

HOMEWORK 8

NAME: KENIGBOLO MEYA STEPHEN
COURSE: DATA MINING

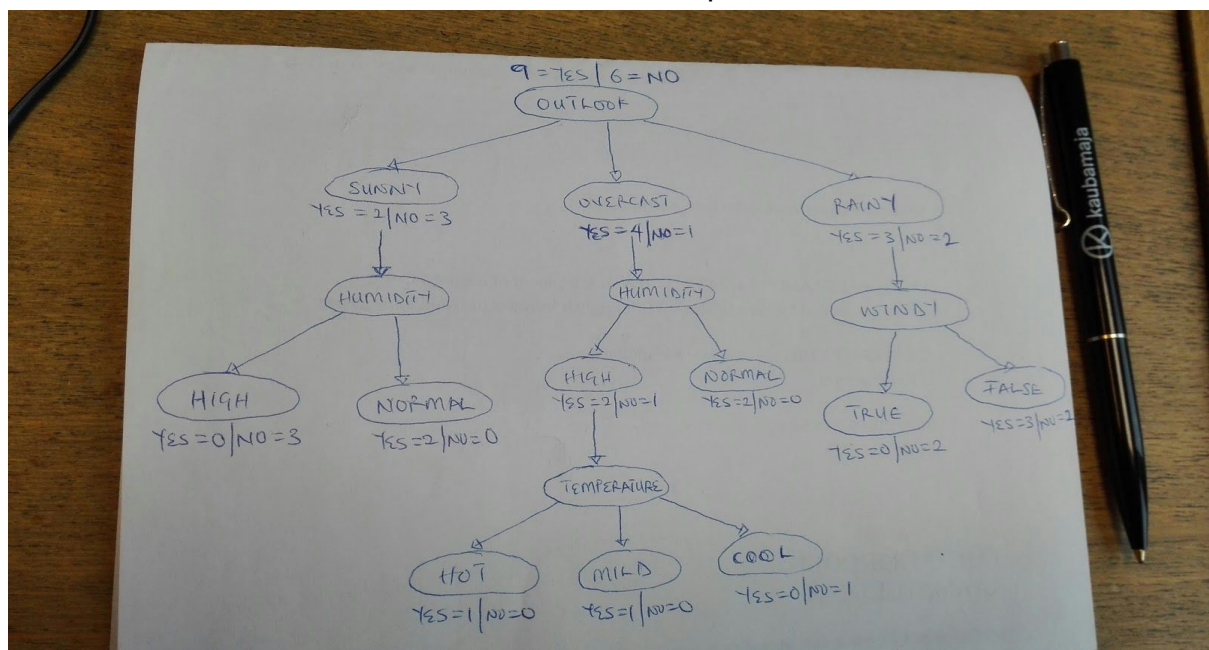
1. Read - <http://www.r2d3.us/visual-intro-to-machine-learning-part-1/> What is the quality of the classifier? Can you understand when it works well and when not?

The quality of any classifier depends on several factors including but not limited to complexity, classification speed etc. In the above article, using the classifier the decision trees classifier can be argued to be about 80% okay because the data included categorical variables and this is what decision trees work better as (i.e. Decision trees cannot predict numerical variables). Also Decision trees are a good way of explaining how the classifier works to the targeted audience and this is due to the fact that they can be easily explained to most people (Bayesian also works well in this case too anyways).

In conclusion I can categorically state that decision tree is a good fit because we are predicting a category from data that is labelled however in the case wherever the data isn't labelled then it will make more sense to perform clustering instead of classification

2. Use this small data example and build a decision tree (manually, explaining all steps/choices).

I made use of the ID3 algorithm for drawing this decision tree and below this image I have listed down both the data used and each step/choices made in the tree.



Given the data below

ord.	Outlook	Temp	Humidity	Windy	Play
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
3	Overcast	Hot	High	FALSE	Yes
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	No
7	Overcast	Cool	Normal	TRUE	Yes
8	Sunny	Mild	High	FALSE	No
9	Sunny	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
14	Rainy	Mild	High	TRUE	No
15	Overcast	Cool	High	FALSE	No

And given the follow up question “providing that there is mild, overcast, high humidity and high wind weather - should one play tennis or not?”, I proceeded by first identifying the best attribute for splitting the the training set of YES = 9 and NO = 6 and selected the outlook for this purpose because the after subsetting for YES and NO I discovered that overcast only has one “NO” response. Splitting the outlook into the three classifications the following results were obtained.

