Adem Malone
Tho Nguyen
Matthew Stroble

row #	What did you do?	How did you do it?	What were the Results?	Code Reference
1	Read through the SwaT Operational Manual to learn more about the water treatment plant system.	Read through the manual sections related to the dataset and taking note of what factors affect each column and how they combine/function together in the system.	Learned specific ranges and units for many of the continuous column data as well as relations between them and the pumps/motorized valves. Also learned how the system in general works.	
2	Finding the best partition of each column in dataset based on the information gain value .	Read the hw8_data.csv, and sort the table based on the selected attribute value. Ran the InfoGain function to find the partitions of the selected column and find the entropyLT and entropyGT of each partition and Using information gain formula with entropy before and after split to decide which partition has the highest information gain value.	The highest information gain value and the best partition of each column from [FIT101:AIT201].	Page 10
3	Convert the continuous data from[FIT101-AIT201] to the nominal data.	Ran the createRangeNominal to run through all values of the selected column and compare with the partition, if the value less than the partition, change it to “LT<partition>” the partition. Otherwise, change the value to “GT<partition>” the partition.	If the value less than the partition, it will change to “LT<value>”, otherwise,”GT<value>”.	Page 11
4	Finding the relation	Using the principal Component from Weka, it	From the dataset before changing to	Page 15

	between attributes in the dataset.	generates the correlation matrix of the. In the correlation matrix, each value in the column show how strong the relation between attribute which the 1 is the highest one.	nominal type, we can validate the relation between which attributes are affected by the is_attack attribute.	
5	Build the decision tree to find which attributes affect the is_attack attribute.	From R, I convert numeric to nominal and from Weka, we change the is_attack to nominal and use the J48 method to generate the tree, we used the information gain formula in R from to validate root of the tree and find it's validity since it has highest . we also use the principal component from Weka to generate the confusion matrix and find the highest value from the first row.	The correction Tree decision with the validation root which is checked by ran information gain function in R, the Correctly Classified Instance is 99.86% and Kappa give us 99.37%.	Page 14 Page 17
6	Find the areas where we can group attributes to multiple independent groups.	We used the Weka for running Kmean clustering and by using the Simple K Mean method for group them into independent groups to find the interesting data.	Generate the two group attributes based on the in_attack attribute.	Page 24 -28
7	Generate the Prediction model how much each attribute affects the is_attack attribute from the dataset.	From Weka, we use the original dataset before changing to nominal, I ran the Linear Regression to see how much attribute to affect to the is_attack attribute based on the slope function from weka. And I could validate whether the slope is accuracy or not based on the stats at bottom of the report. Addition, I ran the linear regression function in R with all attributes to get the graph, therefore, I can have a better view with the	Generated the Linear Regression function with the slope although it is not accurate.	Page 29 -32

		slope.		
8	Generate the prediction model from dataset for how much affects attribute together.	From Weka, I ran the linear regression for each attribute to find which attributes relate together and use the stats from bottom of the report to find the how accuracy of the slope function. From the report, I generate the graph to visualize it.	Generated the Linear Regression function with the slope and find the good relationship of P203 and 205.	Page 33
9	Generate the prediction whether is_attack happen or not and how each attribute affects the result.	In Weka, I convert the is_attack from numeric to nominal in R and uses the weaker learner Adaboost method for the prediction purpose. After it, I ran the AdaboostM1 from Weka to generate the prediction.	Have the report to predict which attributes affects is_attack attribute.	Page 34
10	Used Random Sampling to generate a ⅔ training dataset and used the remaining ⅓ of the dataset for a test dataset. Dataset used the provided numeric dataset using all columns.	Used the built in sample command in R ro randomise the row sellection and multiplied the number of rows by 0.66 and used floor rounding to generate the training dataset, then used that newly generated training dataset to subtract out of the original data to make the testing dataset using the remaining rows.	The method generated a nice ⅔ training dataset and a ⅓ testing test for numeric data.	
11	Used the SuperLearner package to build an ensemble out of the methods, Random Forest, Support Vector Machine, K-nearest neighbors,	Used SuperLearn to build an ensemble using the ensemble methods based on method Risk Estimate. Methods that are too high risk are dropped. SuperLearner by default applies crossfold validation during this process.	Generated a model based on the ensemble methods kept from Risk Estimate.	Code pg. 49

	bayes network general linear modeling, and xgboost on the decision attribute is_attack.			
12	Used the prediction model generated from SuplerLearner in table row #11 to compute a predicted probability result that moved any probabilistic result from >= 0.5 to 1 and <0.5 to 0 and stored it as a predicted result. Then Computed a confusion matrix to see its accuracy.	SuperLearner generates a prediction model based on probability. Used R to force the the data points that fell between 0 and 1 based on rounding. Used the predictedRresult and ran confusionMatrix against the testing dataset that was created in table row #10. Then used the function confusionMatrix between testing dataset and predictedResult to get metrics on accuracy and kappa.	The confusion matrix shows an accuracy of 0.9992 and a kappa of 0.9965. This allowed for validation of the methods used in Weka, J48, kmean, Linear Regression and adaboost.	Code Pg. 49-50 Results pg.47-48
13	Split up the original data set columns to match each subsection of the water treatment plant and built ⅔ training dataset and ⅓ testing data sets for all six subsystems	Built six new subsystem datasets from the original numeric data with each dataset being specific to each of the subsystems in the water treatment plant.	Now have subsystem specific datasets for each subsystem in the treatment plant.	Code pg. 62-67
14	For each subsystem data	Used the built in sample command in R ro randomise	Generated training datasets and testing	

	set, used random sampling to generate ⅔ training and ⅓ testing datasets	the row sellection and multiplied the number of rows by 0.66 and used floor rounding to generate the training dataset, then used that newly generated training dataset to subtract out of the original data to make the testing dataset using the remaining rows for each of the six subsystem datasets.	datasets from random sampling for each of the six subsystems of the water treatment plant.	
15	Used the SuperLearner package to build an ensemble out of the methods, Random Forest, Support Vector Machine, K-nearest neighbors, bayes network general linear modeling, and xgboost on the decision attribute is_attack for each of the six subsystems.	Used SuperLearn to build an ensemble using the ensemble methods based on method Risk Estimate for each of the six subsystems. Methods that are too high risk are dropped. SuperLearner by default applies crossfold validation during this process.	Generated a model based on the ensemble methods kept from Risk Estimate for each subsystem.	Code pg. 62-67 Results Pg. 51-62
16	Used Prediction model from SuplerLearner in table row #11 to compute a predicted probability result that moved any probabilistic result from >= 0.5 to 1 and	SuperLearner generates a prediction model based on probability. Used R to force the the data points that fell between 0 and 1 based on rounding for each subsystem. Used the predictedRresult and ran confusionMatrix against the testing dataset that was created for each subsystem. Then used the function confusionMatrix	Gave a way to validate the results from running Bayesian Networks K2 on each subsection.	Code pg. 62-67 Results pg. 51-62

	<0.5 to 0 and stored it as a predicted result for each subsystem. Then Computed a confusion on each subsystem, except P6.	between testing dataset and predictedResult for each subsystem to get metrics on accuracy and kappa that are specific to each subsystem, except P6.		
17	Used Weka to generate a Bayesian Network using K2	Used Weka Explorer to load nominal dataset and used the Classify tab to generate a Bayesian Network using 3 max parents.	Weka correctly classified 91.73% of instances and generated a Kappa statistic 0.6972.	Weka pg. 72-74 Results pg. 68-71
18	Used Weka to generate another Bayesian Network using K2 with 37 parents.	Used Weka Explorer to load nominal dataset and used the Classify tab to generate a Bayesian Network using 36 max parents.	Weka correctly classified 97.07% of instances and a Kappa of 0.8665.	Weka 79-81 Results pg. 75-78
19	Used Weka to generate a Bayesian Network using K2 with 3 parents on every subsystem dataset.	Used Weka Explorer to load each subsystems nominal dataset and used the Classify tab to generate a Bayesian Network using 3 max parents for each subsystem..	Subsystem P1 and P6 accuracy are much lower and both kappa are less than 0.50. Subsystem P2 to P5 have decent results below.	Weka and results 82-102
20	Ran apriori on dataset to generate associate rules.	Launched Weka Explorer and put in a dataset with all the continuous data as LTs(Less Than) and GTs(Greater Than). Then filtered dataset with “NumericToNominal” and run apriori with a rule count of 1000. Run apriori again but with classIndex set to is_attack.	Generated 1000 associate rules that don’t have limitations and generated another 1000 rules but having is_attack included in all of them.	Pg. 103-105

21	Convert Weka apriori rules to a cleaner, more readable list of rules.	Ran a python program called “ConvertRawRules.py” that takes the Weka results and reformates it.	A clean and consistent rule list that the “RuleY_E_Generator.py” program can recognize and use it.	Pg. 106-107
22	Calculated the Y&E Tables for the cleaned up version of the associate rules from apriori.	Ran a python program called “RuleY_E_Generator.py” which takes the list of the reformatted rules and calculates Y_1 and E_1, as well as the Y_2 and E_2 values.	A text document that contained each Y&E Table of every rule in an array format.	Pg. 108-113
23	Calculate R and R prime values of each rule from apriori.	Ran a python program called “UcfCalculator.py” which used the Y&E tables generated and the Ucf function in an equation to produce R and R prime values.	A text document that contained the calculated R and R prime values of each rule in an array format.	Pg. 114-117
24	Used the formated associate rules from Weka apriori and removed parts of the rule that would result in less error to get a final set of rules.	Ran a python program called “RuleCutting.py” that used the formated rules as well as the text document with the R and R prime values.	A final list of all the rules generated from weka formatted as well as each rule being reduced if it was possible.	Pg. 118-121 Additional Rules from Pg. 122-129

Adem Malone
Tho Nguyen
Matthew Stroble

Essay/Summary:

By running the information gain function, we can see the AIT402 attribute has the highest information gain value (0.2105723925). This means AIT402 is the most affected attribute when the water system is attacked. From J48 method, it generates the tree's decision with the accuracy is ninety nine percent and the kappa statistic is ninety eight percent, which means the tree's decision is very accurate, and it showed the root of the tree is AIT402 as well. To make sure the result from J48 is correct, we used the Addaboost method. Addaboost will predict which attribute will be attacked in the water system, and found that the AIT402 has the highest weight. Using the J48 method, the water system is safe when AIT402 unit is greater than 6.664958. This means when number of chlorine residual and chloramines is as low as possible or nothing,The water system is completely safe. Therefore, the number of of chlorine residual and chloramines is one of the main factors for whether the water system is attacked or not. This makes sense due to the fact that at subsystem 5, it is crucial that the RO system isn't given water with a high level of chlorine. The system itself even has a backup for if, in subsystem 4, the UV Chlorine Destruction Unit fails where the ORP analyser uses sodium bisulphite($NaHSO_3$) to get rid of chlorine residue. It would make sense for an attack to target that location to allow higher concentration of chlorine to flow into the RO system to put it at risk. From the tree's decision, we found the list of attributes : AIT402, LIT301,AIT202, LIT401, AIT203,P301,AIT504,LIT301, DPIT301, AIT201, PIT502 and AIT501. From the tree's decision, we can also see the Raw water supply & storage system and Chemical system don't show up in the tree. Therefore, the water system is safe from beginning to MV201 valve.

We decided to look into each of the subsystems of the water treatment plant as an isolated system. When looking at Supply and Storage (P1) as an isolated system we were able to determine that an attack was not taking place. This was indicated by running the Bayesian Network K2 method in Weka which gave an accuracy of 87.1266%, but reported a kappa of 0. We validated the findings from Weka by using the ensemble methods in SuperLearner. With SuperLearner we calculated a confusion matrix (pg. 54), checked the Risk Estimate (pg. 53) of each method in the ensemble and calculated the Area Under the Curve (pg 55). The same method was used on the remaining five subsystems and we found that in the Pretreatment (P2) was quite easy to detect an attack. We were able to successfully detect attack in Ultrafiltration and backwash (P3), De-Chlorination System (P4), and Reverse Osmosis (P5). We were unable to run a confusion matrix on Permeate Transfer, Clearing and Back-wash (P6) but we are confident that no attack is taking place due to the Risk Estimate reported from each ensemble method from SuperLearner. The Area under the curves report an average of 50% which tell is its as good as random guessing. Weka Reported a kappa of 0, and an accuracy of 87.15% but also reported Relative absolute error of 98.71%.

Out of all the data mining methods that we run on this dataset we found it to be quite surprising that Linear Regression testing completely failed. The graphs that came out of R were difficult to interpret and the most bizzare graph of all was the Scale-Location graph. The only graph that made any sense was the relation between P203 and P205. This graph gave a correlation coefficient of 1 and the graph clearly shows a defined slope. Not only that, but Apriori wasn't as useful as we had hoped initially. The rules generated were indeed very accurate, but they seemed too random or specific to gain any meaningful data from, even after cutting parts of the rules out. Not only that, but getting rules related to is_attack=1 was especially difficult since Weka preferred to generate rules that had more data to go off of. We tried giving it a dataset that had all the is_attack=0 removed but the results weren't as accurate as we wanted. Overall we included the list of rules generated, but didn't find many of them to be as useful as the other methods we did.

```
main<-function(){
  hw8<-read.csv(file="~/CS5402/HW8/hw8_data.csv")
  columnsName<-colnames(hw8)
  #16 which column start from FIT101
  for(i in 15:NROW(columnsName)){
    print(columnsName[i])
    #hw8[c(columnsName[i])]
    "Sort the datase based on the values of ith column
    NOTE: this step must be done before calculte the infomation Gain"
    hw8<-hw8[order(hw8[c(columnsName[[i]])]),]
    "Call infoGain funtion for calculate the information gain"
    partitionValue<-InfoGain(hw8$is_attack,hw8[[c(columnsName[i])]])

hw8[c(columnsName[[i]])]<-createRangeNominal(hw8[c(columnsName[[i]])],parti
tionValue)

  }
  write.csv(hw8,file="~/CS5402/HW8/testFit1.csv")
}
```

```
convertToNominal <- function(numericCol){
  if(length(unique(numericCol)) ==2){
    nominalCol<-ifelse(numericCol==0,'no','yes')
    numericCol<-nominalCol
    return(numericCol)
  }

}
createRangeNominal<-function(dataset,partitionValue){
  #only change data value to nominal when the variance has big value which
It
  # is continous data

dataset<-ifelse(dataset<partitionValue[1],paste("LT",toString(partitionValu
e),sep=""),
                  paste("GT",toString(partitionValue),sep = ""))


  return(dataset)
```

```
}



#Delete last and first value since there entropy is 0
#averageList<-averageList[2:(length(averageList)-1)]

InfoGain <- function(decisionAttribute,continousAttribute){
  decisionAttribute<-convertToNominal(decisionAttribute)
  unique(decisionAttribute)
  #Finding unique values
  distint_value<-unique(continousAttribute)
  #Get the list of partitions
  averageList <-vector(mode="list",length = length(distint_value))
  for(i in 1 : length(distint_value)-1) {
    averageList[i]<-((distint_value[i]+distint_value[i+1])/2)

  }
  #Delete null value in list
  averageList<-Filter(Negate(is.null),averageList)

    " Build the entropy function"

  maxValue<-c()  # value holds the best partition
  informationGain<-c() # list of best parttion
    for(j in 1: (length(averageList))){
      LT<-c()
      GT<-c()

      for( i in 1 : (length(decisionAttribute))){
        if(continousAttribute[[i]]<= averageList[[j]]){
          if( decisionAttribute[[i]]=='yes'){
            LT[length(LT)+1]<-'yes'
          }else{
            LT[length(LT)+1]<-'no'
          }
        }else{
          if( decisionAttribute[[i]]=='yes'){
            GT[length(GT)+1]<-'yes'
          }else{
```

```
            GT[length(GT)+1]<-'no'
          }
        }
      }

      entropyLT<-(-sum( (table(LT)/length(LT))*
log2((table(LT)/length(LT))) ))
      entropyGT<-(-sum( (table(GT)/length(GT))*
log2((table(GT)/length(GT))) ))
      entropyBefore<-(-sum(
(table(decisionAttribute)/length(decisionAttribute)) *

log2((table(decisionAttribute)/length(decisionAttribute))) ))
      #calculate Entropy after Split
      totalInstance<-length(LT)+length(GT)
      entropyAfterSplit<-(length(LT)/totalInstance * entropyLT) +
                           (length(GT)/totalInstance * entropyGT)

        informationGain[length(informationGain)+1] <-
          (entropyBefore - entropyAfterSplit)

    }
  #infoGainList$X2<-informationGain
  #print(infoGainList)
  "Delete last and first value since of the average list since the
entropies are 0"
  averageList<-averageList[2:(length(averageList)-1)]
  informationGain<-informationGain[2:(length(informationGain)-1)]
  "for(k in 1:length(informationGain)){
    print(averageList[[k]])
    print(informationGain[[k]])
  }"
  infoGainList<-data.frame(matrix(ncol = 2,nrow = length(averageList)))
  infoGainList$X1<-averageList
  infoGainList$X2<-informationGain

  #add max value to first index of maxValue list
  #maxValue[length(maxValue)+1]<-max(infoGainList$X2)
  for(i in 1:length(infoGainList$X2)){
    if(infoGainList$X2[i]==max(infoGainList$X2)){
```

```
      #append the partition value to maxValue list
      maxValue[length(maxValue)+1]<-infoGainList$X1[[i]]
    }

  }
  #print(length(maxValue))
  #print(maxValue)
  return(maxValue)
}

main<-function(){
  hw8<-read.csv(file="~/CS5402/HW8/hw8_data.csv")
  columnsName<-colnames(hw8)
  #16 which column start from FIT101
  for(i in 15:NROW(columnsName)){
    print(columnsName[i])
    #hw8[c(columnsName[i])]
    "Sort the datase based on the values of ith column
    NOTE: this step must be done before calculte the infomation Gain"
    hw8<-hw8[order(hw8[c(columnsName[[i]])]),]
    "Call infoGain funtion for calculate the information gain"
    partitionValue<-InfoGain(hw8$is_attack,hw8[[c(columnsName[i])]])

hw8[c(columnsName[[i]])]<-createRangeNominal(hw8[c(columnsName[[i]])],parti
tionValue)

  }
  write.csv(hw8,file="~/CS5402/HW8/testFit1.csv")
}
```

```
library("FSelector")
y<-read.csv(file="~/CS5402/HW8/numericDataset/hw8_data.csv")
is_attact<-y$is_attack
is_attact<-ifelse(is_attact==0,'no','yes')
y$is_attack<-is_attact
View(information.gain(is_attack~.,y))
write.csv(y,file="~/CS5402/HW8/hw8ConvertIsAttackToNominal_data.csv")
```

Attribute Evaluator

Choose **PrincipalComponents** -R 0.95 -A 5

Search Method

Choose **Ranker** -T -1.7976931348623157E308 -N -1

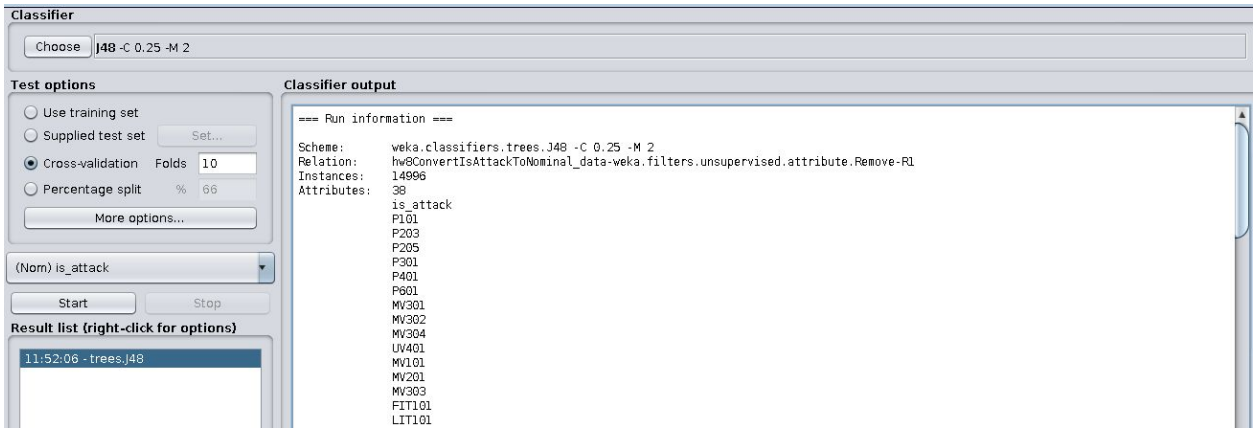
Correlation matrix

1	0.99	0.99	0.21	0.01	0.16	0.02	0.15	-0.07	0.1	0.05	0.98	-0.03	0.05	-0.71	0.62	0.28	0.99	0.23	0.22			
0.99	1	1	0.2	0.01	0.16	0.02	0.14	-0.07	0.1	0.06	0.98	-0.03	0.06	-0.72	0.64	0.29	0.99	0.22	0.21			
0.99	1	1	0.2	0.01	0.16	0.02	0.14	-0.07	0.1	0.06	0.98	-0.03	0.06	-0.72	0.64	0.29	0.99	0.22	0.21			
0.21	0.2	0.2	1	-0.01	0.13	0.02	0.84	0.09	-0.13	-0.39	0.21	-0.05	-0.43	0.13	-0.1	-0.3	0.2	0.98	0.99			
0.01	0.01	0.01	-0.01	1	0	-0	0	-0.04	0.09	0	0.01	0	0.01	-0.01	0.01	0	0.01	-0.01	-0.01			
0.16	0.16	0.16	0.13	0	1	0	0.12	-0.01	0.02	-0.05	0.16	-0.01	-0.06	-0.09	0.07	0.18	0.16	0.13	0.13			
0.02	0.02	0.02	0.02	-0	0	1	0.02	0	-0	0.01	0.02	0.05	0.01	-0.02	0.05	0.03	0.02	0.02	0.02			
0.15	0.14	0.14	0.84	0	0.12	0.02	1	-0.15	-0.05	-0.41	0.14	-0.04	-0.45	0.19	-0.25	-0.38	0.14	0.87	0.83			
-0.07	-0.07	-0.07	0.09	-0.04	-0.01	0	-0.15	1	-0.11	-0.03	-0.06	0.16	-0.04	0.07	-0.1	-0.06	-0.07	-0.02	-0.07			
0.1	0.1	0.1	-0.13	0.09	0.02	-0	-0.05	-0.11	1	0.05	0.1	0.01	0.06	-0.1	0.06	0.04	0.1	-0.09	-0.13			
0.05	0.06	0.06	-0.39	0	-0.05	0.01	-0.41	-0.03	0.05	1	0.04	-0.02	0.93	-0.41	0.31	0.48	0.06	-0.39	-0.39			
0.98	0.98	0.98	0.21	0.01	0.16	0.02	0.14	-0.06	0.1	0.04	1	-0.03	0.05	-0.7	0.61	0.27	0.97	0.22	0.21			
-0.03	-0.03	-0.03	-0.05	0	-0.01	0.05	-0.04	0.16	0.01	-0.02	-0.03	1	-0.02	0.03	-0.1	-0.06	-0.03	-0.04	-0.02			
0.05	0.06	0.06	-0.43	0.01	-0.06	0.01	-0.45	-0.04	0.06	0.93	0.05	-0.02	1	-0.44	0.33	0.52	0.07	-0.43	-0.43			
-0.71	-0.72	-0.72	0.13	-0.01	-0.09	-0.02	0.19	0.07	-0.1	-0.41	-0.7	0.03	-0.44	1	-0.6	-0.65	-0.73	0.12	0.13			
0.62	0.64	0.64	-0.1	0.01	0.07	0.05	-0.25	-0.1	0.06	0.31	0.61	-0.1	0.33	-0.6	1	0.58	0.64	-0.09	-0.1			
0.28	0.29	0.29	-0.3	0	0.18	0.03	-0.38	-0.06	0.04	0.48	0.27	-0.06	0.52	-0.65	0.58	1	0.29	-0.29	-0.29			
0.99	0.99	0.99	0.2	0.01	0.16	0.02	0.14	-0.07	0.1	0.06	0.97	-0.03	0.07	-0.73	0.64	0.29	1	0.22	0.21			
0.23	0.22	0.22	0.98	-0.01	0.13	0.02	0.87	-0.02	-0.09	-0.39	0.22	-0.04	-0.43	0.12	-0.09	-0.29	0.22	1	0.99			
0.22	0.21	0.21	0.99	-0.01	0.13	0.02	0.83	0.07	-0.13	-0.39	0.21	-0.02	-0.43	0.13	-0.1	-0.29	0.21	0.99	1			
-0.59	-0.57	-0.57	-0.58	-0.01	-0.13	0.02	-0.49	0.02	-0.16	0.29	-0.57	-0.03	0.32	0.18	-0.23	-0.22	-0.57	-0.6	-0.59			
0.12	0.12	0.12	-0.19	0.01	-0.07	0.02	-0.26	-0.02	0.08	0.08	0.12	-0.03	0.08	-0.16	0.23	-0.25	0.13	-0.19	-0.2			
0.1	0.1	0.1	-0.12	0.09	0.02	-0	-0.03	-0.11	0.98	0.05	0.1	0.01	0.06	-0.11	0.04	0.05	0.1	-0.08	-0.12			
0.71	0.71	0.71	-0.1	0.02	0.19	0.02	-0.16	-0.11	0.18	0.28	0.69	-0.04	0.31	-0.79	0.75	0.62	0.71	-0.07	-0.09			
-0.1	-0.1	-0.1	0.18	-0.01	0.08	-0.02	0.26	0.02	-0.08	-0.09	-0.1	0.03	-0.1	0.15	-0.25	0.21	-0.11	0.18	0.19			

0.2	0.2	-0.21	0.01	-0.03	0.02	-0.28	-0.03	0.06	0.04	0.2	-0.03	0.04	-0.19	0.24	-0.21	0.2	-0.2	-0.21				
-0.07	-0.06	-0.06	0.05	-0.01	0.05	-0.01	0.08	-0.02	-0.03	-0.25	-0.06	-0.01	-0.29	0.13	-0.04	-0.19	-0.07	0.06	0.05			
0.11	0.11	0.11	-0.11	0	-0.02	0	-0.13	-0.01	0.02	-0.01	0.11	-0	-0.01	-0.09	0.02	-0.3	0.11	-0.11	-0.11			
0.08	0.08	0.08	-0.09	0.06	0.02	-0	-0.04	-0.03	0.73	0.04	0.07	0.01	0.05	-0.08	0.01	0.03	0.08	-0.08	-0.09			
0.13	0.13	0.13	-0.12	0.04	0.05	-0.02	-0.01	-0.05	0.56	0.07	0.12	0.02	0.07	-0.17	-0.04	0.03	0.13	-0.1	-0.12			
-0.06	-0.06	-0.06	0.08	-0.06	-0.02	0.01	0.02	0.03	-0.72	-0.03	-0.06	-0.01	-0.03	0.06	0.02	-0.03	-0.06	0.06	0.08			
-0.09	-0.09	-0.09	0.1	-0.06	-0.03	0.01	0.04	0.04	-0.72	-0.05	-0.09	-0.01	-0.05	0.1	-0.02	-0.04	-0.09	0.08	0.1			
-0.04	-0.04	-0.04	0.06	-0.06	-0.02	0.01	-0.01	0.03	-0.71	-0.02	-0.04	-0.02	-0.02	0.04	0.06	-0.02	-0.04	0.04	0.06			
-0.08	-0.08	-0.08	0.1	-0.06	-0.01	0	0.06	0.03	-0.74	-0.04	-0.07	-0.01	-0.04	0.08	-0.03	-0.01	-0.08	0.09	0.11			
-0.05	-0.05	-0.05	0.07	-0.06	-0.02	0.01	0.01	0.03	-0.72	-0.02	-0.04	-0.01	-0.02	0.04	0.04	-0.02	-0.05	0.05	0.07			
-0.02	-0.02	-0.02	-0.03	0	-0	-0.39	-0.03	-0.06	0	-0.01	-0.02	0.42	-0.01	0.02	-0.07	-0.04	-0.02	-0.03	-0.02			
-0.05	-0.05	-0.05	0.1	0	-0.15	0.03	-0.04	-0.02	0.06	0.05	-0.05	0.04	0.15	0.38	-0.02	-0.05	0.1	0.09				

0.22	-0.59	0.12	0.1	0.71	-0.1	0.2	-0.07	0.11	0.08	0.13	-0.06	-0.09	-0.04	-0.08	-0.05	-0.02	-0.05
0.21	-0.57	0.12	0.1	0.71	-0.1	0.2	-0.06	0.11	0.08	0.13	-0.06	-0.09	-0.04	-0.08	-0.05	-0.02	-0.05
0.21	-0.57	0.12	0.1	0.71	-0.1	0.2	-0.06	0.11	0.08	0.13	-0.06	-0.09	-0.04	-0.08	-0.05	-0.02	-0.05
0.99	-0.58	-0.19	-0.12	-0.1	0.18	-0.21	0.05	-0.11	-0.09	-0.12	0.08	0.1	0.06	0.1	0.07	-0.03	0.1
-0.01	-0.01	0.01	0.09	0.02	-0.01	0.01	-0.01	0	0.06	0.04	-0.06	-0.06	-0.06	-0.06	-0.06	0	0
0.13	-0.13	-0.07	0.02	0.19	0.08	-0.03	0.05	-0.02	0.02	0.05	-0.02	-0.03	-0.02	-0.01	-0.02	-0	-0.15
0.02	0.02	0.02	-0	0.02	-0.02	0.02	-0.01	0	-0	-0.02	0.01	0.01	0.01	0	0.01	-0.39	0.03
0.83	-0.49	-0.26	-0.03	-0.16	0.26	-0.28	0.08	-0.13	-0.04	-0.01	0.02	0.04	-0.01	0.06	0.01	-0.03	-0.04
0.07	0.02	-0.02	-0.11	-0.11	0.02	-0.03	-0.02	-0.01	-0.03	-0.05	0.03	0.04	0.03	0.03	0.03	-0.06	-0.02
-0.13	-0.16	0.08	0.98	0.18	-0.08	0.06	-0.03	0.02	0.73	0.56	-0.72	-0.72	-0.71	-0.74	-0.72	0	0.06
-0.39	0.29	0.08	0.05	0.28	-0.09	0.04	-0.25	-0.01	0.04	0.07	-0.03	-0.05	-0.02	-0.04	-0.02	-0.01	0.05
0.21	-0.57	0.12	0.1	0.69	-0.1	0.2	-0.06	0.11	0.07	0.12	-0.06	-0.09	-0.04	-0.07	-0.04	-0.02	-0.05
-0.02	-0.03	-0.03	0.01	-0.04	0.03	-0.03	-0.01	-0	0.01	0.02	-0.01	-0.01	-0.02	-0.01	-0.01	0.42	-0.05
-0.43	0.32	0.08	0.06	0.31	-0.1	0.04	-0.29	-0.01	0.05	0.07	-0.03	-0.05	-0.02	-0.04	-0.02	-0.01	0.04
0.13	0.18	-0.16	-0.11	-0.79	0.15	-0.19	0.13	-0.09	-0.08	-0.17	0.06	0.1	0.04	0.08	0.04	0.02	0.15
-0.1	-0.23	0.23	0.04	0.75	-0.25	0.24	-0.04	0.02	0.01	-0.04	0.02	-0.02	0.06	-0.03	0.04	-0.07	0.38
-0.29	0.22	-0.25	0.05	0.62	0.21	-0.21	-0.19	-0.3	0.03	0.03	-0.03	-0.04	-0.02	-0.01	-0.02	-0.04	-0.02
0.21	-0.57	0.13	0.1	0.71	-0.11	0.2	-0.07	0.11	0.08	0.13	-0.06	-0.09	-0.04	-0.08	-0.05	-0.02	-0.05
0.99	-0.6	-0.19	-0.08	-0.07	0.18	-0.2	0.06	-0.11	-0.08	-0.1	0.06	0.08	0.04	0.09	0.05	-0.03	0.1
1	-0.59	-0.2	-0.12	-0.09	0.19	-0.21	0.05	-0.11	-0.09	-0.12	0.08	0.1	0.06	0.11	0.07	-0.02	0.09
-0.59	1	-0.19	-0.15	-0.4	0.16	-0.23	-0.05	-0.06	-0.16	-0.05	0.14	0.16	0.13	0.18	0.14	-0.02	-0.27
-0.2	-0.19	1	0.02	0.21	-0.99	0.79	0.23	0.61	0	-0.05	0.06	0.01	0.11	-0.06	0.06	-0.02	0.38
-0.12	-0.15	0.02	1	0.17	-0.03	0.01	-0.05	-0.02	0.78	0.61	-0.78	-0.77	-0.77	-0.78	-0.77	0	0.02
-0.09	-0.4	0.21	0.17	1	-0.2	0.33	-0.13	0.1	0.12	0.12	-0.09	-0.14	-0.05	-0.13	-0.07	-0.03	0.1
0.19	0.16	-0.99	-0.03	-0.2	1	-0.71	-0.25	-0.54	0	0.05	-0.07	-0.02	-0.11	0.06	-0.07	0.03	-0.41
-0.21	-0.23	0.79	0.01	0.33	-0.71	1	0.08	0.71	-0.01	-0.09	0.06	0	0.11	-0.06	0.07	-0.02	0.32

