CS 334: Homework #5

Submission Instructions: The homework is due on Nov 6th at 11:59 PM ET. This is a lighter homework (with total of 70 points). Make sure your program uses Python 3.8. Your submission should consist of two steps (detailed in Homeworks #1, #2). If *either of the two steps are late, then your assignment is late.*

Dataset Description: For this homework, you will be explore the effects of reducing the dimensionality of the data and running an almost random forest on the wine-quality dataset (from homework 1).

1. **(5+10+10+10=35 pts) PCA**

   For this problem, we will try to quantify the impact of dimensionality reduction on logistic regression. You are free to use sklearn packages for this problem.

   (a) (Code) `normalize_feat(xTrain, xTest)`: Normalize the features of the wine quality dataset (where applicable), standardizing features by removing the mean and scaling to unit variance. Your function should return normalized training and test data.

   (b) (Code) `unreg_log(xTrain,yTrain,xTest,yTest)`: Train an *unregularized* logistic regression model on the dataset and predict the probabilities on the test data and calculate the ROC. Your function should return the false positive rates and true positive rates.

   (c) (Code) `run_pca(xTrain, xTest)`: Run PCA on the normalized training dataset. How many components were needed to capture at least 95% of the variance in the original data. Discuss what characterizes the first 3 principal components (i.e., which original features are important). Your function should return 3 things: the transformed xTrain, transformed xTest and the PCA model components(principal axes in feature space).

   (d) (Written)

   - Use `unreg_log` to train an *unregularized* logistic regression models on the normalized dataset and predict the probabilities on the normalized test data and calculate the ROC.
   - Use `unreg_log` to train an *unregularized* logistic regression models on the PCA dataset and predict the probabilities on the appropriately transformed test data (i.e., for PCA, the test data should be transformed to reflect the loadings on the k principal components).
   - Plot the ROC curves for both models (normalized dataset, PCA dataset) on the same graph. Discuss your findings from the ROC plot.

2. **(10+10+20+10+10+5=65 pts) Almost Random Forest**

   For this problem, you will be implementing a variant of the random forest using the decision trees from `scikit-learn`. However, instead of subsetting the features for each node of each tree in your forest, you will choose a random subspace that the tree will be created on. In other words, each tree will be built using a *bootstrap sample and random subset of the features.* The template code for the random forest is available in `rf.py`

   (a) (Code) Build the adaptation of the random forest. Note that the forest will support the following parameters.

- nest: the number of trees in the forest
- maxFeat: the maximum number of features to consider in each tree
- criterion: the split criterion – either gini or entropy
- maxDepth: the maximum depth of each tree
- minSamplesLeaf: the minimum number of samples per leaf node

Note that you'll need to implement the helper functions `generate_bootstrap`, `generate_subfeat`, and the class functions `train` and `predict` in the template code.

- `generate_bootstrap`: Given a feature matrix (xTrain) and the labels (yTrain), generate the bootstrap sample of the feature matrix, the corresponding labels, and the out of bag indices.
- `generate_subfeat`: Given a feature matrix (xBoot) and the maximum number of features, return a numpy2d array where a subset of the features are chosen (this is to be passed into the random forest), and the corresponding indices of the chosen features.
- `train`: Given a feature matrix and the labels, learn the random forest using the data. The return value should be the OOB error associated with the trees up to that point. For example, at 5 trees, calculate the random forest predictor by averaging only those trees where the bootstrap sample does not contain the observation. Each dict element of the class variable `model`, should have a key associated with the bootstrap sample (i.e., 0 for the first bootstrap, 1 for the second bootstrap) and the value is a dict itself with the key `tree` that contains as the value the tree that is built on each bootstrap sample, and `feat` which should reflect the corresponding subsampled of features used in the tree.
- `predict`: Given a feature matrix, predict the responses for each sample.

(b) (Written) Find the best parameters on the wine quality training dataset based on classification error. Justify your selection with a few plots or tables.

(c) (Written) Using your optimal parameters, how well does your version of the random forest perform on the test data? How does this compare to the estimated OOB error?