**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

July 15, 2022

**TABLE OF CONTENTS**

[List of Tables ii](#_Toc108731143)

[List of Exhibits iii](#_Toc108731144)

[List of Appendices v](#_Toc108731145)

[1.0 Introduction 1](#_Toc108731146)

[2.0 Objectives 1](#_Toc108731147)

[3.0 Exploratory Data Analysis 1](#_Toc108731148)

[3.1 Analysis Summary 1](#_Toc108731149)

[3.2 Analysis Findings 2](#_Toc108731150)

[4.0 Feature Evaluation/Extraction 9](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731151)

[4.1 Missing Value Analysis 9](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731152)

[4.2 Correlation Analysis 11](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731153)

[4.3 Principal Component Analysis 13](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731154)

[4.4 Exploratory Clustering Analysis 15](#_Toc108731155)

[4.4.1 Overview 15](#_Toc108731156)

[4.4.2 Results 15](#_Toc108731157)

[5.0 Machine Learning Modeling 16](#_Toc108731158)

[5.1 Logistic Regression 17](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731159)

[5.1.1 Model Overview and Results 17](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731160)

[5.1.2 Best Model Parameters 18](#_Toc108731161)

[5.2 Multinomial Bayes 19](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731162)

[5.2.1 Model Overview and Results 19](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731163)

[5.2.2 Best Model Parameters 20](#_Toc108731164)

[5.3 Decision Tree 21](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731165)

[5.3.1 Model Overview and Results 21](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731166)

[5.3.2 Best Model Parameters 22](#_Toc108731167)

[5.4 Ensemble Forest 23](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731168)

[5.4.1 Model Overview and Results 23](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731169)

[5.4.2 Best Model Parameters 24](#_Toc108731170)

[5.5 Random Forest 25](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731171)

[5.5.1 Model Overview and Results 25](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731172)

[5.5.2 Best Model Parameters 26](#_Toc108731173)

[5.6 Bayesian Optimization 27](#_Toc108731174)

[5.6.1 Overview 27](#_Toc108731175)

[5.6.2 Decision Tree 28](#_Toc108731176)

[5.6.3 Ensemble Forest 28](#_Toc108731177)

[5.6.4 Random Forest 29](#_Toc108731178)

[5.6.5 Model Results 29](#_Toc108731179)

[5.7 Deep Neural Network with Tensorflow/Keras 30](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731180)

[5.7.1 Model Overview and Results 30](file:///C:\Data%20Science%20and%20Analytics\DSA%205900\Final%20Practicum%20-%20Copy\Ramkishore_Rao_DSA5900_Final_Report.docx#_Toc108731181)

[5.7.2 Best Model Parameters 32](#_Toc108731182)

[5.8 Federated Machine Learning with PyTorch and PySft 33](#_Toc108731183)

[5.8.1 What is Federated Machine Learning and Why is it Relevant? 33](#_Toc108731184)

[5.8.2 Modeling Steps 34](#_Toc108731185)

[5.8.3 Model Architecture 35](#_Toc108731186)

[5.8.4 Model Results 35](#_Toc108731187)

[5.9 Summary of Model Evaluations 36](#_Toc108731188)

[5.9.1 Summaries 36](#_Toc108731189)

[5.9.2 Model Prediction Power, Limited Feature Set 36](#_Toc108731190)

[5.9.2.1 Perturbation Approach 37](#_Toc108731191)

[5.9.2.3 Results 37](#_Toc108731192)

[6.0 Conclusions 40](#_Toc108731193)

[7.0 References 41](#_Toc108731194)

[Appendix A: List of Feature Names 43](#_Toc108731195)

[Appendix B: Python code as pdf 47](#_Toc108731196)

# List of Tables

Table 1: Data Breakdown by Target Class

Table 2: Features with More than 10 Pct Missing Values

Table 3: Target Class Breakdown, Final Dataset

Table 4: Correlation Coefficients Between Variables

Table 5: Correlation Coefficients Between Variables and Target Variable

Table 6: Maximum Separation of Farthest Centroids, Input Features, Clustering Analysis

