Capstone 1 Final report
Matthew Gudorf

1. Introduction

The financial loss associated with loans being "charged-off" is incredibly large. According to the FDIC the amount of loans and leases merely past due in 2018 is on the order of 95 billion dollars. This provides motivation to create better metrics and criteria for deciding whether a loan should be provided to potential borrowers. In addition to machine learning metrics, the amount of money lost to charged-off loans or delinquency provide a quantitative metric for future evaluation of the model. That is to say, the model can be validated by seeing a decrease in the amount of capital lost over time and the amount of delinquency. This is perhaps one of the most common types of analysis and applications of machine learning methods but as with most cases, deciding on specific algorithms and parameters to use is mostly black magic. This project provides a comparison between different models based on predictive power and computational accuracy. In addition, this project attempts to find a smarter and more efficient criterion than scanning through the parameter values to find the best model. The process of finding the best model is typically unique to each dataset, hence the lack of generalized results.

The main steps to attain this goal are as follows. We must first clean the dataset, Lending Club loan data (https://www.kaggle.com/wendykan/lending-club-loan-data), by accounting for missing values and deciding on which features in the dataset are important. The feature selection will be determined by an informed opinion of the dataset as well as computational results. An important caveat is that close attention must be paid to whether there is contamination between features. This can be determined by computation of correlations as well as manual interpretation of the dataset. One specific example from this dataset would be the correlation between the remaining amount of principal and the remaining amount of principal that was funded by investors.

Once the data has been cleaned and the important features have been selected, the core machine learning process can begin. This training and predictions using machine learning can begin. Predictions of continuous data and discrete data will be compared between the different models in order to determine the best method for this dataset. The methods tested will include linear and ridge regression for continuous data and logistic regression and random forest classification for discrete data. The predictions and efficiency of each of these models can be compared but the best measure of success would be the actual results regarding future values of delinquency and the amount of charged-off loans. Unfortunately with time dependent predictions the main method of validation requires waiting. The dissemination of results depends on the audience, for a more informed audience there are a variety of metrics which can be used to demonstrate the validity of the method such as the area under the receiver operating characteristic curve, confusion matrix, etc. The data exploration is of course best demonstrated visually. With this coarse outline now in place, let us begin with the actions taken in the data cleaning and wrangling step of this project.

Upon importation into a Jupyter notebook, a warning was displayed indicating that the data contained variables of mixed type (e.g. a combination of python lists and floats in one data feature). This is an issue that could be handled by interpreting the imported data as strings, for instance. There are a number of such properties that need to be taken into account before the prediction models can be created. The process of processing and reformatting the data is commonly referred to as data cleaning or data wrangling. Specifically, my belief is that the important data cleaning decisions to make are: how to handle missing values, redundant information (a feature column identically equal to a single value), multiple types of data. Before a description for how to handle these traits can be offered, a necessary comment must be made towards an inherent problem in even wanting to handle these issues manually. Namely, any custom action must be done in an agnostic way such that it isn't tailored to the current data set, otherwise we right creating a model that doesn't generalize. Therefore the utmost care must be taken whenever a custom data transformation is being contemplated.

Firstly, for the missing data values there are a number of options. For instance the missing values can be imputed by a specified method (replace the missing value with the mean of the data feature, replace with a constant value, etc.), the samples containing missing values can be dropped from the data set, or if the feature is a categorical variable, "missing" can become its own category. Imputation is not acceptable as it will ultimately alter the summary statistics of the data, excluding the statistic maintained by the replacement rule. That is to say, if the missing values are replaced with the mean then the mean will be maintained, but other summary statistics will not be maintained. An alternative would be to just drop all samples with missing values; unfortunately the missing values are distributed in such a manner such that if the samples with missing values are removed from the dataset then there are exactly zero samples left to work with. While the distribution of missing values is not optimal, I developed a strategy to work around their distributions. There are a number of features which are missing values for the vast majority of samples. This should not be confused for a sample that is missing a vast majority of feature values. I first drop the features which have a proportion of missing values above a certain threshold. The distribution of these percentages (i.e. the percent missing per feature) has multiple plateaus. This leads to an interesting choice of where to put the threshold. While it is still arbitrary, the essential choice is whether to include the "plateau" that occurs around 40%. I have not decided upon the best choice but it is evident that these values correspond to a different time period than the features with a smaller proportion of missing values. More on this in the data story section of this project. The main strategy I employ for the current data used in the two-step modeling procedure is to first replace all missing values in categorical feature data with a new category labeled "Missing". Next, I remove all features with a proportion of missing values greater than 30%. Lastly, the remaining samples that have missing (numerical) values are removed from the dataset. The main trade-off in this process is that I keep more samples than features. An unintended consequence is that perhaps this is unfairly biased towards recent data samples. This might not be a terrible idea however because I believe the model should reflect current policies as to incorporate the current standards and practices which have undoubtedly been studied. Once these measures to account for missing values have been performed, I move to the other general measures that are performed to process the data to be ready for each stage of the modeling process. For example, there are some features which contain a single value. As it stands, any partition of the data into training and testing data quite obviously contains only one value. This is unlikely to be an example of sampling bias due to the number of samples; it is simply a systematic issue with how the data is recorded. For example, there is a feature named "policy_code" which only takes a value of 1. Presumably this indicates that the loan was accepted. The alternative to a systematic issue is that the dataset suffers from sampling bias. The number of samples is in the millions and therefore it seems very unlikely that this could be the case. Because of this, these features seem worthless for my purposes. The lack of utility stems from the fact that the training domain being a single value has the implication that it won't know how to handle differing values. There are also features which have a highly imbalanced distribution, namely, the vast majority of samples take one value. This case is not as well motivated as the prior case, but still I believe it is a valid measure to prune these features from the dataset. One last case of hard to manage data features are categorical features which are idiosyncratic in nature. For example, one such feature would be the description for the reason for the loan provided by the borrower. Excluding certain generic categories such as "debt consolidation", there are a very large number of unique responses and hence categories that would need to be encoded. One way of handling this would be to bin all unique responses into an "Other" category. The problem with this approach is that the cutoff would be arbitrary, essential defined by how many categories I deem important. I deemed this too dubious and so I opted for removing these types of categorical features from the dataset, such that they would not be used in the modeling procedure.

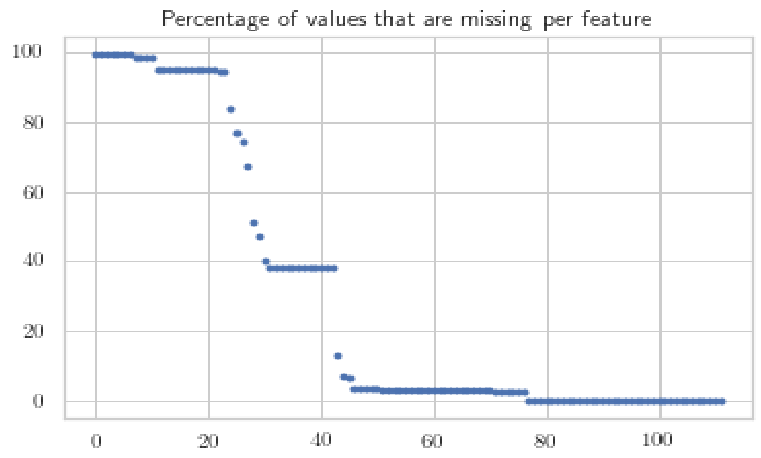


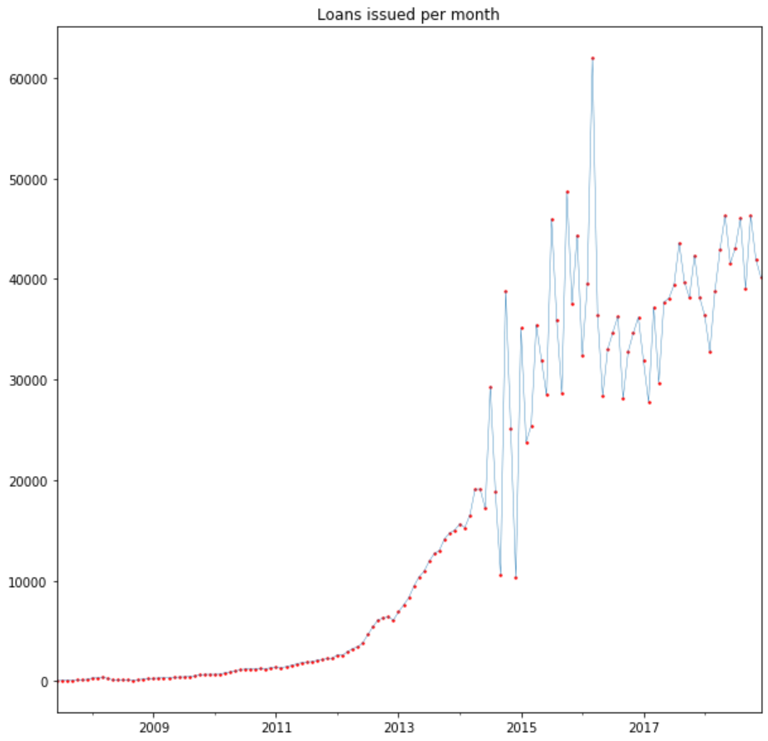
Figure 1 : Percentage

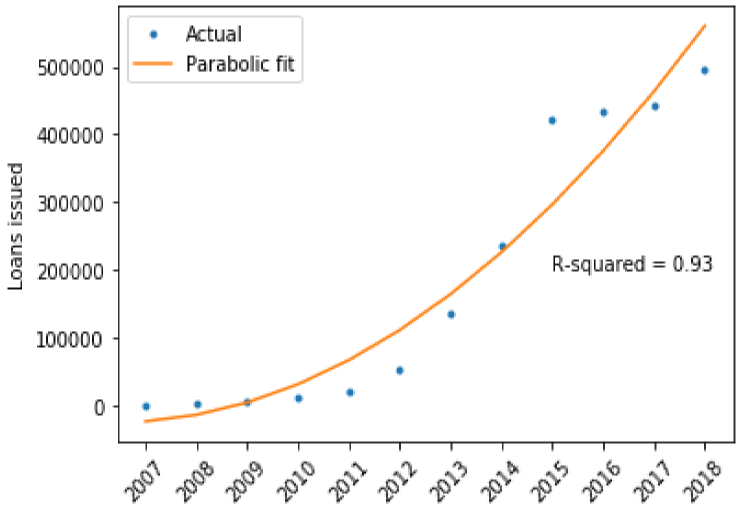
The last component of the data wrangling is processing the data to adhere to our modeling hypotheses. For instance, the loan outcome classification portion of this project wants to predict whether or not a loan will be charged off by its maturity date. Therefore, I need to first subset the loans which have matured, and then choose a way of handling the possible outcomes of each loan (recorded in "loan_status"). For the maturity subselection, the data contains information pertaining to loans of three year terms and five year terms, up to December 1st, 2018. Therefore, I took the subset of loans whose issuance date plus term was equal to or less than this date. It could be argued that there should be two models, one for each loan term, as they might represent distinct populations. While I did process the data using this belief, I did not end up creating separate models for the two populations.

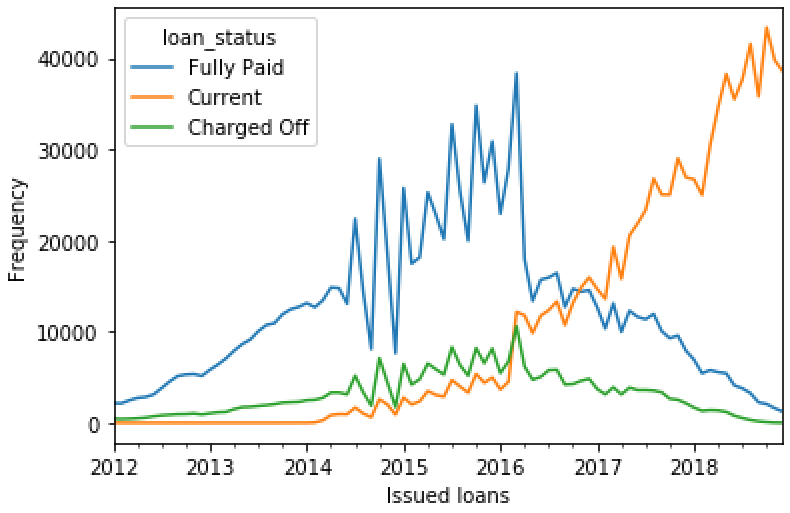
The distribution of values for loan status of the matured loans were largely dominated by "Fully Paid" and "Charged Off". As such, these are the only values that were maintained during the selection process. The value that we want to predict, that is, the state of being "Charged Off" was given a binary value of 1. Likewise, the "Fully paid" was given a value 0. This allows for a formulation of the loan outcome prediction model which attempts to classify as many true positives as possible. For the capital recovery prediction for charged off loans it is natural to take the subset of loans that have been charged off. Even when this selection is made, there is still a large number of loans from which zero capital has been recovered. This is explored more in the modeling process; the model only considers strictly positive recovery amounts as the zero valued recoveries seem to only contribute noise to the model.

