ADVANCED DATA MINING AND PREDICTIVE ANALYTICS

GROUP PROJECT

GROUP 11

STUDENT NAME	STUDENT ID	CONTRIBUTION
Ritika Kalyani	**811289983**	**Coding Lead:** **Wrote and implemented the whole code for the project.** **Conducted reviews from group members and documented the code.**
Akhil Seliveri	**811290035**	**Powerpoint Creator:** **Designed the PowerPoint Slides, ensured all content fom the report is accurately represented in the slides.**
Rajesh Katherasala	**811314095**	**Report Writer:** **Drafted Sections of the report, edited and formatted the final report to ensure clarity**
Navath Vamshi Krishna	**811303145**	**Presenter and Coordinator:** **Organized group calls and delivered the presentation of the project.**

Team collaboration: Participated in brainstorming sessions to define the project objectives, Collaborated on troubleshooting issues, Reviewed each other's work and provided constructive feedback throughout the project.

1.PROJECT GOAL:

This project's main goal is to use machine learning models to forecast the probability of a loan default as well as the scale of the resulting financial loss. This initiative goes beyond conventional binary classification techniques, which classify loans as either "good" or "bad" depending only on the possibility of default. By predicting not only whether a loan would default but also the possible damage to the financial institution in the case of a default, it seeks to offer a more comprehensive risk assessment. This method fits in better with contemporary financial risk management techniques, where estimating the amount of loss is just as crucial as anticipating whether a loan will default in the first place. Financial institutions can make better decisions by managing reserves, allocating capital, and optimizing their loan portfolios with the help of an accurate loss prediction.

In this regard, the initiative addresses a significant gap in the finance sector by advancing beyond basic risk models to more complex, data-driven methods that take into account the likelihood and seriousness of financial consequences. By more accurately anticipating loan defaults and projecting the possible losses for each loan, the model created in this study will assist financial institutions in simplifying their underwriting procedures. This data is crucial for developing more solid, flexible portfolio management techniques as well as for assessing the risk level of individual loans. It makes it possible to allocate resources more effectively, giving lower-risk loans priority and possibly avoiding or pricing high-risk loans differently. Also, by precisely evaluating and reducing risk, this strategy may improve lenders' capacity to satisfy legal obligations for capital reserves.

By establishing a connection between asset management strategies and conventional banking methods, the project will also benefit the larger financial ecosystem. Reducing economic capital consumption through the careful selection of creditworthy clients has frequently been the focus of traditional banking. Asset management, on the other hand, needs a more comprehensive perspective, where investors are worried about both the possible size of losses as well as the probability of default. The project seeks to maximize risk for investors and financial institutions by combining these two viewpoints, providing a two-pronged understanding of credit risk management. By offering more precise, useful forecasts that improve decision-making, increase financial stability, and foster innovation, this research ultimately seeks to show how machine learning can revolutionize financial practices.

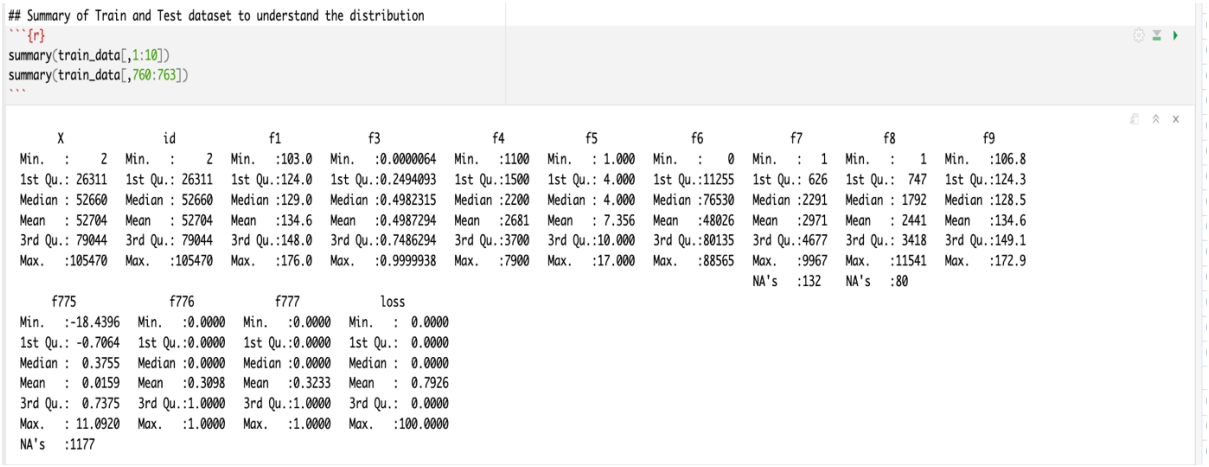
2. OVERVIEW OF DATA, INCLUDING DATA EXPLORATION ANALYSIS:

Data Exploration:

We loaded all of the necessary libraries for data exploration, data visualization, and model construction in the first stage. The data was loaded from our local computer using the "read.csv" function in the following step. After that, we began analyzing the data to gain a deeper understanding of its statistics and structure. Three main files—train_v3.csv, test_v3.csv, and test__no_lossv3.csv—make up the dataset supplied for the project. These files include a variety of attributes related to loan defaults and the losses that accompany them. The target variable, the "loss" caused when a loan fails, is one of 763 variables and 80,000 observations that constitute the

training data. 25,471 observations compose the test data, which has the same structure as the training data. After the model has been trained, the third dataset, test__no_lossv3.csv, can be used to generate predictions because it is comparable to the test data but does not contain the "loss" variable.

