**Nearest Neighbors Implement from Scratch**

**Mingkai Ma z5135897**

**COMP9417, Assignment 2**

**Introduction**

K-nearest neighbor is a basic non-parametric method for classification and regression. In both cases, the input consists of the k closest training examples in the feature space and the output depends on whether the algorithm is used for classification or regression. K-NN algorithm is based on feature similarity. For k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors. For k-NN regression, the output is the property value for the object, the value is the average of the values of its k nearest neighbors. Under some circumstances, we may want to weight nearer neighbors more heavily, therefore, both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more.

The goal of this project is to implement k-NN for both classification and regression, test the k-NN models with two datasets provided and evaluate the models by leave-one-out-cross-validation. In addition to implementing the basic k-NN algorithm, the distance-weighted nearest neighbor (WNN) is also implemented. At the end, some diagrams have been plotted to demonstrate the effect of the k parameters.

**Implementation**

k-NN is a type of instance based and lazy learning. The distance metric used in the project is Euclidean distance. For a given test instance, we calculate the distance between the test instance and all instances in training data, and locate the top k nearest training instances. For classification, taking vote among these k nearest neighbors, the test instance is classified by a majority vote of its neighbors. For regression, the value of test instance is the mean of the values of these k nearest neighbors.

Implementing the distance-weighted nearest neighbors requires additional steps. For WNN, after get the k nearest neighbors, weights for every neighbor is calculated by:

and the decision function for classification is changed to:

the decision function for regression is changed to:

Then leave one out cross validation is used to evaluate the system. One instance in the dataset is used once as test data, and the rest of the instances are used as training data. Repeat the process until all instances have been used as test data. In the project, for classification dataset, there are 351 instances in the training data, so 351 validation process is performed, for regression data set, there are 159 instances (after removing examples with missing values as required), 159 validation process is performed.

In the project, the k-NN algorithm is implemented in Python. There are two class:

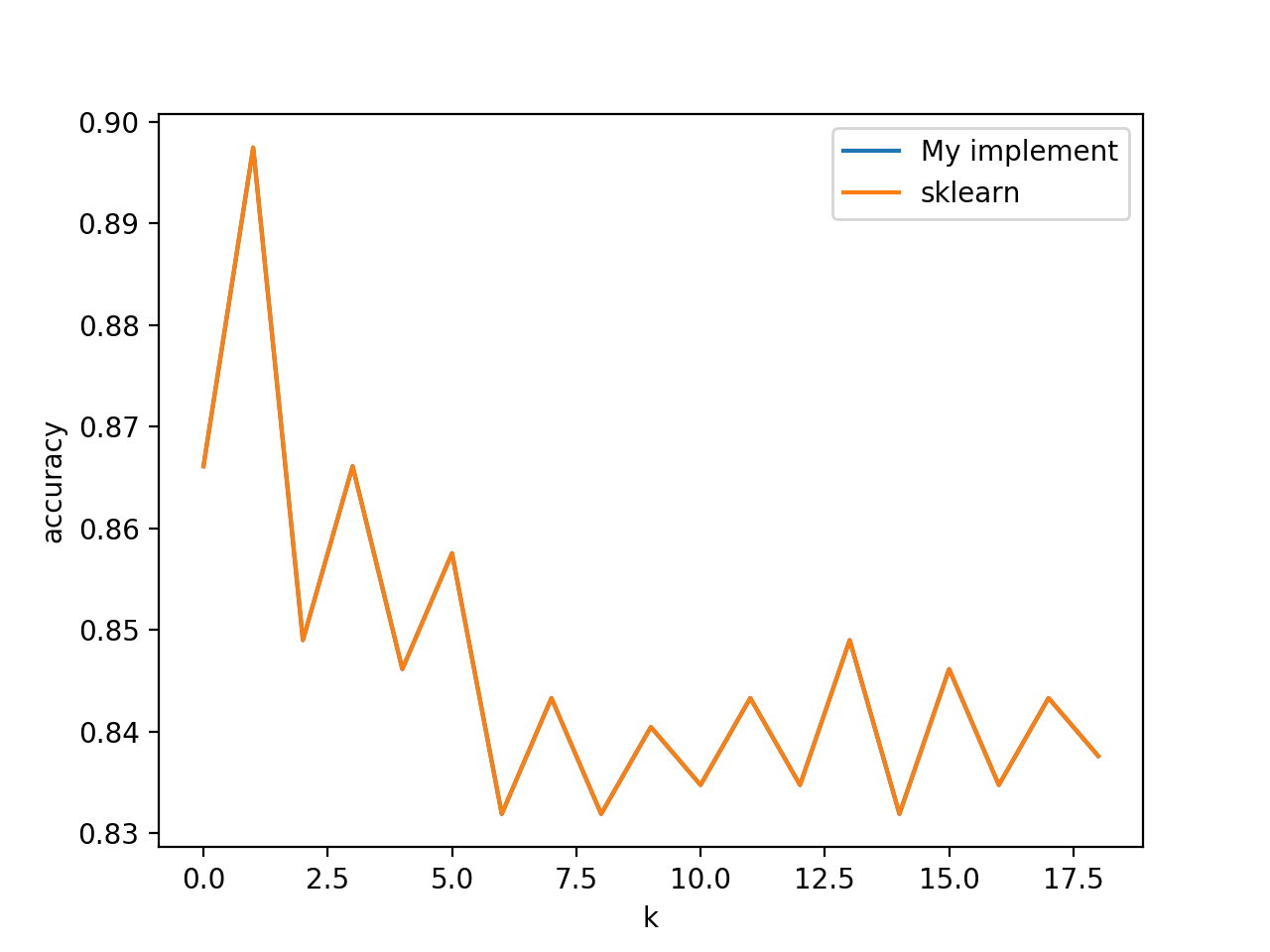




**Experimentation**

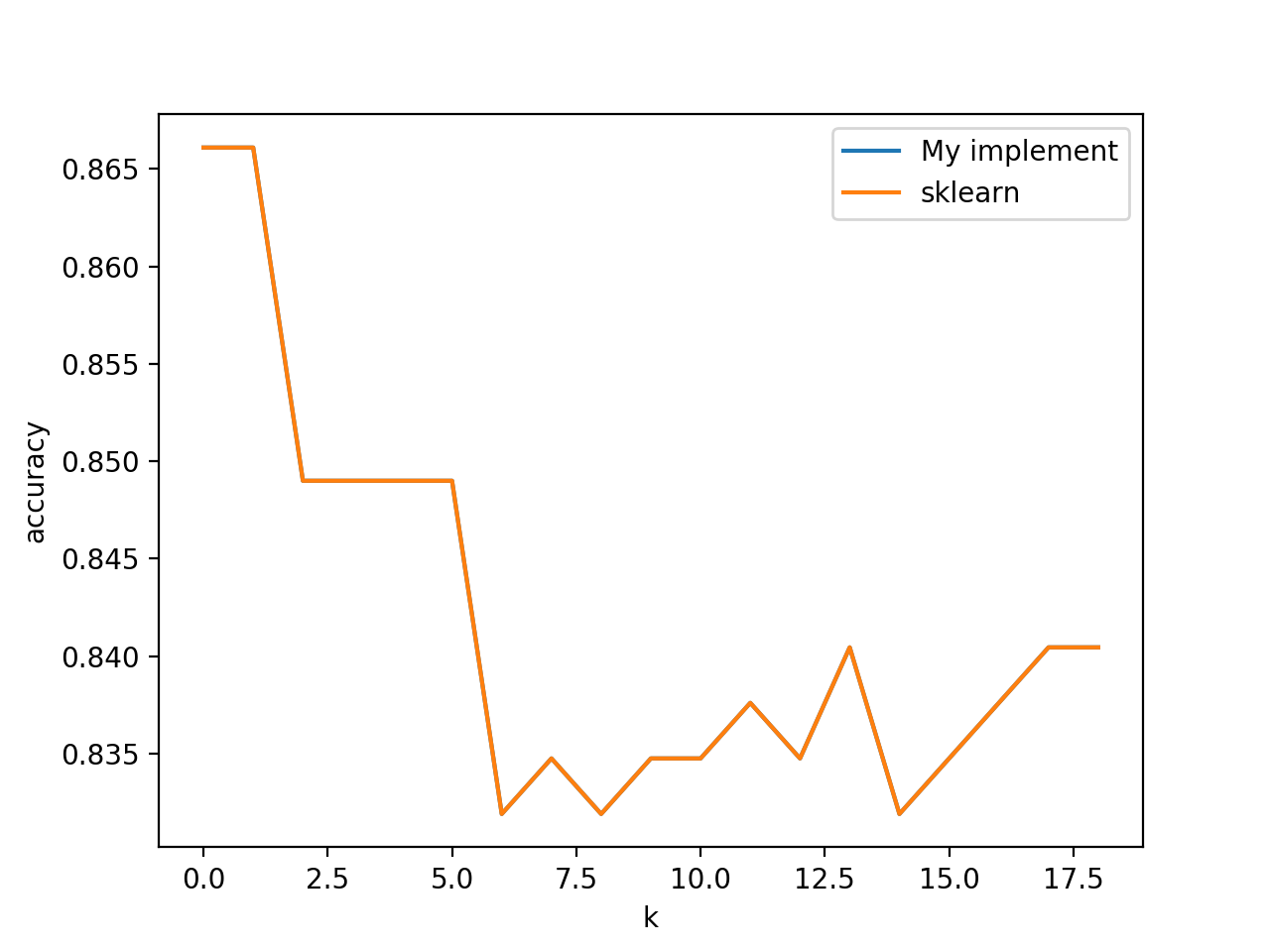
In the project, k-NN classifier is tested on the standard UCI data set ionosphere as provided. Leave-one-out-cross-validation is used to evaluate classifiers. For the purpose of testing, KNeighborsClassifier from sklearn is used. 20 leave-one-out-cross-validation is performed for the range of k from 1 to 20, the result is as follow:

No distance-weighted:



There are two lines in the diagram. The results from my implement is exactly the same with the results from sklearn, so only one line is displayed.

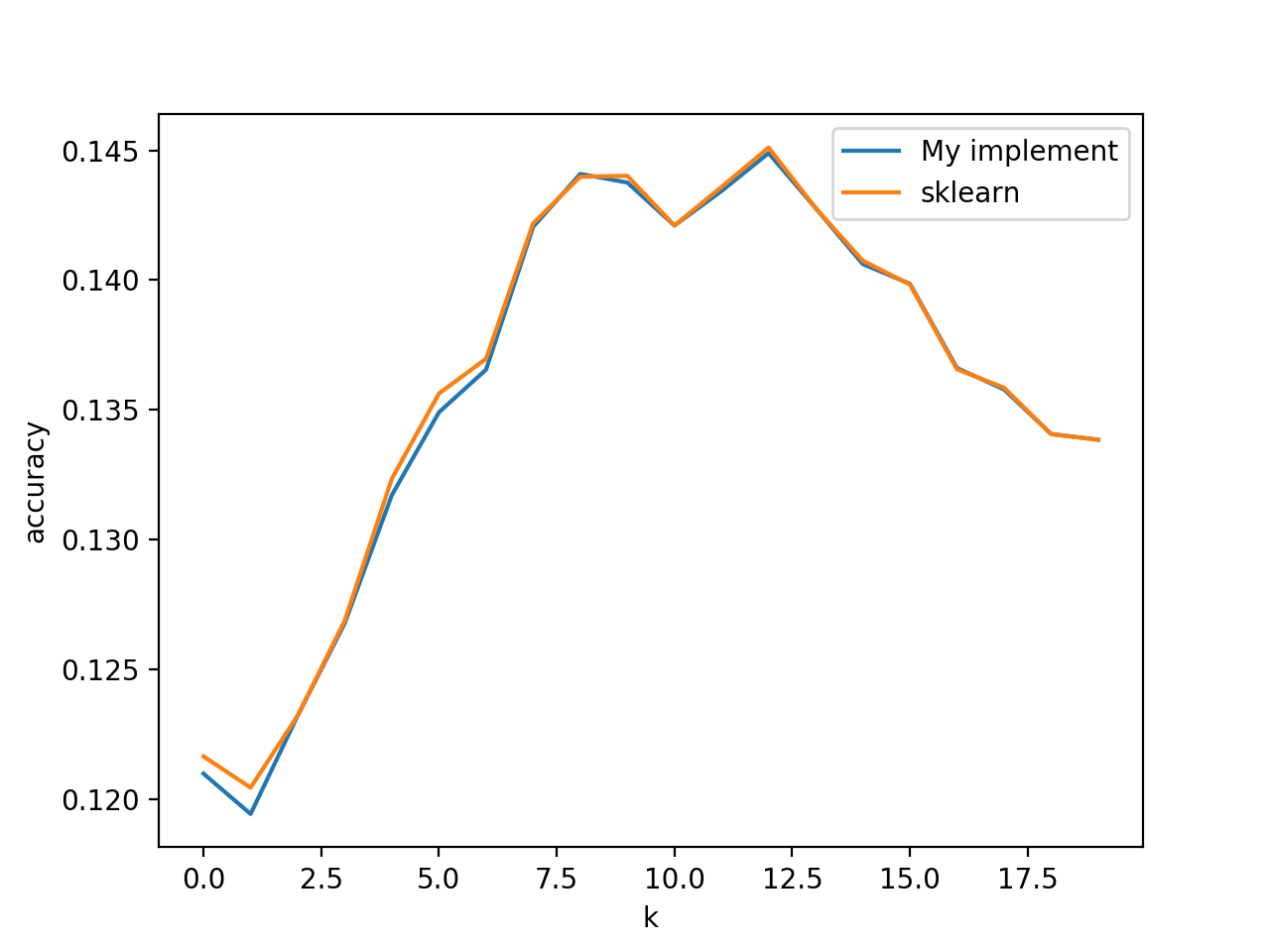
Distance-weighted:



There are two lines in the diagram. The results from my implement is exactly the same with the results from sklearn, so only one line is displayed.

