Capstone 1 Milestone Report
Lending Club Loan Default Modeling

Thomas Loeber

Overview:
This project analyzes data from the peer-to-peer lending platform Lending Club. The goal is to predict whether a borrower will pay back their loan.

Audience and Problem:
The hypothetical client of is either Lending Club itself, or any other lending institution that does not have enough of its own data yet with which to do a similar analysis. Predicting loan defaults is of the major problems banks are facing, and achieving high accuracy has an immediate effect on their bottom line. The resulting model is used to decide whether to accept a customer's application for a new loan – and if so, what loan amount to offer and at which interest rate. In addition, the results could also be used to target marketing material at specific audiences that are most profitable.

Note that, while gaining the last bit of additional accuracy gets increasingly hard in any machine learning application, it also yields disproportionate results in a case such as this: If we are able to predict the default risk more accurately than our competitors, we will be able to identify customers which are particularly profitable: Not necessarily those whose default risk is low, because most of those will likely take their business elsewhere unless we offer them a commensurable low interest rate. Rather, the most profitable customers will be those whose default risk is low r*elative to how most of our competitors view them*. In order to attract those customers, we will want to offer them a slightly lower interest rate than they would get elsewhere, but this discount will be small enough to still yield a greater expected return to the lender. Conversely, we will also identify a segment of customers whose risk of default is larger than implied by the models of our competitors. While we do not want to turn those customers away, we will want to offer them a commensurately higher interest rate, which will drive most of them away to competitors who priced their risk too low.

Data Source:
A subset of the data is publicly available on Lending Club's website, but in order to access all the data I had to create an account. This data set is very comprehensive, it spans about 10 years (from 2007 – shortly after the firm's founding – to the present) and contains roughly 1 million usable observations for 150 variables.

Data Wrangling

I start by checking for any **duplicate** observations, but I did not find any.
Next, I set the issue date of the loan as the index, because this will allow us to more easily slice and group loans by date. Since oftentimes multiple loans were issued on the same date, I create a hierarchical index that also includes the loan ID.

I go on to **drop** any observations that are irrelevant because our target variable – whether someone defaulted on their loan or not – is yet indeterminate. These consist of observations where the loan is still current, and where the loan is late by up to 120 days.
Having gotten rid of unnecessary rows, we now drop unnecessary columns, starting with variables whose values were not yet available at the time the loan was issued . Doing so is important because using such information from the future would create the appearance of better predictive accuracy than the model actually possesses (endogeneity). At the same time, this step is also tricky, because the data documentation is often lacking; as a result, a few such variables are only discovered later on. Furthermore, I delete a few variables which are not relevant, as revealed by the data dictionary.

Next, I address the topic of **missing values**. This is a problem for roughly a third of our variables, because over time Lending Club started collecting additional groups of variables. I decided to discard all variables that have at least 30% missing values (though any threshold between 15% and 65% would have lead to the same results, because no variables had a proportion of missing values falling within this interval).
I made sure not to drop variables where missingness stands for "not applicable" rather than "not collected." Examples are variables such as time since last delinquency, because not all people in the sample have already experienced this event. I identified those variables by the fact that for those cases, missingness is distributed seemingly randomly across time, rather than confined to certain sample periods (usually the beginning of the sample).
Since all variables for which this problem occurred referred to the time that has passed since a specific event, I chose the following **transformation** for them: First, since for some observations the event occurred zero months ago, I added 1 to each value to make all the counts positive. Then, I raised each variable to the power of -0.5. Finally, I set the value to zero for all observations for which the underlying event did not occurr. (The fact that the event did not occur was indicated either by the value being missing or – for earlier observations – zero.) The result is that observations for which the result occurred more recently received a higher score; and the longer ago the event occurred, the closer the score moves to zero. For borrowers for which the event did not occur at

all, the score equals exactly zero, which models that this is equivalent to the event occurring an infinite time ago.

I then go on to check whether variables are of the right **data type**. Originally, all data are imported either as floats or objects. For floats, I identify variables that are in fact integers, and also don't have any missing values (which would require storing them as floats anyway). I then convert them to integers.

Next, I manually inspect all variables of type object. This identifies two variables that should be datetime objects, as well as three variables that should be numeric, but were imported as objects because the unit was appended to the value. I convert all these to the proper type. In addition, I change the datetime variables from absolute date to relative date (time since the loan was issued).
The remaining variables are in fact categorical. After the EDA, I will transform them using one-hot-encoding, after dropping variables with too many unique categories.

