MA 5790

PREDICTIVE MODELING FINAL PROJECT

Michigan Tech

PROJECT TOPIC

LOAN DEFAULT PREDICTION

Submitted by:

HARANADH REDDY RAVI

Submitted to:

PROFESSOR QIUYING SHA

Date of Submission: December 15, 2023

Table of Contents:

Abstract:

This research addresses the challenge financial institutions face in identifying high-risk loan defaulters, a scenario that poses significant financial losses and challenges to responsible lending practices. We propose an advanced predictive model integrating diverse data sources and employing sophisticated machine learning algorithms. Emphasizing feature engineering and interpretability, the model accurately identifies nuanced patterns indicative of potential default risks. Validation using historical loan data demonstrates its efficacy in diverse financial scenarios, offering financial institutions a proactive tool to manage and mitigate loan default risks. By contributing to financial stability, the proposed model enhances responsible lending practices, providing institutions with a means to navigate the lending landscape securely and foster resilience in the financial environment.

Background Summary:

Over the last decade, financial institutions have confronted a persistent challenge in effectively identifying and managing the risks associated with loan defaults. As of 2022, the repercussions of loan defaults extend beyond mere financial losses, posing a substantial threat to the ethical tenets of responsible lending practices. Traditional risk assessment methods have proven inadequate in navigating the intricate dynamics of borrower behavior, necessitating a paradigm shift towards adopting advanced predictive models.

In the backdrop of financial landscapes evolving up until 2021, the demand for predictive models has intensified, prompting a reevaluation of risk management strategies. The intricacies of financial risk, especially from 2015 to 2021, have underscored the limitations of conventional approaches. This underscores the urgency for innovative methodologies, embracing diverse data sources and harnessing the power of sophisticated algorithms.

Recent years, particularly the last five, have greatly emphasized feature engineering and interpretability in predictive modeling. These elements, recognized as crucial components of model development, not only enhance the accuracy of predictions but also ensure a transparent understanding of the model's decision-making process. The validation of these models, leveraging historical loan data spanning the past five years, provides empirical evidence of their efficacy across diverse financial scenarios.

The proposed research aims to tackle the challenges observed in the financial ecosystem between 2015 and 2021. By making use of predictive modeling advancements from 2018 to 2022, proactive risk management tools can be developed to help financial institutions make informed decisions. This will further strengthen their commitment to responsible lending practices in a constantly changing financial landscape.

In this paper, the primary goal is to build a predictive model that can accurately identify individuals likely to default on their loans. This is important for financial institutions to minimize financial losses and ensure responsible lending practices. The study aims to assess and quantify the risk associated with each loan applicant or borrower. By doing so, it helps financial institutions make informed decisions about whether to approve or deny loan applications and what terms to offer.

Variable Introduction and Definitions:

The dataset consists of loan applications from individuals aged 18 to 69, showcasing diversity in financial profiles. The mean age is approximately 43.65 years, with incomes ranging from $15,008 to $149,975 (mean: $82,825). Loan amounts vary from $5,020 to $249,863 (mean: $128,564), and credit scores range from 300 to 849 (average: 576). Employment histories span 0 to 119 months, averaging 58.96 months.

Applicants exhibit varied backgrounds in education, employment, marital status, and loan purposes. Education levels include "Bachelor's," "Master's," and "High School," while employment types range from "Full-time" to "Unemployed." Marital statuses encompass "Married," "Divorced," and "Single," and loan purposes vary from "Auto" to "Business" and "Other."

Approximately 11.94% of loans in the dataset have defaulted, emphasizing the need for predictive modeling to enhance risk assessment and ensure responsible lending practices in the financial landscape.

S.no	Variable	Data Type	Description
1.	Age	Integer	Age of loan applicants
2.	Income	Integer	Income of loan applicants
3.	LoanAmount	Integer	Amount of the loan
4.	CreditScore	Integer	Credit score of loan applicants
5.	MonthsEmployed	Integer	Number of months employed
6.	NumCreditLines	Integer	Number of credit lines
7.	InterestRate	Integer	Interest rate on the loan
8.	LoanTerm	Integer	Term of the loan (in months)
9.	DTIRatio	Integer	Debt-to-income ratio
10.	Education	Factor/Character	Education level of applicants
11.	EmploymentType	Factor/Character	Type of employment
12.	MaritalStatus	Factor/Character	Marital status of applicants
13.	HasMortagage	Factor/Character	Yes/No - Has a mortgage
14.	HasDependents	Factor/Character	Yes/No - Has dependents
15.	LoanPurpose	Factor/Character	Purpose of the loan
16.	HasCoSigner	Factor/Character	Yes/No - Has a co-signer
17.	Default	Integer	Loan default status (0 or 1)

Preprocessing: -

In the initial stages of our loan default prediction project, we performed key preprocessing steps to ensure the quality and integrity of our dataset. Firstly, a thorough check for missing values was conducted, revealing that **the dataset contains no null entries**. This absence of missing values indicates that our dataset is complete and reliable for subsequent analyses. Secondly, a check for duplicate rows was carried out, and the results showed **no duplicate instances** in the dataset. This absence of duplication ensures data integrity and prevents redundancy in our analyses. These preprocessing steps lay the groundwork for a robust and accurate predictive modeling process, providing a solid foundation for our loan default prediction project.

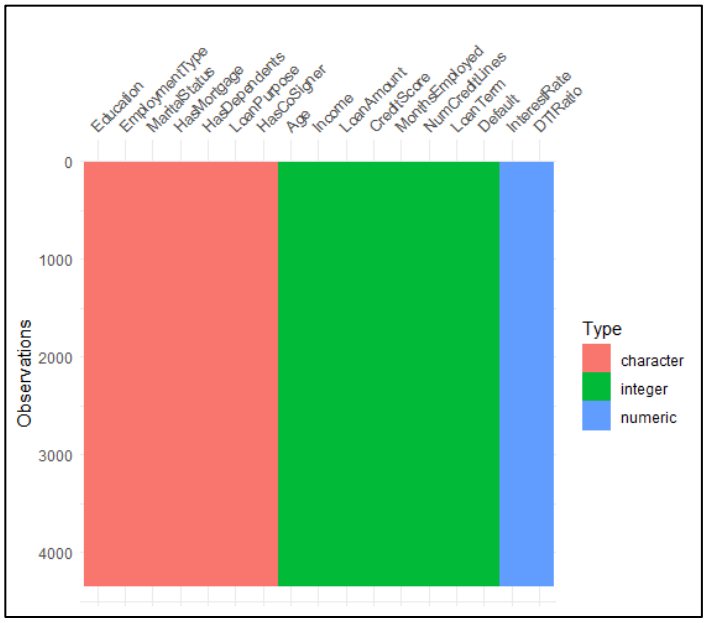


Fig: Given the solid figure shown, there are no missing values in this dataset. If there were missing values, there would be sections of the figure that would be black.

Data Transformations and Exploration:

We initiated the data transformation process by separating numerical and categorical columns. The numerical columns, comprising **10 variables**, were extracted using the select if function. After excluding the target variable "Default," the numerical columns were retained for further analysis. The categorical columns, **encompassing 7 variables**, were similarly isolated using the select if function for character-type columns.

Subsequently, we created dummy variables for the categorical columns to facilitate modeling. The resulting dataset, named dummy_data, **includes 16 dummy variables** representing different categories within the original categorical columns.

Near-Zero Variance Check:

To ensure the robustness of our dataset, we conducted a near-zero variance check on the dummy variables. The results revealed **no near-zero variance variables**, indicating that our dummy variables carry meaningful information.

Integration with Original Dataset:

The dummy variables were then appended to the original numerical dataset, creating a comprehensive **dataset named loan_data with 25 variables**.

Exploratory Data Analysis (EDA):

To gain insights into the distribution of categorical & and numerical variables, histograms, and boxplots were generated for each numerical column. These visualizations provide a preliminary understanding of the data's central tendencies and variations.

