Loan outcome and potential recovery prediction
Capstone 1 : In-depth analysis
Matthew Gudorf

1. Feature selection and engineering

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 predictive model is then only predicting whether or not a loan will default by its 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.

At this point 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 one-hot encoding. Some of the categorical variables have many categories, of the order of a thousand, which complicates things.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, 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.

2. Cross validation setup

The cross validation process as well as the folds will be customly made, instead of using scikit-learn's TimeSeriesSplit() for instance. This to be absolutely sure that the time ordering is respected. Additionally, there are some considerations regarding encoding and preprocessing that are best handled by a custom cross-validation procedure.

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 be 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 was originally to 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, after processing and handling pathological values. Therefore I deemed that this quarterly fold creation is too imbalanced to be practical. With these cross validation folds, the next step in the modeling process is to create a pre-processing pipeline which handles and caresses the data such that it is properly positioned to be modeled with.

3. Loan outcome prediction models

Use only training data to see model performance as a preliminary to the full modeling process. 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)
```

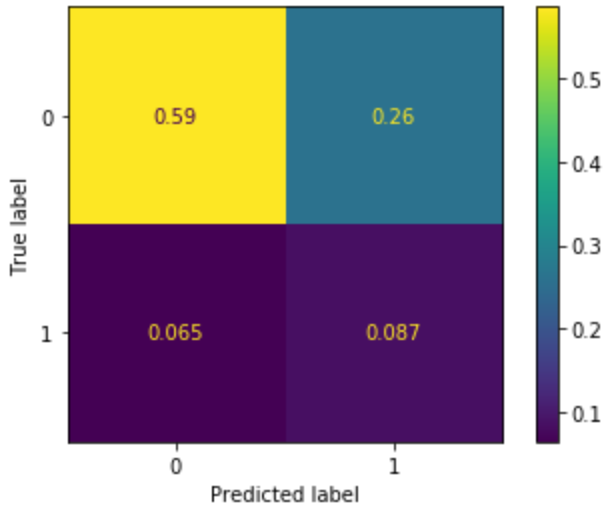

Figure 9. Confusion matrix using cross-validated random forest model using holdout-data set

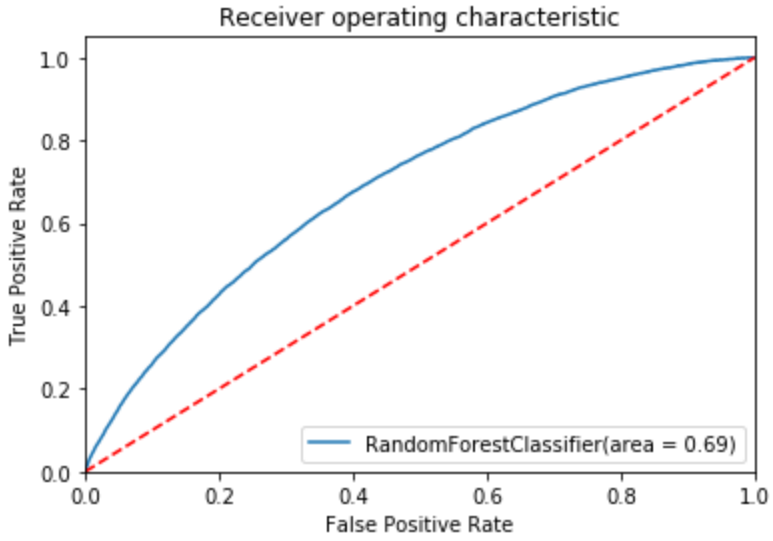

Figure 10. ROC-AUC using cross-validated random forest model using holdout-data set

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)

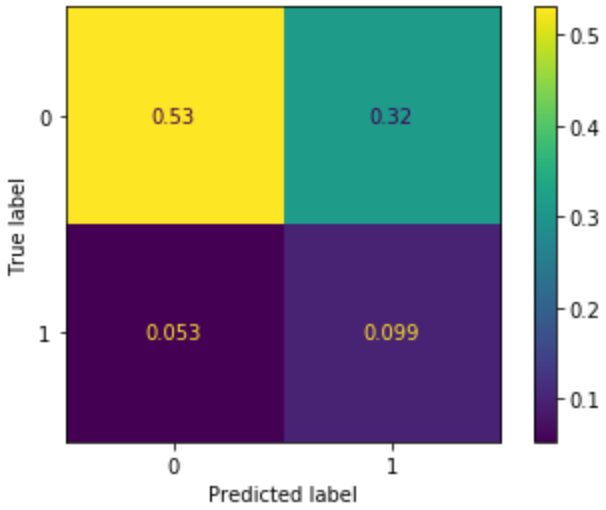

Figure 11. Confusion matrix using cross-validated logistic regression model using holdout-data set

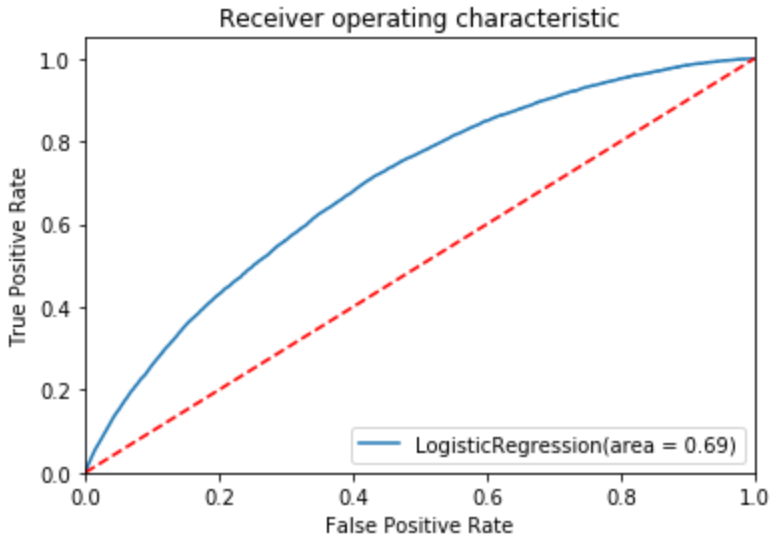

Figure 12. ROC-AUC using cross-validated logistic regression model using holdout-data set

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.

4. Loan recovery prediction models

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 described by the following enumerated list.

