```r
# Load required libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(ggplot2)
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:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(e1071)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(gridExtra)
```

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

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(nnet)

suppressPackageStartupMessages({
library(tidyverse)
library(caret)
library(ggplot2)
library(randomForest)
library(e1071)
library(pROC)
library(gridExtra)
library(xgboost)
library(nnet)
})
```

```r
# Step 1: Load and explore the data
loan_data <- read.csv("D:/project/Loan_default_updated.csv")

# View the structure of the data
str(loan_data)
```

```
## 'data.frame':    20000 obs. of  18 variables:
##  $ LoanID        : chr  "8EGC3UUTY8" "2ZLI6TAHI5" "8WPZH835VS" "HAU91YNH13" ...
##  $ Age           : int  32 64 21 57 35 25 25 58 19 44 ...
##  $ Income        : int  63892 41848 103298 120690 137245 68120 139889 22190 39792 117250 ...
##  $ LoanAmount    : int  66362 177446 111902 30751 176172 147458 195778 79189 191417 110979 ...
##  $ CreditScore   : int  444 693 689 647 585 405 510 320 540 506 ...
##  $ MonthsEmployed: int  90 98 17 46 71 119 86 113 99 84 ...
##  $ NumCreditLines: int  3 3 3 3 1 2 3 3 4 1 ...
##  $ InterestRate  : num  7.45 16.06 23.19 14.86 12.02 ...
##  $ LoanTerm      : int  36 36 24 48 12 48 48 12 36 60 ...
##  $ DTIRatio      : num  0.27 0.46 0.89 0.14 0.4 0.37 0.52 0.12 0.17 0.43 ...
##  $ Education     : chr  "High School" "High School" "Bachelor's" "Master's" ...
##  $ EmploymentType: chr  "Full-time" "Full-time" "Part-time" "Unemployed" ...
##  $ MaritalStatus : chr  "Single" "Single" "Single" "Single" ...
##  $ HasMortgage   : chr  "No" "No" "No" "Yes" ...
##  $ HasDependents : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ LoanPurpose   : chr  "Education" "Other" "Business" "Home" ...
##  $ HasCoSigner   : chr  "No" "No" "No" "Yes" ...
##  $ Default       : int  0 0 1 0 0 1 0 0 0 0 ...
```

```r
# Summary statistics
summary(loan_data)
```

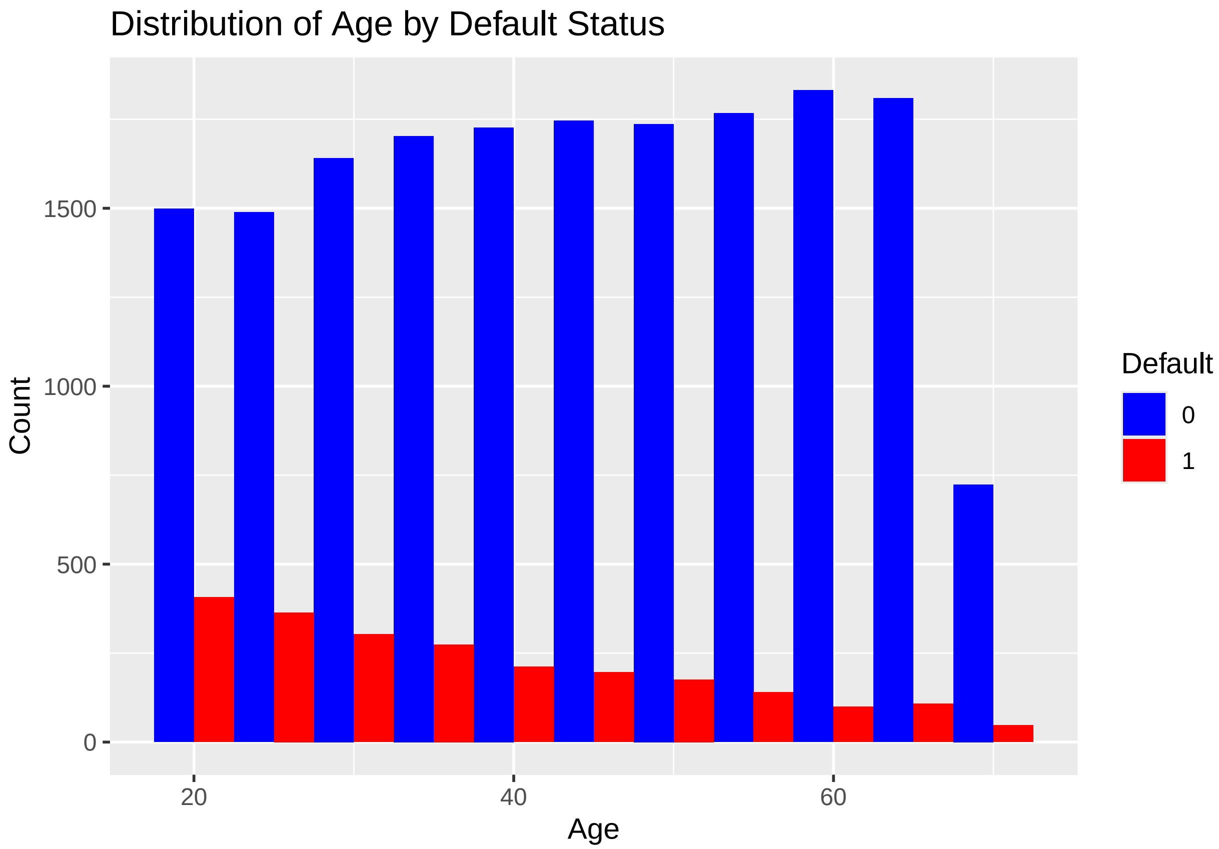
```
##     LoanID               Age            Income         LoanAmount
##  Length:20000       Min.   :18.00   Min.   : 15009   Min.   :  5009
##  Class :character   1st Qu.:31.00   1st Qu.: 48377   1st Qu.: 65308
##  Mode  :character   Median :44.00   Median : 81941   Median :127356
##                     Mean   :43.54   Mean   : 82115   Mean   :127384
##                     3rd Qu.:56.00   3rd Qu.:115763   3rd Qu.:188757
##                     Max.   :69.00   Max.   :149975   Max.   :249992
##   CreditScore    MonthsEmployed   NumCreditLines  InterestRate
##  Min.   :300.0   Min.   :  0.00   Min.   :1.00   Min.   : 2.00
##  1st Qu.:436.0   1st Qu.: 29.00   1st Qu.:2.00   1st Qu.: 7.78
##  Median :573.0   Median : 59.00   Median :3.00   Median :13.43
##  Mean   :573.7   Mean   : 59.38   Mean   :2.51   Mean   :13.49
##  3rd Qu.:712.0   3rd Qu.: 90.00   3rd Qu.:4.00   3rd Qu.:19.28
##  Max.   :849.0   Max.   :119.00   Max.   :4.00   Max.   :25.00
##     LoanTerm        DTIRatio       Education          EmploymentType
##  Min.   :12.00   Min.   :0.1000   Length:20000       Length:20000
##  1st Qu.:24.00   1st Qu.:0.3000   Class :character   Class :character
##  Median :36.00   Median :0.5000   Mode  :character   Mode  :character
##  Mean   :36.14   Mean   :0.5003
##  3rd Qu.:48.00   3rd Qu.:0.7000
##  Max.   :60.00   Max.   :0.9000
##  MaritalStatus      HasMortgage        HasDependents      LoanPurpose
##  Length:20000       Length:20000       Length:20000       Length:20000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  HasCoSigner           Default
##  Length:20000       Min.   :0.0000
##  Class :character   1st Qu.:0.0000
##  Mode  :character   Median :0.0000
##                     Mean   :0.1163
##                     3rd Qu.:0.0000
##                     Max.   :1.0000
```

```r
# Step 2: Data preprocessing
# Convert categorical variables to factors
loan_data$Education <- as.factor(loan_data$Education)
```

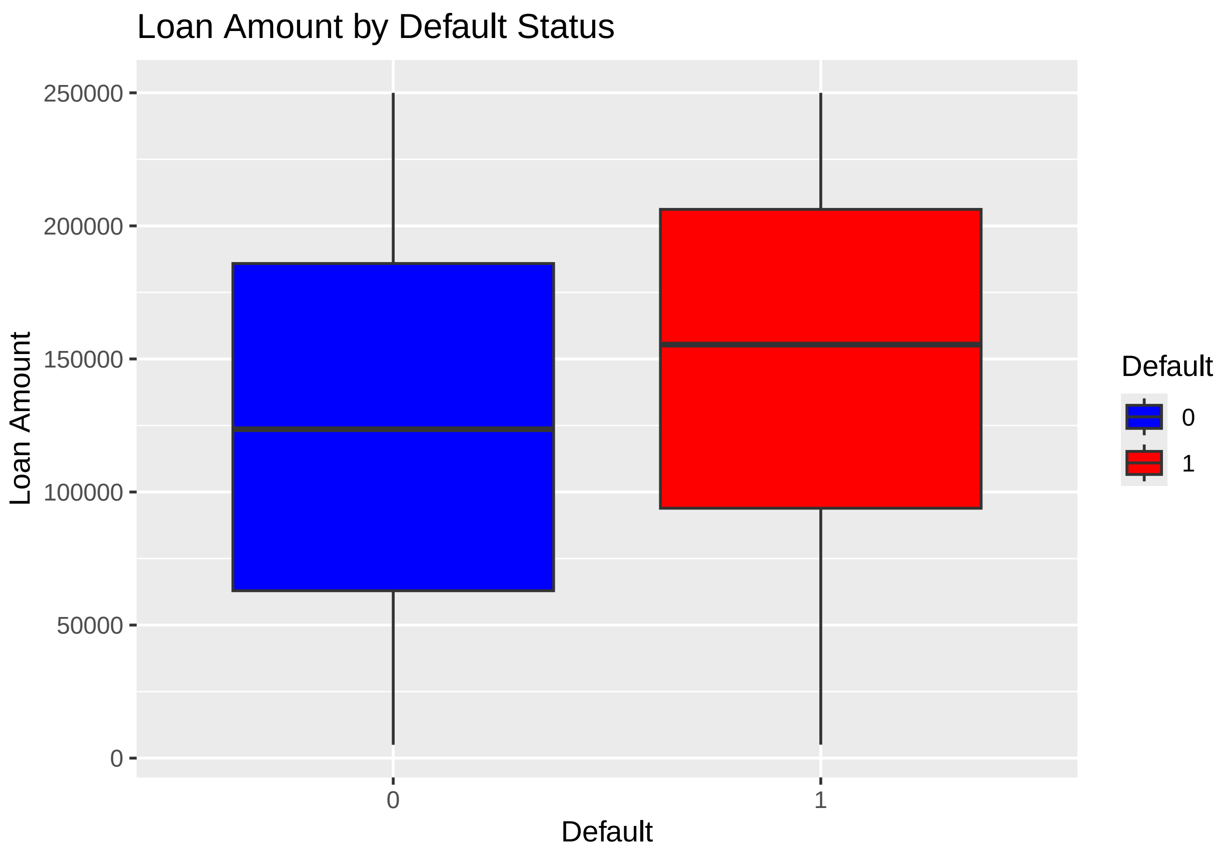
```
loan_data$EmploymentType <- as.factor(loan_data$EmploymentType)
loan_data$MaritalStatus <- as.factor(loan_data$MaritalStatus)
loan_data$LoanPurpose <- as.factor(loan_data$LoanPurpose)
```

