LendingClub Loan Default and Profitability Prediction

Peiqian Li
peiqian@stanford.edu

Gao Han
gh352@stanford.edu

Abstract— **Credit risk is something all peer-to-peer (P2P) lending investors (and bond investors in general) must carefully consider when making informed investment decisions; it is the risk of default as a result of borrowers failing to make required payments, leading to loss of principal and interest. In this project, we build machine-learned models trained on LendingClub (a leading P2P lending platform) historical loan data that help investors quantify credit risks using sci-kit learn [1]. Our classifier, predicting whether a given loan will be fully paid or not, achieves 0.89 in terms of both weighted precision and recall metrics; our regressor leads to a loan selection strategy that invests in 1.76% of available loans with 15% annualized return, when simulated on our independent test set.**

I. Motivation

With the rising popularity of peer-to-peer lending platforms in recent years, investors now have easy access to this alternative investment asset class by lending money to individual borrowers through platforms such as LendingClub, Prosper Marketplace, and Upstart, or to small businesses through Funding Circle.

The process starts with borrowers submitting loan applications to the platform, which performs credit reviews and either approves or denies each application. The platform also uses a proprietary model to determine the interest rate of approved loans based on the credit-worthiness of borrowers. Approved loans are then listed on the platform for investor funding. Investors usually want to diversify their portfolio by only investing a small amount, e.g. $25, in each loan. Hence, it is desirable for investors to be able to independently evaluate the credit risk of a large number of listed loans quickly, and invest in those with lower perceived risks.

This motivates us to build machine-learned classification and regression models that can quantify the credit risk with a LendingClub historical loan dataset. Specifically, we build and evaluate classifiers that predict whether a given loan will be fully paid by the borrower, as well as regressors that predict the annualized net return from investment in a given loan. Finally, we simulate and evaluate a simple loan selection strategy by investing in loans that pass a certain regressor prediction threshold.

II. Related Work

There have been many studies on classification models predicting LendingClub loan default. Chang et al. [2] built Logistic Regression, Naive Bayes, and SVM classifiers, all of which are able to achieve a G-mean score of around 0.86, the geometric mean of true positive and true negative rates. However, we find it questionable that loans with a **Current** status were treated as positive examples, along with **Fully Paid** loans. Since current loans may become default in the future, this practice invariably labels some true negatives as positive. In light of this, we decide to restrict our dataset to finalized loans only.

Tsai et al. [3] also experimented with the three models above along with Random Forest, but with an emphasis on precision at the expense of recall and negative predictive value (i.e. precision for the negative class). They find that Logistic Regression achieves a greater precision than the other models; they also break down the metrics by LendingClub's assigned loan grades (A-G) and subgrades (e.g. A-1). We believe that precision for both classes and their recalls are equally important metrics to optimize for, as a naive model which always predicts positive already achieves a good precision since the majority of examples are positive, but its negative predictive value would be zero.

In addition to classification models that predict loan default, Gutierrez and Mathieson [4] built regression models that predict the annualized return of a given loan. The loan selection strategy derived from a combination of these models was able to achieve better investment performance as measured by the Sharpe ratio than the baseline. This encourages us to build regression models and evaluate an investment strategy that select loans with high enough annualized return predictions.

Pujun et al. [5] built classification and regression models, but the goal was to predict LendingClub loan approval and their assigned interest rates. They applied k-means clustering and PCA techniques to detect latent trends in LendingClub approved loans. One of their most interesting findings is that loan approval standard had been gradually relaxed over the years. This reaffirms the desirability and usefulness of developing an independent and effective model for evaluating credit risks.

III. Dataset and Features

A. Dataset Overview

We worked with public dataset published by Lending Club [6]. Lending Club loans are in either 36-month or 60-month terms; we chose to work with Lending Club loans issued in 2012-2015 so that the loans have at least three years to mature. We filtered out loans whose statuses are not yet final, such as "Current" and "Late (less than 30 days)". We treat "Paid Off" as our positive label, and "Default" or "Charged Off" as negative. This leaves us with a dataset of size 745,529, with 19% negative and 81% positive examples. We split the data using a random $(0.7, 0.3)$ split into training and test sets.

B. Feature Preprocessing

Columns with empty values for most of the rows as well as columns with the same values across all rows are dropped in order to have a cleaner dataset. Free form text columns are also dropped because we posited that these fields would have more noise and are better tackled at a later stage when we have better understanding of the problem.

