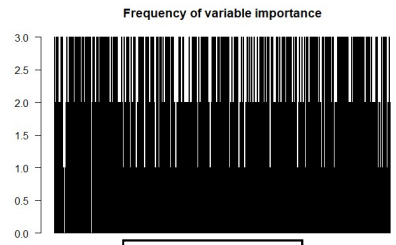
P1:

A R package named "impute" is used for imputation of missing data point. The missing value is replaced by using the "mean" option. A R package named random forest is used to train the model, which uses a CART tree to grow the random forest.

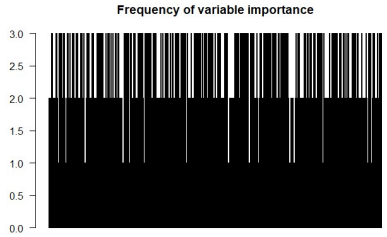
The following is the plotted frequency for the random forest with number of features fixed at 70% of the total number of features and the number of instances vary from 50% - 90%, and with number of instances fixed at 70% of the total number of instances and the number of features vary from 50% - 90%, . The label of y – axis is in form I_S(percentage of sample used)_F(percentage of feature used). The name of each gene is to long to be shown in the plot so the plot is in the order of gene with index from 1 to 800. The index is assigned to genes according to the given ordering in the original data file.

The plot gives the number of appearance of each gene in the random forest model, which should be proportional to frequency of appearance of each gene.

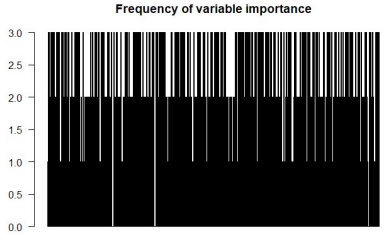
Frequency Plot with Instances fixed at 0.7



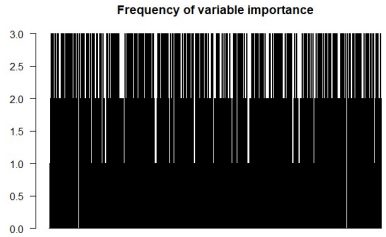
S_0.7_F_0.50



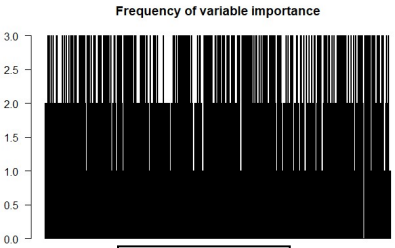
S_0.7_F_0.55



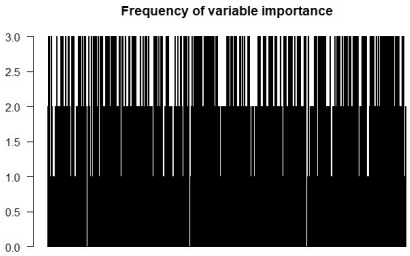
S_0.7_F_0.60



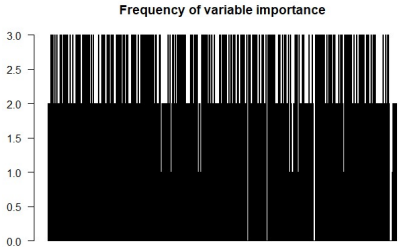
S_0.7_F_0.65



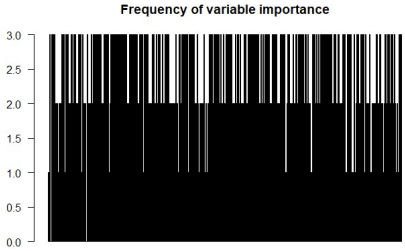
S_0.7_F_0.70



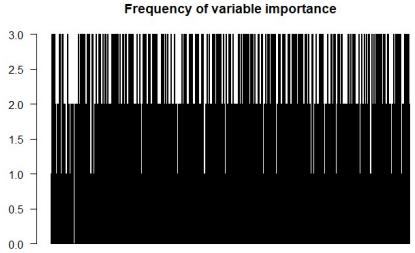
S_0.7_F_0.75



S_0.7_F_0.80

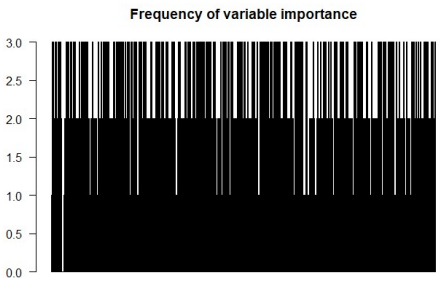


S_0.7_F_0.85

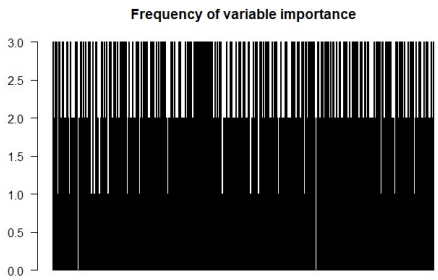


S_0.7_F_0.90

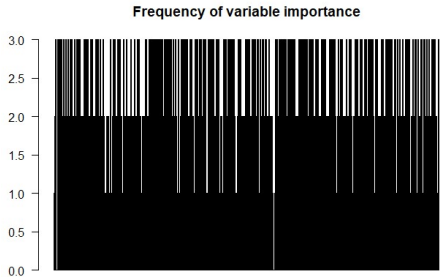
Frequency Plot with Feature fixed at 0.7



