Jeromie Clark

# Capstone Report

## Project Overview

Kaggle, a data science competition site, is currently hosting a competition on Mortgage Default Risk[[1]](#footnote-1) targeting profiles of individuals who may little to no credit history or were victims of “untrustworthy lenders”. Instead of traditional markers like credit score and payment history, the company uses “alternative data, like telco and transactional information” to determine their customers’ creditworthiness.

The contest sponsor has provided a comprehensive, anonymized dataset that includes any previous credit bureau history and open loan balances, details on any outstanding commercial credit and cash loans, credit card balances, previous applications for loans from the same individual, and a history of all payments made on the loans issued. The application dataset is comprised of several hundred features, and hundreds of thousands of records, comprised of a mix of continuous, binary and categorical data.

## Problem Statement

Using the dataset provided, which includes both metadata about each borrower as well as the payment history of each loan, the goal is to successfully create a model that provides an accurate prediction of whether or not a borrower will repay a given loan.

Kaggle provides a labeled application\_train dataset, and a separate, unlabeled application\_test dataset for submission to the contest. For the purposes of this paper, only the application\_train dataset is used, but the preprocessing steps have been built with consideration for processing both the application\_train and application\_test datasets, in order to participate in the actual Kaggle competition in the future. The application\_train dataset is sufficiently large to build robust test and train sets without relying on the Kaggle backend for validation.

## Metrics

Our contest sponsor cares primarily about avoiding borrowers that are likely to have performance problems in the future. To that end, we are using accuracy as the primary performance metric for this project. Accuracy is defined as the ratio between the number of correct predictions to the total number of predictions.

As a secondary concern, erroneously identifying borrowers as potentially problematic is detrimental to our lender’s business, as it unnecessarily reduces their pool of potential customers. The fewer loans our lender makes, the less opportunity they have to profit from issuing a loan.

Therefore, in addition to Accuracy, we must consider both Precision (the ratio of borrowers flagged as problematic that were actually problematic), and Recall (the ratio of borrowers flagged as problematic to all the borrowers that were actually problematic).

## Benchmarks

Because only approximate 8% of borrowers have payment problems, the dataset is skewed in terms of the distribution of problematic and successful borrowers. It is possible to classify approximately 92% of our loans as not problematic, achieving a high (92%) accuracy rate, while in actuality, failing to detect any problematic loans. This naïve predictor was built as a benchmark for comparing the performance of various machine learning algorithms.

Because accuracy alone can be a misleading metric, we must also consider algorithmic performance in terms of precision and recall. These metrics are typically combined to create what is known as the F1 score, defined as the harmonic mean of precision and recall scores. More specifically, we use the F-beta score with a beta value of 0.5 to emphasize precision. The result ranges from 0 to 1, with one being the best possible score.

Our naïve predictor achieves an accuracy score of 0.9190 (~92%), and an F1-score of 0.9341. Our goal for success would be to exceed this benchmark, representing the lender’s current well-tuned ability to make loan decisions.

## Data Exploration

The dataset provided for this contest is broken into multiple files. The primary source of information is application\_train.csv, which contains both the bulk of the information about the applicant and metadata about the loan application itself.

We begin our exploration and model selection using this dataset, and iterate, performing dimensionality reduction (e.g. pruning features that don’t send strong signals for prediction) as we augment the dataset with additional features, examining performance along the way to understand the impact of those new data points.

Much of the data outside the application\_train.csv file requires some feature engineering, as there are one-to-many relationships between the applicant and data points like individual credit card accounts, open balances, previous defaults, etc. This approach attempts to balance computational feasibility (through dimensionality reduction)

The application\_train dataset itself is comprised of 122 individual features. Each application has a unique ID, which we can use as a key to join in data from the other data files. For basic analysis and sanity, I bucketed each feature into one of four categories and wrote a utility function to pull basic statistics about the data sufficient to validate basic sanity and identify any necessary preprocessing steps.

The initial dataset had 61 non-numeric (categorical) features, defined as features comprised of two or more possible strings, which reflect a particular categorization. As an example, the CODE\_GENDER feature (Figure 1) was comprised of ‘F’, M’, and XNA’ values. These representations were analyzed for sanity, and then tracked for One-Hot Encoding during the preprocessing phase.

