Practical machine Learning - Writeup

Dylan

2024-04-22

Background and the Data

Six participants were asked to perform one set of 10 reptitions of the Unilateral Bumbbell Biceps Curl in 5 different fashions

- A: exactly to specification

- B: throwing elbows to the front

- C: lifting dumbbell only half way

- D: lowering the dumbbell only halfway

- F: throwing the hips to the front

Participants wore accelerometer sensors on four locations: arm, forearm, waist, and dumbbell. Sensor readings were recorded while the participants performed the activities at different specification, under supervision of experienced trainer, and labeled accordingly.

Our goal is to train a model to identify the specific manner (`classe` A~F) in which the participant did the exercise.

Selecting Features

Model fitting will be done on portion of the `pml-training.csv` data. Quick look at the data set shows that some metadata like row number, participant name, time stamps, etc. are included in column 1~7. Since they are not sensor readings, we will remove them from model fitting. Some columns also have class of `character`, which we will convert to numeric values.

```
library(caret)
library(dplyr)
library(ggplot2)
```

```
pml_training <- read.csv(file.path(getwd(), "data", "pml-training.csv"))

dim(pml_training)
```

```
## [1] 19622   160
```

```
names(pml_training)[1:8]
```

```
## [1] "X"                    "user_name"            "raw_timestamp_part_1"
## [4] "raw_timestamp_part_2" "cvtd_timestamp"       "new_window"
## [7] "num_window"           "roll_belt"
```

Removing column 1~7 and changing column classes

```
# Transform training data
pml_training <- pml_training %>%
  select(-c(1:7)) %>%
  mutate(
    classe = as.factor(classe),
    across(roll_belt:magnet_forearm_z, ~ as.numeric(.x))
    )
```

Further exploration shows that there are quite a lot of missing values in the data set.

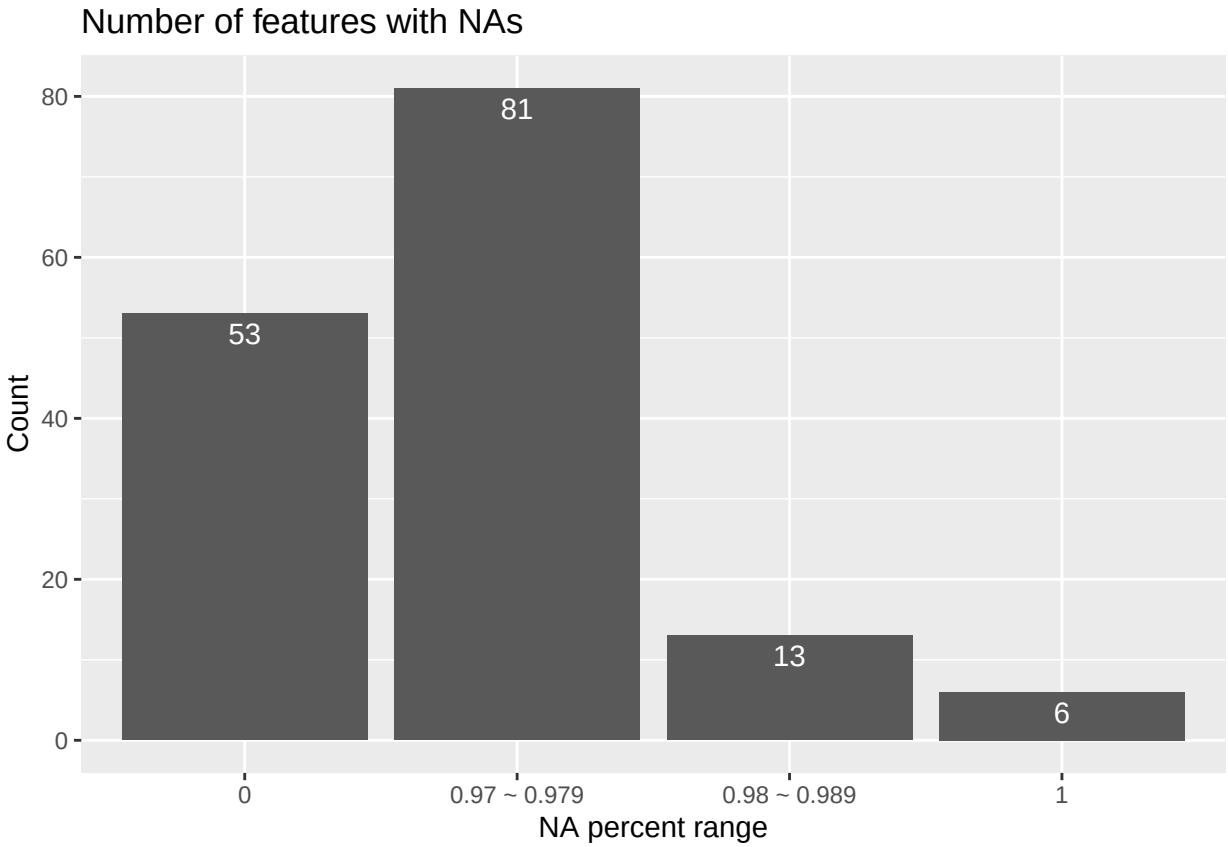
```
col_na <- pml_training %>%
  summarize(across(everything(), ~ sum(is.na(.))))

na_tab <- data.frame(
  col_names = names(col_na),
  na_count = as.numeric(col_na[1,])
)

na_tab <- na_tab %>%
  mutate(
    na_perc = na_count/nrow(pml_training)
  ) %>%
  arrange(desc(na_perc))

na_summary <- na_tab %>% group_by(na_perc) %>%
  summarize(feature_count = n()) %>%
  mutate(na_perc_bracket = case_when(
    na_perc >= 0.97 & na_perc < 0.98 ~ "0.97 ~ 0.979",
    na_perc >= 0.98 & na_perc < 0.99 ~ "0.98 ~ 0.989",
    na_perc >= 0.99 & na_perc < 0.99 ~ "0.99 ~ 0.999",
    na_perc == 1 ~ "1",
    na_perc == 0 ~ "0"
  )) %>%
  group_by(na_perc_bracket) %>%
  summarize(
    feature_count = sum(feature_count)
  ) %>%
  arrange(desc(na_perc_bracket))

ggplot(na_summary, aes(na_perc_bracket, feature_count)) +
  geom_col() +
  geom_text(aes(label = feature_count), color = "white", nudge_y = -2.5) +
  labs(
    title = "Number of features with NAs",
    x = "NA percent range",
    y = "Count"
  )
```



