Part 2 (10 points)

You are required to build and test a classification model using the

drug\_consumption\_cannabis.csv dataset. You must study how to use XBoost in R yourself. You

must write a R code implementing the following requirements:

 Read the drug\_consumption\_cannabis.csv dataset into df.

 Split df into a training set tr and a test set ts with the ratio of 75% - 25%.

 Use the training set tr to build the best model. When choosing the best model, you must

perform parameter tuning.

 Test the model on the test set ts and calculate and include the following performance

measures in your submission:

表格

描述已自动生成

> library(vcd)

> library(xgboost)

> library(caret)

> library(e1071)

>

> df <- read.csv("drug\_consumption\_cannabis.csv")

>

> # Split the dataset

> set.seed(123)

> trainIndex <- createDataPartition(df$C6, p = 0.75, list = FALSE)

> tr <- df[trainIndex,]

> ts <- df[-trainIndex,]

>

> # Set parameter grid for tuning

> grid <- expand.grid(nrounds = c(50, 100, 150),

+ max\_depth = c(4, 6, 8),

+ colsample\_bytree = c(0.6, 0.8, 1),

+ eta = c(0.01, 0.05, 0.1),

+ gamma = 0,

+ min\_child\_weight = 1,

+ subsample = 1)

>

> tr$C6 <- as.factor(tr$C6)

>

> # Build model using cross-validation

> set.seed(123)

> model <- train(

+ C6 ~ ., data = tr, method = "xgbTree",

+ trControl = trainControl(method = "cv", number = 5),

+ tuneGrid = grid, metric = "Accuracy"

+ )

> # Print parameter list

> print(grid)