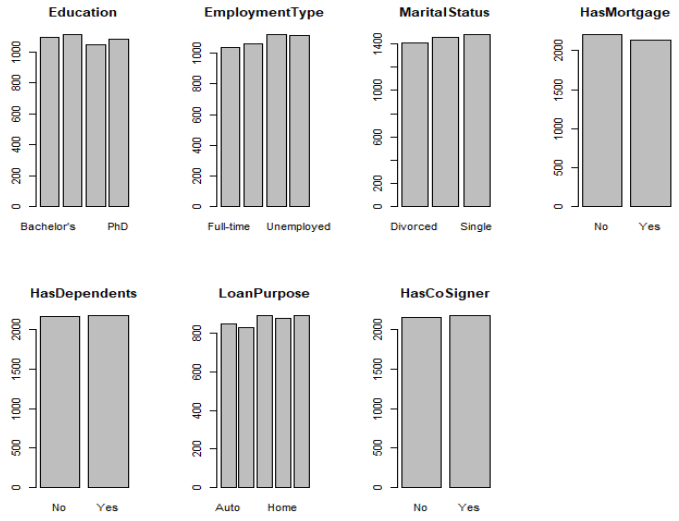
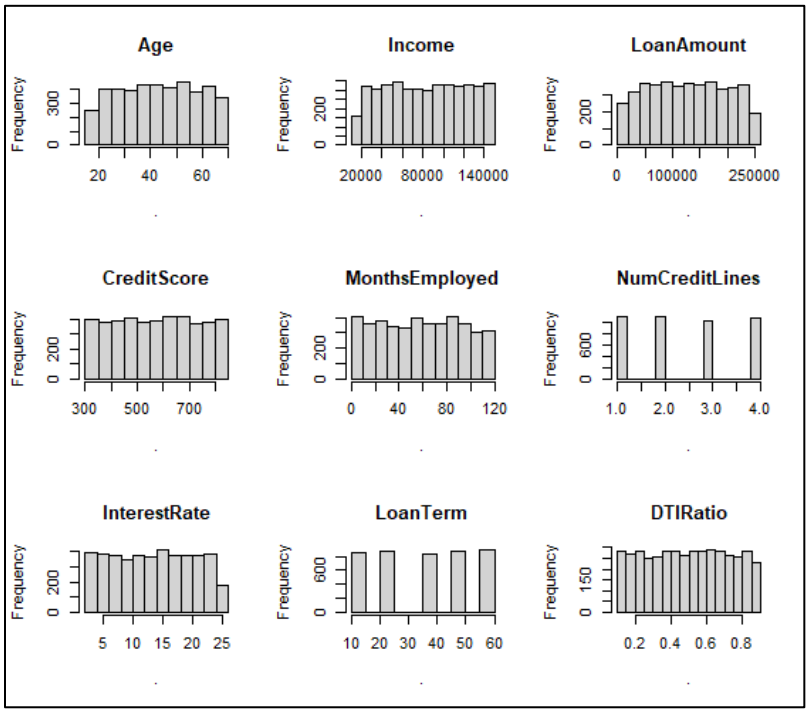


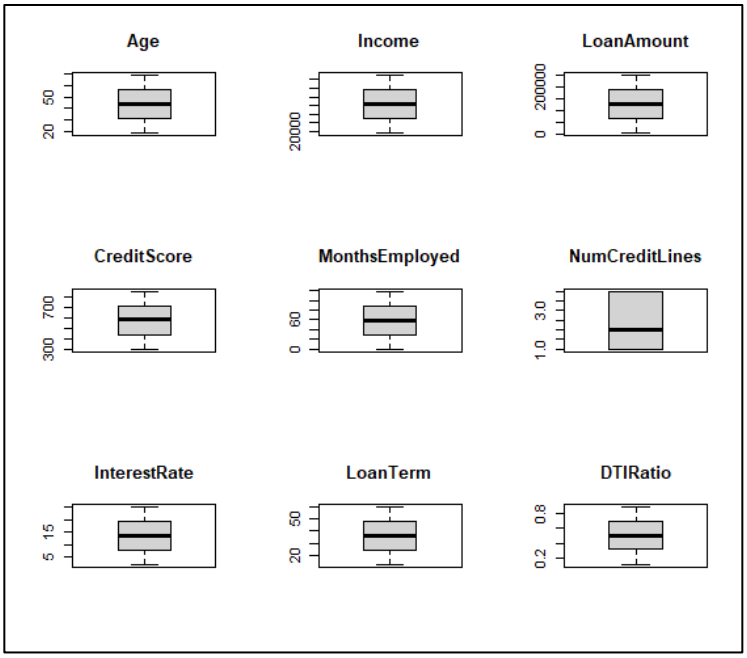
Fig Above are the histograms for each categorical predictor and the response variable, Default. These plots show that the frequencies are balanced for most predictors.

For numerical predictors

Histogram Analysis:



Numerical variables were subjected to histogram analysis to understand the distribution of values within each variable. The histograms revealed a uniform distribution for these variables, suggesting an even spread of values across the range. This uniformity can simplify certain analytical processes and contribute to the robustness of our predictive models.



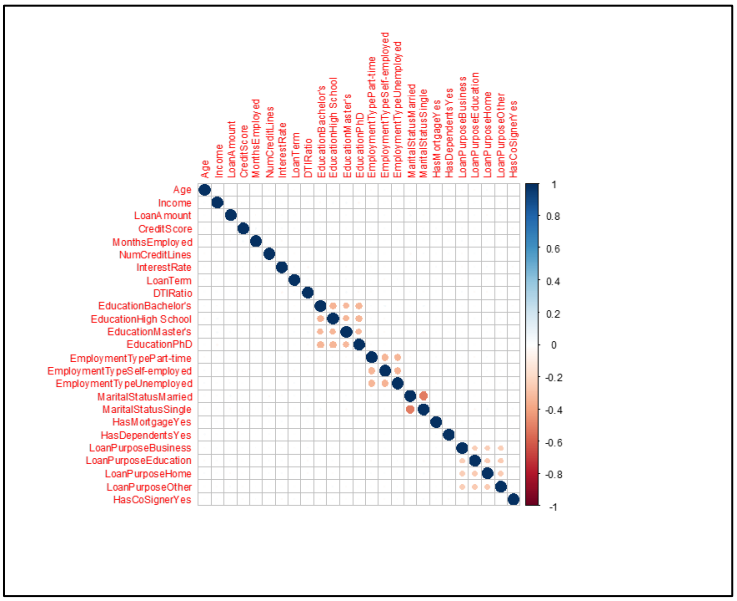
Outlier Detection:

Box plots were utilized to identify potential outliers in numerical variables. Fortunately, the visual inspection of box plots indicated the absence of outliers. The lack of extreme values is advantageous for maintaining the integrity of statistical analyses and ensuring the reliability of our predictive modeling.

This comprehensive data exploration and transformation set the stage for subsequent steps in our loan default prediction project, ensuring that our dataset is prepared for effective modeling and analysis.

Correlation Analysis:

A correlation matrix was calculated for the entire dataset, and a correlation plot was generated using the corrplot library. Highly correlated variables (correlation > 0.80) were identified, revealing instances where variables were duplicative. Notably, variables with identical names, such as "Age" and "Income," demonstrated perfect correlation, indicating the need for further investigation and potential feature reduction.



These perfect correlations are a result of the dummy variable creation process, where each category is represented by its binary variable. It is a common and expected outcome when dealing with categorical features in this manner. The perfect correlation within each variable pair validates the success of the dummy variable creation process and sets the stage for further analysis and modeling.

After Pre-Processing

After preprocessing our dataset is left with **25 predictor columns, 16 of which are categorical and 9 are continuous and and including 1 response variable 26 predictors.** The names are listed below.

```
 [1] "Age"                       "Income"                    "LoanAmount"
 [4] "CreditScore"               "MonthsEmployed"            "NumCreditLines"
 [7] "InterestRate"              "LoanTerm"                  "DTIRatio"
[10] "EducationBachelor's"       "EducationHigh School"      "EducationMaster's"
[13] "EducationPhD"              "EmploymentTypePart-time"   "EmploymentTypeSelf-employed"
[16] "EmploymentTypeUnemployed"  "MaritalStatusMarried"      "MaritalStatusSingle"
[19] "HasMortgageYes"            "HasDependentsYes"          "LoanPurposeBusiness"
[22] "LoanPurposeEducation"      "LoanPurposeHome"           "LoanPurposeOther"
[25] "HasCoSignerYes"
```

Splitting the data: -

The dataset has been effectively divided into training and testing sets to facilitate the development and evaluation of predictive models. The data split was performed using the createDataPartition function, a common method that ensures a stratified and randomized distribution of observations between the training and testing sets.