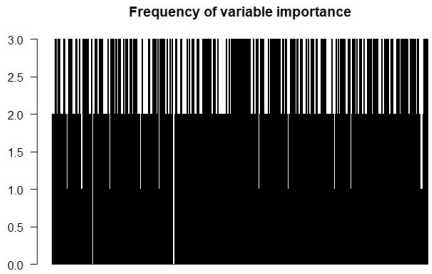
S_0.50_F_0.7



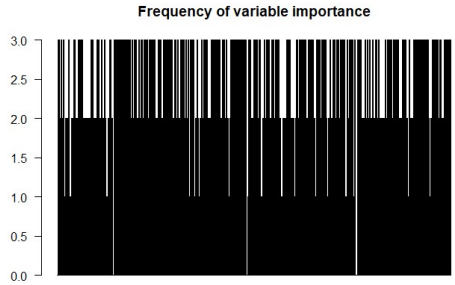
S_0.55_F_0.7



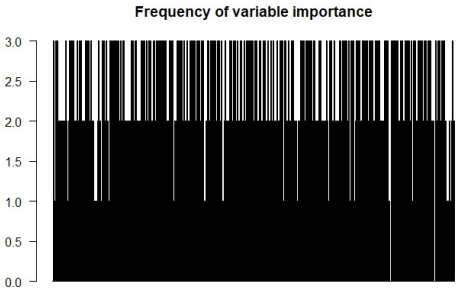
S_0.60_F_0.7



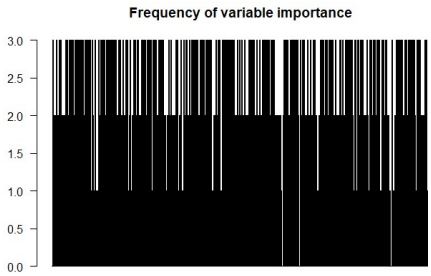
S_0.65_F_0.7



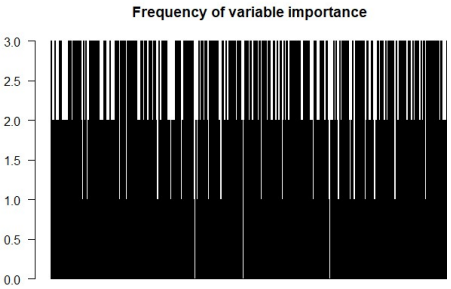
S_0.70_F_0.7



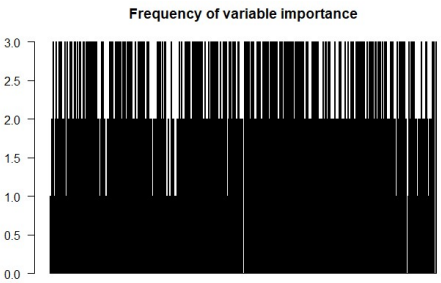
S_0.75_F_0.7



S_0.80_F_0.7



S_0.85_F_0.7



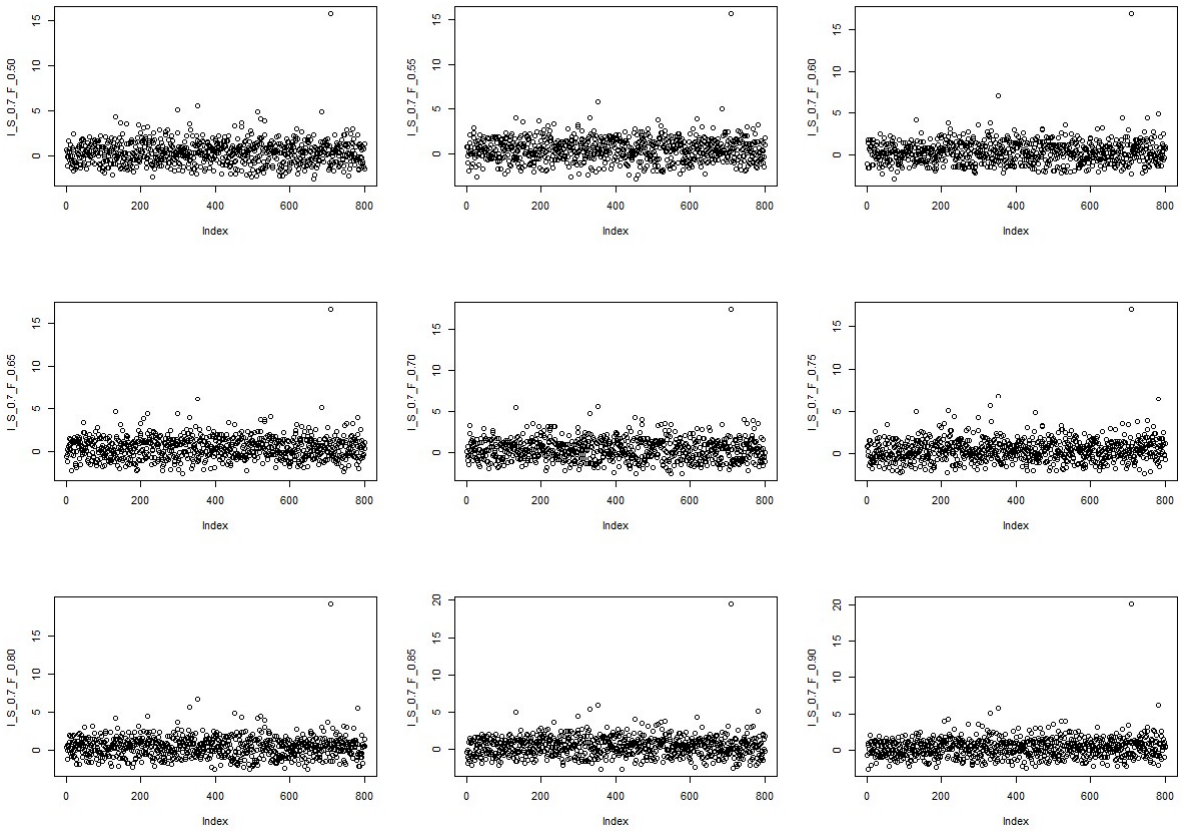
S_0.90_F_0.7

Intuitively we know that the more frequent a feature appears in the random forest, the more likely it will be important for feature selection. That says, for feature selection, we prefer a gene that appears more times in all the trained random forest model. Based on the model we generated, we could compute the total number of appearance of each gene and choose genes with top 100 average number of appearance. A problem of this method is that, there are lots of gene with same total number of appearance, which is clear from the plots above, and it is hard to choose top 100 features when there are more than 200 features have same number of times of appearance and that number is in the range of top 100 far exceed 100. Given this problem we can introduce another term, "Importance", as a measure of gene importance.

