MODEL DEVELOPMENT DOCUMENTATION

Christopher Jose | MSDS 498, Section 57

INTRODUCTION

Predicting default-likely creditees is a necessity among creditors. Due to the advancement of machine learning and technology, this feat is now common practice. Through research carried out by Chung Hua University, a dataset of customer loan status information for a bank has been made available to the public as of 01/26/2016.

This paper outlines the methodology used for building a machine learning model that can predict whether default payments will occur for a customer within the dataset and for future customers with similar data points. The final model was selected through processes called hyperparameter tuning and algorithm selection, which maximized accuracy, the number of correct predictions over total predictions. It was not assumed that another metric, such as precision or recall, more closely corresponds to potential model usage purposes. All analysis was conducted using the R programming language.

Each stage in the machine learning pipeline is reviewed: data review, data quality check, preprocessing, feature engineering, exploratory data analysis, hyperparameter tuning / model selection, algorithm selection, final performance estimation, and room for improvement.

DATA REVIEW

The dataset was taken from the UCI Machine Learning Repository. It contains 30,000 instances of credit card holder information from a major bank in Taiwan. Information includes demographics, credit card payment information, and a binary outcome variable of whether or not default payments were made by the bank (Yes=1, No=0). Data is for each month between April 2005 and September 2005. Data includes 23 initial predictors, including consumer demographics, original amount of provided credit, repayment status, current balance, and previous payment amounts. Observe the Data Dictionary for variable definitions.

DATA DICTIONARY

Table 1

ID	ID of each client
LIMIT_BAL	Amount of given credit in NT dollars (includes individual and family/supplementary credit
SEX	Gender (1=male, 2=female)
EDUCATION	(1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
MARRIAGE	Marital status (1=married, 2=single, 3=others)
AGE	Age in years
PAY_0	Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)
PAY_ j	Repayment status in jth latest month, 2005 (j=2 August, j=3 July, … , j=6 April)
BILL_AMT j	Amount of bill statement in jth latest month, 2005 (j=1 September, j=2 August, j=3 July, … , j=6 April)
PAY_AMT j	Amount of previous payment in jth latest month, 2005 (j=1 September, j=2 August, j=3 July, … , j=6 April)
UTILIZATION j	BILL_AMT j / LIMIT_BAL
BALANCE_ VELOCITY_ j	UTILIZATION_ j / UTILIZATION_ j+1
PAYMENT_RATIO_ j	PAY_AMT j+1 / BILL _ AMT j
default.payment.next. month	Default payment (1=yes, 0=no)

DATA QUALITY

Variable	Comment
LIMIT_BAL	(Min = 10,000, max = 1,000,000) which is reasonable
SEX	Always has value 1 or 2, which is reasonable
EDUCATION	Has “0”, which is not indicated in the data dictionary
MARRIAGE	Has “0”, which is not indicated in the data dictionary
AGE	(Min = 21, max = 79) which is reasonable
PAY_ j	Has values “-2” and “0”; data dictionary shows [-1,1,… 9]
BILL_AMTj	Has negative values, which is possible if creditee overpaid
PAY_AMTj	Lowest value is a min of 0, which means no refunded payments here
Default.payment.next.month	Has two values, which is reasonable
EDUCATION	Zero values were remapped to value 6 (unknown)
MARRIAGE	Zero values were remapped to value 3 (unknown)
PAY_ j	-2 an 0 values were left alone

There are cases of the data appearing to be incorrectly coded. For example,

a. A payment of $60,000 was made prior to the month in which the loan balance went up to $60,000. Either the payment or the balance is likely off by a month for this loan.
b. An august bill amount of -$13,543 went down further to -$14,386 in September. Some bill amounts are negative.

1. I did not adjust these instances or other similar data inconsistencies. I only adjusted engineered features.
2. There were no missing values in the dataset.

TRAIN – TEST – VALIDATION SPLITS

Instances have been placed into train, test, and validation splits as follows:

Train	Test	Validation
15,180	7,323	7,497

FEATURE ENGINEERING

Feature engineering was performed prior to splitting the data into train, test, and validation splits. There was no information leakage from the test/validation splits into the train split, however. Each constructed feature value for an instance only contains information from that instance, and not other instances (where a given instance could be in either of the three splits). The following variables have been constructed as features:

UTILIZATION_ j
BILL_AMT j / LIMIT_BAL
The percentage of original credit that still needs to be paid off during the jth most recent month (j=1 is September, …, j=6 is April).

BALANCE_VELOCITY_ j
UTILIZATION_ j / UTILIZATION_ j+1
(1+%) in the utilization between the jth and j+1st most recent months. This effectively measures the (1+%) change in the monthly balance (BILL_AMT j / BILL_AMT j+1).
For BILL_AMT j+1 <= 0, BALANCE_VELOCITY_ j was set.
For BILL_AMT j <=0, BALANCE_VELOCITY_ j was set.

PAYMENT_RATIO_ j
PAY_AMT j+1 / BILL _ AMT j
Payment in month j as a % of the monthly balance in month j.
PAY_AMT j+1 shows the prior month's payment, meaning month j's payment amount.
There is no payment ratio value for September, since PAY_AMT does not have value for September.

For a month with a payment greater than the balance amount (like 0), the payment ratio was set equal to 1. These cases appear to represent incorrectly coded data, since it makes no sense for a payment to be greater than what is owed. I assume the payment completely satisfies the loan balance, and so I set the payment ratio to 1.

DATA QUALITY – ENGINEERED FEATURES

Data summaries containing distribution statistics of the engineered features are shown below in Table 2. I have not truncated extreme outlier values, which could either aid in model predictiveness or take away valuable information. Outlier truncation would have to be tested via cross validation to estimate its effect on performance.

Some outliers appear valuable:
1. Negative utilizations, due to negative billing amounts, could indicate overpayments or record-keeping errors.
2. Extremely high payment ratios, as I observe in the EDA section, are more associated with non-defaulters, so this could be predictive.