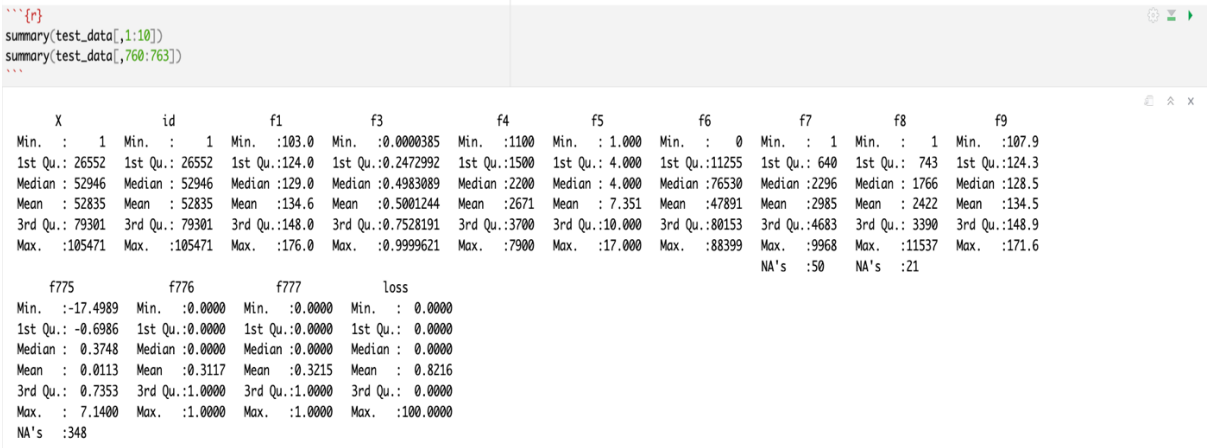
Using the str() method, we can look into the training data's structure and find a variety of numerical and categorical variables, such as IDs and features like f1, f3, f4, and others. The model's target variable, the "loss" variable, has a range of 0 to 100, with many values centered around 0, suggesting that there are many loans with no loss. A review of the first few columns shows that a number of features (such as f1 with 132 missing values and f5 with up to 1177 missing values) have missing data. This means that throughout the preprocessing stage, missing data must be handled appropriately.



```
## Summary of Train and Test dataset to understand the distribution
```{r}
summary(train_data[,1:10])
summary(train_data[,760:763])
```
```

```
       X                id              f1              f3                f4            f5              f6              f7            f8              f9       
 Min.   :    2   Min.   :    2   Min.   :103.0   Min.   :0.0000064   Min.   :1100   Min.   : 1.000   Min.   :    0   Min.   :  1   Min.   :    1   Min.   :106.8  
 1st Qu.: 26311   1st Qu.: 26311   1st Qu.:124.0   1st Qu.:0.2494093   1st Qu.:1500   1st Qu.: 4.000   1st Qu.:11255   1st Qu.: 626   1st Qu.:  747   1st Qu.:124.3  
 Median : 52660   Median : 52660   Median :129.0   Median :0.4982315   Median :2200   Median : 4.000   Median :76530   Median :2291   Median : 1792   Median :128.5  
 Mean   : 52704   Mean   : 52704   Mean   :134.6   Mean   :0.4987294   Mean   :2681   Mean   : 7.356   Mean   :48026   Mean   :2971   Mean   : 2441   Mean   :134.6  
 3rd Qu.: 79044   3rd Qu.: 79044   3rd Qu.:148.0   3rd Qu.:0.7486294   3rd Qu.:3700   3rd Qu.:10.000   3rd Qu.:80135   3rd Qu.:4677   3rd Qu.: 3418   3rd Qu.:149.1  
 Max.   :105470   Max.   :105470   Max.   :176.0   Max.   :0.9999938   Max.   :7900   Max.   :17.000   Max.   :88565   Max.   :9967   Max.   :11541   Max.   :172.9  
                                                                                                                    NA's   :132    NA's   :80                     

      f775              f776            f777             loss        
 Min.   :-18.4396   Min.   :0.0000   Min.   :0.0000   Min.   :  0.0000  
 1st Qu.: -0.7064   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:  0.0000  
 Median :  0.3755   Median :0.0000   Median :0.0000   Median :  0.0000  
 Mean   :  0.0159   Mean   :0.3098   Mean   :0.3233   Mean   :  0.7926  
 3rd Qu.:  0.7375   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:  0.0000  
 Max.   : 11.0920   Max.   :1.0000   Max.   :1.0000   Max.   :100.0000  
 NA's   :1177                                                          
```



```
```{r}
summary(test_data[,1:10])
summary(test_data[,760:763])
```
```

```
       X                id              f1              f3                f4            f5              f6              f7            f8              f9       
 Min.   :    1   Min.   :    1   Min.   :103.0   Min.   :0.0000385   Min.   :1100   Min.   : 1.000   Min.   :    0   Min.   :  1   Min.   :    1   Min.   :107.9  
 1st Qu.: 26552   1st Qu.: 26552   1st Qu.:124.0   1st Qu.:0.2472992   1st Qu.:1500   1st Qu.: 4.000   1st Qu.:11255   1st Qu.: 640   1st Qu.:  743   1st Qu.:124.3  
 Median : 52946   Median : 52946   Median :129.0   Median :0.4983089   Median :2200   Median : 4.000   Median :76530   Median :2296   Median : 1766   Median :128.5  
 Mean   : 52835   Mean   : 52835   Mean   :134.6   Mean   :0.5001244   Mean   :2671   Mean   : 7.351   Mean   :47891   Mean   :2985   Mean   : 2422   Mean   :134.5  
 3rd Qu.: 79301   3rd Qu.: 79301   3rd Qu.:148.0   3rd Qu.:0.7528191   3rd Qu.:3700   3rd Qu.:10.000   3rd Qu.:80153   3rd Qu.:4683   3rd Qu.: 3390   3rd Qu.:148.9  
 Max.   :105471   Max.   :105471   Max.   :176.0   Max.   :0.9999621   Max.   :7900   Max.   :17.000   Max.   :88399   Max.   :9968   Max.   :11537   Max.   :171.6  
                                                                                                                    NA's   :50     NA's   :21                     

      f775              f776            f777             loss        
 Min.   :-17.4989   Min.   :0.0000   Min.   :0.0000   Min.   :  0.0000  
 1st Qu.: -0.6986   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:  0.0000  
 Median :  0.3748   Median :0.0000   Median :0.0000   Median :  0.0000  
 Mean   :  0.0113   Mean   :0.3117   Mean   :0.3215   Mean   :  0.8216  
 3rd Qu.:  0.7353   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:  0.0000  
 Max.   :  7.1400   Max.   :1.0000   Max.   :1.0000   Max.   :100.0000  
 NA's   :348                                                           
```

