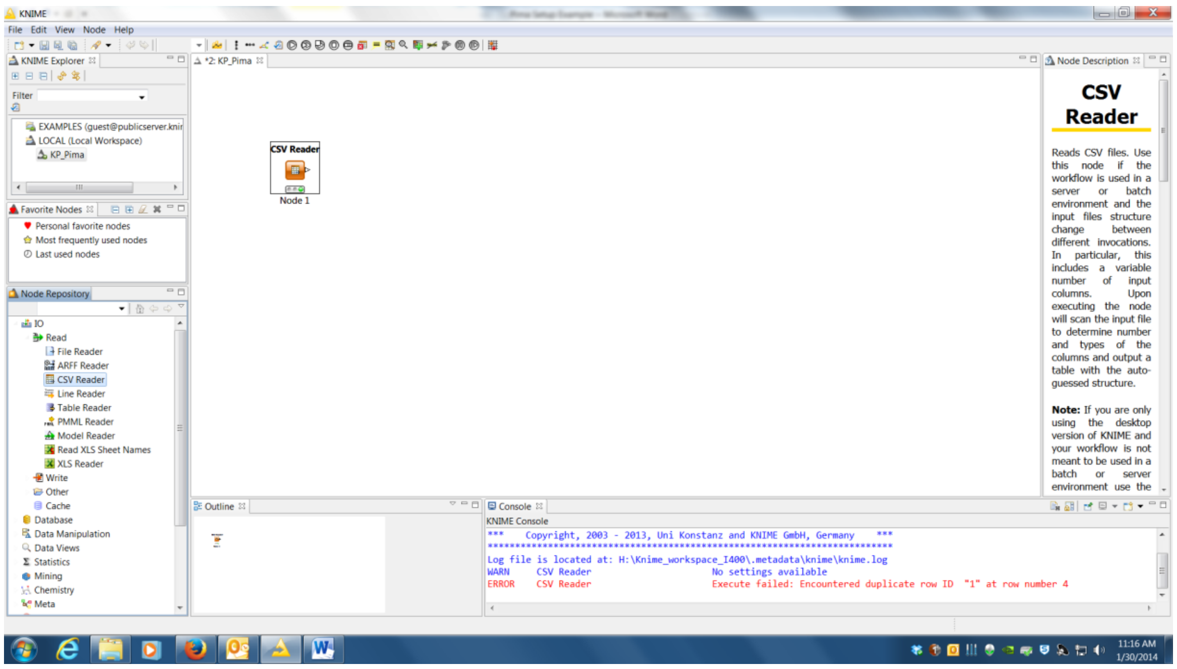
I400 Health Informatics – Data Mining Instructions (KP Project)

Casey Bennett
Spring 2014
Indiana University

1) **Import:** First, we need to import the data into Knime.

- add CSV Reader Node (under IO>>Read)
- right-click and select configure
- browse to file location of csv data file
- uncheck "Has Row Header" box
- Exit out of the configuration screen, right-click and select execute
- If the little traffic light below the node goes to green, then it ran correctly! Good job!



CSV Reader

Reads CSV files. Use this node if the workflow is used in a server or batch environment and the input files structure change between different invocations. In particular, this includes a variable number of input columns. Upon executing the node will scan the input file to determine number and types of the columns and output a table with the auto-guessed structure.

Note: If you are only using the desktop version of KNIME and your workflow is not meant to be used in a batch or server environment use the

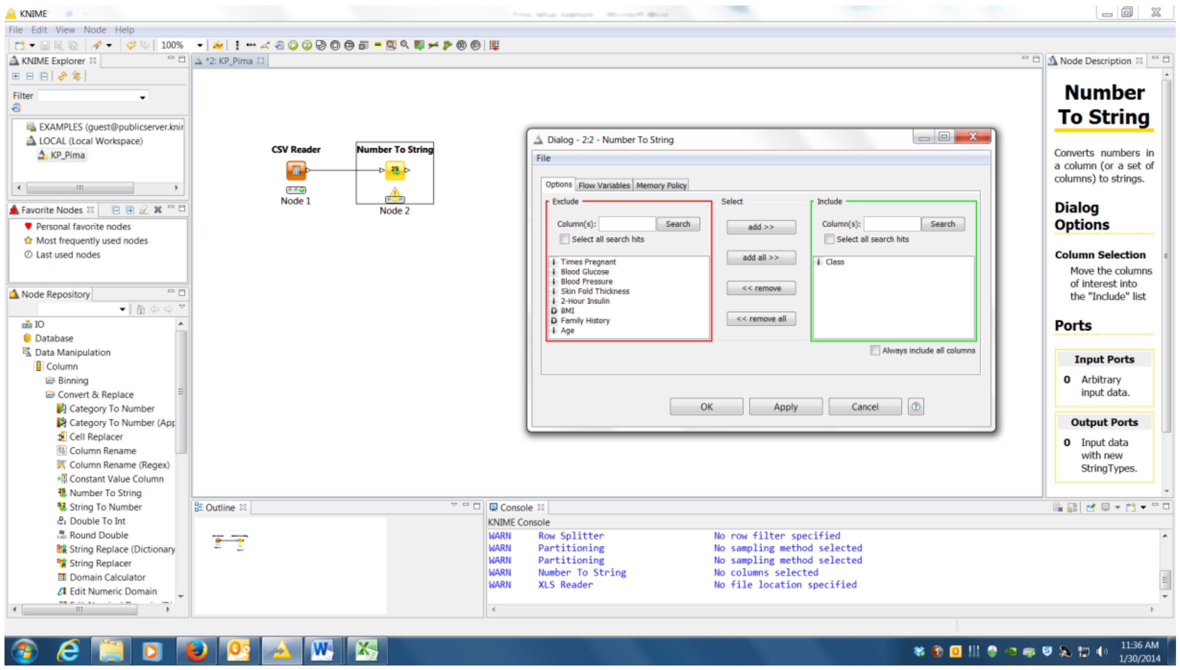
KNIME Console

```
***    Copyright, 2003 - 2013, Uni Konstanz and KNIME GmbH, Germany    ***
**************************************************************************
Log file is located at: H:\Knime_workspace_I400\.metadata\knime\knime.log
WARN     CSV Reader                         No settings available
ERROR    CSV Reader                         Execute failed: Encountered duplicate row ID  "1" at row number 4
```

2) **Adjust Data Types**: Next, we to change the data-types, to make sure that Knime correctly identifies variables as categorical (i.e. strings) or continuous (i.e. numbers). Usually it figures this out on its own, but often in the computer/data world we use 1=yes and 0=No, which confuses most analysis software. So we need to fix this manually. **Hint: You can determine which variables in your dataset are numerical using the Description file (some datasets may have none).

- add Number To String node (under Data Manipulation>>Column>>Convert&Replace)
- connect to CSV Reader node
- right-click and select configure
- Knime will default so that all numbers are converted to strings, using the Description file for your dataset, identify each numerical variable, and remove them from the “Include” list
- Each variables selected should shift over to the left, under “Exclude”
- Make sure the “Class” variable is still on the “Include” side

**In the example below, all the variables except the Class variable are numerical, this is not the case for all datasets



Exclude: Times Pregnant, Blood Glucose, Blood Pressure, Skin Fold Thickness, 2-Hour Insulin, BMI, Family History, Age

Include: Class

Always include all columns

Converts numbers in a column (or a set of columns) to strings.

Column Selection: Move the columns of interest into the "Include" list

Input Ports: 0 Arbitrary input data.

