Capstone Project

Christopher Jose

Machine Learning Engineer Nanodegree

Data Science and Mortgage Loan Defaults

Project Overview

Between 2007-2009, Fannie Mae and Freddie Mac, two government-sponsored enterprises (GSEs) that bundle up mortgages and offer them as securities to investors, went bankrupt from a surge of mortgage defaults (Mamonov and Benbunan-Fich, 2017). Predicting whether a loan will go into default, mortgage loan characteristics that are predictive of whether a loan will default, and the qualitative and quantitative nature of the relationship between these characteristics and defaulting, is thus of importance to the mortgage industry. Predicting which loans are more likely to default is important to the profit of every player in the industry, such as mortgage lenders, insurers, and GSEs.

I consider a loan as delinquent/defaulting if it has missed at least one mortgage payment and has not yet paid it back. Thus, if a loan misses the prior month's payment, but makes the current month's payment, then it is still delinquent for 1 month, since it has not yet made the missed payment.

All analysis was conducted using Python.

Problem Statement

This capstone project will predict whether a loan encounters delinquency for at least 3 consecutive months based on its loan characteristics, determine which loan characteristics are most predictive of this, and the qualitative and quantitative nature of the relationship between these characteristics and the response. I am predicting delinquency for at least 3 consecutive months because this situation indicates a much greater level of distress on the mortgage borrower than defaulting for just a couple months, and represents a much greater risk of foreclosure and an insurance claim (Sun, 2013). This is a binary classification problem in that the outcome variable has two levels: the loan encounters this state (y=1) or it does not during its life (y=0).

Metrics

The AUC and F1 Score metrics will be evaluated to assess model predictability. AUC (Area Under the Curve) is the area under the Receiver Operator Characteristic Curve (ROC Curve). It measures the model's ability to correctly classify Y=1 and Y=0 across all potential thresholds in calculating predicted probabilities. It can be interpreted as the probability that the model will rank a randomly chosen positive observation higher than a randomly chosen negative observation (rank, meaning predictive probability), regardless of the selected threshold. AUC is

a good metric for classification problems because it is not dependent upon the selected threshold, and so it provides a high-level view of the classifier's performance.

F1 score is a weighted average of the recall and precision metrics, and so is useful as providing an assessment of both metrics at the same time. Additional metrics will be computed, but not used in the model selection process. Precision is the percentage of predicted positives that are true positives (positive indicates y=1, meaning 3+ months delinquent). Recall is the percentage of positives that are predicted as positives. A confusion matrix is a table that shows for each actual class value (0,1), how many correct and incorrect predictions were made.

Precision	=	TP / (TP + FP)
Recall	=	TP / (TP + FN)
F1 Score	=	2 * (precision x recall) / (precision + recall)

If model results are used to accept or deny a mortgage loan application, low precision would result in more denied loans that aren't actually likely of defaulting, which would reduce business and profit for mortgage lenders/insurers. Similarly, low recall would result in not predicting more defaults enough defaults, which would also result in less profits for mortgage lenders/insurers.

If precision is high, but recall is low, then the classifier does not predict many of the 3+ month delinquencies, but of predicted delinquencies, most are delinquent. Conversely, if recall is high, but precision is low, then the classifier predicts many of the delinquencies, but many actual non-delinquencies are predicted as delinquencies.

Accuracy has not been selected as a model selection metric because the problem is an imbalanced classification problem. Accuracy would be deceiving in that a high accuracy can still occur even if the model classifies every observation as the majority class (the class we are not interested in, in this case).

The Dataset

The data consists of a portion of Fannie Mae's public Single-Family Historical Loan Performance and Acquisition Datasets: *Acquisition_2016Q1.txt* and *Performance_2016Q1.txt3*. These datasets contain a subset of Fannie Mae's thirty year, single-family, conventional fixed-rate mortgage loans. It consists of 303,403 total loans that originated between 02/2014 – 03/2016.

For the sake of reducing computation time, this exercise has been conducted using a semi-random subset of 5000 observations from this dataset. All 3+ month defaulted loans have been included, as well as all observations containing missing credit score and debt-to-income values.

Data can be accessed at
http://www.fanniemae.com/portal/funding-the-market/data/loan-performance-data.html

The acquisition file provides loan characteristics (the predictors) and has one record per loan. I took independent variables from this file.

The outcome variable was accessed in the performance file. Its value has been set to True if the *Current Loan Delinquency Status* field (called *Delq.Status* in the performance file) is such that:

If *Delq.Status* in (3,4,5,6,7,8,9,10,11,12), then True, else False

In other words, the value is True if at some point the loan was delinquent/defaulting for at least 3 months in a row (*Delq.Status* = 3, meaning 90-110 days have gone by without having paid off all missed mortgage payments). If a value of 3 is reached, the defaulter has had at least 1 missed payment for 3 consecutive months. They could have missed the first month's payment and made one payment in each of the 2nd and 3rd months, but are still behind on one payment in all 3 months (the first month's missed payment). Or the borrower could have missed the first month's payment, then did not make a payment in the second month, and then made two payments in month three. In this case, the loan was still delinquent for 3 consecutive months because there was at least one payment that had not been repaid in each of the months.

The Performance file shows information for each loan for each month after and including its origination month, so each loan has more than one record. Thus, a given loan will have a record for each month that occurs after its origination month and until it's cancellation date.

For each record, there is a *Delq.Status* value that represents the number of days the loan is delinquent for. The *Delq.Status* values for this analysis range from [0,13]. The fact that the maximum isn't much higher is plausible and makes sense, given that the data only includes originated loans and their activity between 2014 – Q1 2016, so we're not going to see delinquency values over the number of months in this range (27 months). A value of 0 means the loan is not delinquent or less than a month delinquent. See below.

Delq.status	Days Delinquent	Description
0	0-30	Behind on at least one payment for under a month or current (= not delinquent)
1	30-59	Behind on at least one payment for 1 month and change
2	60-89	Behind on at least one payment for 2 months and change
3	90-110	Behind on at least one payment for 3 months and change
X	Unknown	This information is unavailable.

Predictors and their field names are shown below. *Origination Season* is constructed from a field in the Acquisitions file that is called *Origination Date*, and will have values ('Fall', 'Winter', 'Spring', 'Summer'), which considers seasonality.

Predictors

Predictors considered for this model include the following:

Index	Name	Description	Type
1	ORIG_CHN	CHANNEL	Categorical
2	ORIG_RT	ORIGINAL INTEREST RATE	Continuous
3	ORIG_AMT	ORIGINAL UNPAID PRINCIPAL BALANCE (UPB)	Continuous
4	ORIG_SE	ORIGINATION SEASON	Categorical
5	OLTV	ORIGINAL LOAN-TO-VALUE (LTV)	Continuous
6	PURPOSE	LOAN PURPOSE	Categorical
7	NUM_BO	NUMBER OF BORROWERS	Discrete
8	DTI	DEBT-TO-INCOME RATIO (DTI)	Continuous
9	CSCORE_B	BORROWER CREDIT SCORE	Continuous
10	FTHB_FLG	FIRST-TIME HOME BUYER INDICATOR	Categorical
11	PROP_TYP	PROPERTY TYPE	Categorical
12	NUM_UNIT	NUMBER OF UNITS	Discrete
13	OCC_STAT	OCCUPANCY STATUS	Categorical
14	STATE	PROPERTY STATE	Categorical
15	MI_PCT	MORTGAGE INSURANCE PERCENTAGE	Continuous
16	CSCORE_C	CO-BORROWER CREDIT SCORE	Continuous
17	MI_TYPE	MORTGAGE INSURANCE TYPE	Categorical
18	RELOCATION_FLG	RELOCATION MORTGAGE INDICATOR	Categorical

Benchmark Model

I will run a logistic regression with its default hyperparameters to serve as a benchmark in comparing the final model, which will have undergone hyperparameter tuning.

