FINAL DRAFT

MACHINE LEARNING
CONSUMER LOAN PROCESSING

By:
Ramkishore Rao
DSA 5900 / Credit Hrs: 4 hrs
Summer 2022
Faculty Advisors: Dr. Trafalis/Dr. Radhakrishnan
Faculty Coordinator: Dr. Beattie

TABLE OF CONTENTS

List of Tables

List of Exhibits

List of Appendices

1.0 Introduction

This project serves as my final practicum for my master's degree in Data Science and Analytics being completed at the University of Oklahoma. As part of this project, various machine learning algorithms were applied to a bank loan dataset (bandora dataset) to aid in the processing of loan applications from consumers at a bank. For this study, a git hub repository developed by Dr. Jeff Heaton for his Deep Learning (DL) (Heaton, 2022) class at Washington University at St. Louis and his accompanying book (Heaton, 2022) were leveraged. In addition, class notes from Dr. Nicholson and from Dr. Diochnos were also utilized during the study.

The primary programming language used was Python, with its pre-existing modules. Tableau has been used during the initial exploration phase of the data.

2.0 Objectives

The main objective of the project is to use the existing bank loan dataset to develop back-end statistics models in order to provide a decision on the loan applications. Training, validation, and testing were performed using the existing dataset. An implementation plan is provided below.

3.0 Exploratory Data Analysis

A bank loan dataset (bandora dataset) that contained 112 features was utilized in this study. Of the 112 features, one of the features was default_date, i.e., this feature had the date on which default occurred. This feature was the target class, and if default had occurred, it was assigned a value of 1 and if default had not occurred, it was assigned a value of 0.

Percentage of data points that belonged to target classes 0 and 1 by total were 66% and 34%, respectively (see Table 1).

Table 1: Data Breakdown by Target Class

Overall Class Counts
Defaulted: 1
Not Defaulted: 0

Target Class	Count of Target Class	% of Total Count of Target Class)
0	156,588	66.0%
1	80,635	34.0%
Grand Total	237,223	100.0%

Count of Target Class and % of Total Count of Target Class) broken down by Target Class.

3.1 Analysis Summary

A few tables and exhibits are provided in the following pages. They present a breakout of aggregated values of several features by target class value (i.e., 0 if debtor has not defaulted and 1 if debtor has defaulted).

3.2 Analysis Findings

Box and whisker plots for features brokwn down by target class shown on Exhibit 1 indicate the following:

1. Higher spread in data and higher maximum observed for Target Class 1 for the following features:
    - Probability of Default
    - Debt Types
    - Interest Servicing
2. No Significant Differences Between Classes observed for the following features:
    - Applied Loan Amount
    - Income types

Lower debtor default rates are attributed to the following based on estimates of aggregated data values breakouts by target class:

1) Higher Income (Exhibit 2)
2) Lower Interest Servicing (Exhibit 3)
3) Higher Previous Credit (Exhibit 4)
4) Better Credit Rating (Exhibit 5)
5) Lower median probability of default and expected loss (Exhibits 6 and 7)
6) Higher Education (Exhibit 10)
7) Higher actual number of previous procured loans (Exhibit 11)
8) More Prompt Payment (Exhibit 12)

