Optimizing Investment Strategy in Peer to Peer Lending

Abiel Gutierrez
Stanford University
abielg@stanford.edu

Drew Mathieson
Stanford University
andrew4@stanford.edu

Abstract

We present several models that attempt to maximize the Sharpe ratio of portfolios selected from loans in Lending Club. Approaches to select which loans to fund include linear and logistic regressions, as well as neural networks. Whereas results in the binary classification task of identifying fully paid and defaulted loans wasn't a significant improvement over the baseline, our models made a good job of selecting fully paid loans whenever we limited the number of fundable loans to 1% of the total loans on the platform. Additionally, our efforts to predict high yield loans were successful. Our portfolio's standard deviations also greatly improved over the baseline, thus providing us with better Sharpe ratios.

1. Introduction

Lending Club provides borrowers and investors a platform where they can apply for and fund loans [1]. Investors choose which borrowers to fund, building portfolios of loans they deem likely to be profitable. Lending Club offers investors a wealth and variety of information pertaining to each loan, prompting the question: Which factors are most useful in predicting the profitability of a loan?

We sought to optimize portfolio loans under the constraint that we could fund 1% of the total loans available on the platform. We evaluated our portfolio using the Sharpe Ratio [2]:

$$S = \frac{R_p - R_f}{\sigma_p}$$

where:

$R_p = portfolio\ return$

$R_f = risk\ free\ rate = .0176$

$\sigma_p = portfolio\ standard\ deviation.$

The Sharpe ratio is a measure of the return earned in excess of the risk-free rate per unit of volatility, and is a popular metric for evaluating investments.

2. Related Work

There's been a considerable amount of work done in evaluating loans with machine learning tools – including several approaches that have used Lending Club data. Tsai et. al [3] worked with several models, including a modified logistic regression for binary loan-default classification with an added parameter β that increased the penalty for false positive predictions, thus increasing their precision metric. They then increased their accuracy by selecting influential words through the Term Frequency-Inverse Document Frequency methodology and adding them as features to their regression. An SVM approach was also implemented; their results improved loan returns by 1-2% when matching default risk with Lending Club's graded loan scheme.

Chang et. al [4] implemented logistic regression, Naïve Bayes, and SVM models to also predict defaulting loans. Their best result was the Gaussian Bayes model, yet they don't define a baseline nor do they compare training and dev set with test set results. Baselines are primordial given that many datasets are skewed towards containing much more non-defaulting loans – in our case we found that an 82% accuracy could be reached simply by predicting all loans as fully paid.

Pujun et. al [5] took a more alternative approach, looking into the loan application process to identify the likelihood that a loan application has of being accepted by the platform. They used a suite of classification, regression, and clustering tasks to derive insights. The most interesting ones include: the application loan approval criteria was relaxed towards 2014, probably in preparation for Lending Club's IPO; educational loans are most likely to be denied, whereas credit card consolidation loans are most likely to be approved; and loan grades are a near perfect predictor of assigned interest rates.

3. Dataset and Features

We downloaded all listed Lending Club loans from 2008 to 2016 [6]. This data contained 850,000 observations of 151 different features. 81 of these features were populated entirely with missing values, and 38 were irrelevant to our question, either because they were useless to our objective (user id, the url link to the loan's Lending Club page, etc.), or because it contained data about the loan's future status which one doesn't know about when allocating investments.

With our remaining 32 features, we eliminated any observations containing missing values (1.2% reduction of our data). Next, we processed our categorical data by allowing each category to remain only if loans with that category consisted of at least 2% of the total dataset. As an example, for the state feature (*Figure 1*), California and New York kept their individual categories, whereas Wyoming and Iowa were grouped together as category 'other.' This was done in an effort to avoid overfitting to small subsets of the data, as well as to reduce the overall dimensionality of our final design matrix, as each category in a feature makes up its own dummy column in the design matrix.

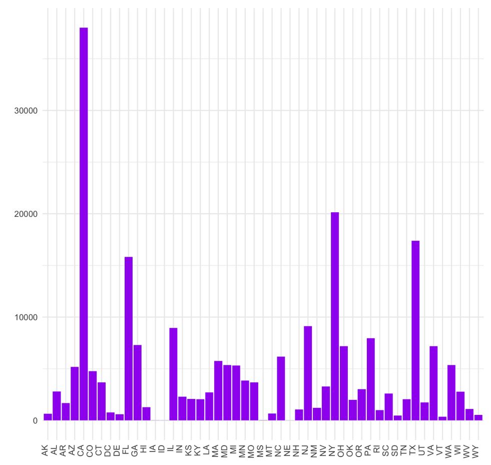

Figure 1: Number of loans issued by state

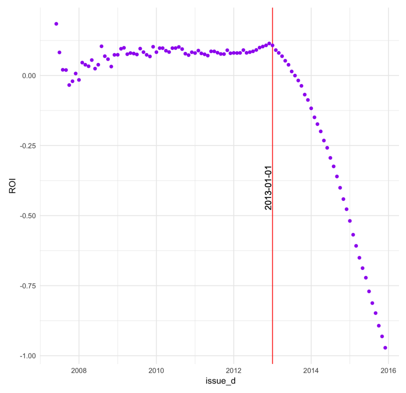

Figure 2: ROI by date. Red line indicates maturity.