Before the modeling it is useful to get an idea of the data that is being used. This is done by both exploring the data using visualization and statistical techniques. It should be noted that whenever a subset of the data is taken, a check is made regarding the balance of the value distribution. If at any point a feature has a single value representing a 99% proportion of the data, it is dropped. The next step after cleaning and wrangling was to explore and get a better feel for the dataset that is being worked with.

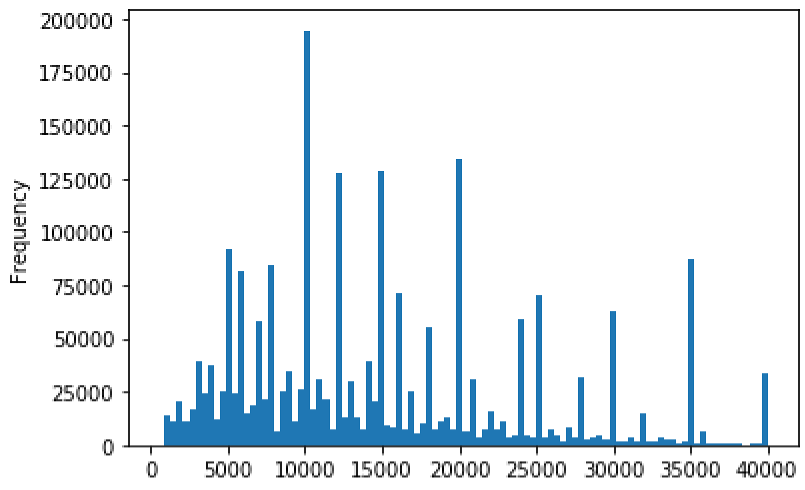
2. Exploratory Data Analysis


Loans issued per month
60000
50000
40000
30000
20000
10000
0
2009
2011
2013
2015
2017


Actual
Parabolic fit
Loans issued
R-squared = 0.93
500000
400000
300000
200000
100000
0
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018

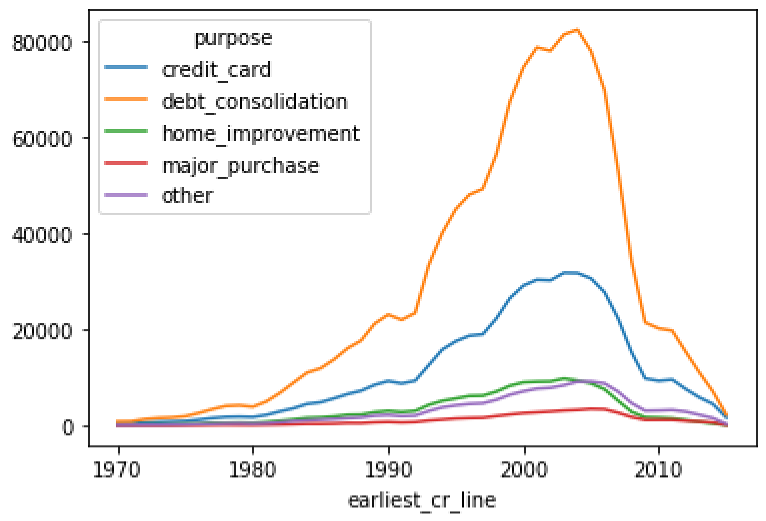


In this dataset there are categorical variables in the form of strings, discrete numerical variables, and continuous numerical variables. The data can also be segmented into information describing the borrower (geographical location, income, etc.) and information about the loan (principal amount, interest rate, etc.). There are also features that are time dependent and can be naturally visualized as a time series. One example of a time dependent feature is the date when each loan was issued. Aggregating by year and performing a polynomial fit assuming parabolic growth for the number of loans issued per year results in a respectable r-squared value (coefficient of determination) value of 0.92. This nonlinear growth implies that our main deliverable, the model which predicts the outcome of a loan should become more important over time. A parabolic fit seems to capture the recent average growth trend but with the caveat that this growth is unsustainable. This is only a figure to give a sense of how the loans have grown up to the present moment. Aggregating by the (top three) statuses of the loan shows the time distribution of when fully paid, current and charged off loans. All of the loans have terms of three or five years so it's understandable that the approximate maximum of the fully paid loans and charged off loans occurs in the same range, three to five years ago. (The most current date in the data set is December 1st 2018). Continuing with the data exploration, a unique and interesting quantity is the distribution of loan amounts; it really shows the affect of human psychology by how the distribution manifests.

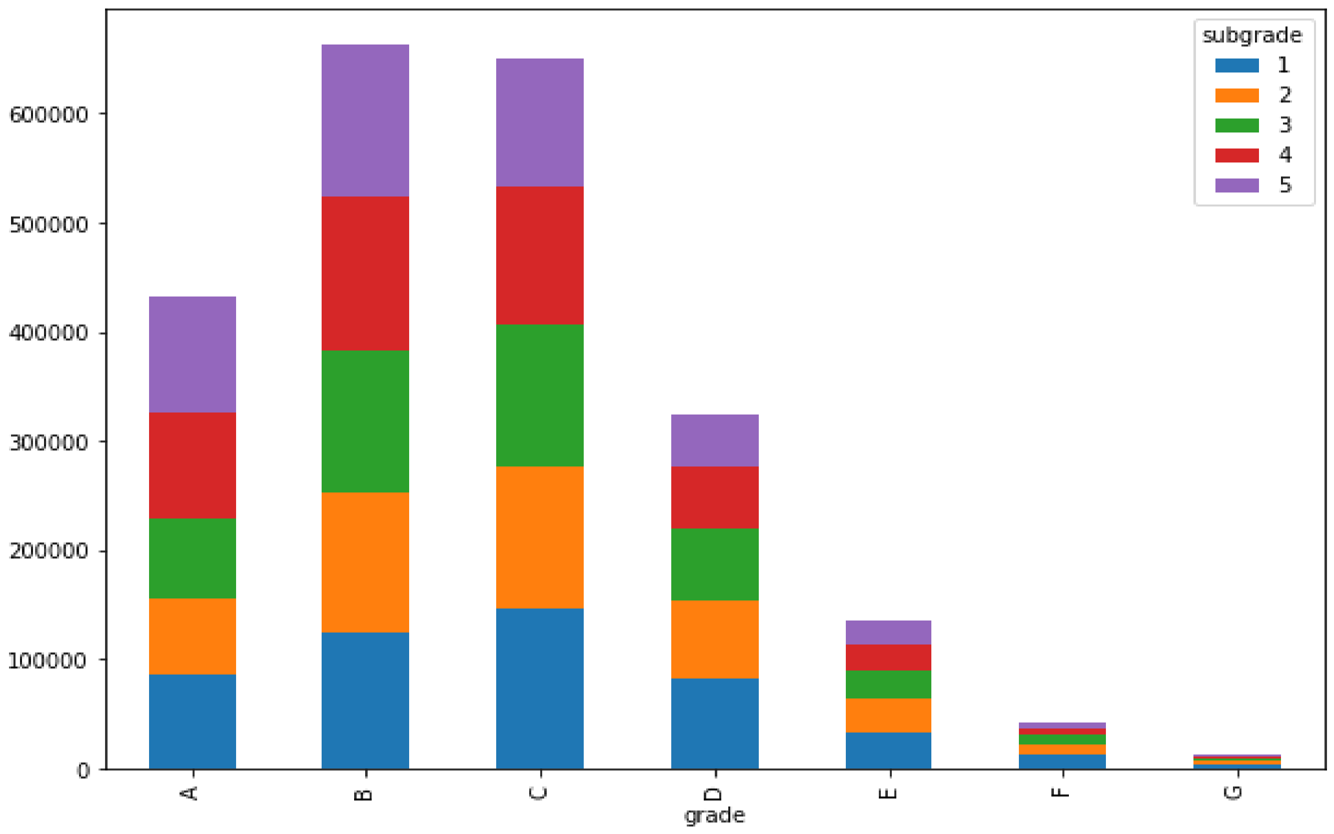


Why is this distribution so strange? As can be seen, the distribution seems to be clustered around "pretty" numbers. This includes round numbers like multiples of five thousand and ten thousand. To show this explicitly, let's look at the top ten most frequent loan values.
This makes the loan amount distribution some strange quasi-discrete distribution.

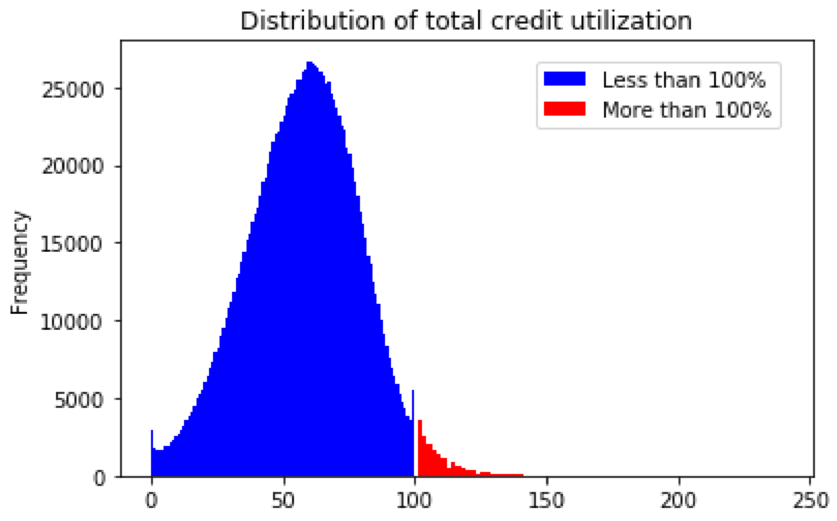
Amount	Count
10000	187236
20000	131006
15000	123226
12000	121681
35000	86285
5000	84765
8000	75033
6000	72089
25000	66453
16000	66418



There doesn't seem to be any distinctive behavior between loan purposes, this would have been evidenced by difference in the mean of each the time series. The goal of stratifying the earliest credit line by the purpose of the loan was to see if different age demographics use loans for different purposes. This would be measured by differences in the mean between these stratified distributions, as I believe it is a fair assumption that everyone gets their first credit line (earliest) around the same age. For the categorical variables the frequency of each category can be displayed in a histogram. Other features were investigated such as the distribution of utilized credit, the distribution of grades and subgrades of loans, the geographical distribution by zip code of the borrower, the date of the earliest credit line of the borrower. For the percentage of utilized credit it can be seen that there is a small portion of the dataset which exceeds the theoretical one-hundred percent threshold. The reason for these values is unknown but it was peculiar enough to merit investigation.. For the time dependency, care must be taken such that we do not accidentally contaminate the modeling process with data from the "future". This is commonly referred to as "data snooping".



subgrade
1
2
3
4
5
0
100000
200000
300000
400000
500000
600000
A
B
C
D
E
F
G
grade



Distribution of total credit utilization
Less than 100%
More than 100%
Frequency
0
5000
10000
15000
20000
25000
0
50
100
150
200
250

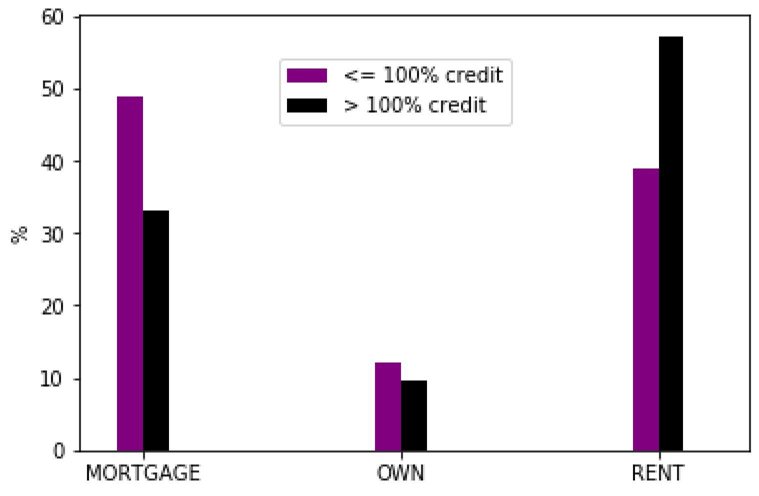
Looking at this distribution, the red region indicates that number of loan borrowers are above their credit limit; the reason for this is unknown but I'll hypothesize that this among with other properties can help stratify the borrowers into fundamentally different classes. Look at the loan amounts and average current balances to see the general loan information of each type of borrower

