BT2103 AY22/23 Sem 2

2023-04-11

```r
# Check data types
cat("Data types of variables:\n")
```

```
## Data types of variables:
```

```r
print(sapply(credit, class))
```

```
##                         ID                  LIMIT_BAL
##                  "integer"                  "integer"
##                        SEX                  EDUCATION
##                  "integer"                  "integer"
##                   MARRIAGE                        AGE
##                  "integer"                  "integer"
##                      PAY_0                      PAY_2
##                  "integer"                  "integer"
##                      PAY_3                      PAY_4
##                  "integer"                  "integer"
##                      PAY_5                      PAY_6
##                  "integer"                  "integer"
##                  BILL_AMT1                  BILL_AMT2
##                  "integer"                  "integer"
##                  BILL_AMT3                  BILL_AMT4
##                  "integer"                  "integer"
##                  BILL_AMT5                  BILL_AMT6
##                  "integer"                  "integer"
##                   PAY_AMT1                   PAY_AMT2
##                  "integer"                  "integer"
##                   PAY_AMT3                   PAY_AMT4
##                  "integer"                  "integer"
##                   PAY_AMT5                   PAY_AMT6
##                  "integer"                  "integer"
## default.payment.next.month
##                  "integer"
```

```r
# Check column names
cat("Column names in the dataset:\n")
```

```
## Column names in the dataset:
```

```r
print(colnames(credit))
```

```
##  [1] "ID"                         "LIMIT_BAL"
##  [3] "SEX"                        "EDUCATION"
##  [5] "MARRIAGE"                   "AGE"
##  [7] "PAY_0"                      "PAY_2"
##  [9] "PAY_3"                      "PAY_4"
## [11] "PAY_5"                      "PAY_6"
## [13] "BILL_AMT1"                  "BILL_AMT2"
## [15] "BILL_AMT3"                  "BILL_AMT4"
## [17] "BILL_AMT5"                  "BILL_AMT6"
## [19] "PAY_AMT1"                   "PAY_AMT2"
## [21] "PAY_AMT3"                   "PAY_AMT4"
## [23] "PAY_AMT5"                   "PAY_AMT6"
## [25] "default.payment.next.month"
```

```r
# Change column names
colnames(credit)[colnames(credit) == "PAY_0"] <- "PAY_1"
colnames(credit)[colnames(credit) == "default.payment.next.month"] <- "DEFAULT"

# Check updated column names
cat("Updated column names in the dataset:\n")
```

```
## Updated column names in the dataset:
```

```r
print(colnames(credit))
```

```
##  [1] "ID"        "LIMIT_BAL" "SEX"       "EDUCATION" "MARRIAGE"  "AGE"
##  [7] "PAY_1"     "PAY_2"     "PAY_3"     "PAY_4"     "PAY_5"     "PAY_6"
## [13] "BILL_AMT1" "BILL_AMT2" "BILL_AMT3" "BILL_AMT4" "BILL_AMT5" "BILL_AMT6"
## [19] "PAY_AMT1"  "PAY_AMT2"  "PAY_AMT3"  "PAY_AMT4"  "PAY_AMT5"  "PAY_AMT6"
## [25] "DEFAULT"
```

```r
# Check for missing values
missing_values <- colSums(is.na(credit))
cat("Missing values in the dataset:\n")
```

```
## Missing values in the dataset:
```

```r
print(missing_values)
```

```
##        ID LIMIT_BAL       SEX EDUCATION  MARRIAGE       AGE     PAY_1     PAY_2
##         0         0         0         0         0         0         0         0
##     PAY_3     PAY_4     PAY_5     PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4
##         0         0         0         0         0         0         0         0
## BILL_AMT5 BILL_AMT6  PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6
##         0         0         0         0         0         0         0         0
##   DEFAULT
##         0
```

