EBO Machine Learning and Artificial Intelligence Research: Predictions of Loan Re-performance

Johnathan Boyce

2/25/2020

Table of Contents

1 Introduction

1.1 Background

PennyMac services loans as part of the Ginnie Mae (GNMA) Early Buyout (EBO) program. GNMA provides mortgage servicers the option to buy out of MBS pools loans that are 90 days delinquent in order to minimize servicing costs for due to advancing payments to the Trust, despite borrowers failing to make payments. There are five paths for loans in this analysis: cures, modifications (mods), paid-in-full (PIFs), short sale / deed-in-lieu (SS/DIL), and foreclosure (FC). Re-performing loans (cures, mods) can be re-delivered into GNMA pools at a premium to par, while PIFs enable servicers to recoup advances of delinquent interest.

1.2 Problem

Since GNMA allows mortgage servicers to re-deliver loans into GNMA pools at a premium to par, PennyMac misses the opportunity to re-deliver loans at a premium price for loans that naturally cure (start re-paying after becoming delinquent) if they are not bought out of as part of the EBO program. In an effort to capture the revenue associated with natural cures (as well as projecting the probabilities of other paths for loans that do not cure), PennyMac is tasked with creating a machine learning model to clearly identify loans that have a high probability for naturally curing, as well as approximating the possibility of remaining paths as part of quantifying the risks associated with the remaining performance paths.

2 Data

2.1 Data Sources

The data sources used for this machine learning and artificial intelligence EBO re-performance project came from servers and tables located within PennyMac's servicing databases.

2.2 Data Collection, Preparation and Exploratory Analysis

In order to utilize the vast machine learning libraries contained within the Python programming language, we imported the data tables into a pandas dataframe. Below is a list of preliminary features used for this portion of the machine learning research project. **Table 2-1** shows a sample of the features obtained from PennyMac's servicing databases.

Table 2-1 – Loan Features Extracted

Feature	Number of Records
LoTypeId	12,626
MEPeriod	12,626
PropertyState	12,626
Units	12,626
PropertyTypeId	12,626
ProductLineCodeId	12,626
InvestorId	12,626
LoanPurposeCodeId	12,626
CurrentPrincipalBalanceAmt	12,626
CurrentInterestRate	12,626
Rate_at_D90	12,626
OriginalInterestRt	12,626
OriginalLoanToValueRatio	12,626
OriginalPrincipalBalanceAmt	12,626
LoanClosingDt	12,626
PropertyTypeDesc	12,626
ProductLineCodeDesc	12,626
InvestorName	12,626
LotypeDesc	12,626
LoanPurposeCodeDesc	12,626
OriginalBorrowerCreditScore	12,626
OccupancyCodeDesc	12,626
cltv_trendix	12,626
cltv_avm	12,626
BeginningLossMitStatus	12,626
Dt	12,626
Flow	12,626
OrigFlow	12,626
OrigDelinquentPaymentCount	12,626
OrigNextPaymentDueDt	12,626
LastDelinquentPaymentCount	12,626
LastNextPaymentDueDt	12,626
PaymentMade	12,626
upb	12,626
CLTV_Trendix_Grp	12,626
CLTV_AVM_Grp	12,626
LTV	12,626
FicoGroup	12,626
OrigDQ	12,626
proptype	12,626
Loan_Purpose	12,626
Occ	12,626
trial	12,626
maxdq	12,626
MaxDQgrp	12,626
DLQ_diff	12,626
DLQ_Diff_Grp	12,626
Total Features	**47**

As part of the data cleaning and preparation process, we evaluate the number of null records. ***Table 2-2*** shows the percentage of null records for each feature. ***Table 2-3*** shows the count of non-null entries for each feature. Note that the features with the null records are the following:

- Units
- LoanPurposeCodeId
- ProductLineCodeDesc
- LoanPurposeCodeDesc
- OriginalBorrowerCreditScore
- cltv_avm
- BeginningLossMitStatus
- CLTV_AVM_Grp
- FicoGroup
- maxdq
- MaxDQgrp
- DLQ_diff
- DLQ_Diff_Grp