Table 2

	UTILIZATION_1	UTILIZATION_2	UTILIZATION_3	UTILIZATION_4	UTILIZATION_5	UTILIZATION_6
Min.	-0.620	-1.396	-1.025	-1.375	-0.877	-1.510
1st Qu.	0.022	0.018	0.016	0.014	0.011	0.008
Median	0.314	0.296	0.273	0.242	0.212	0.185
Mean	0.424	0.411	0.392	0.360	0.333	0.319
3rd Qu.	0.830	0.806	0.755	0.668	0.602	0.582
Max.	6.455	6.380	10.689	5.147	4.936	3.886

	PAYMENT_RATIO_2	PAYMENT_RATIO_3	PAYMENT_RATIO_4	PAYMENT_RATIO_5	PAYMENT_RATIO_6
Min.	0	0	0	0	0
1st Qu.	0.045	0.045	0.038	0.036	0.038
Median	0.102	0.104	0.084	0.077	0.090
Mean	0.580	0.767	0.583	0.439	0.513
3rd Qu.	1	1	1	1	1
Max.	4,444.333	5,001	4,444.333	129.705	690.655

	BALANCE_VELOCITY_2	BALANCE_VELOCITY_3	BALANCE_VELOCITY_4	BALANCE_VELOCITY_5	BALANCE_VELOCITY_6
Min.	-99.010	-169	-1,001	-1,001	-525.534
1st Qu.	0.944	0.948	0.966	0.973	0.969
Median	1	1	1	1	1
Mean	2.467	2.151	2.009	2.052	1.958
3rd Qu.	1.053	1.069	1.110	1.102	1.045
Max.	13,010.380	12,599.670	9,076.714	4,801.250	1,936

EXPLORATORY DATA ANALYSIS

Data is examined for relationships that could lead to better modeling results. Exploratory data analysis (EDA) was performed on the entire dataset.
Ordinal variables include EDUCATION, BINNED_AGE, PAY_ J.
Nominal variables include MARRIAGE, SEX.
Continuous variables include LIMIT_BAL, AGE, BILL_AMT j, PAY_AMT j, PAYMENT_RATIO_ j, BALANCE_VELOCITY j, UTILIZATION_ j.

PREDICTOR RELATIONSHIPS WITH TARGET

CONTINUOUS VARIABLE PREDICTIVENESS OF TARGET

Continuous predictor means for Y=0 vs Y=1 were calculated to give an indication of each variable's predictiveness towards target. If a variable has a sizable difference in its mean for observations where Y=1 vs Y=0, then this indicates it will have predictive power of the outcome Y.

LIMIT_BAL, PAY_AMT j, BALANCE_VELOCITY_ j, UTILIZATION_ j appear to have sizable differences (where 1 correponds to Y=1).

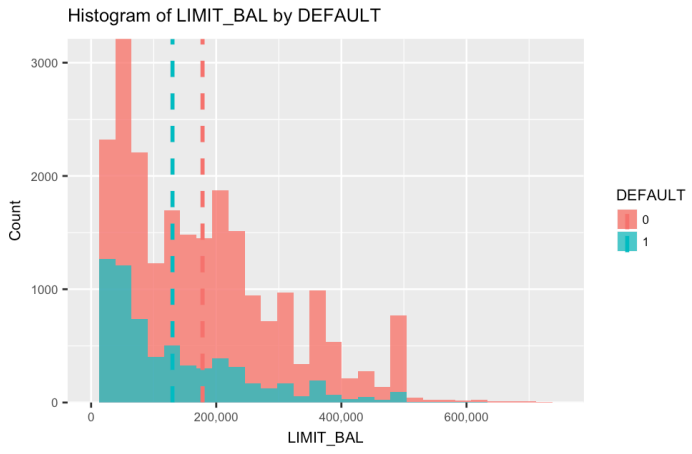
1. LIMIT_BAL Variable

LIMIT_BAL's distribution has a greater percentage of loans with high amounts of credit (500k) for non-defaulters than defaulters. Defaulters have a steeper right-skewed distribution.

The average loan amount for non-defaulters is $178,000, while the average loan amount for defaulters is only $130,000.

This variable should be considered for prediction.

Figure 1



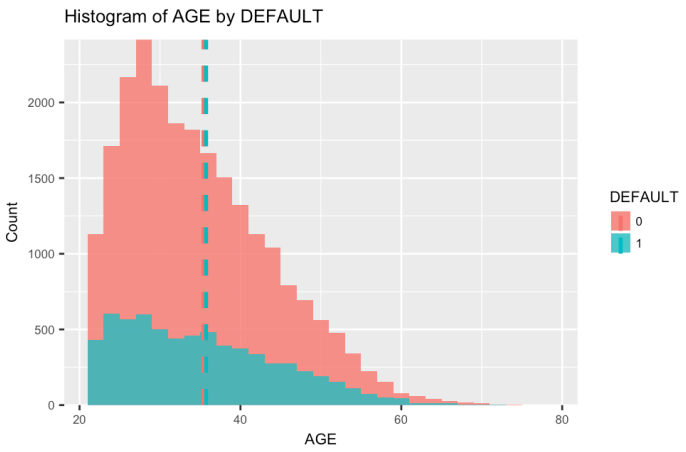
2. AGE Variable

The distribution for non-defaulters has a steeper right skew, with a mode of 29 versus 27 for defaulters. Defaulters have a greater percentage of under 29 creditees, but defaults are slightly more evenly spread across all age groups than the overall pool of loans (less right skew).

The average age is about the same, at around 35.5 for both defaulters and non-defaulters.

This variable might be predictive.

Figure 2



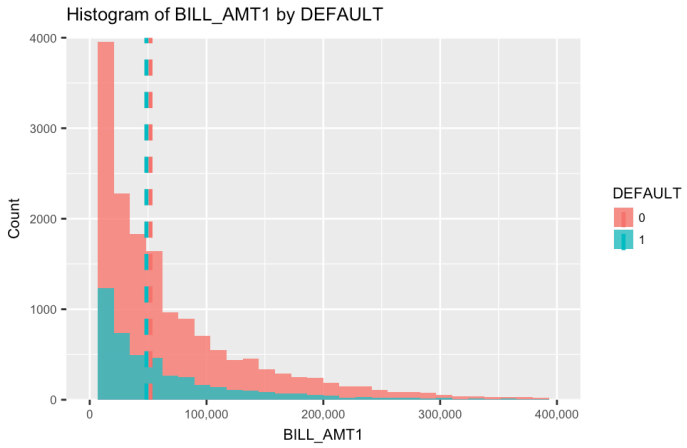
3. BILL_AMT j

The distribution of billing amounts is right skewed for both defaulters and non-defaulters for all months.

The average 9/05 bill amount for non-defaulters is $52,000 and the average bill amount for defaulters is $48,500. I do not see much of a differing pattern between the two target values.

This variable's predictiveness is questionable.

Figure 3

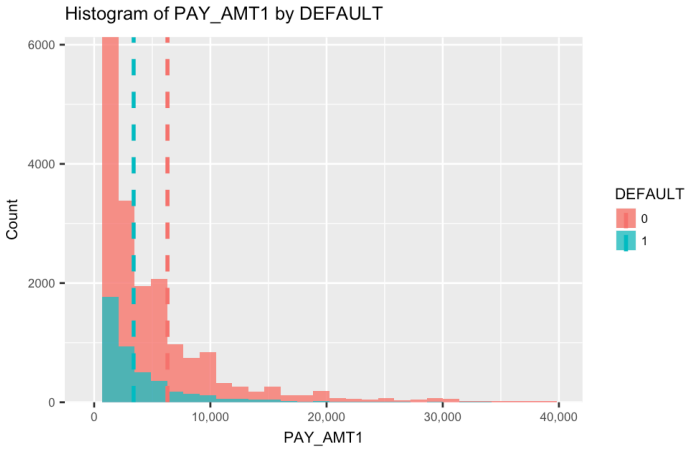


4. PAY_AMT j Variable

Both PAY_AMT1 distributions are right skewed, though non-defaulters have a longer tail and make higher payments. The max non-defaulter pay amount is $880k versus defaulter's $300k.

