Seth Beckett and Ben Shaw
Project 4: CS 6830
Links: Slides, GitHub

# Part 1

## Introduction

In the first part of this project, we seek to predict whether a given patient has a heart disease using a k-nearest neighbor classifier. We examine a dataset[1] which we call the "cleveland" dataset, which gives features of just over 300 patients, such as the resting heart rate and age. Each input has 13 features, including five features which are treated as categorical.

## Methods

Some of the entries in the data are missing–rather, the values are denoted by "?". The problem persists in two distinct columns: in one column (denoted "ca"), we change "?" values to 0.0. In another column (denoted "thal"), we change "?" values to 3.0. These values represent the minimum values obtained by the data in the respective columns. In our training set, the targets are converted to either 1.0 or 0.0: some targets are originally 1.0, 2.0, 3.0, or 4.0, which targets are converted to 1.0.

Using pandas, we conduct one-hot encoding for the following categorical variables: "sex," "cp," "fbs," "exang," and "thal." Afterward, we have 23 input features. We then use sklearn's "Preprocessing" library to standardize the input data.

We wish to select only certain features of the data before turning to a k-nearest neighbors classifier. Rather than try to determine which of the original input features contain the most information, we turn to principal component analysis (PCA). By selecting the principal components whose associated singular values are highest, we choose the features which contain the most variation: the features which are most relevant for classification.

We implement a grid search for the k-parameter in the k-nearest neighbors classifier as well as the number p of principal components. Our grid contains the values of 3, 5, and 7 for both k and p. In each iteration, monte carlo cross validation is used with a test set proportion of 0.1. We implement our grid search three times, using the F1 score, and select a k value of 7 and a p value of 5, as this combination appears to have the best-performing F1 score.

[1]Available at https://archive.ics.uci.edu/ml/datasets/Heart+Disease

## Results

The best F1 score from the grid search occurred with k=7 and p=5, and the score was approximately 0.83. With these parameters, we turned to the sample test dataset, taking this dataset to be the test dataset and taking the entire cleveland dataset to be the training set. After standardization of the entire input dataset, we discovered our top five principal components from the training dataset and projected the test dataset onto those components. Then, we implemented a k-nearest neighbor classifier with k=7 and obtained an F1 score of approximately 0.92. We eagerly await implementing our algorithm for the full test dataset.

# Part 2

## Introduction and Dataset

The second part of this project focuses on implementing a k-nearest neighbor classifier for another dataset. We refer to the chosen dataset as the "leaf" dataset.[2] Each datapoint in the leaf dataset is a one-dimensional time series corresponding to the outline of a tree leaf. Each time series has length 427, and there are 482 data points in the set. There are six classes (types of leaves) in this dataset. The dataset requires no cleaning and is ready to use out-of-the-box.

## Methods

Among the various notions of distance between time series is the well-known measure Dynamic Time Warping (DTW). It is computationally expensive, though generally thought to be much better suited for computing pairwise distances between time series data than a simple euclidean metric. Thus, we implement a k-nearest neighbor classifier where distances are computed using DTW. However, since DTW is computationally expensive, no grid search was implemented: instead, a k-value of 5 was chosen somewhat arbitrarily.

## Results

The F1 score used in part 1 is not applied here, since there are six classes instead of simply two. Instead, the mean accuracy is used. 25% of the dataset was reserved as the test dataset. The algorithm–costly, on account of DTW–was then deployed, and an accuracy of approximately 0.65 was obtained, an improvement over randomly guessing, where an accuracy of less than 0.2 would be expected.

[2] Available at http://www.timeseriesclassification.com/description.php?Dataset=OSULeaf