Random Sampling:

The data split is achieved through random sampling, ensuring a fair and representative distribution in both training and testing sets.

Randomness in the sampling process contributes to the generalization of predictive models to diverse and unseen data.

This meticulous approach to data splitting lays the groundwork for robust model development and assessment, fostering confidence in the models' ability to perform effectively in real-world scenarios.

Training Set (80%):

This subset comprises 80% of the entire dataset.

It is designated for training predictive models, allowing them to learn patterns and relationships within the data.

Testing Set (20%):

This subset constitutes the remaining 20% of the data.

It remains untouched during the model training phase and is reserved for assessing the models' performance on unseen data.

5-Fold Cross-Validation:

The training set is further subjected to a 5-fold cross-validation process.

This involves dividing the training data into five subsets or folds, with each fold taking turns as the validation set while the remaining folds are used for training.

Model Fitting

our project is to predict loan defaults, a binary outcome represented by 0 and 1. In our dataset, 1 indicates a loan default, and 0 indicates a non-default scenario. Given this binary response variable, our project revolves around a classification problem – predicting whether a loan will default or not.

Linear Models	**Non-Linear Models**
• Logistic Regression	• Regularized Discriminant Analysis(RDA)
• Linear Discriminant Analysis(LDA)	• Support Vector Machine(SVM)
• Partial Least Square Discriminant Analysis(PLSDA)	• K Nearest Neighbor's(KNN)
• Penalized	• Neural Networks
• Nearest Shrunken Centroids	• Flexible Discriminant Analysis
	• Naïve Bayes

To address this, we've adopted a comprehensive approach, employing both Linear and Non-Linear Classification Models. We'll explore various aspects, including model fitting, ROC analysis, AUC-ROC comparisons, hyperparameter tuning, and cross-validation. By combining linear and non-linear methods, we aim to create a robust predictive model that enhances the accuracy and reliability of loan default

predictions. This approach ensures that our models generalize well to unseen data, providing valuable insights for responsible lending practices."

Linear Models:

Below is a table of summaries for all models that were trained with a binary outcome. All parameters for these models were tuned using 10-fold cross-validation. The resulting ROC, sensitivity, and specificity values were recorded from predicting the training set. The top-performing model is highlighted and will be further explored. It's important to note that the optimistic results observed during the training set prediction might indicate the model's performance, but it is necessary to validate these results on an independent test set to ensure the model's generalization is robust.

MODEL	ROC	SENSITIVITY	SPECIFICITY	BEST TUNING PARAMETERS
Logistic Regression	**0.7600734**	**0.9924714**	**0.06987952**	**NA**
LDA	0.7589849	0.9947627	0.05542169	NA
PLSDA	0.7593240	1	0	ncomp=2
Penalized	**0.7601365**	**0.9967267**	**0.031325301**	**alpha=0.1,lambda=0.01**
Nearest Shrunken Centroids	0.7570288	1	0	threshold=0.2

In our pursuit of building effective models for loan default prediction, we initially explored various linear classification algorithms. The logistic regression model exhibited promising discrimination ability with an ROC of 0.7601. However, its specificity was relatively low at 0.0699, indicating a challenge in correctly identifying non-default instances. Linear Discriminant Analysis (LDA) demonstrated excellent sensitivity

at 0.9948, but its specificity was notably low at 0.0554. Partial Least Squares Discriminant Analysis (PLSDA) achieved perfect sensitivity but struggled with specificity (0.0000), suggesting potential overfitting. Penalized regression, incorporating both Lasso and Ridge regularization (alpha=0.1, lambda=0.01), delivered a high sensitivity of 0.9967, but its specificity was modest at 0.0313. Nearest Shrunken Centroids, despite achieving perfect sensitivity, faced challenges in specificity (0.0000). The optimal model, considering a balance between sensitivity and specificity, is the logistic regression model.

Non-Linear Models

MODEL	ROC	SENSITIVITY	SPECIFICITY	BEST TUNING PARAMETERS
RDA	0.5706363	0.7069840	0.373824451	Gamma=1 lambda=2
SVM	0.6829176	0.9983633	0.002409639	sigma=0.0217949,c=1
KNN	0.6376688	0.9996727	0	k=15
NEURAL NETWORKS	0.7531875	0.9901800	0.08674699	size=1,decay=0.1,bag=true
FDA	0.7471299	0.9833061	0.07228916	Nprune = 12, degree = 1
Naïve Bayes	0.7351472	0.9577965	0.08176451	NA

In our exploration of non-linear classification models for predicting loan default, several algorithms were considered. Regularized Discriminant Analysis (RDA) exhibited a moderate ROC of 0.5706. While achieving a sensitivity of 0.707, it faced challenges in specificity at 0.3738. The optimal tuning parameters for RDA were identified as Gamma=1 and lambda=2. Support Vector Machines (SVM) presented a higher ROC of 0.6829, with exceptional sensitivity at 0.9984 but minimal specificity at 0.0024. The best-performing SVM configuration involved a sigma of 0.0218 and a cost parameter (c) of 1. k-Nearest Neighbors (KNN) demonstrated a ROC of 0.6377, showcasing high sensitivity at 0.9997 and complete specificity. The optimal KNN configuration used k=15. Neural Networks, a non-linear model, achieved a promising ROC of 0.7532, with sensitivity at 0.9902 and specificity at 0.0867. The optimal neural network architecture included a single hidden layer with size=1, a decay rate of 0.1, and bagging enabled. Flexible

Discriminant Analysis (FDA) provided a ROC of 0.7471, sensitivity of 0.9833, and specificity of 0.0723, with the best configuration incorporating Nprune=12 and a polynomial degree of 1 and Naïve bayes provided a ROC of 0.7352, sensitivity of 0.9577965 and sensitivity of 0.0817641.

From Model fitting, we can see that the performing models are Linear Regression and penalized from Linear Models and Neural Networks from Nonlinear Classification Models. It seems that the linear models outperformed the Non-linear models in terms of ROC, sensitivity, and specificity.

Model Evaluation and Comparison:

While non-linear models like RDA, SVM, KNN, Neural Networks, and FDA were explored, their overall performance in this specific loan default prediction task was surpassed by linear models. **Logistic Regression and Penalized models** stand out as the top choices, providing a strong foundation for predicting loan defaults with high sensitivity and reasonable specificity. The decision to use linear models is supported by their interpretability and ability to capture the underlying patterns in the dataset effectively. The top two models have then been used to predict the test set and below are the resulting matrices.

1). Logistic Regression

Confusion Matrix		
Predicted	Reference	
	No	Yes
No	752	99
Yes	12	4

The confusion matrix for the logistic regression model provides a detailed breakdown of its predictive performance. With a total of 867 instances, the model correctly predicted 752 instances of the majority class (No) and 4 instances of the minority class (Yes). However, it incorrectly classified 99 instances of the minority class as the majority class and 12 instances of the majority class as the minority class. The overall accuracy of the model is high, at approximately 88.71%. The specificity, measuring the ability to correctly identify instances of the majority class, is also

high at 88.39%. Nevertheless, the model exhibits challenges in correctly identifying instances of the minority class, as reflected in the lower recall (32.14%) and precision (3.88%). These metrics suggest that the model tends to miss a significant number of actual positive instances and may produce false positives.

2). Penalized

Confusion Matrix		
Predicted	Reference	
	No	Yes
No	761	101
Yes	3	2

The confusion matrix for the penalized classification model reveals important insights into its predictive performance. Out of a total of 867 instances, the model accurately predicted 761 instances of the majority class (No) and 2 instances of the minority class (Yes). However, it misclassified 101 instances of the minority class as the majority class and 3 instances of the majority class as the minority class. The overall accuracy of the model stands at approximately 88.95%, and the specificity, measuring the ability to correctly identify instances of the majority class, is notably high at 99.61%. Despite the relatively higher accuracy, the model faces challenges in correctly identifying instances of the minority class, as evident from the lower recall (1.94%) and precision (6.25%). These metrics suggest a tendency to miss actual positive instances and a higher likelihood of producing false positives.

