

# Data Mining Assignment #4

Jeremy Keys

10/26/2017

## Part A

(0) I extracted the data and renamed the files.

## Part B

Before pruning/cleaning the data:

ALL\_AML\_grow.train.orig.txt contained: 7130 records, and each record contains: 79 fields, for a total of: 563,270 fields.

ALL\_AML\_grow.test.orig.txt contained: 7130 records, and each record contains: 79 fields, for a total of: 563,270 fields.

1 I completed this section with a combination of head, wc, grep, and awk (and redirection and piping).

After doing the data processing required for (1):

ALL\_AML\_grow.train.noaffy.tmp contained: 7071 records, and each record contains: 70 fields, for a total of: 494,970 fields.

ALL\_AML\_grow.test.noaffy.tmp contained: 7071 records, and each record contains: 79 fields, for a total of: 558,609 fields.

There are 59 control records in the initial train file and 59 control records in the initial test.

```
jereny@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ wc ALL_AML_grow.train.noaffy.tnp
7071 529174 1843979 ALL_AML_grow.train.noaffy.tnp
jereny@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ grep -v control\| ALL_AML_grow.test.orig.txt > ALL_AML_grow.test.noaffy.tnp
jereny@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ wc ALL_AML_grow.test.noaffy.tnp
7071 585738 2028793 ALL_AML_grow.test.noaffy.tnp
jereny@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ head ALL_AML_grow.train.noaffy.tnp | awk -F '\t' '{print NF}'
70
70
70
70
70
70
70
70
70
70
70
70
jereny@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ head ALL_AML_grow.test.noaffy.tnp | awk -F '\t' '{print NF}'
78
79
79
79
79
79
79
79
79
79
79
79
fields.
```

```

34 jerry@jerry-VirtualBox:~/media/sf_NNSU/Data Mining/hw4$ cut -f 2- ALL_AML_grow.test.noaffy.tmp > ALL_AML_grow.test.noaffy.tmp.cut
35 jerry@jerry-VirtualBox:~/media/sf_NNSU/Data Mining/hw4$ cut -f 2- ALL_AML_grow.train.noaffy.tmp > ALL_AML_grow.train.noaffy.tmp.cut
36 jerry@jerry-VirtualBox:~/media/sf_NNSU/Data Mining/hw4$ cut -f 1,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,4
37 6,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78 ALL_AML_grow.test.noaffy.tmp.cut > ALL_AML_grow.test.noaffy.tmp.cutright
38 jerry@jerry-VirtualBox:~/media/sf_NNSU/Data Mining/hw4$ cut -f 1,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,4
39 6,48,50,52,54,56,58,60,62,64,66,68,70,72,74,76,78 ALL_AML_grow.train.noaffy.tmp.cut > ALL_AML_grow.train.noaffy.tmp.cutright
40 jerry@jerry-VirtualBox:~/media/sf_NNSU/Data Mining/hw4$ wc -c cutright
41 7071 275771 1185962 ALL_AML_grow.test.noaffy.tmp.cutright
42 7071 247487 1052781 ALL_AML_grow.train.noaffy.tmp.cutright
43 14142 523256 2237843 total
44 jerry@jerry-VirtualBox:~/media/sf_NNSU/Data Mining/hw4$ head ALL_AML_grow.train.noaffy.tmp.cutright | awk -F '\t' '{print NF}'
45 35
46 35
47 35
48 35
49 35
50 35
51 35
52 35
53 35
54 35
55 35
56 35
57 35
58 35
59 35
60 35
61 35
62 35
63 35
64 35
65 35
66 35
67 35
68 35
69 35
70 35
71 35
72 35
73 35
74 35
75 35
76 35
77 35
78 35
79 35
80 35
81 35
82 35
83 35
84 35
85 35
86 35
87 35
88 35
89 35
90 35
91 35
92 35
93 35
94 35
95 35
96 35
97 35
98 35
99 35
100 35
101 35
102 35
103 35
104 35
105 35
106 35
107 35
108 35
109 35
110 35
111 35
112 35
113 35
114 35
115 35
116 35
117 35
118 35
119 35
120 35
121 35
122 35
123 35
124 35
125 35
126 35
127 35
128 35
129 35
130 35
131 35
132 35
133 35
134 35
135 35
136 35
137 35
138 35
139 35
140 35
141 35
142 35
143 35
144 35
145 35
146 35
147 35
148 35
149 35
150 35
151 35
152 35
153 35
154 35
155 35
156 35
157 35
158 35
159 35
160 35
161 35
162 35
163 35
164 35
165 35
166 35
167 35
168 35
169 35
170 35
171 35
172 35
173 35
174 35
175 35
176 35
177 35
178 35
179 35
180 35
181 35
182 35
183 35
184 35
185 35
186 35
187 35
188 35
189 35
190 35
191 35
192 35
193 35
194 35
195 35
196 35
197 35
198 35
199 35
200 35
201 35
202 35
203 35
204 35
205 35
206 35
207 35
208 35
209 35
210 35
211 35
212 35
213 35
214 35
215 35
216 35
217 35
218 35
219 35
220 35
221 35
222 35
223 35
224 35
225 35
226 35
227 35
228 35
229 35
230 35
231 35
232 35
233 35
234 35
235 35
236 35
237 35
238 35
239 35
240 35
241 35
242 35
243 35
244 35
245 35
246 35
247 35
248 35
249 35
250 35
251 35
252 35
253 35
254 35
255 35
256 35
257 35
258 35
259 35
260 35
261 35
262 35
263 35
264 35
265 35
266 35
267 35
268 35
269 35
270 35
271 35
272 35
273 35
274 35
275 35
276 35
277 35
278 35
279 35
280 35
281 35
282 35
283 35
284 35
285 35
286 35
287 35
288 35
289 35
290 35
291 35
292 35
293 35
294 35
295 35
296 35
297 35
298 35
299 35
300 35
301 35
302 35
303 35
304 35
305 35
306 35
307 35
308 35
309 35
310 35
311 35
312 35
313 35
314 35
315 35
316 35
317 35
318 35
319 35
320 35
321 35
322 35
323 35
324 35
325 35
326 35
327 35
328 35
329 35
330 35
331 35
332 35
333 35
334 35
335 35
336 35
337 35
338 35
339 35
340 35
341 35
342 35
343 35
344 35
345 35
346 35
347 35
348 35
349 35
350 35
351 35
352 35
353 35
354 35
355 35
356 35
357 35
358 35
359 35
360 35
361 35
362 35
363 35
364 35
365 35
366 35
367 35
368 35
369 35
370 35
371 35
372 35
373 35
374 35
375 35
376 35
377 35
378 35
379 35
380 35
381 35
382 35
383 35
384 35
385 35
386 35
387 35
388 35
389 35
390 35
391 35
392 35
393 35
394 35
395 35
396 35
397 35
398 35
399 35
400 35
401 35
402 35
403 35
404 35
405 35
406 35
407 35
408 35
409 35
410 35
411 35
412 35
413 35
414 35
415 35
416 35
417 35
418 35
419 35
420 35
421 35
422 35
423 35
424 35
425 35
426 35
427 35
428 35
429 35
430 35
431 35
432 35
433 35
434 35
435 35
436 35
437 35
438 35
439 35
440 35
441 35
442 35
443 35
444 35
445 35
446 35
447 35
448 35
449 35
450 35
451 35
452 35
453 35
454 35
455 35
456 35
457 35
458 35
459 35
460 35
461 35
462 35
463 35
464 35
465 35
466 35
467 35
468 35
469 35
470 35
471 35
472 35
473 35
474 35
475 35
476 35
477 35
478 35
479 35
480 35
481 35
482 35
483 35
484 35
485 35
486 35
487 35
488 35
489 35
490 35
491 35
492 35
493 35
494 35
495 35
496 35
497 35
498 35
499 35
500 35
501 35
502 35
503 35
504 35
505 35
506 35
507 35
508 35
509 35
510 35
511 35
512 35
513 35
514 35
515 35
516 35
517 35
518 35
519 35
520 35
521 35
522 35
523 35
524 35
525 35
526 35
527 35
528 35
529 35
530 35
531 35
532 35
533 35
534 35
535 35
536 35
537 35
538 35
539 35
540 35
541 35
542 35
543 35
544 35
545 35
546 35
547 
```