```r
# Check for duplicate records
credit.dup <- credit %>%
  select(-ID) # remove ID column since it is not needed to check for duplicates

duplicates <- duplicated(credit.dup) # creates a logical vector indicating duplicated rows
cat("Number of duplicate rows:", sum(duplicates), "\n") # prints the number of duplicated rows
```

```
## Number of duplicate rows: 35
```

```r
# Show the duplicated rows
duplicated_rows <- credit[duplicates,]
cat("Duplicated rows:\n")
```

```
## Duplicated rows:
```

```r
table(duplicates)
```

```
## duplicates
## FALSE  TRUE
## 29965    35
```

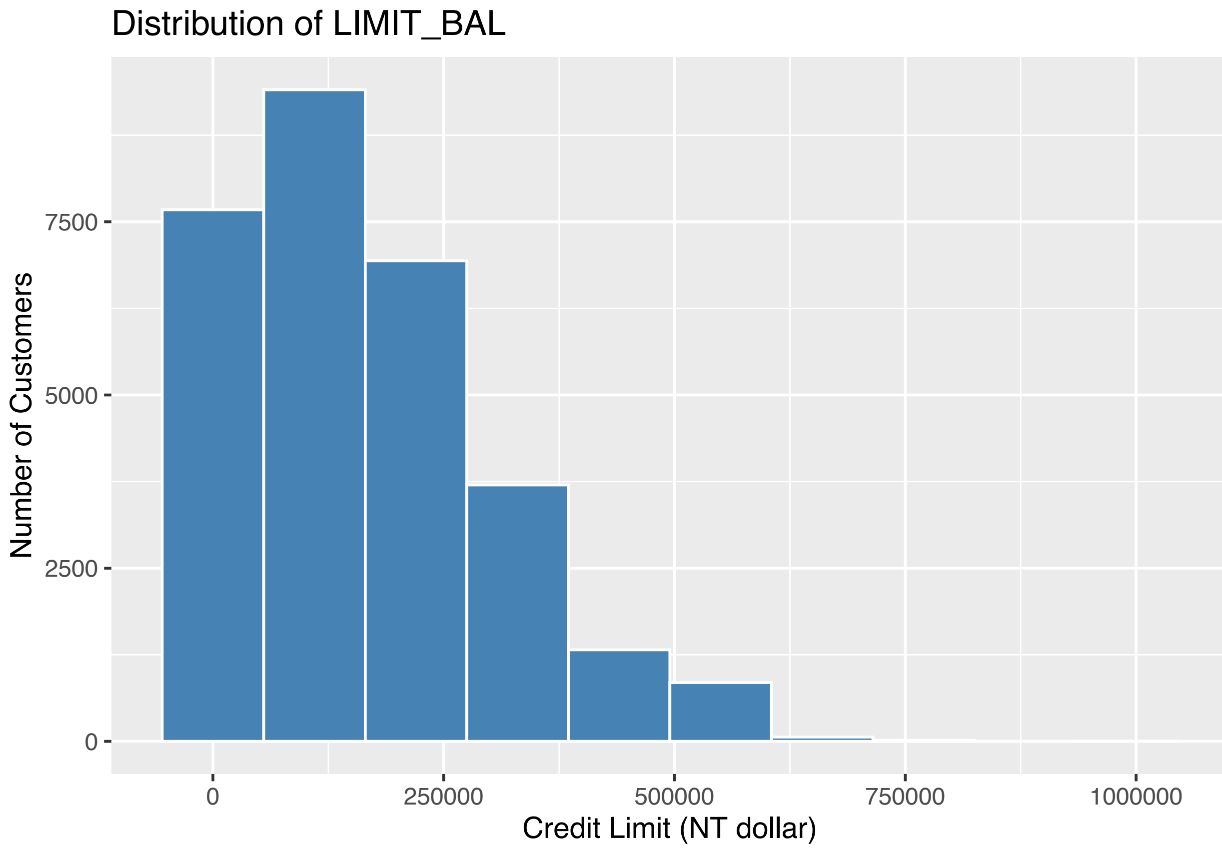
```r
# Remove duplicated rows
credit <- credit[!duplicates, ]
cat("Number of remaining rows:", nrow(credit), "\n")
```

```
## Number of remaining rows: 29965
```

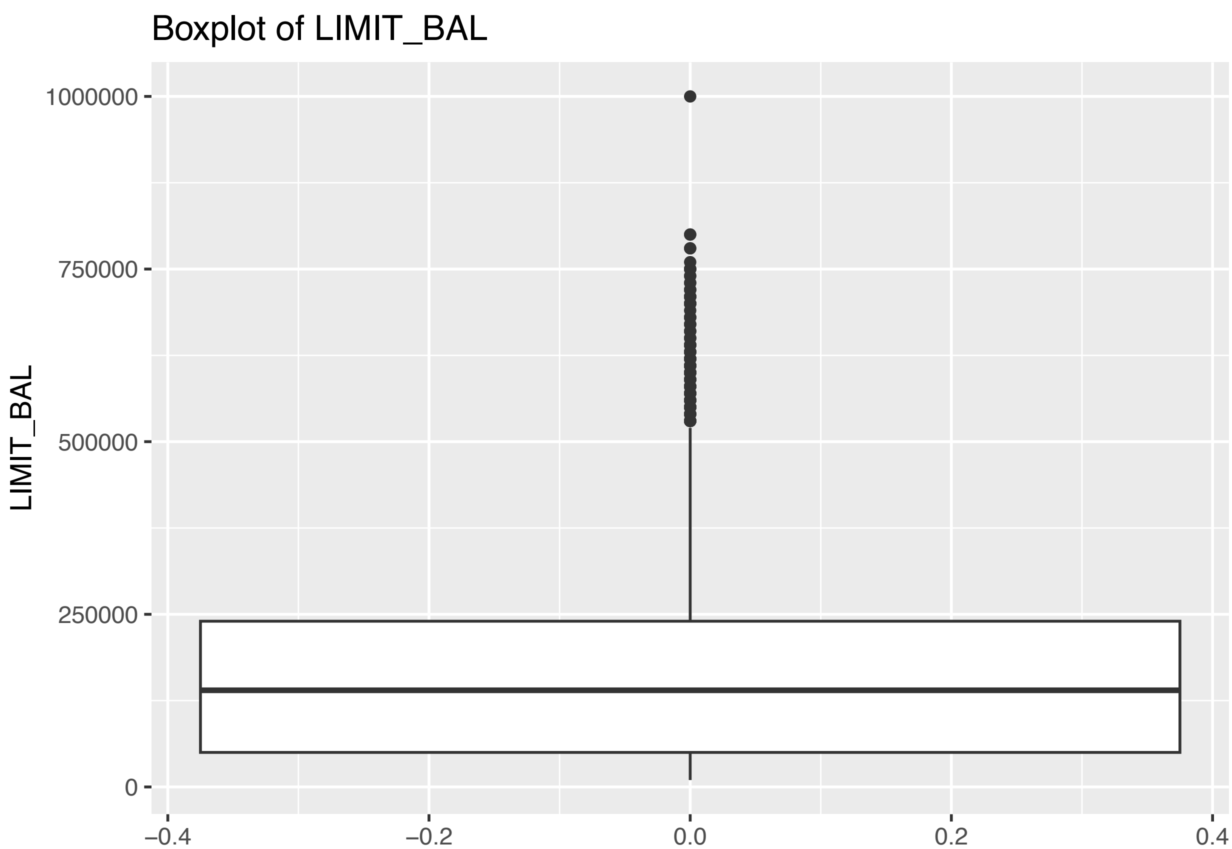
```r
# Check skewness of LIMIT_BAL
limit_bal_skew <- round(psych::skew(credit$LIMIT_BAL), 2)
cat("Skewness of LIMIT_BAL:", limit_bal_skew, "\n")
```

```
## Skewness of LIMIT_BAL: 0.99
```

```r
# Create histogram of LIMIT_BAL
ggplot(credit, aes(x = LIMIT_BAL)) +
  geom_histogram(bins = 10, fill = "steelblue", color = "white") +
  ggtitle("Distribution of LIMIT_BAL") +
  xlab("Credit Limit (NT dollar)") +
  ylab("Number of Customers")
```




```r
# Check for outliers in LIMIT_BAL using a boxplot
ggplot(credit, aes(y=LIMIT_BAL)) +
  geom_boxplot() +
  ggtitle("Boxplot of LIMIT_BAL")
```




```
# Zoom in on the outlier in LIMIT_BAL
print(credit[credit$LIMIT_BAL > 900000, 3:24])
```

```
##      SEX EDUCATION MARRIAGE AGE PAY_1 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6 BILL_AMT1
## 2198   2         1        1  47     0     0     0    -1     0     0    964511
##      BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6 PAY_AMT1 PAY_AMT2
## 2198    983931    535020    891586    927171    961664    50784    50723
##      PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6
## 2198   896040    50000    50000    50256
```

```
# Check if AGE values are within a reasonable range
cat("Number of unrealistic AGE values:", sum(credit$AGE < 18 | credit$AGE > 122), "\n")
```

```
## Number of unrealistic AGE values: 0
```

```
# Check if SEX values are valid
invalid_sex <- !credit$SEX %in% c(1, 2)
cat("Number of invalid SEX values:", sum(invalid_sex), "\n")
```

```
## Number of invalid SEX values: 0
```

```
# Check if EDUCATION values are valid
invalid_education <- !credit$EDUCATION %in% c(1, 2, 3, 4)
cat("Number of invalid EDUCATION values:", sum(invalid_education), "\n")
```

```
## Number of invalid EDUCATION values: 345
```

```r
# Check if MARRIAGE values are valid
invalid_marriage <- !credit$MARRIAGE %in% c(1, 2, 3)
cat("Number of invalid MARRIAGE values:", sum(invalid_marriage), "\n")
```

```
## Number of invalid MARRIAGE values: 54
```

```r
# Check if LIMIT_BAL values are within a reasonable range
cat("Number of unrealistic LIMIT_BAL values:", sum(credit$LIMIT_BAL <= 0 | credit$LIMIT_BAL > 1000000),
```

```
## Number of unrealistic LIMIT_BAL values: 0
```

```r
# Fix invalid entries for EDUCATION and MARRIAGE by shifting them to others category
credit$EDUCATION[credit$EDUCATION %in% c(0, 5, 6)] <- 4 # set invalid values to 4 (others)
credit$MARRIAGE[credit$MARRIAGE == 0] <- 3 # set invalid values to 3 (others)

# Check if EDUCATION values have been fixed
invalid_education <- !credit$EDUCATION %in% c(1, 2, 3, 4)
cat("Number of invalid EDUCATION values after fixing:", sum(invalid_education), "\n")
```

```
## Number of invalid EDUCATION values after fixing: 0
```

```r
# Check if MARRIAGE values have been fixed
invalid_marriage <- !credit$MARRIAGE %in% c(1, 2, 3)