```
# Convert binary variables to factors
loan_data$HasMortgage <- as.factor(loan_data$HasMortgage)
loan_data$HasDependents <- as.factor(loan_data$HasDependents)
loan_data$HasCoSigner <- as.factor(loan_data$HasCoSigner)
loan_data$Default <- as.factor(loan_data$Default)
```

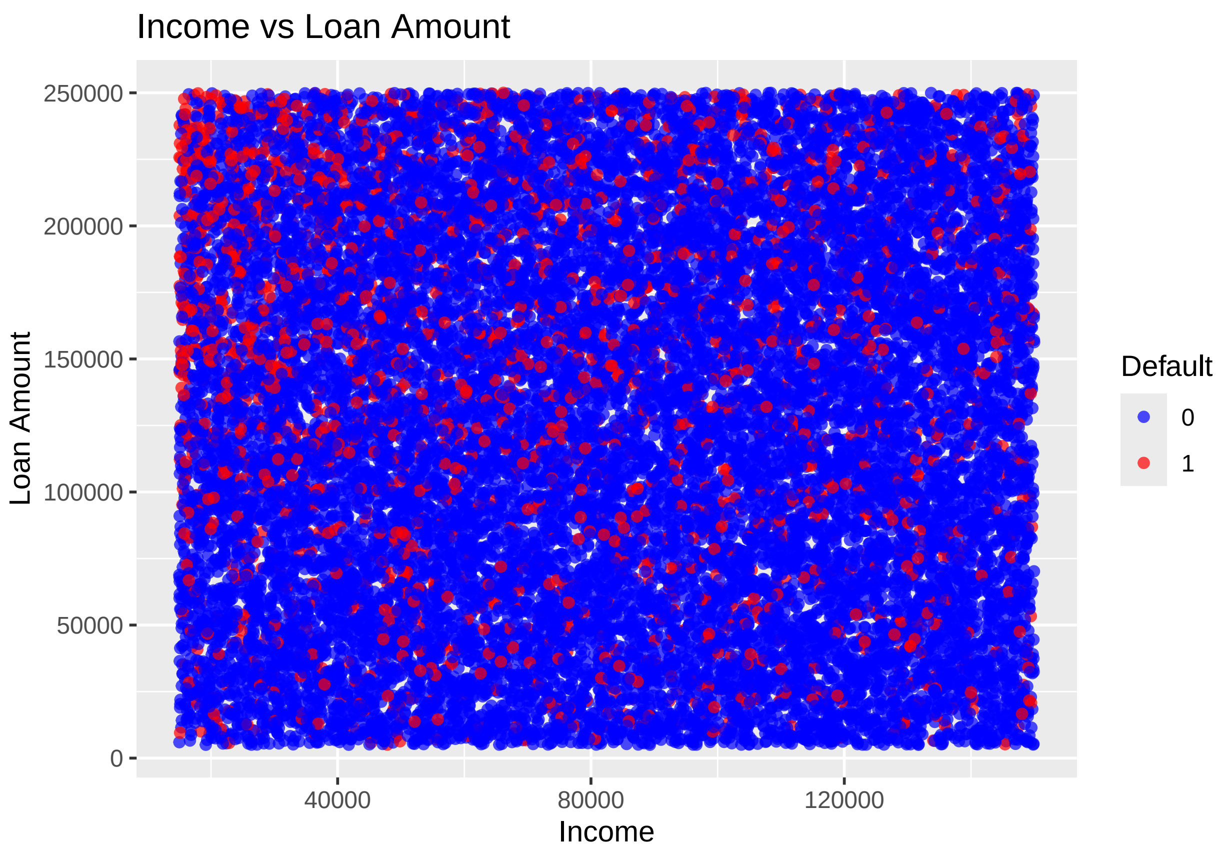
```
# Step 2: Enhanced Data Visualization

