Abhi Gupta

Report

Building off of the last assignment, the pipeline has been updated to incorporate cross-validation and testing of various parameters for each type of model considered. While some variables were simply truncated last time to deal with excessively heavy tails, now the pipeline includes functions to scale the variables. Additionally, there are now pipeline functions to replace certain values within a given variable (for example, one could replace all the 0 ages). To keep this set of results largely comparable to last assignment’s results, I did the same binning and trimming that I had done previously. As with last time, missing values were filled with the median or mode as appropriate. When evaluating a given model and set of parameters, 4-fold CV was used. The evaluation metrics of interest (which will be discussed below) were averaged across these four runs.

Many of the metrics discussed in class were used to evaluate performance. Precision and recall were calculated at a range of thresholds from .05 up to .8. The former is the proportion of predicted cases of financial distress that were indeed true cases, while the latter measures what fraction of actual cases of distress were predicted by the model. In the table below, precision and recall at .05 and .5 thresholds are reported. If this model were to be used by a bank to say, decide whether or not to offer a person a loan, then it is costlier to falsely deem someone financially sound than to reject a more marginal loan applicant. As such, recall is the more important metric. Besides these two measures, the pipeline also calculates the area under the ROC curve (which is a general measure of the model’s performance), the accuracy of the model, and the time taken to train and test the model.

The following models were considered: logistic regression (LR) , k-nearest neighbors (KNN), random forest (RF), extra tress (ET), adaboost (AB), linear SVM (SVM), decision trees (DT), Naïve Bayes (NB), and stochastic gradient descent (SGD). The first table at the end of this document presents the classifier that performed the best on each given metric, with the specific parameter values suppressed for convenience. In order to differentiate between two different parameterizations of the same classifier, the id number of a given run is also included. The second table presents samples from each model type. Within each type, the parameterization with the highest AUC is presented.

In the first table, we see that logistic regressions tend to do well on precision, recall, and f1 measures. Interestingly, the specific parameterizations change pretty substantially depending on which row is considered. While some aspects are consistent- all have a low convergence tolerance and use the L2 norm- the regularization constant C is fairly variable. Also worth noting is that many of the regressions in the table below are rather lopsided, in that one metric is very good while the rest are mediocre. On balance, it seems that logistic regression can perform well if a very specific metric is being targeted, but the imbalances indicates that this model is perhaps too sensitive to changes in parameterizations to be useful in real applications. The model that performed best on the accuracy and AUC fronts is the random forest. Unlike the logistic regressions, this model seems more robust to changes in parameter values. Furthermore, even though these models were not the ones that maximized precision or recall, they both performed similarly well.

In the second table, we can get an idea of the relative performance of the various models. K nearest neighbors, stochastic gradient descent, and SVM all do rather poorly on all metrics. Specifically, their AUC indicates that these are all barely better than guessing randomly. All the tree-based models perform very well across all the precision, recall, and f1 metrics. Also high performing is adaboost. As discussed before, there is reason to care about recall over precision in this setting. Outside of LR, which has the sensitivity issues outlined previously, adaboost has the best recall metrics. Its only real downside is its long train time- if, however, this isn’t a big issue, adaboost looks to be a good approach. If training time is an important factor, then I would recommend random forest.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| num | classifier | best metric | AUC | Accuracy | Precision at .05 | Precision at .5 | Recall at .05 | Recall at .5 | f1 at 0.05 | f1 at 0.5 | test\_time (sec) | train\_time (sec) |
| 6 | RF | AUC | 0.862 | 0.936 | 0.175 | 0.638 | 0.835 | 0.113 | 0.29 | 0.192 | 0.112 | 1.224 |
| 10 | RF | Acc | 0.853 | 0.936 | 0.17 | 0.594 | 0.826 | 0.155 | 0.282 | 0.245 | 0.111 | 1.766 |
| 1 | LR | P .05 | 0.804 | 0.934 | 0.146 | 0.586 | 0.798 | 0.038 | 0.247 | 0.071 | 0.004 | 2.003 |
| 39 | LR | P .5 | 0.804 | 0.934 | 0.152 | 0.587 | 0.783 | 0.051 | 0.254 | 0.093 | 0.004 | 1.174 |
| 0 | LR | R .05 | 0.829 | 0.761 | 0.067 | 0.183 | 1 | 0.743 | 0.125 | 0.294 | 0.013 | 1.565 |
| 2 | LR | R .5 | 0.83 | 0.762 | 0.067 | 0.184 | 1 | 0.744 | 0.125 | 0.295 | 0.004 | 1.02 |
| 14 | LR | f1 .05 | 0.829 | 0.761 | 0.067 | 0.183 | 1 | 0.742 | 0.125 | 0.293 | 0.004 | 1.252 |
| 3 | LR | f1 0.5 | 0.801 | 0.934 | 0.143 | 0.584 | 0.806 | 0.043 | 0.243 | 0.081 | 0.004 | 1.225 |
| 2 | LR | test | 0.83 | 0.762 | 0.067 | 0.184 | 1 | 0.744 | 0.125 | 0.295 | 0.004 | 1.02 |
| 10 | LR | train | 0.573 | 0.933 | 0.067 | 0.167 | 1 | 0 | 0.125 | 0 | 0.004 | 0.061 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| num | clf | AUC | Accuracy | Precision at .05 | Precision at .5 | Recall at .05 | Recall at .5 | f1 at 0.05 | f1 at 0.5 | test\_time (sec) | train\_time (sec) |
| 2 | LR | 0.83 | 0.762 | 0.067 | 0.184 | 1 | 0.744 | 0.125 | 0.295 | 0.004 | 1.02 |
| 57 | KNN | 0.659 | 0.933 | 0.094 | 0.371 | 0.721 | 0.005 | 0.166 | 0.011 | 5.968 | 0.983 |
| 126 | RF | 0.863 | 0.936 | 0.168 | 0.637 | 0.851 | 0.111 | 0.281 | 0.189 | 0.226 | 2.768 |
| 142 | ET | 0.857 | 0.937 | 0.17 | 0.645 | 0.839 | 0.112 | 0.283 | 0.191 | 0.224 | 2.127 |
| 213 | AB | 0.86 | 0.936 | 0.067 | 0.55 | 1 | 0.205 | 0.125 | 0.299 | 0.697 | 9.605 |
| 226 | SVM | 0.534 | 0.929 | 0.068 | 0 | 0.972 | 0 | 0.127 | 0 | 10.2 | 267 |
| 230 | NB | 0.759 | 0.933 | 0.258 | 0.443 | 0.282 | 0.033 | 0.269 | 0.061 | 0.027 | 0.081 |
| 249 | DT | 0.829 | 0.934 | 0.176 | 0.518 | 0.788 | 0.147 | 0.287 | 0.228 | 0.007 | 0.173 |
| 267 | SGD | 0.549 | 0.924 | 0.408 | 0.408 | 0.117 | 0.117 | 0.14 | 0.14 | 0.003 | 0.155 |