Final\_code\_P3.R

hiraltalati

2024-01-12

# 1. Install and load the ISLR package (if not already installed)  
# install.packages(c("crayon", "ISLR", "ggplot2", "boot", "dplyr"))  
  
# Load the required libraries  
library(ISLR)  
library(ggplot2)  
library(gridExtra)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':  
##   
## combine

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(crayon)

##   
## Attaching package: 'crayon'

## The following object is masked from 'package:ggplot2':  
##   
## %+%

# Load the "Default" dataset  
data("Default", package = "ISLR")  
  
# Store the Default dataset in a dataframe  
default\_df <- Default  
  
# Display the first few rows of the dataframe  
head(default\_df)

## default student balance income  
## 1 No No 729.5265 44361.625  
## 2 No Yes 817.1804 12106.135  
## 3 No No 1073.5492 31767.139  
## 4 No No 529.2506 35704.494  
## 5 No No 785.6559 38463.496  
## 6 No Yes 919.5885 7491.559

# Data Processing:  
# Check the structure of the Default dataset  
str(Default)

## 'data.frame': 10000 obs. of 4 variables:  
## $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ student: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 2 1 1 ...  
## $ balance: num 730 817 1074 529 786 ...  
## $ income : num 44362 12106 31767 35704 38463 ...

# Check for missing and infinite values  
missing\_values\_income <- sum(is.na(Default$income))  
missing\_values\_balance <- sum(is.na(Default$balance))  
infinite\_values\_income <- sum(!is.finite(Default$income))  
infinite\_values\_balance <- sum(!is.finite(Default$balance))  
  
  
# Print the number of missing and infinite values  
cat("Number of missing values in the data: ", missing\_values\_income, "\n")

## Number of missing values in the data: 0

cat("Number of infinite values in the data: ", infinite\_values\_income, "\n")

## Number of infinite values in the data: 0

cat("Number of missing values in the data: ", missing\_values\_balance, "\n")

## Number of missing values in the data: 0

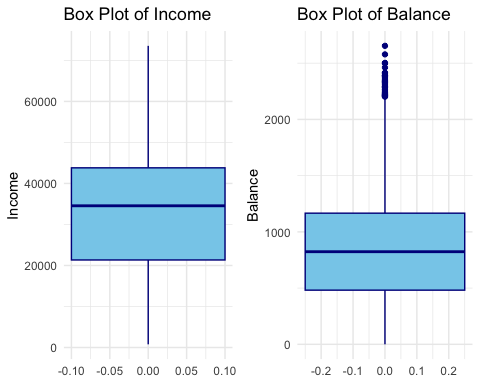
cat("Number of infinite values in the data: ", infinite\_values\_balance, "\n")

## Number of infinite values in the data: 0

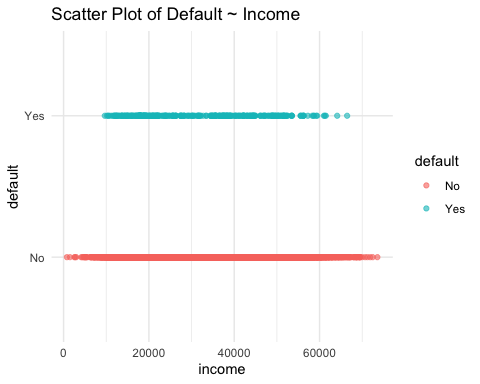
# Check summary statistics for the balance column  
summary(Default)

## default student balance income   
## No :9667 No :7056 Min. : 0.0 Min. : 772   
## Yes: 333 Yes:2944 1st Qu.: 481.7 1st Qu.:21340   
## Median : 823.6 Median :34553   
## Mean : 835.4 Mean :33517   
## 3rd Qu.:1166.3 3rd Qu.:43808   
## Max. :2654.3 Max. :73554

