Due date: 10/16/2024, end of the day.

Please submit an .ipynb file via Canvas

Instructions:

1) The .ipynb file shall include not only the **source code** but also necessary **plots/figures** and **discussions** which include your *observations*, *thoughts,* and *insights*.
2) Please avoid using a single big block of code for everything and then plotting all figures altogether. Instead, use a small block of code for each sub-task which is followed by its plots and discussions. This will make your homework more readable.
3) Please follow common software engineering practices, e.g., by including sufficient **comments** for functions, important statements, etc.

Programming Problem:

In this homework, you will try to implement a decision tree model and learn how to optimize it with the help of sklearn package. We provide a notebook *hw3_24Fall.ipynb* which contains the needed code for step 2 and some instructions which you can follow to help you develop your own code.

Step 1: Data Preparation

Load the Titanic.csv file and examine a sample of the data. Notice that the dataset contains both categorical and numerical features.

1. Address any missing values by imputing them with the feature's mean across the dataset.
2. For this assignment, select a subset of the data including the independent variables: 'pclass', 'sex', 'age', and 'sibsp', and the dependent variable 'survived'.
3. Ensure that 'survived' is a binary variable coded as 1 (yes) or 0 (no).
4. Split the data into training and test sets, using an 80/20 split.

Step 2: Data Processing and Initial Analysis

1. Recognize that the 'age' attribute is continuous. As discussed in our class, decision trees are typically applied to categorical features. Utilize the steps outlined in the provided Jupyter notebook to discretize 'age' using quantile binning.
2. Compute the information gain to determine the optimal first split in the decision tree.

Step 3: Decision Tree Modeling

1. Employ sklearn to train a decision tree model, setting the maximum number of leaf nodes to 20 and the **random_state** to **your student id** number. (Please apply it to all the rest questions)
2. Visualize the complete tree. Note that your tree's structure and size may vary from the example.
3. Implement a function to calculate the accuracy, precision, recall, and F1 score on the test set.

Step 4: Model Optimization

1. Apply GridSearchCV() to identify the optimal max_leaf_nodes parameter, exploring values from 5 to 20, for tree pruning.

2. Plot the pruned tree, which should be more compact than the initially generated tree. Report the performance. (using metrics in step 3.3, same below for step 5)

Step 5: Advanced Modeling

1. Replicate Steps 3 and 4 to construct two additional decision tree models with varying parameters, such as the maximum depth and splitting criteria.
2. Use majority vote to create an ensemble learning model that combines the three decision trees model we trained in the step 4 and step 5.1.
3. Use the *RandomForestClassifier()* function to train a random forest using the optimal tree size you found in step 4. You can set n_estimator as 50. Compare the performance of your random forest and your ensembled model.