Simple Modeling Techniques in R: Return of the Yeast Set

Daniel Kruze

CSCI48900: Data Science

March 7, 2023

The Problem

The microbial yeast sequence data set, which is the same set used for the first miniproject, is a set that describes a large group of samples of various yeast bacteria. The set's tuples represent a yeast colony given by its name and a set of 8 measurements based on various protein sequences within members of that colony. Additionally, each yeast has a "class" attribute that takes from one of nine options depending on the organelle from which the proteins were measured; for example, one yeast may be class "MIT," meaning that the sample from that colony that was used for the given measurements came from the mitochondria of the bacteria.

The idea behind analyzing the data set, for the most part, is that the class of the sample could be predicted (within certain restrictions) using the measurements included per tuple. This would, of course, require the data to be partitioned significantly in order to use simple models (which will be described later) such as what is expected of this analysis.

In order to model the data simply, then, this data must be cleaned to include some binary attributes. The set will need to be cleaned into a smaller group including only those classes that are represented the most, and the predictions can be made for groups of two classes from within that subgroup. Therefore, the set from which the models will pull will include the subset of the original set such that only yeasts belonging to classes represented 100 or more times will be the only yeasts considered. Such classes would be the ME3 class (for membranes with no N-Terminal protein signals,) the MIT class (for mitochondria,) the NUC class (for nuclei,) and the CYT class (for cytoskeletons.) These classes are represented by 163, 244, 429, and 463 samples respectively, out of the original set of 1,484 samples. Additionally, this cleaning will remove duplicates, although in the entire set there are only about 22 non-unique samples. [1]

Models to Be Used

3 Different techniques should be applied, so as to observe their differences in accuracy. The first method will be logistic regression with a binomial system, which will require making subsets for the given cleaned set. These subsets consist of samples belonging to 2 classes at a time, and they can be tested using R's built-in glm() function, as part of the stats library. This model will use the deviations between the measurements (continuous variables) for each instance as variables in a system of Bernoulli equations, from which a line of best fit can be plotted for a graph of the given data in order to predict accurate values for the other data.

Additionally, a shallow neural net (from the R library neuralnet,) can be constructed with only a few repetitions to attempt to process the set of given

measurements into a sequence for a given class. The basis for this neural net will essentially be a logistic regression equation, given as a simple sum using subsets of classes as singular objects. Each class (from the cleaned subset) should be partitioned into its own variable, and samples at random of the data will be used for training, with the class-specific variables being treated as a composite, categorical variable in the given formula (rather than being cast as a binary variable, like in the logistic regression model.)

Lastly, a decision tree can be built such that the given class predicted by the measurements provided is cast as a binary yes/no decision based on specific, continuous values. By repeatedly sampling the entire data set and creating individual logistic regressions for each sample, a series of binary decisions can be made based on the likelihood of certain attributes having certain values for a given class. This can be achieved using the caret library and the rpart.plot library.

Experimental Results

For the first model—the logistic regression model—the results are much less notable for their accuracy with regards to some measurements. Specifically, the model was built using the function given by:

glm(as.factor(Class) ~ MCG + GVH + ALM + MIT + ERL + POX + VAC + NUC, data = dfgauss1/2, family = "binomial")

Where the set “df1gauss1/2” represents two given subsets upon which logistic regression were performed. The first subset included only NUC and CYT class instances, being that they represent the bulk of the data given by the cleaned set. The second set, then, included the ME3 and the MIT instances, which, as two separate groups of instances, have a larger disparity in size and are both represented by far fewer instances per group than the equivalent NUC or CYT groups.

From these models, few relevant statistics can be gathered from the input parameters. The model for the NUC and CYT subset was noticeably far more accurate, with standard error for each measurement (besides the POX measurement) being less than 2. The POX measurement, however, had a standard error reportedly well over 1000, which indicates the presence of significant outliers in the attribute. This would make the POX value a bad attribute around which to try to make predictions using the model, and it would appear that a better choice would be the MIT measurement value, which had the lowest standard error (0.66) and a very low spread (z-value of 0.43) among the attributes given.

The model for the other subset, however, of the ME3 and MIT class instances, had a very poor accuracy. The standard error for 3 of the 8 measurements recorded was over 3000, and every other error was more than 2. This indicates a heavy presence of outliers and a very large spread of data, meaning that using this model to make predictions would likely result in significant failure. Instead, by using the first model (for the NUC and CYT class instances,) a decent predictive data model could be made that may provide relevant statistical results, given by Fig. 1.