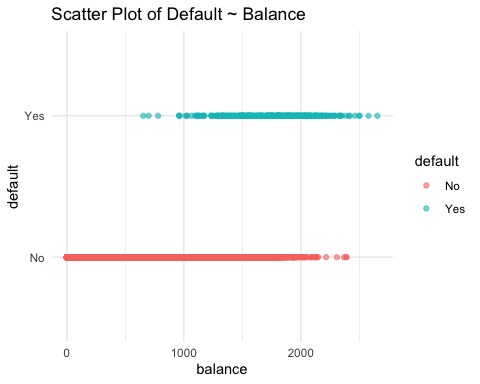
# Box plot: Income  
plot\_income <- ggplot(Default, aes(y = income)) +  
 geom\_boxplot(width = 0.2, fill = "skyblue", color = "darkblue") +  
 labs(title = "Box Plot of Income", y = "Income") +  
 theme\_minimal()   
   
# Box plot: Balance  
plot\_balance <- ggplot(Default, aes(y = balance)) +  
 geom\_boxplot(width = 0.5, fill = "skyblue", color = "darkblue") +  
 labs(title = "Box Plot of Balance", y = "Balance") +  
 theme\_minimal()  
  
# arrange plot side-by-side  
grid.arrange(plot\_income, plot\_balance, ncol = 2)



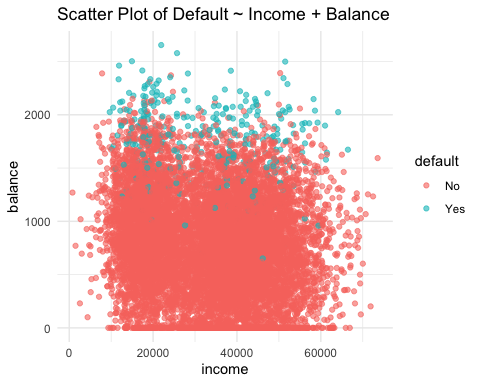
# Scatter plot: Default ~ Income  
ggplot(Default, aes(x = income, y = default, color = default)) +  
 geom\_point(alpha = 0.6) +  
 labs(title = "Scatter Plot of Default ~ Income") +  
 theme\_minimal()



# Scatter plot: Default ~ Balance  
ggplot(Default, aes(x = balance, y = default, color = default)) +  
 geom\_point(alpha = 0.6) +  
 labs(title = "Scatter Plot of Default ~ Balance") +  
 theme\_minimal()



# Scatter plot: Default ~ Income + Balance  
ggplot(Default, aes(x = income, y = balance, color = default)) +  
 geom\_point(alpha = 0.6) +  
 labs(title = "Scatter Plot of Default ~ Income + Balance", x = "income", y = "balance", color = "default") +  
 theme\_minimal()



# Task 1: Logistic regression using glm:  
# Fit logistic regression models  
  
model\_income <- glm(default ~ income, data = Default, family = "binomial")  
model\_balance <- glm(default ~ balance, data = Default, family = "binomial")  
logistic\_model <- glm(default ~ income + balance, data = Default, family = "binomial")  
  
# Display summary of the models  
summary\_income <- summary(model\_income)  
summary\_balance <- summary(model\_balance)  
summary\_logistic <- summary(logistic\_model)  
  
# Create a table for summary results  
summary\_table <- data.frame(  
 Model = c("Income", "Balance", "Income + Balance"),  
 Deviance = c(summary\_income$deviance, summary\_balance$deviance, summary\_logistic$deviance),  
 AIC = c(AIC(model\_income), AIC(model\_balance), AIC(logistic\_model)),  
 BIC = c(BIC(model\_income), BIC(model\_balance), BIC(logistic\_model))  
)  
  
# Print the summary table  
print("Summary Results:")

## [1] "Summary Results:"

print(summary\_table)

## Model Deviance AIC BIC  
## 1 Income 2916.687 2920.687 2935.108  
## 2 Balance 1596.452 1600.452 1614.872  
## 3 Income + Balance 1578.966 1584.966 1606.597

