Übung 4

Alexander Hinneburg

Aufgabe 1

```
# Umbenennen der Dateien:
cp data_set_ALL_AML_train.txt ALL_AML_grow.train.orig.txt
cp data_set_ALL_AML_independent.txt ALL_AML_grow.test.orig.txt

#Zaehlen der Zeilen
wc ALL_AML_*orig.txt
#   7130  533422 1860350 ALL_AML_grow.test.orig.txt
#   7130  590458 2046808 ALL_AML_grow.train.orig.txt
#  14260 1123880 3907158 total

# und der Spalten
head -1 ALL_AML_grow.train.orig.txt | gawk -F"\t" '{print NF}'
# 78
head -1 ALL_AML_grow.test.orig.txt | gawk -F"\t" '{print NF}'
# 70
# das Trennzeichen wird mit -F"\t" auf Tabs gesetzt
```

Aufgabe 1

```
# 1. Loeschen der Kontrollen
# Beispiel: AFFX-BioB-5_at (endogenous control)

# Gibt es noch mehr Eintraege mit "control" ?
grep -n "control" ALL_AML_grow.train.orig.txt

# Ja, drei Zeilen, die nicht am Beginn der Datei sind. Wie sehen diese aus?
# 1.) 3556:Mitotic feedback control protein Madp2 homolog mRNA
# 2.) 6080:Cell division control related protein (hCDCrel-1) mRNA
# 3.) 6340:GB DEF = HRAR- beta 2=retinoic-acid-receptor beta/suspected tumor
#     suppressor {5' region, transcription control region} [human, mRNA Partial, 1730 nt]

# Diese Zeilen sollen drin bleiben, also Muster: "control)"
grep -v "control)" ALL_AML_grow.train.orig.txt > ALL_AML_grow.train.noaffy.tmp
grep -v "control)" ALL_AML_grow.test.orig.txt > ALL_AML_grow.test.noaffy.tmp

# Wieviele Kontrolleintraege?
grep "control)" ALL_AML_grow.train.orig.txt |wc
# 59
grep "control)" ALL_AML_grow.test.orig.txt |wc
# 59
# Macht Sinn, in beiden Dateien dieselbe Anzahl zu finden.
```

Aufgabe 1

```
# Loeschen der Felder Gene Description und call
head -1 ALL_AML_grow.train.orig.txt
# Gene Description        Gene Accession Number   1       call    2 call    3       call    4       call    5     call
      6       call 7       call    8       call    9       call    10      call    11  call    12      call  13      call    14
      call    15      call  16      call    17      call    18      call    19      call    20  call    21      call    22      call
      23      call    24      call
# 25      call    26      call    27      call  34      call    35
# call    36      call    37      call    38      call    28      call
# 29      call    30    call     31      call    32      call    33
# call

# nur die Felder in der Liste werden beibehalten, TAB ist default Trenner
cut -f 2,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,\
41,43,45,47,49,51,53,55,57,59,61,63,65,67,69,71,73,75,77 \
ALL_AML_grow.train.noaffy.tmp > ALL_AML_grow.train.nocall.tmp

cut -f 2,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,\
41,43,45,47,49,51,53,55,57,59,61,63,65,67,69 \
ALL_AML_grow.test.noaffy.tmp > ALL_AML_grow.test.nocall.tmp
```

Aufgabe 1

```
# Test, ob es geklappt hat:
head -1 ALL_AML_grow.train.nocall.tmp
# Gene Accession Number   1       2       3       4       5       6
                        7       8       9       10      11      12    13      14
    15   16      17      18      19      20      21      22      23      24
    25      26      27    34      35      36      37      38      28
              29      30      31      32      33
```

Aufgabe 1

```
# Umwandeln der Tabs in Kommas
cat ALL_AML_grow.train.nocall.tmp | tr "\t" "," >
    ALL_AML_grow.train.nocall.csv
cat ALL_AML_grow.test.nocall.tmp  | tr "\t" "," >
    ALL_AML_grow.test.nocall.csv

# Umbenennen von Gene Accession Number in ID, s ist das substitute
    Kommando von sed
sed "s/Gene.Accession.Number/ID/" ALL_AML_grow.train.nocall.csv >
    ALL_AML_grow.train.ID.csv
sed "s/Gene.Accession.Number/ID/" ALL_AML_grow.test.nocall.csv >
    ALL_AML_grow.test.ID.csv

# Test, ob es geklappt hat:
head -1 ALL_AML_grow.train.ID.csv
# ID,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,
# 25,26,27,34,35,36,37,38,28,29,30,31,32,33
```

Aufgabe 1

- # Spalten und Zeilen der Dateien:
- wc ALL_AML_*.ID.csv
- # 7071 7071 1052762 ALL_AML_grow.test.ID.csv
- # 7071 7071 1177973 ALL_AML_grow.train.ID.csv
- # 14142 14142 2230735 total

- head -1 ALL_AML_grow.train.ID.csv | gawk -F"," '{print NF}'
- # 39
- head -1 ALL_AML_grow.test.ID.csv | gawk -F"," '{print NF}'
- # 35

