BIOS 635 Midterm

Michael V. Edwards
730307019
March 8, 2024

Despite being titled "Identifying Age-Related Conditions", a recent Kaggle competition hosted by InVitro Cell Research presents a dataset without any age variables… or more accurately, without any age labels; The competition presents a dataset of fifty-six anonymously-named variables for use in determining the presence of a condition.[1] Each patient is labeled with one of three age-related conditions (B,D,G) or the absence of any (A). Submissions are judged based on their ability to predict the probability that a patient has any of the three conditions (p_{1i}) or none of the three conditions (p_{0i}), as scored by the balanced logarithmic loss function:

$$\frac{-\frac{1}{N_0}\sum_{i=1}^{N_0} y_{0i}\log\widetilde{p_{0i}} - \frac{1}{N_1}\sum_{i=1}^{N_1} y_{1i}\log\widetilde{p_{1i}}}{2}$$

N_j: true number of patients with status j
y_{ji}: indicator for ith patient's true status j
$\widetilde{p_{ji}} = \max\left(min\left(p_{ji}, 1-10^{-15}\right), 10^{-15}\right)$

The competition also provides a small auxiliary dataset exclusive to the training IDs that contains some dates and categorical variables, notably the distinction between which conditions a patient has. Winning submissions demonstrate the latest machine learning methods, whose performance persists in spite of absent domain information and murky response separation.

I will first select an appropriate model from lecture to get accustomed to the data and practice interfacing with Kaggle. I am especially interested in whether a model improves if it first predicts a patient's specific age-related condition (`Alpha`), or if simply predicting the final binary status (`Class`) is sufficient. As I inevitably approach the limits of lecture model performance, personal experience, and project time constraints, I will appeal to the work of a competition leader to inspire a final attempt.

Data Characteristics

A number of dataset features will be useful for informing my decisions:

- Structure: one observation per subject
- Observations (Complete Cases): 617 (548)
- Covariate Types: 55 continuous, 1 binary
- Covariate Distributions: 9 approximately normal, 42 right-skewed, 0 left-skewed, 4 bimodal
- Response Distribution (Complete Cases): 509 A, 61 B, 18 D, 29 G (446 A, 57 B, 17 D, 28 G)
- Competition Score Range: 0.69314 (uniform guess) to 0.30626 (official winner)

Quadratic Discriminant Analysis (QDA)

I elected to use QDA for my first attempt for the relative viability of its assumptions and its ability to support four response classes. This model assumes that the covariates are normally distributed in each class, but since many of the covariates individually have a skewed distribution, this assumption is strained. Thus, I decided to avoid any further assumptions, such as the equivalence of covariance matrices (for linear discriminant analysis) or conditional independence in each class (for naïve Bayes).

Since the competition's scoring method only requires raw probabilities, there is no need to cross-validate a QDA model over a decision threshold, so I thought fitting this first model would be a quick exercise. In reality, it was an unexpectedly nonlinear and time-consuming process that revealed important considerations useful for fitting any model.

Scoring

Forty-three Kaggle submissions made it clear that the competition would not be able to reliably score my submissions. Since the competition test set has five cases, I opted to designate five random training samples as test samples and score everything myself.

Missing Data

Several of the R functions used in this class, including `predict`, throw errors when operating on missing data. I originally planned to use complete case analysis to ignore the issue and focus on the class material; that is, until I discovered

missing data in the test set. I thus opted to impute any missing data, saving me from throwing out 11% of rows (69/617) via complete case analysis when only 0.4% of cells (131/31,065) had missing data.

There are many imputation methods for continuous variables, but I chose Multiple Imputation by Chained Equations, `MICE` in R, for its performance on data with multiple variable types.[2] I chose the random forest setting because provided a better improvement to my competition score than alternatives like regression trees and lasso linear regression. Being careful not to contaminate the training set with test information, I 1) imputed training values with the full combination of the training set and the auxiliary set, and 2) imputed test values with the combination of the training set and test set stripped of the outcome variable.

Class Imbalance

At least one condition in the full training set has too few observations for `qda` to execute. Even for future models that do successfully perform on the training data, we can expect constraints on cross-validation. In this case, I proceeded using patients' binary outcome status—collapsing classes B, D, and G.

Ultimately, My model predicted that none of the five test cases have any age-related conditions, with probability approximately 1 for each patient. This earned me a score of 9.408226, making my model worse than a uniform guess and worse than 98.9% of official submissions. I then used cross-validation on the training set size to verify that the best QDA performance is on the full training set.

Light Gradient Boosting Machine (LightGBM)

Several degrees of sophistication later, we arrive at the methods of competition leaders. The best-performing unofficial model with public R code was created by user kailex and scores a 0.07467003 with my test set.[3]

LightGBM is responsible for this staggering improvement, and utilizes multiple decision trees that are grown leaf-wise.[4] Leaf-wise growth occurs sequentially and is restrained from unbalanced growth by regularization techniques like Lasso and Ridge regression.[4] When kailex implements this method, they elect to ignore the auxiliary data and predict patients' binary status only. kailex notably does not introduce Lasso or Ridge regularization, but instead caps the number of leaves per tree at 5. Though cross validation is invoked in the code, it is not used for hyperparameter selection, but instead serves to monitor model performance across iterations and halt training whenever there is evidence of stalling or overfitting in the model. This process contributes to the efficiency that LightGBM is known for.[4]

As noted previously, the four true outcome classes (two final outcome classes) are quite unbalanced in the training set. kailex addresses this using downsampling, which is the reduction of samples in the majority class.[5] Specifically, kailex uses the function `themis::step_downsample` with a default ratio of 1 to reduce the number of samples with `Class=0` to that of `Class=1` by removing oversampled cases at random.[5]

kailex does not seem particularly concerned with missing data, as the only direct correction implemented is the imputation of medians for two variables. Surely, we can do better than that. If we implement my imputation approach from before and rebuild kailex's model, the balanced log loss improves to 0.06566658! When I naively switch the `lightlgb::lgb.train` setting from "binary" to "multiclass" and make only essential changes to the code, the model performs quite poorly.[6] It appears several rounds of hyperparameter optimization will be needed to properly test if the model can improve by classifying the true outcome class first.

These results inspire a wellspring of ideas for future iterations and projects. In this project, I was able to compare random forest MICE to a handful of alternatives, but there were several options that failed due to data features like dimensionality—namely KNN and predictive mean matching.[2] Perhaps a form of LightGBM imputation offers the ultimate imputation result. I found it interesting that neither imputation nor model training ever took more than a couple of minutes. I look forward to seeing if this changes with the dimensions of the next competition dataset. Now that I am better prepared to handle data and competition limitations, I look forward to prioritizing cross-validation techniques in my next project, especially after learning from kailex's work. A complete repository of project files can be found on my GitHub page.[7]

1. https://www.kaggle.com/competitions/icr-identify-age-related-conditions
2. https://www.rdocumentation.org/packages/mice/versions/3.16.0/topics/mice
3. https://www.kaggle.com/code/kailex/ic-r-lightgbm-downsampling
4. https://ieeexplore.ieee.org/abstract/document/9362840
5. https://themis.tidymodels.org/reference/step_downsample.html
6. https://search.r-project.org/CRAN/refmans/lightgbm/html/lgb.train.html
7. https://github.com/medwar747/635_Machine_Learning_Midterm