The R package used to grow the random forest has two available type of importance measure stored in the model, mean decrease in accuracy and mean decrease in node impurity. Mean decrease in accuracy is computed from permuting OOB data: For each tree, the prediction error on

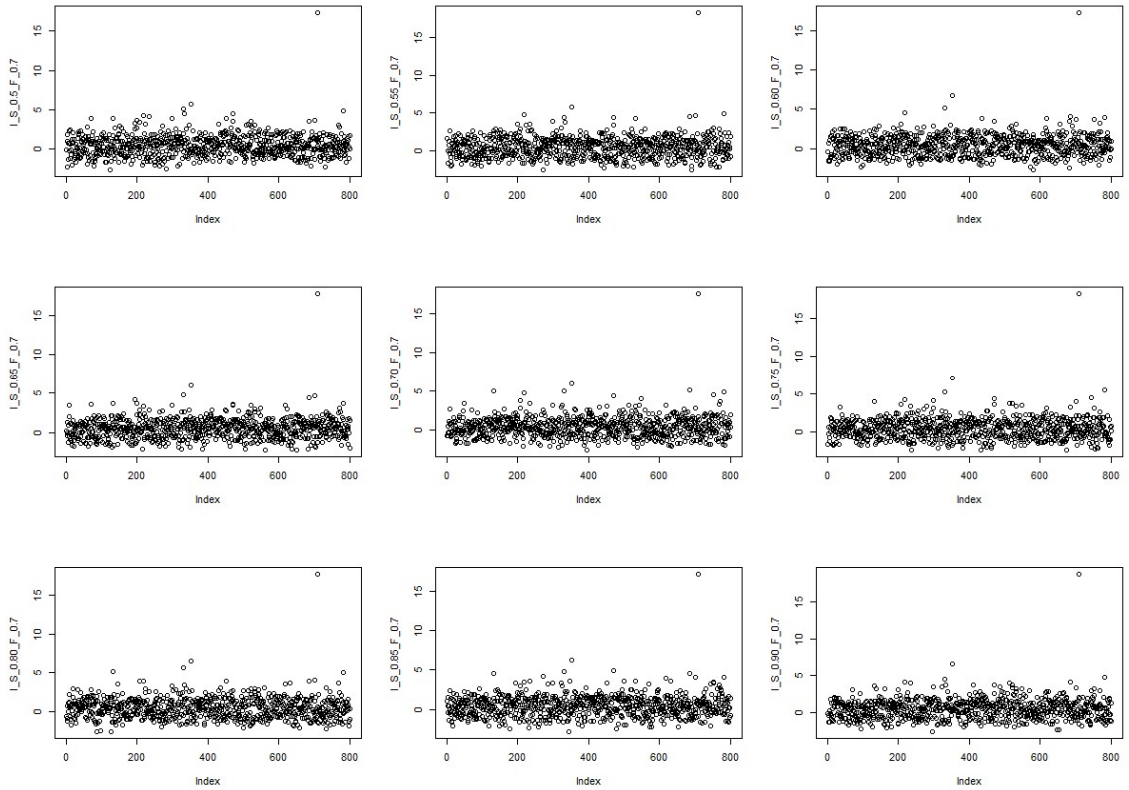
the out-of-bag portion of the data is recorded (error rate for classification). Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and then normalized by the standard deviation of the differences. If the standard deviation of the differences is equal to 0 for a variable, the division is not done (but the average is almost always equal to 0 in that case). The second measure is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For the purpose of feature selection, we prefer features with larger importance value. Since try both method gives similar ranks of variable importance, only the plot of feature importance using decreases in accuracy as the measure is reported here.