SUNNY (YES =2 & NO = 3)

ord.	Outlook	Temp	Humidity	Windy	Play
9	Sunny	Cool	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
8	Sunny	Mild	High	FALSE	No

OVERCAST (YES = 4 & NO = 1)

ord.	Outlook	Temp	Humidity	Windy	Play
3	Overcast	Hot	High	FALSE	Yes
7	Overcast	Cool	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
15	Overcast	Cool	High	FALSE	No

RAINY (YES = 3 & NO = 2)

ord.	Outlook	Temp	Humidity	Windy	Play
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes

10	Rainy	Mild	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	No
14	Rainy	Mild	High	TRUE	No

Since there were no pure sets yet I decided to further split into child nodes for all outlook values. Looking through the Sunny outlook I discovered that I could get two pure subsets by further splitting by Humidity

SUNNY AND HIGH HUMIDITY

ord.	Outlook	Temp	Humidity	Windy	Play
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
8	Sunny	Mild	High	FALSE	No

SUNNY AND NORMAL HUMIDITY

ord.	Outlook	Temp	Humidity	Windy	Play
9	Sunny	Cool	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes

At the node for the Overcast humidity I discovered the possibility to get one pure subset by splitting to a child node via humidity which I did

OVERCAST AND HIGH HUMIDITY

ord.	Outlook	Temp	Humidity	Windy	Play
3	Overcast	Hot	High	FALSE	Yes
12	Overcast	Mild	High	TRUE	Yes
15	Overcast	Cool	High	FALSE	No

OVERCAST AND NORMAL HUMIDITY

ord.	Outlook	Temp	Humidity	Windy	Play
7	Overcast	Cool	Normal	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes

After going through the data of the impure subset I figured out that it will be possible to split to three pure subsets using the temperature. Alas I got three pure subsets as can be seen below

OVERCAST WITH HIGH HUMIDITY AND HOT TEMPERATURE

ord.	Outlook	Temp	Humidity	Windy	Play
3	Overcast	Hot	High	FALSE	Yes

OVERCAST WITH HIGH HUMIDITY AND MILD TEMPERATURE

ord.	Outlook	Temp	Humidity	Windy	Play
12	Overcast	Mild	High	TRUE	Yes

OVERCAST WITH HIGH HUMIDITY AND COOL TEMPERATURE

ord.	Outlook	Temp	Humidity	Windy	Play
15	Overcast	Cool	High	FALSE	No

After getting the pure subsets for overcast, I proceeded to windy and after careful examination I discovered that I could get out two pure subsets using the Windy classification and alas I finally did.

RAIN AND WINDY TRUE

ord.	Outlook	Temp	Humidity	Windy	Play
6	Rainy	Cool	Normal	TRUE	No
14	Rainy	Mild	High	TRUE	No

RAIN AND WINDY FALSE

ord.	Outlook	Temp	Humidity	Windy	Play
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes

In conclusion from the decision tree above we can now give an answer to the question “providing that there is mild, overcast, high humidity and high wind weather - should one play tennis or not?” Which is that “Yes one should play Tennis”