Exhibit 1: Box and Whisker Plots, Select Variables

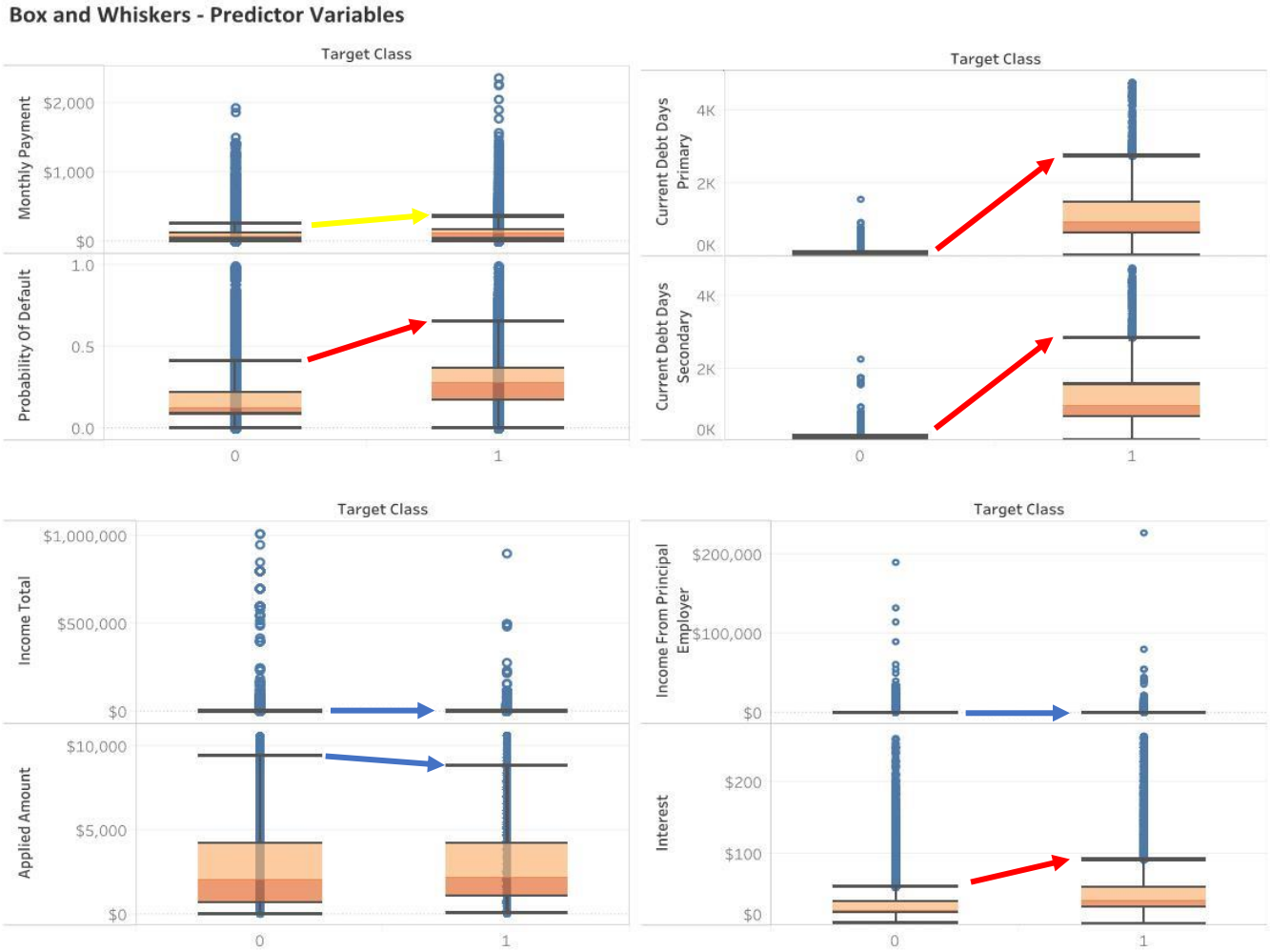


Exhibit 2: Income Breakouts by Target Class

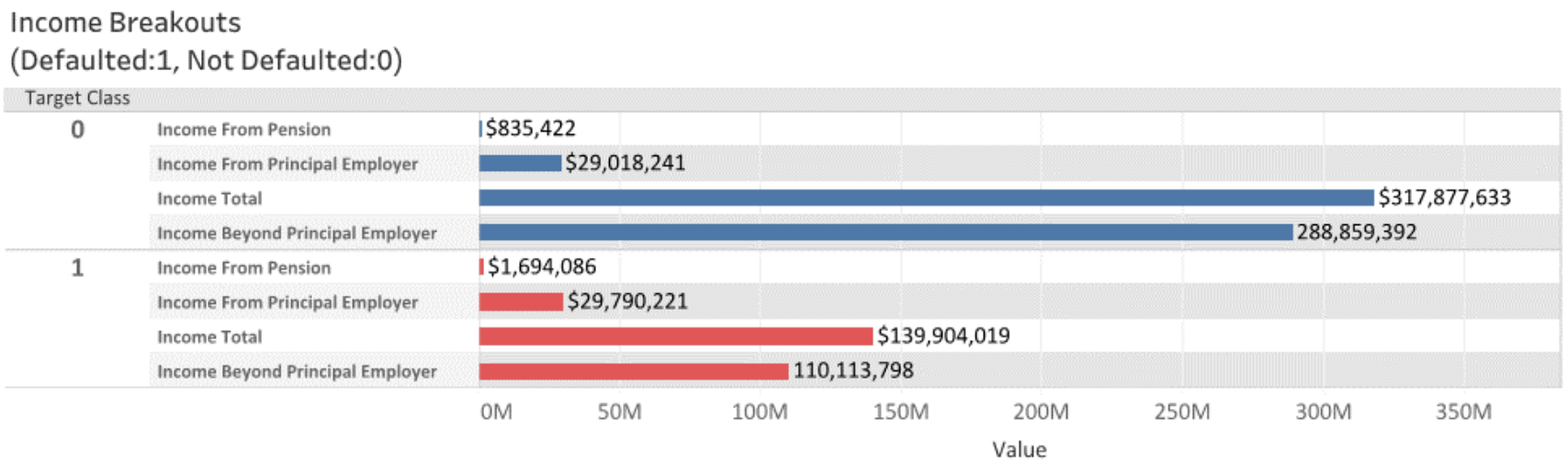


Exhibit 3: Interest Servicing Breakouts by Target Class

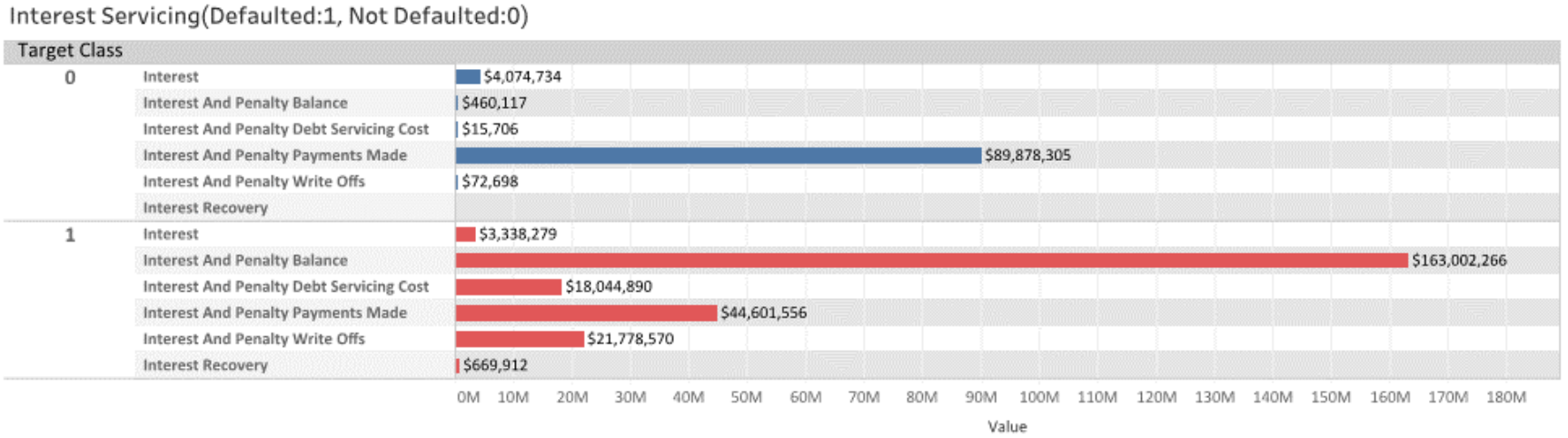


Exhibit 4: Liability Breakouts by Target Class

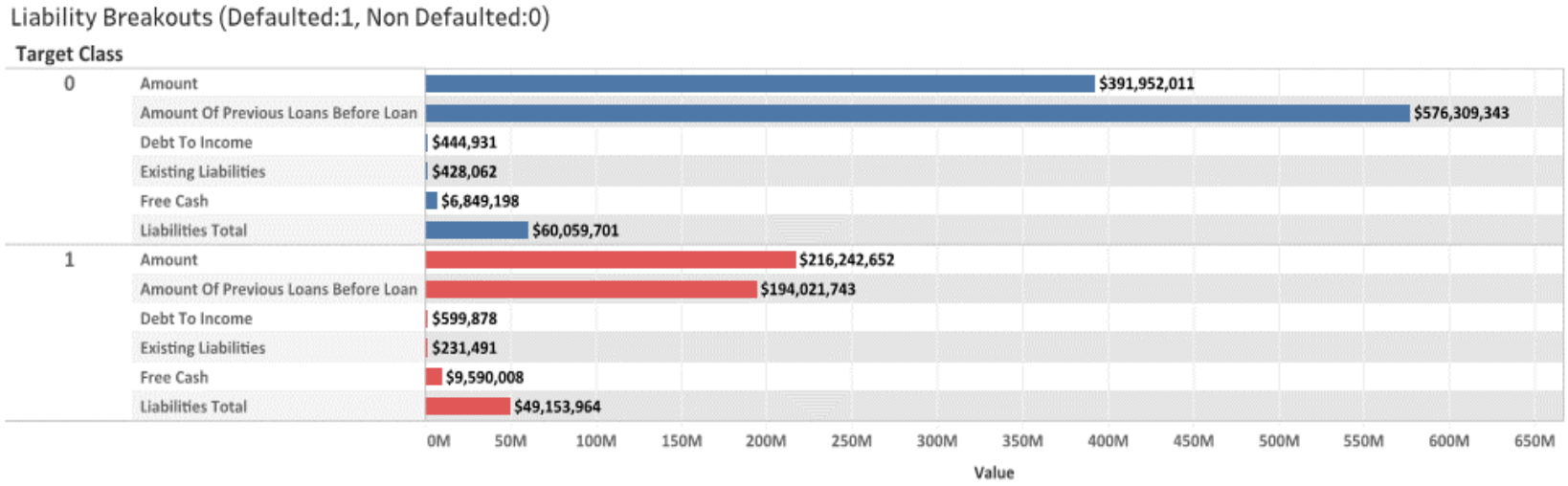


Exhibit 5: Credit Rating by Median Probability of Default

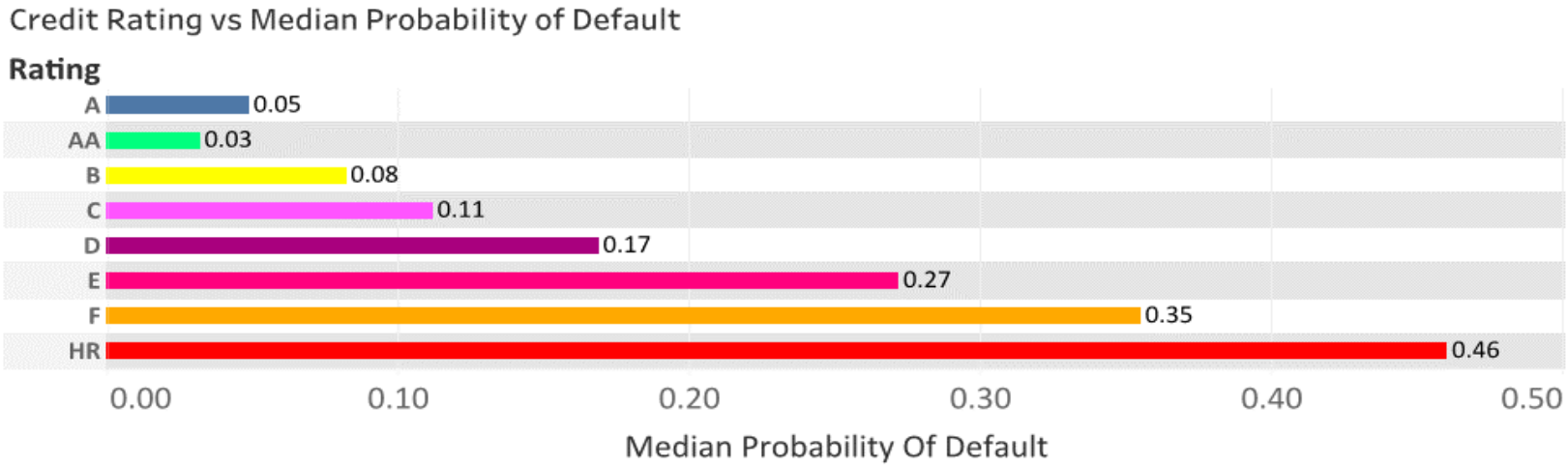


Exhibit 6: Credit Parameters by Target Class - I

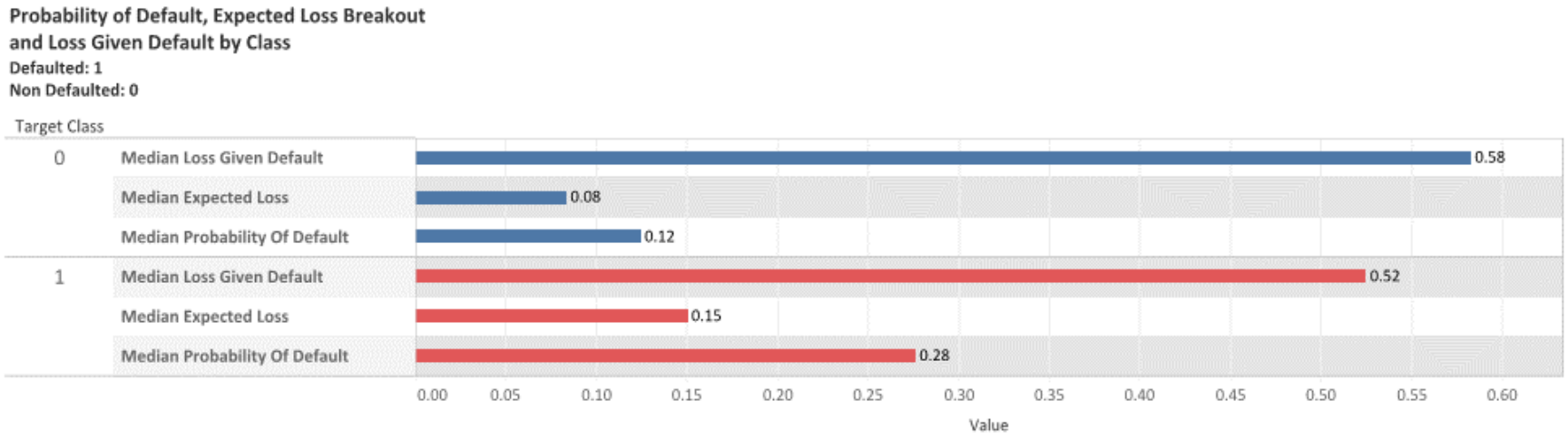
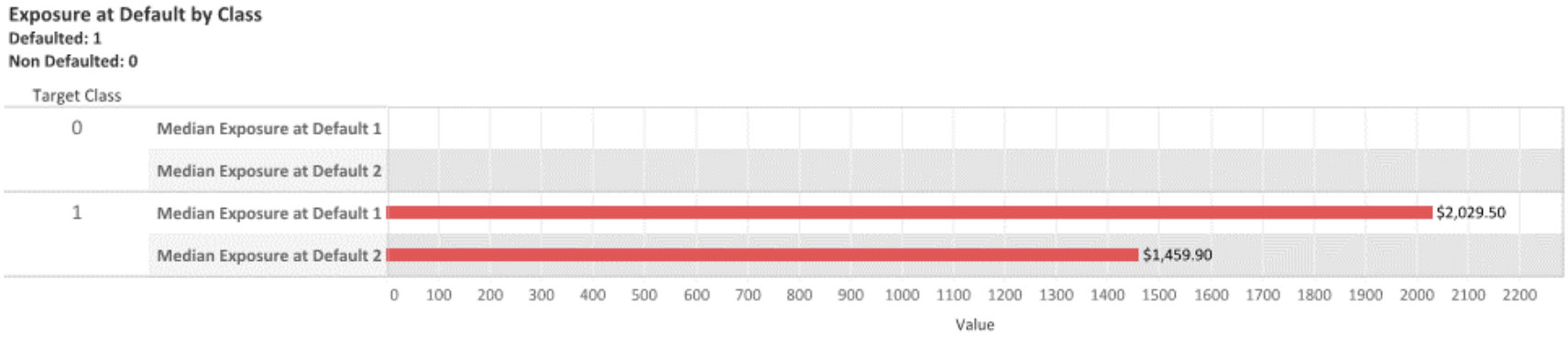


Exhibit 7: Credit Parameters by Target Class - II



Note:

EAD1: Exposure at default, outstanding principal at default, EAD 2: Exposure at default, loan amount less all payments prior to default

Exhibit 8: Employment Status Counts Breakdown by Target Class

Employment Status
Defaulted: 1
Not Defaulted: 0

Target Class	Employment Status							
	-1	0	2	3	4	5	6	Grand Total
0	140,054	5	456	13,782	428	1,147	595	156,467
1	60,581	27	728	16,278	875	860	1,205	80,554
Grand Total	200,635	32	1,184	30,060	1,303	2,007	1,800	237,021

Note:
1: Unemployed, 2: Partially employed, 3: Fully employed, 4: Self-employed, 5: Entrepreneur 6: Retiree

Exhibit 9: Work Experience/Home Ownership Type Counts Breakdown by Target Class

Work Experience/Home Ownership Category Breakouts
Defaulted: 1
Not Defaulted: 0

Target Class	Home Ownership Type	Work Experience					
		2-5 Yrs	5-10 Yrs	10-15 Yrs	15-25 Yrs	<2 Yrs	>25 Yrs
0	0			2		1	1
	1	394	941	917	1,369	205	1,567
	2	629	742	421	287	242	103
	3	436	608	407	348	204	248
	4	226	333	256	209	62	193
	5	15	23	36	31	7	58
	6	108	173	62	97	57	54
	7	162	285	244	326	100	232
	8	105	306	418	545	65	337
	9	18	36	63	96	3	76
	Total	2,093	3,447	2,826	3,308	946	2,869
1	0	8	8	3	5	2	8
	1	483	891	1,077	1,578	194	1,589
	2	872	1,106	728	598	341	209
	3	615	752	685	594	249	410
	4	322	533	458	438	103	484
	5	36	64	76	91	19	106
	6	144	183	150	119	53	86
	7	147	221	201	263	63	216
	8	73	207	355	532	29	418
	9	5	32	51	84	7	52
	Total	2,705	3,997	3,784	4,302	1,060	3,578

Notes:
0: Homeless, 1: Owner 2:Living with parents, 3:Tenant, pre-furnished property, 4: Tenant, unfurnished property, 5: Council house, 6: Joint tenant, 7: Joint ownership, 8: Mortgage, 9:Owner with encumbrance, 10:Other

Exhibit 10: Education/Country Type Counts Breakdown by Target Class

Education/Country Breakout Categories
Defaulted: 1
Not Defaulted: 0

Education	Country	Target Class	
		0	1
-1	EE	201	2
	ES		2
	FI	2,048	185
	Total	2,249	189
0	EE		8
	Total		8
1	EE	12,718	4,819
	ES	460	1,650
	FI	5,869	2,878
	Total	19,047	9,347
2	EE	2,079	2,490
	ES	131	654
	FI	288	798
	SK		4
	Total	2,498	3,946
3	EE	18,943	7,073
	ES	677	2,087
	FI	23,756	10,516
	SK	1	35
	Total	43,377	19,711
4	EE	44,575	17,282
	ES	2,592	7,265
	FI	5,687	3,713
	SK	13	175
	Total	52,867	28,435
5	EE	20,076	5,569

Notes:
1:Primary education, 2:Basic education, 3:Vocational education, 4:Secondary education, 5:Higher education

Exhibit 11: Amount of Previous Credit Breakdown by Target Class

Amount of Previous Credit Breakout
Defaulted: 1
Not Defaulted: 0

No Of Previous Loans Before Loan	Target Class	
	0	1
0	0	0
1	32,686	16,216
2	38,536	17,124
3	35,139	13,671
4	30,320	10,444
5	25,585	8,385
6	21,192	6,600
7	17,682	5,404
8	15,000	4,184
9	12,474	3,402
10	10,060	2,440
Grand Total	238,674	87,870

Exhibit 12: Days to Payments Percentage of Total Breakdown by Target Class

Days to Payments Percentage of Total by Target Class
Defaulted: 1
Non Defaulted: 0

Active Late Category	Target Class		
	0	1	Grand Total
0-7	95.84%	4.16%	100.00%
8-15	97.51%	2.49%	100.00%
16-30	86.07%	13.93%	100.00%
31-60	82.02%	17.98%	100.00%
61-90	60.72%	39.28%	100.00%
91-120	33.15%	66.85%	100.00%
121-150	4.34%	95.66%	100.00%
151-180	2.94%	97.06%	100.00%
180+	0.85%	99.15%	100.00%

4.0 Feature Evaluation/Extraction

The following further data exploration activities are described in this section. It includes a discussion on the following:

1) Missing value analysis;
2) Multi collinearity effects;
3) Correlation between predictor variable and target variable; and
4) PCA analysis to identify how many principal components are able to explain the variance amongst the various continuous variables.

4.1 Missing Value Analysis

Of the 111 predictor variables, several of the categorical variables that do not have numerical value (e.g., Loan Id, Loan Number, etc.) were initially removed from the dataset.

Following this initial data cleansing effort, further analysis was conducted to evaluate features that had more than 10 pct missing data. The features that have more than 10 pct missing data are presented in Table 2. Given the large amount of predictor variables available in the dataset, these features were removed from the dataset. As can be seen later in the modeling effort, removal of these variables does not have significant effect on the prediction performance of the models.

Also note some of these variables such as Planned Principal Post Default, Planned Interest Post Default, those related to Recovery, those related to WriteOffs, and EAD1 and EAD2 should be removed as they were recorded following default and should not be used to predict the target class, and would have been removed from the dataset regardless of the number of missing values.

Table 2: Features with More than 10 Pct Missing Values

Features	Percentage of Total Missing
ContractEndDate	56.58%
DateOfBirth	100.00%
NrOfDependants	84.99%
WorkExperience	84.60%
PlannedPrincipalTillDate	77.04%
CurrentDebtDaysPrimary	63.27%
DebtOccuredOn	63.27%
CurrentDebtDaysSecondary	59.70%
DebtOccuredOnForSecondary	59.70%
PlannedPrincipalPostDefault	66.01%
PlannedInterestPostDefault	66.01%
EAD1	66.01%
EAD2	66.01%
PrincipalRecovery	66.01%
InterestRecovery	66.01%
RecoveryStage	41.56%
StageActiveSince	38.00%
EL_V1	94.55%
Rating_V1	94.55%
Rating_V2	89.40%
ActiveLateCategory	63.51%
WorseLateCategory	34.52%
CreditScoreEsMicroL	13.49%
CreditScoreEsEquifaxRisk	94.85%
CreditScoreFiAsiakasTietoRiskGrade	68.98%
CreditScoreEeMini	45.17%
PrincipalWriteOffs	63.55%
InterestAndPenaltyWriteOffs	63.55%
InterestAndPenaltyBalance	26.65%
PreviousRepaymentsBeforeLoan	37.12%
PreviousEarlyRepaymentsBefoleLoan	74.85%
GracePeriodStart	75.01%
GracePeriodEnd	75.01%
NextPaymentDate	59.58%
NextPaymentNr	39.82%
NrOfScheduledPayments	39.82%
ReScheduledOn	62.77%
PrincipalDebtServicingCost	63.55%
InterestAndPenaltyDebtServicingCost	63.55%
ActiveLateLastPaymentCategory	59.70%

Following the removal of the features noted above, the "surviving" features were further evaluated for "missingness". The percentage of datapoints missing for these features were less than 10% of the total data points. The actual numbers of the missing data points for the features that had missing values are presented on Exhibit 13.