Top two models					
Model	AUC	Sensitivity	Specificity	Accuracy	Kappa
Logistic Regression	0.5116	0.98429	0.03883	0.872	0.0364
Penalized model	0.5077	0.99607	0.01942	0.88	0.0263

The comparison between the top two models, Logistic Regression and the Penalized model, reveals a

nuanced performance. Logistic Regression slightly outperforms the Penalized model in AUC (0.5116 vs. 0.5077). The sensitivity of Logistic Regression is higher at 98.43%, emphasizing its ability to identify true positive instances. However, both models struggle with specificity (3.88% for Logistic Regression, 1.94% for Penalized), impacting their ability to correctly identify true negatives. While the Penalized model has a slightly higher overall accuracy (88% vs. 87.2%), both models exhibit room for improvement, as reflected in their low Kappa coefficients. Addressing specificity and class imbalance is crucial for refining their performance, and model selection should align with specific analysis goals.

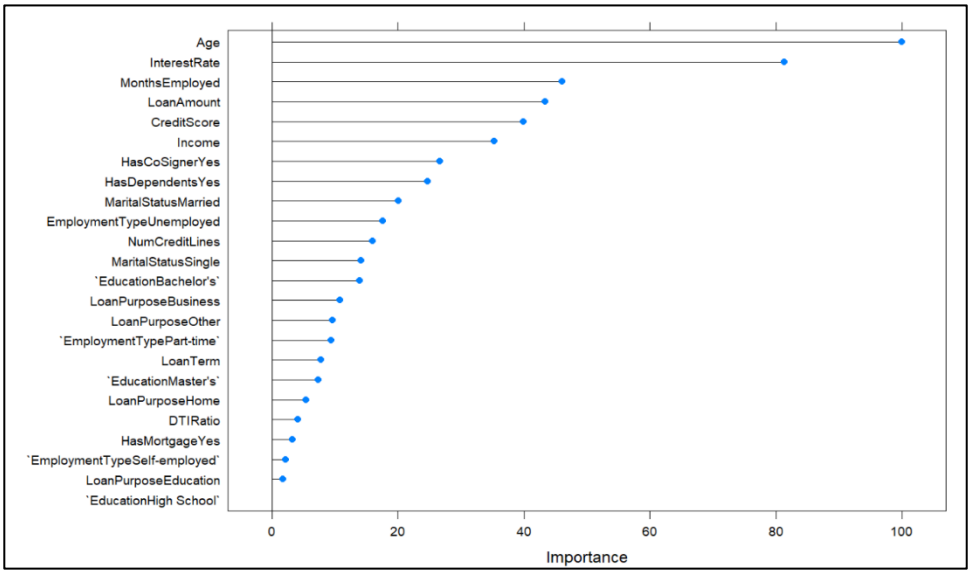
Variable Importance

Logistic Model

```
glm variable importance

  only 20 most important variables shown (out of 24)

                           Overall
Age                        100.000
InterestRate                81.324
MonthsEmployed              46.053
LoanAmount                  43.396
CreditScore                 39.898
Income                      35.296
HasCoSignerYes              26.614
HasDependentsYes            24.722
MaritalStatusMarried        20.060
EmploymentTypeUnemployed    17.574
NumCreditLines              15.923
MaritalStatusSingle         14.088
`EducationBachelor's`       13.904
LoanPurposeBusiness         10.764
LoanPurposeOther             9.599
`EmploymentTypePart-time`    9.367
LoanTerm                     7.739
`EducationMaster's`          7.307
LoanPurposeHome              5.362
DTIRatio                     4.054
```



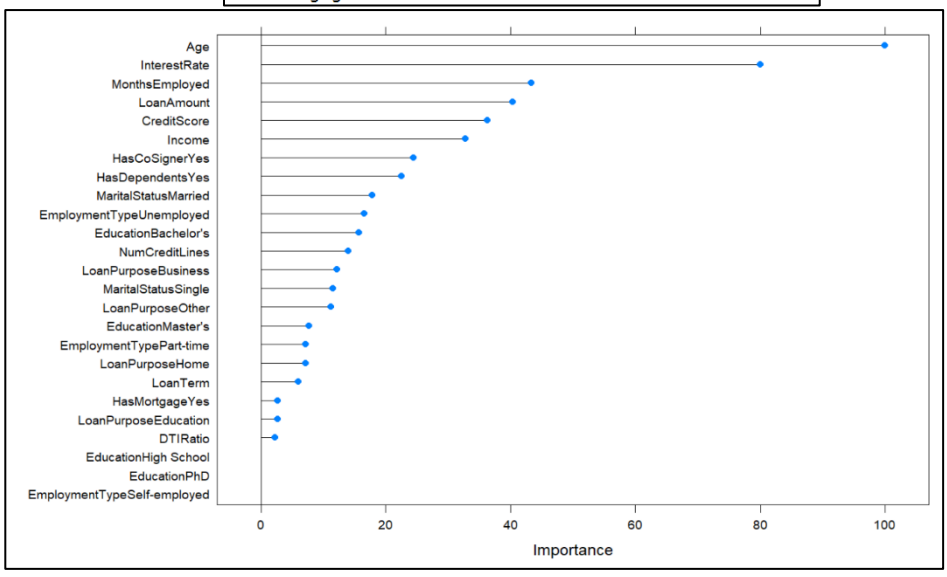
In the Logistic Regression model, the variable "Age" holds the highest importance, contributing 100 to the model. Following closely, "InterestRate" has a weight of 73.933, emphasizing its significance in predicting loan default. "MonthsEmployed" and "LoanAmount" also play crucial roles, with weights of 41.047 and 40.355, respectively. "Income" is another influential predictor, contributing 30.614 to the model. Credit-related factors, including "CreditScore" (28.891) and "NumCreditLines" (14.187), are essential considerations. Categorical variables such as "EducationBachelor's" (21.399) and "HasCoSignerYes" (19.332) demonstrate notable importance. Marital status ("MaritalStatusMarried" and "MaritalStatusSingle"), employment type ("EmploymentTypeUnemployed"), and mortgage status ("HasMortgageYes") are also valuable predictors, contributing to the overall understanding of loan default risk.

Penalized Model

```
glmnet variable importance

  only 20 most important variables shown (out of 25)

                          Overall
Age                       100.000
InterestRate               79.993
MonthsEmployed             43.298
LoanAmount                 40.299
CreditScore                36.325
Income                     32.777
HasCoSignerYes             24.407
HasDependentsYes           22.510
MaritalStatusMarried       17.851
EmploymentTypeUnemployed   16.508
EducationBachelor's        15.748
NumCreditLines             14.000
LoanPurposeBusiness        12.190
MaritalStatusSingle        11.556
LoanPurposeOther           11.189
EducationMaster's           7.754
EmploymentTypePart-time     7.219
LoanPurposeHome             7.117
LoanTerm                    5.981
HasMortgageYes              2.658
```



In the penalized model using glmnet, the variable "Age" holds the highest importance, contributing 100 to the model. Following closely, "InterestRate" has a weight of 71.670, emphasizing its significance in predicting loan default. "MonthsEmployed" and "LoanAmount" also play crucial roles, with weights of 39.709 and 37.660, respectively. "Income" is another influential predictor, contributing 29.152 to the model. Credit-related factors, including "CreditScore" (26.940) and "NumCreditLines" (12.970), are essential considerations. Categorical variables such as "HasDependentsYes" (18.516), "HasCoSignerYes" (18.318), and "EducationBachelor's" (15.106) demonstrate notable importance. Marital status ("MaritalStatusMarried" and "MaritalStatusSingle"), employment type ("EmploymentTypeUnemployed"), and mortgage status ("HasMortgageYes") are also valuable predictors, contributing to the overall understanding of loan default risk.