cat("Number of invalid MARRIAGE values after fixing:", sum(invalid_marriage), "\n")
```

```
## Number of invalid MARRIAGE values after fixing: 0
```

```r
# Create age bins
credit$age_bins <- cut(credit$AGE, breaks = c(20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, Inf), labels =

# Check the distribution of age bins
table(credit$age_bins)
```

```
##
##    0    1    2    3    4    5    6    7    8    9   10
## 3868 7129 5790 4912 3600 2397 1425  572  186   71   15
```

```r
# View 10 random rows of the pay columns
set.seed(123) # Set random seed for reproducibility
random_rows <- sample(nrow(credit), 10) # Generate 10 random row indices
pay_cols <- c("PAY_1", "PAY_2", "PAY_3", "PAY_4", "PAY_5", "PAY_6")
credit[random_rows, pay_cols]
```

```
##       PAY_1 PAY_2 PAY_3 PAY_4 PAY_5 PAY_6
## 18860     0     0     0     0     0     0
## 18908     0     0     0    -1    -1    -2
## 26828     0     0     0     0     0     0
## 25124     0     0     0     0     0     0
```

```
## 28898    -2    -2    -2    -2    -2    -2
## 2987     -2    -1    -1    -1    -1    -1
## 1842     -2    -2    -2    -2    -1    -1
## 25741     0    -1    -1    -1    -1    -1
## 3372      0     0     0     0     0     0
## 29960    -1    -1    -1    -1    -1     0
```

```
# Count the number of data points containing 0 and -2 in the PAY_1 column
pay1_zeros <- sum(credit$PAY_1 == 0, na.rm = TRUE)
pay1_negtwos <- sum(credit$PAY_1 == -2, na.rm = TRUE)
cat("Number of data points containing 0 in PAY_1:", pay1_zeros, "\n")
```

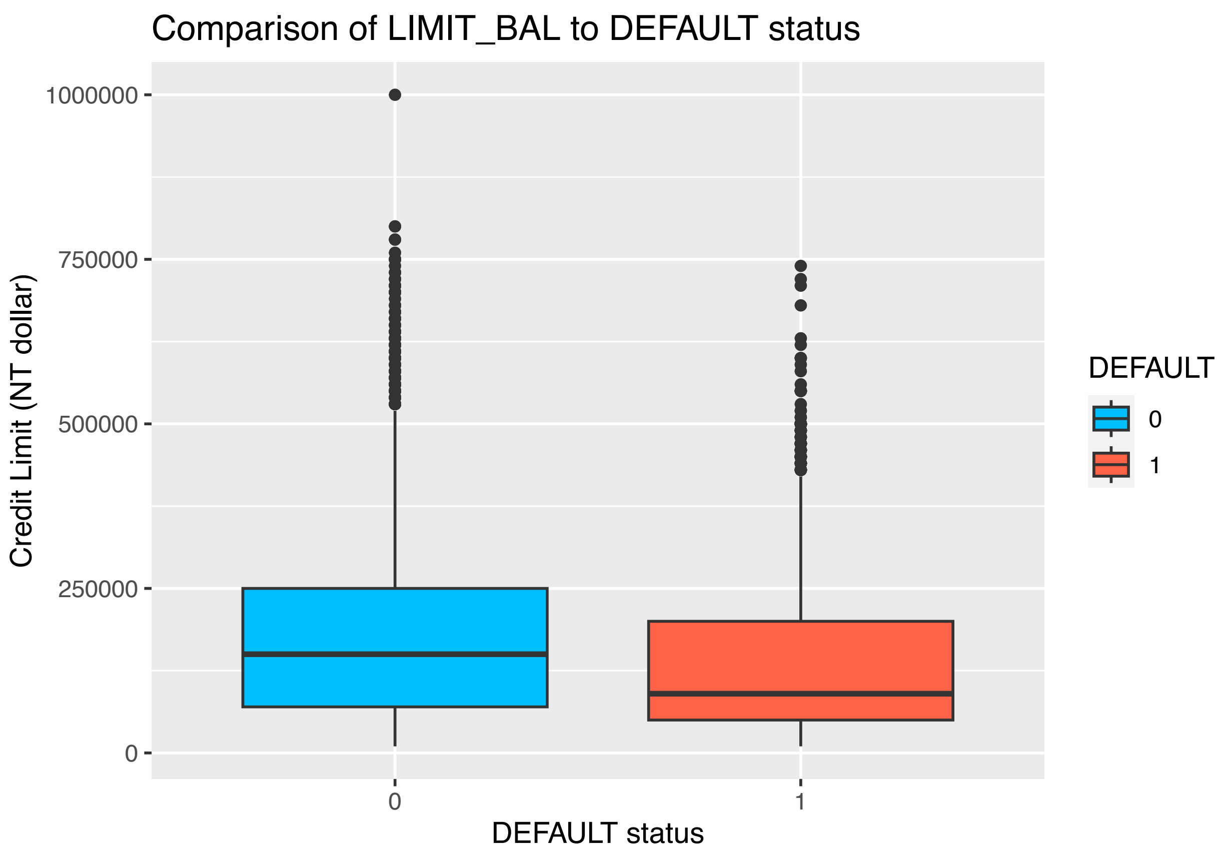
```
## Number of data points containing 0 in PAY_1: 14737
```

```
cat("Number of data points containing -2 in PAY_1:", pay1_negtwos, "\n")
```

```
## Number of data points containing -2 in PAY_1: 2750
```

```
# Remove the ID variable
credit <- credit[, -1]

# Convert categorical variables to factors
credit$SEX <- as.factor(credit$SEX)
credit$EDUCATION <- as.factor(credit$EDUCATION)
credit$MARRIAGE <- as.factor(credit$MARRIAGE)
credit$DEFAULT <- as.factor(credit$DEFAULT)
```

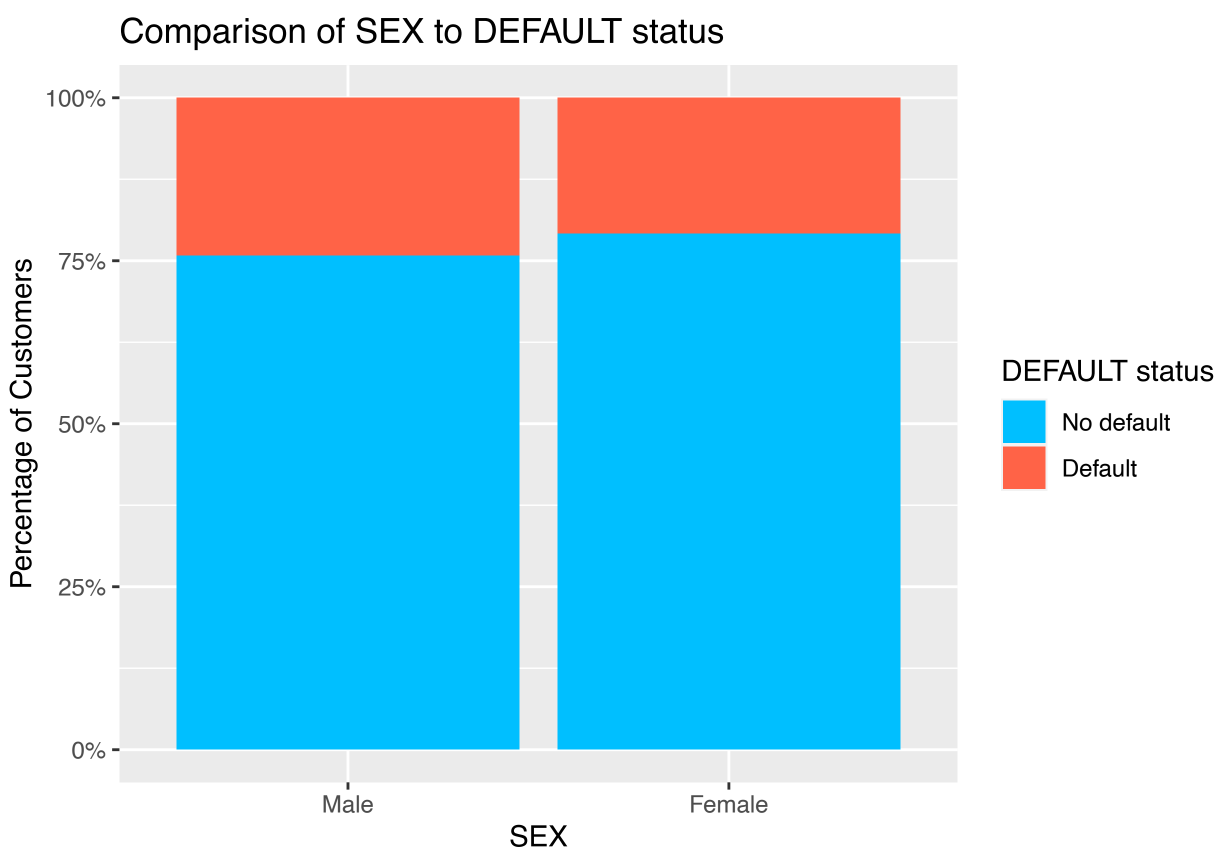
```
# Create box plot of LIMIT_BAL by DEFAULT status
ggplot(credit, aes(x = DEFAULT, y = LIMIT_BAL, fill = DEFAULT)) +
  geom_boxplot() +
  ggtitle("Comparison of LIMIT_BAL to DEFAULT status") +
  xlab("DEFAULT status") +
  ylab("Credit Limit (NT dollar)") +
  scale_fill_manual(values = c("#00BFFF", "#FF6347"))
```




```r
# Load necessary packages
library(scales)
```

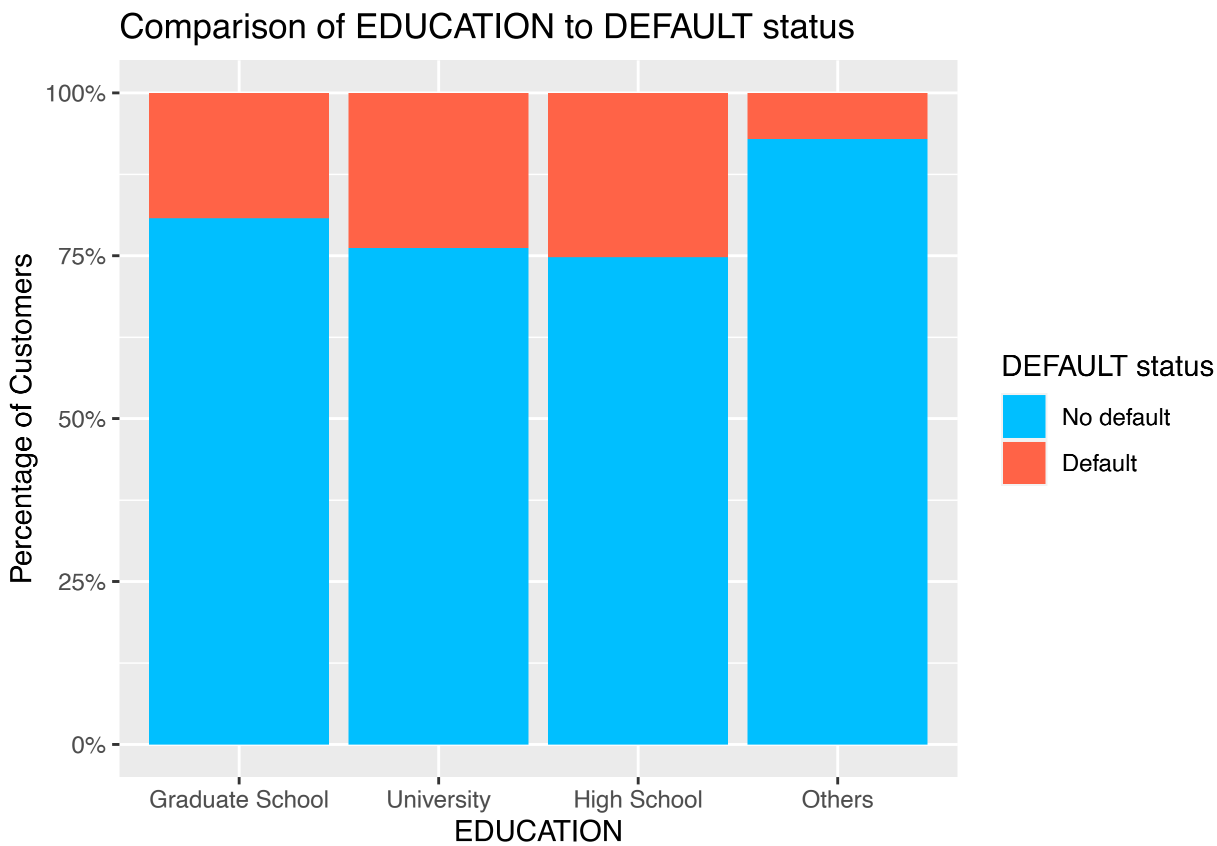
```
##
## Attaching package: 'scales'

## The following objects are masked from 'package:psych':
##
##     alpha, rescale

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

## The following objects are masked from 'package:ggvis':
##
##     fullseq, zero_range
```

```r
# Convert SEX to factor with custom levels
credit$SEX <- factor(credit$SEX, levels = c(1, 2), labels = c("Male", "Female"))

# Create stacked bar plot of SEX by DEFAULT status as percentages
ggplot(credit, aes(x = SEX, fill = DEFAULT)) +
  geom_bar(position = position_fill(reverse = TRUE)) +
  ggtitle("Comparison of SEX to DEFAULT status") +
  xlab("SEX") +
  ylab("Percentage of Customers") +
  scale_fill_manual(values = c("#00BFFF", "#FF6347"), labels = c("No default", "Default")) +
  scale_y_continuous(labels = percent_format()) +
  guides(fill = guide_legend(title = "DEFAULT status"))
```

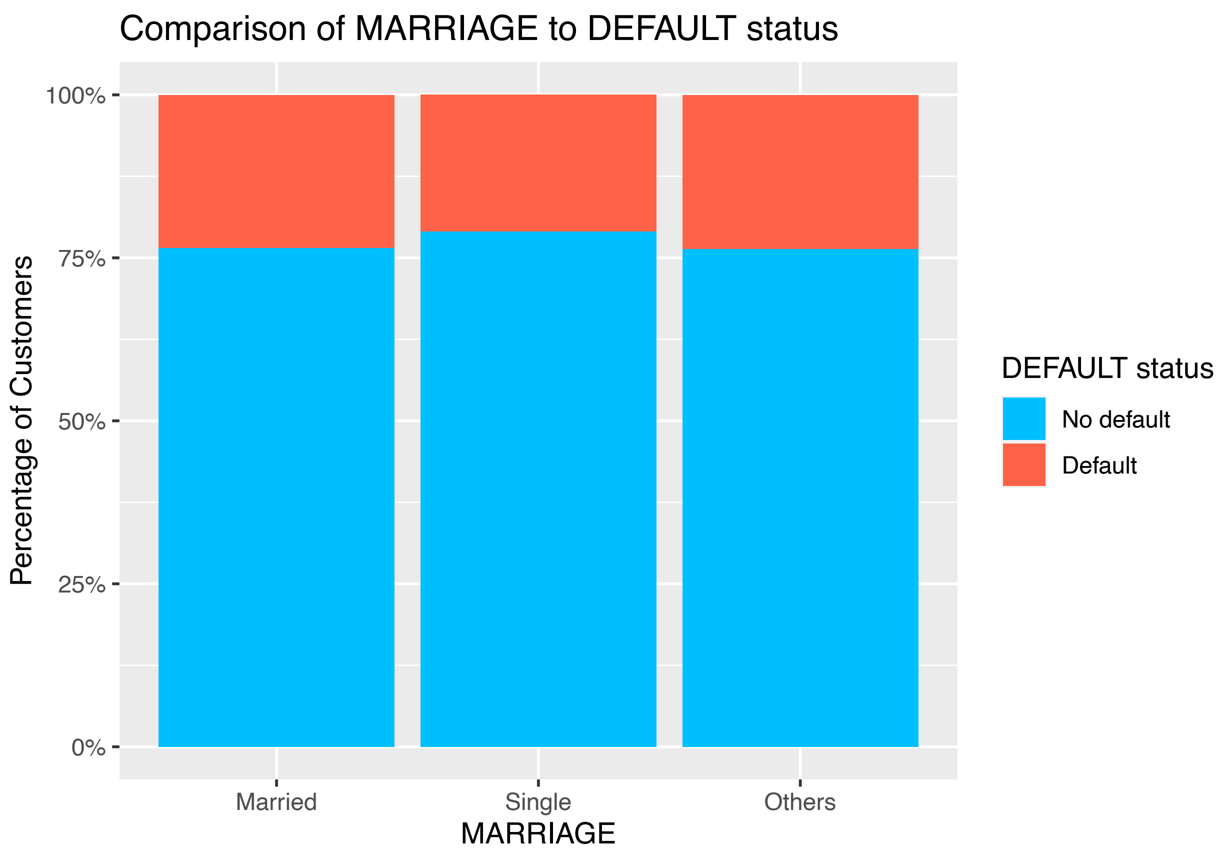



```r
# Convert EDUCATION to factor with custom levels and labels
credit$EDUCATION <- factor(credit$EDUCATION,
                           levels = c(1, 2, 3, 4),
                           labels = c("Graduate School", "University", "High School", "Others"))

