LOAN DEFAULT PREDICTION

Yinan Zhou March 27th, 2024

Introduction

In this comprehensive analysis, we delve into an extensive dataset of loan applicants, with a primary focus on identifying the key factors contributing to loan defaults.

Our study's objective is to enhance our understanding of borrower profiles and determine the most predictive factors for loan defaults. To achieve this, we conducted exploratory data analysis (EDA) using histograms, boxplots, bar charts, and correlation matrix to observe the dataset's distribution. Additionally, we performed hypothesis testing and built various models, including a generalized linear model (GLM), stepwise regression model, and lasso regression model, to identify the best fit for this dataset. We also utilized confusion matrices, ROC curves, and AUC to evaluate the performance of our models.

Through our analysis, we not only reveal the characteristics of individuals more likely to default but also offer strategic insights. These insights can guide the development of more effective lending practices, tailored to mitigate risks, and foster financial reliability. By understanding the underlying patterns and predictors of loan defaults, our aim is to support lenders in making informed decisions that enhance the overall stability and sustainability of financial institutions.

Exploratory Data Analysis

HeadTail of the Dataset Loan

	LoanID	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate
1	I38PQUQS96	56	85994	50587	520	80	4	15.23
2	HPSK72WA7R	69	50432	124440	458	15	1	4.81
3	C1OZ6DPJ8Y	46	84208	129188	451	26	3	21.17
4	V2KKSFM3UN	32	31713	44799	743	0	3	7.07
...	<NA>	...	...	...	...	...	...	...
255344	98R4KDHNND	32	51953	189899	511	14	2	11.55
255345	XQK1UUUNGP	56	84820	208294	597	70	3	5.29
255346	JAO28CPL4H	42	85109	60575	809	40	1	20.9
255347	ZTH91CGL0B	62	22418	18481	636	113	2	6.73

	LoanTerm	DTIRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents
1	36	0.44	Bachelor's	Full-time	Divorced	Yes	Yes
2	60	0.68	Master's	Full-time	Married	No	No
3	24	0.31	Master's	Unemployed	Divorced	Yes	Yes
4	24	0.23	High School	Full-time	Married	No	No
...	...	...	<NA>	<NA>	<NA>	<NA>	<NA>
255344	24	0.21	High School	Part-time	Divorced	No	No
255345	60	0.5	High School	Self-employed	Married	Yes	Yes
255346	48	0.44	High School	Part-time	Single	Yes	Yes
255347	12	0.48	Bachelor's	Unemployed	Divorced	Yes	No

	LoanPurpose	HasCoSigner	Default
1	Other	Yes	0
2	Other	Yes	0
3	Auto	No	1
4	Business	No	0
...	<NA>	<NA>	...
255344	Home	No	1
255345	Auto	Yes	0
255346	Other	No	0
255347	Education	Yes	0

Summary of the Dataset Loan

```
    LoanID              Age            Income          LoanAmount       CreditScore
 Length:255347      Min.   :18.0   Min.   : 15000   Min.   :  5000   Min.   :300.0
 Class :character   1st Qu.:31.0   1st Qu.: 48826   1st Qu.: 66156   1st Qu.:437.0
 Mode  :character   Median :43.0   Median : 82466   Median :127556   Median :574.0
                    Mean   :43.5   Mean   : 82499   Mean   :127579   Mean   :574.3
                    3rd Qu.:56.0   3rd Qu.:116219   3rd Qu.:188985   3rd Qu.:712.0
                    Max.   :69.0   Max.   :149999   Max.   :249999   Max.   :849.0
MonthsEmployed    NumCreditLines    InterestRate      LoanTerm        DTIRatio
 Min.   :  0.00   Min.   :1.000   Min.   : 2.00   Min.   :12.00   Min.   :0.1000
 1st Qu.: 30.00   1st Qu.:2.000   1st Qu.: 7.77   1st Qu.:24.00   1st Qu.:0.3000
 Median : 60.00   Median :2.000   Median :13.46   Median :36.00   Median :0.5000
 Mean   : 59.54   Mean   :2.501   Mean   :13.49   Mean   :36.03   Mean   :0.5002
 3rd Qu.: 90.00   3rd Qu.:3.000   3rd Qu.:19.25   3rd Qu.:48.00   3rd Qu.:0.7000
 Max.   :119.00   Max.   :4.000   Max.   :25.00   Max.   :60.00   Max.   :0.9000
  Education         EmploymentType     MaritalStatus      HasMortgage        HasDependents
 Length:255347      Length:255347      Length:255347      Length:255347      Length:255347
 Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
LoanPurpose         HasCoSigner          Default
 Length:255347      Length:255347      Min.   :0.0000
 Class :character   Class :character   1st Qu.:0.0000
 Mode  :character   Mode  :character   Median :0.0000
                                       Mean   :0.1161
                                       3rd Qu.:0.0000
                                       Max.   :1.0000
```

Structure of the Dataset Loan

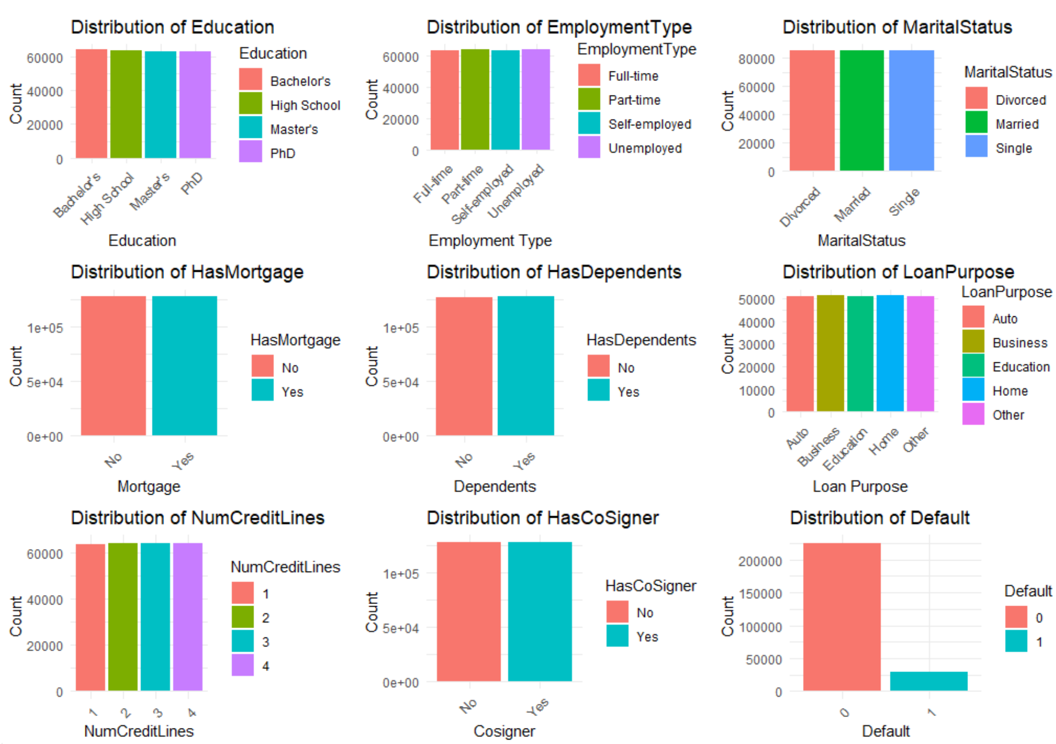
```
'data.frame':      255347 obs. of  18 variables:
 $ LoanID        : chr  "I38PQUQS96" "HPSK72WA7R" "C1OZ6DPJ8Y" "V2KKSFM3UN" ...
 $ Age           : int  56 69 46 32 60 25 38 56 36 40 ...
 $ Income        : int  85994 50432 84208 31713 20437 90298 111188 126802 42053 132784 ...
 $ LoanAmount    : int  50587 124440 129188 44799 9139 90448 177025 155511 92357 228510 ...
 $ CreditScore   : int  520 458 451 743 633 720 429 531 827 480 ...
 $ MonthsEmployed: int  80 15 26 0 8 18 80 67 83 114 ...
 $ NumCreditLines: int  4 1 3 3 4 2 1 4 1 4 ...
 $ InterestRate  : num  15.23 4.81 21.17 7.07 6.51 ...
 $ LoanTerm      : int  36 60 24 24 48 24 12 60 48 48 ...
 $ DTIRatio      : num  0.44 0.68 0.31 0.23 0.73 0.1 0.16 0.43 0.2 0.33 ...
 $ Education     : chr  "Bachelor's" "Master's" "Master's" "High School" ...
 $ EmploymentType: chr  "Full-time" "Full-time" "Unemployed" "Full-time" ...
 $ MaritalStatus : chr  "Divorced" "Married" "Divorced" "Married" ...
 $ HasMortgage   : chr  "Yes" "No" "Yes" "No" ...
 $ HasDependents : chr  "Yes" "No" "Yes" "No" ...
 $ LoanPurpose   : chr  "Other" "Other" "Auto" "Business" ...
 $ HasCoSigner   : chr  "Yes" "Yes" "No" "No" ...
 $ Default       : int  0 0 1 0 0 1 0 0 1 0 ...
```

