Assignment 6

Applied Machine Learning

In this assignment we will generate an ensemble of primitive classifiers, using Bagging method, and compare their performances to the regular classifiers.

This week's problem is the classification of heart failure disease. Download the Kaggle `heart.csv` dataset file from the module content. Load the dataset into your model development framework and examine the features to note that they are a mixture of numerical and nominal features. Apply necessary pre-processing such as nominal to numerical conversions (e.g. `OneHotEncoder`). Make sure to sanity check the pipeline and perhaps run your favorite baseline classifier first.

1. [10 pts] Report 10-fold cross-validation ("CV") performances of the following types of classifiers, using default parameters:
   - `GaussianNB`
   - Linear SVC (use `SVC(kernel='linear', probability=True)`)
   - `MLPClassifier`
   - `DecisionTreeClassifier`

   Now report the `RandomForestClassifier` performance too. Since this is already an ensemble classifier, this one does not need to be done with CV.

2. [10 pts] Generate an ensemble of 100 classifiers for each of the four basic classifiers in Q1. and store each ensemble as a list. In order to create weak sub-classifiers within our ensembles, we underpower some hyperparameters:
   - For the neural network, set the hidden sizes to (3, 3), max iterations to 30, and tolerance to 1e-1.
   - For the decision tree, set max depth to 5 and max features to 5.

   For each of these 4 ensembles, report the performance of the first classifier in the ensemble. I.e., for your ensemble of 100 decision trees, report the performance of just the first weak tree.

3. [20 pts] Write a function `ensemble_fit()` to receive the ensemble (i.e. one of the 4 lists from Q2.) as an input and train it on one of the subsets (i.e. bagging) of the training data. (Hint: `random.sample` could generate the subset of data you'll need.) This way, each classifier will see only a different subset of the training dataset, also called as subsampling the input data for training. Use all features in these subsamples; only subsample the rows/observations.

4. [20 pts] Write a function `ensemble_predict()` to receive the trained ensemble (i.e. one of the lists from Q3.) as input and output a prediction for a given observation. Since each sub-classifier will have its own prediction, use a voting scheme on the returned predictions. (Hint: The final prediction should be the `np.argmax()` of the votes, not merely a "max". Note that `c.predict_proba()` should have better results.)

5. [20 pts] Report 10-fold CV performances of the ensembles with a subsample ratio of 0.2. Compare to a regular decision tree (same subsample ratio). Now repeat these for a subsample ratio of 0.05.

6. [10 pts] Report the 10-fold CV performances of the ensembles for the training subsample ratios of (0.005, 0.01, 0.03, 0.05, 0.1, 0.2). Now train regular versions of those 4 classifiers and report their performance. (Hint: pass the regular classifier to the same ensemble CV in a list of one element. This way, the same script can be used for this entire step)

7. [10 pts] For each of the 4 types of classifier, plot the performances of the ensemble at the different subsample ratios and the performances of the regular classifier at the different subsample ratios on the same plot. Thus, you should have 4 plots, one for each type of classifier. To make it graphically clear which performances are ensemble vs. regular, plotting in 2 different colors is recommended.

   Report your detailed observations.