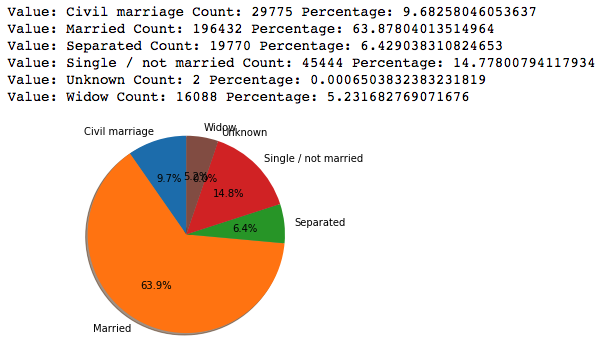


Figure - NAME\_FAMILY\_STATUS Feature

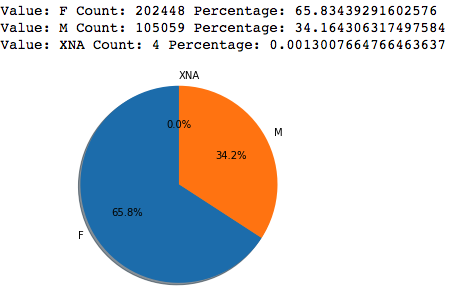


Figure 2 - COUNT\_GENDER Feature

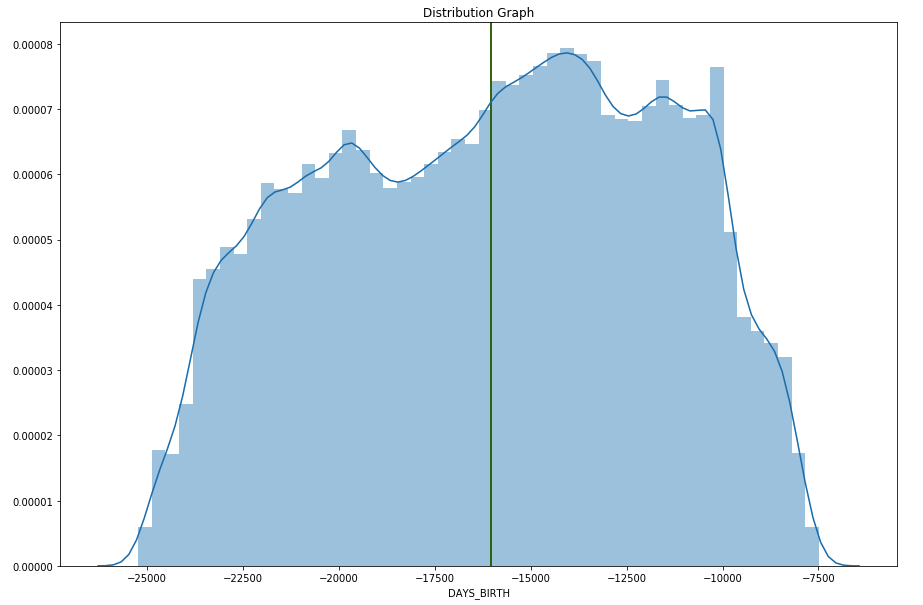
In addition, there was a subset of 6 categorical features which were actually binary in nature (Figure 3), typically characterized by “Yes/No” or “Y/N” fields. We classified those features as “String to Bool”, and ultimately converted those fields to binary (1|0) representations. The meaning is identical, but the binary representation is more effectively processed by standard machine learning algorithms.



Figure - Conversion of Y/N to Boolean Values

Machine learning algorithms tend to perform best under conditions when continuous fields represent a Gaussian distribution.

This graph shows the normal distribution of applicant’s ages, represented as a negative integer, relative to the application date. The data is distributed along a reasonable curve and is well-centered.



In some features; however, we see heavily skewed datasets. This is relatively common, particularly when working with financial data, where you tend to see high counts near zero, and low counts of large values at the opposite end of the spectrum.