Detail of the Variables

	Column_name	Column_type	Data_type	Description
0	LoanID	Identifier	string	A unique identifier for each loan.
1	Age	Feature	integer	The age of the borrower.
2	Income	Feature	integer	The annual income of the borrower.
3	LoanAmount	Feature	integer	The amount of money being borrowed.
4	CreditScore	Feature	integer	The credit score of the borrower, indicating their creditworthiness.
5	MonthsEmployed	Feature	integer	The number of months the borrower has been employed.
6	NumCreditLines	Feature	integer	The number of credit lines the borrower has open.
7	InterestRate	Feature	float	The interest rate for the loan.
8	LoanTerm	Feature	integer	The term length of the loan in months.
9	DTIRatio	Feature	float	The Debt-to-Income ratio, indicating the borrower's debt compared to their income.
10	Education	Feature	string	The highest level of education attained by the borrower (PhD, Master's, Bachelor's, High School).
11	EmploymentType	Feature	string	The type of employment status of the borrower (Full-time, Part-time, Self-employed, Unemployed).
12	MaritalStatus	Feature	string	The marital status of the borrower (Single, Married, Divorced).
13	HasMortgage	Feature	string	Whether the borrower has a mortgage (Yes or No).
14	HasDependents	Feature	string	Whether the borrower has dependents (Yes or No).
15	LoanPurpose	Feature	string	The purpose of the loan (Home, Auto, Education, Business, Other).
16	HasCoSigner	Feature	string	Whether the loan has a co-signer (Yes or No).
17	Default	Target	integer	The binary target variable indicating whether the loan defaulted (1) or not (0).

After executing these codes, we gain a comprehensive understanding of the structure and summary of this dataset, providing insights into its central tendency, dispersion, and distribution. The summary indicates that there are no missing values in this dataset, signifying a complete dataset without any NA values.

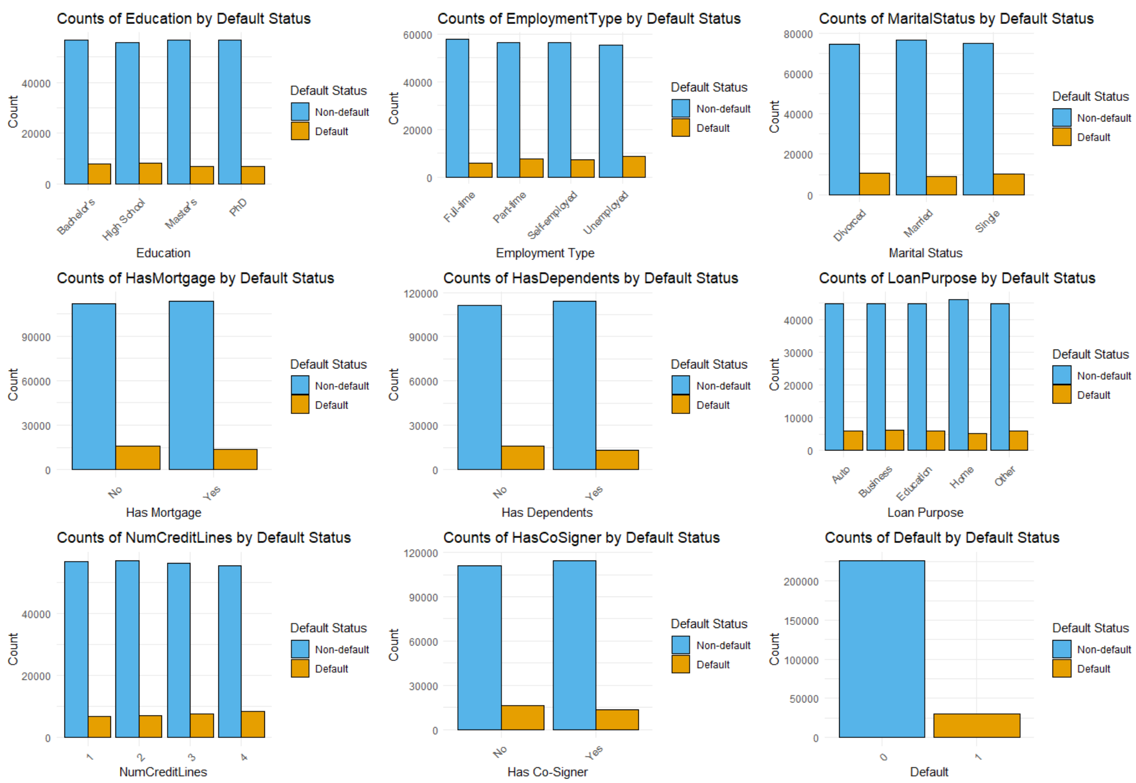
Bar plots of Categorical Variables



The distributions depicted by these categorical variables offer valuable insights into the socio-economic profiles of borrowers and their credit behaviors. This understanding can be instrumental in customizing loan products and conducting more accurate risk assessments.

It's worth noting that the categories of feature variables show nearly equal distribution, suggesting a balanced representation across different categories.

Regarding the target variable, there is an imbalance in the classes, with 11.62% of customers defaulting on their loans and 88.38% of customers not defaulting on their loans. This imbalance may warrant special attention during model training and evaluation to ensure robust predictive performance.



The above graphs display the breakdown of all categorical variables based on their target variable default status.

Counts of Education by Default Status:

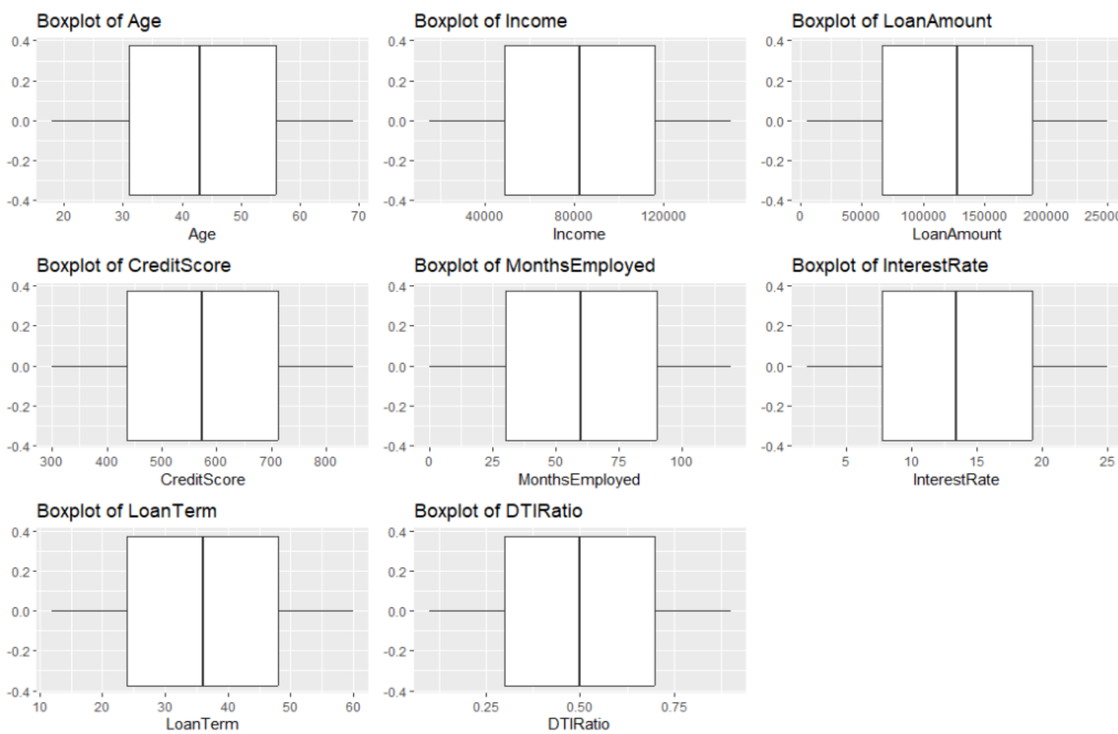
- **Non-default**: The majority of loan applicants across all education levels (Bachelor’s, High School, Master’s, PhD) are non-defaults.
- **Default**: A small percentage of applicants defaulted, with the distribution fairly consistent across education levels. This indicates that education level alone may not be a strong predictor of default.

