Lending Club loan default modeling

Capstone project for Springboard

Thomas Loeber

Problem:

- Accurate prediction of which borrowers will default on their loans
- Squeezing out the last bit of accuracy gives disproportionate results
- Less important (not worth sacrificing accuracy for): Interpretability (e.g., which variables are most important)

Data:

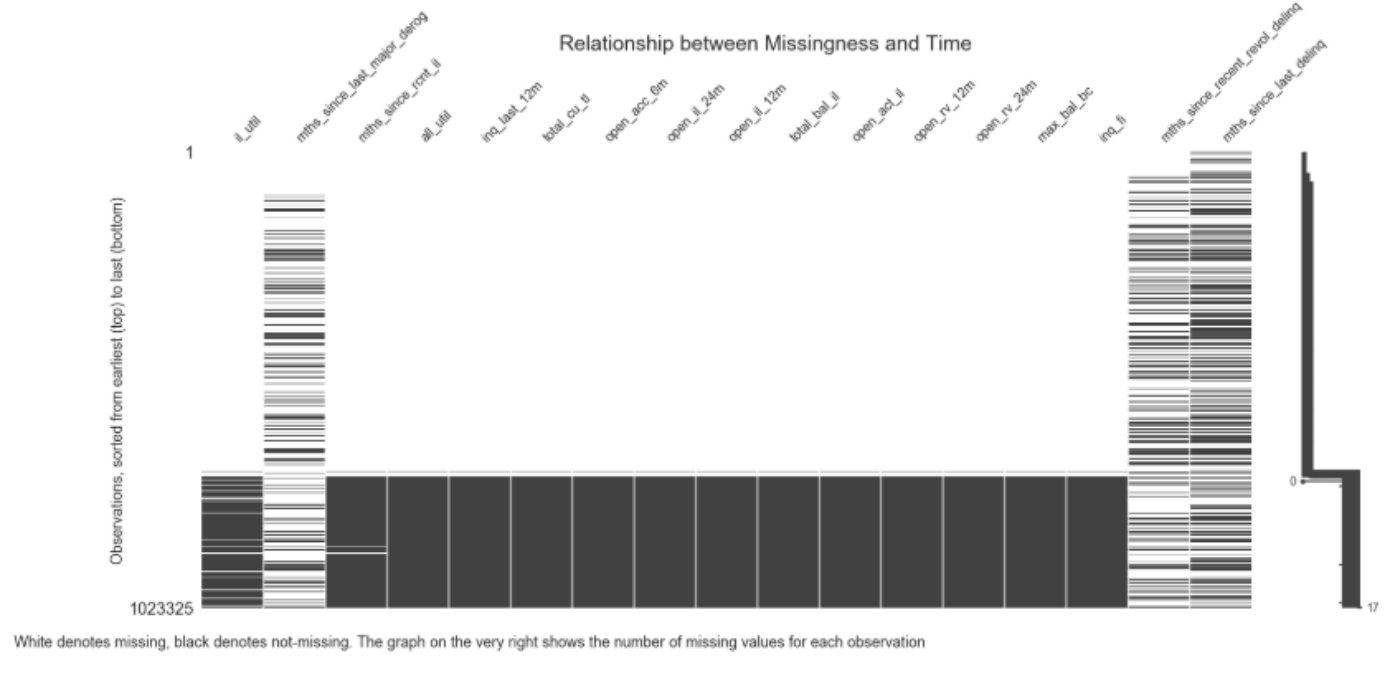
- Publicly available on Lending Club Website
- About 1 million usable observations
- About 150 variables (100 usable)
- We will model default as binary (do not have all the information to predict net present value)

Data Wrangling:

- Dropping variables with more than 50% missing values, impute otherwise using median/mode.
- Dropping observations where outcome is ambiguous or not yet known.
- Dropping endogenous variables (where value is unknown at the time the loan decision is made).
- Converting variables to the proper data type
- Transform skewness by taking the log where necessary. For extremely skewed variables with a mode at the minimum or maximum, add a binary variable for mode (e.g., zero delinquencies).
- Use domain knowledge to engineer better features (e.g., divide installment payment by the applicant's income)

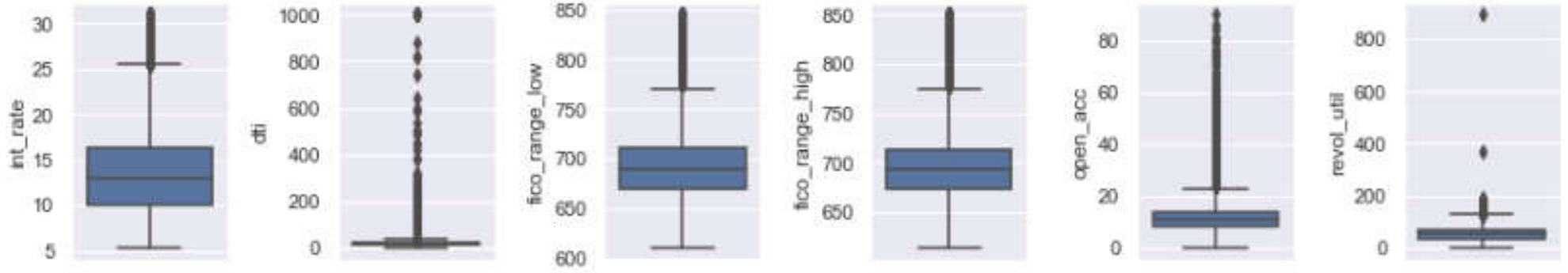
Exploratory Data Analysis 1:

- Inspect whether missing values are caused by the fact thata variable was only collected for a subset of the sample time.
- If the variable has a lot of missing values, but these are distributed across time seemingly randomly, we should take a closer look. (Could mean variable is not actually missing but rather not applicable, such as the number of delinquency for an applicant who never experienced one).



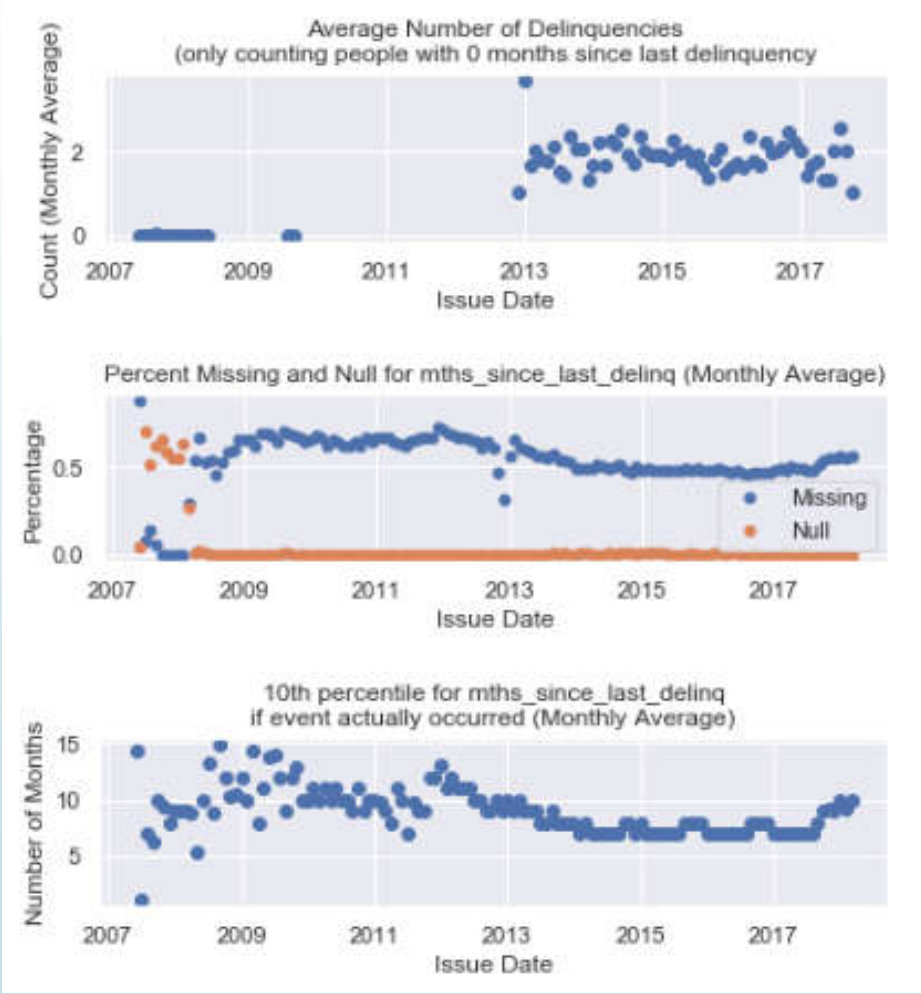
Exploratory Data Analysis 2

- Use box plots to inspect skewness and look for outliers



Exploratory Data Analysis 3:

- Data visualization also helped discover some inconsistent encoding of missing values.



Machine Learning 1

- Classification models used:
  - Logistic regression with elastic net regularization
  - Support vector machines with RBF and polynomial kernels
  - Random forests
  - Gradient boosting(XGBoost)
- These individual models were then combined into an ensemble using a hard voting classifier.

