Performance Analysis of Knn on wave-forms data-set

Anna Karolin Amrithya Balaji

Abstract

The performance of the K-Nearest Neighbors (KNN) algorithm on a challenging wave-forms dataset is explored in this study. We analyze various configurations, including different values for K and distance metrics, and assess KNN's effectiveness using metrics such as accuracy.

1. Introduction

The dataset used in this study has 5000 instances, each of which represents a distinct wave type and is defined by 21 attributes, all of which are embedded with noise. Three different wave classes are involved in the classification task, and an 86% accuracy Bayes classification rate is the ideal setting. We have developed a series of experiments to achieve improved classification, which include optimizing the k-parameter in a k-Nearest Neighbors (kNN) classifier, analyzing techniques for data reduction, and evaluating gains in computational efficiency for the 1NN algorithm.

We also experiment with how the artificially created imbalances in the training data affect the 1000 test wave classification accuracy. Specifically, we will adjust the k-parameter with an emphasis on the F-measure, offering a thorough examination of performance metrics concerning imbalanced datasets.

2. Tuning the best k

KNeighborsClassifier: On a dataset, the k-Nearest Neighbors (KNN) classifier assigns a label to a new point depending on it's nearest neighbours seen during training. The classifier goes through a hyperparameter tuning process after being initially implemented with a default value of neighbors to find the number of neighbours to be considered for highest possible accuracy. Hyperparametrer tuning is necessary for machine learning models in general for better accuracy results during test time.Through a methodical investigation of hyperparameter values, this technique facilitates the determination of the configuration that, when tested via cross-validation, produces the best results.

The hyperparameter tuning is done for the model taking initial k value as 5.To estimate the accuracy a 5-fold cross-validation is carried out for each k values, ranging from 1 to 99. The accuracies_knn list contains the mean accuracy scores. The findings show that, with an accuracy of 85.6%, k should be optimized to be 58 for the dataset.

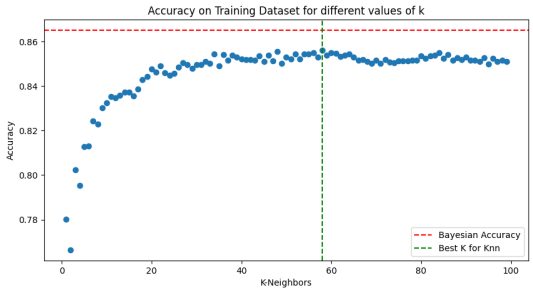

Figure 1. Accuracy for different values of K

Data Reduction: Before training a k-Nearest Neighbors (KNN) classifier, we have used Condensed Nearest Neighbors (CNN) technique. This method allows us to lower the number of sample points by deleting the points that are redundant(points towards the centre of any cluster/group) and points which are always misclassified(outliers and points in the overlapping regions of two clusters). This method speeds up the calculations but at the cost of accuracy as we are deleting sample points.

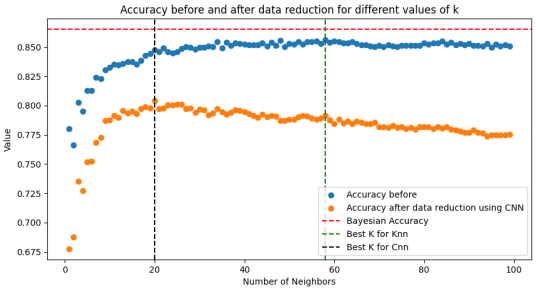

Figure 2. Comparison of Knn and Cnn

To determine the ideal number of neighbors (k) for the KNN classifier trained on the CNN-resampled data, a sim-

ilar hyperparameter tuning procedure is then used. The cross-validation accuracy is calculated for each k as the code iterates through a range of values from 1 to 99. The accuracies_cnn list contains the mean accuracy scores. The outcomes show that 20, along with a matching accuracy, is the ideal k for the CNN-resampled data. With a black dashed line denoting the ideal k value found by the CNN technique and a red dashed line representing a reference Bayesian accuracy, the plot illustrates the relationship between the number of neighbors and the cross-validation accuracy.

