Attribute Selection Methods:

CFSSubsetEval:
CFSSubsetEval stands for Correlation-based Feature Selection Subset Evaluator. Simply, its goal is to find a subset of attributes which have a high predictive ability for the class attribute while having a low correlation between each predictor attribute. More specifically:

> CFS is a simple filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function. The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features.
>
> Quoted from page 69 of the following thesis paper:
> https://www.cs.waikato.ac.nz/~mhall/thesis.pdf

GainRatioAttributeEval:
GainRatioAttributeEval works by evaluating an attribute's gain ratio with respect to the class attribute. The specific formula is shown here:

> https://weka.sourceforge.io/doc.dev/weka/attributeSelection/GainRatioAttributeEval.html
>
> GainR(Class, Attribute) = (H(Class) - H(Class | Attribute)) / H(Attribute)

CorrelationAttributeEval:
CorrelationAttributeEval measures an attribute's worth by measuring the Pearson Correlation Coefficient between the attribute and the class.

InfoGainAttributeEval:
InfoGainAttributeEval works by evaluating an attribute's information gain with respect to the class. The specific formula is shown here:

> https://weka.sourceforge.io/doc.dev/weka/attributeSelection/InfoGainAttributeEval.html
>
> InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute)

ClassifierAttributeEval:
ClassifierAttributeEval evaluates an attribute's predictive power by using a user specified classifier. In this case, I've chosen to use a Naïve Bayes classifier. This will use Naïve Bayes to determine the relationship between each predictor attribute and the classifier attribute.

Classifier Models Used:

Naïve Bayes:

A Naïve Bayes classifier uses Bayes' Theorem to perform a probabilistic prediction of class membership based on the presence of a particular attribute in a particular class. It is called Naïve because we assume class-conditional independence. This means that there is no dependence relation among non-class attributes. This assumption allows us to greatly reduce computation cost.

Logistic Regression:

Logistic regression substitutes our dependent variable for a function of our dependent variable, called the logit function. It is expressed as a linear function of the independent variables. We use regression to estimate the coefficients of our logit function and then our logit function can be substituted into our logistic function. We can then calculate p, the probability that a particular object belongs to a particular class.

Multilayer Perceptron (Neural Network):

Put simply, a neural network is a set of connected input and output units where each connection has a weight associated with it. There is usually at least one hidden layer between the input layer and output layer. This hidden layer returns the inputs to the output layer, which will then inform us of the most likely classification. Each node in the graph will have an activation function (typically logistic or sigmoid) that will help classify the object. In a well-trained network, the output layer should only have a single node that returns close to 1, with all others returning close to 0. The training is done via a learning algorithm called backpropagation. The basic overview of which is adjusting the weights at the connection between each layer and then propagating those adjustments backward to continue adjusting the weights until we reach the input layer. This is done by adjusting the weights to minimize the mean squared error between the network's prediction and the target value for each training tuple. Each adjustment should move the network towards being more capable of making accurate classifications.

J48:

J48 is a decision tree classification algorithm that uses the ID3 algorithm in order to build a decision tree. It works by attempting to split the data into subsets on each attribute and then determining which attribute split causes the maximum amount of information gain. It does this iteratively until all attributes have been split on. It has two notable issues, the first is the potential for finding a local optima instead of a global one. The second, is the potential to overfit to the training data. It is for this reason that smaller decision trees are preferred over larger ones. The tree may also be intentionally pruned as well.

Random Forest:

A random forest is an ensemble learning method that takes advantage of the power of decision trees while attempting to mitigate their disadvantages. The algorithm generates k sample training sets and builds a decision tree on each sample (the name forest comes from its use of multiple trees). While building each tree, a random subset of attributes is selected at each node with the test attribute for the node being selected by some criterion. Classification is then done by majority vote from all trees. The advantage of a random forest (over a single decision tree) is it avoids overfitting and is less susceptible to errors and outliers.