Finally, we filtered out all loans which had not reached full maturity, as their return characteristics could not be determined (Figure 2). This led to a large reduction in the number of observations we were able to observe, but was necessary to perform an accurate analysis. Our final design matrix had 200,000 observations of 32 features, including our target features, loan status (default or non-default), and annual return of investment. We split our data into a training set (70%), a development set (15%), and a test set (15%).

4. Methods

The Sharpe ratio can be increased in two ways. First, we can increase the return of our portfolio. Second, we can decrease the standard deviation of these returns. These two observations drove our decision to pursue models that predicted the annual return on investment for each loan, as well as whether or not the loan would default.

4.1 Linear Regression

We used a linear regression model to predict the annual return on investment for each loan, in an effort to only select high return listings. We built three different models; unregularized linear regression, LR with the Lasso penalty, and LR with the Ridge penalty. These models were built using the cost function:

$$J(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j X_{ij} \right)^2$$

with penalization terms of $\sum_{j=1}^{p} |\beta_j|$ for the Lasso and $\sum_{j=1}^{p} \beta_j^2$ for Ridge.

4.2 Logistic Regression

We used logistic regression to classify loans as likely to default or unlikely to default. If we model the loans as Bernoulli random variables with probability p of defaulting, the variance of a given loan's default rate is p(1-p), which is minimized at extreme values of p. It follows naturally that the variability in a loan's return is tied to the variance of its default rate, and thus a portfolio full of high probability loans will suffer lower volatility. Our model used a sigmoid response in order to predict the probability that each loan would default. Again, we used three different models, unregularized logistic regression, and logistic regression with the lasso and ridge penalties. We used the logistic cost function:

$$J(\beta) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i\left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij}\right)}\right)$$

with the same regularization penalties as above.

4.3 Neural Network

We used a neural network to do binary classification between defaulted and fully paid loans. Our input had 86 sparse features (where the sparsity came from turning categorical features into boolean ones), a 40 unit hidden layer with PReLU activation [7], and a single output unit with sigmoid activation. The entire forward propagation can be seen here:

$$z^{[1]} = W^{[1]}x^{(i)} + b^{[1]}$$

$$a^{[1]} = p(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$y^{(i)} = g(z^{[2]})$$

where *b's* are the intercept term vectors, *p(x)* is the PReLU activation, and *g(x)* is the sigmoid activation function.

Weights had a uniform random initialization of small values – which provided better validation results than Xavier or He initializations. We used the Adam optimizer [8], which improved our loss curves thanks to its use of momentum, bias correction, and notion of previous gradient updates. We chose a learning rate of 0.001 after extensive hyperparemeter tuning, and found that a decay rate of 0.6 with a 5 epoch patience interval on the loss helped us decrease our loss metric, as seen in Figure 3. Due to this potential for a readjusting learning rate, we decided to train for 250 epochs, using mini-batches of size 32.

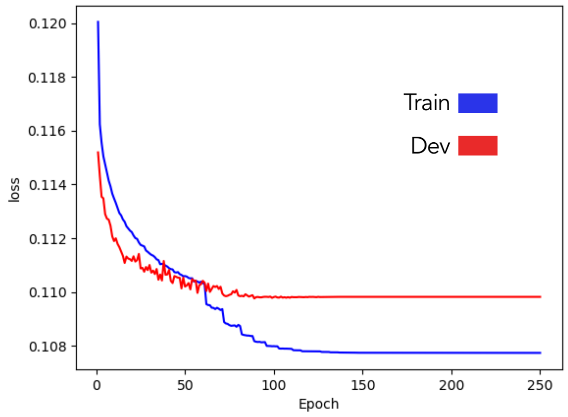

Figure 3: NN loss over epochs. Note the effect of the decaying learning rate in the blue "steps" between epochs 50-100.

PReLU improved our validation accuracy, since it maps negative input to small negative values, rather than zero – thus penalizing misclassifications more aggressively. For loss function we used mean-squared loss.