Table 2-2 – Total Percentage of Null Records

Feature	Percentage of Records
LoTypeId	0.0%
MEPeriod	0.0%
PropertyState	0.0%
Units	4.5%
PropertyTypeId	0.0%
ProductLineCodeId	0.0%
InvestorId	0.0%
LoanPurposeCodeId	0.0%
CurrentPrincipalBalanceAmt	0.0%
CurrentInterestRate	0.0%
Rate_at_D90	0.0%
OriginalInterestRt	0.0%
OriginalLoanToValueRatio	0.0%
OriginalPrincipalBalanceAmt	0.0%
LoanClosingDt	0.0%
PropertyTypeDesc	0.0%
ProductLineCodeDesc	0.0%
InvestorName	0.0%
LotypeDesc	0.0%
LoanPurposeCodeDesc	0.0%
OriginalBorrowerCreditScore	29.9%
OccupancyCodeDesc	0.0%
cltv_trendix	0.0%
cltv_avm	4.7%
BeginningLossMitStatus	60.6%
Dt	0.0%
Flow	0.0%
OrigFlow	0.0%
OrigDelinquentPaymentCount	0.0%
OrigNextPaymentDueDt	0.0%
LastDelinquentPaymentCount	0.0%
LastNextPaymentDueDt	0.0%
PaymentMade	0.0%
upb	0.0%
CLTV_Trendix_Grp	0.0%
CLTV_AVM_Grp	4.7%
LTV	0.0%
FicoGroup	29.9%
OrigDQ	0.0%
proptype	0.0%
Loan_Purpose	0.0%
Occ	0.0%
trial	0.0%
maxdq	8.3%
MaxDQgrp	8.3%
DLQ_diff	8.3%
DLQ_Diff_Grp	8.3%
Total Features	**47**

Table 2-3 – Number of Non-Null Records and Datatypes for the Selected Features

Feature	Number of Non-Null Records	Status	Datatype
LoTypeId	12,626	non-null	category
MEPeriod	12,626	non-null	int32
PropertyState	12,626	non-null	category
Units	12,052	non-null	category
PropertyTypeId	12,626	non-null	category
ProductLineCodeId	12,626	non-null	category
InvestorId	12,626	non-null	category
LoanPurposeCodeId	12,620	non-null	category
CurrentPrincipalBalanceAmt	12,626	non-null	float32
CurrentInterestRate	12,626	non-null	float32
Rate_at_D90	12,626	non-null	category
OriginalInterestRt	12,626	non-null	float32
OriginalLoanToValueRatio	12,626	non-null	float32
OriginalPrincipalBalanceAmt	12,626	non-null	int32
LoanClosingDt	12,626	non-null	datetime64[ns]
PropertyTypeDesc	12,626	non-null	category
ProductLineCodeDesc	12,624	non-null	category
InvestorName	12,626	non-null	category
LotypeDesc	12,626	non-null	category
LoanPurposeCodeDesc	12,620	non-null	category
OriginalBorrowerCreditScore	8,857	non-null	int32
OccupancyCodeDesc	12,626	non-null	category
cltv_trendix	12,626	non-null	float32
cltv_avm	12,027	non-null	float32
BeginningLossMitStatus	4,975	non-null	category
Dt	12,626	non-null	int32
Flow	12,626	non-null	category
OrigFlow	12,626	non-null	category
OrigDelinquentPaymentCount	12,626	non-null	int32
OrigNextPaymentDueDt	12,626	non-null	datetime64[ns]
LastDelinquentPaymentCount	12,626	non-null	int32
LastNextPaymentDueDt	12,626	non-null	datetime64[ns]
PaymentMade	12,626	non-null	int32
upb	12,626	non-null	category
CLTV_Trendix_Grp	12,626	non-null	category
CLTV_AVM_Grp	12,027	non-null	category
LTV	12,626	non-null	category
FicoGroup	8,857	non-null	category
OrigDQ	12,626	non-null	category
proptype	12,626	non-null	category
Loan_Purpose	12,626	non-null	category
Occ	12,626	non-null	category
trial	12,626	non-null	category
maxdq	11,582	non-null	int32
MaxDQgrp	11,582	non-null	category
DLQ_diff	11,582	non-null	int32
DLQ_Diff_Grp	11,582	non-null	category