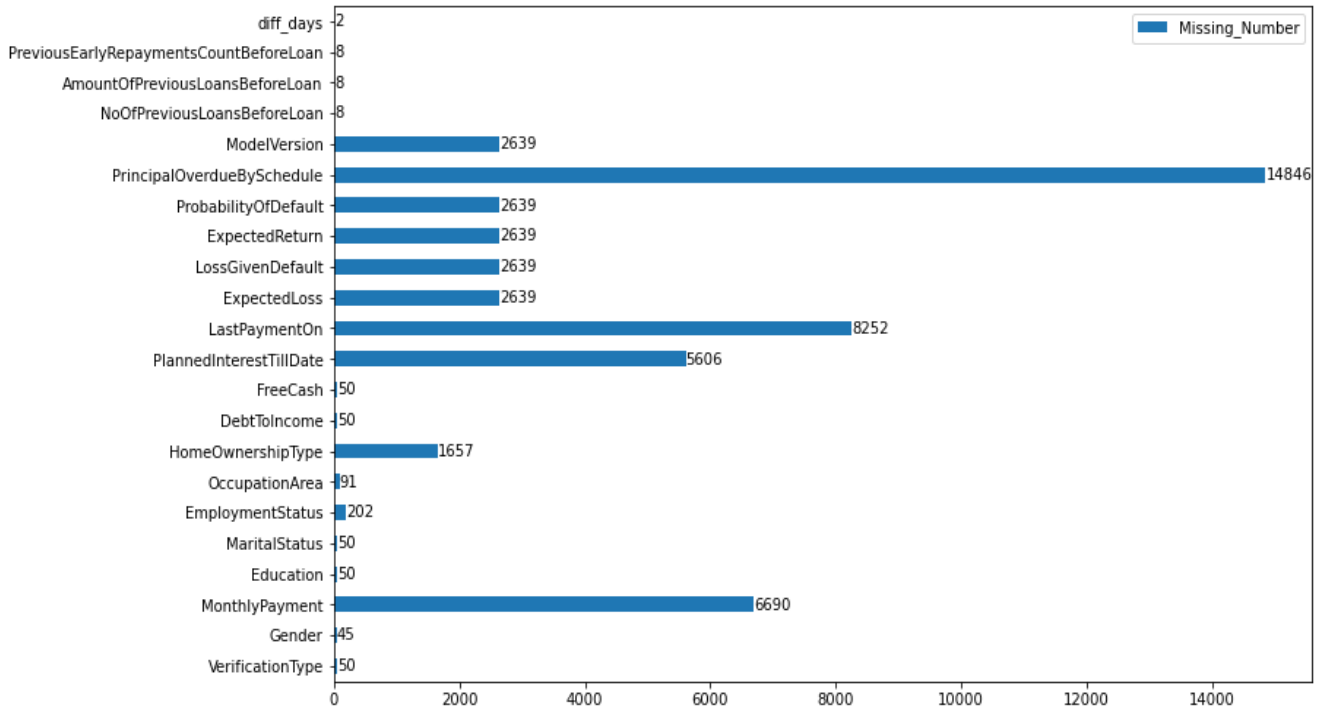
Following the removal of the rows in the dataset with these missing values, the total number of data points remaining in the dataset was 211,240, which is 10.90% less than the original number of 237,223 in the dataset.

The breakdown by target class of the final dataset used in the modeling is presented in Table 3 below:

Table 3:
Target Class Breakdown, Final Dataset

Target Class	Count of Target Class	% of Total Count of Target Class
0	137,895	65.28%
1	73,345	34.72%
Total	211,240	100.00%

Exhibit 13: Missing Values Count for Surviving Features



The distribution of the dataset and the breakdown by target class are similar to the original dataset with the missing values in it (see Table 1). A total of 58 predictor features survived in the final dataset used for further analysis and modeling. Final data cleansing consisted of "minmax" scaling of the continuous variables and one hot dummy encoding (Heaton, J, 2022a) of the categorical variables, where necessary. Note that several of the categorical variables were already assigned "ordinal" scores and did not require dummy encoding. Following this data cleansing and the one hot dummy encoding, 71 predictor variables were generated for the modeling effort.

4.2 Correlation Analysis

Analysis was conducted to assess for multi-collinearity of the surviving predictor variables. This analysis was conducted on unscaled continuous variable data. The predictor variables that have correlation coefficient greater than 0.75 between each other are presented on Table 4. Only 2 pairs (or 4 variables) of the 71 surviving predictor variables have correlation coefficient exceeding 0.9.

These two pairs are marital status and employment status and amount and applied amount. Applied amount is the actual amount requested by the consumer and the amount is the amount of loan that was authorized by the financial institution.

Table 4: Correlation Coefficients Between Variables

Variable_1	Variable_2	Correlation Coeff
MaritalStatus	DebtToIncome	0.767
DebtToIncome	MaritalStatus	0.767
NoOfPreviousLoansBeforeLoan	AmountOfPreviousLoansBeforeLoan	0.77
AmountOfPreviousLoansBeforeLoan	NoOfPreviousLoansBeforeLoan	0.77
UseOfLoan	MaritalStatus	0.774
MaritalStatus	UseOfLoan	0.774
MaritalStatus	OccupationArea	0.774
OccupationArea	MaritalStatus	0.774
Interest	ProbabilityOfDefault	0.785
ProbabilityOfDefault	Interest	0.785
EmploymentStatus	DebtToIncome	0.787
DebtToIncome	EmploymentStatus	0.787
AppliedAmount	MonthlyPayment	0.79
MonthlyPayment	AppliedAmount	0.79
UseOfLoan	EmploymentStatus	0.791
EmploymentStatus	UseOfLoan	0.791
EmploymentStatus	OccupationArea	0.791
OccupationArea	EmploymentStatus	0.791
Interest	ExpectedLoss	0.799
ExpectedLoss	Interest	0.799
ExpectedLoss	ProbabilityOfDefault	0.858
ProbabilityOfDefault	ExpectedLoss	0.858
MaritalStatus	EmploymentStatus	0.928
EmploymentStatus	MaritalStatus	0.928
AppliedAmount	Amount	0.947
Amount	AppliedAmount	0.947

Because the correlation coefficients outside of these 4 variables are not higher than 0.9 (see Table 4), multi-collinearity effects between predictor variables are not considered significant and none of the surviving variables were removed from further analysis.

Also evaluated was the correlation coefficient between the predictor variable and the target variable, and, as expected, a few of the predictor variables, Expected Loss, Probability of Default, Principal_Overdue_by_Schedule, and Status_Late have correlation coefficients exceeding 0.4 (see Table 5). These variables are estimates made during the application process and during loan servicing and not generated following default and hence were not removed from the predictor variable set.

Table 5: Correlation Coefficients Between Variables and Target Variable

Variable_Name	Defaulted
Rating_C	-0.182
Status_Repaid	-0.175
Rating_B	-0.136
AmountOfPreviousLoansBeforeLoan	-0.120
PrincipalPaymentsMade	-0.118
NoOfPreviousLoansBeforeLoan	-0.117
ModelVersion	-0.108
LossGivenDefault	-0.098
Rating_D	-0.080
Rating_AA	-0.070
EmploymentDurationCurrentEmployer_UpTo5Years	-0.067
EmploymentDurationCurrentEmployer_Other	-0.049
diff_days	-0.035
Country_FI	-0.032
MonthlyPaymentDay	-0.029
LoanDuration	-0.016
InterestAndPenaltyPaymentsMade	-0.011
LiabilitiesTotal	0.005
EmploymentDurationCurrentEmployer_UpTo1Year	0.005
PreviousEarlyRepaymentsCountBeforeLoan	0.013
EmploymentDurationCurrentEmployer_Retiree	0.013
IncomeFromLeavePay	0.019
Education	0.020
IncomeOther	0.032
HomeOwnershipType	0.033
EmploymentDurationCurrentEmployer_TrialPeriod	0.035
Amount	0.041
Country_SK	0.045
IncomeFromChildSupport	0.046
IncomeFromSocialWelfare	0.046
ExistingLiabilities	0.049
Restructured_True	0.068
AppliedAmount	0.075
EmploymentDurationCurrentEmployer_UpTo4Years	0.076
IncomeFromFamilyAllowance	0.082
FreeCash	0.084
IncomeFromPension	0.085

Table 5 Continued: Correlation Coefficients Between Variables and Target Variable

Variable_Name	Defaulted
EmploymentDurationCurrentEmployer_UpTo3Years	0.091
NewCreditCustomer_True	0.102
EmploymentDurationCurrentEmployer_UpTo2Years	0.108
PrincipalBalance	0.111
RefinanceLiabilities	0.119
Rating_E	0.120
IncomeFromPrincipalEmployer	0.144
MonthlyPayment	0.160
PlannedInterestTillDate	0.187
OccupationArea	0.237
DebtToIncome	0.245
Rating_HR	0.249
UseOfLoan	0.254
Rating_F	0.256
ExpectedReturn	0.273
ActiveScheduleFirstPaymentReached_True	0.277
MaritalStatus	0.282
EmploymentStatus	0.286
Country_ES	0.298
Interest	0.354
ExpectedLoss	0.409
ProbabilityOfDefault	0.432
PrincipalOverdueBySchedule	0.487
Status_Late	0.758
Defaulted	1.000

4.3 Principal Component Analysis

A Principal Component Analysis (PCA) analysis was conducted to perform exploratory analysis and to evaluate whether the variance in the predictor variables and separation in the target class variables can be explained by reducing dimensions of the predictor variables. The scaling was performed with standard scaler.

An analysis was conducted using only 5,000 dataset points. This analysis indicates that 50% of the variance can be explained with 5 principal components (see Exhibit 14).

Separability in the target class is not clearly discernable when 3 principal components are evaluated (see Exhibit 15).

Exhibit 14: Explained Variance vs Principal Component No.

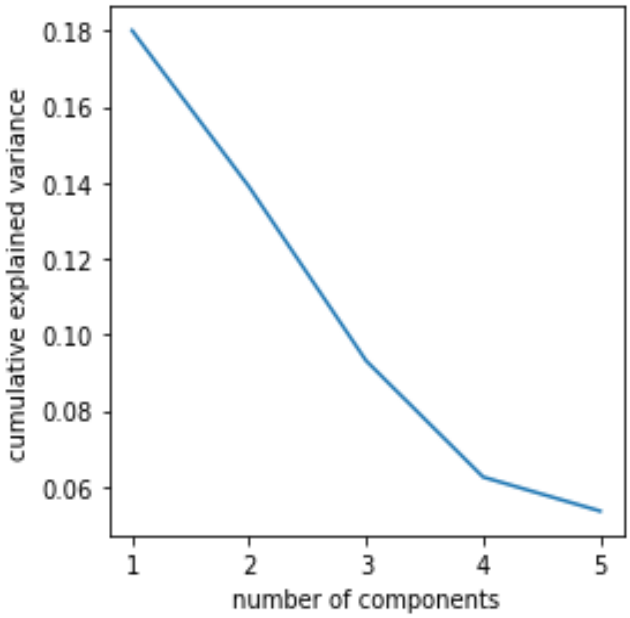
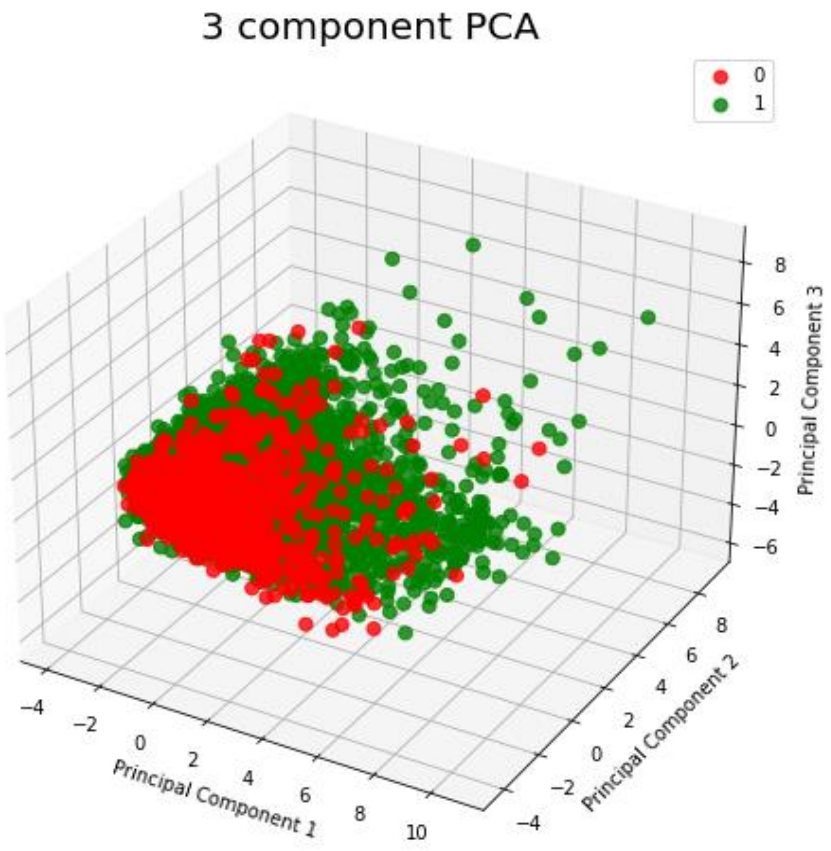


Exhibit 15: Target Class Separation from Three Principal Components



A PCA Bi Plot results from this analysis is presented on Exhibit 16. Based on the "vector" representation of some of the features, it does appear that the first two components may be a reasonable assimilator of a limited set of the continuous predictor variables.

Given the limited separability in target classes noted in Exhibit 15 and a large number of categorical variables (greater than 50 pct of surviving predictor variables), PCA components were not included in the modeling effort and the 71 surviving predictor variables were carried forward for the modeling effort.