# 1. Distribution of Age
p1 <- ggplot(loan_data, aes(x = Age, fill = Default)) +
  geom_histogram(binwidth = 5, position = "dodge") +
  labs(title = "Distribution of Age by Default Status", x = "Age", y = "Count") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"))
print(p1)
```



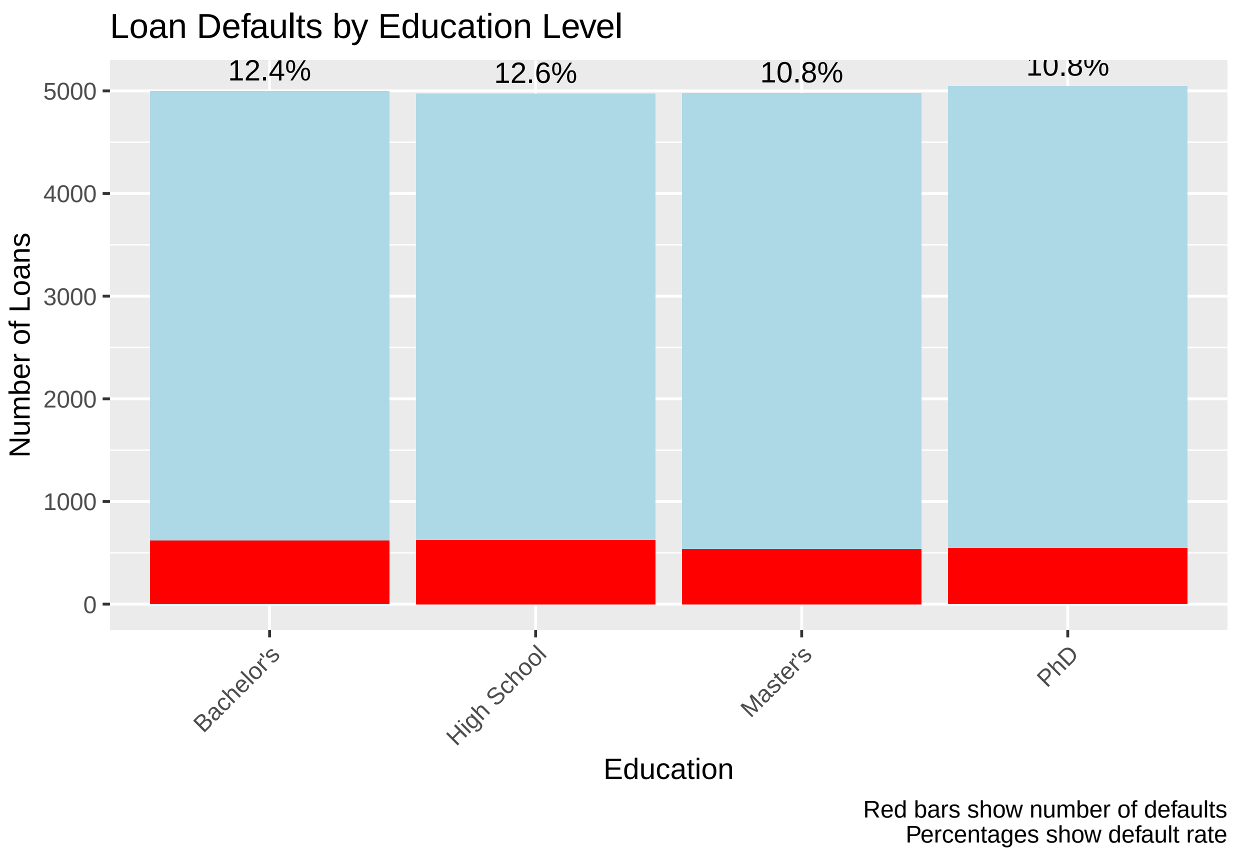
```
# 2. Loan Amount by Default status
p2 <- ggplot(loan_data, aes(x = Default, y = LoanAmount, fill = Default)) +
  geom_boxplot() +
  labs(title = "Loan Amount by Default Status", x = "Default", y = "Loan Amount") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"))
print(p2)
```



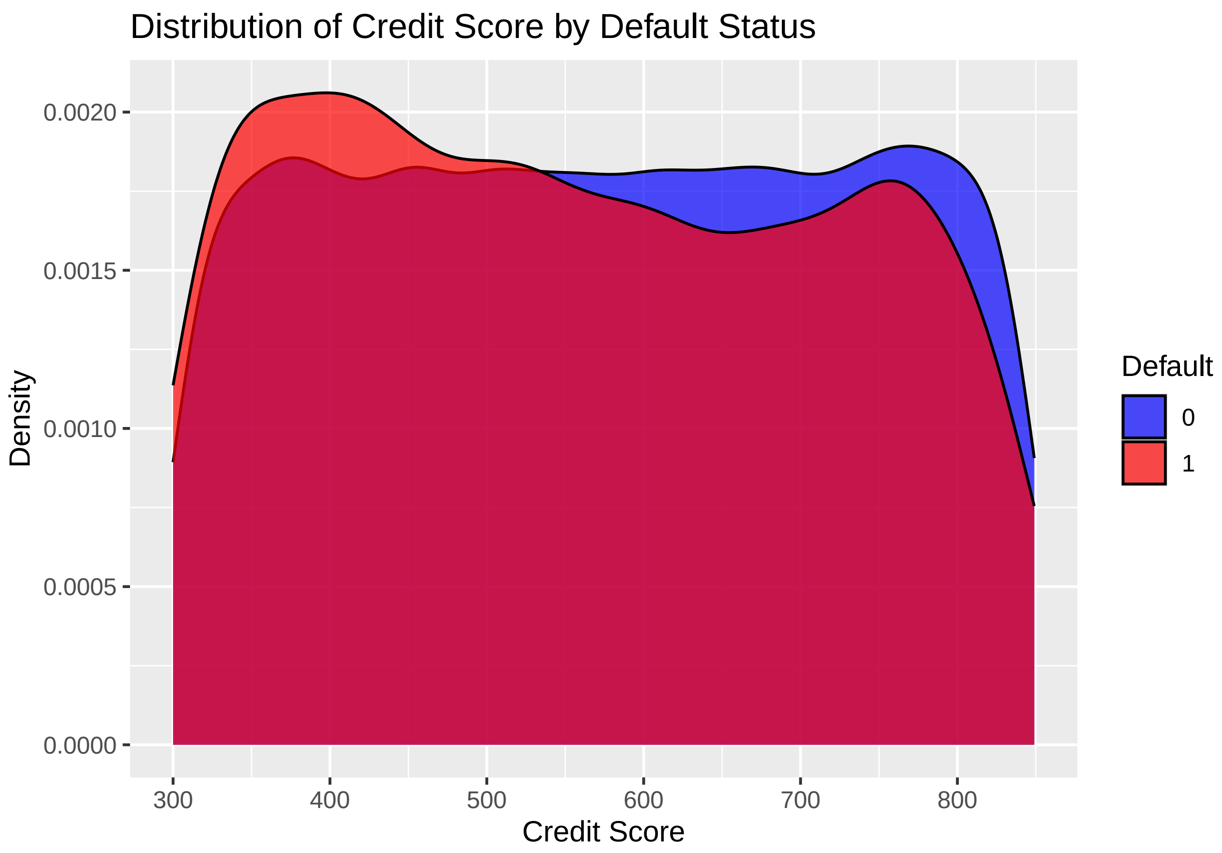
```r
# 3. Income vs LoanAmount, colored by Default status
p3 <- ggplot(loan_data, aes(x = Income, y = LoanAmount, color = Default)) +
  geom_point(alpha = 0.7) +
  labs(title = "Income vs Loan Amount", x = "Income", y = "Loan Amount") +
  scale_color_manual(values = c("0" = "blue", "1" = "red"))
print(p3)
```



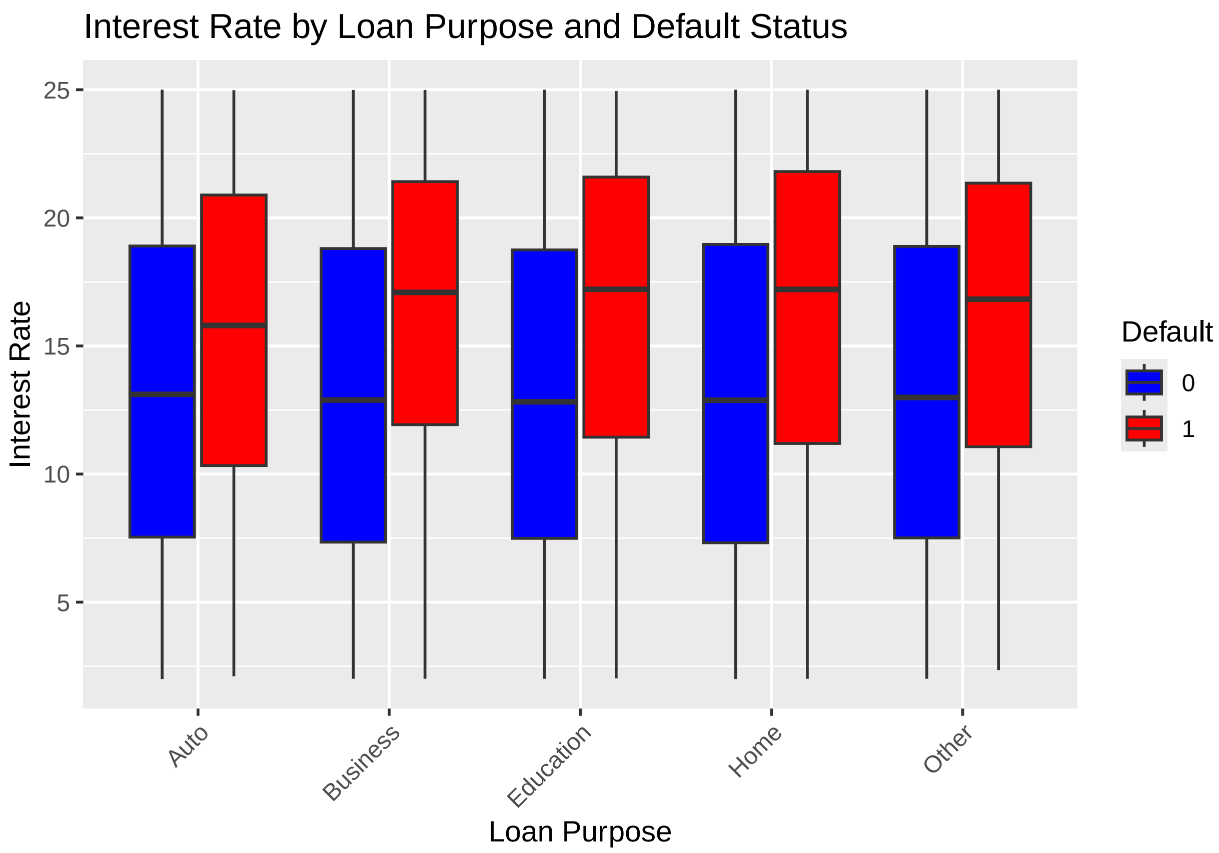
```r
# 4. Default Rate by Education
p4 <- loan_data %>%
  group_by(Education) %>%
  summarise(
    DefaultCount = sum(Default == "1"),
    TotalCount = n(),
    DefaultRate = DefaultCount / TotalCount
  ) %>%
  ggplot(aes(x = Education, y = TotalCount)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  geom_bar(aes(y = DefaultCount), stat = "identity", fill = "red") +
  geom_text(aes(label = sprintf("%.1f%%", DefaultRate*100), y = TotalCount), vjust = -0.5) +
  labs(
    title = "Loan Defaults by Education Level",
    x = "Education",
    y = "Number of Loans",
    caption = "Red bars show number of defaults\nPercentages show default rate"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p4)
```



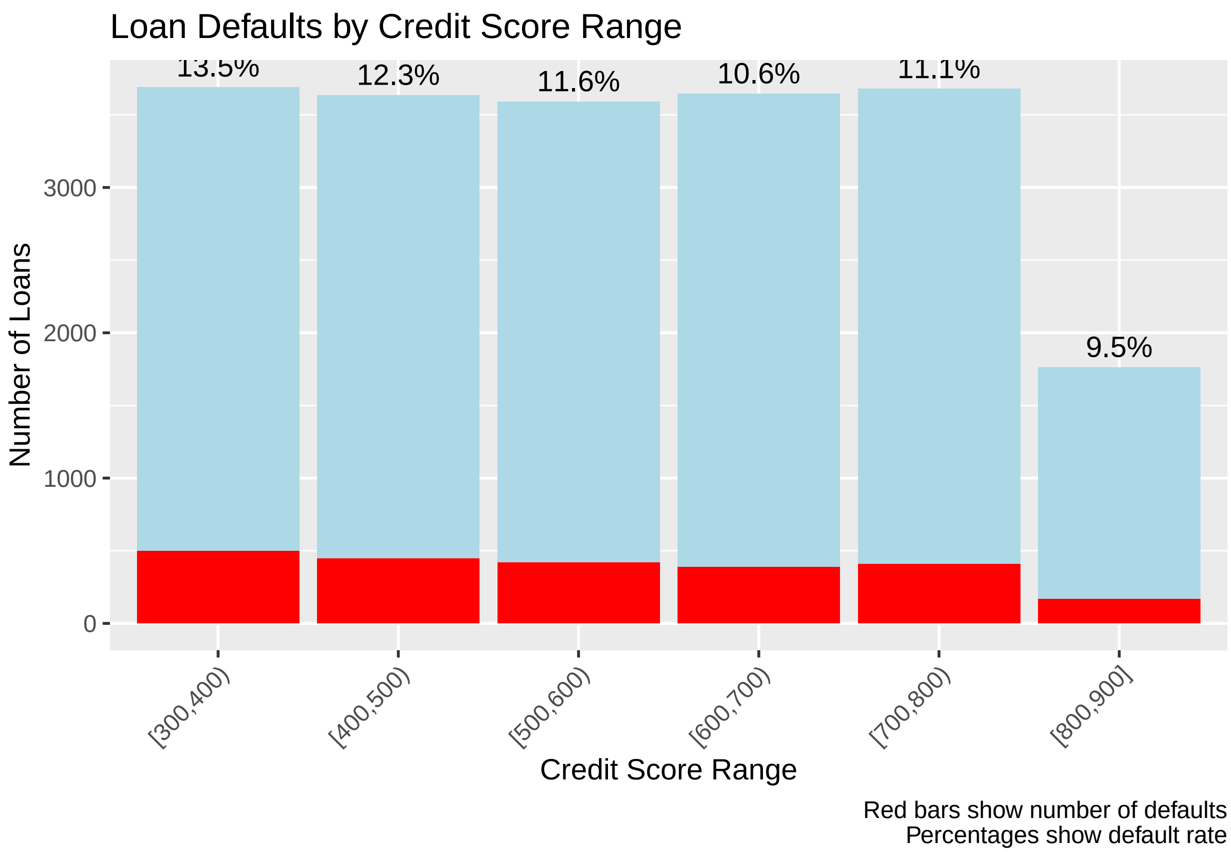
```
# 5. Distribution of Credit Score by Default Status
p5 <- ggplot(loan_data, aes(x = CreditScore, fill = Default)) +
  geom_density(alpha = 0.7) +
  labs(title = "Distribution of Credit Score by Default Status", x = "Credit Score", y = "Density") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"))
print(p5)
```



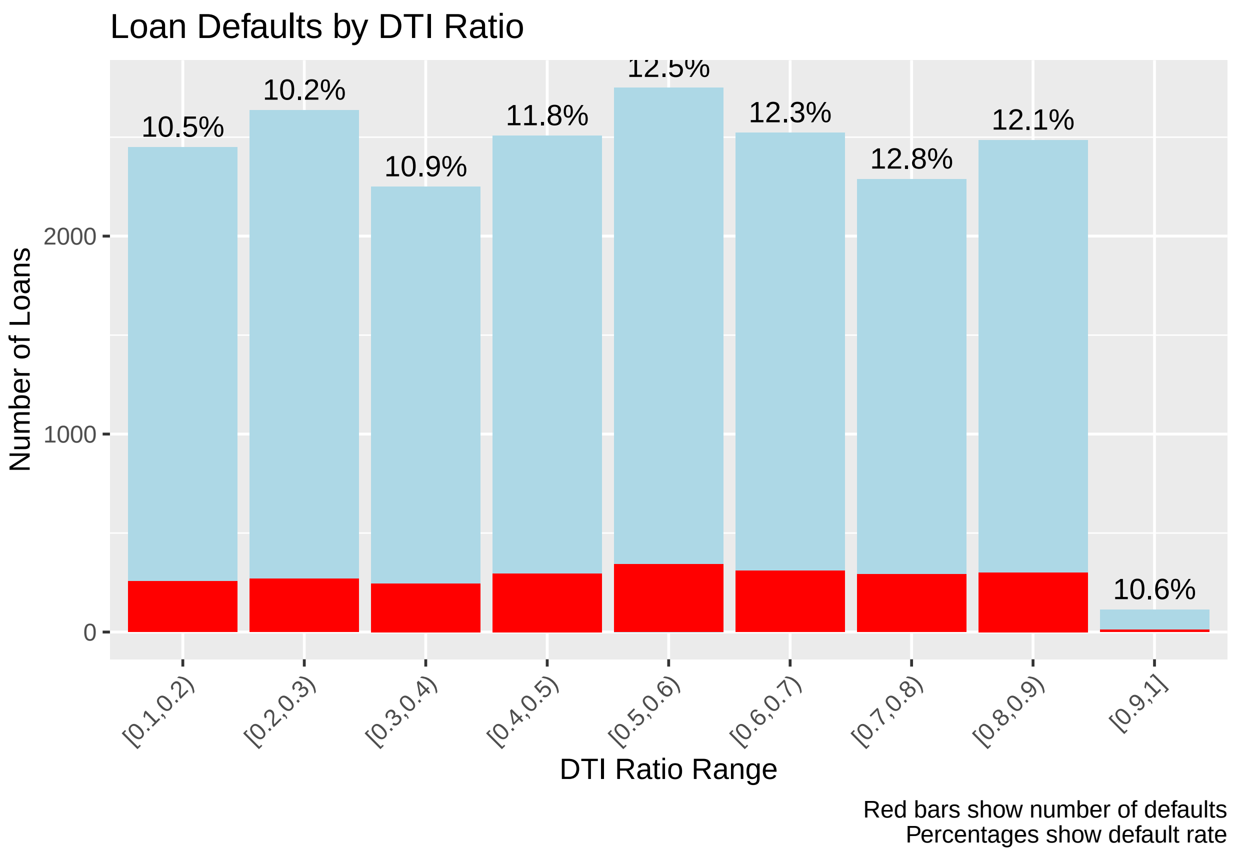
```
# 6. Interest Rate by Loan Purpose
p6 <- ggplot(loan_data, aes(x = LoanPurpose, y = InterestRate, fill = Default)) +
  geom_boxplot() +
  labs(title = "Interest Rate by Loan Purpose and Default Status", x = "Loan Purpose", y = "Interest Ra
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"))
print(p6)
```



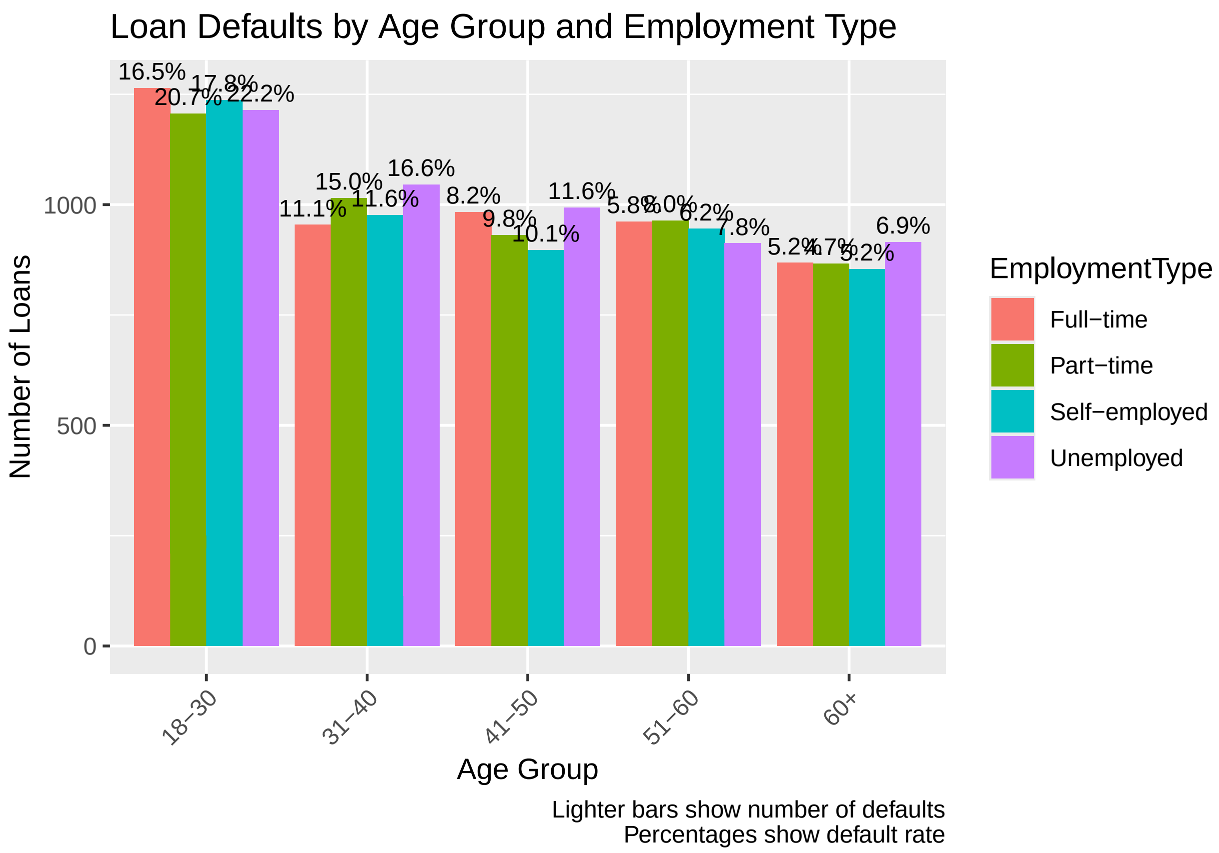
```
# 7. Defaults by Credit Score Range
p7 <- loan_data %>%
  mutate(CreditScoreRange = cut(CreditScore, breaks = seq(300, 900, by = 100),include.lowest = TRUE, ri
