Machine Learning Engineer Nanodegree

Capstone Project

Ekaterina Kravtchenko November 26th, 2018

Abstract

In this project, I created a classification model which predicts, from the point of view of an individual lender, whether peer-to-peer loans will be repaid, or not. To this end, I compared labeled historical Prosper loan data to the repayment status predicted by several supervised learning algorithms. The original dataset includes 113,937 loans, with 81 features per loan, including loan amount, interest rate, demographic information, and actual historical loan status. This information is particularly important for Prosper lenders, for the purpose of determining whether to lend money to a particular borrower, and for the company towards determining whether to offer someone a loan, which terms to offer, and what risk category to assign to a borrower (although the model is geared towards potential lenders).

I. Definition

Project Overview

The main problem in the lending business is determining whether to offer a given borrower a loan, what amount is safe to lend them, and what interest rate to charge them. Peer-to-peer lending companies offer loans, under company-determined terms, to potential borrowers, and then allow private individuals signed up with the service to choose whether to contribute to funding the loan (e.g., a private individual can choose to contribute 5% of the loan amount to a given borrower). These companies have the additional problem of determining how to accurately present loan risk (i.e., risk of non-payment or defaulting) to potential lenders.

In the first case, the company must determine, on the basis of background financial and demographic information, the loan amount, and other factors, whether to offer a potential borrower a loan in the first place. In the second case, potential lenders must decide whether they're willing to lend a potential borrower money. For these lenders, an accurate assessment of risk is very useful.

Given that we have a large database of historical information on which loans were in the end repaid, or not, this is fertile ground for building a supervised learning model to predict repayment status. A version of this model may assist a company in deciding who to offer a loan to in the first place, and on what

terms, and assist potential lenders in deciding who to risk lending money to. My personal motivation was to systematically investigate a rich and complex data set that I previously worked with, to see if I can get additional insight into qualitative patterns I observed, and to see if I can improve on Prosper's less formal predictions of repayment likelihood.

There have been many previous attempts at building models to assist lenders in deciding whether, and how much to invest. Published models use deep learning methods (e.g., https://arxiv.org/abs/1810.03466v2; http://arxiv.org/abs/1811.06471v1), coordinate descent (e.g., http://arxiv.org/abs/1705.08435v2), and a wider variety of models tackle credit worthiness in general (e.g., see https://www.moodysanalytics.com/risk-perspectives-magazine/managing-disruption/spotlight/machine-learning-challenges-lessons-and-opportunities-in-credit-risk-modeling for an overview). As this source discusses, Random Forest and Boosting algorithms are particularly well-suited to this problem.

Problem Statement

The problem to be solved is to predict, from the point of view of a potential lender, whether a loan will be repaid, or not. This can be done in terms of probability of repayment, or a categorical determination by a model. This prolem concerns only historical loans, as the end status of loans currently in repayment is not known. Fundamentally, a lander will want to know whether or not a loan, given the demographic and other data available to them at the outset, is likely to be repaid.

To this end, I built a classification model which predicts loan repayment status (whether the loan was in the end repaid, or not). The accuracy of the model is evaluated using metrics detailed below. The dataset contains a large number of features, many of which are correleated with one another, as shown in the exploratory analysis. To see if I can simplify the dataset, I use Princial Component Analysis (PCA) for dimensionality reduction.

The techniques I use are largely those accepted in determining credit risk - namely, ensemble methods such as random forests and boosting algorithms (in my case, AdaBoost and XGBoost). I also use Logistic Regression, as the easily interpretable results, which give the likelihood that particular loans will default, make it an attractive method. I use a grid analysis to determine optimal parameterization of the most promising model, given performance of the models with their default settings on the training and test datasets.