The average August payment for non-defaulter is nearly double that of defaulters $6300 vs $3400.

Figure 4



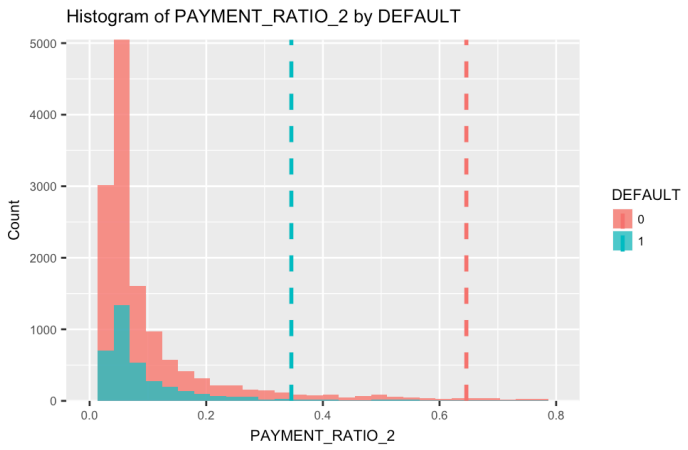
5. PAYMENT_RATIO_ j Variable

The PAYMENT_RATIO_ 2 variable is right-skewed for both target values. Non-defaulters have a longer tail, higher max (4444 vs 103) ,1st quartile, and median. This suggests that higher payment ratios are more likely to indicate non-defaulters.

The average August payment ratio for non-defaulters is nearly double that of defaulters: .65 vs .35.

This variable should be considered for prediction.

Figure 5



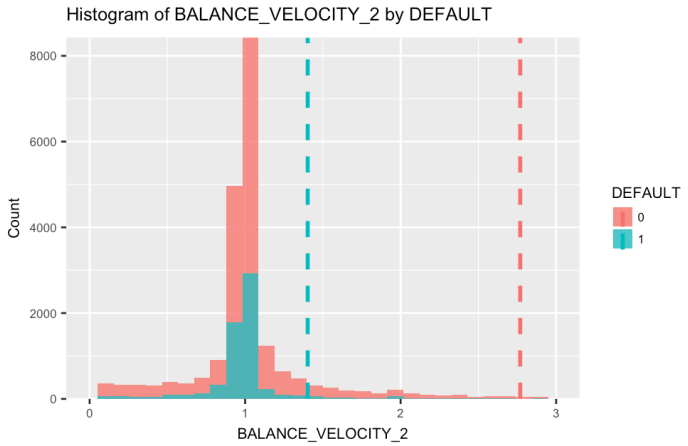
6. BALANCE_VELOCITY_ j Variable

The BALANCE_VELOCITY_ 2 variable has wider much tails for non-defaulters, due to outliers (min=-99 , max=13010) vs (min=-1, max=179) for defaulters. Hence, the mean is somewhat higher. Large magnitude velocities seem to indicate non-defaulters.

Both distributions appear fairly normally distributed, centered at 1.

This variable might be predictive.

Figure 6

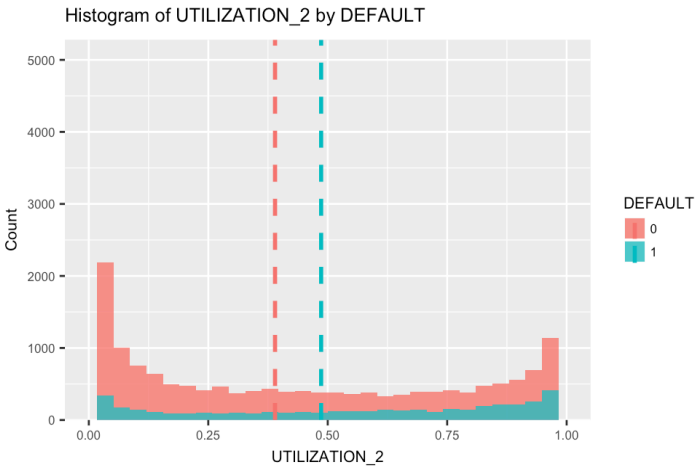


7. UTILIZATION_ j

The UTILIZATION_2 variable has a distribution that is shifted to the right for defaulters, who have a higher 1st quartile, median, mean, and 3rd quartile. However, the distribution for non-defaulters has longer tails (outliers).

This variable should be considered for prediction

Figure 7



CATEGORICAL RELATIONSHIPS WITH TARGET

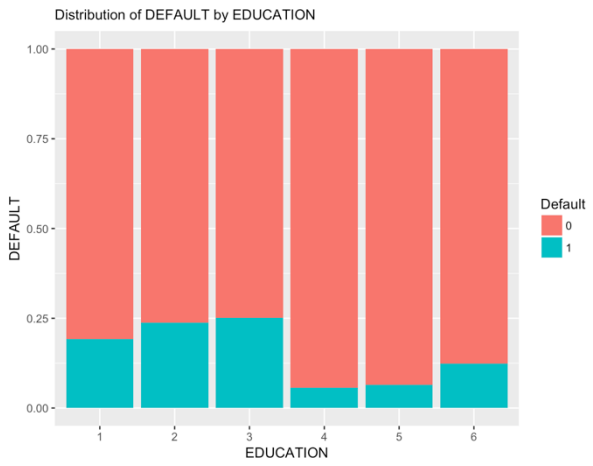
I examine whether each categorical will be predictive of target by inspecting whether there is a sizable difference between each level's proportion of observations with Y=1.

1. EDUCATION Variable

The EDUCATION variable shows a slightly greater proportion of defaults as the level of education increases from high school to graduate school (level 1). Defaults are lower for the other and unknown buckets.

This variable should be considered for prediction.

Figure 8

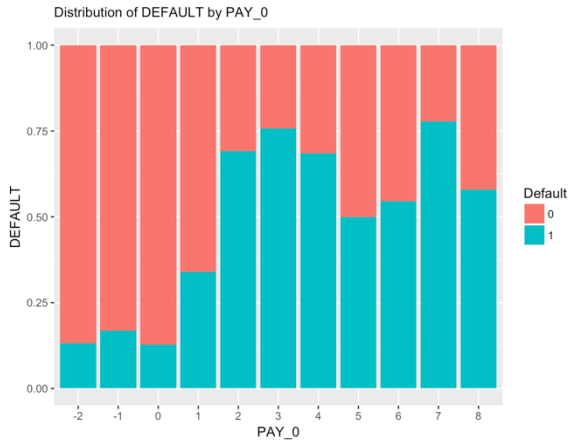


2. PAY_ j Variable

The PAY_ 0 variable shows a greater proportion of defaults as the number of months of payment delay increases. The same pattern holds for the other PAY _ j variables.

This variable should be considered for prediction.

