CAP4770 Assignment 4

For this assignment, you will implement AdaBoost for binary classification and study the behavior of AdaBoost. You will use the given dataset, which was generated by make_moons(). The base classifiers are decision stumps (decision trees with `max_depth = 1`).

Here are the steps that you will carry out.

1. Load the dataset in the files 'moon-all-input.npy' and 'moon-all-output.npy'. Plot and visualize all the instances, as we have done in the lecture slides.

2. The entire dataset has 500 instances, which have already been randomly permutated. Split it into the training and test sets. The first 375 instances shall be in the training set; the remaining 125 instances shall be in the test set. Verify that the test set is well balanced (so that you can use `accuracy_score()` as a performance metric).

3. Implement AdaBoost. This is a big step. You will need to study the lecture slides 36-51 carefully. You will use decision stumps as the base classifiers. Each base classifier is created by something like: `DecisionTreeClassifier(max_depth=1, random_state=42)`
   Please keep the `random_state` value at 42 for ease of grading.

   Please run AdaBoost for 3000 steps, which means you will need 3000 decision stumps. This is more than needed for our case, but it will be useful for studying the behavior of the AdaBoost algorithm.

   Hint: You can store the 3000 based classifiers in a dictionary for easy access with loops. Keep the learning rate $\eta = 1$, although you can experiment with it on the side. Read the manual for `DecisionTreeClassifier` about how to set the sample weights during training.

   Please record the following quantities for each of the base classifiers: the class weight α; the instance weights for all the training instances; the weighted error rate ϵ; the (unweighted) accuracy score on the training set. You will later visualize them.

   Since we run AdaBoost for 3000 steps, this also gives 3000 ensemble models, which we will also call the final models. In other words, after *k* boosting step, there is an ensemble model by combining the first *k* base classifiers. For each of the ensemble models, please make predictions on the training set and the test set, and record the accuracy scores on the training set and the test set.

4. Plot the prediction errors (i.e., 1-accuracy_score) on the training and test sets by the ensemble models. Your x-axis is the indices of the ensemble models: 0 through 2999, which is the same as the boosting step indices. Also, plot another graph by zooming into the first 200 ensemble models to see extra details. Discuss briefly what you observe.
5. Plot the classifier weights α for all 3000 base classifiers. Also zoom into the first 200 classifiers to see extra details.
6. Plot the weighted error rates ϵ for all 3000 base classifiers. Also zoom into the first 200 classifiers to see extra details.
7. Plot the accuracy scores on the training set for all 3000 base classifiers. Also zoom into the first 200 classifiers to see extra details.

8. Next, let us visualize some details about what happens in each boosting step. Please do the following for decision trees with indices 0, 1, 2, 3, 4, 14, 20, 50, 100, and 1000.
    - Plot the tree with the `plot_tree()` function.
    - Plot the decision boundary (a threshold value) together with a scatter plot of the training data instances. Use different symbols and colors for different classes, such as triangle for class 1 and circle for class 0.
    - Use `scatter()` to plot the instance weights used to train this tree. This will be a scatter plot of the training instances. The size of each point will be proportional to the weight of the instance. You will need to scale up the weights to see them. Superimposed the decision threshold onto the same plot.
9. Plot decision boundaries for the final models (ensemble models) with indices 0, 1, 2, 3, 4, 10, 14, 20, 50, 100, 200, 500, and 1000. Show also the training instances in each of the plots. You will need some simple coding here.
10. Discuss what you have learned. Feel free to explore more, as it is difficult to understand the behavior of AdaBoost. If you look at it more carefully, you may have some new discoveries that people didn't know before.