CS 334: Homework #1

Submission Instructions: The homework is due on Sept 18th 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. The entry code for Gradescope is 9RVBYZ.

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 hw1-<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 `hw1-jho334.zip`.

   ```
   $ mv cs334-hw1 hw1-jho334
   $ zip -r hw1-jho334.zip hw1-jho334
   ```

   Submit the single zip file to Canvas.

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`.

   (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 all the commands to load the dataset and plot the features. Make sure that the code is well-documented.

   (a) Load the iris dataset stored in `scikit-learn`.

   (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). Hint: You may want to consider `pandas`, `matplotlib`, or `seaborn` to do this problem.

   (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.

   (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

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

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. Note that this requires a working `knn.py` from the previous problem.

(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`.