*Manually set LoanClosingDt, OrigNextPaymentDueDt, and LastNextPaymentDueDt to datetime datatype

After dropping the eight features/columns, we are left with 39 features/columns (prior to implementing one hot encoding for categorical columns). After deleting null records, we are left with 10,535 rows, which will be further divided into training and test splits (70% training / 30% testing) in order to train and validate this supervised machine learning model. ***Table 2-4*** shows the reconciliation with the original 12,626 loans. The allocation for model training was 7,374 for training, 3,161 for testing, and 2,091 were kicked entirely from the analysis.

Table 2-4 – Data Allocation and Assignment

Train_Test	Flow	Count	Total UPB	% of Total (Count)	% of Total Sample (Count)
⊟ Test	0-DILSS	174	$29,560,480.78	5.50%	1.38%
	1-Prepay	139	$16,432,104.90	4.40%	1.10%
	2-Mod	768	$126,562,109.40	24.30%	6.08%
	3-Cure	799	$114,681,529.53	25.28%	6.33%
	4-Dlq-Pay	236	$34,587,225.39	7.47%	1.87%
	5-Dlq-Nopay	1045	$140,484,735.17	33.06%	8.28%
Test Total		3161	$462,308,185.17	100.00%	25.04%
⊟ Train	0-DILSS	422	$71,148,178.28	5.72%	3.34%
	1-Prepay	288	$41,970,282.24	3.91%	2.28%
	2-Mod	1873	$304,429,060.68	25.40%	14.83%
	3-Cure	1823	$251,360,020.62	24.72%	14.44%
	4-Dlq-Pay	584	$88,589,156.54	7.92%	4.63%
	5-Dlq-Nopay	2384	$316,137,515.33	32.33%	18.88%
Train Total		7374	$1,073,634,213.69	100.00%	58.40%
⊟ OUT	0-DILSS	149	$25,757,006.39	7.13%	1.18%
	1-Prepay	65	$8,541,152.56	3.11%	0.51%
	2-Mod	518	$88,912,594.80	24.77%	4.10%
	3-Cure	400	$57,448,372.30	19.13%	3.17%
	4-Dlq-Pay	178	$26,627,685.80	8.51%	1.41%
	5-Dlq-Nopay	781	$110,614,152.11	37.35%	6.19%
OUT Total		2091	$317,900,963.96	100.00%	16.56%

Now that we have removed unwanted features, we turn our attention to categorical fields and transform these using one hot encoding. The features used for one hot encoding are the following:

- PropertyState
- ProductLineCodeId
- PropertyTypeDesc
- ProductLineCodeDesc
- LotypeDesc
- LoanPurposeCodeDesc
- OccupancyCodeDesc
- OrigDQ
- proptype
- Loan_Purpose
- Occ
- trial
- MaxDQgrp
- DLQ_Diff_Grp

Note: the Flow feature is used as the target for the loan re-performance outcome.

As part of data preparation for modeling, categorical features (versus numeric features) require transformation into a more usable format for use in machine learning by using one hot encoding. This technique transposes each category within a column into its own individual column which now has Boolean entries to signify when the category's existence is true for that instance (represented by 1s and 0s in the column). Once we transform these features using one hot encoding, we can normalize the data prior to model input. ***Figure 2-1*** shows a sample mapping of one hot encoding transformation.

Row Index	LotypeDesc
0	FHA RESIDENTIAL
1	FHA RESIDENTIAL
2	RHS
3	VA RESIDENTIAL
4	FHA RESIDENTIAL

Row Index	LotypeDesc_FHA RESIDENTIAL	LotypeDesc_RHS	LotypeDesc_VA RESIDENTIAL
0	1	0	0
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0

Figure 2-1 – Example of One Hot Encoding Transformation

After creating one hot encoding for the categorical fields, we combine the features into one dataframe using concatenation. An example of the transformed dataframe is shown in ***Table 2-5***. In addition, we normalize the fields using the StandardScaler from the sklearn library. The resulting scaled and combined dataframe (with 119 columns/features not including LoanId) is shown below in ***Table 2-6***. Notice that the values for various features are either positive or negative. Negative values mean that values are "lower than most" when compared to other values for the feature. Positive values mean that the value is "higher than most" of the other values within the feature.