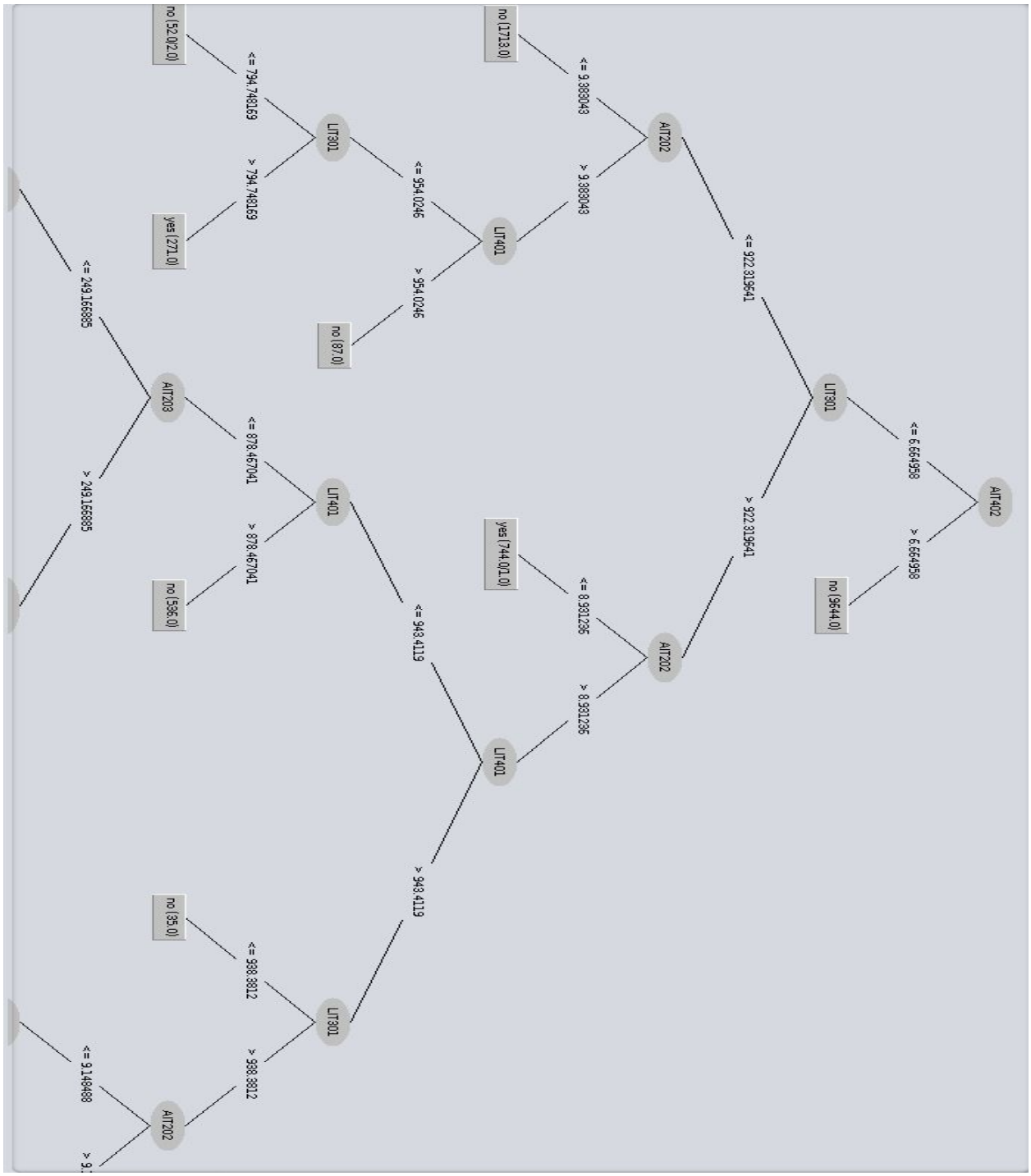
0.2	-0.2	-0.21	-0.23	0.79	0.01	0.33	-0.71	1	0.08	0.71	-0.01	-0.09	0.06	0	0.11	-0.06	0.07	-0.02	0.32
-0.07	0.06	0.05	-0.05	0.23	-0.05	-0.13	-0.25	0.08	1	0.1	-0.07	-0.11	0.07	0.07	0.08	0.05	0.07	0.01	0.02
0.11	-0.11	-0.11	-0.06	0.61	-0.02	0.1	-0.54	0.71	0.1	1	-0.03	0.01	0.06	0.02	0.09	-0.01	0.07	-0	-0.02
0.08	-0.08	-0.09	-0.16	0	0.78	0.12	0	-0.01	-0.07	-0.03	1	0.81	-1	-0.99	-0.99	-1	-0.99	0.01	0.08
0.13	-0.1	-0.12	-0.05	-0.05	0.61	0.12	0.05	-0.09	-0.11	0.01	0.81	1	-0.81	-0.82	-0.82	-0.79	-0.82	0.02	-0.2
-0.06	0.06	0.08	0.14	0.06	-0.78	-0.09	-0.07	0.06	0.07	0.06	-1	-0.81	1	0.99	1	0.99	1	-0.01	-0.04
-0.09	0.08	0.1	0.16	0.01	-0.77	-0.14	-0.02	0	0.07	0.02	-0.99	-0.82	0.99	1	0.98	0.98	0.99	-0.01	-0.05
-0.04	0.04	0.06	0.13	0.11	-0.77	-0.05	-0.11	0.11	0.08	0.09	-0.99	-0.82	1	0.98	1	0.98	1	-0.01	-0.01
-0.08	0.09	0.11	0.18	-0.06	-0.78	-0.13	0.06	-0.06	0.05	-0.01	-1	-0.79	0.99	0.98	0.98	1	0.99	-0	-0.12
-0.05	0.05	0.07	0.14	0.06	-0.77	-0.07	-0.07	0.07	0.07	0.07	-0.99	-0.82	1	0.99	1	0.99	1	-0.01	-0.04
-0.02	-0.03	-0.02	-0.02	-0.02	0	-0.03	0.03	-0.02	0.01	-0	0.01	0.02	-0.01	-0.01	-0.01	-0	-0.01	1	-0.04
-0.05	0.1	0.09	-0.27	0.38	0.02	0.1	-0.41	0.32	0.02	-0.02	0.08	-0.2	-0.04	-0.05	-0.01	-0.12	-0.04	-0.04	1




```
J48 pruned tree
------------------

AIT402 <= 6.664958
|   LIT301 <= 922.319641
|   |   AIT202 <= 9.383043: no (1713.0)
|   |   AIT202 > 9.383043
|   |   |   LIT401 <= 954.0246
|   |   |   |   LIT301 <= 794.748169: no (52.0/2.0)
|   |   |   |   LIT301 > 794.748169: yes (271.0)
|   |   |   LIT401 > 954.0246: no (87.0)
|   LIT301 > 922.319641
|   |   AIT202 <= 8.931236: yes (744.0/1.0)
|   |   AIT202 > 8.931236
|   |   |   LIT401 <= 943.4119
|   |   |   |   LIT401 <= 878.467041
|   |   |   |   |   AIT203 <= 249.166885
|   |   |   |   |   |   LIT401 <= 848.8977: no (619.0)
|   |   |   |   |   |   LIT401 > 848.8977
|   |   |   |   |   |   |   P301 <= 1: no (201.0)
|   |   |   |   |   |   |   P301 > 1
|   |   |   |   |   |   |   |   AIT203 <= 240.938217
|   |   |   |   |   |   |   |   |   AIT202 <= 8.995642: no (58.0)
|   |   |   |   |   |   |   |   |   AIT202 > 8.995642
|   |   |   |   |   |   |   |   |   |   AIT504 <= 24.839785: yes (209.0/1.0)
|   |   |   |   |   |   |   |   |   |   AIT504 > 24.839785
|   |   |   |   |   |   |   |   |   |   |   LIT401 <= 865.431946: yes (17.0)
|   |   |   |   |   |   |   |   |   |   |   LIT401 > 865.431946
|   |   |   |   |   |   |   |   |   |   |   |   LIT301 <= 1033.02844: no (21.0)
|   |   |   |   |   |   |   |   |   |   |   |   LIT301 > 1033.02844: yes (2.0)
|   |   |   |   |   |   |   |   |   AIT203 > 240.938217: no (57.0)
|   |   |   |   |   AIT203 > 249.166885
|   |   |   |   |   |   DPIT301 <= 1.226071: no (35.0)
|   |   |   |   |   |   DPIT301 > 1.226071
|   |   |   |   |   |   |   AIT201 <= 128.332474: yes (414.0/3.0)
|   |   |   |   |   |   |   AIT201 > 128.332474
|   |   |   |   |   |   |   |   PIT502 <= 2.883414: no (3.0)
|   |   |   |   |   |   |   |   PIT502 > 2.883414: yes (40.0)
|   |   |   |   LIT401 > 878.467041: no (536.0)
|   |   |   LIT401 > 943.4119
|   |   |   |   LIT301 <= 938.3812: no (35.0)
|   |   |   |   LIT301 > 938.3812
|   |   |   |   |   AIT202 <= 9.148488
|   |   |   |   |   |   AIT501 <= 7.75109: no (6.0)
|   |   |   |   |   |   AIT501 > 7.75109: yes (6.0)
|   |   |   |   |   AIT202 > 9.148488: yes (226.0)
AIT402 > 6.664958: no (9644.0)

Number of Leaves  :     23

Size of the tree :      45


Time taken to build model: 0.72 seconds
```

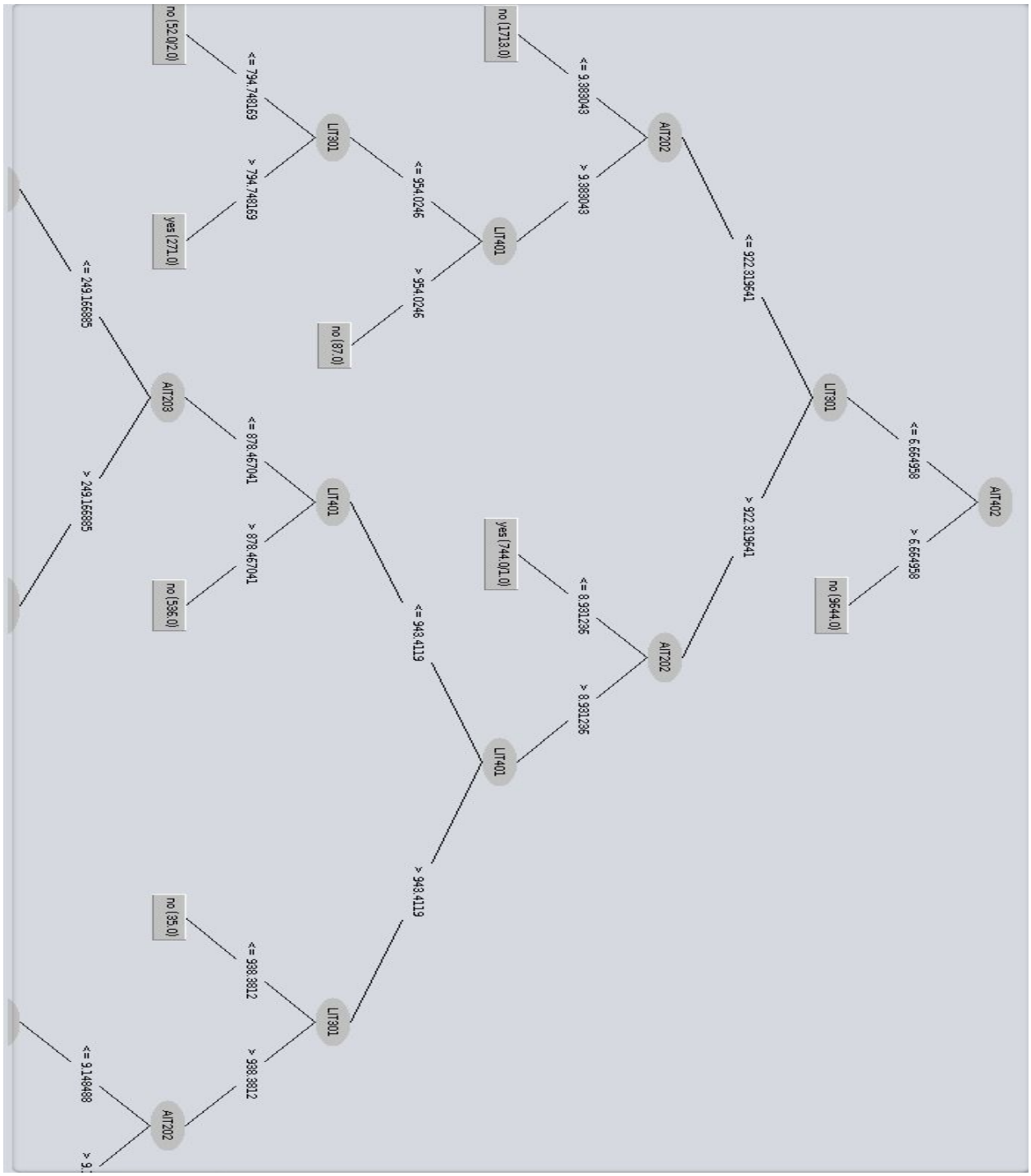
Adem Malone
Tho Nguyen
Matthew Stroble

Tree Decision part 1:



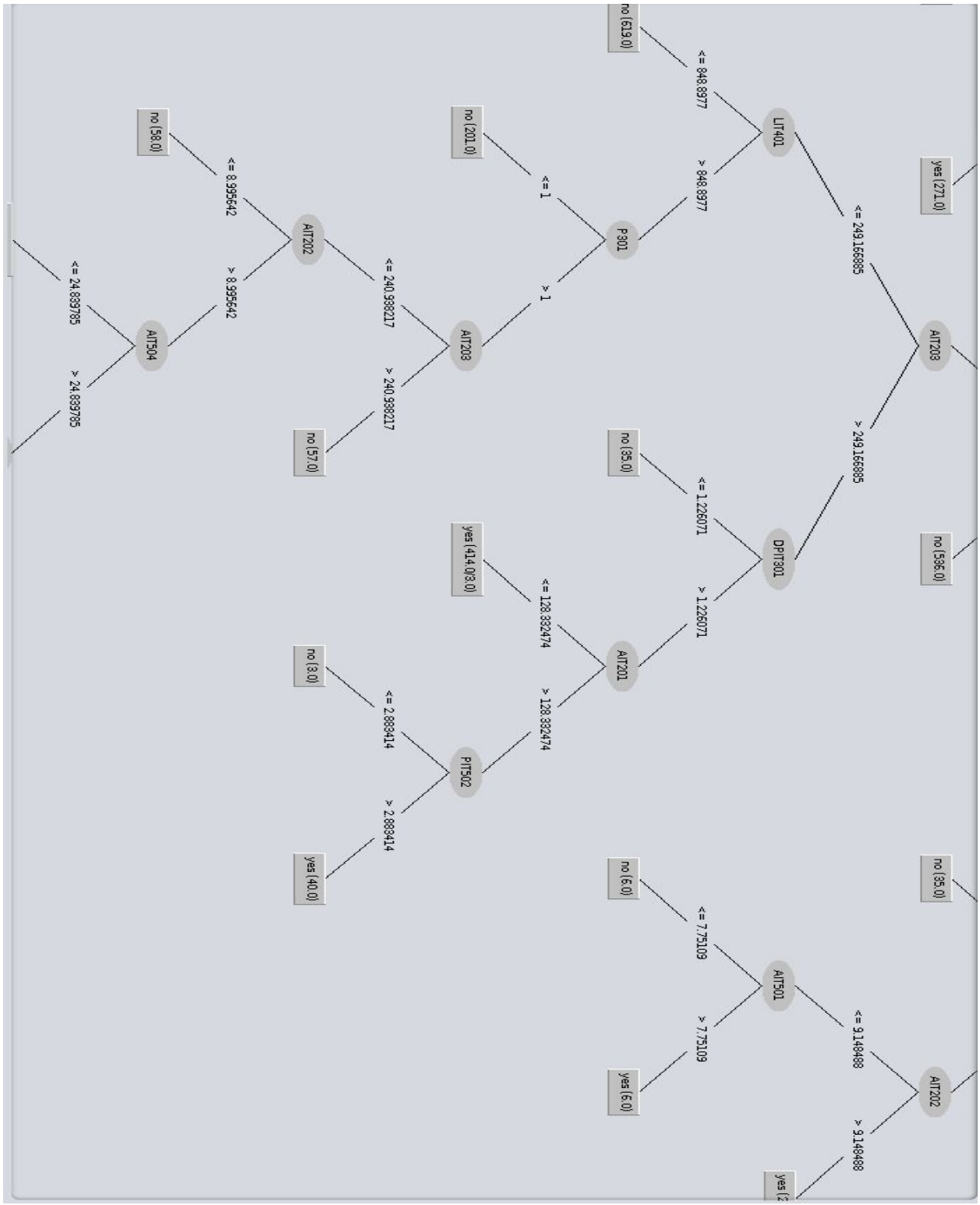
Adem Malone
Tho Nguyen
Matthew Stroble

Tree Decision part :



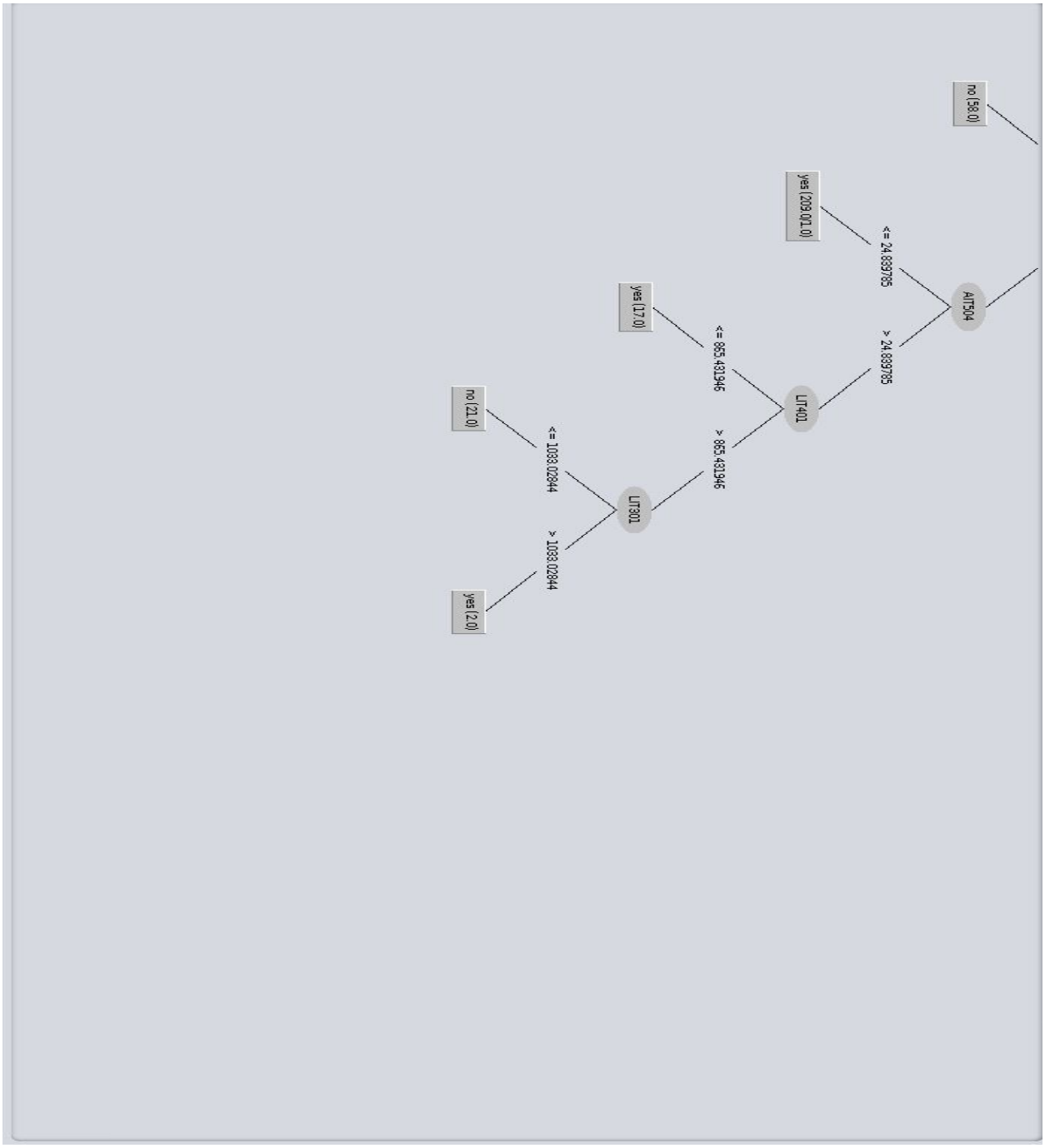
Adem Malone
Tho Nguyen
Matthew Stroble

Tree Decision 2 :



Adem Malone
Tho Nguyen
Matthew Stroble

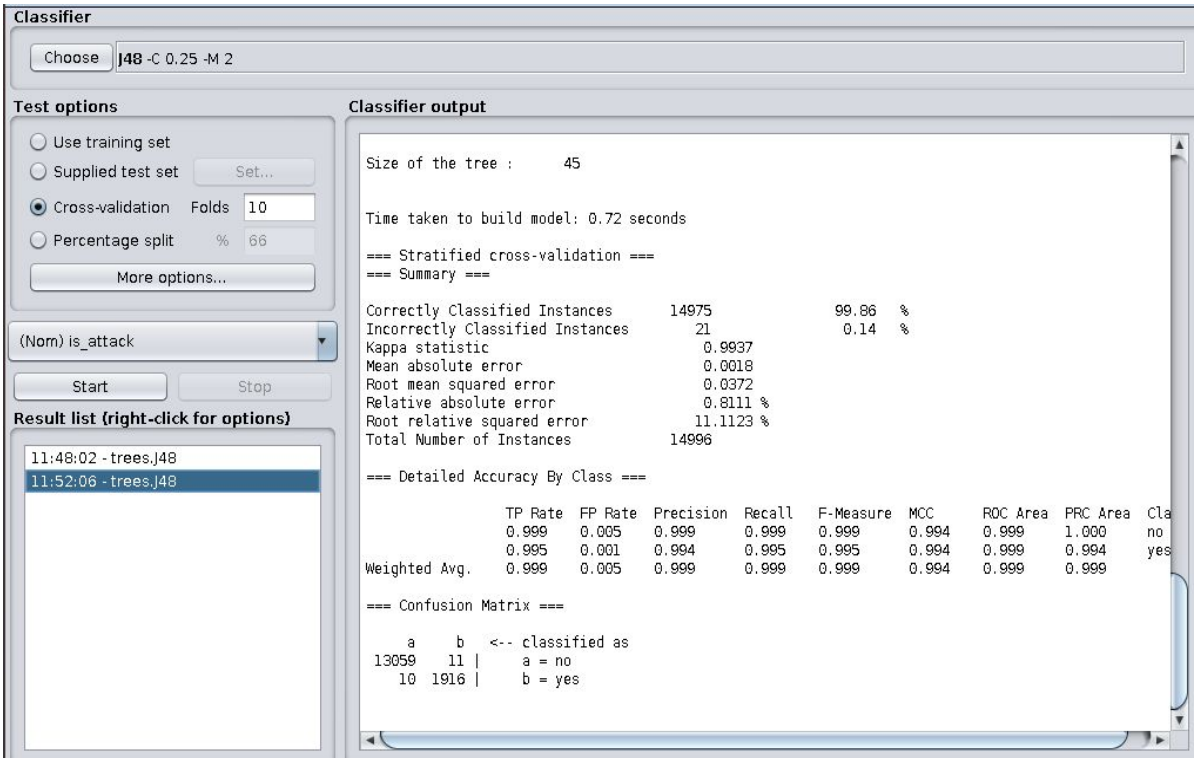
Tree decision part 3:



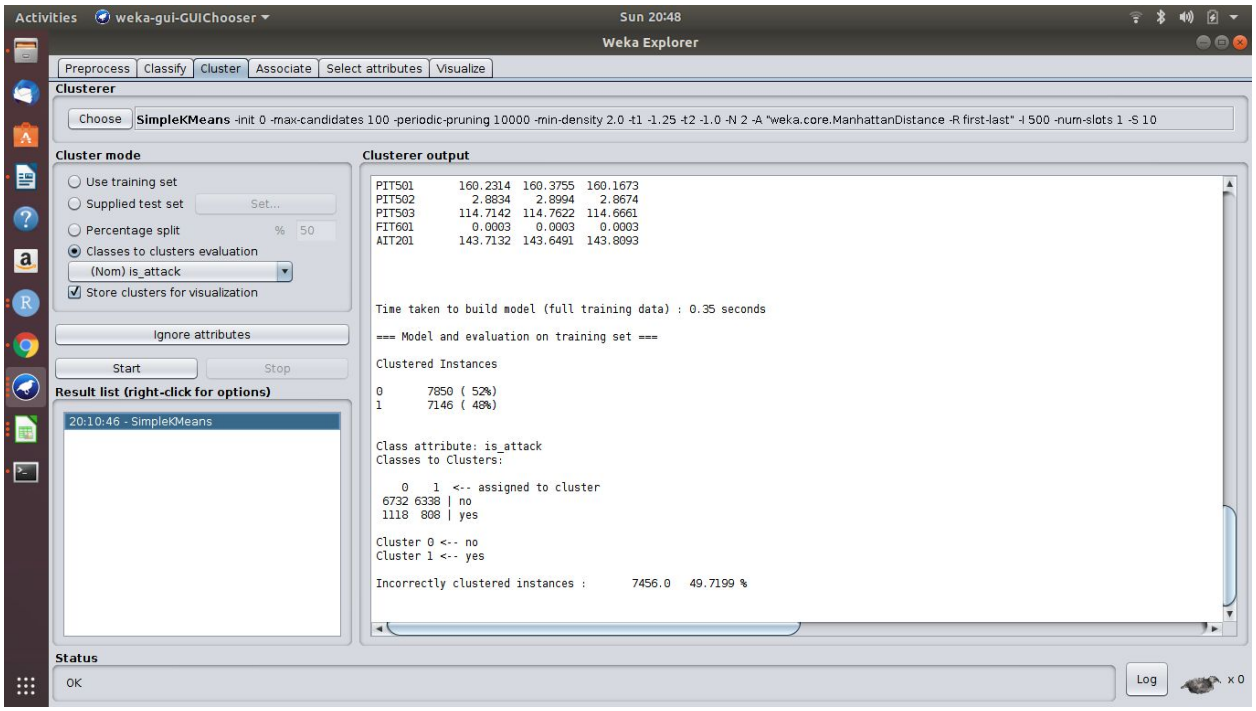
	attr_importance
P101	0.0059094653
P203	0.0052508477
P205	0.0052508477
P301	0.0009454872
P401	0.0000000000
P601	0.0021258681
MV301	0.0000000000
MV302	0.0048306645
MV304	0.0035880857
UV401	0.0022377562
MV101	0.0000000000
MV201	0.0056116099
MV303	0.0000000000
FIT101	0.0116083157
LIT101	0.0286389292
AIT202	0.1804857135
AIT203	0.1445374890
FIT201	0.0146257899
DPIT301	0.1903318149
FIT301	0.0220996207
LIT301	0.1555002645
AIT402	0.2105723925
FIT401	0.0530850688
LIT401	0.0977523670
AIT501	0.1455193681
AIT502	0.1654600569

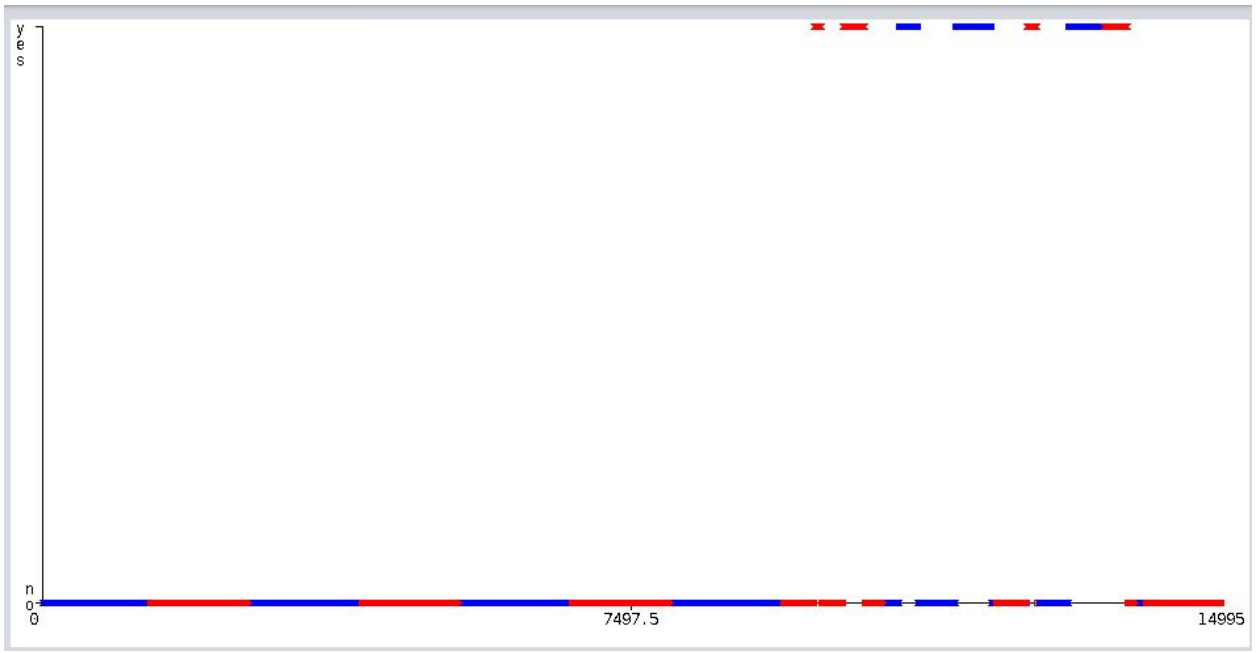
AIT502	0.1654600569
AIT503	0.0009215789
AIT504	0.0878996224
FIT501	0.1030665774
FIT502	0.0870248882
FIT503	0.1133592471
FIT504	0.0346715019
PIT501	0.1778254283
PIT502	0.1129596271
PIT503	0.1711220908
FIT601	0.0050391197
AIT201	0.2173780299

Tree stats:


Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) is_attack
Start
Stop
Result list (right-click for options)
11:48:02 - trees.J48
11:52:06 - trees.J48
Classifier output
Size of the tree : 45
Time taken to build model: 0.72 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 14975 99.86 %
Incorrectly Classified Instances 21 0.14 %
Kappa statistic 0.9937
Mean absolute error 0.0018
Root mean squared error 0.0372
Relative absolute error 0.8111 %
Root relative squared error 11.1123 %
Total Number of Instances 14996
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Cla
0.999 0.005 0.999 0.999 0.999 0.994 0.999 1.000 no
0.995 0.001 0.994 0.995 0.995 0.994 0.999 0.994 yes
Weighted Avg. 0.999 0.005 0.999 0.999 0.999 0.994 0.999 0.999
=== Confusion Matrix ===
a b <-- classified as
13059 11 | a = no
10 1916 | b = yes

Adem Malone
Tho Nguyen
Matthew Stroble





=== Clustering model (full training set) ===

kMeans

======

Number of iterations: 5

Sum of within cluster distances: 54565.088145100686

Initial starting points (random):

Cluster 0:
2,2,2,1,2,1,1,1,1,2,2,2,1,4.371777,537.9983,9.412844,269.3412,2.317624,1.206863,0.000384,950.3973
39,12.022558,0.79557,979.056641,7.722251,71.21251,1016.27789,24.839785,0.801589,0.34674,0.610
441,0.211895,160.695969,2.899433,115.002563,0.00032,143.71315

Cluster 1:
2,2,2,1,2,1,1,1,1,2,2,2,1,4.372097,539.803955,9.244297,259.5232,2.315574,1.200461,0.000512,1107.8
89,4.716739,0.804277,894.6552,7.756217,65.75237,1016.21381,25.147398,0.804153,0.384655,0.6082
64,0.209973,159.446289,2.931471,114.201485,0.00032,121.923866

Missing values globally replaced with mean/mode

Final cluster centroids:

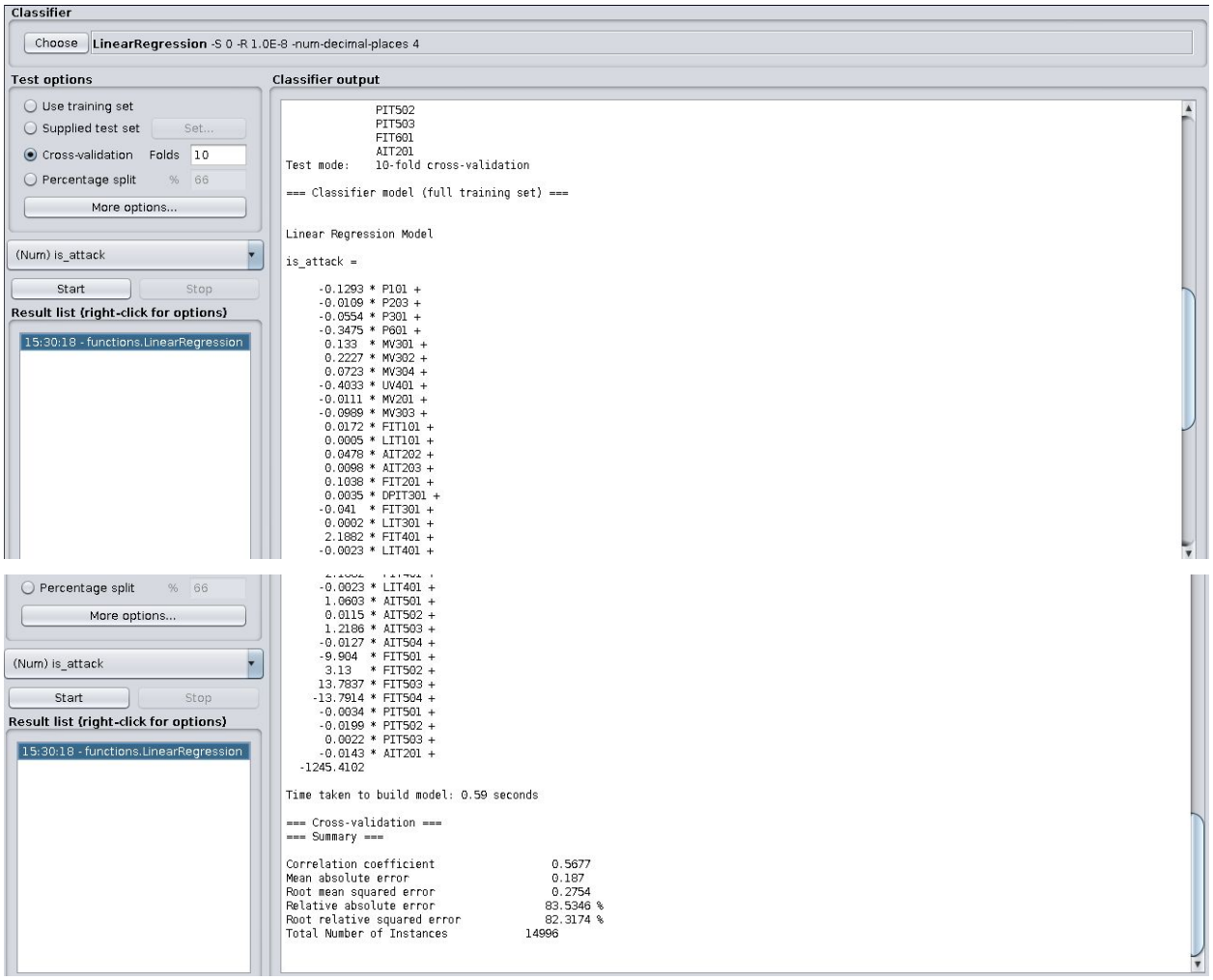
		Cluster#	
Attribute	Full Data	0	1
	(14996.0)	(7850.0)	(7146.0)
P101	1	1	1
P203	1	1	1
P205	1	1	1
P301	1	1	2
P401	2	2	2
P601	1	1	1

MV301	1	1	1
MV302	2	1	2
MV304	1	1	1
UV401	2	2	2
MV101	1	1	1
MV201	1	1	1
MV303	1	1	1
FIT101	0	0	0
LIT101	819.6368	819.4013	819.7938
AIT202	9.2331	9.2437	9.1374
AIT203	246.2189	251.1535	241.3227
FIT201	0.0005	0.0005	0.0005
DPIT301	1.4982	1.2101	12.5232
FIT301	0.0006	0.0005	1.7187
LIT301	981.1587	1008.1149	859.8757
AIT402	9.7411	10.5358	9.4591
FIT401	0.7994	0.7993	0.7994
LIT401	881.274	886.9649	876.1215
AIT501	7.7331	7.7287	7.7347
AIT502	68.9823	69.5206	67.8544
AIT503	1016.2779	1016.2779	1016.2779
AIT504	24.8398	24.8398	24.8398
FIT501	0.8016	0.8017	0.8016
FIT502	0.3469	0.3471	0.3467
FIT503	0.6104	0.6104	0.6104
FIT504	0.2111	0.2111	0.2113
PIT501	160.2314	160.3755	160.1673
PIT502	2.8834	2.8994	2.8674
PIT503	114.7142	114.7622	114.6661
FIT601	0.0003	0.0003	0.0003

```
AIT201      143.7132  143.6491  143.8093

Time taken to build model (full training data) : 0.35 seconds
=== Model and evaluation on training set ===
Clustered Instances
0      7850 ( 52%)
1      7146 ( 48%)

Class attribute: is_attack
Classes to Clusters:

   0    1  <-- assigned to cluster
6732 6338 | no
1118  808 | yes

Cluster 0 <-- no
Cluster 1 <-- yes

Incorrectly clustered instances :       7456.0  49.7199 %
```

Adem Malone
Tho Nguyen
Matthew Stroble

Linear Regression Model :



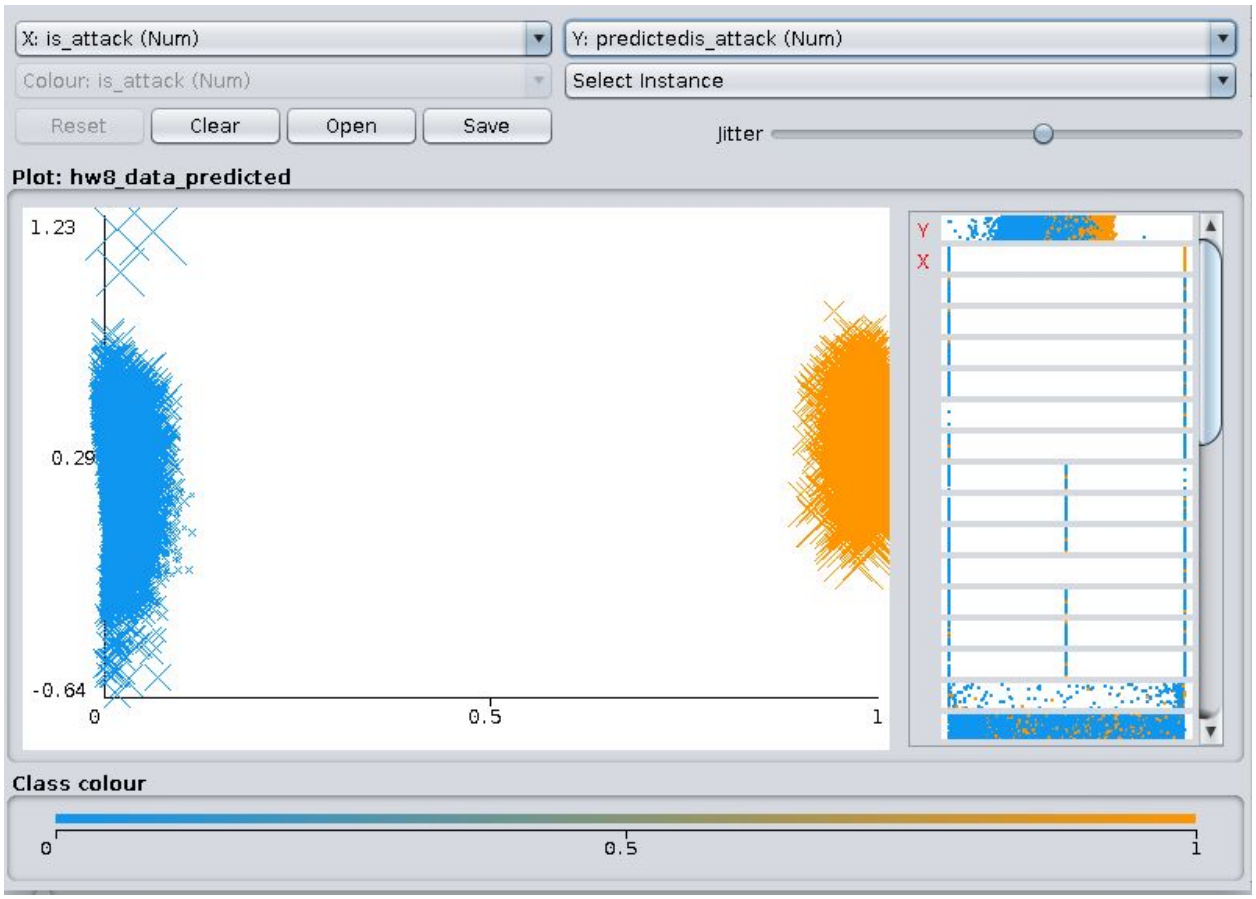
Linear Code from R

```
## Linear regression of the numeric dataset
```