For features with missing values, they are categorized into three cases and treated differently: mean-set, zero-set and max-set. For mean-set fields, we took the average of the non-empty values. One such example is debt-to-income ratio (DTI): borrowers with lower DTI likely have lower risks compared to those with higher DTIs. For loan applicants missing DTI information, it is unreasonable to reward them by assigning zero DTI, hence taking average is a good starting point. In the case of max-set, missing values are replaced with a constant factor multiplied with the maximum value in that column. For instance, if the data for the number of months since last delinquency is missing, it would be unfair to punish the applicants by assigning zero for missing data. Finally, zeros are given for zero-set, which we believe would be a neutral replacement for the missing data.

Categorical features, such as obfuscated zipcode (e.g. "940xx"), are replaced with their one-hot representations. Features with date values are converted into the number of days since epoch. Normalization is then performed at the end on all features so they have zero mean and one standard deviation.

After the above preprocessing, we ended up with 1,097 features. We then ran PCA on the dataset with the hope to further reduce feature size. Unfortunately, the 95% variance threshold corresponds to around 900 features, which is close to 95% of the total number of features and therefore means that we cannot significantly reduce the feature size without sacrificing variances (see Figure 1 for correlation among select numerical features). Hence, we decided to keep all features.

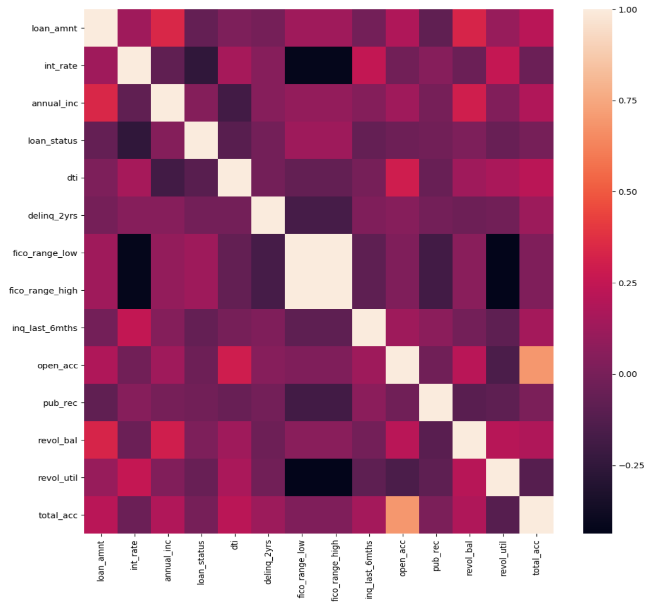

Fig. 1: Feature correlation heatmap

C. Label Definition

For classification model, both **Default** and **Charged Off** are assigned label 0 and **Fully Paid** is assigned label 1. For regression model, we use annualized return rate calculated from loan amount, total payment made by the borrower, and the time interval between loan initiation and the date of last payment.

IV. Classification Problem Overview

Our classification goal is to predict which class the loan belongs to: either **Default** or **Fully Paid**. In the following sections, we will share and discuss our experiments using Logistic Regression, Neutral Networks and Random Forest for classification problem. For metrics to evaluate classification performance, we use confusion matrix whose columns represent predicted values and rows represent true values. We also measure precision, recall, f1-score (the harmonic mean of precision and recall) and weighted average as defined below:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = \frac{2TP}{2TP + FP + FN}$$

Support = the number of true instances for each label

Weighted-avg metric = metric weighted by support

V. Classification Methods and Results

A. Logistic Regression

Logistic Regression takes in a list of features as input and outputs the Sigmoid of a linear combination of features weighted by learned parameters θ, i.e. $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$. To derive optimal parameters, the model iteratively updates weights by minimizing the negative log likelihood with L2 regularization

$$-\sum_{i=1}^{m} \left(y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right) + \lambda \|\theta\|_2^2$$

To tackle the class imbalance problem (only 19% of our dataset are negative examples), we used balanced weight for class labels, which is inversely proportional to class frequencies in the input data:

$$\frac{\text{n\_samples}}{\text{total\_n\_classes} * \text{label\_count}}$$

After running Logistic Regression with the above setting for a maximum of 1000 iterations, we arrived at the following results:

	Predicted Default	Predicted Paid Off
True Default	37,632	5,135
True Paid Off	22,448	158,444

class	precision	recall	f1-score	support
Default	0.63	0.88	0.73	42,767
Paid Off	0.97	0.88	0.92	180,892
Weighted Avg	0.90	0.88	0.88	223,659

As we can see, Logistic Regression is doing fairly well compared to naive models that blindly predict positive for all examples, or randomly guess positive and negative with 50% chance. Thanks to L2 regularization, we did not observe overfitting issues. One thing that we noticed and would like to improve upon is the precision and recall for negative class. Although we used balanced class weights to offset data imbalance, the prediction precision is only slightly better than randomly guessing. Therefore, we suspect there may be non-linear relationships in the dataset that is not learned by Logistic Regression, which leads to our exploration with Neural Network next.