2 I completed this section with cut.

ALL\_AML\_grow.train.noaffy.tmp.cutright contained: 7071. records,  
and each record contains: 35 fields, for a total of: 247,485 fields.

ALL\_AML\_grow.test.noaffy.tmp.cutright contained: 7071 records, and each record contains: 40 fields (except record 1, which contains 39 fields), for a total of: 282,840 - 1\*(record 1 # of fields) fields.

```
40 jupyter@jupyter-VirtualBox: /media/sf_NMSU/Data Mining/hw4$ tr '\t' ' ' < ALL_AML_grow.train.noaffy.tmp.cutright
t > ALL_AML_grow.train.csv
jupyter@jupyter-VirtualBox: /media/sf_NMSU/Data Mining/hw4$ tr '\t' ' ' < ALL_AML_grow.test.noaffy.tmp.cutright
> ALL_AML_grow.test.csv
jupyter@jupyter-VirtualBox: /media/sf_NMSU/Data Mining/hw4$
```

Figure 3:

```

jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ head ALL_AML_grow.train.csv | awk -F ',' '{print NF}'
39
}
35
35
35
35
35
35
35
35
35
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ head ALL_AML_grow.test.csv | awk -F ',' '{print NF}'
39
}
40
40
40
40
40
40

```

Figure 4:

3 I completed this section with tr.

After doing the data processing required for (3):

ALL\_AML\_grow.train.noaffy.tmp.cutright contained: 7071 records,  
and each record contains: 35 fields, for a total of: 7068 fields.

