CIS 432 Final Project

Xue Wang, Ruiyuan Sun, Collin Tracy, Daniyal Kamal, Zixuan Yan

Mar 3, 2021

Background:

An increasing number of homeowners are applying for the Home Equity Line of Credit (HELOC), which generally sets a credit line in the range of $5,000 - $150,000. Banks need to decide if they should give out offers based on applicants' credit report, which indicates their ability to repay within 2 years. Given a wide variety of information and background based on the applicant, Banks will need a reasonable ability to predict if a person has the financial wherewithal (as well as other related factors) to adhere to the specifications of a particular loan. In this project, we devoted ourselves to leveraging all kinds of predictive models, taking advantage of cutting-edge machine learning techniques, and using a dataset to find the most suitable model that can help banks hedge against potential risky applicants, and allow them the flexibility to input their own measures to determine the feasibility of a new applicant on the fly.

Data Overview:

There are 23 different features of applicants that were recorded in this dataset as predictors of their 'Risk performance', which is the target we were aiming to predict. In addition, the dataset contains information on 10,459 applicants to help us understand what components truly affect a person'sRisk performance,' as well as protecting us from variance with a large randomized sample. An applicant's risk performance would be graded as 'Bad', if he/she passed due more than or equal to 90 days at least once over 24 months from when the credit account was opened, or 'Good', if he/she made payments within 90 days. These gradings of good or bad could help Banks understand the potential risk of taking on another client with HELOC, and further help hedge against risky investments from the bank's perspective. The data of 21 out of 23 predictors are continuous, which means the values do not have any abrupt changes in value,

while the values in the feature “MaxDelq2PublicRecLast12M” and “MaxDelqEver” are categorical, meaning there is a classification component to it.

After acquainting ourselves with the data, we used a histogram to plot out the distributions of each feature, and in doing so, noticed

3 special values (-7, -8, -9) may appear in those predictors, indicating “value missing” caused by different reasons. The value of -9 indicated there was no bureau record or investigation, which only appeared in the feature “ExternalEstimate”. We assumed this feature was not essential so we decided to remove all the records that missed this feature. The value of -8 indicated there were no usable/valid trades or inquiries and the value of -7 indicated the condition was not met (no inquiries or no delinquencies). We chose to replace these values with the average value of other values in this feature and added dummy variables to indicate the values -7 and -8 to preserve their information while also attributing more meaningful numbers to our set, rather than an arbitrary negative number.

Limitations:

Despite our best efforts, there are some potential pitfalls that must be known before applying this model in practice to help banks make decisions. The dataset itself is relatively small, meaning that the potential for it to be used in areas with substantially different inflation could be limited. Another thing that should be considered is the accuracy of our predictions. Our best predictive model was accurate X% of the time, which could pose a high risk depending on the amount of money on the line for that particular transaction. These limitations should be thought about carefully before implementing them for your bank or company use.

Models & Result:

We randomly separated the original dataset into the train set (80%) and the test set (20%) using a seed set to 0 to make sure that our results are replicable and consistent from model to model. All the models were validated by the method of cross-validation in order to further protect our models against variance.
The results are as follows:

SVC:

Since SVC is relatively sensitive to different scales of different features, we chose to standardize our data before training SVC. We also tried to tune the hyperparameter C by adjusting its value from 0.01 to 1 and compared the average accuracy under each setting. However, the accuracy values are extremely close.
The final accuracy score we got from the cross-validation is 0.696 for the train set and 0.584 for the test set.

KNN: In order to determine the most ideal hyperparameters for our given problem, we first began by training the model with no parameter tuning. After training it the first time we computed an accuracy score of .679. Following this, we did preliminary cross-validation to determine how the model predicted, obtaining an accuracy score of .677, appropriate values compared to the prior one. Next, we wanted to optimize the model for the problem we are facing. We decided to run a grid search with the included hyperparameters being: n_neighbors, weights, algorithm, leaf size, and p. The grid search identified the most optimal model and finally, we retrained the optimal knn model and received the following train, validation, and test accuracies respectively: .723, .721, and .705. Initially, the grid search output parameters had accuracies of 1.00, .721, and,719. Although the parameters used for this model predicted the test data better overall, the overfitting of the training data was undesirable.

Decision Tree: A decision tree used supervised machine learning algorithms that can perform both regression and classification tasks. For our dataset, the prediction is the categorical variable, for each

feature in the dataset, the decision tree algorithm forms a node, where the most important attribute is placed at the root node. For evaluation, we start at the root node and work our way down the tree by following the corresponding node that meets our condition. This process continues until a leaf node is reached, which contains the prediction or the outcome of the decision tree. In order to find the best tree model with the ideal hyper-parameters, we use a grid search to find the optimal one that predicts the training data set to a degree of 72% accuracy. And it had relatively the same good performance when we used our model on the validation data set. Finally, the hold-out test data only had an accuracy of 70.7%.

Random Forest:

We trained a model using the random forest method, which is a method of bootstrapping that randomly selects features to create decision stumps. Our efforts, after using a grid search to determine the optimal hyperparameters, left us with an optimal model that predicted the training data to a relatively high degree of 75% accuracy; however, we deemed that this model was actually overfitting, a common habit of random forest models. Using the ideal model generated from the training set, the validation set only had an accuracy of 73% and finally, the hold-out test data only had an accuracy of 71%. Even after generalizing the hyperparameters, we were not able to improve the accuracy of the validation and test sets and only improved the accuracy of the training set.

AdaBoost:

The AdaBoost model that we trained uses a unique technique in that it creates a model, and depending on a certain rate specified by the user, creates another model that tries to improve the predictions of the model prior. This helps us protect against the overfitting problem, but our results returned worse than some of the comparable models run, with a 74% training accuracy, 75% validation accuracy, and only 71% hold-out test accuracy.

Logistic Regression:

We scaled the features to make them standardized at the start that we believe will be beneficial to our model performance. We train our first logistic regression model with the training accuracy 0.733 and validation accuracy 0.748. Then, we check coefficients to see whether there are some features that are not contributing to the model. We drop the feature with the lowest coefficient and re-train the model. The model is not improved comparing the cross validation accuracy score. Next step, we considered the regularized model with CV and penalty. We trained 2 models (cv=5 penalty=11&cv=5 penalty=12) and we got the improved model with validation accuracy 0.751. We tested the improved model and we got test accuracy of 0.742 which is close to our training and validation accuracy. At this point, we believe the improved model works, but we still want to see if there is a better probability threshold which can improve the performance. Finally, we got our best model using threshold 0.45, which means we prefer to predict bad risk performance more than good, and the test accuracy is 0.744. That means 74.4% of times the test result is accurate, regardless of bad performance or good performance. Also, our model has a relevantly higher recall number considering the accuracy, the high recall number means your bank can better avoid to loan to the clients with bad risk performance.

Discussions:

The model that we ultimately ended up using was a logistic regressions model. We felt that, given the accuracy statistics, and that we were considering altering the threshold in order to protect the business against risky lenders (artificially increasing false positives as opposed to false negatives), the logistic regression model provided the best bang for our buck.