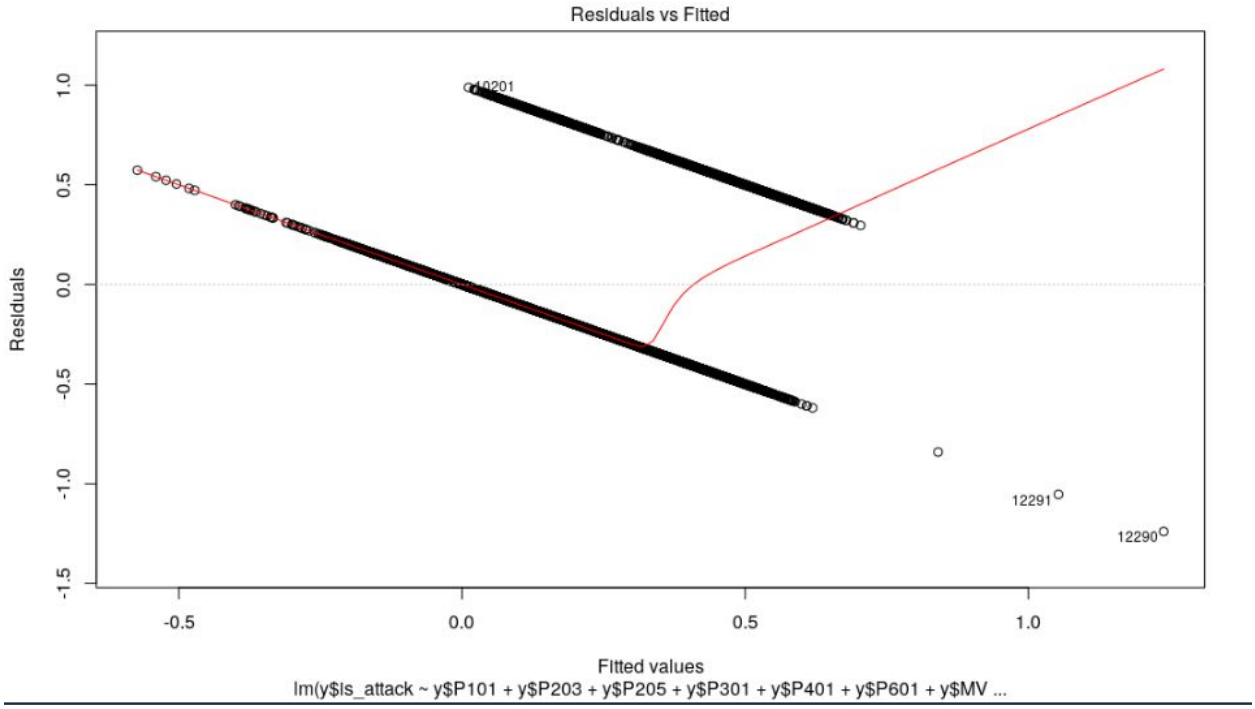
```
fit<-lm(y$is_attack~y$P101+
          y$P203+y$P205+y$P301+y$P401+y$P601+y$MV301+y$MV302+
          y$MV304+y$UV401+y$MV101+y$MV201+y$MV303+y$AIT202+
          y$AIT203+y$FIT201+y$DPIT301+y$FIT301+y$LIT301+y$AIT402+y$FIT401+
          y$LIT401+y$AIT501+y$AIT502+y$AIT503+y$AIT504+y$FIT501+
          y$FIT502+y$FIT503+y$FIT504+y$PIT501+y$PIT502+y$PIT503+
          y$FIT601+y$AIT201 )
plot(fit)
```

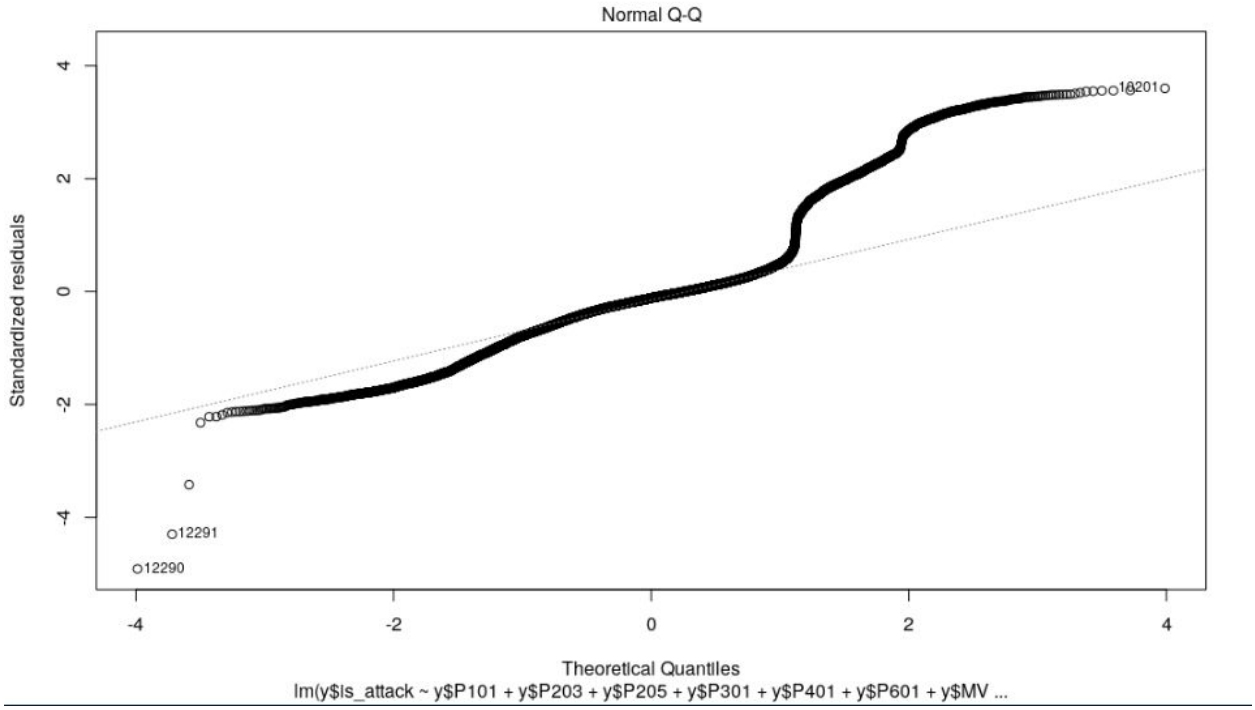
Adem Malone
Tho Nguyen
Matthew Stroble

Linear Graph

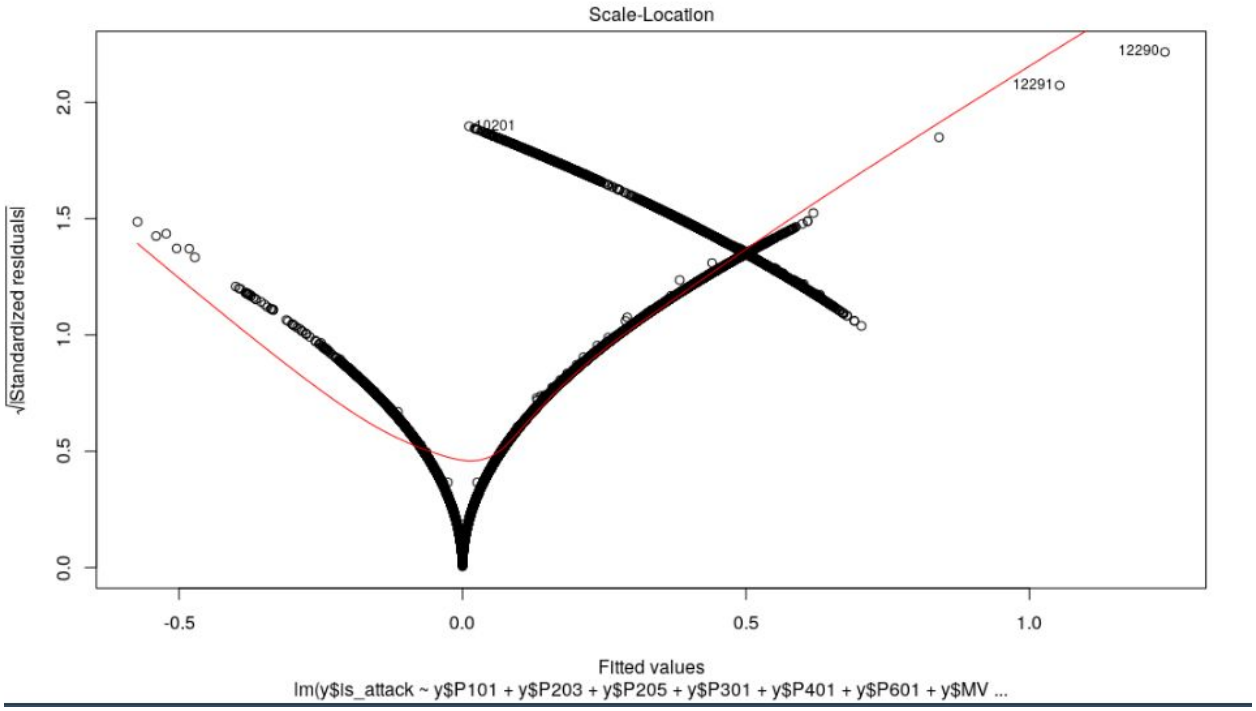


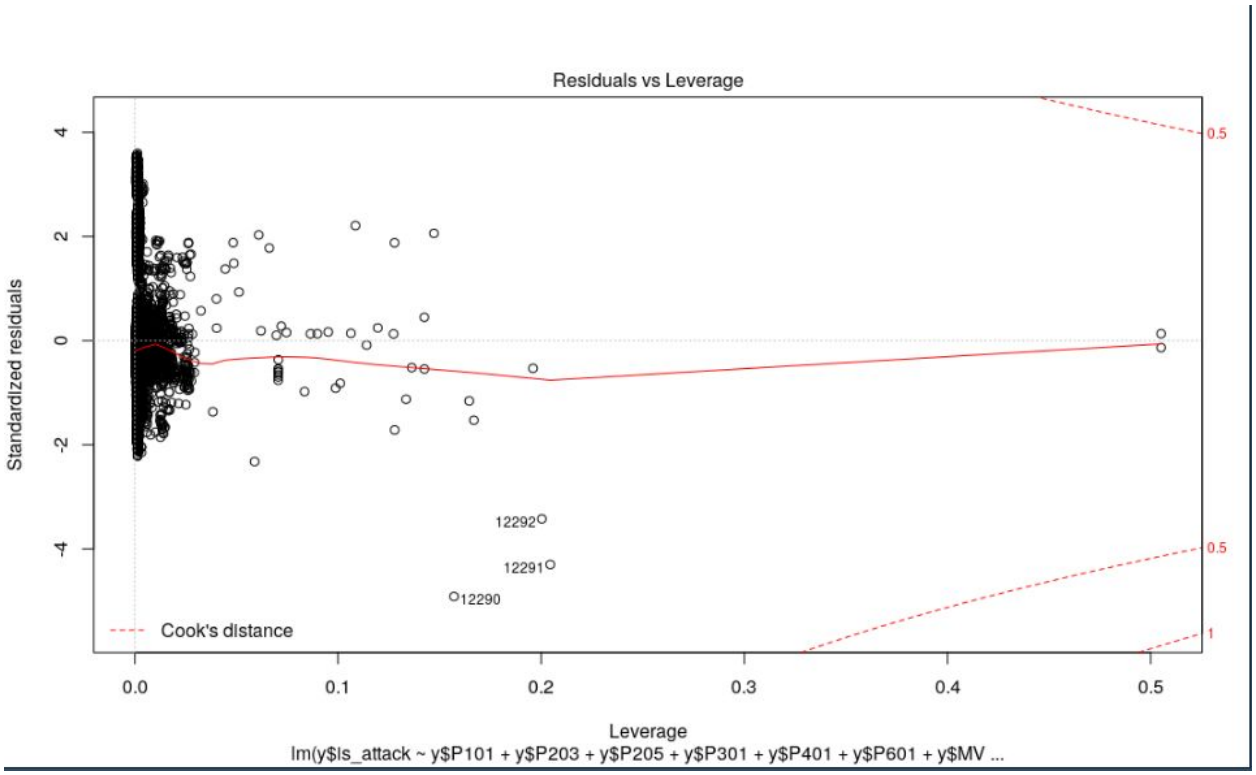
Adem Malone
Tho Nguyen
Matthew Stroble





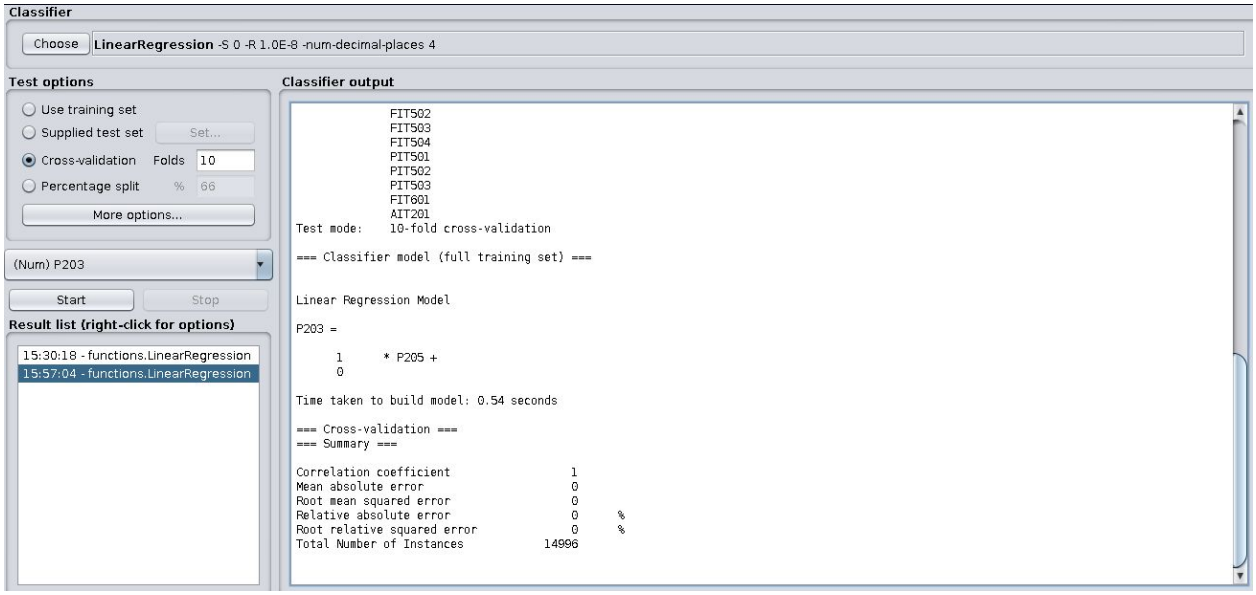
Adem Malone
Tho Nguyen
Matthew Stroble

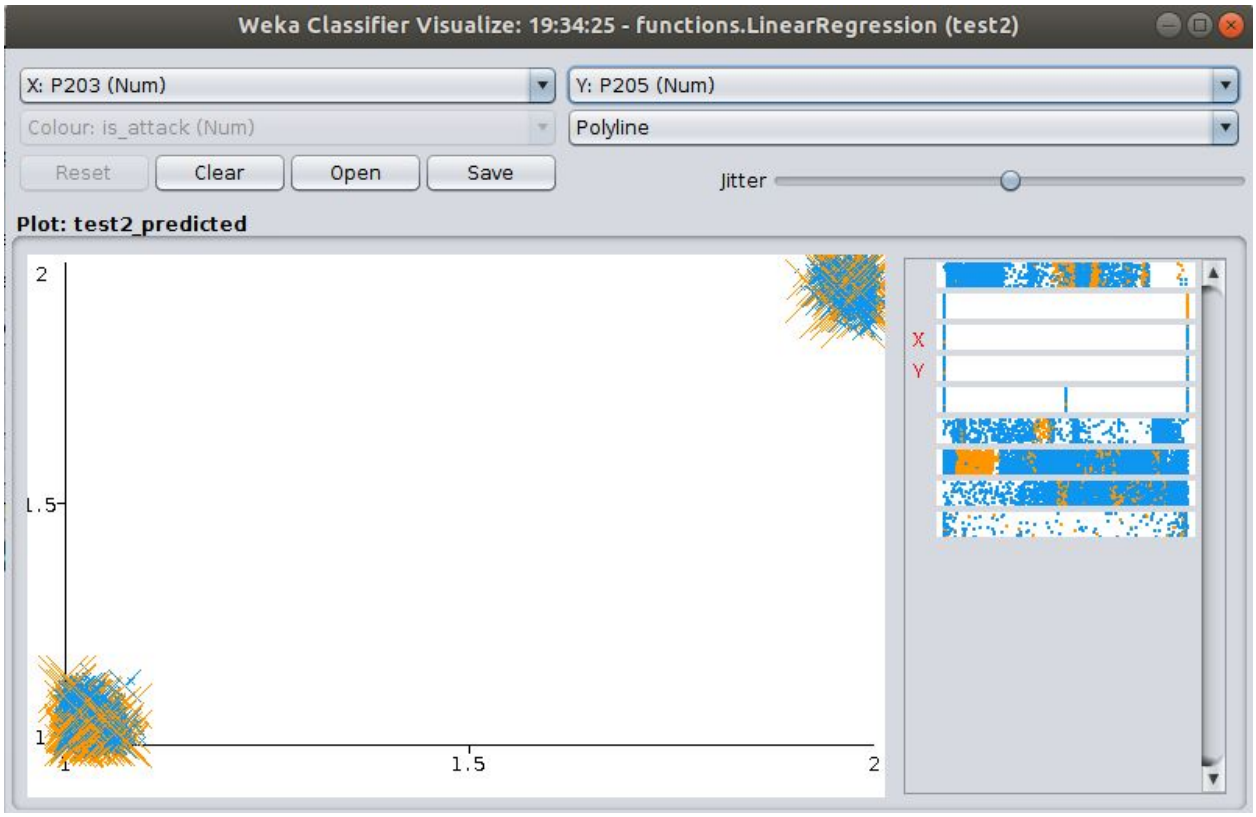




Adem Malone
Tho Nguyen
Matthew Stroble

Linear Regression P203 vs P205





AdaBoost Report

=== Run information ===

Scheme: weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W
weka.classifiers.trees.DecisionStump

Relation: hw8Nominal-weka.filters.unsupervised.attribute.Remove-R1

Test mode: 10-fold cross-validation

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

AdaBoostM1: Base classifiers and their weights:

Decision Stump

Classifications

AIT402 <= 6.677774905 : no

AIT402 > 6.677774905 : no

AIT402 is missing : no

Class distributions

AIT402 <= 6.677774905

no yes

0.6401345291479821 0.3598654708520179

AIT402 > 6.677774905

no yes

1.0 0.0

AIT402 is missing

no yes

0.8715657508668978 0.12843424913310217

Weight: 1.91

Decision Stump

Classifications

AIT402 <= 6.677774905 : yes

AIT402 > 6.677774905 : no

AIT402 is missing : yes

Class distributions

AIT402 <= 6.677774905

no yes

0.20768671193015970.7923132880698402

AIT402 > 6.677774905

no yes

1.0 5.1371606222803254E-17

AIT402 is missing

no yes

0.499999999999974 0.500000000000026

Weight: 1.89

Decision Stump

Classifications

PIT503 <= 114.53794099999999 : no

PIT503 > 114.53794099999999 : no

PIT503 is missing : no

Class distributions

PIT503 <= 114.53794099999999

no yes

0.6225386899819644 0.37746131001803557

PIT503 > 114.53794099999999

no yes

1.0 -4.7326071865815086E-18

PIT503 is missing

no yes

0.71229197851551910.2877080214844809

Weight: 0.91

Decision Stump

Classifications

PIT503 <= 114.53794099999999 : yes

PIT503 > 114.53794099999999 : no

PIT503 is missing : no

Class distributions

PIT503 <= 114.53794099999999

no yes

0.39982266648766507 0.6001773335123349

PIT503 > 114.53794099999999

no yes

0.9999999999999998 1.1674298809448228E-16

PIT503 is missing

no yes

0.5000000000000432 0.49999999999995676

Weight: 0.69

Decision Stump

Classifications

LIT401 <= 790.3550339999999 : yes

LIT401 > 790.3550339999999 : no

LIT401 is missing : no

Class distributions

LIT401 <= 790.3550339999999

no yes

0.006622987773379027 0.993377012226621

LIT401 > 790.3550339999999

no yes

0.6818979335585613 0.3181020664414386

LIT401 is missing

no yes

0.6251384497373392 0.3748615502626609

Weight: 0.89

Decision Stump

Classifications

PIT503 <= 114.53794099999999 : yes

PIT503 > 114.53794099999999 : no

PIT503 is missing : yes

Class distributions

PIT503 <= 114.53794099999999

no yes

0.38790553990603616 0.612094460093964

PIT503 > 114.53794099999999

no yes

0.9999999999999774 2.2734004227713835E-14

PIT503 is missing

no yes

0.4419931177564975 0.5580068822435029

Weight: 0.6

Decision Stump

Classifications

LIT301 <= 923.2408499999999 : no

LIT301 > 923.2408499999999 : yes

LIT301 is missing : no

Class distributions

LIT301 <= 923.2408499999999

no yes

0.8085583213566678 0.1914416786433321

LIT301 > 923.2408499999999

no yes

0.42783805607722536 0.5721619439227746

LIT301 is missing

no yes

0.5683544505231004 0.43164554947689965

Weight: 0.66

Decision Stump

Classifications

AIT202 <= 9.383363500000002 : no

AIT202 > 9.383363500000002 : yes

AIT202 is missing : no

Class distributions

AIT202 <= 9.383363500000002

no yes

0.6831531205244171 0.3168468794755829

AIT202 > 9.383363500000002

no yes

0.23397932369862395 0.7660206763013762

AIT202 is missing

no yes

0.62255164947836990.37744835052163

Weight: 0.82

Decision Stump

Classifications

AIT402 <= 6.677774905 : yes

AIT402 > 6.677774905 : no

AIT402 is missing : yes

Class distributions

AIT402 <= 6.677774905

no yes

0.42863160420997276 0.5713683957900272

AIT402 > 6.677774905

no yes

0.9999999999998577 1.4225072999148033E-13

AIT402 is missing

no yes

0.47721417749149964 0.5227858225085004

Weight: 0.44

Decision Stump

Classifications

LIT401 <= 815.098694 : yes

LIT401 > 815.098694 : no

LIT401 is missing : no

Class distributions

LIT401 <= 815.098694

no yes

0.24371635580241877 0.7562836441975813

LIT401 > 815.098694

no yes

0.64196533451870510.35803466548129487

LIT401 is missing

no yes

0.56994608401045830.4300539159895416

Weight: 0.68

Number of performed Iterations: 10

Time taken to build model: 1.21 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 13630 90.8909 %

Incorrectly Classified Instances 1366 9.1091 %

Kappa statistic 0.4298

Mean absolute error 0.1155

Root mean squared error 0.2369

Relative absolute error 51.5719 %

Root relative squared error 70.8166 %

Total Number of Instances 14996

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class

0.997 0.691 0.907 0.997 0.950 0.511 0.962 0.994 no

0.309 0.003 0.943 0.309 0.466 0.511 0.962 0.764 yes

Weighted Avg. 0.909 0.602 0.912 0.909 0.888 0.511 0.962 0.965

=== Confusion Matrix ===

a b <-- classified as

13034 36 | a = no

1330 596 | b = yes

Adem Malone
Tho Nguyen
Matthew Stroble

SuperLearner using R Complete System Data Set, all subsystems

The following tables and graphs were built using the SuperLearner R package while using a numeric data set containing all subsystems P1, P2, P3, P4, P5, P6. SuplerLearner is a Stacking based ensemble classification package that supports a wide range of packages and methods. The methods used to build this ensemble are, xgbost, ranger, ksvm, kernnelknn using manhattan distance of 5, and bayesglm. The SL is due to the wrapper required to use these methods in SuperLearner. SuperLearning uses Risk assessment and makes a determination on whether a particular method is dropped from the ensemble. The method SL.mean is a baseline indicator during Risk assessment of a method. If the more complex method has a higher risk assessment than SL.mean, it indicates that something may be wrong with the machine learning model or that something is wrong with the data provided or setup. All methods used in SuperLearner were run with 10-fold cross validation.

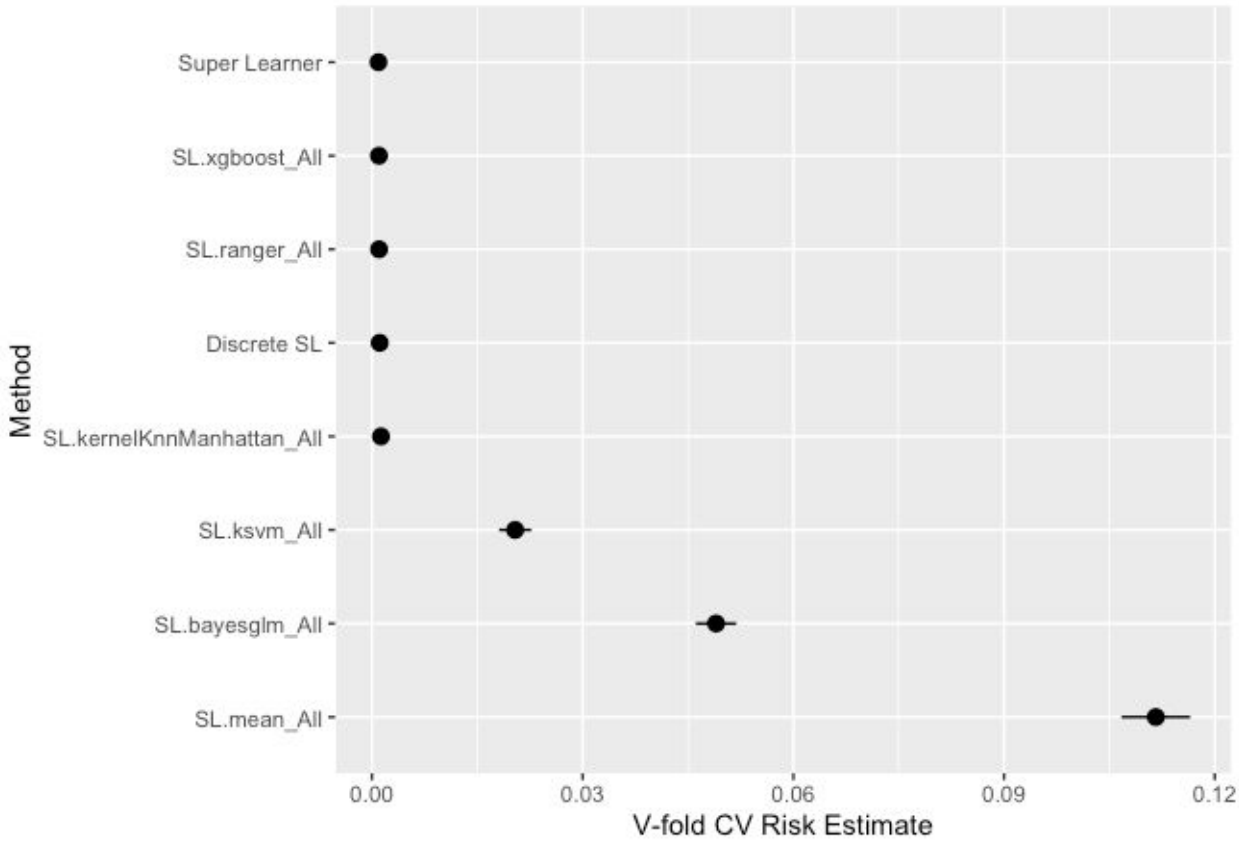
The data set was broken up into a ⅔ training dataset using random sampling and the remaining ⅓ of data was used for the testing dataset. This is noted as xTrain and xTest in the source code. The yTrain and yTest sets are the decision attribute is_attack contained in the data set. A custom wrapper was built for kernnelKnn to force the use of manhattan distance during its calculations. The model that is built from the SuperLearner is stored in the model. We could not find a way to look inside the model to find out how Superlearner made its predictions but we validated it based using other methods and software.

To run the code you must have the SuperLearner R package installed. You will also need to install additional packages needed by SuperLearner as described in its documentation. Due to the system this was run on you may need to adjust the mc.cores variable since the program has been made to work with multicore processing to speed up computation. If all packages are installed just run the superlearner.R file to compute everything we did. To get the plots run the following commands

plot(crossFoldValidation)	to get the Risk assessment graph
confMatrix	to get the confusion matrix statistical output
summary(cv_sl)	to get the Area Under the Curve Calculations

Adem Malone
Tho Nguyen
Matthew Stroble

Superlearner Estimate Risk using provided Numeric Dataset (No changes) Lower is better



SuperLearner Confusion Matrix predictions using Ensemble methods (Stacking)

```
Console   Terminal ×   Jobs ×
~/Documents/theDataMiningProject/Part 2/
> confMatrix
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4438    1
         1    3  657

               Accuracy : 0.9992
                 95% CI : (0.998, 0.9998)
    No Information Rate : 0.871
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9965

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.9993
            Specificity : 0.9985
         Pos Pred Value : 0.9998
         Neg Pred Value : 0.9955
             Prevalence : 0.8710
         Detection Rate : 0.8704
   Detection Prevalence : 0.8706
      Balanced Accuracy : 0.9989

       'Positive' Class : 0
```

Area Under Curve Calculation for SuperLearner 10 Fold CrossValidation

```
> summary(cv_sl)

Call:
CV.SuperLearner(Y = yTrain, X = xTrain, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")

Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

               Algorithm     Ave se     Min     Max
           Super Learner 0.99999 NA 0.99988 1.00000
             Discrete SL 1.00000 NA 0.99998 1.00000
             SL.mean_All 0.50000 NA 0.50000 0.50000
          SL.ranger_All 1.00000 NA 0.99998 1.00000
             SL.ksvm_All 0.99341 NA 0.98757 0.99613
 SL.kernelKnnManhattan_All 0.99998 NA 0.99993 1.00000
         SL.bayesglm_All 0.96343 NA 0.95610 0.96913
          SL.xgboost_All 0.99999 NA 0.99990 1.00000
>
```

superlearner.R

```
library(SuperLearner)
library(KernelKnn)
library(kernlab)
library(dplyr)
library(caret)
library(e1071)
library(bnlearn)
library(arules)
library(arm)
library(parallel)
library(xgboost)
library(ranger)

# listWrappers()
#load data into frame
data = hw8_data

options(mc.cores = 4)
getOption("mc.cores")
set.seed(1, "L'Ecuyer-CMRG")

# Reduce to a dataset random observations.
trainingSet = sample(seq_len(nrow(data)), size = floor(0.66*nrow(data)))
xTrain = data[trainingSet,2:38]
xTest = data[-trainingSet,2:38]

yTrain = as.numeric(data[trainingSet, 1])
yTest = as.numeric(data[-trainingSet, 1])

SL.kernelKnnManhattan = function(...) {
 SL.kernelKnn(..., method = "manhattan", k=5)
}


algorithmList =
list("SL.mean","SL.ranger","SL.ksvm","SL.kernelKnnManhattan","SL.bayesglm","SL.xgboost")

# Risk assessment
system.time({
crossFoldValidation = CV.SuperLearner(yTrain, xTrain, V = 10, parallel = "multicore", family
= binomial(), SL.library = algorithmList)
})

#Building the model
model = SuperLearner(Y = yTrain, X = xTrain, family = binomial(), SL.library =
algorithmList)
```

```
prediction = predict.SuperLearner(model, newdata = xTest)
predictedResult = as.numeric(ifelse(prediction$pred>=0.5,1,0))
confMatrix = confusionMatrix(as.factor(yTest), as.factor(predictedResult))


# Calculating area under the curve using 10 fold cross validation
system.time({
cv_sl = CV.SuperLearner(Y = yTrain, X = xTrain, family = binomial(),
                        V = 10,
                        parallel = "multicore",
                        method = "method.AUC",
                        SL.library = algorithmList)
})

confMatrix
summary(cv_sl)
```

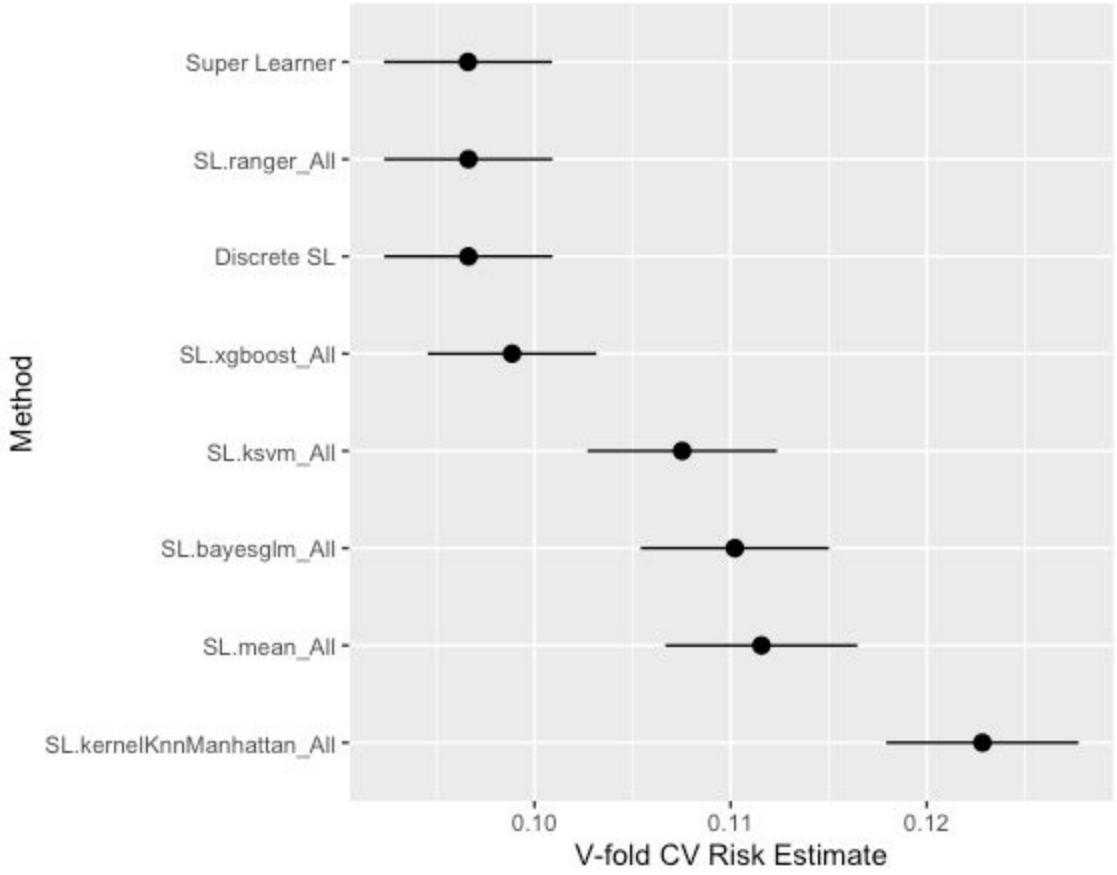
Adem Malone
Tho Nguyen
Matthew Stroble

SuperLearner using R Subsystem Data Sets

The following tables and graphs are the results of running the same SuperLearner techniques on column trimmed dataset. The dataset was split up to match the water treatment plants six subsystems with the is_attack column prepended to the first column for each subsystem P1, P2, P3, P4, P5, and P6. The same ensambles methods xgbost, ranger, ksvm, kernnelknn using manhattan distance of 5, and bayesglm are used on each subsystem dataset. The subsystemSuperLearner.R program will automatically split the original numeric dataset into subsystem specific numeric datasets with the attribute is_attack prepended to the first column in each subsystem dataset. The subsystemSuperLearner.R program will run all methods that were used on the full system dataset on each subsystem data set. The confusion matrix for subsystem P6 could not be calculated due to unknown error.

