PA

Part 1: 決定木の出力

与えられたIonosphereデータセットに対してRを使用して決定木を構築し可視化します。作業の手順は以下の通りです。

a. 作業ディレクトリとパッケージの設定
作業ディレクトリを設定し、必要なパッケージrpartとmlbenchを読み込みます。Ionosphereデータセットもこのステップで読み込みます。

```
# 作業ディレクトリの設定（必要に応じて変更）
setwd("/Users/hayashir/work/Github/UoPeople/CS4407-branch/CS4407/unit5/PA")

# パッケージの読み込み
library(rpart)
library(mlbench)

# データセットの読み込み
ionosphere_data <- read.table("Ionosphere.txt",
                              header = FALSE,
                              sep = ",",
                              stringsAsFactors = TRUE)

# 列名を追加
colnames(ionosphere_data) <- c(paste0("V", 1:34), "Class")

# データの確認
head(ionosphere_data)
summary(ionosphere_data)
```

これにより、Ionosphere.txt データセットが正しく読み込まれ、34列の連続値と1列のクラスラベル（Class： "g" または "b"）を持つデータフレームとして準備されます。

b. 決定木の作成
`rpart()` 関数を使用して、クラス分類のための決定木を作成します。このモデルは、Class（良好な信号か不良な信号か）を予測します。

```r
# 決定木の作成
ionosphere_tree <- rpart(Class ~ ., data = ionosphere_data)

# 決定木の概要を表示
print(summary(ionosphere_tree))
```

```
> ionosphere_tree <- rpart(Class ~ ., data = ionosphere_data)
>
>
> print(summary(ionosphere_tree))
Call:
rpart(formula = Class ~ ., data = ionosphere_data)
  n= 351

          CP nsplit rel error    xerror       xstd
1 0.54761905      0 1.0000000 1.0000000 0.07132675
2 0.20634921      1 0.4523810 0.5079365 0.05741255
3 0.02116402      2 0.2460317 0.3015873 0.04619978
4 0.01000000      5 0.1825397 0.3333333 0.04825916

Variable importance
 V5 V27  V3  V7 V23 V13 V25 V31 V33 V28  V9 V22 V14  V1 V32 V34  V8 V29
 21  12  11  10   8   6   6   5   5   5   3   1   1   1   1   1   1   1

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 351 126 g (0.35897436 0.64102564)
   2) V5< 0.23154 77   4 b (0.94805195 0.05194805) *
   3) V5>=0.23154 274  53 g (0.19343066 0.80656934)
     6) V27>=0.999945 52  13 b (0.75000000 0.25000000)
      12) V1< 0.5 19   0 b (1.00000000 0.00000000) *
      13) V1>=0.5 33  13 b (0.60606061 0.39393939)
        26) V3< 0.73004 8   0 b (1.00000000 0.00000000) *
        27) V3>=0.73004 25  12 g (0.48000000 0.52000000)
          54) V22>=0.47714 9   1 b (0.88888889 0.11111111) *
          55) V22< 0.47714 16   4 g (0.25000000 0.75000000) *
     7) V27< 0.999945 222  14 g (0.06306306 0.93693694) *
 '
```