3. Speeding Up the calculations

Two algorithms are used to speed up that calculations.

3.1. Using KD trees and Ball trees

We select a point at random from the training set and construct a tree with child nodes as the nearest neighbours till we cover all the points. When given a test case we traverse through the tree such that at the given point, the selected node is the nearest neighbour and we explore the children. This method allows us to avoid many sub optimal calculations. But being a greedy approach, the accuracy is lower.

The kdtree and balltree algorithms performed much faster, taking 103 ms and 94.4 ms, respectively, compared to the 1nn algorithm, which took 611 ms.

3.2. By eliminating non viable cases for 1NN(Brute Force)

In this method for each point in the test case we take a point from the training set at random and calculate the distance between them and assign it to minimum distance. We take another point for the training set and find the distance to our test point and if the distance is less than minimum distance we update it. We eliminate all the points in the training who lies inside the circle of radius distance calculated - minimum distance and those outside the circle distance calculated + minimum distance as they cannot be the nearest neighbour by triangle inequality. For each point in the test set we make a copy of the training set and do the above procedure till that copy is empty.

4. Generating artificial imbalance

4.1. Over Sampling a Class

We have created artificial imbalance in the training set using RandomOverSapmler increasing the points in the classes 0 and 2 and explored how the model is affected by the imbalance using accuracy and f1 score. Without imbalance the accuracy and f1 score are 0.825 and 0.825 respectively. When make the data imbalance by making number of data points in the training set of class 2 to be 3000 and 10000, the accuracy values decrease to 0.816 and 0.792 respectively and the f1 score changes to 0.814 and 0.789 respectively.

When the same procedure is done for class 0 in place of class 2, the accuracy values are 0.813 and 0.786 and the f1 values are 0.814 and 0.787. These changes occur as the more the number of data points in the training set for a particular class, the model favors that class.

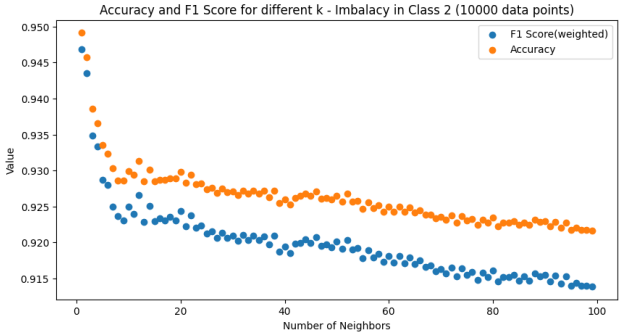

Figure 3. Accuracy vs F1 score after generating artificial imbalance in the dataset (class 2 containing 10000 data points while training)

4.2. Taking an imbalanced sample from the training set

A more imbalanced distribution is guaranteed by the sampling_strategy parameter, which specifies 500 instances for class 0, 1000 instances for class 1, and 1383 instances for class 2. This over-sampled data is then used to train a k-Nearest Neighbors (KNN) classifier with 20 neighbors.

The findings show that on the over-sampled dataset, the KNN model with 20 neighbors obtains an accuracy score of 0.846 and an F1 score of 0.843.

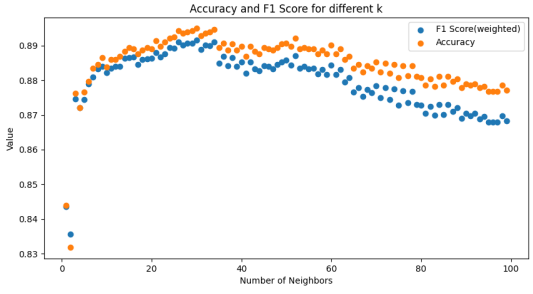

Figure 4. Accuracy vs F1 score after generating artificial imbalance in the dataset

References

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.