Subsystem P1

Subsystem P1 Risk Estimate 10-fold cross validation



Subsystem P1 Confusion Matrix statistics

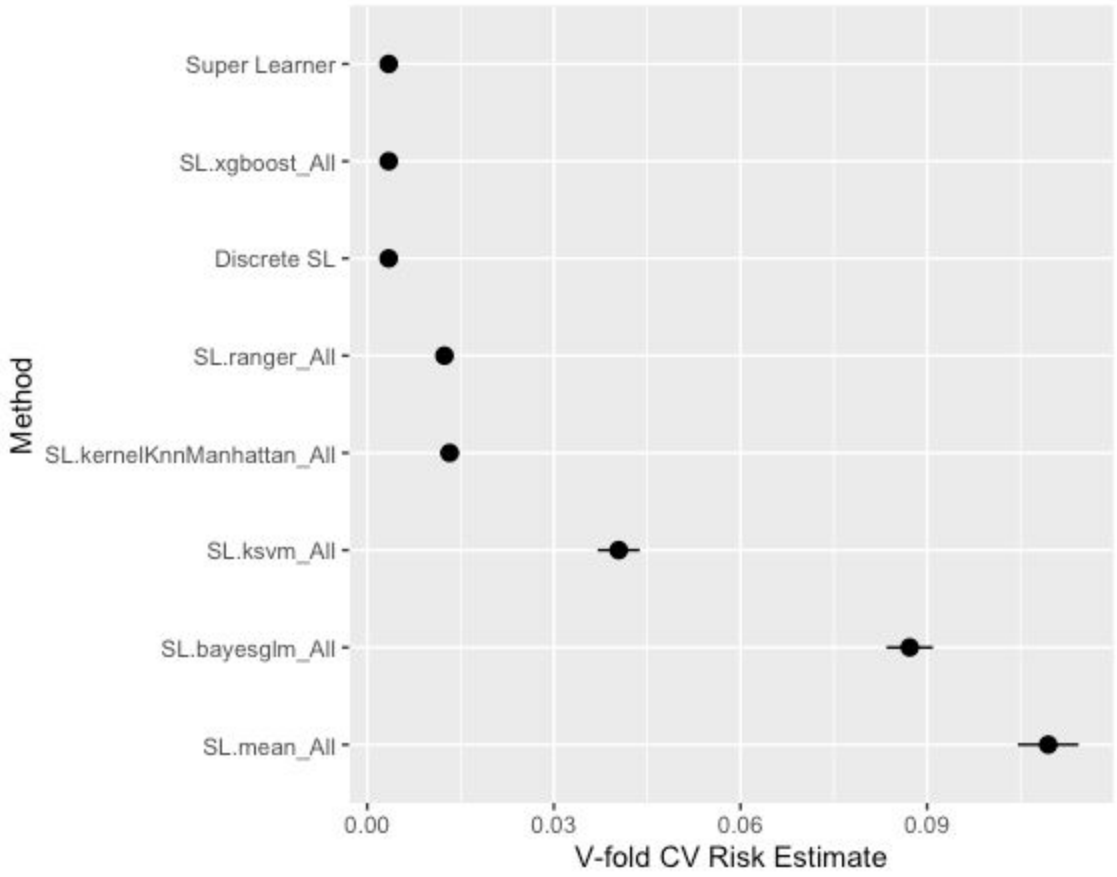
```
> confMatrixSubSysP1
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4420   19
         1  606   54

               Accuracy : 0.8774
                 95% CI : (0.8681, 0.8863)
    No Information Rate : 0.9857
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1248

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.87943
            Specificity : 0.73973
         Pos Pred Value : 0.99572
         Neg Pred Value : 0.08182
             Prevalence : 0.98568
         Detection Rate : 0.86684
   Detection Prevalence : 0.87056
      Balanced Accuracy : 0.80958

       'Positive' Class : 0

>
```

Subsystem P1 Area Under Curve Calculation 10-fold cross validation

```
> summary(cv_slSubSysP1)
Loading required namespace: cvAUC

Call:
CV.SuperLearner(Y = yTrainSubSysP1, X = xTrainSubSusP1, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")


Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

                Algorithm     Ave se     Min     Max
            Super Learner 0.74743 NA 0.71877 0.77427
              Discrete SL 0.74964 NA 0.71179 0.77838
              SL.mean_All 0.50000 NA 0.50000 0.50000
            SL.ranger_All 0.74964 NA 0.71179 0.77838
              SL.ksvm_All 0.54089 NA 0.46805 0.58898
 SL.kernelKnnManhattan_All 0.60437 NA 0.57984 0.62238
          SL.bayesglm_All 0.61073 NA 0.56302 0.69160
           SL.xgboost_All 0.73935 NA 0.71592 0.75657
>
```

Subsystem P2

Subsystem P2 Risk Estimate 10-fold cross validation



Subsystem P2 Confusion Matrix Statistics

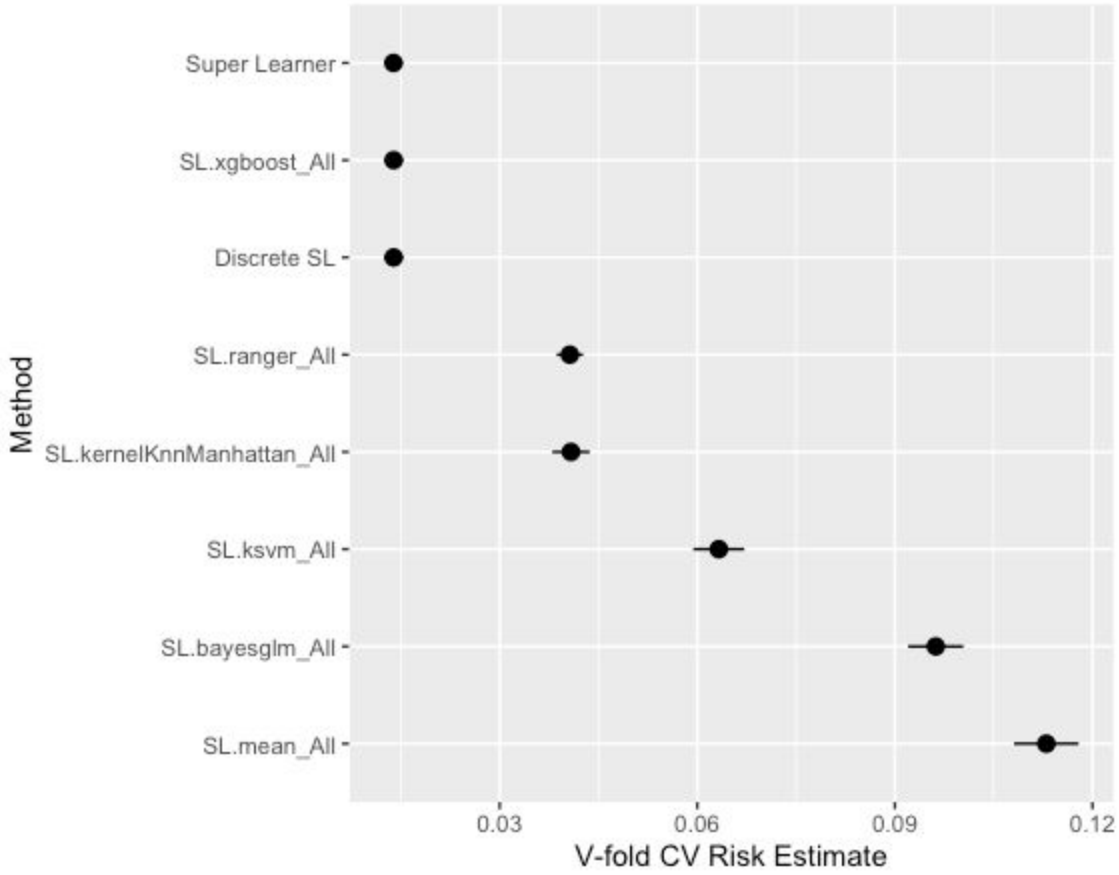
```
> confMatrixSubSysP2
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4407    4
         1   17  671

               Accuracy : 0.9959
                 95% CI : (0.9937, 0.9974)
    No Information Rate : 0.8676
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9822

 Mcnemar's Test P-Value : 0.008829

            Sensitivity : 0.9962
            Specificity : 0.9941
         Pos Pred Value : 0.9991
         Neg Pred Value : 0.9753
             Prevalence : 0.8676
         Detection Rate : 0.8643
   Detection Prevalence : 0.8651
      Balanced Accuracy : 0.9951

       'Positive' Class : 0

> |
```

Subsystem P2 Area Under Curve 10-fold cross validation

```
> summary(cv_slSubSysP2)

Call:
CV.SuperLearner(Y = yTrainSubSysP2, X = xTrainSubSusP2, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")


Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

               Algorithm     Ave se     Min     Max
           Super Learner 0.99970 NA 0.99846 1.00000
             Discrete SL 0.99959 NA 0.99829 1.00000
             SL.mean_All 0.50000 NA 0.50000 0.50000
           SL.ranger_All 0.99949 NA 0.99752 1.00000
             SL.ksvm_All 0.97457 NA 0.95706 0.99053
 SL.kernelKnnManhattan_All 0.99295 NA 0.98474 0.99894
          SL.bayesglm_All 0.79523 NA 0.75967 0.82674
           SL.xgboost_All 0.99959 NA 0.99829 1.00000
>
```

Subsystem P3

Subsystem P3 Risk Estimate 10-fold cross validation



Subsystem P3 Confusion Matrix Statistics

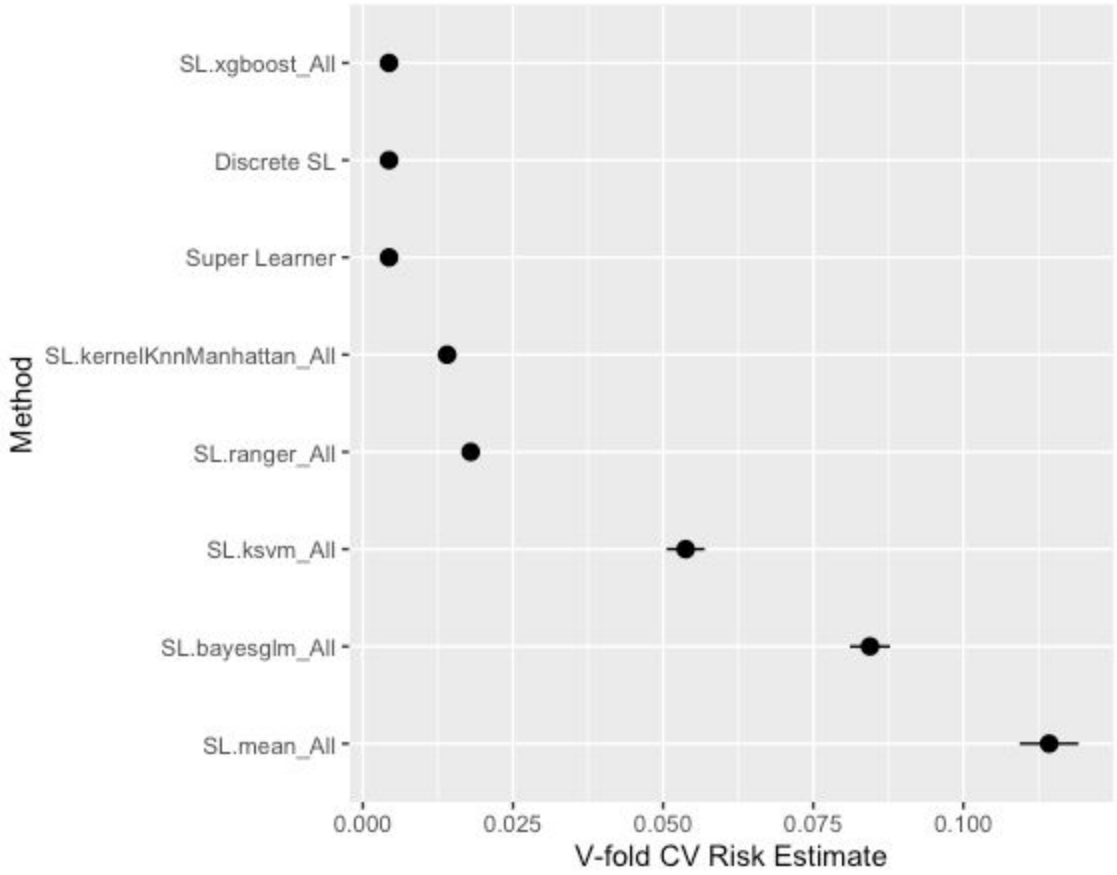
```
> confMatrixSubSysP3
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4415   43
         1   41  600

               Accuracy : 0.9835
                 95% CI : (0.9796, 0.9868)
    No Information Rate : 0.8739
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9252

 Mcnemar's Test P-Value : 0.9131

            Sensitivity : 0.9908
            Specificity : 0.9331
         Pos Pred Value : 0.9904
         Neg Pred Value : 0.9360
             Prevalence : 0.8739
         Detection Rate : 0.8659
   Detection Prevalence : 0.8743
      Balanced Accuracy : 0.9620

       'Positive' Class : 0
```

Subsystem P3 Area Under Curve 10-fold cross validation

```
> summary(cv_slSubSysP3)

Call:
CV.SuperLearner(Y = yTrainSubSysP3, X = xTrainSubSusP3, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")


Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

                Algorithm     Ave se     Min     Max
            Super Learner 0.99732 NA 0.99567 0.99887
              Discrete SL 0.99723 NA 0.99587 0.99853
              SL.mean_All 0.50000 NA 0.50000 0.50000
            SL.ranger_All 0.98734 NA 0.98391 0.99249
              SL.ksvm_All 0.96666 NA 0.95919 0.97725
 SL.kernelKnnManhattan_All 0.95273 NA 0.92969 0.97027
          SL.bayesglm_All 0.76473 NA 0.73437 0.79951
           SL.xgboost_All 0.99723 NA 0.99587 0.99853
> |
```

Subsystem P4

Subsystem P4 Risk Estimate 10-fold cross validation



Subsystem P4 Confusion Matrix Statistics

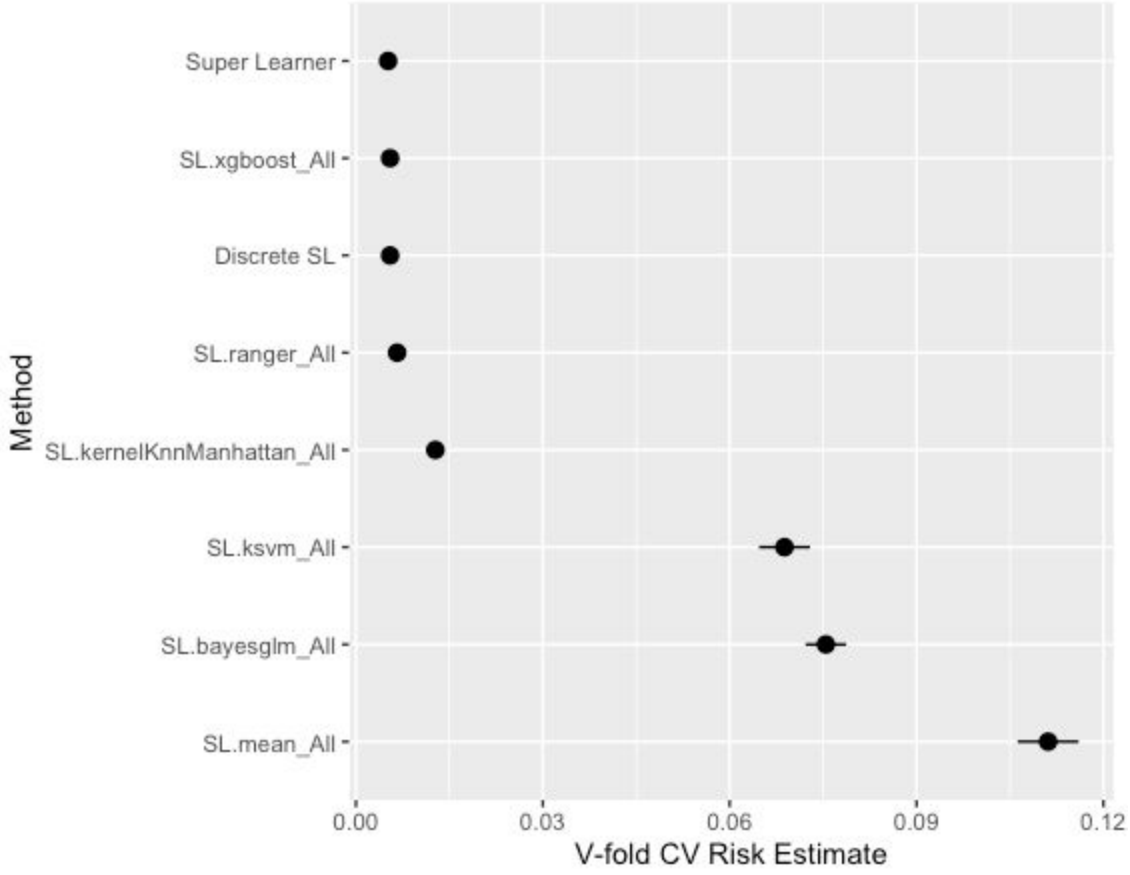
```
> confMatrixSubSysP4
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4464   11
         1   11  613

               Accuracy : 0.9957
                 95% CI : (0.9935, 0.9973)
    No Information Rate : 0.8776
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9799

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9975
            Specificity : 0.9824
         Pos Pred Value : 0.9975
         Neg Pred Value : 0.9824
             Prevalence : 0.8776
         Detection Rate : 0.8755
   Detection Prevalence : 0.8776
      Balanced Accuracy : 0.9900

       'Positive' Class : 0

>
```

Subsystem P4 Area under Curve 10-fold cross validation

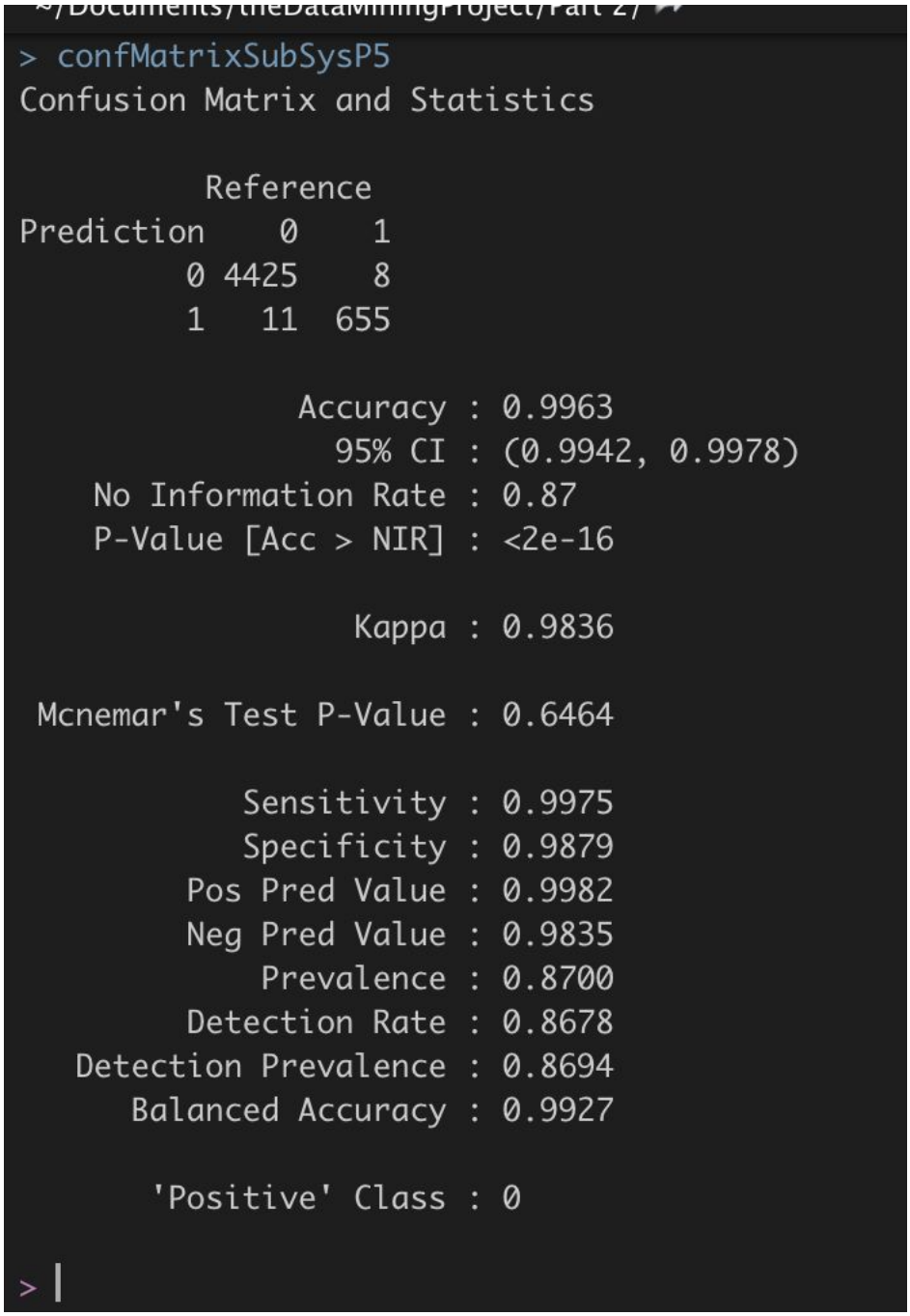
```
> summary(cv_slSubSysP4)

Call:
CV.SuperLearner(Y = yTrainSubSysP4, X = xTrainSubSusP4, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")


Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

               Algorithm     Ave se     Min     Max
           Super Learner 0.99966 NA 0.99908 0.99998
             Discrete SL 0.99958 NA 0.99854 0.99997
             SL.mean_All 0.50000 NA 0.50000 0.50000
           SL.ranger_All 0.99917 NA 0.99766 1.00000
             SL.ksvm_All 0.96645 NA 0.93193 0.97750
 SL.kernelKnnManhattan_All 0.99085 NA 0.97647 0.99916
          SL.bayesglm_All 0.89419 NA 0.85302 0.91660
           SL.xgboost_All 0.99958 NA 0.99854 0.99997
> |
```

Subsystem P5

Subsystem P5 Risk Estimate 10-fold cross validation



Subsystem P5 Confusion Matrix



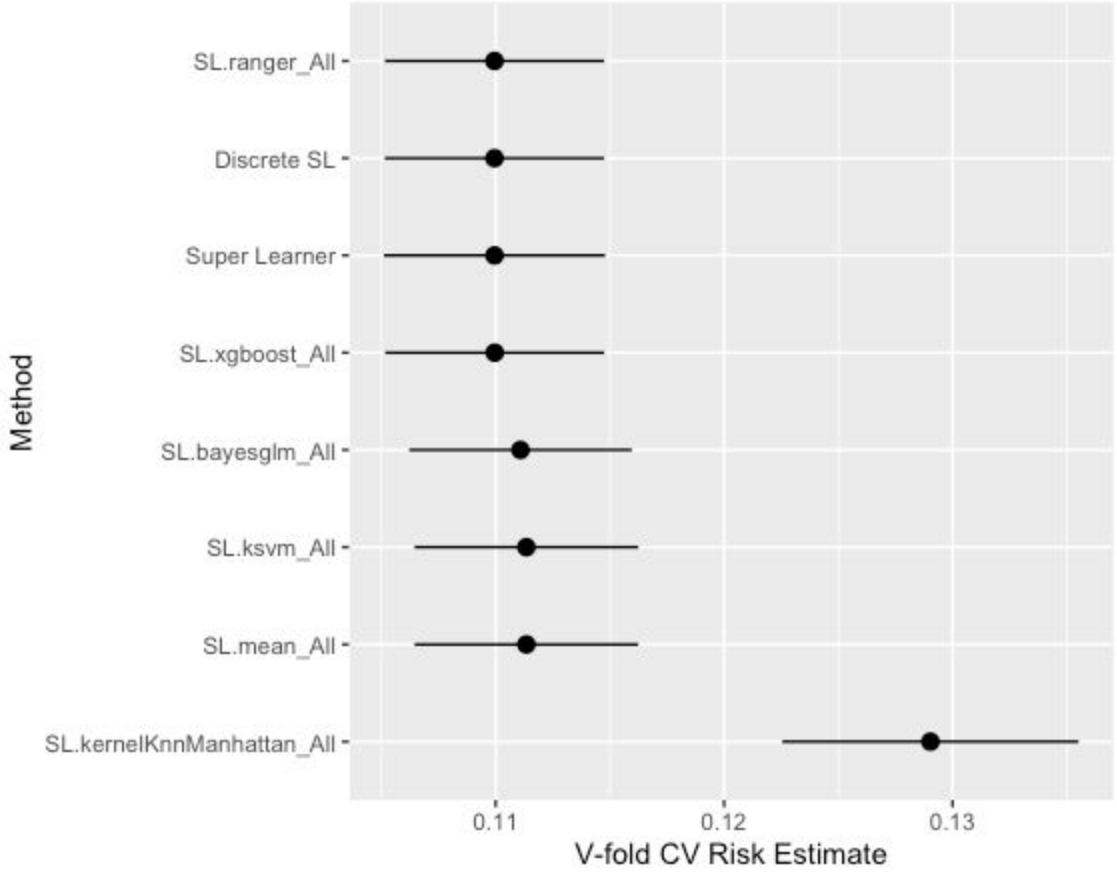
```
> confMatrixSubSysP5
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4425    8
         1   11  655

               Accuracy : 0.9963
                 95% CI : (0.9942, 0.9978)
    No Information Rate : 0.87
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9836

 Mcnemar's Test P-Value : 0.6464

            Sensitivity : 0.9975
            Specificity : 0.9879
         Pos Pred Value : 0.9982
         Neg Pred Value : 0.9835
             Prevalence : 0.8700
         Detection Rate : 0.8678
   Detection Prevalence : 0.8694
      Balanced Accuracy : 0.9927

       'Positive' Class : 0

>
```

Subsystem P5 Area Under Curve 10-fold cross validation

```
> summary(cv_slSubSysP5)

Call:
CV.SuperLearner(Y = yTrainSubSysP5, X = xTrainSubSusP5, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")


Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

               Algorithm     Ave se     Min     Max
           Super Learner 0.99969 NA 0.99856 1.00000
             Discrete SL 0.99975 NA 0.99914 1.00000
             SL.mean_All 0.50000 NA 0.50000 0.50000
          SL.ranger_All 0.99975 NA 0.99914 1.00000
            SL.ksvm_All 0.89974 NA 0.87124 0.92947
 SL.kernelKnnManhattan_All 0.99350 NA 0.98004 0.99907
          SL.bayesglm_All 0.91352 NA 0.89494 0.92602
           SL.xgboost_All 0.99926 NA 0.99707 1.00000
>
```

Subsystem P6

Subsystem P6 Risk Estimate 10-fold cross validation



Subsystem P 6 Area Under Curve 10-fold cross validation

```
> summary(cv_slSubSysP6)

Call:
CV.SuperLearner(Y = yTrainSubSysP6, X = xTrainSubSusP6, V = 10, family = binomial(), SL.library = algorithmList, method = "method.AUC", parallel = "multicore")


Risk is based on: Area under ROC curve (AUC)

All risk estimates are based on V =  10

               Algorithm     Ave se     Min     Max
           Super Learner 0.58665 NA 0.54496 0.62809
             Discrete SL 0.58159 NA 0.54379 0.63429
             SL.mean_All 0.50000 NA 0.50000 0.50000
           SL.ranger_All 0.58107 NA 0.54379 0.63141
             SL.ksvm_All 0.51653 NA 0.43404 0.62802
SL.kernelKnnManhattan_All 0.50330 NA 0.48835 0.51839
         SL.bayesglm_All 0.44089 NA 0.39420 0.48599
          SL.xgboost_All 0.58605 NA 0.55983 0.63429
>
```

subsystemSuperLearner.R

```
library(SuperLearner)
library(KernelKnn)
library(kernlab)
library(dplyr)
library(caret)
library(e1071)
library(bnlearn)
library(arules)
library(arm)
library(parallel)
library(xgboost)
library(ranger)


subSystemP1 = hw8_data[, c("is_attack","P101","MV101","FIT101","LIT101")]
subSystemP2 = hw8_data[,
c("is_attack","P203","P205","MV201","AIT201","AIT202","AIT203","FIT201")]
subSystemP3 = hw8_data[,
c("is_attack","P301","MV301","MV302","MV304","MV303","DPIT301","FIT301","LIT301")]
subSystemP4 = hw8_data[, c("is_attack","P401","UV401","AIT402","FIT401","LIT401")]
subSystemP5 = hw8_data[,
c("is_attack","AIT501","AIT502","AIT503","AIT504","FIT501","FIT502","FIT503","FIT504","PIT50
1","PIT502","PIT503")]
subSystemP6 = hw8_data[, c("is_attack","P601","FIT601")]


SL.kernelKnnManhattan = function(...) {
 SL.kernelKnn(..., method = "manhattan", k=5)
}
```

```
algorithmList =
list("SL.mean","SL.ranger","SL.ksvm","SL.kernelKnnManhattan","SL.bayesglm","SL.xgboost")


options(mc.cores = 3)
getOption("mc.cores")
set.seed(1, "L'Ecuyer-CMRG")

# SubSystemP1 Training and test sets.
trainingSetSubSysP1 = sample(seq_len(nrow(subSystemP1)), size =
floor(0.66*nrow(subSystemP1)))
xTrainSubSusP1 = subSystemP1[trainingSetSubSysP1,2:5]
yTrainSubSysP1 = as.numeric(subSystemP1[trainingSetSubSysP1, 1])

xTestSubSysP1 = subSystemP1[-trainingSetSubSysP1,2:5]
yTestSubSysP1 = as.numeric(subSystemP1[-trainingSetSubSysP1, 1])

# SubSystemP2 Training and test sets.
trainingSetSubSysP2 = sample(seq_len(nrow(subSystemP2)), size =
floor(0.66*nrow(subSystemP2)))
xTrainSubSusP2 = subSystemP2[trainingSetSubSysP2,2:8]
yTrainSubSysP2 = as.numeric(subSystemP2[trainingSetSubSysP2, 1])

xTestSubSysP2 = subSystemP2[-trainingSetSubSysP2,2:8]
yTestSubSysP2 = as.numeric(subSystemP2[-trainingSetSubSysP2, 1])

# SubSystemP3 Training and test sets.
trainingSetSubSysP3 = sample(seq_len(nrow(subSystemP3)), size =
floor(0.66*nrow(subSystemP3)))
xTrainSubSusP3 = subSystemP3[trainingSetSubSysP3,2:9]
yTrainSubSysP3 = as.numeric(subSystemP3[trainingSetSubSysP3, 1])

xTestSubSysP3 = subSystemP3[-trainingSetSubSysP3,2:9]
yTestSubSysP3 = as.numeric(subSystemP3[-trainingSetSubSysP3, 1])


# SubSystemP4 Training and test sets.
trainingSetSubSysP4 = sample(seq_len(nrow(subSystemP4)), size =
floor(0.66*nrow(subSystemP4)))
xTrainSubSusP4 = subSystemP4[trainingSetSubSysP4,2:6]
yTrainSubSysP4 = as.numeric(subSystemP4[trainingSetSubSysP4, 1])

xTestSubSysP4 = subSystemP4[-trainingSetSubSysP4,2:6]
yTestSubSysP4 = as.numeric(subSystemP4[-trainingSetSubSysP4, 1])

# SubSystemP5 Training and test sets.
trainingSetSubSysP5 = sample(seq_len(nrow(subSystemP5)), size =
```

```
floor(0.66*nrow(subSystemP5)))
xTrainSubSusP5 = subSystemP5[trainingSetSubSysP5,2:12]
yTrainSubSysP5 = as.numeric(subSystemP5[trainingSetSubSysP5, 1])

xTestSubSysP5 = subSystemP5[-trainingSetSubSysP5,2:12]
yTestSubSysP5 = as.numeric(subSystemP5[-trainingSetSubSysP5, 1])


# SubSystemP6 Training and test sets.
trainingSetSubSysP6 = sample(seq_len(nrow(subSystemP6)), size =
floor(0.66*nrow(subSystemP6)))
xTrainSubSusP6 = subSystemP6[trainingSetSubSysP6,2:3]
yTrainSubSysP6 = as.numeric(subSystemP6[trainingSetSubSysP6, 1])

xTestSubSysP6 = subSystemP6[-trainingSetSubSysP6,2:3]
yTestSubSysP6 = as.numeric(subSystemP6[-trainingSetSubSysP6, 1])


# Cross fold validation risk calculations for each subsection
system.time({
 CFVSubSysP1 = CV.SuperLearner(yTrainSubSysP1, xTrainSubSusP1, V = 10, parallel =
"multicore", family = binomial(), SL.library = algorithmList)
})

system.time({
CFVSubSysP2 = CV.SuperLearner(yTrainSubSysP2, xTrainSubSusP2, V = 10, parallel =
"multicore", family = binomial(), SL.library = algorithmList)
})

system.time({
CFVSubSysP3 = CV.SuperLearner(yTrainSubSysP3, xTrainSubSusP3, V = 10, parallel =
"multicore", family = binomial(), SL.library = algorithmList)
})

system.time({
CFVSubSysP4 = CV.SuperLearner(yTrainSubSysP4, xTrainSubSusP4, V = 10, parallel =
"multicore", family = binomial(), SL.library = algorithmList)
})

system.time({
CFVSubSysP5 = CV.SuperLearner(yTrainSubSysP5, xTrainSubSusP5, V = 10, parallel =
"multicore", family = binomial(), SL.library = algorithmList)
})

system.time({
CFVSubSysP6 = CV.SuperLearner(yTrainSubSysP6, xTrainSubSusP6, V = 10, parallel =
"multicore", family = binomial(), SL.library = algorithmList)
})
```

```
# SubsystemP1 modeling
modelSubSysP1 = SuperLearner(Y = yTrainSubSysP1, X = xTrainSubSusP1, family = binomial(),
SL.library = algorithmList)
predictionSubSysP1 = predict.SuperLearner(modelSubSysP1, newdata = xTestSubSysP1)
predictedResultSubSysP1 = as.numeric(ifelse(predictionSubSysP1$pred>=0.5,1,0))
confMatrixSubSysP1 = confusionMatrix(as.factor(yTestSubSysP1),
as.factor(predictedResultSubSysP1))


# SybSystemP2 modeling
modelSubSysP2 = SuperLearner(Y = yTrainSubSysP2, X = xTrainSubSusP2, family = binomial(),
SL.library = algorithmList)
predictionSubSysP2 = predict.SuperLearner(modelSubSysP2, newdata = xTestSubSysP2)
predictedResultSubSysP2 = as.numeric(ifelse(predictionSubSysP2$pred>=0.5,1,0))
confMatrixSubSysP2 = confusionMatrix(as.factor(yTestSubSysP2),
as.factor(predictedResultSubSysP2))


# SybSystemP3 modeling
modelSubSysP3 = SuperLearner(Y = yTrainSubSysP3, X = xTrainSubSusP3, family = binomial(),
SL.library = algorithmList)
predictionSubSysP3 = predict.SuperLearner(modelSubSysP3, newdata = xTestSubSysP3)
predictedResultSubSysP3 = as.numeric(ifelse(predictionSubSysP3$pred>=0.5,1,0))
confMatrixSubSysP3 = confusionMatrix(as.factor(yTestSubSysP3),
as.factor(predictedResultSubSysP3))

# SybSystemP4 modeling
modelSubSysP4 = SuperLearner(Y = yTrainSubSysP4, X = xTrainSubSusP4, family = binomial(),
SL.library = algorithmList)
predictionSubSysP4 = predict.SuperLearner(modelSubSysP4, newdata = xTestSubSysP4)
predictedResultSubSysP4 = as.numeric(ifelse(predictionSubSysP4$pred>=0.5,1,0))
confMatrixSubSysP4 = confusionMatrix(as.factor(yTestSubSysP4),
as.factor(predictedResultSubSysP4))

# SybSystemP5 modeling
modelSubSysP5 = SuperLearner(Y = yTrainSubSysP5, X = xTrainSubSusP5, family = binomial(),
SL.library = algorithmList)
predictionSubSysP5 = predict.SuperLearner(modelSubSysP5, newdata = xTestSubSysP5)
predictedResultSubSysP5 = as.numeric(ifelse(predictionSubSysP5$pred>=0.5,1,0))
confMatrixSubSysP5 = confusionMatrix(as.factor(yTestSubSysP5),
as.factor(predictedResultSubSysP5))


# SubsystemP6 modeling
modelSubSysP6 = SuperLearner(Y = yTrainSubSysP6, X = xTrainSubSusP6, family = binomial(),
SL.library = algorithmList)
```

```
predictionSubSysP6 = predict.SuperLearner(modelSubSysP6, newdata = xTestSubSysP6)
predictedResultSubSysP6 = as.numeric(ifelse(predictionSubSysP6$pred>=0.5,1,0))
# something is wrong here
# confMatrixSubSysP6 = confusionMatrix(as.factor(yTestSubSysP6),
as.factor(predictedResultSubSysP6))



# SubSystemP1 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP1 = CV.SuperLearner(Y = yTrainSubSysP1, X = xTrainSubSusP1, family =
binomial(),
                         # For a real analysis we would use V = 10.
                         V = 10,
                         parallel = "multicore",
                         method = "method.AUC",
                         SL.library = algorithmList)
})


# SubSystemP2 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP2 = CV.SuperLearner(Y = yTrainSubSysP2, X = xTrainSubSusP2, family =
binomial(),
                                 # For a real analysis we would use V = 10.
                                 V = 10,
                                 parallel = "multicore",
                                 method = "method.AUC",
                                 SL.library = algorithmList)
})

# SubSystemP3 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP3 = CV.SuperLearner(Y = yTrainSubSysP3, X = xTrainSubSusP3, family =
binomial(),
                                 # For a real analysis we would use V = 10.
                                 V = 10,
                                 parallel = "multicore",
                                 method = "method.AUC",
                                 SL.library = algorithmList)
})

# SubSystemP4 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP4 = CV.SuperLearner(Y = yTrainSubSysP4, X = xTrainSubSusP4, family =
binomial(),
                                 # For a real analysis we would use V = 10.
                                 V = 10,
```

```
                                parallel = "multicore",
                                method = "method.AUC",
                                SL.library = algorithmList)
})


# SubSystemP1 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP4 = CV.SuperLearner(Y = yTrainSubSysP4, X = xTrainSubSusP4, family =
binomial(),
                                # For a real analysis we would use V = 10.
                                V = 10,
                                parallel = "multicore",
                                method = "method.AUC",
                                SL.library = algorithmList)
})


# SubSystemP1 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP5 = CV.SuperLearner(Y = yTrainSubSysP5, X = xTrainSubSusP5, family =
binomial(),
                                # For a real analysis we would use V = 10.
                                V = 10,
                                parallel = "multicore",
                                method = "method.AUC",
                                SL.library = algorithmList)
})

# SubSystemP1 CrossFold Validation for calculating Area Under Curve
system.time({
 cv_slSubSysP6 = CV.SuperLearner(Y = yTrainSubSysP6, X = xTrainSubSusP6, family =
binomial(),
                                # For a real analysis we would use V = 10.
                                V = 10,
                                parallel = "multicore",
                                method = "method.AUC",
                                SL.library = algorithmList)
})
```