The neural net model provides some…interesting results, based on the entire cleaned data set. It is given by the following function:

dfneural.net <- neuralnet(NUCbool+CYTbool+MITbool +ME3bool ~ MCG + GVH + ALM + MIT + ERL + POX + VAC + NUC, data=dfneural.train, hidden=2, rep = 2, err.fct = "ce", linear.output = F, lifesign = "minimal", stepmax = 1000000, threshold = 0.001)

Where the y operands of the formula represent individual attributes (given as a Boolean value) where instances are determined to either be a member of the class corresponding to the equivalent name of the attribute or not. Additionally, "dfneural.train" represents a subset of the parent set made up of random samples (without replacement) from that parent set which are half the size of the parent set. A second subset "dfneural.valid" exists to represent the members of the parent set that are not present in the samples (the other 50%.)

This neural net is wildly inaccurate, due to the low number of repetitions and the large number of input parameters, but note that the low repetitions (only 2) were necessary due to the limitations of processing power on the given hardware (a consumer laptop with an outdated processor and only 8GB of RAM.) From the resulting neural net, however, a computation can be run against the subset of non-sampled data (dfneural.valid) that uses the probabilities calculated [Fig. 2] to attempt to predict Class based on the input parameters. This computation is given by the computer() function, which outputs a prediction function using the neural net given to it and the set of test data set aside earlier. By comparing the predicted values versus the original, non-sampled values, the neural net generated a swath of errors in identification between the four classes, but still manage to predict well over 50% of the given classes accurately, with the lowest accuracy being the CYT instances (which are the largest group in the data set,) with an accuracy of barely 54%.

Lastly, the decision tree's results are primarily evident graphically, as expressed by Fig. 3. The tree itself is created by splitting the parent set into a training set and a non-training set for testing (much like with the neural net.) Once the data has been partitioned, the following function is run to apply a formula (the same one as in the logistic regression model) to the training set a certain number of times:

dtree_fit_info <- train(Class ~., data = training, method = "rpart", parms = list(split = "information"), trControl=trctrl, tuneLength = 10)

Where "trctrl" is a variable using the trainControl() method to represent the aggregation method for the model, and the number of repetitions (3, in this case.)

From the given tree, binary decisions are made at points where the data is predicted to diverge significantly, in this case being when certain measurement thresholds are either too low or too high, with each decision being ordered by how likely a given measurement is different between instances. With this given decision tree, a predict() function can generate a series of

predictions based on testing data (note that in the given function call, the data was set to “training,” which contained 70% of the cleaned data set—the tree is tested using the other 30%,) that can be compared against real values in the original data set. The overall accuracy of this predictive test is approximately 58%, which makes sense considering the comparatively small size of the data set. Comparatively to the neural net, whose testing set was over 150% the size of the decision’s tree’s set, there was far less room for sampling outliers, and there were far fewer predictions to make.

Conclusions

From the initial logistic regression models, the primary conclusion is evident from the graphical representation of the predictive data [Fig. 1] that expresses an accuracy over somewhere around 60%. It could be said that the model is decently accurate, then, for at least the one subset that it pertains to, and according to similar results for the other subset that was modeled (which includes ME3 and MIT class instances,) the error is too high to provide a predictive model that would even reach 50% accuracy when tested. The logistic regression model is clearly very limited, and the inclusion of a large data set with over 1000 values, the limitations become very evident—when the data itself is restricted significantly to give the model a better chance to be accurate, the model still struggles to produce very satisfactory results. Most industries would not accept such a low accuracy for a data set like this, where (in spite of the presence of many outliers,) most of the data is tightly clustered and the data is qualitatively not very sensitive (yeast colonies are not undocumented or difficult to research in any fashion.)

The neural net show similar shortcomings, however many of them are easily attributed to the small scale of the net itself. A large quantity of errors were expectedly noted among the test data (i.e. the non-sampled data from the individual models were being trained between repetitions.) These errors were significant for all four classes being considered, and the accuracy of predictions for CYT, NUC, MIT, and ME3 classes were respectively 54%, 58%, 60%, and 71%. This implies that the smaller groups (classes which were represented by fewer instances) were predicted more accurately—obviously. However, the parent set does contain noticeable outliers, so a model of the same algorithm using different samples could end up being more or less accurate, and obviously more repetitions to train the neural net would be very beneficial.