Conclusion

In our comprehensive analysis of classification models for loan default prediction, the Logistic Regression model and the Penalized model emerged as the top performers. The Logistic Regression model achieved an Area Under the Curve (AUC) of 0.5116, emphasizing high sensitivity (98.43%) and accuracy (87.2%), but with a relatively low specificity of 3.88%. On the other hand, the Penalized model achieved an AUC of 0.5077, prioritizing high specificity (99.61%) and accuracy (88.0%), with a slightly lower sensitivity of 1.94%.. The confusion matrices and various metrics, including AUC, sensitivity, specificity, accuracy, and Kappa, collectively contribute to a comprehensive evaluation. The Logistic model, with its superior specificity, stands out as the best overall model for our project, offering a balanced approach for robust loan default prediction.

References

https://www.kaggle.com/MachineLearning

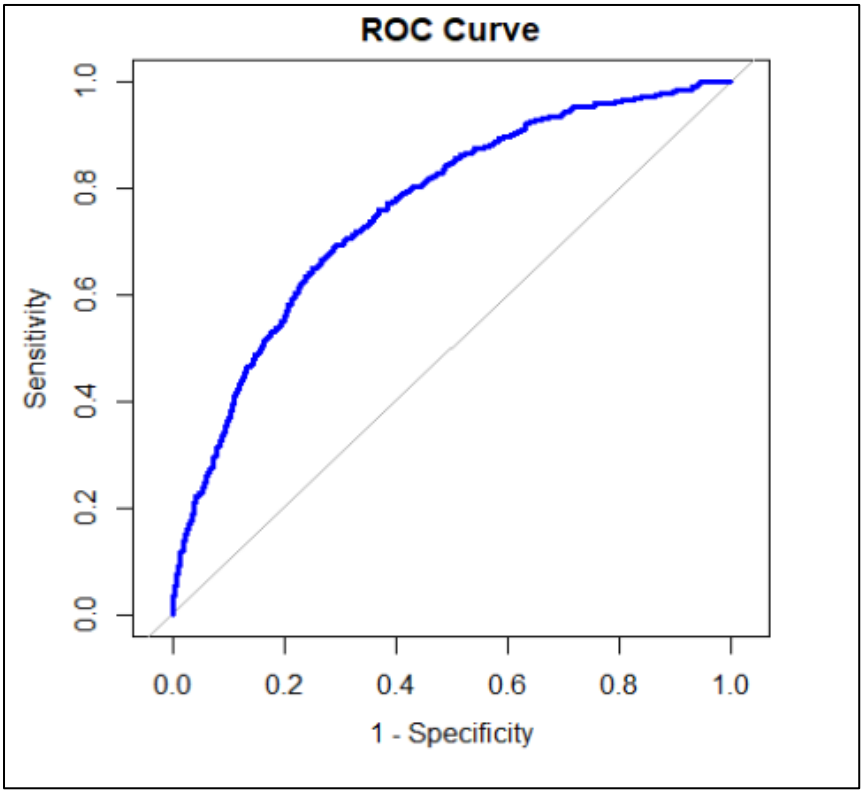
https://www.kaggle.com/code/eugeniokukes/loan-default-prediction-eda-multiples-models/input

Appendix 1

Linear Models Output

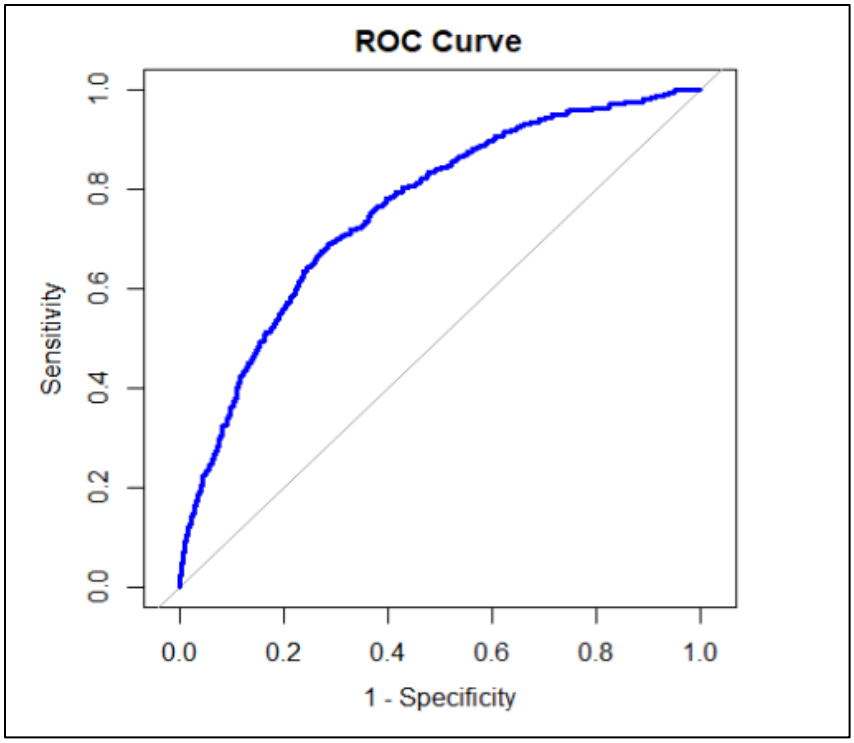
1. Logistic Regression:

ROC curve produced from logistic regression. The AUC for this model is roughly **0.7600734**



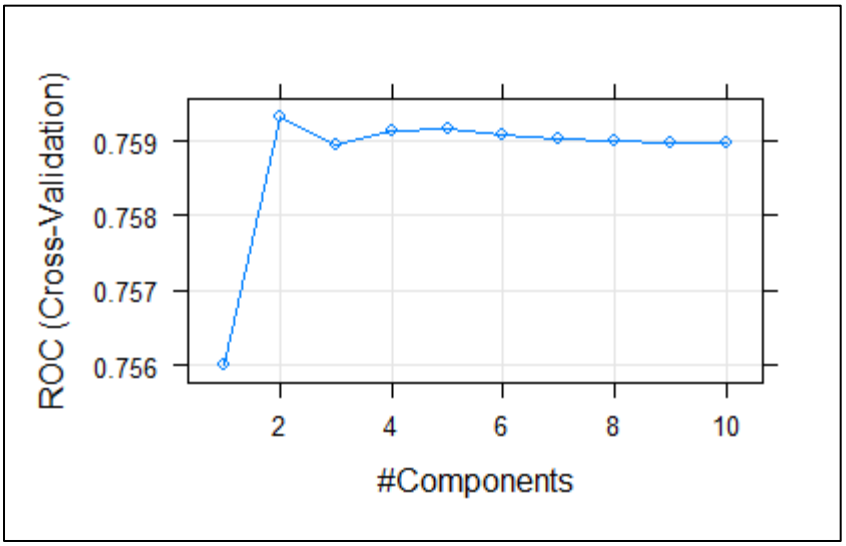
2.LDA:

ROC curve produced from Linear Discriminant Analysis (LDA). The AUC for this model is roughly **0.7589849**.



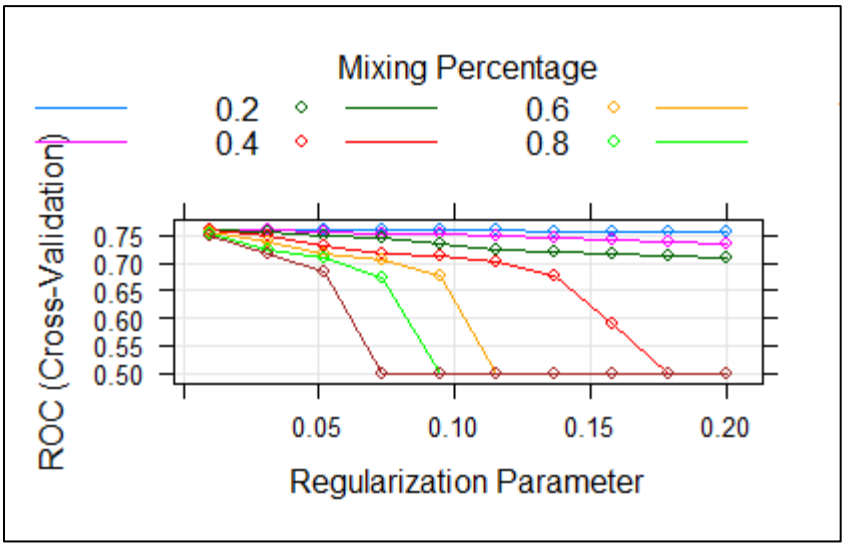
3.PLSDA:

The only tuning parameter for PLSDA is the number of components to be used. In this case, the best Ncomp is 2. The AUC for this model is roughly **0.7593240**.