Output Ports: 0 Input data with new StringTypes.

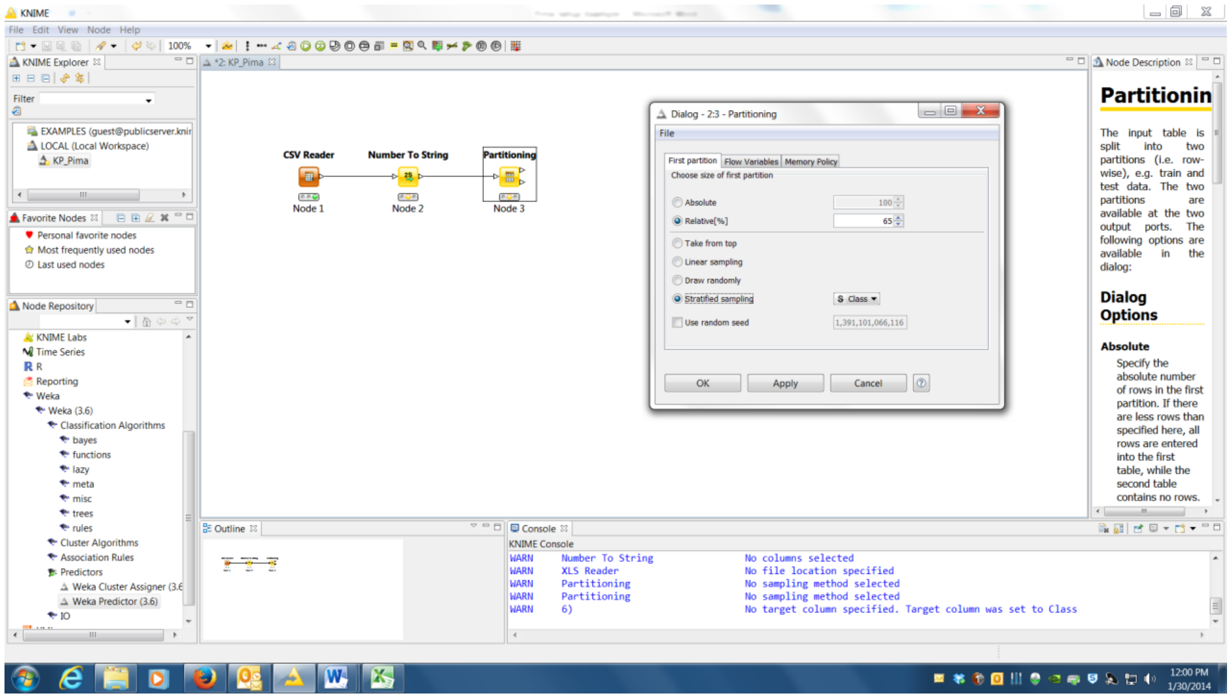
```
KNIME Console
WARN      Row Splitter                  No row filter specified
WARN      Partitioning                  No sampling method selected
WARN      Partitioning                  No sampling method selected
WARN      Number To String              No columns selected
WARN      XLS Reader                    No file location specified
```

3) **Partition**: Now we need to partition the data, which means to divide it up. The reason is that in order to get a valid estimate of the accuracy of any model we build, we need to test the model using data that was *NOT* used to build the model. Using the same data for both steps can result in over-estimates of how good our model is.

We refer to the data used to build the model as the *training set*. We refer to the data used to test the model as the *test set*. Now, technically to do this in the most correct way (and to get the most accurate estimates), we would use an technique called *cross-validation*, which involves dividing the data into a number of equal sets (called *folds*, typically we use 10), training a model using all of the folds but one (i.e. 9 of them), testing on the held-out fold, and then repeating the process 10 times holding each fold out in turn and taking the average performance.

However, to keep things simple, we're just going to use a single training/test set. Typically, this entails using 2/3 of the data to train the model, and 1/3 of the data to test it. We'll round these off to 65% and 35% respectively. So our next step is dividing (i.e. partitioning) our data set.

- add Partitioning node (under Data Manipulation>>Row>>Transform)
- connect to Number To String node
- right-click and select configure
- in the top-half, select the "Relative" radio button, change the % value on the right to 65
- in the bottom-half, select the "Stratified sampling" radio button, make sure the "Class" variable is selected in the drop-down to the right of it



Dialog - 2:3 - Partitioning

First partition | Flow Variables | Memory Policy

Choose size of first partition

- Absolute: 100
- [✓ Relative[%]]: 65
- Take from top
- Linear sampling
- Draw randomly
- [✓ Stratified sampling]: Class
- Use random seed: 1,391,101,066,116

OK | Apply | Cancel

The input table is split into two partitions (i.e. row-wise), e.g. train and test data. The two partitions are available at the two output ports. The following options are available in the dialog:

Dialog Options

Absolute

Specify the absolute number of rows in the first partition. If there are less rows than specified here, all rows are entered into the first table, while the second table contains no rows.

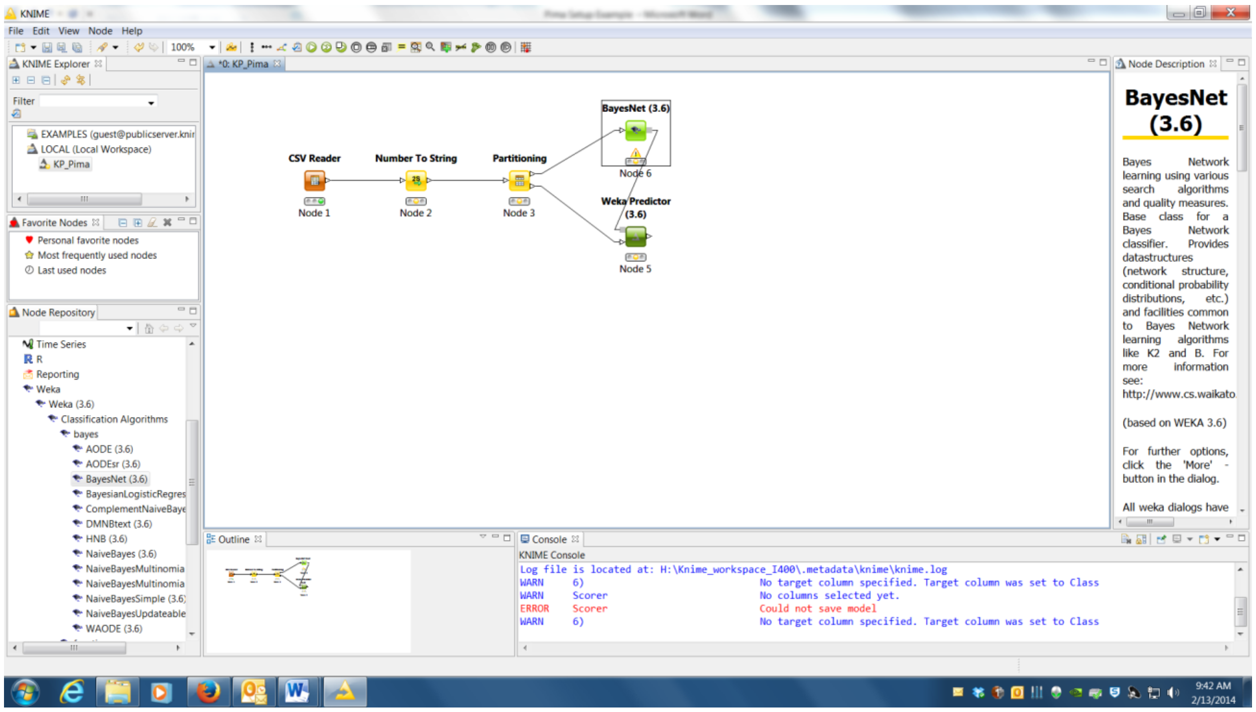
KNIME Console

```
WARN     Number To String              No columns selected
WARN     XLS Reader                    No file location specified
WARN     Partitioning                  No sampling method selected
WARN     Partitioning                  No sampling method selected
WARN     6)                            No target column specified. Target column was set to Class
```

4) **Setup first classifier:** Now we're ready to build some models. There are some other steps we could include, like imputing missing values or pre-discretizing the data, but we'll skip those for now (your datasets are already cleaned up anyway, e.g. there is no missing data). As far as model building, we're going to do three things: build a Bayesian network, build a decision tree, and do some feature selection. We'll do all this using Weka, which is open-source data mining/machine learning library integrated into Knime.