Evaluating a classifier/choosing a best model:

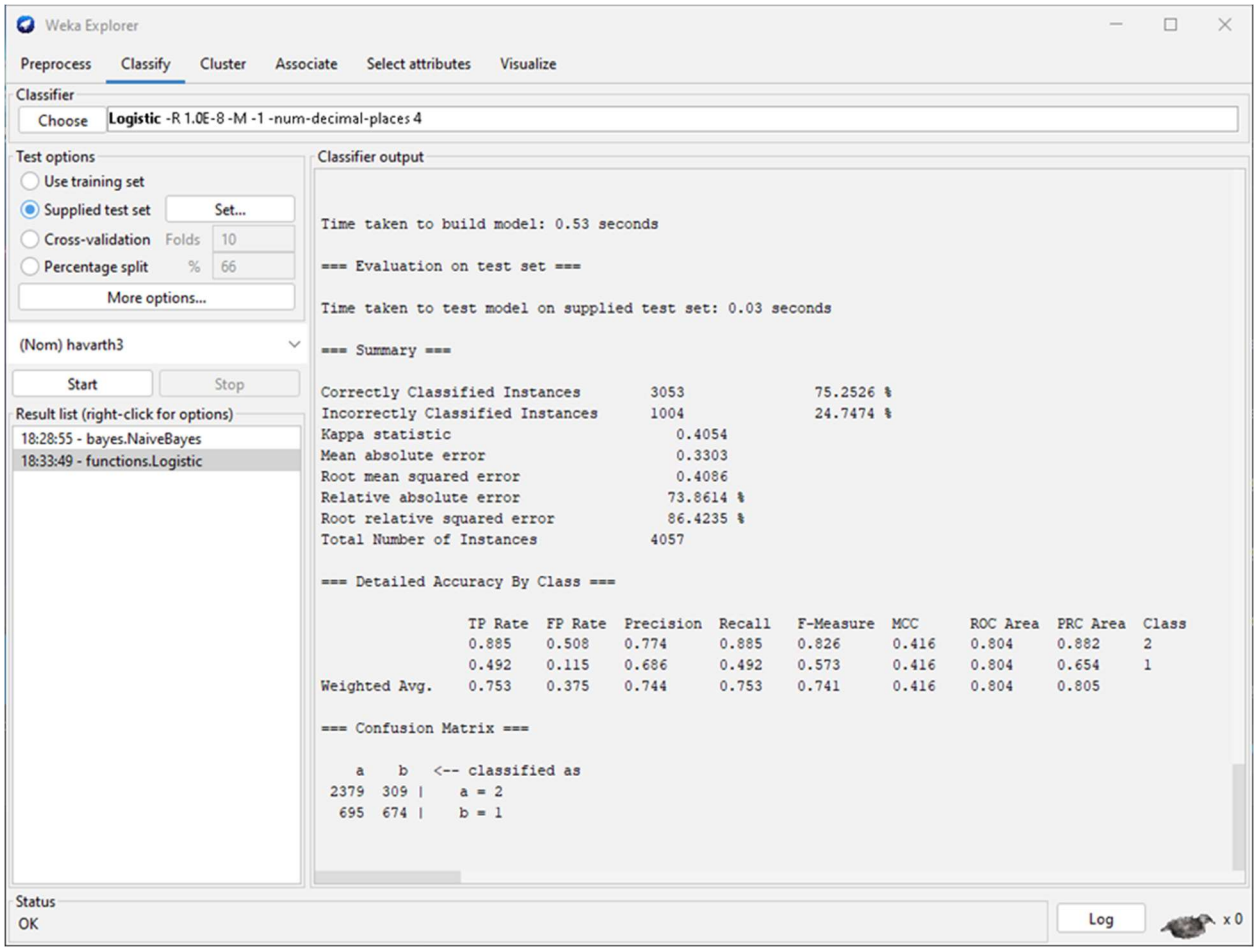
This is a difficult task and is not as simple as selecting the model that had the highest predictive accuracy. Ultimately all of the models were within roughly 5% accuracy of every other, so we must look at other metrics to determine which model best suits our purposes. We also must take into account what it is this data set is classifying as well as how imbalanced the data set is. A higher imbalance means certain metrics will be more or less beneficial in determining the model's effectiveness.