4.4 The Bootstrap

To find the standard deviation of our expected return, we performed the bootstrap on our training, dev, and test sets. The procedure was performed as follows:

1. For i in range(1000)
   a. Take a random sample, with replacement, from the data, of size nrow(data)
   b. Use the model to select optimal loans from the new data set
   c. Record your return from the selected portfolio as return(i)
2. Calculate the standard deviation of the 1000 observed returns

Using the bootstrap, we simulated how returns are affected by fluctuations and differences in the data, and therefore obtain an accurate reading of the variability of our expected return.

5. Experiments & Results

Using our training set, we built each model and tested them on the development set. The model with the best performance on the development set was selected as our best model, to be used on the test set for our final results. Each model was compared to an intuitive baseline. For the linear regression, the baseline model selects only loans with subgrade B3, as B3 had the largest expected return of any grade on the training set (Figure 4). For the logistic regression, the baseline selects only loans with subgrade A1, as they are the least likely to default (Figure 5).

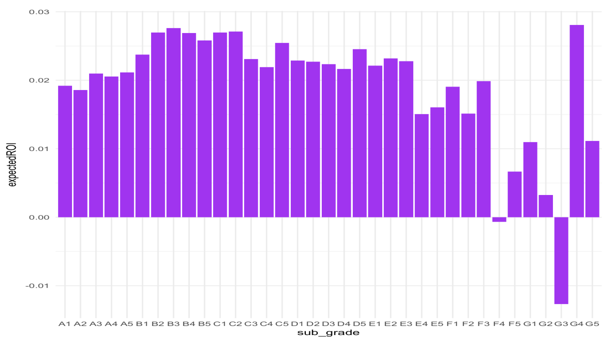

Figure 4: Expected Annual ROI per loan subgrade (A1-G5)

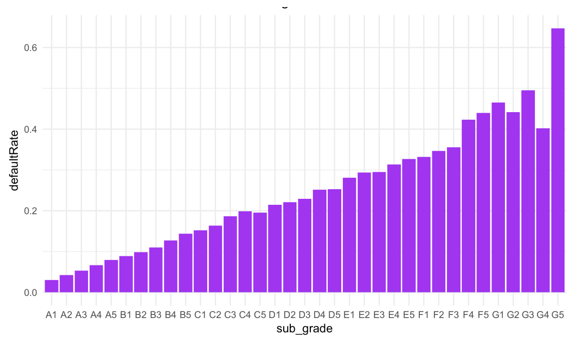

Figure 5: Default rate per loan subgrade (A1-G5)

5.1 Logistic Regression

	Train Return	Train SD	Train Sharpe	Dev Return	Dev SD	Dev Sharpe
Baseline	.0194	.00148	1.213	.0212	.00193	1.864
LR	.0265	.00168	5.303	.0271	.00377	2.524
LR Lasso	.0277	.00191	5.303	.0287	.00313	3.546
LR Ridge	.0271	.00149	6.385	.0277	.00297	3.388

Table 1: Logistic Regression Train & Dev Results

While our logistic model did have superior Sharpe ratios than our baseline model, this was due to averaging higher returns, rather than minimizing the variance of those returns. Given that the goal of the models was to reduce volatility, we were disappointed in these results, despite the higher Sharpe ratios. We suspect the underperformance is due to the fact that the calculations involved in assigning the subgrade for a loan are far more sophisticated than our own models, and likely already include many of the features used in our regression. Thus, it is unsurprising that the baseline model picking only A1 grade loans has lower volatility than our models.

5.2 Linear Regression

	Train Return	Train SD	Train Sharpe	Dev Return	Dev SD	Dev Sharpe
Baseline	.0306	.00248	5.246	.0284	.00505	2.145
LR	.0479	.00255	11.900	.0437	.00622	4.200
LR Lasso	.0405	.00231	9.900	.0402	.00379	5.962
LR Ridge	.0398	.00278	7.992	.0377	.00352	5.715

Table 2: Linear Regression Train & Dev Results

Conversely, we were encouraged by the results of our linear model. It consistently outperformed the baseline, as well as resulting in our highest Sharpe ratios of any model. We suspect the outperformance of the baseline was possibly due to a poorly selected baseline. Were we to redo the experiment, we likely would select a baseline that involved features more indicative of returns, such as the interest rate or the term of the loan. Regardless of the choice of baseline, the results were encouraging, offering us great returns with little increase in volatility.

