# 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 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!
Removing outliers...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...

6.0   5.0   7.0   8.0   4.0   3.0   9.0
-------------------------------------
1066   598   435   55   38   6   0
679   537   167   14   56   4   0
406   128   273   56   15   1   1
55   35   64   15   6   0   0
59   72   14   0   17   1   0
7   10   0   0   3   0   0
3   0   2   0   0   0   0
-------------------------------------
Vertical axis: Reference
Horizontal axis: Prediction
-------------------------------------
Accuracy: 38,95%
```

```
Reading data...Done!
Removing outliers...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   50   0
0   0   50
-------------------------------------
Vertical axis: Reference
Horizontal axis: Prediction
-------------------------------------
Accuracy: 100%

Process finished with exit code 0
```

(Erklärung 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 100% liegt, also dass alle Blüten korrekt klassifiziert werden konnten. Bei den Wein-Daten hingegen ist die Accuracy sehr niedrig, nur auf knapp 39%. Das zeigt uns, dass diese Daten nur sehr schwer auszuwerten sind, da sie scheinbar keine deutlichen Zusammenhänge erkennen lassen. Das liegt aber nur daran, dass bei den Datensätzen sehr viele Spalten zum Einsatz kommen, die nicht alle wirklich notwendig für die Klassifizierung sind. 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“ und „density“ tatsächlich für die Klassifizierung zu verwenden. Folgendes Ergebnis kam dabei heraus:

```
Reading data...Done!
Removing outliers...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...

6.0   5.0   7.0   8.0   4.0   3.0   9.0
-------------------------------------
2198   0   0   0   0   0   0
0   1457   0   0   0   0   0
0   0   880   0   0   0   0
0   0   0   175   0   0   0
0   2   0   0   161   0   0
0   0   0   0   5   15   0
0   0   0   4   0   0   1
-------------------------------------
Vertical axis: Reference
Horizontal axis: Prediction
-------------------------------------
Accuracy: 99,78%
```

Hier ist also doch eine sehr genaue Klassifizierung möglich, solange die richtigen Spalten dafür benutzt werden.

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 0.383773387s!
Classified 10000 datasets in 2.936137456s!
Classified 100000 datasets in 26.687230296s!
```

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 4.214601567s!
Classified 10000 datasets in 32.986494414s!
Classified 100000 datasets in 5m 18.096032038s!
```

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. Deshalb ist bei 100 000 Klassifizierungen die Wartezeit auch schon über fünf Minuten.

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.