So the first thing is to setup the Bayesian Network (all the directions below are for Weka version 3.6, but if you are installed a newer version, e.g. 3.7, no problem). To do this, we need two things, a classifier node and a predictor node.

- add BayesNet node (under Weka>>Weka3.6>>Classification Algorithms>>bayes>>BayesNet)
- connect to Partitioning node, top output arrow (training data)
- add WekaPredictor node (under Weka>>Weka3.6>>Predictors>>Weka Predictor)
- connect to Partitioning node, bottom output arrow (test data)
- Finally, the gray boxes between the BayesNet classifier node and the WekaPredictor node need to be connected, drag-n-drop a connector, when done it should look like the below



BayesNet (3.6)

Bayes Network learning using various search algorithms and quality measures. Base class for a Bayes Network classifier. Provides datastructures (network structure, conditional probability distributions, etc.) and facilities common to Bayes Network learning algorithms like K2 and B. For more information see: http://www.cs.waikato

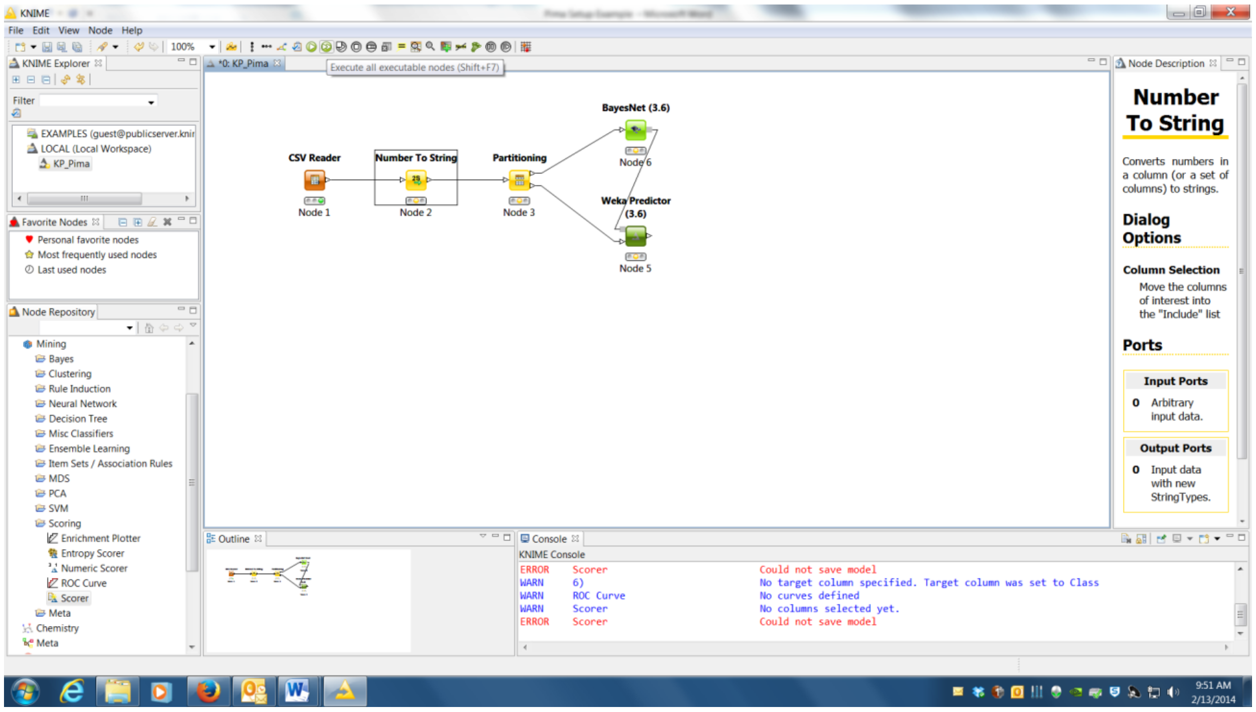
(based on WEKA 3.6)

For further options, click the 'More' - button in the dialog.

All weka dialogs have

```
Log file is located at: H:\Knime_workspace_I400\.metadata\knime\knime.log
WARN     6)                            No target column specified. Target column was set to Class
WARN     Scorer                        No columns selected yet.
ERROR    Scorer                        Could not save model
WARN     6)                            No target column specified. Target column was set to Class
```

5) **Check Progress:** Before we go on, let's make sure what we've setup so far works. Up top along the menu bar, there is a green circle with double arrows on it. If you hover over it, the tooltip should say "Execute all executable nodes". Click on it, if the stoplights under all the nodes turn green, we're in business. Otherwise, come talk to us (or post to the forums) before doing further steps so we can help you get it working.



Number To String

Converts numbers in a column (or a set of columns) to strings.