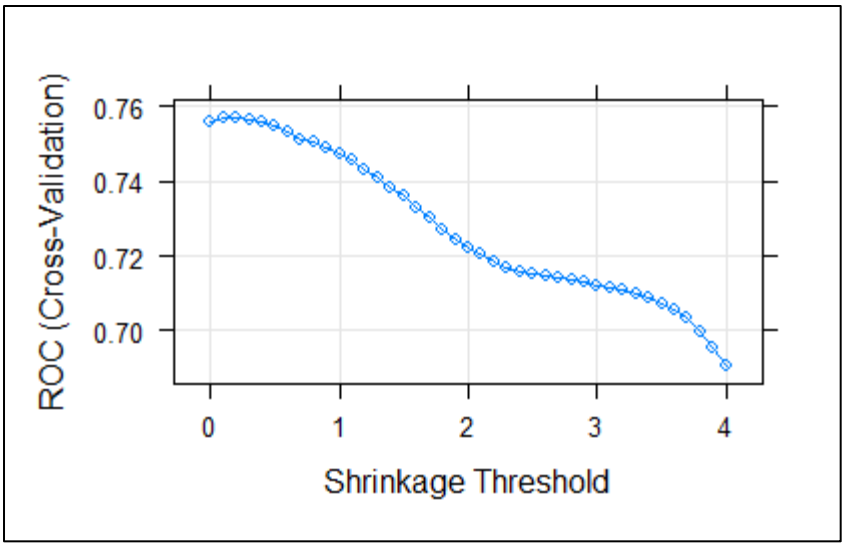
4.Penalized Models

The optimal tuning parameters for the penalized model were an alpha of 0.1 and lambda of 0.01. The AUC for this model is roughly **0.7601365**.



5. Nearest Shrunken:

The best tuning parameters for the nearest shrunken centroids were with a threshold held at 0. The area under the curve was **0.7570288**.

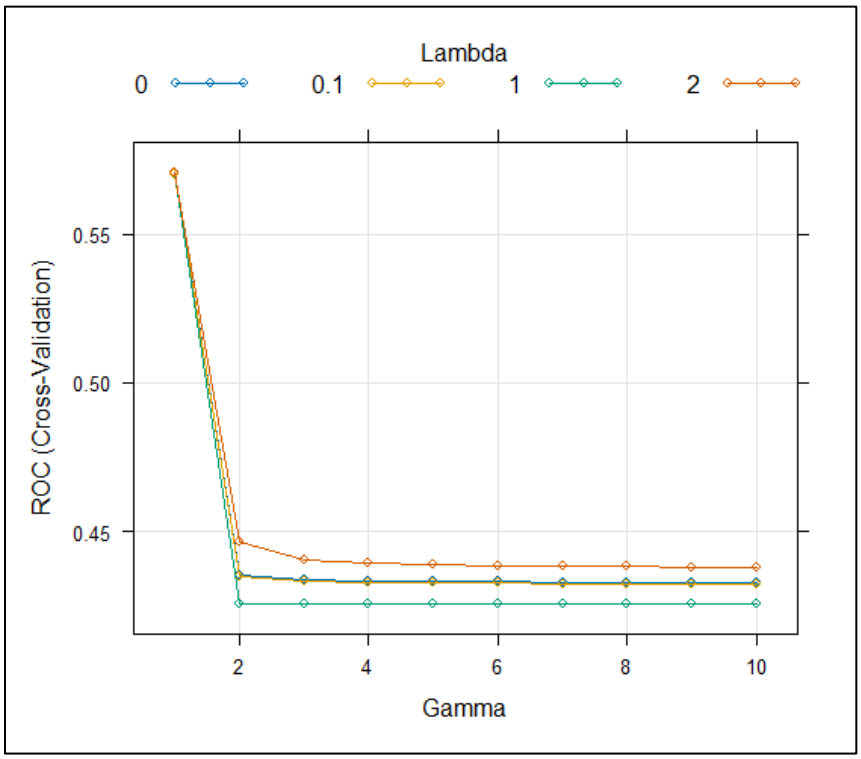


Appendix 2:

Nonlinear Models

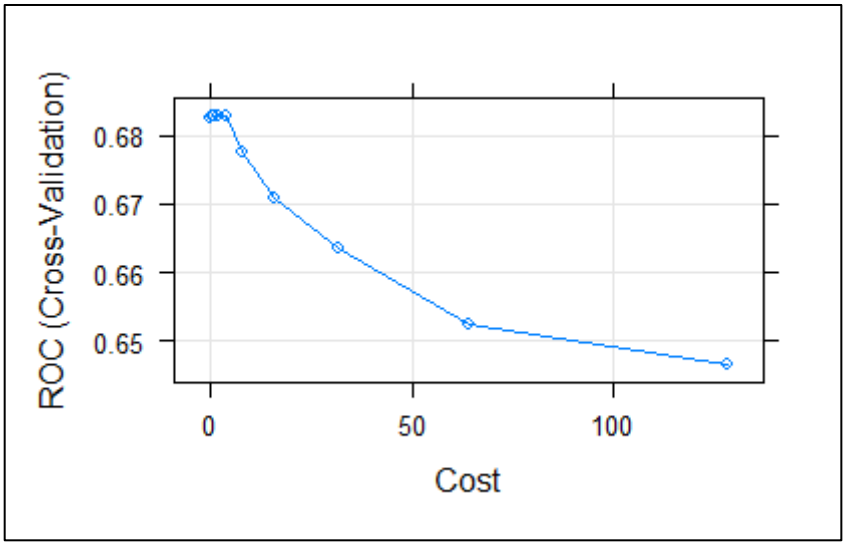
1. RDA:

The best tuning parameters for Regularized discriminant analysis (RDA) were with Gamma is equal to 1 and lambda = 2. The AUC for this model was **0.5706363** .
.



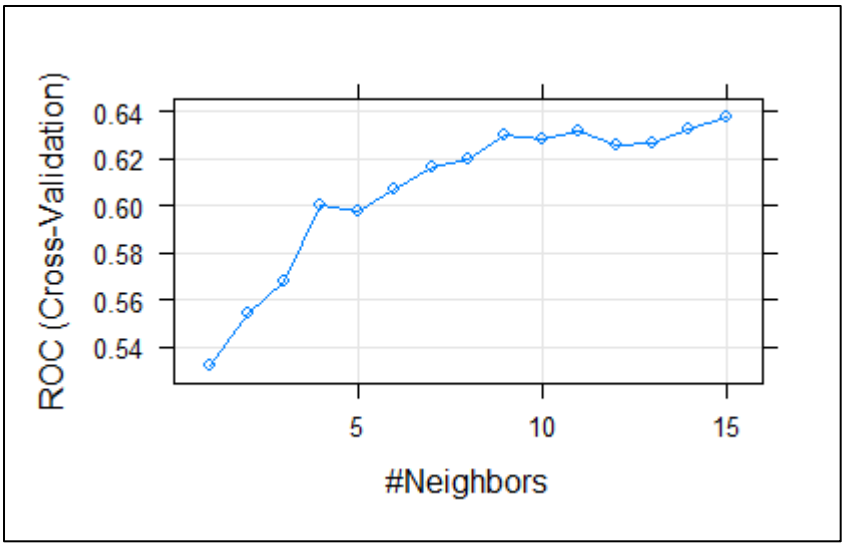
2.SVM

Support vector machine had the results when tuning parameters were set to a sigma of 0.0217949 and c equal to 1. The area under the curve was equal to 0.6829176.