Frequency Plot with Instances fixed at 0.7



Continued next page

Frequency Plot with Feature fixed at 0.7



Using importance as a measure for feature selection seems to give more reasonable result. The rank of every feature is computed for every random forest model and their average rank is sorted in descending order to select top 100 feature. It seems that the top 10 features are stable across random forest models with different parameters, yet there are more variations in the rest features. The following is the index for top 100 feature, ordered by ascending importance.

```
764 376 464 15 80 473 635 310 430 118 456 337 72 624 572 734 175 790 598 134
210 396 136 314 544 422 360 706 224 275 199 208 765 702 729 368 586 747 540
45 333 375 533 189 400 427 322 404 327 256 606 7 150 600 71 298 316 200 399 4
65 439 558 531 222 653 207 542 69 632 223 47 617 770 299 436 650 160 234 192
218 225 236 512 9 470 684 547 522 178 780 532 752 769 743 471 450 329 131 353
 708
```

To check if the ordering of genes are stable, using the same method described above, generate 10 random forest with same parameter (70% instances and 70% features are used here) and order the top 100 gene by ascending importance as above. Compare result from different random forest model and check if the ordering of genes is the same among models. To quantify stability, average of pairwise similarity between top N genes are computed for all 10 random forest models. It seems that the top 5 gene does not change among modles, there are some variations in top 10 genes (average similarity between models are 87.3%) and lots of variations in top 100 genes (average similarity reduced to 32.5%). One thing causes the instability of ordering could be that the features (genes) are correlated to each other so the model can use the features somewhat interchangeably, which causes importance of features to change. One possible way to solve the problem could be adding more trees to the random forest model, but it would require more space for computation.

P2:

The data imputed in P1 is loaded and processed in R. A R package named "e1071" designed for SVM is used to train the model. All model is directly fitted using the svm()function, corresponding kernel type is specified as "linear" for feature selection purpose with linear kernel. The support vector and coefficient are stored in the generated model and can be

Given w in a SVM model $f(x) = (w^T v+b)$, w should be a n*1 vector where n is the number of individual dimensions, which equals to features if using a linear kernel. Write w as $(w_1, w_2, \ldots w_n)^T$, then for each $1 \leq k \leq n$, w_k represents the value of gradient of the decision hyperplane at k^{th} dimension. A dimension (feature) is more important if we have larger absolute value of gradient, and a dimension with its gradient equals to 0 suggests that this dimension, or the feature represented by the dimension has absolutely no impact on the classification result.

Using a linear kernel, each feature is represented by a dimension. Build a svm model with linear kernel, we can compute w of the model and choose the top 100 features with 100 greatest corresponding absolute value of w_k. The software and package used to build svm model is specified at the first paragraph above. For each k^{th} dimension, w_k is computed and its absolute value is taken. We can then sort the value of w_k in descending order and select 100 dimensions with top 100 absolute value of w_k.

The following is the corresponding index of the top 100 most important feature in ascending order (index as defined in problem 1).

```
329 318 90 87 302 31 734 735 618 385 560 361 75 585 351 37 155 741 475 705
725 124 714 641 92 595 348 328 476 777 38 799 537 471 283 534 359 609 431 19
338 544 676 331 633 121 498 101 611 34 528   7 341 363 47 605 297 83 108 761
233 737 515 494 589 192 325 343 629 548 52 186 453 419 698 134 21 326 671 259
266 622 561 156 195 769 508 506 395 762 308 664 86 751 773 628 316 694 72 322
```

Compare with the top 100 features of the random forest, there's only a 26 percent match, which suggest there must be some discrepancy between two models. However, the match will increase to 83% if extend the range to top 200 features. That might suggest the two methods generally agree on which feature is more important but not the specific ordering of them.

Given two quadratic kernels, the Inhomogeneous quadratic kernel will be better for feature selection. As discussed above, we want to compare the value of w_k for the dimension the feature corresponding to and larger value of w_k would suggest the feature is more important. Given (x,y) as example, the homogenous kernel maps to 3 dimension(ignore constant here): x^2, y^2, xy, and the Inhomogeneous kernel maps to 5 dimension(ignore constant here): x^2, y^2, xy, x, y. The dimensions provided by the homogenous kernel cannot provide the corresponding w_K value for individual feature as all dimensions are non-linear transformation of original features, while the dimensions provided by the inhomogeneous kernel can as there are dimensions that are produced by linear transformation from original feature. For similar reasons, it is hard to find correlation between features using polynomial kernels as Pearson Correlation only deals with linear relationship, yet the polynomial kernel includes non-linear transformations. The inhomogeneous kernel could possibly provide some information if only looking at dimensions produced by linear transformation.

P3:

A R package named "arules" using apriori algorithm is used to mine Association Rules. 18 are selected for control and 17 are selected for AD. The choice results in no frequent item sets are rule for min support = 0.1, which agrees with the actual result.

