Kaggle Competition - Give Me Some Credit

Kamden Baer, Charlotte Imbert, Quinn Link, Connor Squellati

November 27, 2023

Project Goal

We created a credit scoring algorithm that predicts the probability a borrower will default on their bank loan. We used historical data consisting of 250,000 borrowers and up to eleven explanatory variables per borrower. This is a binary classification problem: borrowers must be classified as someone who will repay their loan or will default. The performance of our algorithm was evaluated using AUC.

Exploratory Data Analysis

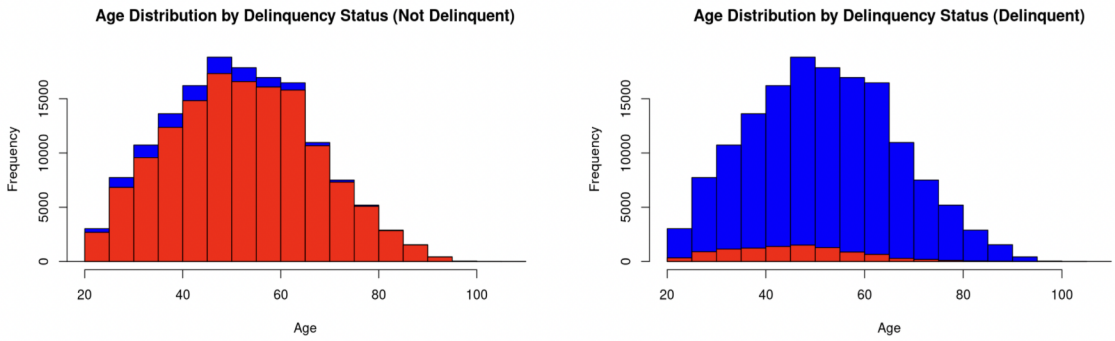

Figure 1: Initial EDA histogram plot overlay with delinquency respective of age. Blue pigmentation is relative to prior graphs with frequency of age for sample in population, red pigmentation is not delinquent. Median age for delinquents is 45.00 relative to age distribution.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
21.00	36.00	45.00	45.93	54.00	101.00

Table 1: Summary statistics for the dataset.

We attempted to visualize our data using PCA. PCA allows us to reduce the dimensionality of our feature space while retaining most of the variance in the data. However, we did not remove

outliers from the data, compromising the efficacy of PCA (see Outlier Detection below). Outliers are different from the rest of the data, significantly influencing total variance. Since PCA seeks to maximize variance when determining the principal components, PCA is sensitive to variance in data. Therefore outliers in our data may have skewed the variances of our components, thus misrepresenting the actual structure of the data. As a result, we were unable to sufficiently reduce the dimensionality of our features for visualization.

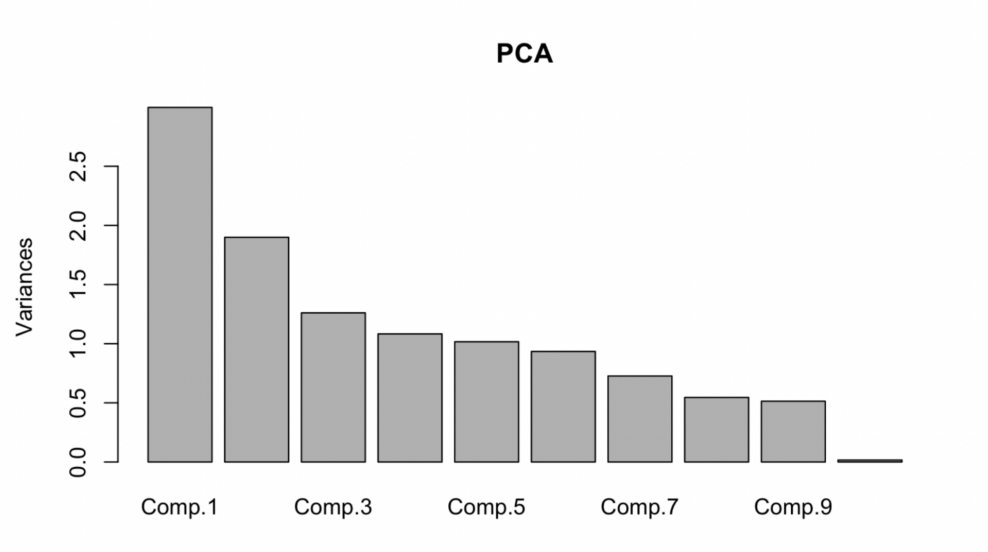

Figure 2: Top 10 principal components and their percentage of variance explained, obtained via PCA.