Exhibit 16: PCA Bi Plot

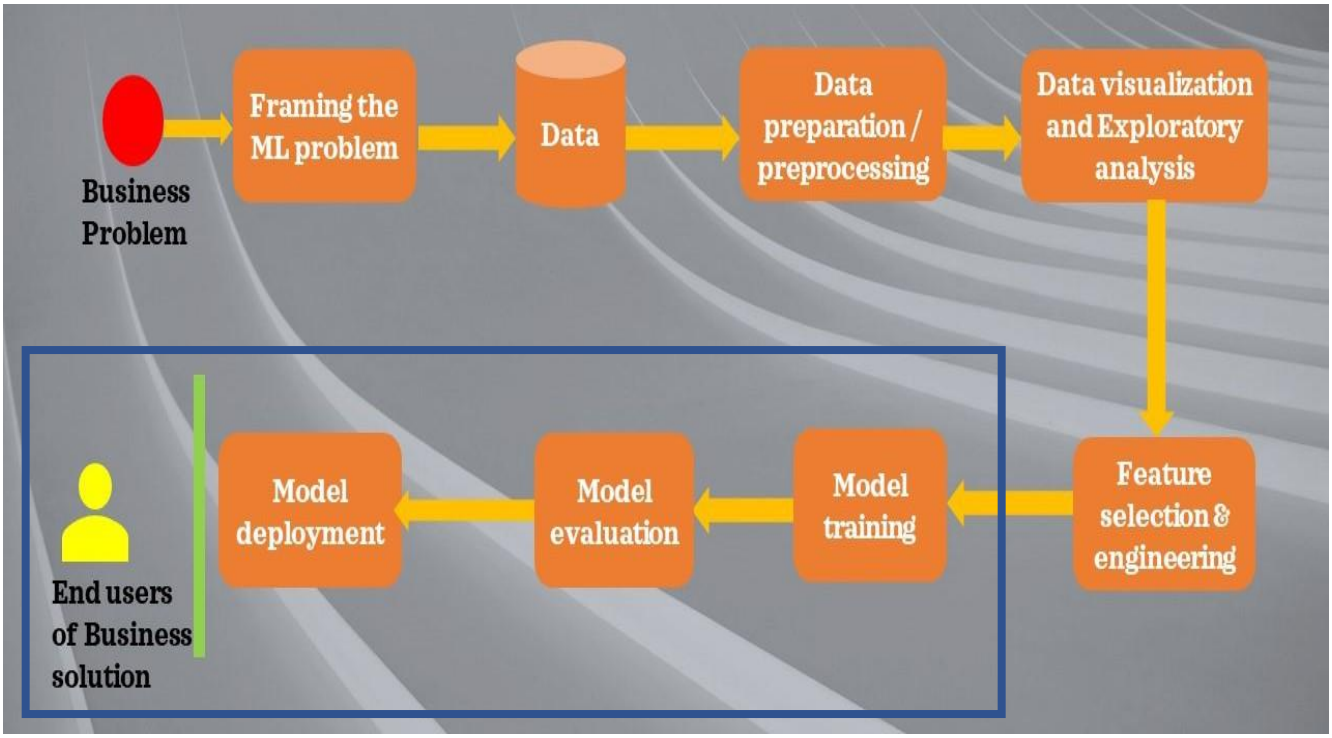
5.0 Machine Learning Modeling

Classification modeling was conducted using the final dataset (from Table 3) that contains 71 predictor variables and 1 target variable (see blue rectangle in schematic below for the work components in this phase). Python packages sklearn and tensorflow/keras were utilized for the development of the machine learning models. PyTorch with a PySyft wrapper was utilized for the remote (federated) machine learning phase of the project.

The final dataset was split into train (80%) and test (20%) components using sklearn's in built functions. The sklearn models were trained with 5-fold cross validation on the train portion of the dataset and its performance was evaluated on the test portion of the dataset.

For Tensorflow/keras, the model was first trained and tested on then full dataset with default parameters without cross validation. For the cross validation and testing portion of the modeling, because of time complexity, the model was trained with 3-fold cross validation on 10% of the dataset. This fraction was split into 80% train and test components.

The focus of PyTorch and PySft modeling effort was to identify the process to be used to train, build, and test the model on a remote dataset and to evaluate its effectiveness in achieving results that are comparable to the other models. Accordingly, to reduce the time required to run the models, 5% of the final dataset was used in the modeling effort. Similar to the workflow for the other models, this fraction of the final dataset was split into train (80%) and test (20%) components.



5.1 Logistic Regression

5.1.1 Model Overview and Results

Logistic regression models a relationship between predictor variables and a categorical response variable (James G, 2017). The log odds per logistic regression for a binary classification problem is given as follows:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta 0 + \beta 1\, X \quad \text{(James G, 2017)}$$

Where: p(X) is the probability that takes a value between 0 and 1, and is used as a predictor for one of the two classes for a binary classification problem based on its value. If the value is between 0 and 0.5, it is assigned to class 0; otherwise it is assigned to class 1.

sklearn's logistic regression module was used to model the logistic regression on the final dataset (sklearn-a). The modeling was conducted as follows:

class sklearn.linear_model.Logistic Regression(*penalty,C, solver='lbfgs', max_iter=200,* x *l1_ratio*).

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 17. Results are provided on Exhibits 18-21.

Exhibit 17: LR Model Hyperparameters

Hyper-parameter	Range	Best Value
Penalty	L1, L2, Elasticnet	L1
C	1,5,10	5
Solver	Lbfgs, liblinear, and saga	liblinear
L1_ratio	0.2,0.6	Ignored

Exhibit 18: LR Model Grid Search CV Results

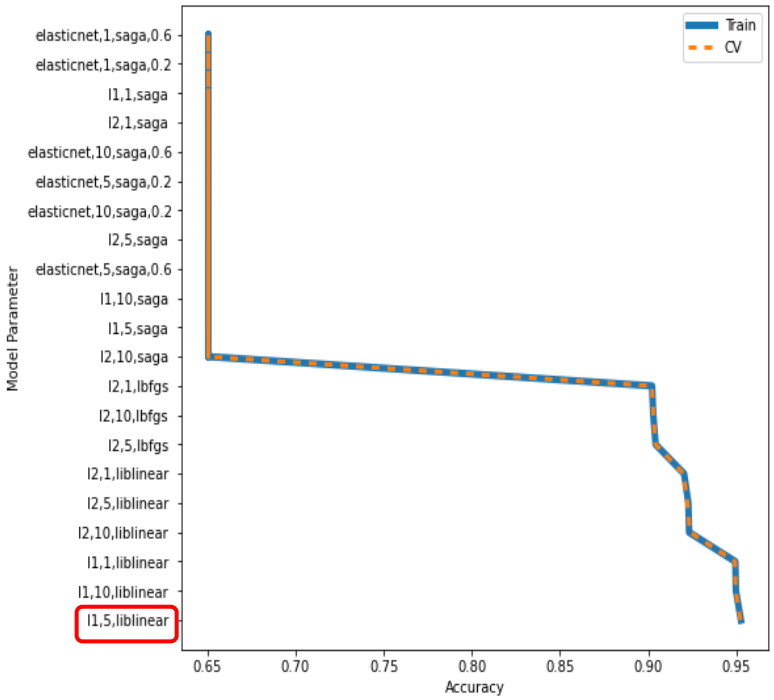


Exhibit 19: Performance Evaluation: Logistic Regression

Confusion Matrix, Test Dataset Following Tuning:

	Predicted No	Predicted Yes
Actual No	26,280	907
Actual Yes	928	13,687

Parameter	Value Following Tuning
RMSE	0.209
Precision	0.938
Accuracy	0.956
Recall	0.936
F1_Score	0.937

Exhibit 20: ROC Curve: Logistic Regression/Best Model Following Tuning

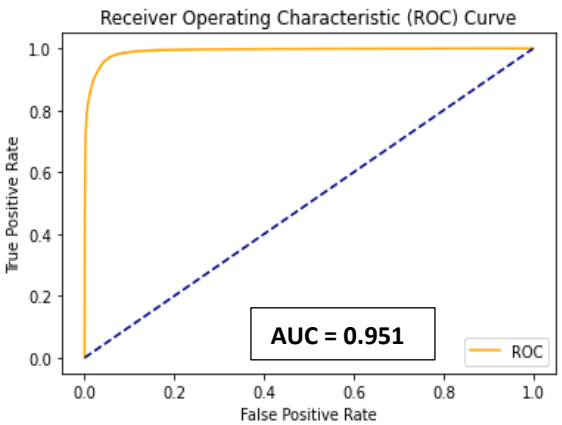
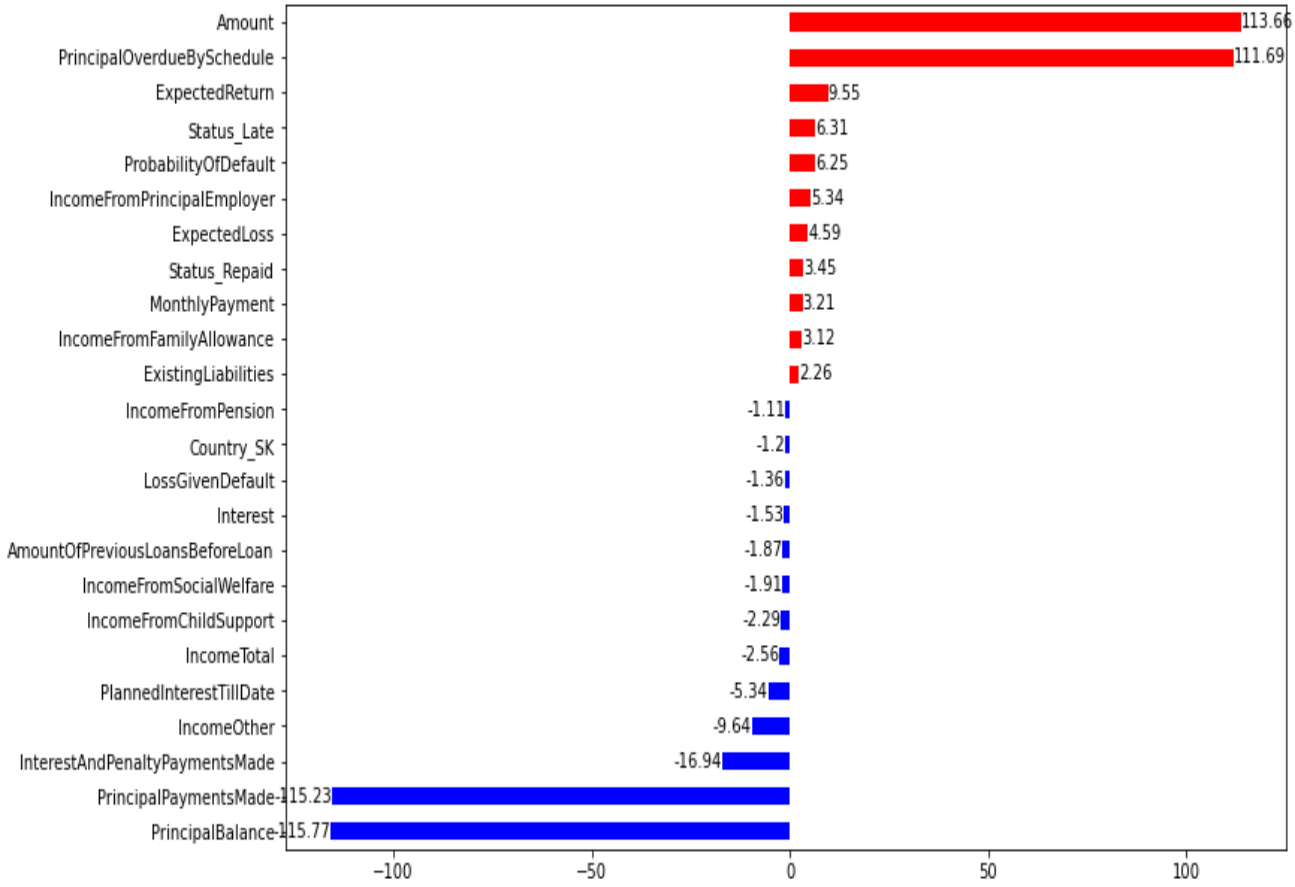


Exhibit 21: Important Features Coefficients: Logistic Regression/Best Model Following Tuning



5.1.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.952 (Exhibit 18) was obtained with the best values of hyperparameters noted on Exhibit 17. The best model was evaluated on the test dataset using these best model parameters. The results from this evaluation indicate that precision, recall, accuracy, F_1 score were all higher than 0.9 (Exhibit 19). The area under the curve of the receiver operating characteristic curve was 0.951 (Exhibit 20), which indicates that the model is effective in separating the target class between 0 and 1.

Top 5 positive coefficients (i.e., β1 values) were obtained for loan amount, *PrincipalOverduebySchedule*, *ExpectedReturn*, *StatusLate*, and *ProbabilityOfDefault*. Top 5 negative coefficients were obtained for *PrincipalBalance*, *PrincipalPaymentMade*, I*nterestAndPenaltyPaymentsMade*, *IncomeOther*, and *PlannedInterestTillDate* (see Exhibit 21). Positive coefficients drive the target class to 1 and negative coefficients drive the target Class to 0. Exhibit 21 can be used for interpretation of the best "logistic regression" model and to identify the features that drove the classification prediction in this model.

5.2 Multinomial Bayes

5.2.1 Model Overview and Results

Multinomial Bayes models help predict that particular observation belongs to a certain class (Y=k) based on the prior probability of the occurrence of a class (πk) and the density function of X (fk(x)) that comes from an observation comes from that kth class:

$$\Pr(Y = k|X = x) = \frac{\pi k f_k(x)}{\sum_1^l \pi l \, f_l(x)}$$ (Hastie, T., 2017)

The denominator is ignored in the calculation.

sklearn's multinominal bayes module was used to model the logistic regression on the final dataset (sklearn-b). The modeling was as follows:

class sklearn.naive_bayes.**MultinomialNB**(**, alpha=1.0, fit_prior=True, class_prior=None*)

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 22. Results are provided on Exhibits 23-26.

