Prediction of manners

Wenjun

2025-02-22

In this project, I would use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

I would first predict the manner in which the participants did the exercise (This is the “classe” variable in the training set). As the outcome variable is categorical variable, the methods used to build the prediction model include decision tree, bagging, random forest, and boosting. Ultimately, I would use the best prediction model to predict 20 different test cases.

Data processing

Since a lot of variables in the pml_training set have a large proportion of missing values, these variables would not be used in prediction model. After data cleaning, the training data set would be separated into sub_train and sub_test set. The model would be built based on sub_train set, and then would be tested on sub_test set.

```
library(readr)
library(caret)
pml_train=read_csv("C:/Users/Lenovo/Desktop/R/rdata/Coursera/Machine_learning/pml_training.cs
v")
pml_test=read_csv("C:/Users/Lenovo/Desktop/R/rdata/Coursera/Machine_learning/pml_testing.csv")
table(pml_train$classe)
library(naniar)
miss_var_summary(pml_train)
library(dplyr)
pml_train_clean=pml_train %>%
    select(where(~mean(is.na(.))==0))
pml_train_clean$classe=factor(pml_train_clean$classe)
set.seed(123)
train_index <- createDataPartition(pml_train_clean$classe, p = 0.7, list = FALSE)
sub_train <- pml_train_clean[train_index, ]
sub_test <- pml_train_clean[-train_index, ]
```