Table 2-5 – Example of Transformed Dataframe after One Hot Encoding

Row Index	LotypeDesc_FHA RESIDENTIAL	LotypeDesc_RHS	LotypeDesc_VA RESIDENTIAL	...	PropertyState_AL	PropertyState_AR	PropertyState_AZ	PropertyState_CA	PropertyState_CO
0	0	0	1	...	0	0	0	0	0
1	1	0	0	...	0	0	0	1	0
2	1	0	0	...	0	0	0	0	0
3	1	0	0	...	0	0	0	0	0
4	0	0	1	...	0	0	1	0	0

Table 2-6 – Example of Normalized Data for Use in the Classification Model

MEPeriod	CurrentPrincipalBalanceAmt	CurrentInterestRate	cltv_trendix	cltv_avm	LastDelinquentPaymentCount	maxdq	DLQ_diff	PropertyState_AL	PropertyState_AR
0.188574068	0.651571918	-0.364578239	-0.28392375	-0.585796204	-0.740636172	-0.112371302	0.143429837	-0.179261846	-0.101295786
0.01795288	-0.816258306	-0.264018481	-0.386203166	-1.121602517	-0.673867371	-0.521541497	-0.409454661	-0.179261846	-0.101295786
0.145918771	-0.813762099	-0.264018481	0.156109244	-0.20700555	0.260895841	0.296798893	-0.409454661	-0.179261846	-0.101295786
0.145918771	-1.244117096	-0.264018481	-0.717229033	-0.61069643	-0.60709857	-0.3169564	0.696314336	-0.179261846	-0.101295786
0.487161146	-1.367050884	-0.364578239	-0.046799003	-0.237756944	-0.740636172	-0.521541497	-0.409454661	-0.179261846	-0.101295786
0.060608177	-0.819362005	-0.364578239	0.178313801	1.339669705	-0.740636172	-0.521541497	-0.409454661	-0.179261846	-0.101295786
0.145918771	-0.71317744	1.64661873	-0.338167427	-0.55052384	0.060589439	-0.521541497	-0.409454661	-0.179261846	-0.101295786
0.273884661	-0.88000137	1.043259579	-1.021947321	-1.201660249	-0.673867371	-0.3169564	0.143429837	-0.179261846	-0.101295786
0.01795288	-0.320941642	1.043259579	-0.325487276	-0.570776331	0.394433443	0.194506344	-0.409454661	-0.179261846	-0.101295786
0.188574068	-1.084404562	1.847738846	-1.611563074	-1.170166128	-0.740636172	0.092213796	-0.409454661	-0.179261846	-0.101295786

3 Methodology - Supervised Learning: Classification

The outcome we're trying to predict (re-performance of delinquent loans) contains five separate classifications. Machine learning algorithms that can be applied for supervised learning classification are the following:

- Decision Tree (explainable)
- Logistic Regression (explainable)
- Support Vector Machine (SVM) - Kernel
- Random Forest
- Neural Network
- Gradient Boosting Tree
- Naïve Bayes
- SVM - Linear

In light of the need to explain results from the machine learning model, it was decided that the Decision Tree algorithm would be the most optimal engine to employ. Primary reasons for using decision trees are due to the transparency in the decisioning waterfall (through exploration of charts of tree leaf nodes and the rules associated with each node), the flexibility to handle both numerical and categorical data, and the fact that it uses only the most important features.

3.1 Decision Tree

We will use the decision tree classification algorithm to build a model to predict loan re-performance of delinquent EBO loans using historical data from PennyMac's servicing databases. As part of the model-build, there are six distinct classifications as part of the model training process. The six separate classifications are the following:

1. Short Sale / Deed-in-Lieu (DILSS)
2. Prepay / Paid-in-Full (Prepay / PIF)
3. Modified
4. Natural Cure
5. Delinquent but made payments
6. Delinquent without payments

3.1.1 Modeling

During model creation, we split our population into training sets and testing sets. These randomly selected populations will have a split of 70% / 30% training to testing, respectively. The training set is used to train the model. Then, the test set is used to validate the model by comparing predicted values with the target values.

