Abstract:

K-nearest neighbor algorithm (KNN) is a supervised machine learning algorithm that can be used to solve classification and regression problems. In this report, KNN is used to classify the age of abalones (which can be found using the number of rings of an abalone) based on 8 other attributes (sex, length, diameter, height, whole weight, shucked weight, viscera weight and shell weight). Further evaluation is conducted on the KNN algorithm using the train-and-test split and k-fold cross validation method by comparing the algorithm's accuracy and computational time.

Based on the comparison findings as described in this report, it is observed that the accuracy of the KNN model to classify the age of abalones yields a low accuracy score of 24%. This is consistently seen in both the train-and-test split method as well as the k-fold cross validation method. However, the computational time using the k-fold cross validation method take around 3 times longer than the train-and-test split method.

Introduction:

In this report, the KNN algorithm will be applied in the multi-class classification of the age of abalones using the 'Abalone' dataset.

Classification requires a training dataset with many examples of inputs and outputs from which to learn. Class labels must be mapped to numeric values before being provided to an algorithm for modelling. A model will then use the training dataset to calculate the best way to map the examples of input data to specific class labels. The modelling algorithms are then evaluated based on their results, for example, based on classification accuracy of the predicted class labels.

For this dataset, there are 4,177 data observations with 8 input attributes (sex, length, diameter, height, whole weight, shucked weight, viscera weight and shell weight) and 1 output variable (number of rings of an abalone, ranging from 1 to 29 rings). Based on the input attributes, the classification model will try to predict the age of the abalones.

The KNN algorithm relies on labelled input data, to learn a function that produces an appropriate output when given new unlabelled data. It works on the premise that similar things are near to each other. Hence by calculating the distance between the test points and the K number of selected training points closest to the test points, the outcome of the test points can be predicted by obtaining the most frequent label within the K number of selected training points. As KNN relies on computing the distances, scaling of data should be done before running the KNN.

KNN is thus used to solve the classification problem described in this report, with the output being the discrete value, rings of an abalone.

Distance calculations used in the KNN algorithm can be calculated by various methods. In this report, Euclidean distance is used as the similarity measure for any two samples and the formula to calculate the distance between the test points and its K-nearest neighbors is given as:

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

After calculation of the Euclidean distance, this is stored in an ordered manner along with the labels of the neighboring points. This ordered collection of distances and labels are then sorted in ascending order by the distances. The first K entries from the sorted collection is then selected and the mode of the K labels is returned. This gives us the predicted output of the test points based on our model.

The value of k in the KNN algorithm is related to the error rate of the model. A small value of k could lead to less stable predictions and overfitting. As the value of k increases, the predictions become more

stable. However, beyond a certain point, it may result in the predictions being skewed due to the larger influence from incorrect nearest neighbors.

Further evaluation is conducted on the KNN algorithm using the train-and-test split and k-fold cross validation method. The train-and-test method splits the data into two subsets, a training data set and a testing data set. The size of the two subsets is commonly expressed as a percentage, example, 60% of the dataset is split into the training data set and 40% of the dataset is split into the testing data set. The model is fitted on the training data set and evaluated on the testing data set. The train-and-test split is useful due to its flexibility and speed.

The k-fold method splits the given sample data to k number of groups. For a single unique group, it is taken as a test data set and the remaining groups are taken as a training data set. The model is fitted on the training data set and evaluated on the testing data set. This procedure is then repeated for each unique group. This means that each sample is used as a test data set once and used to train the model k-1 times. Since the model is trained k-1 times, this method is generally less biased than the train-and-test method which trains the model only once.

Accuracy:

The classification performance of the KNN algorithm for each of the experiments conducted is summarised in table 1. Based on the comparison findings, it is observed that the accuracy of the KNN model to classify the age of abalones yields a low accuracy score of around 24%. This is also observed based on the accuracy given in the classification report in table 2 for the 5-fold cross validation with K = 15 using the classification_report() function provided by the *scikit-learn* library.

As shown in figure 1, the accuracy score is comparable in both the train-and-test split method as well as the k-fold cross validation method, with K = 1 consistently giving the lowest accuracy. This indicates that the bias from having only a single split in the train-and-test split method is not significant compared to the k-fold cross validation method.

	Train-and-Test			Cross-Validation		
Accuracy	0.7 - 0.3	0.6 - 0.4	0.5 - 0.5	5-fold	10-fold	15-fold
K=1	19.40%	20.60%	21.30%	19.70%	20.00%	19.80%
K=5	23.50%	23.80%	23.70%	23.30%	23.00%	23.00%
K=10	24.70%	23.50%	23.90%	24.80%	24.50%	24.60%
K=15	25.70%	24.50%	24.50%	24.70%	24.60%	24.50%
K=20	23.10%	26.30%	25.00%	25.40%	25.50%	25.50%