Additionally, in the paper, *What Can We Learn from Past Mistakes? Lessons from Data Mining the Fannie Mae Mortgage Portfolio* (the URL link is in section *References*), predictive models are also constructed to predict loans in the Fannie Mae data that will reach 3 months of delinquency. However, a broader set of the same Fannie Mae data is examined. The modelers include loan originations between 2000-2014, around 20 gigabytes of data. In addition to the benchmark model, I will treat their results as a comparison to assess the validity of my results.

The modelers perform exploratory data analysis, build predictive models (Logistic Regression, SVM, and more), use 10-fold cross validation to produce classification metrics (each metric value is an average of that metric from k-built models), and evaluate feature importance using a feature permutation-based method. I can directly compare my classification metrics and feature importance with those of the study to assess the validity of my model results. The study optimized recall and calculated precision, recall, and accuracy. I can compare my models to the study through these metrics. However, I should see some differences because I'm working with just a small subset of newer Fannie Mae data (2015-2016 originations) that isn't included in the modeler's data (only includes up to 2014 originations).

When referencing this study later in the report, I will refer to it as the Mamonov study.

Exploratory Data Analysis

Regarding the distribution of the outcome variable, only 649 out of the 303,403 loans defaulted for at least 3 consecutive months (the number of y=1 observations). Thus, this is an imbalanced classification problem, since the y=1 class of the dependent has far less observations than the y=0 class. After subsetting the data to 5,000 observations, 12.98% of the observations are defaults, so it's still an imbalanced classification problem.

Numeric predictor correlations are shown below. Most numeric predictors do not appear to be especially correlated with one another, except for CSCORE_B and ORIG_RT, which show a moderately negative correlation of -.423. As credit score rises, the origination rate (interest rate on the mortgage loan) falls, which makes sense.

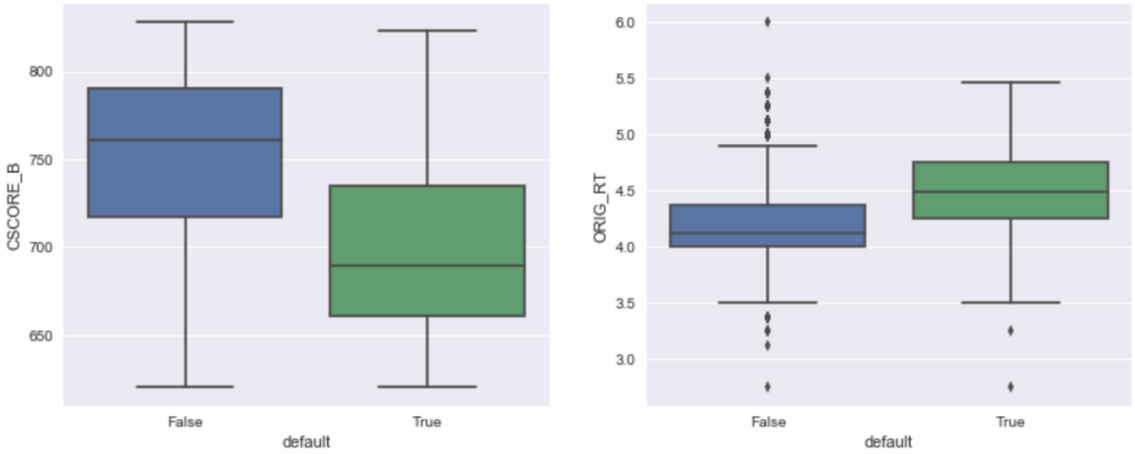
	ORIG_RT	ORIG_AMT	ORIG_TRM	OLTV	NUM_BO	DTI	CSCORE_B	NUM_UNIT
ORIG_RT	1.000	-0.143	-0.008	0.104	-0.107	0.156	-0.423	0.143
ORIG_AMT	-0.143	1.000	-0.013	0.015	0.156	0.074	0.056	0.137
ORIG_TRM	-0.008	-0.013	1.000	0.000	-0.009	0.025	-0.016	0.007
OLTV	0.104	0.015	0.000	1.000	-0.025	0.072	-0.091	-0.101
NUM_BO	-0.107	0.156	-0.009	-0.025	1.000	-0.093	0.146	-0.036
DTI	0.156	0.074	0.025	0.072	-0.093	1.000	-0.195	0.002
CSCORE_B	-0.423	0.056	-0.016	-0.091	0.146	-0.195	1.000	0.022
NUM_UNIT	0.143	0.137	0.007	-0.101	-0.036	0.002	0.022	1.000
default	0.222	-0.041	0.009	0.100	-0.134	0.130	-0.355	-0.033

For each predictor, I inspected above whether there is a sizable difference between its mean value for each value of the outcome variable, meaning for loans that eventually default for 3 consecutive months or more and loans that do not.

Most predictors appear to have values that are close, regardless of the outcome's level. Origination rate (ORIG_RT), Origination Loan-to-Value (OLTV), and Borrower Credit Score (CSCORE_B), might have sizable enough differences to offer value as predictors. The average credit score for defaulters, 699, is typically regarded as indicating "good credit", while the average credit score for non-defaulters, is typically regarded as indicating "excellent credit" (O'Shea, B.). This apparent difference in credit score quality between defaulters and non-defaulters is indicative of a predictive relationship between credit score and defaulting.

	ORIG_RT	ORIG_AMT	ORIG_TRM	OLTV	NUM_BO	DTI	CSCORE_B	NUM_UNIT
default								
False	4.210	239472.305	359.928	76.906	1.471	34.172	751.262	1.043
True	4.441	224808.937	359.963	81.441	1.267	37.496	698.847	1.015

Inspecting boxplots for CSCORE_B and ORIG_RT, grouped by defaults and non-defaults, shows a clearer relationship between these predictors and the outcome variable.



For each categorical predictor, I inspected whether there is a sizable difference between each level's proportion of observations that defaulted. If a certain level has many defaults compared with other levels, then when the predictor has as a value that level, then it is more likely that the outcome will have value 'default'.