# Task 1.1: Logistic Regression Curve:  
  
# Install and load the pROC library if not already installed  
# install.packages("pROC")  
library(pROC)

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

##   
## Attaching package: 'pROC'

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

# Function to plot ROC curve  
plot\_roc\_curve <- function(predicted\_probs, observed\_values, title) {  
 roc\_curve <- roc(observed\_values, predicted\_probs)  
 auc\_value <- auc(roc\_curve)  
   
 ggplot(roc\_curve, aes(x = 1 - specificity, y = sensitivity)) +  
 geom\_line(color = "blue") +  
 geom\_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +  
 labs(title = paste("ROC Curve -", title, "\nAUC =", round(auc\_value, 3)),  
 x = "1 - Specificity (False Positive Rate)",  
 y = "Sensitivity (True Positive Rate)") +  
 theme\_minimal()  
}  
  
# Check levels of the response variable  
table(Default$default)

##   
## No Yes   
## 9667 333

# Convert response variable to a binary outcome (assuming "No" as the reference level)  
Default$binary\_default <- as.factor(ifelse(Default$default == "Yes", 1, 0))  
  
# Logistic regression curve with ROC curve:  
  
predicted\_probs\_income <- predict(model\_income, type = "response") #for Default ~ Income   
predicted\_probs\_balance <- predict(model\_balance, type = "response") #for Default ~ Balance  
predicted\_probs\_income\_balance <- predict(logistic\_model, type = "response") #for Default ~ Income + Balance  
  
# Plot ROC curve  
roc\_curve\_income\_data <- roc(Default$binary\_default, predicted\_probs\_income)

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

## Setting direction: controls < cases

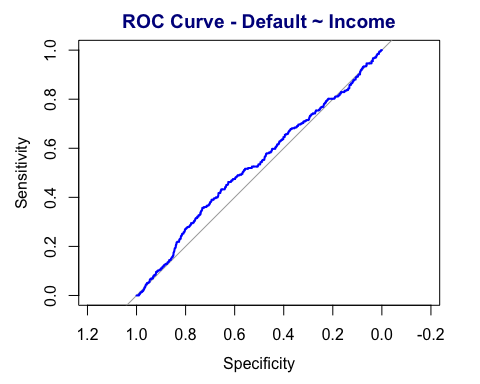
roc\_curve\_balance\_data <- roc(Default$binary\_default, predicted\_probs\_balance)

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

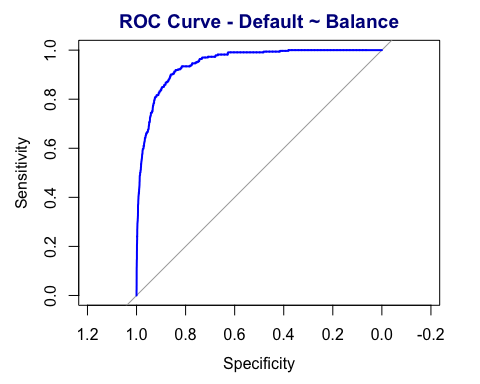
roc\_curve\_income\_balance\_data <- roc(Default$binary\_default, predicted\_probs\_income\_balance)

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

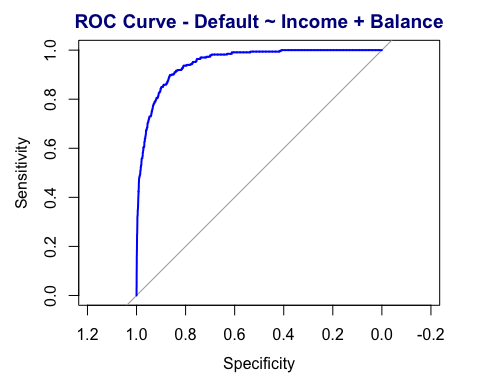
# Plot ROC curve  
plot(roc\_curve\_income\_data, col = "blue", main = "ROC Curve - Default ~ Income", col.main = "darkblue")



plot(roc\_curve\_balance\_data, col = "blue", main = "ROC Curve - Default ~ Balance", col.main = "darkblue")



plot(roc\_curve\_income\_balance\_data, col = "blue", main = "ROC Curve - Default ~ Income + Balance", col.main = "darkblue")