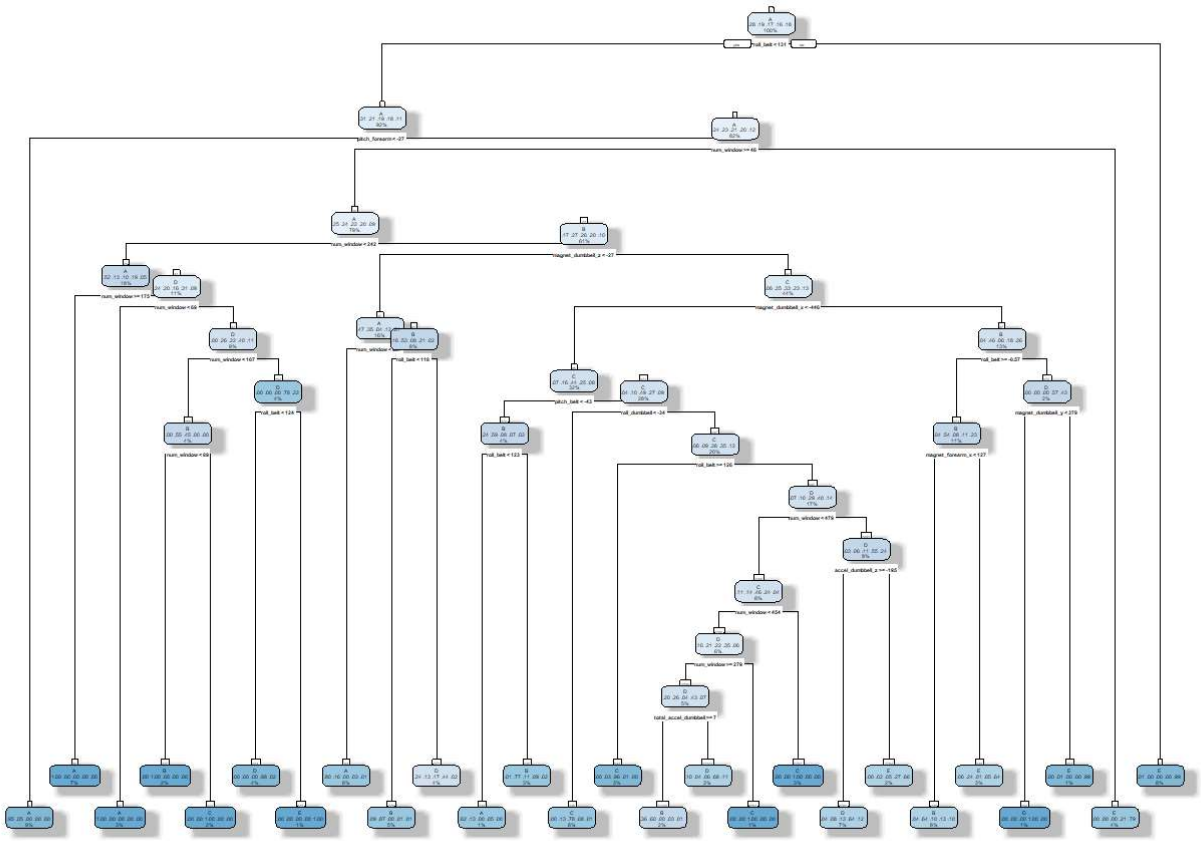
1.Decision tree

First of all, the decision tree model was used to build the prediction model based on sub_train set, and 10-fold cross-validation method was used to improve the accuracy of model. When the model was tested on the sub_test set, the out-of-sample accuracy was 0.83.

```
library(caret)
ctrl <- trainControl(method = "cv",number =10)
set.seed(1234)
tree_model <- train(
  classe ~ .,
  data = sub_train[,-1:-5],
  method = "rpart",
  trControl = ctrl,
  tuneGrid = expand.grid(cp = seq(0.01, 0.5, 0.01))
)
pred_tree <- predict(tree_model, newdata=sub_test)
pred_tree=factor(pred_tree,levels=levels(pml_train_clean$classe))
conf_mat_tree <- confusionMatrix(pred_tree,sub_test$classe)
print(conf_mat_tree$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.8273577      0.7822750      0.8174552      0.8369345      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
library(rpart.plot)
rpart.plot(tree_model$finalModel,
           box.palette = "Blues",
           shadow.col = "gray",
           nn = TRUE)
```



2. Bagging

The bagging method with 5-fold cross-validation was applied to build a prediction model based on sub_train data set. The cross-validation processes were repeated 2 times in order to improve accuracy. The final out-of-sample accuracy of bagging model on sub_test data was 0.996.

```
library(caret)
library(doParallel)
library(MLmetrics)
cl <- makePSOCKcluster(4)
registerDoParallel(cl)
### Build model
ctrl_bag <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 2,
  classProbs = TRUE,
  summaryFunction = multiClassSummary,
  allowParallel = TRUE
)
set.seed(1234)
bag_model <- train(
  classe ~ .,
  data = sub_train[,-1:-5],
  method = "treebag",
  trControl = ctrl_bag,
  nbagg = 30,
  metric = "logLoss"
)
stopCluster(cl)

### Predict sub_test data
pred_bag <- predict(bag_model, sub_test)
conf_mat_bag <- confusionMatrix(pred_bag, sub_test$classe)
print(conf_mat_bag$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9960918      0.9950564      0.9941415      0.9975209      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

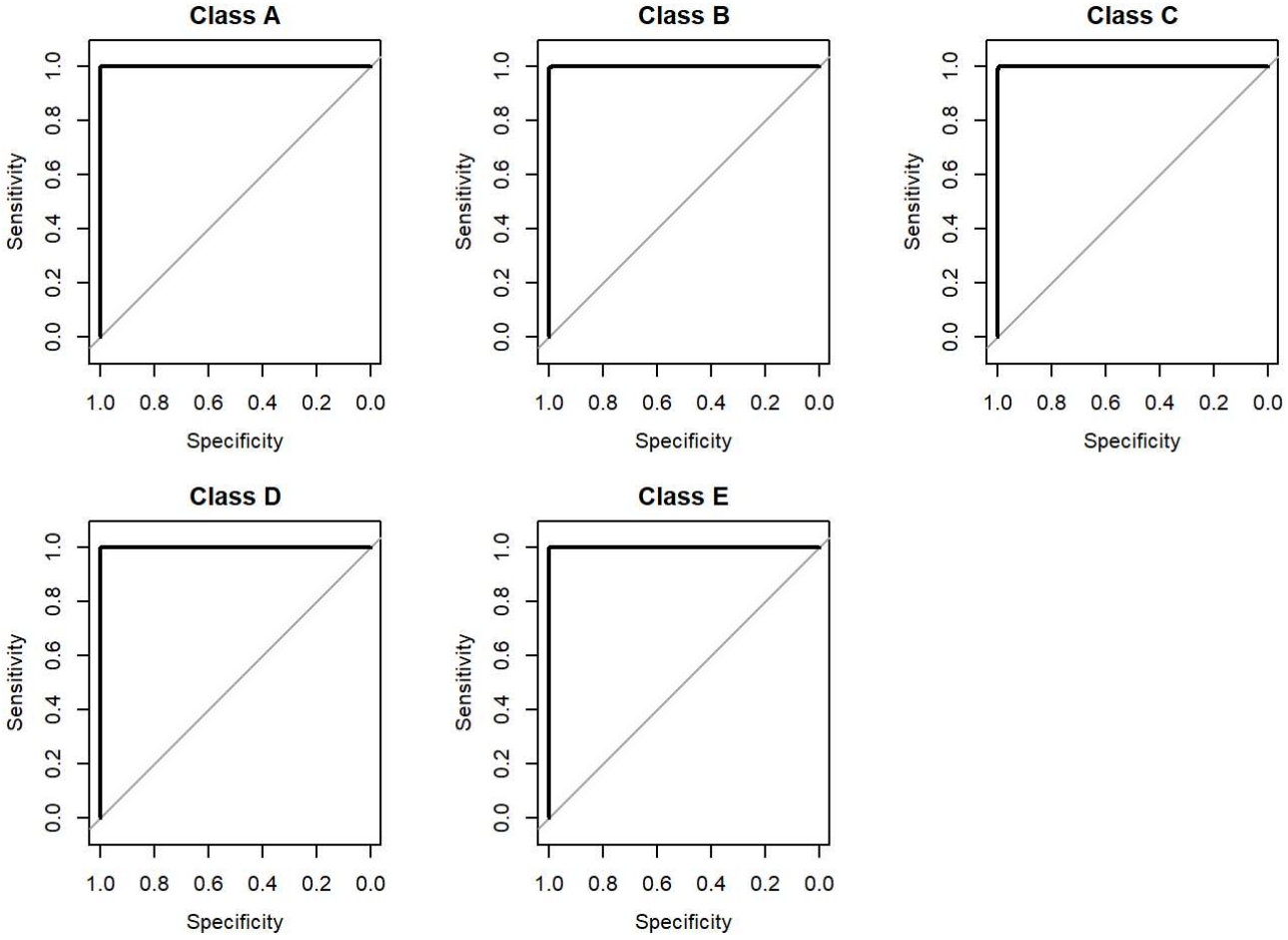
3. Random Forest

In this section, random forest was utilized to build the prediction model. The model was validated by 5-fold and repeated 2 times. And the results showed that the out-of sample accuracy on sub_test data was 0.998, which was excellent.

