# Data Wizards (Group 4) Project 2

Di Chen Mai Castellano Tyler Kussee Spencer (Hutchison) Yang

## Introduction to dataset

In our pursuit of statistical inquiry, we have chosen to explore the Vehicle Loan Default dataset, comprising approximately 41 columns, with one designated as the response variable. Encompassing diverse information, the dataset delves into loan details, including date of birth, employment type, and credit score, alongside loan-related specifics such as disbursal details and loan-to-value ratios. The dataset presents challenges, notably in the form of odd date and time length columns, requiring standardization and transformation into comprehensible formats conducive to model development.

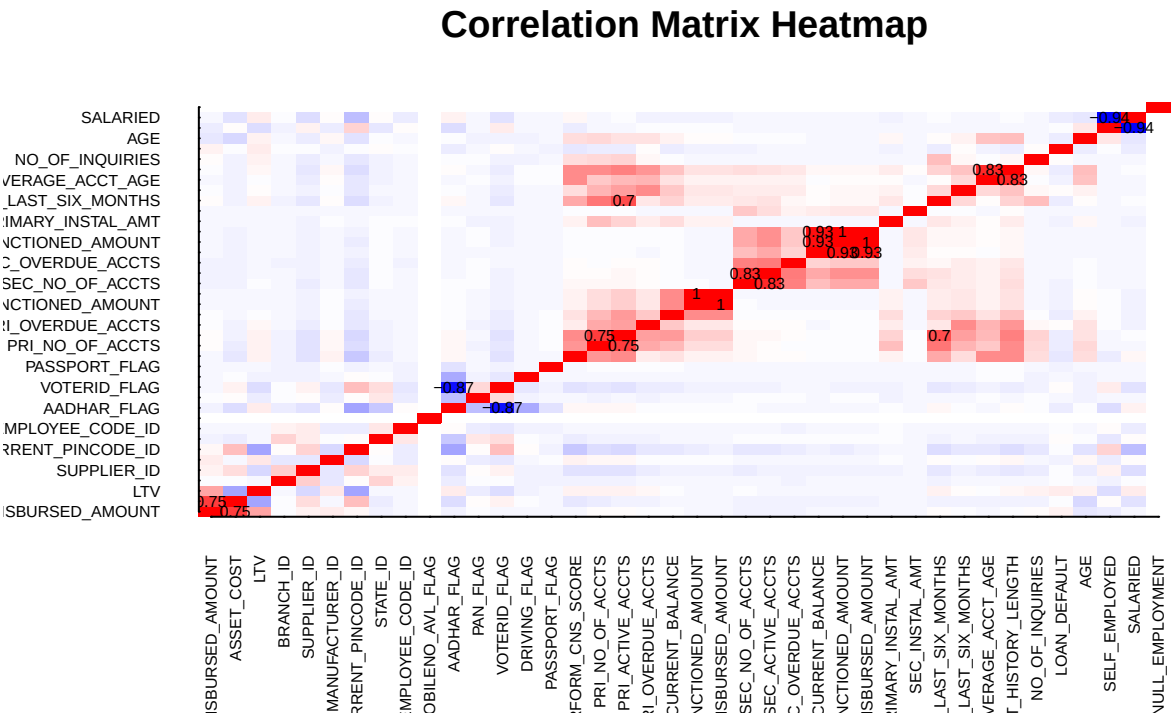
We want to discover the most influential explanatory variables driving loan default, and their impact within the dataset. We also want to find the optimal modeling approach for harnessing the training data, evaluating various methodologies to identify the most effective. Ultimately, our investigation extends to which among them best identifies the underlying dynamics of vehicle loan default prediction.

The data was pulled from the Vehicle Loan Default Prediction datasets available on Kaggle. As mentioned earlier, we have approximately 41 columns, with one of the columns designated as the response variable.

## Feature extraction and description

In the case of this dataset, we need to find the features that have the biggest impact on our "LOAN_DEFAULT" variable.

First, we want to check the correlations between different variables.