Metrics

The evaluation metrics I am primarily concerned with are the following:

- Recall, which calculates how many of the completed loans are in fact labeled as completed:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- Precision, which calculates how many of those loans labeled completed are, in fact, completed (as opposed to defaulted). This would appear to be the metric of most interest, as we are most concerned with predicting loans likely to default:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- The F1 score, which seeks a balance between recall and precision:

$$2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

II. Analysis

Data Exploration

The original dataset is linked here: https://s3.amazonaws.com/udacity-hosted-downloads/ud651/prosperLoanData.csv

I previously cleaned and performed an exploratory analysis of this dataset for the Data Analyst Nanodegree. This is a very large historical dataset of peer-to-peer loans, including background financial and demographic information on borrowers, the risk status assigned by Prosper itself and likely lender yield, and most importantly, information on whether past loans were in fact repaid, or not. The dataset contains information on 113,937 loans, with 81 continuous or categorical features per loan.

I use a slightly modified version of the dataset, which I already cleaned and organized, which is generated by the following R code: https://github.com/eskrav/udacity-data-analyst/blob/master/explore-and-summarize/explore-and-summarize.Rmd. I further cleaned and modified it for use in this project.

Exploratory Visualization

As can be seen in Figure 1, some features in this dataset are quite clearly (as well as, in this case, purposefull) predictive of loan repayment status.

This is information explicitly provided by the loan company for the purpose of helping the potential lender decide whether a borrower is trustworthy. Although

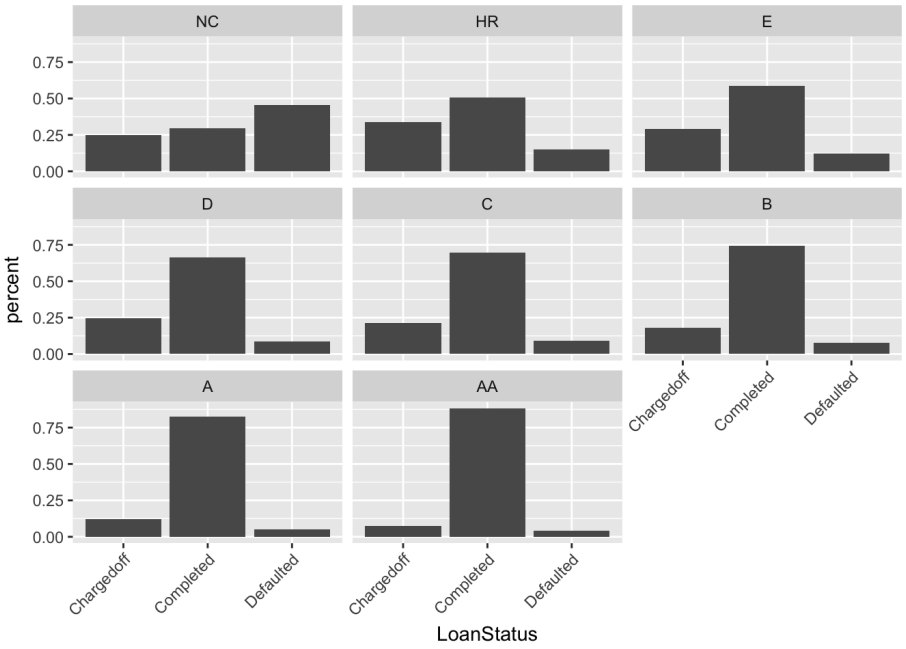


Figure 1: Loan Repayment by Prosper Rating

these loan ratings are by no means perfectly predictive, it is clearly the case that the higher the rating assigned by the company, the higher the likelihood of loan repayment.

Other features are not specifically designed by the company to be predictive of loan repayment, and yet appear quite informative, as can be seen in Figure 2.

Here, one can clearly see that the more the potential borrower earns, the more likely they (unsurprisingly) are to replay the loan.

More visualizations of how various features in the dataset correlate with likelihood of loan completion and higher returns for the lender can be found here: https://github.com/eskrav/udacity-data-analyst/blob/master/explore-and-summarize/explore-and-summarize.html.