ALL\_AML\_grow.test.noaffy.tmp.cutright contained: 7071 records,  
and each record contains: 40 fields (except for the first, for a total of:  
7068 fields.

This step is equivalent to transforming the file to a CSV file with newlines.

4 I completed this section with Notepad++ gui.

Changing the first field from one with spaces from one to without did not change the field count of either file.

5 I completed this section with a Python script

```
import sys
#constants
```

```

DEBUG = True
INPUT_FILE_NAME_TRAIN = "ALL_AML_grow.train.csv"
INPUT_FILE_NAME_TEST = "ALL_AML_grow.test.csv"
OUTPUT_FILE_NAME_TRAIN = "ALL_AML_train.csv.normalized"
OUTPUT_FILE_NAME_TEST = "ALL_AML_test.csv.normalized"

def debug(logMsg):
    if DEBUG:
        print(logMsg)

#just for a little bit of house keeping/debugging, we will keep a running list o
modified_lines_test = []

def proc_file(input_file_name, output_file_name):
    #https://stackoverflow.com/questions/3277503/how-do-i-read-a-file-line-b
    with open(input_file_name) as f:
        modified_lines = []

        #open the file which will be the train output; we don't want to
        out_file = open(output_file_name, "w")

        #create a list of lines, stripped of the newline
        content = [line.rstrip('\n') for line in f]
        firstValue = True
        firstRecord = True

        for line in content:
            modified_line = []
            int_strings = line.split(',')
            #print(int_strings)

            for s in int_strings:
                if (firstValue):
                    firstValue = False
                    modified_line.append(int_strings[0]);
                    int_strings.pop(0);
                    continue

                if(s.isspace() or s == ""):
                    continue

                if(int(s) < 20):
                    modified_line.append(str(20))
                elif (int(s) > 1600):
                    modified_line.append(str(1600))
                else:

```

```

                                modified_line.append(s)
                                #join the normalized values together again with commas,
                                modified_lines.append(",".join(modified_line) + "\n")
                                firstValue = True

                                out_file.writelines(modified_lines)
                                return modified_lines

modified_lines_train = proc_file(INPUT_FILE_NAME_TRAIN, OUTPUT_FILE_NAME_TRAIN)
modified_lines_test = proc_file(INPUT_FILE_NAME_TEST, OUTPUT_FILE_NAME_TEST)
#print(modified_lines_train)

```

**This code did not modify the structure of the files, except by adding a newline to the end of each file. If I find the newlines to be impediments, I will remove them manually with a text editor.**

## 6 I completed this section with another Python script.

```

#https://unix.stackexchange.com/questions/60590/is-there-a-command-line-utility-
import csv, sys
#constants DEBUG = True
INPUT_FILE_NAME_TRAIN = "ALL_AML_train.csv.normalized"
INPUT_FILE_NAME_TEST = "ALL_AML_test.csv.normalized"
OUTPUT_FILE_NAME_TRAIN = "ALL_AML_gcol_train.tmp"
OUTPUT_FILE_NAME_TEST = "ALL_AML_gcol_test.tmp"

def transposeRows(input_file_name, output_file_name):
    with open(input_file_name) as file:
        out_file = open(output_file_name, "w")
        rows = list(csv.reader(file))
        writer = csv.writer(out_file)
        for col in range(0, len(rows[0])):
            writer.writerow([row[col] for row in rows])

transposeRows(INPUT_FILE_NAME_TRAIN, OUTPUT_FILE_NAME_TRAIN)
transposeRows(INPUT_FILE_NAME_TEST, OUTPUT_FILE_NAME_TEST)

```

**A portion of this code is copied directly from Stack Overflow. I did not know how to idiomatically transpose a file in Python. The source is cited in the source file.**

```

jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ wc *.tmp
 34      34 926173 ALL_AML_gcol_test.tmp
 38      38 1036509 ALL_AML_gcol_train.tmp
 72      72 1962682 total
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ head ALL_AML_gcol_test.tmp | awk -F ',' '{print NF}'
7068
7068
7068
7068
7068
7068
7068
7068
7068
7068
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$

