Other Methods for Dealing with Missingness

Contents

1 Introduction

We will explore other methods of dealing with missingness, and compare them. We will run each method, and then provide a comparison at the end.

1.1 Requirements

We load the required packages with the code below.

```r
# List of required packages
packages <- c("rpart", "caret", "pROC", "ggplot2", "dplyr")

# Function to check if a package is installed, and install it if it's missing
install_if_missing <- function(pkg) {
  if (!require(pkg, character.only = TRUE)) {
    install.packages(pkg, dependencies = TRUE)
  }
}

# Install any missing packages
invisible(lapply(packages, install_if_missing))
```

```
## Loading required package: rpart

## Loading required package: caret

## Loading required package: ggplot2
## Loading required package: lattice

## Loading required package: pROC

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

##
## Attaching package: 'pROC'

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

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

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

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

```r
# Load the necessary packages
lapply(packages, library, character.only = TRUE)
```

```
## [[1]]
##  [1] "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"   "rpart"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
##
## [[2]]
##  [1] "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"   "rpart"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
##
## [[3]]
##  [1] "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"   "rpart"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
##
## [[4]]
##  [1] "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"   "rpart"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
##
## [[5]]
##  [1] "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"   "rpart"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
```

```r
# rpart      - Recursive partitioning for decision trees
# caret      - For confusion matrix and precision/recall calculations
# pROC       - For ROC curve
# ggplot2    - For plotting
# dplyr      - For data manipulation
```

2 Other Methods to Deal with Missingness

2.1 Analysis with Missingness Encoded as ?

This treats missingness as its own category.

```r
# Get the current working directory
current_dir <- getwd()
cat("Current directory:", current_dir, "\n")
```

```
## Current directory: C:/Users/megar/Documents/Code/Data Science Toolbox/dst-group-project-1/report
```

```r
# Get the parent directory
parent_dir <- dirname(current_dir)
cat("Parent directory:", parent_dir, "\n")
```

```
## Parent directory: C:/Users/megar/Documents/Code/Data Science Toolbox/dst-group-project-1
```

```r
# Set the working directory to the parent directory
setwd(parent_dir)
cat("New working directory:", getwd(), "\n")
```

```
## New working directory: C:/Users/megar/Documents/Code/Data Science Toolbox/dst-group-project-1
```

```r
# Load the training and test datasets
X_train <- read.csv("data/X_train_smote.csv", row.names = 1)  # Load the SMOTE
↪  transformed training features
y_train <- read.csv("data/y_train_smote.csv", row.names = 1)  # Load the SMOTE
↪  transformed training labels
X_test <- read.csv("data/X_test.csv", row.names = 1)     # Load the test features
y_test <- read.csv("data/y_test.csv", row.names = 1)     # Load the test labels

# Combine X_train and y_train into one data frame for rpart
train_data <- cbind(X_train, y_train)

# Drop the "education" column from the training data if it exists
train_data <- train_data[, !names(train_data) %in% "education"]

# Replace NAs with "?" for both training and test datasets
train_data[is.na(train_data)] <- "?"
X_test[is.na(X_test)] <- "?"
```

```
# Fit the classification tree using rpart
fit <- rpart(income ~ ., data = train_data, method = "class", control = rpart.control(cp
↪  = 1e-6))

# Get the cost-complexity pruning table and identify the best cp based on minimum xerror
cptable <- fit$cptable
best_cp <- cptable[which.min(cptable[,"xerror"]), "CP"]
cat("Best CP:", best_cp, "\n")
```

```
## Best CP: 9.903278e-05
```

```
# Prune the tree using the best cp
pruned_tree <- prune(fit, cp = best_cp)

# Get predicted probabilities for the positive class
pred_probs <- predict(pruned_tree, X_test, type = "prob")[, "<=50K"]  # Adjust based on
↪  your positive class

# Compute ROC curve for the model with predictions
roc_curve_model_encoded <- roc(y_test[, 1], pred_probs, levels = c("<=50K", ">50K"))
```

```
## Setting direction: controls > cases
```

```
# Calculate AUC for the model
auc_value_model_encoded <- auc(roc_curve_model_encoded)

# Print AUC value for the model
cat("AUC for Model (Missing Data Encoded):", auc_value_model_encoded, "\n")
```

```
## AUC for Model (Missing Data Encoded): 0.8885526
```

```
# Create a data frame for the model's ROC data
roc_data_model_encoded <- data.frame(
  FPR = 1 - roc_curve_model_encoded$specificities,  # False Positive Rate
  TPR = roc_curve_model_encoded$sensitivities         # True Positive Rate
)
```

2.2 Complete Case Analysis

We start with a simple method of dealing with missing data: simply deleting all missing data. This is not really a good strategy here because we have about 50000 observations and 3000 missing some values. This was not considered a negligible amount of missingness, but we will just consider this as a baseline.