	Comp.1	**Comp.2**
RevolvingUtilizationOfUnsecuredLines	0.003692821	0.13993740
Age	-0.058446962	0.02050613
NumberOfTime30.59DaysPastDueNotWorse	0.572928185	0.03764458
DebtRatio	-0.007768919	0.19347098
MonthlyIncome	-0.009712328	0.28382320
NumberOfOpenCreditLinesAndLoans	-0.069030950	0.46887634
NumberOfTimes90DaysLate	0.574400172	0.01647108
NumberRealEstateLoansOrLines	-0.044324755	0.52127595
NumberOfTime60.89DaysPastDueNotWorse	0.574515370	0.01656865
NumberOfDependents	-0.003409505	0.25740515
OutlierScores	0.037734278	0.54973744

Table 2: PCA Loadings for Comp.1 and Comp.2

Imputation

We downloaded our data from Kaggle.com. Kaggle provided two data files: a training dataset with 150,000 observations and a test dataset of 100,000 observations. Only two columns had missing data: monthly income and number of dependents. Initially we tried multivariate imputation by chained equations (mice). Mice works by performing an initial imputation using a simple technique such as mean or median. Variables with missing data are imputed sequentially, using regression models to predict missing values using other variables in the dataset including the previously imputed data. When we performed mice imputation we observed a substantial change in our sample mean. This indicates that our choice of imputation model was inappropriate for our data. Ultimately we decided to impute by using the median for each variable to fill in missing data. We did this under the assumption that the missing data is randomly distributed. For example, we do not assume that people with different incomes fail to provide complete information at different rates. Under this assumption we do not introduce bias into our sample by imputing the median. What is more, the "No Free Lunch Theorem" states that every model works best in an infinite number of situations. Because of this and the poor performance of our mice model, we imputed using the median as it is a simple yet robust method.

Outlier Detection

Upon first glance at the data, we could see there were some anomalous values for certain variables, for example a debt ratio of 5710, or a monthly income of $208333. These values may have simply been due to the fact that these borrowers had unusually high values for these variables, but they could also be attributed to a potential error or typo during the data collection stage. In order to detect all outlier values, we used the find_HDoutliers function in the stray package. This function is built for multidimensional data with large numbers of rows and columns, which is ideal given that our dataset has 150,000 observations of 11 variables. One of the inputs of this function is k, referring to the number of nearest neighbors. A k value of 1 was chosen as higher values were very computationally intensive and took a significant amount of time to run. Those values determined to be outliers by the find_HDoutliers function were assigned an outlier score by the function (out_scores) in order to classify how anomalous they were. A higher outlier score indicates a more anomalous observation. A histogram of these outlier scores was plotted. The vast majority of observations were not considered outliers, however a small proportion had outlier scores above 0.05. The maximum outlier score was 0.538, detected using the which.max function. We did not remove any of the outlying observations from the dataset. Instead, we added a feature to our dataset in the form of the outlier scores (OutlierScores) for each observation, in order to determine if it had any predictive power. Incorporating an outlier score as a feature can help the model identify and correctly classify these rare instances. If this additional feature does not significantly contribute to distinguishing between classes, the SVM model will assign it a lower weight, having a minimal impact on the decision boundary.

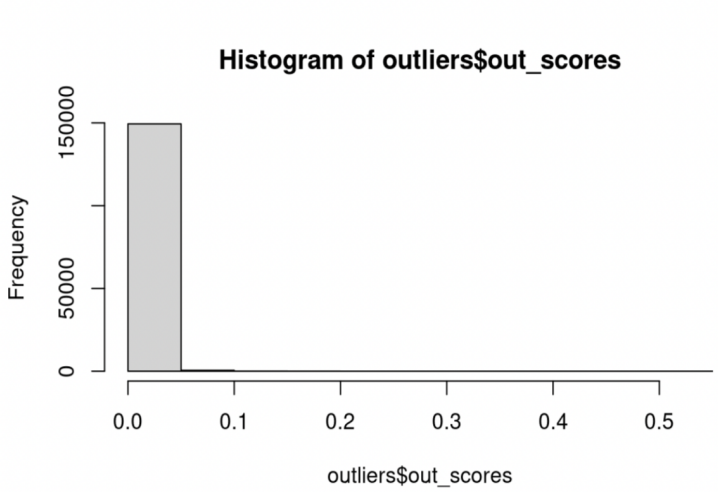

Figure 3: Histogram showing the distribution of the outlier scores resulting from the find_HDoutliers function. A high outlier score indicates that the observation is more anomalous. We can see that the histogram is very right skewed, with the vast majority of the observations having outlier scores below 0.05. It also shows that there were some observations with outlier scores above 0.05, and as high as 0.538.