  group_by(CreditScoreRange) %>%
  summarise(
    DefaultCount = sum(Default == "1"),
    TotalCount = n(),
    DefaultRate = DefaultCount / TotalCount
  ) %>%
  ggplot(aes(x = CreditScoreRange, y = TotalCount)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  geom_bar(aes(y = DefaultCount), stat = "identity", fill = "red") +
  geom_text(aes(label = sprintf("%.1f%%", DefaultRate*100), y = TotalCount), vjust = -0.5) +
  labs(
    title = "Loan Defaults by Credit Score Range",
    x = "Credit Score Range",
    y = "Number of Loans",
    caption = "Red bars show number of defaults\nPercentages show default rate"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p7)
```



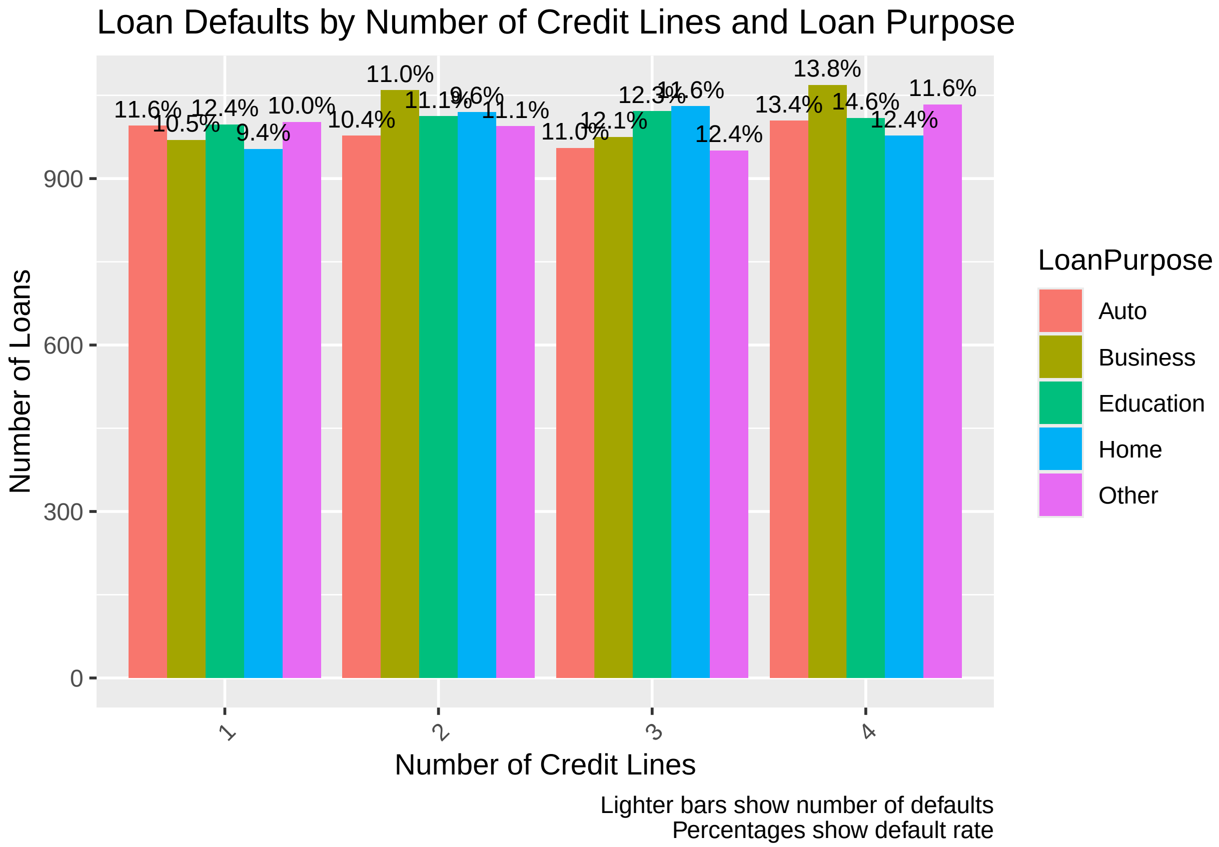
```r
# 8. Defaults by DTI Ratio
p8 <- loan_data %>%
  mutate(DTIRange = cut(DTIRatio, breaks = seq(0, 1, by = 0.1),include.lowest = TRUE, right = FALSE)) %
  group_by(DTIRange) %>%
  summarise(
    DefaultCount = sum(Default == "1"),
    TotalCount = n(),
    DefaultRate = DefaultCount / TotalCount
  ) %>%
  ggplot(aes(x = DTIRange, y = TotalCount)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  geom_bar(aes(y = DefaultCount), stat = "identity", fill = "red") +
  geom_text(aes(label = sprintf("%.1f%%", DefaultRate*100), y = TotalCount), vjust = -0.5) +
  labs(
    title = "Loan Defaults by DTI Ratio",
    x = "DTI Ratio Range",
    y = "Number of Loans",
    caption = "Red bars show number of defaults\nPercentages show default rate"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p8)
```



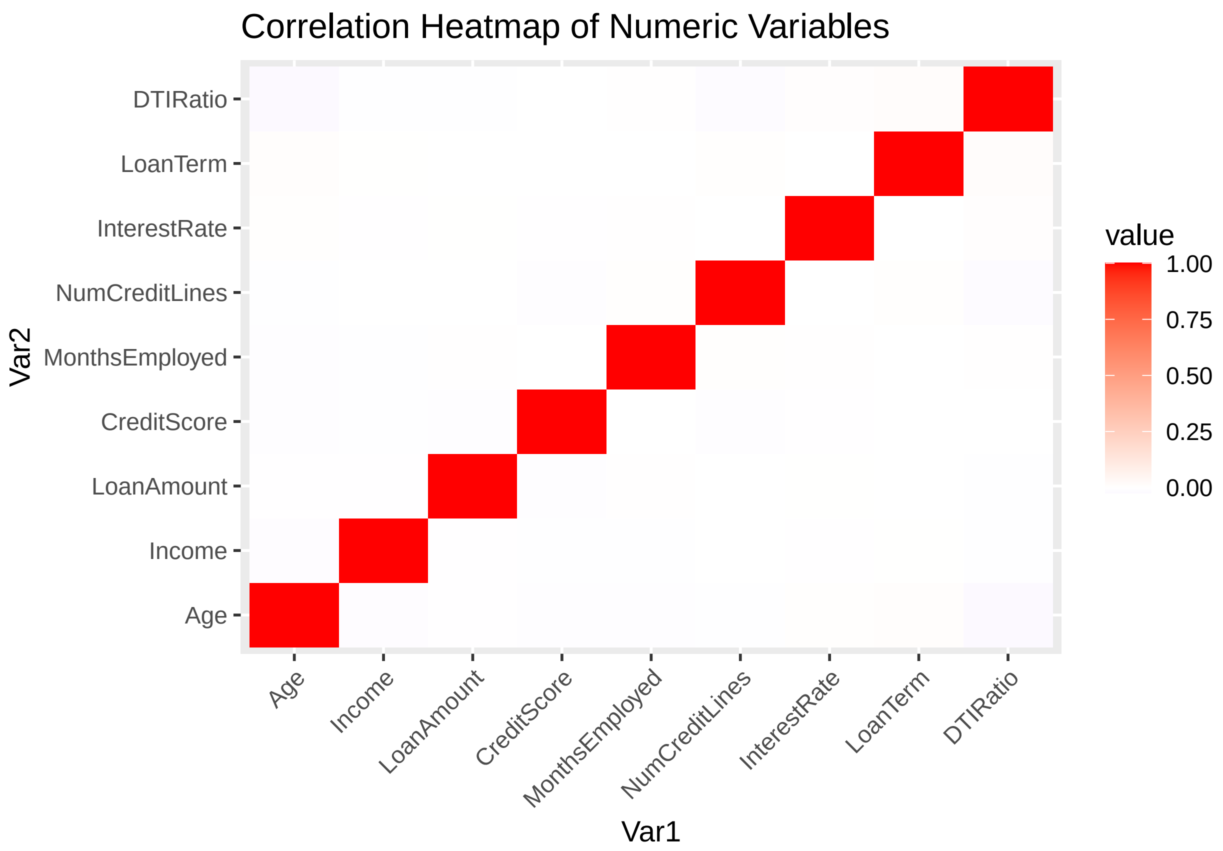
```r
# 9. Defaults by Age Group and Employment Type
p9 <- loan_data %>%
  mutate(AgeGroup = cut(Age, breaks = c(0, 30, 40, 50, 60, 100),
                        labels = c("18-30", "31-40", "41-50", "51-60", "60+"))) %>%
  group_by(AgeGroup, EmploymentType) %>%
  summarise(
    DefaultCount = sum(Default == "1"),
    TotalCount = n(),
    DefaultRate = DefaultCount / TotalCount,
    .groups = 'drop'  # Explicitly drop grouping after summarizing
  ) %>%
  ggplot(aes(x = AgeGroup, y = TotalCount, fill = EmploymentType)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_bar(aes(y = DefaultCount), stat = "identity", position = "dodge", alpha = 0.5) +
  geom_text(aes(label = sprintf("%.1f%%", DefaultRate * 100), y = TotalCount),
            position = position_dodge(width = 0.9), vjust = -0.5, size = 3) +
  labs(
    title = "Loan Defaults by Age Group and Employment Type",
    x = "Age Group",
    y = "Number of Loans",
    caption = "Lighter bars show number of defaults\nPercentages show default rate"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p9)
```