Table 7: Search Space, Decision Tree Bayesian Optimization

Table 8: Search Space, Ensemble Forest Bayesian Optimization

Table 9: Search Space, Random Forest Bayesian Optimization

# List of Exhibits

Exhibit 1: Box Plots, Select Continuous 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

Exhibit 8: Employment Status Counts Breakdown by Target Class

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

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

Exhibit 11: Amount of Previous Credit Breakdown by Target Class

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

Exhibit 13: Missing Values Count for Surviving Features

Exhibit 14: Explained Variance vs Principal Component No.

Exhibit 15: Target Class Separation from Three Principal Components

Exhibit 16: PCA Bi-Plot

Exhibit 17: Elbow Analysis, K Means Clustering

Exhibit 18: K Means Visualization

Exhibit 19: LR Model Hyperparameters

Exhibit 20: LR Grid Search CV Results

Exhibit 21: Performance Evaluation, Logistic Regression

Exhibit 22: ROC Curve, Logistic Regression, Best Model Following Tuning

Exhibit 23: Importance Feature Coefficients, Logistic Regression, Best Model Following Tuning

Exhibit 24: MNB Model Hyperparameters, Multinomial Bayes

Exhibit 25: MNB Grid Search CV Results

Exhibit 26: Performance Evaluation, Multinomial Bayes

Exhibit 27: ROC Curve, Multinomial Bayes, Best Model Following Tuning

Exhibit 28: Important Features Coefficients Difference Between Classes Naïve Bayes/Best Model Following Tuning

