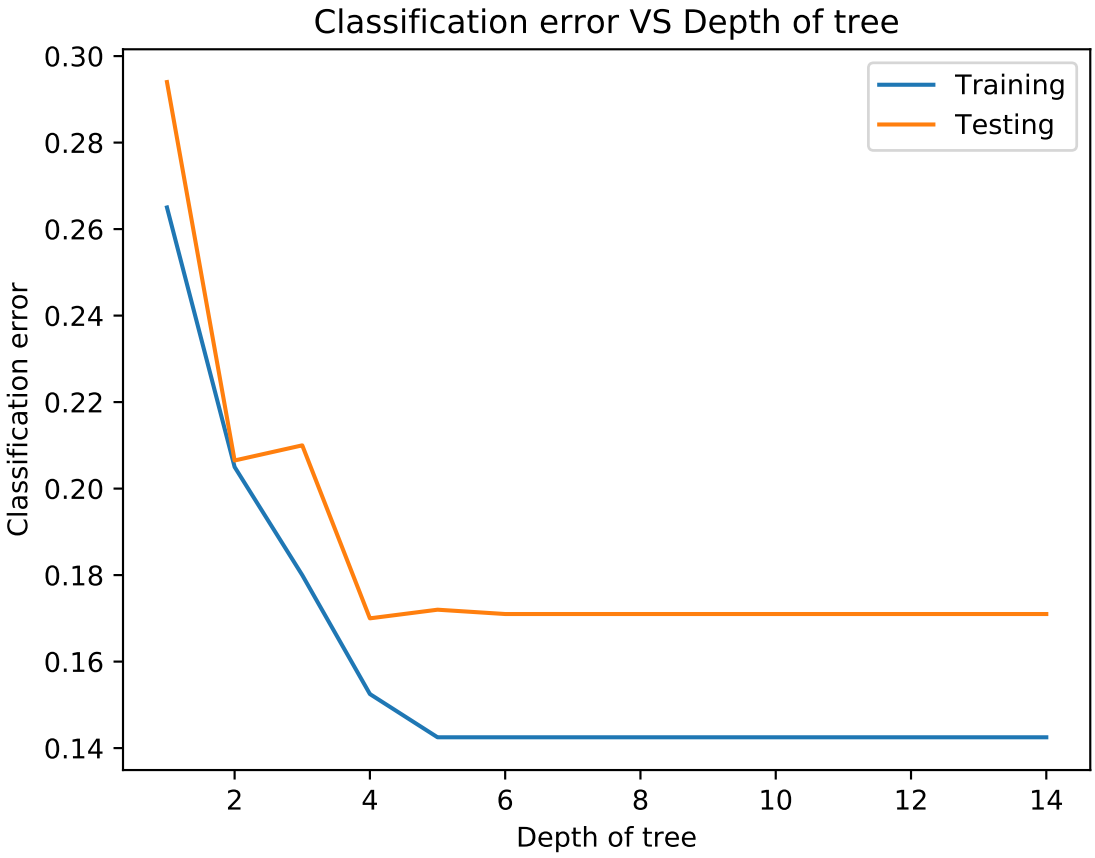
2.5 Cost of Fitting Decision Trees

O(m n d logn)

Explanation: Cost of fitting a decision stump = O(nd logn). Assuming the stump splits into 2 submodels A and B (where A + B = n), the cost of fitting the decision stumps at the submodels = O(Ad logA) + O(Bd log B) which is approximately O(nd logn). Thus, cost is O(mnd log n) for a tree of depth m.

