1. **Vertebral Column Data Set**

   This Biomedical data set was built by Dr. Henrique da Mota during a medical residence period in Lyon, France. Each patient in the data set is represented in the data set by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (in this order): pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis. The following convention is used for the class labels: DH (Disk Hernia), Spondylolisthesis (SL), Normal (NO) and Abnormal (AB). In this exercise, we only focus on a binary classification task NO=0 and AB=1.[1]

   (a) Download the Vertebral Column Data Set from: `https://archive.ics.uci.edu/ml/datasets/Vertebral+Column`.

   (b) Pre-Processing and Exploratory data analysis:

      i. Make scatterplots of the independent variables in the dataset. Use color to show Classes 0 and 1.

      ii. Make boxplots for each of the independent variables. Use color to show Classes 0 and 1 (see ISLR p. 129).

      iii. Select the first 70 rows of Class 0 and the first 140 rows of Class 1 as the training set and the rest of the data as the test set.

   (c) Classification using KNN on Vertebral Column Data Set

      i. Write code for k-nearest neighbors with Euclidean metric (or use a software package).

      ii. Test all the data in the test database with k nearest neighbors. Take decisions by majority polling. Plot train and test errors in terms of k for $k \in \{208, 205, \ldots, 7, 4, 1, \}$ (in reverse order). You are welcome to use smaller increments of k. Which k^* is the most suitable k among those values? Calculate the confusion matrix, true positive rate, true negative rate, precision, and F_1-score when $k = k^*$.[2]

      iii. Since the computation time depends on the size of the training set, one may only use a subset of the training set. Plot the *best test error rate*,[3] which is obtained by some value of k, against the size of training set, when the size of training set is $N \in \{10, 20, 30, \ldots, 210\}$.[4] Note: for each N, select your training set by choosing the first $\lfloor N/3 \rfloor$ rows of Class 0 and the first $N - \lfloor N/3 \rfloor$ rows of Class 1 in the training set you created in 1(b)iii. Also, for each N, select the optimal k from a set starting from $k = 1$, increasing by 5. For example, if $N = 200$, the optimal k is selected from $\{1, 6, 11, \ldots, 196\}$. This plot is called a *Learning Curve*.

   Let us further explore some variants of KNN.

[1] Make sure that you convert labels to 0 and 1, otherwise you may not obtain correct answers.

[2] We will learn in the lectures what these mean, for now research how they are computed and compute them.

[3] Obviously, use the test data you created in 1(b)iii

[4] For extra practice, you are welcome to choose smaller increments of N.

(d) Replace the Euclidean metric with the following metrics[5] and test them. Summarize the test errors (i.e., when $k = k^*$) in a table. Use all of your training data and select the best k when $\{1, 6, 11, \ldots, 196\}$.

i. Minkowski Distance:

A. which becomes Manhattan Distance with $p = 1$.

B. with $\log_{10}(p) \in \{0.1, 0.2, 0.3, \ldots, 1\}$. In this case, use the k^* you found for the Manhattan distance in 1(d)iA. What is the best $\log_{10}(p)$?

C. which becomes Chebyshev Distance with $p \to \infty$

ii. Mahalanobis Distance.[6]

(e) The majority polling decision can be replaced by weighted decision, in which the weight of each point in voting is *inversely proportional* to its distance from the query/test data point. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away. Use weighted voting with Euclidean, Manhattan, and Chebyshev distances and report the best test errors when $k \in \{1, 6, 11, 16, \ldots, 196\}$.

(f) What is the lowest training error rate you achieved in this homework?

[5] You can use sklearn.neighbors.DistanceMetric. Research what each distance means.

[6] Mahalanobis Distance requires inverting the covariance matrix of the data. When the covariance matrix is singular or ill-conditioned, the data live in a linear subspace of the feature space. In this case, the features have to be transformed into a reduced feature set in the linear subspace, which is equivalent to using a pseudoinverse instead of an inverse.