Research Data Scientist - Technical Exercise

Richard J. McAlexander

9/20/2022

Introduction

This memo introduces a novel method for addressing systematicity in ML systems for predicting successful hires for job applicants. The main problem is that if a hiring process uses the same ML model to predict successful hires, individuals who are slightly below the cutoff for the binary classifier (and thus are very similar to candidates who are slightly above the cutoff) will be consistently and systematically not recommended for roles.

The goal is to fix this problem with a method that has the following characteristics. The first is that the method preserves some of the predictive accuracy of the ML models. Replacing the output of a binary classifier with random draws from a binomial distribution will introduce systematicity since each prediction from a model will be completely independent of every other prediction. However, the predictive output will have no relation to the underlying quality of the actual candidates.

The second desirable characteristic is that the method is accompanied by an easily computable and interpretable metric that measures the level of agreement/disagreement between ML models. The third goal is to produce a method itself that computationally efficient and scales well. Below I outline such a method, apply it to a dataset, and discuss limitations, possible extensions, and alternative approaches.

Random Re-ordering to Address Systematicity.

My proposed method is to randomly manipulate subsets of the data and then estimate a model on the randomly manipulated data. The manipulation should occur in the following way. For a subset of columns in the dataframe, select a random subset of rows in the data frame and subtract each value of that row subset from the maximum value of the column. In effect this re-orders random subsets of the data on a reversed scale. For binary variables, what this does is flips the labeling from zero to one and from one to zero. Performing this on on a subset of the actual rows in each column is necessary, since flipping all zeros to ones simply flips the sign on the coefficient for that variable (in parametric models) and has no effect on prediction.[1] This method is superior than simply randomly permuting a subset of the data since random permutations introduce random noise—this method re-orders the data values so that individual candidates that had higher values of a variable (for example, years of experience) now have lower values (and vice versa). Randomly selecting the columns to re-order and randomly selecting the rows within the column (and the size of each) introduces an extra layer of randomness. These parameters could be easily fixed and specified by the user.

For each new training dataset created by the re-ordering, we then train the model. These could be either a series of the same models (say, logit models since they are computationally efficient) or different families of models (for example, a random forest trained on one dataset, a neural net trained on another, etc.). Since random forest models and neural nets are quite different in themselves, the user should not need to estimate as many models as they would if they were only using logits. Thus combining different types of models trained

[1] Note that one should not re-order more than half the rows of any column, since the maximum level of re-ordering introduces in one column is when half the rows are re-ordered. Re-ordering more rows decreases amount of arbitrariness but increases computational complexity.

on different re-ordered datasets should reduce the extent to which each model makes the same prediction as the other.

The main metric for this exercise is the share of models that produce the same prediction, or SMA (share of model agreement) on the held-out test set. If we estimate fifty logit models on randomly re-ordered datasets, then the SMA is the percentage of candidates where all fifty models predict 'hired' or all predict 'not hired'. The SMA describes the output of all the models on all the candidates. If we are interested in understanding which candidates have disagreements among some of the models (so that, for a given candidate, some models predict 'hired' and some predict 'not hired') we can look directly at the models predictions for each candidate to extract the candidate where the model output is not in complete agreement. These would be the candidates who are just below or just above the prediction threshold of the binary classification model.

Figure 1 shows how increasing the number of logit models estimated decreases the SMA. This enables the user to calibrate the desired level of SMA by increased the number of models estimated on separate re-ordered datasets. Note that estimating multiple models will be computationally costly if we move from logits to something more complex, like a random forest.

Another way to increase the SMA to the desired level is to estimate different types of models on different randomly re-ordered datasets. Here I create three datasets that are each randomly re-ordered, and estimate a KNN classification, a random forest, and gradient boosted tree model. Figure 2 shows that the share of agreement is much lower than for the logit models.

Caveats and Alternatives

There are some caveats. Re-ordering multiple variables in random ways that were previously highly correlated may make those variables less correlated, which produces unusual cases or limits the overall predictive accuracy of the model. Similarly, one-hot encoding a categorical variable, and then re-ordering some of the dummy variables that were created from the categorical variable, can produce contradictory cases. One solution to these problems is to perform dimension reduction on highly correlated continuous variables, and then re-order the output of the dimension reduction. For mutually exclusive dummies, simply re-ordering one variable while dropping others may address this. In addition, training multiple models increases computational cost.