# Create stacked bar plot of EDUCATION by DEFAULT status as percentages with labels
ggplot(credit, aes(x = EDUCATION, fill = DEFAULT)) +
  geom_bar(position = position_fill(reverse = TRUE)) +
  ggtitle("Comparison of EDUCATION to DEFAULT status") +
  xlab("EDUCATION") +
  ylab("Percentage of Customers") +
  scale_fill_manual(values = c("#00BFFF", "#FF6347"), labels = c("No default", "Default")) +
```

```
  scale_y_continuous(labels = percent_format()) +
  guides(fill = guide_legend(title = "DEFAULT status"))
```



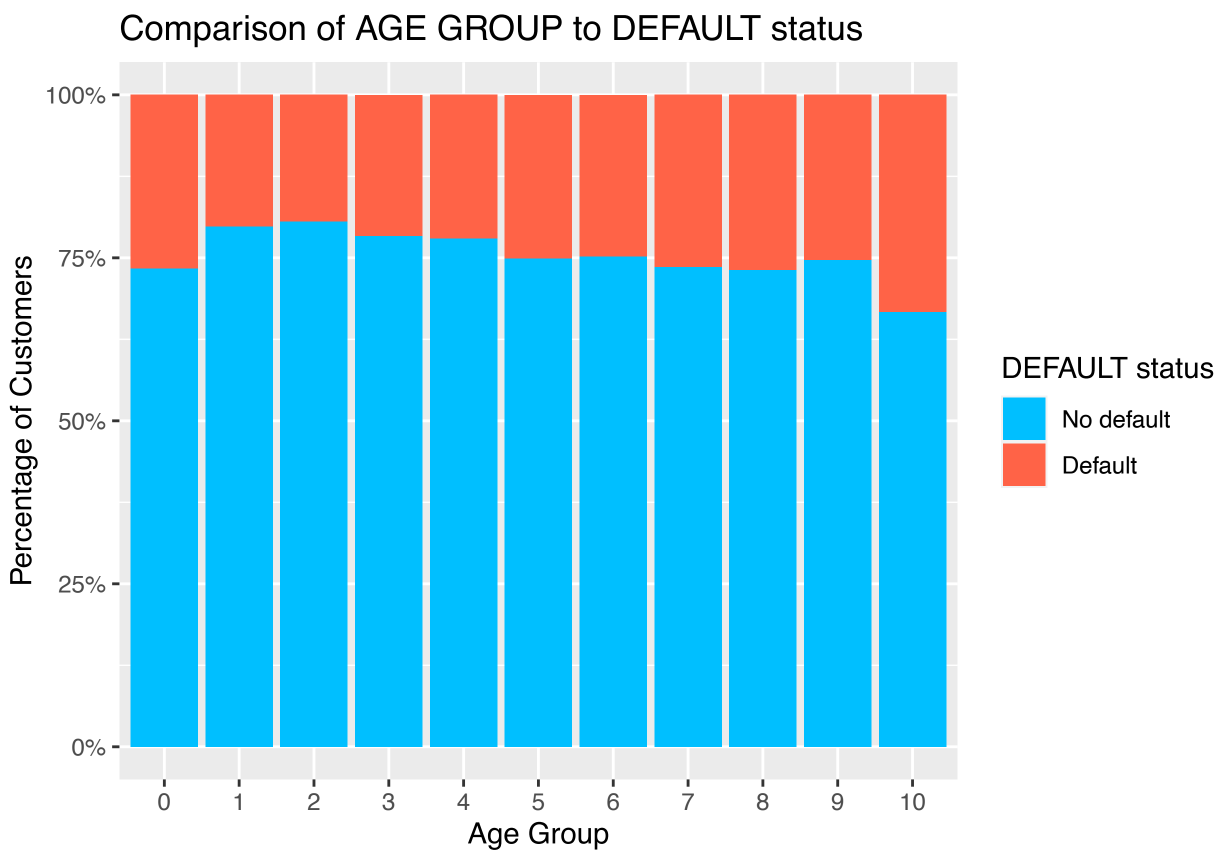

```
# Convert MARRIAGE to factor with custom levels and labels
credit$MARRIAGE <- factor(credit$MARRIAGE,
                          levels = c(1, 2, 3),
                          labels = c("Married", "Single", "Others"))

# Create stacked bar plot of MARRIAGE by DEFAULT status as percentages with labels
ggplot(credit, aes(x = MARRIAGE, fill = DEFAULT)) +
  geom_bar(position = position_fill(reverse = TRUE)) +
  ggtitle("Comparison of MARRIAGE to DEFAULT status") +
  xlab("MARRIAGE") +
  ylab("Percentage of Customers") +
  scale_fill_manual(values = c("#00BFFF", "#FF6347"), labels = c("No default", "Default")) +
  scale_y_continuous(labels = percent_format()) +
  guides(fill = guide_legend(title = "DEFAULT status"))
```




```r
# Create a 2x2 contingency table of DEFAULT by MARRIAGE
cont_table <- table(credit$DEFAULT, credit$MARRIAGE)

# Run chi-squared test for independence
chisq.test(cont_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  cont_table
## X-squared = 27.489, df = 2, p-value = 1.074e-06
```

```r
# Create stacked bar plot of AGE_BINS by DEFAULT status as percentages
ggplot(credit, aes(x = age_bins, fill = DEFAULT)) +
  geom_bar(position = position_fill(reverse = TRUE)) +
  ggtitle("Comparison of AGE GROUP to DEFAULT status") +
  xlab("Age Group") +
  ylab("Percentage of Customers") +
  scale_fill_manual(values = c("#00BFFF", "#FF6347"), labels = c("No default", "Default")) +
  scale_y_continuous(labels = percent_format()) +
  guides(fill = guide_legend(title = "DEFAULT status"))
```

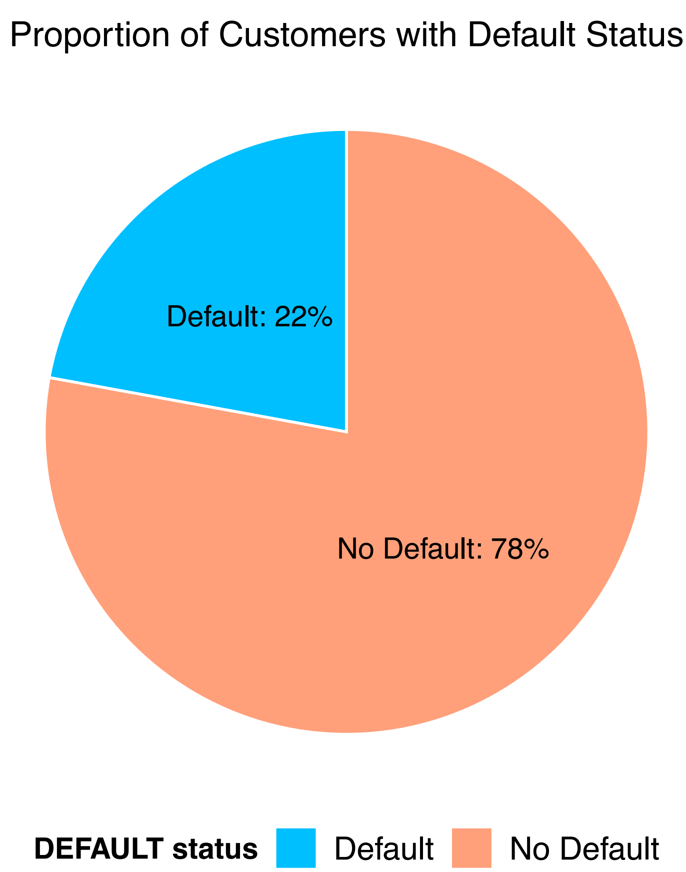



```r
# Fit a logistic regression model to predict default based on age
model <- glm(DEFAULT ~ AGE, data = credit, family = "binomial")

# Print the model summary to check if age is a significant predictor
summary(model)
```

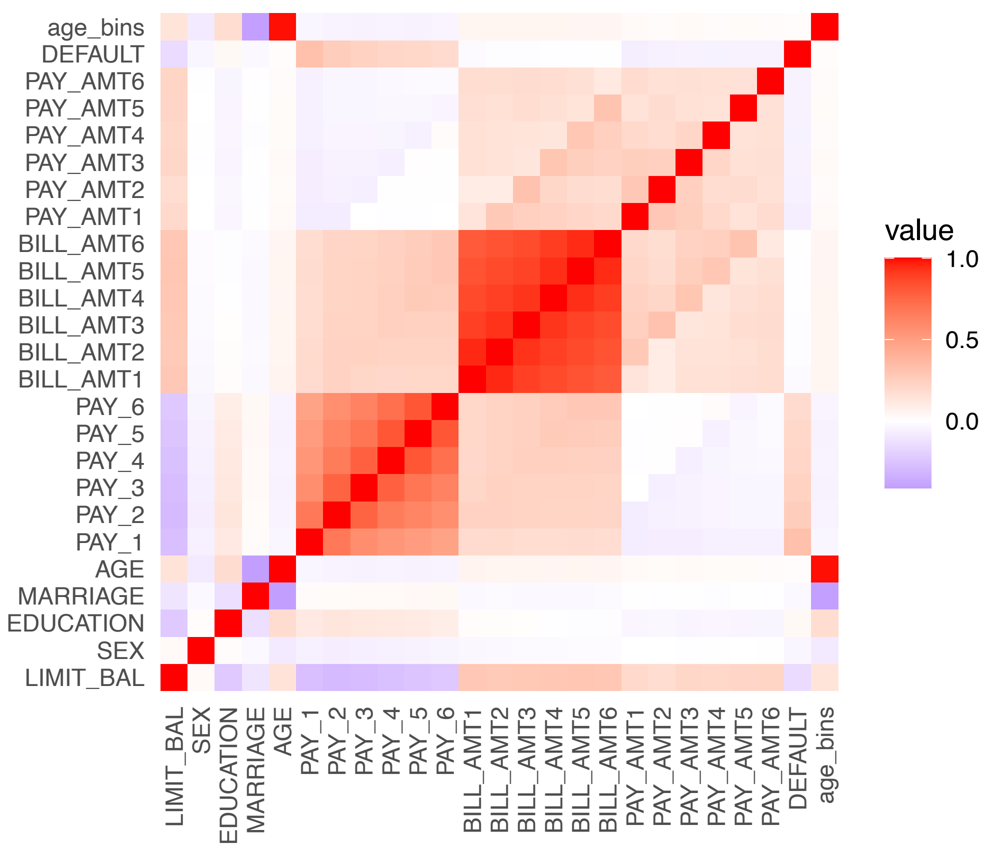
```
##
## Call:
## glm(formula = DEFAULT ~ AGE, family = "binomial", data = credit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7565  -0.7110  -0.7011  -0.6934   1.7600
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.384131   0.055265 -25.045   <2e-16 ***
## AGE          0.003536   0.001500   2.357   0.0184 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 31673  on 29964  degrees of freedom
## Residual deviance: 31667  on 29963  degrees of freedom
## AIC: 31671
```

```
##
## Number of Fisher Scoring iterations: 4
```

```r
# Calculate proportion of customers with and without default
prop_default <- mean(credit$DEFAULT == 1)
prop_no_default <- mean(credit$DEFAULT == 0)

#creating pie graph
pie_data <- data.frame(DEFAULT = c("Default", "No Default"),
                       Proportion = c(prop_default, prop_no_default))

ggplot(pie_data, aes(x="", y=Proportion, fill=DEFAULT)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar(theta = "y") +
  ggtitle("Proportion of Customers with Default Status") +
  theme_void() +
  theme(legend.position = "bottom") +
  guides(fill = guide_legend(title = "DEFAULT status")) +
  geom_text(aes(label = paste0(DEFAULT, ": ", round(Proportion * 100), "%")),
            position = position_stack(vjust = 0.5)) +
  scale_fill_manual(values=c("#00BFFF", "#FFA07A")) + # set fill colors