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 scikit-learn's SGDRegression (stochastic gradient descent) and Ridge regression (regularized linear least squares). The performance on the training set was perhaps too good; an indication of overfitting, but alas we press on. 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.

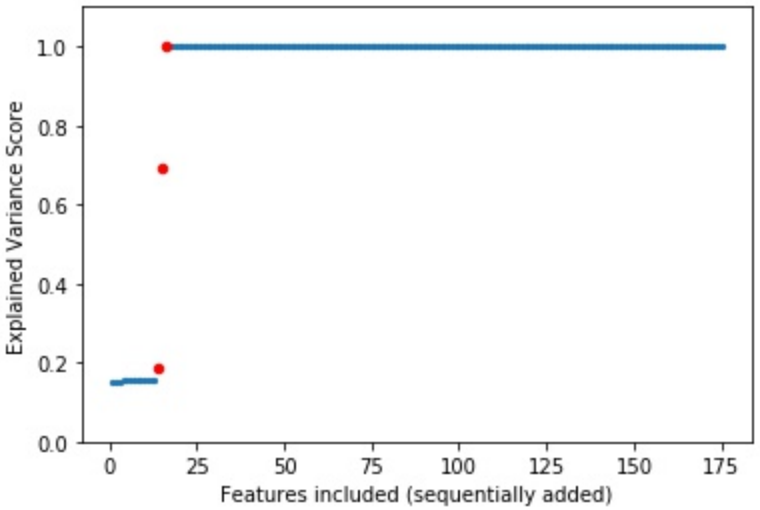

Figure 15. Dependence of explained variance on the number of included features (not randomized).

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.

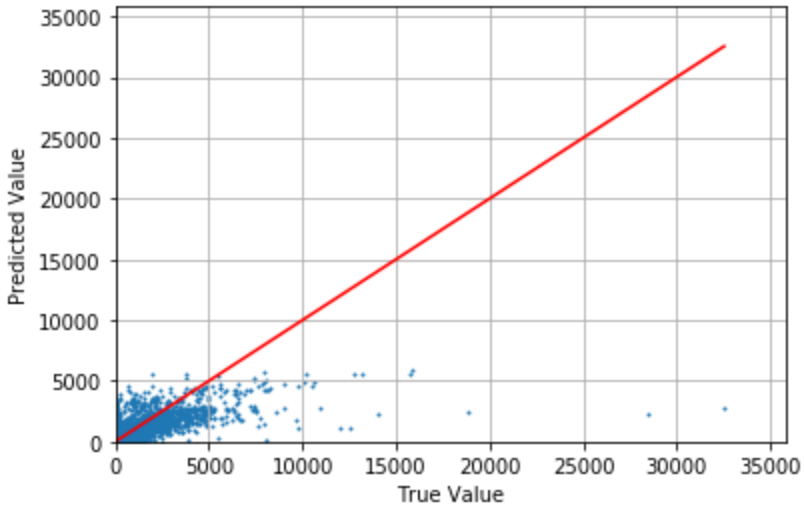

Figure 16. SGD Regressor (minus three features) explained variance score :

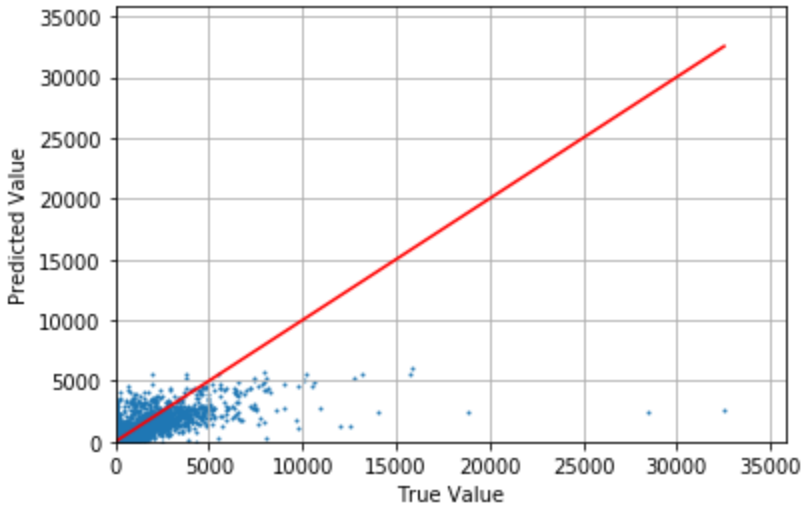

Figure 17. Ridge Regressor (minus three features) explained variance score :

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.

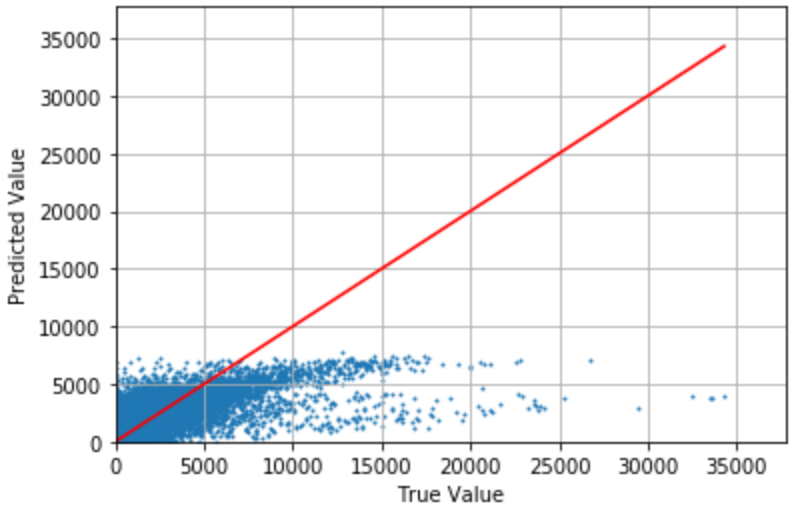

Figure 18. Final ridge regression model (minus 3 features): Explained variance score 0.46042358045918

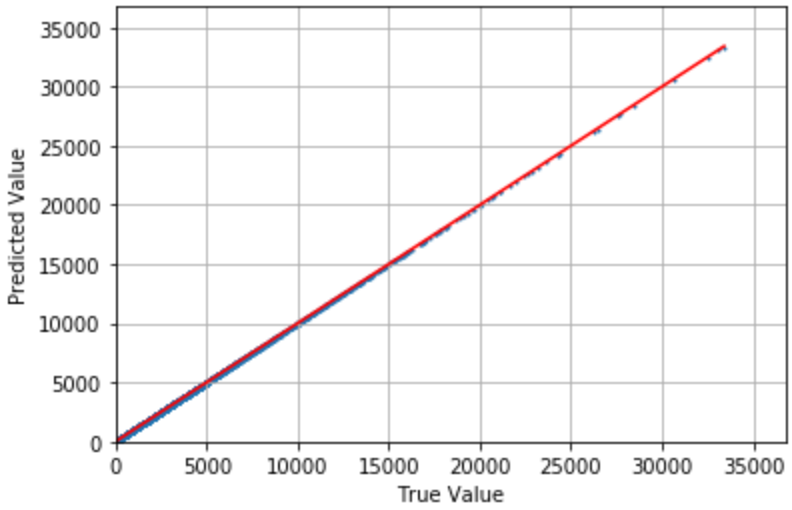

Figure 19. Final ridge regression model (including three "dubious" features) : Explained variance score 0.99999738611

The above plots represent results from scaling with MinMaxScaler on the feature data. 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. 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; even though the model appears to have high variance (overfitting).

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.