# Task 2: Interpretation of the Logistic models:  
  
# Model Validation  
  
# 1. Install and load the required packages  
# install.packages(c("ISLR", "ggplot2", "pROC", "dplyr", "boot"))  
  
# Load the required libraries  
library(ISLR)  
library(ggplot2)  
library(pROC)  
library(dplyr)  
library(boot)  
  
# Load the "Default" dataset  
data("Default", package = "ISLR")  
  
# Task 1: Logistic regression using glm  
model\_income <- glm(default ~ income, data = Default, family = "binomial")  
model\_balance <- glm(default ~ balance, data = Default, family = "binomial")  
logistic\_model <- glm(default ~ income + balance, data = Default, family = "binomial")  
  
# Display summary of the models  
summary\_income <- summary(model\_income)  
summary\_balance <- summary(model\_balance)  
summary\_logistic <- summary(logistic\_model)  
  
# Create a table for summary results  
summary\_table <- data.frame(  
 Model = c("Income", "Balance", "Income + Balance"),  
 Deviance = c(summary\_income$deviance, summary\_balance$deviance, summary\_logistic$deviance),  
 AIC = c(AIC(model\_income), AIC(model\_balance), AIC(logistic\_model)),  
 BIC = c(BIC(model\_income), BIC(model\_balance), BIC(logistic\_model))  
)  
  
# Print the summary table  
print("Summary Results:")

## [1] "Summary Results:"

print(summary\_table)

## Model Deviance AIC BIC  
## 1 Income 2916.687 2920.687 2935.108  
## 2 Balance 1596.452 1600.452 1614.872  
## 3 Income + Balance 1578.966 1584.966 1606.597

# Task 2: Interpretation of the Logistic models  
coefficients\_income <- coef(model\_income)  
odds\_ratios\_income <- exp(coefficients\_income)  
  
coefficients\_balance <- coef(model\_balance)  
odds\_ratios\_balance <- exp(coefficients\_balance)  
  
coefficients\_income\_balance <- coef(logistic\_model)  
odds\_ratios\_income\_balance <- exp(coefficients\_income\_balance)  
  
# Create a table for interpretation results  
interpretation\_table <- data.frame(  
 Variable = c("Income", "Balance", "Income + Balance"),  
 Coefficients = c(coefficients\_income[2], coefficients\_balance[2], coefficients\_income\_balance[2]),  
 Odds\_Ratios = c(odds\_ratios\_income[2], odds\_ratios\_balance[2], odds\_ratios\_income\_balance[2])  
)  
  
# Print the interpretation table  
print("Interpretation Results:")

## [1] "Interpretation Results:"

print(interpretation\_table)

## Variable Coefficients Odds\_Ratios  
## 1 Income -8.352575e-06 0.9999916  
## 2 Balance 5.498917e-03 1.0055141  
## 3 Income + Balance 2.080898e-05 1.0000208

# Task 2.1: Visualization of Model Validation:  
# Create dataframes for plotting  
  
df\_income <- data.frame(predictor = "income",  
 odds\_ratio = odds\_ratios\_income,  
 lower\_ci = exp(confint(model\_income)[, 1]),  
 upper\_ci = exp(confint(model\_income)[, 2]))

## Waiting for profiling to be done...

## Waiting for profiling to be done...

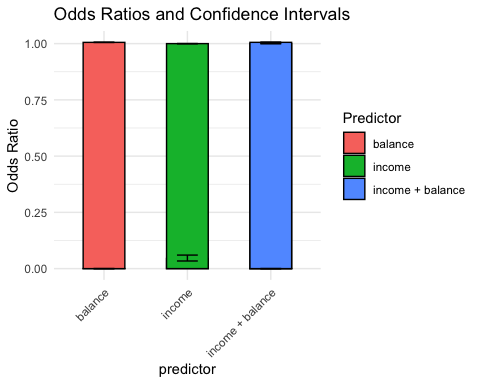
df\_balance <- data.frame(predictor = "balance",  
 odds\_ratio = odds\_ratios\_balance,  
 lower\_ci = exp(confint(model\_balance)[, 1]),  
 upper\_ci = exp(confint(model\_balance)[, 2]))