Exhibit 29: Decision Tree Model Hyperparameters

Exhibit 30: Decision Tree Grid Search CV Results

Exhibit 31: Performance Evaluation, Decision Tree

Exhibit 32: ROC Curve, Logistic, Decision Tree, Best Model Following Tuning

Exhibit 33: Features Importance Decision Tree/Best Model Following Tuning

Exhibit 34: Ensemble Forests Model Hyperparameters

Exhibit 35: Ensemble Forests Grid Search CV Results

Exhibit 36: Performance Evaluation, Ensemble Forests

Exhibit 37: ROC Curve, Ensemble Forests, Best Model Following Tuning

Exhibit 38: Features Importance Ensemble Forests /Best Model Following Tuning

Exhibit 39: Random Forest Model Hyperparameters

Exhibit 40: Random Forest Grid Search CV Results

Exhibit 41: Performance Evaluation, Random Forest

Exhibit 42: ROC Curve, Random Forest, Best Model Following Tuning

Exhibit 43: Features Importance Random Forests/Best Model Following Tuning

Exhibit 44: Bayesian Optimization, Algorithm

Exhibit 45: Bayesian Optimization Convergence, Decision Tree

Exhibit 46: Bayesian Optimization Convergence, Ensemble Forest

Exhibit 47: Bayesian Optimization, Convergence, Random Forest

Exhibit 48: Bayesian Optimization, Summary of Results

Exhibit 49: Performance Evaluation, NN, Tensor Flow/Keras, Default Parameters

Exhibit 50: Keras/Tensorflow Model Hyperparameters

Exhibit 51: Keras/Tensorflow Model Training Errors, Best Model Retraining

Exhibit 52: Keras/Tensorflow Model Training Accuracy, Best Model Retraining

Exhibit 53: Performance Evaluation, Tensorflow/Keras

Exhibit 54: Important Features Weights Neural Net/Best Model Following Tuning

Exhibit 55: ROC Curve: Tensor Flow/Keras/Default

Exhibit 56: ROC Curve, TensorFlow/Keras, Best Model Following Tuning

Exhibit 57: Federated ML Process Layout

Exhibit 58: Federated ML Connection Layout

Exhibit 59: Performance Evaluation: PyTorch and PySft

Exhibit 60: Federated ML Training Errors

Exhibit 61: Federated ML ROC Curve

Exhibit 62: Overall Models Performance Evaluation

Exhibit 63: 20 Feature Set Mix/Performance Evaluation

Exhibit 64 10 Feature Set Mix/Performance Evaluation

Exhibit 65: 6 Feature Set Mix/Performance Evaluation

Exhibit 66 4 Feature Set Mix/Performance Evaluation

Exhibit 67 2 Feature Set Mix/Performance Evaluation

Exhibit 68: Model Performance Metrics vs No. of Predictor Features

Exhibit 69: ROC Curve, 10 Feature Set Model

# List of Appendices

Appendix A: List of Feature Names

Appendix B: Python code as pdf

# 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

Themain 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**



## 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**

Chart, box and whisker chart

Description automatically generated

**Exhibit 2: Income Breakouts by Target Class**

A picture containing timeline

Description automatically generated

**Exhibit 3: Interest Servicing Breakouts by Target Class**

Timeline

Description automatically generated

**Exhibit 4: Liability Breakouts by Target Class**

A picture containing bar chart

Description automatically generated

**Exhibit 5: Credit Rating by Median Probability of Default**

Chart, bar chart

Description automatically generated

**Exhibit 6: Credit Parameters by Target Class - I**

A picture containing timeline

Description automatically generated

**Exhibit 7: Credit Parameters by Target Class - II**

Chart, bar chart

Description automatically generated

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**

Table

Description automatically generated

Note:

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

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

Table

Description automatically generated

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

Table

Description automatically generated

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

Notes:

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

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

Table

Description automatically generated

**Exhibit 11: Amount of Previous Credit Breakdown by Target Class**

Table

Description automatically generated

# 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; and
5. Exploratory clustering analysis.

## 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**

Table

Description automatically generated

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**

Chart, bar chart

Description automatically generated

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**

Table

Description automatically generatedTable

Description automatically generated

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 Continued: Correlation Coefficients Between Variables and Target Variable**

Table

Description automatically generated

**Table 5: Correlation Coefficients Between Variables and Target Variable**

Table

Description automatically generated

## 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.**

Chart, line chart

Description automatically generated

**Exhibit 15: Target Class Separation from Three Principal Components**

Chart, scatter chart

Description automatically generated

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**

Chart, scatter chart

Description automatically generated with medium confidence

## 4.4 Exploratory Clustering Analysis

### 4.4.1 Overview

Exploratory unsupervised learning in the form of clustering analysis was performed using the K-means clustering algorithm using the sklearn package.

### 4.4.2 Results

Elbow analysis that shows a plot of cumulative within cluster sum of squares (WCSS) vs No. of Clusters is depicted on Exhibit 17 for fully scaled data (all values between 0 and 1). Cumulative WCSS (sum of squared distance (SSE) between the data points and their respective assigned clusters centroid) is an indicator of the spread within each cluster. A clear elbow cannot be discerned from the exhibit, and the value of the cumulative WCSS of 500,000, which is indicative of K means not adequately separating data into the first 12 clusters.

For purpose of this analysis, 6 cluster K-means was further investigated. Values of maximum spread between cluster centroids (inter cluster spread) ranged from 0.096 to 0.424 (scale of 0 to 1) for the 10 top ranked continuous variables (Table 6). This spread appears low and maybe an indicator of poor performance of k-means on the data. Lastly, a scatter plot between two continuous variables further demonstrates of lack of separation into clusters by this method (see Exhibit 18).

Because of this apparent lack of promise, unsupervised learning was not further investigated.