```
# 10. Defaults by Number of Credit Lines and Loan Purpose
p10 <- loan_data %>%
  group_by(NumCreditLines, LoanPurpose) %>%
  summarise(
    DefaultCount = sum(Default == "1"),
    TotalCount = n(),
    DefaultRate = DefaultCount / TotalCount,
    .groups = 'drop'  # Explicitly drop grouping after summarizing
  ) %>%
  ggplot(aes(x = as.factor(NumCreditLines), y = TotalCount, fill = LoanPurpose)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_bar(aes(y = DefaultCount), stat = "identity", position = "dodge", alpha = 0.5) +
  geom_text(aes(label = sprintf("%.1f%%", DefaultRate * 100), y = TotalCount),
            position = position_dodge(width = 0.9), vjust = -0.5, size = 3) +
  labs(
    title = "Loan Defaults by Number of Credit Lines and Loan Purpose",
    x = "Number of Credit Lines",
    y = "Number of Loans",
    caption = "Lighter bars show number of defaults\nPercentages show default rate"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p10)
```



```
# Correlation heatmap for numerical variables
numeric_vars <- loan_data %>% select_if(is.numeric)
cor_matrix <- cor(numeric_vars)
p11 <- ggplot(data = reshape2::melt(cor_matrix)) +
  geom_tile(aes(x = Var1, y = Var2, fill = value)) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Heatmap of Numeric Variables")
print(p11)
```



```
# Step 4: Split the data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(loan_data$Default, p = 0.8, list = FALSE)
train_data <- loan_data[trainIndex,]
test_data <- loan_data[-trainIndex,]
```

```
# Step 5: Define the model formula
model_formula <- Default ~ Age + Income + LoanAmount + CreditScore + MonthsEmployed + NumCreditLines +
```

```
# Step 6: Train models

# Logistic Regression
logit_model <- glm(model_formula, data = train_data, family = "binomial")

# Random Forest
rf_model <- randomForest(model_formula, data = train_data)

# Support Vector Machine
svm_model <- svm(model_formula, data = train_data, kernel = "radial", probability = TRUE)

# XGBoost
train_matrix <- model.matrix(model_formula, data = train_data)[,-1]
test_matrix <- model.matrix(model_formula, data = test_data)[,-1]
dtrain <- xgb.DMatrix(data = train_matrix, label = as.numeric(train_data$Default) - 1)
xgb_model <- xgboost(data = dtrain, nrounds = 100, objective = "binary:logistic")
```

