Modeling Techniques with Bias Mitigation: Yeast Set Reprise, Second Movement

Daniel Kruze

CSCI48900: Data Science

April 4, 2023

The Problem

The microbial yeast sequence data set, as has been so ubiquitous in these reports, 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, is that the class of the sample could be predicted (within certain restrictions) using the measurements included per tuple. Obviously, however, not every kind of class is represented the same way, with some classes being massively overrepresented.

To be brief, the CYT class is overrepresented the hardest with 444 instances [1] alone in the dataset (mind that the data set has had duplicates removed, which removed around 20 members from the raw dataset.) Therefore, the sets from which the models will pull will include the full set of 1462 instances [1], a group of two subsets split around the class parameter (155 CYT instances, 277 other instances,) [1] and then a modified group of those subsets with bias mitigation included. The models will be trained using about 60% of the entire dataset, and then tested using 40% of it.

Models and Motivations

2 Different techniques will be applied, in accordance with the assignment. The first method will be a shallow neural net (from the R library neuralnet,) 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 training set) should be partitioned into its own variable, with the class-specific variables being treated as a composite, categorical variable in the given formula.

Secondly, 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.

Unmitigated Results

For the first model—the neural net model—testing of the entire set provides

some decent results, not unlike what could have been guessed beforehand. The neural net was generated using the following function:

dfneuralnet <- neuralnet(NUCbool+CYTbool+MITbool +ME3bool+ME2bool+ME1bool+EXCbool+VACbool+ERLbool+POXbool ~ MCG + GVH + ALM + MIT + ERL + POX + VAC + NUC, data=dftrain, hidden=2, rep = 5, err.fct = "ce", linear.output = F, lifesign = "minimal", stepmax = 100000, 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, “dftrain” represents a subset of the parent set made up of random samples (without replacement) from that parent set which are 60% of the original size. A second subset “dftest” exists to represent the members of the parent set that are not present in the samples (the remaining 40%.)

According to the results of this net’s predictions, nearly half of the classes in the set were simply not even present in the testing set (ERL, EXC, POX, and VAC classes,) yet those classes were used as predictive results on over 20 out of 585 instances in the set. Of the remaining classes (whose weights are obviously much lower, [Fig. 1]) the results were most accurate for the NUC class instances, and most CYT instances (the overrepresented member) were actually predicted to be CYT, by a factor of about 2 to 1.

The following function is run to apply a formula (the same one as in the neural model) to the training set a certain number of times, in order to create the decision tree:

dtree_fit_info <- train(Class ~., data = dftrainD, 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,) and “dftrainD” is the training set modified to not include the sequence parameter.

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. By using the same predict() function used with the neural net to make predictions in the testing set, it would seem, again, that POX, ERL and VAC class instances are simply not represented at all, although they were predicted to exist slightly fewer times than with the neural net. CYT values were predicted most accurately by far, with NUC values and MIT values close behind—this is consistent, considering those are the three most represented classes. In general, this model was seemingly better at accounting for under and over representation.

Partitioned Results

Using the partitioned subsets of the testing set, results vary. The CYT instances account for about 30% of the entire testing set, so naturally it would

make a big difference for accuracy to remove and isolate it.

As for the neural net, predicting CYT values alone proved just as inaccurate as with the general model, resulting in over half of the CYT instances being predicted as NUC instances. Additional testing of the non-CYT values did, however, involved a *significant* reduction in the predicted CYT values by over 60 predictions (out of 277,) yet a continued presence of the NUC predictions as the most common. Additionally, there were no predictions made for ERL or EXC classes.

The decision tree fared much better (as expected according to the previous results,) with 137 out of 155 CYT instances correctly predicted. In the set of non-CYT instances, NUC samples still make up the bulk of the predictions, with over 150 out of 277 predictions (NUC samples make up about 174 of these samples in actuality, meaning that this was an accurate result. [1]) It should be noted that EXC and ERL instances are still not being predicted whatsoever by this model.

Mitigation Attempt

It is clear from the results that not only is the CYT overrepresented, but that NUC is also slightly overrepresented, and ERL and EXC are underrepresented. This goes for the testing set, yes, but also for the training set, because neither model managed to make predictions with the latter classes, and the neural net made far too many predictions for NUC.

To mitigate this, two things were done to the training set using a probability matrix: the top two most represented values, CYT and NUC, were weighted very low (0.025,) while the bottom two most represented values, ERL and EXC, were weighted very high (0.2.) These changes were *also* applied to the testing set, while both sets remained the same size.

The neural net, after retraining, did show some slight improvement in error and was able to mitigate some of the erroneous predictions of NUC where they were not appropriate. Additionally, more predictions were made for ERL and EXC (roughly 20 each,) which were not always out of place this time as some of the test values were, indeed, instances with such classes. As predicted, MIT and ME3 class predictions were also made more frequently, being that their weight was now comparatively higher than the most represented classes.

