Peer Lending Risk Predictor

Kevin Tsai
kevin0259@live.com

Sivagami Ramiah
sivagamiramiah@yahool.com

Sudhanshu Singh
ssingh.leo@gmail.com

Abstract

Warren Buffett famously stated two rules for investing: Rule #1. Never lose money, and Rule #2. Never forget Rule #1. Recent Peer Lending opportunities provide the individual investor to earn an interest rate significantly higher than that of a savings account. However, a default on the loan by the borrower means the investor will lose her entire principal. In this paper, we will use Machine Learning algorithms to classify and optimize peer lending risk.

Introduction

Individual investors who prefer fixed-income assets are faced with extremely low yield in the recent years. Bank accounts pay less than one percent, and Treasury Bonds pay low single digit percentages. At the same time, consumer debt interest rates have remained high, with unsecured consumer debt such as credit card rates at over twenty, and sometimes thirty percent. This has created a new space where peer lending companies match individual investors who are looking for a higher yield with borrowers who are looking for a lower interest rate.

LendingClub.com, Prosper.com, and Upstart.com are examples of such companies. The Loan process starts with the prospective borrower filling out an application online, stating reason for the loan, the loan amount, employment and income, and a battery of other information. There is usually a vetting process by these companies which also includes a risk grading, and then the loans are made available to investors. Once a loan has attracted enough investment dollars, it is funded. These lending companies make an upfront through a fixed percentage discount point(s).

A given portion of these borrowers will be late in payment and possibly even default on their principal. These lending companies state they will perform their due diligence to recover money from loans that are in arrears. However, because any loss is borne solely by the investor, it is imperative that the investor carefully select the investment opportunities so as to avoid default risk while maintaining a healthy return.

While this paper will use data publicly available from LendingClub.com, this analysis can apply equally to other fixed-income, fixed-term investment with feature data. LendingClub, like other peer lending companies, provide some form of risk grading, which usually rises as the loan interest rate rises. The goal of this paper is to reduce exposure to loan defaults and exceed LendingClub's return given the same risk level.

Data Description

The LendingClub data used in this paper spans years 2007 through 2013.

Table 1: Interest and Default Rate per Grade

Loan Grade	Default (y=0)	Paid (y=1)	Available Loans	Interest Rate	Default Rate
A	1,340	16,395	17,735	7.47%	7.56%
B	4,113	24,155	28,268	11.62%	14.55%
C	4,418	17,534	21,952	14.77%	20.13%
D	3,585	10,236	13,821	17.54%	25.94%
E	1,970	4,377	6,347	20.12%	31.04%
F	1,032	1,702	2,734	22.62%	37.75%
G	254	409	663	23.82%	38.31%
Total	16,712	74,808	91,520	13.47%	18.26%

The raw data contains multiple loan statuses, including fully paid, charged/defaulted, late, in grace, issued, and current. Because the goal of this paper is to predict and avoid loans that will default, and to invest in loans that will be fully paid, the classification will require loan statuses only fully paid (*y=1*) and charged/default (*y=0*). The LendingClub data also includes nearly 50 features. Feature selection is discussed in a following section.

As one would expect, higher interest rates correspond to higher default rates. If an investor were to invest blindly, she would get an average interest rate of 13.47%. However, she will also face a default rate of nearly one in five loans. If the investor were to follow LendingClub's loan grading and choose to be conservative, she may choose to invest only in A-grade loans, where she will earn about 7.5% interest at a default risk of also 7.5%. If she were willing to take a higher risk, she can choose G-grade loans at an average interest rate of about 24% but with a default risk of greater than one-in-three.

Method

Because our primary goal is to not lose money, our optimization will focus on severely discriminating against loans with potential for default, meaning we will strongly favor a loan classified as good must be good as the primary metric. We can afford to incorrectly eliminate good loans as bad, as at any given time, there are many more loans available to invest in than dollars to be invested. Therefore, in this paper, we will focus on precision at the cost of recall and overall accuracy.

After identifying this pool of loans with low probability of default, we will compare our return rate to LendingClub's return rate given the same default risk rate.

Data Preprocessing

LendingClub data required significant cleansing before ingestion:

- RegExp to clean HTML tags and other unwanted characters.
- Discrete/categorical features expanded into separate binary columns, including text preparation for TF-IDF (see section on TF-IDF).
- Stanford-NLP (Manning, 2014), guava, Lucene for text processing.
- For serializing and de-serializing CSV files, we used JSefa API.

