**kNN TigerFish Report**

**Dane Acena**

# **Problem Description**

Given a data set from the Clemson Wildlife and Fisheries Biology graduate students of two newly discovered species of fish in Lake Hartwell, develop a kNN classification algorithm that will determine a fish is a TigerFish1 (the positive case) or a TigerFish0.

# **Data Description**

The initial data consisted of 300 records representing features of either TigerFish1 species or TigerFish0 species with three tab-delimited entries. The first two are floats indicating the measured body length and dorsal fin length of each fish, respectively. The last element is a digit, either “1” or “0” identifying the species of fish as either “TigerFish1” or “TigerFish0”. A plot of the initial dataset is as shown in Figure 1.

![A screenshot of a cell phone

Description automatically generated]()

Figure : The Initial Data Set

# **Training a kNN Algorithm**

The k Nearest Neighbor algorithm was developed using 5-fold cross-validation. When the data set is loaded it gets randomized and split 80-20. The 80% goes to a *Training Set,* and the remaining 20% goes to a *Test set*, which will be set aside be put to use later on. The algorithm further divided *Training Set* into five folds with 48 records each. From those five sets, the algorithm recursively creates a *Train[i].txt* (contains 192 records each) combining four sets and create *Val[i].txt* (contains 48 records each) with the leftover fold. **validationMode()** recursively executes Train[i].txt and Val[i].txt via **crossValidate()** with odd values of k from 1 through 21. In each iteration of the test, each record of the Val[i].txt set gets tested against the Train[i].txt set and **getNeighbors()** looks at the *k* points nearest the *val* record using **euclideanDistance()**. Then, **getResponse()** determines the majority type of neighbor**.** The accuracy and error are acquired through **getAccuracy()** and **getError()** respectively.

For each test iteration of each test set and k value, the number misclassification is recorded in a tab-delimited file, as shown in Figure 2.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 |
| Test1 Errors | ﻿6 | 5 | 5 | 6 | 6 | 6 | 6 | 7 | 6 | 6 | 6 |
| Test2 Errors | 3 | 3 | 3 | 3 | 4 | 4 | 5 | 5 | 5 | 5 | 5 |
| Test3 Errors | 9 | 7 | 7 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| Test4 Errors | 5 | 6 | 5 | 3 | 4 | 4 | 4 | 4 | 5 | 4 | 5 |
| Test5 Errors | 5 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 |
| TOTALS | **28** | **23** | **22** | **20** | **23** | **23** | **24** | **25** | **25** | **23** | **24** |

Figure : Misclassifications for different values of k on the five training sets­­

Additionally, to visually illustrate the accuracy a plot was also produced by the algorithm, as shown in Figure 3.

# 

Figure : Average accuracy for different values of k (PLACEHOLDER)

As shown above, k=7 provided the best accuracy. K=7 was chosen for the test set with the k Nearest Neighbor algorithm.

# **Results**

Results of the validation of the test set against the training set are shown in the confusion matrix as shown in figure 4

The test set contained 34 records representing TigerFish0 and 26 records representing TigerFish1. 57 of the 60 records identified correctly; denoting an *accuracy* of 0.95. *Precision* is 0.96, out of the 25 that were predicted to be TigerFish1, one was TigerFish0. The *recall* is 0.92, two out of the 35 that were predicted to be TigerFish0 are misclassified. The **F1** score is 0.94

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predicted TigerFish1 | |
| N | Y |
| Actual  TigerFish1 | N | TN =33 | FP =1 |
| Y | FN =2 | TP =24 |

Figure : Confusion Matrix