ORIG_SE and RELOCATION_FLG, appear to offer the biggest differences between the proportion of defaults among their levels. For predictor ORIG_SE, loans originated during the Autumn saw 16.5% of their loans as eventually defaulting, compared to loans that originated in Spring, which only saw 8% of their loans defaulting (see below). Each of these levels had a sizable number of originated loans, thus giving greater credibility to these default percentages. For predictor RELOCATION_FLG, which represents mortgages that companies make available to relocating employees, only 1.3% of these mortgages default, compared with all other mortgages (non-relocation mortgages), which default at a rate of 13%.

ORIG_SE	Autumn	Spring	Summer	Winter
default				
False	496	372	13	3470
True	98	36	6	509

```
Autumn 0.165
Spring 0.0882
Summer 0.3158
Winter 0.1279
```

RELOCATION_FLG	N	Y
default		
False	4275	76
True	648	1

```
N 0.1316
Y 0.013
```

Sanity Check: Do predictor min/max values look reasonable?

The *Fannie Mae Single-Family Loan Performance Data Glossary* provides variable information, including ranges. I look at the data's actual ranges to make sure values are not outside the ranges shown on the glossary.

1. ORIG_RT appears to have reasonable min/max values (2.75,6).
2. ORIG_AMT does appears to have reasonable min/max values. For instance, the Mamonov study found zero values for ORIG_AMT, which makes no sense, and so these records were removed.
3. ORIG_TRM's values are within range.
4. OLTV looks okay. Fannie Mae has stated in the loans whose LTVs are above 97% are excluded so that all data is consistent with current underwriting guidelines.
5. NUM_BO can take a value between [1,10], according to the glossary, and min/max values here are [1,6], which is therefore reasonable.
6. DTI can be 1-64% according to the glossary, so this looks reasonable.
7. CSCORE_B looks okay, since values can range from 300-850, and are blank if they fall outside this range.
8. NUM_UNIT can range from 1-4, so the data looks reasonable.

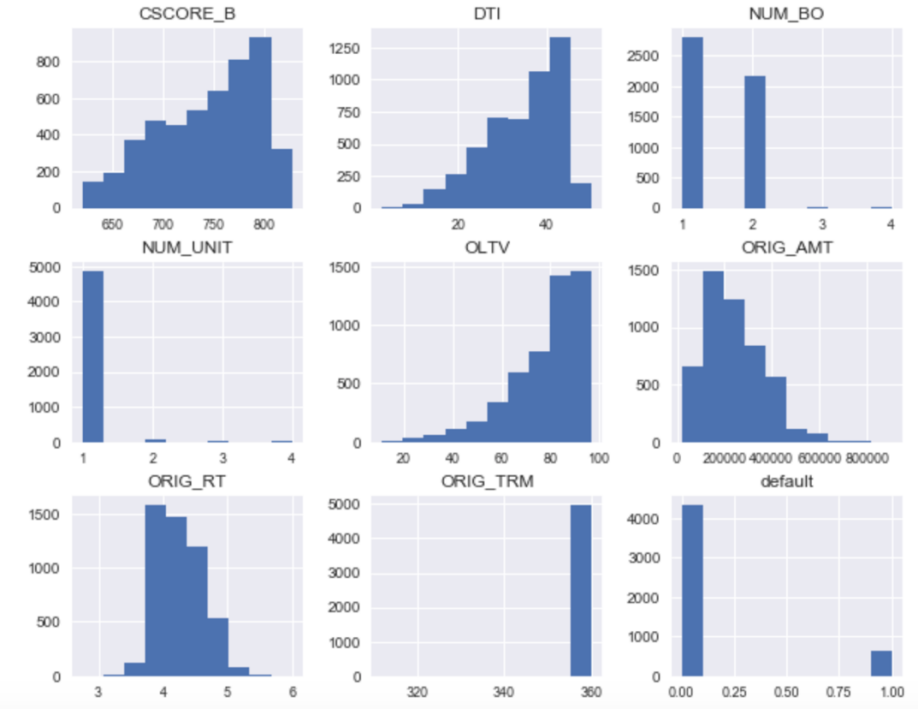
	ORIG_RT	ORIG_AMT	ORIG_TRM	OLTV	NUM_BO	DTI	CSCORE_B	NUM_UNIT
min	2.750	20000.000	312.000	11.000	1.000	3.000	620.000	1.000
max	6.000	903000.000	360.000	97.000	4.000	50.000	828.000	4.000

Missing Values Check

Within the final dataset of 5,000 observations, two predictors have missing values: DTI and CSCORE_B (72, 114 missing values, respectively). In consulting Fannie Mae's glossary file, DTI is missing if it's 0, >65, or unknown. CSCORE_B is missing if it's <300, >850, or unknown. Since low, high, and unknown values are included in both variables, and since both variables are correlated with the response, these missing values can be indicative of a greater likelihood of both defaulting and non-defaulting loans. Throwing away these 186 observations is unnecessary information loss, and given that these observations could potentially contain both high and low values, the middle of these values will be approximated using each variable's median, and this median will be used as part of median imputation for filling in the missing values.

Checking Predictor Distributions

Numeric predictor histograms show that CSCORE_B, DTI, OLTV and ORIG_AMT have skewed distributions. CSCORE_B and ORIG_AMT also have much wider ranges than the other predictors. Most ORIG_TRM's values are 360.



Data Wrangling

A seasonality field with values, 'Winter', 'Spring', 'Summer', 'Autumn', was created to represent the season in which the loan was originated. This field captures any potential relationship between the season of the origination date of the loan and whether it eventually defaults for 3 consecutive months.

A region field was created to capture the region of the state that the loan was originated in, and has values, 'Northeast', 'South', 'Midwest', 'West'.

3 variables, 'MI_PCT', 'CSCORE_C', 'MI_TYPE', were originally considered, but were removed since most of their values were missing. 'ORIG_TRM' was removed because of having low variance (see its above histogram). Most loans have annual terms.

Numeric predictors were log-transformed to reduce the effect of skewness and outliers. Most distributions remained skewed but were more constrained after the transformation.

Categorical predictors were converted to dummies, with one level removed to avoid the dummy-variable trap.

Preprocessing Steps

Classifiers were trained and tested using 3-fold nested stratified cross-validation. A randomized search using 3-fold stratified cross validation was conducted within each of the 3 inner train folds to optimize hyperparameters and the number of features used in the models. A pipeline with the following preprocessing steps, in the following order, was applied to the data within the inner folds of the nested cross validation (in the randomized search) and then to each outer fold in training the model based on the resultant optimized hyperparameters. This was done to ensure no data leakage of the preprocessing methods into any test set data.

1. Median Imputation
2. Feature Scaling
3. Feature Selection
4. SMOTE method

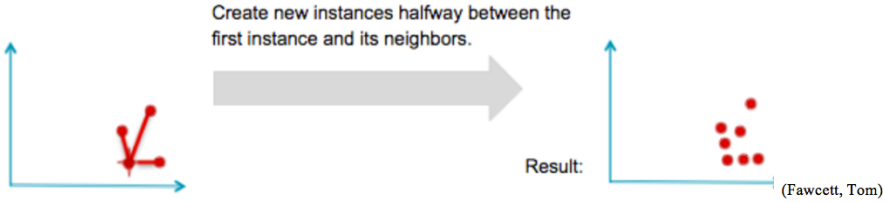
1. Median imputation was applied to the two numeric predictors with missing values (CSCORE_B, ORIG_AMT), since missing values for these variables can either indicate very low, high, or unknown values, as mentioned in the *Missing Values Check* section, and there would be wasteful information loss in throwing away these observations.