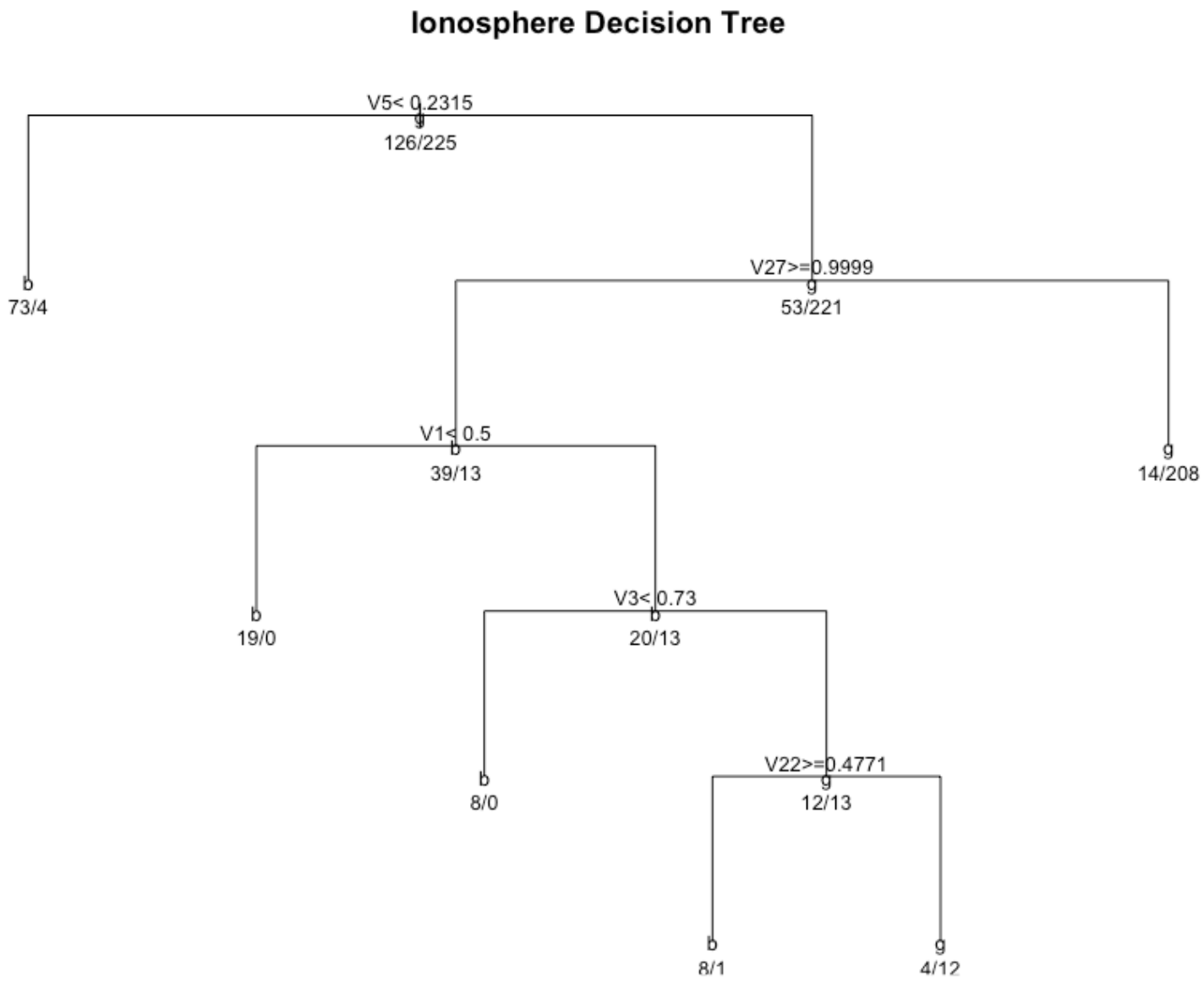
Figure 1 決定木の概要

c. 決定木の可視化

`plot()` と `text()` 関数を用いて、作成した決定木を可視化します。実行するコマンドは以下の通りです:

```
# マージンを調整
par(mar = c(4, 4, 4, 4))
# 決定木の描画
plot(ionosphere_tree, uniform = TRUE, main = "Ionosphere Decision Tree")
# ノードにラベルを追加
text(ionosphere_tree, use.n = TRUE, all = TRUE, cex = 0.8)
```

コマンドを実行した結果を以下に記載します:



Part 2: 精度の評価

このセクションでは、Ionosphereデータセット を訓練データとテストデータに分割し、作成した決定木モデルの予測精度を評価します。具体的には、テストデータを使用してモデルがどれだけ正確にクラス（"g" または "b"）を分類できるかを確認します。

a. データの分割
データセットを訓練データとテストデータに分割します。 `sample()` 関数を使用して、全データの70%を訓練データに、30%をテストデータに設定します。

```
set.seed(123)

# データを7:3に分割
train_indices <- sample(1:nrow(ionosphere_data), 0.7 * nrow(ionosphere_data))
train_data <- ionosphere_data[train_indices, ]
test_data <- ionosphere_data[-train_indices, ]
```

b. 訓練データで決定木を作成
訓練データを用いて、新しく決定木を構築します。

```
# 訓練データを使った決定木の作成
ionosphere_tree_train <- rpart(Class ~ ., data = train_data)

# モデル概要の表示
print(summary(ionosphere_tree_train))
```

```
> print(summary(ionosphere_tree_train))
Call:
rpart(formula = Class ~ ., data = train_data)
  n= 245

         CP nsplit rel error    xerror       xstd
1 0.4943820      0 1.0000000 1.0000000 0.08458323
2 0.2359551      1 0.5056180 0.5280899 0.06924820
3 0.0100000      2 0.2696629 0.3033708 0.05507279

Variable importance
 V5 V27  V7 V31 V13 V23 V33 V29  V3  V9 V28
 23  18   9   8   7   7   7   7   6   4   3


node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 245 89 g (0.36326531 0.63673469)
  2) V5< 0.02313 44  0 b (1.00000000 0.00000000) *
  3) V5>=0.02313 201 45 g (0.22388060 0.77611940)
    6) V27>=0.999945 39  9 b (0.76923077 0.23076923) *
    7) V27< 0.999945 162 15 g (0.09259259 0.90740741) *
>
```

Figure 3 テストデータによる決定木の概要

c. テストデータで予測

`predict()` 関数を使用して、テストデータに基づく予測を行います

```
# テストデータでの予測
test_predictions <- predict(ionosphere_tree_train, newdata = test_data, type = "class")
```

d. 精度の計算

テストデータの実際のクラスラベルと予測結果を比較し、モデルの精度を計算します。

```
# 予測と実際の値を比較するクロス集計表を作成
confusion_matrix <- table(test_predictions, test_data$Class)

# モデルの精度を計算
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# 結果の表示
print(confusion_matrix)
print(paste("Accuracy: ", round(accuracy * 100, 2), "%"))
```