CS 334: Homework #2

Submission Instructions: The homework is due on Gradescope. A part of your homework will be automatically graded by a Python autograder. The autograder will support Python 3.10. Additional packages and their versions can be found in the `requirements.txt`. Please be aware that the use of other packages and/or versions outside of those in the file may cause your homework to fail some test cases due to incompatible method calls or the inability to import the module. We have split homework 2 into 2 parts on Gradescope: the coding portion and the written answer portion. If *either of the two parts is late, then your homework is late.*

1. **Upload PDF to HW2-Written Assignment**: Create a single high-quality PDF with your solutions to the non-coding problems. The solutions can be typed and/or written and scanned but the resulting pdf *must be legible* and make sure to tag the section appropriately on your PDF!

2. **Submit code to the HW2-Code Assignment**: Your submitted code must contain the following files: 'dt.py', 'q2.py', 'README.txt'. You are welcome to submit other files but the autograder will only copy these files when running the test cases so make sure they are self-contained (i.e., capable of running standalone). Make sure you always upload *ALL* of these files when you (re)submit. The `README.txt` file *must contain a signed honor statement* that contains the following words:

   ```
   /* THIS CODE IS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING CODE
   WRITTEN BY OTHER STUDENTS OR LARGE LANGUAGE MODELS SUCH AS CHATGPT.
   Your_Name_Here */

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

1. **Decision Tree Implementation** (5+20+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 homework 1 (q3 and q4). The template code is found in `dt.py`. The program takes maximum depth, minimum leaf samples as required arguments to be used as stopping criteria, and datasets as optional arguments (by default it uses the train/test datasets from hw1 q4). 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 DecisionTree class methods.

   (a) (Code) Implement the `calculate_split_score` helper function. The arguments to the function are `y`, a 1D numpy array of size n that contains which class (0 or 1) and the criteria ('gini' or 'entropy') and output the gini or entropy score associated with the array.

   (b) (Code) Implement the `train` function of decision tree. The arguments to the function are `xFeat`, a numpy 2D array of size $n \times d$ where each row represents a sample, and each column a feature, and y, a numpy 1D array of size n that contains which class (i.e., 0 or 1). You can add variables to the class to store whatever information you need. No pruning is required in your implementation.

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

(d) (Written) 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 dataset (hw1 q4 datasets) as a function of these terms (you can have 2 plots, one for each variable by fixing the other variable at a default value; or have a 3D plot to understand the relationship of accuracy to both values).

(e) (Written) 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 homework 1 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) (Code) 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 a numpy 2D array ($n \times d$) where each row represents a sample and each column a feature,
iii. `y` is a numpy 1D array of size n 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.

Note that the sampling should not be deterministic and will likely vary each time you run it!

(b) (Code) 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) (Code) 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) (Written) 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 regard 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) (Written) 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) (Written) 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 5%, 10%, and 20% 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) (Written) 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) (Written) 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 datasets (how they are impacted, not in the sense of sensitivity and specificity).