The following is the table summarizing the Number of and Association Rules and Frequent Item set discovered for case dataset:

Entry in form: # of rules(# of frequent item set)

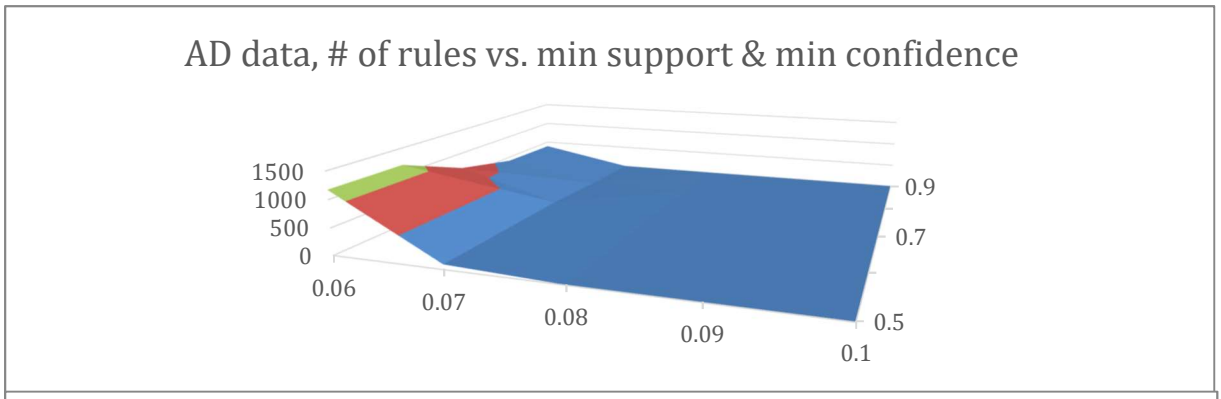
	Min support = 0.06	**Min support = 0.07**	**Min support = 0.08**	**Min support = 0.09**	**Min support = 0.10**
Confidence = 0.5	**5977(1836)**	**312(142)**	**8(4)**	**0**	**0**
Confidence = 0.6	**5977(1836)**	**312(142)**	**8(4)**	**0**	**0**
Confidence = 0.7	**5277(1486)**	**312(142)**	**8(4)**	**0**	**0**
Confidence= 0.8	**4548(1235)**	**150(61)**	**8(4)**	**0**	**0**
Confidence = 0.9	**3900(1171)**	**70(26)**	**0(0)**	**0**	**0**

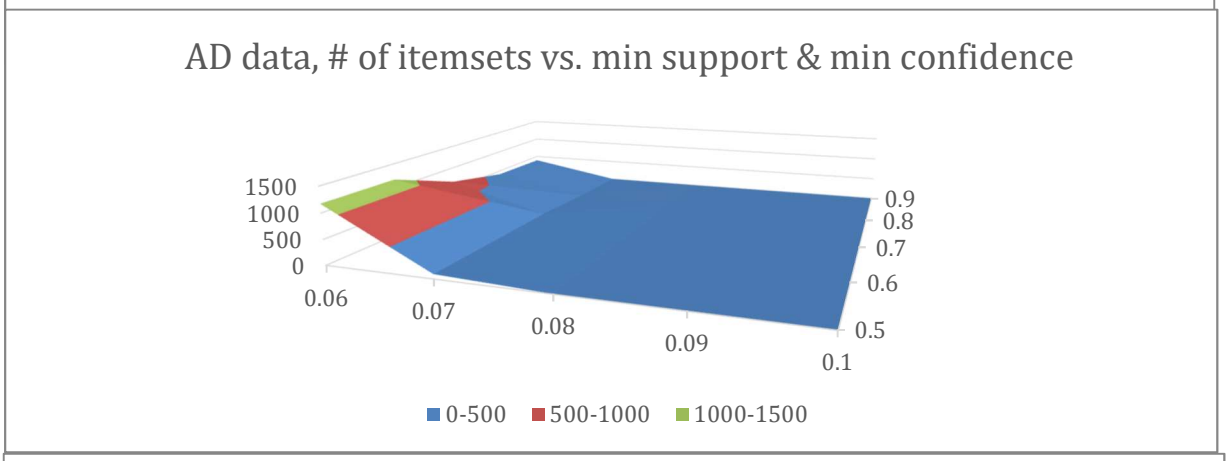
The following is the table summarizing the Number of and Association Rules and Frequent Item set discovered for ctrl dataset:

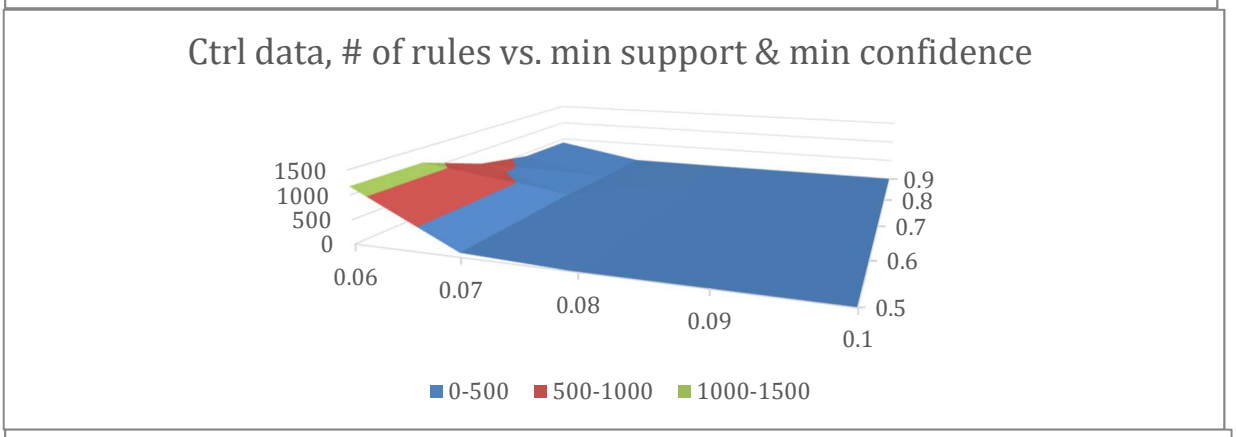
Entry in form: # of rules(# of frequent item set)