2. Feature scaling was applied using the following formula to ensure that features would be treated equally within the classifiers:

$$xnew = (x - xmin) / (xmax - xmin)$$

3. Feature selection was applied to reduce the potential for over-fitting by removing features introducing noise into the model. By using sklearn's meta-transformer, *SelectFromModel*, a randomized search selected the optimum variable importance threshold in the array [1e-5,1e-4,1e-3]. All variables with thresholds higher than the selected threshold from a Random Forest classifier fitted to the data (preprocessed via median imputation and feature scaling) were included in the models. Higher thresholds were tested, but resulted in no variables being selected, so these were removed from consideration.

4. SMOTE was applied (Synthetic Minority Oversampling technique) to resample the minority class (defaults) to balance defaults and non-defaults at a 50/50 ratio. SMOTE finds the 5-nearest neighboring minority observations and for each neighbor, creates a new observation having the minority class, and whose predictor values are such that the point is halfway between the two points in the feature space.



(Fawcett, Tom)

Addressing the Class Imbalance

Since the dataset is imbalanced, models will be more prone to correctly predicting the majority class's value (non-default), at the expense of defaulting observations, thus resulting in high accuracy, but low values for other statistics, like precision. To address this class imbalance, accuracy is not optimized for in the randomized search, but rather, AUC and F1 score, stratified cross validation is used, and SMOTE is used. Stratified cross validation causes roughly the same percentage of both outcome labels in the train and test folds. SMOTE makes new observations with minority class values.

Model Building

Given that this is a classification problem, 8 classifiers were considered using the sklearn library: Decision Trees, Logistic Regression, K Nearest Neighbors, Random Forests, Support Vector Machines, AdaBoost, Gradient Boosting Trees, and Naïve Bayes.

Decision Trees take an observation and examine one of its feature's values. The tree asks a yes/no question about the feature's value, and depending on the answer, then another feature's value is examined via a yes/no question. This process continues until a label (Y=1, Y=0) is assigned to the observation.

Logistic Regression maximizes the log likelihood of the observed data and outputs a linear predictor ($b1x1 + b2x2 + \ldots$) that can be interpreted as the log-odds of Y. To get P(Y=1), then, exponentiate the log odds to get the odds, and then P(Y=1) = odds / (1+odds). These formulas / interpretations assume the predictor data has not been transformed (log transformed or scaled). Unlike linear regression, logistic regression assumes that Y has a binomial distribution instead of a normal distribution.

K Nearest Neighbors predicts the class of new observations as the class belonging to the majority of the K nearest observations (or neighbors). K must be specified. The nearest observations are determined by a specified distance metric.

Random Forests constructs a multitude of decision trees. Each tree is different because of having been created from two randomization processes: tree bagging and feature bagging. Tree bagging involves randomly sampling the training set to build each tree. Feature bagging involves randomly sampling a subset of features to build each split. Each tree spits out a prediction and the final prediction is based on the predictions from all the trees.

Support Vector Machines finds a line called a hyperplane that divides up observations, such that observations that fall on one side of the line are predicted as Y=1, and those that fall on the other side are predicted as Y=0. The hyperplane is placed to maximize the distance to the nearest point of each class.

Gradient Boosting Trees starts with a classifier tree that predicts y and sequentially builds new classifier trees off each prior classifier tree, which reduce the train error in each prior classifier tree.

AdaBoost takes a model and enhances it ("boost"s it) by building a series of alterations of the model off the data. Each subsequent time the model is constructed, it is altered to focus more on the data that it predicted incorrectly the previous time it was built (the model is "ada"pted to the data). AdaBoost then averages the predictions of all models and considers models that gave more accurate predictions.

Naïve Bayes takes advantage of Bayes Theorem and by assuming that features are independent. It predicts an observation as having class Y=1 or Y=0 via the Bayes Theorem formula: P(Y=1 | data) = P(data | Y=1) x P(Y=1) / P(data), where data is the observed feature values for the observation. P(Y=1 | data) is the posterior probability and the label with the highest posterior probability, either Y=1 or Y=0, is the selected label for the observation.

Parameters for each classifier were specified as follows:

```
classifiers = [
    DecisionTreeClassifier(random_state=0),
    LogisticRegression(random_state=0),
    KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2,
                         metric='minkowski', metric_params=None, n_jobs=1),
    RandomForestClassifier(warm_start=True, oob_score=True,random_state=0,n_estimators=300),
    svm.SVC(random_state=0,probability=True),
    AdaBoostClassifier(base_estimator=base, n_estimators = 100, random_state=0) ,
    GradientBoostingClassifier(random_state=0, max_features='sqrt'),
    GaussianNB()]
```

Default parameters were used for all methods, except for those tuned in the randomized search, and except for the following methods:

1. For Random Forests, 300 trees were specified.
2. AdaBoost builds 100 decision tree stumps (max depth of 1).
3. For Gradient Boosting, the maximum features considered at each tree split was specified as the square root of the number of features, which is considered as a rule of thumb to reduce the possibility of over-fitting (Jain, A.)

100-500 trees, at least in the case of gradient boosting classifiers, is generally considered as enough trees, so I use at least 100 trees in Random Forests, Adaboost and Gradient Boosting. Gradient Boosting uses 100 tree by default (Friedman, Jerome).

Randomized search is conducted on the pipeline, which differs from Grid Search in that only a fixed number of parameter settings are tried out. For each classifier, a pipeline is fitted and fits the number of features to select and certain hyperparameters for the classifier. n_iter is a hyperparameter in the randomized search algorithm that controls the number of parameter settings to consider. Since the default value for n_iter is 10 (a hyperparameter of the randomized search), and certain methods (Naïve Bayes) have less than 10 hyperparameter settings, to prevent an error, n_iter's value was set to the number of hyperparameters in the classifier's pipeline (# of feature selection parameters (=3) + # of classifier hyperparameters).

```
pipeline = Pipeline([('median_imputer',median_imputer),('scaler', scaler),
                     ('fselect',feature_selection), ('smote', smote),
                     ('clf', classifiers[k])])
```

Feature selection threshold values that were considered in the randomized search are:
.00001, .0001, .001.

Hyperparameters specific to each method are shown below. *Max_depth* controls the size of the decision tree and the greater the size, the greater the complexity, and the more prone the model is to over-fitting (thus less generalization abilities). C controls the amount of regularization for both Logistic Regression and SVM, where the smaller the value, the greater the penalization on model complexity, which helps reduce over-fitting and increases generalization. Learning Rate controls the impact of each tree on the model and how quickly the model fits to the data. Using a lower rate can prevent over-fitting. *Max Features* sets the size of the random subset of features generated at each split, and using a lower value helps prevent over-fitting.