```
# Remove rows with missing data from the training and test sets
train_data[train_data == "?"] <- NA
train_data <- na.omit(train_data)

# Fit the classification tree using rpart
fit <- rpart(income ~ ., data = train_data, method = "class", control = rpart.control(cp
↪  = 1e-6))
```

```
# Get the cost-complexity pruning table and identify the best cp based on minimum xerror
cptable <- fit$cptable
best_cp <- cptable[which.min(cptable[,"xerror"]), "CP"]
cat("Best CP:", best_cp, "\n")
```

```
## Best CP: 9.078005e-05
```

```
# Prune the tree using the best cp
pruned_tree <- prune(fit, cp = best_cp)

# Get predicted probabilities for the positive class
pred_probs <- predict(pruned_tree, X_test, type = "prob")[, "<=50K"]  # Adjust based on
↪  your positive class

# Compute ROC curve for the model with predictions
roc_curve_model_deleted <- roc(y_test[, 1], pred_probs, levels = c("<=50K", ">50K"))
```

```
## Setting direction: controls > cases
```

```
# Calculate AUC for the model
auc_value_model_deleted <- auc(roc_curve_model_deleted)

# Print AUC value for the model
cat("AUC for Model (Missing Data Deleted):", auc_value_model_deleted, "\n")
```

```
## AUC for Model (Missing Data Deleted): 0.8887446
```

```
# Create a data frame for the model's ROC data
roc_data_model_deleted <- data.frame(
  FPR = 1 - roc_curve_model_deleted$specificities,  # False Positive Rate
  TPR = roc_curve_model_deleted$sensitivities         # True Positive Rate
)
```

2.3 Surrogate Splits

We reuse the code from the previous file.

NOTE: We preferred to do this rather than saving the files and loading them as it sometimes led to non-reproducible issues for the other group members.

```
# NEED TO RELOAD DATA: We deleted some of it before. We preferred to do it this way
↪  rather than changing the order
# of the methods to avoid reloading it. This is because we wanted the simplest methods
↪  first.

# Get the current working directory
current_dir <- getwd()
cat("Current directory:", current_dir, "\n")
```

```
## Current directory: C:/Users/megar/Documents/Code/Data Science Toolbox/dst-group-project-1/report
```

```r
# Get the parent directory
parent_dir <- dirname(current_dir)
cat("Parent directory:", parent_dir, "\n")
```

```
## Parent directory: C:/Users/megar/Documents/Code/Data Science Toolbox/dst-group-project-1
```

```r
# Set the working directory to the parent directory
setwd(parent_dir)
cat("New working directory:", getwd(), "\n")
```

```
## New working directory: C:/Users/megar/Documents/Code/Data Science Toolbox/dst-group-project-1
```

```r
# Load the training and test datasets
X_train <- read.csv("data/X_train_smote.csv", row.names = 1)  # Load the SMOTE
↪  transformed training features
y_train <- read.csv("data/y_train_smote.csv", row.names = 1)  # Load the SMOTE
↪  transformed training labels
X_test <- read.csv("data/X_test.csv", row.names = 1)    # Load the test features
y_test <- read.csv("data/y_test.csv", row.names = 1)    # Load the test labels

# Combine X_train and y_train into one data frame for rpart
train_data <- cbind(X_train, y_train)

# Drop the "education" column from the training data if it exists
train_data <- train_data[, !names(train_data) %in% "education"]

# Fit the classification tree using rpart
fit <- rpart(income ~ ., data = train_data, method = "class", control = rpart.control(cp
↪  = 1e-6))

# Get the cost-complexity pruning table and identify the best cp based on minimum xerror
cptable <- fit$cptable
best_cp <- cptable[which.min(cptable[,"xerror"]), "CP"]
cat("Best CP:", best_cp, "\n")
```

```
## Best CP: 0.0001485492
```

```r
# Prune the tree using the best cp
pruned_tree <- prune(fit, cp = best_cp)

# Get predicted probabilities for the positive class
pred_probs <- predict(pruned_tree, X_test, type = "prob")[, "<=50K"]  # Adjust based on
↪  your positive class

# Compute ROC curve for the model with predictions
roc_curve_model <- roc(y_test[, 1], pred_probs, levels = c("<=50K", ">50K"))
```

```
## Setting direction: controls > cases
```

```r
# Calculate AUC for the model
auc_value_model <- auc(roc_curve_model)