Exhibit 22: MNB Model Hyperparameters

Hyper--parameter	Range	Best Value
Alpha	1E-4, 1E-2, 1E-1, 1	1

Exhibit 23: MNB Grid Search CV Results

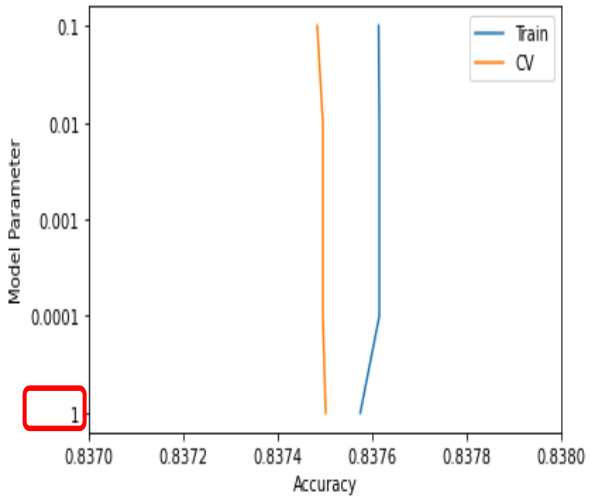


Exhibit 24: Performance Evaluation: Multinomial Bayes

Confusion Matrix, Test Dataset Following Tuning:

	Predicted No Default	Predicted Yes Default
Actual No Default	24,283	2,904
Actual Yes Default	928	13,687

Parameter	Value Following Tuning
RMSE	0.399
Precision	0.789
Accuracy	0.841
Recall	0.743
F1_Score	0.765

Exhibit 25: ROC Curve: Multinomial Bayes/Best Model Following Tuning

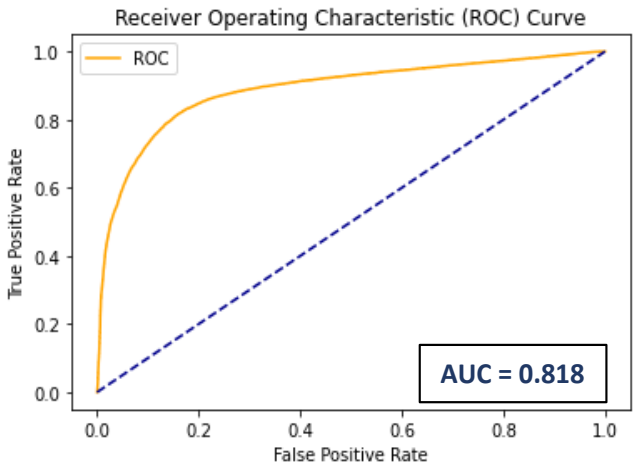
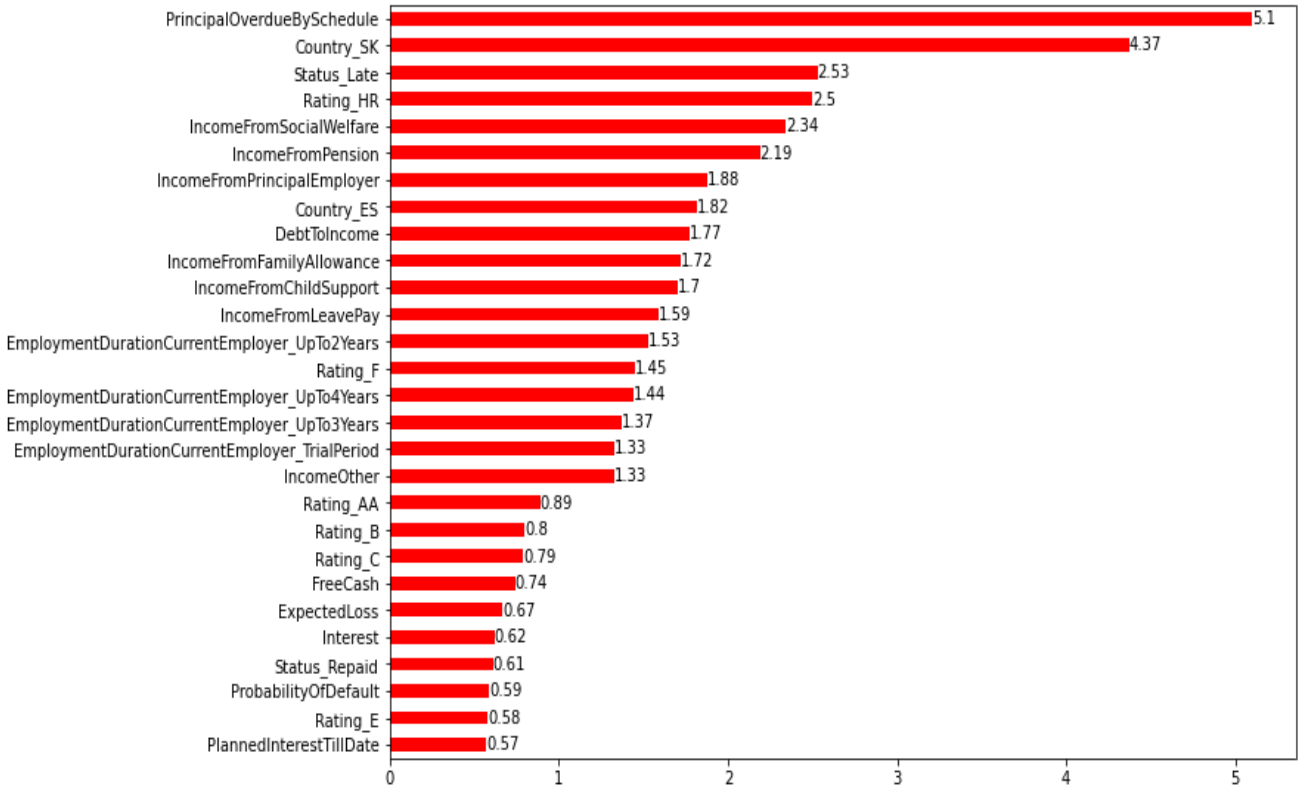


Exhibit 26: Important Features Coefficients Difference Between Classes Naïve Bayes/Best Model Following Tuning



5.2.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.838 (Exhibit 23) was obtained with the best values of hyperparameters noted on Exhibit 22. The best model was evaluated on the test dataset using these best model parameters. The results from this evaluation indicate that precision, recall, accuracy, F_1 score were all lower than 0.9 (between 0.7 and 0.9) and were lower than the other models evaluated in this study (Exhibit 24). The area under the curve of the receiver operating characteristic curve was 0.818 (Exhibit 25), which indicates that the model is less effective than the other evaluated models in separating the target class between 0 and 1.

The model provides estimates of the probability that a feature predicts a class 0 and a class 1 based on its values. Exhibit 26 depicts estimates of the absolute difference between these values for the features used in the modeling. Higher values of these estimates can be used an indicator of the relative importance of the feature in this model for separating the result for the target into it two disparate classes (0 or 1).

5.3 Decision Tree

5.3.1 Model Overview and Results

Decision Tree is a Supervised learning algorithm that is used for classification. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

Decision tree classifiers use either Gini Impurity Index or Information Gain (entropy) at a given node to create a split in the decision tree. Features that have the lowest Gini Impurity Index or highest Information Gain are placed at a given node.

sklearn's Decisiontree Classifier module was used to model the logistic regression on the final dataset (sklearn-c). The modeling was as follows:

class sklearn.tree.DecisionTreeClassifier (*criterion*, *max_depth*)

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 27. Results are provided on Exhibits 28-31.

Exhibit 27: Decision Tree Model Hyperparameters

Hyper--parameter	Range	Best Value
Criterion	Gini and Entropy	Entropy
Max_Depth	5,10,20	20

Exhibit 28: Decision Tree Grid Search CV Results

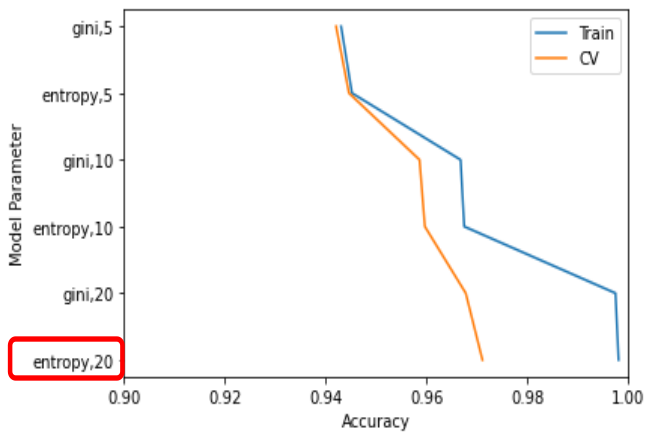


Exhibit 29: Performance Evaluation: Decision Tree

Confusion Matrix, Test Dataset Following Tuning:

	Predicted No	Predicted Yes
Actual No	26,663	554
Actual Yes	591	14,024

Parameter	Value Following Tuning
RMSE	0.166
Precision	0.962
Accuracy	0.973
Recall	0.960
F1_Score	0.961

Exhibit 30: ROC Curve: Decision Tree/Best Model Following Tuning

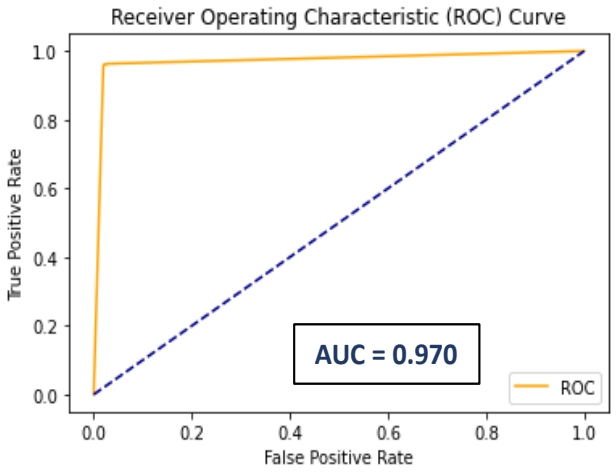
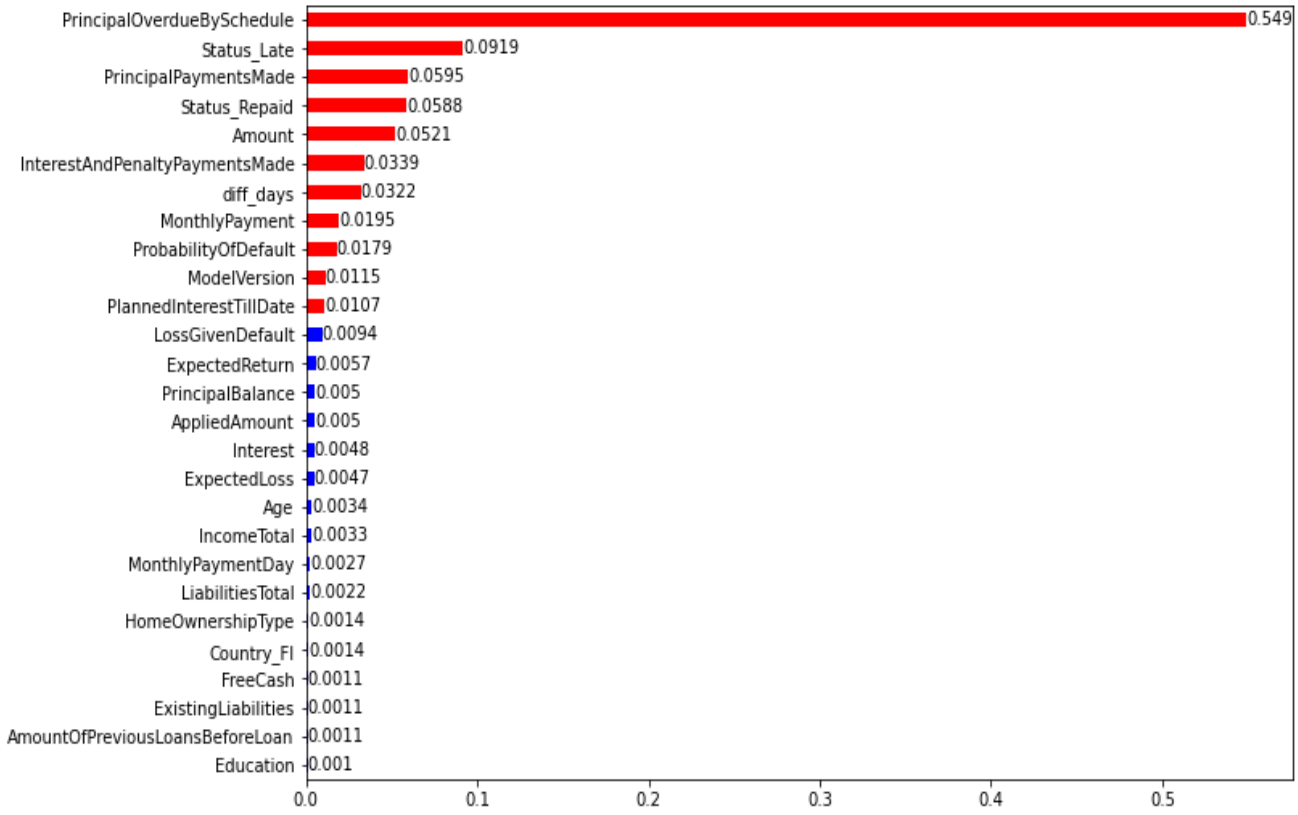


Exhibit 31: Features Importance Decision Tree/Best Model Following Tuning



5.3.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.971 (Exhibit 28) was obtained with the best values of hyperparameters noted on Exhibit 27. The best model was evaluated on the test dataset using these best model parameters. The results from this evaluation indicate that precision, recall, accuracy, F_1 score were all higher than 0.9 (Exhibit 29). The area under the curve of the receiver operating characteristic curve was 0.970 (Exhibit 30), which indicates that the model is effective in separating the target class between 0 and 1.

The five features with the most importance to model prediction were *PrincipalOverduebySchedule*, *StatusLate*, PrincipalPaymentsMade, *StatusRepaid*, and *loan amount* (see Exhibit 31). Exhibit 31 can be used for interpretation of the best "decision tree" model and to identify the features that drove the classification prediction in this model.

5.4 Ensemble Forests

5.4.1 Model Overview and Results

Ensemble AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

For our analysis, the Ensemble Model was built on a base estimator of a Decision Tree Classifier with a maximum depth of 1. The Decision Tree Classifier is considered a weak classifier as it only has a maximum depth of 1. In this study, sklearn's Adaboot classifier that implements the algorithm known as AdaBoost-SAMME is utilized (Zhu, H., 2009). Despite the classifier much weaker than the Decision Tree Classifier (max_depth of 20 in Section 5.3), the results of this model do not suffer much in comparison.

sklearn's ensemble AdaBoost Classifier module was used to model the logistic regression on the final dataset (sklearn-d). The modeling was as follows:

class sklearn.ensemble.AdaBoostClassifier (*n_estimators, learning_rate=1.0*)

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 32. Results are provided on Exhibits 33-36.

