Based on the results of kNN1, I get a classification accuracy of 48.0% (when $k = 1$). In my kNN algorithm, I calculate the Euclidean Distance between features. Due to the nature of the Euclidean Distance, the highest prediction accuracy will be in the case where all features have the same scale (they are standardized). For clarity, let's provide a mathematical example: we have two features located on a plane with x and y axes. The scale of points on the x-axis is $\frac{1}{100}$, and the scale of points on the y-axis is 100. Let's calculate the Euclidean Distance between two random features. Suppose we get $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = \sqrt{0.01^2 + 100^2} \approx 100.0000005$; now, for the sake of example, let's eliminate the x-value: $\sqrt{(x_1 - x_2)^2} = 0.01$. We then get $\sqrt{(y_1 - y_2)^2} = 100$. As we can see, , even after eliminating the x-value, we obtained approximately the same distance in both examples. Thus, the first feature is devalued due to the significantly larger scale of the second feature. This results in a disruption of the balance between the influence of input variables, represented in different scales, on the output variable. This influence is conditioned not by the real dependency, but by the change in scale. We want to consider all features equally, so they all must have the same scale (be standardized). To achieve this, we need to normalize our train and test data.

In machine learning, normalization refers to the method of preprocessing features in train data to bring them to a common scale without losing information about the differences in ranges. I will use Z-normalization for the kNN2 algorithm. The formula for Z-normalization is:

$$Z = \frac{X - \mu}{\sigma}$$

where x is the observed value, μ is the mean, and σ is standard deviation.

Let's demonstrate how we use Z-normalization to standardize data for calculations and the operation of the algorithm. Let's find the $\mathbb{E}(X)$ (expected value) and $Var(X)$ (variance) of $Z = \frac{X - \mu}{\sigma}$. Let X be the random feature in the train dataset, $\mu = \mathbb{E}(X)$ is the mean of feature X, and $\sigma = \sqrt{Var(X)}$ is the standard deviation.

Expected value:

$$\mathbb{E}(Z) = \mathbb{E}\left(\frac{X - \mu}{\sigma}\right) = \underbrace{\mathbb{E}\left(\frac{X}{\sigma}\right)}_{=\frac{\mu}{\sigma}} - \underbrace{\mathbb{E}\left(\frac{\mu}{\sigma}\right)}_{=\frac{\mu}{\sigma}} = 0$$

(As $\mathbb{E}(X) = \mu$, then $\mathbb{E}\left(\frac{X}{\sigma}\right) = \frac{\mu}{\sigma}$; as the expected value of the constant is the constant itself, then $\mathbb{E}\left(\frac{\mu}{\sigma}\right) = \frac{\mu}{\sigma}$).

Variance (I will use $\sqrt{\sigma^2}$ for clarity):

$$Var(Z) = Var\left(\frac{X - \mu}{\sqrt{\sigma^2}}\right) = Var\left(\frac{X}{\sqrt{\sigma^2}}\right) + Var\left(\frac{\mu}{\sqrt{\sigma^2}}\right) = \frac{1}{\sigma^2} \cdot \sigma^2 + 0 = 1$$

(As the variance of a constant is zero, then $Var\left(\frac{\mu}{\sqrt{\sigma^2}}\right) = 0$; There is a property that a constant is extracted from under the variance sign squared, so $Var\left(\frac{X}{\sqrt{\sigma^2}}\right) = \frac{1}{(\sqrt{\sigma^2})^2} Var(X) = \frac{1}{\sigma^2} \cdot \sigma^2 = 1$).

We found that for a random feature, as a result of z-normalization, the Expected Value (Mean) $\mathbb{E}(Z) = 0$ and the Variance $Var(Z) = 1$, which indicates that all features in the train data are standardized.

Let's implement Z-normalization in the kNN2 algorithm.

```
double temp[][][][] = Normalize(train, test);
train = temp[0];
test = temp[1];
```

After loading data into the train and test arrays from the text files train_data and test_data respectively, and before finding the Euclidean Distance, we initiate a 3D array that will eventually store the normalized train array at position [0] and test array at position [1] (Initially, I wanted to use Pair, but it would require the use of the additional javafx library). The Normalize method is called with the train and test arrays as parameters.

```
public static double[][][][] Normalize(double[][][] train, double[][][] test) {
    double[] mean = new double[FEATURE_SIZE];
    double[] standardDeviation = new double[FEATURE_SIZE];;

    for (int i = 0; i < FEATURE_SIZE; i++) {
        double sum = 0;
        for (int j = 0; j < TRAIN_SIZE; j++) {
            sum += train[j][i];
            sumOfSquares += train[j][i] * train[j][i];
        }
        mean[i] = sum / TRAIN_SIZE;
        standardDeviation[i] = Math.sqrt((sumOfSquares / TRAIN_SIZE) -
                                (mean[i] * mean[i]));
    }

    for (int z = 0; z < TRAIN_SIZE; z++) {
        for (int v = 0; v < FEATURE_SIZE; v++) {
            train[z][v] = (train[z][v] - mean[v]) / standardDeviation[v];
        }
    }
    for (int z = 0; z < TEST_SIZE; z++) {
        for (int v = 0; v < FEATURE_SIZE; v++) {
            test[z][v] = (test[z][v] - mean[v]) / standardDeviation[v];
        }
    }

    double[][][][] NormalizeOutput = new double[2][TRAIN_SIZE][FEATURE_SIZE];
    NormalizeOutput[0] = train;
    NormalizeOutput[1] = test;
    return NormalizeOutput;
}
```

First, we find the sum of features in one column (the 1st feature in the 1st row + the 1st feature in the 2nd row + ... + the first feature in the 200th row). Then, we find the sum of the squares of each feature in one column (the 1st feature in the 1st row squared + the 1st feature in the 2nd row squared + ... + the 1st feature in the 200th row squared). Then, we find the mean from one column using the formula:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

where x_i is feature value in the column, $\sum x_i$ is the sum of feature values in one column found previously and n is the number of feature values in one column (equal to the number of rows (patterns) = TRAIN_SIZE).
Then we find the standard deviation from one column of feature values:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \overline{X}^2$$

$$s.d. = \sigma = \sqrt{\sigma^2}$$

Where $(\sum_{i=1}^{n} x_i^2)$ is the sum of squares, n is the number of patterns (rows), $\overline{X}^2$ is the squared mean, σ^2 is variance and $s.d.$ (σ) is standard deviation.

Now we can Z-normalize our dataset. We apply the formula $Z = \frac{X - \mu}{\sigma}$ to each feature in our train and test data. It is important to mention that we will not calculate the expected value (mean) and variance using test data, and will only use data obtained from train to normalize the test data. There are reasons for that:

1. The first reason, the most important in the context of the kNN algorithm, is that during training we are only allowed to use the training data, we cannot use the test data;

2. The second reason is that the distribution of features in the training data and the test data can be (and most likely will be) different. Accordingly, if we use other data (obtained from test data) for normalizing the test data (Variance and Mean), then we will get different normalization for train and test, but we need the same to have the same scale (standardization).

As a result of using Z-normalization, I get a classification accuracy of 94.5% (when $k = 1$) in kNN2.

References

1. https://wiki.loginom.ru/articles/data-normalization.html#normalizatsiya-srednim-z-normalizatsiya
2. http://www.statology.org/z-score-normalization/
3. http://www.mun.ca/biology/scarr/Mean_&_Variance.html
4. https://en.wikipedia.org/wiki/Variance