The average **balance** for borrowers utilizing **more than 100%** of their credit: $182575.92
The average **balance** for borrowers utilizing **less than 100%** of their credit: $143809.02

The average **loan amount** for borrowers utilizing **more than 100%** of their credit: $11729.86
The average **loan amount** for borrowers utilizing **less than 100%** of their credit: $15274.88

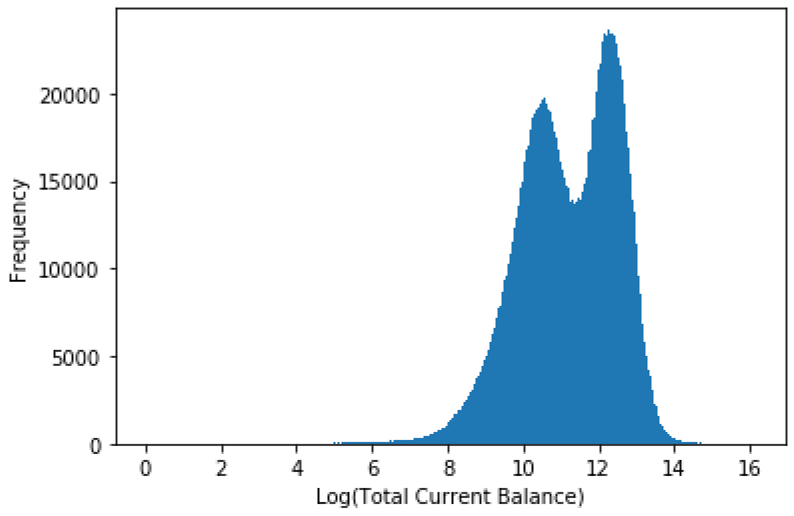
The average current balance is actually greater for people utilizing less of their credit, This is not indicative of not paying off loans; in fact, people that use less credit on average have loans with higher principal amounts. To get a sense as to whether the separation by credit utilization serves as significant borrower classifier, we can look at the differences in summary statistics between the two populations. The quantity that stood out to me is the total balance excluding mortgage. The averages for each sample population were computed to be approximately 40000 dollars; but it seems that this value nearly doubles when excluding mortgage balances. It could be that perhaps there is a difference in home ownership status between the two groups.

While this bar plot does not prove anything it does seem to imply that there is a difference in housing status between the two groups. It seems that a majority of the borrowers

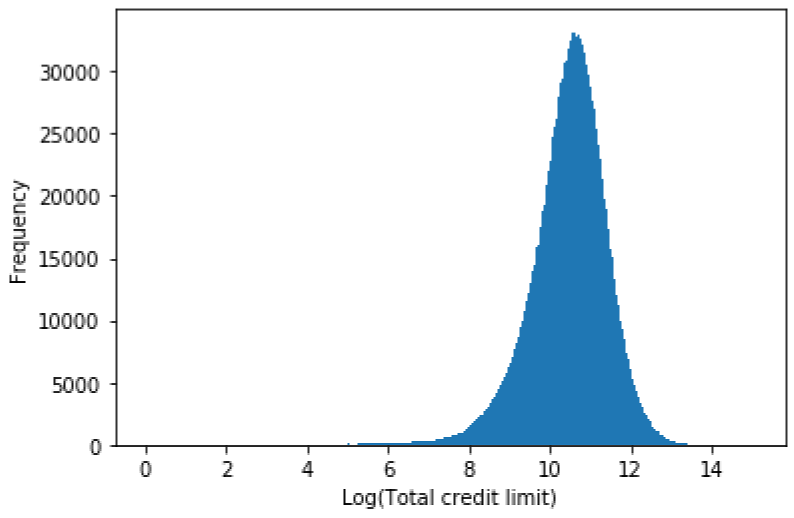


over their credit limits are renters while the other borrowers are more likely to either own or mortgage a home. The question I wanted to answer was this: can home ownership status be used as a distinguishing factor in any other manner? To explore this line of thinking, let's look at the total current balances of all borrowers.

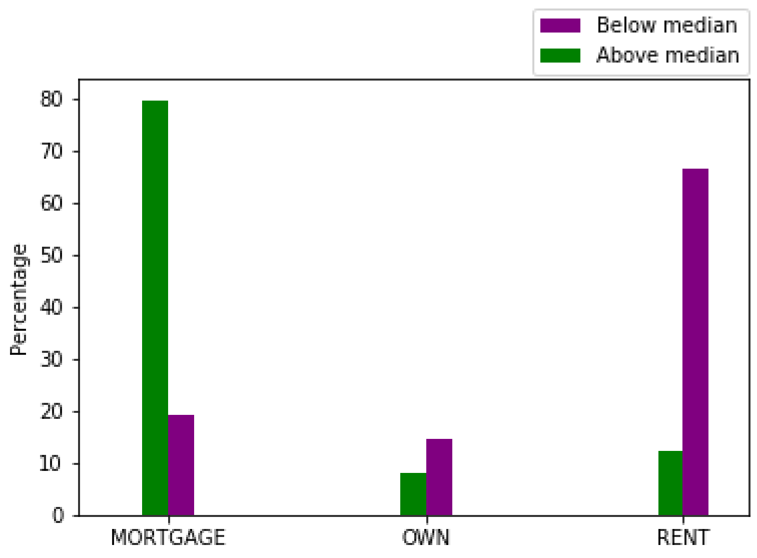
I would have guessed that the total current balance of all accounts would have a normal distribution. Plotting the histogram, however, shows another story. As can be seen, the distribution is distinctly bi-modal. As I want to explore the effect of home ownership, I then plot



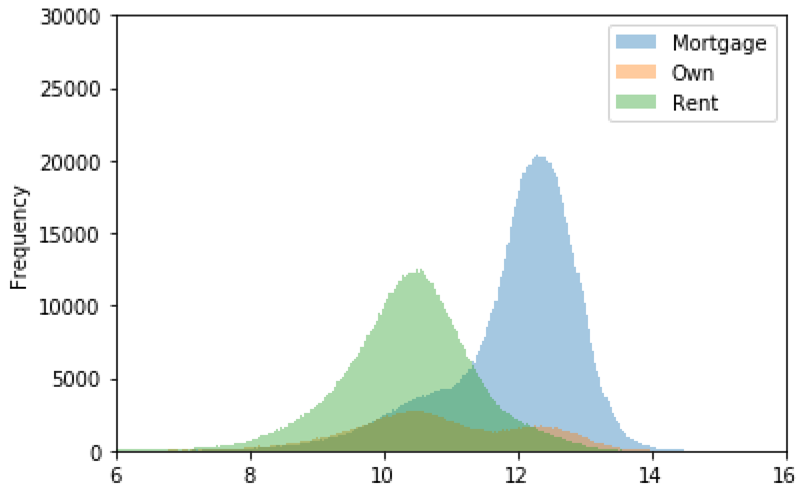
the total balance excluding mortgages. The effect of home ownership status seems immediate as the two distributions are entirely different.



To investigate the reason for the bimodal distribution I split the total current balance data in half (via the median value) and then create a bar plot for the home ownership status for each half. The stratification is much more drastic than was the case with credit utilization.



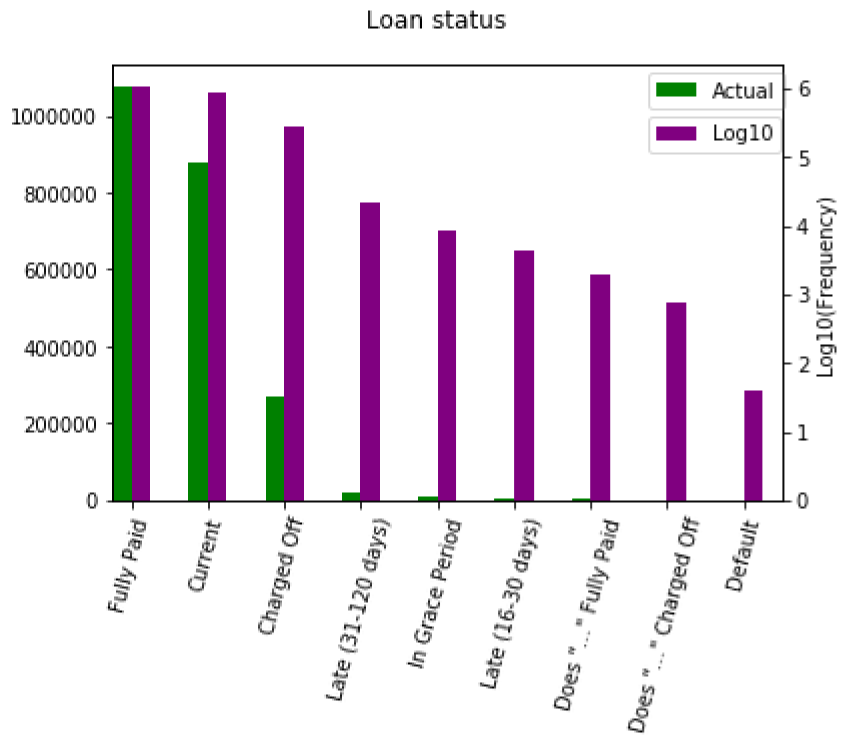
Let's visualize this by plotting the total current balance (log) distributions with respect to each category.



Own and mortgage and perhaps the distribution for those who have mortgages still look like they're bimodal, but the total current balance distribution for renters seems to be unimodal. This almost seems like it's hinting at the possibility that there are hidden populations of customers.

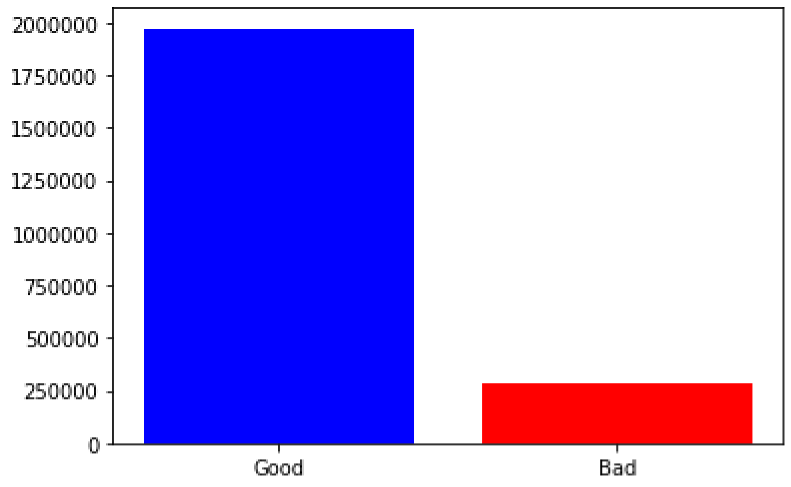
The idea I have in my head at least is that perhaps there is a useful way of subdividing or partitioning the data such that these subsets could each be treated as their own population; that is to say, each would have a separate model instead of there being single models for loan outcome prediction and recovered capital regression. The problem is that the next step forward, breaking down the mortgage and own categories is not obvious, even after repeating the same steps. So, in summary, it might be useful to further investigate or subdivide the borrowers into distinct populations, but up until now I have not found an obvious means of doing so.

To continue, the main goal of this project is to produce a model which accurately predicts the outcome of a loan. A bad outcome is defined as when either a borrower is late on payments or the borrower has charged off the loan. A good outcome is when the loan is paid in full. This “good” and “bad” dichotomy is an artificial construction resulting from aggregation by loan status. This frames our problem as a binary classification problem which can be modeled by logistic regression and random forest classification. With these methods and other considerations, a model for loan status prediction can be formulated.