Exhibit 32: Ensemble Forests Model Hyperparameters

Hyper--parameter	Range	Best Value
N_estimators	5,10,20, 50,100	100
L_rate	.1, .5, 1.0, 5.0,10.0	1.0

Exhibit 33: Ensemble Forests Grid Search CV Results

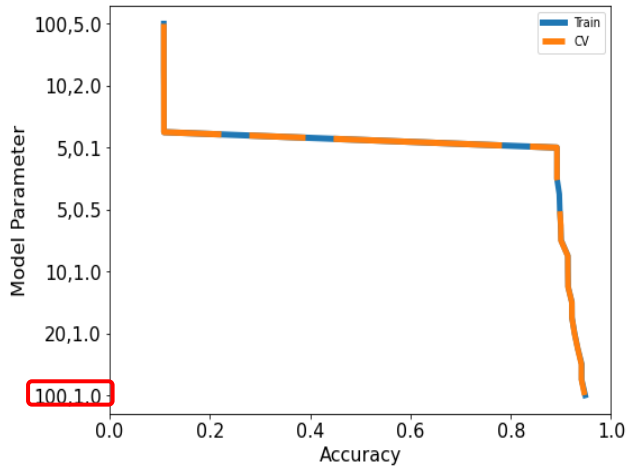


Exhibit 34: Performance Evaluation: Ensemble Forests

Confusion Matrix, Test Dataset Following Tuning:

	Predicted No	Predicted Yes
Actual No	26,238	949
Actual Yes	591	14,024

Parameter	Value Following Tuning
RMSE	0.231
Precision	0.934
Accuracy	0.947
Recall	0.913
F1_Score	0.923

Exhibit 35: ROC Curve: Ensemble Forests/Best Model Following Tuning

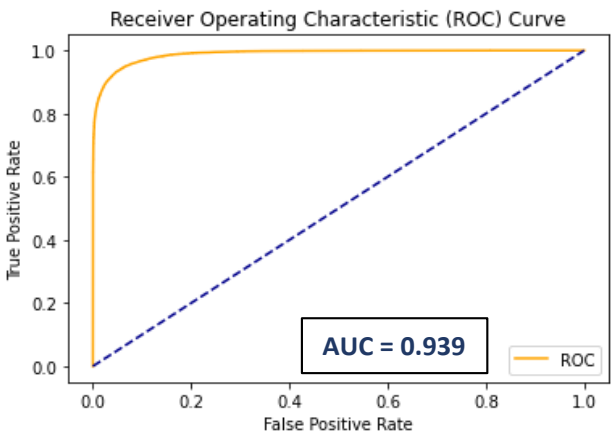
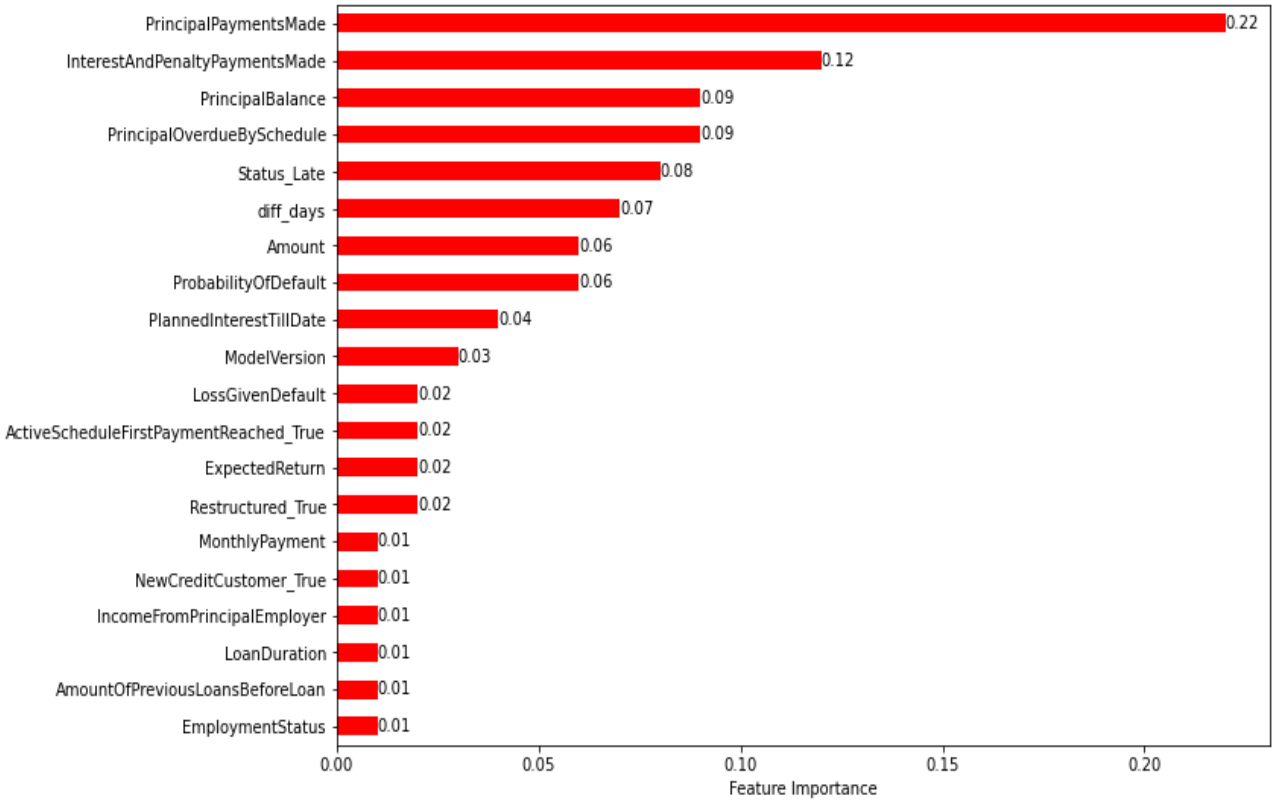


Exhibit 36: Features Importance Ensemble Forests/Best Model Following Tuning



5.4.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.947 (Exhibit 33) was obtained with the best values of hyperparameters noted on Exhibit 32. The best model was evaluated on the test dataset using these best model parameters. The results from this evaluation indicate that precision, recall, accuracy, F_1 score were marginally lower than the stronger and unboosted Decision Tree Classifier, but were all higher than 0.9 (Exhibit 34). The area under the curve of the receiver operating characteristic curve was 0.939 (Exhibit 35), which indicates that the model is effective in separating the target class between 0 and 1.

Despite the fact that this model boosted a much weaker Decision Tree Classifier than that utilized in Section 5.3, model results were comparable. It is worth noting that the strength of the weak Decision Tree Classifier boosted by this algorithm is much lower on the lower end for some hyperparameters (mean CV score of less than 0.2) when compared to the best model with l_rate of 1.0 and number of estimators of 100.

The five features with the most importance to model prediction were PrincipalPaymentsMade, *InterestandPenaltyPayment*Made, *PrincipalBalance*, *PrincipalOver DueBy Schedule*, and *StatusLate* (see Exhibit 36). Exhibit 36 can be used for interpretation of the best "ada-boost" model and to identify the features that drove the classification prediction in this model.

5.5 Random Forest

5.5.1 Model Overview and Results

Random forests or **random decision forests** is an ensemble learning method for classification that operates by constructing a multitude of decision trees at training time. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

sklearn's ensemble RandomForest Classifier module was used to model the logistic regression on the final dataset (sklearn-e). The default gini impurity criterion for feature selection at the nodes. Default max_depth was utilized, which allows the nodes to expand until all leaves are pure or until all leaves contain less than 2 samples required to split an internal node.

The modeling was conducted as follows:

class sklearn.ensemble.RandomForestClassifier(*n_estimators, l_rate)*

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 37. Results are provided on Exhibits 38-41.

Exhibit 37: Random Forests Model Hyperparameters

Hyper--parameter	Range	Best Value
N_estimators	5,10,20, 50,100	100
L_rate	.1, .5, 1.0, 5.0,10.0	1.0

Exhibit 38: Random Forests Grid Search CV Results

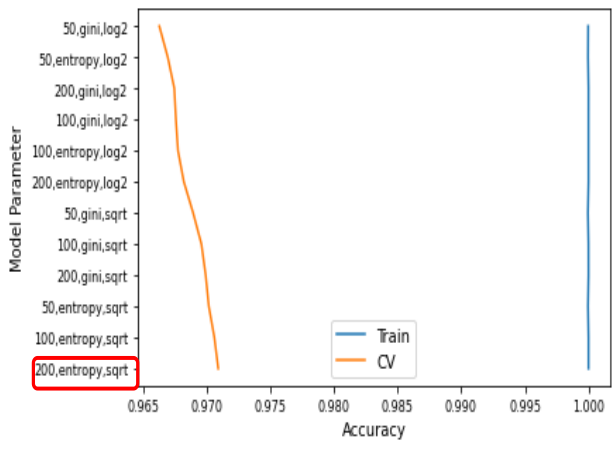


Exhibit 39: Performance Evaluation: Random Forest

Confusion Matrix, Test Dataset Following Tuning:

	Predicted No	Predicted Yes
Actual No	26,854	333
Actual Yes	826	13,789

Parameter	Value Following Tuning
RMSE	0.163
Precision	0.976
Accuracy	0.972
Recall	0.943
F1_Score	0.945

Exhibit 40: ROC Curve: Random Forest/Best Model Following Tuning

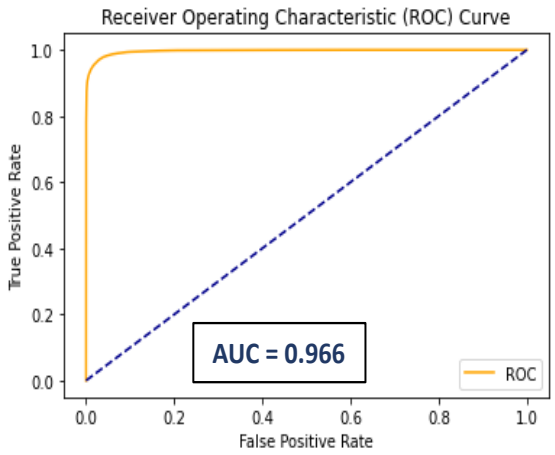
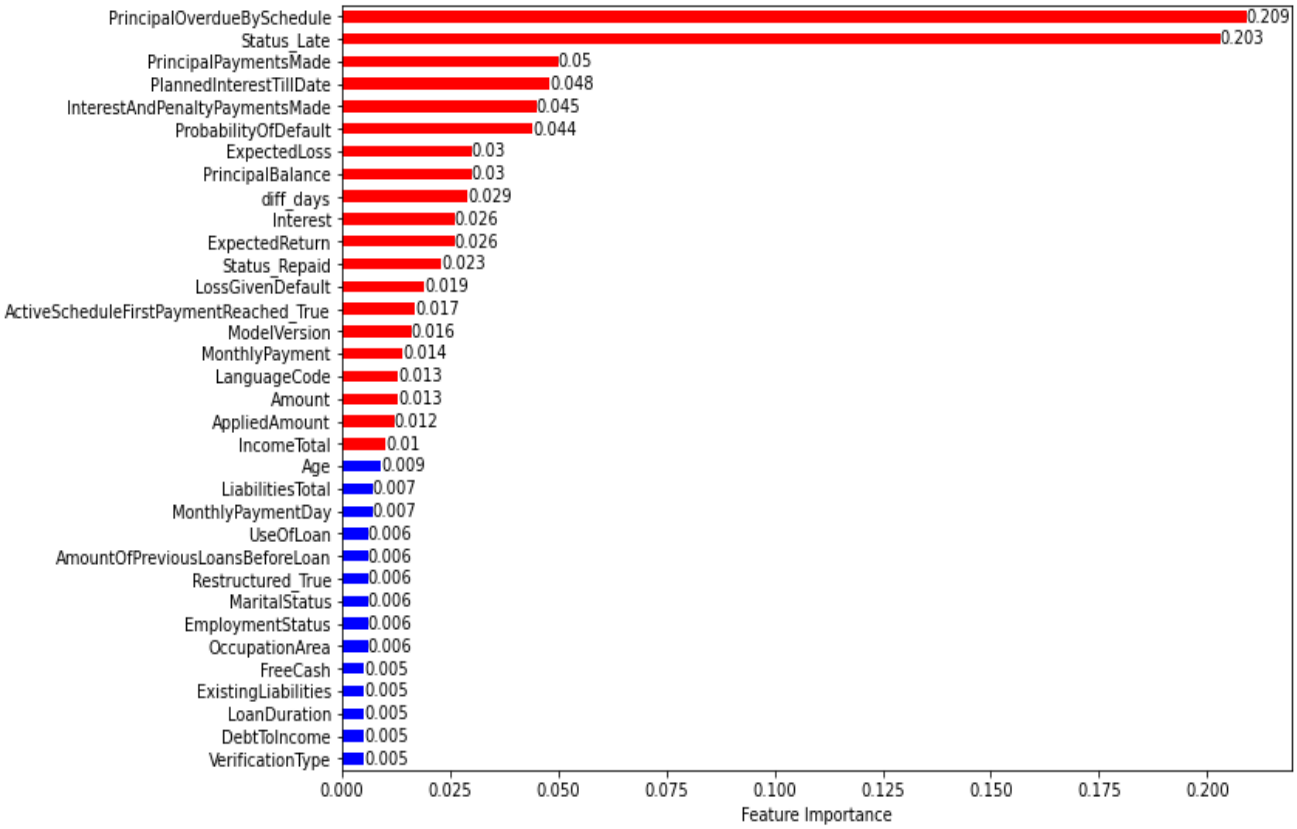


Exhibit 41: Important Features Importance Random Forest/Best Model Following Tuning



5.5.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.971 (Exhibit 38) was obtained with the best values of hyperparameters noted on Exhibit 37. The best model was evaluated on the test dataset using these best model parameters. The results from this evaluation indicate that precision, recall, accuracy, F_1 score were all higher than 0.9 (Exhibit 39). The area under the curve of the receiver operating characteristic curve was 0.966 (Exhibit 40), which indicates that the model is effective in separating the target class between 0 and 1.

The five features with the most importance to model prediction were *PrincipalOverduebySchedule*, *StatusLate*, PrincipalPaymentsMade, *PlannedInterestsTillDate*, and *InterestandPenaltyPaymentsMade* (see Exhibit 41). Exhibit 41 can be used for interpretation of the best "decision tree" model and to identify the features that drove the classification prediction in this model.

5.6 Deep Neural Network with Tensorflow/Keras

5.6.1 Model Overview and Results

Deep neural network model was developed using Tensorflow/Keras to train, validate, and test the final dataset. The architecture for the neural network was as follows:

1) Input layer with 71 neurons corresponding to 71 predictor variables.
2) 3 Hidden layers: Layer 1 with 100 neurons; Layer 2 with 50 neurons, and Layer 3 with 25 neurons. Each accepts the sum of the products of linear input of weights and input values and the output activation of each layer is set to be RELU.
3) 1 output layer with 1 neuron with a sigmoid activation.

The neural network was first trained on the entire final dataset, with a 80% train and 20% test split. Training was conducted using default parameters noted on Exhibit 42.

Following this initial preliminary run, the Tensorflow/Keras model was subjected to 3-Fold cross validation. skLearn's GridSearch CV was utilized to perform hyperparameter tuning during this phase. Exhibit 43 identifies the various hyperparameters chosen during this study and the results of the analyses. Note that because of the significant time complexity of this phase of the modeling, only a 10% fraction of the final dataset was used for training, validation, and testing. This fraction was then split into 80% train (and validation) and test components. The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 43. Results are provided on Exhibits 44-47.