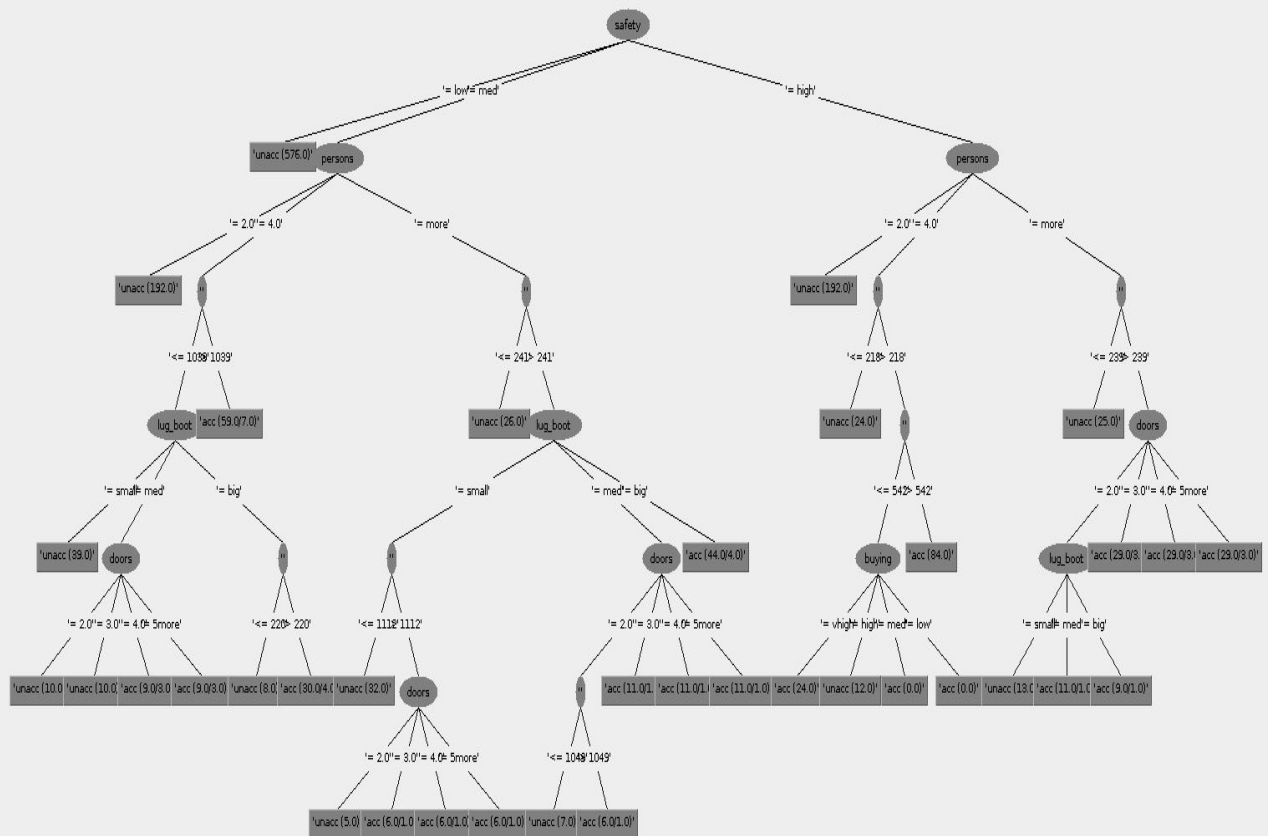
3. Use the Cars data set and apply decision trees for classification. Describe the tree. (you can use R, or Weka (install Weka from [here](#)), or python...). Compare the decision tree approach to the association rules derived from the same data.

- To make your life easier, we recommend you remove observations with two infrequent classes - *good* and *v-good*. You can get the resulting dataset [here](#)
- in R, you can use library *rpart* to build the trees and *rpart.plot* to visualize them

In order to do this task I first converted the data into a .csv file by reading the .txt file into R, formatting it properly and writing it out to a .csv file using the following commands.

After this I proceeded to read the data into Weka and then proceeded to click on the classify tab and selected the J48 classifier to build the tree. I used the default parameters however I opted to build this tree with a training set. The resulting tree is as seen below

Tree View



In the appendix section of this Homework you will find the complete breakdown of the entire tree that was visualized above.