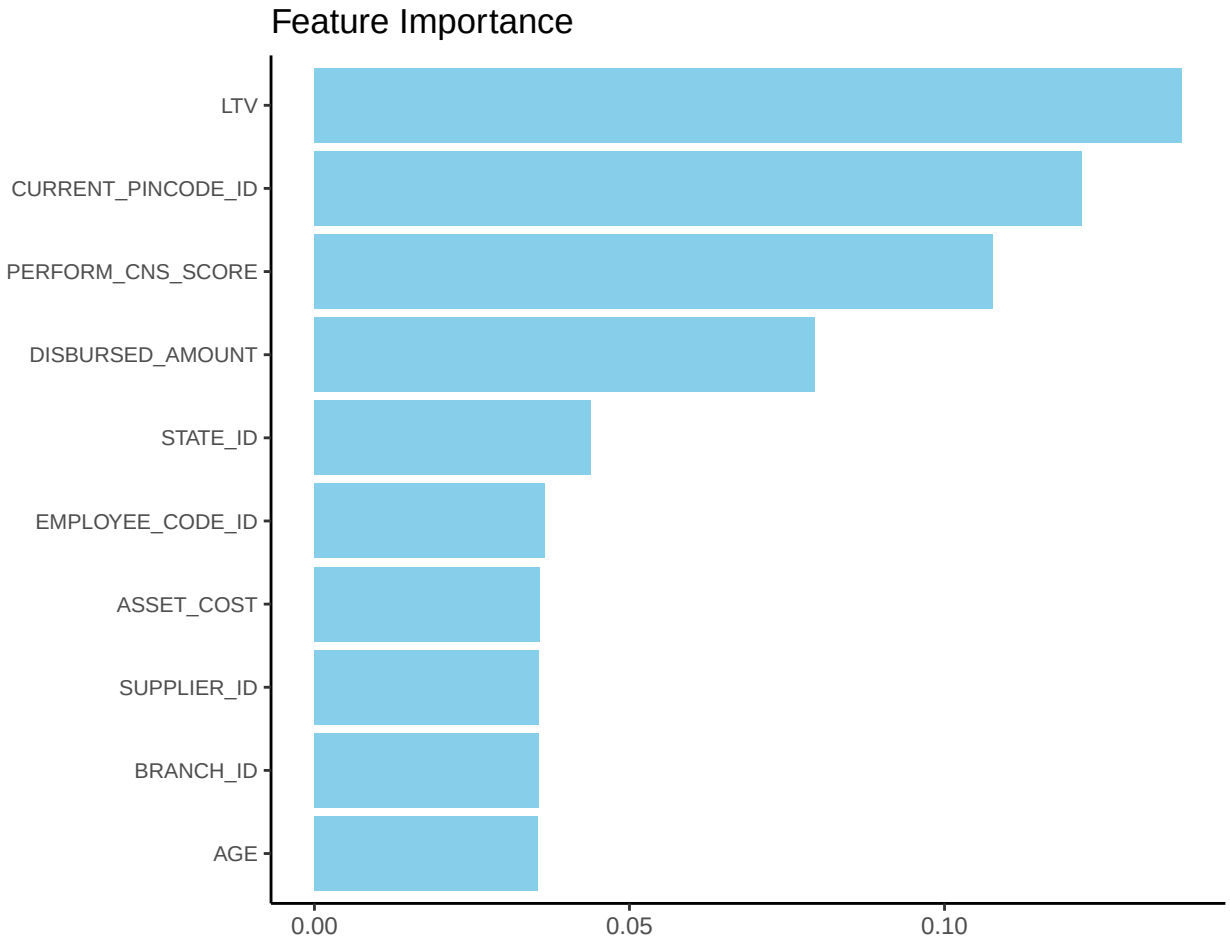


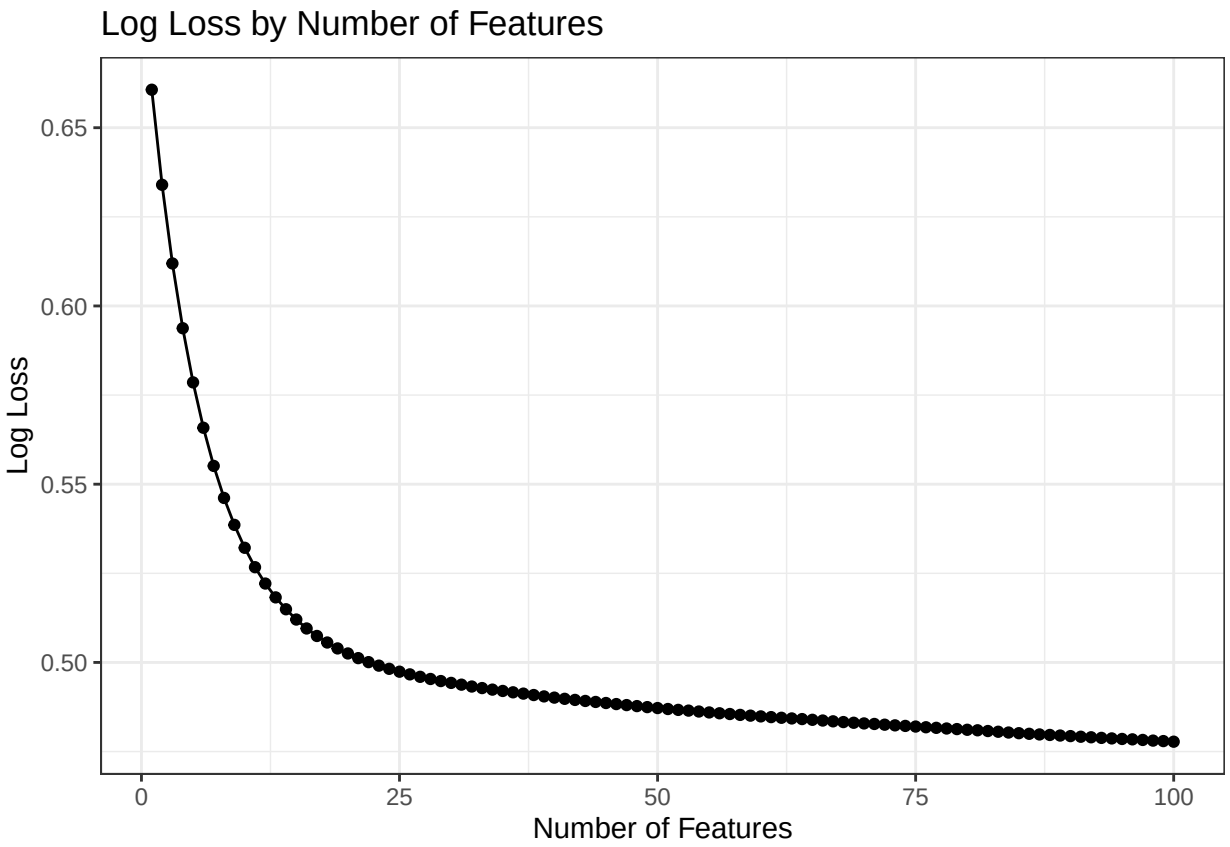
From the correlation matrix, we observe that darker shades of red signify robust positive correlations, while deeper shades of blue denote strong negative correlations.

The size and intensity of the squares within the heatmap correspond to the magnitude of the correlation coefficients between variables. Notably, variables such as PRI.DISBURSED.AMOUNT, LTV, SEC.NO.OF.ACCTS, SEC.ACTIVE.ACCTS, PRI.CURRENT.BALANCE, and AVERAGE.ACCT.AGE exhibit notable correlations with other independent variables. However, to derive conclusive insights, further in-depth analysis and interpretation are warranted.

In order to find the features that have the biggest impact, we will be using the XGBoost machine learning algorithm package to train our matrix and label vector from the training data, setting the parameters, and performing cross-validation to find the optimal number of rounds for training the model.

Once the parameters and optimal number of rounds were found, we trained the model and then proceeded to calculate the feature importance scores. From there, we select the top 10 features based on their importance to the model, along with plotting the scores to visualize the relative importance of each feature.