Exhibits 48 and 49, show AUC for the receiver operating characteristic curves, for the default and the best "tuned" model, respectively.

Exhibit 42: Performance Evaluation: Keras/Tensorflow, Default Parameters
Confusion Matrix, Test Dataset:

	Predicted No Default	**Predicted Yes Default**
Actual No Default	26,101	1,086
Actual Yes Default	1,768	12,847

Parameter	**Value**
RMSE	0.261
Precision	0.922
Accuracy	0.931
Recall	0.879
F1_Score	0.900

Exhibit 43: Keras/Tensorflow Model Hyperparameters

Hyper--parameter	**Range**	**Best Value**
Optimizer	rmsprop, adam	adam
Inits	glorot_uniform, normal, unform	glorot_uniform
Epochs	50,100,150	150
Batches	5,20	5
Default: Only Change: Inits: random_normal; No Batch; Early Stopping Allowed		

Exhibit 44: Keras/Tensorflow Training Errors, Best Tuned Model Retraining

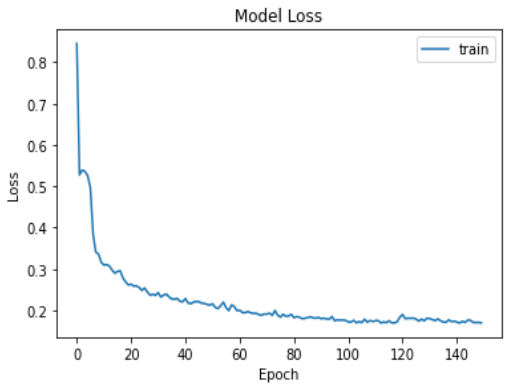


Exhibit 45: Keras/Tensorflow Training Accuracy, Best Model Retraining

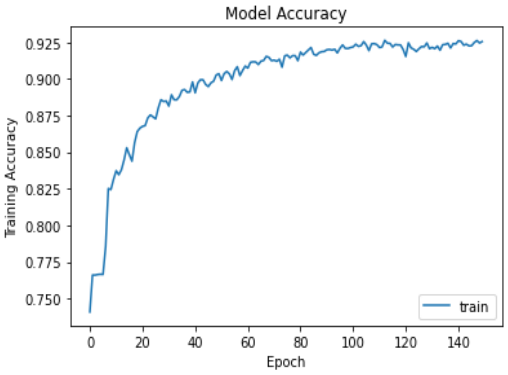


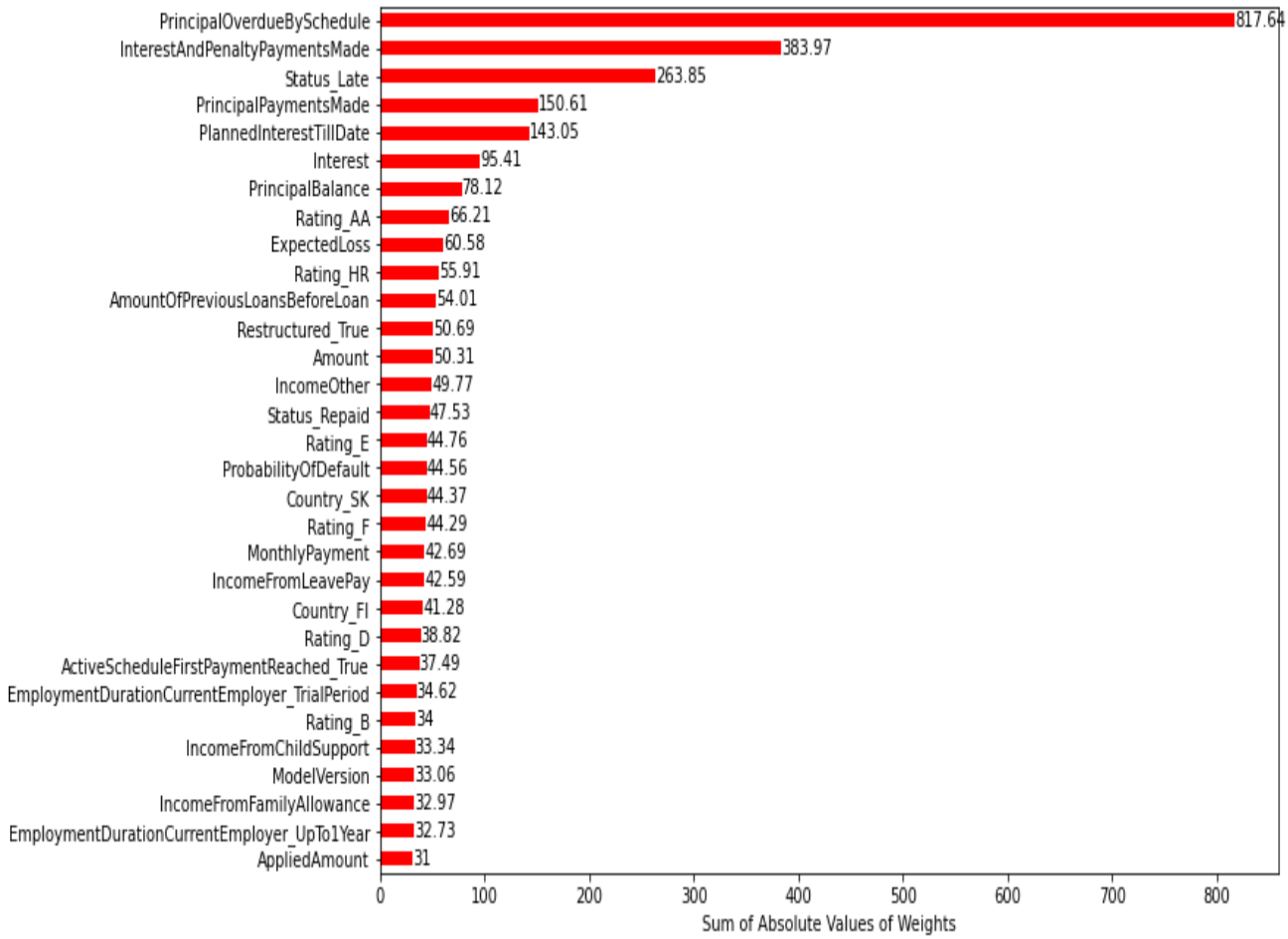
Exhibit 46: Performance Evaluation: Keras, Best Model Following Tuning

Confusion Matrix, Test Dataset Following Tuning (10% of Dataset):

	Predicted No	**Predicted Yes**
Actual No	630	308
Actual Yes	44	3,018

Parameter	**Value**
RMSE	0.249
Precision	0.907
Accuracy	0.912
Recall	0.986
F1_Score	0.945

Exhibit 47: Important Features Weights Neural Net/Best Model Following Tuning



5.6.2 Best Model Parameters

Based on the results of the tuning, best hyperparameters were selected (see Exhibit 43). The best model was evaluated on the test dataset using these best model parameters. The results from this evaluation indicate that precision, recall, accuracy,, and F_1 score were all higher than 0.9 (Exhibit 46). The area under the curve of the receiver operating characteristic curve was 0.980 (Exhibit 49), which is the highest of all the models evaluated during this study.

Note that the top rows from the final dataset were chosen for the training and testing. The distribution of the target class within this segment of the dataset was different from the overall distribution. Despite this, the AUC for the ROC curve was the highest for this model and its performance relative to other performance metrics were similar to the best "tree" models – decision tree and random forest.

It is worth noting that the performance of the neural network on the entire dataset using the default model was also reasonable. The AUC for the ROC curve on the test dataset for this model was also 0.98 (Exhibit 48). The precision, accuracy, F_1 score were greater than or equal to 0.9, and recall was marginally below 0.9. With hyperparameter tuning, it is conceivable that the results of the modeling on the entire dataset will likely be similar to those obtained from the 10% of the final dataset.

Features that had the highest final weights assigned to them on the best tuned model is presented in descending order of weights on Exhibit 47. The five features with the highest weights were *PrincipalOverduebySchedule*, *InterestandPenaltyPaymentsMade, StatusLate*, PrincipalPaymentsMade, and *PlannedInterestTillDate* (see Exhibit 47)

Exhibit 48: ROC Curve:

TensorFlow/Keras Default

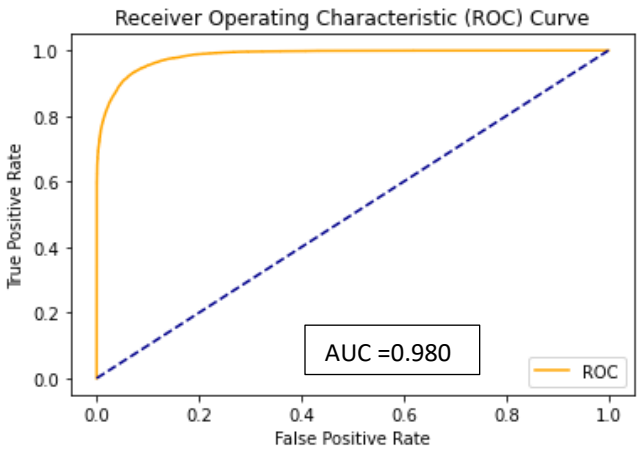
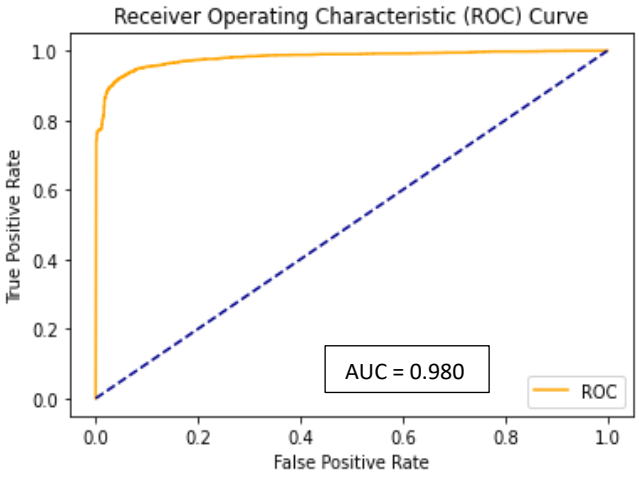


Exhibit 49: ROC Curve: Tensor Flow/Keras/Best Model Following Tuning



5.7 Federated Machine Learning with PyTorch and PySft

5.7.1 What is Federated Machine Learning and Why is it Relevant?

The traditional AI algorithms require centralizing data on a single machine or a server. The limitation of this approach is that all the data collected is sent back to the central server for processing before sending it back to the devices.

Federated Learning is a centralized server first approach. It is a distributed ML approach where multiple users collaboratively train a model. The concept of federated learning was first introduced in Google AI's 2017 blog. Here, remote raw data is distributed without being moved to a single server or data center. The central server selects a few remote nodes and sends the initialized version containing model parameters of an ML model to all the remote nodes. Each remote node now executes the model, trains the model on their local data, and has a local version of the model at each node. Once trained the models are then sent to the centralized server for aggregation and model evaluation.

Federated Learning leverages techniques from multiple research areas such as distributed systems, machine learning, and privacy. FL is best applied in situations where the on-device data is more relevant than the data that exists on servers. Federated learning provides edge devices with state of the art ML without centralizing the data and privacy by default. Thus it handles the unbalanced and non-Independent and Identically Distributed (IID) data of the features in mobile devices. A lot of data is generated from smartphones that can be used locally at the edge with on-device inference. Since the server does not need to be in the loop for every interaction with the locally generated data, this enables fast working with battery saving and better data privacy.

For this study, Facebook's PyTorch with a PySyft wrapper was utilized to perform a "test" run for the execution of federated ML. Process and connection layouts are depicted on Exhibits 50 and 51, respectively.