Feature Importance

We initially fitted a random forest classifier to our imputed dataset in order to understand our data in more depth. We used the default values for the parameters in R, which was 500 for the number of trees and 3 variables tried at each split. We then looked at the importance of each feature in this random forest fit, which returned the mean decrease in Gini impurity attributed to each variable in the dataset. This showed us that our OutlierScores feature, added during basis expansion, was the most important feature (MeanDecreaseGini = 5289.7007), followed by the RevolvingUtilizationOfUnsecuredLines variable (MeanDecreaseGini = 2270.1243). The least important variables in terms of reducing the Gini Impurity were NumberOfDependents, NumberOfTime60.89DaysPastDueNotWorse, and NumberRealEstateLoansOrLines. The Gini purity for a feature indicates the importance of that feature for predicting the target variable. High Gini purity suggests that a feature significantly contributes to the model's ability to separate classes.

```
Call:
 randomForest(formula = SeriousDlqin2yrs ~ ., data = impoutcredit)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 4.61%
Confusion matrix:
       0    1 class.error
0 138212 1762  0.01258805
1   5159 4867  0.51456214
                                       MeanDecreaseGini
RevolvingUtilizationOfUnsecuredLines           2270.1243
age                                            1604.9773
NumberOfTime30.59DaysPastDueNotWorse            792.4015
DebtRatio                                      1909.4183
MonthlyIncome                                  1644.9255
NumberOfOpenCreditLinesAndLoans                1351.7764
NumberOfTimes90DaysLate                        1197.4075
NumberRealEstateLoansOrLines                    621.1504
NumberOfTime60.89DaysPastDueNotWorse            691.4496
NumberOfDependents                              782.5901
OutlierScores                                  5289.7007
```

Figure 4: Feature importances (Gini impurity) resulting from the random forest classifier model, revealing that OutlierScores is the most important feature for predicting whether a customer will default on a loan.

Decision Tree

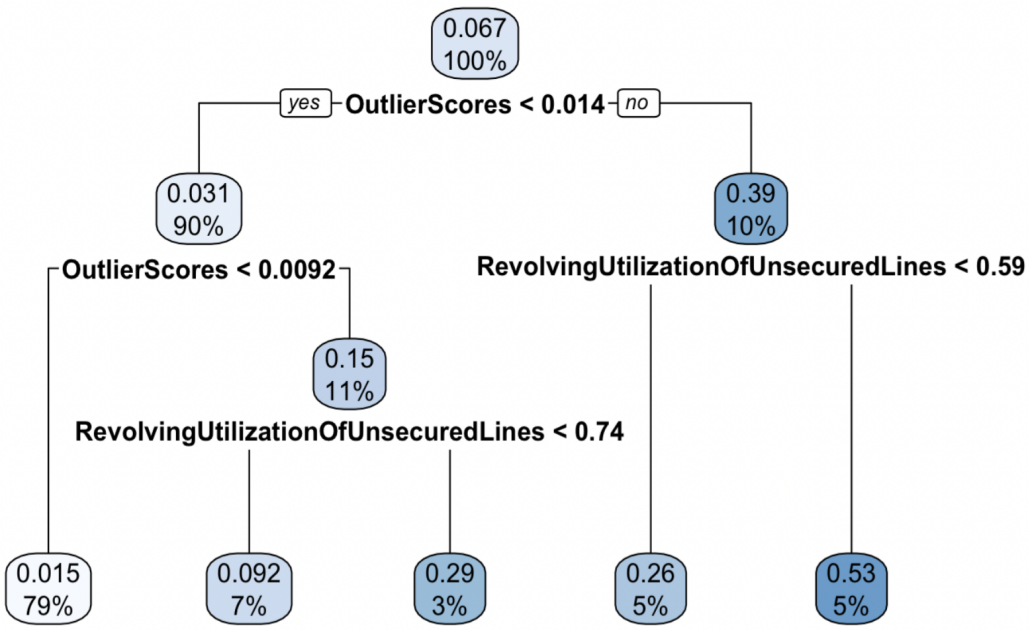

Figure 5: Decision tree resulting from using the rpart function on our imputed data. This agrees with the feature importance values obtained by fitting a random forest, as we can see that OutlierScores and RevolvingUtilizationOfUnsecuredLines were the most important variables for determining the splits of the tree. The decimal values show the fraction of the observations within each node with a response value of 1 (i.e. will be delinquent within 2 years).

Training, Tuning, and Implementing our SVM Learner

SVM relies on a subset of data (support vectors). SVMs are less likely to overfit compared to models that might weigh all features equally. We chose SVM because we saw through our exploration of Gini purity that certain features are more informative than others when trying to separate our data. SVMs are computationally expensive as training an SVM involves solving a quadratic optimization problem. The number of computations required to train an SVM grows quadratically as the number of data points increase. Considering the computational intensity of training one SVM model on a large data set, tuning the learner's parameters posed a challenge. Even using a grid search to tune two parameters (cost and gamma) is not feasible as the number of computations increases exponentially as we increase the range of values we search over. To limit the computational intensity of our tuning routine we used Bayesian optimization to tune our learner. Instead of searching over thousands of combinations of cost and gamma we made 30 SVM models. We separated out data from our training data to act as testing data. This allowed us to evaluate the performance of each SVM model by calculating AUC. We picked the combination of parameters that maximized AUC.

With the Bayesian optimized cost and gamma parameters we created a trained SVM model on our entire training data set. The SVM model created a decision boundary which can be used to classify

new observations as type 0 (will repay their loan) or type 1 (will not repay their loan). We also trained our SVM to calculate the probability of a prediction belonging to each class. This score reflects our confidence in the prediction. We used this trained model to predict the probabilities of those in the test data having serious delinquency in the past two years. These predictions were then submitted to Kaggle which returned an AUC of 0.67084.

Figure 6: Final submission of project with an AUC score of 0.67084