My Analysis

Nicole

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
# Load necessary libraries
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

# Load necessary libraries
# Load the training and testing data
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
test_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
train_data <- read.csv(train_url, na.strings = c("NA", "", " "))
test_data <- read.csv(test_url, na.strings = c("NA", "", " "))
```

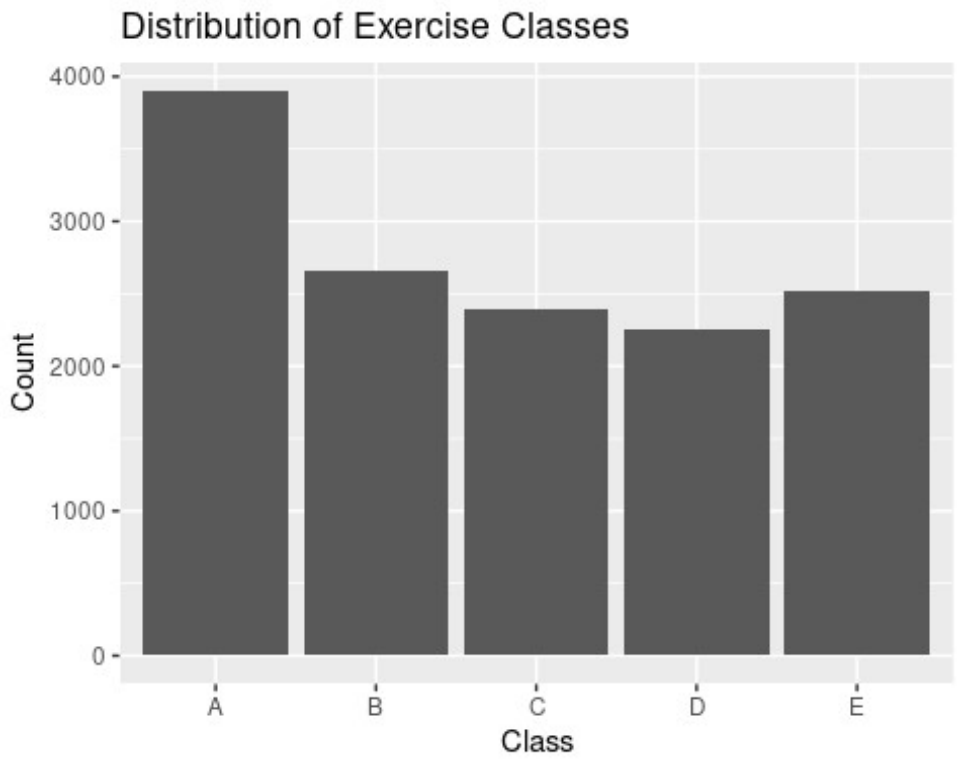
Including Plots

```
# Remove columns with mostly NA values and irrelevant columns
train_data <- train_data %>% select_if(~ sum(is.na(.)) < 0.9 *
nrow(train_data))
train_data <- train_data[, -c(1:7)] # Remove metadata columns (e.g., ID)
```

You can also embed plots, for example:

```
set.seed(123)
trainIndex <- createDataPartition(train_data$classe, p = 0.7, list = FALSE)
training <- train_data[trainIndex,]
validation <- train_data[-trainIndex,]

# Visualize the distribution of classes
ggplot(training, aes(x = classe)) +
  geom_bar() +
  labs(title = "Distribution of Exercise Classes", x = "Class", y = "Count")
```




```r
# Check correlation for feature selection
correlations <- cor(training[, sapply(training, is.numeric)])

# Train a Random Forest model
# Convert `classe` to a factor if it's not already

training$classe <- as.factor(training$classe)
validation$classe <- as.factor(validation$classe)

# Convert any character columns to numeric or remove them if irrelevant
training <- training %>% mutate_if(is.character, as.numeric)
validation <- validation %>% mutate_if(is.character, as.numeric)