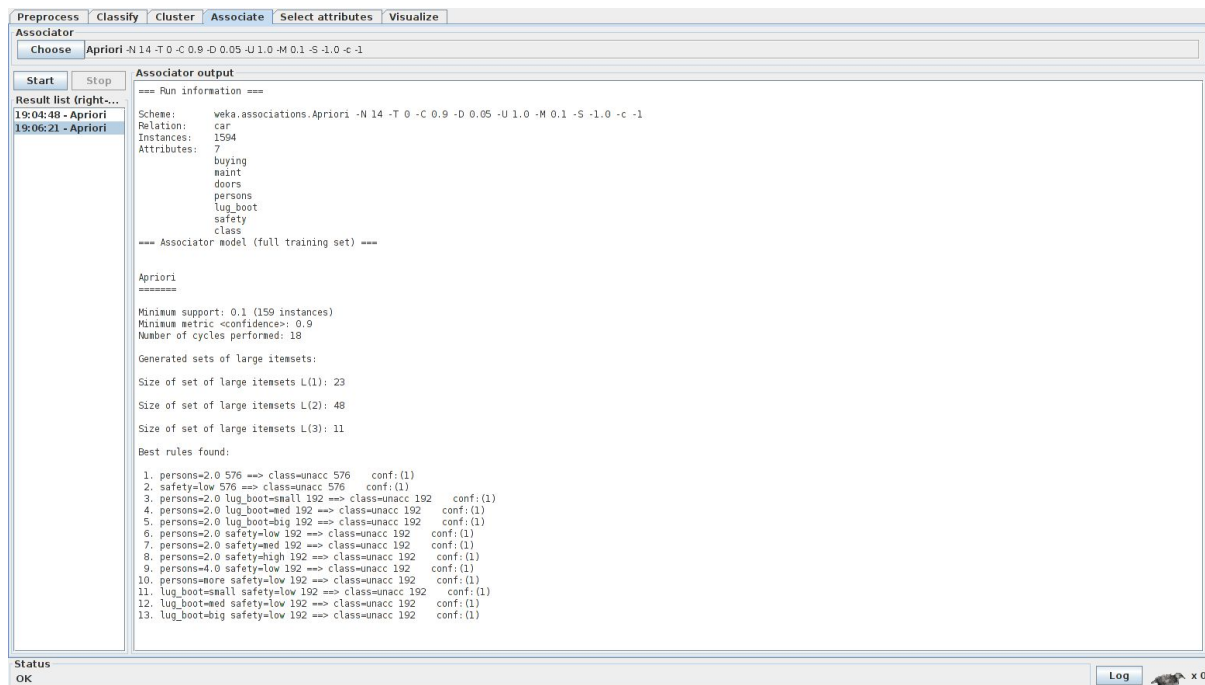
Using the training set as my test option resulted in giving me 97.5533% classification accuracy which is quite high as can be seen in the summary below.

=== Evaluation on training set ===

=== Summary ===

Correctly Classified Instances	1555	97.5533 %
Incorrectly Classified Instances	39	2.4467 %
Kappa statistic	0.9353	
Mean absolute error	0.0417	
Root mean squared error	0.1443	
Relative absolute error	11.3844 %	
Root relative squared error	33.7486 %	
Total Number of Instances	1594	

I also used the apriori algorithm in Weka to mine the rules and the following output was gotten after setting the number of rules to be generated to be 14

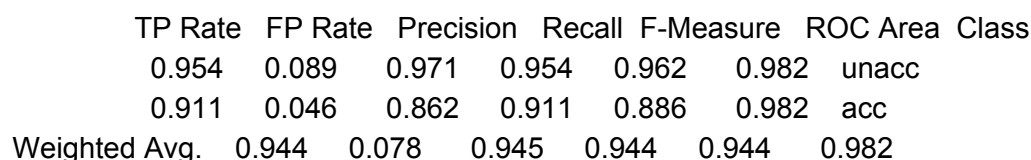


Best rules found:

1. persons=2.0 576 ==> class=unacc 576 conf:(1)
2. safety=low 576 ==> class=unacc 576 conf:(1)
3. persons=2.0 lug_boot=small 192 ==> class=unacc 192 conf:(1)
4. persons=2.0 lug_boot=med 192 ==> class=unacc 192 conf:(1)
5. persons=2.0 lug_boot=big 192 ==> class=unacc 192 conf:(1)
6. persons=2.0 safety=low 192 ==> class=unacc 192 conf:(1)
7. persons=2.0 safety=med 192 ==> class=unacc 192 conf:(1)
8. persons=2.0 safety=high 192 ==> class=unacc 192 conf:(1)
9. persons=4.0 safety=low 192 ==> class=unacc 192 conf:(1)
10. persons=more safety=low 192 ==> class=unacc 192 conf:(1)
11. lug_boot=small safety=low 192 ==> class=unacc 192 conf:(1)
12. lug_boot=med safety=low 192 ==> class=unacc 192 conf:(1)
13. lug_boot=big safety=low 192 ==> class=unacc 192 conf:(1)

In regards to comparison I can say that from my understanding, the basic difference between association rules and decision trees is simply that association rules basically focus on detecting relationships between categorical variables in the data set whereas the decision trees basically is purely a classification technique used predict a target (i.e. they map the set of record features into the class attribute which is the object of classification and target variable.

a) Decision Tree



=== Confusion Matrix ===

```

a  b  <-- classified as
1154 56 | a = unacc
34 350 | b = acc