Algorithms and Techniques

For this task, I picked three Ensemble methods commonly used for the task of classification - one averaging method, and two boosting methods. I also picked Logistic Regression, as having an output of probabilities allows for more interpretability, flexibility after the fact in setting thresholds for decisions. I broadly retain some of my previous descriptions of these models, where I have

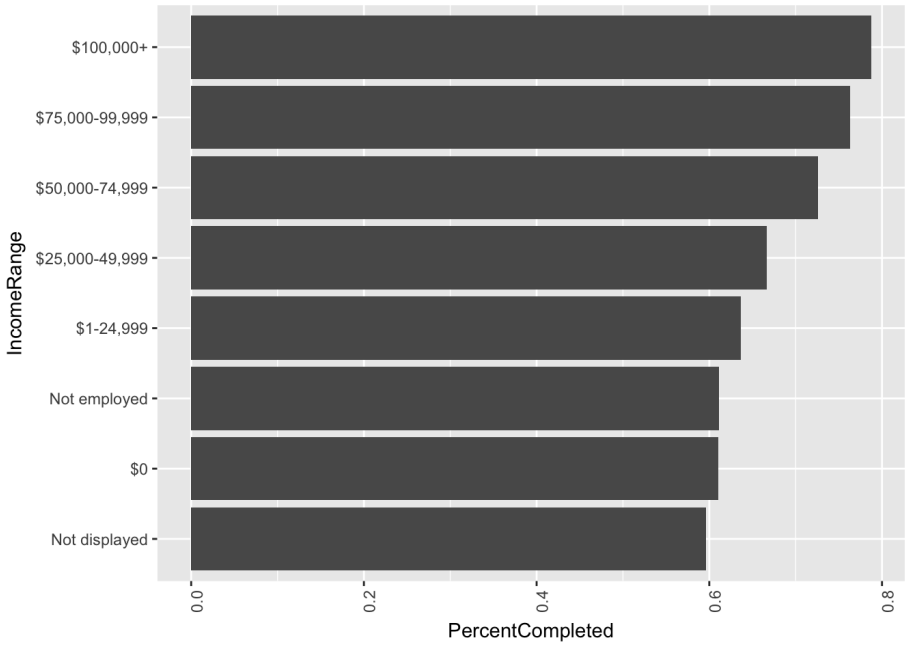


Figure 2: Loan Repayment by Income Range

previously used them. All of these models will be initially evaluated on the training and test data with their default settings. The model(s) with the most promising metrics will be further optimized.

For strengths and weaknesses of the relevant models, I primarily consulted the following sources, keeping in mind that heuristics do not necessarily apply to each data set:

https://medium.com/@randylaosat/machine-learning-whats-inside-the-box-861f5c7e72a3

https://medium.com/@vijaya.beeravalli/comparison-of-machine-learning-classification-models-for-credit-card-default-data-c3cf805c9a5a

https://www.dummies.com/programming/big-data/data-science/machine-learning-dummies-cheat-sheet/

https://hackernoon.com/boosting-algorithms-adaboost-gradient-boosting-and-xgboost-f74991cad38c

https://medium.com/@grohith327/gradient-boosting-and-xgboost-90862daa6c77

Ensemble Methods: Random Forest

The strengths of this model are the following: it natively handles categorical variables; it is less prone to overfitting than a single decision tree – therefore, it is more likely to select relevant features; and it frequently outperforms other methods on accuracy. **These features make this model a good candidate for this problem, which has numerous categorical and continuous features. Given that after one-hot conversion, there were 88 total features, a model which will automatically select the most important features is of particular importance.** Additionally, it can handle imbalanced data, such as this dataset, and is a flexible algorithm that does not require lots of parameter tuning.

The weaknesses of this model are that it won't perform well with a bad set of features; it's not very transparent, and it's hard to interpret what's going on in the algorithm; further, too many trees can slow down the algorithm.

Ensemble Methods: AdaBoost

AdaBoost is particularly well-suited for boosting the performance of decision trees on binary classification tasks. It typically does not overfit despite excellent accuracy (although it is to date unclear exactly how), and frequently outperforms other methods. **These features make it a good model to attempt for this problem, given that the algorithm can be used to boost the performance of classifiers that work well natively with categorical data, such as decision trees, and like RF will automatically select those features that are most important.** It additionally can handle missing variables, and does not require variable transformation, and has relatively few parameters that need tweaking.

The weaknesses of this model are that it does not deal well with noisy data, and the efficiency of the algorithm is affected by outliers, since the algorithm attempts to fit each point.