The decision tree however yielded some very different results compared to the previous tree. Surprisingly, there were still no predictive results for ERL (or VAC or POX,) represented in the graph, and there were in fact *more* leaf nodes that pointed to CYT. There were, however, *fewer* nodes pointing to NUC, implying that perhaps equating them as the same weight may not have the desired effect, being that CYT is so much more overrepresented than NUC. This was represented during prediction, when predictions of NUC became much more spread out across every class (whether that prediction was right or not,) and there was an increase in predictions of CYT and even MIT instances (MIT being the third most represented class.)

Conclusions

Clearly, the inclusion of weighting affected both models' accuracy, with the

neural net becoming slightly more accurate and the decision tree becoming slightly less. The decision tree for all of the data managed to make decently accurate predictions from the get-go, although the failure to account for the most underrepresented members of the data in the predictions was felt both before and after mitigation, and likewise in the neural net. The neural net, however, showed that the mitigation technique may not have been a poor choice, being that it seemed to benefit from being trained using a more diverse set of data—some predictions that were entirely nonexistent before were now accurate in some cases.

Conclusively it could be said that the weighting technique for mitigation is only moderately successful for the given modeling techniques, and, if employed, will be more useful for neural net generation (which benefits from being trained with diverse data) than for decision trees.

Figures

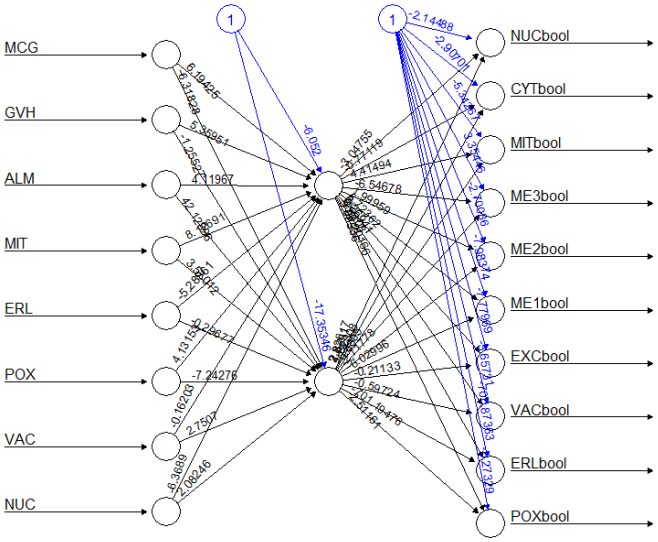

Figure 1

This map represents the numerical components of the neural net generated for the entire data set. Note that the weights are radically low for some of the underrepresented classes (POX, ERL, etc.) but radically high for the overrepresented variables. This implies that weights are placed most significantly on NUC and CYT instances, which is consistent with the predictive results. Curiously, the NUC instances seem to bear the most weight, which is why the net erroneously predicts NUC values very often, especially when they ought to be CYT values. What this means for the net overall is that the NUC and CYT ought to be reweighted manually (as described above.)

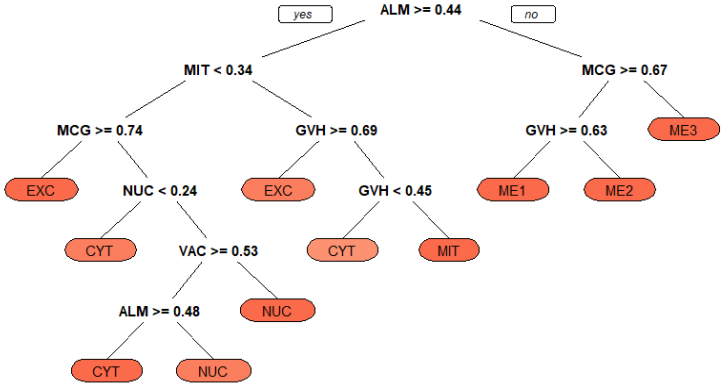

Figure 2

This map shows the traversal through the decision tree for the entire data set. Note that leaves for many of the underrepresented classes are not present, but a leaf *is* present for the EXC class. While there are few leaves for NUC classes in this diagram, the predictive results show a very high accuracy in recognizing NUC instances compared to the neural net. This is likely why the mitigation technique (reweighting) made the tree *less* accurate for those instances, being that they were already counterbalanced by the model, and then ended up being weighted too low. It is worth noting that while the above writing implies that the predictions for CYT were accurate already and then even *more* predictions were made in that direction after mitigation, the increase in predictions represents a *decrease* in accuracy.

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].