An alternate metric instead of simply taking the share of models that agree on their prediction would be to use a metric such as Cohen's kappa coefficient (k). This is commonly used to measure intercoder reliability for categorical items. This is useful especially for comparing the output of multiclass classification predictive models. The kappa coefficient is simply to calculate and accounts for the probability of agreement or disagreement across models happening by chance.

A potential downside of randomly re-ordering the data is that, by introducing random noise, it reduces the overall predictive power of the ML models. This could be a larger risk when datasets are smaller, since models may neglect the information contained in the re-ordered columns in favor of a column that has more relevant information. Less predictive power means less utility for the clients using the software.

Implementation

How would we use this method to actually make predictions about who will be a successful hire? At prediction time, after running models on re-ordered datasets, we take the measure indicating the amount of agreement across models for each candidate and use that to select which candidates will be predicted to be a successful hire. For example, if we have 100 candidates, and 40 of those candidates were predicted to be hired across all our models, and 40 of those candidates were predicted not to be hired across all our models, then we can be fairly confident that those 80 predictions are accurate because they do not change when the data is re-ordered. The question remains: what then do we do with the 20 candidates where some models predicted them to be hired, and other models predicted them to not be hired?

The precise answer is case dependent. One option is to randomly introduce a cutoff that is bounded between the the SMA for the candidate with the highest SMA that is less than one and the candidate with the lowest SMA that is greater than zero. This will produce a cutoff point where some candidates who had disagreements