```

Figure 5:

```

jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ man awk
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ awk '{ $1=$1; print }' ALL_AML_idclass.test.txt > ALL_AML_idclass.test.txt.cut
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ awk '{ $1=$1; print }' ALL_AML_idclass.train.txt > ALL_AML_idclass.train.txt.cut
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ cut -f 1-2 ALL_AML_idclass.train.txt.cut > ALL_AML_idclass.train.txt.cutright
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ cut -f 1-2 ALL_AML_idclass.test.txt.cut > ALL_AML_idclass.test.txt.cutright
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ man cut
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ cut -d " " -f 1-2 ALL_AML_idclass.test.txt.cut > ALL_AML_idclass.test.txt.cutright
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$ cut -d " " -f 1-2 ALL_AML_idclass.train.txt.cut > ALL_AML_idclass.train.txt.cutright
jeremy@jeremy-VirtualBox:/media/sf_NMSU/Data Mining/hw4$

```

Figure 6:

7 I completed this section with awk and cut.

8 I completed this section with yet another Python script.

```

import sys, csv
INPUT_FILE_GCOL_TRAIN = "ALL_AML_gcol.train.tmp.cutcomma"
INPUT_FILE_NAME_TRAIN = "ALL_AML_idclass.train.txt.cutright"
OUTPUT_FILE_NAME_TRAIN = "ALL_AML_gcol_class.train.csv"
INPUT_FILE_GCOL_TEST = "ALL_AML_gcol.test.tmp.cutcomma"
INPUT_FILE_NAME_TEST = "ALL_AML_idclass.test.txt.cutright"
OUTPUT_FILE_NAME_TEST = "ALL_AML_gcol_class.test.csv"
def stitchFiles(input_file_name_samples, input_file_name_id_class, output_file_name):
    #https://stackoverflow.com/questions/3277503/how-do-i-read-a-file-line-by-line
    with open(input_file_name_samples) as samplesFile:
        modified_lines = []
        #which will
        input_file_id_class = open(input_file_name_id_class)
        #open the file which will be the train output; we don't want to
        out_file = open(output_file_name, "w")
        #create a list of lines, stripped of the newline
        sample_content = [line.rstrip('\n') for line in samplesFile]
        #create a list of "ID Class" strings, stripped of the newline

```

```

id_class_content = [line.rstrip('\n') for line in input_file_id_
print("len of sample_content: " + str(len(sample_content)))
print("len of id_class_content: " + str(len(id_class_content)))
i = 0
for i in range(0, len(sample_content)):
    sample_line = sample_content[i]
    id_class_line = id_class_content[i]
    id_class_lst = id_class_line.split(" ")
    modified_line = id_class_lst[0] + "," + sample_line + ",
    #join the stitched line
    modified_lines.append(modified_line)
out_file.writelines(modified_lines)
return modified_lines

```

```

modified_lines_train = stitchFiles(INPUT_FILE_GCOL_TRAIN, INPUT_FILE_NAME_TRAIN,
modified_lines_test = stitchFiles(INPUT_FILE_GCOL_TEST, INPUT_FILE_NAME_TEST, OU

```

## Part C

### 1

I finished this part with the WEKA ARFF viewer.

### 2

=== Classifier model (full training set) ===

ID:

< 27.5 -> ALL

>= 27.5 -> AML (38/38 instances correct)

Time taken to build model: 0.13 seconds

=== Evaluation on test set ===

This classifier (OneR), with ID not excluded, correctly classified 14/34 test records.

### 3

#### OneR

=== Classifier model (full training set) ===

X95735\_at:

< 994.0 -> ALL

>= 994.0 -> AML (38/38 instances correct)

Time taken to build model: 0.09 seconds

**OneR excluding ID, with the training set as the test set, correctly classified all instances.**

**OneR w/ ALL\_AML\_gcol\_class.test-no-id.arff**

```
=== Classifier model (full training set) ===
X95735_at:
< 994.0 -> ALL
>= 994.0 -> AML (38/38 instances correct)
Time taken to build model: 0.09 seconds
```

**OneR excluding ID correctly classified 21/34 instances, for a ~61.76% accuracy.**

**J48**

```
=== Classifier model (full training set) ===
J48 pruned tree
-----
X95735_at <= 938: ALL (27.0)
X95735_at > 938: AML (11.0)
Number of Leaves : 2
Size of the tree : 3
```

**J48 excluding ID, with the training set as the test set, correctly classified all instances.**

**J48 w/ ALL\_AML\_gcol\_class.test-no-id.arff**

```
=== Classifier model (full training set) ===
J48 pruned tree
-----
X95735_at <= 938: ALL (27.0)
X95735_at > 938: AML (11.0)
Number of Leaves : 2
Size of the tree : 3
```

**J48 excluding ID correctly classified 21/34 instances, for a ~61.76% accuracy.**

**Naive Bayes**

**Naive Bayes excluding ID, with the training set as the test set, correctly classified all instances.**



**Naive Bayes w/ ALL\_AML\_gcol\_class.test-no-id.arff**

**Naive Bayes excluding ID correctly classified 21/34 instances, for a  
~61.76% accuracy.**