Loan Default Prediction

How Mixed-Models can be used to better Percentage of Default on Loans

Gayatri Sawant
Computer Science
Rice University
Houston Texas United States
grs7@rice.edu

Thera Fu
Computer Science
Rice University
Houston Texas United States
mf43@rice.edu

ABSTRACT

The use of machine learning in asset management is a part of data science that is slowly gaining traction in banks and start-ups. However, traditional methods can only predict the probability of a loan being defaulted on or not. Therefore, a more convenient method for determining the total loss involved in default is necessary. In this project, we propose a way of determining the percentage of loss if a loan is granted to an applicant. The data is taken from a 2013 competition by Imperial College of London[1]. The features are anonymized to a degree that the problem may only be solved using pure machine learning techniques. In this project, we primarily carry out algorithm comparisons to evaluate the reliability of our method. The experiment results show that mixed models which iteratively treat the problem as a classification and a regression problem have the least loss at 0.9 mean absolute error while keeping the recall and precision high. However, when compared to the leaderboard for the competition, our model was not competitive.

INTRODUCTION

This project is the result of a competition by the Imperial College of London regarding loan default prediction. The objective of this project is that given a set of features about a candidate, including previous loans, to determine if a loan request should be approved. This is determined by predicting the likely percentage loss to be incurred if the applicant does default on the loan.

This method of approaching this problem is a departure from traditional methods which distinguish between good or bad parties in a binary way. Changing this typically binary problem to a regression problem by seeking to anticipate and incorporate both the default and the severity of the losses, bridges the gap between traditional banking and asset management.

RELATED WORK

Since this project was inspired by a Kaggle competition, there are several submissions published on Kaggle. One public Kaggle notebook by Victor Pareira[2] suggested several regression variations of classification models, such as Random Forest and Gradient Boosting. However, the primary suggestion for the notebooks regarding this problem is that feature manipulation and dimensionality reduction plays a larger part in model results.

In addition to the competition notebooks, we also looked at some research papers. A paper in Procedia Computer Science[3] describes the use of a random forest algorithm to find defaults. It also describes how to deal with missing data in the context of a random forest. A paper by Leifei Zhou and Hong Wang also used the random forest for a large imbalance loan default dataset and demonstrated it performs better than KNN and SVM approaches[4]. Another paper describing such ensemble modeling was on ResearchGate[5].

PREPROCESSING

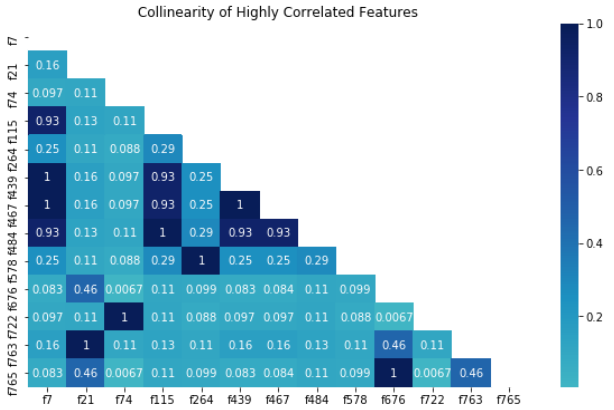
The dataset includes over 100,000 independent observations and around 770 feature columns, the features are largely anonymized and detrended, to the point that the features are labeled in order of their occurrence. As a result, strictly data science and machine learning methods were used to choose and tune models. The target variable, loss, is an integer percentage value from 0 to 100, indicating the amount by which the loan defaulted. A zero means the loan is paid in full.

Our first step included converting all of our features to a numerical format. We also had some columns with missing values. In this case, we needed to either drop the row or to replace it with another value. Since this is independent data we could not use forward fill techniques. Instead, the mean over each feature was used to replace these values.

Feature modeling and selection was also a large part of our data preprocessing, as we had very high dimensional data. For feature reduction, correlation matrix and PCA were the two primary methods that we used.

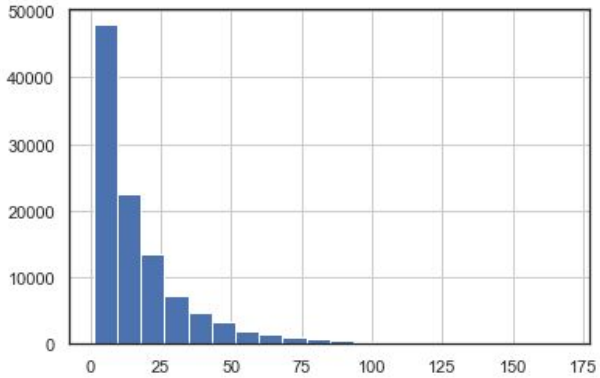
When we checked the Pearson correlation of the features to each other. It was found that over 450 of the features were almost perfectly correlated with a correlation coefficient of 0.99 or higher, a sample of which can be seen in *Graph 1*. As such, we retained a singular of each perfectly correlated feature, reducing our feature dimensionality significantly.