nrounds max\_depth colsample\_bytree eta gamma min\_child\_weight subsample

1 50 4 0.6 0.01 0 1 1

2 100 4 0.6 0.01 0 1 1

3 150 4 0.6 0.01 0 1 1

4 50 6 0.6 0.01 0 1 1

5 100 6 0.6 0.01 0 1 1

6 150 6 0.6 0.01 0 1 1

7 50 8 0.6 0.01 0 1 1

8 100 8 0.6 0.01 0 1 1

9 150 8 0.6 0.01 0 1 1

10 50 4 0.8 0.01 0 1 1

11 100 4 0.8 0.01 0 1 1

12 150 4 0.8 0.01 0 1 1

13 50 6 0.8 0.01 0 1 1

14 100 6 0.8 0.01 0 1 1

15 150 6 0.8 0.01 0 1 1

16 50 8 0.8 0.01 0 1 1

17 100 8 0.8 0.01 0 1 1

18 150 8 0.8 0.01 0 1 1

19 50 4 1.0 0.01 0 1 1

20 100 4 1.0 0.01 0 1 1

21 150 4 1.0 0.01 0 1 1

22 50 6 1.0 0.01 0 1 1

23 100 6 1.0 0.01 0 1 1

24 150 6 1.0 0.01 0 1 1

25 50 8 1.0 0.01 0 1 1

26 100 8 1.0 0.01 0 1 1

27 150 8 1.0 0.01 0 1 1

28 50 4 0.6 0.05 0 1 1

29 100 4 0.6 0.05 0 1 1

30 150 4 0.6 0.05 0 1 1

31 50 6 0.6 0.05 0 1 1

32 100 6 0.6 0.05 0 1 1

33 150 6 0.6 0.05 0 1 1

34 50 8 0.6 0.05 0 1 1

35 100 8 0.6 0.05 0 1 1

36 150 8 0.6 0.05 0 1 1

37 50 4 0.8 0.05 0 1 1

38 100 4 0.8 0.05 0 1 1

39 150 4 0.8 0.05 0 1 1

40 50 6 0.8 0.05 0 1 1

41 100 6 0.8 0.05 0 1 1

42 150 6 0.8 0.05 0 1 1

43 50 8 0.8 0.05 0 1 1

44 100 8 0.8 0.05 0 1 1

45 150 8 0.8 0.05 0 1 1

46 50 4 1.0 0.05 0 1 1

47 100 4 1.0 0.05 0 1 1

48 150 4 1.0 0.05 0 1 1

49 50 6 1.0 0.05 0 1 1

50 100 6 1.0 0.05 0 1 1

51 150 6 1.0 0.05 0 1 1

52 50 8 1.0 0.05 0 1 1

53 100 8 1.0 0.05 0 1 1

54 150 8 1.0 0.05 0 1 1

55 50 4 0.6 0.10 0 1 1

56 100 4 0.6 0.10 0 1 1

57 150 4 0.6 0.10 0 1 1

58 50 6 0.6 0.10 0 1 1

59 100 6 0.6 0.10 0 1 1

60 150 6 0.6 0.10 0 1 1

61 50 8 0.6 0.10 0 1 1

62 100 8 0.6 0.10 0 1 1

63 150 8 0.6 0.10 0 1 1

64 50 4 0.8 0.10 0 1 1

65 100 4 0.8 0.10 0 1 1

66 150 4 0.8 0.10 0 1 1

67 50 6 0.8 0.10 0 1 1

68 100 6 0.8 0.10 0 1 1

69 150 6 0.8 0.10 0 1 1

70 50 8 0.8 0.10 0 1 1

71 100 8 0.8 0.10 0 1 1

72 150 8 0.8 0.10 0 1 1

73 50 4 1.0 0.10 0 1 1

74 100 4 1.0 0.10 0 1 1

75 150 4 1.0 0.10 0 1 1

76 50 6 1.0 0.10 0 1 1

77 100 6 1.0 0.10 0 1 1

78 150 6 1.0 0.10 0 1 1

79 50 8 1.0 0.10 0 1 1

80 100 8 1.0 0.10 0 1 1

81 150 8 1.0 0.10 0 1 1

> best\_parameters <- model$bestTune

> print(best\_parameters)

nrounds max\_depth eta gamma colsample\_bytree min\_child\_weight subsample

59 100 4 0.1 0 0.8 1 1

>

> # Forecast

> predictions <- predict(model, ts)

>

> # Convert ts$C6 to factor

> ts$C6 <- as.factor(ts$C6)

>

> # Convert predictions to factor with same levels as ts$C6

> predictions <- factor(predictions, levels = levels(ts$C6))

>

> # Calculate confusion matrix

> confusion\_matrix <- confusionMatrix(predictions, ts$C6)

>

> # Function to extract performance metrics

> extract\_metrics <- function(confusion\_matrix) {

+ tp <- confusion\_matrix[2, 2]

+ fp <- confusion\_matrix[1, 2]

+ tn <- confusion\_matrix[1, 1]

+ fn <- confusion\_matrix[2, 1]

+

+ tpr <- tp / (tp + fn)

+ fpr <- fp / (fp + tn)

+ precision <- tp / (tp + fp)

+ recall <- tpr

+ f\_measure <- 2 \* (precision \* recall) / (precision + recall)

+ mcc\_numerator <- as.numeric(tp) \* as.numeric(tn) - as.numeric(fp) \* as.numeric(fn)

+ mcc\_denominator <- sqrt(as.numeric(tp + fp) \* as.numeric(tp + fn) \* as.numeric(tn + fp) \* as.numeric(tn + fn))

+ mcc <- mcc\_numerator / mcc\_denominator

+ kappa\_obj <- vcd::Kappa(as.table(confusion\_matrix))

+ kappa\_value <- kappa\_obj$statistic["Kappa"]

+

+ return(c(TPR = tpr, FPR = fpr, Precision = precision, Recall = recall, F\_measure = f\_measure, MCC = mcc, Kappa = kappa))

+ }

>

> # Calculate metrics for each class

> class\_0\_metrics <- extract\_metrics(confusion\_matrix$table)

> class\_1\_metrics <- extract\_metrics(matrix(c(confusion\_matrix$table[2,2], confusion\_matrix$table[2,1], confusion\_matrix$table[1,2], confusion\_matrix$table[1,1]), nrow = 2))

>

> # Calculate weighted average metrics

> n\_class\_0 <- sum(df$C6 == 0)

> n\_class\_1 <- sum(df$C6 == 1)

> weighted\_avg\_metrics <- (n\_class\_0 \* class\_0\_metrics + n\_class\_1 \* class\_1\_metrics) / (n\_class\_0 + n\_class\_1)

>

> # Display metrics

> cat("Class 0 Metrics:\n")

Class 0 Metrics:

> print(class\_0\_metrics)

TPR FPR Precision Recall F\_measure MCC Kappa.Kappa

0.8485804 0.3051948 0.8512658 0.8485804 0.8499210 0.5424863 0.5424800

>

> cat("Class 1 Metrics:\n")

Class 1 Metrics:

> print(class\_1\_metrics)

TPR FPR Precision Recall F\_measure MCC Kappa.Kappa

0.6903226 0.1487342 0.6948052 0.6903226 0.6925566 0.5424863 0.5424800

>

> cat("Weighted Average Metrics:\n")

Weighted Average Metrics:

> print(weighted\_avg\_metrics)

TPR FPR Precision Recall F\_measure MCC Kappa.Kappa

0.7423756 0.2001960 0.7462670 0.7423756 0.7443157 0.5424863 0.5424800