In addition, we will limit the max depth of the decision tree to four rather than twelve (even though a max depth of twelve resulted in the highest accuracy). Limiting max depth mitigates the tendency for overfitting the model. And while the higher max depth yielded a higher accuracy, it resulted in overfitting the model. A symptom of overfitting was evidenced by the fact that a majority of the classification paths had a 100% of the total probability, rather than probabilities being spread across most if not all of the classification paths (which is what we would observe and expect in reality).

In ***Figure 3-1***, we see the resulting decision tree that includes leaf nodes, using a max depth of four.

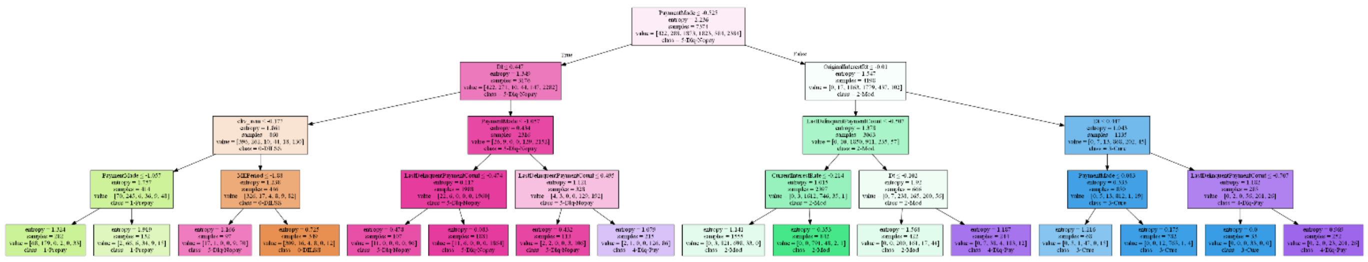


Figure 3-1 – Decision Tree Leafs and Nodes Using Max Depth of Four

3.1.2 Results

3.1.2.1 Decision Tree Leaf Nodes

Looking more closely at the decision tree leaf nodes, features that are shown as node decisioning rules are the following:

- PaymentMade
- Dt
- OriginalInterestRt
- cltv_avm

- MEPeriod (presumed month of buyout)
- LastDelinquentPaymentCount (dlq payments at time of classification)
- CurrentInterestRate

The most important determinant in this decision tree is the PaymentMade feature due to it being the highest node in the decision tree. PaymentMade represents the number of payments at the time of buyout. Next in priority in this waterfall are the Dt and OriginalInterestRt features. Dt is the month the classification occurs, while OriginalInterestRt is the original rate of the loan.

3.1.2.2 Classification Reports

Table 3-1 shows the classification report results using a max depth of four (MD4) while ***Table 3-2*** shows the classification report results using a max depth of twelve (MD12). Precision in the report is the measure of the accuracy of predictions of the respective class. Recall is a measure of the percentage of the positives that were properly predicted. F1-score is the harmonic mean of the precision (true positives per predicted positive) and the recall (true positives per real positive), resulting in a score that ranges between 1.0 and 0.0 (1.0 being the best score).

$$Precision = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

$$Recall = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

$$F1\ Score = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

Where:

TP = True Positive – when a case was positive and predicted positive

TN = True Negative – when a case was negative and predicted negative

FP = False Positive – when a case was negative but predicted positive

FN = False Negative – when a case was positive but predicted negative

Table 3-1 – Classification Report – Decision Tree with Max Depth = 4

Classification	precision	recall	f1-score	support	Classification Description
0	83%	77%	80%	174	0-DILSS
1	66%	78%	72%	139	1-Prepay
2	64%	97%	77%	768	2-Mod
3	95%	47%	63%	799	3-Cure
4	69%	86%	77%	236	4-Dlq-Pay
5	97%	90%	93%	1,045	5-Dlq-Nopay