We also experimented with PCA to try to reduce the dimensionality further. With the features selected from correlation analysis, the first 100 principal elements yield an explained variance of about 0.89. In terms of performance in later models, we have found that taking the first 100 components often doesn't improve but doesn't hurt the performance of the models, except in rare cases. However, it does significantly improve the computing time since the size of the dataset becomes smaller.



Graph 1: Perfectly Correlated Features

Another aspect that was tackled was the different distribution of features in our dataset. As can be seen in *Graph 2*, a number of our features (about 50) are skewed beyond 10 standard deviations.



Graph 2: Exponentially Skewed Feature

To counter this, we took the log of the data to normalize the distribution. While doing so, we ensured that the log did not result in undefined values by skewing the distribution to the left by the minimum value - 1.

MODELS

The goal of our modeling stage was to predict the numerical percentage of loss as an integer of 0 to 100. We have experimented with directly using regression models, such as decision tree, random forest, logistic regression, gradient boost regression, and SVM. Since we have very imbalanced data, we also attempted to

leverage classification models to distinguish between defaulted and non-defaulted observations.

Classification models we have tried include KNN, random forest, decision tree, naive Bayers, isolation forest, support vector classifier, stochastic gradient descent, gradient boosting classifier. We also attempted to combine the classification and regression models using a two-step process, which will be explained in the mixed model section.

The tree-based models such as Decision Tree, Random Forest, and Gradient Descent were chosen in order to analyze the high-density of features to pick the "important" ones. Other models like Support Vector Machines were chosen in order to clearly distinguish between the classes in the data. Due to the issues with the skew in our target class, down-sampling was alternatively used to train these models better.

The metric we are using to evaluate our regression models is the mean absolute error (MAE). When the prediction is the mean of the training dataset, the MAE is about 1.49. When the prediction is set to all 0s, the MAE of the testing set is about 0.79, which also reflects the large imbalance toward zero. For classification, we will be using the precision, recall, and F1 score of the target class.

1 Classification

Due to the large skew in our target class, regression methods would be harder pressed to differentiate between the larger class (negative class) and the other possible values. As such, we thought to use a classification model to better estimate the larger class.

To use classification models, we transform the labels of our data into a binary format, with a 0 indicating that the loan did not default, and a 1 indicating a positive percentage of a defaulted loan. The ratio of negative and positive samples of our data is roughly 10: 1.

The classification models we experimented with include K-Neighbours Classifier, Decision Tree, Random Forest, Support Vector Classifier, Gaussian Naive Bayers, Isolation Forest, Stochastic Gradient Descent, and Gradient Boosting Classifier. In this section, we will briefly explore the models and their best results.

Decision tree and random forest models generally perform well on our dataset since our input data is high dimensional and imbalanced. Decision trees and random forest classifiers were not only able to produce valid results but were able to finish training within short training times, which becomes a serious bottleneck for some other models we tried. With the decision tree, we were able to get an F1 score of 0.15 and an accuracy of 0.81.

KNN is also a preferable model in terms of training times; however, it is somewhat limited by the data imbalance since the majority of points would have mostly zeros as their neighbors. Therefore, to avoid predicting all zeros, we had to limit the number of neighbors to 1 or 2. The best result we were able to get with the KNN classifier was an F1 score of 0.13 and an accuracy of 0.83.

For the Naive Bayers model, again, due to the class imbalance, most of the predictions are skewed to zero. But if we try to adjust the threshold carefully, we were able to get an F1-score of 0.2 and an accuracy of 0.79. It is also worth noting that the naive Bayers model was able to generate the highest precision of about 0.19 when tuned appropriately. Another interesting observation is that the predicted probabilities are often very close to each other and split into several groups, which might suggest that some features still resemble each other to a large degree.

Isolation forest is the only unsupervised model in our selection. It is selected because it fits the nature of our input and output data well. In practice, isolation forest is more robust to the class imbalance problem and

was able to be tuned to a larger degree. The resulted in an accuracy of 0.83 and an f1-score of 0.1.

Support Vector Classification was also selected as it works well with high dimensional data. As such, it is better able to separate data in higher dimensional spaces. Unlike PCA, it does not always look at linear separability, making it more effective in some cases. It had an accuracy of 0.9 and an f1-score of 0.01.

2 Regression

As the desired output format is an integer number, we also wanted to try regression models directly. The regression models we tried are decision tree, random forest, logistic regression, gradient boost regression, and SVM.