In the example below, we’re looking at the applicant’s total income. Our range of applicant incomes is huge, with a minimum value of 25650, and a maximum value of 117,000,000. The average and mean values are 168797, with a standard deviation of 237122. As a result, you see that the graph is heavily skewed, with a small bump towards the left end of the range (the vertical bar is the median), and a long tail of continuous values extending out to the right of the range (Figure 4).

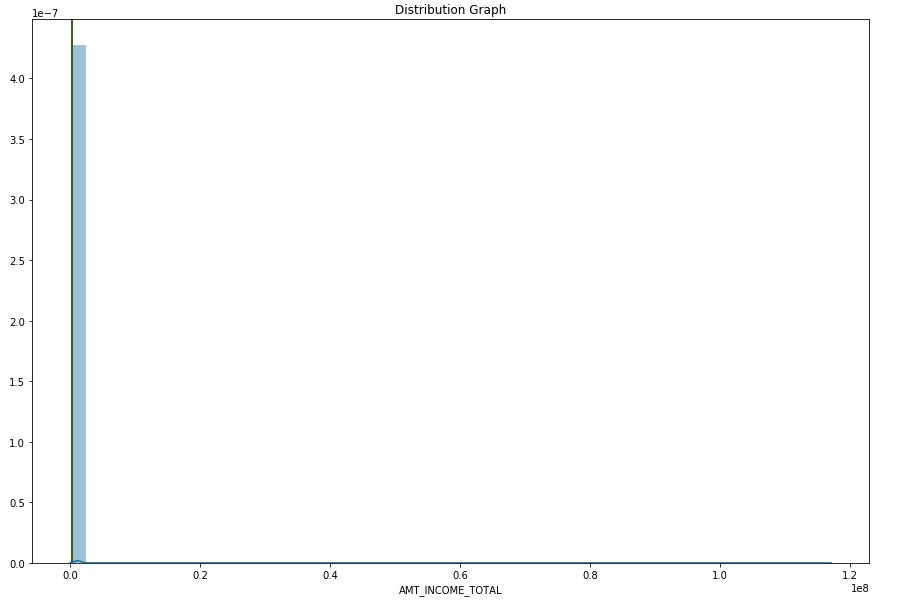


Figure - AMT\_INCOME\_TOTAL Unprocessed

Financial data like this tends to be lognormally distributed, and by taking the log of the values, we can typically restore symmetry to our data. After processing, we now see a symmetrical curve, nicely centered around the median. In addition to the 61 continuous numeric fields, we identify another 11 fields for log transformation during the pre-processing phase.

## 

## Additional Preprocessing

A handful of fields contained problematic individual values that were problematic during processing. Those features were handled independently.

DAYS\_EMPLOYED included values that indicated that some people had worked for 365243 days, or approximately 100 years. This seemed like they were probably a result of a bug, and in order to avoid penalizing the applicants for a software issue, the value was replaced with the mean value of all the values in the set.

A number of normalized housing data fields like “HOUSETYPE\_MODE” contained categorical String values, with the exception of NaN when a value was not specified. This doesn’t work when one-hot encoding categorical values, so the string value “not specified” was substituted.

In order to scale a group of continuous numeric fields to a comparable range, those fields must exclusively contain numerical data. In a number of instances, fields where numeric data was not supplied were populated with NaN (Not A Number) values. To facilitate processing, those values were converted to 0 using the NumPy library’s nan\_to\_num() function, which converts NaN to 0, -inf to MININT, and inf to MAXINT.

When machine learning algorithms evaluate numeric features, they may apply weight to larger values. In order to ensure that the algorithm used applies equal weight to values of independent features, we put them all in a relative scale from 0 to 1. This scaling approach maintains the shape of the value distributions in each feature, while solving the problem of artificially weighting various features.

Because some fields may have contained outliers, the scikit learn RobustScaler class was initially considered to mitigate the effect of any outliers in the set; however, when we evaluated the performance of our candidate algorithms between data preprocessed with RobustScaler and StandardScaler, StandardScaler performed slightly better. We also compared data with and without log transformations, but found that the log-transformed data performed marginally better.