Adem Malone
Tho Nguyen
Matthew Stroble

K2 Bayesian Network using Weka on Complete System

Utilizing Weka Explorer, we used a new dataset that was created from the original dataset that was converted to nominal using R based on information gain. This new nominal data set was imported into Weka and the remaining numeric data for is_attack, pumps and motorised valves were converted to nominal data using the weka filter NumarecToNominal Filter. The Weka classification tool was used and our chosen classifier was BayesNet. BayesNet was configured to run with 3 parents, using K2 search algorithm. Weka was then run using “Use Training Method” for its test options and is_attack was used as the decision attribute. High resolution images are available on request

BayesNet K2 Full System 3 max Parents

=== Run information ===

Scheme: weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 3 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5
Relation: updateDataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last
Instances: 14996
Attributes: 38
is_attack
P101
P203
P205
P301
P401
P601
MV301
MV302
MV304
UV401
MV101
MV201
MV303
FIT101
LIT101
AIT202

AIT203
FIT201
DPIT301
FIT301
LIT301
AIT402
FIT401
LIT401
AIT501
AIT502
AIT503
AIT504
FIT501
FIT502
FIT503
FIT504
PIT501
PIT502
PIT503
FIT601
AIT201
Test mode: evaluate on training data

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

Bayes Network Classifier
not using ADTree
#attributes=38 #classindex=0
Network structure (nodes followed by parents)
is_attack(2):
P101(2): is_attack
P203(2): is_attack P101
P205(2): is_attack P203
P301(2): is_attack P101 P203
P401(2): is_attack
P601(2): is_attack P203 P301
MV301(3): is_attack
MV302(3): is_attack P301 P101
MV304(3): is_attack MV302 P301
UV401(2): is_attack P301 P101
MV101(3): is_attack MV302 P203

MV201(3): is_attack P101 P203
MV303(3): is_attack MV304 MV301
FIT101(2): is_attack MV101 P101
LIT101(2): is_attack P203 MV101
AIT202(2): is_attack MV302 P203
AIT203(2): is_attack AIT202 MV201
FIT201(2): is_attack P203 P301
DPIT301(2): is_attack P301 AIT202
FIT301(2): is_attack P301 MV201
LIT301(2): is_attack FIT201 MV302
AIT402(2): is_attack LIT301 AIT202
FIT401(2): is_attack AIT402 UV401
LIT401(2): is_attack AIT203
AIT501(2): is_attack AIT402 UV401
AIT502(2): is_attack AIT501 DPIT301
AIT503(2): is_attack MV101 AIT501
AIT504(2): is_attack AIT502 MV302
FIT501(2): is_attack AIT501 FIT401
FIT502(2): is_attack AIT501 AIT402
FIT503(2): is_attack AIT402 AIT502
FIT504(2): is_attack FIT501 FIT503
PIT501(2): is_attack AIT501 FIT503
PIT502(2): is_attack AIT402 MV302
PIT503(2): is_attack PIT501 AIT503
FIT601(2): is_attack P203 MV101
AIT201(2): is_attack LIT301 AIT502
LogScore Bayes: -136483.56006778337
LogScore BDeu: -137509.12664243678
LogScore MDL: -138058.32646374448
LogScore ENTROPY: -136207.3352490072
LogScore AIC: -136592.3352490072

Time taken to build model: 0.31 seconds

=== Evaluation on training set ===

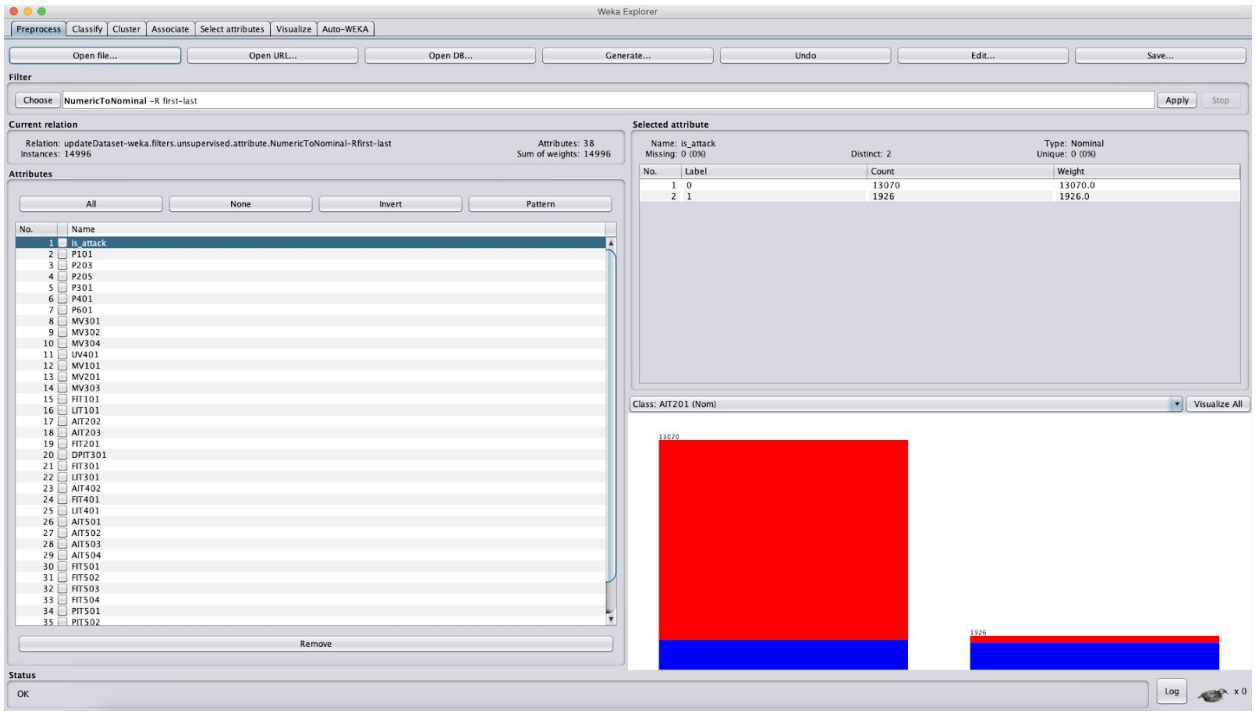
Time taken to test model on training data: 0.12 seconds

=== Summary ===

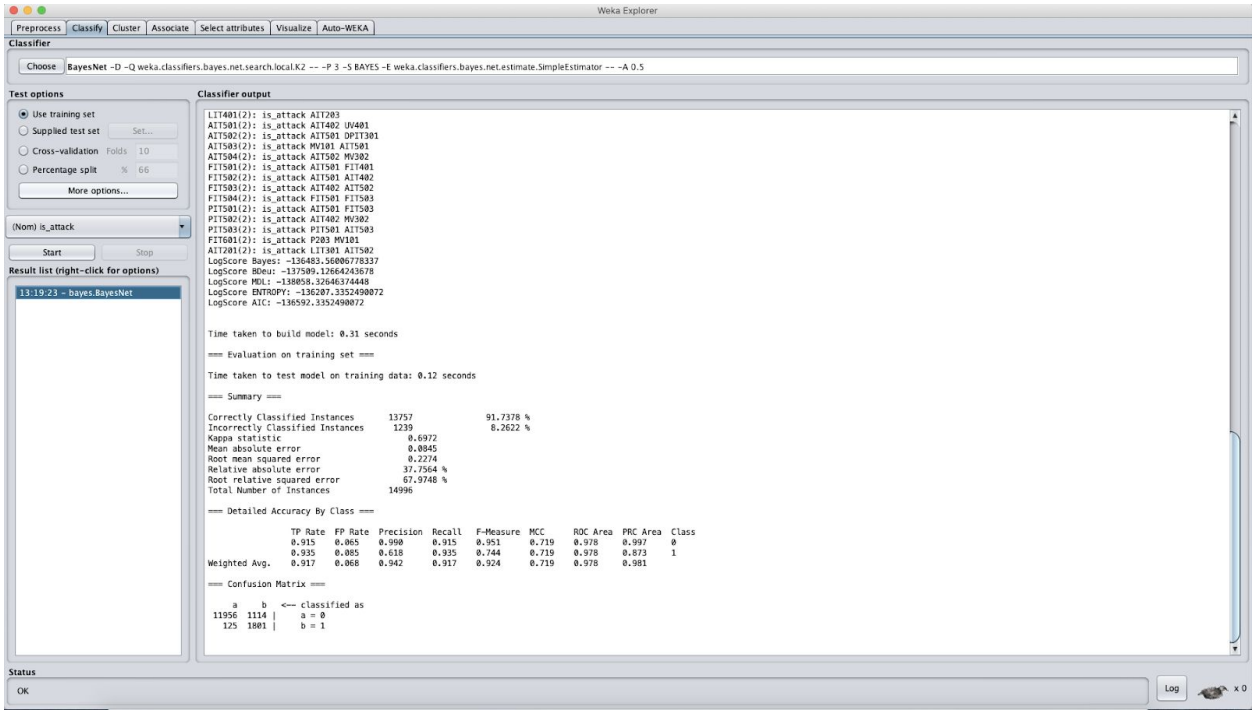
```
Correctly Classified Instances       13757               91.7378 %
Incorrectly Classified Instances      1239                8.2622 %
Kappa statistic                          0.6972
Mean absolute error                      0.0845
Root mean squared error                  0.2274
Relative absolute error                 37.7564 %
Root relative squared error             67.9748 %
Total Number of Instances            14996

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.915    0.065    0.990      0.915    0.951      0.719    0.978     0.997     0
               0.935    0.085    0.618      0.935    0.744      0.719    0.978     0.873     1
Weighted Avg.  0.917    0.068    0.942      0.917    0.924      0.719    0.978     0.981

=== Confusion Matrix ===

    a     b   <-- classified as
 11956  1114 |     a = 0
   125  1801 |     b = 1
```

Weka preprocessing

Adem Malone
Tho Nguyen
Matthew Stroble

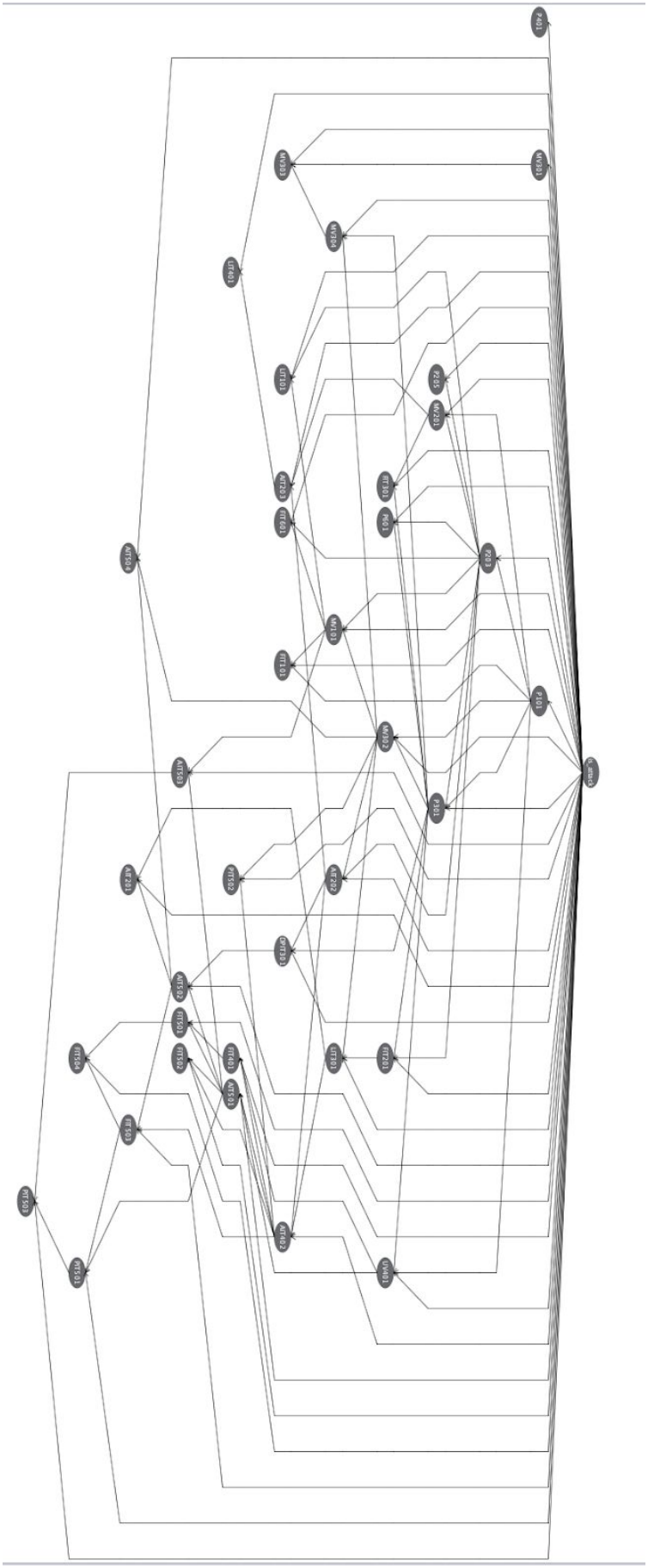


Weka BayesNet Classify

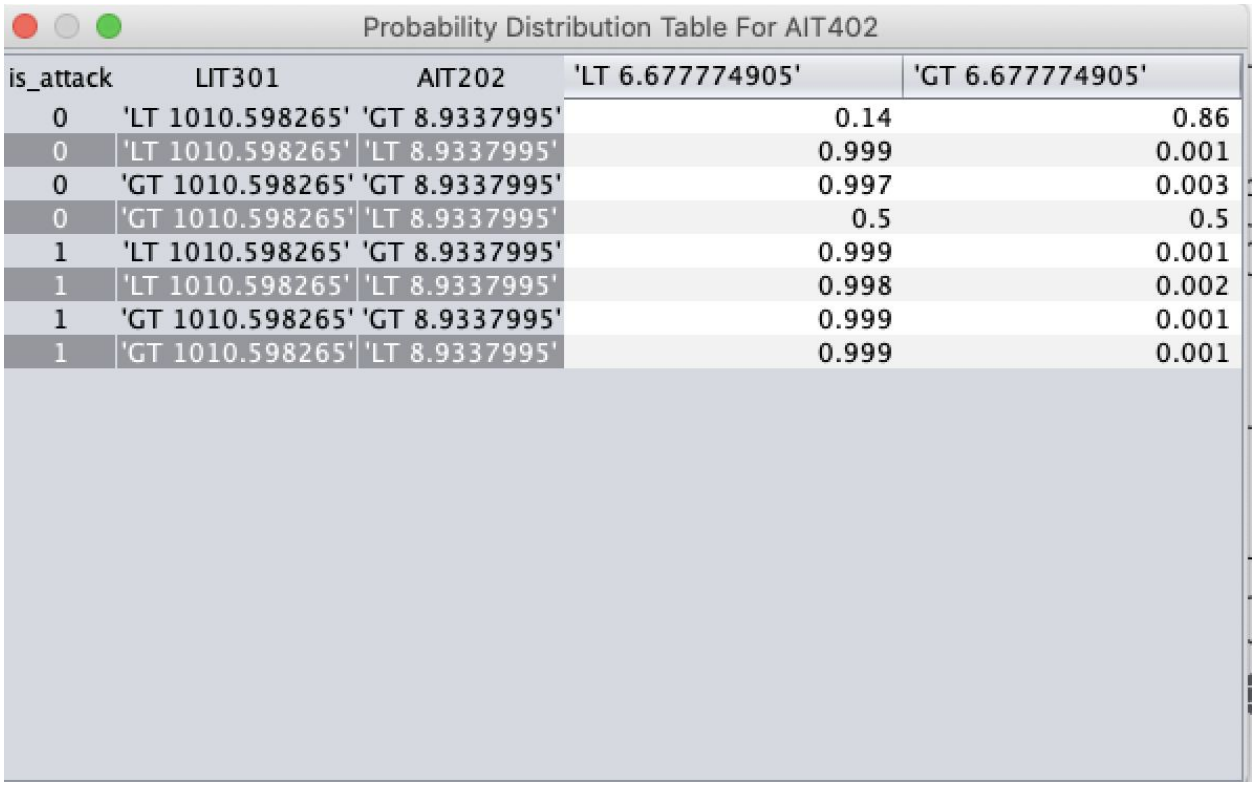


Weka Bayesian Network K2 3 Parents Next page

Adem Malone
Tho Nguyen
Matthew Stroble



BayesNet K2 Node AIT402 with connections



is_attack	LIT301	AIT202	'LT 6.677774905'	'GT 6.677774905'
0	'LT 1010.598265'	'GT 8.9337995'	0.14	0.86
0	'LT 1010.598265'	'LT 8.9337995'	0.999	0.001
0	'GT 1010.598265'	'GT 8.9337995'	0.997	0.003
0	'GT 1010.598265'	'LT 8.9337995'	0.5	0.5
1	'LT 1010.598265'	'GT 8.9337995'	0.999	0.001
1	'LT 1010.598265'	'LT 8.9337995'	0.998	0.002
1	'GT 1010.598265'	'GT 8.9337995'	0.999	0.001
1	'GT 1010.598265'	'LT 8.9337995'	0.999	0.001

BayesNet K2 Full System 36 max parents

Scheme: weka.classifiers.bayes.BayesNet -D -Q weka.classifiers.bayes.net.search.local.K2 -- -P 37 -S BAYES -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 0.5
Relation: updateDataset-weka.filters.unsupervised.attribute.NumericToNominal-Rfirst-last
Instances: 14996
Attributes: 38

- is_attack
- P101
- P203
- P205
- P301
- P401
- P601
- MV301
- MV302
- MV304
- UV401
- MV101
- MV201
- MV303
- FIT101
- LIT101
- AIT202
- AIT203
- FIT201
- DPIT301
- FIT301
- LIT301
- AIT402
- FIT401
- LIT401
- AIT501
- AIT502
- AIT503
- AIT504
- FIT501
- FIT502
- FIT503
- FIT504
- PIT501

PIT502
PIT503
FIT601
AIT201
Test mode: evaluate on training data

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

Bayes Network Classifier
not using ADTree
#attributes=38 #classindex=0
Network structure (nodes followed by parents)
is_attack(2):
P101(2): is_attack
P203(2): is_attack P101
P205(2): is_attack P203
P301(2): is_attack P101 P203 P205
P401(2): is_attack
P601(2): is_attack P203 P301
MV301(3): is_attack
MV302(3): is_attack P301 P101 P601 P401
MV304(3): is_attack MV302 P301 P101
UV401(2): is_attack P301 P101 MV304 P401
MV101(3): is_attack MV302 P203 MV304 MV301
MV201(3): is_attack P101 P203
MV303(3): is_attack MV304 MV301 P301
FIT101(2): is_attack MV101 P101 MV201
LIT101(2): is_attack P203 MV101 MV302 P601 FIT101 P301
AIT202(2): is_attack MV302 P203 MV303 P301 UV401
AIT203(2): is_attack AIT202 MV201 LIT101 MV101 P301 P601 P203 UV401
FIT201(2): is_attack P203 P301 LIT101 AIT203 AIT202 MV101
DPIT301(2): is_attack P301 AIT202 P203 MV304 P601 AIT203 FIT201 LIT101
FIT301(2): is_attack P301 MV201 UV401 MV304 AIT203 MV303 DPIT301
LIT301(2): is_attack FIT201 MV302 P301 AIT203 P101 DPIT301 MV101 UV401 LIT101 MV304 FIT101
AIT402(2): is_attack LIT301 AIT202 MV201 DPIT301 MV302 MV101 AIT203 P601 LIT101 FIT201
FIT401(2): is_attack AIT402 UV401 DPIT301 AIT203 P203 P301 LIT301 MV101 LIT101 MV302 P601 FIT101 MV304 MV301 FIT301 P101
LIT401(2): is_attack AIT203

AIT501(2): is_attack AIT402 UV401 FIT201 FIT401 P301 FIT301 MV101 AIT202 DPIT301 MV304 LIT301 FIT101
AIT502(2): is_attack AIT501 DPIT301 AIT402 P203 P301 AIT203 UV401 MV101 FIT401 FIT301 LIT301 MV304
AIT503(2): is_attack MV101 AIT501 AIT502 AIT203 P301 FIT401 MV201 DPIT301 LIT101 AIT402 FIT301 LIT401 P601 FIT101 MV303 MV301
AIT504(2): is_attack AIT502 MV302 LIT301 AIT203 AIT501 P101 MV101 AIT503 FIT301 LIT101 AIT402 FIT401 LIT401 FIT101 P601
FIT501(2): is_attack AIT501 FIT401 AIT503 AIT402 AIT502 FIT301 LIT101 DPIT301 AIT504 P101 MV101 UV401 AIT202 P601
FIT502(2): is_attack AIT501 AIT402 FIT301 AIT502 MV101 AIT503 LIT101 AIT504 P301 P101 AIT203 UV401 LIT301 MV302 P601 LIT401 MV301 AIT202
FIT503(2): is_attack AIT402 AIT502 UV401 FIT201 MV302 AIT503 AIT504 P601 FIT301
FIT504(2): is_attack FIT501 FIT503 AIT504 FIT401 MV101 MV201 DPIT301 AIT503 AIT502 FIT301 LIT101 AIT501 AIT402 LIT401 P203 AIT203 LIT301 AIT202
PIT501(2): is_attack AIT501 FIT503 AIT502 P301 AIT402 P101 AIT503 AIT504 P601
PIT502(2): is_attack AIT402 MV302 PIT501 P601 AIT503 AIT202 AIT502 LIT301 LIT101 FIT301 AIT501
PIT503(2): is_attack PIT501 AIT503 PIT502 AIT504 P203 FIT501 AIT402 MV101 P601
FIT601(2): is_attack P203 MV101 AIT502 P301 AIT503 AIT402 PIT502 FIT502 LIT401 AIT203 AIT504 DPIT301 FIT301 MV304 PIT501 LIT301 LIT101 AIT202 MV201
AIT201(2): is_attack LIT301 AIT502 AIT402 PIT502 FIT201 P301 AIT503 AIT202
LogScore Bayes: -107549.3804655661
LogScore BDeu: -1.1811591887389927E8
LogScore MDL: -4.457444504403842E7
LogScore ENTROPY: -5708047.88342283
LogScore AIC: -1.379212888342461E7

Time taken to build model: 120.24 seconds

=== Evaluation on training set ===

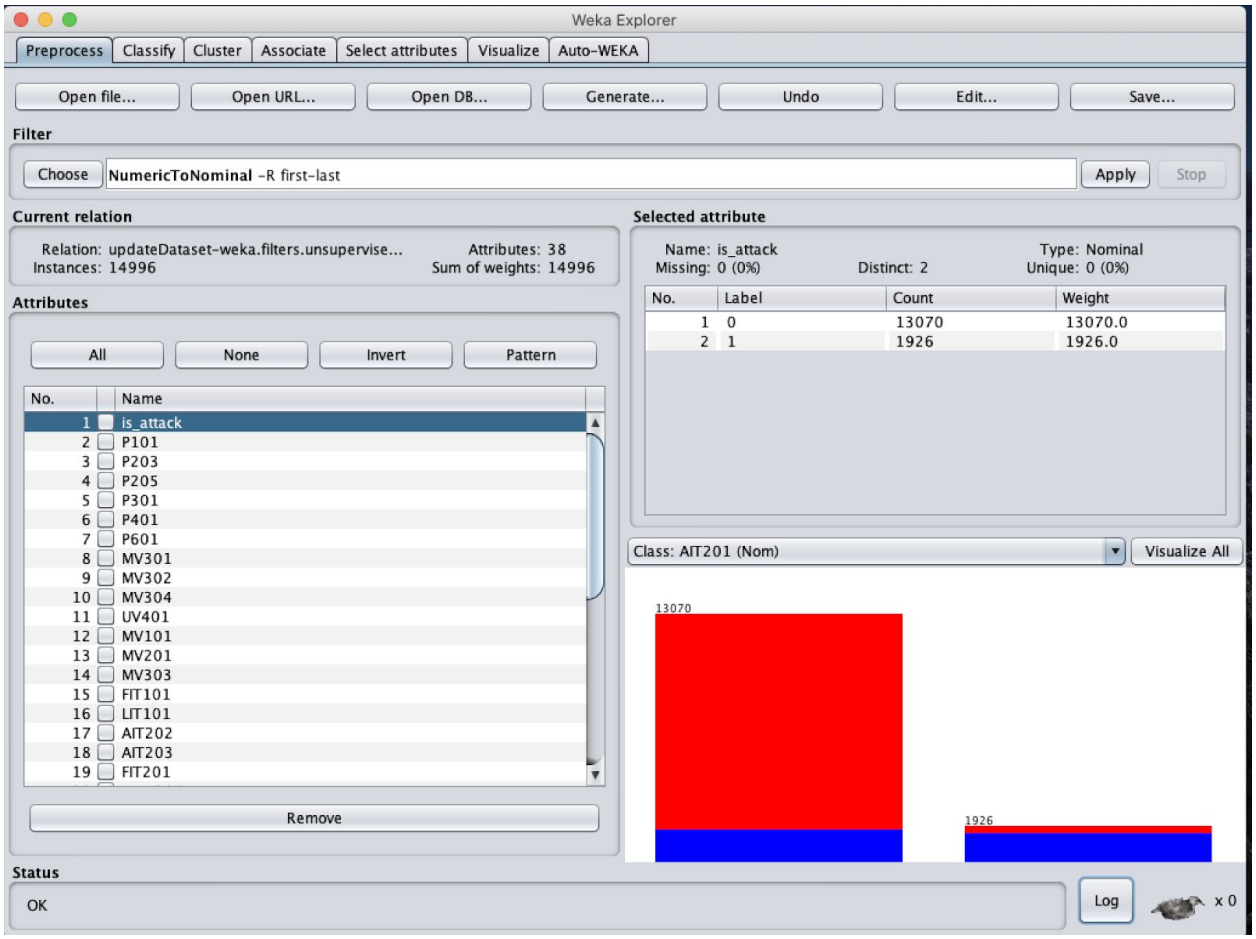
Time taken to test model on training data: 0.15 seconds

=== Summary ===

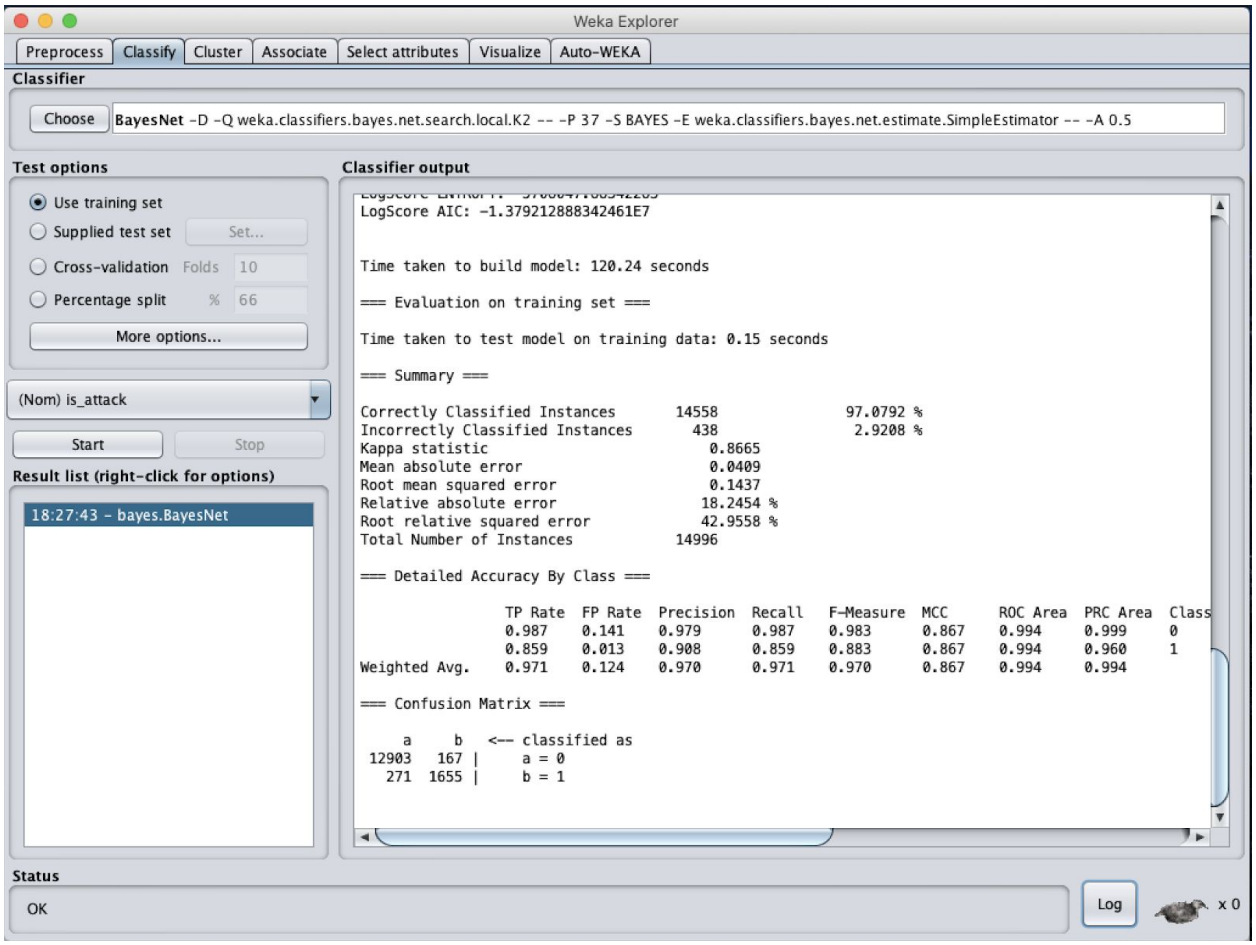
Correctly Classified Instances	14558	97.0792 %
Incorrectly Classified Instances	438	2.9208 %
Kappa statistic	0.8665	
Mean absolute error	0.0409	

```
Root mean squared error                  0.1437
Relative absolute error                 18.2454 %
Root relative squared error             42.9558 %
Total Number of Instances               14996

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.987    0.141    0.979      0.987    0.983      0.867    0.994     0.999     0
               0.859    0.013    0.908      0.859    0.883      0.867    0.994     0.960     1
Weighted Avg.  0.971    0.124    0.970      0.971    0.970      0.867    0.994     0.994

=== Confusion Matrix ===

     a     b   <-- classified as
 12903   167 |     a = 0
   271  1655 |     b = 1
```

Weka Explorer

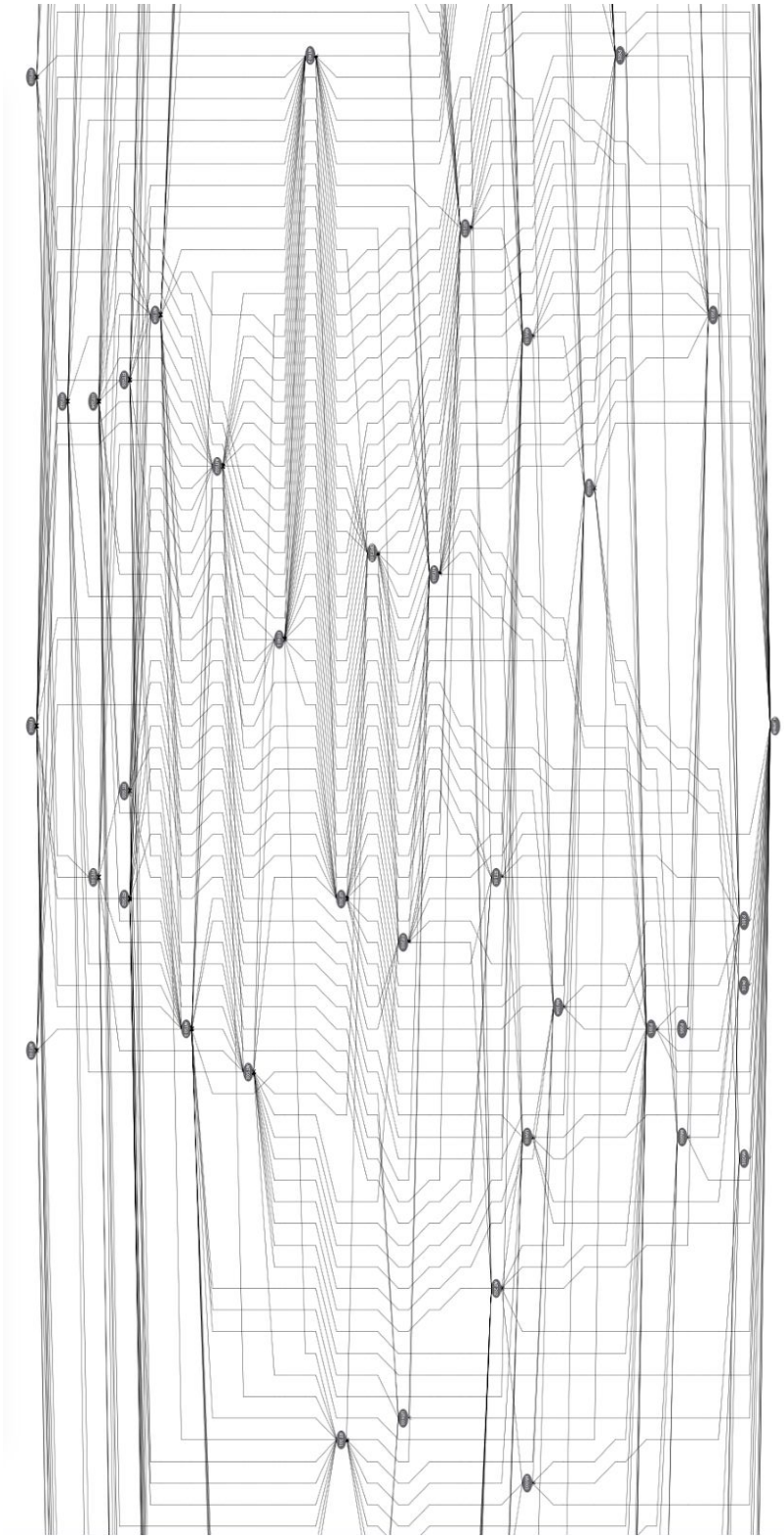


Weka Classify



Adem Malone
Tho Nguyen
Matthew Stroble

Weka Graph



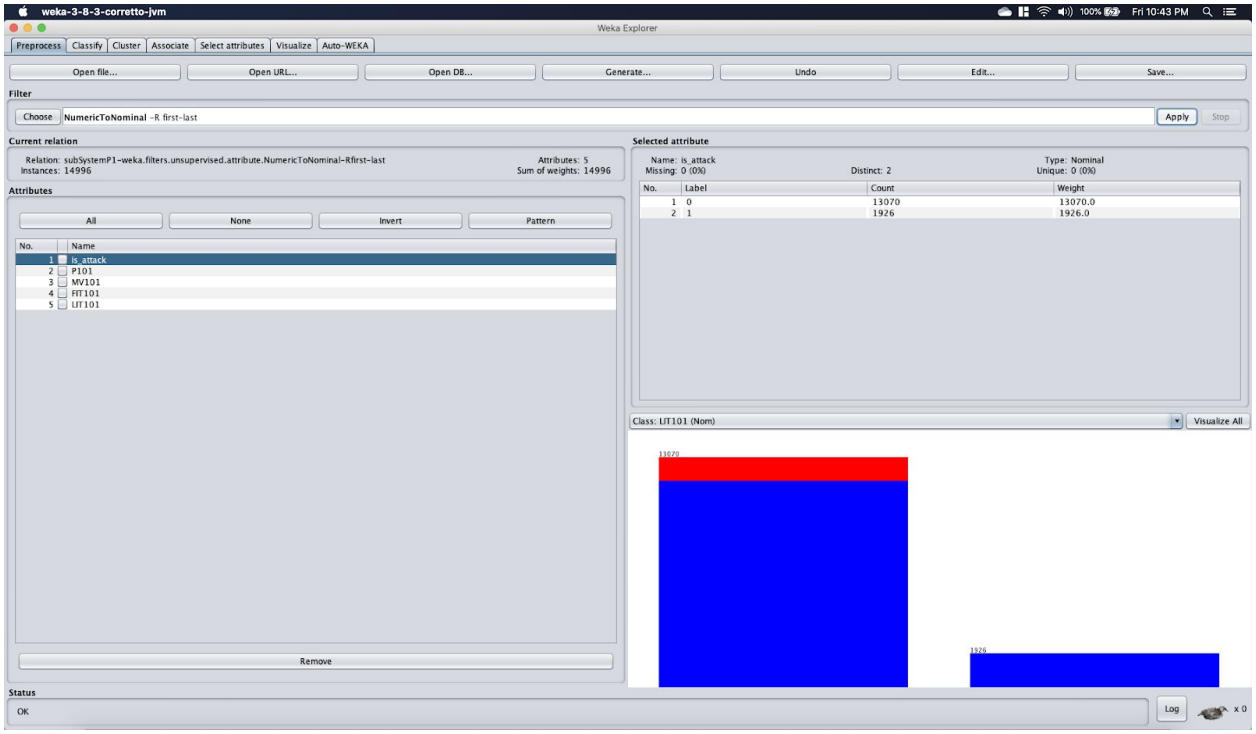
Adem Malone
Tho Nguyen
Matthew Stroble

K2 Bayesian Network using Weka on Subsystem Sets

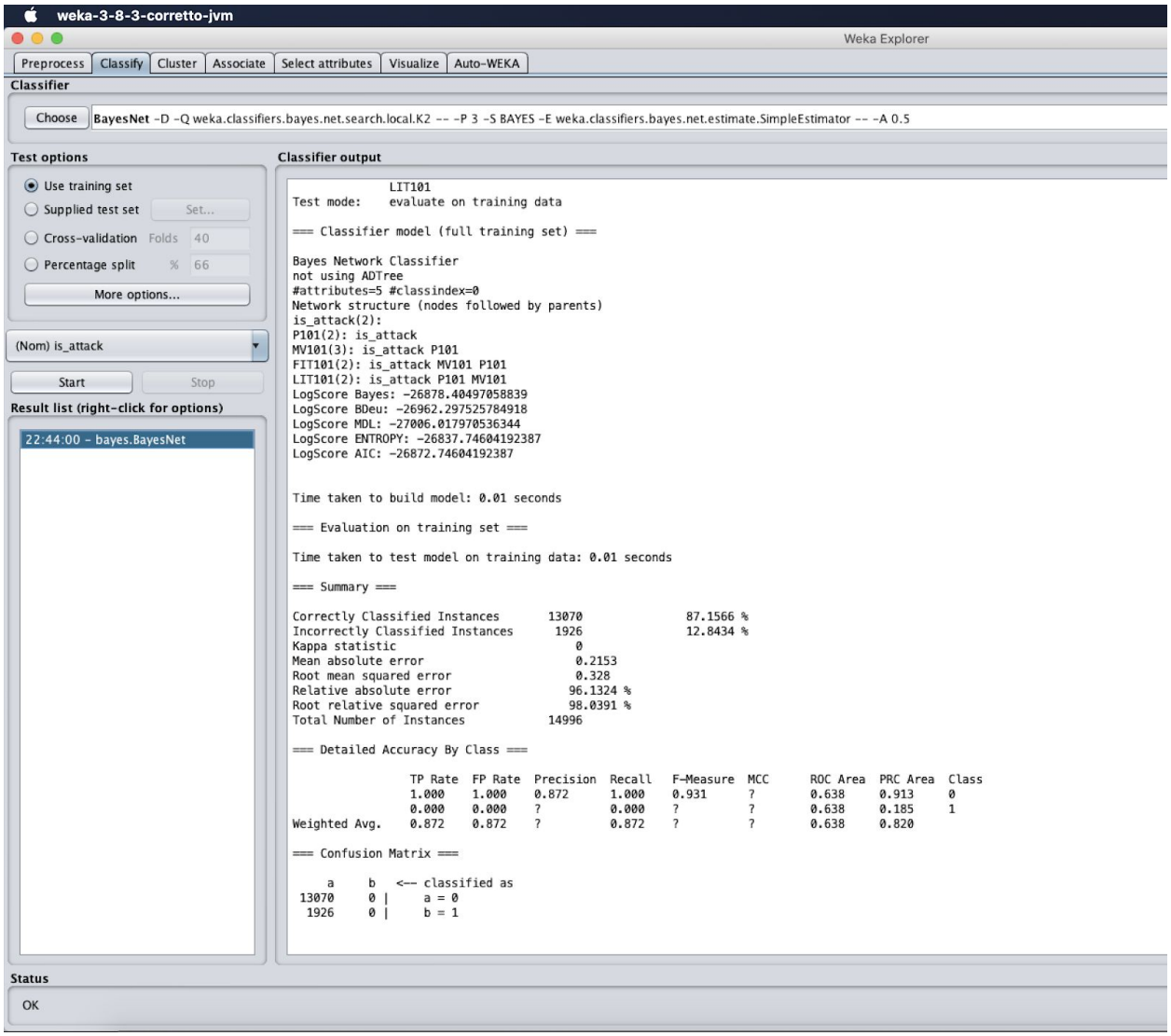
Utilizing Weka, using the Full System nominal dataset we used R to break up the full system nominal data set in R into six subsystem specific data sets for the subsystems P1, P2 P3, P4, P5, and P6. The six new subsystem specific dataset each have the attribute is_attack prepended to the first column. Each of the six nominal datasets were imported into Weka during that subsystems bayesian network run, and any remaining numeric data for is_attack, and columns that still contained numeric data were converted to nominal using the weka filter NumarecToNominal. The Weka classification tool was used and our chosen classifier was BayesNet. BayesNet was configured to run with 3 parents, using K2 search algorithm. Weka was then run using “Use Training Method” for its test options and is_attack was used as the decision attribute for each subsystem dataset. High resolution images are available on request.