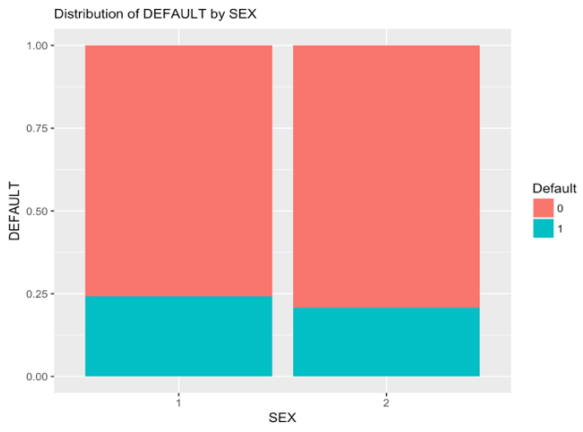
Figure 9



3. SEX Variable

The SEX variable does not show a sizable difference in the proportion of defaults across male and female levels. This variable is likely not predictive of DEFAULT.

Figure 10



VARIABLE PREDICTIVENESS OF TARGET VIA DECISION TREE

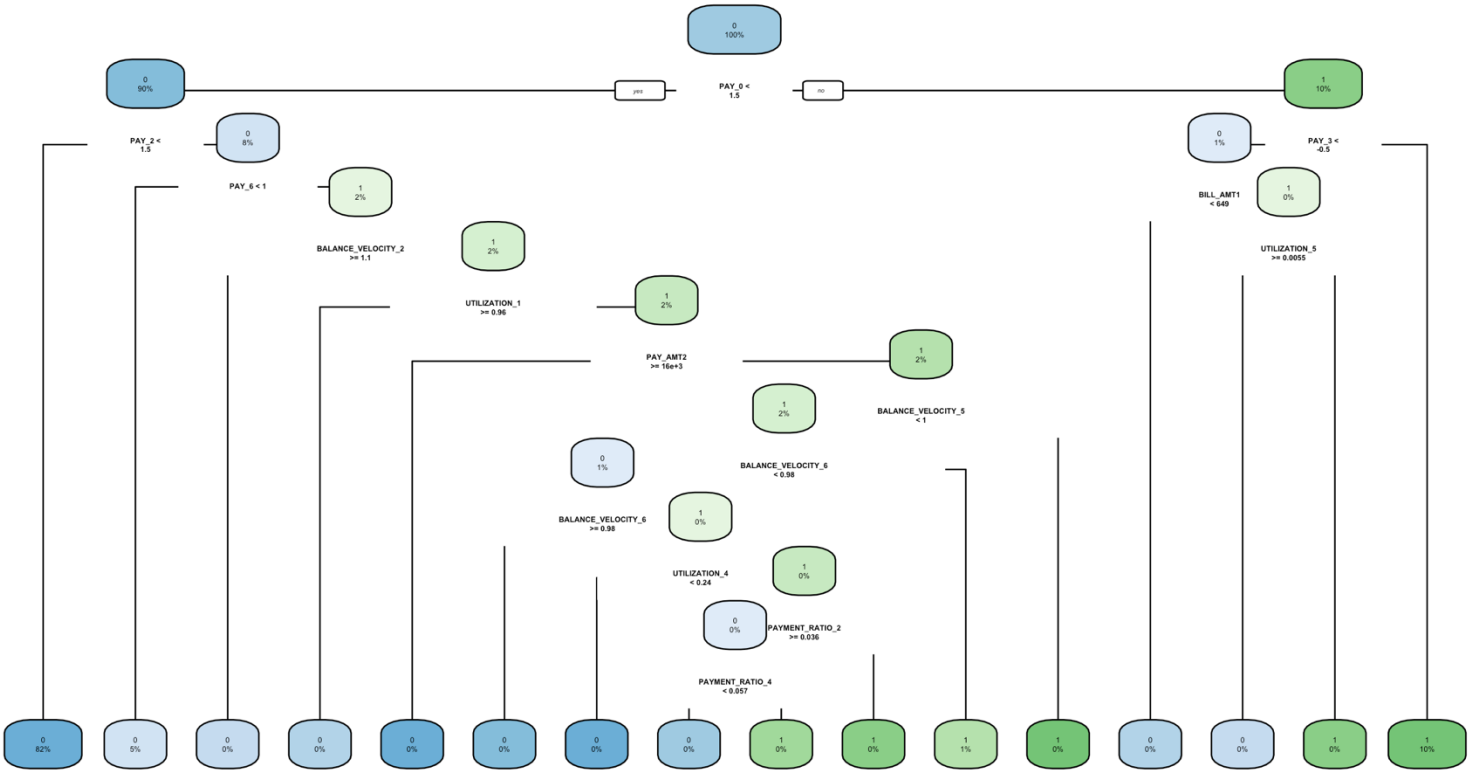
A decision tree (Figure 11) was constructed for a preliminary inspection of variable importances (Table 3), which were calculated from the importance level of each variable in constructing the tree. PAY_ j variables dominated, with PAY_0 having the most importance. PAY_0 < 1.5 divides the tree in two. This means that if there is a payment delay of at least 2 months, then the model predicts default payments will be made. This is the most important information contained in the predictors that is related to target (in building the decision tree).

Table 3

Variable Importance - Decision Tree

	tree.variable.importance
PAY_0	1,572.48
PAY_2	345.81
PAYMENT_RATIO_2	229.10
PAY_4	78.00
PAY_5	76.50
PAY_3	72.70