Dialog Options

Column Selection

Move the columns of interest into the "Include" list

Ports

Input Ports

0 Arbitrary input data.

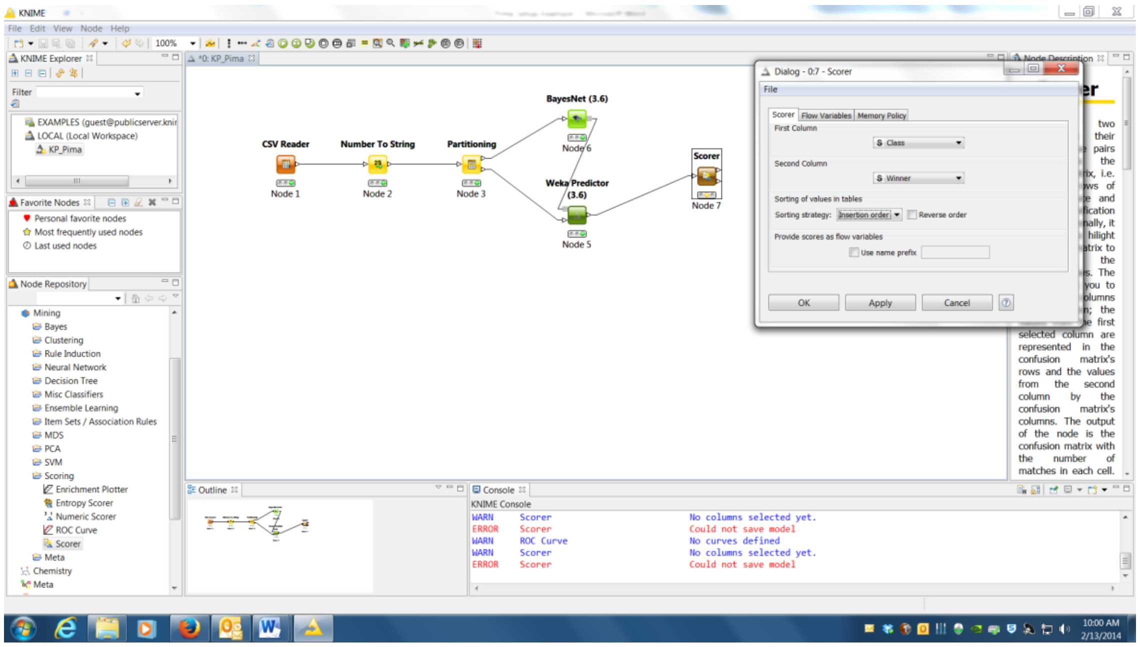
Output Ports

0 Input data with new StringTypes.

```
KNIME Console
ERROR   Scorer                          Could not save model
WARN    6)                              No target column specified. Target column was set to Class
WARN    ROC Curve                       No curves defined
WARN    Scorer                          No columns selected yet.
ERROR   Scorer                          Could not save model
```

6) **Accuracy Scorer:** Before we train any models, we need to be able to "score" them, i.e. tell how good they are. We'll do this using two metrics: accuracy and AUC. I'll explain the differences in class, for now, you can just take them as two numbers which represent scores. It's just like a basketball game, higher scores are better. We'll add the accuracy scorer in this step, and the AUC in step #7.

- add Scorer node (under Mining>>Scoring)
- connect to WekaPredictor node
- right-click and select configure
- under the "first column" section, select the drop-down and change to "Class", the "second column" should be set to winner



Dialog - 0:7 - Scorer

File

Scorer | Flow Variables | Memory Policy

First Column: S Class

Second Column: S Winner

Sorting of values in tables

Sorting strategy: Insertion order ☐ Reverse order

Provide scores as flow variables

☐ Use name prefix

OK Apply Cancel

... two ... their ... pairs ... the ...rix, i.e. ...ws of ...e and ...fication ...ally, it ... hilight ...atrix to ... the ...s. The ...you to ...olumns ...n; the ...e first selected column are represented in the confusion matrix's rows and the values from the second column by the confusion matrix's columns. The output of the node is the confusion matrix with the number of matches in each cell.

Console

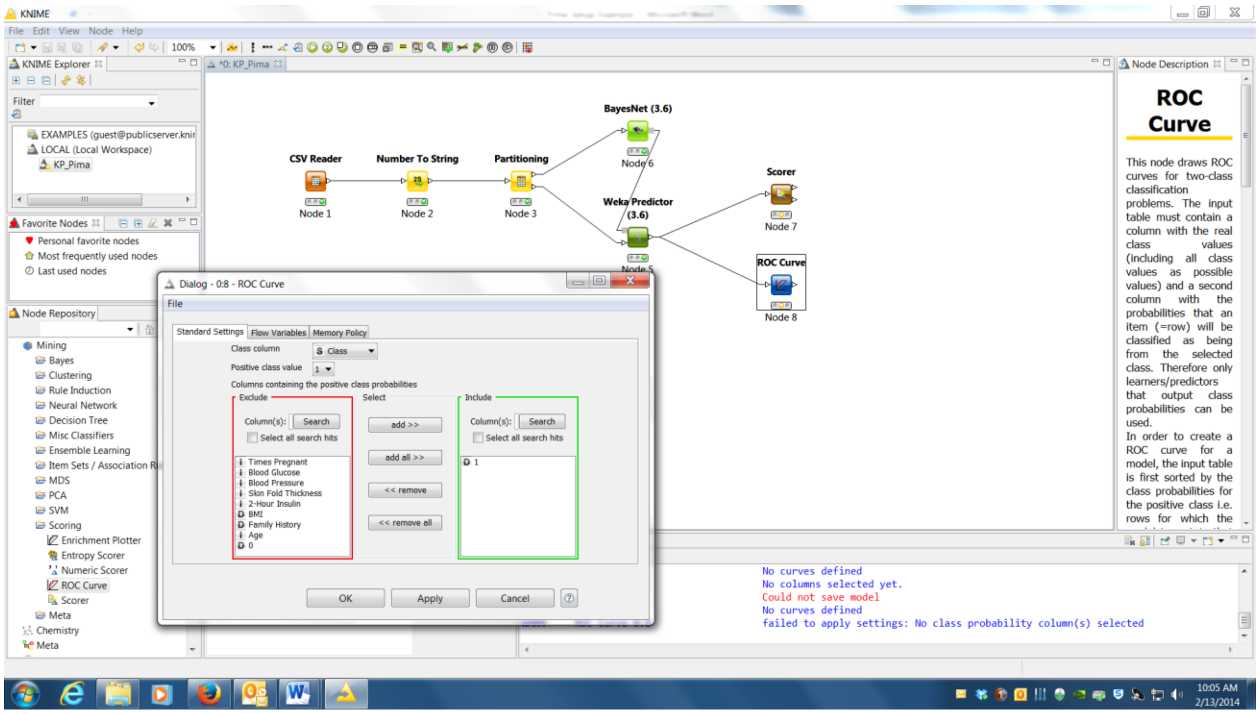
KNIME Console

```
WARN    Scorer                          No columns selected yet.
ERROR   Scorer                          Could not save model
WARN    ROC Curve                       No curves defined
WARN    Scorer                          No columns selected yet.
ERROR   Scorer                          Could not save model
```

10:00 AM
2/13/2014

7) **AUC scorer:** Add the AUC scorer

- add ROC Curve node (under Mining>>Scoring)
- connect to WekaPredictor node
- right-click and select configure
- change the “Class column” drop-down to “class”, change the “Positive class value” drop-down to 1, and in the Exclude/Include lists below, make sure the “1” prob column is added to the “Include” list



ROC Curve

This node draws ROC curves for two-class classification problems. The input table must contain a column with the real class values (including all class values as possible values) and a second column with the probabilities that an item (=row) will be classified as being from the selected class. Therefore only learners/predictors that output class probabilities can be used.
In order to create a ROC curve for a model, the input table is first sorted by the class probabilities for the positive class i.e. rows for which the

Dialog - 0:8 - ROC Curve

File

Standard Settings | Flow Variables | Memory Policy

Class column: S Class

Positive class value: 1

Columns containing the positive class probabilities

Exclude	Select	Include
Column(s): Search	add >>	Column(s): Search
Select all search hits	add all >>	Select all search hits
Times Pregnant	<< remove	1
Blood Glucose	<< remove all	
Blood Pressure		
Skin Fold Thickness		
2-Hour Insulin		
BMI		
Family History		
Age		
0		

OK | Apply | Cancel

```
No curves defined
No columns selected yet.
Could not save model
No curves defined
failed to apply settings: No class probability column(s) selected
```