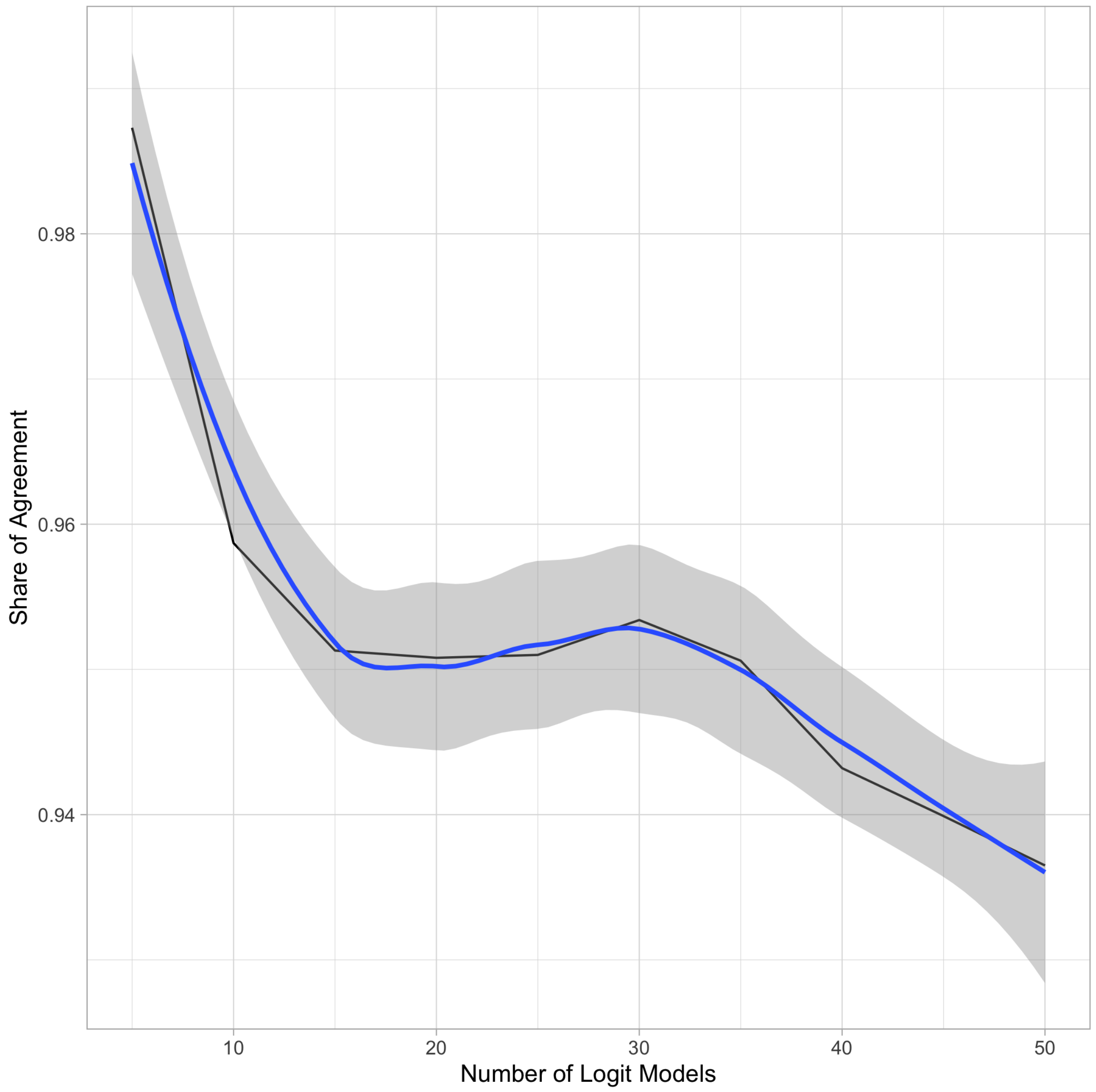

Figure 1: Share of Model Agreement as a Function of Number of Logit Models Estimated

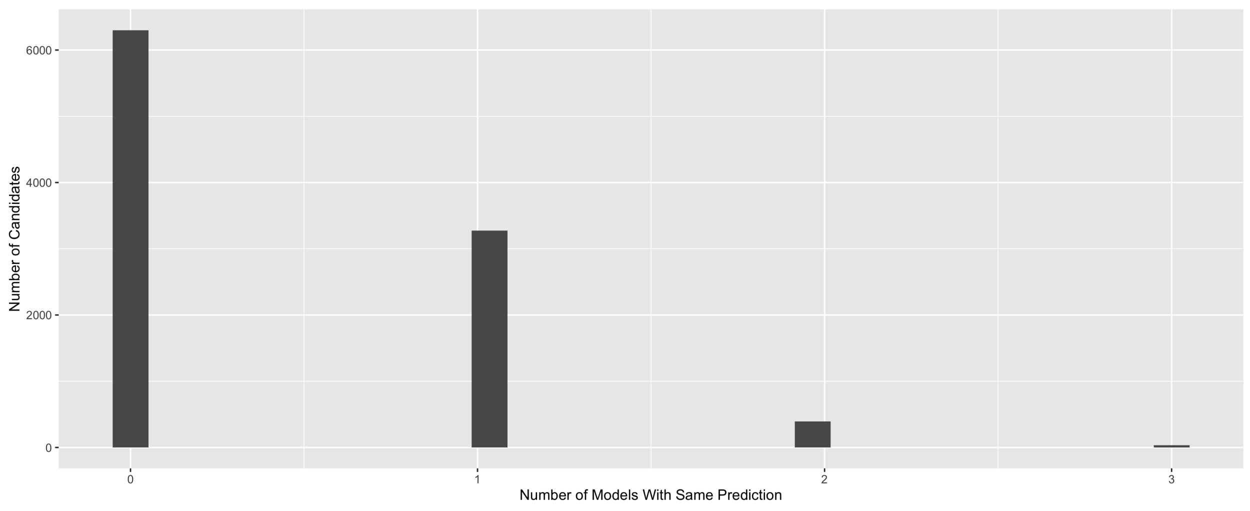

Figure 2: Share of Model Agreement for KNN, RF, and Xgboost Models

among the models are treated as being predicted to be 'hired' if they are above the cutoff, and 'not hired' if they are below the cutoff. The other factor is the number of 'hired' predictions. If as the cutoff is lower, then more candidates will be recommended to be hired. The amount of recommended hires may be determined by the customers needs, and can be increased/decreased by varying the cutoff around the candidate-level SMA.

Conclusion

Here I have an introduced a method to address systematicity in ML models. The method introduces random re-ordering of variables into a training dataset, estimates a model, and then repeats this process a specified number of times. Then, we compute the share of estimated models that have the same prediction for each candidate. This metric can then be used to find out which candidates the models agree should be 'hired' or 'not hired', and which candidates the models disagree on. These candidates are the candidates that are very close to the decision threshold and otherwise would be systematically from being hired if only one model was used for prediction. This method also preserves the predictive accuracy of the models since it preserves the prediction for when all models predict the same value for a candidate.