## Waiting for profiling to be done...  
## Waiting for profiling to be done...

df\_income\_balance <- data.frame(predictor = "income + balance",  
 odds\_ratio = odds\_ratios\_income\_balance,  
 lower\_ci = exp(confint(logistic\_model)[, 1]),  
 upper\_ci = exp(confint(logistic\_model)[, 2]))

## Waiting for profiling to be done...  
## Waiting for profiling to be done...

# Combine dataframes  
df\_plot <- rbind(df\_income, df\_balance, df\_income\_balance)  
  
# Plotting  
ggplot(df\_plot, aes(x = predictor, y = odds\_ratio, ymin = lower\_ci, ymax = upper\_ci, fill = predictor)) +  
 geom\_col(position = "dodge", width = 0.5, color = "black") +  
 geom\_errorbar(position = position\_dodge(width = 0.5), width = 0.25, color = "black") +  
 labs(title = "Odds Ratios and Confidence Intervals", y = "Odds Ratio", fill = "Predictor") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



# Task 3: Bootstrapping method for logistic regression analysis:  
  
# Load the required packages  
library(boot)  
  
# Create a function to obtain bootstrap coefficients  
bootstrap\_func <- function(data, indices, model\_formula) {  
 sampled\_data <- data[indices, ]  
 fit <- glm(formula(model\_formula), data = sampled\_data, family = "binomial")  
 return(coef(fit))  
}  
  
# Set the seed for reproducibility  
set.seed(123)  
  
# Number of bootstrap samples  
num\_boot\_samples <- 1000  
  
# Income model  
bootstrap\_results\_income <- boot(  
 data = Default,  
 statistic = bootstrap\_func,  
 R = num\_boot\_samples, # Increase the number of bootstrap samples  
 model = 'default ~ income'  
)  
  
# Calculate bootstrap confidence intervals  
boot\_ci\_income <- boot.ci(bootstrap\_results\_income, type = "perc")  
  
  
# Balance model  
bootstrap\_results\_balance <- boot(  
 data = Default,  
 statistic = bootstrap\_func,  
 R = num\_boot\_samples, # Increase the number of bootstrap samples  
 model = 'default ~ balance'  
)  
  
# Calculate bootstrap confidence intervals  
boot\_ci\_balance <- boot.ci(bootstrap\_results\_balance, type = "perc")  
  
# Income + Balance model  
bootstrap\_results\_income\_balance <- boot(  
 data = Default,  
 statistic = bootstrap\_func,  
 R = num\_boot\_samples, # Increase the number of bootstrap samples  
 model = 'default ~ income + balance'  
)  
  
# Calculate bootstrap confidence intervals  
boot\_ci\_income\_balance <- boot.ci(bootstrap\_results\_income\_balance, type = "perc")  
  
  
# Create a table for bootstrap results  
bootstrap\_table <- data.frame(  
 Model = c("Income", "Balance", "Income + Balance"),  
 Lower\_CI = c(boot\_ci\_income$perc[4], boot\_ci\_balance$perc[4], boot\_ci\_income\_balance$perc[4]),  
 Upper\_CI = c(boot\_ci\_income$perc[5], boot\_ci\_balance$perc[5], boot\_ci\_income\_balance$perc[5])  
)  
  
# Print the bootstrap results table  
print("\nBootstrap Confidence Intervals:")

## [1] "\nBootstrap Confidence Intervals:"

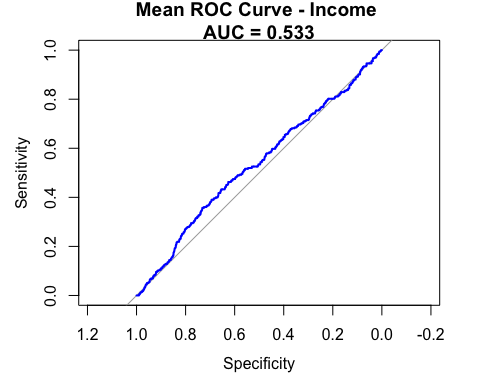
print(bootstrap\_table)