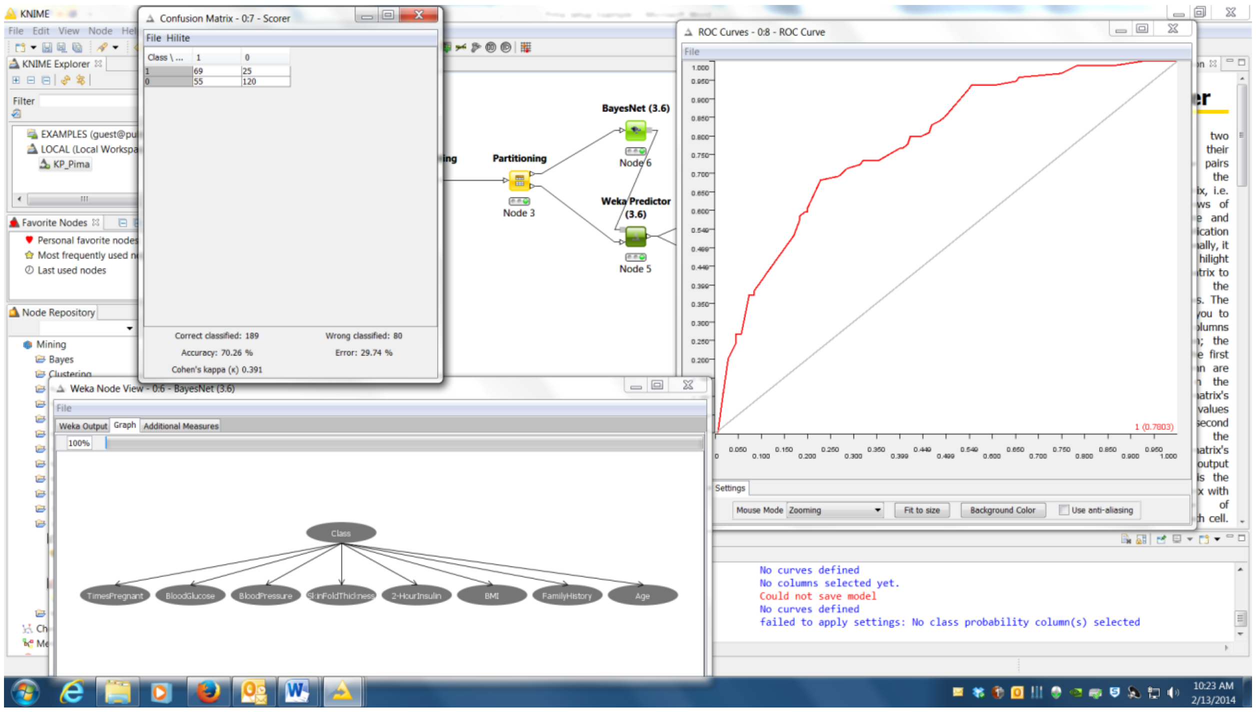
10:05 AM
2/13/2014

8) **Test BayesNet**: Okay let's try running it.

- Click the double-arrow green button "Execute all executable nodes" up along the menu bar, all the stoplights should turn green

We're going to look at three things: a visualization of the model, the accuracy score, and the AUC score.

- Right click on the Scorer node, select "View: Confusion Matrix", the accuracy score is down toward the bottom
- Right click on the ROC Curve node, select "View: ROC curves", the AUC score is will be the number inside parentheses in the lower right corner of the graph
- Right click on the BayesNet node, select "View: Weka Node View", select the "Graph" tab, you should see a visual of the BayesNet you just trained. Clicking on any of the variable nodes in the graph will show the CPT (conditional probability table) for that node. The graph can also be exported as .png or .svg image, but it's often easier to take a screenshot of it.
- HINT: you can move the visualization image around in the window by holding down the left-click and dragging it. Right-clicking will bring up options like auto-scale that may make the graph easier to read.



Confusion Matrix - 0:7 - Scorer

Class \ ...	1	0
1	69	25
0	55	120

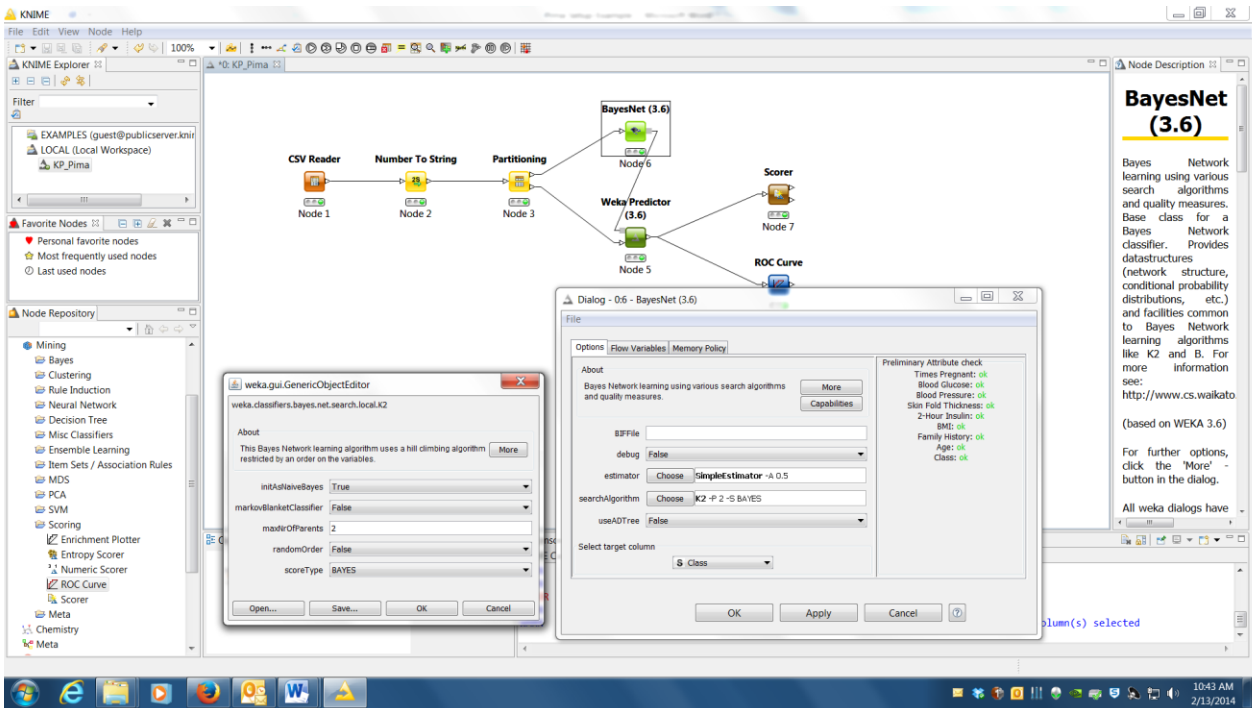
Correct classified: 189 Wrong classified: 80
Accuracy: 70.26 % Error: 29.74 %
Cohen's kappa (κ) 0.391

```
No curves defined
No columns selected yet.
Could not save model
No curves defined
failed to apply settings: No class probability column(s) selected
```

9) **Run BayesNet:** The first thing you'll notice from the graph of the BayesNet above is that all the variable nodes in the graph directly connect to the Class node, and not to each other. Basically, we've trained a Naïve Bayes classifier, which assumes that none of the variables have any conflating/correlated effects. For instance, we are assuming that the risks of higher BMI do not increase with age or vary by gender. To adjust this, we are going to alter the configuration, and rerun the model.

From this point forward, you should record three things for each model: 1) a screenshot of the model visualization, 2) the accuracy score, and 3) the AUC score.

