Analysis of Movement Data

Ishaan Gupta

2023-12-17

Load the Data

```r
# Load training data
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
train_data <- read.csv(url(trainUrl))

# Load testing data
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
test_data <- read.csv(url(testUrl))
```

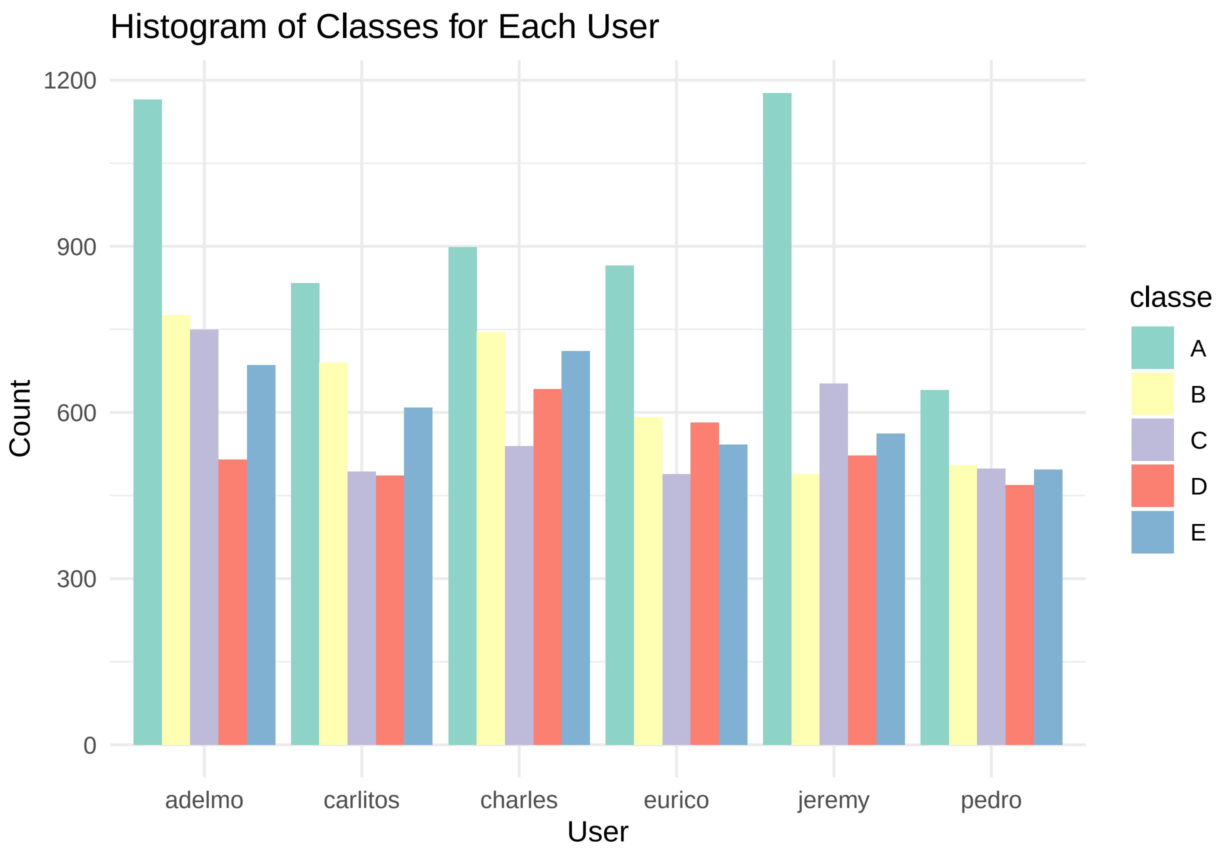
Data Exploration

The Data set used in the analysis describes six participants who were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl. Each exercise was categorized as exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

We can see the distribution of the classes for each participant in:

```r
# Assuming train_data is your data frame
library(ggplot2)

# Create a histogram for each user
ggplot(train_data, aes(x = user_name, fill = classe)) +
  geom_bar(position = "dodge", stat = "count") +
  labs(title = "Histogram of Classes for Each User",
       x = "User",
       y = "Count") +
  scale_fill_brewer(palette = "Set3") +  # You can choose a different color palette if needed
  theme_minimal()
```



Data Cleanup

We remove all the columns with many missing values.

```
library(caret)
```

```
## Loading required package: lattice
```

```
remove_cols <- nearZeroVar(train_data, saveMetrics = TRUE)
train_data <- train_data[, !remove_cols$nzv]
test_data <- test_data[, !remove_cols$nzv]
```

We remove columns with NAs

```
remove_cols <- colSums(is.na(train_data)) == 0
train_data <- train_data[, remove_cols]
test_data <- test_data[, remove_cols]
```

We remove variables with timestamp or user_name since they are not likely to affect the classe. Also, we remove the 'X' column since it is just an index field.

```
# Identify and exclude columns with names starting with "X" or containing "timestamp" or "user_name"
remove_cols <- grepl("timestamp|user_name|X", names(train_data))
train_data <- train_data[, !remove_cols]
test_data <- test_data[, !remove_cols]
```

Partitioning Train and Validation Data

We split the preprocessed training set into a training dataset (70%) and a validation dataset (30%). The validation dataset will be used for cross-validation in subsequent steps.

```
training_idx <- createDataPartition(train_data$classe, p = 0.70, list = FALSE)
val_data <- train_data[-training_idx, ]
train_data <- train_data[training_idx, ]
```

Random Forest

We will use the Random Forest algorithm to predict the class because it automatically selects important variables and is robust to correlated covariates & outliers in general.
We will use 5-fold cross validation when applying the algorithm.

```
forest <- train(classe ~ ., data = train_data, method = "rf", trControl = trainControl(method = "cv", nu
forest
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: ’A’, ’B’, ’C’, ’D’, ’E’
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10988, 10991, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9938126  0.9921729
##   27    0.9965060  0.9955805
##   53    0.9935208  0.9918043
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Validation

Now, we test the performance of our random forest model on the validation set

```r
val_results <- predict(forest, val_data)
confusionMatrix(factor(val_data$classe), val_results)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    2 1136    1    0    0
##          C    0    1 1025    0    0
##          D    0    0    3  961    0
##          E    0    0    0    4 1078
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9981          
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2848          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9976          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9991   0.9961   0.9959   1.0000
## Specificity            1.0000   0.9994   0.9998   0.9994   0.9992
## Pos Pred Value         1.0000   0.9974   0.9990   0.9969   0.9963
## Neg Pred Value         0.9995   0.9998   0.9992   0.9992   1.0000
## Prevalence             0.2848   0.1932   0.1749   0.1640   0.1832
## Detection Rate         0.2845   0.1930   0.1742   0.1633   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9994   0.9992   0.9980   0.9976   0.9996
```

```r
accuracy <- postResample(val_results, factor(val_data$classe))
```

The estimated accuracy of our model is 99.8130841%.

Test Set

Now we can test our model on the final test set

```r
predict(forest, test_data)
```

```
##  [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
```