Counts of Employment Type by Default Status:

- **Non-default**: Most applicants are employed full-time, followed by part-time and self-employed/unemployed.
- **Default**: Default rates are relatively higher among unemployed applicants compared to those who are employed full-time or part-time. This suggests that employment status is a significant factor in predicting defaults.

The target variable classes are almost equally distributed among all categories of feature variables. This is a positive indication, as it suggests that each feature in the dataset has a relationship with the target variable.

Boxplots of Numerical Variables



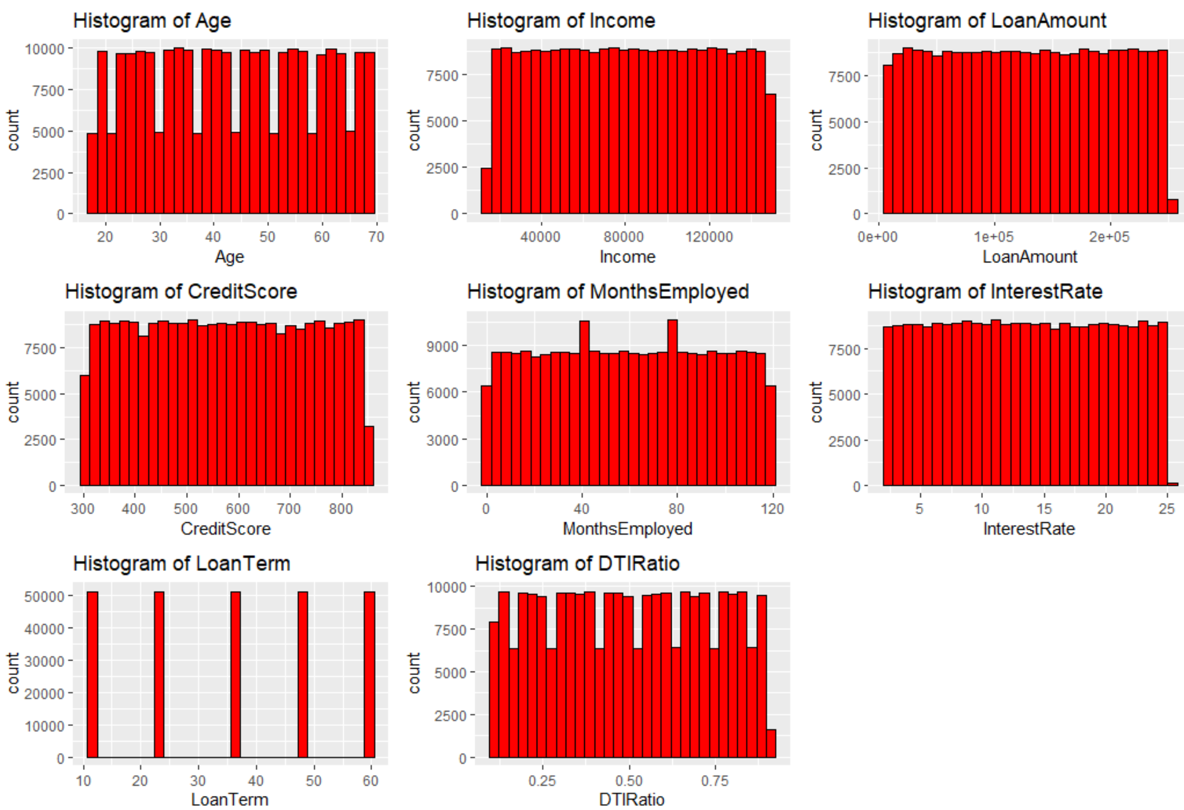
The boxplots provide a clear visual representation of the central tendency and spread of various numerical features. These boxplots help us understand the distribution of all the numerical variables in the dataset, including the presence of outliers and the typical range (interquartile range).

For the **box plot of Age**, the median age of applicants is around 40. The interquartile range (IQR) spans from approximately 30 to 50 years old. There are no significant outliers, indicating a fairly normal distribution of age among applicants.

In the **box plot of Income**, the median income of applicants is around $80,000. The IQR ranges from approximately $60,000 to $100,000. The income distribution appears fairly symmetric, with no significant outliers.

Observation: There is no significant data skewness or outliers in the numerical data, indicating a fairly normal and symmetric distribution across these features.

Histograms of Numerical Variables



Based on these histograms, we can easily understand the distribution of each numerical variable. The distribution appears relatively uniform across most numerical variables, with specific patterns evident in some features.

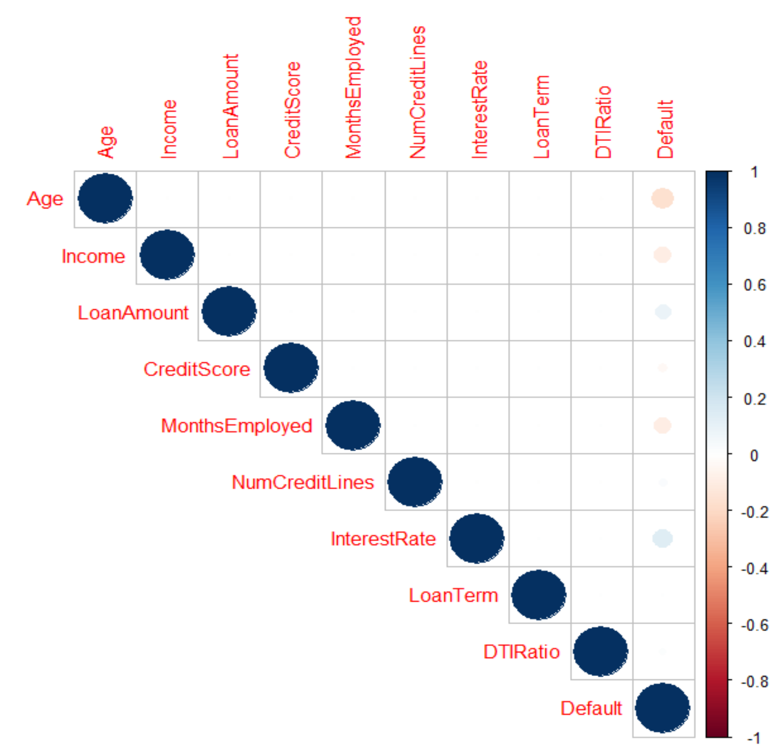
- **Histogram of Age**: The distribution of age is relatively uniform across the range, with slight dips at certain intervals. The majority of applicants are between 20 and 70 years old.
- **Histogram of Income**: Income shows a relatively uniform distribution, with a slight decrease towards the higher end. Most applicants have incomes between $0 and $150,000, with fewer applicants having incomes above $150,000.
- **Histogram of Interest Rate**: Interest rates show a relatively uniform distribution across the range, with most rates falling between 5% and 20%.
- **Histogram of Loan Term**: The distribution of loan terms shows distinct peaks at specific intervals, indicating that certain loan terms are more common.