- Close the various views from the last step (Accuracy, AUC, Weka Node view)
- right click on the BayesNet node, select configure
- under the "search algorithm" option, left-click on the white area where it says "K2 -P 1 -S BAYES" (don't hit choose, just click in the white area)
- a new screen will pop-up with a number of options, change the "maxNrOfParents" option from 1 to 2
- Rerun the model, record a visualization, the accuracy, and the AUC score



weka.gui.GenericObjectEditor

weka.classifiers.bayes.net.search.local.K2

About

This Bayes Network learning algorithm uses a hill climbing algorithm restricted by an order on the variables.

initAsNaiveBayes: True
markovBlanketClassifier: False
maxNrOfParents: 2
randomOrder: False
scoreType: BAYES

Dialog - 0:6 - BayesNet (3.6)

About

Bayes Network learning using various search algorithms and quality measures.

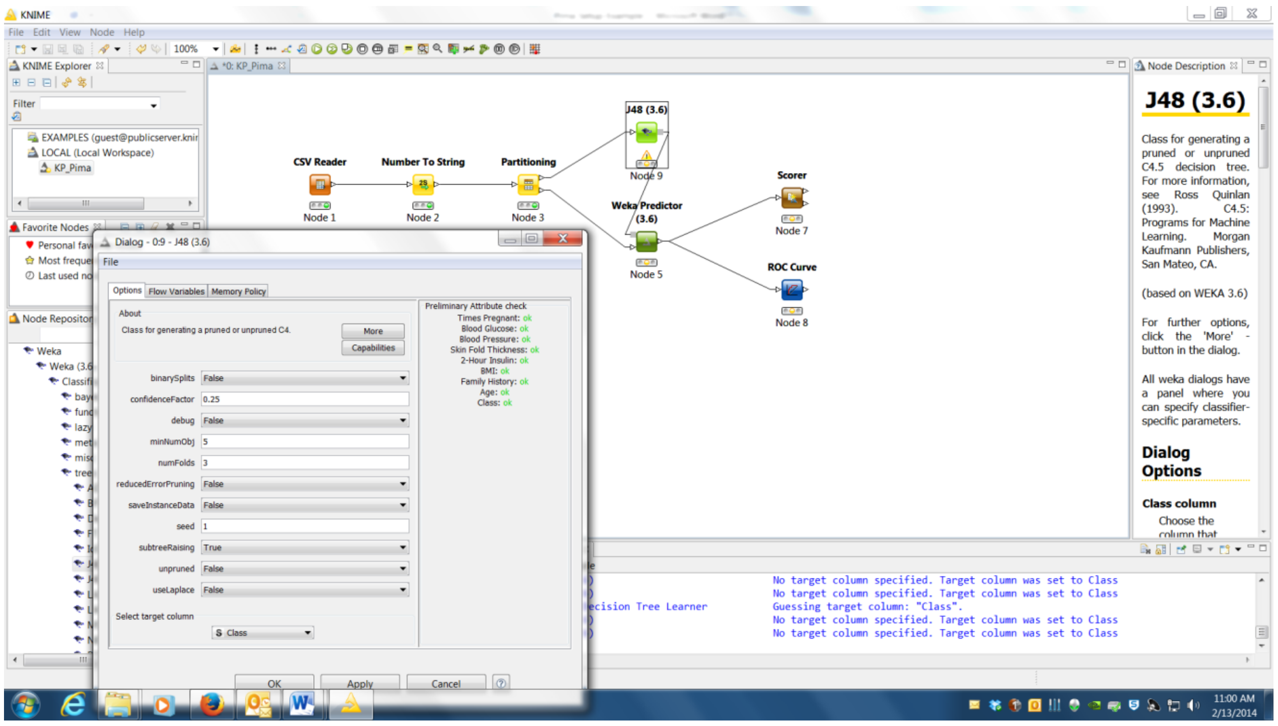
BIFFile:
debug: False
estimator: SimpleEstimator -A 0.5
searchAlgorithm: K2 -P 2 -S BAYES
useADTree: False

Select target column: S Class

Preliminary Attribute check
Times Pregnant: ok
Blood Glucose: ok
Blood Pressure: ok
Skin Fold Thickness: ok
2-Hour Insulin: ok
BMI: ok
Family History: ok
Age: ok
Class: ok

10) **Run Decision Tree:** Now we're gonna do the same thing, but using a simple C4.5 decision tree. C4.5 is one of the "classic" decision tree algorithms. Weka and Knime call it "J48".

- Close the various views from the last step (Accuracy, AUC, Weka Node view)
- Right-click on the BayesNet node, select delete
- add J48 node (under Weka>>Weka3.6>>Classification Algorithms>>trees)
- connect to Partitioning node, top output arrow (training data)
- connect the J48 node to the WekaPredictor node by linking the gray boxes
- right click on the J48 node, select configure
- change the "minNumObj" option to 5
- Run the model, record a visualization, the accuracy, and the AUC score



About
Class for generating a pruned or unpruned C4.

```
binarySplits          False
confidenceFactor      0.25
debug                 False
minNumObj             5
numFolds              3
reducedErrorPruning   False
saveInstanceData      False
seed                  1
subtreeRaising        True
unpruned              False
useLaplace            False
```

Select target column: S Class

```
Preliminary Attribute check
Times Pregnant: ok
Blood Glucose: ok
Blood Pressure: ok
Skin Fold Thickness: ok
2-Hour Insulin: ok
BMI: ok
Family History: ok
Age: ok
Class: ok
```

J48 (3.6)

Class for generating a pruned or unpruned C4.5 decision tree. For more information, see Ross Quinlan (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA.

(based on WEKA 3.6)

For further options, click the 'More' - button in the dialog.

All weka dialogs have a panel where you can specify classifier-specific parameters.

Dialog Options

Class column

Choose the column that

```
)                        No target column specified. Target column was set to Class
)                        No target column specified. Target column was set to Class
ecision Tree Learner     Guessing target column: "Class".
)                        No target column specified. Target column was set to Class
)                        No target column specified. Target column was set to Class
```

