


Nicht-lineare Datenstrukturen

Der ID3-Algorithmus II

Kopiere das Projekt  07-ID3-Algorithmus aus dem Tauschordner und öffne es in BlueJ.

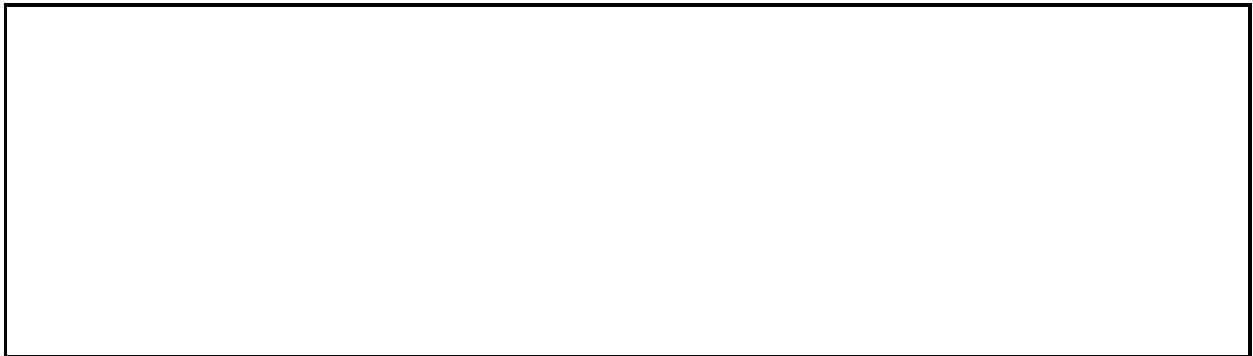
Aufgabe 1

Erstelle eine Instanz des `DescisionTreeBuilders` und führe die Methode `calculateEntropy()` aus. Vergleiche die angezeigten Daten mit der berechneten *Entropie* und dem *Informationsgewinn* der Attribute.




Aufgabe 2

Führe die Methode `buildTree()` auf dem Objekt aus. Der Entscheidungsbaum wird nun rekursiv aus den Trainingsdaten aufgebaut.


Skizziere den entstandenen Baum, indem du mit dem Inspektor von BlueJ die Objekte erkundest.



Aufgabe 3

Die Trainingsdaten werden aus der Datei  `titanic50.csv` im Projektordner geladen. Der Dateiname wird im `DescisionTreeBuilder` in der Klassenvariablen `TRAININGDATA` festgelegt. Statt 50 Passagierdaten sind auch Datensätze mit 300 ( `titanic300.csv`) und 800 ( `titanic800.csv`) vorhanden.

Erstelle den Baum mit einem der anderen Trainingsdatensätze und vergleiche das Ergebnis mit dem ersten Baum.

 **Hinweis:** Zum Vergleich kannst du die Methode `classifyTestdata()` verwenden, um einen Satz unbekannter Passagierdaten zu laden und durch den Entscheidungsbaum klassifizieren zu lassen.

Aufgabe 4

Mit größeren Datensätzen werden die Bäume feiner ausdifferenziert und lassen sich durch den Inspektor nur umständlich betrachten.

Implementiere die Methode `private void printTree(BinaryTree<DecisionNode> pTree, int pDepth)` wie folgt:

Die Methode läuft rekursiv durch den Baum. Wenn der aktuelle Baum nicht leer ist, wird der Inhalt der Wurzel ausgegeben. Dann wird die Methode rekursiv erst auf dem linken, dann auf dem rechten Teilbaum aufgerufen. Die Rekursion bricht ab, wenn ein Blattknoten erreicht ist (der Teilbaum leer ist).

Gestartet wird die Ausgabe mit der Methode `printTree()`.

Aufgabe 5

Die maximale Anzahl an Entscheidungen im Baum kann beim Instanzieren des `DescisionTreeBuilders` über den Parameter `pMaxDepth` festgelegt werden. Teste verschiedene Werte für den Parameter und lass dir den entstehenden Baum mit `printTree()` anzeigen.

Aufgabe 6

Das rekursive Durchlaufen des Baumes, wie in der Methode `printTree()`, nennt man *Traversierung*.

Lies im Buch den Abschnitt 5.2 (ab Seite 145) über Binärbäume - insbesondere über die verschiedenen Arten der *Traversierung*. Bearbeite dann auf Seite 150 Aufgabe 3 a) und b). (b) in Pseudocode).