Figure 11



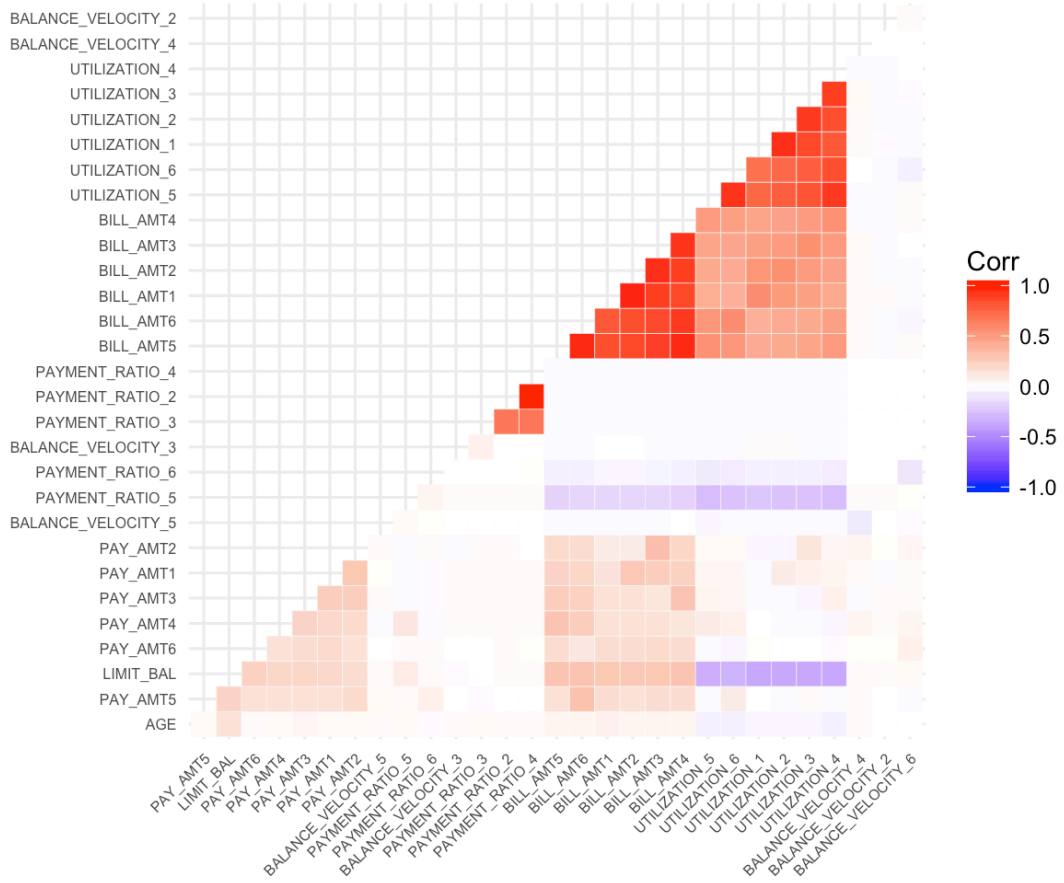
MULTICOLLINEARITY

Observe the correlation matrix heatmap. Multiple continuous predictors are correlated with each other, hence there is multicollinearity. Multicollinearity can mean data redundancy, in that groups of correlated predictors are all supplying the same information to the algorithm. This unnecessarily makes the algorithm run more slowly. It can make estimates of predictor significance wrong and misleading. Also, more predictors potentially means more noise captured by the algorithm, which reduces its generalization abilities. Noise is trivial, one-time only patterns in the data that are not reflective of future patterns, and so should not be captured by the algorithm. However, if a group of mulicollinear predictors are all heavily correlated with target and don't introduce too much noise, it could help generalizability to keep them in. The only way to be sure, however, is to estimate generalizability with and without multicollinear predictors, such as through K-Folds Cross Validation.

From figure 12, notice that:
1. UTILIZATION j variables are strongly positively correlated.
2. BILL_AMT j are strongly positively correlated.
3. Some of the PAYMENT_RATIO's are strongly positively correlated.

Both PAYMENT_RATIO j and UTILIZATION j appeared correlated with target in the prior EDA section because of having means that differed sizably by the value of target. BILL_AMT did not appear sizably related to target. This analysis leaves all predictors in for most algorithms, under the basis of having more information for predictiveness, potentially at the cost of introducing more noise. Future analyses can incorporate feature selection techniques into a cross validation to assess whether their removal would boost generalizability.

Figure 12



PREDICTIVE MODELING: METHODS AND RESULTS

METHODOLOGY

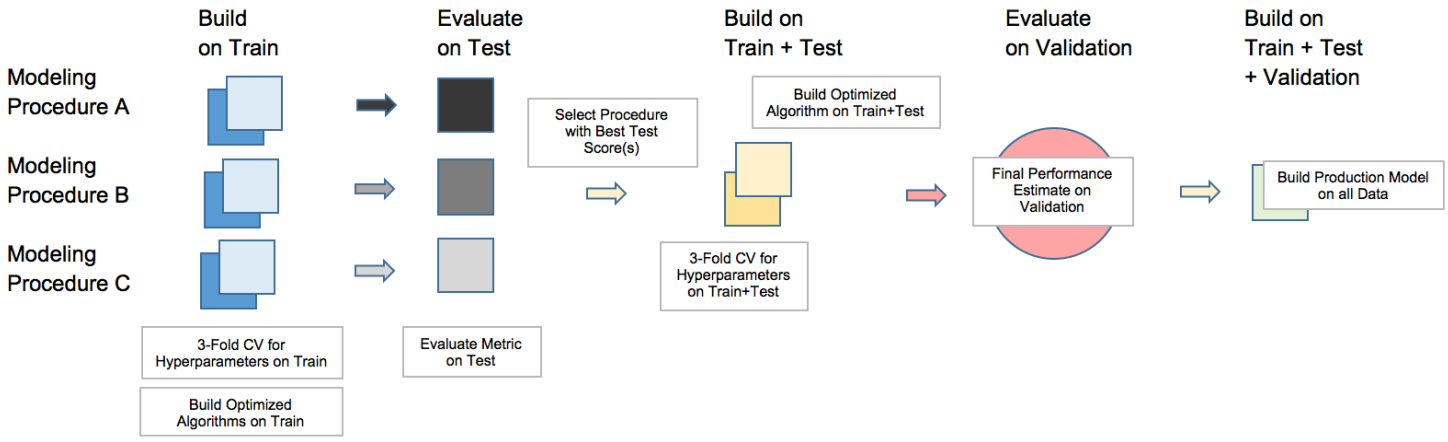
Model selection / hyperparameter tuning, modeling procedure selection, and a final performance estimation of the selected procedure were carried out according to a variant of the 3-way holdout method. Data was split into train, test, and validation.

1. 3-fold cross validation was performed on train for hyperparameter tuning, as part of the algorithm / modeling procedure selection. Competing algorithms include: Gradient Boosted Machines, Random Forests, Stepwise Logistic Regression, and K-Nearest Neighbors.
2. Optimized models from train were tested on test, where the modeling procedure (algorithm and data preprocessing steps) whose model had the best mix of recall, false positive rate, and accuracy was selected.
3. 3-fold cross validation was then rerun on data from both train and test to hyperparameter tune the selected procedure for its performance estimation.
4. The selected procedure was rerun on train and test data using optimized hyperparameters from the 3-fold cross validation.
5. An independent performance estimate for the selected procedure was calculated by testing the resultant model on validation.

The below visual illustrates the above model building steps

Source: Christopher Jose

Figure 13



The best metrics in step 2, as part of the "modeling procedure competition" were not used as final performance estimates of the selected procedure. These values are optimistically biased as generalizability estimates because they are used in selecting the final algorithm / modeling procedure (where the one with the lowest cv estimate is purposely selected).

A performance estimate of the selected procedure must be carried out after all steps in the modeling process have been executed: feature selection/transformation, model selection / hyperparameter tuning, and algorithm / modeling procedure selection. Step 2 calculates performance estimates, but then the algorithm selection step takes place from these estimates. Thus, these estimations are no longer estimates of the entire model building process. They are biased. Final performance estimates of the entire modeling process must take place after the algorithm selection step and not before.