As stated before, the decision tree and random forest models generally work well with our data and are more resilient to the class imbalance problem. With decision trees, our best result is 1.11 with a maximum depth around 30 and using Gini impurity as the separator, and random forest generates similar results and exhibits similar behaviors when using all data as samples, instead, when doing bootstrap sampling, the model is prone to predict all zeros, presumably because zeros are more likely to be selected in the re-sampling process. For decision tree models, with a shallower tree, the predictions skew more towards zeros, but with a sufficiently deep tree, the predictions display a distribution similar to the desired output.

Another method we have tried to alleviate the class imbalance problem is using box cox transformation to transform the labeled data into a normal distribution and transform the predictions back to calculate metrics. This method has proven to be relatively helpful for simple regression models. For example, for decision tree regressors, it was able to provide an improvement of around 0.1 in MAE.

We also selected gradient boosting as a regressor because it provides many similar characteristics as decision trees, and it is able to overcome the overgeneralization of random forests, which, in our specific case, should hopefully give us more non-zero results. The best MAE we were able to get from the gradient boosting regressor is 1.24.

Other regression models we tried to include were logistic regression and support vector machine regressors. And their respective MAEs are 0.81 and 0.82. However, though the number is good, the distribution of predictions consists of basically all 0s, which is also a common behavior we were seeing across many other models when tuned differently, which is not particularly desirable.

3 Mixed Models

To leverage the power of classification models to reduce class imbalance while still producing our desired integer predictions, we tried to combine the classification and regression models in a two-step process.

First, we train both the classification and regression model on the training data. For the testing data, we first run the input data through the classifier; then, only for those models that were predicted as positive by the classifier do we pass it into the regression model to get an integer number as the output. For numbers that are predicted as 0 in the classifier, we use 0 as the output directly.

By using the classifier as a filter, we were able to get slightly more balanced results for the regression models. And since the classifiers are used here as a preliminary filter, we tuned them to preserve as many true positives as possible and the number of false positives doesn't matter as much.

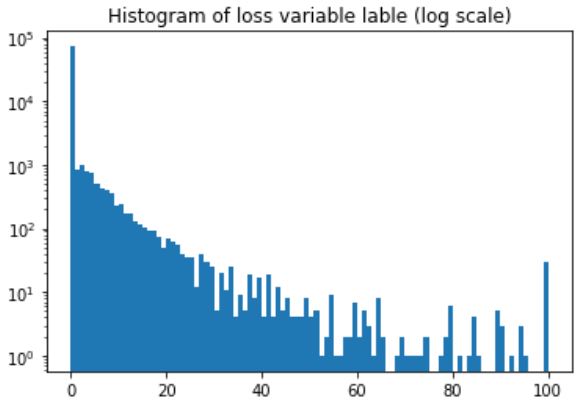
We tried a large number of combinations of classification models and regression models out of the ones we have discussed. For classifiers, naive Bayers works better in this setting because we could more easily tune the threshold to get our desired output distribution. For regression models, we have used the better performing one from previous experiments: decision tree, since in practice, it is the most robust against predicting all zeros. The combination of naive

Bayes and decision tree gives us an MAE of 0.88, which is the best we can get so far with result distributions that resemble the actual distribution instead of all zeros.

RESULTS

In the previous section, we have discussed most of the models we have experimented with, even the ones that didn't yield the most promising results. In this section, we will be including some visualizations of the results from our best-performing models: decision trees and mixed models (naive Bayes + decision tree) since the output distribution is an important factor for this project.

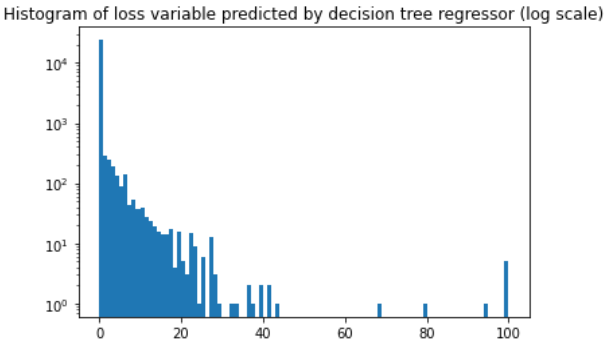
First, here in *Graph 3* of the labeled loss distribution. As it is on a log scale, we can see that zero values make up a substantial amount of the data, and the 100 percentage section also consists of a relatively large amount.