Chart, line chart

Description automatically generated

**Exhibit 17: Elbow Analysis, K-means Clustering**

6 CLUSTER

MODEL SELECTED

Cumulative WCSS

Chart, scatter chart

Description automatically generated

**Exhibit 18: K-means Visualization**

**Table 6:**

**Maximum Separation of Farthest Centroids, Input Features, Clustering Analysis**

|  |  |  |
| --- | --- | --- |
| **Rank** | **Spread** | **Feature Names** |
| 1 | 0.424 | DebtToIncome |
| 2 | 0.398 | LossGivenDefault |
| 3 | 0.223 | PrincipalBalance |
| 4 | 0.172 | ProbabilityOfDefault |
| 5 | 0.163 | PrincipalPaymentsMade |
| 6 | 0.152 | ExpectedLoss |
| 7 | 0.135 | PlannedInterestTillDate |
| 8 | 0.109 | PrincipalOverdueBySchedule |
| 9 | 0.098 | AppliedAmount |
| 10 | 0.096 | Interest |

# 5.0 Machine Learning Modeling

Supervised learning via 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. Training was conducted using two optimization approaches for comparative analysis: grid search optimization and Bayesian optimization.

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:

(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, max\_iter=200*,  *l1\_ratio*).

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 19. Results are provided on Exhibits 20-23.

**Exhibit 20: LR Model Grid Search CV Results**

Chart, line chart

Description automatically generated

**Exhibit 19: 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 22: 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 21: ROC Curve: Logistic Regression/Best Model Following Tuning**

Chart, line chart

Description automatically generated

**AUC = 0.951**

**Exhibit 23: Important Features Coefficients: Logistic Regression/Best Model Following Tuning**

Graphical user interface

Description automatically generated with medium confidence

### 5.1.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.952 (Exhibit 20) was obtained with the best values of hyperparameters noted on Exhibit 19. 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 22). The area under the curve of the receiver operating characteristic curve was 0.951 (Exhibit 21), 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 23). Positive coefficients drive the target class to 1 and negative coefficients drive the target Class to 0. Exhibit 23 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 and the density function of X (fk(x)) that comes from an observation comes from that kth class:

(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*,  *fit\_prior=True*)

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 24. Results are provided on Exhibits 25-28.

**Exhibit 25: 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** | 3,762 | | 10,853 | |
| **Parameter** | | **Value Following Tuning** | |
| RMSE | | 0.399 | |
| Precision | | 0.789 | |
| Accuracy | | 0.841 | |
| Recall | | 0.743 | |
| F1\_Score | | 0.765 | |

**Exhibit 24: MNB Model Hyperparameters**

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

**Exhibit 26: MNB Grid Search CV Results**

****

**Exhibit 27: ROC Curve: Multinomial Bayes/Best Model Following Tuning**



**Exhibit 28: Important Features Coefficients Difference Between Classes Naïve Bayes/Best Model Following Tuning**

Chart, bar chart

Description automatically generated

### 5.2.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.838 (Exhibit 26) was obtained with the best values of hyperparameters noted on Exhibit 24. 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 25). The area under the curve of the receiver operating characteristic curve was 0.818 (Exhibit 27), 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 28 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.

**Exhibit 31: 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 | |

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 29. Results are provided on Exhibits 30-33.

**Exhibit 29: Decision Tree Model Hyperparameters**

|  |  |  |
| --- | --- | --- |
| **Hyper-**  **-parameter** | **Range** | **Best Value** |
| Criterion | Gini and Entropy | Entropy |
| Max\_Depth | 5,10,20 | 20 |

**Exhibit 32: ROC Curve: Decision Tree/Best Model Following Tuning**



**Exhibit 30: Decision Tree Grid Search CV Results**



**Exhibit 33: Features Importance Decision Tree/Best Model Following Tuning**

Chart

Description automatically generated

### 5.3.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.971 (Exhibit 30) was obtained with the best values of hyperparameters noted on Exhibit 29. 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 31). The area under the curve of the receiver operating characteristic curve was 0.970 (Exhibit 32), 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 33)*.*  Exhibit 33 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 Forest

### 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.

**Exhibit 36: Performance Evaluation: Ensemble Forest**

Confusion Matrix, Test Dataset Following Tuning:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Predicted**  **No** | | **Predicted**  **Yes** | |
| **Actual**  **No** | 26,238 | | 949 | |
| **Actual**  **Yes** | 1,276 | | 13,339 | |
| **Parameter** | | **Value Following Tuning** | |
| RMSE | | 0.231 | |
| Precision | | 0.934 | |
| Accuracy | | 0.947 | |
| Recall | | 0.913 | |
| F1\_Score | | 0.923 | |

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*)

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 34. Results are provided on Exhibits 35-38.

