Homework 2: Tree-based Models **CSCE 633**

Due: 11:59pm on March 8, 2024

Instructions for homework submission

a) There are two sections in this homework: 1) Write the solution to the first section in Latex. 2) For the programming questions, explain your thought process, results, and observations in a markdown cell after the code cells. Do not just include your code without justification.

b) You need to submit a zip file:

- The name of the .zip file: FirstName_LastName_HW2.zip
- The zipped folder includes the following files:
  - A pdf (generated using Latex) for the math section: FirstName_LastName_HW2.pdf
  - A .ipynb jupyter notebook file for the programming section: FirstName_LastName_HW2.ipynb
  - A csv file with 2 main columns. One for the tree classifier predictions (part A) with the name pred_tree and one for the XGBoost predictions (part B) with the name pred_xgboost. File name: FirstName_LastName_preds.csv

c) **You can not use available ML libraries for part A.** You can still use libraries such as NumPy, Pandas, Matplotlib to implement the model and plot the data or result. You can also use auxiliary functions from sklearn such as train_test_split. For part B, you can use XGBoost library.

d) Start early :)

e) Total: 100 points

Math Questions

Problem 1: Information Gain (20 points)

NOTE: This is not a programming assignment, so you may NOT use programming tools to help solve this problem. Show your work.

Suppose you are given 6 training points as seen below, for a classification problem with two binary attributes X_1 and X_2 and three classes $Y \in \{1, 2, 3\}$. You will use a decision tree learner based on information gain.

(1) Calculate the conditional entropy for both X_1 and X_2.

(2) Calculate the information gain if we split based on 1) X_1 or 2) X_2.

(3) Report which attribute is used for the first split. Draw the decision tree using this split.

(4) Conduct classification for the test example $X_1 = 0$ and $X_2 = 1$.

X_1	X_2	Y
1	1	1
1	1	1
1	1	2
1	0	3
0	0	2
0	0	3

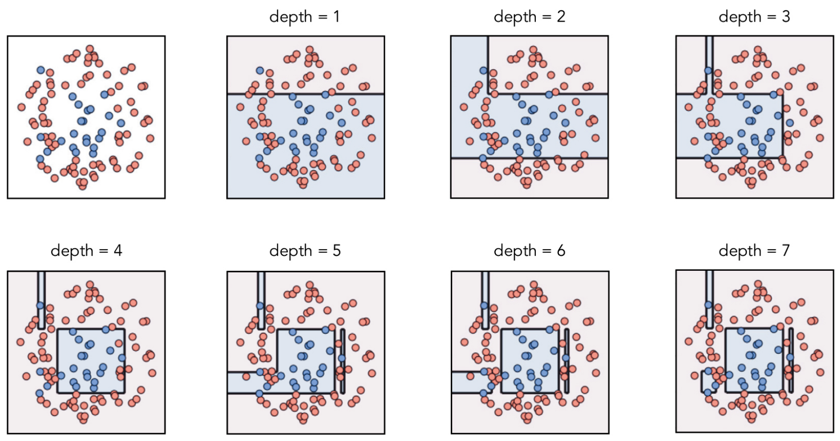
Programming Questions

Part A - Classification Tree (50 points)

In this problem, you will be coding up a classification tree from scratch. Trees are a special class of graphs with only directed edges without any cycles. They fall under the category of directed acyclic graphs or DAGs. So, trees are DAGs where each child node has only one parent node.

Since trees are easy to design recursively, it is super important that you are familiar with recursion. So, it is highly recommended that you brush up on recursion and tree-based search algorithms such as depth-first search (DFS) and breadth-first search (BFS).

The figure below shows the growth of a tree to a maximum depth of seven on a two-class classification dataset.



As the tree grows, note how many parts of the input space do not change as leaves on the deeper branches become pure. By the time we reach a maximum depth of seven, there is considerable overfitting.

Our dataset is Loan Dataset. In part A, we will create a decision tree as a binary classifier.

A-1 Data Processing and EDA

1. There are 2 data splits for this homework, data_train and data_test. The data_test doesn't have ground truth labels, you need to use the trained model to do inference on it. For validation, you can split the training data to train and validation sets. Read the data. (Try *read_csv()* function in *pandas* library)

2. Print the training data. How does the data look like? Add a short description about the data. (You may use *head()* function in *pandas* library)

3. Return the shape of the data. Shape means the dimensions of the data. (In Python, *pandas* dataframe instances have a variable *shape*)

4. Does the data have any missing values? How many are missing? Process the data to remove these missing values. You can drop the corresponding rows or apply imputation. (In *pandas*, check out *isnull()* and *isnull()*.sum())

5. Extract the features and the label from the data. Our label is Loan_Status in this case.

6. Plot the histograms of all the variables in the data.

7. What are the different feature types in this data (e.g, continuous vs categorical). Apply feature encoding if needed. What is the feature transformation you used and why?

A-2 Implementation

1. Using the data you pre-processed above, implement a classification tree from scratch for prediction. You are NOT allowed to use machine learning libraries for the tree model.

2. Build the tree model using training data and Gini Index as the splitting criteria. Validate the trained model with validation data.

3. Using the trained model, conduct inference on the test data and save the predicted result in a separate file called FirstName_LastName_preds.csv with the column name pred_tree.

4. Below are the steps you may want to consider.

   Define a splitting criteria: 1) this criteria assigns a score to a split.

   Create the split: 1) split the dataset by iterating over all the rows and feature columns; 2) evaluate all the splits using the splitting criteria; 3) choose the best split.

   Build the tree: 1) decide when to stop growing (when the tree reaches the maximum allowed depth or when a leaf is empty or has only 1 element); 2) split recursively by calling the same splitting function; 3) create a root node and apply recursive splitting.

   Predict with the tree: For a given data point, make a prediction using the tree.

Part B - Boosting (30 points)

Now that we implemented classification trees in part A, we would like to use a decision-tree-based ensemble Machine Learning algorithm for Loan Dataset. You can use the same pre-processed dataset as part A.

1. Define a function *train_XGBoost* to use an *XGBoost* model with L2 regularization that returns a dictionary. The keys of the returned dictionary would be the lambda parameter (for L2 regularization) and the corresponding value is the mean AUC of 10 fold cross validation. Use the following lambda parameters: [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3].

2. Define a function *train_XGBoost* to use an *XGBoost* model with L2 regularization that returns a dictionary with keys as lambda parameter for the following values: [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3] and the corresponding mean auc as value pairs. Compute the mean AUC values with 10 fold cross-validation. Describe your hyperparameter tuning procedures and the optimal *lambda* value for l2 regularization.

3. Train and test the model with the best parameters you found.

4. Plot the ROC curve for the XGBoost model on validation data and also print the area under curve measurements. Include axes labels, legend, and title in the Plot.
5. Compare the validation result you obtained using XGBoost with the results using the tree you implemented in part A. Which model did perform better and why?
6. Using the trained model, conduct inference on the test data and save the predicted result into the predictions csv file with **pred_xgboost** as the column name.