CS 334: Homework #1

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 1 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 HW1-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 HW1-Code Assignment**: Your submitted code must contain the following files: 'q1.py', 'q2.py', 'knn.py', 'q4.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. **Numerical Programming** (1+4+4+1=10 points): For this problem, we will perform a speed comparison of two types of code, comparing the non-vectorized code (for-loop) and a vectorized version (Numpy). The template for the problem is in `q1.py`. Please get yourself familiar with the NumPy package (`https://numpy.org/doc/stable/`) which will be useful for all homeworks.

   (a) Fill in the code for `gen_random_samples`. Create an array of 5 million random numbers using `numpy.random.randn`.

   (b) Fill in the code for `sum_squares_for`. Compute the sum of squares using a for loop for each element of the array. Time how long it takes to compute this value for the samples.

   (c) Fill in the code for `sum_square_np`. Compute the sum of squares `numpy.dot`. Time how long it takes to compute this value for the samples.

   (d) How much faster was the vectorized approach compared to the first approach? You can execute the code from the command line:

   ```
   $python q1.py
   ```

2. **Visualization Exploration**(1+7+6+6=20 points): For this problem, we will explore the iris dataset[1]. Create your own python file `q2.py` that contains the commands to load the dataset and plot the features. Make sure that the code is well-documented.

[1]Details of the dataset can be found at `https://en.wikipedia.org/wiki/Iris_flower_data_set`.

Please get yourself familiar with `scikit-learn` dataset loading functionalities (`https://scikit-learn.org/stable/tutorial/basic/tutorial.html#loading-example-dataset`) and `pandas.DataFrame` (`https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html`) which will be useful for all homeworks. You may want to consider `pandas.DataFrame.boxplot` and `pandas.DataFrame.plot` for the visualization in this problem. Other alternative packages are `matplotlib` or `seaborn`.

(a) Load the iris dataset stored in `scikit-learn` by creating a function called `load_iris()` that returns a `pandas DataFrame` with both the features and the target variable (i.e., 150 rows, 5 columns with the headers sepal length (cm), sepal width (cm), petal length (cm), petal width (cm), target.

(b) For each feature (sepal length, sepal width, petal length, petal width), plot the boxplot of the distribution of the feature as a function of the species type. In other words, for sepal length, there should be one plot with 3 boxes (i.e., Iris Setosa, Iris Versicolour, and Iris Virginica).

(c) Explore the different types of features based on the petal and sepal distribution. For petal and sepal separately, plot a scatter plot of the samples with the length on the x-axis and the width on the y-axis, and each species type colored a different color.

(d) Based on your exploration of the data in parts (b) and (c), come up with a set of "rules" that could classify the species type.

3. **K-NN Implementation** (10+20+5+7+8=50 points): For this problem, you will implement the knn algorithm using the Euclidean distance for a small synthetic dataset with only 2 features. The dataset has been split into 4 different files, `q3xTrain.csv, q3yTrain.csv, q3xTest.csv, q3yTest.csv`, where the features are located in the `q3xTrain.csv, q3xTest.csv` files and the corresponding labels are in the `q3yTrain.csv, q3yTest.csv` files. The template code is found in `knn.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 knn.py <k>` or use another python file to call the Knn object (you can see `q4.py` for how you can use the class).

(a) Implement the `train` function of knn. 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. For example, if you wanted to store a string associated with the class, you can add a variable `coolName` and refer to it in any method within the class as `self.coolName`.

(b) Implement the `predict` function of knn. 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) Implement the `accuracy` function in the file. Given an array of predicted values (yHat) and the true labels (yTrue), what is the percentage of correctly classified samples?

(d) What is the training accuracy and test accuracy of the data for different values of k? Plot the accuracy of train and test as a function of k. Hint: you can use `pandas.DataFrame.plot.line` or `seaborn.lineplot`.

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

4. **K-NN Performance** (5+5+5+5=20 points): For this problem, we will explore the impact of preprocessing and irrelevant features to predict the quality of a red wine given some of the physicochemical properties including acidity, citric acid, sulphates, and residual sugar[2]. Similar to the previous question, the dataset has been split into 4 different files, `q4xTrain.csv, q4yTrain.csv, q4xTest.csv, q4yTest.csv`. The template code for this problem is found in `q4.py`. You *can* use the `scikit-learn` module for this problem and if you are unable to get Problem 3 to function properly, you can use the knn implementation in `scikit-learn`.

   (a) Fill in the `standard_scale` function to scale the training data to have 0 mean and unit variance. The transformation should then be applied to the test data. It is important to note that the test data may not have 0 mean and unit variance.

   (b) Fill in the `minmax_range` function to scale the training data so all the features lie between the [0,1] values. The transformation should then be applied to the test data. It is important to note that the test data may not lie between 0 and 1.

   (c) Fill in the `add_irr_feature` function to add two irrelevant features to the training and test data. The data for each column should be drawn from a Gaussian (normal) distribution with 0 mean and standard deviation of 1.

   (d) Evaluate the accuracy of the model on the test dataset for the different preprocessing techniques as a function of k. What conclusions can you draw with regards to the different forms of preprocessing and the sensitivity to irrelevant features for this dataset?

[2]The original dataset is described here `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`.