Logistic Regression

The strengths of this model are that it's fairly easy to interpret in terms of probabilities; relatively unlikely to overfit; fast; well-suited for binary classification tasks; and explanatory variables can be continuous or categorical. **These features, and in particular the fact that results are probabilities, make this model a good candidate for this problem, since probabilities allow for post-hoc adjustment of the threshold for whether a lender should fund a particular loan, perhaps depending on their personal finances or risk aversion.**

The downsides of this model are that it's not particularly good at capturing complex or non-linear relationships between features, or dealing with multiple/non-linear decision boundaries; generally, it's not very flexible.

Ensemble Methods: XGBoost

XGBoost was recommended by a reviewer, is highly robust to irregularities in data, and like AdaBoost, is a boosting algorithm which tries to create a strong classifier from a series of weaker classifiers. **Given that the dataset I have is quite noisy, with quite a bit of missing or imputed data, and is particularly suited to decision trees, this algorithm would be appropriate to try.**

Benchmark

The benchmark model I am using would be the following analysis, which ended up using a Random Forest classifier: https://www.kaggle.com/jschnessl/prosper-analysis/notebook. I was not able to immediately locate any other analysis using the same dataset, or reasonably current Prosper loan data.

This model achieves a Recall score of 0.74, a Precision score of 0.44, and an F1 score of 0.55. As the author of this model points out, this model would result in a return of -23.25%, excluding additional fees, which would most likely be unacceptable in the real world (although, as I pointed out in my exploratory analysis of Prosper loan data, the predictive models, or other methods, that Prosper uses appear to grossly overestimate lender return).

III. Methodology

Data Preprocessing

The pre-processing steps can be viewed in more detail in this file: https://github.com/eskrav/udacity-machine-learning/tree/master/capstone-project/capstone_project.html. The data that I imported was already pre-cleaned, explored, and summarized, as stated above.

All current loans had already been removed from the dataset, as I was solely interested in historical loans which have already been repaid, or defaulted. The few historical cancelled loans had been removed from the data, as they do not result in any gain or yield for the lender, and therefore are not of much interest to lenders deciding whether to lend money.

Several new features were already been added to the original data - for example, a simple string and numerical binary factor indicating whether the loan was completed or not (with no regard for precise default status); a continuous feature indicating what percent of their investment the lenders in fact earned back on the historical loans, and so forth.

In total, the dataset contained 55084 data points corresponding to individual loans, with 32 continuous and 23 categorical features. Although the data was

cleaned, there was still missing data that was not imputed, and non-normally-distributed continuous features which needed to be transformed, as well as scaled along with other continuous variables.

First Steps

Some features irrelevant to the task at hand, or redundant with other features, were immediately removed:

The following are internal loan and borrower identifiers, which are no use to the task at hand: `ListingKey`, `ListingNumber`, `LoanKey`, `LoanNumber`, `MemberKey`.

The following concern internal group membership, about which we do not have any further information (further, most loans do not belong to groups): `CurrentlyInGroup`, `GroupKey`.

The following concern information that is not available to the lender at the time they are making the decision of whether to lend money, or is not directly informative about whether the loan is closed, or not: `ClosedDate`, `LP_CustomerPayments`, `LP_CustomerPrincipalPayments`, `LP_InterestandFees`, `LP_ServiceFees`, `LP_CollectionFees`, `LP_GrossPrincipalLoss`, `LP_NetPrincipalLoss`, `LP_NonPrincipalRecoverypayments`, `LoanCurrentDaysDelinquent`, `LoanFirstDefaultedCycleNumber`, `LoanMonthsSinceOrigination`, `PercentYield`.

The following are redundant with the `Rating` feature, which combines pre- and post-2009 Prosper rating schemes: `ProsperRating.num`, `ProsperRating.alpha`, `CreditGrade`.

The following are redundant with the label of interest (`Completed.num`), which indicates whether the loan was repaid in full, or defaulted: `LoanStatus`, `Completed`. The following are redundant with `CreditScore`: `CreditScoreRangeLower`, `CreditScoreRangeUpper`. The following is redundant with the loan origination date: `LoanOriginationQuarter`. The Following is redundant with the borrower rate: `BorrowerAPR`. The following is redundant with monthly income, which as a continuous feature is more easily interpretable: `IncomeRange`. The following is redundant with age of credit history at the time the loan was originated: `FirstRecordedCreditLine`.