Table 1. Classification performance comparison

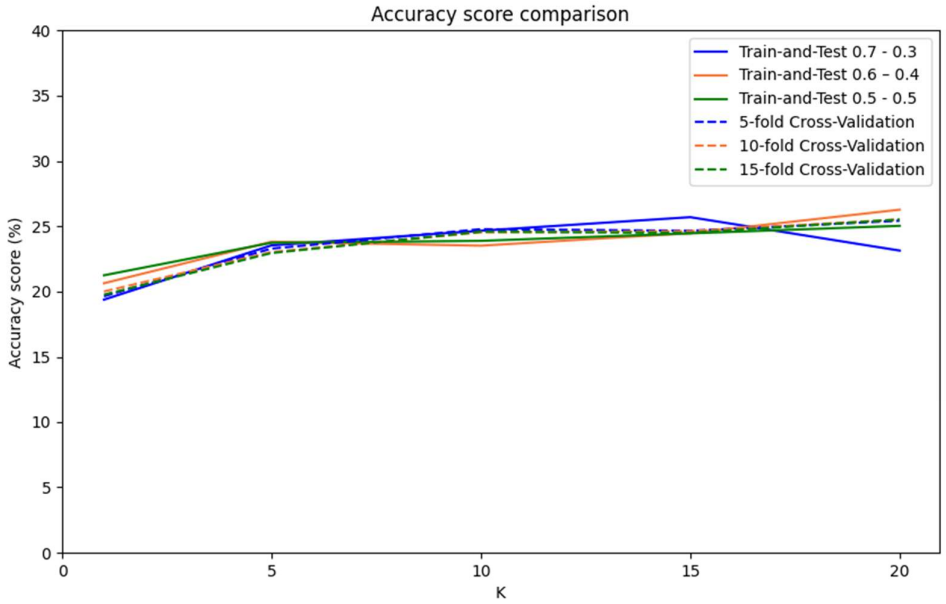

Figure 1. Classification performance comparison

precision	recall	f1-score	support	
1	0	0	0	1
2	0	0	0	1
3	0	0	0	15
4	0.38	0.44	0.41	57
5	0.29	0.29	0.29	115
6	0.29	0.31	0.3	259
7	0.3	0.35	0.32	391
8	0.3	0.32	0.31	568
9	0.24	0.35	0.28	689
10	0.2	0.27	0.23	634
11	0.24	0.23	0.23	487
12	0.16	0.08	0.11	267
13	0.14	0.05	0.07	203
14	0.19	0.02	0.04	126
15	0	0	0	103
16	0.07	0.01	0.02	67
17	0.08	0.02	0.03	58
18	0	0	0	42
19	0	0	0	32
20	0	0	0	26
21	0	0	0	14
22	0	0	0	6
23	0	0	0	9
24	0	0	0	2
25	0	0	0	1
26	0	0	0	1
27	0	0	0	2
29	0	0	0	1
accuracy			0.24	4177
macro avg	0.1	0.1	0.09	4177
weighted avg	0.22	0.24	0.23	4177

Table 2. Classification report for 5-fold cross validation with K = 15

Running Time:

The computational time for the classification for each of the experiments conducted is summarised in table 3. Based on the comparison findings, it is observed that the computational time of the KNN model using the k-fold cross validation method is about 3 times longer than the train-and-test split method (refer to figure 2). This is consistent with the methodology of the k-fold cross validation method, as since the model is trained k-1 times, the computational time for the classification is thus longer than that for the train-and-test method. Furthermore, the computational time for the k-fold cross validation increases with the increase in the number of folds. Therefore, the computational time for the 15-fold cross validation model is the longest amongst the different experiment setups.

	Train-and-Test			Cross-Validation		
Run-Time	0.7 - 0.3	0.6 - 0.4	0.5 - 0.5	5-fold	10-fold	15-fold
K=1	0.43	0.494	0.536	1.531	1.677	1.747
K=5	0.423	0.495	0.53	1.517	1.702	1.757
K=10	0.421	0.487	0.536	1.531	1.682	1.768
K=15	0.422	0.498	0.539	1.544	1.702	1.76
K=20	0.424	0.49	0.54	1.505	1.704	1.764

Table 3. Computational time comparison

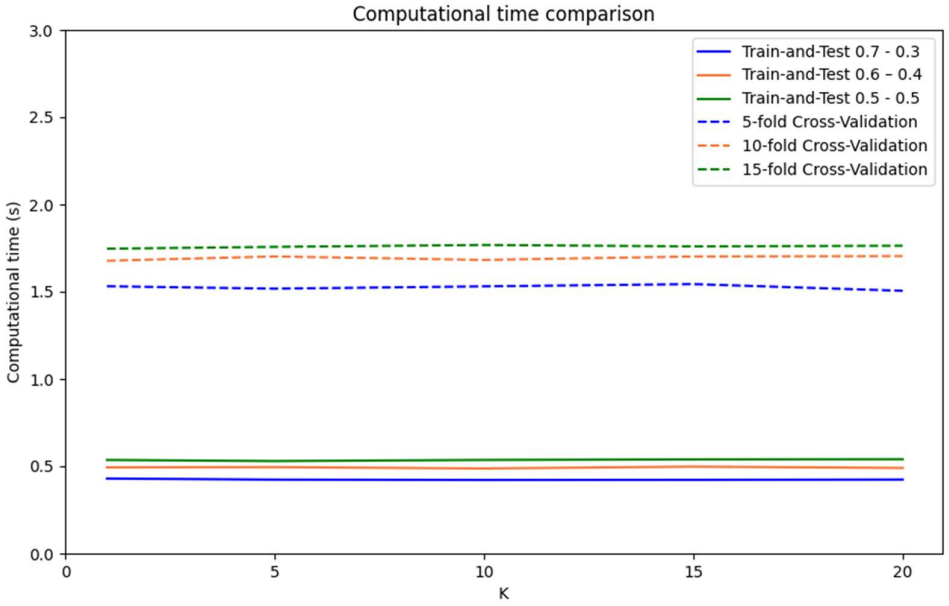

Figure 2. Computational time comparison