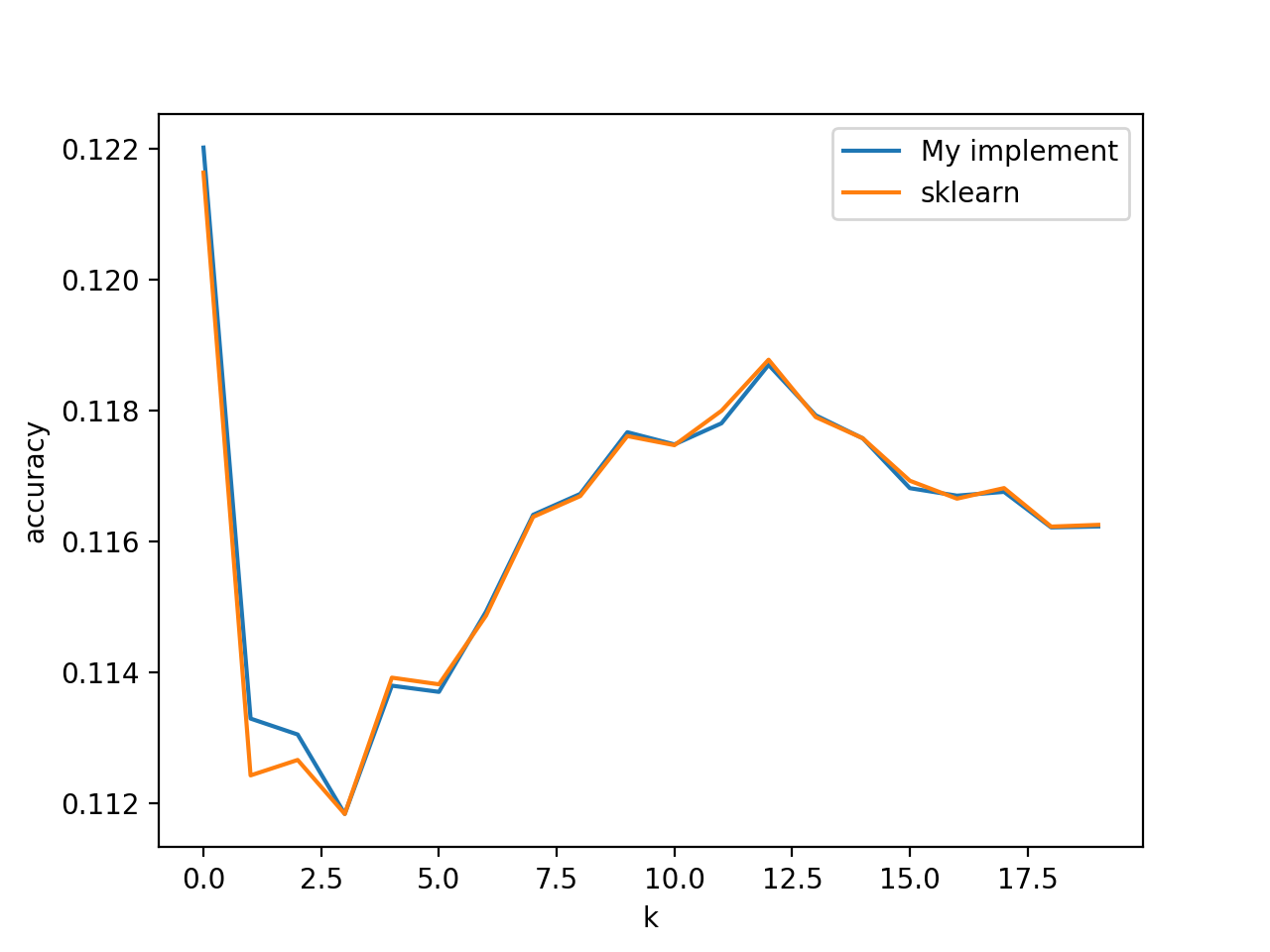
k-NN regressor is tested on the standard UCI data set autos for range of values of k from 1 to 20. KNeighborsRegressorfrom sklearn is used. 20 leave-one-out-cross-validation is performed for the range of k from 1 to 20, the result is as follow:

no distance-weighted:



notice that there are slightly difference. That’s because there are some instances that have the same distance, my implement choose the same instances in random order.

Distance-weighted:



notice that there are slightly difference. That’s because there are some instances that have the same distance, my implement chooses the same instances in random order.

**Conclusion**

From the diagrams above, it is obvious that the value of k has influence on the performance of the k-NN system, and distance-weighted nearest neighbors algorithm is slightly better than the k-NN.

**Reference**