Almost all loans are completely funded (`PercentFunded`), so this feature was removed.

Datetime features were converted to datetime, and then numerical timestamps.

Data Imputation

`EstimatedEffectiveYield` was a measure not available prior to mid-2009. However, it measures the borrower rate minus expected fees, which can be imputed. Given the data is generally not normally distributed (which will be examined further below), unless stated otherwise, missing data is imputed with

the median of the relevant metric. In this case, this is the median difference between borrower rate and estimated effective yield, as implemented in https://www.kaggle.com/jschnessl/prosper-analysis/notebook. Other measures of estimated gain or loss, which were only implemented post-2009, cannot be easily imputed, although a linear regression might be used in a future analysis to infer likely values. For now, I leave these measures as is, and do not use them in the later analyses, as the values are missing for about half the dataset.

For most continuous features, missing values are imputed using the median feature values. The mean is avoided due to the distribution of values for many of these features being skewed. The missing values for features related to prior Prosper Loans are coded as 0, given that information is only present for those borrowers who had prior Prosper loans.

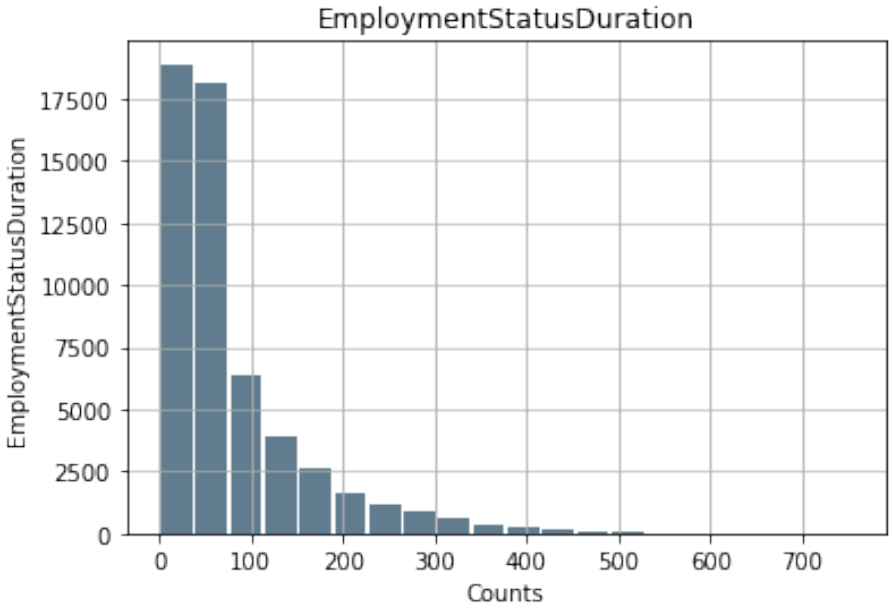
The following categorical variables are highly complex, and would need to me grouped meaningfully (for example, by region, working class, or state GDP) to be useful: `BorrowerState` and `Occupation`. For the time being, I have dropped these features from analysis.