  theme(plot.title = element_text(hjust = 0.5), # center plot title
        legend.title = element_text(face = "bold"), # bold legend title
        legend.text = element_text(size = 12)) # increase legend text size
```



```r
# Load necessary libraries
library(reshape2)

# Create a correlation matrix of all variables in the credit dataset
credit2 <- as.data.frame(sapply(credit, as.numeric))

corr_mat <- cor(credit2)

# Create a heatmap of the correlation matrix
ggplot(data = melt(corr_mat), aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
  theme_minimal() +
  coord_fixed() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank())
```




```r
# Select variables with absolute correlation > 0.2
highly_correlated <- findCorrelation(corr_mat, cutoff = 0.2, verbose = FALSE)

# Subset the credit dataset to only include highly correlated variables
credit_subset <- credit[, highly_correlated]
```

```r
# Check the dimensions of the subsetted dataset
dim(credit_subset)
```

```
## [1] 29965    17
```

```r
# Load necessary packages
library(e1071)

# Split data into training and testing sets
set.seed(123)
train_index <- sample(1:nrow(credit), size = round(0.7*nrow(credit)), replace = FALSE)
train_data <- credit[train_index,]
test_data <- credit[-train_index,]

# Calculate proportion of default vs non-default in whole dataset
prop_table_all <- prop.table(table(credit$DEFAULT))

# Calculate proportion of default vs non-default in training set
prop_table_train <- prop.table(table(train_data$DEFAULT))

# Calculate proportion of default vs non-default in test set
prop_table_test <- prop.table(table(test_data$DEFAULT))

# Compare proportions using chi-squared test of independence
chisq_all_train <- chisq.test(prop_table_all, prop_table_train)
```

```
## Warning in chisq.test(prop_table_all, prop_table_train): Chi-squared
## approximation may be incorrect
```

```r
chisq_all_test <- chisq.test(prop_table_all, prop_table_test)
```

```
## Warning in chisq.test(prop_table_all, prop_table_test): Chi-squared
## approximation may be incorrect
```

```r
chisq_train_test <- chisq.test(prop_table_train, prop_table_test)
```

```
## Warning in chisq.test(prop_table_train, prop_table_test): Chi-squared
## approximation may be incorrect
```

```r
# Print results
cat("Chi-squared test comparing proportions of default vs non-default in whole dataset and training set
```

```
## Chi-squared test comparing proportions of default vs non-default in whole dataset and training set:
```

```r
chisq_all_train
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  prop_table_all and prop_table_train
## X-squared = 0, df = 1, p-value = 1
```

```r
cat("Chi-squared test comparing proportions of default vs non-default in whole dataset and test set:")
```

```
## Chi-squared test comparing proportions of default vs non-default in whole dataset and test set:
```

```r
chisq_all_test
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  prop_table_all and prop_table_test
## X-squared = 0, df = 1, p-value = 1
```

```r
cat("Chi-squared test comparing proportions of default vs non-default in training set and test set:")
```

```
## Chi-squared test comparing proportions of default vs non-default in training set and test set:
```

```r
chisq_train_test
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  prop_table_train and prop_table_test
## X-squared = 0, df = 1, p-value = 1
```

```r
# Train SVM model
svm_model <- svm(DEFAULT ~ LIMIT_BAL + SEX + EDUCATION + MARRIAGE + age_bins +
             PAY_1 + PAY_2 + PAY_3 + PAY_4 + PAY_5 + PAY_6 +
             PAY_AMT1+ PAY_AMT2 + PAY_AMT3 + PAY_AMT4 + PAY_AMT5 + PAY_AMT6 +
             BILL_AMT1  + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 + BILL_AMT5 + BILL_AMT6 , data = train_data)