ALGORITHM SELECTION

GRADIENT BOOSTED MACHINES

Gradient Boosted Machines (GBM) starts with a classifier tree that predicts target and sequentially builds new classifier trees off each prior classifier tree, which reduce the train error in each prior tree. The below metrics are from training a hyperparameter optimized GBM algorithm on train and testing on test:

Table 4

Metric	Train	Test
Accuracy	.8200	.8249
True Positive Rate	.9519	.9461
False Positive Rate	.0481	.0539

Table 4 suggests that there are no signs of overfitting, since accuracy and the false positive rate actually increase from train to test, with a slight decrease in the true positive rate (recall).

Hyperparameter tuning to maximize accuracy was conducted for the interaction.depth and n.trees hyperparameters, whose optimal values were 50 and 3, respectively. Interaction.deph is the number of required splits for the decision tree. N.trees is the number of required decision trees (iterations). Observe below how cv accuracy goes down as iterations goes up, which indicates overfitting. Hyperparameters shrinkage and n.minobsinnode were held constant at .1 and 10, respectively.

Figure 14 shows a decrease in cv accuracy as the number of trees / iterations increases, for the majority of decision tree sizes (Max Tree Depth). This suggests signs of overfitting as model complexity (via more iterations) increases. Figure 15 shows that GBM variable importances have PAY_0 contributing the most.

Figure 14

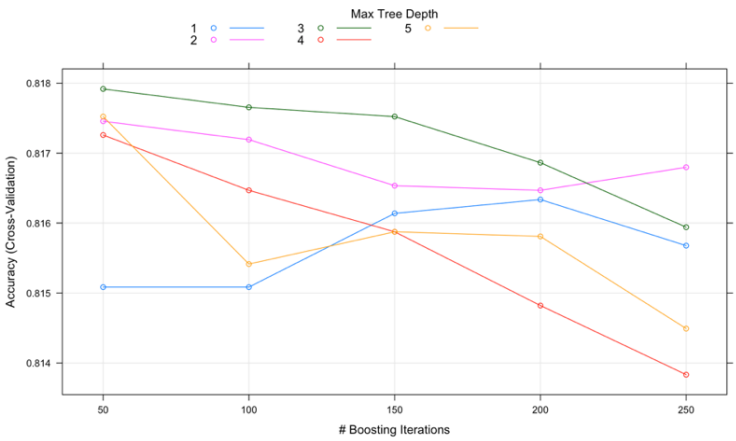
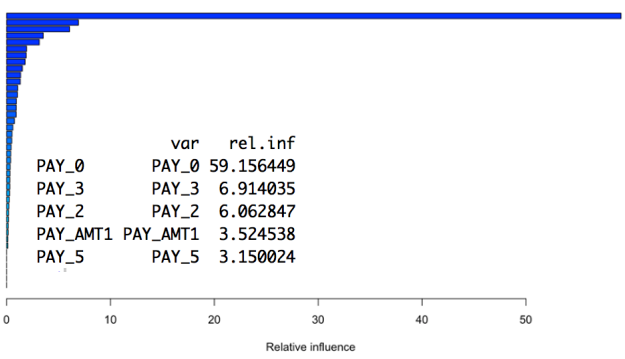


Figure 15



RANDOM FORESTS

Random Forests (RF) 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. The final prediction is based on the predictions from all trees.

The below metrics are from training a hyperparameter optimized RF algorithm on train and testing on test:

Table 5

Metric	Train	Test
Accuracy	.9991	.8199
True Positive Rate	.9999	.9334
False Positive Rate	.0000	.0666

Table 5 suggests that RF is overfitting the train set, since the true positive rate and accuracy are nearly perfect, while taking a dip on the test split.

Hyperparameter tuning to maximize accuracy was conducted for the mtry hyperparameter, the number of predictors available for splitting at each decision tree node. Its optimal value was found to be 42 (all predictors). Observe below how cv accuracy goes up as mtry goes up. However, perhaps using less predictors, reducing the number of trees, and more, could reduce model complexity sufficiently on train to increase generalizability on test.

Figure 16

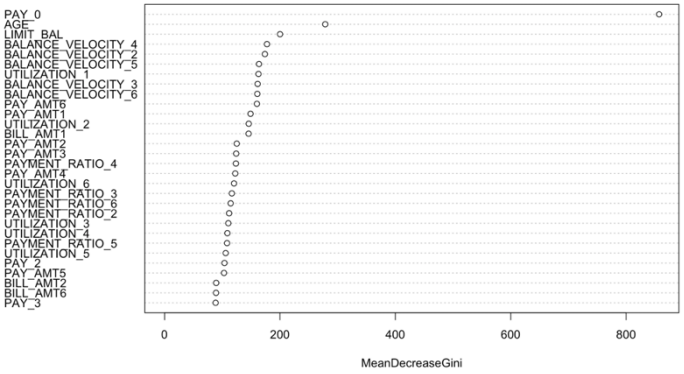


Figure 17

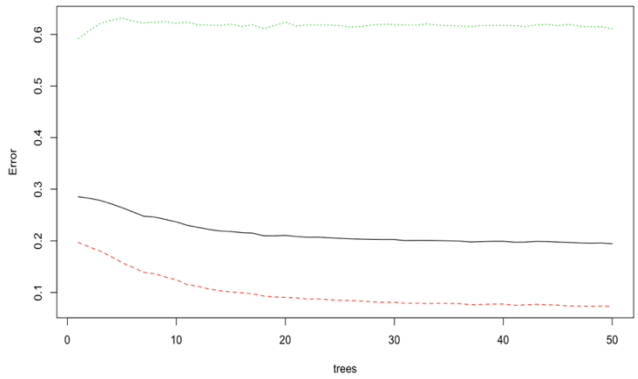


Figure 16 shows RF variable importances. Top variables in fitting the model include PAY_0, AGE, LIMIT_BAL, and BALANCE_VELOCITY_4 (PAY_0 was #1 for GBM and the Decision Tree, too).

Figure 17 shows the Overall Out-of-Bag Error Estimate (OOB) in black, the error estimate for Y=1 in green, and error estimate for Y=0 in dotted red as the number of decision trees increases. OOB estimates the generalization error (misclassification rate) by having classifiers built in the RF make predictions on training examples not included in their training.

Average overall OOB is .22
OOB for Y=1 is .099
OOB for Y=0 is .62
The OOB error estimate for predicting defaults is highest, at 62%.