The decision tree seems to be about as generally accurate as the neural net, with similar accuracies in prediction for each given class. Using the discussion above of the neural net, the respective accuracies of predictions for the four given classes were 52%, 83%, 53%, and 58% for CYT, ME3, MIT, and NUC class instances respectively. This would imply that the decision tree, while technically more accurate qua the testing set it was given, was actually slightly worse at predicting instances of CYT and MIT classes, slightly better at predicting instances of ME3 classes, and surprisingly the exact same for predicting NUC instances. The larger training set and higher number of repetitions allowed

it to predict some parameters with a decent accuracy, but was obviously restricted in some capacities by the quality of the testing set (which was randomly sampled.) The model could then be refined significantly by training it with more repetitions, and it could be more adequately compared to the neural net if the training and testing data were set as equal partitions of the parent set.

From the given models, a few comparative conclusions can be drawn. As expected, the neural net and the decision tree were both more similar to one another than either were to the general linear model with logistic regression. That model could not actually predict values of every given class using the original set, and necessitated the data be partitioned into very specific subsets, which may not be very helpful or statistically meaningful to consider. Not only that, but one of the models for a particular subset was essentially worthless for prediction whatsoever, meaning that this kind of modeling technique may not be a good choice for the given data set. Additionally, the neural net and the decision tree, while both decently accurate, have their own limitations and drawbacks; the neural net is difficult to implement and requires significant processor attention, while the decision tree, simple as it is, may require a lot of training data to actually predict anything very well.

Certainly, then, when deciding on how to model the data in this given set and predict the class of a yeast sample based on its protein-chain measurements alone, it would be ideal to use a less simple data model, like a neural net with multiple dimensions or a large number of repetitions. That, or one might need to simply reduce the number of attributes considered by the model to try and use only those with significant variety, i.e. not considering the POX or NUC values (the POX value was actually not even included in the graphical representation of the decision tree, because it was likely never anything but 0.00 for every instance.) Between the applied models discussed above, the neural net would appear to be the best choice for predictive accuracy, while the decision tree may be the most effective way to generate a visual representation of the trends in the data.

Figures

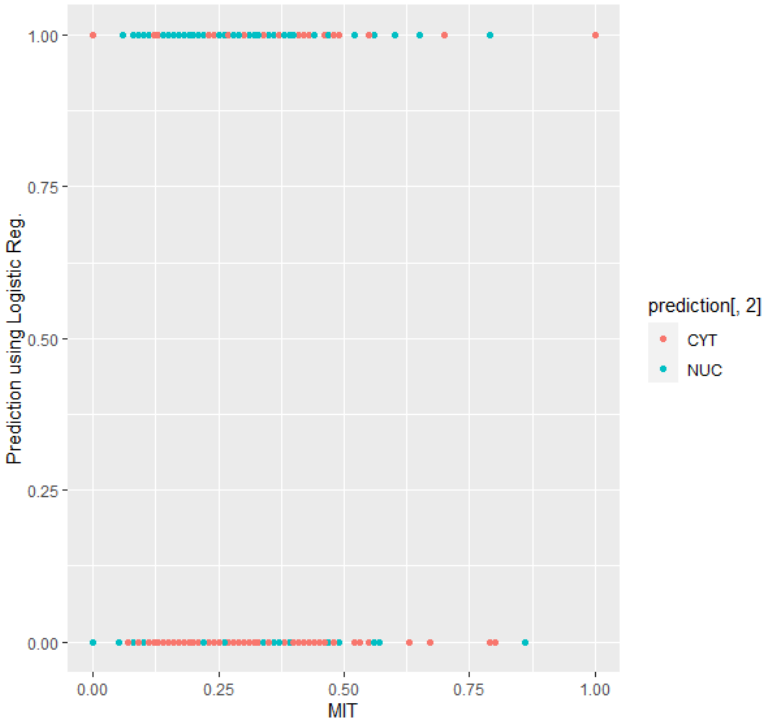

Figure 1

This plot represents the range 0—1 of the related logistic regression model for the subset of yeast samples from class NUC or CYT. The predicted data (y-axis) represents the probability that a sample from this subset will be a CYT or a NUC instance, based on the MIT measurement, using the more accurate of the two models. Note that the probabilities are not necessarily 0 or 1 exactly, but rather much higher or much lower than either upper or lower bound, but are displayed either as high or as low as possible given the scale. What this means is that the model can correctly predict when a sample is a CYT or NUC instance based on their MIT value more than half the time, which is only a slight degree of accuracy.

