CENG313 Introduction to Data Science

Fall 2020-2021

Lecturer: Dr. Duygu Sarıkaya

Teaching Assistant: Kevser Özdem

Gazi University, Department of Computer Engineering

Assignment 2 due on 21st of December, Monday 23:59

Assignment 2: KNN (K-Nearest Neighbors) Classifier

In this assignment you will be working on implementing the KNN classifier algorithm, and using this algorithm to classify Iris flower species. **Important Note: For this assignment you will be implementing the KNN classifier algorithm from scratch using Python. You are not allowed to use a KNN method from a library (such as skicit learn) but instead you will be implementing this algorithm and defining a method that uses this algorithm yourself.** You can use scikit learn library for other aspects of the assignment (such as loading the dataset, splitting the training and test sets, evaluating performance etc.). Your method should work for an arbitrary number of K (for K=1, K=3, K=5 etc. where 0< K ≤ total number of training samples).

Iris Dataset:

The Iris flower data set or Fisher's Iris data set is a multivariate data set. The data is collected to quantify the morphologic variation of Iris flowers of three related species. The dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. You can load the Iris dataset using scikit-learn (sklearn.datasets. load_iris)

For this assignment, you will first split your dataset in train and test sets (%70 training set,%30 test set), and then you will classify the test set data using KNN classifier algorithm you have implemented from scratch and the training set. You will evaluate your test set and report the accuracy you get for (K=1, K=3 and K=5). You can write up your findings along with the accuracies you received for K=1, K=3 and K=5 in a comment section at the end.

A *k*-nearest neighbors classifier.

- Stores the training set.
- To make a prediction for a new data point, the algorithm finds the point in the training set that is closest to the new point.
- Then it assigns the label of this training point to the new data point.

The *k* in *k*-nearest neighbors signifies that instead of using only the closest neighbor to the new data point, we can consider any fixed number *k* of neighbors in the training (for example, the closest three or five neighbors). Then, we can make a prediction using the majority class among these neighbors.

You will submit a jupyter notebook (ipynb file) with executable Python script with comments that explain the code. You should import all the libraries you will use at the top of your notebook. Please refer to course slides, tutorials and practicals to set up a running Python environment, Jupyter notebook and to import these libraries. You can check the documentation of each library (available online) to get more information about the functions you will use.

Tips:

You can use Euclidian distance to find the closes neighbors.

You can sort all distances to your query data point in the test set), then take the K neighbors.

Important Note:

You will receive points only if your script executes, the KNN is written from scratch and works for an arbitrary number of K, if you have covered each point mentioned (splitting the dataset, classification, evaluation), and written comments that explain each main step, shown findings (accuracy) for K=1, K=3 and K=5.

This is an individual assignment, meaning that you will be working on it alone (please check the Class Rules and Expectations below, also available in the syllabus)

Submission:

You will submit a jupyter notebook (ipynb file) with executable Python script and comments (explanations). The file will be uploaded on lms (guzem). You can upload a zip file that contains the jupyter notebook (ipynb file).

Grading:

The total is 100 points. You will receive points only if your script executes, the KNN is written from scratch, works for an arbitrary number of K, and works correctly, if you have covered each point mentioned (splitting the dataset, classification, evaluation), and written comments that explain each main step, shown findings (accuracy) for K=1, K=3 and K=5.

Course Rules and Expectations

All work on programming assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, however, everything that is turned in for each assignment must be your own work. In particular, it is not acceptable to: submit another person's assignment as your own work (in part or in its entirety), get someone else to do all or a part of the work for you, submit a previous work that was done for another course in its entirety (self- plagiarism), submit material found on the web as is etc. These acts are in violation of academic integrity (plagiarism), and these incidents will not be tolerated. Homeworks, programming assignments, exams and projects are subject to Turnitin (https://www.turnitin.com/) checks.