## Model Lower\_CI Upper\_CI  
## 1 Income -3.404395 -2.805551  
## 2 Balance -11.364361 -9.967937  
## 3 Income + Balance -12.479442 -10.799186

# Task 3.2: Logistic Regression Curve after bootstrapping:  
  
# Function to plot ROC curve after bootstrapping  
plot\_roc\_curve\_bootstrap <- function(predicted\_probs\_vector, labels, model\_name) {  
 # Check if predicted\_probs\_vector is a numeric vector  
 if (!is.numeric(predicted\_probs\_vector)) {  
 stop("'predicted\_probs\_vector' must be a numeric vector.")  
 }  
   
 # Calculate mean ROC curve  
 mean\_roc\_curve <- pROC::roc(labels, predicted\_probs\_vector)  
 auc\_value <- pROC::auc(mean\_roc\_curve)  
   
 # Plot mean ROC curve  
 plot(mean\_roc\_curve, col = "blue", main = paste("Mean ROC Curve -", model\_name, "\nAUC =", round(auc\_value, 3)))  
}  
  
# Example usage:  
  
# Plot ROC curve after bootstrapping for Default ~ Income  
roc\_curve\_bootstrap\_income <- plot\_roc\_curve\_bootstrap(  
 predicted\_probs\_income,  
 labels = Default$default,  
 model\_name = "Income"  
)

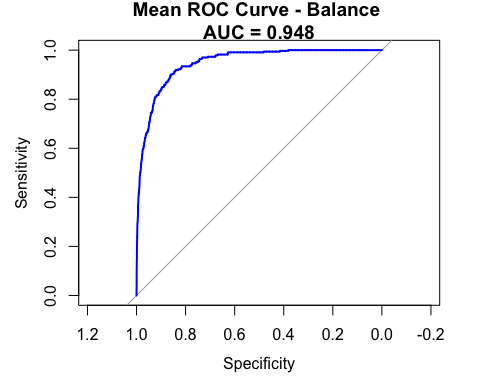
## Setting levels: control = No, case = Yes

## Setting direction: controls < cases



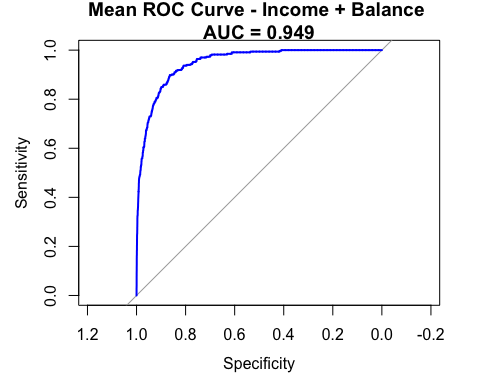
# Plot ROC curve after bootstrapping for Default ~ Balance  
roc\_curve\_bootstrap\_balance <- plot\_roc\_curve\_bootstrap(  
 predicted\_probs\_balance,  
 labels = Default$default,  
 model\_name = "Balance"  
)

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases



# Plot ROC curve after bootstrapping for Default ~ Income + Balance  
roc\_curve\_bootstrap\_income\_balance <- plot\_roc\_curve\_bootstrap(  
 predicted\_probs\_income\_balance,  
 labels = Default$default,  
 model\_name = "Income + Balance"  
)

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases



# Display the plots  
print(roc\_curve\_bootstrap\_income)

##   
## Call:  
## roc.default(response = labels, predictor = predicted\_probs\_vector)  
##   
## Data: predicted\_probs\_vector in 9667 controls (labels No) < 333 cases (labels Yes).  
## Area under the curve: 0.5327