In addition to the general summary statistics of the features, the exploration of the test data and the test data with no loss column (test_no_loss) shows similar distributions and patterns. Notably, the test dataset also contains a mix of numerical and categorical data, with some missing values. By analyzing the summary statistics of both the training and testing datasets, we can further identify patterns, such as correlations between features or any outliers that may skew the model's predictions. For instance, the features f775, f776, and f777 appear to have a substantial number of zeros, suggesting they may be binary indicators or placeholders for missing data. This exploration

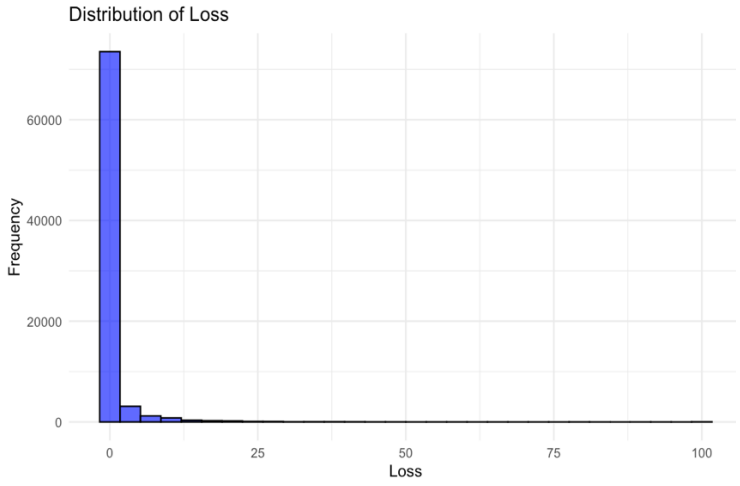
gives us crucial insight into how to handle missing values, outliers, and feature selection in the model-building process.

Data Visualization:

To better understand the distribution and characteristics of the "loss" variable in the dataset, several visualizations were created:

Histogram of Loss Values:

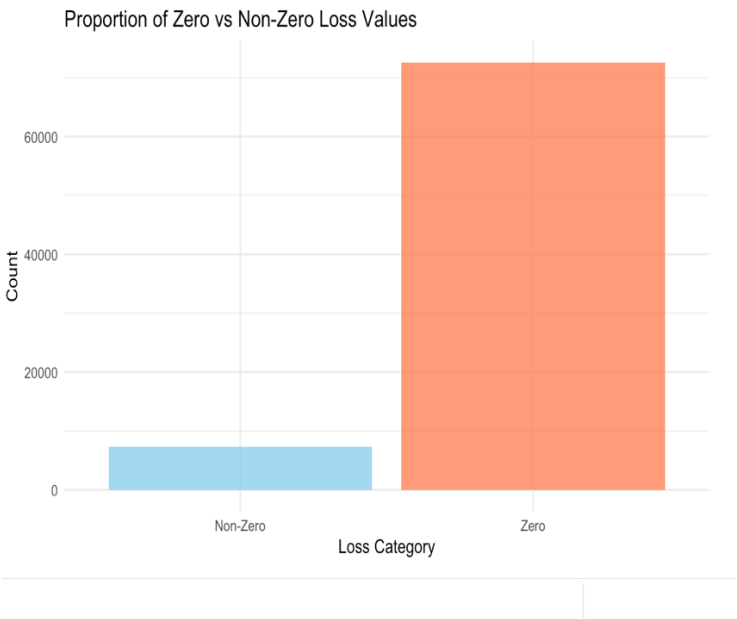
To see how the loss variable was distributed, a histogram was plotted. By revealing patterns, skewness, and any possible outliers, this figure helped to shed insight into the distribution of loss values. To provide a detailed picture of the data, the plot used 30 bins, with the bars coloured blue and outlined in black for improved visibility.



The distribution in the histogram is biased to the right. This indicates that the distribution decreases toward greater loss values and that there are more data points with lower loss values. The mode, or the center of the distribution, seems to lie between 10 and 15. The values in the distribution range from about 0 to 80, making it fairly dispersed.