accuracy	79%	3,161		
macro avg	79%	79%	77%	3,161
weighted avg	84%	79%	79%	3,161

Table 3-2 – Classification Report – Decision Tree with Max Depth = 12

Classification	precision	recall	f1-score	support	Classification Description
0	85%	86%	85%	174	0-DILSS
1	81%	75%	78%	139	1-Prepay
2	85%	89%	87%	768	2-Mod
3	88%	83%	85%	799	3-Cure
4	80%	84%	82%	236	4-Dlq-Pay
5	93%	94%	93%	1,045	5-Dlq-Nopay

accuracy	88%	3,161		
macro avg	85%	85%	85%	3,161
weighted avg	88%	88%	88%	3,161

As a rule of thumb, the weighted average of F1-Score should be used to compare classifier models, not global accuracy. When comparing the F1-Score, MD12 shows a higher score of 88% compared to 79% for MD4. While the initial reaction might be to use results from MD12, the proper approach is to use MD4. The thought behind this approach centers around the desire to minimize the overfitting problem (models that are more generalized are more likely to produce similar results when predicting classifications out-of-sample). In addition, a notable observation of the MD4 report shows that while the recall for 3-Cure was relatively low (47%), **it displayed a high precision (95%). This is especially desirable when considering EBOs, since one of the major revenue-capturing scenarios is buying out loans that naturally cure.**

3.1.2.3 Confusion Matrix

Figure 3-2 below shows the confusion matrix to compare the predicted versus targeted classes, as part of the process for supervised learning model training and development. Note that the matrix mirrors the "recall" statistics captured in the Classification Report. The strongest classification recall result was Class 2 (2-Mod) with a 97% recall rate. Although the True Label for Class 3 (3-Cure) had the lowest recall rate of 47%, the next largest predicted label for the Class 3 True Label was Class 2 (2-Mod), which is also part of the re-performing Classes (1-Prepay, 2-Mod, and 3-Cure). This result is encouraging due to the fact that the engine for this supervised learning model appears to be picking up the "positive" aspects of the re-performing Classes.

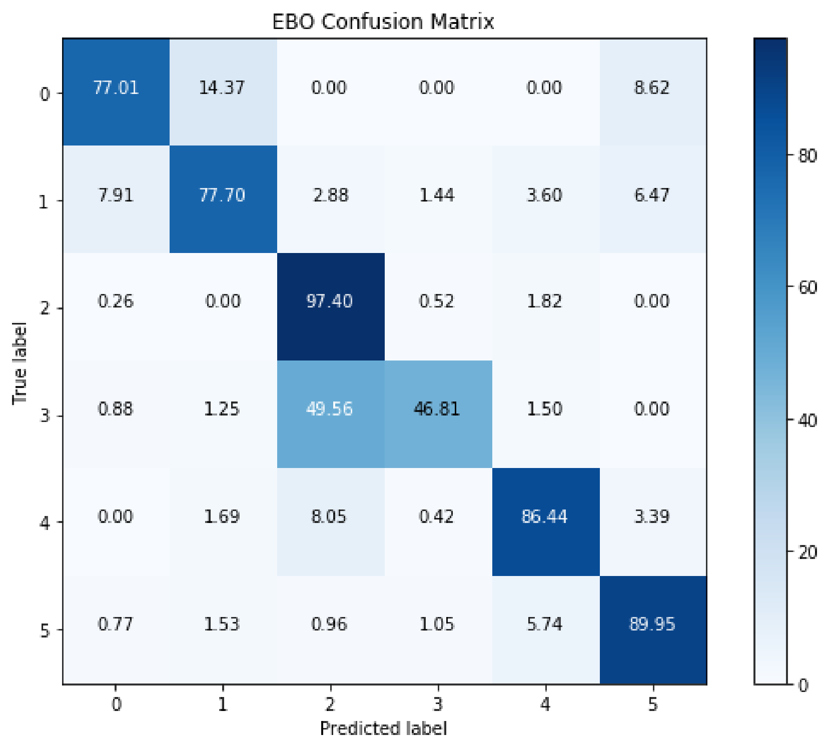
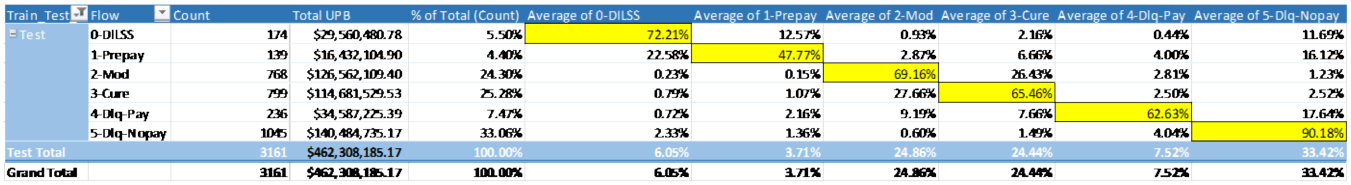