**Exhibit 34: Ensemble Forest 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 35: Ensemble Forest Grid Search CV Results**

**Exhibit 37: ROC Curve: Ensemble Forest/Best Model Following Tuning**



Chart, line chart

Description automatically generated

**Exhibit 38: Features Importance Ensemble Forest/Best Model Following Tuning**

**Chart, bar chart

Description automatically generated**

### 5.4.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.947 (Exhibit 35) was obtained with the best values of hyperparameters noted on Exhibit 34. 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 36). The area under the curve of the receiver operating characteristic curve was 0.939 (Exhibit 37), 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 38)*.*  Exhibit 38 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](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification) that operates by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) 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 classification on the final dataset (sklearn-e). 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, criterion, max\_features)*

The noted hyperparameters were tuned per Grid Search CV with 5-fold cross validation per Exhibit 39. Results are provided on Exhibits 40-43.

**Exhibit 41: 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 | | 0.960 | |

**Exhibit 39: Random Forest Model Hyperparameters**

|  |  |  |
| --- | --- | --- |
| **Hyper-**  **-parameter** | **Range** | **Best Value** |
| N\_estimators | 50,100,200 | 200 |
| Criterion | Gini, entropy | entropy |
| Max\_features | sqrt, log2 | sqrt |

**Exhibit 40: Random Forest Grid Search CV Results**



**Exhibit 42: ROC Curve: Random Forest/Best Model Following Tuning**



**Exhibit 43: Important Features Importance Random Forest/Best Model Following Tuning**

**Chart

Description automatically generated**

### 5.5.2 Best Model Parameters

Based on the results of the tuning, the highest mean CV score of 0.971 (Exhibit 40) was obtained with the best values of hyperparameters noted on Exhibit 39. 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 41). The area under the curve of the receiver operating characteristic curve was 0.966 (Exhibit 42), 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 43)*.* Exhibit 43 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 Bayesian Optimization

### 5.6.1 Overview

1. For n = 1,2,…. do
2. Select new xn+1 by optimizing objective function α (in this case 1 – Accuracy)

xn+1 = argmin α (x, Dn) where Dn is surrogate function, (in this case a Gaussian Process)

1. Query the objective function to obtain yn+1
2. augment data Dn+1 = {Dn, (xn+1 ,yn+1)}
3. update statistical model
4. end for

**Exhibit 44: Bayesian Optimization, Algorithm**

Bayesian optimization is an elegant solution to optimization of the objective functions (such as accuracy, RMSE, etc.) required for hyperparameter tuning of machine learning models (Jones, 2001). It is being used in academia to solve numerous areas (Shahriari, 2016).

For our study, this approach of optimization was evaluated on three of the models, decision tree, ensemble forest, and random forest for comparison with the grid search optimization outlined in sections 5.3, 5.4, and 5.5.

Bayesian optimization algorithm utilizes gaussian process and maintains a posterior distribution for this function as observations are made (Snoek, 2012). A range of hyperparameter values are optimized in this process and each subsequent value is picked to allow for optimization of the current best result of the objective function (see Exhibit 44 for the algorithm).

The three main ingredients of the Bayesian optimization process used in our study are the following:

1. **Search space of Hyperparameters**: A range and type of values of the hyperparameters for the three models that were optimized using this algorithm are presented in sections 5.6.2, 5.6.3, and 5.6.4. The prior distributions for the hyperparameters assigned in our study are uniform and log-uniform for integer and real parameter values and a list of values for categorical hyperparameter values.
2. **Objective function**: The objective function is the main evaluator used in the optimization. For our study, the set of hyperparameters are optimized to obtain the lowest mean 5-fold cross validation scores of 1-accuracy score for each iteration.
3. **Surrogate function** and **Acquisition function**: sklearn’s gaussian process was utilized as the surrogate function. The **surrogate function** can be interpreted as an approximation of the objective function. It is used to propose parameter sets to the objective function that likely yield an improvement in terms of accuracy score.