B. Neural Network

We constructed a fully connected neural network with 4 hidden layers of shape $(10, 10, 5, 3)$ and Sigmoid activation for all neurons. We arrived at these hyper-parameter values by experimenting with various settings. Inputs are pushed through the model layer by layer. For neurons in each layer, the j-th output in layer i is computed as

$$a_j^{[i]} = g({W_j^{[i]}}^T x + b_j^{[i]}).$$

The final output of the network uses cross entropy (log loss) as loss function:

$$o = -(y \log(\hat{y}) + (1-y)\log(1-\hat{y})).$$

To arrive at optimal parameters, the model iteratively updates weights within each layer using Gradient Descent-based solver with a mini-batch size of 200, learning rate of 0.001 and L2 regularization penalty of 0.0001.

We obtained the following results:

Training set result

	Predicted Default	Predicted Paid Off
True Default	74,345	24,818
True Paid Off	24,210	398,497

class	precision	recall	f1-score	support
Default	0.75	0.75	0.75	99,163
Paid Off	0.94	0.94	0.94	422,707
Weighted Avg	0.91	0.91	0.91	521,870

Test set result

	Predicted Default	Predicted Paid Off
True Default	30,545	12,222
True Paid Off	11,864	169,028

class	precision	recall	f1-score	support
Default	0.72	0.71	0.72	42,767
Paid Off	0.93	0.93	0.93	180,892
Weighted Avg	0.89	0.89	0.89	223,659

The model has high variance and is suffering from overfitting. Compared with the logistic regression model, this neural network model achieves a better weighted precision at the expense of weighted recall and the difference between precision and recall is less polarized compared to that of the Logistic Regression.

C. Random Forest

Random Forest classifier is one of the tree ensemble methods that make decision splits using a random subset of features and combine the output of multiple weak classifiers to derive a strong classifier of lower variance at the cost of higher bias.

We started off our venture into Random Forest with 200 trees using Gini loss $1 - \sum_{j=0}^{1} p_j^2$.

Decision splits are based on at most 50 features to reduce variance. After training, we reached the following result:

Training set result

	Predicted Default	Predicted Paid Off
True Default	81,517	17,646
True Paid Off	863	421,844

class	precision	recall	f1-score	support
Default	0.99	0.82	0.90	99,163
Paid Off	0.96	1.00	0.98	422,707
Weighted Avg	0.97	0.96	0.96	521,870

Test set result

	Predicted Default	Predicted Paid Off
True Default	27,760	15,007
True Paid Off	8,750	172,142

class	precision	recall	f1-score	support
Default	0.76	0.65	0.70	42,767
Paid Off	0.92	0.95	0.94	180,892
Weighted Avg	0.89	0.89	0.89	223,659

Although the performance is on par with Neural Network and Logistic Regression, Random Forest's overfitting problem is much more prominent than any other models even after restricting the maximum number of features considered for decision splits to 50.

D. Classification Model Conclusion

Based on our explorations with Logistic Regression, Neural Network and Random Forest, we are able to achieve weighted average of 0.89 for both precision and recall. More specifically, our classification results appear to be better than the works done by the previous project [7] in terms of higher precision and recall, and more logically reasonable and practical than the work done by Chang et al. [2]. However, classification models can only predict the probability of loan defaults. This does not offer us a very fine-grained view in terms of how much return each loan can generate, which is essential for investors. Therefore, we would also like to predict the expected return rate, which naturally leads to our experiments with regression models next.

VI. Regression Problem Overview

We strive to predict the investment return if we were to invest in a given loan. Our goal is to build regression models that predict the **net annualized return** (NAR) of a given loan in a way similar to how LendingClub calculates NAR for investors [8]. For a given example x, our label y is the NAR defined as

$$y = (\frac{x_{TP}}{x_{LA}})^{\frac{1}{365/D}} - 1$$

where x_{LA} is the loan amount, x_{TP} is total payment made by the borrower, and D is the number of days between loan funding and date of last payment.

We evaluate regression models in terms of mean square error (MSE) and coefficient of determination R^2.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2}{\sum_{i=1}^{m}(y^{(i)} - \bar{y})^2}$$

where $\hat{y}^{(i)}$ is model prediction on $x^{(i)}$, and $\bar{y} = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}$ is the mean of the true labels. The coefficient of determination tells us how much variability of the true NARs can be explained by the model.

