CS 334: Homework #2

Submission Instructions: The homework is due on Sept 30th at 11:59 PM ET. Make sure your program uses Python 3.7. Your submission should consist of two steps (detailed below). If *either of the two steps are late, then your assignment is late.*

1. **Upload PDF to Gradescope**: Create a single high-quality PDF with your solutions to all the problems. The solutions can be typed and/or written and scanned but the resulting pdf *must* be easily readable.

2. **Submit a single zip containing all the source code to Canvas**: Make sure your code is complete, well-commented, and all the files are included in the directory. Also include any additional code files or auxiliary data that is needed to run your code or were used to generate any of your answers in the PDF. These files should be named appropriately so we can easily tell what question this is related to. This *must also include the honor statement*, a `README.txt` file that contains the following words:

   ```
   /* THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING CODE
   WRITTEN BY OTHER STUDENTS.
   Your_Name_Here */

   I collaborated with the following classmates for this homework:
   <names of classmates>
   ```

   Rename your code directory to hw2-<eid> and zip it. As an example, for an eid of jho334, the following lines on a unix terminal would rename everything and zip it into the file `hw2-jho334.zip`.

   ```
   $ mv cs334-hw2 hw2-jho334
   $ zip -r hw2-jho334.zip hw2-jho334
   ```

   Submit the single zip file to Canvas.

1. **Decision Tree Implementation** (25+10+7+8=50 points): For this problem, you will implement the decision tree algorithm. The algorithm should work on both of the datasets from the previous homework (q3 and q4). The template code is found in `dt.py`. You *are not allowed* to use any `scikit-learn` modules in this problem. You can either execute the file from the command line by running the following command `python dt.py <max depth> <min leaf samples>` or use another python file to call the D.

   (a) Implement the `train` function of decision tree. The arguments to the function are `xFeat`, an $n \times d$ array where each row represents a sample, and each column a feature, and y, a 1-D array of size $n \times 1$ that contains which class (0 or 1). You can add variables to the class to store whatever information you need.

   (b) Implement the `predict` function of decision tree. This will take in a 2D array ($m \times d$), and should output a 1-D array ($m \times 1$) that represents the predicted class of each sample.

   (c) What is the training accuracy and test accuracy of the data for different values of max depth and minimum number of samples in a leaf? Plot the accuracy of train and test as a function of both of these terms (Hint: Consider a 3D plot to better understand the relationship of both values on accuracy).

(d) What is the computational complexity of the train and predict function you implemented in terms of the training size (n), the number of features (d), and the maximum depth (p)? You must justify your answer.

2. **(5+6+5+7=23 pts) Exploring Model Assessment Strategies**

We will be using the wine quality dataset from the previous homework to explore the different model assessment strategies covered in class. The template code is found in `q2.py`. You can use any of the `scikit-learn` modules in this problem.

(a) Implement the holdout technique to assess the model. The arguments to the function `holdout` are `model, xFeat, y, testSize`. The parameter specifications are:

i. `model` is a `DecisionTreeClassifier` object from `sklearn`
ii. `xFeat` is an $n \times d$ array where each row represents a sample and each column a feature,
iii. `y` is a 1-D array of size $n \times 1$ that contains which class (0 or 1)
iv. `testSize` is a `float` between 0 and 1 that represents the proportion of the dataset to include in the test split

The output of the function is the following:

i. `trainAuc` is the AUC on the training dataset
ii. `testAuc` is the AUC on the holdout dataset.
iii. `timeElapsed` is the time the function took.

(b) Implement the k-fold cross-validation approach for a model, where the performance is reported from the k-different validation. The arguments to the function are `model, xFeat, y, k`. See (a) for the definitions of the first three parameters. k is the user-specified value for k-fold cross validation. For the output, the `trainAuc`, and `testAuc` should reflect the average across the k-different folds.

(c) Implement Monte Carlo Cross-validation approach with s samples. The arguments to the function are `model, xFeat, y, testSize, s`. See (a) for definitions of the first four parameters. s is the number of samples for performing the validation assessment. Similar to (b) the `trainAuc`, and `testAuc` should reflect the average across the s samples. Hint: You may find it useful to re-use code from (a).

(d) Run the `q2.py` script from the command line to get a table of the AUC and the time. Comment on how the different model selection techniques compare with one another with regards to AUC, robustness of the validation estimate, and the computational time of the three different hold-out techniques.

3. **(10+6+6+5=27 pts) Robustness of Decision Trees and K-NN**

In this problem, we will explore how sensitive decision trees and k-nn are on the wine-quality dataset.

(a) Use k-fold cross validation to find the optimal hyperparameters for k-nn and decision tree. What are the optimal parameters, θ_{opt}, for each model? Also make sure to justify your choice of k (associated with k-fold and not to be confused with k-NN).

(b) For the optimal hyperparameter found in (a), $\theta_{\text{knn-opt}}$, train the k-nn on the entire training dataset and evaluate both AUC and accuracy on the test dataset. Create 3 datasets where you randomly remove 1%, 5%, and 10% of the original training data. For each random subset, train a new k-nn model using the same parameter, θ_{opt}, and evaluate both AUC and accuracy on the test dataset.

(c) For the optimal decision tree parameters found in (a), $\theta_{\text{dt-opt}}$, train a decision tree on the entire training dataset and evaluate both AUC and accuracy on the test dataset. Using the random subsets from (b), train a new decision tree model with the same parameters and evaluate both AUC and accuracy on the test dataset.

(d) Report the AUC and accuracy of the 8 different models from (b) and (c) in a table. Comment on the sensitivity of the two algorithms to the training dataset.