Subsystem P1

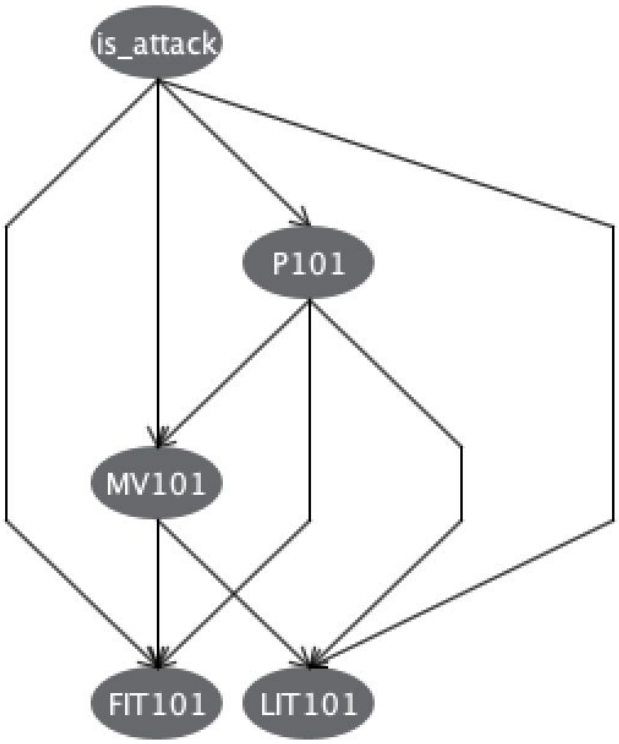
Weka Preprocessing



Weka Classify



Weka Bayesian Network K2 3 Parents



Weka Bayesian Network K2 Node LIT101

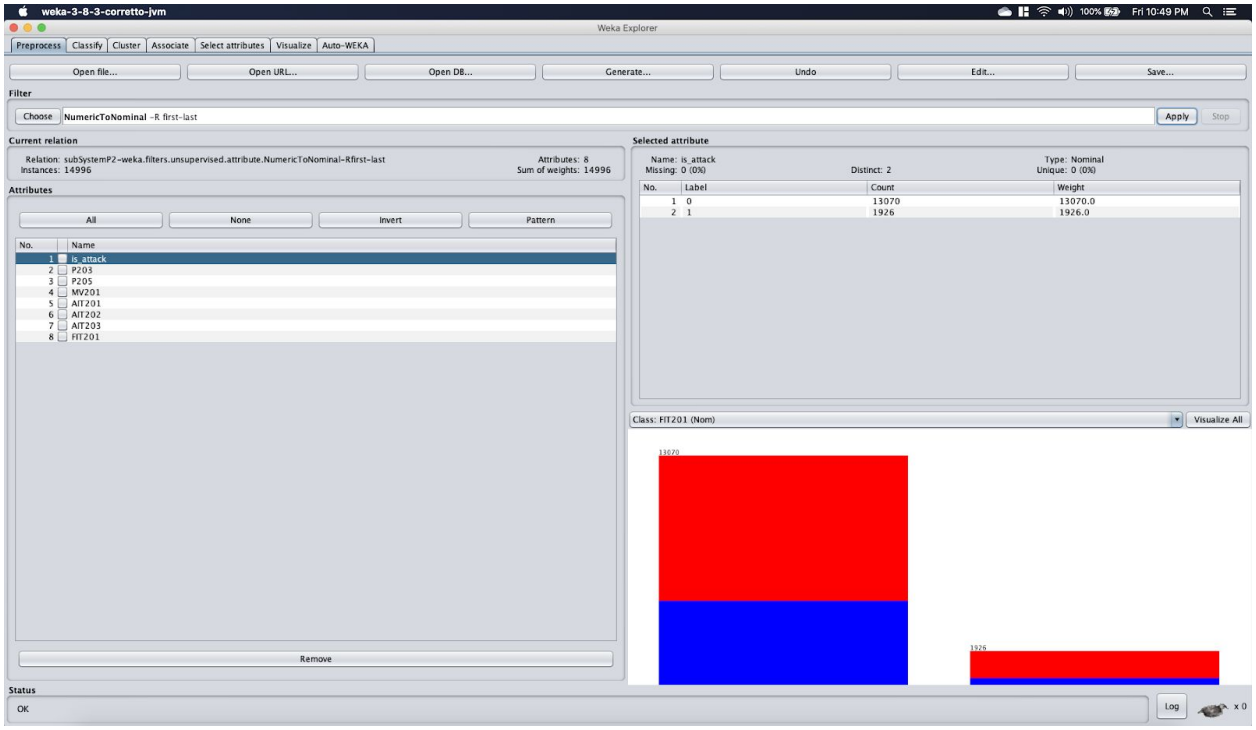
Probability Distribution Table For LIT101

is_attack	P101	MV101	'GT 545.1423'	'LT 545.1423'
0	1	0	0.986	0.014
0	1	1	1	0
0	1	2	0.964	0.036
0	2	0	0.007	0.993
0	2	1	0.85	0.15
0	2	2	0.409	0.591
1	1	0	0.95	0.05
1	1	1	1	0
1	1	2	0.998	0.002
1	2	0	0.5	0.5
1	2	1	0.999	0.001
1	2	2	0.992	0.008

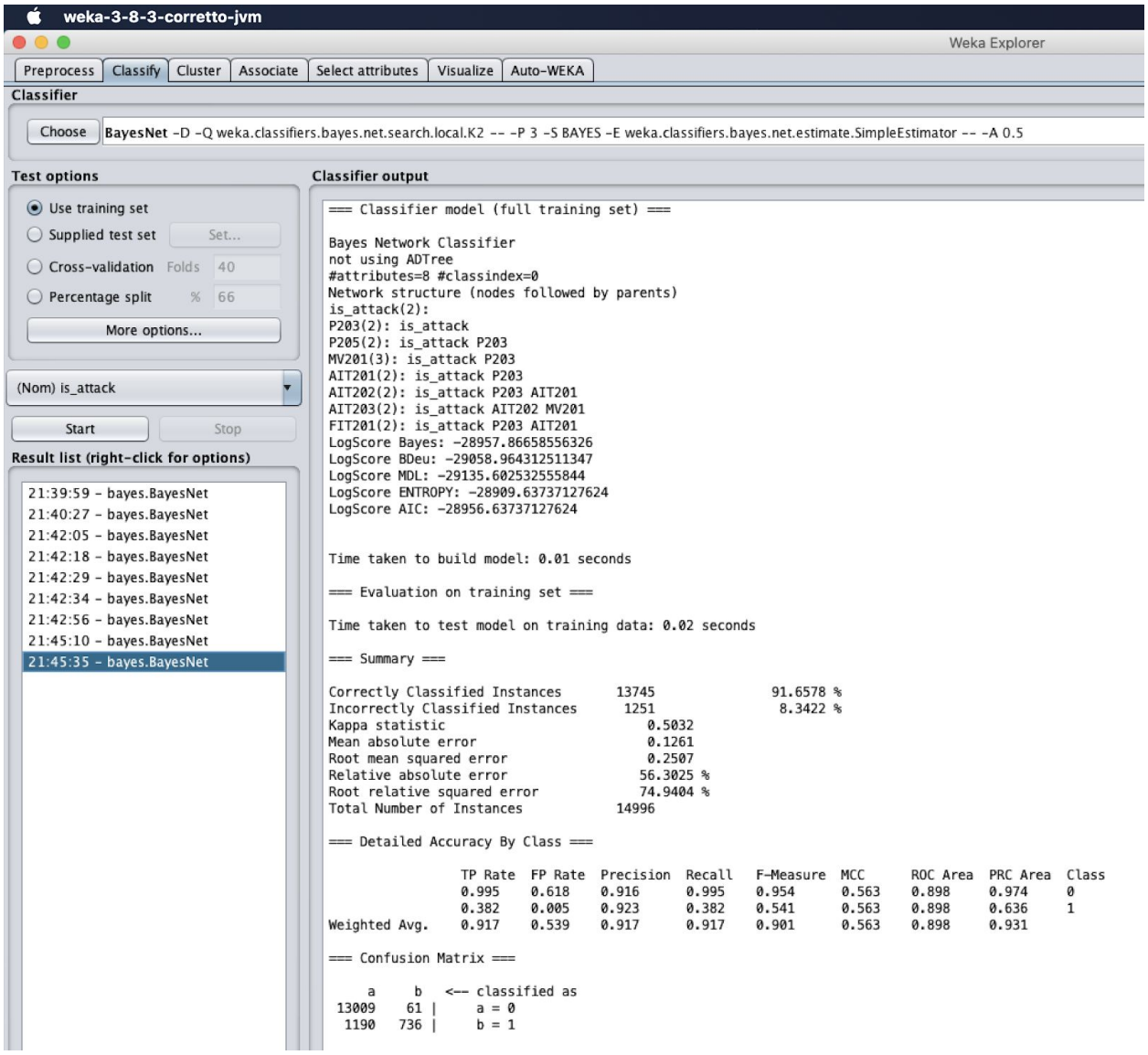
Adem Malone
Tho Nguyen
Matthew Stroble

Subsystem P2

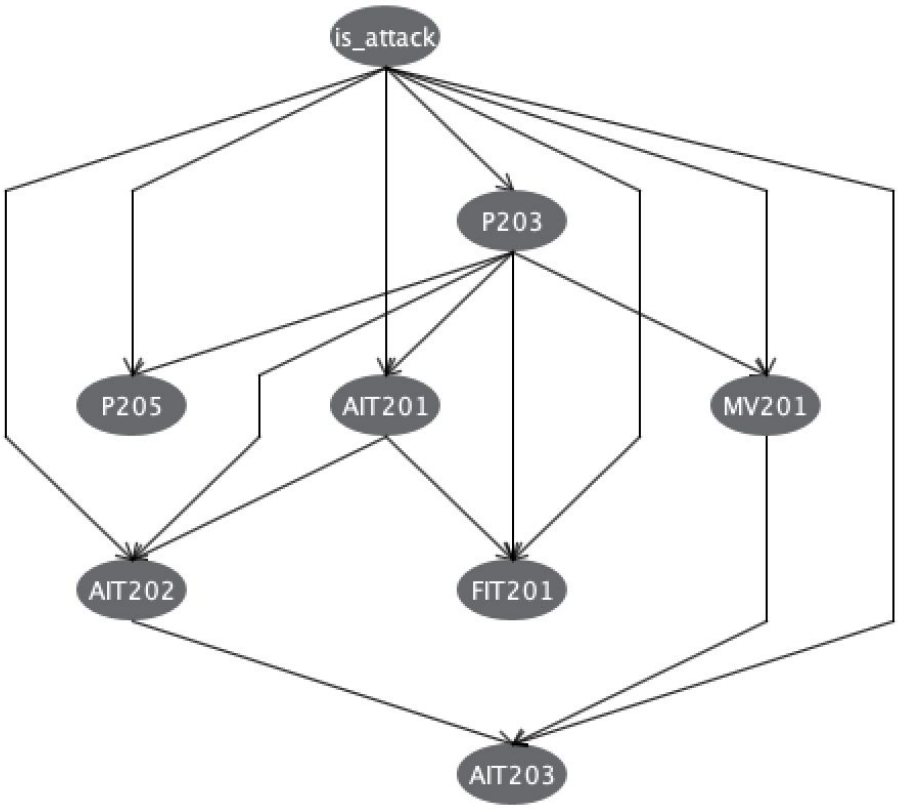
Weka Preprocessing



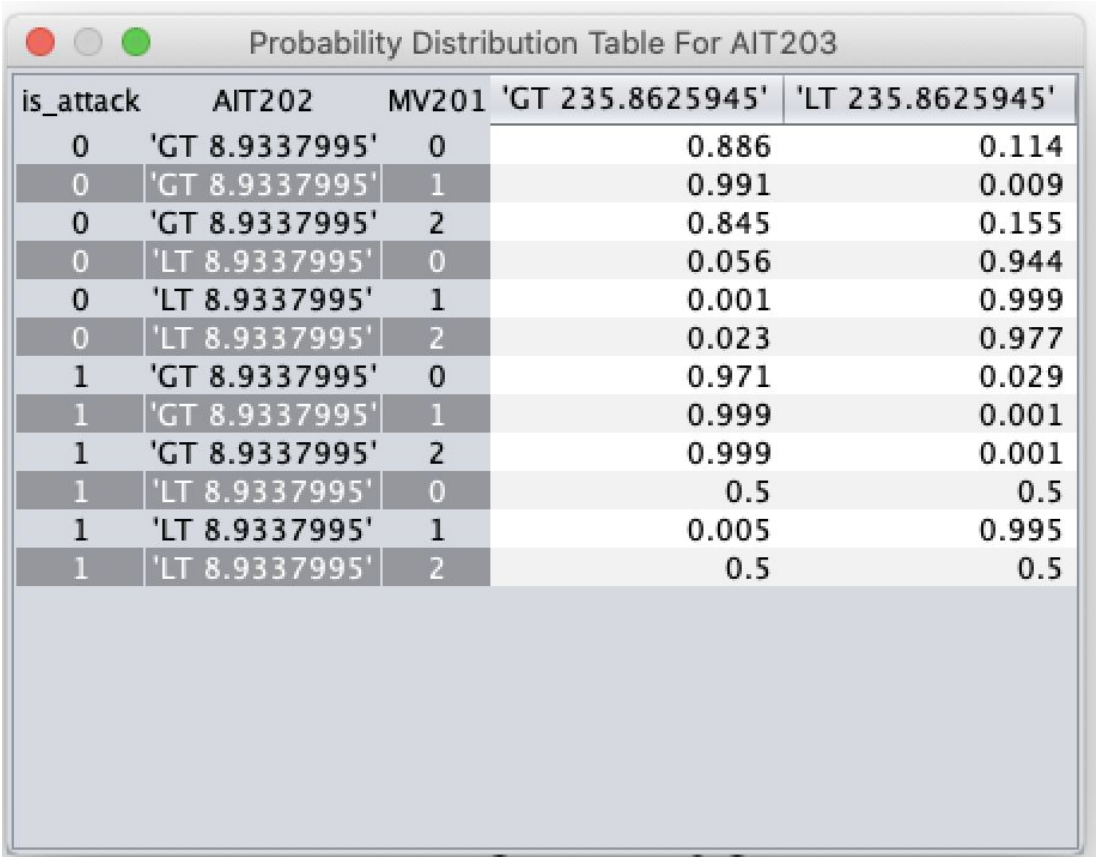
Weka Classify



Weka Bayesian Network K2 3 Parents



Weka Bayesian Network K2 Node AIT203

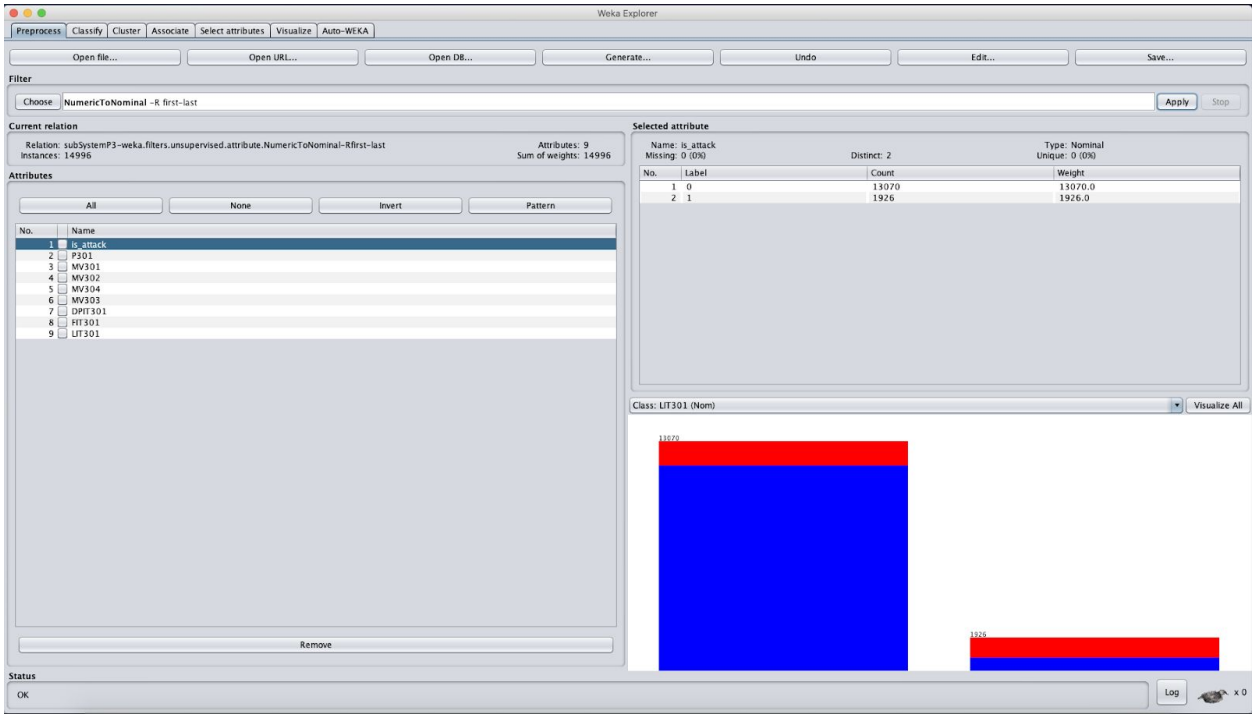


is_attack	AIT202	MV201	'GT 235.8625945'	'LT 235.8625945'
0	'GT 8.9337995'	0	0.886	0.114
0	'GT 8.9337995'	1	0.991	0.009
0	'GT 8.9337995'	2	0.845	0.155
0	'LT 8.9337995'	0	0.056	0.944
0	'LT 8.9337995'	1	0.001	0.999
0	'LT 8.9337995'	2	0.023	0.977
1	'GT 8.9337995'	0	0.971	0.029
1	'GT 8.9337995'	1	0.999	0.001
1	'GT 8.9337995'	2	0.999	0.001
1	'LT 8.9337995'	0	0.5	0.5
1	'LT 8.9337995'	1	0.005	0.995
1	'LT 8.9337995'	2	0.5	0.5

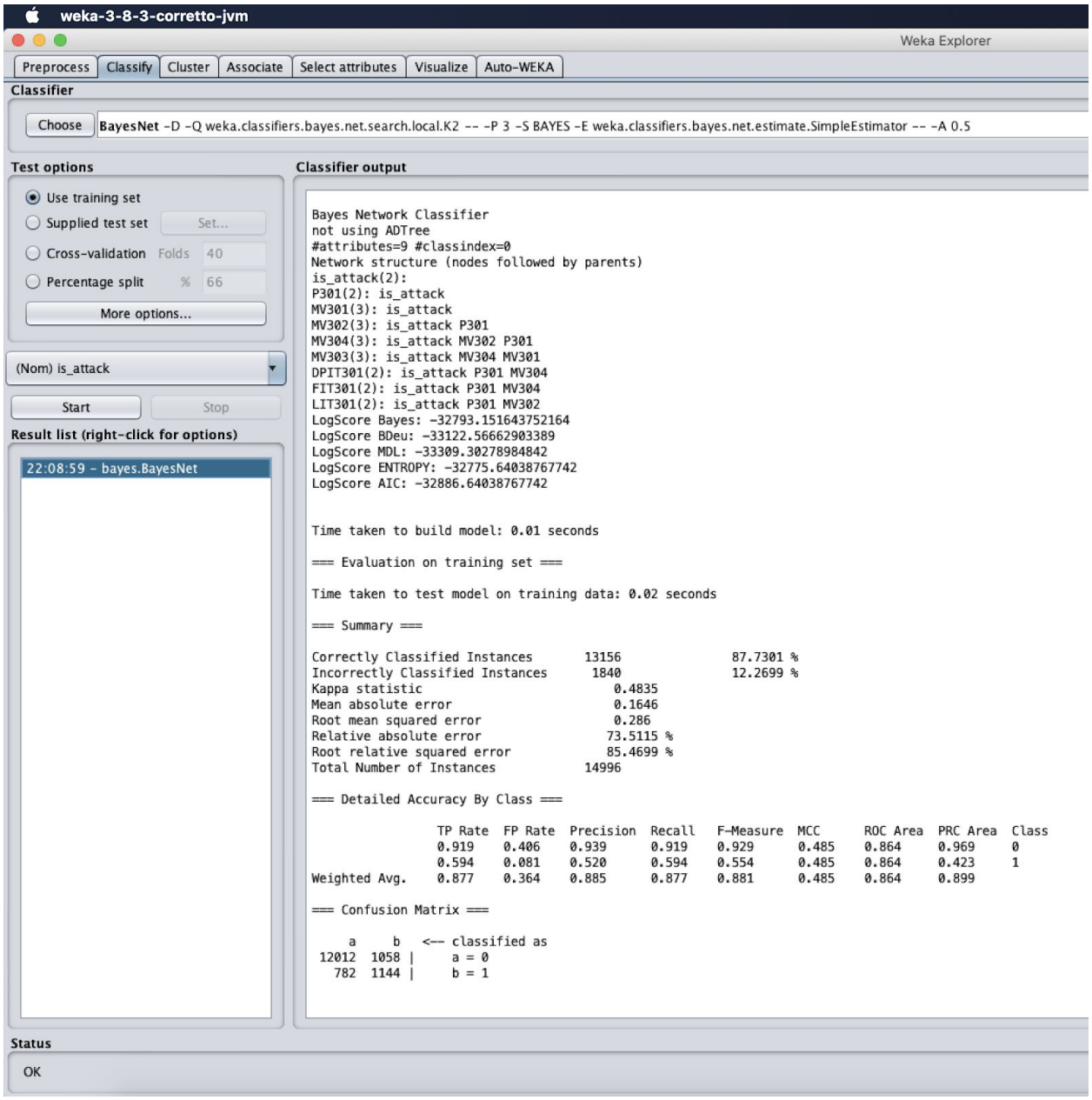
Adem Malone
Tho Nguyen
Matthew Stroble

Subsystem P3

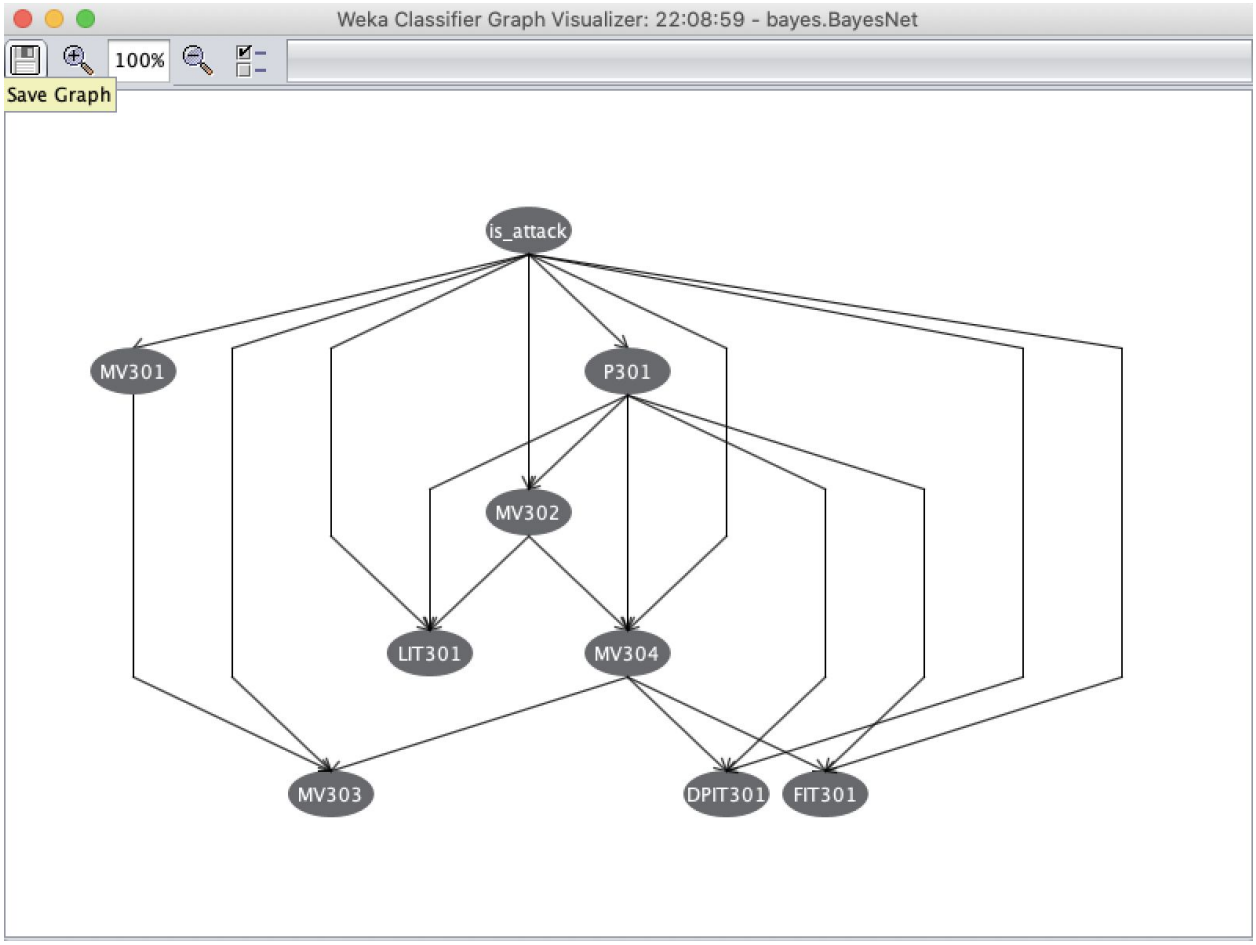
Weka Preprocessing



Weka Classify



Weka Bayesian Network K2 3 Parents



Weka Bayesian Network K2 Node MV303

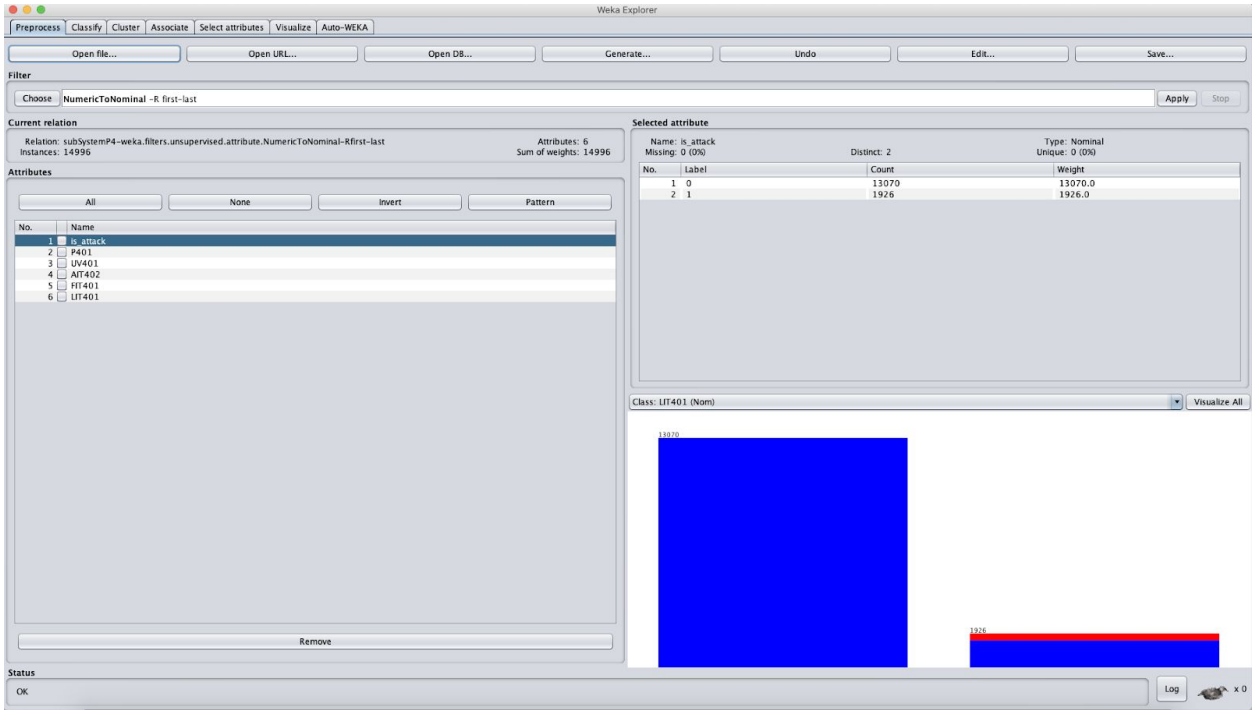
Probability Distribution Table For MV303

is_attack	MV304	MV301	0	1	2
0	0	0	0.333	0.333	0.333
0	0	1	0.005	0.903	0.092
0	0	2	0.333	0.333	0.333
0	1	0	0.515	0.03	0.455
0	1	1	0	1	0
0	1	2	0.333	0.111	0.556
0	2	0	0.333	0.333	0.333
0	2	1	0.032	0.811	0.157
0	2	2	0.333	0.333	0.333
1	0	0	0.333	0.333	0.333
1	0	1	0.333	0.333	0.333
1	0	2	0.333	0.333	0.333
1	1	0	0.333	0.333	0.333
1	1	1	0	0.999	0
1	1	2	0.333	0.333	0.333
1	2	0	0.333	0.333	0.333
1	2	1	0.333	0.333	0.333
1	2	2	0.333	0.333	0.333