- **Histogram of DTI Ratio (Debt-to-Income Ratio)**: The distribution of DTI ratios is relatively uniform, with a slight decrease towards the higher end. Most applicants have DTI ratios between 0 and 0.75 (or 75%).

These histograms provide a visual representation of the frequency distribution for each numerical feature. The data appears to be relatively uniformly distributed across most features, with some features showing distinct patterns that could be important for further analysis.

Correlation Matrix for Numeric Variables

```
> cor(matrix_data)
                       Age        Income    LoanAmount   CreditScore MonthsEmployed
Age             1.0000000000 -0.0012440952 -0.0022127413 -5.481709e-04  -3.413880e-04
Income         -0.0012440952  1.0000000000 -0.0008653257 -1.430447e-03   2.674877e-03
LoanAmount     -0.0022127413 -0.0008653257  1.0000000000  1.261270e-03   2.816836e-03
CreditScore    -0.0005481709 -0.0014304474  0.0012612696  1.000000e+00   6.128273e-04
MonthsEmployed -0.0003413880  0.0026748770  0.0028168363  6.128273e-04   1.000000e+00
NumCreditLines -0.0008897680 -0.0020164097  0.0007944049  1.604201e-05   1.267119e-03
InterestRate   -0.0011273828 -0.0023034253 -0.0022911190  4.361387e-04   9.557276e-05
LoanTerm        0.0002633451 -0.0009981963  0.0025379660  1.130365e-03  -1.166064e-03
DTIRatio       -0.0046891917  0.0002054967  0.0011224209 -1.039252e-03   1.764627e-03
Default        -0.1677831649 -0.0991194845  0.0866591772 -3.416649e-02  -9.737383e-02
               NumCreditLines  InterestRate      LoanTerm      DTIRatio       Default
Age             -8.897680e-04 -1.127383e-03  0.0002633451 -0.0046891917 -0.1677831649
Income          -2.016410e-03 -2.303425e-03 -0.0009981963  0.0002054967 -0.0991194845
LoanAmount       7.944049e-04 -2.291119e-03  0.0025379660  0.0011224209  0.0866591772
CreditScore      1.604201e-05  4.361387e-04  0.0011303655 -0.0010392521 -0.0341664938
MonthsEmployed   1.267119e-03  9.557276e-05 -0.0011660636  0.0017646268 -0.0973738290
NumCreditLines   1.000000e+00 -2.966494e-04 -0.0002257909 -0.0005862297  0.0283297218
InterestRate    -2.966494e-04  1.000000e+00  0.0008920080  0.0005753188  0.1312730153
LoanTerm        -2.257909e-04  8.920080e-04  1.0000000000  0.0022730945  0.0005446977
DTIRatio        -5.862297e-04  5.753188e-04  0.0022730945  1.0000000000  0.0192359810
Default          2.832972e-02  1.312730e-01  0.0005446977  0.0192359810  1.0000000000
```



A correlation plot (or correlation matrix) provides a visual representation of the relationships between different variables in a dataset. Values close to 1 indicate a strong positive correlation, while values close to -1 indicate a strong negative correlation. For example, we can see that Default and Interest Rate are positively correlated (0.13), indicating that as interest rates increase, the likelihood of default increases. Conversely, Default and Age are negatively correlated (-0.17), suggesting that older individuals are less likely to default on loans. "Credit Score" is also negatively correlated with Loan Default, meaning higher credit scores are associated with lower default rates.

Additionally, all predictor variables are not highly correlated with each other, indicating no multicollinearity issues.

Statistical Tests

Chi-Square Testing

Are Defaults Related to Education level?

1. **State the Hypotheses:**

H0: Default (Yes or No) is independent of the Education level.

H1: Default (Yes or No) is dependent of the Education level.

	Non-Default	Default
Bachelor's	56577	7789
High School	55673	8230
Master's	56633	6908
PhD	56811	6726

2. **Find the critical value:**

The number of degrees of freedom is found by multiplying 1 less than a number of rows in the contingency table by 1 less than the number of columns. Our table consisted of 4 rows and 2 columns. We will find the number of degrees of freedom by multiplying 3 by 1. So, d.f = 3. With alpha = 0.05, we find the critical value is 7.815.

3. **Calculate the Chi-Square Test Value (Using R):**

Chi-Square Result:

```
Pearson's Chi-squared test

data:  observed
X-squared = 214.02, df = 3, p-value < 2.2e-16
```

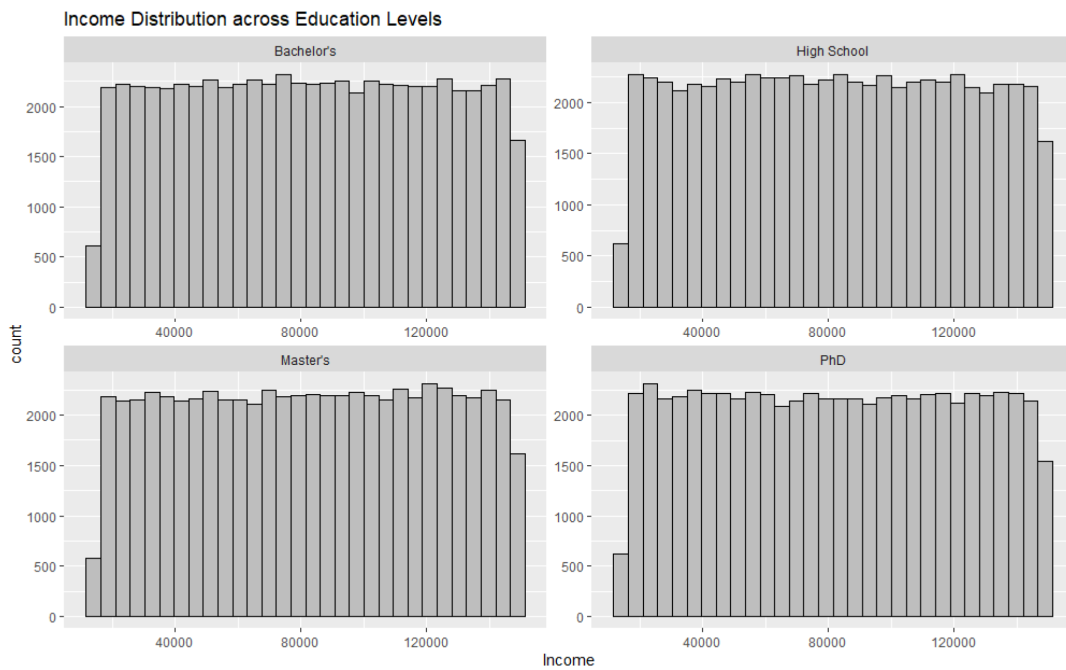
4. **Make the decision:**

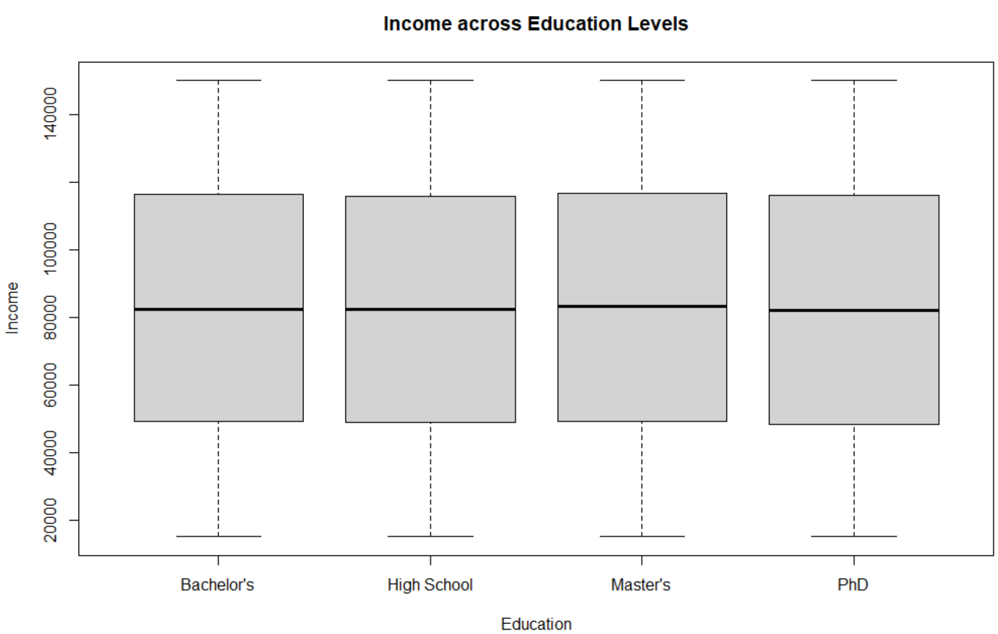
Since our chi-square test value of 214.02, which is far greater than the critical value of 7.815, and the p-value is extremely small, far less than the alpha value 0.05, we reject the null hypothesis.