Data Balancing

The raw data from LendingClub has a default-to-paid off rate of 18.26% vs. 81.74%. This skew will negatively impact algorithms such as Logistic Regression that optimizes across the entire training set. For training, therefore, we balanced the training data file to have a 50/50 split.

Standard Data Files

We created three data files to be used across all of our machine learning algorithms. All data files are randomized.

1. Balanced Training.
2. Balanced Test – this file was used to plot the learning curve.
3. Prior Test – this file keeps the same distribution as the source data and is used to calculate precision.

Feature Selection

Apart from our own intuition and insight we got from the data by running a few algorithms, we also ran exhaustive feature selection search in Weka and MatLab to find most dominant features.

1. InfoGain: InfoGain(Class,Attribute) = H(Class) - H(Class | Attribute) Evaluates the worth of an attribute by measuring the

information gain with respect to the class by comparing worth with and without the attribute. Top 4 features: *emp_title, interest_rate, loan_amount, annual_income*.

2. InfoGain for Logistic Regression. Top features: *loan_amount, term, interest_rate, installment, employment_length, annual_income, debt_to_income, revolving_utilization.*
3. Correlation-based Feature Subset Selection (Hall, 1998) and Genetic Algorithm Search (Goldberg, 1989): Evaluates the value of a subset of attributes by considering the individual predictive ability of each feature and minimize redundancy. Subsets of features that are highly correlated with the class while having low intercorrelation are preferred. Top 4 features: *debt_to_income, emp_title, int_rate, term.*
4. Matlab's sequentialfs forward search feature selection algorithm identified the following features: *int_rate, is_income_verified, annual_income, and loan_purpose.*

Modified Logistic Regression

Motivation

Standard Logistic Regression attempts to maximize the log likelihood of the estimates (Ng). Since log($h(x^{(i)})$) and log(1-$h(x^{(i)})$) are always zero or negative, misclassification results in a large negative number. As our goal is to minimize default risk by focusing on increasing precision, we modify the log likelihood estimate by multiplying the $y^{(i)}=0$ term by a penalty factor beta (β).

$$l(\theta)=\sum_{i=0}^{m} y^{(i)}\log\left(h(x^{(i)})\right)+\beta\left(1-y^{(i)}\right)\log\left(1-h(x^{(i)})\right) \quad (1)$$

If the classifier incorrectly classifies a defaulted loan (*y(i)=0*) as a good loan (*h(x(i))~1*), the log likelihood of this same will be multiplied by the factor β. A high β will cause the classifier to avoid incorrectly classifying defaulted loans (*y(i)=0*) as paid off loans (*y(i)=1*), even if that means increasing the misclassification of a paid off loans as defaulted loans. This effectively gives preference for precision at the cost of recall and overall accuracy. Introducing β into the log likelihood, it follows that the first and second derivatives are:

$$\frac{\partial}{\partial\theta_j} l(\theta)=(y-\ (y+\beta+\beta y)h(x))\,x_j \quad (2)$$

$$\frac{\partial^2 l(\theta)}{\partial\theta_j\partial\theta_k}=-(y+\beta+\beta y)h(x)(1-h(x))\,x_j x_k^T \quad (3)$$

Note that when $\beta=1$, all three equations above revert to the original Logistic Regression equations. With the first and second derivatives, we used Newton-Raphson as the optimization method:

$$\theta := \theta - H^{-1}\,\nabla_\theta l(\theta) \quad (4)$$

Effect of Penalty Factor Beta

As we have many more data points than dimensions, m>>n, we decided not to use regularization. As can be seen in the Figure 1, the size of the data set puts us in the high bias range.