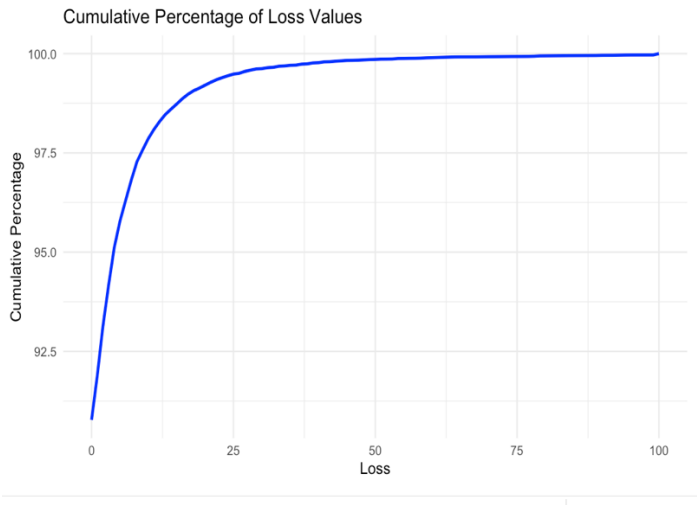
Proportion of Zero vs. Non-Zero Loss:

The number of records with zero loss as compared to non-zero loss was shown using a bar plot. This visualization gave a rapid overview of the frequency of zero losses in the dataset by highlighting the percentage of records in which no loss occurred. For zero losses, the bars are sky blue, whereas for non-zero losses, they are coral.
A larger number of observations in the dataset have a loss value of 0 compared to those with any non-zero loss values.



Cumulative Percentage of Loss data:

To display the cumulative percentage of loss data, a line plot was made. Understanding the accumulation of loss values throughout the sample was made easier by this cumulative distribution. It was very helpful for figuring out any notable thresholds in the data and seeing how greater loss levels behaved.



A considerable amount of the data has relatively low loss values, as indicated by the curve's sharp rise. After that, the curve flattens out, indicating that most of the data is inside a particular range of loss values. The curve then increases once more, although less rapidly, suggesting that fewer observations had larger loss values.

Gaining a basic knowledge of the loss distribution and spotting important data trends that would guide future analysis and model construction were made possible thanks in large part to these visualizations.

Data Handling and Cleaning:

During the data preprocessing phase, the following steps were performed to handle missing values across the training, testing, and test-no-loss datasets:

1.Missing Value Summary:

We began by checking for missing values in the three datasets. The results showed:

- Train data: 593,281 missing values.
- Test data: 191,194 missing values.
- Test data without loss: 191,194 missing values.

2.Missing Value Percentage:

A percentage of missing values per column was calculated to identify columns with a high proportion of missing values. Key statistics include:

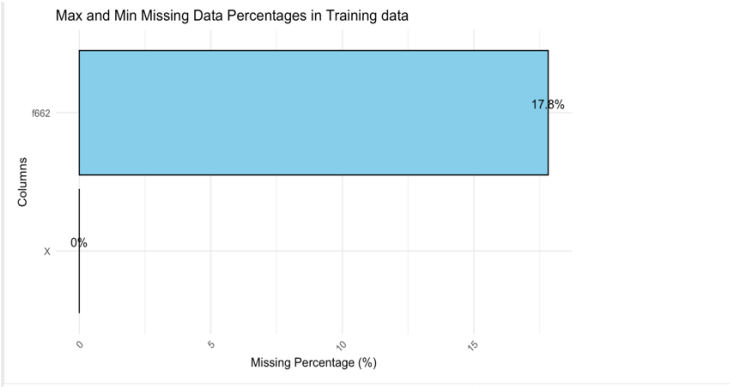
- The minimum missing percentage in all datasets was 0%.
- The maximum missing percentage was 17.83% in the train dataset and 17.95% in the test dataset.

From the training and test datasets, columns with more than 50% missing values were found and eliminated. This reduced the analysis's exposure to columns with a high percentage of missing data.

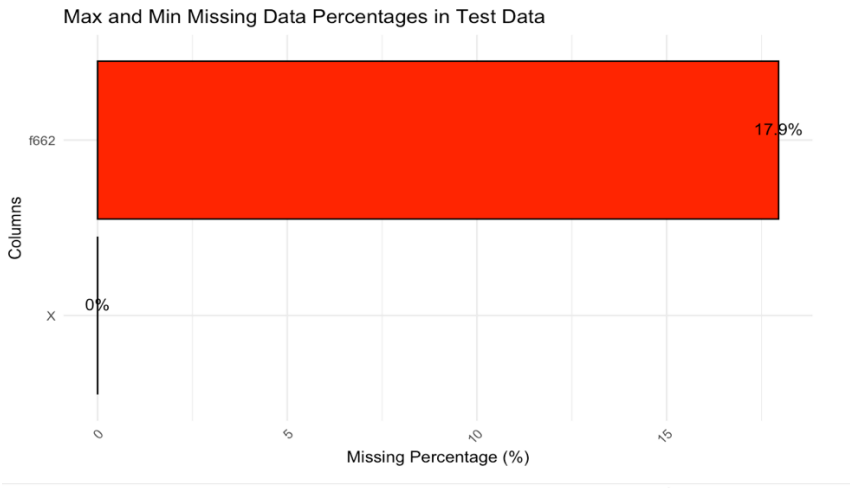
3.Visualizing Missing Data:

To show the columns in the training and test datasets with the highest and lowest percentages of missing data, bar charts were made. Decisions about how to manage missing data were guided by these plots, which showed the columns with the highest percentages of missing values.