Method	**Hyperparameter**	**Values**
Decision Tree	Max Depth	2,3,4,5
Logistic Regression	C	.0001, .01, 1, 100, 10000
K Nearest Neighbors	K	3,4,5,6,7
Random Forests	Max Features	sqrt,.2,.5,.8
Support Vector Machines	C, Gamma	C: [.001, .01, .1, 1, 10], Gamma: [.001, .01, .1, 1]
AdaBoost	Learning Rate	.01,.1,1,2
Gradient Boosting	Learning Rate, Max Depth	Learning Rate [.001,.01,.1], Max Depth:[3,4,5]
Naïve Bayes	None	None

Model Evaluation and Validation

An initial model evaluation was conducted using the pipeline, but excluding SMOTE and hyperparameter tuning. Precision and accuracy prove be higher than in using the refined models, but F1 Score, Recall, and AUC are lower.

In optimizing for F1 score, Gradient Boosting has the highest AUC and F1 Score. In optimizing for AUC, Gradient Boosting also has the highest F1 score and close to the highest AUC. Classifiers in the tables below are ranked from top to bottom by F1 Score in descending order. Recall improves substantially in the refined models.

Without using SMOTE and the randomized search to tune hyperparameters, the following results were generated:

	Method	AUC	Accuracy	F1 Score	Precision	Recall
7	Naive Bayes	0.651	0.857	0.403	0.441	0.373
2	K Nearest Neighbors	0.595	0.862	0.306	0.441	0.236
0	Decision Tree	0.600	0.810	0.302	0.289	0.316
5	AdaBoost	0.584	0.872	0.283	0.522	0.196
3	Random Forests	0.584	0.868	0.281	0.489	0.200
6	Gradient Boosting Classifier	0.575	0.869	0.260	0.504	0.177
1	Logistic Regression	0.546	0.871	0.175	0.536	0.106
4	Support Vector Machines	0.500	0.870	0.000	0.000	0.000

In optimizing the randomized search for F1 Score, the following results were generated:

	Method	AUC	Accuracy	F1 Score	Precision	Recall
6	Gradient Boosting Classifier	0.730	0.777	0.437	0.326	0.667
5	AdaBoost	0.706	0.796	0.426	0.337	0.584
4	Support Vector Machines	0.730	0.740	0.417	0.294	0.718
1	Logistic Regression	0.727	0.743	0.416	0.295	0.706
0	Decision Tree	0.720	0.741	0.412	0.296	0.692
3	Random Forests	0.646	0.840	0.384	0.386	0.384
2	K Nearest Neighbors	0.610	0.711	0.297	0.218	0.474
7	Naive Bayes	0.623	0.477	0.294	0.180	0.821

In optimizing the randomized search for AUC, the following results were generated:

	Method	AUC	Accuracy	F1 Score	Precision	Recall
6	Gradient Boosting Classifier	0.729	0.761	0.428	0.311	0.686
4	Support Vector Machines	0.730	0.740	0.417	0.294	0.718
5	AdaBoost	0.719	0.757	0.417	0.304	0.667
1	Logistic Regression	0.727	0.743	0.416	0.296	0.704
0	Decision Tree	0.718	0.739	0.409	0.293	0.690
3	Random Forests	0.641	0.837	0.376	0.377	0.378
2	K Nearest Neighbors	0.622	0.696	0.308	0.219	0.522
7	Naive Bayes	0.623	0.477	0.294	0.180	0.821

Justification

The benchmark logistic regression with no hyperparameter tuning produced the metrics shown below in its 3-fold cross validation. The final model offered improvements in AUC, Accuracy, F1 Score, and Precision, while having a slightly lower Recall. Final model metrics are from the previous page, where F1 score was optimized for within the inner folds of the nested cross validation. While final model results are an improvement, the AUC, F1 Score, and Precision are not good enough to use in a production environment and do not solve the problem of adequately predicting default-likely loans. Given the poor precision scores, these models would identify too many loans as potentially defaulting when they would likely not be at risk of defaulting.

Benchmark Model

	Method	AUC	Accuracy	F1 Score	Precision	Recall
0	Logistic Regression	0.723	0.733	0.409	0.287	0.710

Final Model

	Method	AUC	Accuracy	F1 Score	Precision	Recall
6	Gradient Boosting Classifier	0.730	0.777	0.437	0.326	0.667

The Mamonov study produced the following metrics (each being the average of its 10 values from 10-fold CV:

Metric	Lowest Scoring Model	Score	Highest Scoring Model	Score
Recall	Random Forests	0.589	Neural Networks	0.834
Precision	Random Forests	0.309	Gradient Boosting	0.428
Accuracy	Support Vector Machines	0.593	Gradient Boosting	0.692

Mamonov study models were optimized for recall with a prediction threshold of 30% using 10-fold stratified CV and Fannie Mae data from Q1 2000 to Q1 2014 (22 million records). My models optimized F1 score (or AUC) using default sklearn prediction thresholds, 3-fold stratified CV and a subset of 5,000 loans from Q1 2016 data.

My highest scores are comparable to Mamonov results, though I used far less data:

Metric	Benchmark High Score	My High Score
Recall	0.834	0.821
Precision	0.428	0.386
Accuracy	0.692	0.840

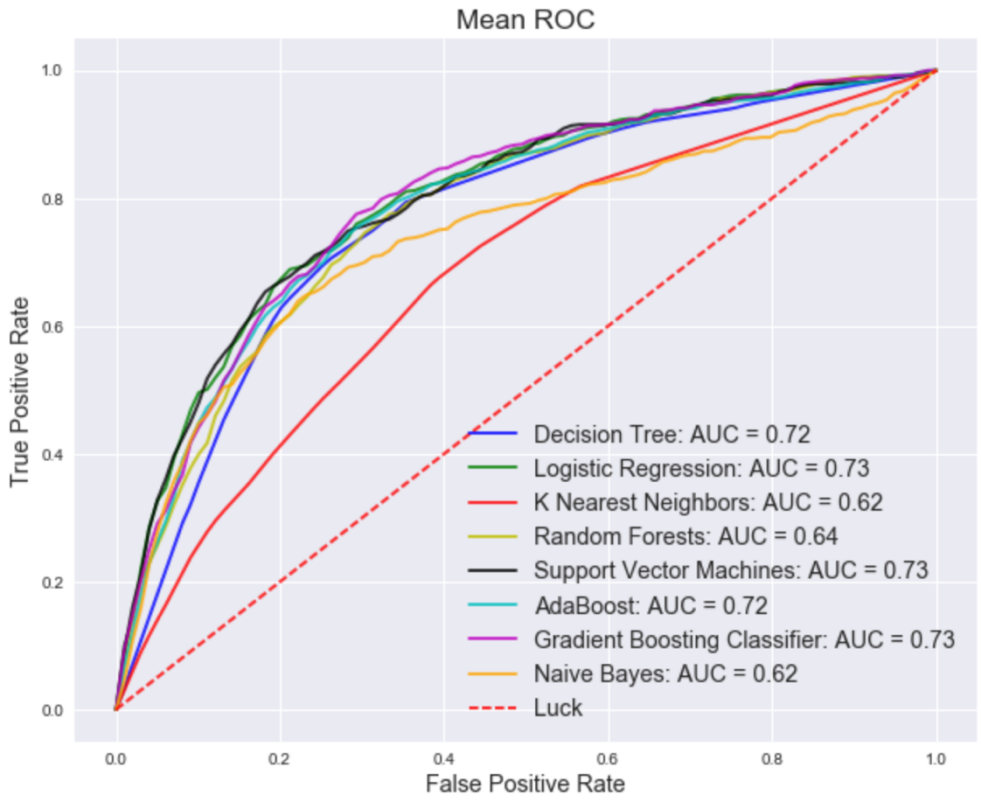
Like the Mamonov results, my models had poor precision (predicted too many non-defaults as defaults), while having better recall (predicted a decent portion of actual defaults as defaults), and had good accuracy.