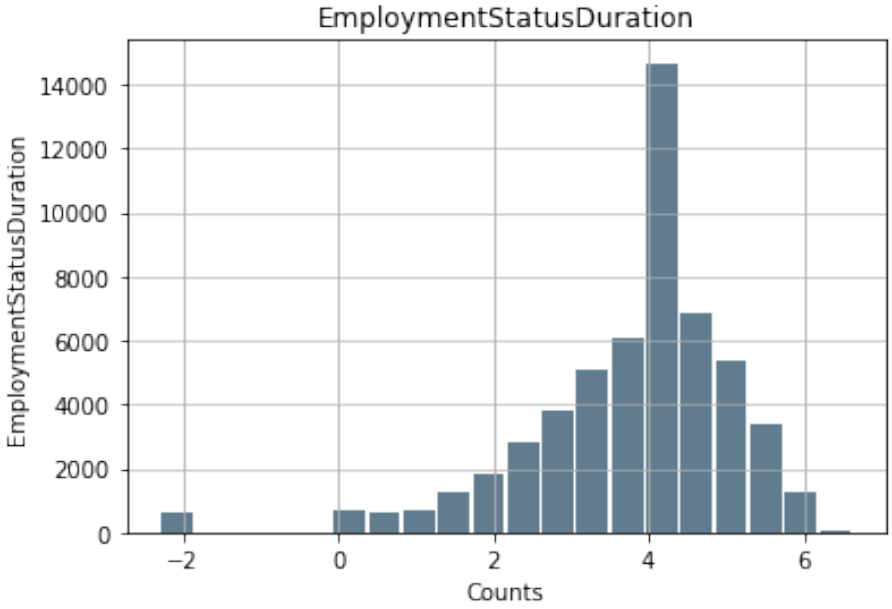
Missing values for the following categorical features are coded as "Unknown": `EmploymentStatus`, `Rating`.

At the end of this process, there are 55,084 records left, with 69.12% of these loans completed.

Data Transformation

After plotting the continuous features in this dataset, it became apparent that many were significantly right- or left-skewed. To prevent extreme values from significantly influencing any algorithms I use, I log-transformed all features that showed noticeable skew. For instance, while not quite normally distributed, `EmploymentStatus` shows significantly less skew after being log-transformed:





Afterwards, all categorical data was dummy coded, since many algorithms take only numerical input. After dummy coding, the dataset is leave with 88 total features.

Dimensionality Reduction

I then used Pricipal Component Analysis 9PCA) to see if the features in my dataset could be reduced to fewer dimensions, which could improve algorithm performance, and reduce feature redundancy. I first split the data into training and testing datasets, and scaled all numerical features to ensure that they are treated equally by algorithms, rather than any being given undue weight.

Several training and testing sets were created: one that simply lacked the columns with missing values, one that reduced the features to 10 components, one that reduced them to 3 components, one that reduced them to the most influential 30% of features, and one that reduced them to the most influential 10% of features.

Implementation

I first calculated the metrics of a naive predictor - which assumed a loan was always repaid - and which any algorithm I chose would need to perform better than. This naive predictor has an accuracy score of 0.6912, an F1-score of 0.8174, a precision score of 0.6912, and a recall score of 1.

I then borrowed a testing and training pipeline from a previous Udacity project, which systematically takes each algorithm, trains it on 1%, 10%, and 100% of the data, evaluates the results on a subportion of the testing set, and gives benchmark speed and accuracy metrics for my algorithms of choice (prior to any fine-tuning).

Overall, the AdaBoost and XGBoost algorithms appeared to consistently perform better than the Random Forest and Logistic Regression algorithms. AdaBoost was generally faster than XGBoost, and on the full dataset (minus features with missing values), seemed to strike a good balance between speed, high precision on the test set (~77%), high recall (~84%), and a relatively high F1-score (~81%). As both of these algorithms are well-suited to the problem, I use AdaBoost given its faster performance and slight advantage on relevant metrics.

Refinement

To improve the model, I did a grid search analysis of the AdaBoostClassifier, gathering the F1, precision, and recall metrics for each model. I varied the number of estimators (between 10 and 1000), the learning rate (between 0.001 and 1), and the algorithm (SAMME vs. SAMME.R). The optimal model chosen by the grid search was (. . . . . ..).

IV. Results

Model Evaluation and Validation

As mentioned, the optimal model in the end was the AdaBoost Classifier, with (. . . . . ..). This was identified through a process of evaluating different candidate models with several different versions of the dataset, and then optimizing the most promising model using a grid search through the common range of parameters.

I would argue that the model is robust to changes in the training data, given that I have tested it with several variably reduced versions of the dataset, and it in general performed well, and typically better than other models or on par/close to the performance of XGBoost. Further, it generally performed well on significantly smaller versions of the training data (1%, 10%).

I further ran a sensitivity analysis

Justification

Overall, my model performs higher than both the benchmark I chose, and the naive predictor: (. . . . . ..).

