Other Methods for Dealing with Missingness

Contents

1 Introduction

We have explored surrogate splits in the previous section. This did not provide a satisfactory performance. Hence, we here explore other methods of dealing with missingness from [2] and compare them using the performance metrics we found before. 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.

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

# 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

## Loading required package: readr
```

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

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

```
## [[4]]
##  [1] "readr"     "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"
##  [7] "rpart"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [13] "methods"   "base"
##
## [[5]]
##  [1] "readr"     "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"
##  [7] "rpart"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [13] "methods"   "base"
##
## [[6]]
##  [1] "readr"     "dplyr"     "pROC"      "caret"     "lattice"   "ggplot2"
##  [7] "rpart"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [13] "methods"   "base"
```

```
# 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
```

We load the data below.

```
# Data loading
# 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/VivekP
```

```
# 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
```

```
# 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
```

```
# Load the training and test datasets
X_train <- read.csv("data/X_train.csv", row.names = 1)  # Load the training features
y_train <- read.csv("data/y_train.csv", row.names = 1)  # Load the 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)
```

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

We load the analysis using surrogate splits from before, to compare with the other methods of dealing with missingness that we will look at.

```r
# Fit the classification tree using rpart
roc_data_model_surrogate <- read_csv("roc_data/roc_data_model_surrogate.csv")
```

```
## Rows: 32 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): FPR, TPR
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2 Other Methods to Deal with Missingness

Besides surrogate splits, there are several other methods to deal with missingness. As mentioned in Section 5.2, these are: - Treating missingness as its own category - Complete case analysis. - Imputation.

We now explain each in detail, and explain how they are implemented.

2.1 Analysis with Missingness Encoded as ?

This treats missingness as its own category. This is particularly easy to implement in our case, since all the missing variables were categorical.

```r
# 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: 0.0007910558
```

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

# Get predicted probabilities for the positive class
```

```r
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
```

```r
# 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): 0.8803486
```

```r
# 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
)
```

The AUC with missing data encoded as ? was around 0.880. We store the data and view it together with the other methods of dealing with missingness later.

2.2 Complete Case Analysis

Complete case analysis is a simple method of dealing with missing data. We just delete all missing data. This is not a good strategy here because we have about 50000 observations and about 3000 (or 6%) missing some values. This may not considered a negligible amount of missingness, and since missingness had relations to income, with more low-income individuals having missing data, the data may not be MCAR (see Section 5.1 for definition), which may bias our predictions. However, we will ignore these issues and consider this method as a baseline.

```r
# 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: 0.0006328446
```

```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_deleted <- roc(y_test[, 1], pred_probs, levels = c("<=50K", ">50K"))
```

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

```r
# 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): 0.8794634
```

```r
# 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
)
```

The AUC with missing data deleted was also around 0.880. We store the data and view it together with the other methods of dealing with missingness later.

2.3 Imputation

Imputation simply refers to filling in a missing value. we will here do single imputations using the mode.

Note: There is a more sophisticated way to perform imputation, called **multiple imputation** [1]. Multiple imputation (MI) is a method for dealing with missing data by filling in the gaps multiple times, creating several "complete" datasets. Each of these datasets is analyzed separately, and the results are combined to account for the uncertainty in the missing values. A common approach to MI is the Multiple Imputation by Chained Equations (MICE), which is versatile and applicable in various contexts. This technique generates multiple predictions for each missing value, allowing the analysis of the imputed data to reflect the uncertainty inherent in these estimates.

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

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

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

```r
# We put a hidden code chunk before to reload the data #

# Custom function to calculate the mode
get_mode <- function(x) {
  unique_x <- unique(x)
  unique_x[which.max(tabulate(match(x, unique_x)))]
}

# Function to replace NA with the mode
impute_with_mode <- function(data) {
  for (i in 1:ncol(data)) {
    if (is.factor(data[[i]]) || is.character(data[[i]])) {
      mode_value <- get_mode(data[[i]])
      data[[i]][is.na(data[[i]])] <- mode_value
    }
  }
  return(data)
}

# Impute missing values in the training and test datasets
train_data <- impute_with_mode(train_data)
X_test <- impute_with_mode(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 for imputed model:", best_cp, "\n")
```

```
## Best CP for imputed model: 8.582841e-05
```

```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_imputed <- roc(y_test[, 1], pred_probs, levels = c("<=50K", ">50K"))
```

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

```r
# Calculate AUC for the model
auc_value_model_imputed <- auc(roc_curve_model_imputed)

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

```
## AUC for Model (Imputed): 0.8881656
```