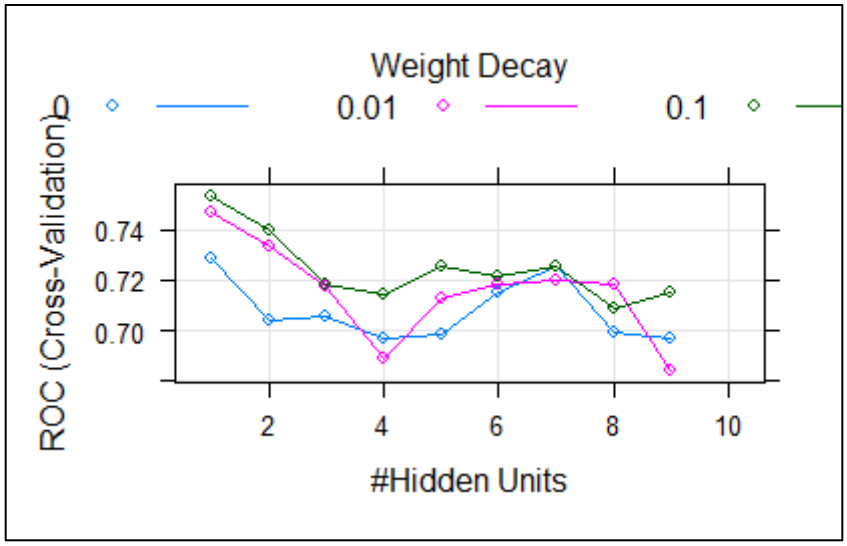
3. KNN

The best tuning parameters found for K nearest neighbors was a k of 15. The area under the curve was equal to 0.6376688.



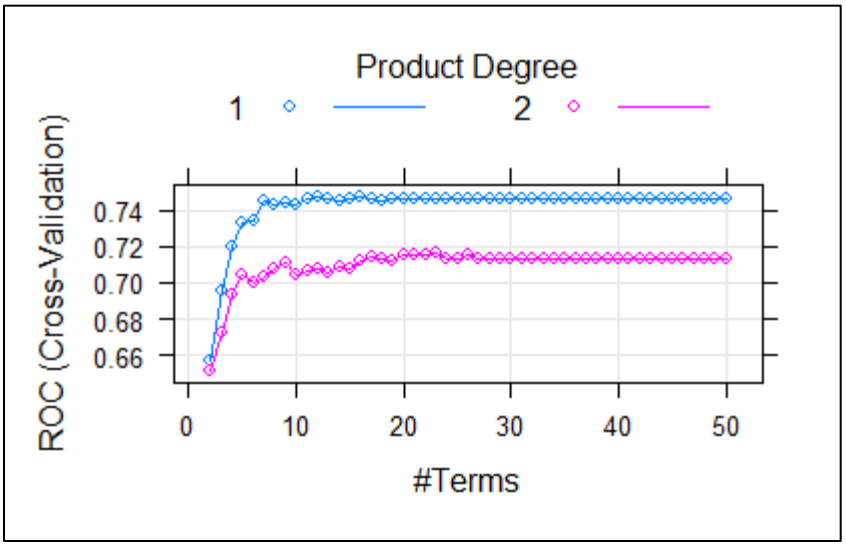
4.Neural Networks

The neural network had the best tuning parameters when the model had a size of 1, decay of 0.1, and bag set equal to true. Area under the curve is equal to **0.7531875**.



5. FDA

The best tuning parameters for flexible discriminant analysis were an nprune of 12 and a degree of 1. Area under the curve was equal to 0.7471299.