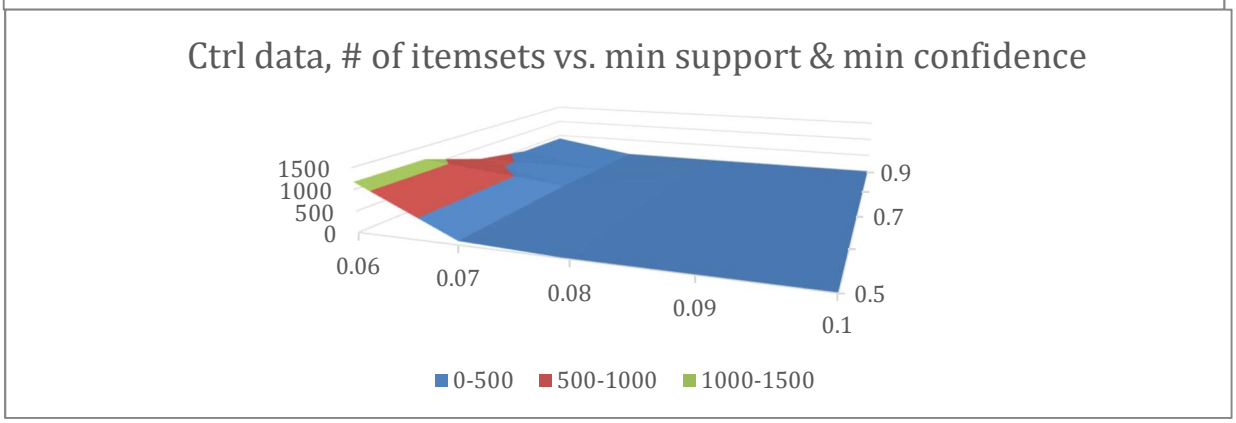
	Min support = 0.06	**Min support = 0.07**	**Min support = 0.08**	**Min support = 0.09**	**Min support = 0.10**
Confidence = 0.5	**2837(1173)**	**187(90)**	**8(4)**	**2(1)**	**0**
Confidence = 0.6	**2837(1173)**	**187(90)**	**8(4)**	**2(1)**	**0**
Confidence = 0.7	**1835(672)**	**187(90)**	**8(4)**	**2(1)**	**0**
Confidence= 0.8	**1299(414)**	**63(28)**	**8(4)**	**2(1)**	**0**
Confidence = 0.9	**903(390)**	**20(8)**	**2(1)**	**2(1)**	**0**

3D plot for both tables at next page .


AD data, # of rules vs. min support & min confidence
1500
1000
500
0
0.06
0.07
0.08
0.09
0.1
0.9
0.7
0.5


AD data, # of itemsets vs. min support & min confidence
1500
1000
500
0
0.06
0.07
0.08
0.09
0.1
0.9
0.8
0.7
0.6
0.5
0-500
500-1000
1000-1500


Ctrl data, # of rules vs. min support & min confidence
1500
1000
500
0
0.06
0.07
0.08
0.09
0.1
0.9
0.8
0.7
0.6
0.5
0-500
500-1000
1000-1500


Ctrl data, # of itemsets vs. min support & min confidence
1500
1000
500
0
0.06
0.07
0.08
0.09
0.1
0.9
0.7
0.5
0-500
500-1000
1000-1500

This Page Summarize Top 50 frequent set and their support for AD dataset.