5. **Conclusion:**

Therefore, we will conclude that the default rate is dependent on the level of education.

Kruskal-Wallis Test





From the above two graphs, we observed that the distribution of income across the four education levels appears to follow a uniform distribution, which contradicts the assumption of normal distribution required for an ANOVA test. Therefore, we opted to use the Kruskal-Wallis Test to determine if there are any significant differences in income among the four education levels.

1. **State the hypotheses:**

H0: There is no difference in income in the four education levels.

H1: There is a difference in income in the four education levels.

2. **Find the critical values:**

Use the chi-square table with d.f. = k − 1, where k = the number of groups. With $\alpha = 0.05$ and d.f. = 4 − 1 = 3, the critical value is 7.815.

3. **Calculate the test value (Using R):**

```
        Kruskal-Wallis rank sum test

data:  Income by Education
Kruskal-Wallis chi-squared = 8.6481, df = 3, p-value = 0.03435
```

4. **Make the decision:**

Since the test value of 8.6481 is larger than the critical value of 7.815, the decision is to reject the null hypothesis.

5. **Conclusion:**

There is enough evidence to reject the claim that there is no difference in income in the four education levels. Therefore, the observed differences are statistically significant at a significance level of $\alpha = 0.05$.

Data Analysis

Undersampling

Dealing with imbalanced data for the target variable “Default” requires careful consideration to ensure that the model doesn't become biased towards the majority class. Traditional machine learning algorithms trained on imbalanced data tend to be biased towards the majority class. They may achieve high accuracy by simply predicting the majority class most of the time. However, this can result in poor performance in correctly identifying the minority class (“Default” being “yes”), which is crucial for our project.

To address this bias and improve prediction accuracy, we implemented undersampling to balance the dataset. This involved reducing the dataset to 29,653 observations for each class, ensuring a more equitable representation of both classes and enhancing the model's ability to accurately identify the minority class.

Numeric Encoding

From the dataset structure, we can see that there is a mix of numeric and categorical predictors. Therefore, we performed numeric encoding by converting categorical variables into factors and further converting factor levels into integer representations, while keeping only the resulting integer and numeric columns in the data frame. We used label encoding (converting categorical labels into numerical labels) to represent categorical data numerically.

```
> # Numeric Encoding
> char_cols <- sapply(df, is.character)
> for (col in names(df)[char_cols]) {
+   df[[paste0(col, "_factor")]] <- factor(df[[col]], levels = unique(df[[col]]))
+   df[[paste0(col, "_integer")]] <- as.integer(df[[paste0(col, "_factor")]])
+ }
> df <- df[sapply(df, is.integer) | sapply(df, is.numeric)]
> str(df)
'data.frame':	255347 obs. of  17 variables:
 $ Age                 : int  56 69 46 32 60 25 38 56 36 40 ...
 $ Income              : int  85994 50432 84208 31713 20437 90298 111188 126802 42053 132784 ...
 $ LoanAmount          : int  50587 124440 129188 44799 9139 90448 177025 155511 92357 228510 ...
 $ CreditScore         : int  520 458 451 743 633 720 429 531 827 480 ...
 $ MonthsEmployed      : int  80 15 26 0 8 18 80 67 83 114 ...
 $ NumCreditLines      : int  4 1 3 3 4 2 1 4 1 4 ...
 $ InterestRate        : num  15.23 4.81 21.17 7.07 6.51 ...
 $ LoanTerm            : int  36 60 24 24 48 24 12 60 48 48 ...
 $ DTIRatio            : num  0.44 0.68 0.31 0.23 0.73 0.1 0.16 0.43 0.2 0.33 ...
 $ Default             : int  0 0 1 0 0 1 0 0 1 0 ...
 $ Education_integer   : int  1 2 2 3 1 3 1 4 1 3 ...
 $ EmploymentType_integer: int  1 1 2 1 2 2 2 1 3 3 ...
 $ MaritalStatus_integer : int  1 2 1 2 1 3 3 2 1 2 ...
 $ HasMortgage_integer   : int  1 2 1 2 2 1 1 2 1 1 ...
 $ HasDependents_integer : int  1 2 1 2 1 2 2 2 2 2 ...
 $ LoanPurpose_integer   : int  1 1 2 3 2 3 4 4 5 1 ...
 $ HasCoSigner_integer   : int  1 1 2 2 2 1 1 1 2 1 ...
```

Split the data into train and test sets

We splited the loan default dataset into training and testing sets for predictive modeling, with the training sets containing approximately 70% of the data and the testing sets containing the remaining data. Also, we prepared the input and output variables for modeling purposes.

```
# Split the data into training and testing sets
set.seed(123)
trainIndex <- sample(x = nrow(df), size = nrow(df) * 0.7)
train_data <- df[trainIndex, ]
test_data <- df[-trainIndex, ]

train_x <- model.matrix(Default ~ ., train_data) [, -1]
test_x <- model.matrix(Default ~ ., test_data) [, -1]

train_y <- train_data$Default
test_y <- test_data$Default
```

Logistic Regression Model

Model Fitting

We fitted a logistic regression model with a binomial family and a logit link function using the "glm" function, which is suitable for binary classification problems. In this logistic regression model, we modeled the "Default" variable as the response variable, while using all other variables 16 in the training dataset as predictors. The fitted model was saved as "ols_model" for further analysis and interpretation.

After examining the summary of "ols_model", we obtained various statistical measures. Notably, the p-values associated with certain predictor variables were observed to be relatively high (greater than 0.05), indicating a potentially insignificant impact on predicting the "Default" variable compared to other predictors. These variables include "LoanTerm" and "Education."

```
> # Fit logistic regression model
> ols_model <- glm(Default ~ ., family = binomial(link = "logit"), data = train_data)
> summary(ols_model)

Call:
glm(formula = Default ~ ., family = binomial(link = "logit"),
    data = train_data)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             4.379e-01  1.071e-01   4.087 4.37e-05 ***
Age                    -3.870e-02  7.534e-04 -51.369  < 2e-16 ***
Income                 -8.195e-06  2.734e-07 -29.971  < 2e-16 ***
LoanAmount              3.900e-06  1.542e-07  25.289  < 2e-16 ***
CreditScore            -7.309e-04  6.835e-05 -10.693  < 2e-16 ***
MonthsEmployed         -9.820e-03  3.163e-04 -31.049  < 2e-16 ***
NumCreditLines          7.638e-02  9.712e-03   7.865 3.69e-15 ***
InterestRate            7.017e-02  1.680e-03  41.779  < 2e-16 ***
LoanTerm               -2.784e-04  6.383e-04  -0.436    0.663
DTIRatio                2.555e-01  4.685e-02   5.455 4.91e-08 ***
Education_integer      -2.316e-02  9.704e-03  -2.387    0.017 *
EmploymentType_integer  7.246e-02  9.898e-03   7.321 2.46e-13 ***
MaritalStatus_integer  -7.594e-02  1.339e-02  -5.672 1.41e-08 ***
HasMortgage_integer     1.482e-01  2.166e-02   6.842 7.80e-12 ***
HasDependents_integer   2.384e-01  2.168e-02  10.995  < 2e-16 ***
LoanPurpose_integer    -3.622e-02  7.694e-03  -4.708 2.51e-06 ***
HasCoSigner_integer     2.655e-01  2.170e-02  12.238  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 57551  on 41513  degrees of freedom
Residual deviance: 49612  on 41497  degrees of freedom
AIC: 49646

Number of Fisher Scoring iterations: 3
```