Figure 1: Learning Curve: Modified Logistic Regression

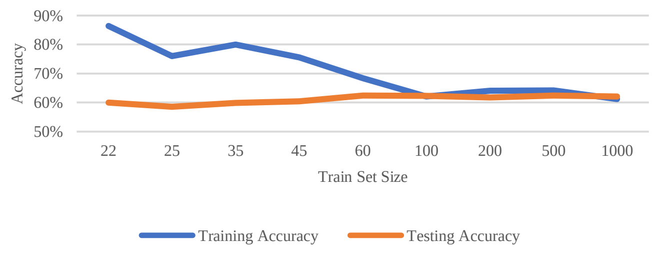
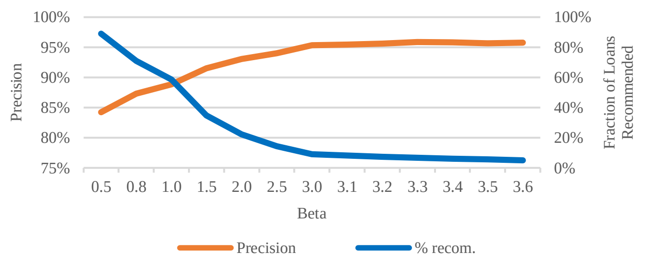


Figure 2 shows the inverse relationship between Fraction of Loans Recommended and precision. This is expected, as β increases, the classifier is more discriminate against defaulted loans.

Figure 2: Effect of Penalty Factor β



As seen in Table 2, non-penalized Logistic Regression (*β=1*) recommends 58.5% of the available loans at a precision of 88.9%. When *β>1*, precision increases and peaks at 95.9% at a β of 3.3, at the cost of recall and testing accuracy. For our purpose, this is fine, as high precision means low risk of losing money in a defaulted loan.

Table 2: Effect of Penalty Factor β

Beta	Training Accuracy	Testing Accuracy	Precision	Recall	Fraction Recommended
0.5	57.3%	79.2%	84.2%	91.7%	89.0%
1.0	64.0%	63.7%	88.9%	63.6%	58.5%
2.0	58.9%	37.4%	93.0%	25.3%	22.2%
3.0	54.2%	26.5%	95.4%	10.6%	9.1%
3.2	53.6%	25.0%	95.6%	8.6%	7.4%
3.3	53.3%	24.5%	**95.9%**	7.9%	6.8%
3.4	53.0%	23.9%	95.8%	7.2%	6.1%

Support Vector Machines (SVM)

Since SVMs have been a promising tool for data classification, we used LibSVM (Chih-Jen Lin) and Liblinear (Lin) libraries for our 2-class classification problem.

The main idea in SVM is to map data into a high dimensional space and find a separating hyperplane with the maximal margin. Given training vectors $x_k \epsilon R_n$, $k = 1,...,m$ in two classes and a vector of labels $y_k \epsilon R_m$, such that $y_k \epsilon \{1,-1\}$, SVM solves a quadratic optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + \sum_{k=1}^{\infty} \xi_k, \quad (5)$$

$$s.t.\ \ y_k(w^T\phi(x)+b) \geq 1-\xi_k,\ \ \xi_k{\geq}0,\ k=1,...,m$$

If data is linear, a separating hyper plane may be used to divide the data. However, in our data set, as the number of instances is larger than the number of features m >> n, mapping data to higher dimensional spaces(i.e., using nonlinear kernels) would be a better approach per section C.3 of the "A Practical Guide to Support Vector Classification" (Chih-Wei Hsu) on LibSVM. Besides running Liblinear algorithm on our data set gave us only 56% training accuracy. This tells us that our data is non-linear.

We noticed that Liblinear's running time is very fast irrespective of the sample size (less than a minute) whereas LibSVM is relatively slow and it takes 2 minutes for 20,000 samples. This is due to the fact that the complexity of the SMO algorithm implemented in LibSVM is O(n^2) or O(n^3) whereas in Liblinear it's O(n) (n is the number of samples).

To run the LibSVM algorithm on our data, we followed the procedure stated in the above mentioned SVM guide to format the data, choose the kernel, identify the best parameters and train the whole training set. Data scaling didn't add much value to the classification as our data model doesn't have variance issue.

A snapshot of four labeled feature vectors in libSVM data format:

```
-1 1:24000 2:24000 3:24000 5:18.55 6:874.3  7:17 ...
 1 1:9000  2:9000  3:9000  5:12.12 6:299.45 7:28 ...
 1 1:9450  2:9450  3:9450  5:16.29 6:333.59 7:22 ...
 1 1:7000  2:7000  3:6950  5:7.51  6:217.77 7:32 11.1...
```

Model Selection: The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and the soft margin parameter C. We started with the Gaussian/Radial Basis Function (RBF) kernel with a single parameter γ and found it to be the best kernel, when compared to polynomial and sigmoid kernels, for our classification problem.