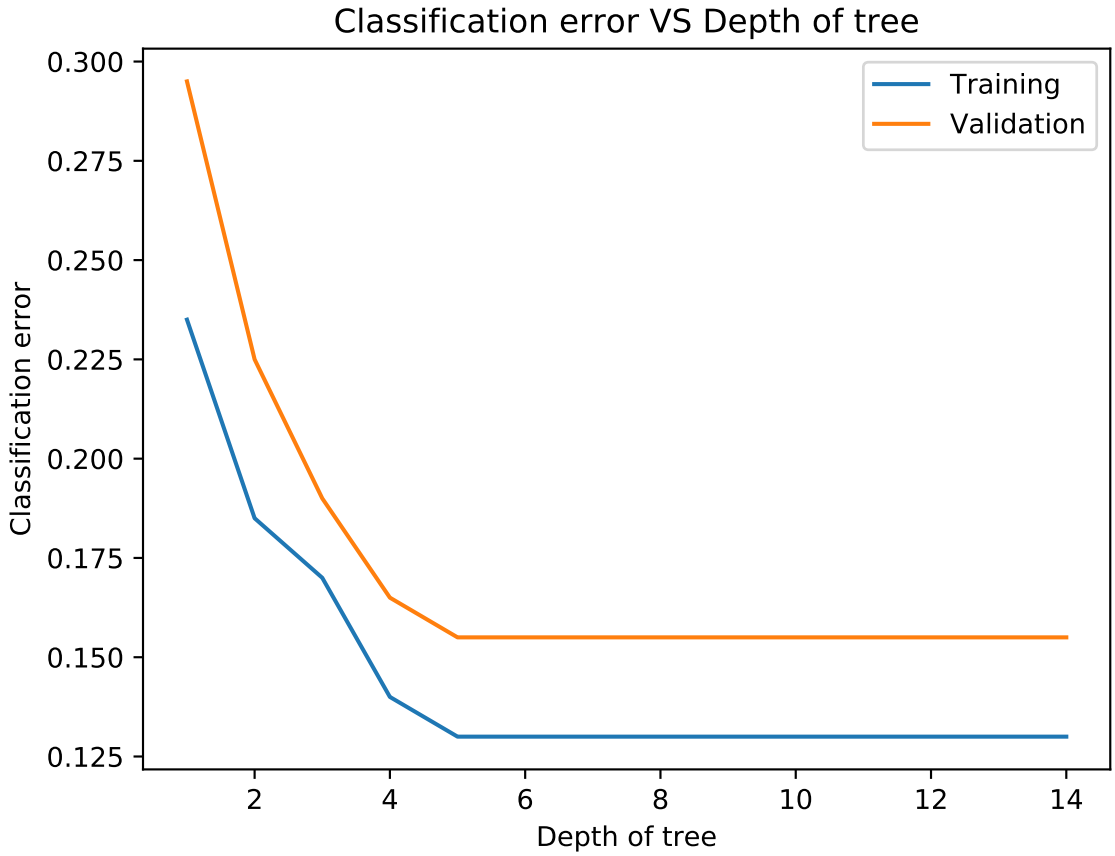
3 Training and Testing

3.1 Training and Testing Error Curves



As depth of the tree increases, both training and testing classification error decrease. However, while training error stays relatively constant for depths approximately ¿=5, testing error stays relatively constant for depths approximately ¿= 4. Furthermore, there seems to be slight increases in testing error when depth increases from 2 to 3, as well as from 4 to 5.