The hyperparameters that are put forward for evaluation by the objective function are selected by applying a criterion to the surrogate function. This criterion is defined by a acquisition function.

Sklearn’s gp\_minimize function is used with the default gp\_hedge parameter for the acquisition function. Gp\_hedge probabilistically chooses one of the above three acquisition functions at every iteration. These are:

* "LCB" for lower confidence bound;
* "EI" for negative expected improvement; and
* "PI" for negative probability of improvement.

Reference to a github repository for explanation of the acquisition functions is provided as reference (skopt-a). Default values for the rest of the skopt’s gp\_minimize were retained for this study.

### 5.6.2 Decision Tree

Bayesian optimization algorithm using the process applied above was applied to the Decision Tree classifier. Default values were retained for this model, except for the hyperparameters that were tuned.

The search spaces for the hyperparameters that were tuned are presented in Table 7 below.

Three hyperparameters were tuned in this study. Criterion had a categorical search space while the other two had integer search spaces. Results (mean\_cv(1-accuracy score)) are provided on Exhibit 45.

**Table 7: Search Space**

**Decision Tree**



Chart, line chart

Description automatically generated

**Exhibit 45: Bayesian Optimization Convergence, Decision Tree**

**Criterion: Entropy**

**Min\_samples\_split: 5**

**Max\_Depth: 20**

### 5.6.3 Ensemble Forest

Bayesian optimization algorithm using the process applied above was applied to the adaboost’s ensemble forest classifier. Similar to the process used in Section 5.4, a weak decision tree classifier was boosted by the adaboost model.

The search spaces for the hyperparameters that were tuned are presented in Table 8 below.

Two hyperparameters were tuned in this study. N\_estimators had an integer search space while the l\_rate had real value search space. Results (mean\_cv(1-accuracy score)) are provided on Exhibit 46.

**Table 8: Search Space**

**Ensemble Forest**

****

Chart, line chart

Description automatically generated

**Exhibit 46: Bayesian Optimization Convergence, Ensemble Forest**

**N\_estimators 196**

**l\_rate: 1.625**

### 5.6.4 Random Forest

Bayesian optimization algorithm using the process applied above was applied to the random forest classifier. Aside from the hyperparameters outlined below, the remainder of the parameters used in the model were identical to those used in the modeling conducted in Section 5.5.

The search spaces for the hyperparameters that were tuned are presented in Table 9 below.

Three hyperparameters were tuned in this study. Criterion had a categorical search space while the other two had integer search spaces. Results Results (mean\_cv(1-accuracy score)) are provided on Exhibit 47.

**Table 9: Search Space**

**Random Forest**



Chart, line chart

Description automatically generated

**Exhibit 47: Bayesian Optimization Convergence, Random Forest**

**Criterion: Entropy**

**Max\_Features: sqrt**

**N\_estimators: 300**

### 5.6.5 Model Results

Each of the tuned classifier models were fitted with the selected hyperparameters on the training portion of the final dataset. Following this fitting, the fitted models were evaluated on the test dataset and the results are provided on Exhibit 48.

For the decision tree and the random forest classifier, the results were similar to those obtained from the grid search optimization. Selected hyperparameters were identical for decision tree and for random forest, the only difference was in the no\_of trees of 300 relative to the 200 in the grid search optimization.

Adaboost showed an improvement for all of the metrics relative to grid search optimization.

**Exhibit 48:**

**Bayesian Optimization, Summary of Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Decision Tree**  **BO** | **AdaBoost**  **BO** | **Random Forest**  **BO** |
| RMSE | 0.166 | 0.199 | 0.164 |
| Precision | 0.962 | 0.950 | 0.976 |
| Accuracy | 0.973 | 0.960 | 0.972 |
| Recall | 0.960 | 0.936 | 0.944 |
| F\_1 Score | 0.961 | 0.943 | 0.960 |
| AUC | 0.970 | 0.954 | 0.966 |

Note: Final results identical for Decision Tree and also for random forest within tolerance bounds. Adaboost scores showed improvement.