```

b) NaiveBayes

The screenshot shows the Orange3 software interface with the NaiveBayes classifier selected. The 'Classifier output' tab displays the following results:

Test options:

- Use training set
- Supplied test set
- Cross-validation** (Folds: 10)
- Percentage split (%: 66)

Classifier output:

persons

2.0	577.0	1.0
4.0	313.0	199.0
more	323.0	187.0
[total]	1213.0	387.0

lug_boot

small	451.0	106.0
med	393.0	136.0
big	369.0	145.0
[total]	1213.0	387.0

safety

low	577.0	1.0
med	358.0	181.0
high	278.0	205.0
[total]	1213.0	387.0

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	1442	90.4642 %
Incorrectly Classified Instances	152	9.5358 %
Kappa statistic	0.7256	
Mean absolute error	0.1737	
Root mean squared error	0.278	
Relative absolute error	47.4695 %	
Root relative squared error	65.0166 %	
Total Number of Instances	1594	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
unacc	0.96	0.271	0.918	0.96	0.939	0.972	unacc
acc	0.729	0.04	0.854	0.729	0.787	0.972	acc
Weighted Avg.	0.905	0.215	0.902	0.905	0.902	0.972	

=== Confusion Matrix ===

```

a  b  <-- classified as
1162 48 | a = unacc
104 280 | b = acc

```

Applying Naive bayes for the decision Tree classifier the following values are obtainable for 2x2 tables of TP, FP, FN, TN

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	1442	90.4642 %
Incorrectly Classified Instances	152	9.5358 %
Kappa statistic	0.7256	
Mean absolute error	0.1737	
Root mean squared error	0.278	
Relative absolute error	47.4695 %	
Root relative squared error	65.0166 %	
Total Number of Instances	1594	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.96	0.271	0.918	0.96	0.939	0.972	unacc
	0.729	0.04	0.854	0.729	0.787	0.972	acc
Weighted Avg.	0.905	0.215	0.902	0.905	0.902	0.972	

=== Confusion Matrix ===

```

a  b  <-- classified as
1162 48 | a = unacc
104 280 | b = acc

```

From the above output I can say that the Decision Tree classifier (J48) is better than the Naive Bayes classifier for this data because of the difference in accuracy. While the J48 classifier gave an accuracy (correctly classified instances) of 94.3538 % the Naive Bayes classifier gave an accuracy (correctly classified instances) of 90.4642 % however the time taken to generate the models differ as it took 0.04 seconds for the Decision tree as opposed to 0.02 seconds for the Naive Bayes.

5. Use the Titanic data set - compare your classifiers learned from Titanic data - decision trees, Bayes rules, association rules - and try to characterise the rules observed in data using these approaches. How can they be interpreted against each other?

a) Decision Trees (J48)

The screenshot shows the WEKA software interface with the J48 classifier selected. The 'Test options' section shows 'Cross-validation' with 'Folds' set to 10. The 'Classifier output' section displays the J48 pruned tree, stratified cross-validation summary, detailed accuracy by class, and the confusion matrix.

Classifier output

```

J48 pruned tree
-----
<= 1490: No (1490.0)
> 1490: Yes (711.0)

Number of Leaves :    2
Size of the tree :    3

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      2200          99.9546 %
Incorrectly Classified Instances        1           0.0454 %
Kappa statistic                    0.999
Mean absolute error                 0.0005
Root mean squared error             0.0213
Relative absolute error              0.1039 %
Root relative squared error          4.5581 %
Total Number of Instances          2201

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      -----  -
      0.999    0.001    0.999    0.999    1.000    1.000    No
      1.000    0.000    1.000    1.000    1.000    1.000    Yes
Weighted Avg.    1.000    0.000    1.000    1.000    1.000    1.000

=== Confusion Matrix ===

a  b  <-- classified as
1489 1 | a = No
  12 0 | b = Yes

```

=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2

Relation: titanic

Instances: 2201

Attributes: 5

Class

Sex

Age

Survived

Test mode:10-fold cross-validation

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

J48 pruned tree

<= 1490: No (1490.0)

> 1490: Yes (711.0)