According to the feature importance scores and our graph, we found the following features to be the most important:

- LTV
- CURRENT_PINCODE_ID
- PERFORM_CNS_SCORE
- CTV
- DISBURSED_AMOUNT
- STATE_ID
- SUPPLIER_ID
- PRI_SANCTIONED_AMOUNT
- AGE
- EMPLOYEE_CODE_ID

## Building classifiers and results

The dataset, containing 10 explanatory variables as identified above and 1 response variable, was divided into training and testing sets, with an 80-20 split. A logistic regression model was then trained on the training data. Initial predictions on the training data were classified using a threshold of 0.5, and subsequent accuracy and confusion matrices were computed. To achieve a better balance, the threshold was adjusted to 0.4. Adjusted predictions were made accordingly, and the new accuracy, precision, and recall were calculated.

The initial logistic regression model had an accuracy of 78.15% but showed significant class imbalance, predicting nearly all instances as the majority class (class 0). Adjusting the threshold to 0.4 slightly reduced accuracy but improved the

detection of the minority class (class 1). Despite this, recall remained very low at 0.19%, indicating poor detection of minority class instances. Precision improved to 37.26%, meaning the model correctly predicted the minority class 37.26% of the time, but still missed most actual minority instances.

Subsequently, two XGBoost models were trained. The initial model used a maximum depth of 6, a learning rate (eta) of 1, and 100 boosting rounds, with the binary:logistic objective and logloss evaluation metric. Cross-validation with early stopping (5 rounds) was applied to determine the optimal number of boosting rounds, which was then used to train the final model.

Predictions were made on the training data with a threshold of 0.5. The accuracy of both models on the training data was calculated, resulting in 80.76% and 78.25%, respectively. The first model appears to have a better balance between precision of 69.5% and recall of 21.24%. However, a recall of 21.24% is relatively low. The next steps should include exploring further optimization to improve the model's performance. These optimization include adjusting the decision threshold, tuning the maximum depth of the trees, adjusting the learning rate, and creating learning curve plot.