## 5.7 Deep Neural Network with Tensorflow/Keras

### 5.7.1 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 49.

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 50 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 50. Results are provided on Exhibits 51-54.

Exhibits 55 and 56, show AUC for the receiver operating characteristic curves, for the default and the best “tuned” model, respectively.

**Exhibit 49: 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 51: Keras/Tensorflow Training Errors, Best Tuned Model Retraining**

**Shape

Description automatically generated with low confidence**

**Exhibit 50: 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 53: 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 52: Keras/Tensorflow Training Accuracy, Best Model Retraining**

**A picture containing shape

Description automatically generated**

**Exhibit 54: Important Features Weights Neural Net/Best Model Following Tuning**

**Letter

Description automatically generated with medium confidence**

### 5.7.2 Best Model Parameters

Based on the results of the tuning, best hyperparameters were selected (see Exhibit 50). 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 53). The area under the curve of the receiver operating characteristic curve was 0.980 (Exhibit 55), 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 56). 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 54. The five features with the highest weights were *PrincipalOverduebySchedule*, *InterestandPenaltyPaymentsMade, StatusLate*, PrincipalPaymentsMade, and *PlannedInterestTillDate* (see Exhibit 54)

**Exhibit 56: ROC Curve: Tensor Flow/Keras/Best Model Following Tuning**

A picture containing square

Description automatically generated

**Exhibit 55: ROC Curve:**

**TensorFlow/Keras Default**

A picture containing square

Description automatically generated

AUC = 0.980

AUC =0.980

## 5.8 Federated Machine Learning with PyTorch and PySft

### 5.8.1 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](https://analyticsindiamag.com/federated-learning/) 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](https://ai.googleblog.com/2017/04/federated-learning-collaborative.html). 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 57 and 58, respectively.

**Exhibit 57: Federated ML Process Layout**

Graphical user interface, application, Teams

Description automatically generated

**Exhibit 58: Federated ML Connection Layout**

A picture containing graphical user interface

Description automatically generated

### 5.8.2 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 57)
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

### 5.8.3 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

### 5.8.4 Model Results

Results of the modeling are depicted on Exhibits 59 to 61. 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 60: Federated ML Training Errors**

Shape

Description automatically generated

**Exhibit 59: 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 61: Federated ML ROC Curve**

A picture containing polygon

Description automatically generated

AUC =0.966

## 5.9 Summary of Model Evaluations

### 5.9.1 Summaries

A comparison of the performance of the models presented in this study relative to the various performance metrics is presented in Exhibit 62 below. For the decision tree and random forest, scores from the grid search optimization are presented, but for the ensemble forest, scores from the Bayesian optimization are presented.

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 62: 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 Forest | 0.199 | 0.950/0.936 | 0.960/0.943 | 0.950 |
| Random Forest | **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 |

### 5.9.2 Model Prediction Power, Limited Feature Set

In order to evaluate whether models that contain a smaller subset of input features can retain the prediction power of the “full” feature set models evaluated in this study, a focused evaluation using the random forest classifier model tuned with the hyperparameters selected from the Bayesian optimization process on a smaller subset of input features was conducted.

The perturbation approach utilized and the results from this evaluation are presented in sections 5.9.2.1 and 5.9.2.2, respectively.

### 5.9.2.1 Perturbation Approach

As part of this approach, the feature importance scores (see Exhibit 43) for all 72 input features from the random forest model (i.e., the baseline model) were sorted in descending order. Five separate modeling runs were conducted using the following subset of features:

1. Top 20 features from the baseline model (see Exhibit 63);
2. Top 10 features from the baseline model (see Exhibit 64);
3. Top 6 features from the baseline model (see Exhibit 65);
4. Top 4 features from the baseline model (see Exhibit 66); and
5. Top 2 features from the baseline model (see Exhibit 67).