```
## [1]  train-logloss:0.531163
## [2]  train-logloss:0.442379
## [3]  train-logloss:0.387999
## [4]  train-logloss:0.353253
## [5]  train-logloss:0.329644
## [6]  train-logloss:0.313897
## [7]  train-logloss:0.302286
## [8]  train-logloss:0.293986
## [9]  train-logloss:0.286677
## [10] train-logloss:0.281176
## [11] train-logloss:0.276095
## [12] train-logloss:0.271274
## [13] train-logloss:0.266555
## [14] train-logloss:0.262272
## [15] train-logloss:0.258651
## [16] train-logloss:0.255583
## [17] train-logloss:0.251232
## [18] train-logloss:0.250062
## [19] train-logloss:0.246839
## [20] train-logloss:0.244208
## [21] train-logloss:0.240750
## [22] train-logloss:0.237436
## [23] train-logloss:0.236162
## [24] train-logloss:0.233699
## [25] train-logloss:0.232494
## [26] train-logloss:0.231951
## [27] train-logloss:0.228774
## [28] train-logloss:0.226315
## [29] train-logloss:0.223607
## [30] train-logloss:0.220778
## [31] train-logloss:0.218245
## [32] train-logloss:0.215488
## [33] train-logloss:0.214560
## [34] train-logloss:0.213624
## [35] train-logloss:0.212225
## [36] train-logloss:0.211367
## [37] train-logloss:0.210683
## [38] train-logloss:0.209052
## [39] train-logloss:0.207623
## [40] train-logloss:0.205659
## [41] train-logloss:0.204366
## [42] train-logloss:0.202947
## [43] train-logloss:0.201537
## [44] train-logloss:0.200712
## [45] train-logloss:0.200391
## [46] train-logloss:0.199757
## [47] train-logloss:0.198419
## [48] train-logloss:0.197840
## [49] train-logloss:0.196564
## [50] train-logloss:0.194731
## [51] train-logloss:0.192522
## [52] train-logloss:0.192425
## [53] train-logloss:0.191652
## [54] train-logloss:0.190472
## [55] train-logloss:0.188664
## [56] train-logloss:0.186794
## [57] train-logloss:0.184714
## [58] train-logloss:0.183541
## [59] train-logloss:0.182268
## [60] train-logloss:0.180846
## [61] train-logloss:0.178570
## [62] train-logloss:0.176252
## [63] train-logloss:0.175341
## [64] train-logloss:0.173482
## [65] train-logloss:0.171607
## [66] train-logloss:0.170210
## [67] train-logloss:0.169549
## [68] train-logloss:0.168336
## [69] train-logloss:0.167552
## [70] train-logloss:0.166283
## [71] train-logloss:0.164436
## [72] train-logloss:0.163171
## [73] train-logloss:0.161324
## [74] train-logloss:0.159795
## [75] train-logloss:0.158725
## [76] train-logloss:0.158543
## [77] train-logloss:0.157509
## [78] train-logloss:0.155904
## [79] train-logloss:0.154985
## [80] train-logloss:0.154161
## [81] train-logloss:0.152845
## [82] train-logloss:0.150986
## [83] train-logloss:0.150499
## [84] train-logloss:0.149597
## [85] train-logloss:0.147604
## [86] train-logloss:0.146536
## [87] train-logloss:0.145782
## [88] train-logloss:0.144728
## [89] train-logloss:0.144012
## [90] train-logloss:0.143175
## [91] train-logloss:0.142167
## [92] train-logloss:0.140451
## [93] train-logloss:0.139559
## [94] train-logloss:0.137789
## [95] train-logloss:0.136984
## [96] train-logloss:0.136628
## [97] train-logloss:0.135779
## [98] train-logloss:0.135277
## [99] train-logloss:0.134325
## [100]    train-logloss:0.133391
```

```r
# Neural Network
nn_model <- nnet(model_formula, data = train_data, size = 5, maxit = 1000)
```

```
## # weights:  131
## initial  value 11452.396461
## iter  10 value 5864.572403
## iter  20 value 5674.655357
## iter  30 value 5615.678861
## iter  40 value 5579.516906
## iter  50 value 5565.816839
## iter  60 value 5558.912330
## iter  70 value 5557.801268
## iter  80 value 5556.942591
## iter  90 value 5427.281334
## iter 100 value 5308.636645
## iter 110 value 5241.389524
## iter 120 value 5218.803304
## iter 130 value 5200.098556
## iter 140 value 5153.934609
## iter 150 value 5140.665848
## iter 160 value 5136.430441
## iter 170 value 5126.914344
## iter 180 value 5120.376555
## iter 190 value 5119.835185
## iter 200 value 5119.644315
## iter 200 value 5119.644306
## final  value 5119.641803
## converged
```

```r
# Step 7: Make predictions on test data
logit_pred <- predict(logit_model, newdata = test_data, type = "response")
rf_pred <- predict(rf_model, newdata = test_data, type = "prob")[,2]
svm_pred <- predict(svm_model, newdata = test_data, probability = TRUE)
xgb_pred <- predict(xgb_model, newdata = test_matrix)
nn_pred <- predict(nn_model, newdata = test_data, type = "raw")
```

```r
# Function to calculate metrics
calculate_metrics <- function(actual, predicted, threshold = 0.5) {
  predicted_class <- factor(ifelse(predicted > threshold, "1", "0"), levels = levels(actual))
  cm <- confusionMatrix(predicted_class, actual)
  auc <- as.numeric(roc(actual, predicted)$auc)

  data.frame(
    Accuracy = cm$overall["Accuracy"],
    Precision = cm$byClass["Precision"],
    Recall = cm$byClass["Recall"],
    F1_Score = cm$byClass["F1"],
    AUC = auc
  )
}

# Calculate metrics for each model
logit_pred_class <- factor(ifelse(logit_pred > 0.5, "1", "0"), levels = levels(test_data$Default))
logit_metrics <- calculate_metrics(test_data$Default, logit_pred)
```

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

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

```r
rf_pred_class <- factor(ifelse(rf_pred > 0.5, "1", "0"), levels = levels(test_data$Default))
rf_metrics <- calculate_metrics(test_data$Default, rf_pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
svm_pred_class <- factor(ifelse(attr(svm_pred, "probabilities")[,2] > 0.5, "1", "0"), levels = levels(te
svm_metrics <- calculate_metrics(test_data$Default, attr(svm_pred, "probabilities")[,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
xgb_pred_class <- factor(ifelse(xgb_pred > 0.5, "1", "0"), levels = levels(test_data$Default))
xgb_metrics <- calculate_metrics(test_data$Default, xgb_pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

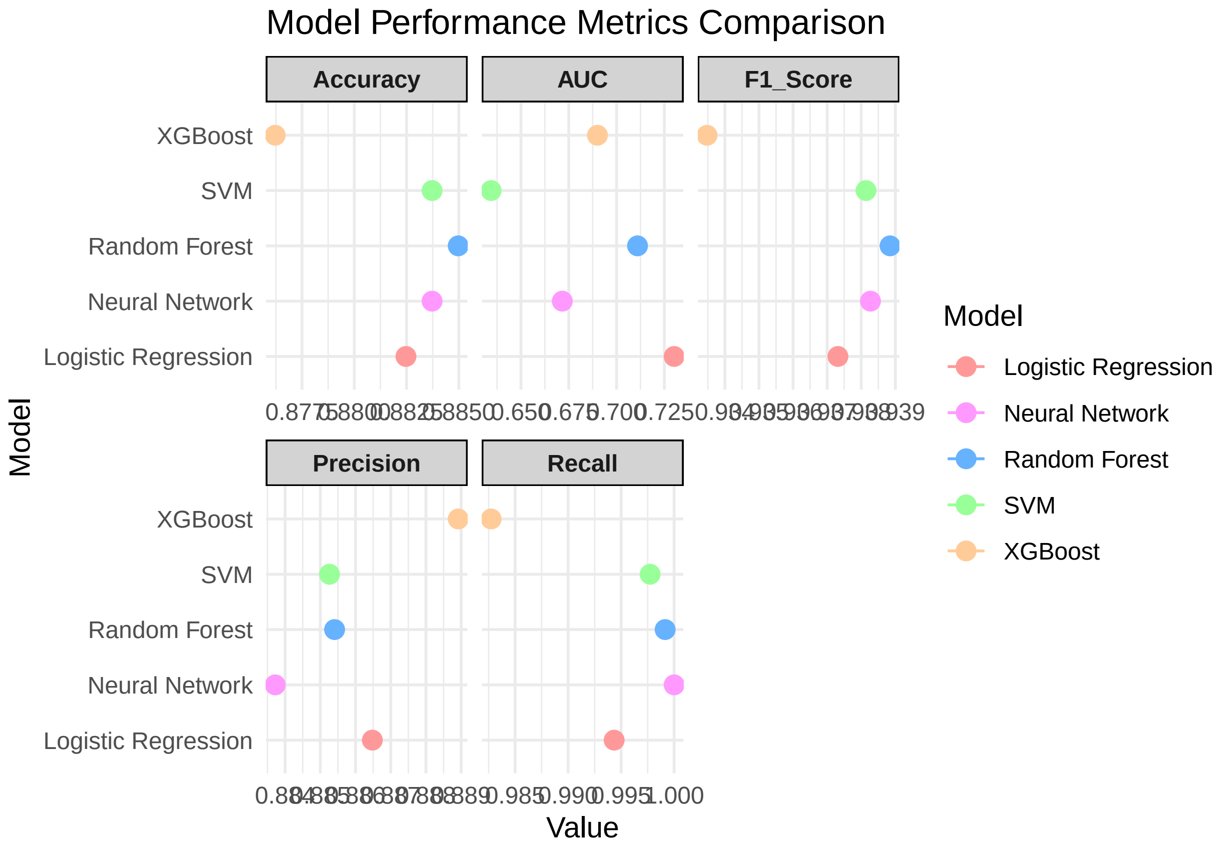
```r
nn_pred_class <- factor(ifelse(as.vector(nn_pred) > 0.5, "1", "0"), levels = levels(test_data$Default))
nn_metrics <- calculate_metrics(test_data$Default, as.vector(nn_pred))
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

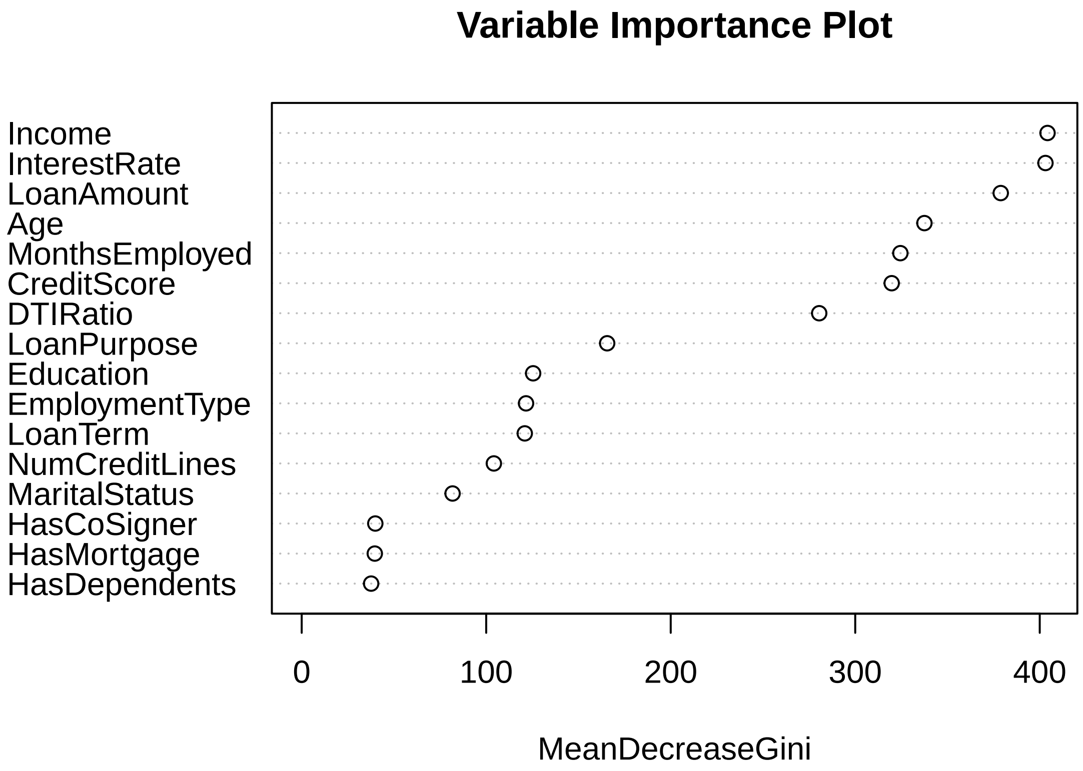
```r
# Combine all metrics
all_metrics <- rbind(
  cbind(Model = "Logistic Regression", logit_metrics),
  cbind(Model = "Random Forest", rf_metrics),
  cbind(Model = "SVM", svm_metrics),
  cbind(Model = "XGBoost", xgb_metrics),
  cbind(Model = "Neural Network", nn_metrics)
)