Maximum and Minimum Missing Data Percentages in Training data plot:



Maximum and Minimum Missing Data Percentages in Test data plot:



4. Imputation of Missing Values:

The median of each column was used for the imputation of the remaining missing values. This method was chosen to prevent data misinterpretation that could arise from employing the mean, particularly when dealing with skewed variables. The train, test, and test-no-loss datasets were all subjected to the imputation function.

5. Imputation Verification to ensure Data Integrity:

Following imputation, a last check made sure that none of the datasets still contained any missing values. This stage made sure there were no further problems with data integrity and that the datasets were prepared for modeling.

```r
# Step 1: Define a function to handle missing values for numeric data
handle_missing_numeric <- function(data) {
  # Replace missing values with the median of each column
  data <- data %>%
    mutate(across(everything(), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))
  return(data)
}

# Step 2: Apply the function to train and test datasets
train_data <- handle_missing_numeric(train_data)
test_data <- handle_missing_numeric(test_data)
test_no_loss <- handle_missing_numeric(test_no_loss)

# Step 3: Verify that no missing values remain
cat("Remaining missing values in train data:", sum(is.na(train_data)), "\n")
cat("Remaining missing values in test data:", sum(is.na(test_data)), "\n")
cat("Remaining missing values in test_no_loss data:", sum(is.na(test_no_loss)), "\n")
```

```
Remaining missing values in train data: 0
Remaining missing values in test data: 0
Remaining missing values in test_no_loss data: 0
```

Data Preprocessing:

1.Near-Zero Variance Feature Removal:

Finding and eliminating near-zero variance (NZV) features from the datasets was the first step in the data pretreatment procedure. Because of their low to nonexistent variability, NZV characteristics give machine learning models very limited predictive potential. Columns with NZV were filtered out using the caret package's nearZeroVar() function. This process ensured a more effective and appropriate feature set for modeling by reducing the training dataset to 741 items (including the target variable). To ensure consistency, the test datasets were aligned to contain only the same characteristics as the training dataset after processing.

```r
# Removing zero variance features from datasets
# Step 1: Remove the target column before handling near-zero variance features
target <- train_data$loss  # Save the target variable
train_data <- train_data[, !colnames(train_data) %in% "loss"]  # Remove the target column

# Step 2: Identify near-zero variance features in the predictor features
nzv <- nearZeroVar(train_data, saveMetrics = TRUE)

# Step 3: Retain only features that are not near-zero variance
train_data <- train_data[, !nzv$nzv, drop = FALSE]

# Step 4: Add the target column back to the filtered train_data
train_data$loss <- target

# Step 5: Align test_data and test_no_loss with filtered train_data
test_data <- test_data[, colnames(test_data) %in% colnames(train_data), drop = FALSE]
test_no_loss <- test_no_loss[, colnames(test_no_loss) %in% colnames(train_data), drop = FALSE]

# Verify that all datasets now have consistent features
cat("Features in train_data (including target):", ncol(train_data), "\n")
cat("Features in test_data:", ncol(test_data), "\n")
cat("Features in test_no_loss:", ncol(test_no_loss), "\n")
```

```
Features in train_data (including target): 741
Features in test_data: 741
Features in test_no_loss: 740
```

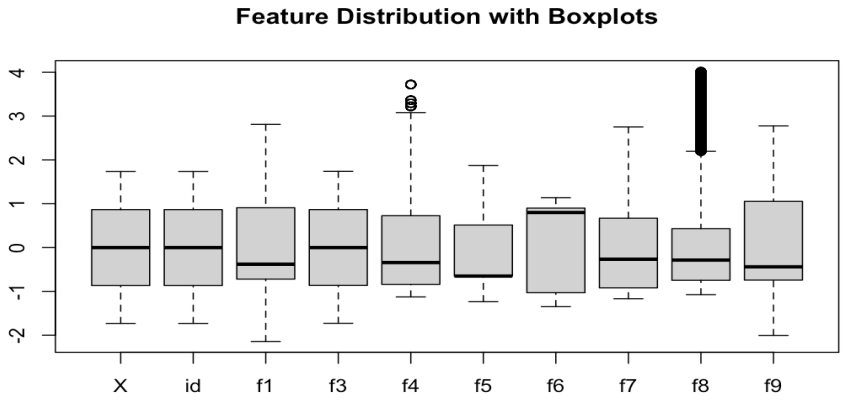
2. Data Scaling and Standardization:

The predictor variables were standardized in order to prepare the data for machine learning models that are sensitive to feature scales, such as neural networks and support vector machines. All numeric predictor variables were scaled and centered to have a mean of 0 and a standard deviation of 1 using the preProcess function from the caret package. This stage makes sure that features don't have their initial scale have an undue impact on how well the model performs. The test datasets were consistently subjected to the identical scaling transformation as was used on the training dataset. In order to preserve its original distribution, which is essential for regression modeling, the target variable (loss) was not scaled.