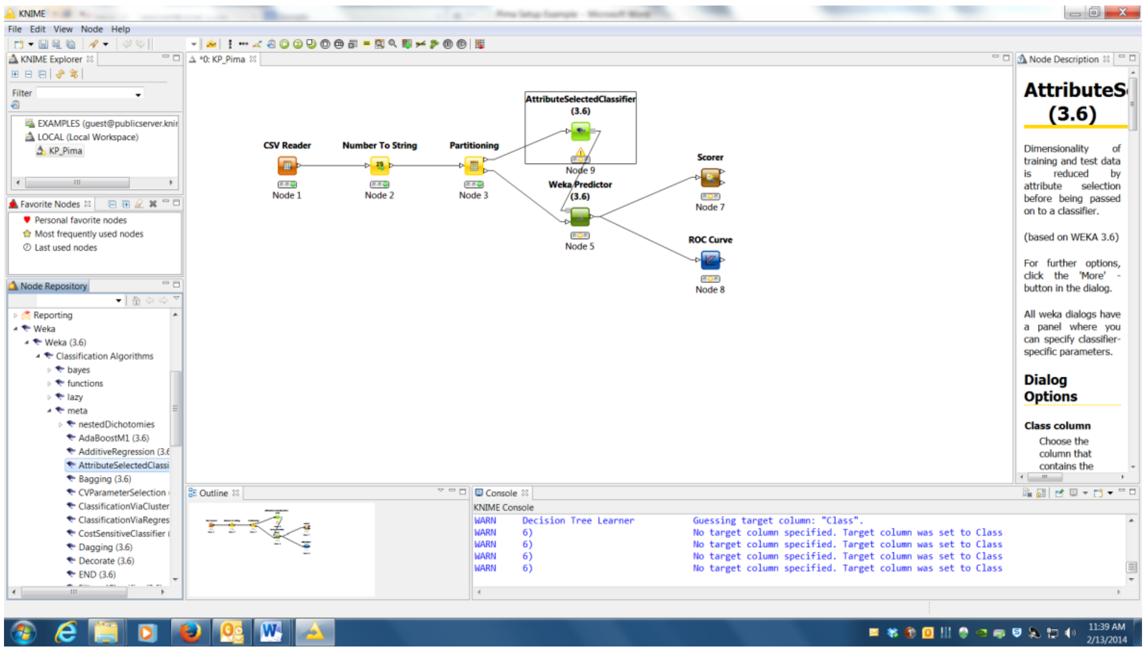
11:00 AM
2/13/2014

11) **Run Feature Selection:** The last thing we're going to do is feature selection ("feature" is just another word for "variable"). One thing we may want to know is whether some variables (e.g. age, gender) actually make a difference in the predictions we might want to make about certain diseases or treatments. Perhaps we are creating a public health screening program, and we want to know which factors to screen for. Maybe we are building a mobile app, and want to know which variables to track. Parsimonious models (i.e. models with the fewest necessary variables) are also easier to understand and explain to other people.

There are many ways to do this – generally the best way is to train lots of models using varying sets of features. We refer to this as a *wrapper-based approach*. However, we can use simpler methods that "rank" the features in terms of expected importance *prior* to building any models. Some of these ranker methods are univariate (evaluate each variably independently, e.g. gain ratio) and some are multivariate (evaluate all variables simultaneously, accounting for correlations, e.g. relief-F). We'll keep it simple here, and use a univariate ranker method, based on a chi-square evaluator (the same chi-square you may be familiar with from statistics). We'll then train another BayesNet model (just like step #9).

Setting this up will take a number of steps, which will be explained over the next several pages.

- Close the various views from the last step (Accuracy, AUC, Weka Node view)
- Right-click on the BayesNet node, select delete
- add AttributeSelectedClassifier node (under Weka>>Weka3.6>>Classification Algorithms>>meta)
- connect to Partitioning node, top output arrow (training data)
- connect the AttributeSelectedClassifier node to the WekaPredictor node by linking the gray boxes



AttributeS (3.6)

Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier.

(based on WEKA 3.6)

For further options, click the 'More' - button in the dialog.

All weka dialogs have a panel where you can specify classifier-specific parameters.

Dialog Options

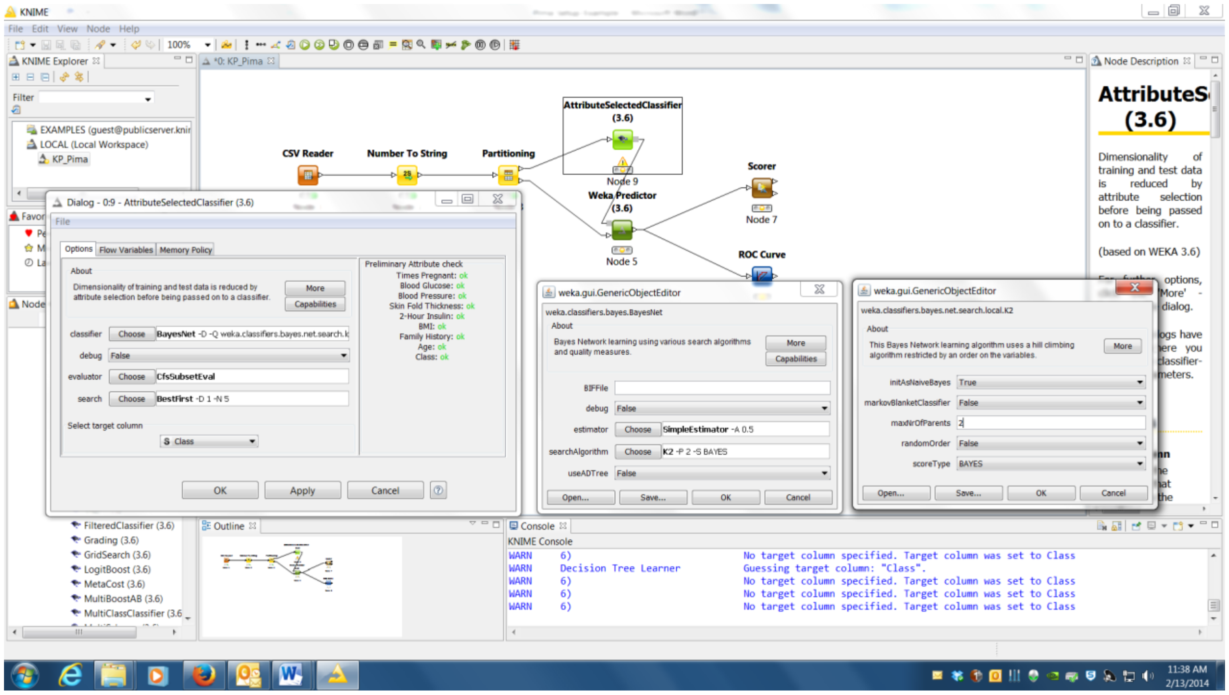
Class column

Choose the column that contains the

KNIME Console

```
WARN    Decision Tree Learner          Guessing target column: "Class".
WARN    6)                             No target column specified. Target column was set to Class
WARN    6)                             No target column specified. Target column was set to Class
WARN    6)                             No target column specified. Target column was set to Class
WARN    6)                             No target column specified. Target column was set to Class
```

- right click on the AttributeSelectedClassifier node, select configure
- for the “classifier” option, click on Choose, and change it to BayesNet (under the bayes)
- left-click in the white area of the classifier where it says “BayesNet”
- a new window will pop-up, left-click on the white area where it says “K2 -P 1 -S BAYES” (don’t hit choose, just click in the white area)
- a new window will pop-up with a number of options, change the “maxNrOfParents” option from 1 to 2 … it should look like the below



Dialog - 0:9 - AttributeSelectedClassifier (3.6)

Options | Flow Variables | Memory Policy

About: Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier.

classifier: BayesNet -D -Q weka.classifiers.bayes.net.search.k
debug: False
evaluator: CfsSubsetEval
search: BestFirst -D 1 -N 5
Select target column: S Class

Preliminary Attribute check
Times Pregnant: ok
Blood Glucose: ok
Blood Pressure: ok
Skin Fold Thickness: ok
2-Hour Insulin: ok
BMI: ok
Family History: ok
Age: ok
Class: ok

weka.gui.GenericObjectEditor
weka.classifiers.bayes.BayesNet
About: Bayes Network learning using various search algorithms and quality measures.