FEATURE SELECTION - STEPWISE SELECTION - LOGISTIC REGRESSION

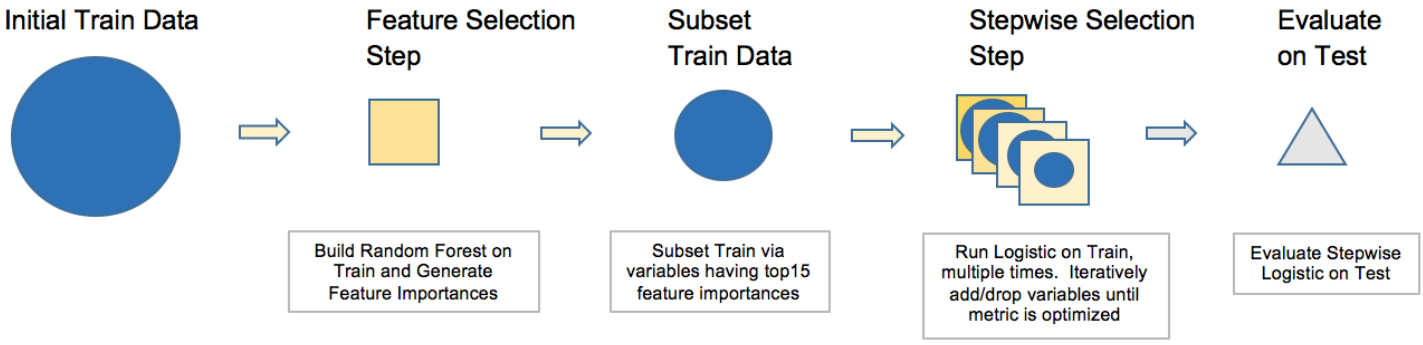
This procedure consists of the following

1. A feature selection step—the top 15 feaures were selected from Random Forests importances, built off train.
2. A 2nd feature selection step—Logistic Regression (LR) was run off subsetted train via a stepwise selection algorithm.
3. Evaluate LR on test with features optimized from above two processes.

The below visual illustrates the above modeling steps

Source: Christopher Jose

Figure 18



LR maximizes the log likelihood of the observed data and outputs a linear predictor (b1x1 + b2x2 +...) that can be interpreted as the log-odds of Y. To get P(Y=1), 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, LR assumes that Y has a binomial distribution instead of a normal distribution.

LR was implemented according to the stepwise selection algorithm. Stepwise Selection starts with a null model and iteratively adds variables that most optimize the evaluation metric (AIC, in this case). But at each step, before adding a variable, if the removal of a given variable improves AIC, then it is removed. Thus, stepwise selection is like forward selection, but with a backward selection action carried out first, at each step.

The backward selection component of stepwise selection actually does not take into effect during the first two iterations. The reasoning is as follows (this logic is in accordance with the StepAIC funciton in R):

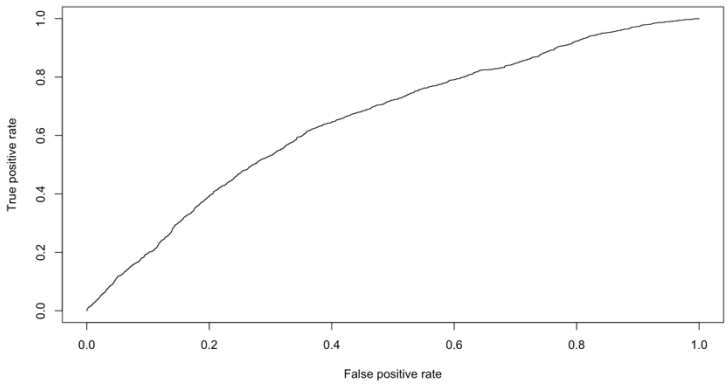
1. Upon adding the first variable, when checking to remove the first variable, AIC would just be null model AIC, which is guaranteed suboptimal or else the 1st variable never would have been selected.

2. Upon adding the second variable, when checking to remove the first variable, model AIC would just be the AIC corresponding to a model with just the second variable, which is guaranteed suboptimal since this variable wasn't selected in the first step.

In evaluating Stepwise LR on test, probabilties are outputted. Below is the ROC curve, which plots recall and (1-specificity) for each potential probability threshold value. Finding the optimal probability threshold for the below curve is needed to convert probability predictions to binary / label predictions.

Test ROC Curve

Figure 19



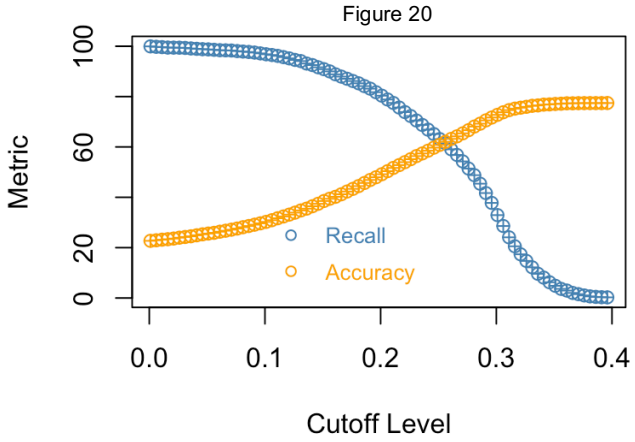
In setting a probability threshold, train metrics were examined over many probability thresholds. A threshold of .29 was selected to achieve a train accuracy of .70, yielding a true positive rate (recall) of .42 (see Table 6).

Train Metrics vs Threshold

Table 6
Stepwise Logistic Metrics on Train

	cutoffs	accuracy	recall	precision	f1_score
57	0.281	68.096	48.788	35.084	40.816
58	0.286	69.236	45.399	35.683	39.959
59	0.291	70.547	41.689	36.571	38.962
60	0.296	71.680	37.774	37.348	37.560
61	0.301	72.754	32.924	37.984	35.274
62	0.306	73.775	28.601	38.911	32.969

Figure 20



Observe above the tradeoff between true positive rate (recall) and accuracy/precision in both Table 6 and Figure 20. Train was used to optimize accuracy and the threshold, just as how train was used to optimize hyperparameters with respect to accuracy for the other classifiers (though the optimal threshold was not determined via 3-fold CV). Since algorithm selection will occur on test, it is not desirable to have test information leaking into model building, which would have happened had the threshold been optimized using test metrics. Test leakage into model building would cause bias in the test evaluation metrics.

The below metrics are from training a stepwise selection optimized logistic algorithm on train and testing on test:

Table 7

Metric	Train	Test
Accuracy	.7028	.7024
True Positive Rate	.7834	.7764
False Positive Rate	.2166	.2236

Table 7 suggests that Stepwise LR is not overfitting, since the metrics are fairly similar. However, both the train and test metrics seem fairly low, relative to other algorithms. There is too much bias in the algorithm and not enough signal (underlying patterns) in the data is being captured by the algorithm. Perhaps the feature selection methods helped cause this.

Stepwise Logistic Modeling Summary

Coefficients from the stepwise selection optimized LR model on train are presented in the table below. Variables statistically significant at the .001 level include LIMIT_BAL, PAY_AMT1, BILL_AMT5, PAY_AMT2, AGE, and BILL_AMT1.

Coefficients can be interpreted as the amount of change in the log odds for a one-unit increase in the predictor, holding all other variables constant. e^(coefficient) is the new odds relative to the old odds for a one-unit increase in the predictor. Then, e^(coefficient) – 1 is the % change in the odds for a one-unit increase in the predictor.

For a $100,000 increase in LIMIT_BAL,
exp(-0.000002803*$100000) = .755 The new odds of defaulting is 75.6% of the old odds.
exp(-0.000002803*$100000)-1 = -.244 The odds of defaulting decreases 24%.