print(all_metrics)
```

```
##                         Model  Accuracy Precision    Recall  F1_Score       AUC
## Accuracy  Logistic Regression 0.8824706 0.8864783 0.9943407 0.9373166 0.7301185
## Accuracy1       Random Forest 0.8849712 0.8854062 0.9991511 0.9388461 0.7109690
## Accuracy2                 SVM 0.8837209 0.8852624 0.9977363 0.9381402 0.6344086
## Accuracy3             XGBoost 0.8762191 0.8889173 0.9827391 0.9334767 0.6899988
## Accuracy4      Neural Network 0.8837209 0.8837209 1.0000000 0.9382716 0.6716180
```

```r
# Step 9: Visualize model performance
metrics_long <- all_metrics %>%
  pivot_longer(cols = -Model, names_to = "Metric", values_to = "Value")

# Create a custom color palette
model_colors <- c("Logistic Regression" = "#FF9999", # Light red
                  "Random Forest" = "#66B2FF",        # Light blue
                  "SVM" = "#99FF99",                 # Light green
                  "XGBoost" = "#FFCC99",             # Light orange
                  "Neural Network" = "#FF99FF")      # Light purple

# Create the plot
ggplot(metrics_long, aes(x = Value, y = Model, color = Model)) +
  geom_point(size = 3) +
  geom_errorbar(aes(xmin = Value, xmax = Value), width = 0.2) +
  facet_wrap(~ Metric, scales = "free_x", ncol = 3) +
  labs(title = "Model Performance Metrics Comparison",
       x = "Value",
       y = "Model") +
  scale_color_manual(values = model_colors) +
  theme_minimal() +
  theme(strip.background = element_rect(fill = "lightgray"),
        strip.text = element_text(face = "bold"))
```



```
# Step 10: Variable Importance (for Random Forest)
varImpPlot(rf_model, main = "Variable Importance Plot")
```



```r
# Step 11: ROC Curve Comparison
plot(roc(test_data$Default, logit_pred), col = "blue", main = "ROC Curve Comparison")
```

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

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

```r
plot(roc(test_data$Default, rf_pred), col = "red", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(roc(test_data$Default, attr(svm_pred, "probabilities")[,2]), col = "green", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

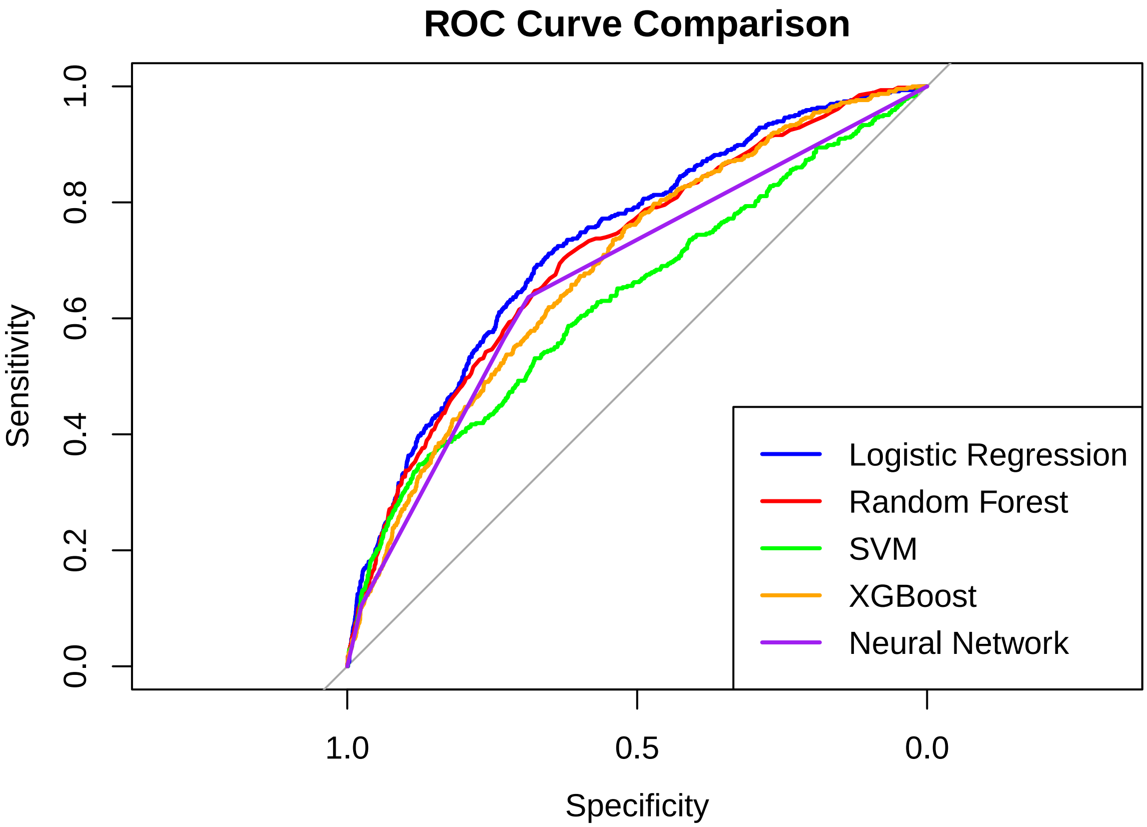
```r
plot(roc(test_data$Default, xgb_pred), col = "orange", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

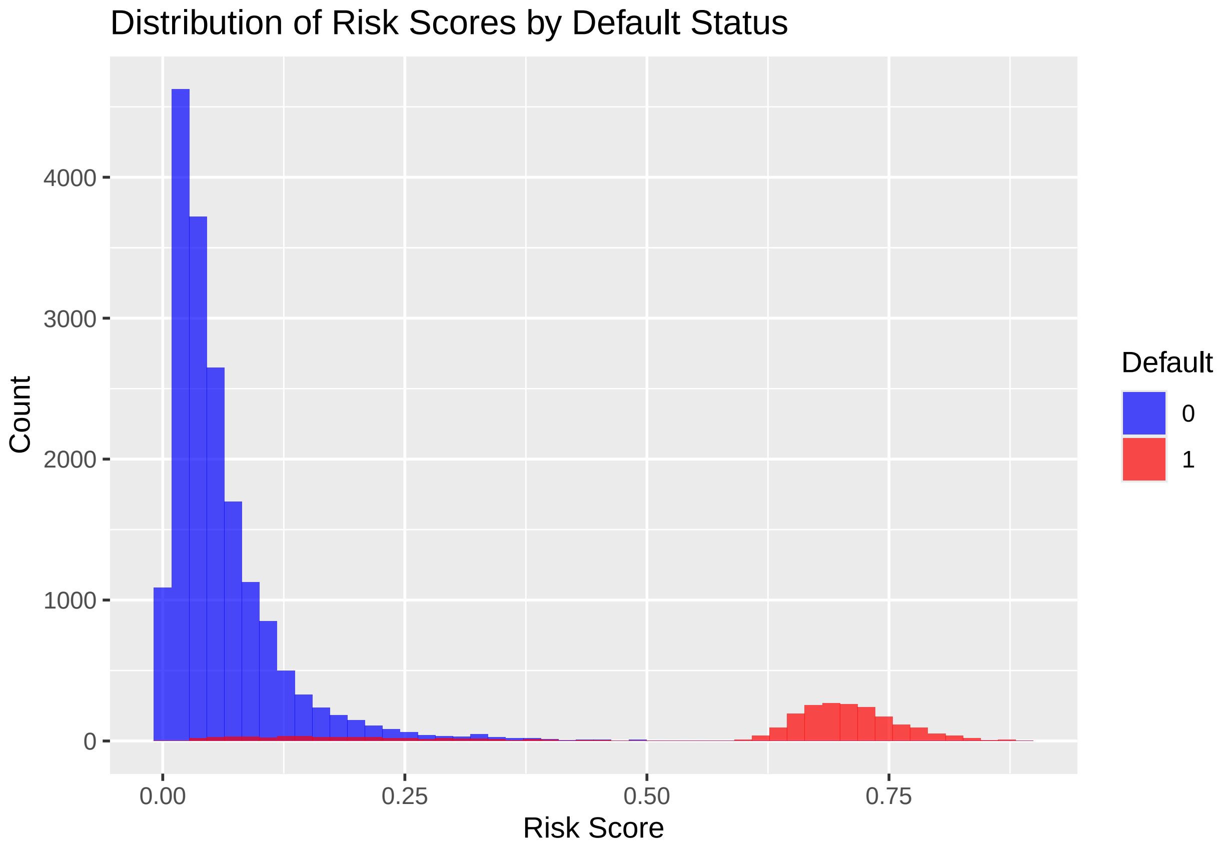
```
plot(roc(test_data$Default, as.vector(nn_pred)), col = "purple", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