Model Evaluation

Subsequently, we computed predicted probabilities for both the training set and the test set using "ols_model" and determined the corresponding classes based on a probability threshold of 0.5. If a predicted probability is greater than or equal to 0.5, it is labeled as “Yes”; otherwise, it is labeled as “No.” Following this, we analyzed the confusion matrix to assess the model's accuracy on both the training set and the test set.

```
> # Train set predictions
> probabilities.train1 <- predict(ols_model,newdata = train_data,type ='response')
> predicted.classes.train1 <- as.factor(ifelse(probabilities.train1 >=0.5,"1","0"))
>
> # Model accuracy
> train_data$Default <- as.factor(train_data$Default)
> confusionMatrix(data = predicted.classes.train1, reference = train_data$Default, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 13997  6511
         1  6766 14240

               Accuracy : 0.6802
                 95% CI : (0.6757, 0.6847)
    No Information Rate : 0.5001
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.3604

 Mcnemar's Test P-Value : 0.0275

            Sensitivity : 0.6862
            Specificity : 0.6741
         Pos Pred Value : 0.6779
         Neg Pred Value : 0.6825
             Prevalence : 0.4999
         Detection Rate : 0.3430
   Detection Prevalence : 0.5060
      Balanced Accuracy : 0.6802

       'Positive' Class : 1

> # Test set predictions
> probabilities.test1 <- predict(ols_model,newdata = test_data,type ='response')
> predicted.classes.test1 <- as.factor(ifelse(probabilities.test1 >=0.5,"1","0"))
>
> # Model accuracy
> test_data$Default <- as.factor(test_data$Default)
> confusionMatrix(data = predicted.classes.test1, reference = test_data$Default, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5956 2769
         1 2934 6133

               Accuracy : 0.6795
                 95% CI : (0.6725, 0.6863)
    No Information Rate : 0.5003
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.3589

 Mcnemar's Test P-Value : 0.02988

            Sensitivity : 0.6889
            Specificity : 0.6700
         Pos Pred Value : 0.6764
         Neg Pred Value : 0.6826
             Prevalence : 0.5003
         Detection Rate : 0.3447
   Detection Prevalence : 0.5096
      Balanced Accuracy : 0.6795

       'Positive' Class : 1
```

Through the confusion matrix, we can calculate many evaluation metrics for classification models, which help assess the model's performance across different categories and facilitate performance comparisons.

Predictions on the training set		Predictions on the test set	
TN 13997	FN 6511	TN 5956	FN 2769
FP 6766	TP 14240	FP 2834	TP 6133
• Accuracy: 0.6802 • Recall/ Sensitivity (TPR): 0.6862 • Specificity (TNR): 0.6741 • Precision: 0.6779		• Accuracy: 0.6795 • Recall/ Sensitivity (TPR): 0.6889 • Specificity (TNR): 0.6700 • Precision: 0.6764	

For the training set, the confusion matrix shows:

- True Negatives (TN): 13997 instances correctly predicted as non-default.
- True Positives (TP): 14240 instances correctly predicted as default.
- False Positives (FP): 6766 instances incorrectly predicted as default.
- False Negatives (FN): 6511 instances incorrectly predicted as non-default.

For the test set, the confusion matrix shows:

- True Negatives (TN): 5956 instances correctly predicted as non-default.
- True Positives (TP): 6133 instances correctly predicted as default.
- False Positives (FP): 2934 instances incorrectly predicted as default.
- False Negatives (FN): 2769 instances incorrectly predicted as non-default.

The model demonstrates reasonable accuracy and sensitivity, with accuracies around 68% and sensitivities of approximately 69% for both training set and test set. Additionally, the model's accuracy and other metrics, such as sensitivity, specificity and precision are relatively consistent between the training set and the test set. The differences in performance are not substantial, suggesting that the model does not suffer from overfitting.

Stepwise Selection

Model Fitting

The stepwise regression (or stepwise selection) involves iteratively adding and removing predictors from the predictive model to identify the subset of variables that produces the best-performing model, minimizing prediction error. (Hayes,2022)

We conducted stepwise selection and fitted a model, followed by generating a summary of the stepwise model. The summary reveals that this model comprises 15 variables, fewer than the “ols_model” model. Moreover, the AIC value is 49645, which is lower than the AIC of the “ols_model” (49646). This lower AIC indicates that the model achieves a better balance between goodness of fit and model complexity. In other words, a lower AIC signifies that the model effectively captures the data's variability while employing fewer parameters or features, thus avoiding overfitting.

```
> summary(step_model)

Call:
glm(formula = Default ~ Age + Income + LoanAmount + CreditScore + 
    MonthsEmployed + NumCreditLines + InterestRate + DTIRatio + 
    Education_integer + EmploymentType_integer + MaritalStatus_integer + 
    HasMortgage_integer + HasDependents_integer + LoanPurpose_integer + 
    HasCoSigner_integer, family = binomial(link = "logit"), data = train_data)

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)    
(Intercept)             4.277e-01  1.046e-01   4.090 4.31e-05 ***
Age                    -3.870e-02  7.533e-04 -51.368  < 2e-16 ***
Income                 -8.195e-06  2.734e-07 -29.972  < 2e-16 ***
LoanAmount              3.900e-06  1.542e-07  25.288  < 2e-16 ***
CreditScore            -7.309e-04  6.835e-05 -10.694  < 2e-16 ***
MonthsEmployed         -9.819e-03  3.163e-04 -31.048  < 2e-16 ***
NumCreditLines          7.640e-02  9.712e-03   7.867 3.64e-15 ***
InterestRate            7.017e-02  1.679e-03  41.778  < 2e-16 ***
DTIRatio                2.554e-01  4.685e-02   5.453 4.96e-08 ***
Education_integer      -2.318e-02  9.704e-03  -2.389   0.0169 *  
EmploymentType_integer  7.250e-02  9.898e-03   7.325 2.39e-13 ***
MaritalStatus_integer  -7.593e-02  1.339e-02  -5.671 1.42e-08 ***
HasMortgage_integer     1.482e-01  2.166e-02   6.843 7.76e-12 ***
HasDependents_integer   2.384e-01  2.168e-02  10.995  < 2e-16 ***
LoanPurpose_integer    -3.622e-02  7.694e-03  -4.707 2.51e-06 ***
HasCoSigner_integer     2.655e-01  2.170e-02  12.239  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 57551  on 41513  degrees of freedom
Residual deviance: 49613  on 41498  degrees of freedom
AIC: 49645

Number of Fisher Scoring iterations: 3
```

Model Evaluation

Next, we used the stepwise regression model to make predictions on both the train set and test set and analyzed the confusion matrix respectively.

```
> # Train set predictions
> probabilities.train2 <- predict(step_model,newdata = train_data,type ='response')
> predicted.classes.train2 <- as.factor(ifelse(probabilities.train2 >=0.5,"1","0"))
> 
> # Model accuracy
> confusionMatrix(data = predicted.classes.train2, reference = train_data$Default, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 14006  6523
         1  6757 14228
                                          
               Accuracy : 0.6801          
                 95% CI : (0.6756, 0.6846)
    No Information Rate : 0.5001          
    P-Value [Acc > NIR] : < 2e-16         
                                          
                  Kappa : 0.3602          
                                          
 Mcnemar's Test P-Value : 0.04319         
                                          
            Sensitivity : 0.6857          
            Specificity : 0.6746          
         Pos Pred Value : 0.6780          
         Neg Pred Value : 0.6823          
             Prevalence : 0.4999          
         Detection Rate : 0.3427          
   Detection Prevalence : 0.5055          
      Balanced Accuracy : 0.6801          
                                          
       'Positive' Class : 1               
```

```
> # Test set predictions
> probabilities.test2 <- predict(step_model,newdata = test_data,type ='response')
> predicted.classes.test2 <- as.factor(ifelse(probabilities.test2 >=0.5,"1","0"))
>
> # Model accuracy
> confusionMatrix(data = predicted.classes.test2, reference = test_data$Default, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5960 2766
         1 2930 6136

               Accuracy : 0.6799
                 95% CI : (0.6729, 0.6867)
    No Information Rate : 0.5003
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.3597

 Mcnemar's Test P-Value : 0.03079

            Sensitivity : 0.6893
            Specificity : 0.6704
         Pos Pred Value : 0.6768
         Neg Pred Value : 0.6830
             Prevalence : 0.5003
         Detection Rate : 0.3449
   Detection Prevalence : 0.5096
      Balanced Accuracy : 0.6798

       'Positive' Class : 1
```

Through the confusion matrix, we can calculate many evaluation metrics for classification models, which help assess the stepwise model's performance across different categories and facilitate performance comparisons.

