Intro1.

The data is from kaggle.com, lending club is the current leading peer to peer online loan platform market share wise. Our research goal is to answer this question with a simple Y/N to a new borrower given information. Since a loan default will cause both principal and uncollected interests losses, we want to prioritize on an as high as possible default case recall rate, also consider default case F-score and overall accuracy.

Intro 2

Here are some graphs descriptions. The loan status is our Y labels, and to simplify situation into binary choices, we combine the current, fully paid grace period, grace period as TRUE, “charged off, default, late” as False. The just issued are dropped.

Date Prep.

Speaking of which, we are already in the domain of data cleaning.

1. .We redefine the Y to True or false as I just said, the newly approved loans are omitted.
2. We start by deleting all the goal-irrelevant and redundant features. For example, information for administrative purpose like member\_id and issuded\_date, next\_due\_time etc. are dropped. And we also delete columns like grade, since all the information is contained in subgrade. Secondly, the features have the same values across all cases are dropped, like policy\_code and paymnt plan. Thirdly, we remove all the features contain more than 50% NA in them.
3. There is an interesting feature called description, which asks the application to fill in whatever they wanna say and it is optionally required. As a result, some left blank while some wrote a 200 words essay. We first wanna to use tf-idf (term frequency and inverse document frequency) to deeply dig it, nevertheless, due to the limitation of computation power, we settled with counting the string length of the description instead of ignoring it, which turned out to be worthwhile.
4. Step 4 is plain.
5. And we proceed step 5 with large caution. only (cov > 0.95) and multicollinearity associated with VIF>100(variance inflation factor) accountable columns are dropped. Very conservatiove
6. Normalize all numeric data as 0 to 1.
7. For logistic regression and boosting, we expand all the categorical to design matrix of each sublevel with dummy variable.

Naïve bayes

Now the data is ready. We launched by the most classic model, NB classifier for it’s simple and effective, well to some extent.

Since we all knew that NB is sensitive to collinearity, we have to further delete two more features to make it start working. Since it ignores the interactions between variables, some of the information in the data is ignored. This makes it an inherently high bias model, which makes it hard to accurately represent many complex situations. ; it has a high approximation error but as a result it also does not overfit. (A model with high variance attempts to model all of the data including the noise in the data). The results are shown in the table, the classifier performs well on the positive class but badly on the negative class, which might because of the high bias of the model.

But NB performs well with small data sets and missing data.

Logistic 1.

Then, we fired logistic regression with L1 regularization (LASSO) on the dataset hoping to achieve better result. As we known, LR is not very sensitive to the high correlations between features. But it is affected greatly by the imbalanced data distribution. We didn’t notice this until we saw the result under default setting where cut-off threshold is 0.5, meaning a probability higher than 0.5 will be classified as positive. Without any doubt, the models works for the True cases, aka, folks pay back, but the result for the minority case, false, folks who default, seems dreadful.

Logistic 2.

And this is caused by the imbalanced data. Out solution is to do the ROC curve approach. It is a [graphical plot](https://en.wikipedia.org/wiki/Graph_of_a_function) that illustrates the performance of a [binary classifier](https://en.wikipedia.org/wiki/Binary_classifier)system as its discrimination threshold is varied. Ideally, when the false positive rate is 0, the true positive rate is already 1. And from here, we took the closet point to (0,1) as our optimal cutoff threshold, 0.924. Now, we plug this new cutoff into our logistic regression, we could see with the same performance on True cases and overall accuracy, there is a considerable improvement on the prediction on the False cases.

Boosting 1.

Could we further improve our data? Definitely, especially, we are not entirely sure about how the data are obtained and whether logistic regression’s linear combination over all features is a good treatment? It could be entirely plausible that some nonlinear treatment is favored. Also, we cannot be completely confident about the reliability of step-wise gradient methods like shrinkage, LASSO, or LARS to select features. The way out here is component-wise gradient boosting. It optimizes the prediction accuracy and select the functional form and variables at the same time. . The resulting prediction rule is mathematically equivalent to General Additive Model, easy to interpret.

Boosting 2.

Due to the same reason, computation limitation, we use the data as is instead of oversampling the minority. After 5-fold CV, you could see, in either case, boosting has higher accuracy than basic logistic regression. We also tried to generate some points on the ROC curve, and it looks like the optimal cutoff should be near 0.8, which has a much better overall performance and close False recall.

RF. 1

Finally, we went for Random Forest, because we have heard so many good things about it.

First, parameter tuning. From the bottom plot, we could see growing 50-100 trees pretty much gives similar performance, so we went with 100 hoping for a stronger forest. From the right side plot, we could see the number of features per tree should be chosen as 5, which meets the conventional suggestions, square root of total feature then floor.

RF.2

Then, to mitigate the imbalance problem, we tried to tune the discriminative cutoff of voting, but looks like the default 50-50 setting is robust, which also fits the RF’s relatively immune to sample imbalance property. Given no need to do cross validation in RF, we tried to increase the weight on minorities, which is ruthlessly crushed by the fact we don’t have enough memory to allocate the data.

From here, we could see the final performance are very impressive and the rumor RF beats lots of algorithms is indeed true.

Comparison

Finally, when we put the performance together, we have to go back to the particular question: NB Logistic regression and boosting vs. Random forest, which is better? Here are some of our thoughts:

Both LR/boosting and RF are scalable and efficient, RF tends to edge out in terms of performance, the higher false recall of LR is at the cost of very low precision. But logistic regression has its own advantages like could incorporate new inputs online and provide potentially more useful probabilistic outputs. And NB is fast but doesn’t quite fit for this data set.

Since our client is sadly ourselves, so we might choose the best performing one, or we will take a lesson from the Netflix Prize and Middle Earth, just use an ensemble method to choose them all.

Some final thoughts, when data is huge, the performance gap of different algorithms is not that dramatic and it comes down to the speed and easiness.

Better data usually beats better algorithm, and good feature design gets you a long way.