items	support
{GI_10864030.S=1,GI_14249467.I=1}	0.08522727
{GI_10864030.S=1,GI_14916472.S=1}	0.08522727
{GI_12232430.S=1,GI_13376532.S=1}	0.08522727
{GI_13787194.S=1,GI_14165460.I=1}	0.08522727
{GI_10800147.S=1,GI_13376672.S=1}	0.07954545
{GI_10800147.S=1,GI_13376746.S=1}	0.07954545
{GI_10800147.S=1,GI_13787194.S=1,GI_14165460.I=1}	0.07954545
{GI_10800147.S=1,GI_13787194.S=1}	0.07954545
{GI_10800147.S=1,GI_14165460.I=1}	0.07954545
{GI_10800147.S=1,GI_14249251.S=1}	0.07954545
{GI_10800147.S=1,GI_14249309.S=1}	0.07954545
{GI_10864030.S=1,GI_13376528.S=1}	0.07954545
{GI_11056009.S=1,GI_12232456.S=1}	0.07954545
{GI_11056009.S=1,GI_13376477.S=1}	0.07954545
{GI_11321582.S=1,GI_14589955.S=1}	0.07954545
{GI_11545919.S=1,GI_14195633.S=1}	0.07954545
{GI_12232430.S=1,GI_13376532.S=1,GI_14042952.S=1}	0.07954545
{GI_12232430.S=1,GI_14042952.S=1}	0.07954545
{GI_12232456.S=1,GI_12597630.S=1}	0.07954545
{GI_12232456.S=1,GI_13376553.S=1}	0.07954545
{GI_12545396.I=1,GI_14165460.I=1}	0.07954545
{GI_13375741.S=1,GI_14150038.S=1}	0.07954545
{GI_13376532.S=1,GI_13540508.S=1}	0.07954545
{GI_13376532.S=1,GI_13787194.S=1}	0.07954545
{GI_13376532.S=1,GI_14042952.S=1}	0.07954545
{GI_13376672.S=1,GI_13376746.S=1}	0.07954545
{GI_13376672.S=1,GI_14249251.S=1}	0.07954545
{GI_13376746.S=1,GI_13787194.S=1}	0.07954545
{GI_13376746.S=1,GI_14249309.S=1}	0.07954545
{GI_13430873.S=1,GI_14165460.I=1}	0.07954545
{GI_13540508.S=1,GI_13787194.S=1}	0.07954545
{GI_14043068.S=1,GI_14456711.S=1}	0.07954545
{GI_14165460.I=1,GI_14249309.S=1}	0.07954545
{GI_14211842.S=1,GI_14589955.S=1}	0.07954545
{GI_14249251.S=1,GI_14249309.S=1}	0.07954545
{GI_14249467.I=1,GI_14916472.S=1}	0.07954545
{GI_14591913.S=1,GI_14591915.S=1}	0.07954545
{GI_10800147.S=1,GI_13376672.S=1,GI_13376746.S=1}	0.07386364
{GI_10800147.S=1,GI_13376672.S=1,GI_14249251.S=1}	0.07386364
{GI_10800147.S=1,GI_13376672.S=1,GI_14249309.S=1}	0.07386364
{GI_10800147.S=1,GI_13376746.S=1,GI_14249251.S=1}	0.07386364
{GI_10800147.S=1,GI_13376746.S=1,GI_14249309.S=1}	0.07386364
{GI_10800147.S=1,GI_14249251.S=1,GI_14249309.S=1}	0.07386364
{GI_10864030.S=1,GI_13376528.S=1,GI_14249467.I=1}	0.07386364
{GI_10864030.S=1,GI_13376528.S=1,GI_14916472.S=1}	0.07386364
{GI_10864030.S=1,GI_14249467.I=1,GI_14916472.S=1}	0.07386364
{GI_11056009.S=1,GI_12232456.S=1,GI_13376553.S=1}	0.07386364
{GI_12232430.S=1,GI_13376532.S=1,GI_13540508.S=1,GI_14042952.S=1}	0.07386364
{GI_12232430.S=1,GI_13376532.S=1,GI_13540508.S=1}	0.07386364
{GI_12232430.S=1,GI_13376532.S=1,GI_13787194.S=1}	0.07386364

There are 70 rules with confidence > 0.8 originated from the TOP 50 frequent itemset, and total 3900 rules with confidence >0.8. Top 10 rules originated from TOP 50 frequent itemset are report here.

	lhs	rhs	support	confidence	lift	count
[1]	{GI_12232430.S=1, GI_14042952.S=1}	=> {GI_13376532.S=1}	0.07954545	1.0000000	10.352941	14
[2]	{GI_13376532.S=1, GI_14042952.S=1}	=> {GI_12232430.S=1}	0.07954545	1.0000000	10.352941	14
[3]	{GI_12232430.S=1, GI_13376532.S=1}	=> {GI_14042952.S=1}	0.07954545	0.9333333	9.662745	14
[4]	{GI_12232430.S=1, GI_14042952.S=1}	=> {GI_13540508.S=1}	0.07386364	0.9285714	9.613445	13
[5]	{GI_13540508.S=1, GI_14042952.S=1}	=> {GI_12232430.S=1}	0.07386364	1.0000000	10.352941	13
[6]	{GI_12232430.S=1, GI_13540508.S=1}	=> {GI_14042952.S=1}	0.07386364	1.0000000	10.352941	13
[7]	{GI_13376532.S=1, GI_14042952.S=1}	=> {GI_13540508.S=1}	0.07386364	0.9285714	9.613445	13
[8]	{GI_13540508.S=1, GI_14042952.S=1}	=> {GI_13376532.S=1}	0.07386364	1.0000000	10.352941	13
[9]	{GI_13376532.S=1, GI_13540508.S=1}	=> {GI_14042952.S=1}	0.07386364	0.9285714	9.613445	13
[10]	{GI_11056009.S=1, GI_12232456.S=1}	=> {GI_13376553.S=1}	0.07386364	0.9285714	9.613445	13

This Page Summarize Top 50 frequent set and their support for ctrl dataset.