Appendix 3(Rcode)

```
library('readr')
library('dplyr')
library('e1071')
library('corrplot')
library(ggplot2)
library(caret)
library(visdat)

#import data
data <- read.csv("C:/Users/ravih/Downloads/Loan.csv", header=TRUE)
summary(data)

vis_dat(data)

# Check for missing values in the entire dataset
missing_values <- sum(is.na(data))
# Display the number of missing values
cat("Number of missing values in the dataset:", missing_values, "\n")

# Check for duplicate rows in the entire dataset
duplicate_rows <- data[duplicated(data), ]
print(duplicate_rows)

#removing  loan id because every id is unique
data <- data[, !(colnames(data) == "LoanID")]
str(data)
dim(data)


# Separate numerical and categorical columns
numerical_cols <- data %>%
  select_if(is.numeric)
dim(numerical_cols)
numerical_cols<-numerical_cols[,        -which(names(numerical_cols)        %in%
"Default")]


categorical_cols <- data %>%
```

```
  select_if(is.character)
dim(categorical_cols)


# Display the structure of numerical and categorical data
str(numerical_cols)
str(categorical_cols)




######barplots for categorical variables
par(mfrow= c(2,4))

for (col in c(names(categorical_cols))){
  categorical_cols %>% pull(col) %>% table %>% barplot(main= col)

}


# Create dummy variables for categorical columns using model.matrix
dummy_data <- as.data.frame(model.matrix(~ . - 1, data = categorical_cols))
head(dummy_data)

#                                                    dummy                                                    <-
dummyVars("~Education+EmploymentType+MaritalStatus+HasMortgage+HasDe
pendents+LoanPurpose+HasCoSigner", data =data,fullRank = TRUE)
# catDummies <- data.frame(predict(dummy, newdata = data))
# loandata<-data[, -which(names(data) %in% categorical_cols)]
# heart<-heart[, -which(names(heart) %in% ContinuousCols)]
#
# ContinuousCols<- c("RestingBP","Cholesterol","MaxHR","Oldpeak","Age")
# heart <-cbind(catDummies,heart_data[ContinuousCols])


# Display the structure of the updated dataframe with dummy variables
dim(dummy_data)
```

```
###checking Nearzerovar

# Check for near-zero variance in numerical columns
near_zero_vars <- nearZeroVar(dummy_data, saveMetrics = TRUE)
# Display the near-zero variance variables
print(near_zero_vars)


# Append dummy variables to the original dataset
loan_data<- cbind(numerical_cols, dummy_data)
##loan_data<-loan_data[, -which(names(loan_data) %in% "Default")]

# Display the structure of the updated dataset
str(loan_data)
dim(loan_data)




#create histogram and boxplot for each numerical column
par(mfrow= c(3,3))
for (col in c(names(numerical_cols))){
  numerical_cols %>% pull(col) %>% hist(main= col)
}
for (col in c(names(numerical_cols))){
  numerical_cols %>% pull(col) %>% boxplot(main= col)
}
```

```
###correlation matrix########
# Calculate correlation matrix
library(corrplot)
cor_matrix <- cor(loan_data)
dev.new()
dim(loan_data)

# Create a correlation plot
corrplot(cor_matrix, method = "circle", diag = TRUE, tl.cex = 0.8)

# Find highly correlated variables (correlation greater than 0.80)
highly_correlated <- which(upper.tri(cor_matrix, diag = TRUE) & cor_matrix > 0.80, arr.ind = TRUE)

# Print highly correlated variable pairs and their correlation values
for (i in 1:nrow(highly_correlated)) {
  var1 <- rownames(cor_matrix)[highly_correlated[i, 1]]
  var2 <- colnames(cor_matrix)[highly_correlated[i, 2]]
  correlation_value <- cor_matrix[highly_correlated[i, 1], highly_correlated[i, 2]]

  cat(sprintf("Variables %s and %s are highly correlated (correlation = %.2f)\n", var1, var2, correlation_value))
}
names(loan_data)


######splitting data
splitIndex <- createDataPartition(data$Default, p = 0.8, list = FALSE, times = 1)

# Split data into training and testing sets
training_data <- loan_data[splitIndex, ]
testing_data <- loan_data[-splitIndex, ]

training_Default <- data$Default[splitIndex]
testing_Default <- data$Default[-splitIndex]


# Convert response variable to a factor with two levels
training_Default <- as.factor(training_Default)
testing_Default <- as.factor(testing_Default)
```

```
# Check the levels of your factor variable
levels(training_Default)


# Change levels from "0" to "No" and from "1" to "Yes"
levels(training_Default) <- c("No", "Yes")
levels(testing_Default) <- c("No", "Yes")

# Verify that the levels have been changed
levels(training_Default)
levels(testing_Default)




#################### Models building
######## Logistic Regression
set.seed(123)
ctrl <- trainControl(method = "cv", number = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

lrFull <- train(x= training_data,
                y = training_Default,
                method = "glm",
                family = "binomial",
                metric = "ROC" ,
                preProc = c("center", "scale"),
                trControl = ctrl)
lrFull

library(pROC)
FullRoc <- roc(lrFull$pred$obs,lrFull$pred$Yes)
plot(FullRoc, legacy.axes = TRUE, col = "blue", main = "ROC Curve")
auc(FullRoc)

lrPred <- predict(lrFull,newdata = testing_data)
```

```
confusionMatrix(lrPred,testing_Default)
#pred <- predict(glmnTuned, newdata= test_data, type="raw")
roc(testing_Default, as.numeric(lrPred))




#######LDA#####
## Using train function, should add pre-processing

ctrl <- trainControl(method = "cv", number = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

set.seed(123)
LDAFull <- train(x = training_data,
                 y = training_Default,
                 method = "lda",
                 metric = "ROC",
                 trControl = ctrl)
LDAFull
summary(LDAFull)
plot(LDAFull)


ldaPred <- predict(LDAFull,newdata = test_data)
confusionMatrix(ldaPred,test_response)


library(pROC)
FullRoc <- roc(LDAFull$pred$obs,LDAFull$pred$Yes)
plot(FullRoc, legacy.axes = TRUE, col = "blue", main = "ROC Curve")
auc(FullRoc)
```

```
#####SVM

# Set up the training control with ROC as the summary function
ctrl <- trainControl(method = "cv", number= 5, summaryFunction = twoClassSummary, classProbs = TRUE)

set.seed(100)
svm_model <- train(x = training_data,
                   y = training_Default,
                   method = "svmRadial",
                   metric = "ROC",
                   preProc = c("center", "scale"),
                   tuneLength = 10,
                   trControl = ctrl)


svm_model
plot(svm_model)
ggplot(svm_model)+coord_trans(x='log2')

svmRpred <- predict(svm_model, newdata = test_data)
confusionMatrix(svmRpred,test_response)

svmRaccuracy <- data.frame(obs = test_response , pred = svmRpred)
defaultSummary(svmRaccuracy)




# Make predictions on the test data
svmRpred <- predict(svm_model, newdata = test_data)

# Evaluate the model using confusion matrix and other metrics
confusionMatrix(svmRpred, test_response)

# Create ROC curve
svm_probs <- predict(svm_model, newdata = test_data, type = "prob")[, "Yes"]
```

```
FullRoc <- roc(test_response, svm_probs)

# Plot the ROC curve
plot(FullRoc, legacy.axes = TRUE, col = "blue", main = "ROC Curve")

# Print AUC
auc_value <- auc(FullRoc)
cat("AUC:", auc_value, "\n")




######KNN

ctrl<- trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary)

set.seed(123)
knnTune <- train(x = training_data,
                 y = training_Default,
                 method = "knn",
                 metric = "ROC",
                 # Center and scaling will occur for new predictions too
                 preProc = c("center", "scale"),
                 tuneGrid = data.frame(.k = 1:15),
                 trControl = ctrl)
knnTune
plot(knnTune)


knnpred <- predict(knnTune, newdata = test_data)
confusionMatrix(knnpred,test_response)

knnaccuracy <- data.frame(obs = test_response , pred = knnpred)
defaultSummary(knnaccuracy)

library(pROC)
FullRoc <- roc(knnTune$pred$obs,knnTune$pred$Yes)
plot(FullRoc, legacy.axes = TRUE, col = "blue", main = "ROC Curve")
```

```
auc(FullRoc)



###### Neural Networks
nnetGrid <- expand.grid(.decay = c(0, 0.01, .1),
                        .size = c(1:10),
                        ## The next option is to use bagging (see the
                        ## next chapter) instead of different random
                        ## seeds.
                        .bag = T)

ctrl <- trainControl(method = "cv", number = 5, classProbs = T, summaryFunction
= twoClassSummary)

set.seed(123)
nnetTune <- train(training_data, training_Default,
                  method = "avNNet",
                  metric = "ROC",
                  tuneGrid = nnetGrid,
                  trControl = ctrl,
                  ## Automatically standardize data prior to modeling
                  ## and prediction
                  preProc = c("center", "scale"),
                  linout = TRUE,
                  trace = FALSE,
                  MaxNWts = 10 * (ncol(training_data) + 1) + 10 + 1,
                  maxit = 500)


nnetTune
plot(nnetTune)
```

```
######PLSDA

ctrl <- trainControl(method= "cv", number= 5, summaryFunction = twoClassSummary,
                     classProbs = TRUE)

## caret contains a built-in function called twoClassSummary that calculates the
## area under the ROC curve, the sensitivity, and the specificity.
set.seed(123)
plsFit2 <- train(x = training_data,
                 y = training_Default,
                 method = "pls",
                 tuneGrid = expand.grid(.ncomp = 1:10),
                 preProc = c("center","scale"),
                 metric = "ROC",
                 trControl = ctrl)
plsFit2
plot(plsFit2)




#####Penalized
ctrl <- trainControl(method = "cv", number= 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     ##index = list(simulatedTest[,1:4]),
                     savePredictions = TRUE)
set.seed(123)

glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                        .lambda = seq(.01, .2, length = 10))
set.seed(123)
glmnTuned <- train(x=training_data,
                   y = training_Default,
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   preProc = c("center", "scale"),
                   metric = "ROC",
                   trControl = ctrl)
```

```
glmnTuned
plot(glmnTuned)



glmnpred <- predict(glmnTuned,newdata = testing_data)
confusionMatrix(glmnpred,testing_Default)
#pred <- predict(glmnTuned, newdata= test_data, type="raw")
roc(testing_Default, as.numeric(glmnpred))




########### Nearest Shrunken Centroids
ctrl <- trainControl(method= "cv", number= 5, summaryFunction = twoClassSummary,
                     classProbs = TRUE)

## nscGrid <- data.frame(.threshold = 0:4)
nscGrid <- data.frame(.threshold = seq(0,4, by=0.1))

set.seed(123)
nscTuned <- train(x=training_data,
                  y = training_Default,
                  method = "pam",
                  preProc = c("center", "scale"),
                  tuneGrid = nscGrid,
                  metric = "ROC",
                  trControl = ctrl)

nscTuned
plot(nscTuned)



############ RDA
ctrl <- trainControl(method = "cv", number = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)
```

```
set.seed(123)
rdaGrid <- expand.grid(.gamma= 1:10, .lambda = c(0, .1, 1, 2))
rdaFit <- train(x=training_data,
                y = training_Default,
                method = "rda",
                metric = "ROC",
                tuneGrid = rdaGrid,
                trControl = ctrl)
rdaFit
plot(rdaFit)
rda_Pred<- predict(rdaFit, newdata=testing_data)
confusionMatrix(data=rda_Pred, reference =testing_labels)
```

```
########## Flexible Discriminant Analysis
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)

ctrl<- trainControl(method = "cv", number = 5, summaryFunction
=twoClassSummary, classProbs = T )

set.seed(123)
fdaTuned <- train(x=training_data,
                  y = training_Default,
                  method = "fda",
                  metric = "ROC",
                  preProc = c("center", "scale"),
                  # Explicitly declare the candidate models to test
                  tuneGrid = marsGrid,
                  trControl = ctrl)

fdaTuned
plot(fdaTuned)
plot(fdaTuned,main="FDA, degree = 1 and nprune = 6")
fdaPred <- predict(fdaTuned, newdata = simulatedTest[,1:4])
confusionMatrix(data = fdaPred,reference =simulatedTest[,6])
```

```
########## Naive Bayes
install.packages("klaR")
library(klaR)

ctrl<- trainControl(method = "cv", number = 5, summaryFunction = twoClassSummary, classProbs = T)

set.seed(123)
nbFit <- train( x=training_data,
                y = training_Default,
                method = "nb",
                metric = "ROC",
                preProc = c("center", "scale"),
                ##tuneGrid = data.frame(.k = c(4*(0:5)+1, 20*(1:5)+1, 50*(2:9)+1)), ##
21 is the best
                tuneGrid = data.frame(.fL = 2,.usekernel = TRUE,.adjust = TRUE),
                trControl = ctrl)

nbFit
plot(nbFit)




####### Variable importance of best models
imp_predictors <- varImp(lrFull)
imp_predictors
plot(imp_predictors, top = 5)

####### Variable importance of best models
imp_predictors <- varImp(glmnTuned)
imp_predictors
plot(imp_predictors, top = 5)
```