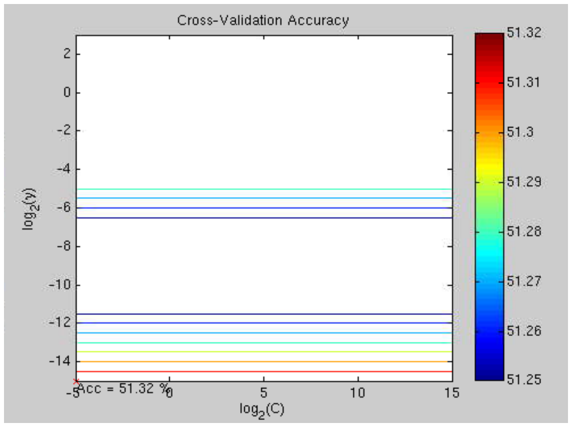
$$K(x_i, x_j) = exp^{-\gamma\left|\left|x_i, x_j\right|\right|^2} \quad (6)$$

We selected the best combination of *C* and γ by the grid search algorithm with exponentially growing sequences of *C* and γ,

$$C \epsilon \{2^{-5}, 2^{-3}, \ldots, 2^{13}, 2^{15}\} \ (7) \ ; \ \gamma \epsilon \{2^{-15}, 2^{-13}, \ldots, 2^{1}, 2^{3}\} \ (8)$$

Each combination of parameter choices was checked using 5 fold cross validation, and the parameters with best cross-validation accuracy were used to train the entire training set. Figure 3 shows the contour plot of parameter selection for Gaussian/RBF kernel using LibSVM.

Figure 3: Contour Plot: LibSVM



Solver: C-SVC classification model gave the highest precision percentage among the three solvers in LibSVM available for classification.

Penalty factor/weight (-wi): is used to set the parameter C of class i to weight*C. Since our primary objective is to increase the precision of our classification at the cost of recall and overall accuracy, we introduced higher weight (-w-1 1.5) for negative class to penalize false positives.

Table 3: Results for Gaussian Kernel
(Kernel Type -t 2 (RBF), Default Weight for Positive Class-w1 1)

Solver	Weights	Train Acc.	Test Acc.	Precision	Recall	% Recommendation
-s 0	-w-1 0.5	50.0	81.7	81.7	100.0	100.00
-s 0	-w-1 1	61.5	61.4	87.6	61.5	57.41
-s 0	-w-1 1.5	53.9	26.4	**93.7**	10.7	9.32
-s 1	-w-1 1	56.9	53.3	86.0	51.2	48.67
-s 1	-w-1 1.5	59.2	41.2	90.7	31.3	28.26
-s 2	-w-1 1	48.9	49.1	81.2	49.0	49.32
-s 2	-w-1 1.5	48.9	49.1	81.2	49.0	49.32

C-SVC solver (-s 0) with highest precision 93.7% recommends 9.32% loans at the cost of recall.

Figure 4: Precision, Recall, Prediction curve: LibSVM

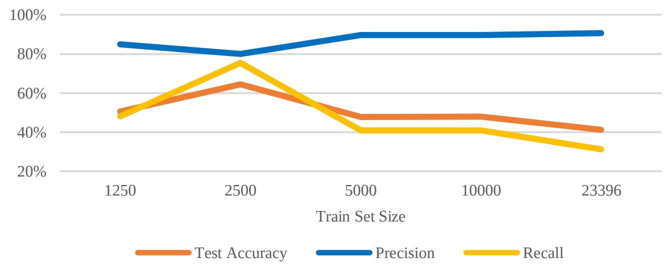


Figure 4 shows the AUC for precision, recall and prediction. AUC for precision is the highest.

Naïve Bayes

The Naïve Bayes (George H. John, 1995) implementation is taken directly from the Multinomial Event Model from CS229 Class Notes 2:

$$\mathcal{L}(\phi, \phi_{k|y=0}, \phi_{k|y=1}) = \prod_{i=1}^{m}\left(\prod_{j=1}^{n_i} p(x_j^{(i)}|y; \phi_{k|y=0}, \phi_{k|y=1})\right) p(y^{(i)}|\phi_y) \quad (9)$$

Continuous-valued features such as *annual income*, *revolving balance utilization*, and *loan amount* were discretized. After discretizing these features into a bucket of size 25% from 1%, we got an increase of 4% in precision.