## Appendix

```
##An if statement for checking if a package is installed

if (!require(tidycensus)) {
  install.packages("tidycensus")
}
if (!require(tidyverse)) {
  install.packages("tidyverse")
}
if (!require(dplyr)) {
  install.packages("dplyr")
}
if (!require(ggplot2)) {
  install.packages("ggplot2")
}
if (!require(caret)) {
  install.packages("caret")
}
if (!require(xgboost)) {
  install.packages("xgboost")
}
#if (!require(DiagrammeR)) {
  #install.packages("htmltools")
#}

#Load libraries
library(tidycensus)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(caret)
library(xgboost)
#library(DiagrammeR)

# Reading in the csv
train <- read.csv('train.csv')
train <- train %>% select(-"DISBURSAL_DATE")
train$DATE_OF_BIRTH <- as.Date(train$DATE_OF_BIRTH, format = "%d-%m-%Y")
train$AGE <- as.Date('01-01-2019', format = "%d-%m-%Y") - train$DATE_OF_BIRTH
train$AGE <- as.integer(floor(train$AGE / 365.25))
```

```r
train <- train %>% select(-"DATE_OF_BIRTH", -"PERFORM_CNS_SCORE_DESCRIPTION")

# Calculate strange data fields
train$acctyr <- as.numeric(gsub("yrs.*", "", train$AVERAGE_ACCT_AGE))
train$acctmo <- as.numeric(gsub(".*yrs|mon", "", train$AVERAGE_ACCT_AGE))
train$crdtyr <- as.numeric(gsub("yrs.*", "", train$CREDIT_HISTORY_LENGTH))
train$crdtmo <- as.numeric(gsub(".*yrs|mon", "", train$CREDIT_HISTORY_LENGTH))

# Replace strange data fields
train$AVERAGE_ACCT_AGE <- round(train$acctyr + train$acctmo / 12, 2)
train$CREDIT_HISTORY_LENGTH <- round(train$crdtyr + train$crdtmo / 12, 2)
# Remove calc fields
train <- train %>% select(-acctyr, -acctmo, -crdtyr, -crdtmo)
# Create emloyment dummy fields
train$SELF_EMPLOYED <- ifelse(train$EMPLOYMENT_TYPE == "Self employed", 1, 0)
train$SALARIED <- ifelse(train$EMPLOYMENT_TYPE == "Salaried", 1, 0)
train$NULL_EMPLOYMENT <- ifelse(is.na(train$EMPLOYMENT_TYPE), 1, 0)
# Remove employment_type
train <- train %>% select(-EMPLOYMENT_TYPE)
# Pull CNS score letter grade, removed to use XGBoost for now since it only takes integer/numbers
#train$PERFORM_CNS_SCORE_DESCRIPTION <- as.factor(substr(train$PERFORM_CNS_SCORE_DESCRIPTION,

# Remove rows with any null values
train <- train[complete.cases(train), ]

# Remove duplicate rows
train <- train[!duplicated(train), ]

# converting Loan Default to a factor for binary classification
tempTrain <- train
tempTrain$LOAN_DEFAULT <- as.factor(train$LOAN_DEFAULT)

colnames(tempTrain)

# Convert all columns to numeric
tempTrain[] <- lapply(tempTrain, as.numeric)

# Check the structure of the 'train' dataset to verify numeric conversion
str(tempTrain)

# Check for NaN or infinite values in the correlation matrix and replace with 0
correlation_matrix <- cor(tempTrain)
correlation_matrix <- as.matrix(correlation_matrix)
correlation_matrix[is.nan(correlation_matrix) | is.infinite(correlation_matrix)] <- 0

# Plot correlation matrix directly without clustering with variable labels
image(1:nrow(correlation_matrix), 1:ncol(correlation_matrix), correlation_matrix,
      main = "Correlation Matrix Heatmap",
      xlab = "",
      ylab = "",
      col = colorRampPalette(c("blue", "white", "red"))(100),
      axes = FALSE)

# Add labels to the axes with smaller font size and remove numbers
axis(1, at = 1:ncol(correlation_matrix), labels = colnames(correlation_matrix), las = 2, cex.axis = 0.5, tcl
axis(2, at = 1:nrow(correlation_matrix), labels = rownames(correlation_matrix), las = 2, cex.axis = 0.5, tcl
```

```r
featureMatrix <- as.matrix(train[, -which(names(train) == "LOAN_DEFAULT")])
labelVector <- as.numeric(as.character(train$LOAN_DEFAULT))

# Set the parameters
params <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  max_depth = 6,
  eta = 0.1,
  nthread = 2,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Cross-validation for rounds
cvResults <- xgb.cv(
  params = params,
  data = featureMatrix,
  label = labelVector,
  nfold = 5,
  nrounds = 100,
  early_stopping_rounds = 10,
  verbose = FALSE
)

# Train the XGBoost model
xgbModel <- xgboost(
  params = params,
  data = featureMatrix,
  label = labelVector,
  nrounds = cvResults$best_iteration
)

# Select the top features
importanceMatrix <- xgb.importance(model = xgbModel)
topFeatures <- importanceMatrix$Feature[1:10]
print(topFeatures)

#Modelling the features importance
importanceMatrix <- xgb.importance(model = xgbModel)
xgb.plot.importance(importanceMatrix)