Interpreting AUC

In optimizing for AUC in the randomized search, 10-fold CV produces mean ROC curves and mean AUC's as shown in the graph below. Each of the ROC curves in the graph is the *mean* ROC curve for a given model, based on the 3 ROC curves from testing that model on each of the 3 test folds.

The median AUC of all model mean AUC's is 72%, which can be interpreted as the typical probability that a model will assign a randomly chosen positive observation a higher probability than a randomly chosen negative observation. Another interpretation is that 72% is the average recall across all potential specificity values.

An AUC of 50% would indicate the model does no better than predicting randomly, so these models do have some predictive value.



Confusion Matrices

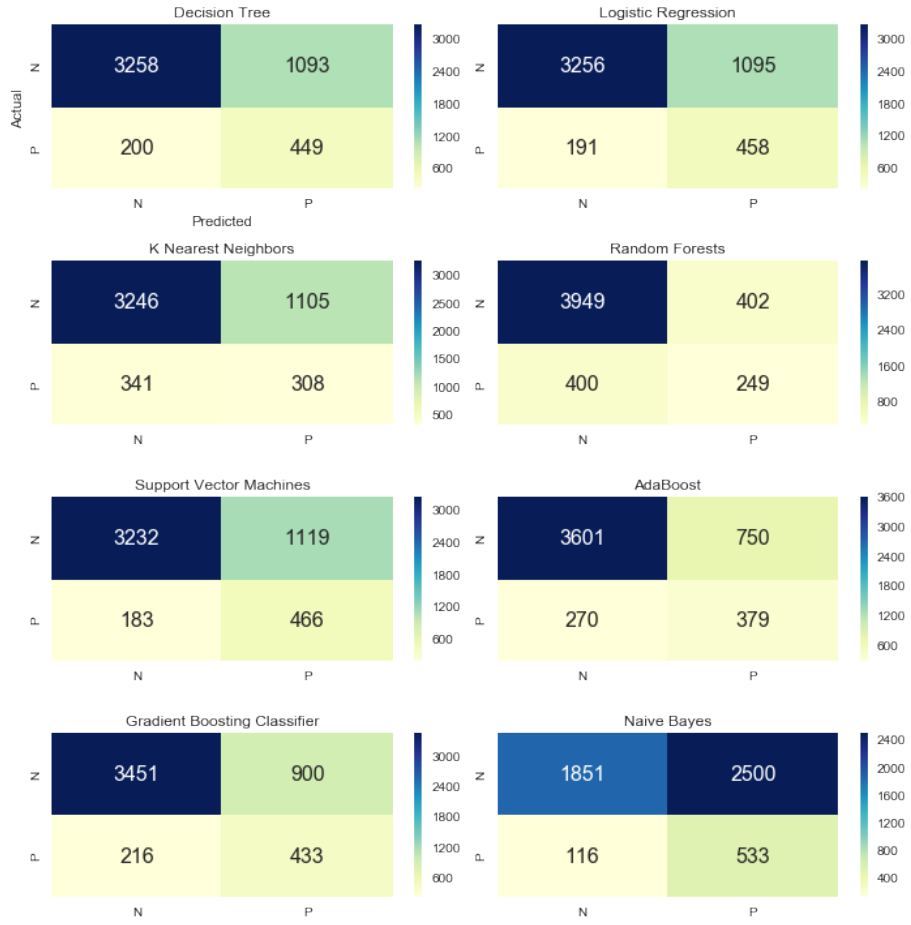
Confusion matrices show, from left-to-right, top-to-bottom: TN, FP, FN, and TP predictions.

TN = True Negative
FP = False Positive
FN = False Negative
TP = True Positive

Since most observations are negative (non-defaults) and the models do a good job predicting these observations, True Negatives have the highest value in each confusion matrix.

Classifier metrics like precision, recall, and accuracy, can be calculated from these matrices and compared (approximately) to the metrics in the first table in the Model Evaluation section:

For example, for SVM, precision = TP / (TP +FP) = 466 / (466 + 1119) = .294.



Feature Importance

Feature importances were generated using a Random Forest, the chi-squared test, and the Recursive Feature Elimination method (RFE). The chi-squared test requires non-negative features and the greater the test statistic, the greater the feature importance.

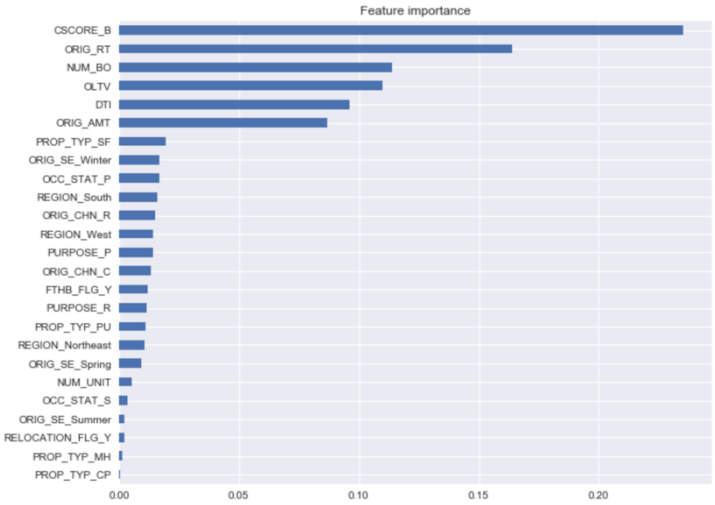
RFE recursively removes the least important features. In the rankings below, the lower the RFE ranking, the more important the feature and the farther along in the process the feature will remain without being removed.

CSCORE_B and NUM_BO are ranked in the top 5 in all 4 methods. Predictors that are ranked in the top 5 in at least 3 out of the 4 methods include: OLTV, and ORIG_RT. Predictors ranked in the top 5 in 2 out of the 4 methods include: DTI and RELOCATION_FLG_Y.