Random Forest

Random Forest (Breiman, 2001) works as large collection of de-correlated B bag of trees and training data D of $\{(x_1,y_1),\ldots,(x_m,y_m)\}$.

```
1. for i=1:B
      - choose bootstrap sample D_i from D.
      - construct tree t_i using D; such that, at each node chose n random
        subset of features and only consider splitting on those features.
   end for
2. Once all trees are built, run test data through aggregated predictor.
3. Given x, take majority vote (for y=0,1) from different bags of tree.
```

Train accuracy was very high at 99%, but test set was very low, at about 60%. Because the trees that are grown very deep and learn highly irregular patterns, they overfit their training sets. Having more trees in the bag reduce the variance. We fine-tuned the model by gradually varying the tree size, number of random features, and depth to minimize out-of-bag errors: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

Table: 4 Abridged Out-of-Bag Error Minimization

Tree size	Depth	Random Features	Out of bag error
5	5	4	0.3823
15	15	4	0.3919
15	10	4	**0.3725**
10	10	5	0.3778

TF-IDF

The text attributes are very sparse as only a few loans have descriptions. After removing stop words, the vocabulary was 22,681 unique words. We used Term Frequency-Inverse Document Frequency (TF-IDF) to identify words better associated with either *y=1* or *y=0* loans:

$$tfidf(t,d,D) = tf(t,d) \text{ x } idf(t,D) \quad (10)$$

TF is a measure of how often a word appears in a document, normalized to document length. The more often it appears, the more weight it gives:

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d): w \epsilon d\}} \quad (11)$$

IDF is a measure of how special a word is. A word that exists only in a small fraction of the body of documents will have high weight:

$$idf(t,D) = \log \frac{N}{|\{d \epsilon D: t \epsilon d\}|} \quad (12)$$

TF-IDF ranking is constructed by ordering TF-IDF scored in descending order; the higher the score, the lower the ordinal rank (i.e. rank #1 has highest TF-IDF score). The top ranking (highest TF-IDF scores) words have similar ranking in both classes, which means these words are not discriminative of the class. However, as we looked at words in lower ranks, we started to see the differentiation that allowed us to better classify loans to be either *y=1* or *y=0*. A given word is associated more with the class with the lower TF-IDF rank, and the larger the difference between the two ranks, the more discriminate the word.

Table 5: Rank TF-IDF

Word	y = 0	y = 1
God	594	999
Steady	177	792
University	494	342
Refinance	118	128

As seen in Table 5, the word Steady is associated more with loans that are defaulted (TF-IDF ranking 594) versus paid off (lower rank of 999), whereas the word University is associated more with loans that are paid off (TF-IDF ranking 342) versus defaulted (lower rank 494). We checked the data, many applicants who mentioned that they will have a steady income or steady cash-flow and didn't have a permanent earning at present they end up defaulting the loan at later time. Based on this process, we created additional binary features such as IS_STEADY, IS_GOD, IS_UNIVERSITY. This gave 3% increase in performance on LibSVM over numeric-only classification.

Data Visualization

Our classification work in higher dimension space led us to believe our data is not linearly separable, and using a larger penalty factor β, we are operating in high precision, low recall space. To confirm this, we used Principal Components Analysis (PCA) to reduce the data dimension for visualization. Standard procedures were used:

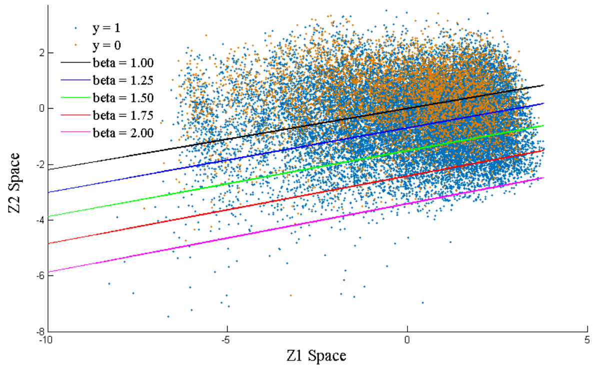
1. Perform mean subtraction and variance scaling on source data.
2. With normalized data, calculate covariance matrix:

$$\Sigma = \frac{1}{m} X^T X \quad (13)$$

3. Use SVD to identify the first and second principal components

As seen in Figure 5, there is significant overlap across the two data sets *y=1* and *y=0*. However, there is separation, as it also appears the center of mass for the two labels are distinct. Applying a large penalty factor β to Logistic Regression effectively shifts the decision boundary away from the center of mass to the region predominantly *y=1*. Below the red and magenta decision boundaries is a small fraction of data point with a high concentration of *y=1*; this region is high precision. However, these decision boundaries also leave above most of the data points, both *y=1* and *y=0*; this is why recall is very low. This confirms the behavior we see in higher dimensional space.

Figure 5: Logistic Regression on PCA Data



Performance

Comparing Algorithms

In this paper, we used four algorithms: Naïve Bayes, Random Forest, SVM, and Modified Logistic Regression (MLR). In our implementation, only SVM and MLR were instrumented with the ability to preferentially bias for one classification over the other. Therefore, since our goal is to optimize for precision, SVM and MLR had the best performance for our goal, as shown in Table 6. While the highest precision is from MLR on two-dimension PCA data, this classification only recommended 0.6% of loans, compared to over 6% for SVM and MLR. 0.6% recommendation means that for every 1,000 loans offered, only 6 will be recommended. This would be relatively impractical in a real investment strategy.

Table 6 also attempt to compare the performance of our algorithms to the different LendingClub Sub-Grades. For example, MLR with 96.0% precision at 3.23 beta is closest to LendingClub Subgrade A1 at 95.9%. Investing based on MLR, the investor will earn 1.7% higher average interest rate than investing based on LendingClub's A1 Sub-Grade, even though both have the same risk of default.

Table 6: Best Case Performance Comparison of Algorithms

	CS229 Best Performance			LendingClub Equivalent		
	Best β	Best Precision	Interest	Grade	Precision	Interest
Naïve Bayes	-	88.5%	11.0%	B1	88.6%	10.0%
Random Forest	-	88.8%	11.1%	B1	88.6%	10.0%
SVM-RBF	1.5	93.7%	6.9%	A2	94.2%	6.5%
MLR	3.23	96.0%	7.6%	A1	95.9%	5.9%
MLR on PCA	1.99	98.9%	8.0%	A1	95.9%	5.9%

Comparing to LendingClub

Emphasizing our original goal: to reduce exposure to loan defaults and exceed LendingClub.com's return given the same risk level. We will now compare the performance[1] of our classifier to that of LendingClub's. Our methodology is as follows:

1. Calculate the equivalent precision from LendingClub's data. For example, the precision of Grade A loans is the paid off loans in Grade A divided by all loans in Grade A. Repeat for all grades
2. Adjust β until our Modified Logistic Regression classifier precision is at the same precision level of the specific LendingClub Grade target.

[1] Because our Modified Logistic Regression (MLR) algorithm afforded the most flexibility in adjusting β, we will focus our comparison to LendingClub using MLR.

3. Compare at each precision level the interest rates and fraction of loans our classifier selected, compared to interest rates and fraction of loans that LendingClub selected.

Notice in Table 7 for Grade A, LendingClub classified 19.38% of total loans at an average interest rate of 7.5%. MLR classified 25.98% of total loans at an average interest rate of 10.3%. This means MLR offers the investor 6.6% more loan choices, with an average interest rate of 2.8% higher than LendingClub classification.

Table 7: Investment Performance by Grade

	LendingClub			Modified Logistic Regression		
Grade	Precision	LC Fraction	LC Rate	Beta	MLR Fraction	MLR Rate
A	92.4%	19.38%	7.5%	1.815	25.98%	10.3%
B	85.4%	30.89%	11.6%	0.900	38.58%	12.8%
C	79.9%	23.99%	14.8%	0.615	17.61%	15.8%
D	74.1%	15.10%	17.5%	0.436	9.97%	17.9%
E	69.0%	6.94%	20.1%	0.312	4.81%	20.1%
F	62.3%	2.99%	22.6%	0.218	2.51%	22.3%
G	61.7%	0.72%	23.8%	-	0.54%	23.8%

For the more conservative investor, the Table 8 is a breakdown for Sub-Grade A1-A5. Notice the similar effect where MLR is able to classify more loans in the higher precision category and at a higher interest rate.

