Dokumentation K-Nearest-Neighbour Algorithmus

Wir haben uns dazu entschlossen ein Framework zu bauen, welches für sämtliche Datensätze verwendet werden kann. Das Framework ist über verschiedene Methoden einfach konfigurierbar. Hier ein Beispiel der Verwendung bei der Klassifizierung der Wein-Daten:

```
/* WHITE WINE */
classifier.setDelimiter(";");
classifier.ignoreColumns(new int[]{1,2});
classifier.setOutputColumnCount(12);
classifier.setDataBeginRowCount(2);
classifier.readData( filename: "datafiles/winequality-white-data.csv");


/* CLASSIFY NORMAL */
classifier.doKFoldCross();

/* CLASSIFY TIMED */
classifier.measureClassifyingTime( number: 100000);
```

Und hier nun das Ergebnis, welches beim Durchführen auf der Konsole ausgegeben wird:

```
Reading data...Done!
Categorizing all datasets...Done!
Creating packs...Done!
Doing pass 1...Done!
Doing pass 2...Done!
Doing pass 3...Done!
Doing pass 4...Done!
Doing pass 5...Done!
Doing pass 6...Done!
Doing pass 7...Done!
Doing pass 8...Done!
Doing pass 9...Done!
Doing pass 10...Done!
Printing results...

5.0   4.0   6.0   7.0   8.0   3.0   9.0
-------------------------------------
726   12    629   83    7     0     0
88    5     61    9     0     0     0
575   7     1370  237   9     0     0
162   1     505   211   1     0     0
29    0     97    35    14    0     0
11    1     7     1     0     0     0
1     0     4     0     0     0     0
-------------------------------------
Vertical axis: Reference
Horizontal axis: Prediction
-------------------------------------
Accuracy: 47,49%
```

```
Reading data...Done!
Categorizing all datasets...Done!
Creating packs...Done!
Doing pass 1...Done!
Doing pass 2...Done!
Doing pass 3...Done!
Doing pass 4...Done!
Doing pass 5...Done!
Doing pass 6...Done!
Doing pass 7...Done!
Doing pass 8...Done!
Doing pass 9...Done!
Doing pass 10...Done!
Printing results...

Iris-setosa   Iris-versicolor   Iris-virginica
-------------------------------------
50    0     0
0     48    2
0     2     48
-------------------------------------
Vertical axis: Reference
Horizontal axis: Prediction
-------------------------------------
Accuracy: 97,33%
```

(Erklärungen zum Bild: Die Confusion Matrix hier ist nicht sortiert, die Werte über der ersten Linie stellen die unterschiedlichen Kategorien dar, die vertikale Achse hat dieselben Kategorien in derselben Reihenfolge. Die horizontale Achse stellt die Kategorie dar, die der Algorithmus berechnet hat, und die vertikale Achse stellt die Kategorie dar, die tatsächlich im Datensatz eingetragen ist. Die Werte innerhalb der Matrix sind teilweise etwas verschoben und nicht immer direkt unter der horizontalen Achse.)

Man sieht, dass die Accuracy für die Iris-Blüten bei etwa 97% liegt, also dass fast alle Blüten korrekt klassifiziert werden konnten. Bei den Wein-Daten hingegen ist die Accuracy sehr niedrig, nur auf etwa 47%. Das zeigt uns, dass diese Daten nur sehr schwer auszuwerten sind, da sie scheinbar keine deutlichen Zusammenhänge erkennen lassen. Deshalb haben wir uns anschließend mit der Möglichkeit unseres Frameworks herumgespielt, ein paar Spalten der Datensätze zu ignorieren. Letztendlich haben wir uns dazu entschieden nur die Spalten für „alcohol“, „density“, „volatile acidity“ und „total sulfur dioxide“ tatsächlich für die Klassifizierung zu verwenden. Folgendes Ergebnis kam dabei heraus:

```
6.0   7.0   8.0   3.0   5.0   4.0   9.0
-------------------------------------
1527   232   8   0   430   1   0
545   260   2   0   73   0   0
113   38   15   0   9   0   0
12   1   0   0   7   0   0
711   65   6   0   670   5   0
93   12   0   0   58   0   0
4   1   0   0   0   0   0
-------------------------------------
Vertical axis: Reference
Horizontal axis: Prediction
-------------------------------------
Accuracy: 50,47%
```

Hier ist also eine etwas genauere Klassifizierung möglich, wir kommen hier auf etwa 50%.

Anschließend haben wir noch mithilfe des Algorithmus die Zeit für 1 000, 10 000 und 100 000 Klassifizierungen gemessen und sind auf folgendes Ergebnis für die Iris-Blüten gekommen:

```
Classified 1000 datasets in 0.030272486s!
Classified 10000 datasets in 0.112270726s!
Classified 100000 datasets in 0.752573944s!
```

Diese Daten sind also relativ schnell auszuwerten, da es nur recht wenige Datensätze zum Lernen gibt und nur insgesamt drei unterschiedliche Möglichkeiten der Klassifizierung vorhanden sind.

Die Werte für die Wein-Datensätze für 1 000, 10 000 und 100 000 Klassifizierungen sind folgende:

```
Classified 1000 datasets in 1.929400301s!
Classified 10000 datasets in 10.380347772s!
Classified 100000 datasets in 49.298243778s!
```

Hier ist also zu sehen, dass es deutlich länger benötigt als bei den Iris-Datensätzen. Das liegt daran, dass mehr Datensätze zum Lernen vorhanden sind und es insgesamt auch mehrere Möglichkeiten der Klassifizierung gibt.

Zusätzlich haben wir entdeckt, dass der Algorithmus mit Java 8 ungefähr um das 10-fache schneller arbeitet als unter Java 9. Daher sind alle obigen Auswertungen mit Java 8 durchgeführt worden. Den Grund für die Verlangsamung unter Java 9 konnten wir leider nicht herausfinden.