Assignment 3 – KNN

Seah Zu Xiang

Abstract

The objective of this assignment is to measure the accuracy of the KNN algorithm and how it fairs as you increase the number of K and the difference testing it using 'train and test' versus 'cross validation'.

In summary, the accuracy of the KNN algorithm generally increases as you increase the number of K. The comparison between 'cross validation' and 'train and test' seems to be around the same with 'train and test' being marginally more accurate. Changine 'train and test' percentage and number of folds also have marginal difference, where 'train and test' is more accurate with higher train percentage and cross validation being more accurate with higher fold.

1. Introduction

1.1 Introduce the Problem

Briefly describe the classification problem.

Finding the number of rings of an abalone is an incredibly cumbersome process as it involes cutting a sample of the abalone, staining it and counting the number of rings through a microscope.

1.2 Method - KNN algorithm

The KNN algorithin uses the features of an abalone to predict the number of rings an abalone has.

This is done by comparing each feature of the new abalone with the trained data's feature to find the Euclidean distance. This is done by treating each feature as an 'axis' and finding the distance by calculating the magnitude of the difference.

$$Distance = \sqrt{\sum_{i=1}^{n} (x1_i - x2_i)^2}$$

Where n = number of features and x1 = new abalone and x2 = one of the trained data.

Once you have all the distance, get the KNN (K nearest neighbours) by getting 1st K number of shortest distance. Compare the K outputs and take the highest recurring output. If none of them are recurring, take the 1st output. This will be the predicted value for the rings of the abalone.

2. Accuracy

Report your findings of comparison on classification performance.

The accuracy of the algorithm is determined by comparing the predicted output with the test output, if it matches it's counted as a test positive.

$$Accuracy = \frac{Test\ Positive}{All\ Test} * 100\%$$

Accuracy	Train-and test			Cross validation		
	0.7 - 0.3	0.6 – 0.4	0.5 - 0.5	5-fold	10-fold	15-fold
K=1	20.33%	19.34%	19.96%	20.23%	19.75%	20.08%
K=5	21.61%	22.68%	23.07%	23.27%	23.34%	23.07%
K=10	23.13%	22.86%	23.93%	23.41%	24.64%	24.49%
K=15	25.52%	24.90%	25.61%	24.97%	24.73%	24.73%
K=20	27.75%	25.43%	26.81%	24.99%	25.47%	25.42%

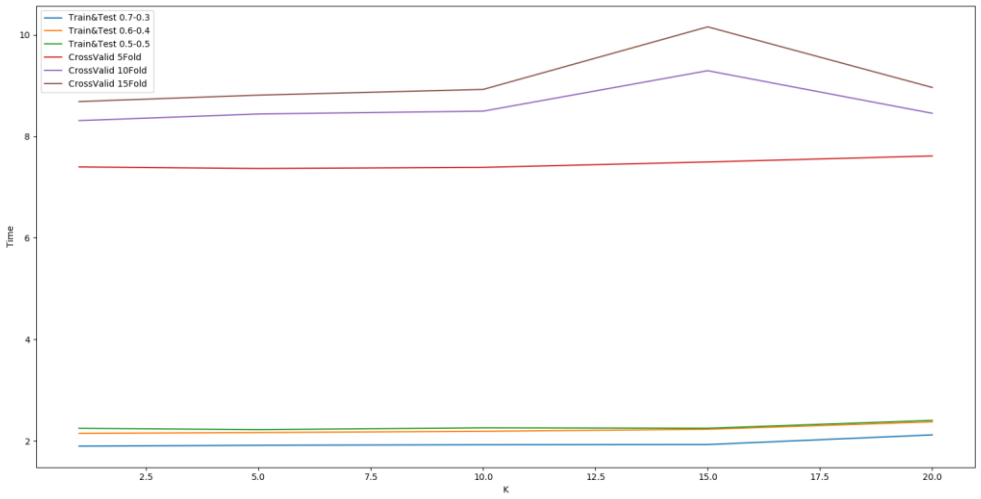
Attached below is the classification report for the 5-fold cross validation where k = 15.

Classfication report for 5-fold Cross Validation where k = 15				
	Precision	Recall	F1-score	support
1.0	0.00	0.00	0.00	1
2.0	0.00	0.00	0.00	1
3.0	0.00	0.00	0.00	15
4.0	0.38	0.46	0.42	57
5.0	0.29	0.28	0.28	115
6.0	0.31	0.34	0.32	259
7.0	0.27	0.31	0.29	391
8.0	0.31	0.32	0.31	568
9.0	0.25	0.37	0.30	689
10.0	0.21	0.29	0.25	634
11.0	0.24	0.23	0.24	487
12.0	0.14	0.07	0.10	267
13.0	0.12	0.04	0.06	203
14.0	0.05	0.01	0.01	126
15.0	0.00	0.00	0.00	103
16.0	0.20	0.04	0.07	67
17.0	0.11	0.02	0.03	58

18.0	0.00	0.00	0.00	42
19.0	0.00	0.00	0.00	32
20.0	0.00	0.00	0.00	26
21.0	0.00	0.00	0.00	14
22.0	0.00	0.00	0.00	6
23.0	0.00	0.00	0.00	9
24.0	0.00	0.00	0.00	2
25.0	0.00	0.00	0.00	1
26.0	0.00	0.00	0.00	1
27.0	0.00	0.00	0.00	2
29.0	0.00	0.00	0.00	1
Accurracy			0.25	4177
Macro avg	0.10	0.10	0.10	4177
Weighted avg	0.22	0.25	0.23	4177

3. Running time

Run-time	Train-and test			Cross validation		
	0.7 - 0.3	0.6 – 0.4	0.5 - 0.5	5-fold	10-fold	15-fold
K=1	1.90s	2.14s	2.24s	7.39s	8.31s	8.69s
K=5	1.91s	2.16s	2.22s	7.37s	8.44s	8.81s
K=10	1.92s	2.19s	2.27s	7.39s	8.50s	8.93s
K=15	1.93s	2.23s	2.25s	7.50s	9.29s	10.16s
K=20	2.12s	2.37s	2.40s	7.61s	8.46s	8.96s



The timings are a lot longer for ‘cross validation’ compared to ‘train-and test’ as cross validation does more comparison.

Also the duration taken to compute also increases as the number of fold increases for ‘cross validation’ as the number of computation is (NumberOfSamples / fold)* (NumberOfSamples / fold * (fold - 1)).

‘Train-and test’ also takes longer as train percentage decreases as the number of computation is (NumberOfSamples * TestPercentage)* (NumberOfSamples * TrainPercentage).

In general, duration increases as K increases as you need to check more values for reccurring values.