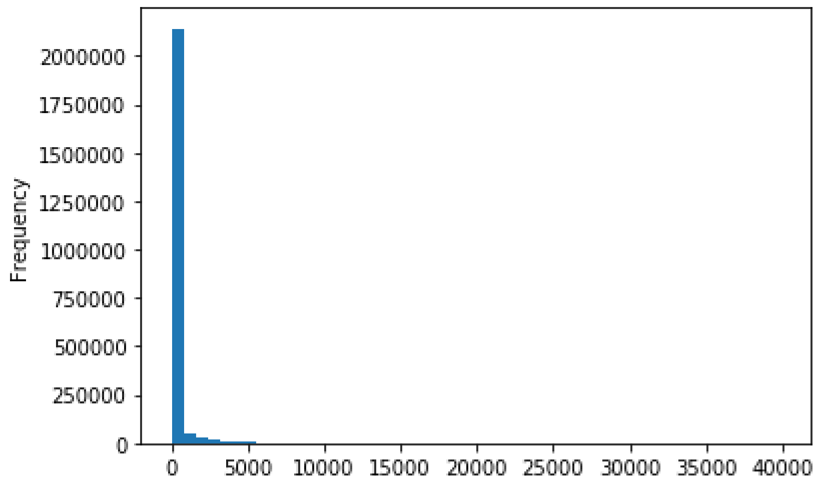


As we can see by this figure, the vast majority of loans fall into three categories: "Fully paid", "Current", and "Charged Off". Because of the time dependence of the problem, the quantity being modeled is modified to respect this distribution. The modification is to predict the outcome of loans by their maturity date. Therefore we can prune the loans which have not matured yet.

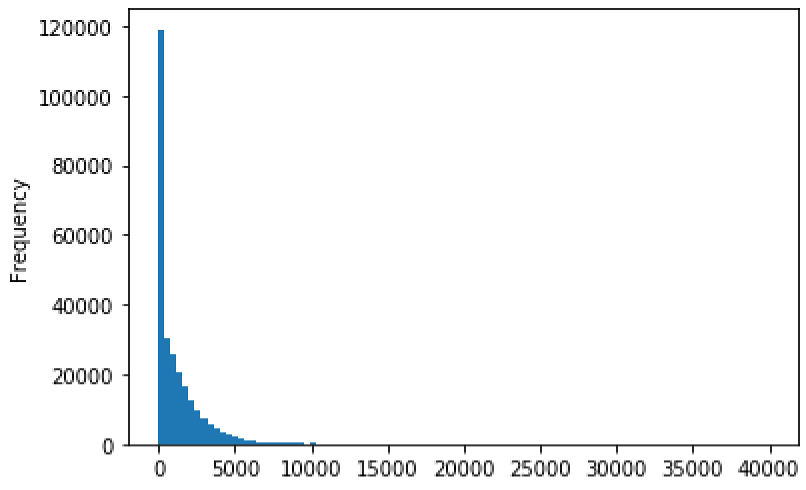
In addition, to make the problem a relatively balanced, binary classification problem the only categories that are retained are "Fully paid" and "Charged Off". The binary distribution plot shows the proportion of loans with bad outcomes.



What can be done with loans of bad standing? Instead of the money being lost to the ether, financial institutions will naturally attempt to recover capital of charged off loans. This amount of capital recovered can be modeled as a continuous variable via regression techniques such as linear regression and many others. This variable is important because it is closely associated with loans of bad status, and provides a recommendation or course of action for loans that have become charged off or delinquent. Unsurprisingly the recovery amount is correlated to loan status but this is a misleading quantity as loans of good status do not need recovery to begin with. Because of this, I postulate that it may be wise to completely filter out loans of good status before performing any modeling. The main concern is the order of these time dependent variables in regards to training and testing data sets for the models as well as for cross validation. If future data is accidentally used in the training then the predictions are essentially worthless. This goes down all the way to the level of normalization; that is, if normalizing the data then one should be sure to only normalize with respect to the training set.



How should these be accounted for in the modeling process? For the loan classification process, all time dependent quantities need to be removed from the model training process for the loan classification problem, as the model is attempting to predict a decision made at a certain point in time. The model for the recovery amounts is agnostic of time as it merely wants to predict the amount that can be recovered going forward; it is not a time sensitive decision; therefore, no considerations need to be made for the second stage of the capital recovery model.
As can be seen, nearly all of the loans have zero dollars recovered; but this is because we did not take into account the fact that most loans do not need to have money recovered from
them, as they are not charged off (as indicated by the binary bar plot above). Subsetting by the loans that have been charged off should give us a better idea of the distribution.



It seems that there are still a large number of charged-off loans that have had no money recovered from them which serves as motivation for this stage of the project. Analysis as to whether this can be used as a target variable is made in the modeling notebook.

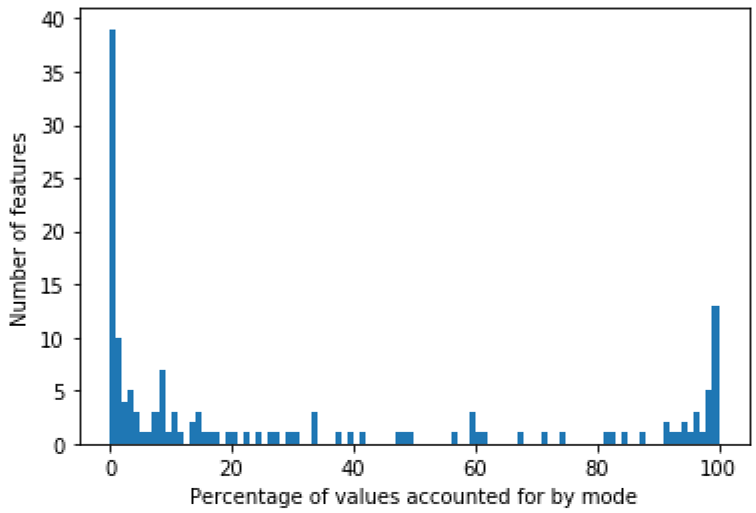
3. Statistical analysis

Before the precise formulation of any hypotheses sets, recall that some features are quite imbalanced and also there are both categorical and numerical variables. It is my belief that it is wiser to investigate numerical data first, specifically data defined on a continuous domain (approximately continuous, as most variables are in regards to money). This choice is motivated by my idea that continuous variables better represent the unique nature of individual human beings. The idea is to take distributions of continuous variables and find statistically significant categories of people not captured by any of the categorical variables. There is no indication a priori on where to start so a search began to find an anomaly.
The quantities that are analyzed are:

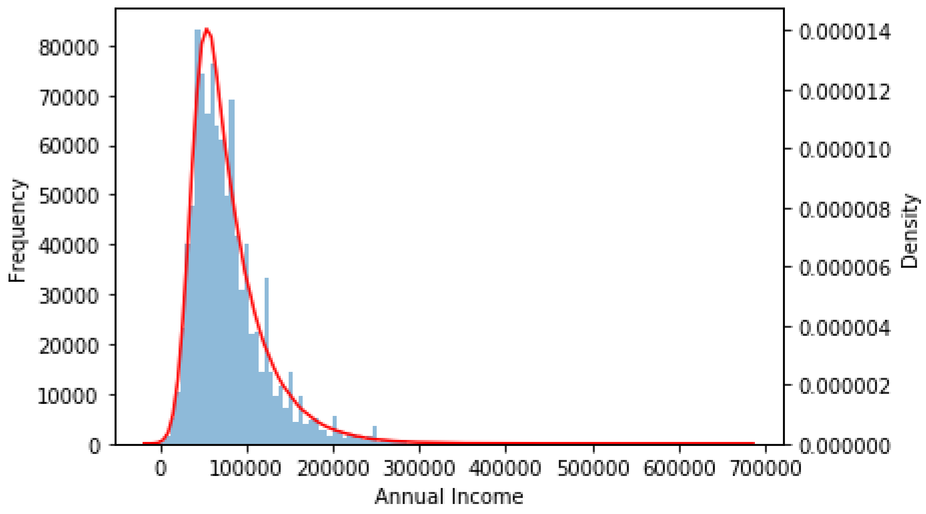
1. The annual income (without some very extreme outliers)
2. Total payments made to date
3. The percentage of credit being utilized
4. Number of open accounts
5. Debt to income ratio

Income inequality is a relatively common topic in pop culture such that it is fairly well known that the distribution of annual income does not follow a normal distribution. It is not clear if this will remain the case for the distribution of annual incomes of loan borrowers. It is questions like these that motivated me to investigate the distributions of the various quantities. Specifically, using the Kolmogorov-Smirnov test I attempt to classify the different distributions that occur in the data set. Another motivation for this investigation is the balance of the various feature data. We can look at the percentages of the total number of samples that each feature's mode represents to get a better idea of the imbalances.

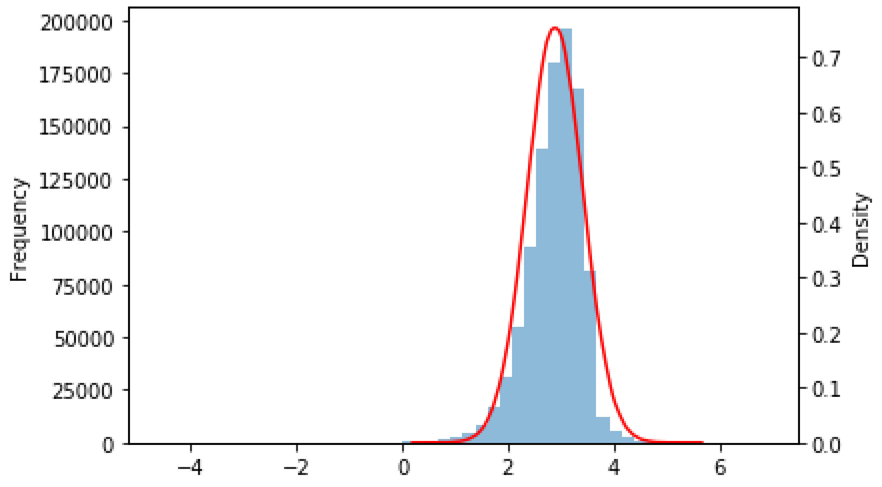
Another quantity previously reported was the utilization of credit; I believed that it



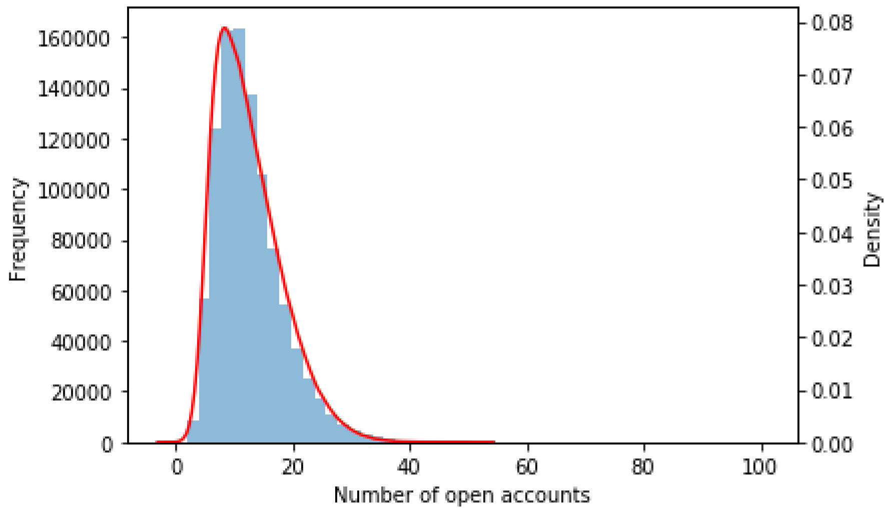
followed a normal distribution but it appears upon further analysis that my claim is that it follows a log-normal distribution.