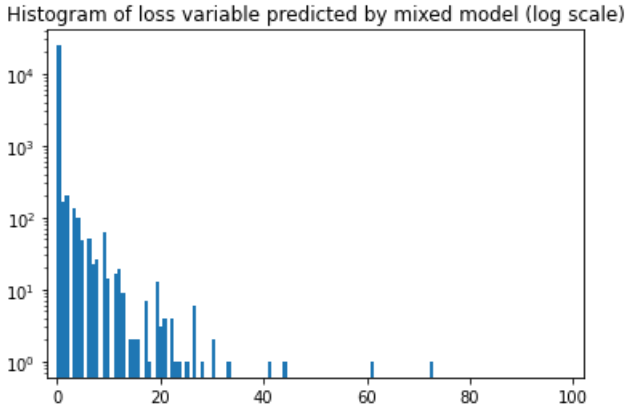
Graph 3: Log-scale distribution of Target Variable

Then, in *Graph 4* is the output distribution of the decision tree, which had an MAE of about 1.11. As we can see, the decision tree model was able to correctly capture the result distribution, including the excess at 0 and 100, and it was able to predict a fair amount of non-zero values. The values between 50 and 100 are less well represented, but the values between 0 and 50 generally fit the true distribution.

Finally, *Graph 5* shows the output distribution of our mixed model, which had an MAE of 0.88. The model still managed to predict many non-zero values and the distribution of the result is close to the true distribution in the lower percentage ranges. However, compared to the labeled data, the excess of loss value of 100 is lost in the predictions given by our mixed model.



Graph 4: Log-scale distribution of Decision Tree Regressor prediction



Graph 5: Log-scale distribution of Mixed Model (Naive Bayes + Decision Tree) prediction

DISCUSSION

As can be seen in the models described above, a clear distinction can be seen between the performances of the models. In terms of classification models, some issues propagate to give the results above.

1 Classification

In terms of Decision Trees and Random Forests, the models result in a lower classification error. This is due to the nature of these models which focus on the thresholds between each feature and how it can be used to separate the data. The decision trees are better at classifying than the random forest as they do not generalize to multiple weighted features but instead fixate on the set features to decide the outcome.

Doing slightly better than Decision Trees is Gaussian Naive Bayes. This can be attributed to Naive Bayes' ability to ignore the curse of dimensionality and with the large availability of data, it is better able to train its probabilities. K-Neighbours Classification is, however, affected by these issues, thus paving the way for lower accuracy and f1-score.

2 Regression

The logistic regression function is by definition linear. Thus, it has trouble learning the non-linear decision boundaries in our data as evidenced by our failed PCA computation. This is likely the same reason why Support Vector Machines failed despite setting the kernel to non-linear. Another issue with Logistic Regression is that it is prone to high bias towards classes, explaining why it was put off by the large skew in our dataset towards zero.

3 Mixed

As described in the models' section above, the mixed model leverages the strength of each model to better manage the skew of the target class and achieve a better MAE score. This ability, however, comes with a significant problem. As can be seen in *Graph 5*, the ability of the regression to predict a skewed distribution is also lower with many gaps falling at an interval of about 2-3 bins. This is due to the lower distribution of data available to the regressor, thus allowing it to be merged into a certain bin. However, on the whole, this sequence of models performs better than any singular model described earlier.

CONCLUSION

The biggest challenge of this project is the large class imbalance of the target variable and a large number of independent features. By experimenting with a large set of classification and regression models, we found that using a two-step mixed model with a classifier as a filter can improve the result of the regression

The next steps going forward would have been to leverage the strengths of each model in terms of the recall and precision to weight the probabilities produced by each. This would allow us to leverage the advantages of multiple models within the classification and regression of the mixed model.

REFERENCES

[1] Loan Default Prediction - Imperial College London, https://www.kaggle.com/c/loan-default-prediction/

[2] Pereira, V. H. (2019, January 17). Loan Default Prediction. Retrieved October 08, 2020, from https://www.kaggle.com/panamby/loan-default-prediction

[3] Zhu, Qiu, et al., "A study on predicting loan default based on the random forest algorithm"

[4] Zhou, L., & Wang, H. (2012). Loan Default Prediction on Large Imbalanced Data Using Random Forests. TELKOMNIKA Indonesian Journal of Electrical Engineering, 10(6). doi:10.11591/telkomnika.v10i6.1323

[5] Akanmu, Semiu & Gilal, Abdul. (2019). A Boosted Decision Tree Model for Predicting Loan Default in P2P Lending Communities. 2249-8958. 10.35940/ijeat.A9626.109119

CONTRIBUTION

The code was divided up equally including overlap between exploratory data analysis and model creation, tuning, and evaluation. This can be seen clearly in the way that the code is divided into different notebooks. The code is available on Github at the following link: https://github.com/GaiSawant/Loan-Default-Prediction---ICL. This document was also equally contributed to, the onus of describing our work falling to ourselves