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

# Calculate confusion matrix
conf_mat_svm <- confusionMatrix(test_data$DEFAULT, svm_pred)

conf_mat_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6681  300
##          1 1344  664
##
##                Accuracy : 0.8171
##                  95% CI : (0.809, 0.8251)
##     No Information Rate : 0.8928
##     P-Value [Acc > NIR] : 1
##
```

```
##                  Kappa : 0.3531
##
##  Mcnemar's Test P-Value : <2e-16
##
##            Sensitivity : 0.8325
##            Specificity : 0.6888
##         Pos Pred Value : 0.9570
##         Neg Pred Value : 0.3307
##             Prevalence : 0.8928
##         Detection Rate : 0.7432
##   Detection Prevalence : 0.7766
##      Balanced Accuracy : 0.7607
##
##       'Positive' Class : 0
##
```

```
cm.svm.test <- table(test_data$DEFAULT, svm_pred)

cm.svm.test
```

```
##    svm_pred
##        0    1
##   0 6681  300
##   1 1344  664
```

```
TP.svm <- cm.svm.test[2,2]
TN.svm <- cm.svm.test[1,1]
FP.svm <- cm.svm.test[1,2]
FN.svm <- cm.svm.test[2,1]

accuracy.svm.test <- (TP.svm + TN.svm) /(TP.svm+TN.svm+FP.svm+FN.svm)
sensitivity.svm.test <- TP.svm/(FN.svm+TP.svm)
specificity.svm.test <- TN.svm/(TN.svm+FP.svm)
precision.svm.test <- TP.svm/(FP.svm+TP.svm)
aca.svm.test <- (sensitivity.svm.test + specificity.svm.test) / 2
hm.svm.test <- 1 / ( ( (1/sensitivity.svm.test) + (1/specificity.svm.test) ) / 2)

accuracy.svm.test
```

```
## [1] 0.8171098
```

```
sensitivity.svm.test
```

```
## [1] 0.3306773
```

```
specificity.svm.test
```

```
## [1] 0.9570262
```

```
precision.svm.test
```

```
## [1] 0.6887967
```

```
aca.svm.test
```

```
## [1] 0.6438518
```

```
hm.svm.test
```

```
## [1] 0.4915213
```

```
# Convert DEFAULT to factor with levels "No default" and "Default"
credit$DEFAULT <- factor(credit$DEFAULT, levels = c(0, 1), labels = c("No default", "Default"))



# Train logistic regression model
glm_model <- glm(DEFAULT ~ LIMIT_BAL + SEX + EDUCATION + MARRIAGE + age_bins +
             PAY_1 + PAY_2 + PAY_3 + PAY_4 + PAY_5 + PAY_6 +
             PAY_AMT1+ PAY_AMT2 + PAY_AMT3 + PAY_AMT4 + PAY_AMT5 + PAY_AMT6 +
             BILL_AMT1  + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 + BILL_AMT5 + BILL_AMT6,
             family = binomial(link = "logit"), data = train_data)

# Make predictions on test data
glm_pred <- predict(newdata = test_data, glm_model, type = "response")
glm_pred <- ifelse(glm_pred > 0.5, 1, 0)

# Convert predicted data to factor with same levels as test data
glm_pred_factor <- factor(glm_pred, levels = levels(test_data$DEFAULT))

# Calculate confusion matrix
conf_mat_glm <- confusionMatrix(test_data$DEFAULT, glm_pred_factor)

# Print confusion matrix
conf_mat_glm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6766  215
##          1 1530  478
##
##                Accuracy : 0.8059
##                  95% CI : (0.7975, 0.814)
##     No Information Rate : 0.9229
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2703
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8156
##             Specificity : 0.6898
##          Pos Pred Value : 0.9692
##          Neg Pred Value : 0.2380
##              Prevalence : 0.9229
##          Detection Rate : 0.7527
##    Detection Prevalence : 0.7766
##       Balanced Accuracy : 0.7527
##
##        'Positive' Class : 0
##
```

```r
cm.glm.test <- table(test_data$DEFAULT, glm_pred)

cm.glm.test
```

```
##    glm_pred
##        0    1
##   0 6766  215
##   1 1530  478
```

```r
TP.glm <- cm.glm.test[2,2]
TN.glm <- cm.glm.test[1,1]
FP.glm <- cm.glm.test[1,2]
FN.glm <- cm.glm.test[2,1]

accuracy.glm.test <- (TP.glm + TN.glm) /(TP.glm+TN.glm+FP.glm+FN.glm)