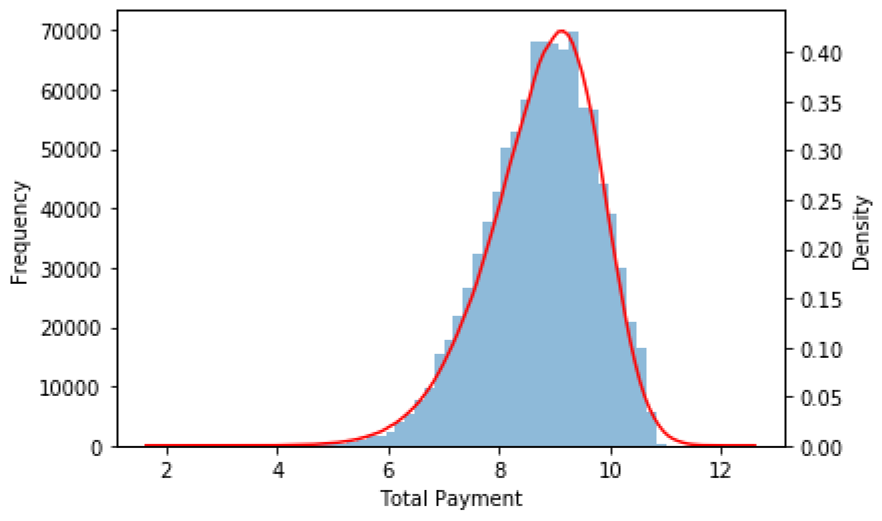
The debt to income ratio is yet another quantity which is close to exponentially distributed has a skewness that says otherwise. Therefore, instead I took the logarithm of the distribution such as to use a skew-normal model for the logarithm.An alternative model for an exponentially distributed random variable Y is to take the log and model it with either a skew-normal or normal distribution. This is likely a better choice because the skewness of the variates is not what would be expected from an exponential distribution, where it is a constant value independent of the commonly used exponential distribution parameter. In order to apply the logarithm, however, the values equal to zero have to be removed to avoid transformed values of negative infinity; this is throwing away data from the sample distribution which is ill motivated. Therefore there is a trade off between accuracy and sample size when fitting models to data of this type.



The number of open credit lines, a discrete valued quantity, nearly looks continuous when



plotted as a histogram with bin width greater than or equal to one. This presents an interesting test; given a discrete variable how well can it be modeled with a continuous distribution, taking only the integer part when sampling?



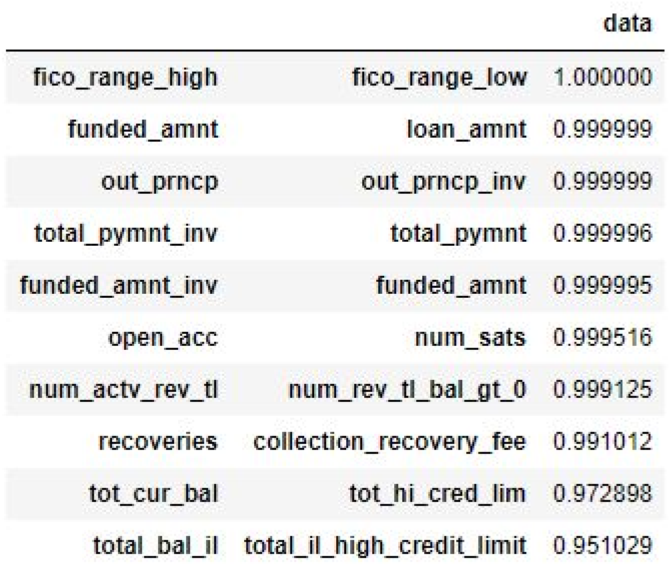
The Kolmogorov-Smirnov test compares cumulative distribution functions of sample and reference distribution (one-sample) or compares the distributions of two-samples.

To claim that the approximated kernel density estimates for the distributions are accurate depictions of our sampling distributions, statistical testing is required. This could come in many different flavors but because I mainly want to compare sample to predicted distributions I will employ the Kolmogorov-Smirnov test to the sets of variates created above.

Quantity	Predicted Distribution	P-value from K-S test
Annual Income	Exponential	1.1840154413373579e-05
Log(Total Payment)	Log-normal	9.305272930319027e-09
Log(Total Payment)	Skew-normal	0.012102373785575339
Utilized Credit Percentage	Log-normal	0.00
Active account number	Skew-normal	3.942810987345747e-29
Log(Debt-to-income ratio)	Skew-normal	3.13097821399214e-23

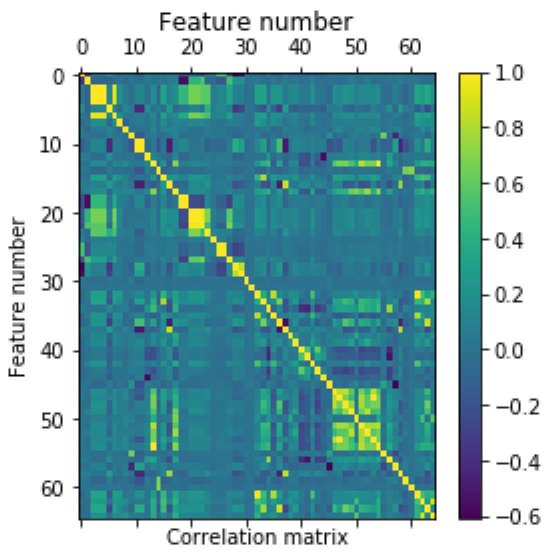
In each of the following, the null hypothesis is that the loan data in question is distributed in the way we claim. The alternative hypothesis is that it is not distributed in this way. More precisely, because I am using the Kolmogorov-Smirnov test, the hypothesis testing involves the CDF's not the PDF's. Therefore if the p-value of the hypothesis tests are less than

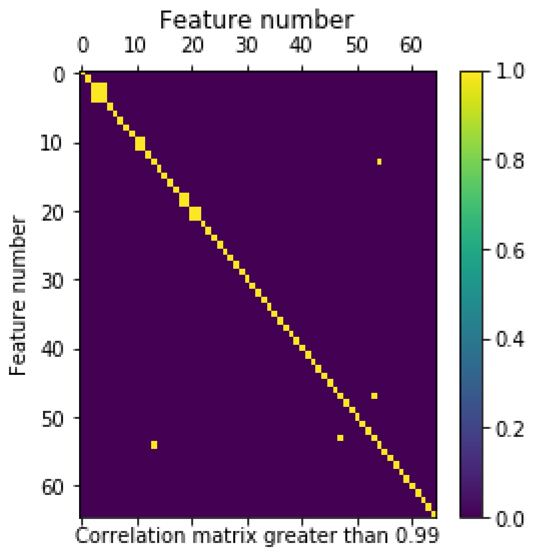
0.05 then the null hypothesis is rejected with 95% confidence. In other words, the CDFs are not equal to our claim. The only claim that has any merit whatsoever is the logarithm of the total payments being a skew normal distribution. It seems that the kernel density estimate plots are more misleading than I thought.



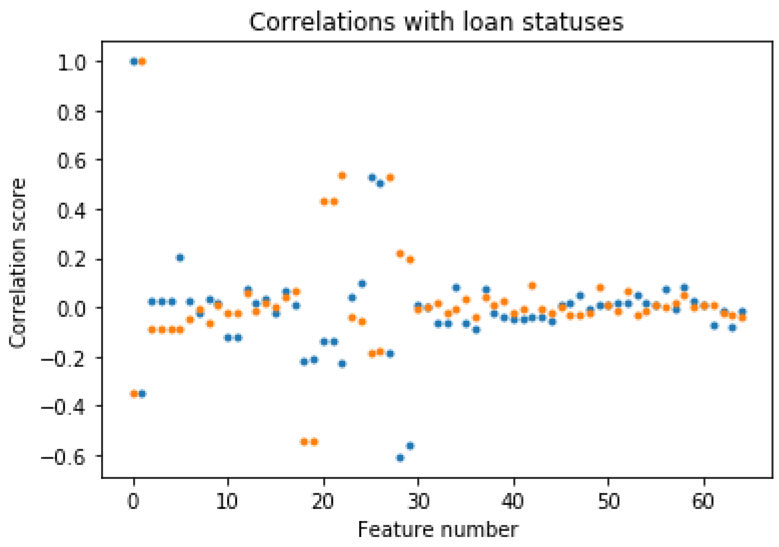
		data
fico_range_high	fico_range_low	1.000000
funded_amnt	loan_amnt	0.999999
out_prncp	out_prncp_inv	0.999999
total_pymnt_inv	total_pymnt	0.999996
funded_amnt_inv	funded_amnt	0.999995
open_acc	num_sats	0.999516
num_actv_rev_tl	num_rev_tl_bal_gt_0	0.999125
recoveries	collection_recovery_fee	0.991012
tot_cur_bal	tot_hi_cred_lim	0.972898
total_bal_il	total_il_high_credit_limit	0.951029

There are a number of pairs of features with pearson correlation scores greater than 0.999 for a specific and relatively obvious reason. Specifically, some features are essentially identical; an example being: the funded amount of a loan and funded aamount of a loan from investors. If investors represent the overwhelming majority of loan funding then these features are nearly identical which seems to be the case upon inspection. I decided to not include the very highly correlated features as they should contribute little to the modeling process other than computation time. The correlations in their totality can be visualized by plotting the correlation matrix as an color coded image.



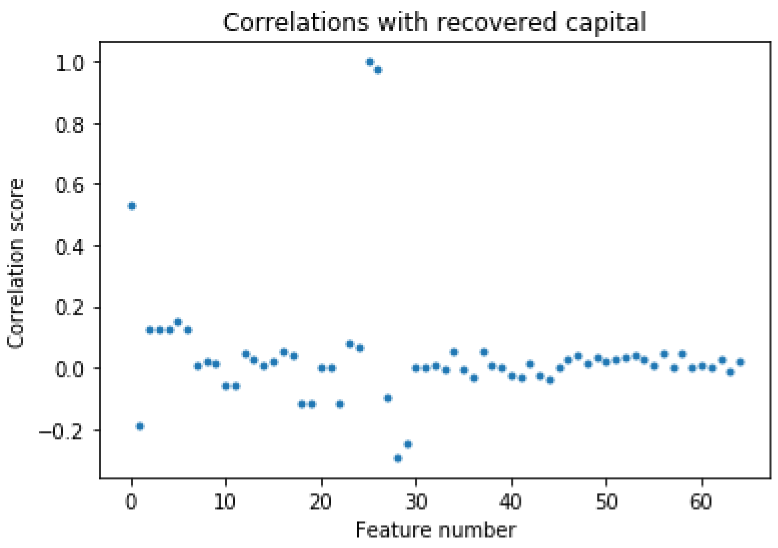


Likewise, by filtering out all values less than a threshold, chosen here to be a value of 0.99, the highest correlation pairs are demonstrated. Note that the matrix is symmetric, and so unique pairs are represented by choosing the values either above or below the diagonal. Because the main targets are the loan status and the recovery amounts, it is prudent to investigate the correlations with these variables. Because the loan status is a categorical variable, it would have to be encoded in order for this to be well defined. Pandas has one-hot encoding via a function "get_dummies" which allows for one-hot encoding. Using this to convert the categorical data to numerical, I could then compute the correlations of this with the other numerical features. As can be seen, other than the autocorrelations, the remainder of the features have correlation scores between -0.6 and 0.6, approximately. This is sufficient (to me) to not have to drop any of the



other numerical features before the modeling process.

Likewise, the same can be computed for the recovered amount of capital. The largest correlation is between the recovered amount and the collection recovery fee; that is, the money that is recovered the more must be paid for its recovery. Therefore, in order to avoid a biased model, I drop the collection recovery fee data from the feature data.With this knowledge I believed that I was ready to begin the modeling process, but it turned out that there were still a number of



considerations that needed to be made.