```
library(caret)
library(randomForest)
library(pROC)
library(doParallel)
cl <- makePSOCKcluster(4)
registerDoParallel(cl)

### Build model
ctrl_rf <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 2,
  classProbs = TRUE,
  summaryFunction = multiClassSummary,
  allowParallel = TRUE,
  sampling = "up"
)

tuneGrid_rf <- expand.grid(
  .mtry = c(2, sqrt(ncol(sub_train)-1),ncol(sub_train)-1))
set.seed(1234)
rf_model <- train(
  classe ~ .,
  data = sub_train[,-1:-5],
  method = "rf",
  trControl = ctrl_rf,
  tuneGrid = tuneGrid_rf,
  ntree = 300,
  metric = "logLoss",
  importance = TRUE
)

stopCluster(cl)

### Predict sub_test data
pred_classes_rf <- predict(rf_model, sub_test)
pred_probs_rf   <- predict(rf_model, sub_test, type = "prob")
conf_mat_rf <- confusionMatrix(pred_classes_rf, sub_test$classe)
print(conf_mat_rf$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9981308      0.9976358      0.9966580      0.9990666      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
### ROC curve
roc_list <- lapply(levels(sub_test$classe), function(cls){
  roc(response = as.numeric(sub_test$classe == cls),
      predictor = pred_probs_rf[[cls]])
})