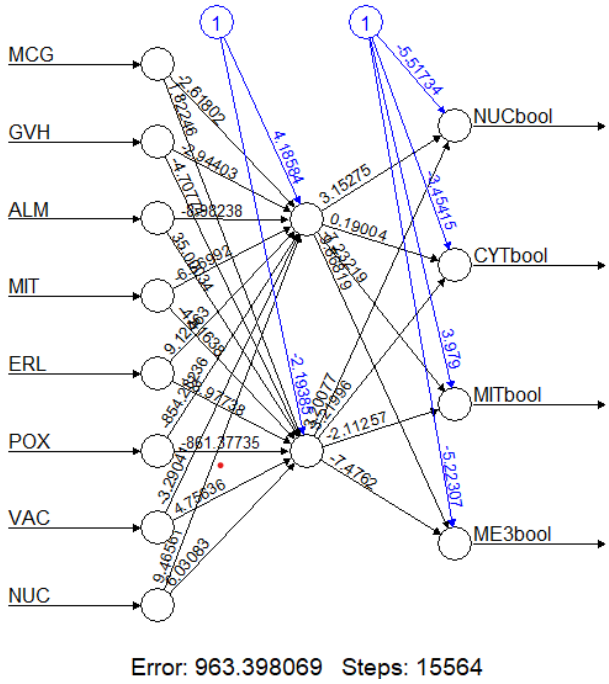

Figure 2

This map represents the numerical components of the neural net and how the probability of an instance's class being recognized from its measurements can change from repetition to repetition. Note, as usual, that the values in blue represent bias probabilities. According to this representation, after the neural net modeled each parameter 2 times using the given formula, predictions of class from the given data are given by their respective weights according to how often each class value is represented by a certain range in the data. Obviously, the error is massive due to the low amount of repetitions and large amount of input parameters. This implies that the neural net is not very accurate whatsoever, and would be particularly inaccurate for prediction—this is also implied by the large bias weights for each parameter.

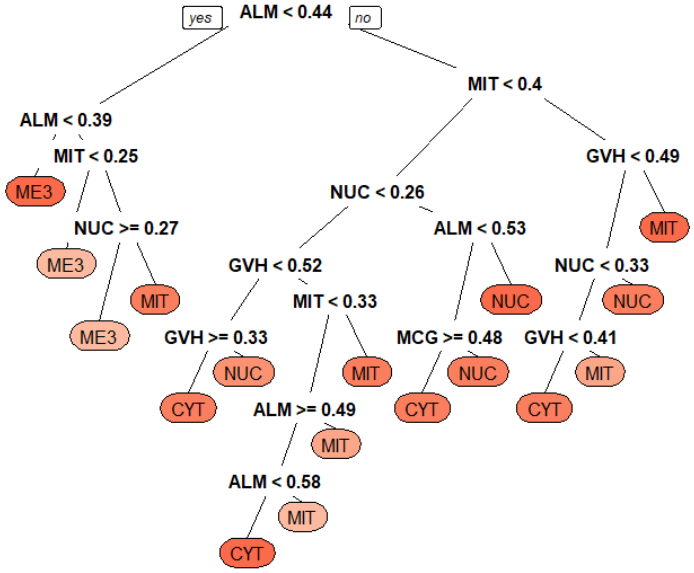

Figure 3

This simple chart shows the resulting decision tree (simplified to exclude counts and weights for each decision) from the given model. As can be observed, there are many breakpoints where decisions can be made, and the saturation of the color of each leaf is proportionally representative to the likelihood that that class will pertain to a given instance. Overall, this implies (from a distance,) that the ALM value is the root decision, i.e. the decision that partitions the data into the most relevant way, balancing the amount of instances on each "side" of the tree (child of the root and resultant subtrees) proportionally to how many instances per class may fall into one of two categories (partitions.) Note that a NUC or CYT class instance *cannot* have an ALM of less than 0.44—this is what the root stands to do, in that it divides all the NUC and CYT instances to one side and most of the MIT and ME3 instances to another. Note also that not every measurement attribute is even included (they may not all be relevant to the given decisions.)

References

1. K. Nakai, “Yeast Data Set,” *UCI Machine Learning Repository*, 01-Sep-1996. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Yeast. [Accessed: 07-Mar-2023].