Exploratory Data Analysis
The main purpose of the explanatory data analysis (EDA) I carried out was to, firstly, assist in data cleaning by identifying potential problems in the data, and secondly to assist in feature engineering.

Another common use of EDA is to discover interesting relationships between variables (e.g., a non-linear effect that we may try to model by including a variable's square term). Let me justify why I did not make much use of this: Usually, we are interested in the *partial* effect of a variable, i.e. holding other variables constant. If we use visualizations, it is generally only practical to plot the effects of two or three variables at a time. This tends to give a misleading picture of partial effects (e.g., correlated predictors will give rise to spurious correlations). While one might argue that it can't hurt to try finding something interesting that way, we need to take the opportunity cost into account, so I think time is better invested elsewhere. For instance, if we use a flexible model such as gradient boosting, we don't have to worry about trying to visually detect all nonlinear effects and interactions between predictors.

A better use of our time is to use EDA to **find problems with the data**. The reason is this often draws on our domain knowledge, as well as general human knowledge about the world. Both of these are hard to encode into a model, such as the fact that we know that certain variables has to fall within a particular range. Another common problem is that some missing values may have been encoded as zeros or missing values standing for "not applicable" rather than that the true value is unknown.

In fact, I detected both of these problems with Lending Club's data through data visualization. The first problem was revealed while I was deciding which variables have a high enough ratio of valid to missing values to impute the missing values (rather than dropping the whole variable). In making this decision, I not only considered the proportion of valid observations, but also visualized how the missing values are distributed over time. This showed that – unsurprisingly – a high fraction of missingness was usually caused by the fact that the variable was only collected for a subset of the sample period. However, it was suspicious that for some variables the missing values were distributed seemingly randomly across time. (About half the values were missing, so it is unlikely that such a high proportion simply failed to be properly recorded).
This prompted me to take a closer look, which revealed that missingness seemed to denote not "unknown", but rather "not applicable": For example, the variable "time since last delinquency" was not applicable to borrowers who never experienced a delinquency. Unfortunately, we don't have two different types of missing values available to encode each. I solved this problem by setting the "time-since" variable to negative infinity for each observation where the event count (e.g., the number of delinquencies) was zero. (Setting it to negative infinity works because I later transformed the "time-since" variables by raising them to the power of -0.5, which gives a higher score to applicants to whom the event occurred more recently, and converges to zero as the time-since approaches infinity).

In the process of making sense of these inconsistencies, data visualization also uncovered a related problem, namely that "not applicable" was encoded not always as missing, but in some cases instead as zero. It turned out that this practice seems to have changed at some point in the middle of the sample period. Since the data dictionary did not address this, I had to use a number of different plots to figure out what was going on. For example, I looked only at the subset of borrowers who had a zero for the number of months since the last delinquency, and then plotted the monthly average for the number of delinquencies. This revealed that the average monthly number of delinquencies was zero until 2010, but from 2013 on hovered around 2 (there were no observations for the years in between). This suggested that the practice of encoding missing values as zeros stopped somewhere between 2010 and 2013, but this interpretation raises some other inconsistencies, which further plotting showed to be due to a change in Lending Club's lending behavior. (This explained, for example, why there were no zeros at all during the middle of the sample period: By that time, "not applicable" was denoted by "missing", and the lending behavior was still so cautious as to not accept any applicants who had just experienced a delinquency zero months ago.)

A different use of data visualization was to assist in **feature engineering**. The goal was to create features that are less skewed, in particular by taking the logarithm where necessary. I eventually automated this task by quantifying skewness, but in the process it was still helpful to use data visualization in order to come up with good measures and to make sure no errors were introduced. For example, the measure of skewness I settled on looked at how much closer/further the median was from the upper quartile compared to the lower quartile. Thus, this measure did not work for variables that were so skewed that all three quartiles fell onto the same value, so I had to visually decide what was going on with these distributions. Furthermore, plotting some of the extremely skewed variables showed that for many variables, skewness was caused by an inflated number of zeros (located at the minimum) or ones (located at the maximum). For example, for most applicants the number of derogatory public records was zero. This prompted me to take a second step to remediate skewness, namely to create another binary feature for these extremely skewed variables, telling us which values are equal to this inflated minimum or maximum.

Overall, thus, exploratory data analysis through data visualization was able to both detect some problems with the data, and also helped create better features. While I tried to automate these manual tasks as much as possible (e.g., by writing functions that try if taking the logarithm decreases skewness), some of these tasks are (still) too hard or time-consuming to automate.