items	support
{GI_11056009.S=1,GI_13376477.S=1}	0.09042553
{GI_11641260.S=1,GI_13376659.S=1}	0.08510638
{GI_12025664.S=1,GI_14589955.S=1}	0.08510638
{GI_12232430.S=1,GI_12232456.S=1}	0.08510638
{GI_10800147.S=1,GI_14165460.I=1}	0.07978723
{GI_10864030.S=1,GI_11641260.S=1}	0.07978723
{GI_10864030.S=1,GI_13376659.S=1}	0.07978723
{GI_10864030.S=1,GI_13376746.S=1}	0.07978723
{GI_10864030.S=1,GI_14249467.I=1}	0.07978723
{GI_11545766.S=1,GI_13128859.S=1}	0.07978723
{GI_12232430.S=1,GI_13376477.S=1}	0.07978723
{GI_12232456.S=1,GI_13787194.S=1}	0.07978723
{GI_12232456.S=1,GI_14042952.S=1}	0.07978723
{GI_13112047.A=1,GI_13376321.S=1}	0.07978723
{GI_13376182.S=1,GI_13787194.S=1}	0.07978723
{GI_13376746.S=1,GI_14249467.I=1}	0.07978723
{GI_13376847.S=1,GI_13540514.S=1}	0.07978723
{GI_13787194.S=1,GI_14249309.S=1}	0.07978723
{GI_14043068.S=1,GI_14456711.S=1}	0.07978723
{GI_14165460.I=1,GI_14249309.S=1}	0.07978723
{GI_14249467.I=1,GI_14916472.S=1}	0.07978723
{GI_10800147.S=1,GI_13376182.S=1}	0.07446809
{GI_10800147.S=1,GI_14165460.I=1,GI_14249309.S=1}	0.07446809
{GI_10800147.S=1,GI_14249309.S=1}	0.07446809
{GI_10800147.S=1,GI_14600335.S=1}	0.07446809
{GI_10800413.S=1,GI_11321582.S=1}	0.07446809
{GI_10800413.S=1,GI_14589955.S=1}	0.07446809
{GI_10864008.I=1,GI_12965196.S=1}	0.07446809
{GI_10864008.I=1,GI_13112047.A=1}	0.07446809
{GI_10864008.I=1,GI_14591905.S=1}	0.07446809
{GI_10864030.S=1,GI_11641260.S=1,GI_13376659.S=1}	0.07446809
{GI_10864030.S=1,GI_13376746.S=1,GI_14249467.I=1}	0.07446809
{GI_10864030.S=1,GI_14249309.S=1}	0.07446809
{GI_10864030.S=1,GI_14600335.S=1}	0.07446809
{GI_10864030.S=1,GI_14916472.S=1}	0.07446809
{GI_10947033.S=1,GI_14165470.S=1}	0.07446809
{GI_11037056.S=1,GI_13376321.S=1}	0.07446809
{GI_11038670.S=1,GI_11545766.S=1}	0.07446809
{GI_11038670.S=1,GI_13128859.S=1}	0.07446809
{GI_11056009.S=1,GI_12232430.S=1,GI_13376477.S=1}	0.07446809
{GI_11056009.S=1,GI_12232430.S=1}	0.07446809
{GI_11096339.S=1,GI_13194196.S=1}	0.07446809
{GI_11321582.S=1,GI_12025664.S=1}	0.07446809
{GI_11321582.S=1,GI_13540514.S=1}	0.07446809
{GI_11321582.S=1,GI_14211842.S=1}	0.07446809
{GI_11321582.S=1,GI_14589955.S=1}	0.07446809
{GI_11968040.S=1,GI_13376551.S=1}	0.07446809
{GI_11968040.S=1,GI_13376672.S=1}	0.07446809
{GI_12025664.S=1,GI_13027654.A=1}	0.07446809
{GI_12232372.S=1,GI_14165257.S=1}	0.07446809

There are 20 rules with confidence > 0.8 originated from the TOP 50 frequent itemset, and total 903 rules with confidence >0.8. Top 10 rules originated from TOP 50 frequent itemset are report here.

	lhs	rhs	support	confidence	lift	count
[1]	{GI_13376477.S=1}	=> {GI_11056009.S=1}	0.09042553	0.9444444	9.864198	17
[2]	{GI_11056009.S=1}	=> {GI_13376477.S=1}	0.09042553	0.9444444	9.864198	17
[3]	{GI_12232430.S=1, GI_14042952.S=1}	=> {GI_12232456.S=1}	0.07446809	1.0000000	10.444444	14
[4]	{GI_12232456.S=1, GI_14042952.S=1}	=> {GI_12232430.S=1}	0.07446809	0.9333333	9.748148	14
[5]	{GI_12232430.S=1, GI_13376477.S=1}	=> {GI_11056009.S=1}	0.07446809	0.9333333	9.748148	14
[6]	{GI_11056009.S=1, GI_12232430.S=1}	=> {GI_13376477.S=1}	0.07446809	1.0000000	10.444444	14
[7]	{GI_13376182.S=1, GI_13787194.S=1}	=> {GI_14249309.S=1}	0.07446809	0.9333333	9.748148	14
[8]	{GI_13376182.S=1, GI_14249309.S=1}	=> {GI_13787194.S=1}	0.07446809	1.0000000	10.444444	14
[9]	{GI_13787194.S=1, GI_14249309.S=1}	=> {GI_13376182.S=1}	0.07446809	0.9333333	9.748148	14
[10]	{GI_10800147.S=1, GI_14165460.I=1}	=> {GI_14249309.S=1}	0.07446809	0.9333333	9.748148	14

For a rule A → B discovered in this problem, it suggest that if a certain person has high expression level of gene A, then it is very likely that he will also have high expression level of gene B. Therefore, to reduce the number of features, since the antecedent of a rule will lead to consequence of a rule, we could use the antecedent features of selected rules (which can be filtered by lift >1) to represent consequence of a rule to reduce the total number of features used in further analysis. We could start from all features or all frequent itemsets, remove those are consequence of a selected rule, and add those are antecedent of a rule, then return the result set as selected features.

The following is the pseudo code for the feature selection algorithm

```
Result = all frequent item sets
For each rule in Rules:
    If lift(rule) =<1
        Delete rule from Rules
    End if
End for
If Rules = NULL then terminate.
Else
    For r in result:
        If r = consequence of a rule
        Remove r from result
    End for
End if
```