Note that for several of the baseline models (i.e., logistic regression, decision tree, ensemble forest, and random forest classifiers), aside from some marginal differences, the top 20 features were similar, with xx % of the features being identical across the classifiers.

The random forest model trained with 5-fold cross validation mean accuracy scores with Bayesian optimization were fitted to these subsets of features in the train portion of the dataset and the fitted model was tested on the test portion of the dataset.

### 5.9.2.3 Results

Model evaluations on the test dataset for the various performance metrics and the feature importance scores for the smaller subset models are presented on Exhibits 63-67. The performance metric scores were compared with the baseline model that contained all 72 predictor features. Where model performance was better than the baseline model, the result is highlighted in blue and where it is poorer than the baseline model, it is highlighted in red.



**RMSE: 0.144**

**Accuracy: 0.979**

**Precision: 0.981**

**Recall: 0.960**

**F\_1 Score: 0.970**

**AUC: 0.975**

Feature Importance

**Exhibit 63: 20 Feature Set Mix/Performance Evaluation**

For the 20 and the 10 input feature set models, model performance on the test dataset exhibited some improvement over the baseline model (see Exhibits 63 and 64). For the 6, 4, and 2 input feature set models, model performance declined relative to the baseline model (see Exhibits 65-67).

Chart, bar chart

Description automatically generated

**RMSE: 0.146**

**Accuracy: 0.979**

**Precision: 0.982**

**Recall: 0.956**

**F\_1 Score: 0.969**

**AUC: 0.973**

Feature Importance

**Exhibit 64: 10 Feature Set Mix/Performance Evaluation**

Chart, bar chart

Description automatically generated

**RMSE: 0.176**

**Accuracy: 0.969**

**Precision: 0.969**

**Recall: 0.941**

**F\_1 Score: 0.955**

**AUC: 0.962**

Feature Importance

**Exhibit 65: 6 Feature Set Mix/Performance Evaluation**

Chart

Description automatically generated

Feature Importance

**RMSE: 0.252**

**Accuracy: 0.937**

**Precision: 0.936**

**Recall: 0.879**

**F\_1 Score: 0.906**

**AUC: 0.923**

**Exhibit 66: 6 Feature Set Mix/Performance Evaluation**



Feature Importance

**RMSE: 0.296**

**Accuracy: 0.912**

**Precision: 0.921**

**Recall: 0.819**

**F\_1 Score: 0.867**

**AUC: 0.891**

**Exhibit 67: 2 Feature Set Mix/Performance Evaluation**

**Summary of Evaluations**

Exhibit 68 shows model performance vs no. of predictor features. Models with 10 or more features perform better than the baseline model and those below 10 features perform poorer than the baseline model.

Furthermore, the percentage of common features between the various classifiers included in this study decline as the total number of features drop below 20, with less than xx% common for 10 input features.

Higher Performing Models Range

R

**Exhibit 68: Model Performance Metrics vs No. of Predictor** **Features**

Chart, line chart

Description automatically generated

Care should be taken to ensure a balanced tradeoff between model simplicity and better interpretability (smaller number of input features) and robustness in model predictability and consistency in feature selection/importance (larger number of input features) while selecting the optimum subset of features to be utilized for classification.

A ROC curve for the 10 feature set model that shows marginal improvement over the baseline model is depicted on Exhibit 69.

AUC =0.973

**Exhibit 69: ROC Curve, 10 Feature Set Model**

# 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](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

Hastie T, 2017: The Elements of Statistical Learning

sklearn-b: [sklearn.naive\_bayes.MultinomialNB — scikit-learn 1.1.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html)

sklearn-c: [sklearn.tree.DecisionTreeClassifier — scikit-learn 1.1.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)

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](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html)

sklearn-e: [sklearn.ensemble.RandomForestClassifier — scikit-learn 1.1.1 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

https://scikit-optimize.github.io/stable/auto\_examples/bayesian-optimization.html

**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