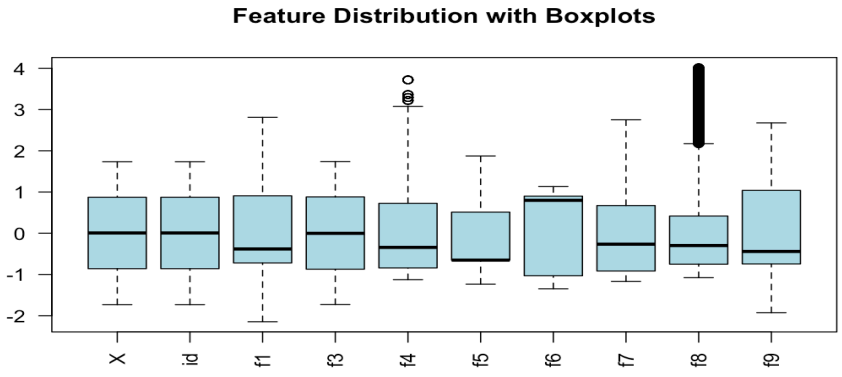
With 741 features in the training and test datasets (without the loss column in the latter), all datasets now have the same feature configurations. Now that the scaled features are prepared for model input, convergence rates and accuracy will increase, especially for algorithms that depend on normalized inputs. Standardizing the data and eliminating NZV features guarantees that the dataset is simplified, clear of unnecessary variables, and prepared for efficient modeling.

Additionally, we used boxplots to display the scaled data for the first ten features in both the test and training datasets. After scaling, these boxplots ensure consistency across the datasets by highlighting each feature's distribution, variability, and any outliers. This stage confirms that the features are similar across datasets and that the standardization procedure was done successfully.

For Train Data:



For Test data:



Feature Selection:

In machine learning, feature selection is an essential procedure that enhances model performance by determining which features are most important for prediction. It assists in lowering the dimensionality of the data, which optimizes the model, accelerates training, and lowers the possibility of overfitting. Feature selection improves interpretability and guarantees that the model concentrates on the most important variables by removing unnecessary or irrelevant

characteristics. This procedure is a crucial part of data preprocessing since it increases computational efficiency and can result in greater generalization on unseen data.

1.Correlation-Based Feature Selection:

This strategy reduced duplication and avoided multicollinearity by identifying and eliminating strongly correlated characteristics. In order to concentrate on the predictor factors, the goal variable (loss) was first eliminated. Features with a correlation higher than 0.9 were flagged after a correlation matrix was calculated. A smaller set of predictors was obtained by eliminating these strongly associated traits. The filtered dataset was then updated to include the target variable once more. The test datasets (test_data and test_no_loss) were aligned with the remaining characteristics of the filtered training dataset in order to guarantee consistency. Valid comparisons and model predictions were made possible by this alignment, which ensured that all datasets had the same predictor structure.

Following filtering, the test_no_loss dataset comprised 273 features (without the target variable) compared to 274 features in the training and test datasets (containing the target variable). Consistency of maintained features across datasets was verified by feature alignment testing.

2. Lasso Regression for Feature Selection:

A statistical method called Lasso regression was employed to choose features according to their predictive significance. The predictor variables and the target variable (loss) were separated in order to further analyze the filtered training data from the previous stage. The Lasso regression model was used, which shrinks the coefficients of less significant characteristics to zero by including L1 regularization. To find the ideal regularization parameter (lambda) that reduced prediction error, cross-validation was used. The optimal lambda value found was roughly 0.0142. The model's coefficients were extracted using this ideal value, and features with non-zero coefficients were deemed significant. 69 features were chosen as the most significant for predicting the target variable, excluding the intercept term.

This procedure improved the efficiency and interpretability of future modeling by reducing the dataset's complexity while preserving the features most important for loss prediction.
Lasso regression efficiently performs feature selection by using L1 regularization to reduce less significant feature coefficients to zero. It contributes to the reduction of dataset dimensionality and enhances model interpretability by finding and keeping only the most significant predictors.

Splitting the training dataset:

One of the most important steps in creating and evaluating machine learning models is splitting a dataset. It involves dividing the data into distinct subsets, usually a validation (or testing) set and a training set. The model is constructed using the training set, and its performance on unseen data is evaluated using the validation set. This method guarantees that the model generalizes effectively to new data and helps prevent overfitting. Depending on the size of the dataset, the split is usually done in an 80-20 or 70-30 ratio, and repeatability is ensured by using a random seed. The data has

been split into training and validation sets to evaluate the model's performance. Using an 80-20 split ensures that the training set contains 80% of the data, while the validation set holds the remaining 20%, providing a separate dataset to test the model's generalizability. The process was made reproducible by setting a random seed.

3.DETAILS OF YOUR MODELING STRATEGY (I.E., WHAT TECHNIQUE AND WHY):

In this project, several machine learning techniques were employed to predict loan defaults and the severity of losses, with each method chosen based on its strengths and suitability for the problem. Below is a detailed explanation of the four modeling approaches used:

1.Lasso Regression Model:

A regularization method called Lasso regression (Least Absolute Shrinkage and Selection Operator) penalizes big coefficients in linear regression models, which helps to lessen overfitting. By applying an L1 penalty, it essentially does feature selection by shrinking part of the coefficients to zero. In high-dimensional datasets with numerous predictors, Lasso's ability to recognize and preserve only the most pertinent characteristics enhances both model performance and interpretability. In order to avoid overfitting and increase prediction accuracy, Lasso was selected for this task since it can manage a high number of characteristics while streamlining the model by eliminating unnecessary variables.

Cross-validation was used to train the model and find the ideal lambda value, which regulates the level of regularization. The model's performance on the validation set was assessed using Mean Absolute Error (**MAE**), which produced a value of **1.442777**. The **ideal lambda** was determined to be **0.004956714**. Lasso is a good fit for this dataset because of its capacity to simplify the model and improve interpretability.