Machine learning 2

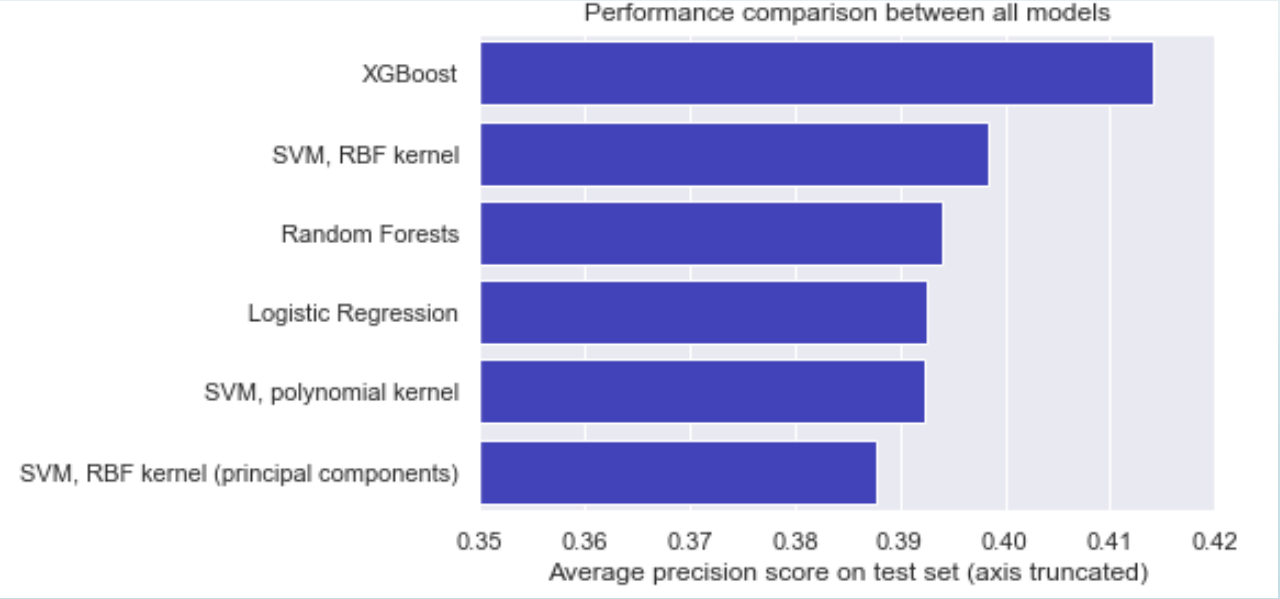
- Challenge: Large sample size, so training took too long.
- Solutions:
  - Estimate efficient version of each model (e.g., use stochastic gradient descent for logistic regression, and XGBoost for gradient boosting).
  - Use Bayesian hyperparameter optimization rather than brute-force grid- or randomized search
  - Single validation set instead of 5-fold cross-validation where necessary
  - Train most computationally expensive models on AWS
  - Try reducing number of features using principal component analysis, if necessary
  - Use only a subset of observations, if necessary

Machine learning 3

- Bayesian hyperparameter optimization
  - Implemented using Hyperopt
  - Learns from the results of past combinations of hyperparameters it tried.
  - This allows an intelligent update of our confidence which combinations of hyperparameters are most promising.
  - As a result, we need less iterations to achieve a given accuracy, making it less costly than a brute-force approach such as grid- or randomized search
- Optimized metric: Average precision on test set (roughly equivalent to area under precision-recall curve)

Results 1: Individual classifiers

- XGBoost performs by far best
- The remaining classifiers are in the same ballpark
- SVM with RBF kernel (on original data) performs next-best. The same model on principal components performs worst.



Results 2: Ensemble

- Use hard voting classifier, which takes a majority vote between the predictions from individual models
- Could not use soft voting classifier because getting probabilities from SVMs would have been too computationally costly
- Weighted XGBoost by a factor of 2, since it performed by far the best (and its predictions were also less correlated with the other models)
- The resulting ensemble performed about as well as XGBoost, but did not quite match it.
- This is not too surprising, given the XGBoost's dominance
- Nevertheless, the ensemble performs better than the weighted average of its component models. This illustrates the power of ensembles. We just would need to add more individual models, or find a better way of weighting them.

Accuracy of best model (XGBoost)

- If we want to identify about half the applicants who will eventually default (recall = 0.49), we have to be willing to accept that only 40% of the people we predict will default will actually do so (precision = 0.4).
- Since the cost of one default is higher than the cost of turning down one borrower who would pay back, this seems like a reasonable trade-off.
- Depending on the precise costs associated with both cases, it may be worth to further adjust this trade-off. Unfortunately, the data did not allow modeling the net present value of each loan instead of a binary measure of default. This would have given us a more precise cut-off to use.

	precision	recall	f1-score	support
0	0.86	0.81	0.83	122083
1	0.40	0.49	0.44	31416
micro avg	0.74	0.74	0.74	153499
macro avg	0.63	0.65	0.64	153499
weighted avg	0.77	0.74	0.75	153499

Recommendations:

- It is possible to cut the number of defaults in half, but this entails making fewer loans.
- Take into account how scaling down the number of loans affects the business, in particular with regard to fixed costs.
- It is possible to further adjust the decision threshold. This allows to gradually reduce the number of loans. (Start by filtering out applicants whose default risk is highest, and then gradually lower this threshold to the desired level.)
- Depending on the precise return from squeezing out an additional bit of accuracy, it may be worth investing into improving the predictive models further.

Future scope:

- Instead of dropping categorical variables with more than 50 categories (because of one-hot encoding), add binary variables for most frequent categories. Use natural language processing to extract meaning from text fields.
- Take the time-series nature into account (temporarily correlated residuals), though there are some challenges.
- Train a deep learning model
- Add other (not too-correlated) models to ensemble
- Use more sophisticated weighting when combining individual classifiers (take both relative performance as well as correlation with other models into account)