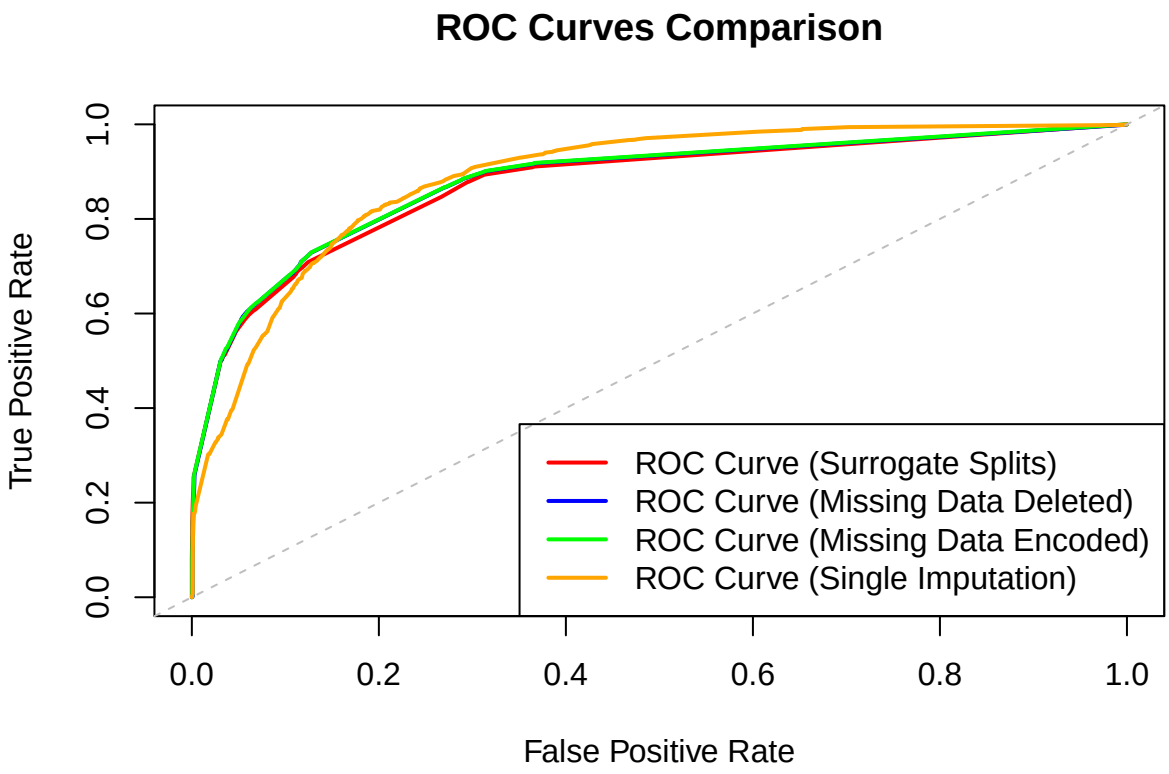
```r
# Create a data frame for the model's ROC data
roc_data_model_imputed <- data.frame(
  FPR = 1 - roc_curve_model_imputed$specificities,  # False Positive Rate
  TPR = roc_curve_model_imputed$sensitivities         # True Positive Rate
)
```

The AUC with missing data deleted was around 0.888. We store the data and view it together with the other methods of dealing with missingness later.

3 Comparisons of Performance

We had explored many measures of performance in the previous file. As discussed in Section 5.2.2, the accuracy is not a good measure of performance here due to the imbalance, and we will favour the ROC curve instead. We will compare the ROC curves obtained from each method.

```r
# Create the plot for the saved ROC curve
plot(roc_data_model_surrogate$FPR, roc_data_model_surrogate$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 the ROC curve for the model with missing data encoded as "?"
lines(roc_data_model_imputed$FPR, roc_data_model_imputed$TPR, col = "orange", 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)", "ROC Curve (Single Imputation)"),
       col = c("red", "blue", "green", "orange"), lwd = 2)
```




```r
# Save ROC data for all models to CSV files
# Create the 'roc_data' directory if it doesn't exist
if (!dir.exists("roc_data")) {
  dir.create("roc_data")
}

# Save ROC data for all models to CSV files in the 'roc_data' directory
write.csv(roc_data_model_deleted, "roc_data/roc_data_model_deleted.csv", row.names =
↪  FALSE)
write.csv(roc_data_model_encoded, "roc_data/roc_data_model_encoded.csv", row.names =
↪  FALSE)
write.csv(roc_data_model_imputed, "roc_data/roc_data_model_imputed.csv", row.names =
↪  FALSE)
```

The results were surprising. The tree was quite robust to the method we chose, and the ROC curves and AUC were almost identical. However, it looks like using simple imputation may be better overall as it has the highest area under the ROC curve in this case. Having explored methods to deal with missing data, we now look at the use of SMOTE to deal with imbalance.

4 References

[1] Van Buuren, Stef. "Multiple imputation of discrete and continuous data by fully conditional specification." Statistical methods in medical research 16.3 (2007): 219-242. [2] Scheffer, Judi. "Dealing with missing data." (2002).