2.Ridge Regression Model:

In contrast to Lasso, Ridge regression does not completely eliminate any predictors but instead shrinks coefficients using an L2 penalty. Rather, it lessens the size of the associated feature coefficients, preventing overfitting and addressing multicollinearity without removing variables. When the majority of elements are anticipated to influence the prediction and the objective is to preserve all variables while lessening the impact of less significant ones, this method can be helpful. Ridge regression was used to supplement Lasso because it offers a good substitute when the model must manage any correlations between features and all features are thought to have some predictive power.Cross-validation was used to train Ridge, just like Lasso, and the **ideal lambda** was determined to be **1.11874**.

The validation set was used to assess the model's performance, and the **MAE** was determined to be **1.444355.** By placing a penalty on high coefficients, Ridge helps stabilize the solution and is especially useful when multicollinearity is present.

3.Random Forest Regression Model:

Several decision trees are constructed using the Random Forest ensemble learning technique, which aggregates their predictions to increase accuracy and decrease overfitting. Complex datasets benefit greatly from its exceptional ability to capture non-linear correlations and interactions between attributes. Random Forest keeps flexibility while lowering the chance of overfitting by choosing subsets of features at random for each split. This method was selected because it is resistant to noise and overfitting and can handle sizable datasets with high-dimensional features. Additionally, feature importance metrics are provided, which might aid in identifying the model's most significant predictors. Random Forest was tuned by adjusting the 'mtry' parameter, which determines the number of features considered at each split, using cross-validation. With 200 trees and an 'mtry' of 5, the model was trained on the dataset and evaluated based on MAE, yielding a **result of 1.597924**. The robustness of Random Forest in handling both categorical and continuous variables, along with its ability to manage missing data, made it a powerful model for this problem.

```r
# Enable parallel processing for faster processing
cl <- makeCluster(detectCores() - 2)
registerDoParallel(cl)

# Initialise cross-validation folds
train_control <- trainControl(method = "cv", number = 3, verboseIter = TRUE)

# Train a Random Forest model
rf_model <- train(loss ~ ., data = train_split, method = "rf",
                  trControl = train_control,
                  tuneGrid = expand.grid(mtry = c(5, 10, 15)),
                  ntree = 200)

# Stop parallel processing
stopCluster(cl)
registerDoSEQ()

# Predict on the validation set
rf_preds <- predict(rf_model, valid_split)

# Evaluate performance using Mean Absolute Error (MAE)
rf_mae <- mean(abs(valid_split$loss - rf_preds))
cat("Random Forest MAE on validation set:", rf_mae, "\n")
```

```
Aggregating results
Selecting tuning parameters
Fitting mtry = 5 on full training set
Random Forest MAE on validation set: 1.597924
```

4.Neural Network:

Layers of interconnected nodes create neural networks, which are capable of modeling extremely complicated, non-linear relationships in data. They work especially well when more complex patterns in the data are missed by conventional techniques like linear regression. Because neural networks can learn intricate mappings between inputs and outputs—particularly when the dataset is vast and contains non-linear interactions—they were selected for this project. They are flexible and adaptable, which makes them a strong competitor for complicated prediction tasks like severity estimate and loan default prediction, even if they are computationally demanding.
Cross-validation was used for training, and parameters like decay rate and network size were improved. With a decay rate of 0.1 and three layers, the model was assessed using MAE on the

validation set. Despite doing rather well, the Neural Network did not outperform the other models, as indicated by the resulting **MAE of 1.449678**. Although neural networks can be computationally costly and need to be carefully adjusted to prevent overfitting, they are especially helpful for capturing intricate patterns.

```
Aggregating results
Fitting final model on full training set
# weights:  214
initial  value 1205167.835667
iter  10 value 1201235.751237
iter  20 value 1200850.523437
iter  30 value 1200746.601154
final  value 1200735.200387
converged
Neural Network

64000 samples
   69 predictor

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 42667, 42666, 42667
Resampling results:

  RMSE      Rsquared      MAE
  4.324852  0.0002384399  1.430994

Tuning parameter 'size' was held constant at a value of 3
Tuning parameter 'decay' was
 held constant at a value of 0.1
Neural Network MAE on validation set: 1.449678
```

4.ESTIMATION OF MODEL'S PERFORMANCE:

Regression models are frequently evaluated using the Mean Absolute Error (MAE), especially when comparing model performance. It calculates the average size of the errors in a collection of forecasts without taking into account the direction they are going, that is, whether the prediction is higher or lower than the actual number. In this instance, the mean absolute difference (MAE) between the target variable loss's actual and anticipated values is computed by averaging the variations over all test samples. A lower MAE indicates better predictive accuracy. MAE gives a clear picture of how far the predictions deviate from the actual values.

In this project, the performance of several machine learning models—Random Forest, Lasso, Ridge, and Neural Network—is evaluated and contrasted using MAE. We may measure how well each model predicts the loss values by calculating the mean absolute error (MAE) for each model on the test set. The model with the lowest MAE is regarded as having the best prediction performance; a smaller MAE indicates that the model is more accurate. This metric provides a straightforward yet efficient way to compare the relative performance of different models and is particularly helpful when the objective is to reduce the degree of prediction error.