100 of the accelerometer related features contain more than 97% NA. Only 53 features have 0% NA.

We will use only the features without missing values for modeling.

```
# Filter for features with 0 NA
select_col <- na_tab %>%
  filter(na_perc == 0)

# extract col names
name_vec <- select_col$col_names

# extract select columns
pml_training_select <- pml_training %>%
  select(one_of(name_vec))
```

The Setup

Now that we have chosen the features to fit the model, we further partition the data into testing and training set.

```
set.seed(6525485)
# index
inTrain <- createDataPartition(y = pml_training_select$classe,                                    p=0.75, lis
# filter
training <- pml_training_select[inTrain,]
```

```
testing <- pml_training_select[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 14718    53
```

```
## [1] 4904   53
```

Model Fitting

We will use random forest to fit the model through the `caret` package. Random forest has the advantages of providing high accuracy, flexibility over non-linear data, while staying fairly robust to overfitting.

Training Specification

First, we make some specifications to training control.

We will expand tuning grid for `mtry` parameter beyond the default. Next, we specify the cross-validation control in the `trainControl()` function. We choose k = 5 fold for cross-validation because anything bigger than that will really inflate the computation time.

```
myTuneGrid <- data.frame(
  .mtry = c(5, 15, 25, 35, 45, 52)
)

myControl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = FALSE
)
```

Fitting the Random Forest Model

Now we are ready to fit the model using the specifications we have saved.

```
# Takes quite a while to run...
fit_rf_2 <- train(
  classe ~.,
  data = training,
  method = "rf",
  tuneGrid = myTuneGrid,
  trControl = myControl
)
```