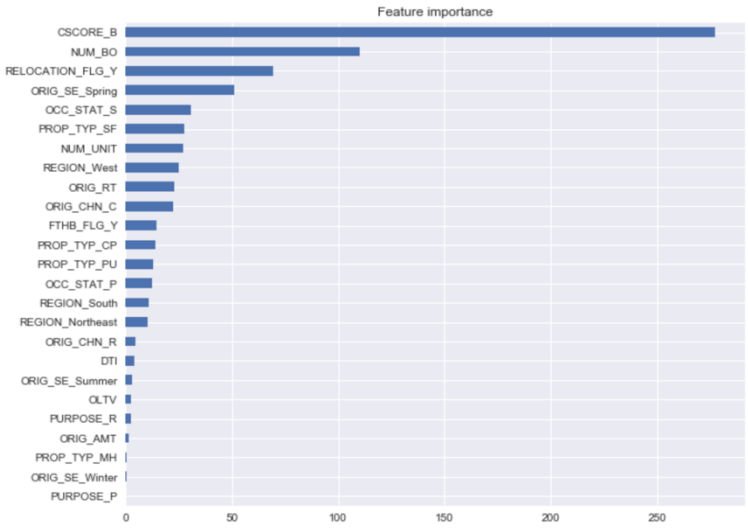
The Mamonov study provides variable importances using Neural Networks and Random Forests. In at least one of these methods, top 5 variables include CSCORE_B, OLTV, and DTI. ORIG_RT is in the top 10 in both methods.

The Mamonov study's results did not show NUM_BO or RELOCATION_FLG as important variables, though all other top variables in this analysis were included in their top 10. It makes sense that the top variables are the most important. Credit score, interest rate, loan-to-value, and debt-to-income are all intuitively indicative of a borrower's financial status and thus ability to make mortgage payments.

Random Forest Classifier



SelectKBest - Chi-Squared Test Statistic



RFE - Random Forest

Predictor	Ranking
ORIG_RT	1
RELOCATION_FLG_Y	1
OLTV	1
NUM_BO	1
CSCORE_B	1
NUM_UNIT	2
DTI	3
PROP_TYP_CP	4
ORIG_SE_Summer	5

Logistic Regression

Predictor	Ranking
ORIG_RT	1
OLTV	1
NUM_BO	1
DTI	1
CSCORE_B	1
ORIG_AMT	2
PROP_TYP_SF	3
OCC_STAT_P	4
ORIG_CHN_R	5

Exploring the Relationship Between Predictors and Outcome

The above feature importance analysis suggested the importance of the following variables in predicting 3+month defaults:

CSCORE_B, NUM_BO, OLTV, ORIG_RT, DTI, RELOCATION_FLG_Y

A logistic regression with default parameters was fitted to the entire subset of 5000 observations using the preprocessing pipeline shown in the *Model Building* section, but excluding the feature selection step, so that all features could be examined. Coefficients do not have normal interpretations because numeric predictor predictors were log transformed and scaled. However, larger coefficients still indicate that the predictor results in a greater likelihood of Y=1 and smaller coefficients indicate that the predictor results in a lower likelihood of Y=1.

As debt-to-income, origination rate, and origination loan-to-value ratio increase (DTI, ORIG_RT, OLTV), the likelihood of default increases the most, compared to other predictors. This makes sense because increasing values for these variables indicates a worse financial status of the borrower, so for them to indicate a greater likelihood of defaulting, makes sense.

As number of bedrooms and credit score increase, and for relocating employees (NUM_BO, CSCORE_B, RELOCATION_FLG_Y), the likelihood of default decreases the most, compared to other predictors. This also makes sense because these variables indicate a better financial status of the borrower, so for them to indicate a lower likelihood of defaulting, makes sense.

	Coefficient
ORIG_RT	2.843
DTI	2.168
OLTV	1.477
ORIG_SE_Summer	1.013
PROP_TYP_SF	0.803
OCC_STAT_P	0.637
OCC_STAT_S	0.455
PURPOSE_P	0.373
PURPOSE_R	0.371
PROP_TYP_PU	0.319
ORIG_CHN_C	0.292
ORIG_CHN_R	0.252
ORIG_AMT	0.165
PROP_TYP_MH	0.066
REGION_South	0.064
FTHB_FLG_Y	-0.112
REGION_West	-0.119
REGION_Northeast	-0.193
ORIG_SE_Winter	-0.341
ORIG_SE_Spring	-0.589
PROP_TYP_CP	-1.434
NUM_UNIT	-2.112
RELOCATION_FLG_Y	-2.501
NUM_BO	-2.694
CSCORE_B	-4.084

The Final Model

Given that Gradient Boosting (GBM) had the best metrics when optimizing F1 score and the best mix of F1 and AUC in optimizing AUC score in the nested cross validations, it was selected as the final model. Randomized search was then run on the entire data set of 5,000 observations to again tune feature selection thresholds and GBM hyperparameter settings within the same kind of pipeline used in the nested cross validation for performance testing and model selection. The optimal pipeline produced contained the following hyperparameter settings:

```
{'clf__learning_rate': 0.01, 'clf__max_depth': 4, 'fselect__threshold': 0.0001}
```

The feature selection threshold results in the selection of most features: 24 out of 25 features (all features, except PROP_TYP_CP). This optimized pipeline was then fit to the entire training set to produce a final model for predicting future, unseen data. Given the nested cross-validation performance metrics, this model will identify many non-defaults as defaults, and so is likely not good enough for deployment in a real-world setting.

Coding Complications

I had wanted to use the Weight of Evidence algorithm in handling the STATE categorical variable. I wrote a custom function for it, but including it in the preprocessing pipeline would have required it to have been made into a class with fit and transform methods. I found it easiest to just make a REGION variable. For the preprocessing pipeline, I was limited to just sklearn functions with fit and transform methods.

I had intended on implementing a learning curve using the selected classifier (gradient boosting), however, an error was generated using the hyperparameters selected from the randomized search. I could not resolve the error. Other classifiers worked in the learning curve code.

In making the ROC plot, AUC's were originally calculated using the auc() function, but resultant AUC values were not consistent with those created in the model evaluation and selection process, which used sklearn's roc_auc_score(). The auc() function uses the trapezoidal rule, which I have read can sometimes overstate AUC values, which in this case, did result in higher AUC's than those generated from roc_auc_score(). I simply used AUC scores generated from the model evaluation phase in the ROC plot.

Final Model Robustness Check

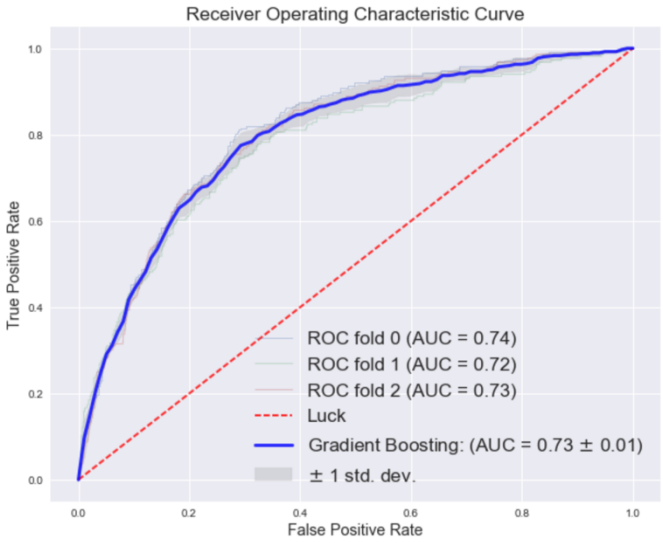
In examining the robustness of the selected model, Gradient Boosting, I inspected the metrics in each of the 3 outer folds within the 3-fold nested cross validation. As shown below, all metrics appear to have consistent values across all folds. Each row shows results for a fold.