BIFFile:
debug: False
estimator: SimpleEstimator -A 0.5
searchAlgorithm: K2 -P 2 -S BAYES
useADTree: False

weka.gui.GenericObjectEditor
weka.classifiers.bayes.net.search.local.K2
About: This Bayes Network learning algorithm uses a hill climbing algorithm restricted by an order on the variables.

initAsNaiveBayes: True
markovBlanketClassifier: False
maxNrOfParents: 2
randomOrder: False
scoreType: BAYES

Node Description

AttributeS (3.6)

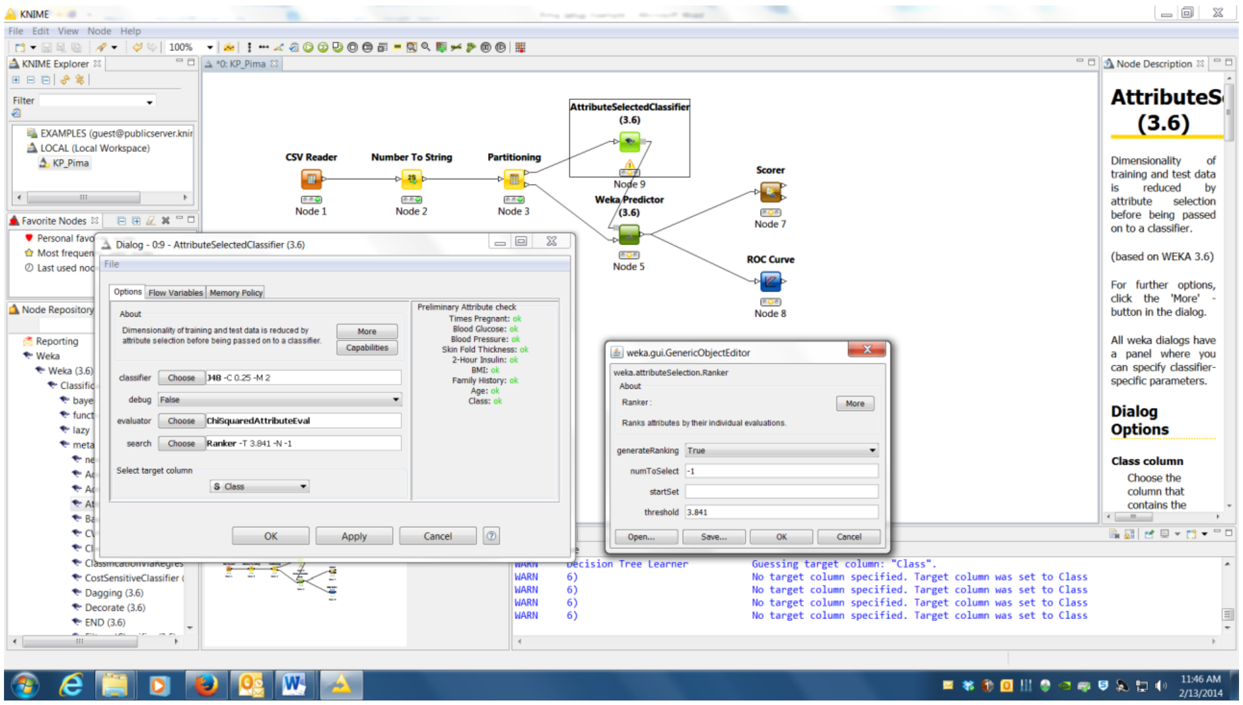
Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier.

(based on WEKA 3.6)

KNIME Console
```
WARN    6)                      No target column specified. Target column was set to Class
WARN    Decision Tree Learner   Guessing target column: "Class".
WARN    6)                      No target column specified. Target column was set to Class
WARN    6)                      No target column specified. Target column was set to Class
WARN    6)                      No target column specified. Target column was set to Class
```

11:38 AM
2/13/2014

- hit “OK” on the two smaller screens you just opened and go back to the main AttributeSelectedClassifier configuration window (the larger gray screen in the screenshot above)
- for the “evaluator” option, click on Choose, and change it to ChiSquareAttributeEval
- for the “search” option, click on Choose, and change it to Ranker
- for chi-square, the significance level for df=1 (degrees-of-freedom, which is 1 since this is a binary classification), is 3.841, left-click on the white area in the “search” section where it says “Ranker –T …”
- a new screen will pop-up, change the “threshold” option to 3.841 (see below)



Dialog - 0:9 - AttributeSelectedClassifier (3.6)

Options | Flow Variables | Memory Policy

About

Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier.

classifier: J48 -C 0.25 -M 2
debug: False
evaluator: ChiSquaredAttributeEval
search: Ranker -T 3.841 -N -1

Select target column: S Class

Preliminary Attribute check
Times Pregnant: ok
Blood Glucose: ok
Blood Pressure: ok
Skin Fold Thickness: ok
2-Hour Insulin: ok
BMI: ok
Family History: ok
Age: ok
Class: ok

weka.gui.GenericObjectEditor

weka.attributeSelection.Ranker

About

Ranker :

Ranks attributes by their individual evaluations.

generateRanking: True
numToSelect: -1
startSet:
threshold: 3.841

Dimensionality of training and test data is reduced by attribute selection before being passed on to a classifier.

(based on WEKA 3.6)

For further options, click the 'More' - button in the dialog.

All weka dialogs have a panel where you can specify classifier-specific parameters.

Dialog Options

Class column

Choose the column that contains the

```
WARN    Decision Tree Learner    Guessing target column: "Class".
WARN    6)                       No target column specified. Target column was set to Class
WARN    6)                       No target column specified. Target column was set to Class
WARN    6)                       No target column specified. Target column was set to Class
WARN    6)                       No target column specified. Target column was set to Class
```

- Run the model, record a visualization, the accuracy, and the AUC score … compare these to the results from step #9
- We also want to note which features made the cut, and which didn't, right click on the AttributeSelectedClassifier node, select "View : Weka Node View"
- Under the "Weka Output" tab, you should find a section labeled "Ranked Attributes" near the top. The chi-squared value is the decimal value on the left, with the variable name on the right. Only features that passed the threshold are included.
- Record these features, including their chi-squared value, also determine which variables were excluded

There are lots of other classification algorithms in Knime/Weka, e.g. multi-layer perceptron neural networks, SVMs, random forests, ensemble classifiers. The R stats package is also integrated. You can play around with these if you want. We chose the algorithms above mainly because they produce pretty pictures (i.e. human-readable), be aware that many algorithms do not.

12) **Bigger Picture:** These kinds of approaches are utilized in the healthcare industry to solve various problems and/or gain insight into challenging questions. Some examples (doing very similar things to what you did above) are below:

- Casey Bennett, Tom Doub, and Rebecca Selove (2012) "EHRs Connect Research and Practice: Where Predictive Modeling, Artificial Intelligence, and Clinical Decision Support Intersect." *Health Policy and Technology*. 1(2): 105-114. http://www.caseybennett.com/uploads/HLPT11_PublishedVersion.pdf

- Casey Bennett, Tom Doub, April Bragg, et al. (2011) "Data Mining Session-Based Patient Reported Outcomes (PROs) in a Mental Health Setting: Toward Data-Driven Clinical Decision Support and Personalized Treatment." *Proceedings of the IEEE Health Informatics and Systems Biology Conference*. 229-236. http://www.caseybennett.com/uploads/Data_Mining_PROs_Final__Arxiv.pdf

- Casey Bennett and Kris Hauser (2013) "Artificial Intelligence Framework for Simulating Clinical Decision-Making: A Markov Decision Process Approach." *Artificial Intelligence in Medicine*. 57(1): 9-19. http://www.caseybennett.com/uploads/Bennett_AI_ClinicalDecisionMaking__Article_in_Press_.pdf

The things you did above are also very similar to the techniques used by IBM's Watson to learn from data!

- David Ferrucci, et al. (2013) "Watson: Beyond Jeopardy!" *Artificial Intelligence*. 199: 93-105. http://www.researchgate.net/publication/237052357_Watson_Beyond_Jeopardy!/file/60b7d51b0ef16d8fa8.pdf