Predicting Default Risk of Lending Club Loans

Shunpo Chang, Simon Kim, Genki Kondo
CS229: Machine Learning, Stanford University

Goal

Maximize expected returns from Lending Club investments by predicting the probability of default of loans.

Method

Given an imbalanced dataset, such as the Lending Club dataset where the rate of positive examples is about 85%, accuracy does not indicate the true performance of the model, as the accuracy will depend on the overall default rate of the test data set. Thus we will look at sensitivity and specificity:

$$\text{specificity} = \frac{TN}{TN+FP} \qquad \text{sensitivity} = \frac{TP}{TP+FN}$$

In order to combine both sensitivity and specificity, we will use the G-mean:

$$G = \sqrt{\text{sensitivity} * \text{specificity}}$$

We will train several learning models to find the best prediction, and to establish performance, we will train each model with the first 70% of the loans and test the trained models on the last 30% of the loans in our dataset.

Data preparation

Raw data

- Lending club dataset
- 60 features

Filtered data

- Filter out non-descriptive fields such as loan ID
- Remove fields for which > 10% of loans were missing data for
- Remove loans for which at least one field is missing

Expanded data

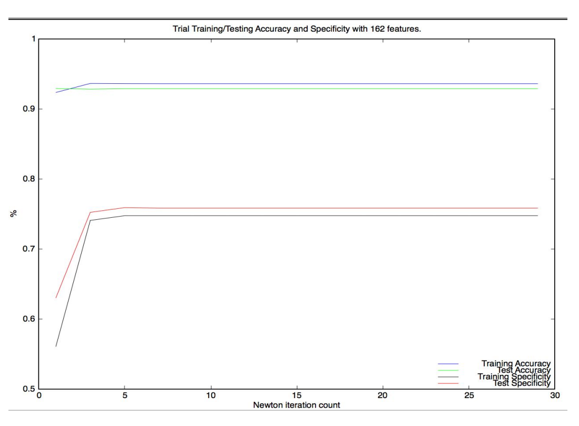
- Convert categorical fields into multiple boolean fields
- Join zip code with census data
- Conduct TF-IDF on loan description text to identify relevant keywords

Models

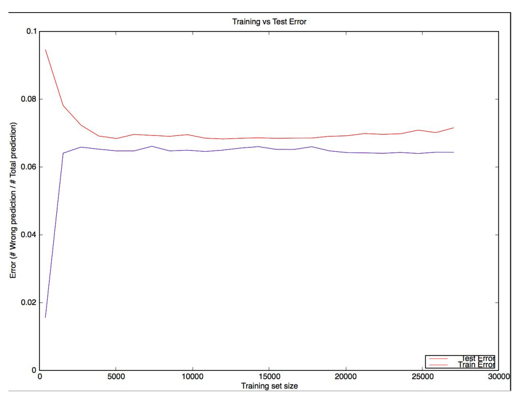
Logistic Regression

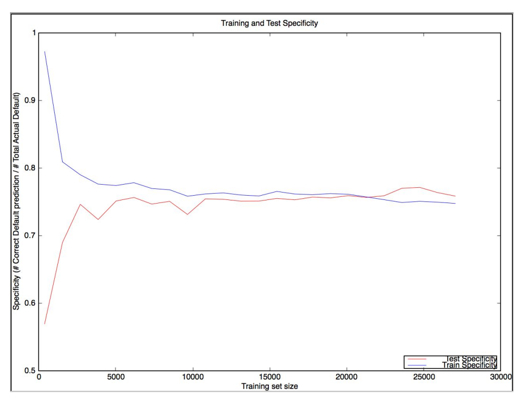
We first started modeling with logistic regression with Newton's method. To get boundaries of iterations needed for Newton as well as understand data features better, we first trained with all available features.

Training and test converges to an optimal solution within 5 Newton iterations.



From the diagnostic of bias vs variance, we see that the model still has high bias, but default prediction may benefit from better feature selections:





Based on the ablative analysis on all available features, we filtered out features that decreased the test specificity, and without hurting overall test accuracy, we managed to bump specificity from 75.8% to 77.1% for our logistic regression model with better feature selection.