Table 8: Investment Performance by Sub-Grade A

	LendingClub			Modified Logistic Regression		
Sub-Grade	Precision	LC Fraction	LC Rate	Beta	MLR Fraction	MLR Rate
A1	95.9%	2.52%	5.9%	3.230	7.18%	7.6%
A2	94.2%	2.94%	6.5%	2.860	3.16%	8.0%
A3	93.2%	3.50%	7.4%	2.570	3.21%	8.4%
A4	91.6%	5.23%	7.8%	2.120	6.56%	9.2%
A5	90.1%	5.18%	8.6%	1.815	5.87%	10.3%

What is most interesting is the breakdown of Sub-Grade A1 loans in Table 9. MLR classified 7.18% of loans in the equivalent of Sub-Grade A1 risk group, compared to LendingClub's 2.52%. That means LendingClub misclassified a number of high grade loans with high interest rate into a lower category. Compare Sub-Grade A1-3 breakout from MLR with Sub-Grade A4 from LendingClub; the former interest rate is 12-18% at a precision of 92.1%, whereas the latter is 9.2% at 91.6% precision. The astute investor can take advantage of this discrepancy to invest in loans that pay unusually high interest rates given their lower risk levels.

Table 9: Intra-Sub-Grade A1 Breakdown

Sub-Sub-Grade	Interest Rate	Total	Paid	Precision
A1-1	0-8%	1,561	1,511	96.8%
A1-2	8-12%	313	293	93.6%
A1-3	12-18%	89	82	92.1%
A1-4	18+%	9	8	88.9%

Lessons Learned

1. We initially achieved 98+% test accuracy using most of our algorithms, only to later find out that our data included information that is not available when a loan is first offered, such as *late_fee*, *recovery_fee*, and *current_fico_score*. This allowed our algorithms to "cheat" by looking into the future. We subsequently removed these features.
2. Some of our algorithms such as Logistic Regression are sensitive to population skew that was natural in our data. We rectified this problem by manually balancing *y=1* and *y=0* to 50/50.
3. When tuning parameters of a machine learning algorithm it's better to use a training sample with smaller size, otherwise it will take a very long time to find the optimal parameters as the algorithm need to run large number of iterations on the training set. We learned this lesson from the grid search algorithm for LibSVM.
4. As suggested by Prof. Ng, we learned that it's better to start with a quick and dirty approach before spending time and efforts on an approach that might not work.

Conclusion

As we stated earlier, our primary objective is to obtain a high precision that translates to low risk of losing money in a defaulted loan. We employed 4 machine learning algorithms on this task and found out that Logistic Regression outperformed LibSVM, Naïve Bayes and Random Forest. The secret ingredient to boost the precision is to increase the penalty factor for the negative class. This yielded higher precision at the cost of recall and prediction accuracy.

Future Work

1. An area of future work would be to perform sentence and sentiment analysis on those text features that might help in improving the overall accuracy and precision.
2. Ensemble of classifiers to build a strong classifier.
3. Reinforcement machine learning algorithm can be used to classify active loans with time series data.
4. Classifying loan data from other peer lending platforms to see how our classifiers perform on varying platforms.

References

Breiman. (2001). Random Forests Machine Learning. Retrieved from https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

George H. John, P. L. (1995). Estimating Continuous Distributions in Bayesian Classifiers. *Eleventh Conference on Uncertainty in Artificial Intelligence*, 338-345.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning.

Hall, M. A. (1998). Correlation-based Feature Subset Selection for Machine Learning.

Lin, C.-C. C.-J. (n.d.). LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2:27:1--27:27. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/libSVM

Manning, C. D. (2014). Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 55-60. Retrieved from http://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf

Ng, A. (n.d.). CS229 Class Notes 1 "Supervised Learning" and Notes 2 "Generative Learning Algorithms".

Acknowledgements

We are grateful to Professor Andrew Ng who has been the greatest inspiration in our Machine Learning journey. We would also like to thank all the TAs who have been immensely helpful. Last but not least, our heartfelt thanks to our spouses, children & parents for being really supportive and understanding.