# Print AUC value for the model
cat("AUC for Model:", auc_value_model, "\n")
```

```
## AUC for Model: 0.879037
```

```r
# Create a data frame for the model's ROC data
roc_data_model <- data.frame(
  FPR = 1 - roc_curve_model$specificities,  # False Positive Rate
  TPR = roc_curve_model$sensitivities         # True Positive Rate
)
```

3 Comparisons of Performance

We compare the ROC curves obtained from each method.

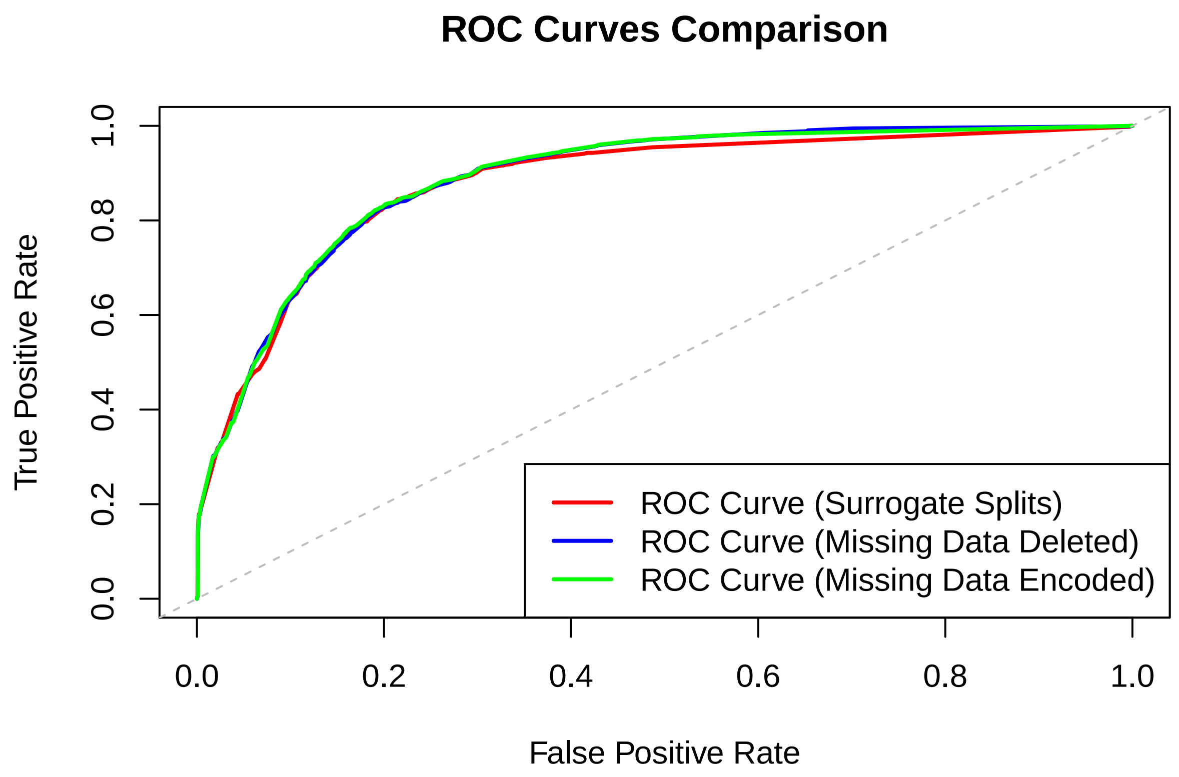
```r
# Create the plot for the saved ROC curve
plot(roc_data_model$FPR, roc_data_model$TPR, type = "l", col = "red", lwd = 2,
     xlim = c(0, 1), ylim = c(0, 1),
     main = "ROC Curves Comparison",
     xlab = "False Positive Rate",
     ylab = "True Positive Rate")

# Add the ROC curve for the model with missing data deleted
lines(roc_data_model_deleted$FPR, roc_data_model_deleted$TPR, col = "blue", lwd = 2)

# Add the ROC curve for the model with missing data encoded as "?"
lines(roc_data_model_encoded$FPR, roc_data_model_encoded$TPR, col = "green", lwd = 2)

# Add a diagonal line for random guessing
abline(0, 1, lty = 2, col = "grey")

# Add a legend to the plot
legend("bottomright", legend = c("ROC Curve (Surrogate Splits)", "ROC Curve (Missing Data
↪  Deleted)", "ROC Curve (Missing Data Encoded)"),
       col = c("red", "blue", "green"), lwd = 2)
```



The results shocked us. The tree was quite robust to the method we chose, and the ROC curves and AUC were almost identical.

Having exhausted most of the methods of dealing with missingness we explored, as well as the use of SMOTE to treat imbalance, we moved to considering ensembles.