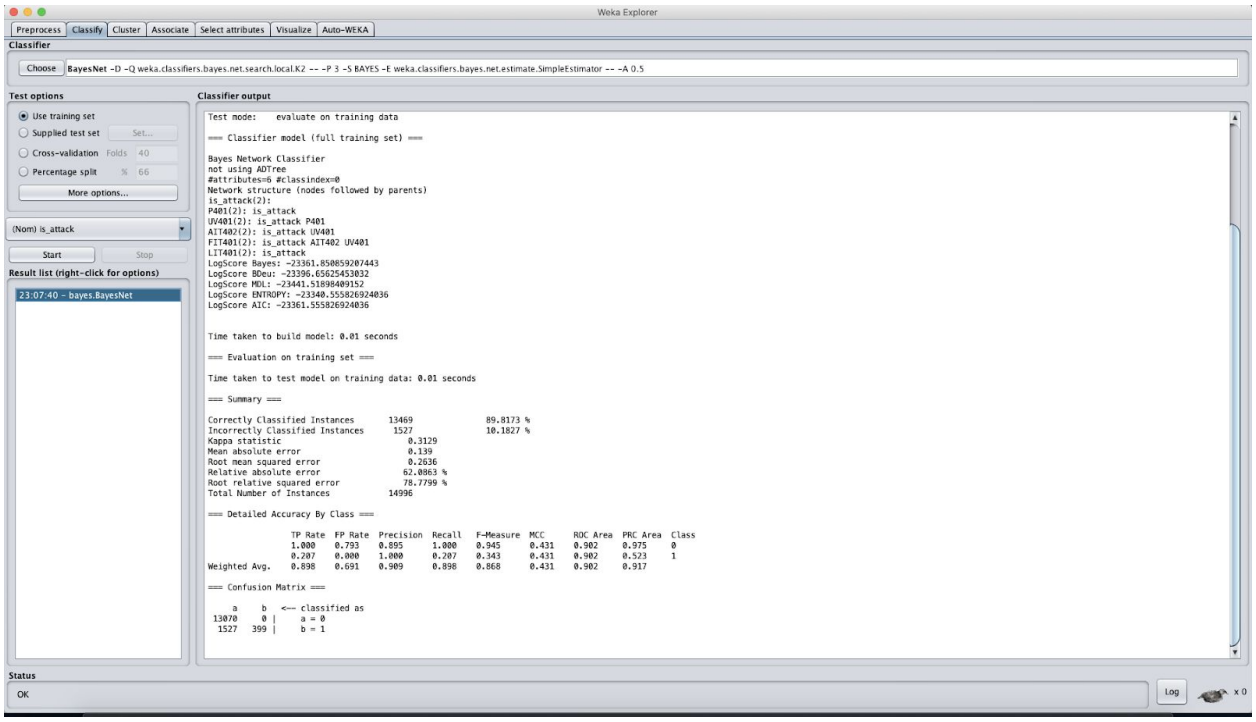
Adem Malone
Tho Nguyen
Matthew Stroble

Subsystem P4

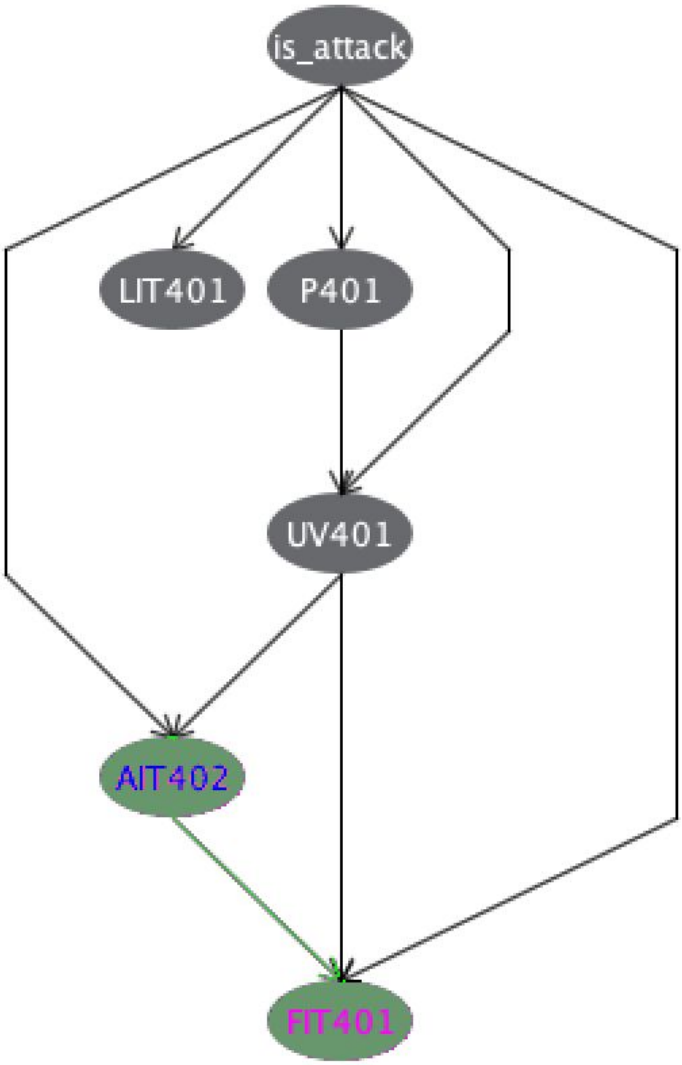
Weka Preprocessing



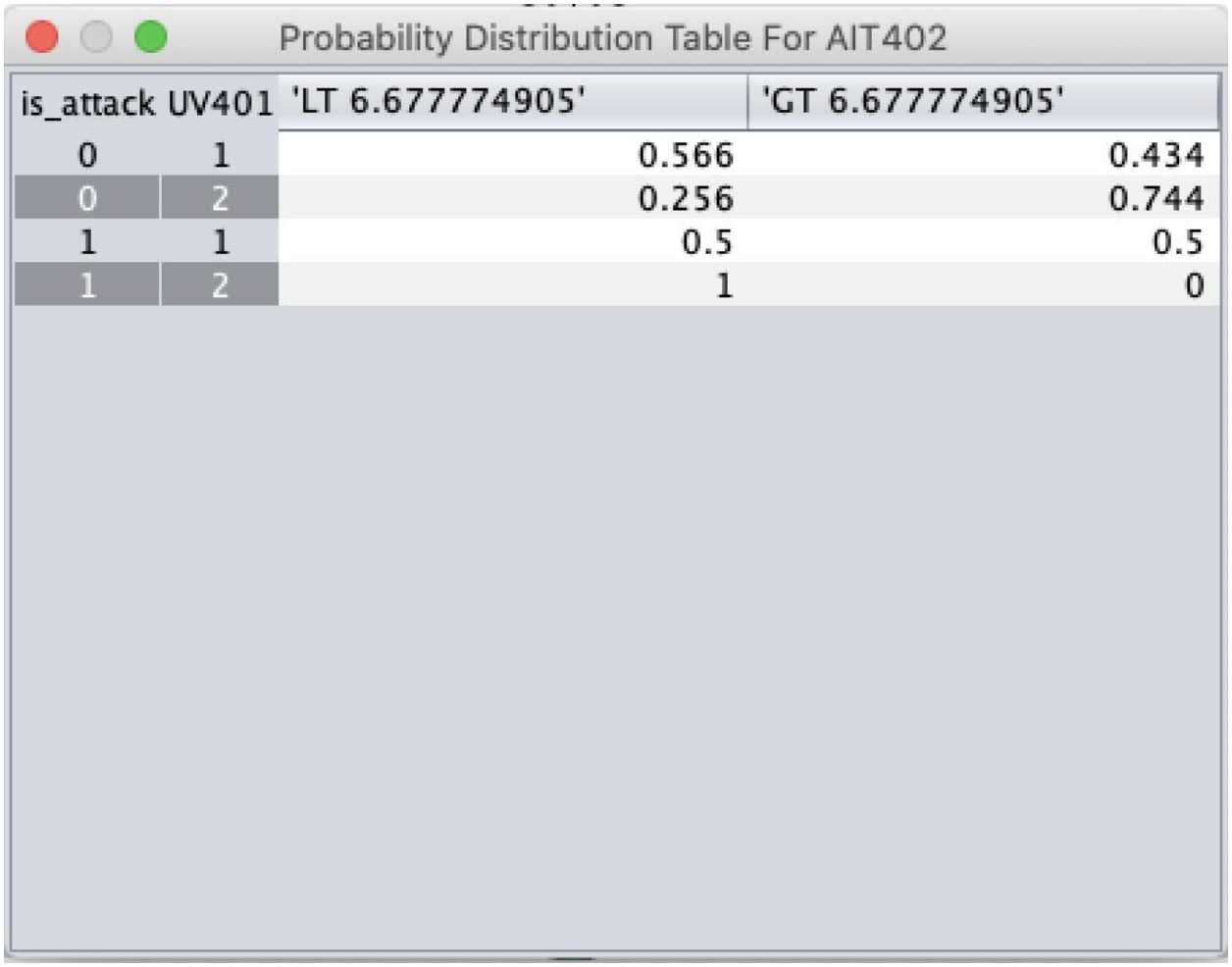
Weka Classify



Weka Bayesian Network K2 3 Parents



Weka Bayesian Network K2 Node AIT402

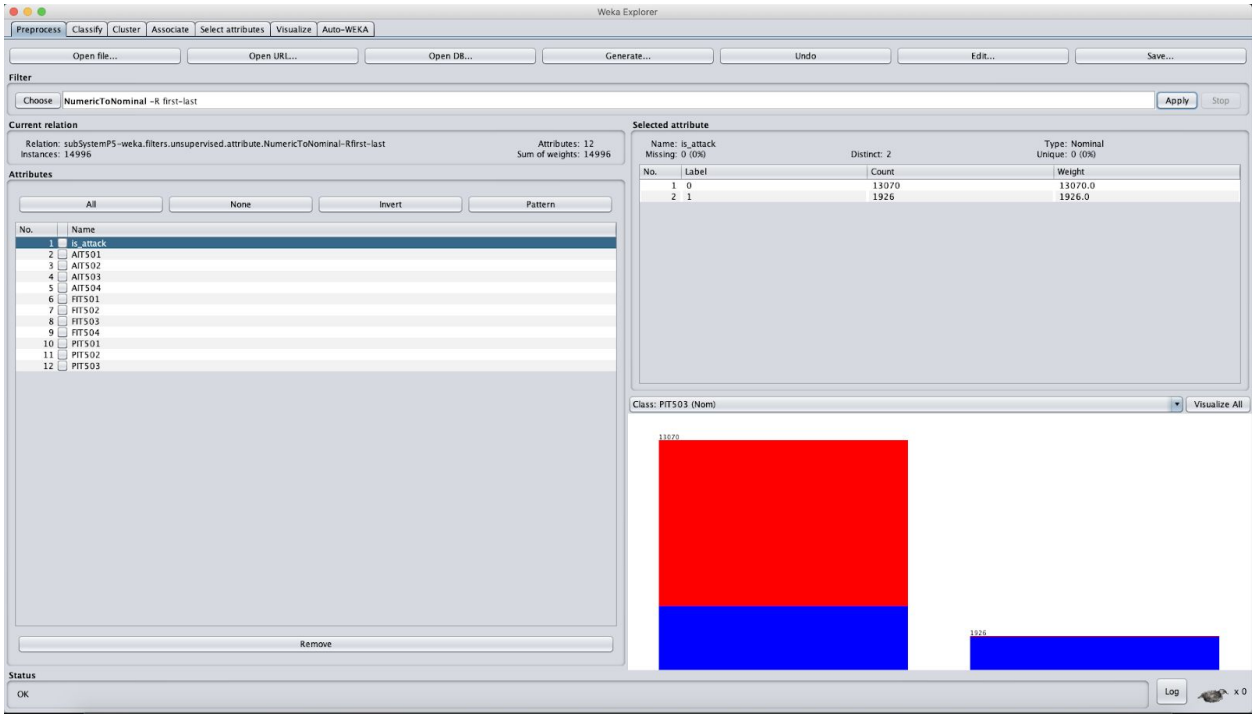


is_attack	UV401	'LT 6.677774905'	'GT 6.677774905'
0	1	0.566	0.434
0	2	0.256	0.744
1	1	0.5	0.5
1	2	1	0

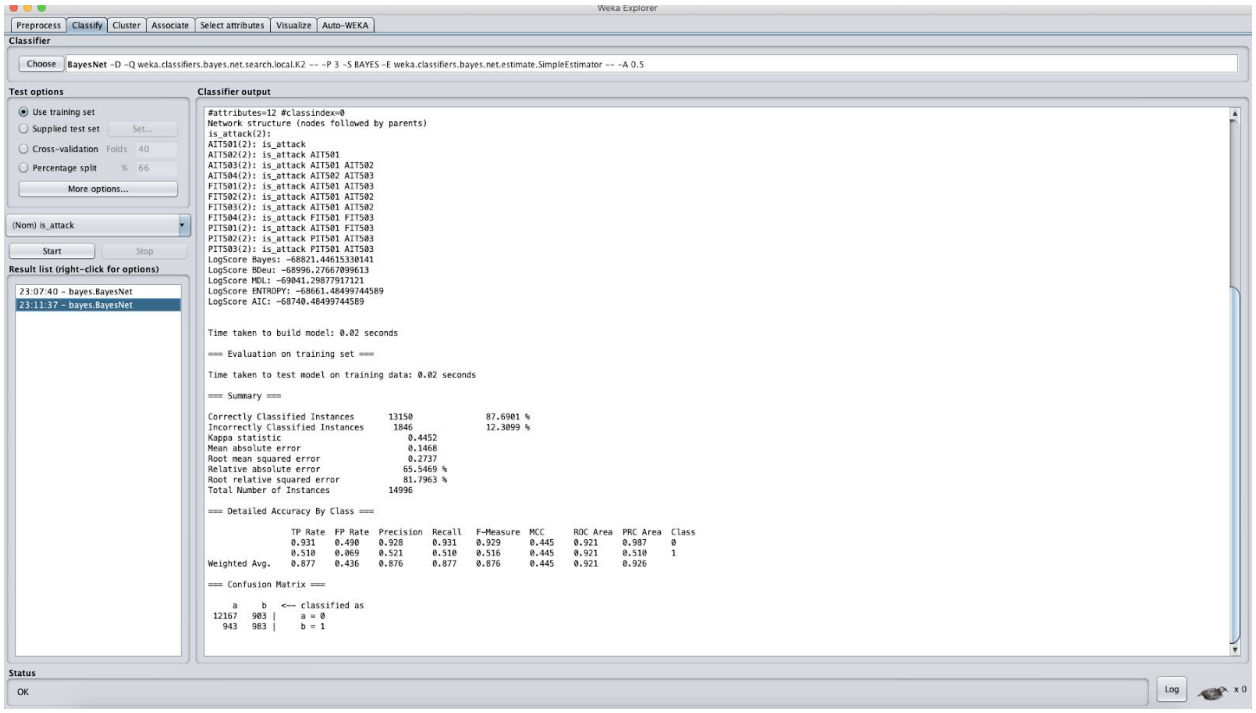
Adem Malone
Tho Nguyen
Matthew Stroble

Subsystem P5

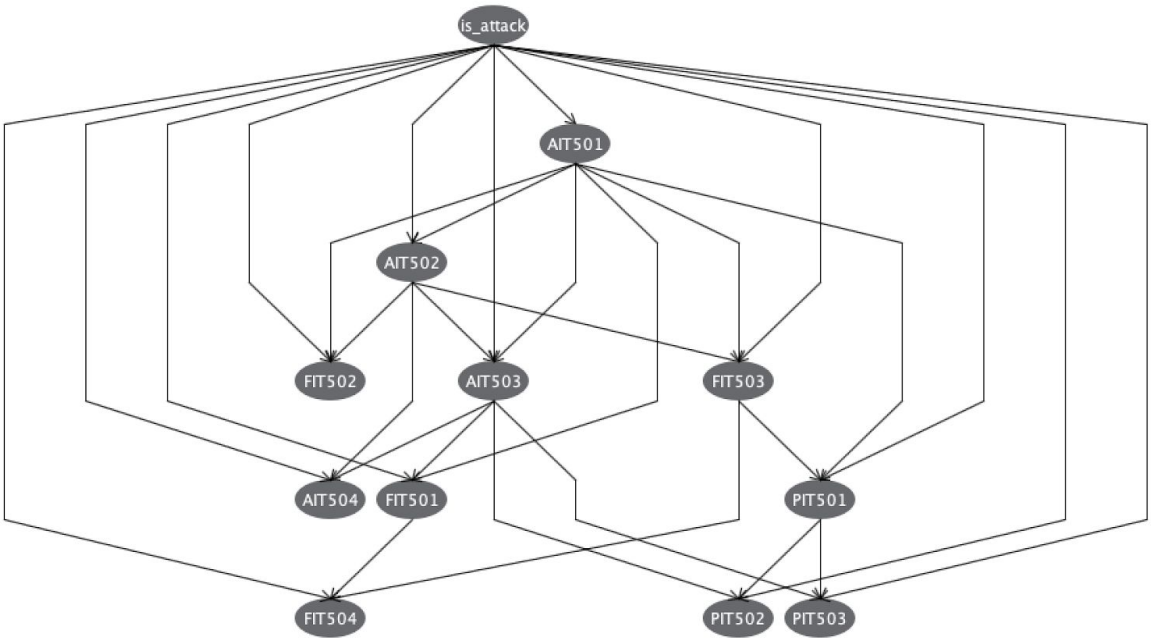
Weka Preprocessing



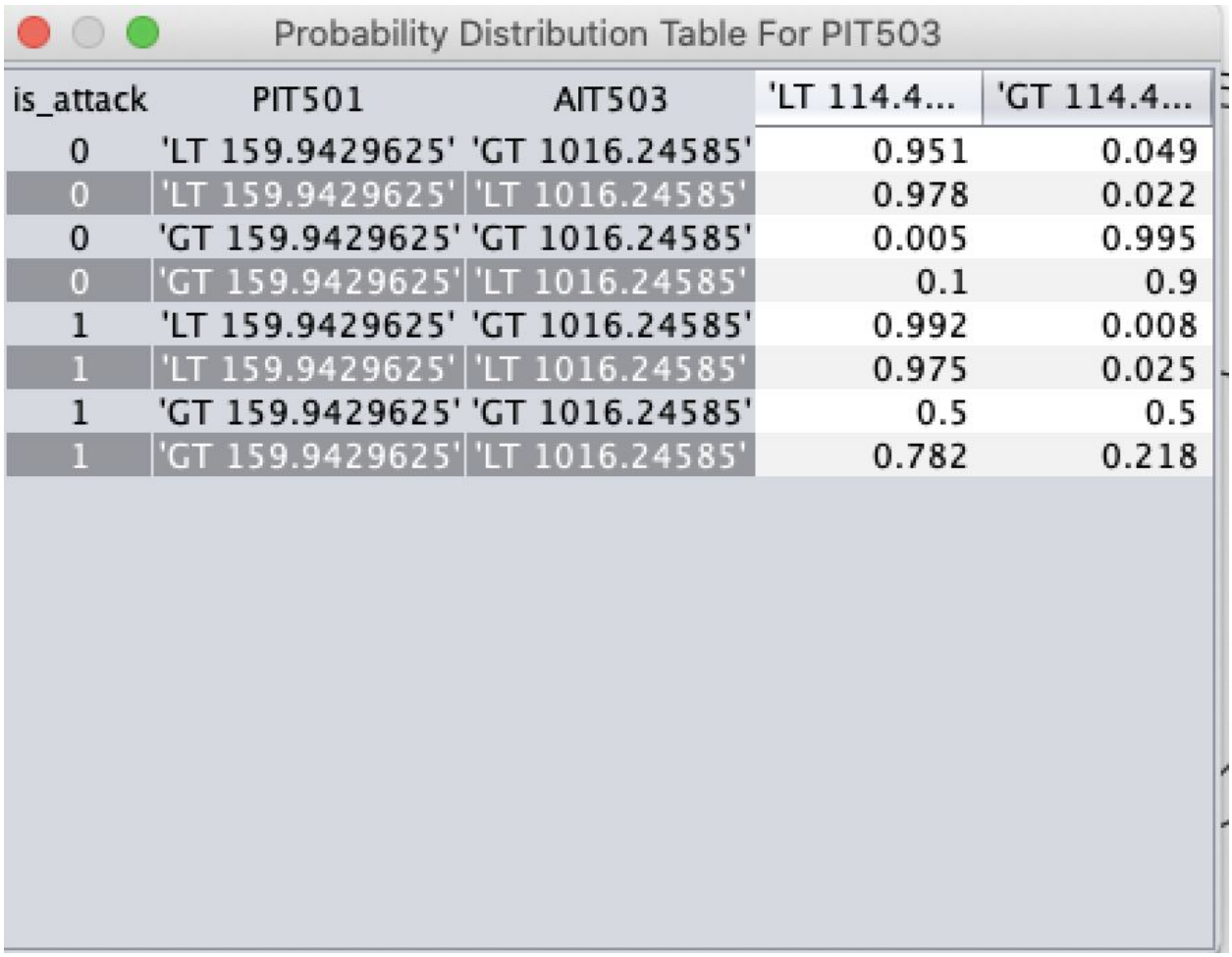
Weka Classify



Weka Bayesian Network K2 3 Parents



Weka Bayesian Network K2 Node PIT503

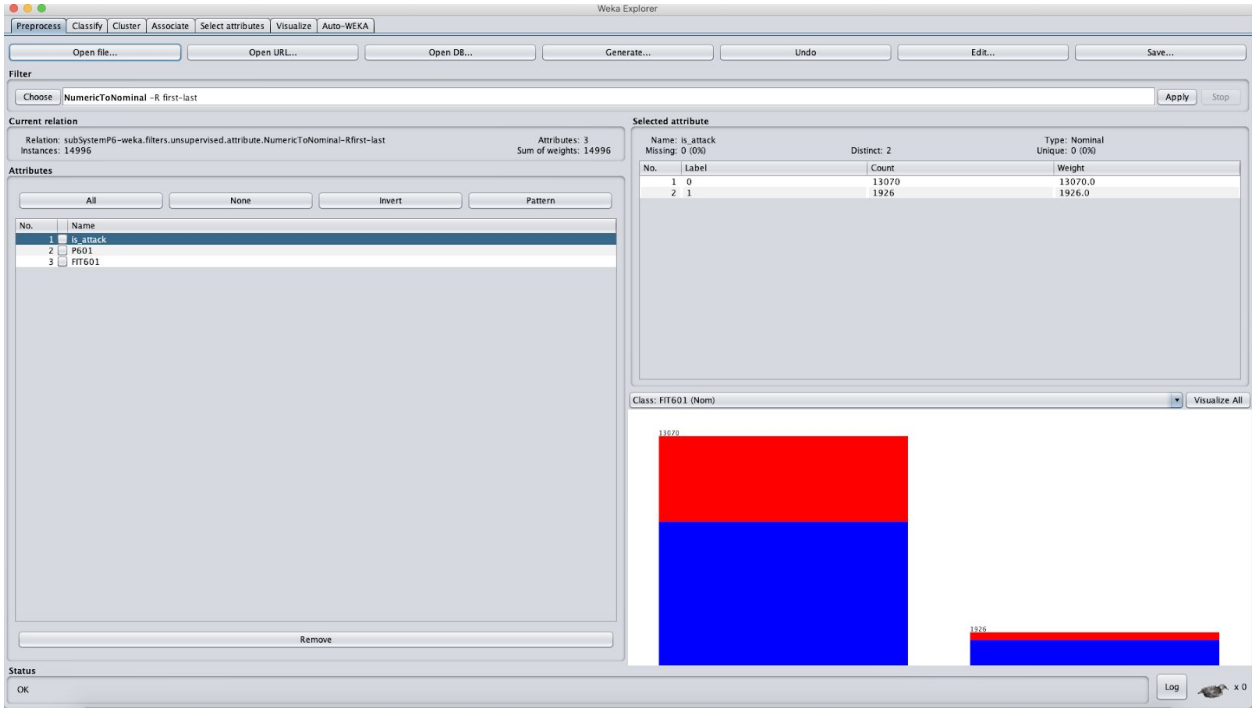


is_attack	PIT501	AIT503	'LT 114.4...	'GT 114.4...
0	'LT 159.9429625'	'GT 1016.24585'	0.951	0.049
0	'LT 159.9429625'	'LT 1016.24585'	0.978	0.022
0	'GT 159.9429625'	'GT 1016.24585'	0.005	0.995
0	'GT 159.9429625'	'LT 1016.24585'	0.1	0.9
1	'LT 159.9429625'	'GT 1016.24585'	0.992	0.008
1	'LT 159.9429625'	'LT 1016.24585'	0.975	0.025
1	'GT 159.9429625'	'GT 1016.24585'	0.5	0.5
1	'GT 159.9429625'	'LT 1016.24585'	0.782	0.218

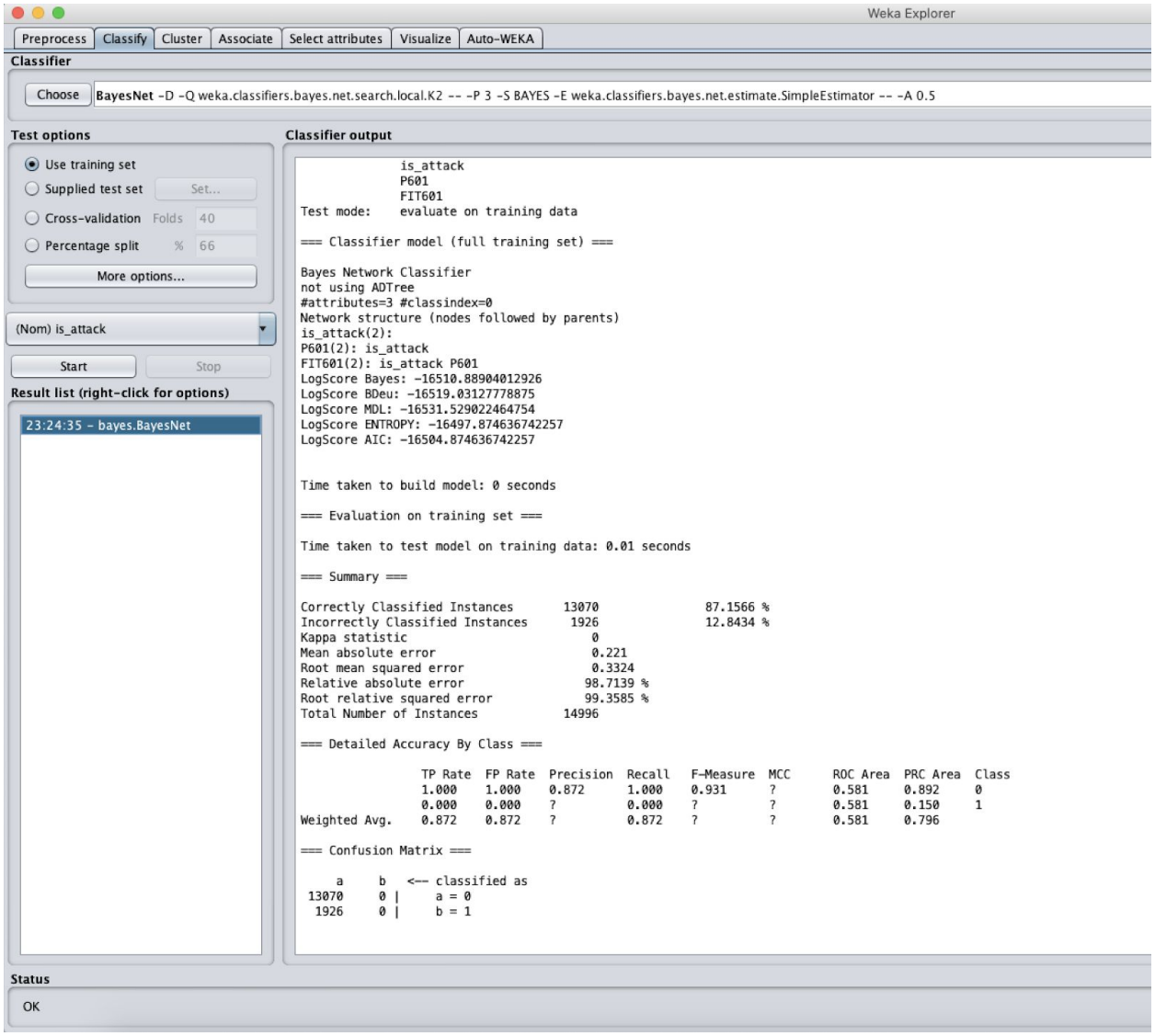
Adem Malone
Tho Nguyen
Matthew Stroble

Subsystem P6

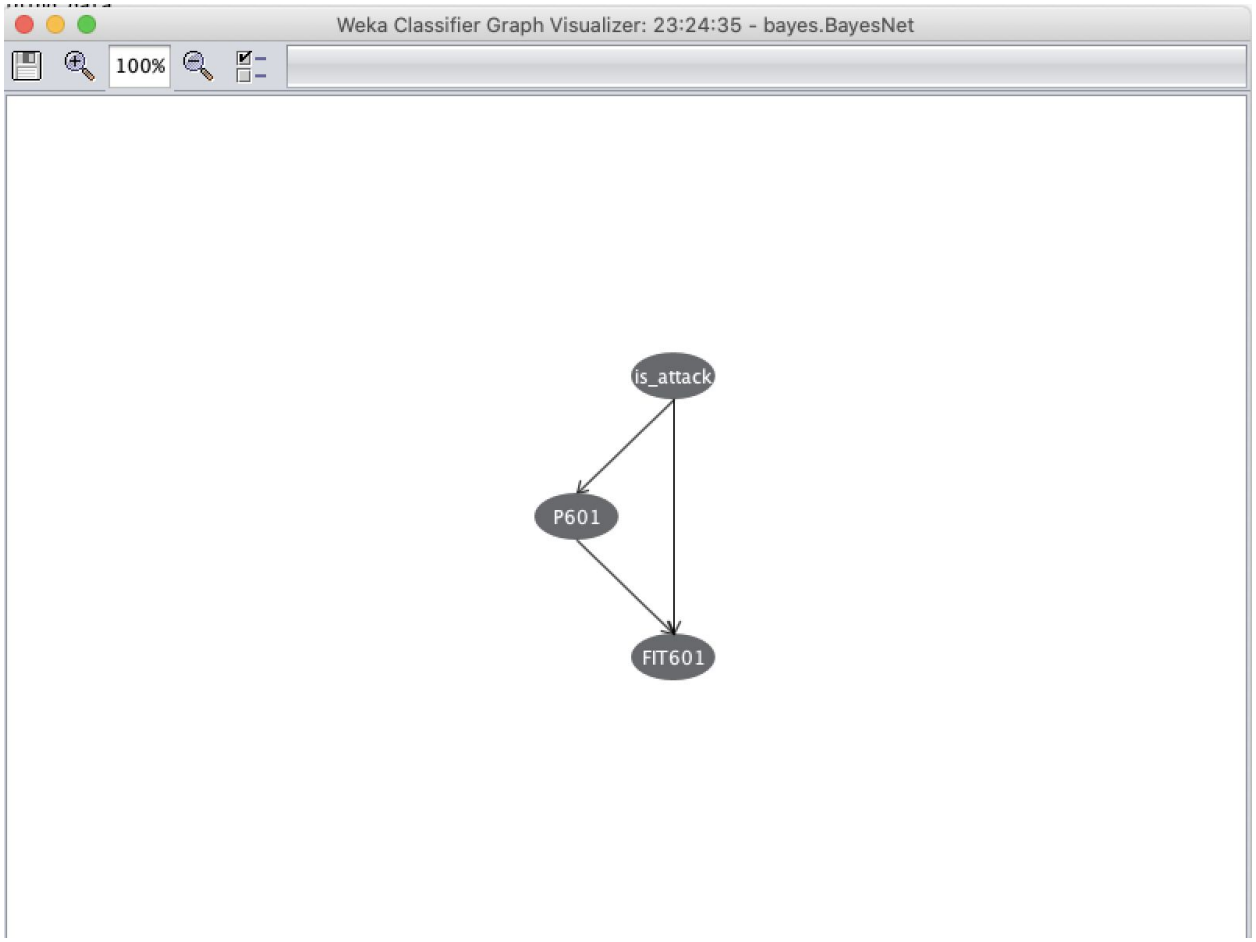
Weka Preprocessing



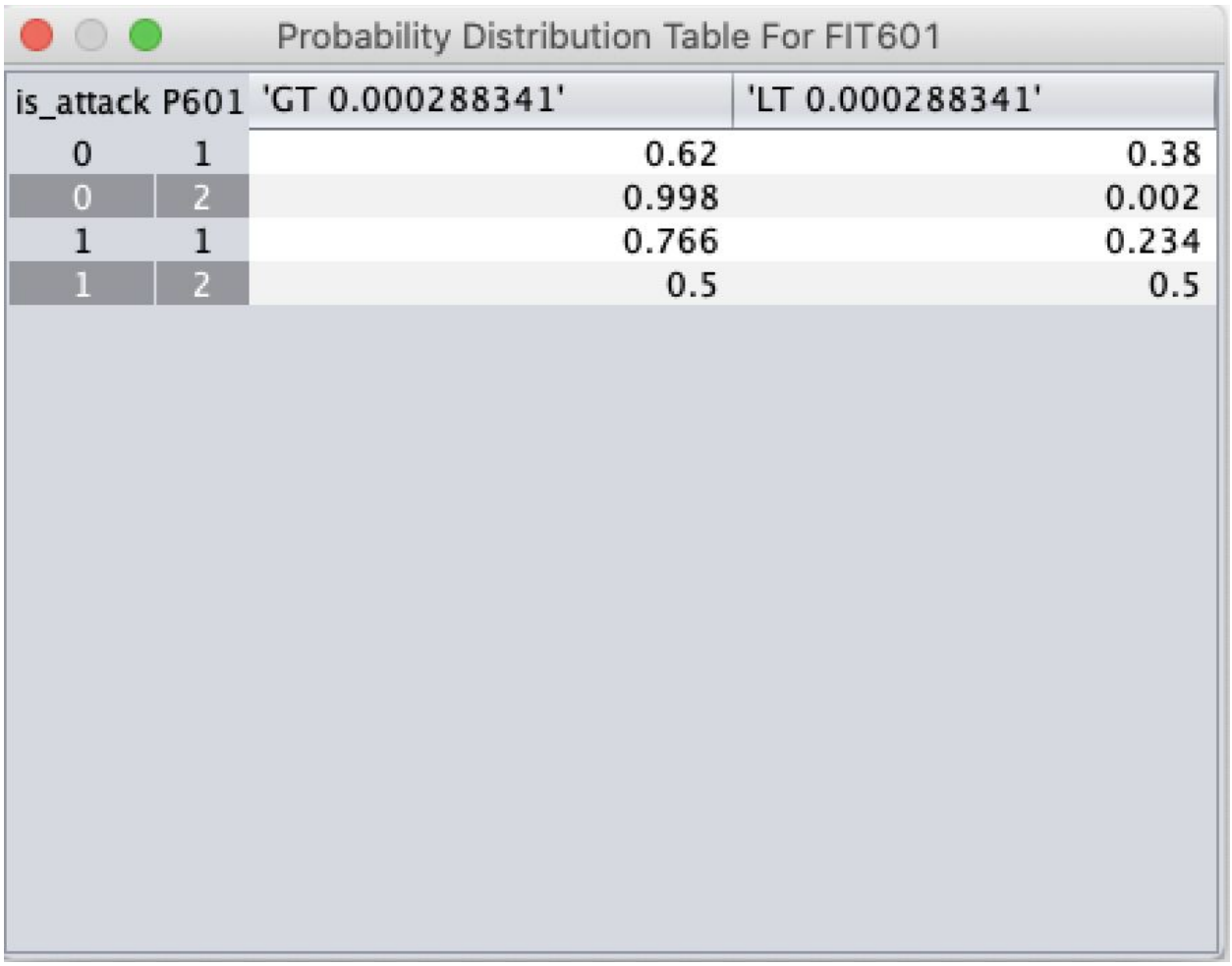
Weka Classify



Weka Bayesian Network K2 3 Parents



Weka Bayesian Network K2 Node FIT601

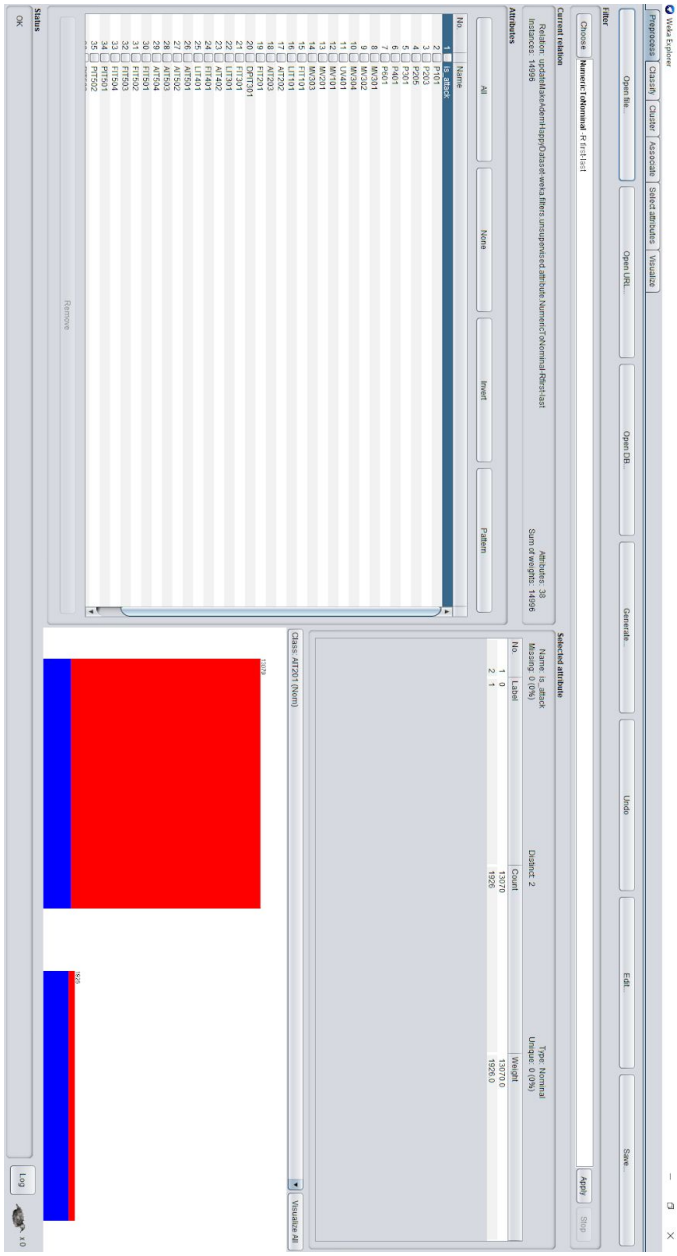


is_attack	P601	'GT 0.000288341'	'LT 0.000288341'
0	1	0.62	0.38
0	2	0.998	0.002
1	1	0.766	0.234
1	2	0.5	0.5

Adem Malone
Tho Nguyen
Matthew Stroble

Running Apriori on dataset as well as cutting down on associate rules generated.

Open Weka Explorer and use the dataset with the continuous data changed to LT(less than) and GT(greater than) values. Next you have to choose the filter “NumericToNominal” and apply it to the dataset. One this is done, it is now ready to go to the Associate tab to run Apriori.