Exhibit 50: Federated ML Process Layout

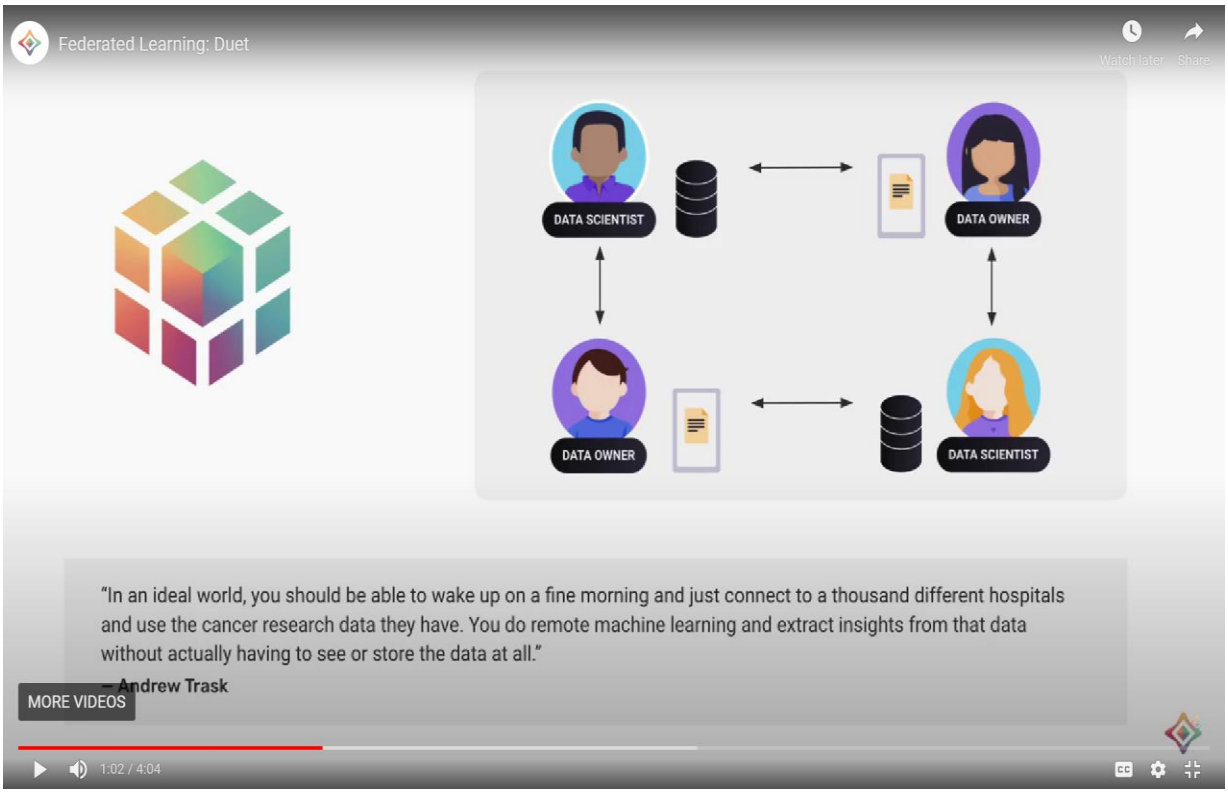
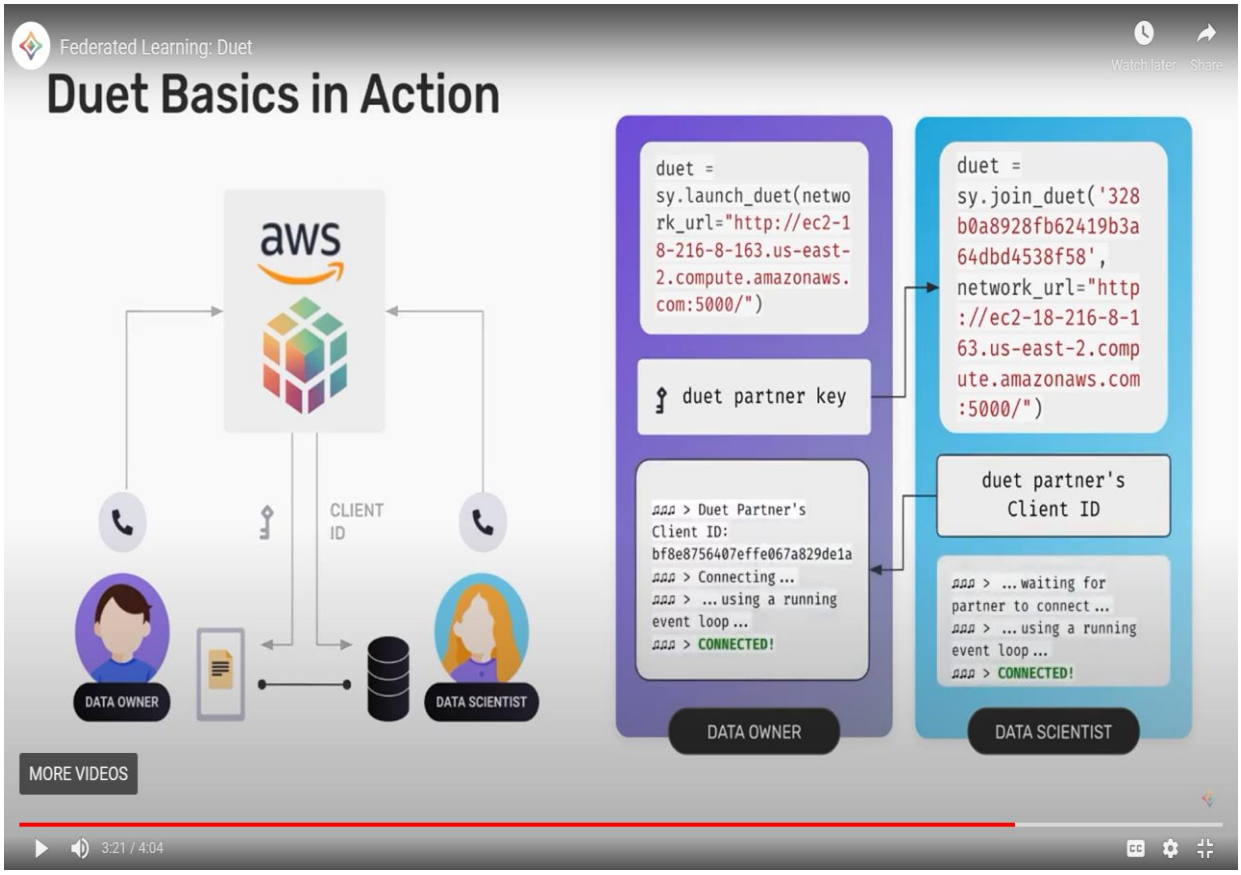


Exhibit 51: Federated ML Connection Layout



5.7.2 Modeling Steps

The steps used for the remote federated ML in this study are provided below. 2 Jupyter notebooks were developed; one for the data owner and a second one for the data scientist to simulate the federated ML.

The focus of PyTorch and PySft modeling effort was to identify the process to be used to train, build, and test the model on remote dataset and to evaluate its effectiveness in achieving results that are comparable to the other models. Accordingly, to reduce the time required to run the models, 5% of the final dataset was used in the modeling effort. Similar to the workflow for the other models, this fraction of the final dataset was split into train (80%) and test (20%) components.

The steps followed were as follows:

1) Data Owner/Data Scientist interacted via PySyft and PyGrid/Amazon Web Service (see Exhibit 51)
2) Data Owner sent data to Data Scientist upon request from Data Scientist
3) Data Scientist made requests via Pysft to Data Owner
4) Data Scientist created the neural network model architecture
5) Data Scientist sent the model to Owner
6) Training occurred on the Remote Server

7) Model Sent to Data Scientist Once Trained

8) Data Scientist Tested the Model using test set data – Sckit Learn Packages

5.7.3 Model Architecture

The neural network model architecture and model parameters were as follows:

1) 3 Hidden Layers: 100, 50, and 25 Neurons, RELU Activation

2) 1 Output Layer, 2 Neurons, Log_soft_max Activation

3) 300 Epochs

4) Optimizer: Adam

5) learning_rate = .01

6) nn.functional.nll_loss

5.7.4 Model Results

Results of the modeling are depicted on Exhibits 52 to 54. Model results indicated that the precision, accuracy, recall, and F_1 scores all exceeded 0.85, and the AUC score was 0.966. The model results indicate the viability of this application for the classification on the loan dataset. Further fine tuning and optimization and testing on the full final dataset should yield results comparable to the best performing models in this study.

Exhibit 52: Performance Model, PyTorch and PySft

	Predicted No	**Predicted Yes**
Actual No	1,262	99
Actual Yes	97	632

Parameter	**Value**
RMSE	0.306
Precision	0.865
Accuracy	0.906
Recall	0.867
F1_Score	0.867

Exhibit 53: Federated ML Training Errors

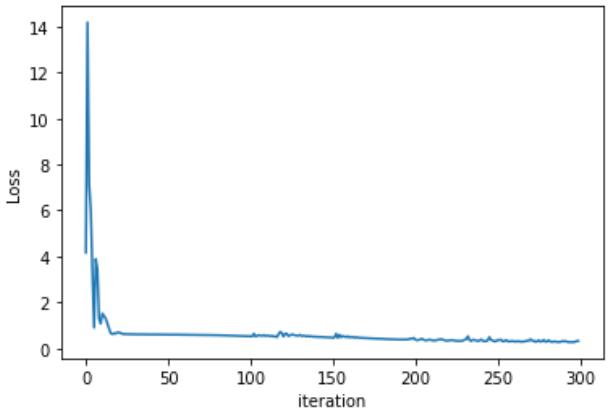
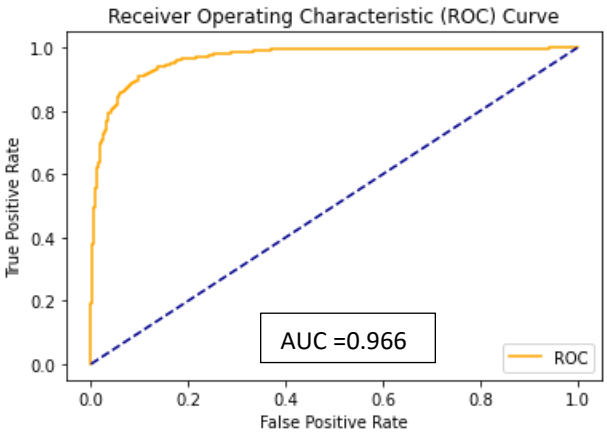


Exhibit 54: Federated ML ROC Curve



5.8 Summary of Model Evaluations

A comparison of the performance of the models presented in this study relative to the various performance metrics is presented in Table 55 below.

1) Accuracy, and F_1 scores were highest for the decision tree model.
2) Recall was the highest for Tensorflow/Keras neural network model.
3) RMSE was the lowest for the Random forest model.
4) Precision was the highest for the Random forest model.
5) Better tuning of the Random forest model, which has a high time complexity, should allow it to outperform the Decision Tree model.
6) AUC was the highest for Tensorflow/Keras neural network model.
7) Ensemble forest which boosted a weak decision tree classifier compared favorably with the stronger Decision Tree Classifier presented in table below.
8) Remote ML with PyTorch/PySft provided results that were comparable to other models.

Exhibit 55: Overall Models Performance Evaluation

Parameter	RMSE	Precision /Recall	Accuracy/ F_1 Score	AUC
Logistic Regression	0.209	0.938/0.936	0.956/0.937	0.951
Multinomial Bayes	0.399	0.789/0.743	0.789/0.765	0.818
Decision Tree	0.166	0.962/0.960	**0.973**/**0.961**	0.970
Ensemble Forests	0.231	0.934/0.913	0.947/0.923	0.939
Random Forests	**0.163**	**0.976**/0.943	0.972/0.960	0.966
Tensor Flow/Keras NN	0.249	0.907/**0.986**	0.912/0.945	**0.980**
PyTorch/PySft	0.306	0.865/0.867	0.906/0.867	0.966

6.0 Conclusions

All the machine learning models, except Naïve Bayes provided consistent results. Precision, accuracy, recall, F1_scores were all above 0.85, and above 0.9 for all models, except remote ML performed by PyTorch/PySft.

If PyTorch/PySft model has a better architecture and undergoes tuning it should result in results comparable to the other models. Remote ML performed by PyTorch/PySft, which was only performed on a small fraction of the dataset (5 pct of the total) and was not tuned for hyperparameters still showed results that were comparable to other models. Remote ML models, when performed by PyTorch/PySft, can be trained remotely on multiple distributed systems and results can be aggregated and tested on the central server.

7.0 References

Bandora dataset:	Loan Dataset file from https://www.bondora.com/en/public-reports
Heaton, J., 2022	Applications of Deep Neural Networks with Keras, Jeff Heaton, Spring 2022.0
Heaton, J, 2022a:	Refer to Section 2.2.2 Encoding Categorical Variables as dummies, Applications of Deep Neural Networks with Keras, Jeff Heaton, Fall 2022.0
James G, 2017:	Introduction to Statistical Learning in R
sklearn-a:	sklearn.linear_model.LogisticRegression — scikit-learn 1.1.1 documentation
Hastie T, 2017:	The Elements of Statistical Learning
sklearn-b:	sklearn.naive_bayes.MultinomialNB — scikit-learn 1.1.1 documentation
sklearn-c:	sklearn.tree.DecisionTreeClassifier — scikit-learn 1.1.1 documentation
Zhu, H. 2009:	Zhu, H. Zou , S. Rosset, T. Hastie, “Multi-class AdaBoost”, 2009.
sklearn-d:	sklearn.ensemble.AdaBoostClassifier — scikit-learn 1.1.1 documentation
sklearn-e:	sklearn.ensemble.RandomForestClassifier — scikit-learn 1.1.1 documentation

APPENDICES

Appendix A: List of Feature Names

Feature No	Feature Name
1	ReportAsOfEOD
2	LoanId
3	LoanNumber
4	ListedOnUTC
5	BiddingStartedOn
6	BidsPortfolioManager
7	BidsApi
8	BidsManual
9	PartyId
10	NewCreditCustomer
11	LoanApplicationStartedDate
12	LoanDate
13	ContractEndDate
14	FirstPaymentDate
15	MaturityDate_Original
16	MaturityDate_Last
17	ApplicationSignedHour
18	ApplicationSignedWeekday
19	VerificationType
20	LanguageCode
21	Age
22	DateOfBirth
23	Gender
24	Country
25	AppliedAmount
26	Amount
27	Interest
28	LoanDuration
29	MonthlyPayment
30	County
31	City
32	UseOfLoan
33	Education
34	MaritalStatus
35	NrOfDependants
36	EmploymentStatus
37	EmploymentDurationCurrentEmployer
38	EmploymentPosition
39	WorkExperience
40	OccupationArea
41	HomeOwnershipType
42	IncomeFromPrincipalEmployer
43	IncomeFromPension
44	IncomeFromFamilyAllowance
45	IncomeFromSocialWelfare
46	IncomeFromLeavePay

47 IncomeFromChildSupport
48 IncomeOther
49 IncomeTotal
50 ExistingLiabilities
51 LiabilitiesTotal
52 RefinanceLiabilities
53 DebtToIncome
54 FreeCash
55 MonthlyPaymentDay
56 ActiveScheduleFirstPaymentReached
57 PlannedPrincipalTillDate
58 PlannedInterestTillDate
59 LastPaymentOn
60 CurrentDebtDaysPrimary
61 DebtOccuredOn
62 CurrentDebtDaysSecondary
63 DebtOccuredOnForSecondary
64 ExpectedLoss
65 LossGivenDefault
66 ExpectedReturn
67 ProbabilityOfDefault
68 PrincipalOverdueBySchedule
69 PlannedPrincipalPostDefault
70 PlannedInterestPostDefault
71 EAD1
72 EAD2
73 PrincipalRecovery
74 InterestRecovery
75 RecoveryStage
76 StageActiveSince
77 ModelVersion
78 Rating
79 EL_V0
80 Rating_V0
81 EL_V1
82 Rating_V1
83 Rating_V2
84 Status
85 Restructured
86 ActiveLateCategory
87 WorseLateCategory
88 CreditScoreEsMicroL
89 CreditScoreEsEquifaxRisk
90 CreditScoreFiAsiakasTietoRiskGrade
91 CreditScoreEeMini
92 PrincipalPaymentsMade
93 InterestAndPenaltyPaymentsMade
94 PrincipalWriteOffs

95	InterestAndPenaltyWriteOffs
96	PrincipalBalance
97	InterestAndPenaltyBalance
98	NoOfPreviousLoansBeforeLoan
99	AmountOfPreviousLoansBeforeLoan
100	PreviousRepaymentsBeforeLoan
101	PreviousEarlyRepaymentsBefoleLoan
102	PreviousEarlyRepaymentsCountBeforeLoan
103	GracePeriodStart
104	GracePeriodEnd
105	NextPaymentDate
106	NextPaymentNr
107	NrOfScheduledPayments
108	ReScheduledOn
109	PrincipalDebtServicingCost
110	InterestAndPenaltyDebtServicingCost
111	ActiveLateLastPaymentCategory
112	Target Class: Defaulted

Appendix B: Python code as pdf