CS 378h: Machine Learning **Spring 2024**

Homework 1 - Programming

Lecture: Prof. Adam Klivans

Keywords: decision trees

Read the online documentation on decision trees and random forests in scikit-learn to find out how to use decision trees and random forests. Notice that training a classifier is done using the `fit` method, and that for decision trees this is done using a more sophisticated evolution (known as CART) of the ID3 algorithm covered in class.

On random seeds: Many functions in scikit-learn, including models as well as utilities, use randomization. For ease of grading, we will fix a random seed for questions 1, 2, and 3 so as to make behavior deterministic. We will use a random seed of 10. This can generally be done by passing in `random_state=10` to the function; please consult documentation if unsure. For cross-validation methods, though, you will likely need to set the `cv` argument instead. You can do this by setting `cv=KFold(n_splits=<number of splits>,random_state=10,shuffle=True))`.

1. [**10 points**] Use the breast cancer data set from Homework 0 to create a training set. Recall that the label is 0 if the patient's data indicates a malignant cancer and 1 otherwise. Compute the base rate of malignant cancer occurrence over the entire data set. In other words, what would be your best guess for the probability of malignant cancer of a single example using only the labels in the training set? This question is very simple, so try not to overthink it.

2. The goal is to build a decision tree that, based on the other features in the set, predicts whether or not a patient has malignant cancer. So this is a classification problem. Using `tree.DecisionTreeClassifier` and other functions in the scikit-learn library, one can build a decision tree and calculate both its training accuracy when fitted to the entire data set as well as its accuracy using 10-fold cross validation (which gives a better idea of true accuracy).

   In this question you will need to complete two sub-components:

   (a) [**6 points**] Make a plot visualizing the performance of a `tree.DecisionTreeClassifier` as you search for an optimal `max_depth` parameter. Vary the depth of your decision tree using `max_depth` = 1,2,...,10 and record the results from the following evaluation procedures for each setting:

   - The accuracy when training and testing on the full dataset.
   - 10-fold cross-validated accuracy.

   Plot the results of both evaluation procedures on the same plot with evaluation scores on the y-axis and `max_depth` values on the x-axis. Use 10 as your random seed/state for the decision tree and the cross-validation. Use a legend to label both evaluation procedures.

   (b) [**4 points**] Answer the questions below based on the results of 2a. Write your answers in the corresponding field in the markdown cell that is present in the HW1 template notebook. You can do this by double clicking the markdown cell and writing your answer directly in the cell. Pressing enter will re-render the markdown.

i. What setting of `max_depth` gave the best accuracy w.r.t. the **full-dataset** accuracy? If more than one setting equaled the best accuracy, list each of the best settings.

ii. What setting of `max_depth` gave the best accuracy w.r.t. the **cross-validated** accuracy? If more than one setting equaled the best accuracy, list each of the best settings.

3. This question explores random forest classifiers by using scikit-learn's `ensemble.RandomForestClassifier`. You will make two plots and answer questions about them.

   (a) [**2 points**] For the first plot, use a `ensemble.RandomForestClassifier` and the best depth you found 2(b)ii as `max_depth`. We will now find the optimal setting of a second parameter, `n_estimators`. Vary the number of trees in the forest via the parameter `n_estimators` and plot its 10-fold cross-validated accuracy (use `n_estimators` $= 1, 2, \ldots, 20$). Again, use 10 as your random seed for your classifier and cross-validation.

   (b) [**2 points**] Do you see an improvement using random forests versus using a single tree? (Note: use the `n_estimators`=1 result as the result for a single tree.)

   (c) [**2 points**] What setting of `n_estimators` gave the best accuracy w.r.t. the **cross-validated** accuracy? If more than one setting equaled the best accuracy, list each of the best settings.

   (d) [**2 points**] For the second plot, again use a `ensemble.RandomForestClassifier`, but this time you will fix the `n_estimators` parameter and instead attempt to find the optimal setting of a `max_depth`. Use your answer to 3c as the setting for `n_estimators` and follow the procedure from 2a to find the best setting for `max_depth`. This time, only plot the results from 10-fold cross-validation and not the training set, but the plot should be the same structure as in 2a otherwise (use `max_depth` = 1,2,...,10). Again, use 10 as your random seed.

   (e) [**2 points**] In the plot in 3d, is the optimal setting of `max_depth` the same as in 2(b)ii? If not, what is the new optimal setting of `max_depth`?

4. For this last question, we will explore the dependability of our estimates.

   (a) [**6 points**] Make a plot using the following procedure:

      i. Using `random_state` values from $0, 1, \cdots, 99$ calculate the 10-fold cross-validation accuracy of different `tree.DecisionTreeClassifiers` with `max_depth` settings from $1, 2, \cdots, 10$.

      ii. Then record the best `max_depth` settings for each `random_state`. Be sure to check whether multiple settings achieve the best accuracy.

      iii. Plot the counts for the best `max_depth` settings as a bar chart with the `max_depth` settings on the x-axis and the 'best parameter counts' on the y-axis (number of times that parameter was selected as the best max depth setting). This will be the sampling distribution of our 'best parameter' estimate.

      Note: this calculation might take some time. For debugging, try a smaller range of `random_states`.

   (b) [**2 points**] What are the top two most frequent parameter settings?