Figure 3-2 – Confusion Matrix for Targeted Classification with Max Depth = 4

3.1.2.4 Probabilities by Classification

Preliminary results are promising when evaluating the probabilities broken out by the six classifications. Overall, predicted probabilities for classifications are in-line with the supervised target classification. By using ***Table 2-4***, we can compare the re-performance (cures, mods, and PIFs) for the supervised classes versus predicted classes. Re-performance for the supervised classes was 54.0% (4.4% + 24.3% + 25.3%). Re-performance for the predicted classes roughly had the same re-performance of 53.0% (3.7% + 24.9% + 24.4%) using data from ***Table 3-3*** found below. Re-performance is very similar when comparing supervised classes for training and predicted classes from the test split. More work needs to be done, but the initial results are extremely promising.

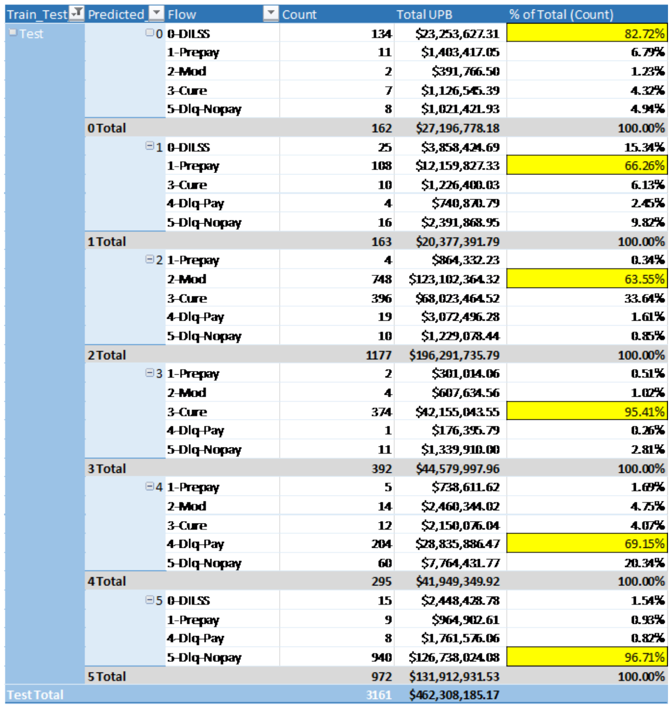
Table 3-3 – Testing Split Probabilities By Classification



Train_Test	Flow	Count	Total UPB	% of Total (Count)	Average of 0-DILSS	Average of 1-Prepay	Average of 2-Mod	Average of 3-Cure	Average of 4-Dlq-Pay	Average of 5-Dlq-Nopay
Test	0-DILSS	174	$29,560,480.78	5.50%	72.21%	12.57%	0.93%	2.16%	0.44%	11.69%
	1-Prepay	139	$16,432,104.90	4.40%	22.58%	47.77%	2.87%	6.66%	4.00%	16.12%
	2-Mod	768	$126,562,109.40	24.30%	0.23%	0.15%	69.16%	26.43%	2.81%	1.23%
	3-Cure	799	$114,681,529.53	25.28%	0.79%	1.07%	27.66%	65.46%	2.50%	2.52%
	4-Dlq-Pay	236	$34,587,225.39	7.47%	0.72%	2.16%	9.19%	7.66%	62.63%	17.64%
	5-Dlq-Nopay	1045	$140,484,735.17	33.06%	2.33%	1.36%	0.60%	1.49%	4.04%	90.18%
Test Total		3161	$462,308,185.17	100.00%	6.05%	3.71%	24.86%	24.44%	7.52%	33.42%
Grand Total		3161	$462,308,185.17	100.00%	6.05%	3.71%	24.86%	24.44%	7.52%	33.42%