4. Model creation and in-depth analysis

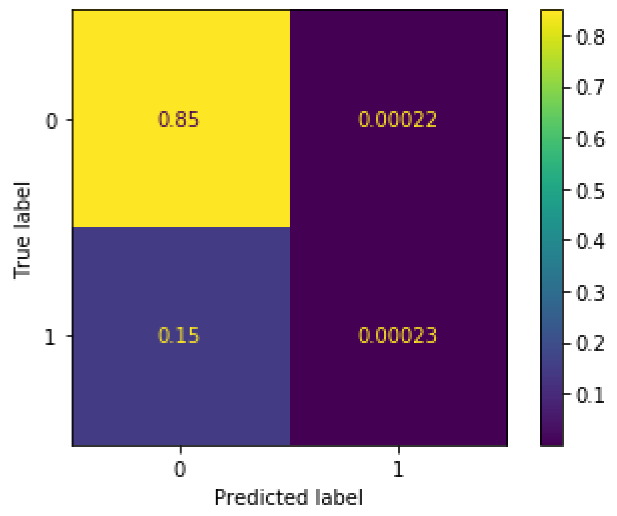
For the loan outcome prediction portion of this project, only the loans which have both matured are considered. Additionally, in this subset of loans there are a number of outcomes but by far the most highly populated are loans which have been fully paid or charged off. This can be formulated as a binary classification problem: the loans that are charged-off
will be denoted as "successes" or by the integer 1. Likewise, loans which have been fully paid off will be denoted as "failure" by assigning these the integer 0. The first main hurdle that we need to overcome is the time dependent nature of the problem. If not handled carefully, we could accidentally perform what is known as "data snooping", which is the contamination of the model by inclusion of information from the future. The two main effects that this time dependency has are Multiple time dependent variables and time ordered cross validation folds. To account for the first of these issues, all variables that are recorded at times more recent than the date that the loan was issued are removed. I could tell which features these were because their descriptions typically contained descriptive statements such as "in the past twelve months". Some of the features are ambiguous, however, and so I lean on the side of caution by removing these features as well. By including only loans which have matured, the correct description that these models produce is predict whether a loan will default by the maturity date. The maturity date depends on the term of the loan; it might be wise to separate the loans by term but currently this is not done. The cross validation process as well as the folds will be customly made, instead
of using scikit-learn's TimeSeriesSplit() for instance. This is to be as confident as possible that there is no data snooping. Additionally, there are some considerations regarding encoding and preprocessing that are best handled by a custom cross-validation procedure.
Have the cross validation folds I now needed to create a procedure which correctly preprocesses them before testing. I made the folds such that the number of training samples is cumulative over time. This attempts to reflect the collection of data over time. I also ensure that I created a "hold-out" set of data that will used for final predictions and analysis after all cross-validation and model learning has been accomplished. In order to split the loan data into meaningful folds The loan issuance dates are aggregated by month, but from the metadata we know that the data is reported quarterly. Using this as motivation, the hold-out data will be the most recent quarter. Because the number of loans has grown over time, the last quarter represents one seventh of all of the data. Therefore I deemed that this quarterly fold creation is too imbalanced to be practical. After the folds were created, I needed to account for the two different data types in the feature data, numerical (float) and categorical (object, could be cast as string). The categorical variables will be encoded and represented by binary numerical columns via what is know as one-hot encoding. Practically speaking it gets more detailed; some of the categorical variables have many categories, of the order of a thousand.The most diverse categorical variables will be handled in one of two ways: we shall argue that they are either unnecessary or reduce the number of categories via binning.

The following are the choices made for these variables: originally the choice was made to remove all but the first two digits of the zip_code (there are only three recorded). This in fact lowers the resolution to a point where the state of residence is more precise; therefore, I believe that the zip_code can be dropped as it isn't informative. For the categorical variables which may be cast as datetime variables, **['earliest credit_line', 'issue_d']** the number of categories will be reduces by application of a binning strategy. The binning for the issuance date is naturally handled when the cross-validation folds are being produced, as this is the determining time dependent quantity. More specifically, we drop the issuance date from the calculation because based on the folds, the training and testing data will never intersect by definition; which I believe implies that the model will not know what to do. For the earliest_credit_line dates it is less obvious as to how to proceed. The earliest credit line is thought of to be a proxy for age groups. Depending on the context, demographic age groups can be quite large in terms of the age spread. Therefore, I believe that only creating a few bins for this variable is sufficient. Applying the same argument comparing zip_code and addr_state, I remove the 'grade' of the loan in favor of keeping the subgrade. The grades take the values such as 'A', 'B', 'C' while the sub-grade is more specific 'A1', 'A2', ... 'E4', 'E5'.

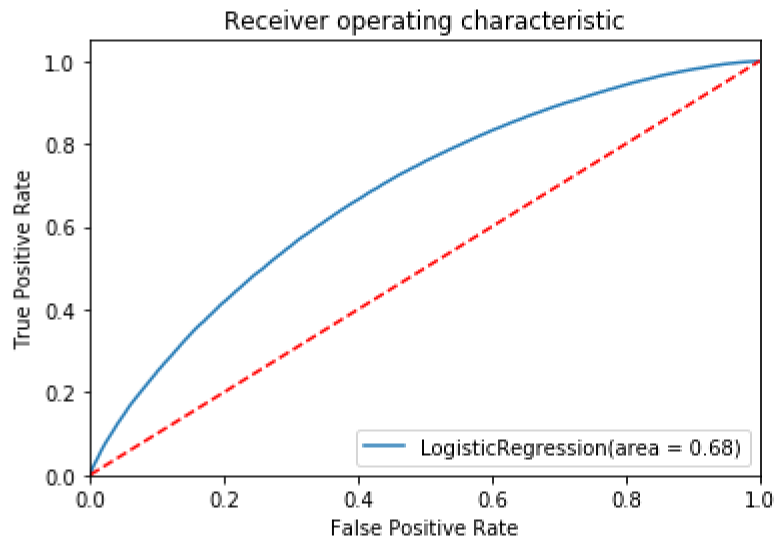
Therefore before the modeling process the following steps shall be taken:

1. Remove target variable from feature data
2. Remove more coarse categorical variables
3. Encode datetime-like categorical variables with KBinsDiscretizer (OneHotEncoding strategy).
4. Encode remaining categorical variables with OneHotEncoder (scitkit-learn)

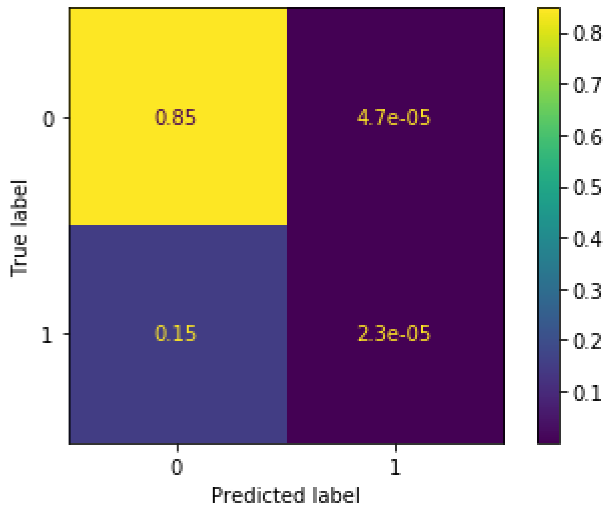
Note: To avoid collinearity in the one-hot encoding process, the category with the lowest frequency is dropped (for each feature). The first test of these methods was to develop an intuition as to what to include in the cross-validation process. To model, the data of course needed to be processed as previously described. There are a number of choices for the method with which to scale the numerical data. The tests are performed by using the entirety of the training data (named traintest because the testing folds come from this subset) to fit the model. The model is then tested using this data as well as a set of "holdout" data which the model does not know about. The holdout data, by virtue of the cross-validation folds, corresponds to the most recent loan data. The preliminary tests that were run were naive modeling using only the default hyperparameters of the two classifiers under investigation. What I show for each model is the confusion matrix and the ROC-AUC curves for each model.



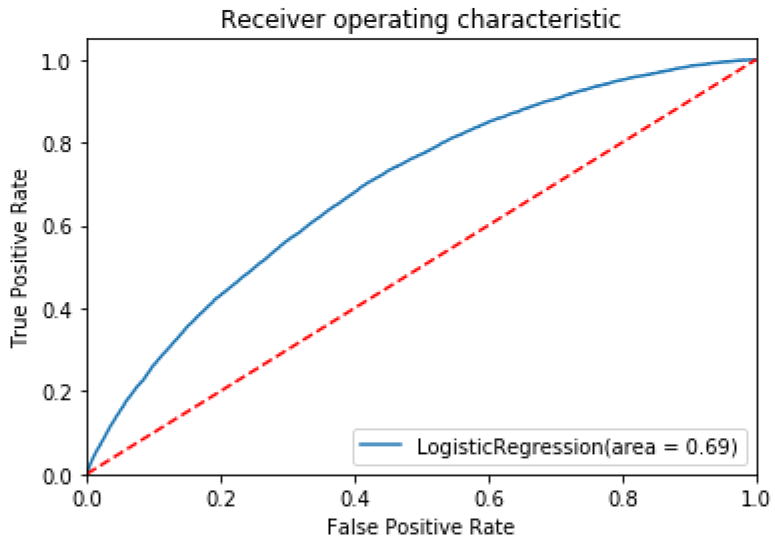
Confusion matrix for logistic regression where training = testing set (null test)



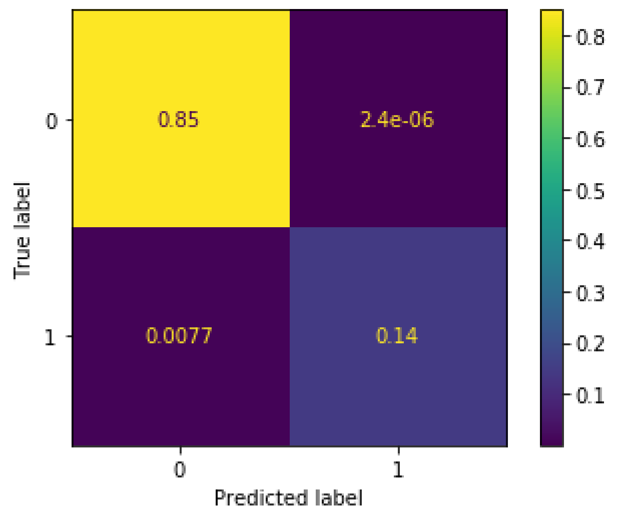
ROC-AUC curve for logistic regression where training = testing set (null test)



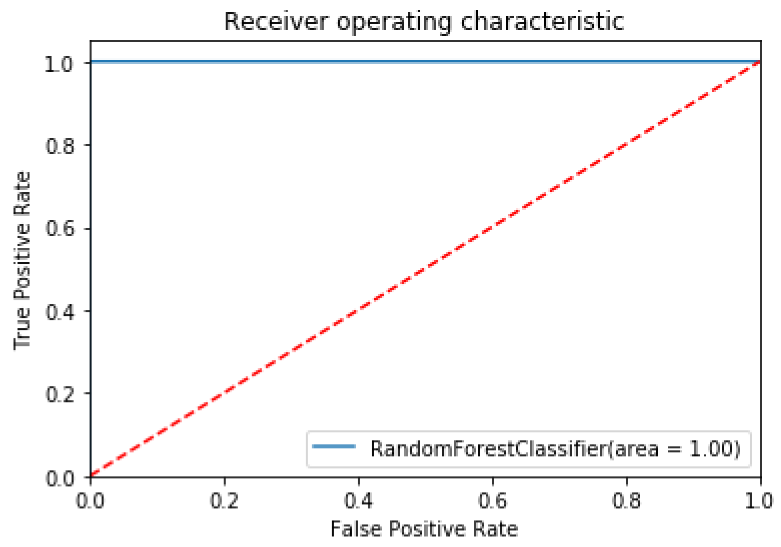
Confusion matrix for logistic regression by using different training and testing data



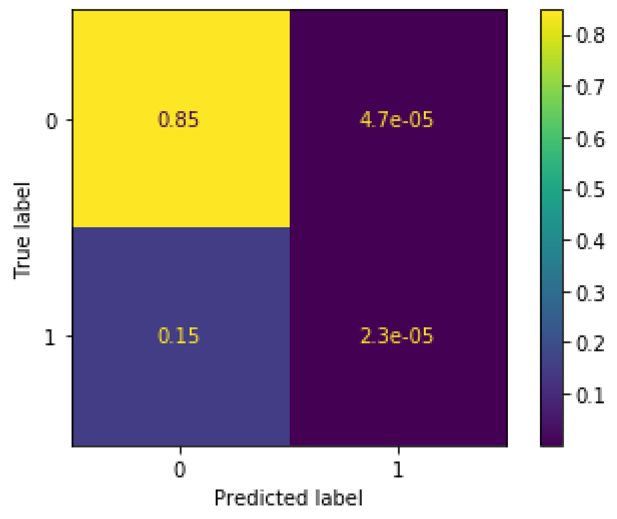
ROC-AUC curve for logistic regression by using different training and testing data



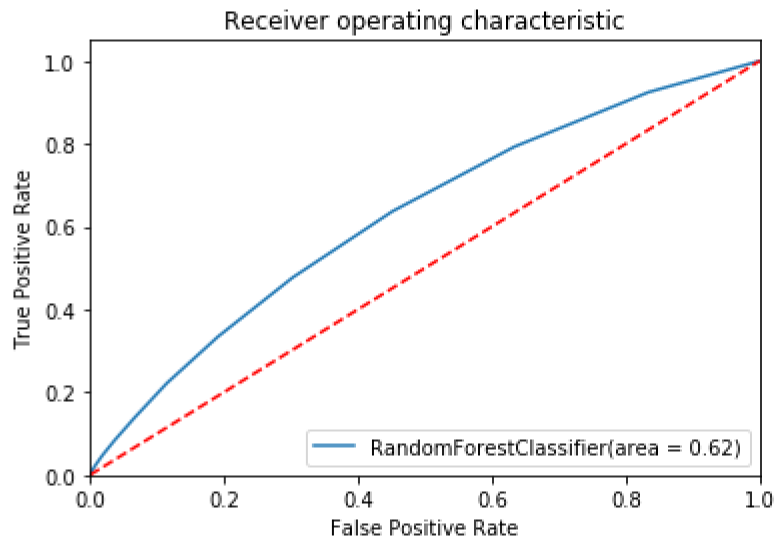
Confusion matrix for random forest where training = testing set (null test)