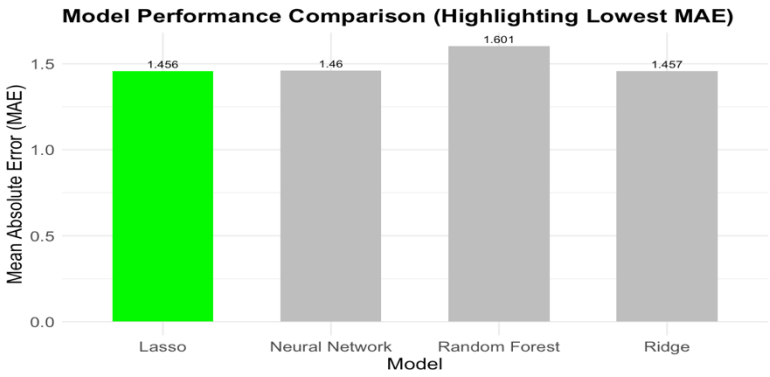
The code above evaluates the performance of four different regression models (Lasso, Ridge, Random Forest, and Neural Network) using the Mean Absolute Error (MAE) metric on the test dataset. First, the code prepares the test data by excluding the target variable loss from the feature set. Then, each model is used to predict the loss values on the test dataset, and the MAE is calculated by comparing the predicted values to the actual loss values.

The MAE values for each model are as follows:

Lasso MAE = **1.4557,**
Ridge MAE = **1.4566,**
Random Forest MAE = **1.6014,** and
Neural Network MAE = **1.4604.**

These values indicate that Lasso regression provides the best predictive performance, as it has the lowest MAE among all the models. The Lasso regression provides the best predictive performance, as indicated by its **lowest MAE (1.4557)** among the four models. The reason Lasso outperforms the others is due to its ability to perform both regularization and feature selection simultaneously. Lasso (Least Absolute Shrinkage and Selection Operator) applies an L1 penalty, which forces some of the model coefficients to shrink to zero, effectively eliminating irrelevant or less important features. This results in a simpler, more interpretable model with potentially fewer overfitting issues, as it reduces the complexity of the model by focusing only on the most important predictors. The reduction in complexity often leads to improved generalization to unseen data, as evidenced by its relatively lower MAE on the test set compared to Ridge regression, Random Forest, and Neural Network models, which may struggle with overfitting or require more complex parameter tuning.

To visually present these results, a bar plot is created using ggplot2. This plot highlights the model with the lowest MAE by coloring it green, while the other models are colored grey. The MAE values are displayed on top of each bar for easy comparison. The plot provides a clear, intuitive view of which model performs best in terms of prediction accuracy. The Lasso model, with the lowest MAE, is clearly highlighted in green, demonstrating its superior performance over the other models in this analysis.



Saving Prediction to Different file with only ID and "loss" column:

In the final step of the modeling process, the full training dataset, consisting of both the train_split and valid_split, is prepared and used to train the Lasso regression model on the entire set of predictor variables. The full dataset is then scaled using the column means and standard deviations derived from the training data. This ensures that the scaling of the predictors is consistent across the training and test datasets, helping the model generalize better. The Lasso model is then fitted using cross-validation to find the optimal regularization parameter (lambda), which in this case is **0.0024309.**

In the final step, after generating the predictions from the trained Lasso regression model, the predictions are saved to a CSV file for submission. This is done by first creating a new data frame that includes the id column from the test dataset (test_no_loss$id) and the predicted loss values (lasso_final_preds). The write.csv() function is then used to save this data frame to a file named "final_predictions.csv", without including row names in the output file. This step ensures that the predicted values are stored in a format that can easily be reviewed or submitted for further analysis or evaluation.

```r
# Read the predictions file
final_predictions <- read.csv("/Users/ritikakalyani/ADM Project/final_predictions.csv")
# Print the first few rows of the predictions
head(final_predictions,15)
```

Description: df [10 × 2]

	id <int>	loss <int>
1	7933	0
2	101860	0
3	62580	0
4	1760	0
5	48008	0
6	9308	0
7	27862	1
8	87760	0
9	53334	0
10	72303	0

1–10 of 10 rows

5.INSIGHTS AND CONCLUSIONS:

To forecast the probability of loan defaults and the magnitude of losses for financial investors, we used a variety of machine learning models throughout this research, such as Lasso, Ridge, Random Forest, and Neural Networks. The main emphasis was on performance optimization using cross-validation methods, hyperparameter tuning, and strict feature selection. The best model was Lasso regression, which had the lowest mean absolute error (MAE) and produced a cost-effective solution with fewer features without sacrificing predictive accuracy. This illustrates the value of Lasso's feature selection capabilities while reducing overfitting, particularly in high-dimensional datasets like the one utilized in this study. Other models, such as Random Forest and Ridge, performed similarly but had somewhat higher MAE, indicating that they might profit from additional tuning or improvements in hyperparameters.

An ideal model for forecasting loan defaults and loss severity was produced by the effective use of Lasso regression in conjunction with its feature selection capabilities. The methodology for the project also stressed the significance of data preprocessing, which includes handling missing values, scaling, and making sure that the training and testing datasets are properly aligned. The experiment demonstrated the importance of model selection in managing financial risk by carefully preparing the data, using appropriate feature selection techniques, and assessing model performance using MAE. Although there is always space for improvement, the models' results showed that using the right machine learning techniques—particularly regularization techniques like Lasso—can greatly help with better risk management and prediction making in financial situations.

The project emphasized the significance of model interpretability and validation in financial decision-making, in addition to the fundamental modeling process. The comparison with other models, such Ridge and Random Forest, helped highlight the trade-offs between accuracy and complexity, even though Lasso regression demonstrated great predictive performance. The capacity of each model to generalize to unknown data was clearly demonstrated by the validation using the Mean Absolute Error (MAE) measure. These revelations not only aided in improving the prediction procedure but also highlighted how important cross-validation and careful hyperparameter modifying are to producing solid, trustworthy results. The project made sure that the model's output matched expectations in the real world by recording the final forecasts to a CSV file and comparing their range to the initial loss values.