Problem Set 1: Trees
IMT 575: Scaling, Applications, and Ethics

Instructor: Lovenoor (Lavi) Aulck
University of Washington
Spring Quarter 2020

March 31, 2020

Instructions

All problem sets must be submitted as a Jupyter notebook. You must organize your analysis and utilize the notebook's markdown capabilities to present your solutions. Partial credit will be awarded for each question for which a serious attempt at finding an answer has been shown. Please see the course syllabus for more instructions/information. A rubric for this assignment is available on the Assignment's canvas page. Unless otherwise noted, this assignment is due on **April 10, 2020.**

Part 1: Creating helper functions

1. **Create a helper function to calculate Gini impurity called "calcGini".** This function should take in two values which represent a count of each class (we can assume only two classes for this problem set) and output the Gini impurity.
2. **Create a helper function to calculate entropy called "calcEntropy".** This function should take in two values which represent a count of each class (we can assume only two classes for this problem set) and output the entropy. (Note: be sure the function appropriately accounts for zero values)
3. **Create a helper function to calculate weighted sums called "weightedSum".** This function should take in two lists - one of values and one of weights/counts - and should output the weighted sum of the values as weighted by weights/counts.
4. **Verify that your functions are working as expected.** Use the same dataset/calculations from lecture and calculate the Gini impurity and entropy for each column. The data can be found in the toyData.csv file.

Part 2: Building trees

Note that you do not need to use tree objects to complete this assignment but are welcome to do so. You are still expected to store information on your tree and corresponding splits in a pandas DataFrame or similar data structure (Question 8). You must calculate all split points using your helper functions and do so from scratch unless otherwise instructed.

5. **Import the Boston using dataset using sklearn.datasets.** Create a new column called "highPriced" that is a boolean indicator of whether the target value in the dataset (i.e. MEDV or median home value) is greater than $35K. The variable "highPriced" will be our response/output variable going forward. What are some interesting characteristics of this variable?

6. **Using your helper functions and with highPriced as your output, find what the best split is along the AGE variable using each of Gini impurity and entropy as the splitting criterion.** (Hint: it may help to build this process as a function because you will be doing it many times throughout this problem set.) Assume the "left" side of each split contains all values less than and the "right" side contains all values equal to or greater than. What is the optimal split point when using Gini impurity? What about when using entropy? What if we calculate the same using the CRIM variable? Comment on any similarities and differences.

7. **Import sklearn's DecisionTreeClassifier and find what the optimal split is along the AGE variable using entropy.** Show that the optimal value as calculated by sklearn's classifier is 37.25. If this is different than what you calculated, explain why and adjust your outputs so it gives you the same value. (*Hint: you may have implemented a splitting method that looked at axis-aligned splits at every data point whereas sklearn looks at axis-aligned splits between every pair of data points.*)

8. **Using the RM, LSTAT, and RAD variables, build a a decision tree with 2 levels (3 split points, 4 leaf nodes) based on entropy.** Show the splits as a pandas DataFrame with the following columns: level (indicates which level of the tree the split occurs. You should have a 1 once and 2 twice), col (indicates the column across which the split occurs), and threshold (indicates the threshold used in the split). Keep in mind that when building this tree, each split necessitates that you find the optimal split for that data subset across all three variables. Your output should resemble the table below.

level	col	threshold
1	first split here (variable/column)	first split here (threshold/value)
2	Left side split here (variable/column)	Left side split here (threshold/value)
2	Right side split here (variable/column)	Right side split here (threshold/value)

9. **Visualize your splits on an X-Y plane**. Set the x-axis to be RM and the y-axis to be LSTAT. Plot the target variable and use straight lines to indicate the axis-aligned splits for your 2-level decision tree. Make sure your visualization is understandable and comment on what you see.

Part 3: Making predictions

10. **Create a training/test split for your data.** Assume the data is already randomized and select every 5th observation from the original dataset to be in your test data (i.e. when working in Pandas, your test dataframe should contain index 0, 5, 10, 15, ...) and all other observations in your training data. How many observations are in your training and test datasets, respectively? How does the target variable look across the two datasets?

11. **Create a baseline set of predictions.** In classification tasks, when reaching a leaf node that is not pure, predictions are often made by randomly selecting from training observations in the corresponding node. Suppose you had a tree with zero splits across your training dataset (i.e. the entirety of your training dataset) and you made predictions for your test dataset. In other words, "predict" values for your test dataset by randomly sampling (with replacement) from the target variable across the entirety of your training dataset. How do your predictions

fare in terms of accuracy and other metrics? (Note: when comparing subsequent classifiers, use not only accuracy, but other metrics as well. Also, contextualize differences you may find.)

12. **Now, use a 2-level decision tree to make predictions.** Re-train a 2-level decision tree (i.e. a tree with 4 leaf nodes) using the training dataset with only the RM and LSTAT variables as inputs. How does it compare to the decision tree you built using the entirety of the data? Again, at each leaf node, randomly sample the output values from the training observations to make predictions for the test set. How do the predictions from this classifier compare to the baseline you created in question 11?

Part 4: Comparing to out-of-the-box classifiers

13. **Use sklearn's DecisionTreeClassifier to recreate the decision tree that is trained on the training data.** There may be slight differences in the predictions but show that 1) the splits are at the same locations as the tree you built and 2) the probabilities for the leaf nodes are also the same. How do the predictions from this classifier compare to both your decision tree as well as the baseline you created in question 11?

14. **Use sklearn's BaggingClassifier to create a bagging classifier whose base is a DecisionTreeClassifier with 2 levels and entropy as the split criterion.** Train it on your training dataset and make predictions for your test dataset. How does this bagging classifier perform compared to the decision tree you created in question 13?

15. **Use sklearn's RandomForestClassifier to create a random forest classifier whose base is a decision tree with 2 levels and entropy as the split criterion.** Train it on your training dataset and make predictions for your test dataset. How does this random forest classifier perform compared to the classifiers you created in questions 13 and 14?