The results are stronger, but only marginally so. Ideally, I would like to work on this problem further to improve the performance of my model, and to attempt to optimize multiple promising models to see which gives the best performance. Given the potential money at stake (as I showed in my exploratory analysis, lenders are likely to lose far more money than Prosper estimates), it would be important to develop a more precise model.

V. Conclusion

Free-Form Visualization

Although I believe that predicting loan default, from the lender's point of view, is a very important goal, as can be seen in the plot, it is clear that Prosper are already able to predict, within some margin of error, whether a loan will default or not. Although it is problematic that up to ~7% of the highest-rated loans default, it is nevertheless clear that these ratings are quite informative about the likelihood of a loan being charged off in the end.

Reflection

Overall, throughout this project, I took a previously cleaned and explored dataset, and prepared it for use in a supervised learning algorithm. I did so by imputing or excluding missing data, transforming features that were not normally

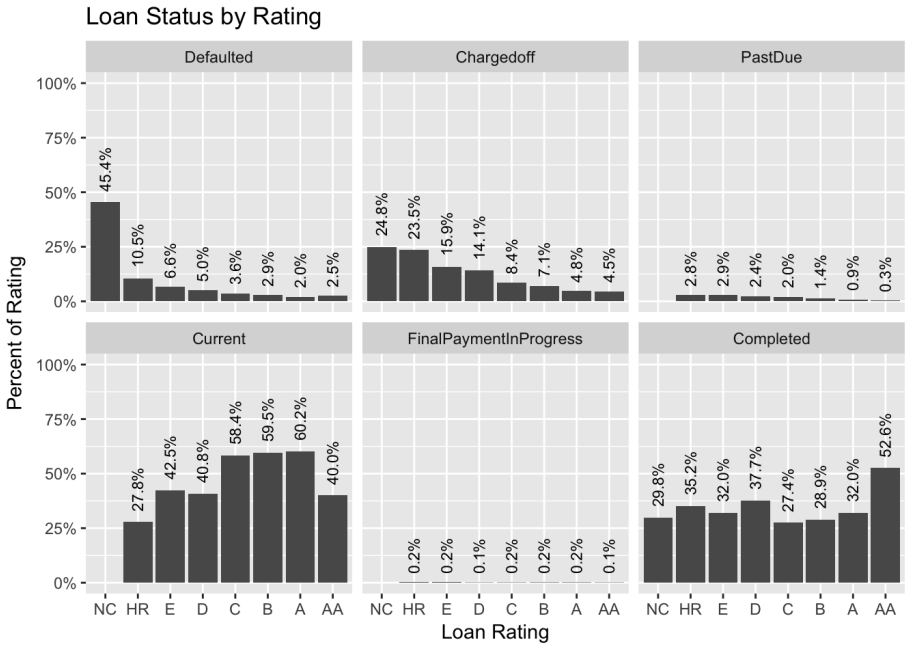


Figure 3: Loan Status by Prosper Rating

distributed, scaling numerical variables to prevent significantly higher or lower numerical values from giving the feature undue weight, and dummy-coding the categorical variables. I then created several different testing sets, with variable amounts of dimensionality reduction using principal component analysis. Finally, I the testing and training data through base versions of the models I was most interested in using, and chose the most promising model based on its performance on the metrics of interest, and its training/prediction speed. Finally, I performed a grid search to find the optimal parameters for this model.

Improvement

I believe, first of all, that this model could be improved through consulting more closely those models that were in the past successful in predicting credit trustworthiness, and implementing those features of the models used that made them successful, to the degree possible. Second, I would like to work at optimizing more than one model, in order to have a better line of coparison than that between several unoptimized models. Finally, I would like to build a Deep Neural Network model to see if it improves classification performance on this dataset. I would also like to use any models I come up with on similar datasets, to see if their performance generalizes.

Second, during my initial exploratory analysis, I found that Prosper systematically overestimated how much money lenders were likely to earn from a given loan. In the future, I would also like to attempt a regression analysis that can perhaps predict, with greater accuracy, how much a lender stands to gain or lowe from a given loan.
