Programming Questions

Part A - Classification Tree

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).

Your submission should include a script that can be run seamlessly and performs all the following steps one after another. Submission with a runtime error would result in lost points.

Our dataset is Loan Dataset. You will try to use your tree as binary classifier.

A-1 Data Processing and EDA

1. There should be 3 dataset splits for this homework, data_train, data_valid and data_test. The data_test doesn't have ground truth labels, you need to use the trained model to do inference on it. 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? Return the number of missing values. (In *pandas*, check out *isnull()* and *isnull()*.sum())
5. Drop all the rows with any missing data. (In *pandas*, check out *dropna()*. The *dropna()* accepts an argument *inplace*, check out what it does and when it comes in handy.)
6. Extract the features and the label from the data. Our label is Loan_Status in this case.
7. Plot the histograms of all the variables in the data. Provide a brief discussion on your intuition regarding the variables and the resulting histograms.

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 like scikit-learn here.
2. Train the model using training data and use validation data to validate the trained model.
3. Using the trained model, conduct inference on the test data and save the predicted result in a separate file called HW2 Test Result.csv.
4. Below are suggested steps you may want to consider.

   Define a splitting criteria: 1) this criteria assigns a score to a split; 2) this criteria might be the Gini Index.

   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

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 dataset pre-processing method as part A.

1. Define a function *train XGBoost* to use a *XGBoost* model with L2 regularization that returns a dictionary with *alpha vals* as keys and corresponding *mean auc* as value pairs. In the function, apply bootstrapping and repeat the training process for *n bootstraps* = 100 times, and for each time train the model for *max iter* iterations with all *alpha*s from *alpha vals*. Compute the AUC values with the validation set and append the values each time to list *aucs xgboost*. Then calculate *mean auc* over time for each *alpha*. Please describe your hyperparameter tuning procedures and optimal *alpha* in *alpha vals* = [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3] that gives the best model.
2. Train and test the model with the best parameters you found.
3. Plot the ROC curve for the XGBoost model and also print the area under curve measurements. Include axes labels, legend, and title in the Plot.