Number of Leaves : 2

Size of the tree : 3

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	2200	99.9546 %
Incorrectly Classified Instances	1	0.0454 %
Kappa statistic	0.999	
Mean absolute error	0.0005	
Root mean squared error	0.0213	
Relative absolute error	0.1039 %	
Root relative squared error	4.5581 %	
Total Number of Instances	2201	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.999	0	1	0.999	1	1	No
	1	0.001	0.999	1	0.999	1	Yes
Weighted Avg.	1	0	1	1	1	1	

=== Confusion Matrix ===

```

a  b  <-- classified as
1489  1 |  a = No
    0 711 |  b = Yes

```

2) Bayes Rule (NaiveBayes)

The screenshot shows the Weka GUI with the 'Classify' tab selected. The 'Classifier' dropdown is set to 'NaiveBayes'. The 'Test options' section shows 'Cross-validation' with 'Folds' set to 10. The 'Result list' shows the current run as '22:09:35 - bayes.NaiveBayes'. The 'Classifier output' pane displays the following information:

Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===

=== Summary ===

Metric	Value	Percentage
Correctly Classified Instances	2092	95.0477 %
Incorrectly Classified Instances	109	4.9523 %
Kappa statistic	0.8911	
Mean absolute error	0.0704	
Root mean squared error	0.1776	
Relative absolute error	16.0906 %	
Root relative squared error	37.9688 %	
Total Number of Instances	2201	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.927	0	1	0.927	0.962	0.994	No
	1	0.073	0.867	1	0.929	0.994	Yes
Weighted Avg.	0.95	0.024	0.957	0.95	0.951	0.994	

=== Confusion Matrix ===

```

a  b  <-- classified as
1381 109 |  a = No
    0 711 |  b = Yes

```

=== Run information ===

Scheme: weka.classifiers.bayes.NaiveBayes

Relation: titanic

Instances: 2201

Attributes: 5

Class

Sex

Age

Survived

Test mode: 10-fold cross-validation

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

Naive Bayes Classifier

	Class	
Attribute	No	Yes
	(0.68)	(0.32)

=====

mean	745.5	1846
std. dev.	430.1259	205.2478
weight sum	1490	711
precision	1	1

Class		
3rd	529.0	179.0
1st	123.0	204.0
2nd	168.0	119.0
Crew	674.0	213.0
[total]	1494.0	715.0

Sex		
Male	1365.0	368.0
Female	127.0	345.0
[total]	1492.0	713.0

Age		
Child	53.0	58.0
Adult	1439.0	655.0
[total]	1492.0	713.0

Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	2092	95.0477 %
Incorrectly Classified Instances	109	4.9523 %
Kappa statistic	0.8911	
Mean absolute error	0.0704	
Root mean squared error	0.1776	
Relative absolute error	16.0906 %	
Root relative squared error	37.9688 %	
Total Number of Instances	2201	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.927	0	1	0.927	0.962	0.994	No
	1	0.073	0.867	1	0.929	0.994	Yes
Weighted Avg.	0.95	0.024	0.957	0.95	0.951	0.994	

=== Confusion Matrix ===

```

a  b  <-- classified as
1381 109 | a = No
0 711 | b = Yes

```

3) Association Rule(Apriori) -> Performed this with R

```

139 #''[r]
140 #load package for frequent set mining
141 library(arules)
142 library(arulesviz)
143
144 titanic <- read.table( "C:/Users/Kenigbolo PC/Desktop/Data Mining/titanic.txt", sep = ',', header = TRUE)
145
146 #observe the data
147 #first 6 observations
148 head(titanic)
149 #types of features
150 str(titanic)
151 #dimensionality of the data
152 dim(titanic)
153 |
154 |
155 #run apriori algorithm with default settings
156 rules = apriori(titanic)
157
158 #inspection of the result
159 inspect(rules)
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Apriori

Parameter specification:

```

confidence minval smax arem  aval originalSupport support minlen maxlen target  ext
0.8 0.1 1 none FALSE TRUE 0.1 1 10 rules FALSE

```

Algorithmic control:

```

filter tree heap memopt load sort verbose
0.1 TRUE TRUE FALSE TRUE 2 TRUE

```

Absolute minimum support count: 220

set item appearances ...[0 item(s)] done [0.00s].
 set transactions ...[10 item(s), 2201 transaction(s)] done [0.00s].
 sorting and recoding items ... [9 item(s)] done [0.00s].
 creating transaction tree ... done [0.00s].
 checking subsets of size 1 2 3 4 done [0.00s].
 writing ... [27 rule(s)] done [0.00s].
 creating S4 object ... done [0.00s].

lhs	rhs	support	confidence	lift
1 {}	=> {Age=Adult}	0.9504771	0.9504771	1.0000000
2 {Class=2nd}	=> {Age=Adult}	0.1185825	0.9157895	0.9635051
3 {Class=1st}	=> {Age=Adult}	0.1449341	0.9815385	1.0326798
4 {Sex=Female}	=> {Age=Adult}	0.1930940	0.9042553	0.9513700
5 {Class=3rd}	=> {Age=Adult}	0.2848705	0.8881020	0.9343750
6 {Survived=Yes}	=> {Age=Adult}	0.2971377	0.9198312	0.9677574
7 {Class=Crew}	=> {Sex=Male}	0.3916402	0.9740113	1.2384742
8 {Class=Crew}	=> {Age=Adult}	0.4020900	1.0000000	1.0521033
9 {Survived=No}	=> {Sex=Male}	0.6197183	0.9154362	1.1639949
10 {Survived=No}	=> {Age=Adult}	0.6533394	0.9651007	1.0153856
11 {Sex=Male}	=> {Age=Adult}	0.7573830	0.9630272	1.0132040
12 {Sex=Female,Survived=Yes}	=> {Age=Adult}	0.1435711	0.9186047	0.9664669
13 {Class=3rd,Sex=Male}	=> {Survived=No}	0.1917310	0.8274510	1.2222950
14 {Class=3rd,Survived=No}	=> {Age=Adult}	0.2162653	0.9015152	0.9484870
15 {Class=3rd,Sex=Male}	=> {Age=Adult}	0.2099046	0.9058824	0.9530818
16 {Sex=Male,Survived=Yes}	=> {Age=Adult}	0.1535666	0.9209809	0.9689670
17 {Class=Crew,Survived=No}	=> {Sex=Male}	0.3044071	0.9955423	1.2658514
18 {Class=Crew,Survived=No}	=> {Age=Adult}	0.3057701	1.0000000	1.0521033
19 {Class=Crew,Sex=Male}	=> {Age=Adult}	0.3916402	1.0000000	1.0521033
20 {Class=Crew,Age=Adult}	=> {Sex=Male}	0.3916402	0.9740113	1.2384742
21 {Sex=Male,Survived=No}	=> {Age=Adult}	0.6038164	0.9743402	1.0251065
22 {Age=Adult,Survived=No}	=> {Sex=Male}	0.6038164	0.9242003	1.1751385
23 {Class=3rd,Sex=Male,Survived=No}	=> {Age=Adult}	0.1758292	0.9170616	0.9648435
24 {Class=3rd,Age=Adult,Survived=No}	=> {Sex=Male}	0.1758292	0.8130252	1.0337773
25 {Class=3rd,Sex=Male,Age=Adult}	=> {Survived=No}	0.1758292	0.8376623	1.2373791
26 {Class=Crew,Sex=Male,Survived=No}	=> {Age=Adult}	0.3044071	1.0000000	1.0521033
27 {Class=Crew,Age=Adult,Survived=No}	=> {Sex=Male}	0.3044071	0.9955423	1.2658514

From analyzing the decision trees, Bayes rules, association rules it is obvious that sex clearly had the most significant relationship demonstrated within the dataset in terms of the rate of survival. It is also worth noting that the J48 classifier, using the test data set resulted in ~100% (99.9546 %) correctly classified instances as opposed to ~94% (95.0477 %) when using the NaiveBayes classifier.

6. (Bonus 1p) How to detect and avoid overfitting? What is the good (optimal?) size of the decision tree classifiers? Use the above Cars data, and for comparison use one of the two data sets - the Mushroom ([LINK](#)) or the Connect 4 ([LINK](#)).

APPENDIX
DECISION TREE FOR TASK 3

Test mode: evaluate on training data

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

J48 pruned tree