ROC-AUC curve for random forest where training = testing set (null test)



Confusion matrix for random forest by using different training and testing data

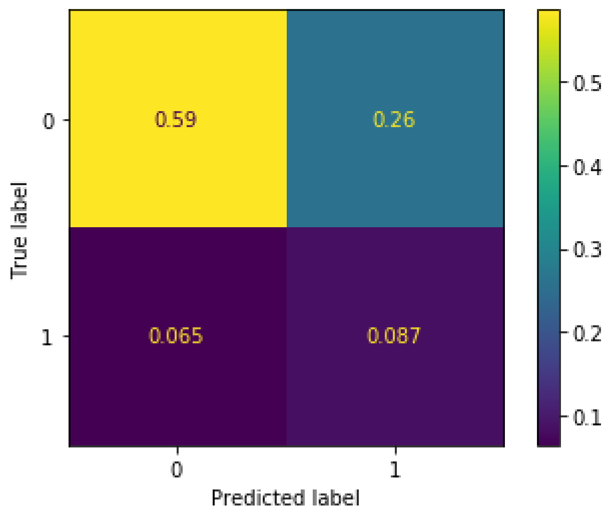


ROC-AUC curve for random forest by using different training and testing data

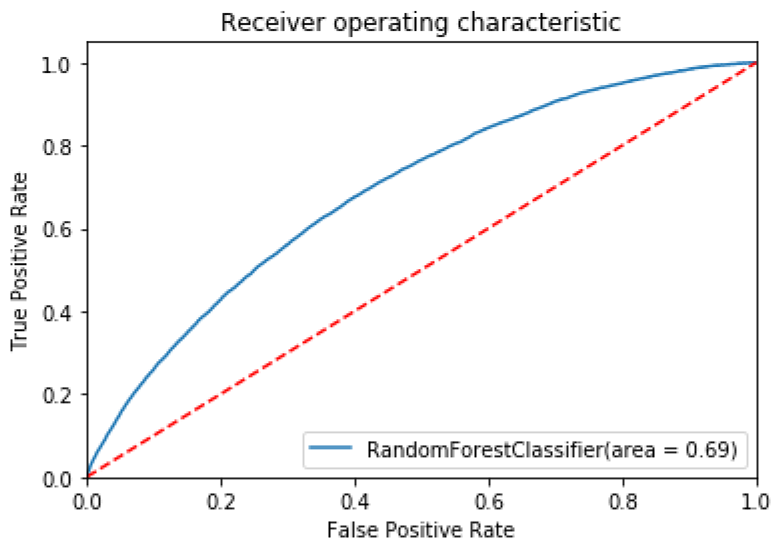
For me there are two main takeaways from these tests. Firstly, the class weights need to be balanced to represent the frequency at which defaults occur (approximately 1/7 of the total samples). Second, it is easy to overtrain the random forest classifier, as indicated by the tremendous difference in performance between training and testing data. The main goal is to reduce capital loss not maximize profits. Therefore, we value prediction of when a loan will be charged off more than fully paid. We can account for this by changing the class weights in the classification process. This model will reject loans that would have been fully paid in order to avoid loans that will become charged off. In other words, the goal is to maximize the number of true positives, where "positive" in this case is equivalent to a loan being charged off. It's clear that the class weights need to be balanced because otherwise the models are modeling the null

information rate. Therefore, for everything that proceeds I shall weight the binary classes by their inverse frequency. The choices for the hyper-parameters aren't obvious so the cross-validation process attempts to cast a broad (but coarse) net
on the hyper-parameter space. For the random forest classification, the hyperparameters are not very intuitive to me, so I varied most of them including: the number of trees (estimators), the maximum depth of each tree, the minimum number of samples required to split, the minimum samples required in a leaf. The best results were given by the following set of parameters

```
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight='balanced',
                       criterion='gini', max_depth=27, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=200, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=25,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```



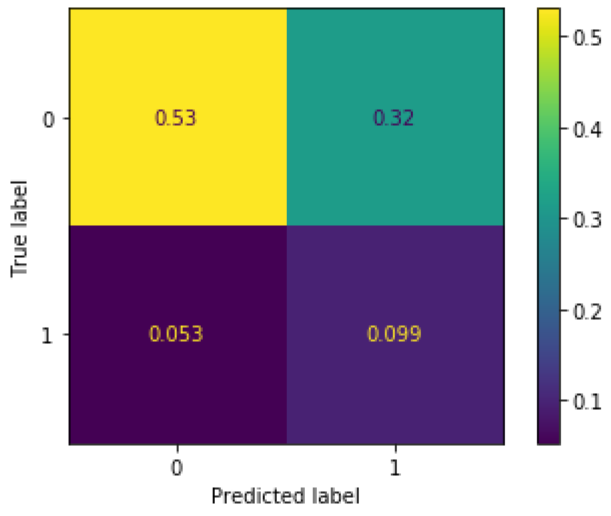
Confusion matrix using cross-validated random forest model using holdout-data set



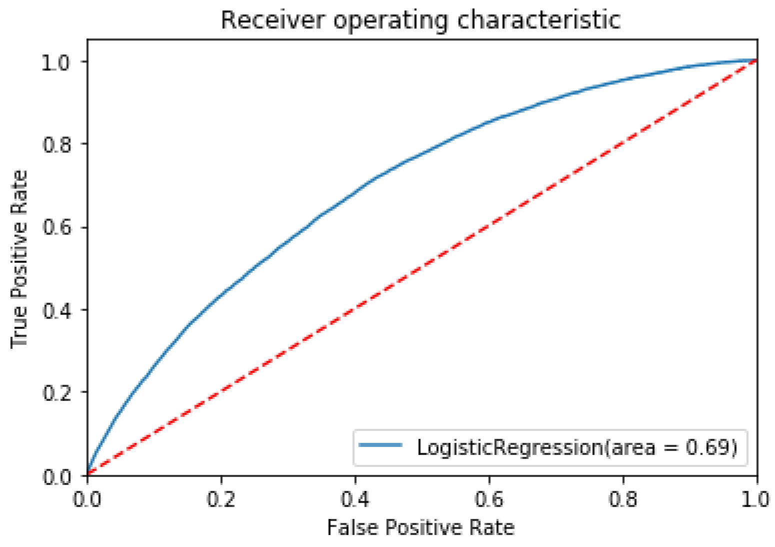
ROC-AUC using cross-validated random forest model using holdout-data set

Likewise, for the logistic regression model I only tested the regularization constant ('C', smaller means stronger regularization) and the tolerance. This resulted in the following model.

```
LogisticRegression(C=0.25, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=500, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```



Confusion matrix using cross-validated logistic regression model using holdout-data set



ROC-AUC using cross-validated logistic regression model using holdout-data set

e comparable performances. The logistic regression model has decent performance in regards to the main goal which is attempting to identify true positives, which it accomplished about 66% of the time. Because there are so many false positives, however, I do not believe that this is sufficiently accurate to determine whether or not to provide loans, as it would reject far too many applicants. Therefore the best that this model can do is to flag accounts which are at risk of defaulting. This is still a proactive measure but it would only be correct around 25% of the time and so it would take an additional investigation to see whether or not this is worth the effort. This is supported by the increase in both recall and precision of the logistic regression.

The deliverable model that is about to be produced is one that predicts the amount of recoverable money from a "charged off" loan. This is performed via two different regression methods, Ridge regression and stochastic gradient descent. There is no time dependent component of the hypothesis in this case; that is, no special time order is required. To produce these two models all data needs to be in a numerical form, which requires the encoding of categorical variables.

The procedure for how we first produce these models is as follows:

1. Take the subset of the data which corresponds to charged-off loans (already done).
2. Perform preprocessing operations (Categorical variable encoding and numerical variable rescaling).
3. Cross-validate a Ridge regression model
4. Cross-validate a StochasticGradientDescent (**SGD)** model
5. Analyze the effectiveness of each model with various metrics.
6. Decide on a final deliverable model.

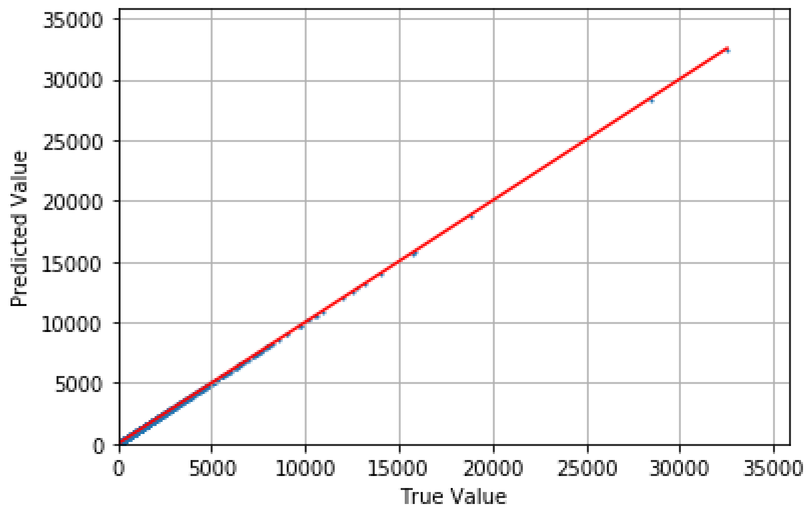
There are many categorical type variables which need to be encoded in order to be used for regression. There are a number of categorical variables with a large (100+) number of categories. In order to better handle this fact there are a number of features which we prune from our analysis.

The prediction is occuring in the present moment. All of the data corresponding to charged-off loans can be used, and it does not have to be treated as a time series, as we are not trying to predict the future. As can be seen, the variables with the second to eight largest number of categories are related to dates; These features, from the cleaning process, are a mixture of dates (month-year) and the category 'Missing'. These variables will be encoded using binning, specifically KBinsDiscretizer using the One-hot encoding strategy with a small number of bins per feature vector. The manner with which we bin is to represent the dates by their year, and then use a uniform bin-width strategy with three bins. Simultaneously, the "Missing" values are one-hot-encoded separately, such that the result is three bins for the date's years and an additional bin for the 'Missing' Category. The largest number of categories belongs to the zip code variable; which is reported as the first three digits (followed by 'xx'). We can reduce the number of categories by aggregating by the first *two* digits; but if we do this it is almost nearly the same as identifying by the state; In fact, rounding to two digits is actually less specific than identifying by the state. We already have the state in 'addr_state' feature data so there is no motivation to keep the zip codes if reducing them to two digits. Additionally, the grade of each loan is less precise that the subgrade of each loan such that we dispose of this as well.

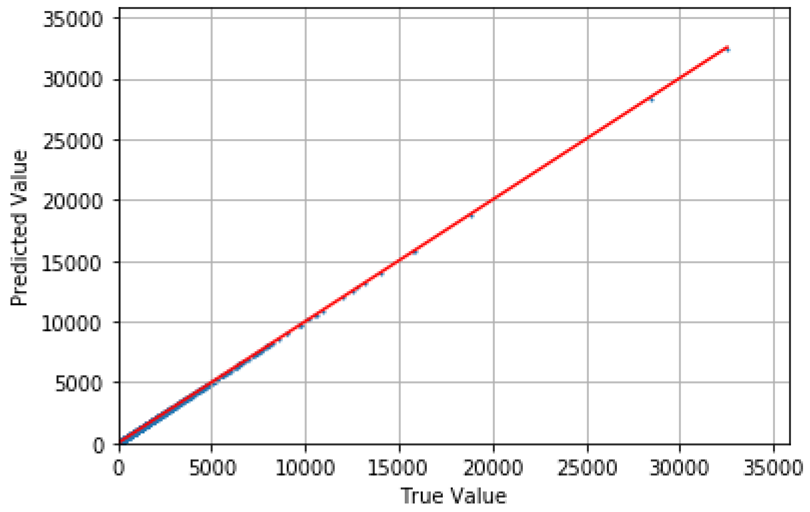
I left these next calculations here for posterity, but it turns out that handling the datetime-like variables in this binning manner introduced a lot of noise into the models.