# Modeling

Using the scikit-learn flowchart[[2]](#footnote-2) for choosing an algorithm as a guide, we ultimately selected three algorithms to compare based on the characteristics of our data.

Our dataset is based on historical loan data, and is already labeled by whether or not the application payed on time. meaning that there’s already a feature that definitively tells us whether or not our applicants have defaulted on loans. Because we’re working with a labeled dataset, we’re simply looking for a Classification algorithm that can predict the binary state of whether or not a loan will be problematic.

Our dataset is large and feature-rich, with hundreds of features and hundreds of thousands of records. This rules out Support Vector Curves and Naïve Bayes algorithms, both of which are too computationally intensive to be practical with datasets of this size.

The candidates that we selected for evaluation were LinearRegression, which is actually a classification algorithm, as well as the RandomForest and AdaBoost ensemble methods.

Linear Regression models the response between a dependent variable and one or more explanatory variables by attempting to derive a line that best separates the two classes of data (loans with problems and loans without problems). The algorithm does this by choosing a random line, measuring the distance between points in the dataset to that line as “error”, then moving the line to minimize the total error against all points. Error is generally calculated as Ordinary Least Squares, or the sum of the squared error value.

Random Forests and AdaBoost are an ensemble methods, which use multiple instances of estimators to produce meta-results that are more accurate than individual instances of the underlying estimators themselves. Random Forests uses a number of Decision Tree classifiers on subsets of the training data to improve the predictive accuracy of the algorithm and control the tendency of a single Decision Tree to overfit the training data[[3]](#footnote-3).

AdaBoost takes a slightly different approach, fitting to the original dataset, then focusing in on misclassified data points with each iteration, in an effort to focus in on the more difficult cases to classify.

In practice, our evaluation showed that LinearRegression fails miserably in this particular application failing to identify any records as potentially problematic. Random Forests and AdaBoosts performed comparably. While Random Forests is about twice as fast as AdaBoost in training and prediction tasks, AdaBoost performs slightly better in terms of F1 score.

## Model Tuning

## Solution

Explore ensemble learning techniques with the goal of producing a model that provides accurate predictions based on our dataset.

### Benchmarking

Since this is a supervised learning problem, there is a clear benchmark by which we can compare our performance. In addition, this is an active Kaggle competition, and we can compare the score of this kernel with other proposed kernels in the competition.

### Project Design

#### Data Exploration

* Import the project data and get a feel for what’s there by creating labeled tables.
* Learn what fields are typically populated vs. what fields are sparse
* Look at what values fields are populated with and formulate plans for re-encoding and normalizing values as necessary for optimal analysis
* Look at the distribution of values across fields to understand their usefulness as signals
* Calculate some basic statistics about the dataset to answer fundamental questions

#### Data Preparation

* Identify interesting features
* Understand and normalize feature encoding and distribution
* As needed, one-hot encode non-numeric tables

#### Prepare Training and Testing Data

In this project, it looks like the sponsor has already created training, testing and validation sets; however, I will create them as required using best practices.

#### Select and Evaluate Candidate Models

Logistic Regression and AdaBoost will be interesting to look at for this project, but I’ll probably spend some time to more fully explore the options available in scikit-learn and look for papers in the field that might provide better ideas.

I’ll create some prototype models and compare candidates, and select one for further tuning based on my observations.

#### Model Tuning

I will refer to available literature to identify sensible starting ranges for various hyperparameters in the models used, and employ automated processes like GridSearch for evaluating and tuning those hyperparameters where that makes sense.

#### Feature Selection

I will calculate the importance of various features in the dataset and select a subset sufficient to produce high-quality results, while optimizing for feasible processing times.

## Project Resources

A full description of the Kaggle competition, along with the test dataset can be found at:   
https://www.kaggle.com/c/home-credit-default-risk

1. https://www.kaggle.com/kailex/tidy-xgb-0-777/data [↑](#footnote-ref-1)
2. http://scikit-learn.org/stable/tutorial/machine\_learning\_map/index.html [↑](#footnote-ref-2)
3. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [↑](#footnote-ref-3)