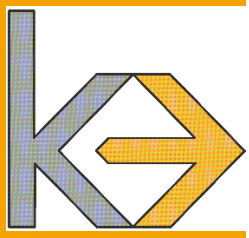
# Implementierung kNN

**Projekt der Vorlesung Maschinelles Lernen: Symbolische Ansätze**

TECHNISCHE
UNIVERSITÄT
DARMSTADT



## 1 Über dieses Projekt

Im Rahmen dieses Projektes werden Sie einen k-Nearest Neighbor Klassifizierer implementieren, wie er auch in der Vorlesung und Übung vorgestellt wird. Hierzu wird ein Framework bereitgestellt, welches Kompatibilität mit WEKA gewährleistet. Bei korrekter Implementation sind die Resultate des Klassifizieres identisch mit der WEKA Version IBk. Neben dem Framework werden auch JUnit TestCases zur Verfügung gestellt mit welchen die korrekte Funktionsweise überprüft werden kann.
Die Aufgaben aus Kapitel 4.1 und 4.2 sind fast ausschließlich mit Informationen aus dem Vorlesungsscript zu lösen, wohin gegen Kapitel 4.3 etwas Transferleistung und eigene Überlegungen erfordert.

## 2 Installation

Das Projekt ist in Java geschrieben und es wird empfohlen Eclipse[1] als IDE zu verwenden. Das Projektarchiv enthält entsprechende Eclipse Projektsignaturen. Zudem wird JUnit 4 benötigt, welches aber bereits Bestandteil der Eclipse Java Development Tools[2] ist. Die Anbindung an WEKA [3] kann entweder über das Hinzufügen der weka.jar Datei zum classpath erfolgen oder indem man den Quellcode als Projekt einbindet. Das Projekt wurde für WEKA 3.7.11 geschrieben, ist aber auch mit WEKA 3.7.10 kompatibel. Die entsprechende Datei finden Sie unter Developer Version auf der WEKA Download Seite.

## 3 Das Framework

Zu Implementieren ist die Klasse tud.ke.ml.project.classifier.NearestNeighbor. Diese erbt von tud.ke.ml.project.framework.classifier.ANearestNeighbor, welche ein lightweight Framework definiert, so wie das Handling der Konfiguration. Die Klasse weka.classifiers.lazy.keNN definiert die Schnittstelle zu WEKA und darf genauso wie die abstrakte Basisklasse nicht modifiziert werden.

Instanzen sind im Framework als Liste von Objekten gespeichert. Ein jedes Objekt ist entweder ein String, und repräsentiert in diesem Falle ein nominales Attribut, oder ein Double, welches für ein numerisches Attribut steht. Das Klassenattribut ist immer nominal.

Gestartet wird das Projekt über tud.ke.ml.project.main.RunWEKA. Der neue Classifier ist nun in allen Teilen des Programmes unter lazy.keNN zu finden. Die Klasse main.SimpleRun ist ein einfaches Beispiel, welches während der Entwicklung genutzt werden kann, falls Sie nicht immer die WEKA GUI verwenden wollen. Im Package tud.ke.ml.project.junit befinden sich zudem die erwähnten JUnit Tests. junit.SimpleValidation und junit.AdvancedValidation testen verschiedene Konfigurationen des Klassifizieres. Sollten diese Tests alle erfolgreich durchlaufen, ist dieser sehr wahrscheinlich korrekt implementiert.

Zu allen relevanten Methoden ist eine Javadoc-Dokumentation vorhanden.

### 3.1 JUnit

JUnit tests können ausgeführt werden, in dem die entsprechende Library dem Projekt hinzugefügt wird. In Eclipse ist diese bereits vorhanden und muss daher nicht mehr extra hinzu gefügt werden. Zum testen muss eine JUnit-java Datei ausgewählt werden und per Kontextmenü Run As→JUnit Test ausgeführt werden. Rote Kreuze bedeuten dass der Test eine Exception ausgelöst hat. Blaue Kreuze repärsentieren einen fehlgeschlagenen JUnit Test. Für eine genauere Beschreibung von JUnit, siehe `http://www.vogella.com/tutorials/JUnit/article.html`.

## 4 Die Aufgabe

Im Folgenden ist zu beachten, dass zum Testen alle Methoden implementiert sein müssen. Es ist aber natürlich möglich hierfür erst einmal nur Dummy Methoden zu verwenden, die einfach einen fixen Wert zurückgeben.
Definieren sie zuerst getMatrikelNumbers, da diese Funktion für die finale Bewertung verwendet werden!

[1] `https://www.eclipse.org/`
[2] `http://www.eclipse.org/jdt/`
[3] `http://www.cs.waikato.ac.nz/ml/weka/`

## 4.1 Der Basis Klassifizierer

1. Implementieren Sie die Methode learnModel, welche die als Argument übergebenen Trainingsdaten intern speichert. Des Weiteren soll die Methode vote implementiert werden, in dem sie getUnweightedVotes aufrufen. Die daraus resultierende Map muss an getWinner übergeben werden um die aus den Votes resultierende Klasse zu bestimmen.
2. getNearest muss für alle Instanzen des gespeicherten Modells die Manhattan Distanz zur übergebenen Test-Instanz berechnen. Dies soll mit Hilfe der zu Implementierenden Methode determineManhattanDistance erfolgen. Die Liste, die zurückgegeben wird, muss auf die nächsten getkNearest Elemente beschränkt sein. Die Liste beeinhaltet Pair Objekte, bestehend aus der Instanz sowie dem Abstand.
3. Wenn Sie nun noch die Methode getWinner implementieren, sollten Sie eine erste, lauffähige Implementation haben. Die genannte Methode soll den Index der Klasse mit den meisten Votes zurückgeben.

## 4.2 Inverse Distance Weighting und Euclidean Distance

1. Implementieren Sie nun getWeightedVotes, welches die Stimmen als Summe der inversen Distanzen berechnet. vote muss diese Methode in Abhängigkeit von isInverseWeighting aufrufen.
2. Genauso muss die Methode getNearest in Abhängigkeit von getMetric die euklidsche Distanz als Entfernungsmaß nutzen. Hierzu soll die Methode determineEuclideanDistance genutzt und implementiert werden.

## 4.3 Normalization and Tie-Breaking

1. Es ist meistens sinvoll die Attributwerte zu normalisieren. Dafür soll die Methode normalizationScaling die nötigen Skalierungs- und Translationsfaktoren zurückliefern, abhängig von isNormalizing. In der Methode getNearest muss nun das resultierende Array aufgeteilt und als protected double[] scaling, bzw. protected double[] translation gespeichert werden, wobei für jedes Attribut (auch das potentielle Klassenattribut) ein Eintrag vorhanden sein muss. Diese Arrays müssen nun genutzt werden, um die Attributwerte entsprechend zu normalisieren, d.h. das Resultat von normalizationScaling darf nicht direkt übernommen werden!
2. In der Methode getWinner kann es vorkommen, dass mehr als eine Klasse die meisten Stimmen bekommen hat. Implementieren Sie hierfür eine Entscheidungsfunktion und begründen Sie diese.
3. In der Methode getNearest kann es passieren, dass mehrere Instanzen den gleichen Abstand haben. Wie sollte man hierbei vorgehen ? Begründen und implementieren Sie ihre Methode.