VII. Regression Methods and Results

A. Linear Regression

The goal of linear regression is to find a linear hyperplane that minimizes the ordinary least squares. Specifically, it finds parameters θ that minimizes

$$J(\theta) = \sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$$

Performance of linear regression:

Split	**MSE**	R^2
train	0.040	0.243
test	5.014	-9.494×10^{22}

The extremely skewed MSE and R^2 values on the test set clearly indicate a high-variance problem of the model which overfits the training examples. To rectify this, we employ L2 regularization in our next model.

B. Ridge Regression

Ridge regression adds an L2 regularization term to the cost function of linear regression

$$J(\theta) = \sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 + \alpha\|\theta\|_2^2$$

but otherwise works the same way as linear regression.

Performance of ridge regression with $\alpha = 1$:

Split	**MSE**	R^2
train	0.040	0.243
test	0.040	0.238

As expected, L2 regularization mitigated the problem of overfitting, giving similar metrics for both train and test sets. $R^2 = 0.24$ means that 24% of the NAR's variability can be explained by the ridge regression model. We next try non-linear models to further decrease MSE and increase R^2.

C. Neural Network

The fully-connected neural network regression model is very similar to the classifier described earlier in section V-B. The only difference is that all neurons use the ReLU activation function $f(x) = \max(0, x)$, and the neural network tries to minimize the squared loss on the training set.

We used the Adam stochastic gradient-based optimizer [9], a batch size of 200, L2 regularization penalty parameter of 0.0001, four hidden layers with $(20, 10, 5, 3)$ neurons, and obtained the following results:

Split	**MSE**	R^2
train	0.036	0.324
test	0.037	0.306

We see that the neural network regressor performs much better than ridge regression thanks to its ability to model non-linear relationships.

D. Random Forest

A decision tree regression model infers decision rules from example features by finding a feature split for each non-leaf node that maximizes the variance reduction as measured by MSE. The mean of leaf-node example labels is the output of the decision tree regressor.

Decision trees tend to overfit, especially when the tree is deep and leaf nodes comprise too few examples. Limiting the maximum depth or the minimum leaf node examples not only reduces overfitting, but also speeds up training significantly, as random forest model builds numerous decision trees before taking the average of their predictions.

Specifically, random forest regressor repeatedly builds decision trees on a bootstrap sample drawn from the training set, and considers a random subset of features as candidates when finding an optimal split.

Performance results of Random Forest regressor with various depth limits:

Max Depth	4	8	10
Train MSE	0.037	0.035	0.034
Test MSE	0.037	0.036	0.036
Train R^2	0.298	0.329	0.356
Test R^2	0.295	0.312	0.315

From these results, we see that as we allow the decision trees to grow deeper, bias increases while variance decreases. The performance of Random Forest regressor beats both ridge regression and neural network, likely due to the fact that decision trees are able to capture very nuanced and non-linear relationships.

VIII. Loan Selection Strategy

Our best Random Forest regressor achieves a root-MSE of $\sqrt{0.036} = 0.19$ on the test set, which implies that the predicted NAR is estimated to differ from the true NAR by 0.19. While this may appear very large at first glance, the model can actually be very useful in formulating a loan selection strategy. Loan defaults usually happen soon after loan funding, and the chance of default decreases as more payment is made. As a result, most true NARs of defaulted loans are well below -0.5, so the model can still very accurately tell us that investing in loans like these likely result in losses.

In light of this, we experimented with the strategy of investing in loans with model NAR predictions higher than a reasonable threshold $M > 0$. Intuitively, the threshold M can serve as a parameter investors can tune according to their investment account size: the bigger M is, the more stringent the loan selection is, so less amount of money can be invested, but hopefully the annualized return will be higher due to investing in loans more selectively.

In order to determine a reasonable range of values for M, we rank the training set examples by model predictions from high to low. Figure 2 shows a decreasing annualized return as we invest in more loans, which is consistent with our expectation that less stringent threshold results in a lower average annual return.

For a specific threshold $M = 0.132$, on both training and test set, the strategy yields an annualized return of 15% with 1.7% loans picked and invested.

