1. **Problem Description**: In this Lab, you will implement KNN with different distance metrics on a Breast Cancer data set. You can find the dataset at

   https://archive.ics.uci.edu/ml/datasets/Breast+Cancer

   The goal is to predict from 9 features whether breast cancer patients had recurrence events or not. The features are all *categorical* features, in the sense that they have *levels*. For example, the levels of node-caps are yes and no. The levels of breast-quad are left-up, left-low, right-up, right-low, and central. The dataset includes 201 instances of no recurrence and 85 instances of recurrence. You are given training data that has 160 instances of the no-recurrence class and 68 instances of the recurrence class.

   Your task is to determine the so-called hyper-parameters of the algorithm, which are k and the distance metric used to determine the nearest neighbors.

2. **Data pre-processing**: There are many steps that are needed for most datasets before a Machine Learning algorithm can be implemented on them. For this dataset:

   (a) First, replace missing values of each feature with the *mode* of that feature. For example, if the feature node-caps is missing in an instance, replace the missing value with whichever level of node-caps is more common in your data (yes or no?). In this data set, missing values are shown with '?'

   (b) Dealing with categorical features: Machine Learning algorithms work with numeric values, so categorical features have to be converted to numeric features. Some categorical features are ordinal, which means there is an order among levels. An example of such a feature is inv-nodes. The smallest numbers of nodes were encoded to 0-2, while the largest were encoded to 36-39.

   The ordinal features in this data set are: age, tumori-size, inv-nodes, and deg-malig. Convert their levels to numbers. For example for inv-nodes, 0-2 should be converted to 1, 3-5 should be converted to 2, 6-8 should be encoded to 3, and so on.

   On the other hand, some categorical features do not have any order. The features menopause, node-caps, breast, breast-quad, and irradiat are such features in this dataset. If such a feature has only two levels, its two levels will be converted to 0 and 1 in the data set. For example, node-cap=yes, becomes nodecap = 1. If the feature has more than one level, we replace it with one *auxilliary* binary feature for each level. For example, breast-quad has five levels: left-up, left-low, right-up, right-low, central. It will be replaced with five auxiliary features. Now, assume that the value of breast-quad is right-up for an instance. The values of those auxiliary features will be: left-up=0, left-low=0, right-up = 1, right-low=0, central=0. This way of converting the categorical features to numeric features is called one-hot encoding.

3. **Determining hyperparameters using a test set**: We will use l-norm based distance:

$$d_l(x, x^*) = \left(\sum_{j=1}^{p} |x_j - x_j^*|^l \right)^{1/l}$$

in which x_j is the j^{th} numeric feature of instance x in the training set, and x_j^* is the j^{th} feature of the test point x^*. We also use consider the edit distance. When $l \to 0$, $d_l(x, x^*)$ can be shown to be equal to the number of corresponding elements of x and x^* that are not equal. When $l \to \infty$ $d_l(x, x^*)$ can be shown to be equal to the maximum difference between elements of x and x^*. Your code should use a function that a distance function $\textsc{distance}(L, x, x^*)$ that acts as follows:

$$\textsc{distance}(L, x, x^*) = \begin{cases} \text{edit distance} = \text{number of non-matching elements of } x \text{ and } x^* & L = -1 \\ d_L(x, x^*) = \left(\sum_{j=1}^{p} |x_j - x_j^*|^L \right)^{1/L} & L \in \{1, 2, 3, 4, 5, 6\} \\ \max_{j=1}^{p} |x_j - x_j^*| & L = 1000 \end{cases}$$

Note that the choice of $L = -1$ for the edit distance (whose code is given to you) is only symbolic to represent $\lim_{l \to 0} d_l(x, x^*)$

Consider $L \in \{-1, 1, 2, 3, 4, 5, 6, 1000\}$ and $k \in \{1, 2, 3, \ldots, 30\}$.

For each pair of (k, L), and for an instance x^* in the test set, determine the k nearest neighbors of x^* in the training set, using the distance metric $\textsc{distance}(L, x, x^*)$, and predict a label for x^* based on the majority of the labels of the nearest neighbors. Break possible ties in favor of the positive class, i.e., recurrence-events. Compare the majority vote label you predicted from the training set with the true label of x^*. Next, calculate the percentage of misclassified points in your test set, i.e. the percentage of the labels in the test set that were predicted incorrectly. Summarize the results in a data structure and print them in a table. Find the k and L that result in the lowest misclassification rate in the test set.

Here is how your code will be graded:

- General soundness of code: 60 pts.
- Passing multiple test cases: 40 pts. The test cases will be based on different splits of the data into training and testing. Your code should return the correct (k, L). There might be more than one pair of (k, L) that have the lowest misclassification rate (or highest accuracy = 1-misclassification rate)

4. (20 pts **Extra Credit**, **Weighted kNN**):

   Implement Weighted kNN, in which you should convert the labels of instances tino numeric labels (no-recurrence-events = 0 , re-currence-events=1), and do kNN like above, with the exception that instead of majority polling, you should use weighted majority polling, which calculates a score for each test point x^* based on its distances from its K-Nearest neighbors in the training set, $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$. Assume the numeric labels of those k nearest neighbors are $y_1, y_2, \ldots, y_k$. Calculate the following score for x^*:

$$score(x^*) = \frac{\sum_{i=1}^{k} y_i / [\text{DISTANCE}(L, x^{(i)}, x^*)]}{\sum_{i=1}^{k} 1/[\text{DISTANCE}(L, x^{(i)}, x^*) + \epsilon]}$$

in which you are giving more weight to the neighbors that are closer to x^*. Here, we assume $\epsilon = 0.0001$. Then if $score(x^*) \geq 0.5$, you classify it in the positive (recurrence-events) class, otherwise in the negative (no-recurrence-events). Everything else will be the same as vanilla kNN. Determine the best (k, L) based on test misclassification rate.

Note: You must submit Extra Credit along with your main homework. If you submit Extra Credit Late, your whole homework will be late.