We see from ***Table 3-4*** the breakout of the precision for the predicted classes and the status of what the actual target values were. While F1-Score is the preferred measure, looking closely at the composition of the "misses" can tell a story of whether or not the model generally has "close misses" versus "massive misses." Close misses would be Classes that fall in either the same re-performance path (cures, mods, PIFs) or the same non-performing path (SS/DIL, Dlq Pay, or Dlq Nopay). In addition, for each predicted class in ***Table 3-4***, we see the precision highlighted in yellow. The remaining actual classes are also shown for each predicted Class.

Table 3-4 – Distribution of the Predicted Classes versus Actual Classes



Train_Test	Predicted	Flow	Count	Total UPB	% of Total (Count)
Test	0	0-DILSS	134	$23,253,627.31	82.72%
		1-Prepay	11	$1,403,417.05	6.79%
		2-Mod	2	$391,766.50	1.23%
		3-Cure	7	$1,126,545.39	4.32%
		5-Dlq-Nopay	8	$1,021,421.93	4.94%
	0 Total		162	$27,196,778.18	100.00%
	1	0-DILSS	25	$3,858,424.69	15.34%
		1-Prepay	108	$12,159,827.33	66.26%
		3-Cure	10	$1,226,400.03	6.13%
		4-Dlq-Pay	4	$740,870.79	2.45%
		5-Dlq-Nopay	16	$2,391,868.95	9.82%
	1 Total		163	$20,377,391.79	100.00%
	2	1-Prepay	4	$864,332.23	0.34%
		2-Mod	748	$123,102,364.32	63.55%
		3-Cure	396	$68,023,464.52	33.64%
		4-Dlq-Pay	19	$3,072,496.28	1.61%
		5-Dlq-Nopay	10	$1,229,078.44	0.85%
	2 Total		1177	$196,291,735.79	100.00%
	3	1-Prepay	2	$301,014.06	0.51%
		2-Mod	4	$607,634.56	1.02%
		3-Cure	374	$42,155,043.55	95.41%
		4-Dlq-Pay	1	$176,395.79	0.26%
		5-Dlq-Nopay	11	$1,339,910.00	2.81%
	3 Total		392	$44,579,997.96	100.00%
	4	1-Prepay	5	$738,611.62	1.69%
		2-Mod	14	$2,460,344.02	4.75%
		3-Cure	12	$2,150,076.04	4.07%
		4-Dlq-Pay	204	$28,835,886.47	69.15%
		5-Dlq-Nopay	60	$7,764,431.77	20.34%
	4 Total		295	$41,949,349.92	100.00%
	5	0-DILSS	15	$2,448,428.78	1.54%
		1-Prepay	9	$964,902.61	0.93%
		4-Dlq-Pay	8	$1,761,576.06	0.82%
		5-Dlq-Nopay	940	$126,738,024.08	96.71%
	5 Total		972	$131,912,931.53	100.00%
Test Total			3161	$462,308,185.17	

4 Discussion and Conclusion

The model for predicting re-performance of EBO loans is off to an excellent start, as the foundational set of features and modeling assumptions for the decision tree generated impressive results. There are additional opportunities to improve the results without overfitting. Some of these areas for improvement revolve around removal of features that likely will not be available at time of prediction, utilizing market rates and data (such as swap rates and Fed Funds rates), and economic releases (such as unemployment and manufacturing indices).

Additional areas for enhancement include:

- Implementing Gradient boosting trees
- Addition of economic release features
- Removal of features that do not materially affect predictions

Further exploration of the above-mentioned potential measures for model enhancement will take place in the next phase of this project.