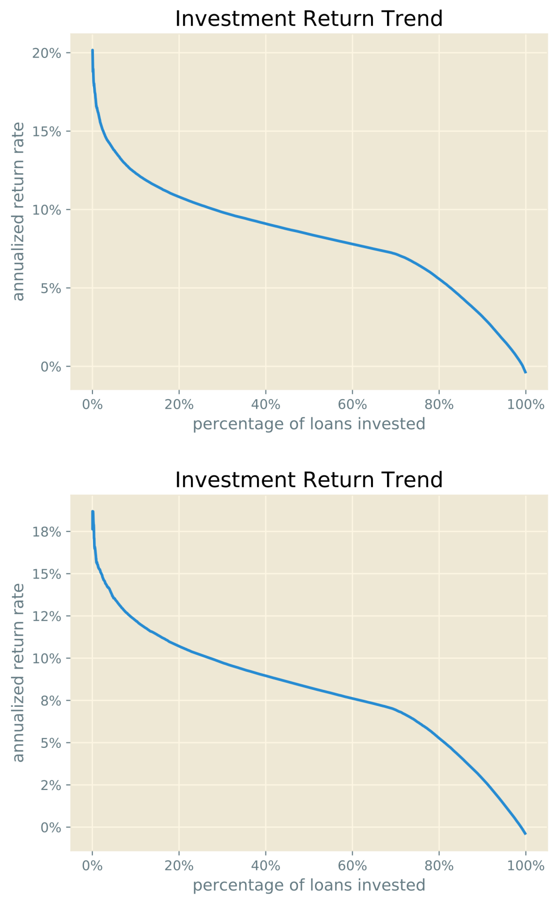

Fig. 2: Annualized return vs. percentage of loans invested on training (top) and test (bottom) sets.

IX. Conclusion

Comparing our models with those from related work, ours have better precision / recall and are more practical in terms of enabling implementable investment strategies. In the case of classification models, Random Forest achieved 0.89 weighted average precision and recall. But it is also important to note that the Random Forest and Neural Network models do have higher variance than desired and have space for improvement. For the regression counterpart, Random Forest is able to attain 0.315 coefficient of determination and to deliver predictions that lead to a **profitable** and **actionable** loan selection strategy in the sense that the return rate is higher than S&P 500's 10% annualized return for the past 90 years [10].

X. Future Work

We obtained a regression prediction threshold based on the training set, and simulated the strategy on the test set. Both sets comprise loans initiated within the same periods (2012-2015). We can check to see if the strategy generalizes to future loans by testing it on 2016-2018 loans that have finalized. Practically speaking, this would be a much more useful metric for investors.

We worked with a 70% training and 30% test split for simplicity in this project. The absence of a development set didn't afford us much opportunity to tune the hyper-parameters of our models, such as the number of decision trees to use in random forest models, and the number of hidden layers and neurons of each layer in neural network models. Having a small development set would enable us to tune some hyper-parameters quickly to help improve model performance metrics.

There are definitely factors that contribute to default not captured by features in our dataset. We can add external features such as macroeconomic metrics that have been historically correlated to bond default rate. For categorical features like employment title, we can join them with signals such as average income by industry, similar to what Chang et al. [2] did for zip code with average income of each neighborhood.

We can also make better use of existing features in the LendingClub dataset. One example is loan description which the borrower enters at the time of loan application. Instead of dropping such freeform features, we can try applying some statistical natural language processing techniques such as TF-IDF as Chang et al. [2] did.

Finally, we notice that LendingClub also publishes declined loan datasets [6]. We can add these declined loans as negative examples to our dataset, which helps further alleviate the class imbalance problem.

XI. Contributions

The two of us paired up on all components of this project, including dataset cleaning, feature engineering, model formulation / evaluation, and the write-up of this report and the poster.

Codebase: `https://goo.gl/Sxf1Rm`

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] S. Chang, S. D.-o. Kim, and G. Kondo, "Predicting default risk of lending club loans," 2015.

[3] K. Tsai, S. Ramiah, and S. Singh, "Peer lending risk predictor," *CS229 Autumn*, 2014.

[4] A. Gutierrez and D. Mathieson, "Optimizing investment strategy in peer to peer lending," 2017.

[5] B. Pujun, C. Nick, and L. Max, "Demystifying the workings of lending club,"

[6] "Lending club statistics — lendingclub." `https://www.lendingclub.com/info/download-data.action`. (Accessed on 12/08/2018).

[7] "Predict lendingclubs loan data." `https://rstudio-pubs-static.s3.amazonaws.com/203258_d20c1a34bc094151a0a1e4f4180c5f6f.html`. (Accessed on 12/08/2018).

[8] "How we measure net annualized return — lendingclub." `https://www.lendingclub.com/public/lendersPerformanceHelpPop.action`. (Accessed on 12/08/2018).

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[10] "What is the average annual return for the s&p 500?." `https://www.investopedia.com/ask/answers/042415/what-average-annual-return-sp-500.asp`. (Accessed on 12/08/2018).