# Remove columns with mostly NA values and irrelevant columns
training <- training %>% select_if(~ sum(is.na(.)) < 0.9 * nrow(training))
validation <- validation %>% select_if(~ sum(is.na(.)) < 0.9 *
nrow(validation))

# Fit the random forest model
set.seed(123)
model_rf <- randomForest(classe ~ ., data = training, importance = TRUE)

# Check model accuracy on validation set
pred_valid <- predict(model_rf, validation)
conf_matrix <- confusionMatrix(pred_valid, validation$classe)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1131    4    0    0
##          C    0    4 1022   10    4
##          D    0    0    0  954    4
##          E    0    0    0    0 1074
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9930   0.9961   0.9896   0.9926
## Specificity            0.9991   0.9992   0.9963   0.9992   1.0000
## Pos Pred Value         0.9976   0.9965   0.9827   0.9958   1.0000
## Neg Pred Value         1.0000   0.9983   0.9992   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1922   0.1737   0.1621   0.1825
## Detection Prevalence   0.2851   0.1929   0.1767   0.1628   0.1825
## Balanced Accuracy      0.9995   0.9961   0.9962   0.9944   0.9963
```

```
# Define cross-validation parameters
control <- trainControl(method = "cv", number = 5)
model_cv <- train(classe ~ ., data = training, method = "rf", trControl =
control)

# Expected out-of-sample error
print(model_cv)
```

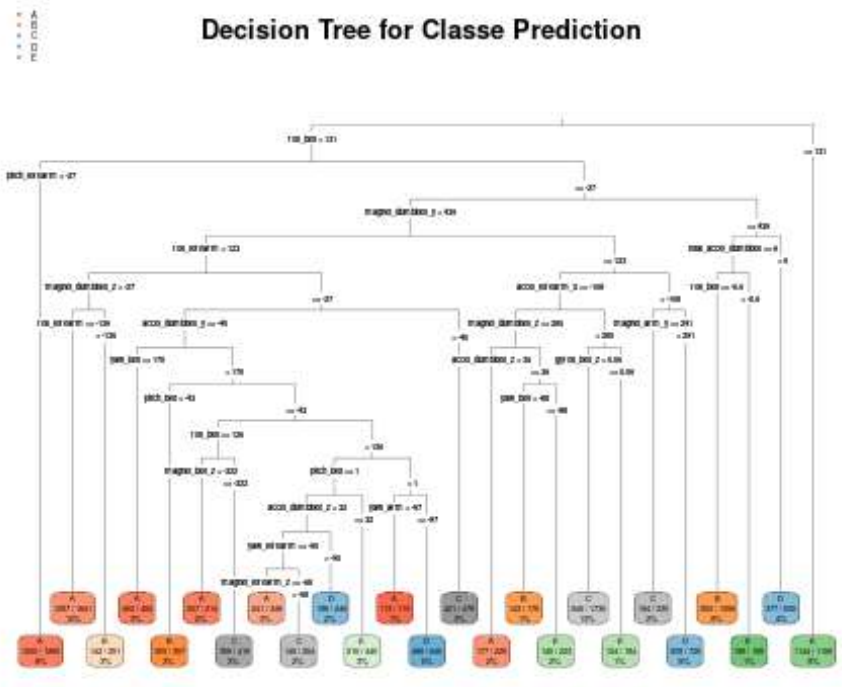
```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10991, 10988, 10990, 10989
```

```
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9901726  0.9875671
##   27    0.9909005  0.9884891
##   52    0.9859503  0.9822257
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```r
# Load necessary libraries
library(rpart)
library(rpart.plot)

# Create a single decision tree for visualization
set.seed(123)
model_tree <- rpart(classe ~ ., data = training, method = "class")

# Plot the decision tree
rpart.plot(model_tree, type = 3, extra = 102, fallen.leaves = TRUE, main =
"Decision Tree for Classe Prediction")
```




```r
# Predict on test data
predictions <- predict(model_rf, test_data)
predictions
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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

# Save predictions for submission
write.csv(predictions, "predictions.csv", row.names = FALSE)
```