sensitivity.glm.test <- TP.glm/(FN.glm+TP.glm)
specificity.glm.test <- TN.glm/(TN.glm+FP.glm)
precision.glm.test <- TP.glm/(FP.glm+TP.glm)
aca.glm.test <- (sensitivity.glm.test + specificity.glm.test) / 2
hm.glm.test <- 1 / ( ( (1/sensitivity.glm.test) + (1/specificity.glm.test) ) / 2)

accuracy.glm.test
```

```
## [1] 0.8058738
```

```r
sensitivity.glm.test
```

```
## [1] 0.2380478
```

```r
specificity.glm.test
```

```
## [1] 0.9692021
```

```r
precision.glm.test
```

```
## [1] 0.6897547
```

```r
aca.glm.test
```

```
## [1] 0.603625
```

```r
hm.glm.test
```

```
## [1] 0.3822182
```

```r
# Load necessary packages
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(caret)

# Train random forest model
rf_model <- randomForest(DEFAULT ~ LIMIT_BAL + SEX + EDUCATION + MARRIAGE + age_bins +
             PAY_1 + PAY_2 + PAY_3 + PAY_4 + PAY_5 + PAY_6 +
             PAY_AMT1+ PAY_AMT2 + PAY_AMT3 + PAY_AMT4 + PAY_AMT5 + PAY_AMT6 +
             BILL_AMT1  + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 + BILL_AMT5 + BILL_AMT6,
```

```r
          data = train_data, ntree = 500, importance = TRUE)

# Make predictions on test data
rf_pred <- predict(rf_model, newdata = test_data)

# Calculate confusion matrix
conf_mat_rf <- confusionMatrix(test_data$DEFAULT, rf_pred)

# Print confusion matrix
conf_mat_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6565  416
##          1 1272  736
##
##                Accuracy : 0.8122
##                  95% CI : (0.804, 0.8202)
##     No Information Rate : 0.8718
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3619
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8377
##             Specificity : 0.6389
##          Pos Pred Value : 0.9404
##          Neg Pred Value : 0.3665
##              Prevalence : 0.8718
##          Detection Rate : 0.7303
##    Detection Prevalence : 0.7766
##       Balanced Accuracy : 0.7383
##
##        'Positive' Class : 0
##
```

```r
# Calculate performance metrics
TP_rf <- conf_mat_rf$table[2, 2]
TN_rf <- conf_mat_rf$table[1, 1]
FP_rf <- conf_mat_rf$table[1, 2]
FN_rf <- conf_mat_rf$table[2, 1]

accuracy_rf <- (TP_rf + TN_rf) / (TP_rf + TN_rf + FP_rf + FN_rf)
sensitivity_rf <- TP_rf / (FN_rf + TP_rf)
specificity_rf <- TN_rf / (TN_rf + FP_rf)
precision_rf <- TP_rf / (FP_rf + TP_rf)
aca_rf <- (sensitivity_rf + specificity_rf) / 2
hm_rf <- 1 / ( ( (1/sensitivity_rf) + (1/specificity_rf) ) / 2)

# Print performance metrics
```

```r
cat("Accuracy of random forest model:", accuracy_rf, "\n")
```

```
## Accuracy of random forest model: 0.8122149
```

```r
cat("Sensitivity of random forest model:", sensitivity_rf, "\n")
```

```
## Sensitivity of random forest model: 0.3665339
```

```r
cat("Specificity of random forest model:", specificity_rf, "\n")
```

```
## Specificity of random forest model: 0.9404097
```

```r
cat("Precision of random forest model:", precision_rf, "\n")
```

```
## Precision of random forest model: 0.6388889
```

```r
cat("Average Class Accuracy of random forest model:", aca_rf, "\n")
```

```
## Average Class Accuracy of random forest model: 0.6534718
```

```r
cat("Harmonic Mean of random forest model:", hm_rf, "\n")
```

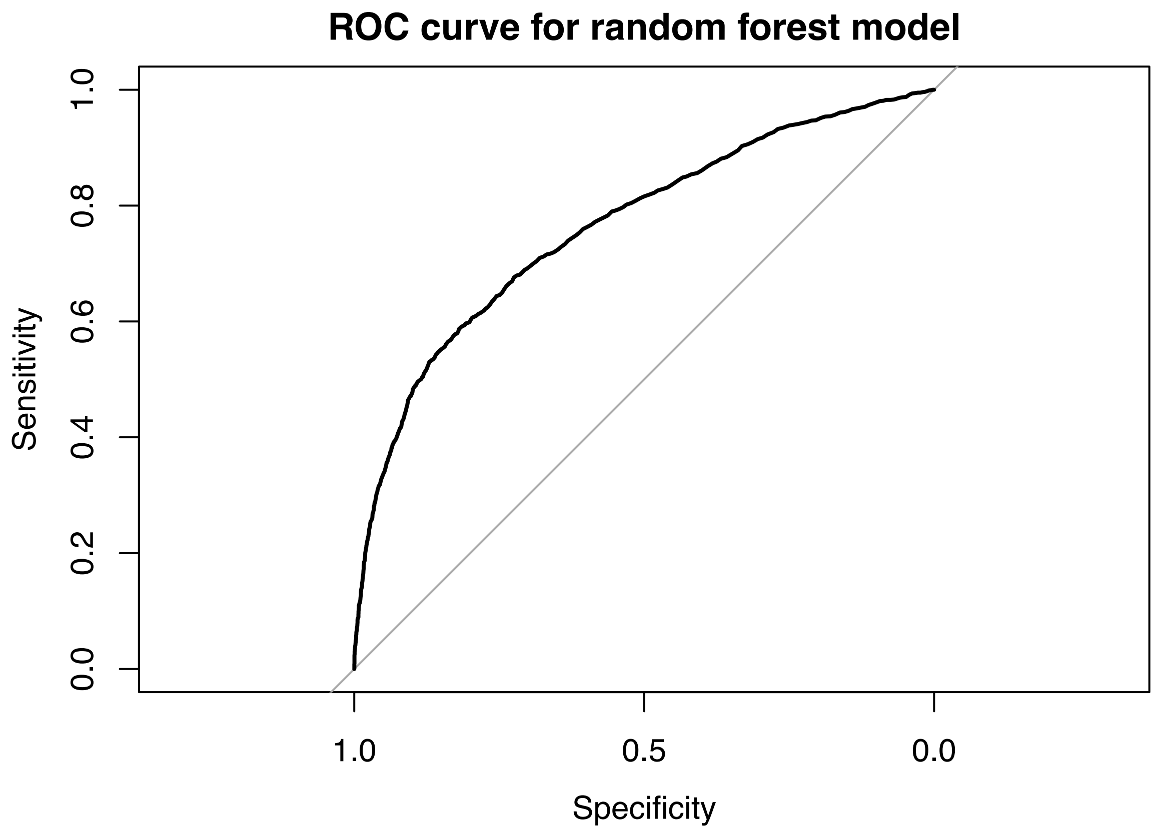
```
## Harmonic Mean of random forest model: 0.527478
```

```r
# Create ROC curve
roc_rf <- roc(test_data$DEFAULT, predict(rf_model, newdata = test_data, type = "prob")[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Plot ROC curve
plot(roc_rf, main = "ROC curve for random forest model")
```




```r
# Calculate AUC
auc_rf <- auc(roc_rf)
cat("AUC of random forest model:", auc_rf, "\n")
```

```
## AUC of random forest model: 0.7639172
```

```r