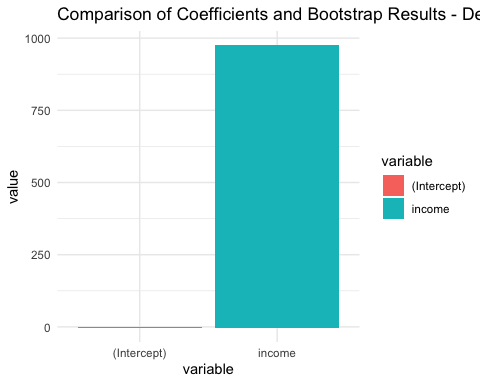
print(roc\_curve\_bootstrap\_balance)

##   
## Call:  
## roc.default(response = labels, predictor = predicted\_probs\_vector)  
##   
## Data: predicted\_probs\_vector in 9667 controls (labels No) < 333 cases (labels Yes).  
## Area under the curve: 0.948

print(roc\_curve\_bootstrap\_income\_balance)

##   
## Call:  
## roc.default(response = labels, predictor = predicted\_probs\_vector)  
##   
## Data: predicted\_probs\_vector in 9667 controls (labels No) < 333 cases (labels Yes).  
## Area under the curve: 0.9491

# Task 3.3: Compare before and after bootstrap results:  
  
# Visualize the differences of before and after bootstrapping:  
# Extract original coefficients  
  
original\_coefficients\_income <- coef(model\_income)  
original\_coefficients\_balance <- coef(model\_balance)  
original\_coefficients\_income\_balance <- coef(logistic\_model)  
  
# Combine coefficients and bootstrap results  
df\_comparison\_income <- data.frame(  
 variable = c(names(original\_coefficients\_income), rep("income", length(boot\_ci\_income$percent))),  
 value = c(original\_coefficients\_income, rep(NA, length(boot\_ci\_income$percent)))  
)  
  
df\_comparison\_balance <- data.frame(  
 variable = c(names(original\_coefficients\_balance), rep("balance", length(boot\_ci\_balance$percent))),  
 value = c(original\_coefficients\_balance, rep(NA, length(boot\_ci\_balance$percent)))  
)  
  
df\_comparison\_income\_balance <- data.frame(  
 variable = c(names(original\_coefficients\_income\_balance), rep(c("income", "balance"), each = length(boot\_ci\_income\_balance$percent))),  
 value = c(original\_coefficients\_income\_balance, rep(NA, 2 \* length(boot\_ci\_income\_balance$percent)))  
)  
  
# Fill in the rows corresponding to bootstrap results  
df\_comparison\_income$value[length(names(original\_coefficients\_income)) + 1:length(boot\_ci\_income$percent)] <- boot\_ci\_income$percent  
df\_comparison\_balance$value[length(names(original\_coefficients\_balance)) + 1:length(boot\_ci\_balance$percent)] <- boot\_ci\_balance$percent  
  
# Fill in the rows corresponding to bootstrap results for income  
df\_comparison\_income\_balance$value[1:length(original\_coefficients\_income\_balance)] <- original\_coefficients\_income\_balance  
df\_comparison\_income\_balance$value[length(original\_coefficients\_income\_balance) + 1:length(boot\_ci\_income\_balance$percent)] <- boot\_ci\_income\_balance$percent  
  
# Fill in the rows corresponding to bootstrap results for balance  
df\_comparison\_income\_balance$value[length(original\_coefficients\_income\_balance) + length(boot\_ci\_income\_balance$percent) + 1: length(boot\_ci\_income\_balance$percent)] <- NA  
  
# Plotting  
ggplot(df\_comparison\_income, aes(x = variable, y = value, fill = variable)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 labs(title = "Comparison of Coefficients and Bootstrap Results - Default ~ Income") +  
 theme\_minimal()



ggplot(df\_comparison\_balance, aes(x = variable, y = value, fill = variable)) +  
 geom\_bar(stat = "identity", position = "dodge")