3.2 Validation Set

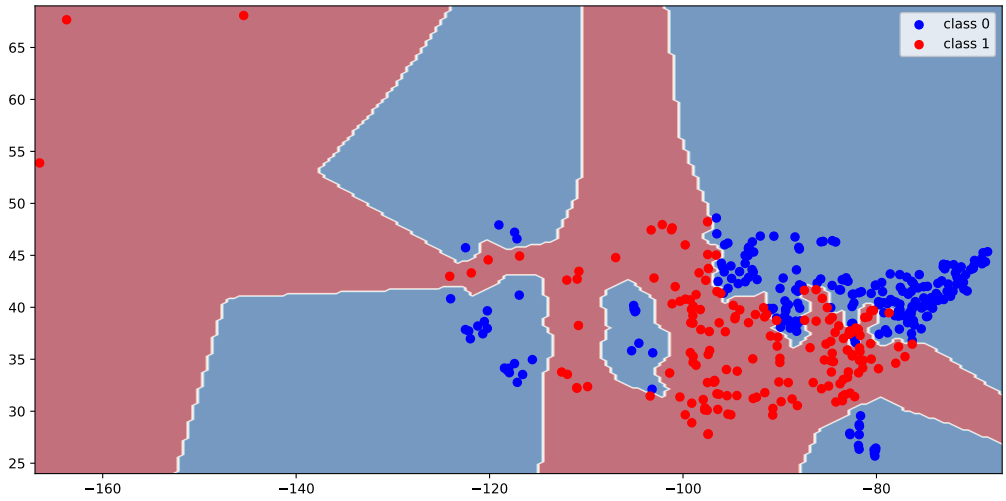


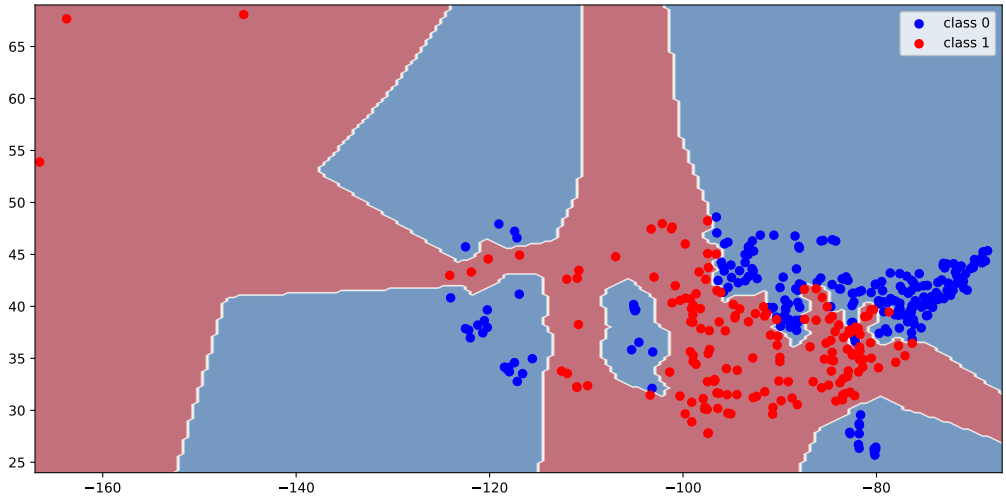
As shown, a decision tree depth of 5 would minimize the validation set error. This value does not change if the training and validation sets are switched. To estimate the depth more reliably, k-fold cross validation could be used.

4 K-Nearest Neighbours

4.1 KNN Prediction

1. Predict function inside knn.py
2. For k = 1, training error is 0 and test error is 0.065. For k = 3, training error is 0.028 and test error is 0.066. For k = 10, training error is 0.072 and test error is 0.097. These numbers are much lower than the training and test errors obtained using the decision tree.
3. My KNN Plot, followed by KNeighborsClassifier Plot