This is a dataset that is classifying whether a particular patient has a specific medical diagnosis. As such, the trade off between true positive rate (TPR) and false positive rate (FPR) is very relevant to us. This tradeoff is measured by the area under the Receiver Operating Characteristics (ROC) curve. We will use this to our advantage in determining the 'best' model. We still want to take overall accuracy into account, as the model that had higher predictive accuracy is still, overall, classifying more objects correctly. We will thus try to find the model that has both the highest accuracy and the highest ROC area.

Using these metrics to determine a 'best' model, we determine that the model made using Logistic Regression with the attributes selected using cfsSubsetEval is the best model. It had the highest accuracy of all models built while also having the highest ROC area.

The attributes in best-train are the following:
employ1, children, deaf, pneuvac4, diffwalk, diffdres, diabete3, physhlth, genhlth, persdoc2, checkup1, chcocncr, chccopd1, cvdcrhd4, x.age.g, x.age80, x.age65yr, x.rfhlth, x.exteth3

The performance measures from the Logistic Regression model built on the cfsSubsetEval dataset is as follows:



Choosing 5 most relevant attributes:

The easiest way to narrow down which attributes are most relevant to the class attribute is to simply count how often attributes appear in the subsets used to create the models. Using this approach, there are 7 attributes that are contained in every subset. Those are:

employ1, diffwalk, chccopdl, x.age.g, x.age80, x.age65yr, x.rfhlth

While it is difficult to narrow the attributes down further (from 7 to 5), we can attempt to do so as 4 of the 5 attribute evaluators used generated a metric to determine how relevant a particular predictor attribute is to the class attribute. With this in mind, we similarly count how many of those attributes had higher values in their respective metrics. The 5 attributes that seem to be most relevant are:

emply1, diffwalk, chccopdl, x.age80, x.age65yr

What was learned:

For me, what was most learned is just how subjective the field of data science can be. This project asked us to create 25 different models, 5 models each from 5 different subsets of the original data and ultimately none of them was clearly better than any of the others. Ultimately, it required more understanding of weka's outputs and the information generated to be able to determine a 'best' model. For these examples, the model with the best ROC area just happened to also have the highest accuracy – that is by no means a guarantee.

Even after deciding upon a 'best' model for this project, I'm still putting best in quotation marks because it is a subjective best. It is me, as the data scientist, saying that I believe it is the model that will give us the best predictive accuracy for the purposes of the dataset and the model. If this dataset had been for a different purpose or had a different make up of tuples – for example had been highly imbalanced – it would be even more important to look at other metrics, such as the Matthews Correlation Coefficient (MCC), which is a more accurate correlation measurement for a dataset with highly imbalanced classes.

I really appreciate what I believe to be the purpose of this project. Not only does it ask us to make use of a lot of what we've learned throughout this course, it also – as mentioned – really shows exactly why having a strong understanding of the science behind these algorithms and the output data is very important. It is rather simple to learn how to run these algorithms with a little googling; truly understanding what it all means is entirely different.

Other Observations:

After creating all these different subsets and then being asked to pick 5 predictor attributes I believe are most relevant to the class attribute, it got me wondering if using some sort of bagging method for attribute selection would be beneficial here. The answer to a lot of questions in this field seem to be something along the lines of "you have to use your best judgement." Knowing that, and thus knowing that picking a true 'best' set of attributes is going to be subjective, I'm inclined to say that the answer is yes: we could run a series of attribute selection algorithms and then use a meta bagging algorithm to determine a subset of attributes that are most likely to improve our final model.