par(mfrow = c(2,3))
for(i in 1:5){
  plot(roc_list[[i]], main = paste("Class", levels(sub_test$classe)[i]))
}
```



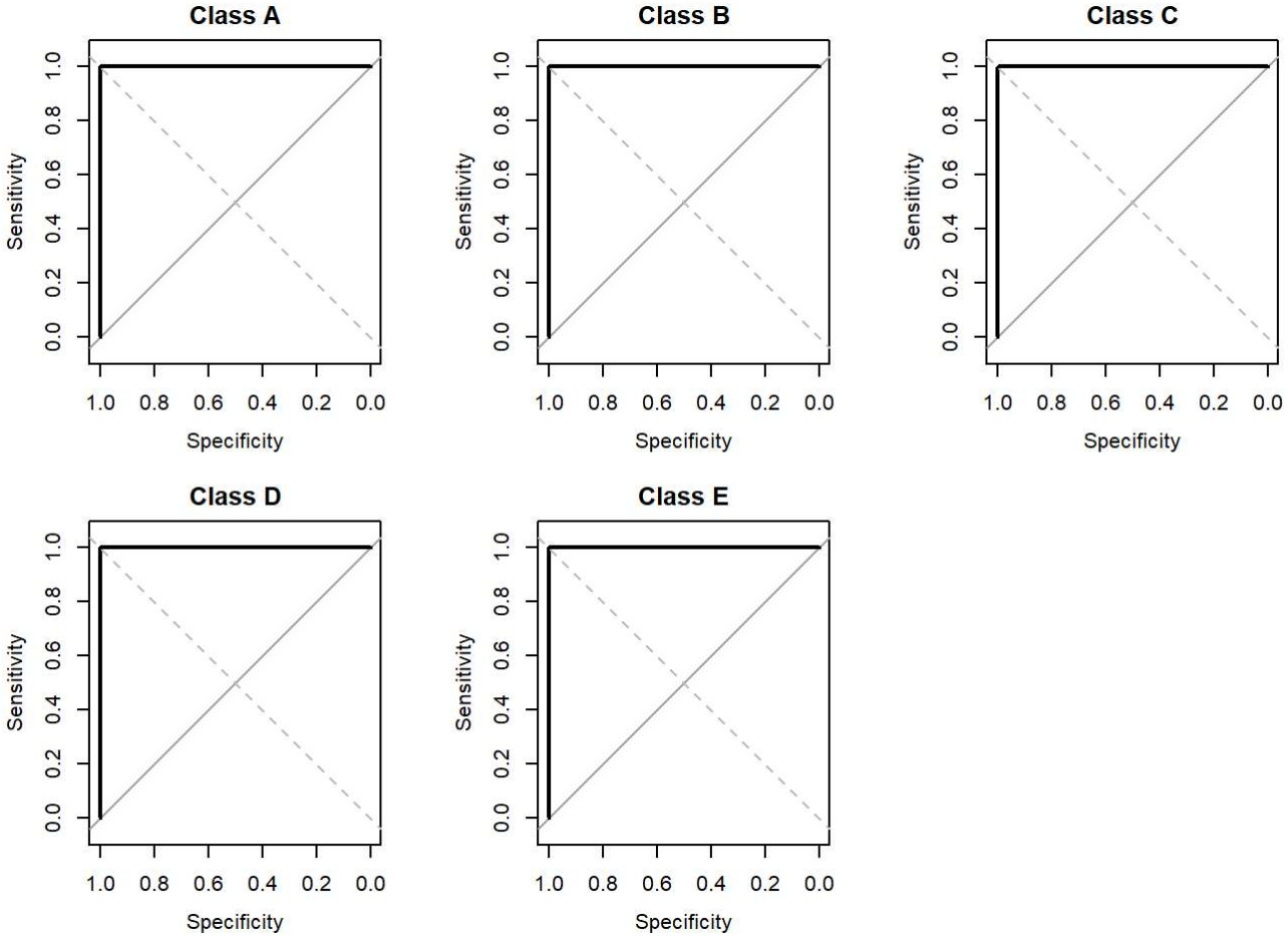
4.boosting

Lastly, boosting was used to build a prediction model, also with 5-fold cross-validation which repeated twice. The results showed that the out-of sample accuracy was 0.999 for sub_test data.

```
library(caret)
library(gbm)
library(doParallel)
library(pROC)
library(recipes)
cl <- makePSOCKcluster(4)
registerDoParallel(cl)

### Build model
ctrl_boosting <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 2,
  classProbs = TRUE,
  summaryFunction = multiClassSummary,
  allowParallel = TRUE,
  sampling = "smote"
)

tuneGrid_boosting <- expand.grid(
  interaction.depth = c(3,5),
  n.trees = c(100, 200),
  shrinkage = c(0.01, 0.1),
  n.minobsinnode = 10
)
set.seed(1234)
gbm_model <- train(
  classe ~ .,
  data = sub_train[,-1:-5],
  method = "gbm",
  trControl = ctrl_boosting,
  tuneGrid = tuneGrid_boosting,
  metric = "logLoss",
  verbose = FALSE
)
stopCluster(cl)

### Predict sub_test
pred_classes_gbm <- predict(gbm_model, sub_test)
pred_probs_gbm <- predict(gbm_model, sub_test, type = "prob")
conf_mat_gbm <- confusionMatrix(pred_classes_gbm, sub_test$classe)
print(conf_mat_gbm$overall)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9986406      0.9982806      0.9973232      0.9994129      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
# ROC curve
roc_list_gbm <- lapply(levels(sub_test$classe), function(cls){
  pROC::roc(response = as.numeric(sub_test$classe == cls),
            predictor = pred_probs_gbm[[cls]])
})
par(mfrow = c(2,3))
for(i in 1:5){
  plot(roc_list_gbm[[i]], main = paste("Class", levels(sub_test$classe)[i]))
  abline(a=0, b=1, lty=2, col="grey")
}
```



5. Prediction test data set

Since random forest and boosting methods achieved the best out-of-sample accuracy, these two models were used to predict the 20 test data. Both prediction results were showed below. The results from two models were consistent.

```
pred_test_rf=predict(rf_model,pml_test)
pred_test_boost=predict(gbm_model,pml_test)
print(pred_test_rf)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

```
print(pred_test_boost)
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```