Predictions on the training set		Predictions on the test set	
TN 14006	FN 6523	TN 5960	FN 2766
FP 6757	TP 14228	FP 2930	TP 6136
• Accuracy: 0.6801 • Recall/ Sensitivity (TPR): 0.6857 • Specificity (TNR): 0.6746 • Precision: 0.6780		• Accuracy: 0.6799 • Recall/ Sensitivity (TPR): 0.6893 • Specificity (TNR): 0.6704 • Precision: 0.6768	

For the training set, the confusion matrix shows:

- True Negatives (TN): 14006 instances correctly predicted as non-default.
- True Positives (TP): 14228 instances correctly predicted as default.
- False Positives (FP): 6757 instances incorrectly predicted as default.
- False Negatives (FN): 6523 instances incorrectly predicted as non-default.

For the test set, the confusion matrix shows:

- True Negatives (TN): 5960 instances correctly predicted as non-default.
- True Positives (TP): 2766 instances correctly predicted as default.

- False Positives (FP): 2930 instances incorrectly predicted as default.
- False Negatives (FN): 2766 instances incorrectly predicted as non-default.

The stepwise model has similar accuracies on the training set (0.6801) and the test set (0.6799), indicating that it generalizes well to new, unseen data. The precision values for both sets (training: 0.6780, test: 0.6893) are quite close to the recall rates (training: 0.6857, test: 0.6855), indicating that the model is balanced in its ability to identify both positive and negative instances accurately.

Regularization – LASSO

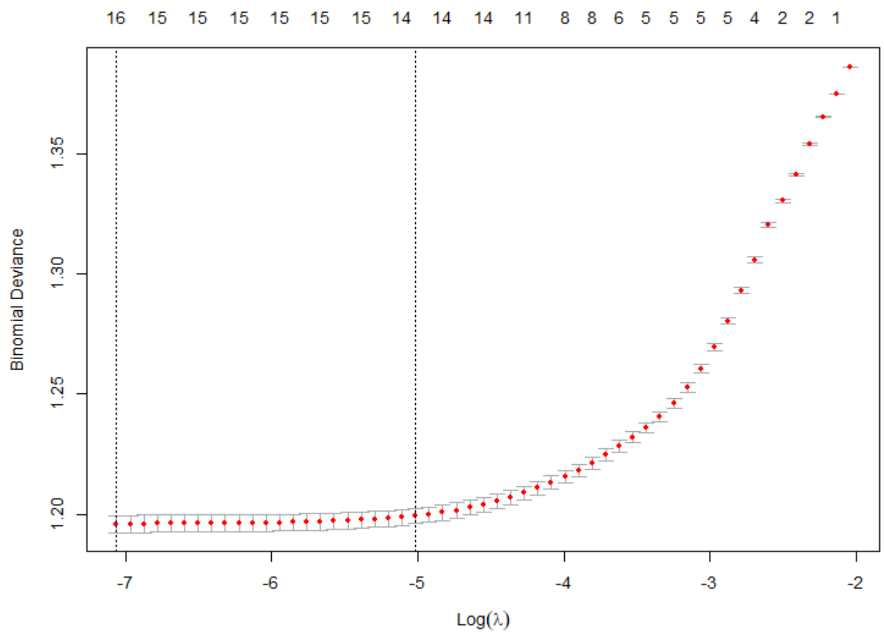
The Optimal Values of Lambda

LASSO regression balances model simplicity and accuracy by adding a penalty term to linear regression, encouraging sparse solutions with some coefficients forced to zero. This makes LASSO ideal for feature selection, automatically discarding irrelevant or redundant variables. (Kumar,2024)
In Lasso regression, lambda is a hyperparameter that controls the strength of regularization, with larger values of lambda leading to more regularization (shrinking coefficients towards zero). "λ min" is the lambda value that produces the lowest mean cross-validated error, while "λ1se" is the lambda value that leads to the most regularized model where the cross-validated error is within one standard error of the minimum.

We performed LASSO Regression using cross-validated LASSO regression with 10 folds and estimated the optimal values of lambda ("lambda_min_LASSO" and "lambda_1se_LASSO") using cross-validation. The outcome returned "lambda_min_LASSO" as 0.0008593558, while "lambda_1se_LASSO" was estimated to be 0.007302405.

```
> # Estimate lambda.min and lambda.1se
> lambda_min_lasso <- cv_lasso$lambda.min
> lambda_1se_lasso <- cv_lasso$lambda.1se
> # Compare lambda values
> lambda_min_lasso
[1] 0.0008573771
> lambda_1se_lasso
[1] 0.006638359
```

Additionally, the lambda values were compared and visualized using a plot as follows:



The y-axis typically shows the mean squared error (MSE), while the x-axis displays the logarithm of lambda (log(λ)). At the top of the plot, there's a display of the count of non-zero coefficients in the model for each lambda value.

The two vertical dotted lines on the plot represent log(λmin) and log(λ1se), respectively. In this scenario, the LASSO model with λmin has 16 non-zero coefficients, while the LASSO model with λ1se has 14 non-zero coefficients, making it the simplest yet effective model.

Model Fitting

Next, we fitted the LASSO regression model using lambda.1se, and extracted coefficients as follows:

```
> # Fit a Lasso regression model using lambda.1se
> lasso_model_1se <- glmnet(train_x, train_y, family = "binomial", alpha = 1, lambda = lambda_1se_lasso)

> coef(lasso_model_1se)
17 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)              5.450170e-01
Age                     -3.576524e-02
Income                  -7.258246e-06
LoanAmount               3.374684e-06
CreditScore             -5.154669e-04
MonthsEmployed          -8.668728e-03
NumCreditLines           4.711154e-02
InterestRate             6.360702e-02
LoanTerm                 .
DTIRatio                 1.169437e-01
Education_integer        .
EmploymentType_integer   4.205751e-02
MaritalStatus_integer   -3.620876e-02
HasMortgage_integer      8.267229e-02
HasDependents_integer    1.687596e-01
LoanPurpose_integer     -1.323508e-02
HasCoSigner_integer      1.967602e-01
```

The output indicates that the LASSO model with λ1se automatically removed two insignificant predictors by setting their coefficients to zero. These two insignificant predictors are: "LoanTerm" and "Education".

Model Evaluation

Afterwards, we used a LASSO regression model with lambda.1se to make predictions on both the train set and test set and analyzed the confusion matrix respectively.

```
> # Train set predictions
> probabilities_train_1se <- predict(lasso_model_1se, newx = train_x, type = "response")
> predicted_classes_train.1se <- ifelse(probabilities_train_1se >= 0.5, 1, 0)
>
> # Convert variables to factors for confusionMatrix
> train_y <- as.factor(train_y)
> predicted_classes_train.1se <- as.factor(predicted_classes_train.1se)
>
> # Model accuracy using confusionMatrix
> confusionMatrix(data = predicted_classes_train.1se, reference = train_y, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 13972  6543
         1  6791 14208

               Accuracy : 0.6788
                 95% CI : (0.6743, 0.6833)
    No Information Rate : 0.5001
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.3576

 Mcnemar's Test P-Value : 0.03243

            Sensitivity : 0.6847
            Specificity : 0.6729
         Pos Pred Value : 0.6766
         Neg Pred Value : 0.6811
             Prevalence : 0.4999
         Detection Rate : 0.3422
   Detection Prevalence : 0.5058
      Balanced Accuracy : 0.6788

       'Positive' Class : 1

> # Test set predictions
> probabilities_test_1se <- predict(lasso_model_1se, newx = test_x, type = "response")
> predicted_classes_test.1se <- ifelse(probabilities_test_1se >= 0.5, 1, 0)
>
> # Convert variables to factors for confusionMatrix
> test_y <- as.factor(test_y)
> predicted_classes_test.1se <- as.factor(predicted_classes_test.1se)
>
> # Model accuracy using confusionMatrix
> confusionMatrix(data = predicted_classes_test.1se, reference = test_y, positive = "1")
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 5930 2774
         1 2960 6128

               Accuracy : 0.6777
                 95% CI : (0.6708, 0.6846)
    No Information Rate : 0.5003
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.3554

 Mcnemar's Test P-Value : 0.01456

            Sensitivity : 0.6884
            Specificity : 0.6670
         Pos Pred Value : 0.6743
         Neg Pred Value : 0.6813
             Prevalence : 0.5003
         Detection Rate : 0.3444
   Detection Prevalence : 0.5108
      Balanced Accuracy : 0.6777

       'Positive' Class : 1
```

Through the confusion matrix, we can calculate many evaluation metrics for classification models, which help assess the stepwise model's performance across different categories and facilitate performance comparisons.