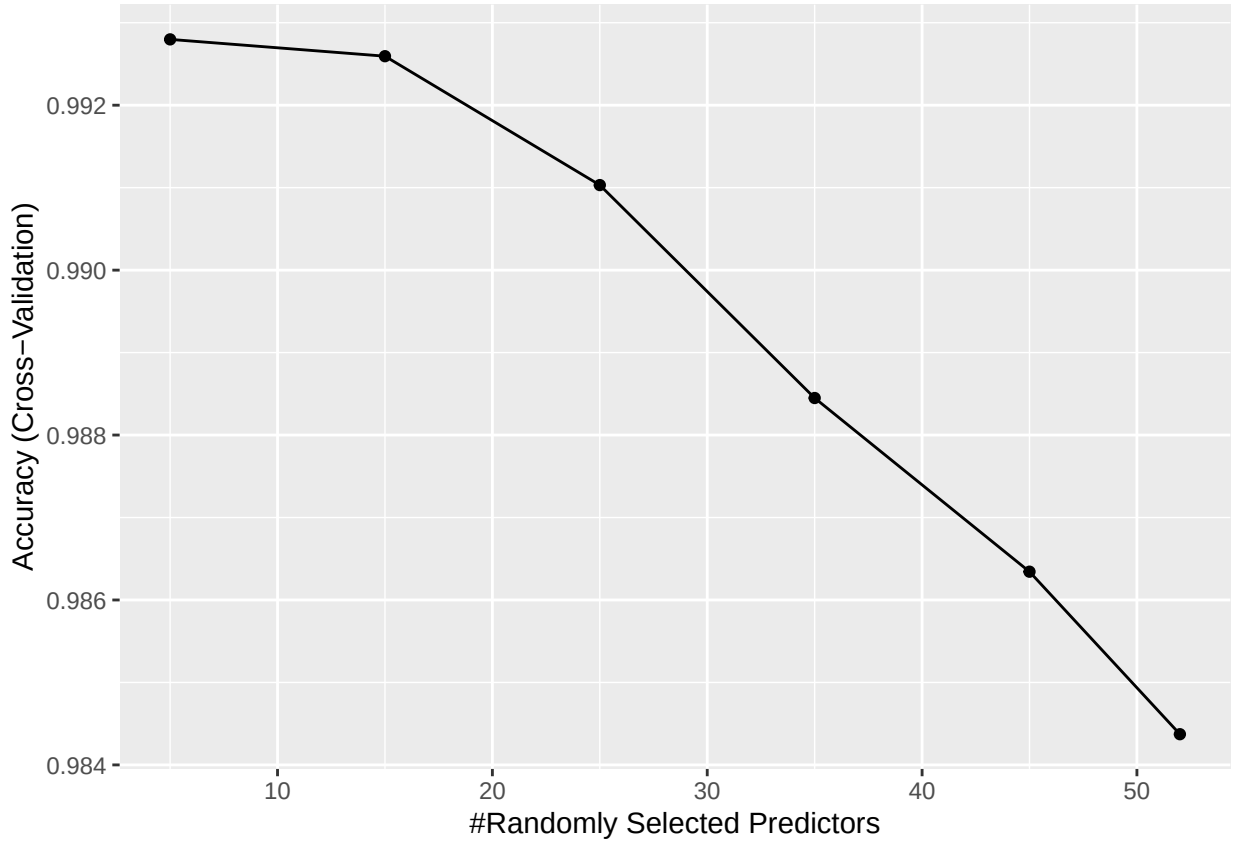
Let's take a look at the summary information on the model:

```
fit_rf_2
```

```
## Random Forest
##
## 14718 samples
```

```
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11773, 11775, 11775, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    5    0.9927980  0.9908888
##   15    0.9925942  0.9906310
##   25    0.9910314  0.9886544
##   35    0.9884492  0.9853877
##   45    0.9863430  0.9827234
##   52    0.9843726  0.9802318
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.
```

```
ggplot(fit_rf_2)
```



With k = 5 fold, the estimated out-of-sample accuracy for mtry = 5 is 0.9934775.

Now we apply the model to the testing set to assess performance.

```r
# Using predict() on testing set
pred_rf2_test <- predict(fit_rf_2, newdata = testing)

# Confusion Matrix to assess
confusionMatrix(testing$classe, pred_rf2_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    2  945    2    0    0
##          C    0    3  852    0    0
##          D    0    0   17  787    0
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9986   0.9968   0.9782   0.9962   1.0000
## Specificity            1.0000   0.9990   0.9993   0.9959   0.9993
## Pos Pred Value         1.0000   0.9958   0.9965   0.9789   0.9967
## Neg Pred Value         0.9994   0.9992   0.9953   0.9993   1.0000
## Prevalence             0.2849   0.1933   0.1776   0.1611   0.1831
## Detection Rate         0.2845   0.1927   0.1737   0.1605   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9993   0.9979   0.9887   0.9960   0.9996
```

The resulting accuracy rate is 0.9939, which is very close to accuracy on training set.

Predicting Quiz Cases

Now to apply the method to the quiz cases.

We will apply the same transformation and feature selection steps as we did in the training process.

```r
pml_test <- read.csv(file.path(getwd(), "data", "pml-testing.csv"))

# Same for testing data
pml_test <- pml_test %>%
  # Remove meta data
```

```
  select(-c(1:7)) %>%
  mutate(
    # transform data to numeric
    across(roll_belt:magnet_forearm_z, ~ as.numeric(.x))
    ) %>%
  # Select non-NA features
  select(problem_id, one_of(name_vec))
```

```
## Warning: Unknown columns: `classe`
```

Using the quiz data set, apply the random forest model to get prediction for each problem ID case.

```
# predict on pml_test
quiz_df <- data.frame(
  problem_id = pml_test$problem_id,
  pred_classe = predict(fit_rf_2, newdata = pml_test)
)

quiz_df
```

```
##    problem_id pred_classe
## 1           1           B
## 2           2           A
## 3           3           B
## 4           4           A
## 5           5           A
## 6           6           E
## 7           7           D
## 8           8           B
## 9           9           A
## 10         10           A
## 11         11           B
## 12         12           C
## 13         13           B
## 14         14           A
## 15         15           E
## 16         16           E
## 17         17           A
## 18         18           B
## 19         19           B
## 20         20           B
```