```
safety = low: unacc (576.0)
safety = med
| persons = 2.0: unacc (192.0)
| persons = 4.0
| | buying = vhigh
| | | maint = vhigh: unacc (12.0)
| | | maint = high: unacc (12.0)
| | | maint = med
| | | | lug_boot = small: unacc (4.0)
| | | | lug_boot = med: unacc (4.0/2.0)
| | | | lug_boot = big: acc (4.0)
| | | maint = low
| | | | lug_boot = small: unacc (4.0)
| | | | lug_boot = med: unacc (4.0/2.0)
| | | | lug_boot = big: acc (4.0)
| | buying = high
| | | lug_boot = small: unacc (16.0)
| | | lug_boot = med
| | | | doors = 2.0: unacc (4.0)
| | | | doors = 3.0: unacc (4.0)
| | | | doors = 4.0: acc (4.0/1.0)
| | | | doors = 5more: acc (4.0/1.0)
| | | lug_boot = big
| | | | maint = vhigh: unacc (4.0)
| | | | maint = high: acc (4.0)
| | | | maint = med: acc (4.0)
| | | | maint = low: acc (4.0)
| | buying = med
| | | maint = vhigh
| | | | lug_boot = small: unacc (4.0)
| | | | lug_boot = med: unacc (4.0/2.0)
| | | | lug_boot = big: acc (4.0)
| | | maint = high
| | | | lug_boot = small: unacc (4.0)
| | | | lug_boot = med: unacc (4.0/2.0)
| | | | lug_boot = big: acc (4.0)
| | | maint = med: acc (12.0)
```


- | | | maint = low: acc (6.0)
- | | buying = low: acc (36.0/6.0)
- | persons = more
 - | | lug_boot = small
 - | | | buying = vhigh: unacc (16.0)
 - | | | buying = high: unacc (16.0)
 - | | | buying = med
 - | | | | maint = vhigh: unacc (4.0)
 - | | | | maint = high: unacc (4.0)
 - | | | | maint = med: acc (4.0/1.0)
 - | | | | maint = low: acc (4.0/1.0)
 - | | | buying = low
 - | | | | maint = vhigh: unacc (4.0)
 - | | | | maint = high: acc (4.0/1.0)
 - | | | | maint = med: acc (4.0/1.0)
 - | | | | maint = low: acc (4.0/1.0)
 - | | | lug_boot = med
 - | | | buying = vhigh
 - | | | | maint = vhigh: unacc (4.0)
 - | | | | maint = high: unacc (4.0)
 - | | | | maint = med: acc (4.0/1.0)
 - | | | | maint = low: acc (4.0/1.0)
 - | | | buying = high
 - | | | | maint = vhigh: unacc (4.0)
 - | | | | maint = high: acc (4.0/1.0)
 - | | | | maint = med: acc (4.0/1.0)
 - | | | | maint = low: acc (4.0/1.0)
 - | | | buying = med: acc (13.0/2.0)
 - | | | buying = low: acc (10.0/1.0)
 - | | | lug_boot = big
 - | | | buying = vhigh
 - | | | | maint = vhigh: unacc (4.0)
 - | | | | maint = high: unacc (4.0)
 - | | | | maint = med: acc (4.0)
 - | | | | maint = low: acc (4.0)
 - | | | buying = high
 - | | | | maint = vhigh: unacc (4.0)
 - | | | | maint = high: acc (4.0)
 - | | | | maint = med: acc (4.0)
 - | | | | maint = low: acc (4.0)
 - | | | buying = med: acc (12.0)
 - | | | buying = low: acc (8.0)
- safety = high
 - | persons = 2.0: unacc (192.0)
 - | persons = 4.0
 - | | maint = vhigh

```

| | | buying = vhigh: unacc (12.0)
| | | buying = high: unacc (12.0)
| | | buying = med: acc (12.0)
| | | buying = low: acc (12.0)
| | maint = high
| | | buying = vhigh: unacc (12.0)
| | | buying = high: acc (12.0)
| | | buying = med: acc (12.0)
| | | buying = low: acc (6.0)
| | maint = med: acc (30.0)
| | maint = low: acc (24.0)
| persons = more
| | maint = vhigh
| | | buying = vhigh: unacc (12.0)
| | | buying = high: unacc (12.0)
| | | buying = med: acc (12.0/1.0)
| | | buying = low: acc (12.0/1.0)
| | maint = high
| | | buying = vhigh: unacc (12.0)
| | | buying = high: acc (12.0/1.0)
| | | buying = med: acc (12.0/1.0)
| | | buying = low: acc (5.0/1.0)
| | maint = med: acc (30.0/4.0)
| | maint = low: acc (26.0/4.0)

```

Number of Leaves : 79

Size of the tree : 108