Predictions on the training set		Predictions on the test set	
TN 13972	FN 6543	TN 5930	FN 2774
FP 6791	TP 14208	FP 2960	TP 6128
• Accuracy: 0.6788 • Recall/ Sensitivity (TPR): 0.6847 • Specificity (TNR): 0.6729 • Precision: 0.6766		• Accuracy: 0.6777 • Recall/ Sensitivity (TPR): 0.6884 • Specificity (TNR): 0.6670 • Precision: 0.6743	

For the training set, the confusion matrix shows:

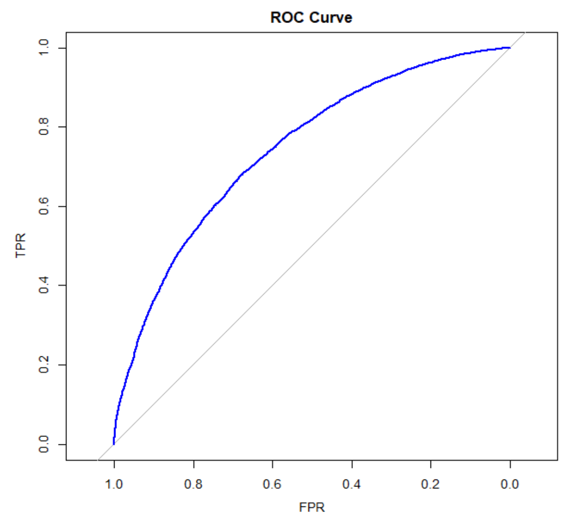
- True Negatives (TN): 13972 instances correctly predicted as non-default.
- True Positives (TP): 14208 instances correctly predicted as default.
- False Positives (FP): 6791 instances incorrectly predicted as default.
- False Negatives (FN): 6543 instances incorrectly predicted as non-default.

For the test set, the confusion matrix shows:

- True Negatives (TN): 5930 instances correctly predicted as non-default.
- True Positives (TP): 6128 instances correctly predicted as default.
- False Positives (FP): 2960 instances incorrectly predicted as default.
- False Negatives (FN): 2774 instances incorrectly predicted as non-default.

The accuracy is similar for both the training (0.6788) and test sets (0.6777), indicating that the model generalizes well and does not suffer significantly from overfitting. Moreover, the balance between recall and precision indicates that the model has a reasonable trade-off between identifying positive cases and minimizing false positives.

Also, we generated the Receiver Operating Characteristic (ROC) curve using the “roc” function from the “pROC” package. The ROC Curve provides insight into a model's ability to differentiate between classes.




```
> # Calculate the area under the ROC curve
> auc_value <- ROC1$auc
> auc_value
Area under the curve: 0.7428
```

The x-axis of the ROC curve represents the False Positive Rate (FPR), while the y-axis of the ROC curve represents the True Positive Rate (TPR). An ideal ROC curve would hug the top-left corner of the plot, indicating a high TPR (Sensitivity) and a low FPR (1 -Specificity). (Nahm,2022) Based on the returned ROC Curve, its proximity to the top-left corner (high TPR, low FPR) indicates a well-performing model with a greater capacity to accurately classify positive instances while reducing false positives.

The Area Under the Curve (AUC) is a widely used metric for assessing a model's performance in distinguishing between positive and negative classes. Ideally, a model with perfect discrimination would have an AUC value of 1. (Nahm,2022) With an AUC value of 0.7428, our binary classifier exhibits discriminatory power in distinguishing between default and non-default cases. However, there is still room for improvement.

Model Comparisons

In this project, we fitted three models, trained them on the training set, and made predictions on both the training set and the test set. The following table shows the performance of these three models:

	Logistic Regression Model		Stepwise Regression Model		LASSO Regression Model	
	Train Set	Test Set	Train Set	Test Set	Train Set	Test Set
Accuracy	0.6802	0.6795	0.6801	0.6799	0.6788	0.6777
Recall	0.6862	0.6889	0.6857	0.6893	0.6847	0.6884
Specificity	0.6741	0.6700	0.6746	0.6704	0.6729	0.6670
Precision	0.6779	0.6764	0.6780	0.6768	0.6766	0.6743

From the table, we can see that the Logistic Regression Model, Stepwise Regression Model, and LASSO Regression Model have similar performance in predicting loan defaults. However, the LASSO Regression Model stands out due to its simplicity and its effective prevention of overfitting. Therefore, the LASSO Regression Model is preferred.

Conclusion

In this project, Loan default prediction models were developed and compared, ultimately selecting a Lasso Regression Model because of simplicity and avoid overfitting. Fourteen predictors significantly impact loan default. These predictors are "Age," "Income," "LoanAmount," "CreditScore," "MonthsEmployed," "NumCreditLines," "InterestRate," "DTIRatio," "EmploymentType," "MaritalStatus," "HasMortgage," "HasDependents," "LoanPurpose," and "HasCoSigner."

In the loan default prediction, both False Negatives (FN) and False Positives (FP) carry negative consequences for the bank. FN occurs when the bank approves a loan that ends up defaulting, leading to financial losses and an increased risk of defaulting loans in the bank's portfolio. This can impact profitability and risk management significantly. FP happens when the bank rejects a loan application that would have been successfully repaid, resulting in missed revenue opportunities and potential customer dissatisfaction if deserving loan applications are denied.

However, in the context of loan default prediction, FN is generally considered worse for the bank. This is because the financial impact of approving loans that are not repaid (FN) can be more severe than missing out on potential revenue from rejected loans (FP), making it a more critical error to avoid.

To minimize loan defaults, we provide the following suggestions for the bank:

- **Robust Credit Evaluation:** Implement a thorough credit assessment process that considers multiple factors such as credit history, income stability, and debt-to-income ratio. Utilize advanced analytics and machine learning models to assess creditworthiness accurately.
- **Risk-Based Pricing:** Adjust interest rates and loan terms based on the borrower's credit risk profile to reflect the level of risk. Higher-risk borrowers may be offered loans with higher interest rates or shorter repayment terms to mitigate default risk.
- **Regular Monitoring and Reviews:** Implement regular monitoring and reviews of loan portfolios to identify early warning signs of potential default. Use data analytics to track payment patterns, credit utilization, and other indicators that may signal financial distress.

- **Loan Modifications and Assistance:** Provide options for loan modifications, refinancing, or temporary payment relief for borrowers facing financial difficulties. Proactive assistance can help prevent defaults and maintain customer relationships.

References

Hayes, A. (2022). Stepwise Regression: Definition, Uses, Example, and Limitations. *Investopedia.* https://www.investopedia.com/terms/s/stepwise-regression.asp

Kumar, D. (2024). A Complete Understanding of LASSO Regression. *Great Learning*. https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/

Nahm. F. S. (2022). Receiver Operating Characteristic Curve: Overview and Practical Use for Clinicians. *Korean Journal of Anesthesiology*, 75(1), 25–36. https://doi.org/10.4097/kja.21209

Yasser, M. (2022). Loan Default Dataset. *Kaggle.* https://www.kaggle.com/datasets/yasserh/loan-default-dataset