For each model we make a preliminary test on the training data; this is similar to the no-information rate analysis in classification problems. That is, we train and test with the same data set. If the model is inaccurate in this case then there is something seriously wrong with the data or the model. While there are an incredible number of options for fine tuning, my main investigation is into the efficacy of the numerical feature transformers. I used scitkit learn’s

SGDRegression (stochastic gradient descent) and Ridge regression (regularized linear least squares). The preliminary models performance was too good to be true.

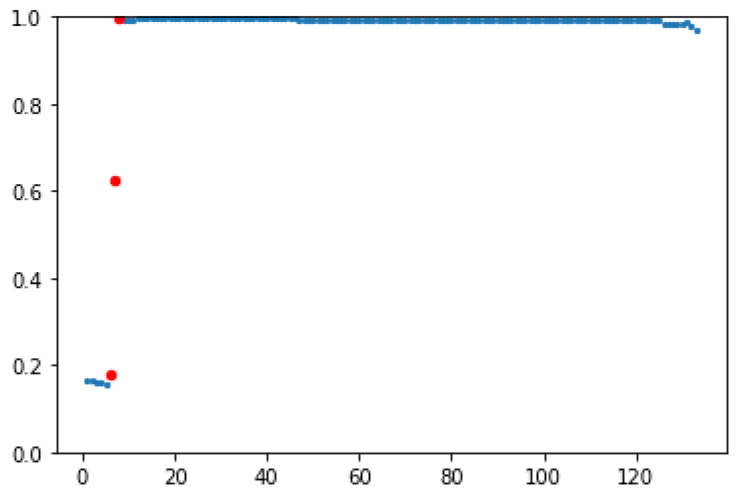


Stochastic gradient regressor: explained variance score 0.999988226779694



Ridge regressor : explained variance score 0.9999959491424831

Note that the reason why these results are so dubious is because they were actually performed on separate train and test sets. It looks as though it was testing versus the training data but again, that is not the case. The explained variance of these initial models is essentially equal to unity. I investigate this peculiarity by a crude yet effective method of performing the same training and testing routine, but adding one feature at a time.



Explained variance (y) vs. the numbers of features included (x)

There is an incredible jump in explained variance that happens as three specific features (the first jump is small in relative terms but significant, represented by red dots). As if to indicate a collinearity between these columns and the target variable. I do not believe that there is any contamination between these features but at the same time they seem to perform too well as predictive variables. According to the metadata, the definitions of the three features are:

1. total_pymnt : Payments received to date for total amount funded

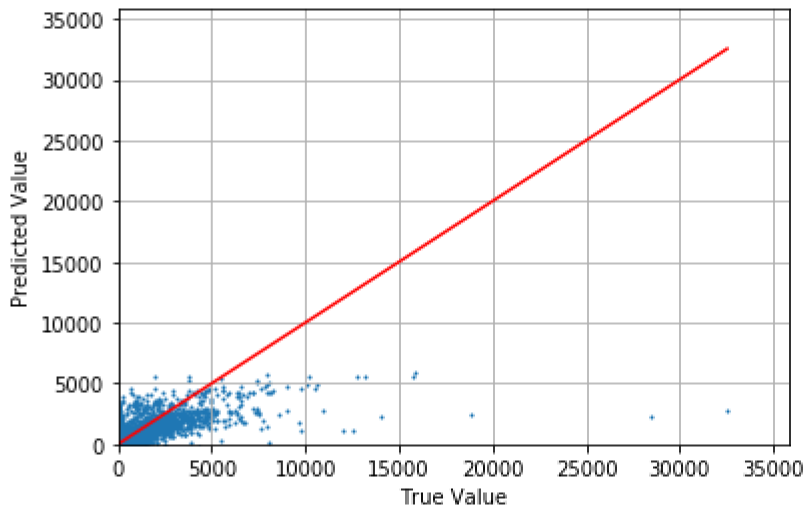
2. total_rec_prncp : Principal received to date

3. total_rec_int : Interest received to date

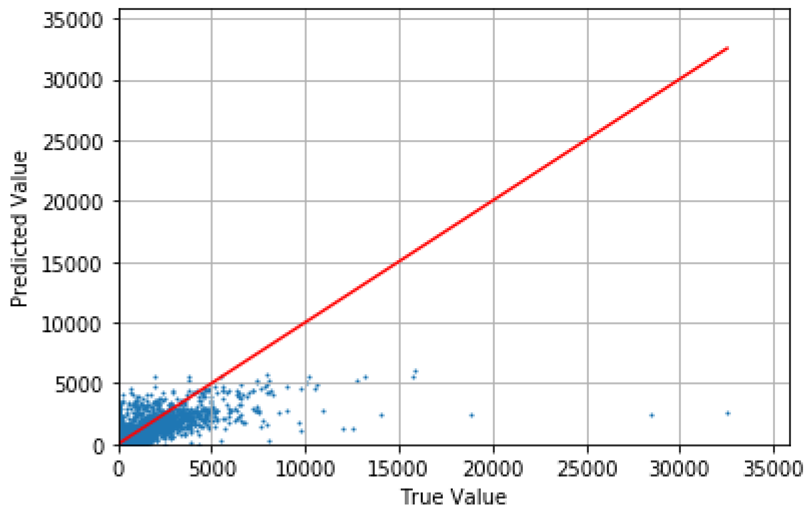
And the target variable:

recoveries : post charge off gross recovery

Therefore my recommendation would be to uncover a more detailed explanation of the relation between these features and the target variable. Until then, a cautious application of this model is recommended. Unfortunately the performance in the absence of these three features is pretty terrible, as can be seen by the models without these three features. The explained variance drops from near unity to values approximately equal to 0.25.



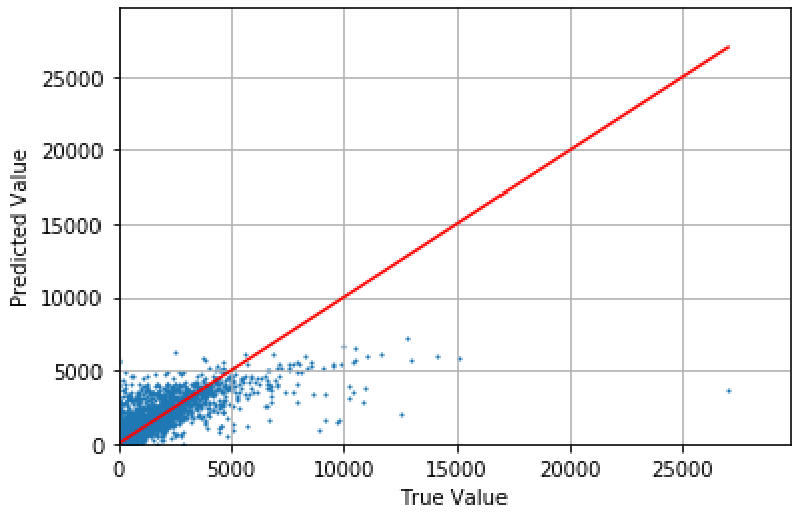
SGD Regressor (minus three features) explained variance score :



Ridge Regressor (minus three features) explained variance score :

Note that this isn't a time dependent problem so a possible is that total payments perhaps include post charge off gross recovery. and there does not seem to be any glaring relations between these three variables. It may be that these features are simply very good descriptors. With no way to know or follow up, I continue towards the real modeling procedure.

One way of improving the models that exclude the suspicious features was to subset the data even further, only including charged off loans which have non-zero recovery amounts. This idea originated from the fact that the distribution of recovery values has a mode equal to zero. By including this measure, the explained variance score increased to around 0.47.



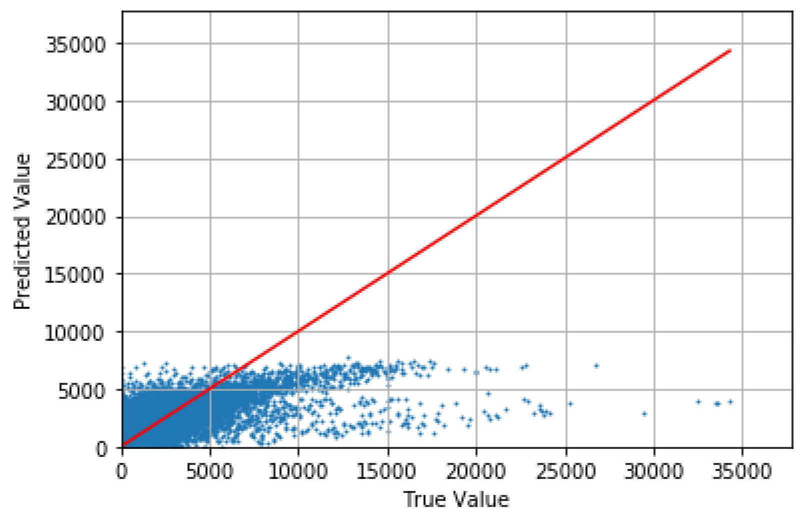
The main cross validation process tested sets of parameters as well as different column transformers that acted on the numerical feature data. This is of course in addition to the two different models being used as mentioned earlier. In summary the feature data column transformers tested included: QuantileTransformer(), MinMaxScaler(), and StandardScaler(). The models tested were SGDRegressor() and Ridge() from scikit learn, the hyperparameters tested included the tolerance and regularization constants in both models. In every case the Ridge regression outperforms the stochastic gradient descent. The final ridge regression model had parameters

Ridge(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=None, normalize=False, random_state=None, solver='auto', tol=0.1)

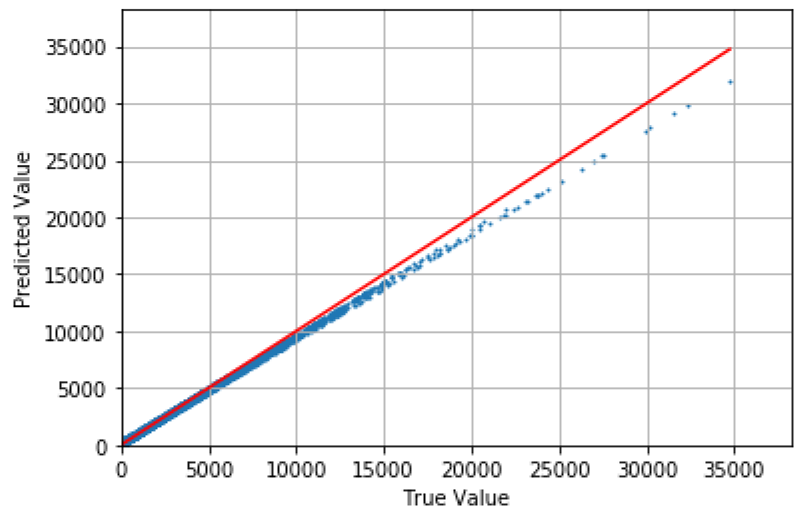
And the final results on the different numerical transformers were

Transformer	Mean squared error	ExplainedVariance
QuantileTransformer	2278262.7349281064	0.4270508478802073
MinMaxScaler	2205599.158832141	0.44532419977748505
StandardScaler	2381178.413802254	0.40166608575310014

Such that for the final models, with and without the three dubious features, using MinMaxScaler() provided results given by the two following figures.



Final ridge regression model (minus 3 features): Explained variance score 0.46042358045918



Final ridge regression model (including three dubious features) : Explained variance score 0.99599738611

The above plots represent results from scaling with MinMaxScaler on the feature data.

1. The prediction vs. true values for ridge regression without the suspicious features
2. The prediction vs. true values for ridge regression with the suspicious features

It's quite clear that the inclusion of all of the features performs dramatically better,

therefore the course of action would be to discover whether or not the inclusion of these features is valid. This is outside of my power, however, so my investigation ends here.
In summary, Ridge regression with small regularization (larger value = smaller regularization per scikit-learn's docs) but a relatively strict tolerance seems to perform the best. The prediction for the recovered amount of capital seems the always be smaller than the true value, which I believe is better.

6. Future work

This project attempted to create a two-stage recommendation system which included proactive and reactive measures for capital recovery from defaulted loans. The performance of the first stage was subpar and the performance of the second stage was highly dependent on a connection between data features that could not be known with the current level of information. In light of this analysis, I would not recommend using these models as I had originally intended. The first stage could at best be used as a system to flag loans which may be at risk of defaulting so that proactive measures could be taken. The second stage could be used if it turns out that the data is not contaminated as the unrealistic performance seems to indicate.