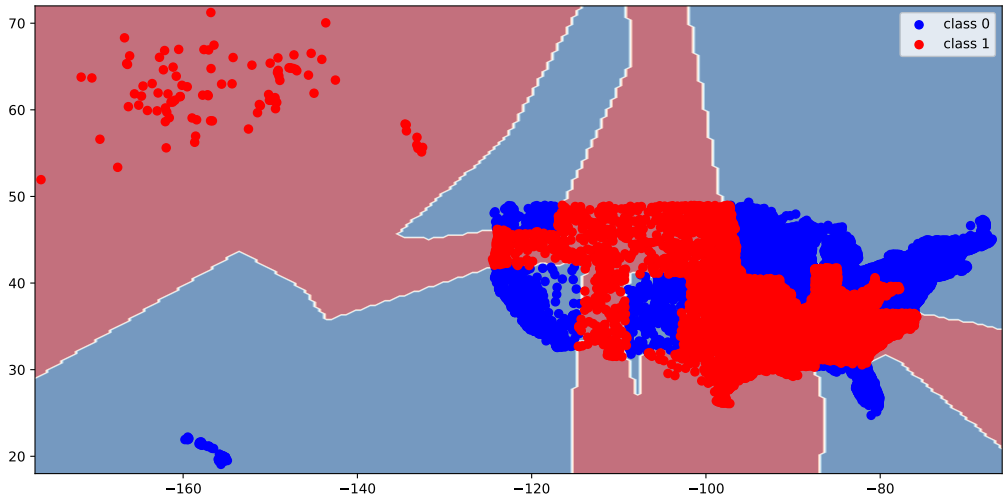




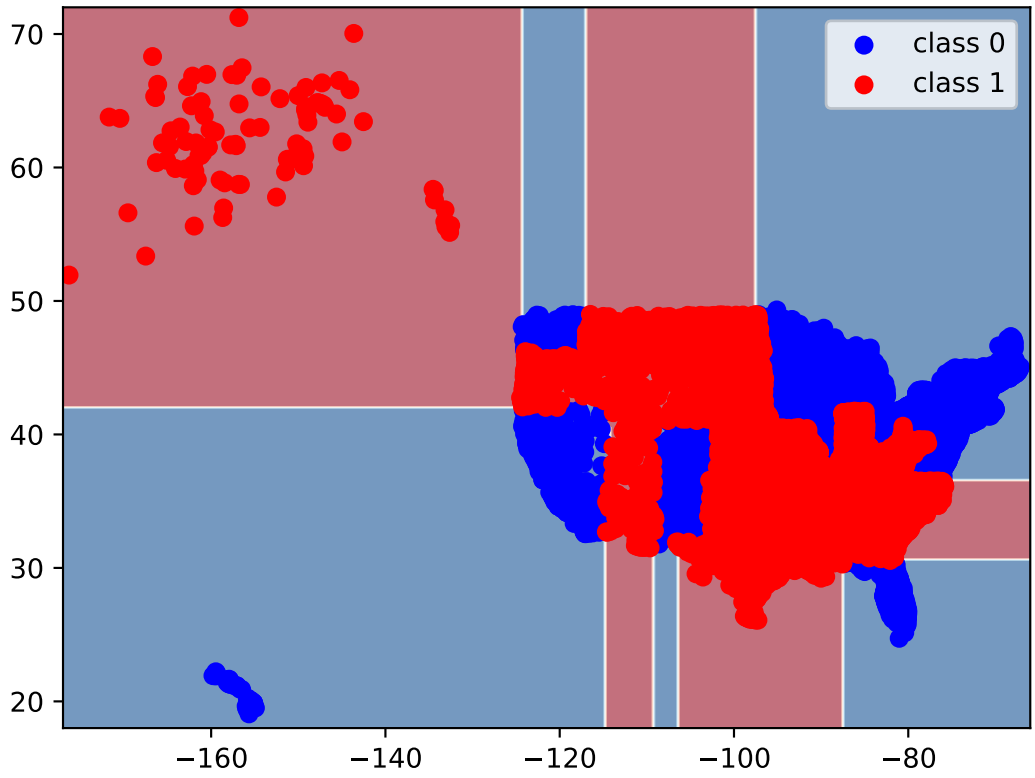
4. Training error is 0 for k = 1 because training had been done on that example before, and thus a match will always be found that is a distance of 0 away from the test data. Thus, it will always be labelled correctly.
5. Create a validation set and use this to do k-fold cross validation to obtain the best k.

4.2 Condensed Nearest Neighbours

1. CNN took 3.546399 seconds to make a prediction. Using KNN for this dataset, I stopped the process after 25s.
2. Training error is 0.008 and testing error is 0.018. Number of variables in the subset is 457.
3. CNN Plot:



1. Since the dataset is only a subset of all the data provided, there will be some error introduced when basing predictions on only 1 neighbour.
2. O(dts)
3. Training error is 0.138 and testing error is 0.210. Since there are only 30 samples in the subset of data used by CNN, underfitting occurs and the model is unable to accurately predict the training and test data.
4. DecisionTreeClassifier took 0.032084 seconds. Training error: 0, Testing error: 0.012. Overall, I would prefer using decision trees for this data set as it works well with low testing error and is very fast.
5. DecisionTreeClassifier Plot:



5 Very-Short Answer Questions

1. We can make assumptions about test outputs by looking at nearby points on the scatterplot. Furthermore, a graph cannot really be used if the plots are too randomly scattered. Looking at the scatterplot before running machine learning algorithms might save time.
2. Labels might depend on data from other instances. Furthermore, data might change over time and the distribution may not be the same.
3. A validation set is used to choose hyper-parameters, and the best performing model. A test set is used to check the accuracy of the selected approach.
4. The number of parameters grows with the size of the dataset. As such, fewer assumptions about the data is required, providing better results when the true data distribution is unknown or cannot be easily approximated.
5. Standardization does not affect the accuracy of a decision tree classifier since both the split value and the sample value are scaled the same. However, KNN accuracy becomes better, as it reduces the scale of features with large values, which tend to dominate the distances and result in negligence of smaller-scale features. Standardization reduces this scale.

6. There is no training phase in KNN as training data is merely stored, thus k does not affect it. Prediction runtimes are of O(nd), and thus increasing k would result in a linear increase in prediction runtime.
7. Increasing k in KNN generally reduces the training error but increases the approximation error, as the impact of label noises cancel one another out. However, if k is too large, training error may increase and approximation error decrease as the majority class will be the main prediction in most cases.
8. Increasing n decreases the training error, but also decreases how well the training error approximates the test error.