HW5

2023-04-01

```
# Import data set
train <- read.delim("vowel.train.txt", header = TRUE, sep = ",")
train = train[-1]
str(train)
```

```
## 'data.frame':    528 obs. of  11 variables:
##  $ y   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ x.1 : num  -3.64 -3.33 -2.12 -2.29 -2.6 ...
##  $ x.2 : num  0.418 0.496 0.894 1.809 1.938 ...
##  $ x.3 : num  -0.67 -0.694 -1.576 -1.498 -0.846 ...
##  $ x.4 : num  1.779 1.365 0.147 1.012 1.062 ...
##  $ x.5 : num  -0.168 -0.265 -0.707 -1.053 -1.633 ...
##  $ x.6 : num  1.627 1.933 1.559 1.06 0.764 ...
##  $ x.7 : num  -0.388 -0.363 -0.579 -0.567 0.394 0.217 0.322 -0.435 -0.512 -0.466 ...
##  $ x.8 : num  0.529 0.51 0.676 0.235 -0.15 -0.246 0.45 0.992 0.928 0.702 ...
##  $ x.9 : num  -0.874 -0.621 -0.809 -0.091 0.277 0.238 0.377 0.575 -0.167 0.06 ...
##  $ x.10: num  -0.814 -0.488 -0.049 -0.795 -0.396 -0.365 -0.366 -0.301 -0.434 -0.836 ...
```

```
train$y = as.factor(train$y)
test <- read.delim("vowel.test.txt", header = TRUE, sep = ",")
test = test[-1]
test$y = as.factor(test$y)
```

Fit a random forest or gradient boosted model to the "vowel.train" data using all of the 11 features using the default values of the tuning parameters.

```
library(randomForest)

# Fit a random forest model with default parameters
fit_rf <- randomForest(y ~ ., data = train)
```

Use 5-fold CV to tune the number of variables randomly sampled as candidates at each split if using random forest, or the ensemble size if using gradient boosting.

```
library(caret)
# Set up the training control for cross-validation
train_control <- trainControl(method = "cv", number = 5)

# Tune the number of variables randomly sampled as candidates at each split
tune_grid <- expand.grid(mtry = 1:11)
set.seed(123)
fit_rf_tuned <- train(y ~ ., data = train, method = "rf",
                      importance = TRUE, proximity=TRUE,
                      tuneGrid = tune_grid, trControl = train_control)

fit_rf_tuned$bestTune # 1 variable
```

```
##   mtry
```

```
## 1    1
```

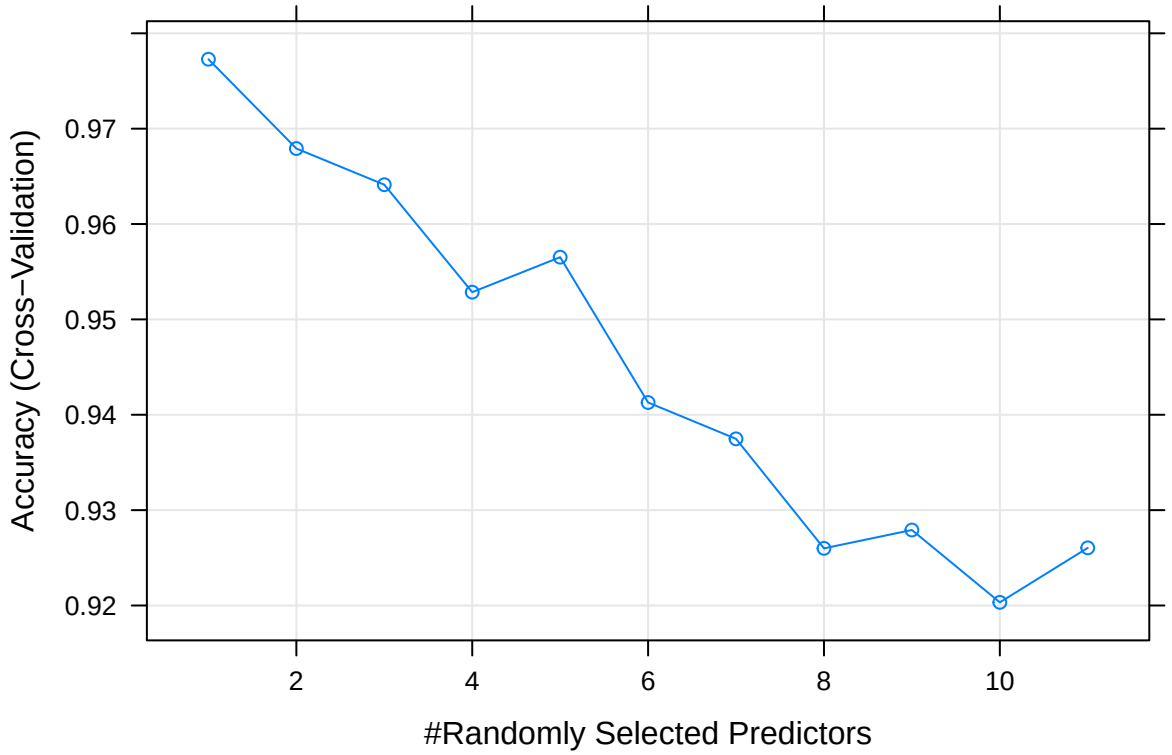
```
(fit_rf_tuned$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE,      proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 2.08%
## Confusion matrix:
##     1  2  3  4  5  6  7  8  9 10 11 class.error
## 1  48  0  0  0  0  0  0  0  0  0  0  0.00000000
## 2   0 48  0  0  0  0  0  0  0  0  0  0.00000000
## 3   0  0 48  0  0  0  0  0  0  0  0  0.00000000
## 4   0  0  0 47  0  1  0  0  0  0  0  0.02083333
## 5   0  0  0  0 46  1  0  0  0  0  1  0.04166667
## 6   0  0  0  0  0 46  0  0  0  0  2  0.04166667
## 7   0  0  0  0  2  0 45  1  0  0  0  0.06250000
## 8   0  0  0  0  0  0  0 48  0  0  0  0.00000000
## 9   0  0  0  0  0  0  0  1 46  1  0  0.04166667
## 10  0  0  0  0  0  0  1  0  0 47  0  0.02083333
## 11  0  0  0  0  0  0  0  0  0  0 48  0.00000000
```

```
print(fit_rf_tuned)          ## summary of fit object
```

```
## Random Forest
##
## 528 samples
##  10 predictor
##  11 classes: '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 424, 423, 420, 423, 422
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.9772829  0.9750054
##    2    0.9679175  0.9647027
##    3    0.9641259  0.9605282
##    4    0.9528561  0.9481230
##    5    0.9565249  0.9521608
##    6    0.9412847  0.9353900
##    7    0.9374752  0.9311987
##    8    0.9259914  0.9185599
##    9    0.9279144  0.9206758
##   10    0.9203296  0.9123301
##   11    0.9260446  0.9186184
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 1.
```

```
plot(fit_rf_tuned)          ## plot OOB MSE as function of # of trees
```



With the tuned model, make predictions using the majority vote method, and compute the misclassification rate using the 'vowel.test' data.

```
# Make predictions on the test data using the tuned model
pred_rf <- predict(fit_rf_tuned, newdata = test)

# Compute the misclassification rate
(misclassification_rate <- mean(pred_rf != test$y))
```

```
## [1] 0.4458874
```