Aufgabe 1

```
# Normalisieren
# Kopfzeile retten
head -1 ALL_AML_grow.train.ID.csv > ALL_AML_grow.train.norm.csv
head -1 ALL_AML_grow.test.ID.csv > ALL_AML_grow.test.norm.csv

# normalisierte Daten anfuegen
sed "1,1d" ALL_AML_grow.train.ID.csv | \
gawk -v FS=, -v OFS=, '{for (i=2;i<=NF;i++){if($(i)<20) $(i)=20;if($(i)>16000) $(i)=16000}
      print $0 }' \
>> ALL_AML_grow.train.norm.csv

sed "1,1d" ALL_AML_grow.test.ID.csv | \
gawk -v FS=, -v OFS=, '{for (i=2;i<=NF;i++){if($(i)<20) $(i)=20;if($(i)>16000) $(i)=16000}
      print $0 }' \
>> ALL_AML_grow.test.norm.csv

# Transponieren
cat ALL_AML_grow.train.norm.csv | gawk -v FS=, -v OFS=, -f transpose.awk >
      ALL_AML_gcol.train.csv
cat ALL_AML_grow.test.norm.csv | gawk -v FS=, -v OFS=, -f transpose.awk >
      ALL_AML_gcol.test.csv
```

Aufgabe 1

```
# Inhalt von Transpose.awk
# NR == 1 {
#    n = NF
#    for (i = 1; i <= NF; i++)
#        row[i] = $i
#    next
# }
# {
#    if (NF > n)
#        n = NF
#    for (i = 1; i <= NF; i++)
#        row[i] = row[i] "," $i
# }
# END {
#    for (i = 1; i <= n; i++)
#        print row[i]
# }
```

Aufgabe 1

- # Spalten und Zeilen der Dateien:
- wc ALL_AML_gcol.*.csv
- # 35 35 955774 ALL_AML_gcol.test.csv
- # 39 39 1067326 ALL_AML_gcol.train.csv
- # 74 74 2023100 total

- head -1 ALL_AML_gcol.train.csv | gawk -F"," '{print NF}'
- # 7071
- head -1 ALL_AML_gcol.test.csv | gawk -F"," '{print NF}'
- # 7071

Aufgabe 1

```
# ID und Klasse extrahieren
echo "ID,Class" >ALL_AML_idclass.train.txt
echo "ID,Class" >ALL_AML_idclass.test.txt

cat table_ALL_AML_samples.txt | \
gawk -v "FS=[\t ]+" '{if ($1>=1 && $1<39) print $1 "," $2}'
     >>ALL_AML_idclass.train.txt

cat table_ALL_AML_samples.txt | \
gawk -v "FS=[\t ]+" '{if ($1>=39 && $1<=72) print $1 "," $2}'
     >>ALL_AML_idclass.test.txt

wc ALL_AML_idclass.*.txt
# 35  35 247 ALL_AML_idclass.test.txt
# 39  39 266 ALL_AML_idclass.train.txt
# 74  74 513 total
```

Aufgabe 1

```
# Zusammenfuegen
sort -n ALL_AML_gcol.train.csv > ALL_AML_gcol.train.sort.csv
sort -n ALL_AML_gcol.test.csv > ALL_AML_gcol.test.sort.csv

join -t , ALL_AML_gcol.train.sort.csv ALL_AML_idclass.train.txt >
    ALL_AML_gcol.train.join.csv
join -t , ALL_AML_gcol.test.sort.csv ALL_AML_idclass.test.txt >
    ALL_AML_gcol.test.join.csv
```

Aufgabe 2.1

- OneRule
  - === Classifier model (full training set) ===
    ID:
    < 27.5 -> ALL
    >= 27.5 -> AML
    (38/38 instances correct)
  - Genauigkeit 100%
  - Klassifikator macht aber keinen Sinn, da das ID Attribut gewählt wurde, und dies sich sicher nicht verallgemeinern läßt.

Aufgabe 2.2

- OneR
  - === Classifier model (full training set) ===
    X95735_at:
    < 994.0 -> ALL
    >= 994.0 -> AML
    (38/38 instances correct)
- J48
  - pruned tree:
    X95735_at <= 938: ALL (27.0)
    X95735_at > 938: AML (11.0)
- NaiveBayesSimple:
  - Fehler: Attribute mit Std=0
- OneR und J48 liefern fast gleiche Ergebnisse
- Vor NaïveBayes sollte eine Attributauswahl vorgenommen werden

Aufgabe 2.3

- Datenvorbereitung ist sehr aufwendig
  - Daten wurden nicht für die Analyse designed
  - Datenintegration oft 70% der Arbeit eine DM Projektes
- Ausnahmen und Fehler treten überall auf
  - Spezifikationen sind unvollständig
  - Daten werden oft anders bearbeitet =>
    keine Erfahrung mit DM
- Kenntnis von vielen Werkzeugen nützlich
  - Unix-Tools für große Datenmengen zum Testen
  - Datenbanken oder Dataware-Houses für den regelmäßigen Betrieb
- Dokumentation der Operationen ist wichtig