# Sort the data by importance in descending order
importance_data <- importanceMatrix[order(importanceMatrix$Importance, decreasing = TRUE), ]

# Create the bar plot
ggplot(importance_data, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  labs(title = "Feature Importance") +
  theme_classic() +
  theme(axis.text.y = element_text(size = 8), axis.title.x=element_blank(), axis.title.y=element_blank()) +
  coord_flip()

# Subset the dataset with the selected features
filteredData <- train[, c(topFeatures, "LOAN_DEFAULT")]

# Training/Testing split
```

```r
set.seed(1)
n <- dim(filteredData)[1]
train_indices <- sample(1:n, size = round(0.8 * n))  # 80% for training
test_indices <- setdiff(1:n, train_indices)  # Remaining 20% for testing

train.data <- (filteredData[train_indices, 1:11])
test.data <- (filteredData[test_indices, 1:11])

train.labels <- as.numeric(filteredData$LOAN_DEFAULT[train_indices])
test.labels <- as.numeric(filteredData$LOAN_DEFAULT[test_indices])

#Verify for accuracy
head(train.data)

#Fit the Logistic regression on train data
logistic_model <- glm(LOAN_DEFAULT ~ ., data = train.data, family = binomial)

# Predictions on train data
y_pred_train <- predict(logistic_model, newdata = train.data, type = "response")
y_pred_class_train <- ifelse(y_pred_train > 0.5, 1, 0)

# Model evaluation on train data
train_accuracy <- mean(y_pred_class_train == train.labels)
train_confusion_matrix <- table(train.labels, y_pred_class_train, dnn = c("Actual", "Predicted"))

# Print accuracy and confusion matrix for train data
print(paste("Train Accuracy:", train_accuracy))
train_confusion_matrix

# Adjust the threshold to see if it improves the balance
threshold <- 0.4
y_pred_class_train_adjusted <- ifelse(y_pred_train > threshold, 1, 0)
train_confusion_matrix_adjusted <- table(train.labels, y_pred_class_train_adjusted, dnn = c("Actual", "Predi

adjusted_train_accuracy <- mean(y_pred_class_train_adjusted == train.labels)
precision <- train_confusion_matrix_adjusted[2, 2] / sum(train_confusion_matrix_adjusted[, 2])
recall <- train_confusion_matrix_adjusted[2, 2] / sum(train_confusion_matrix_adjusted[2, ])

# Print adjusted accuracy, precision, and recall for train data
print(paste("Adjusted Train Accuracy:", adjusted_train_accuracy))
train_confusion_matrix_adjusted
print(paste("Precision:", precision))
print(paste("Recall:", recall))

# XGBoost Classifier

#Matrix for explanatoty variable
train.data <- as.matrix(train.data[ , 1:10])
test.data <- as.matrix(test.data[ , 1:10])

str(train.data)
str(test.data)

xgb.loan = xgboost(data = train.data, label = train.labels,
                   max.depth = 6, eta = 1, nrounds = 100,
                   objective = "binary:logistic", eval_metric = "logloss")
```

```r
# Early Stopping using cross-validation
xgb.loan.cv = xgb.cv(data = train.data, label = train.labels,
                     max.depth = 6, eta = 1, nrounds = 100,
                     nfold = 5, early_stopping_rounds = 5,
                     objective = "binary:logistic", eval_metric = "logloss")

sel_rounds = xgb.loan.cv$best_iteration

xgb.loan2 = xgboost(data = train.data, label = train.labels,
                    max.depth = 6, eta = 1, nrounds = sel_rounds,
                    objective = "binary:logistic", eval_metric = "logloss")

# Fitted response labels
pred1 = predict(xgb.loan, train.data)
pred.xgb.loan = ifelse(pred1 > 0.5, 2, 1)

pred2 = predict(xgb.loan2, train.data)
pred.xgb.loan2 = ifelse(pred2 > 0.5, 2, 1)

# Plotting the first two trees in the boosted classifier
xgb.plot.tree(model = xgb.loan, trees = 1:2)

# Misclassification errors in the training data
train_conf_matrix1 <- table(filteredData$LOAN_DEFAULT[train_indices], pred.xgb.loan)
train_conf_matrix2 <- table(filteredData$LOAN_DEFAULT[train_indices], pred.xgb.loan2)

#Accuracy
train_accuracy1 <- (train_conf_matrix1[1,1] + train_conf_matrix1[2,2]) / sum(train_conf_matrix1)
precision1 <- train_conf_matrix1[2, 2] / sum(train_conf_matrix1[, 2])
recall1 <- train_conf_matrix1[2, 2] / sum(train_conf_matrix1[2, ])

train_accuracy1
precision1
recall1

train_accuracy2 <- (train_conf_matrix2[1,1] + train_conf_matrix2[2,2]) / sum(train_conf_matrix2)
precision2 <- train_conf_matrix2[2, 2] / sum(train_conf_matrix2[, 2])
recall2 <- train_conf_matrix2[2, 2] / sum(train_conf_matrix2[2, ])

train_accuracy2
precision2
recall2
```