	AUC	Accuracy	F1 Score	Precision	Recall
0	0.739	0.768	0.440	0.321	0.700
1	0.715	0.737	0.403	0.286	0.685
2	0.733	0.779	0.440	0.327	0.671

Conclusion

Free-Form Visualization

ROC curves and AUC for each fold of the "winning" model, Gradient Boosting, is shown to the right, with the mean curve in bold. ROC curves are similar, regardless of the fold, further demonstrating that the model appears fairly robust to the data fed to it.



Reflection

Model results, predicting mortgage loans that default for 3 consecutive months, as judged by F1 score and AUC, were poor/fair, though variables with the highest feature importances and their relationship with the outcome variable made intuitive sense. These results were comparable to the Mamonov study models' and offered an improvement over the initial models built without SMOTE or hyperparameter tuning.

The task of predicting mortgage defaults offered several challenges. To start with, the exact nature of the outcome variable had to be determined. The performance dataset has multiple rows for each loan, each representing its status x months into its life. I could have predicted how far into each loan's life that it was expected to become delinquent for the first time. Given that the performance data has foreclosure dates, I could have predicted whether the loan would eventually foreclose or how long it would take for the loan to foreclose (which would be close to zero if the loan was not expected to foreclose). There is a difference between foreclosure and delinquency. If a borrower misses at least one payment, then the loan becomes delinquent during this time. Eventually, the lender may decide to take ownership of the home and sell it in a

foreclosure sale if the lender has a first lien on the home. A first lien lender has the right to take ownership of the home before other lenders that have liens on the home, like second lien lenders.

The classification imbalance would have been less severe if the outcome had been defined as whether a loan becomes delinquent at least once during its lifetime. However, this situation is less indicative of severe financial distress and an eventual foreclosure sale and claim to a mortgage insurer than when a borrower has missed several payments in a row (when the loan has been delinquent for several consecutive months). Thus, if the outcome had been defined as at least one month of delinquency, there would have been many "good" borrowers with Y=1 values, meaning borrowers with better financial metrics, like credit score, debt-to-income ratio, and loan-to-value ratios. This would have potentially diluted the data and weakened the correlations and made the models less predictive. This scenario is also likely of less interest to lenders and insurers, which is why the Mamonov study defined the outcome as 3+ months of delinquency, as well.

Understanding the data and definitions of the data was a challenge. I had trouble figuring out the exact definition of *Delq.Status* and what each row in the performance dataset represented (the status of the loan x months into its life).

Another difficult aspect of the project was figuring out and implementing the nested cross validation to avoid data leakage in the model performance estimates. I have not done this before and figuring out the right code to get this done was a challenge. I could not include certain preprocessing steps in the pipeline, like the weight of evidence algorithm for transforming the state variable. All pipeline estimators need fit and transform methods, and since sklearn did not have a weight of evidence method, I would have needed to have made my own from scratch.

Summary of Analysis

- A small subset of Fannie Mae Q1 2016 data was used
- Preliminary data wrangling involved creating the outcome variable (3+ month defaulters), creating 2 new features, and removing 3 predictors with too many missing values
- Data was explored to do a preliminary assessment of correlations, a reasonability check, predictor distributions, and missing values
- Data was further preprocessed by log transforming numeric predictors to address skewness, converting categorical variables into binary dummy variables, and removing a variable with low variance
- Nested Cross Validation was implemented for 8 classifiers using a pipeline with preprocessing steps, where a randomized search was used in the inner folds for model and feature selection hyperparameter tuning, and both inner and outer folds used 3-fold stratified cross validation.
- The pipeline included the following steps: median imputation, min/max feature scaling, feature selection using a random forest, SMOTE for 50/50 ratio over-sampling to address the class imbalance, and implementing the classifier
- The Nested CV was run once to optimize F1 score in the randomized search, once to optimize AUC score in the optimized search, and once without SMOTE or hyperparameter tuning for comparison to the refined methods

- All models were evaluated based on F1 Score and AUC, though Accuracy, Precision, and Recall were still calculated.
- Gradient Boosting had the best mix of F1 score and AUC, both in optimizing for AUC score and F1 score in the randomized search.
- Gradient Boosting (GBM) metrics were compared with the benchmark model, an un-tuned Logistic Regression. GBM had better metrics, except for Recall. Results were also compared with the Mamonov study, and were fairly similar.
- Based on model results, these models are not good enough for actual deployment to predict default-likely loans. Too many loans would be classified as default-likely, when they are not actually likely of defaulting (precision is too low).
- The final model (Gradient Boosting) was built on all of the data, where hyperparameter tuning and feature selection was also redone using 3-fold stratified cross validation.
- The final model's robustness was examined by calculating metrics for each fold in the nested cross validation that optimized F1 score. Metrics appear consistent across folds, thus suggesting robustness in the final model.

Improvement

Project improvements include:

- Implementing the State of Evidence algorithm to apply the STATE variable
- Tuning every estimator hyperparameter instead of just a couple
- Further exploring why feature importance results differ between each method
- Trying other feature selection methods, like best subset selection (though sklearn does not have this method)
- Tuning the SMOTE algorithm such that over-sampling does not result in 50/50 minority/majority class proportions in the dataset
- Including additional features from the data, like co-borrower credit score, seller, and CLTV
- Implementing more methods, like Neural Networks, which had the best recall score in the Mamonov study, and XGBoost
- Including the entire dataset in the analysis, instead of just a small subset
- Combining Fannie Mae data with outside data indicating macro-economic trends (there were many more mortgage defaults during the economic crisis of 2007-2008)
- Including more data about the mortgage loan borrowers, if it was available from Fannie Mae, such as education level and employment, as recommended by the Mamonov study

References

Fawcett, T. (2016, August 25). Learning from Imbalanced Classes. Retrieved from https://svds.com/learning-imbalanced-classes/

Friedman, J. H. (1999). Greedy Function Approximation: A Gradient Boosting Machine. IMS. Retrieved from https://statweb.stanford.edu/~jhf/ftp/trebst.pdf.

Jain, A. (2016, May 27). Complete Guide to Parameter Tuning in Gradient Boosting (GBM) in Python. Retrieved from https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python

Mamonov, S., & Benbunan-Fich, R. (2017). What Can We Learn from Past Mistakes? Lessons from Data Mining the Fannie Mae Mortgage Portfolio. *Journale of Real Estate Research*,*39*(2), 235-262. Retrieved November 26, 2017, from http://pages.jh.edu/jrer/papers/pdf/past/vol39n02/9873-04.235_262.pdf

O'Shea, B. (2018, January 23). Credit Score Ranges: How Do You Compare? Retrieved from https://www.nerdwallet.com/blog/finance/credit-score-ranges-and-how-to-improve/