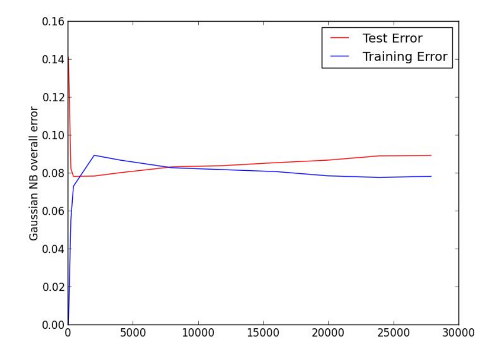
Accuracy	Precision	Sensitivity	Specificity	G-mean
92.8%	96.6%	95.2%	77.1%	85.7%

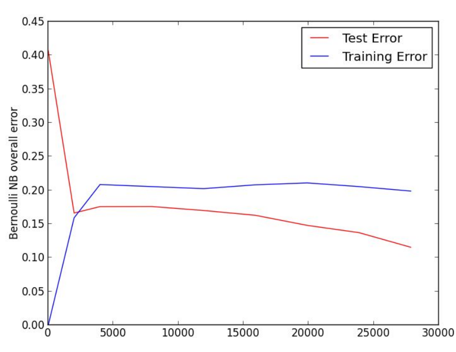
Naive Bayes

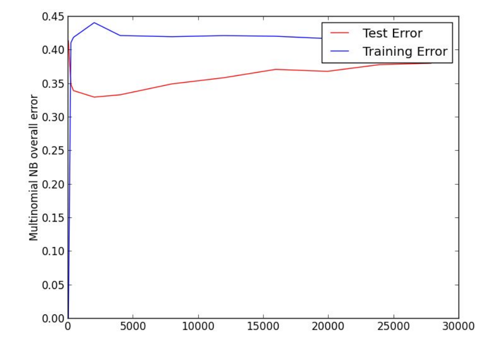
All values for features had to be converted to integers to work for Naive Bayes classification, and Laplace smoothing factor of 1 was used. Using Python Scikit's Naive Bayes library, the following results were achieved:

Naive Bayes	Accuracy	Precision	Sensitivity	Specificity	G-mean
Gaussian	**91.1%**	**96.6%**	**92.7%**	**80.4%**	**86.3%**
Bernoulli	88.5%	91.0%	96.2%	36.9%	59.6%
Multinomial	61.9%	88.7%	64.4%	45.8%	54.3%

Then, In order to determine whether we are seeing high bias or high variance, we compare the training error to the test error for each case of Naive Bayes:







Gaussian Naive Bayes shows the lowest test and training error and thus best prediction result. However, the error rate for Gaussian Naive Bayes is still relatively high (> 8%). Therefore, an additional feature of median income by zip code was included in the dataset and the test error for Gaussian Naive Bayes decreased from 8.94% to 8.65% (by about 0.3%).

SVM

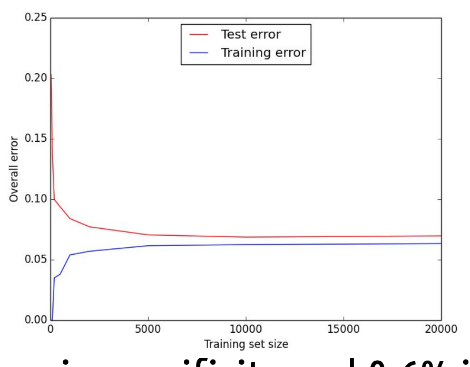
Since the training data is likely not linearly separable, and not guaranteed to be separable even in higher-dimensional feature spaces, we used L1 regularization (soft margin SVM). For training data points ($x^{(i)}$, $y^{(i)}$), the model is the result of the optimization:

$$min_{\gamma,w,b}\frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, ..., m$$

The performance of an SVM model depends on the kernel used, the parameters of the kernel, and the soft margin parameter C. We ran SVM with various kernels:

Kernel	Accuracy	Precision	Sensitivity	Specificity	G-mean
Linear	**93.0%**	**96.6%**	95.3%	**77.7%**	**86.1%**
Polynomial	93.0%	94.5%	97.6%	61.8%	77.7%
RBF	92.8%	93.9%	98.0%	57.5%	75.1%
Sigmoid	90.8%	91.8%	**98.2%**	41.0%	63.5%

Using a linear kernel, we see that the model has a high bias:



We added more :a" above), and see a 0.1% increase in specificity and 0.6% increase in G-mean. We then iterated on various values of C but ultimately C = 1 performed the best.

Accuracy	Precision	Sensitivity	Specificity	G-mean
93.7%	96.9%	96.3%	78.0%	86.7%

Results

From our model comparison, we found that Naive Bayes with Gaussian has the highest specificity (80%), and thus default rate prediction is best with Naive Gaussian. If we apply the model and avoid investing in the predicted-to-default loans, we can increase the return of investment* for the test set from 10.1% to over **15.8%** (**56%** increase).

* Note: $Return\ on\ Investment\ (ROI) = \frac{Total\ payment\ amount\ funded\ by\ investors}{The\ total\ amount\ committed\ by\ investors\ for\ that\ loan\ at\ that\ point\ in\ time}$