# Use k-fold cross validation to evaluate the model
set.seed(123)
cv <- trainControl(method = "cv", number = 10, savePredictions = TRUE)
rf_model_cv <- train(DEFAULT ~ LIMIT_BAL + SEX + EDUCATION + MARRIAGE + age_bins +
             PAY_1 + PAY_2 + PAY_3 + PAY_4 + PAY_5 + PAY_6 +
             PAY_AMT1+ PAY_AMT2 + PAY_AMT3 + PAY_AMT4 + PAY_AMT5 + PAY_AMT6 +
             BILL_AMT1  + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 + BILL_AMT5 + BILL_AMT6,
           data = train_data, method = "rf", trControl = cv, tuneGrid = expand.grid(mtry = 2))
rf_model_cv$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD    KappaSD
## 1    2 0.8154073 0.3252287 0.006435947 0.02955361
```

```r
# Load necessary packages
library(nnet)

# Train neural network model
nn_model <- nnet(DEFAULT ~ LIMIT_BAL + SEX + EDUCATION + MARRIAGE + age_bins +
```

```
             PAY_1 + PAY_2 + PAY_3 + PAY_4 + PAY_5 + PAY_6 +
             PAY_AMT1+ PAY_AMT2 + PAY_AMT3 + PAY_AMT4 + PAY_AMT5 + PAY_AMT6 +
             BILL_AMT1  + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 + BILL_AMT5 + BILL_AMT6,
             data = train_data, size = 5, decay = 5e-4, maxit = 500)
```

```
## # weights:  186
## initial  value 22185.443682
## iter  10 value 10901.668508
## iter  20 value 10882.102489
## iter  30 value 10875.649825
## iter  40 value 10873.142330
## iter  50 value 10868.279801
## iter  60 value 10865.230765
## iter  70 value 10862.083790
## iter  80 value 10858.779138
## iter  90 value 10856.685207
## iter 100 value 10855.432087
## iter 110 value 10855.316782
## iter 120 value 10855.274526
## iter 130 value 10854.679835
## iter 140 value 10854.410283
## iter 150 value 10854.050409
## iter 160 value 10853.450406
## iter 170 value 10851.903187
## iter 180 value 10851.813560
## iter 190 value 10851.750779
## iter 200 value 10851.596824
## iter 210 value 10851.588263
## final  value 10851.588004
## converged
```

```
# Make predictions on test data
nn_pred <- predict(nn_model, newdata = test_data, type = "class")

# Convert nn_pred to factor with the levels of test_data$DEFAULT
nn_pred <- factor(nn_pred, levels = levels(test_data$DEFAULT))

# Calculate confusion matrix
conf_mat_nn <- table(test_data$DEFAULT, nn_pred)

# Ensure conf_mat_nn has the correct structure
if (ncol(conf_mat_nn) == 1) {
  conf_mat_nn <- cbind(conf_mat_nn, c(0, 0))
  colnames(conf_mat_nn) <- c(0, 1)
}

TP_nn <- conf_mat_nn[2, 2]
TN_nn <- conf_mat_nn[1, 1]
FP_nn <- conf_mat_nn[1, 2]
FN_nn <- conf_mat_nn[2, 1]

accuracy_nn <- (TP_nn + TN_nn) / (TP_nn + TN_nn + FP_nn + FN_nn)
sensitivity_nn <- TP_nn / (FN_nn + TP_nn)
```

```r
specificity_nn <- TN_nn / (TN_nn + FP_nn)
precision_nn <- TP_nn / (FP_nn + TP_nn)
aca_nn <- (sensitivity_nn + specificity_nn) / 2
hm_nn <- 1 / ( ( (1/sensitivity_nn) + (1/specificity_nn) ) / 2)

# Print confusion matrix and performance metrics
conf_mat_nn
```

```
##    nn_pred
##        0    1
##   0 6981    0
##   1 2008    0
```

```r
cat("Accuracy of neural network model:", accuracy_nn, "\n")
```

```
## Accuracy of neural network model: 0.7766159
```

```r
cat("Sensitivity of neural network model:", sensitivity_nn, "\n")
```

```
## Sensitivity of neural network model: 0
```

```r
cat("Specificity of neural network model:", specificity_nn, "\n")
```

```
## Specificity of neural network model: 1
```

```r
cat("Precision of neural network model:", precision_nn, "\n")
```

```
## Precision of neural network model: NaN
```

```r
cat("Average Class Accuracy of neural network model:", aca_nn, "\n")
```

```
## Average Class Accuracy of neural network model: 0.5
```

```r
cat("Harmonic Mean of neural network model:", hm_nn, "\n")
```

```
## Harmonic Mean of neural network model: 0
```