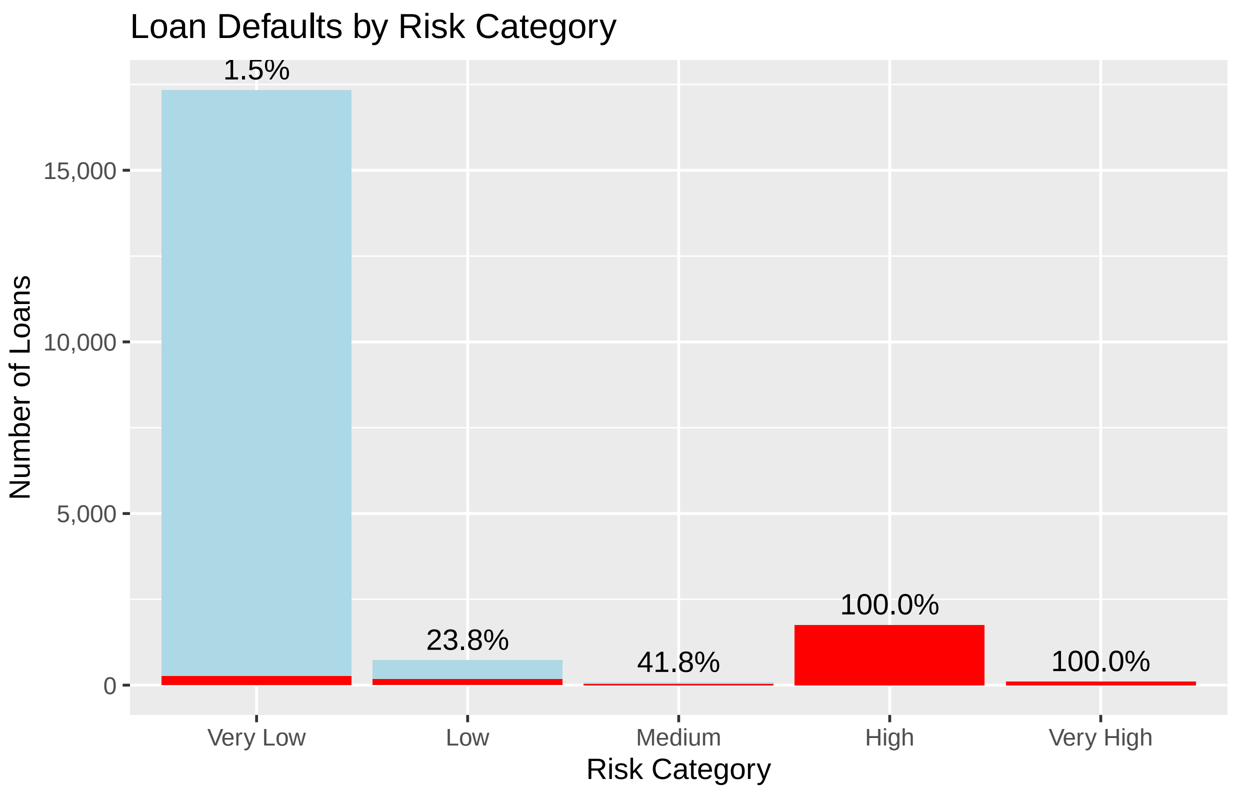
```
legend("bottomright", legend = c("Logistic Regression", "Random Forest", "SVM", "XGBoost", "Neural Netw
       col = c("blue", "red", "green", "orange", "purple"), lwd = 2)
```



```
# Step 12: Enhanced Risk Scoring System
# Calculate risk scores (probability of default)
risk_scores <- predict(rf_model, newdata = loan_data, type = "prob")[,2]

# Create risk categories
risk_categories <- cut(risk_scores,
                       breaks = c(0, 0.2, 0.4, 0.6, 0.8, 1),
                       labels = c("Very Low", "Low", "Medium", "High", "Very High"),include.lowest = TRU

# Add risk scores and categories to the original data
loan_data$RiskScore <- risk_scores
loan_data$RiskCategory <- risk_categories

# Distribution of Risk Scores
p12 <- ggplot(loan_data, aes(x = RiskScore, fill = Default)) +
  geom_histogram(bins = 50, position = "identity", alpha = 0.7) +
  labs(title = "Distribution of Risk Scores by Default Status",
       x = "Risk Score", y = "Count") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"))
print(p12)
```

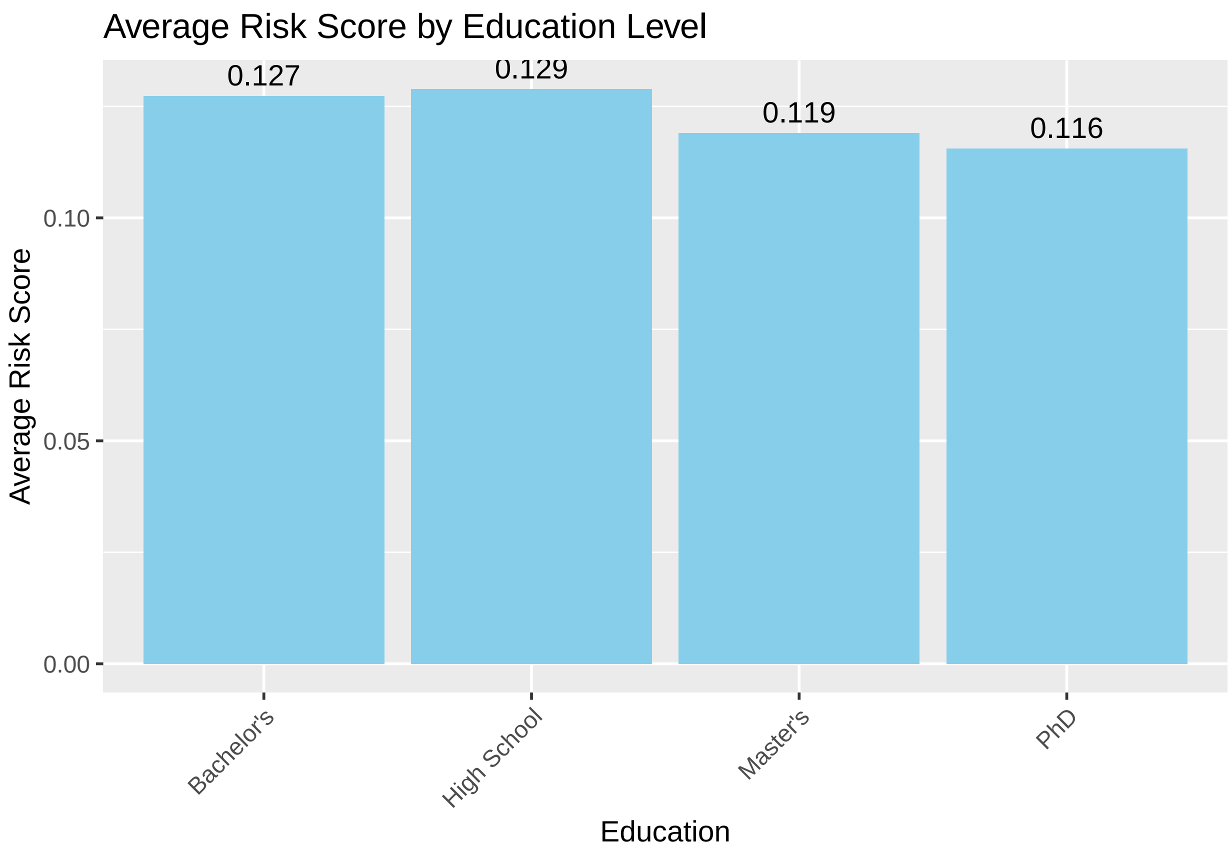


```
# Default Rate by Risk Category
p13 <- loan_data %>%
  group_by(RiskCategory) %>%
  summarise(
    DefaultCount = sum(Default == "1"),
    TotalCount = n(),
    DefaultRate = DefaultCount / TotalCount
  ) %>%
  ggplot(aes(x = RiskCategory, y = TotalCount)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  geom_bar(aes(y = DefaultCount), stat = "identity", fill = "red") +
  geom_text(aes(label = sprintf("%.1f%%", DefaultRate*100), y = TotalCount), vjust = -0.5) +
  scale_y_continuous(labels = scales::comma) +
  labs(
    title = "Loan Defaults by Risk Category",
    x = "Risk Category",
    y = "Number of Loans",
    caption = "Red bars show number of defaults\nPercentages show default rate"
  )
print(p13)
```



Red bars show number of defaults
Percentages show default rate

```r
# Average Risk Score by Categorical Variables
p14 <- loan_data %>%
  group_by(Education) %>%
  summarise(AvgRiskScore = mean(RiskScore)) %>%
  ggplot(aes(x = Education, y = AvgRiskScore)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = sprintf("%.3f", AvgRiskScore)), vjust = -0.5) +
  labs(title = "Average Risk Score by Education Level", x = "Education", y = "Average Risk Score") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
print(p14)
```



Average Risk Score by Education Level
0.127
0.129
0.119
0.116
0.10
0.05
0.00
Average Risk Score
Bachelor's
High School
Master's
PhD
Education