In checking the .001 significant coefficients for reasonability (see Table 8),
As LIMIT_BAL, PAY_AMT j increase, odds of default goes down. This suggests that the higher the loan and the greater the payments, the less likely a creditee will default. This makes sense. A creditor would not want to give bigger loans to riskier clients. Strangely, an increasing BILL_AMT5 raises default likelihood, while a decreasing BILL_AMT1 decreases default likelihood. Increasing age raises default likelihood, which is possible.

Notice that the number of predictors has decreased from the subset provided by RF feature selection (15 to 12 predictors).

Table 8

Stepwise Logistic

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-0.999	0.080	-12.524	0	***
LIMIT_BAL	-0.00000	0.00000	-12.913	0	***
PAY_AMT1	-0.00002	0.00000	-5.541	0.00000	***
BILL_AMT5	0.00000	0.00000	4.716	0.00000	***
PAY_AMT2	-0.00002	0.00000	-5.771	0	***
AGE	0.010	0.002	4.698	0.00000	***
PAY_AMT4	-0.00001	0.00000	-2.860	0.004	**
PAY_AMT3	-0.00001	0.00000	-2.846	0.004	**
BILL_AMT1	-0.00000	0.00000	-3.986	0.0001	***
BILL_AMT3	0.00000	0.00000	2.498	0.012	*
PAYMENT_RATIO_2	-0.120	0.054	-2.247	0.025	*
PAY_AMT5	-0.00001	0.00000	-2.173	0.030	*
PAY_AMT6	-0.00000	0.00000	-1.905	0.057	*

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

K-NEAREST NEIGHBORS

K-Nearest Neighbors (KNN) 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.

Feature normalization was implemented as a preprocessing step or else large-variance features would have dominated the algorithm. This step transforms variable ranges to [0,1].

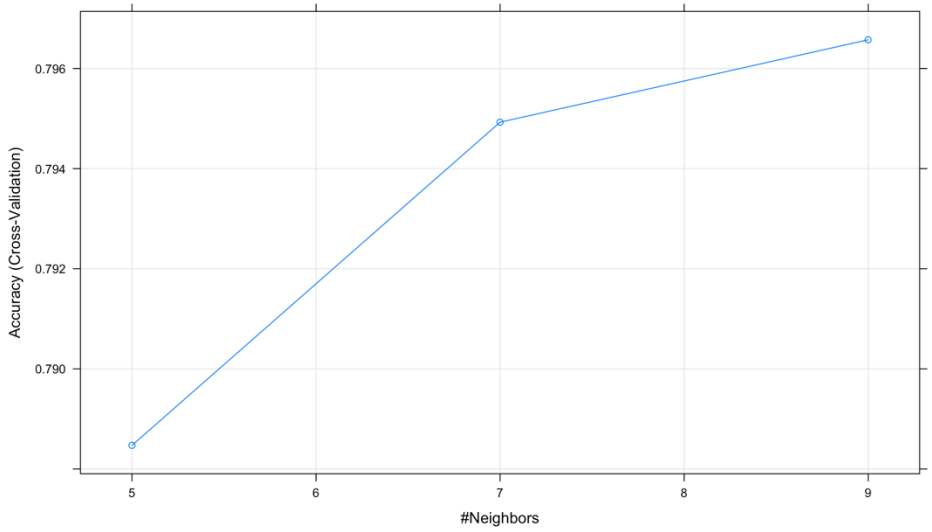
The below metrics are from training a hyperparameter optimized KNN algorithm on train and testing on test:

Table 9

Metric	Train	Test
Accuracy	.8221	.8057
True Positive Rate	.9564	.9426
False Positive Rate	.0436	.0574

Table 9 suggests no signs of overfitting, since test metrics decrease just slightly from those of train. Hyperparameter tuning to minimize accuracy was conducted for the k hyperparameter, the number of clusters. Its optimal value was found to be 42 (all predictors). Observe below how cv accuracy goes up as k goes up.

Figure 21



COMPARISON OF RESULTS

An aggregation of evaluation metrics on test are shown below. GBM is selected because of having the highest test accuracy, highest test true positive rate, lowest test false positive rate, and did not show signs of overfitting. KNN is the runner up, does not show signs of overfitting, and has high metrics on both train and test. In comparison, RF clearly overfit train. It had a huge drop in metrics from train to test. Stepwise LR did not overfit, but was too biased by not fitting well enough on train (and hence test). It scored both poor train and test metrics. If not enough signal is captured, then generalization won't be done well.

Table 10

	GBM	RF	Logit	KNN
Test Accuracy	.8249	.8199	.7024	.8057
Test True Positive Rate	.9461	.9334	.7764	.9426
Test False Positive Rate	.0539	.0666	.2236	.0574
Train Accuracy	.8200	.9991	.7028	.8221
Train True Positive Rate	.9519	.9999	.7834	.9564
Train False Positive Rate	.0481	.0000	.2166	.0436

FINAL PERFORMANCE ESTIMATE

As the selected algorithm, GBM was then hyperparameter tuned and trained on all of the train and test data. An independent performance estimation was conducted on the validation split. GBM's validation split metrics are also favorable and similar to the test metrics generated during the algorithm selection phase.

Table 11

Metric	Validation
Accuracy	.8234
True Positive Rate	.9515
False Positive Rate	.0485

CONCLUSION

Predicting default likely creditees is do-able and can be done well. The selected algorithm in this paper, Gradient Boosted Machines, has an estimated misclassification rate of only 17.66%. Recall is extremely high, at 95.15%, meaning of actual defaulters in the validation set, 95.15% were correctly predicted as being default likely. All of the algoritms performed well, suggesting that predicting future defaulters among customers at major banks in Taiwan is absolutely feasible, and doing so can stand to benefit both banks and customers.

Even with these good results, there is a lot of room for improvement:

1. Feature transformation steps like outlier truncation, normalization, logging, binning, and more, can be evaluated incorporated.
2. Feature selection based on techniques including 1) mulicollinearity 2) pairwise correlation 3) correlation 4) principal components analysis with target, can be implemented to reduce data redundancies.
3. Alternative splitting schemes like nested cross validation can provide less biased performance estimates for hyperparameter tuning, algorithm selection, and the final performance estimate.
4. Evaluation based on AUC and ROC offers a non-threshold dependent evaluation process.
5. Additional feature engineering might find predictors that are more predictive of target
6. Additional algorithms, like Naïve Bayes, Support Vector Machines, Neural Networks, and more, can be considered.
7. Additional hyperparameters can be tuned to better optimize existing algorithms.
8. Visuals can be produced to estimate the bias and variance of models, such as the change in train and test error as model complexity increases.
9. Algorithm selection can be conducted by consulting statistical tests.

BIBLIOGRAPHY

Yeh, I. (2016, January 26). Default of credit card clients Data Set. Retrieved from https://archive.ics.uci.edu/ml/datasets/default of credit card clients