5.3 Neural Network

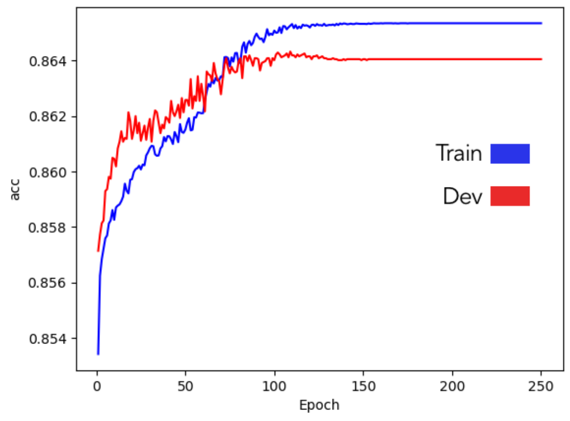

Figure 6: NN accuracy over epochs on train and dev sets.

	Training	Validation	Test
Baseline	85.20%	85.22%	85.17%
Neural Net	86.54%	86.41%	85.09%

Table 3: Neural Network loan default classification results on the entire dataset.

As seen on Table 3, the neural network did not perform better than the baseline (which just predicts that all loans are fully paid) on the test set, when evaluated on *all the loans*. However, if we analyze the model's result by the use case we proposed above, which is investing only in the top 1% most promising loans, we get a 95.5% accuracy on loan default prediction. This means that the model does a good job in classifying loans that are likely to be paid off; it doesn't do as great with loans that carry more ambiguity.

There was only a 0.13% accuracy gap between the training and validation set (Figure 6), which suggests that the test results might have been greater if we tried to fit the model a bit more aggressively to the data. This was also evident when we tried dropout regularization – it brought both the training and the dev accuracy down, so we decided to remove it from the model. Results could have improved if we had a larger data set and if our features weren't as sparse – the input feature vector had a size of 86, but those amounted to only about 20 features once you removed the sparsity that resulted from categorical variables. This meant that many Boolean instances of categorical features rarely occurred, and thus the model couldn't make much sense of them.

	Annualized Return	STD Dev	Sharpe Ratio
Baseline (A1 grade loans)	.0191	.00303	.513
Logistic Regression w/ Lasso	.0262	.00405	2.137
Baseline 2 (B3 grade loans)	.0157	.00646	(.285)
Linear Regression w/ Lasso	.0359	.00579	3.165
Neural Network	.0267	.00621	1.474

Table 4: Sharpe Ratio comparison of all models

Lastly, table 4 provides a comparison of all our models with regards to the Sharpe ratio metric that we set out to maximize. The linear regression model with lasso regularization provided the best metric, carried by the 3.59% annualized return that it produced. The logistic regression and neural network models also beat both baselines, meaning that the models in fact provide better portfolios than what any of the individual Lending Club subgrade schemes can offer to one.

6. Conclusion and Future Work

In this project, we pursued individual optimizing of the return on investment as well as minimizing the volatility of the loan. In order to maximize the Sharpe ratio, however, we must develop a model that jointly optimizes these conditions. Moving forward, we hope to explore creating a meaningful combination of our models to achieve this goal. By picking only loans that score well from both a return and a volatility perspective, we expect to achieve optimal Sharpe ratios. Another avenue for improvement can be to leverage the language found in the loan's descriptions, as other papers have done. This would provide a more qualitative rather than quantitative framework to evaluate the loans presented.

7. Contributions

Drew focused on cleaning the data and performing the regressions. Abiel focused on developing, tuning, and training the neural network model. Both of us worked on the poster and the final report.

8. References

[1] https://www.lendingclub.com/
[2] https://web.stanford.edu/~wfsharpe/art/sr/sr.htm
[3] Kevin Tsai, Sivagami Ramiah, and Sudhanshu Singh. Peer Lending Risk Predictor. CS 229 Autumn 2014. Stanford University.
[4] Shunpo Chang, Simon Dae-Oong Kim, Genki Kondo. Predicting Default Risk of Lending Club. CS 229 Autumn 2015. Stanford University.
[5] Bhatnagar Pujun, Chow Nick, Lai Max. Demystifying the workings of Lending Club. CS 229. Stanford University.
[6] https://www.lendingclub.com/info/download-data.action
[7] Trottier, L., Giguère, P., & Chaib-draa, B. (2016). Parametric exponential linear unit for deep convolutional neural networks. arXiv preprint arXiv:1605.09332.
[8] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[9] Ted O'Rourke. Lending Club – Predicting Loan Outcomes. June 19, 2016. https://rpubs.com/torourke97/190551
[10] François Chollet et. al. Keras (2015). https://github.com/keras-team/keras
[11] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
[12] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing In Science & Engineering, 9(3), 90-95.
[13] R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.
[14] Hadley Wickam. Tidyverse. R packages for data scientists. https://www.tidyverse.org/
[15] Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.