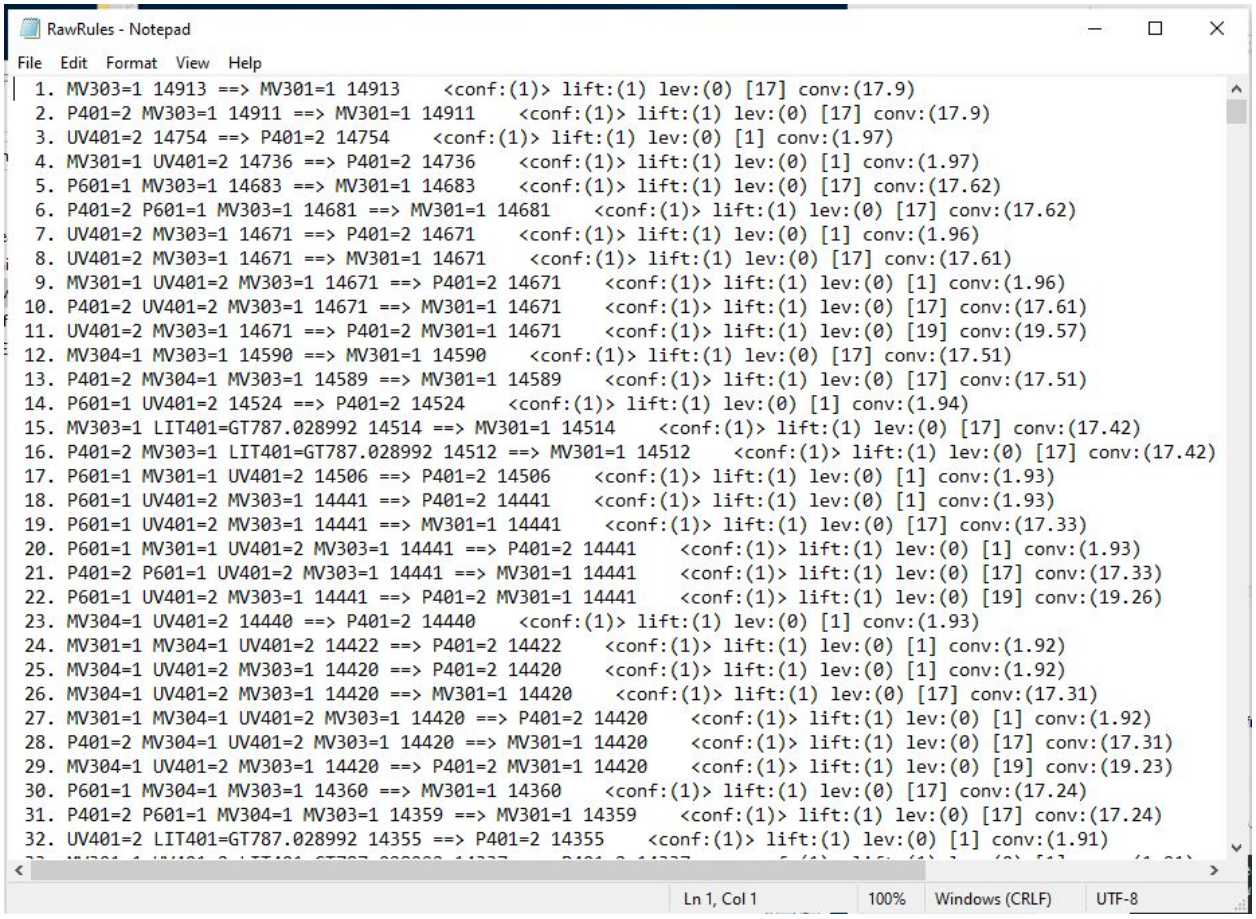


Adem Malone
Tho Nguyen
Matthew Stroble

Next we run Apriori in weka to generate 1000 rules with the nominal dataset. Also generated another 1000 rules but with the condition that every rule must contain is_attack. In order to do this we went to apriori's settings and set car to True and classIndex to 1

We put the result from weka and put the raw rules in a RawRules.txt file as below:

RawRules.txt:

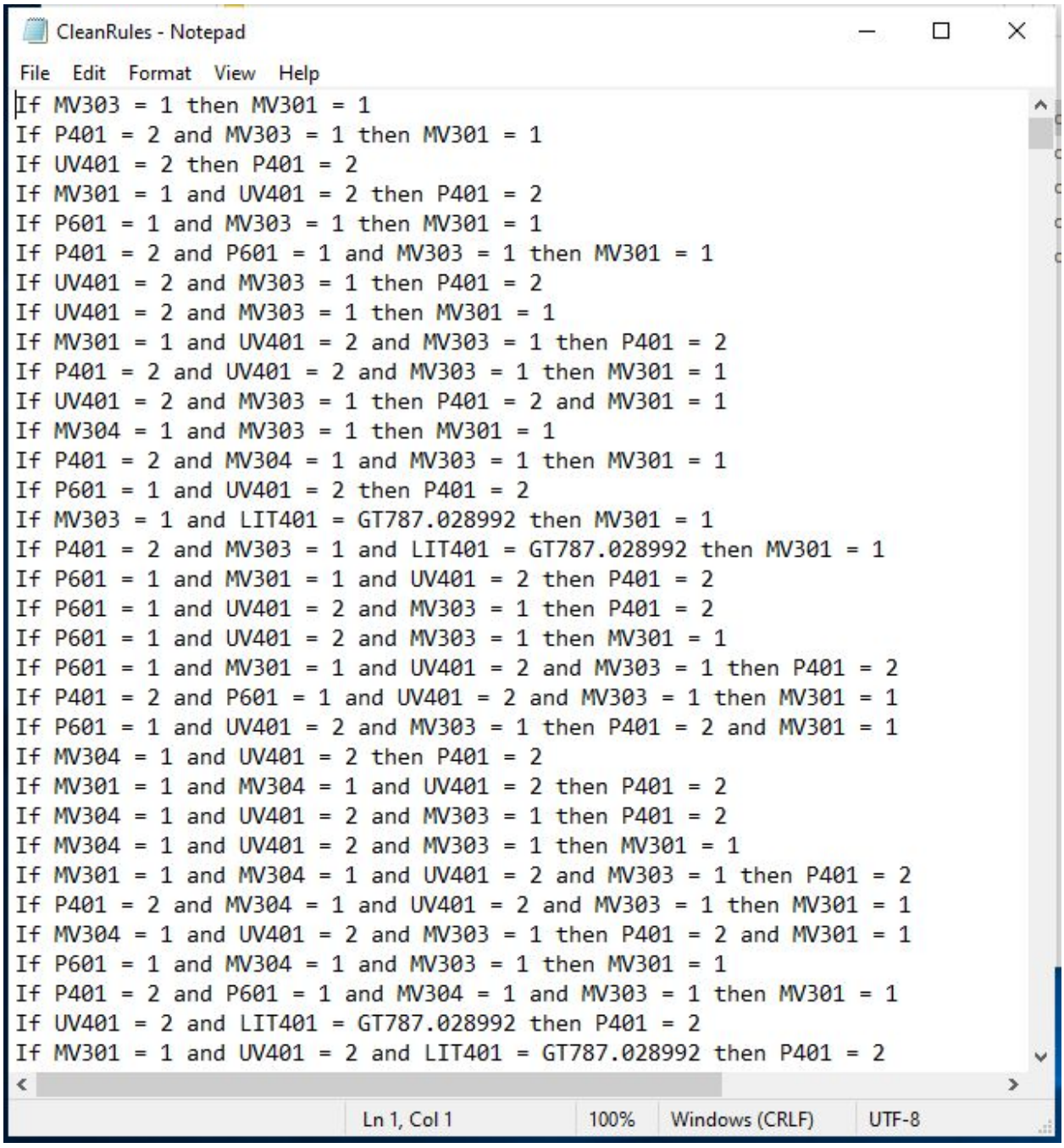


After filling the RawRules.txt, we then run the ConverRawRules.py. This python code reformats the rules to make them clean and put into a CleanRules.txt file.

ConvertRawRules.py:

```
f1 = open('CleanRules.txt','w')
f2 = open('RawRules.txt', 'r')

AllLines = f2.readlines()
cleanRules = [""] * len(AllLines)
#Goes through all rules made by Weka apriori and formats
#it in a way for RuleY_E_Generator.py to recognize
for Rule in AllLines:
      List = Rule.split()
      string = "If"
      Then = False
      for i in range(len(List)):
            Part = []
            if("=" in List[i]):
                  if(i == 1 or Then):
                        Part = List[i].split("=")
                        string = string + " " + Part[0] + " = " + Part[1]
                        Then = False
                  elif(List[i] != "==>"):
                        Part = List[i].split("=")
                        string = string + " and " + Part[0] + " = " +
Part[1]
                  else:
                        string = string + " then"
                        Then = True
      # if("is_attack" in string):
      #     f1.write(string + "\n")
      f1.write(string + "\n")
f1.close()
f2.close()
```

Resulting CleanRules.txt:



One we have the clean rules, we can feed it into the RuleY_E_Generator.py. This code calculates the Y and E tables, which has Y_1 and E_1 values as well as all the Y_2 and E_2 values. It then puts them into a Y_E_Tables.txt file into an array where the first part of the array is the Y_1 value followed by the Y_2 values in order of the rules. The second part of the array is the E values in the same structure and order as the Y's.

RuleY_E_Generator.py:

```
import csv
fields = []
rows = []
csvLocation = ("updateMakeAdemHappyDataset.csv")

r = csv.reader(open(csvLocation))
lines = list(r)
maxRows = 14996

def YandE_Generator(RawRule):
      ColumnNames = []
      Sign = []
      Number = []
      ExpectName = False
      ExpectSign = False
      ExpectNumber = False
      ExpectAnd = False

      BeforeThen = 0

      Rule = RawRule.split()
      for i in range(len(Rule)):
            if(Rule[i] == "If"):
                  ExpectName = True

            elif(Rule[i] == "and" or Rule[i] == "then" and ExpectAnd):
                  ExpectName = True
                  ExpectAnd = False
                  if(Rule[i] == "then"):
                        BeforeThen = len(ColumnNames)
```

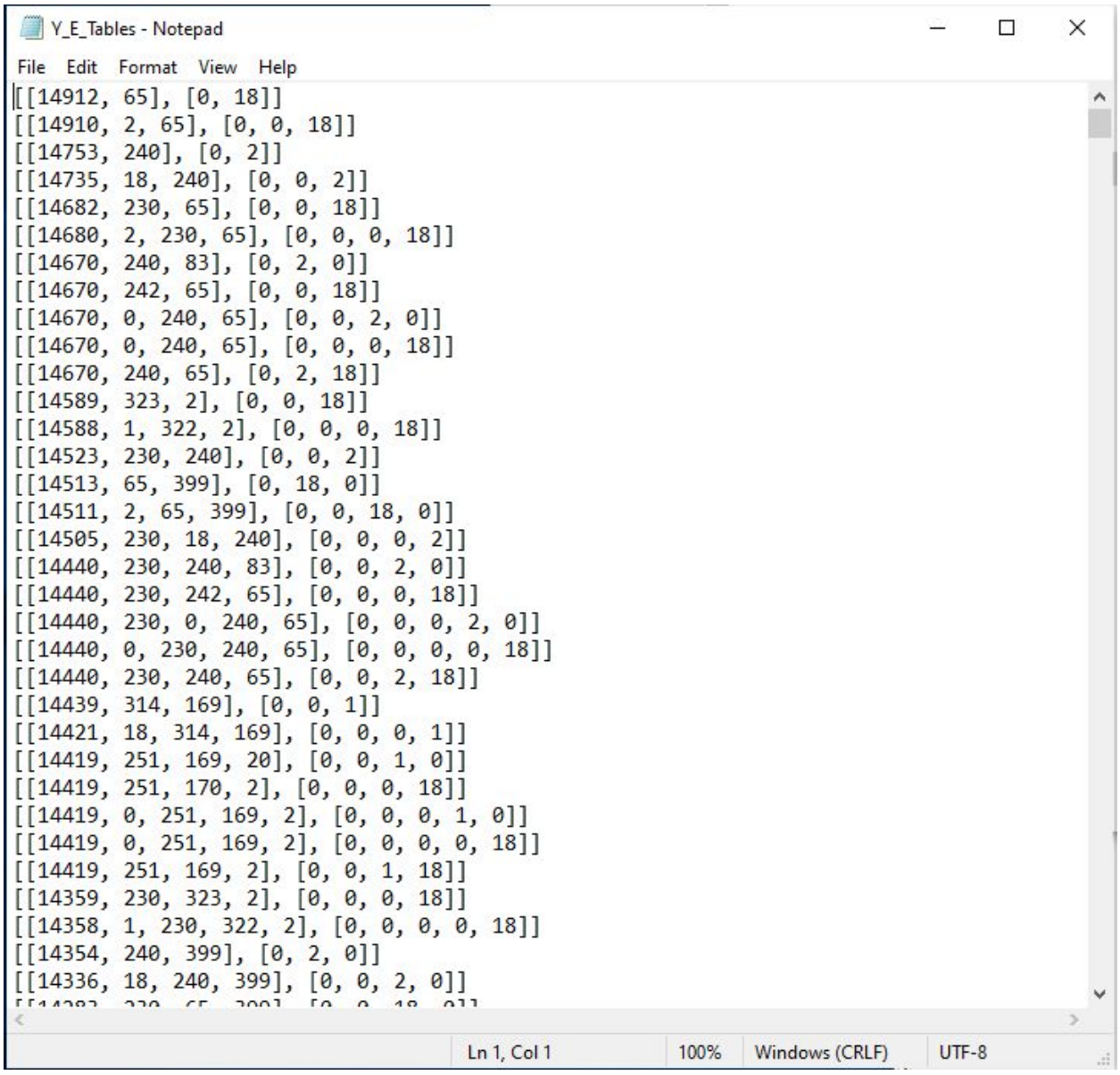
```
            elif(Rule[i] == ">=" and ExpectSign):
                  Sign.append(Rule[i])
                  ExpectNumber = True
                  ExpectSign = False

            elif(Rule[i] == ">" and ExpectSign):
                  Sign.append(Rule[i])
                  ExpectNumber = True
                  ExpectSign = False

            elif(Rule[i] == "<=" and ExpectSign):
                  Sign.append(Rule[i])
                  ExpectNumber = True
                  ExpectSign = False

            elif(Rule[i] == "<" and ExpectSign):
                  Sign.append(Rule[i])
                  ExpectNumber = True
                  ExpectSign = False

            elif(Rule[i] == "=" and ExpectSign):
                  Sign.append(Rule[i])
                  ExpectNumber = True
                  ExpectSign = False

            else:
                  if(ExpectName):
                        ColumnNames.append(Rule[i])
                        ExpectName = False
                        ExpectSign = True
                  elif(ExpectNumber):
                        Number.append(Rule[i])
                        ExpectNumber = False
                        ExpectAnd = True
                  else:
                        print("Error: Not a valid rule")
                        print(RawRule)
                        break

      ColumnPositions = []
```

```
      for j in range(len(ColumnNames)):
            for k in range(38):
                  if(ColumnNames[j] in lines[0][k]):
                        ColumnPositions.append(k)

      AfterThen = len(ColumnNames) - BeforeThen

      E1 = 0
      Y1 = 0
      Y2 = [0] * (len(ColumnPositions) - AfterThen)
      E2 = [0] * (len(ColumnPositions) - AfterThen)

      for Turn in range(-1, len(ColumnPositions) - AfterThen):
            for r in range(1, maxRows):
                  for c in range(len(ColumnPositions)):
                        NoNegating = True
                        if(c == Turn):
                              NoNegating = False
                        StatementAcc = False
                        if(Sign[c] == ">="):
                              if(NoNegating):
                                    if(lines[r][ColumnPositions[c]] >=
Number[c]):
                                          StatementAcc = True
                              else:
                                    if(lines[r][ColumnPositions[c]] <
Number[c]):
                                          StatementAcc = True
                        elif(Sign[c] == ">"):
                              if(NoNegating):
                                    if(lines[r][ColumnPositions[c]] >
Number[c]):
                                          StatementAcc = True
                              else:
                                    if(lines[r][ColumnPositions[c]] <=
Number[c]):
                                          StatementAcc = True
                        elif(Sign[c] == "<="):
                              if(NoNegating):
```

```
                                    if(lines[r][ColumnPositions[c]] <=
Number[c]):
                                          StatementAcc = True
                              else:
                                    if(lines[r][ColumnPositions[c]] >
Number[c]):
                                          StatementAcc = True
                        elif(Sign[c] == "<"):
                              if(NoNegating):
                                    if(lines[r][ColumnPositions[c]] <
Number[c]):
                                          StatementAcc = True
                              else:
                                    if(lines[r][ColumnPositions[c]] >=
Number[c]):
                                          StatementAcc = True
                        elif(Sign[c] == "="):
                              if(NoNegating):
                                    if(lines[r][ColumnPositions[c]] ==
Number[c]):
                                          StatementAcc = True
                              else:
                                    if(lines[r][ColumnPositions[c]] !=
Number[c]):
                                          StatementAcc = True
                        else:
                              print("Somethings not right")
                        #before then and statement is false then just stop
                        if(not(c >= len(ColumnPositions) - AfterThen) and
not(StatementAcc)):
                              break
                        #After then, meaning all statements before then
were true so start
                        #counting Y1 and E1 as well as Y2 and E2
                        if((c >= len(ColumnPositions) - AfterThen)):
                              #Checks if a condition after then is correct,
if not then E's go up and breaks
                              if(StatementAcc):
                                    if(c == len(ColumnPositions) - 1):
                                          if(Turn == -1):
```

```
                                                Y1 += 1
                                          else:
                                                Y2[Turn] += 1
                                    else:
                                          if(Turn == -1):
                                                E1 += 1
                                          else:
                                                E2[Turn] += 1
                                          break
      Y2.insert(0, Y1)
      E2.insert(0, E1)
      YE_Table = []
      YE_Table.append(Y2)
      YE_Table.append(E2)
      return YE_Table



print(YandE_Generator("If FIT101 = LT4.385068655 then P401 = 2"))
count = 0
f1 = open('CleanRules.txt','r')
f2 = open('Y_E_Tables.txt', 'w')
AllRules = f1.readlines()
for Rule in AllRules:
      count = count + 1
      print(count)
      f2.write(str(YandE_Generator(Rule)) + "\n")

f1.close()
f2.close()
```

Y_E_Tables.txt:



Adem Malone
Tho Nguyen
Matthew Stroble

With the Y_E_Tables.txt, it is now possible to calculate the R and R prime values of the rules by running UcfCalculator.py. In order to calculate the R value as well as every R prime value, we use the following equations:

$$R = U_{25\%}(E_1, E_1 + Y_1)$$

$$R \text{ prime} = U_{25\%}(E_1 + E_2, E_1 + E_2 + Y_1 + Y_2)$$

We then put each of these values into a UcfValesOfYE.txt file. Each rule is generated an array with the first number being R and the next numbers being the R primes of each part of the rule.

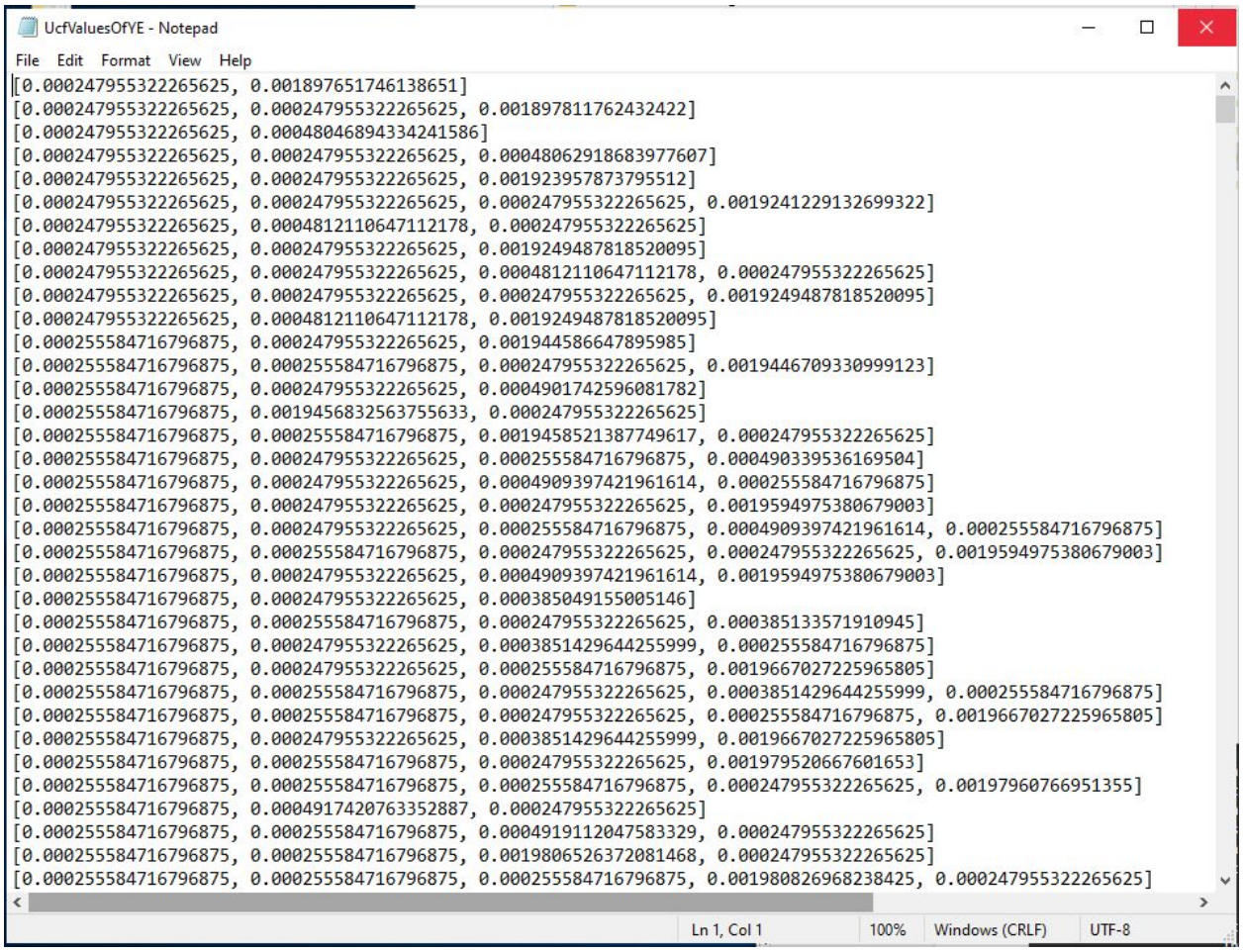
UcfCalculator.py:

```
#The function binP and calcBin is not my code
#but the rest is. I had to use it since I didn't
#know how to calculate Ucf() without using wolfram alpha
#Source: https://stackoverflow.com/questions
#/13059011/is-there-any-python-function-library-
#for-calculate-binomial-confidence-intervals

#Calculatee Ucf for associate rule dropping/cleaning
def binP(N, p, x1, x2):
    p = float(p)
    q = p/(1-p)
    k = 0.0
    v = 1.0
    s = 0.0
    tot = 0.0
    while(k<=N):
        tot += v
        if(k >= x1 and k <= x2):
            s += v
        if(tot > 10**30):
            s = s/10**30
            tot = tot/10**30
            v = v/10**30
        k += 1
        v = v*q*(N+1-k)/k
    return s/tot

def calcBin(vx, vN, vCL = 95):
    vx = float(vx)
    vN = float(vN)
```

```
    #Set the confidence bounds
    vTU = (100 - float(vCL))/2
    vTL = vTU
    vP = vx/vN
    if(vx==0):
        dl = 0.0
    else:
        v = vP/2
        vsL = 0
        vsH = vP
        p = vTL/100

        while((vsH-vsL) > 10**-5):
            if(binP(vN, v, vx, vN) > p):
                vsH = v
                v = (vsL+v)/2
            else:
                vsL = v
                v = (v+vsH)/2
        dl = v
    if(vx==vN):
        ul = 1.0
    else:
        v = (1+vP)/2
        vsL =vP
        vsH = 1
        p = vTU/100
        while((vsH-vsL) > 10**-5):
            if(binP(vN, v, 0, vx) < p):
                vsH = v
                v = (vsL+v)/2
            else:
                vsL = v
                v = (v+vsH)/2
        ul = v
    return (dl, ul)

f1 = open('CleanRules.txt','r')
f2 = open('Y_E_Tables.txt', 'r')
f4 = open('UcfValuesOfYE.txt', 'w')
```

```
AllRules = f1.readlines()
AllYE_Tables = f2.readlines()
count = 0
for i in range(len(AllRules)):
    count += 1
    print(count)
    Y = []
    E = []
    Y_E_Table = AllYE_Tables[i]
    New = Y_E_Table.split(', ')
    YPart = False
    EPart = False
    for j in New:
        if('[[' in j or YPart):
            YPart = True
            NewJ = j.replace("[", "")
            Y.append(int(NewJ.replace("]", "")))
        elif('[' in j or EPart):
            EPart = True
            NewJ = j.replace("[", "")
            E.append(int(NewJ.replace("]", "")))
        if(']' in j):
            YPart = False
    Total1 = Y[0] + E[0]
    UcfValue = calcBin(Y[0], Total1)
    R = []
    R.append(1 - UcfValue[0])
    for k in range(1, len(Y)):
        Total2 = Total1 + Y[k] + E[k]
        UcfValue = calcBin(Y[0] + Y[k], Total2)
        R.append(1 - UcfValue[0])
    f4.write(str(R) + "\n")

f1.close()
f2.close()
f4.close()
```

UcfValues:


UcfValuesOfYE - Notepad
File Edit Format View Help
[0.000247955322265625, 0.001897651746138651]
[0.000247955322265625, 0.000247955322265625, 0.001897811762432422]
[0.000247955322265625, 0.00048046894334241586]
[0.000247955322265625, 0.000247955322265625, 0.00048062918683977607]
[0.000247955322265625, 0.000247955322265625, 0.001923957873795512]
[0.000247955322265625, 0.000247955322265625, 0.000247955322265625, 0.0019241229132699322]
[0.000247955322265625, 0.0004812110647112178, 0.000247955322265625]
[0.000247955322265625, 0.000247955322265625, 0.0019249487818520095]
[0.000247955322265625, 0.000247955322265625, 0.0004812110647112178, 0.000247955322265625]
[0.000247955322265625, 0.000247955322265625, 0.000247955322265625, 0.0019249487818520095]
[0.000247955322265625, 0.0004812110647112178, 0.0019249487818520095]
[0.000255584716796875, 0.000247955322265625, 0.001944586647895985]
[0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.0019446709330999123]
[0.000255584716796875, 0.000247955322265625, 0.0004901742596081782]
[0.000255584716796875, 0.0019456832563755633, 0.000247955322265625]
[0.000255584716796875, 0.000255584716796875, 0.0019458521387749617, 0.000247955322265625]
[0.000255584716796875, 0.000247955322265625, 0.000255584716796875, 0.0004903395361695O4]
[0.000255584716796875, 0.000247955322265625, 0.0004909397421961614, 0.000255584716796875]
[0.000255584716796875, 0.000247955322265625, 0.000247955322265625, 0.0019594975380679003]
[0.000255584716796875, 0.000247955322265625, 0.000255584716796875, 0.0004909397421961614, 0.000255584716796875]
[0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.000247955322265625, 0.0019594975380679003]
[0.000255584716796875, 0.000247955322265625, 0.0004909397421961614, 0.0019594975380679003]
[0.000255584716796875, 0.000247955322265625, 0.000385049155005146]
[0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.000385133571910945]
[0.000255584716796875, 0.000247955322265625, 0.0003851429644255999, 0.000255584716796875]
[0.000255584716796875, 0.000247955322265625, 0.000255584716796875, 0.0019667027225965805]
[0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.0003851429644255999, 0.000255584716796875]
[0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.000255584716796875, 0.0019667027225965805]
[0.000255584716796875, 0.000247955322265625, 0.0003851429644255999, 0.0019667027225965805]
[0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.001979520667601653]
[0.000255584716796875, 0.000255584716796875, 0.000255584716796875, 0.000247955322265625, 0.00197960766951355]
[0.000255584716796875, 0.0004917420763352887, 0.000247955322265625]
[0.000255584716796875, 0.000255584716796875, 0.0004919112047583329, 0.000247955322265625]
[0.000255584716796875, 0.000255584716796875, 0.0019806526372081468, 0.000247955322265625]
[0.000255584716796875, 0.000255584716796875, 0.000255584716796875, 0.0019808026968238425, 0.000247955322265625]
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

Now with the R and R prime values generated for each rule, we can run RuleCutting.py which compares the R and R prime values of each rule. If R prime is lower than R, that part of the rule related to that R prime is a candidate to be dropped/cut out. If there are more than one candidate for each rule, we pick the part with the lowest R prime to be dropped. We then remove the cut out rule part and then reformat the rule into a FinishedRules.txt file.

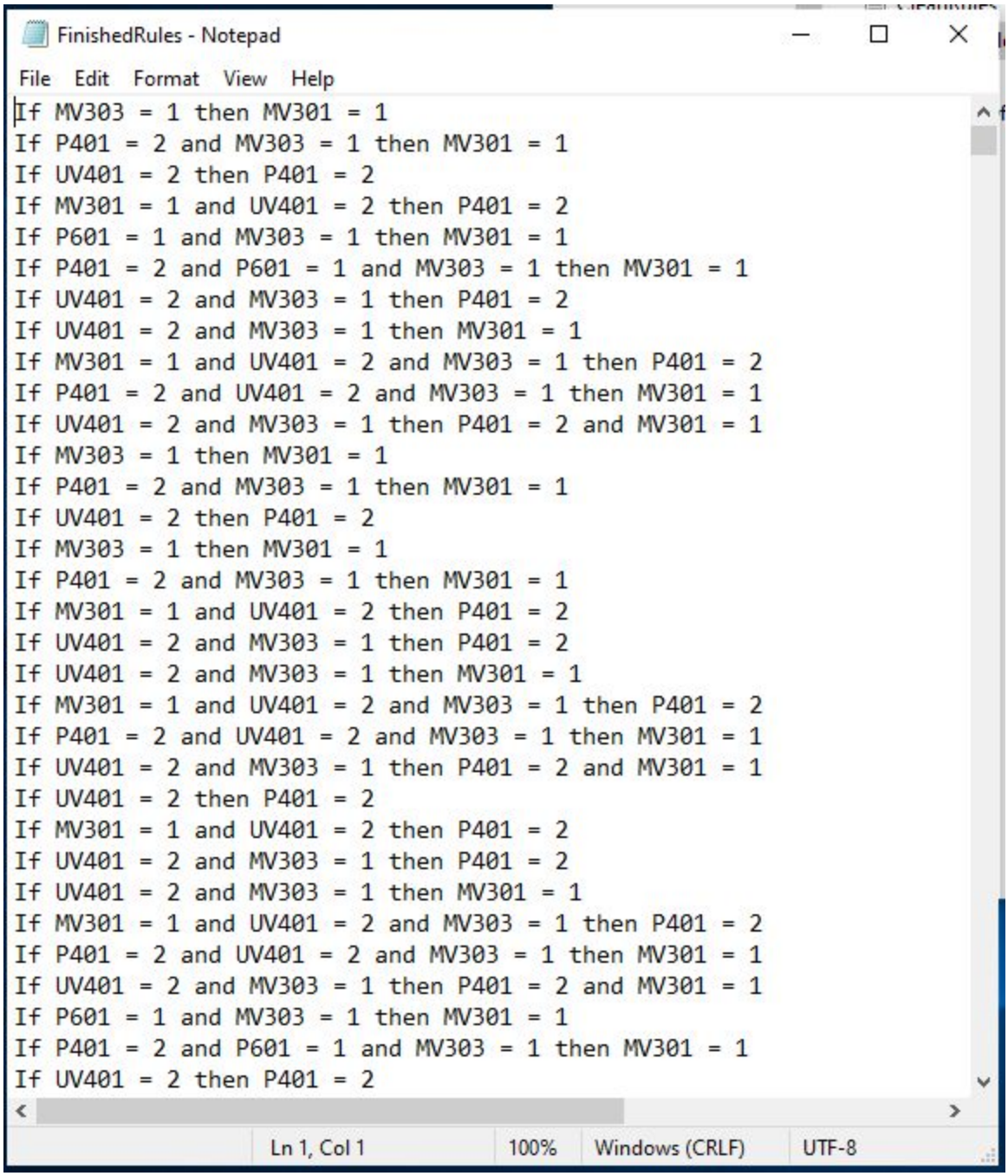
RuleCutting.py:

```
import re

def remove_values(List, value):
      return[val for val in List if val != value]

f1 = open('CleanRules.txt','r')
f2 = open('UcfValuesOfYE.txt', 'r')
f3 = open('CutRules.txt', 'w')
f4 = open('FinishedRules.txt', 'w')
AllRules = f1.readlines()
AllUcf = f2.readlines()
for i in range(len(AllRules)):
      Rule = AllRules[i]
      UcfR = AllUcf[i]
      Rule = Rule.split()
      Rule.remove("If")
      # if("and" in Rule):
      #     Rule.remove("and")
      Rule = remove_values(Rule, "and")
      Rule.insert(Rule.index("then")+1, "fill")
      Rule.insert(Rule.index("then")+1, "fill")
      RuleList = []
      ChosenRule = 0
      for a in range(0, len(Rule), 3):
            if(Rule[a] == "then"):
                  RuleList.append(Rule[a])
            else:
                  RuleList.append(Rule[a] + Rule[a+1] + Rule[a+2])
      UcfR = UcfR.replace("[", "")
      UcfR = UcfR.replace("]", "")
      UcfR = UcfR.replace(",", "")
      UcfR = UcfR.split()
```

```
	DropRules = []
	for j in range(1, len(UcfR)):
		if(len(UcfR) > 2):
			if(UcfR[j] < UcfR[0]):
				DropRules.append(j)
	for k in Rule:
		if(len(DropRules) == 1):
			ChosenRule = DropRules[0]
		elif(len(DropRules) > 1):
			ChosenRule = UcfR.index(min(UcfR))
	if(ChosenRule != 0):
		RuleList.pop(ChosenRule-1)
	FormatedRules = "If"
	for rulePos in range(len(RuleList)):
		RuleString = RuleList[rulePos]
		if("=" in RuleString):
			Location = RuleString.find("=")
			NewString = RuleString[:Location] + " " +
RuleString[Location:Location+1] + " " + RuleString[Location+1:]
			RuleList[rulePos] = NewString
	for b in range(len(RuleList)):
		if(b == 0):
			FormatedRules = FormatedRules + " " + RuleList[b]
		elif(RuleList[b] == "then"):
			FormatedRules = FormatedRules + " then"
		else:
			if(RuleList[b-1] != "then"):
				FormatedRules = FormatedRules + " and " +
RuleList[b]
			else:
				FormatedRules = FormatedRules + " " + RuleList[b]
	WasCut = ""
	if(ChosenRule != 0):
		WasCut = "Was cut"
	f3.write(AllRules[i] + FormatedRules + "\t\t\t\t" + WasCut + "\n" +
"\n")
	f4.write(FormatedRules + "\n")


f1.close()
```

```
f2.close()
f3.close()
f4.close()
```

FinishedRules.txt:



Note that this process only cuts out a part of a rule once if possible. If it was possible to cut more than one part of a rule, they would have to be put into the CleanRules.txt file and go through RuleY_E_Generator.py, UcfCalculator.py, and RuleCutting.py again to cut out another part if necessary

Adem Malone
Tho Nguyen
Matthew Stroble

Weka Apriori association rules generated with unnecessary parts of rules cut out. Only 100 rules shown for both no restrictions and having is_attack restriction.

No restrictions:

If MV303 = 1 then MV301 = 1
If P401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If P601 = 1 and MV303 = 1 then MV301 = 1
If P401 = 2 and P601 = 1 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If MV303 = 1 then MV301 = 1
If P401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 then P401 = 2
If MV303 = 1 then MV301 = 1
If P401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 then P401 = 2
If UV401 = 2 and MV303 = 1 then P401 = 2
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If UV401 = 2 and MV303 = 1 then P401 = 2
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If P601 = 1 and MV303 = 1 then MV301 = 1
If P401 = 2 and P601 = 1 and MV303 = 1 then MV301 = 1
If UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If P601 = 1 and MV303 = 1 then MV301 = 1
If P401 = 2 and P601 = 1 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2

If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If UV401 = 2 then P401 = 2
If MV303 = 1 then MV301 = 1
If P401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 then P401 = 2
If MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If UV401 = 2 then P401 = 2
If UV401 = 2 and MV303 = 1 then P401 = 2
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If UV401 = 2 and MV303 = 1 then P401 = 2
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If P601 = 1 and MV303 = 1 then MV301 = 1
If P401 = 2 and P601 = 1 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2
If UV401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 and MV303 = 1 then P401 = 2
If P401 = 2 and UV401 = 2 and MV303 = 1 then MV301 = 1
If UV401 = 2 and MV303 = 1 then P401 = 2 and MV301 = 1
If P601 = 1 and MV303 = 1 then MV301 = 1
If P401 = 2 and P601 = 1 and MV303 = 1 then MV301 = 1
If MV303 = 1 then MV301 = 1

If P401 = 2 and MV303 = 1 then MV301 = 1
If FIT301 = LT1.72693199 then P401 = 2
If UV401 = 2 then P401 = 2
If UV401 = 2 then P401 = 2
If MV301 = 1 and FIT301 = LT1.72693199 then P401 = 2
If UV401 = 2 and LIT401 = GT787.028992 then P401 = 2
If MV303 = 1 then MV301 = 1
If P401 = 2 and MV303 = 1 then MV301 = 1
If MV301 = 1 and UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 then P401 = 2
If MV301 = 1 and UV401 = 2 and LIT401 = GT787.028992 then P401 = 2
If AIT202 = GT8.9337995 then MV301 = 1
If AIT202 = GT8.9337995 then MV303 = 1
If AIT202 = GT8.9337995 then LIT401 = GT787.028992
If MV303 = 1 then MV301 = 1
If MV301 = 1 and AIT202 = GT8.9337995 then MV303 = 1
If AIT202 = GT8.9337995 then MV301 = 1 and MV303 = 1
If AIT202 = GT8.9337995 and LIT401 = GT787.028992 then MV301 = 1
If MV301 = 1 and AIT202 = GT8.9337995 then LIT401 = GT787.028992
If AIT202 = GT8.9337995 then MV301 = 1 and LIT401 = GT787.028992
If AIT202 = GT8.9337995 and LIT401 = GT787.028992 then MV303 = 1
If MV303 = 1 and AIT202 = GT8.9337995 then LIT401 = GT787.028992
If AIT202 = GT8.9337995 then MV303 = 1 and LIT401 = GT787.028992

Has is_attack:

If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0

If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0

If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0

If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and UV401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and UV401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0

If P401 = 2 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and MV301 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P601 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and P601 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If UV401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and UV401 = 2 and LIT301 = LT1010.598265 and AIT201 = GT128.5087125 then is_attack = 0
If P601 = 1 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and P601 = 1 and MV301 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P601 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0
If P401 = 2 and P601 = 1 and MV303 = 1 and LIT301 = LT1010.598265 and LIT401 = GT787.028992 and AIT201 = GT128.5087125 then is_attack = 0