Course Project ML

Hazem Haffouz

2024-01-02

Leveraging Accelerometer Data to Predict Exercise Quality: A Machine Learning Approach"

Introduction

This project aims to harness accelerometer data from devices like Jawbone Up, Nike FuelBand, and Fitbit to predict the manner in which exercises are performed. Utilizing data collected from accelerometers on the belt, forearm, arm, and dumbbell of participants, the project endeavors to classify barbell lifts into one of five categories, representing the quality of the exercise performed. This report outlines the methodology adopted, from exploratory data analysis (EDA) through to predictive modeling, culminating in the application of a Random Forest model to predict exercise quality in test cases.

```
# 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 4.7-1.1
```

```
## 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
```

```
library(rpart)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.3.2

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

##
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':
##
##     importance
```

```
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.3.2

##
## Attaching package: 'psych'

## The following object is masked from 'package:randomForest':
##
##     outlier

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
# Data Loading
training_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training_data <- read.csv(training_url, na.strings=c("NA","#DIV/0!", ""))
testing_data <- read.csv(testing_url, na.strings=c("NA","#DIV/0!", ""))
```

Exploratory Data Analysis (EDA)

The EDA commenced with a summary and structural analysis of the training and testing datasets, which included diverse accelerometer measurements. Key descriptive statistics were reviewed, and the presence of null or missing values was assessed. Unique value analysis helped in understanding the data's diversity. A critical aspect of the EDA was visualizing the distribution of the 'classe' variable in the training data using a bar plot, providing insights into the balance of classes.

```
# Exploratory Data Analysis
summary(training_data[1:5])
```

```
##        X          user_name         raw_timestamp_part_1 raw_timestamp_part_2
##  Min.   :    1   Length:19622       Min.   :1.322e+09    Min.   :   294
##  1st Qu.: 4906   Class :character   1st Qu.:1.323e+09    1st Qu.:252912
##  Median : 9812   Mode  :character   Median :1.323e+09    Median :496380
##  Mean   : 9812                      Mean   :1.323e+09    Mean   :500656
##  3rd Qu.:14717                      3rd Qu.:1.323e+09    3rd Qu.:751891
##  Max.   :19622                      Max.   :1.323e+09    Max.   :998801
##  cvtd_timestamp
##  Length:19622
##  Class :character
##  Mode  :character
##
##
##
```

```
describe(training_data[1:5])
```

```
##                      vars     n         mean        sd       median
## X                       1 19622 9.811500e+03   5664.53       9811.5
## user_name*              2 19622 3.350000e+00      1.70          3.0
## raw_timestamp_part_1    3 19622 1.322827e+09 204927.68 1322832920.0
## raw_timestamp_part_2    4 19622 5.006561e+05 288222.88     496380.0
## cvtd_timestamp*         5 19622 1.098000e+01      5.75         11.0
##                          trimmed       mad        min        max  range  skew
## X                    9.811500e+03   7272.89          1      19622  19621  0.00
## user_name*           3.310000e+00      2.97          1          6      5  0.07
## raw_timestamp_part_1 1.322836e+09 237001.02 1322489605 1323095081 605476 -0.16
## raw_timestamp_part_2 5.007597e+05 367764.86        294     998801 998507  0.00
## cvtd_timestamp*      1.101000e+01      7.41          1         20     19 -0.05
##                      kurtosis      se
## X                       -1.20   40.44
## user_name*              -1.26    0.01
## raw_timestamp_part_1    -1.01 1462.95
## raw_timestamp_part_2    -1.19 2057.58
## cvtd_timestamp*         -1.26    0.04
```

```
sum(is.null(training_data))
```

```
## [1] 0
```

```r
sum(is.na(training_data))
```

```
## [1] 1925102
```

```r
head(unique(training_data$columnName))
```

```
## NULL
```

```r
summary(testing_data[1:5])
```

```
##        X          user_name         raw_timestamp_part_1 raw_timestamp_part_2
##  Min.   : 1.00   Length:20          Min.   :1.322e+09    Min.   : 36553
##  1st Qu.: 5.75   Class :character   1st Qu.:1.323e+09    1st Qu.:268655
##  Median :10.50   Mode  :character   Median :1.323e+09    Median :530706
##  Mean   :10.50                      Mean   :1.323e+09    Mean   :512167
##  3rd Qu.:15.25                      3rd Qu.:1.323e+09    3rd Qu.:787738
##  Max.   :20.00                      Max.   :1.323e+09    Max.   :920315
##  cvtd_timestamp
##  Length:20
##  Class :character
##  Mode  :character
##
##
##
```

```r
describe(testing_data[1:5])
```

```
##                       vars  n         mean        sd       median      trimmed
## X                        1 20         10.5      5.92         10.5 1.050000e+01
## user_name*               2 20          4.2      1.47          5.0 4.310000e+00
## raw_timestamp_part_1     3 20 1322777592.3 230560.29 1322673138.5 1.322774e+09
## raw_timestamp_part_2     4 20     512167.4 303068.11     530705.5 5.197185e+05
## cvtd_timestamp*          5 20          6.8      3.37          7.0 6.940000e+00
##                             mad        min        max  range  skew kurtosis
## X                          7.41          1         20     19  0.00    -1.38
## user_name*                 1.48          1          6      5 -0.70    -0.75
## raw_timestamp_part_1 258053.20 1322489635 1323095002 605367  0.32    -1.49
## raw_timestamp_part_2 394438.32      36553     920315 883762 -0.14    -1.52
## cvtd_timestamp*            4.45          1         11     10 -0.18    -1.54
##                             se
## X                         1.32
## user_name*                0.33
## raw_timestamp_part_1 51554.85
## raw_timestamp_part_2 67768.09
## cvtd_timestamp*           0.75
```

```r
sum(is.null(testing_data))
```

```
## [1] 0
```

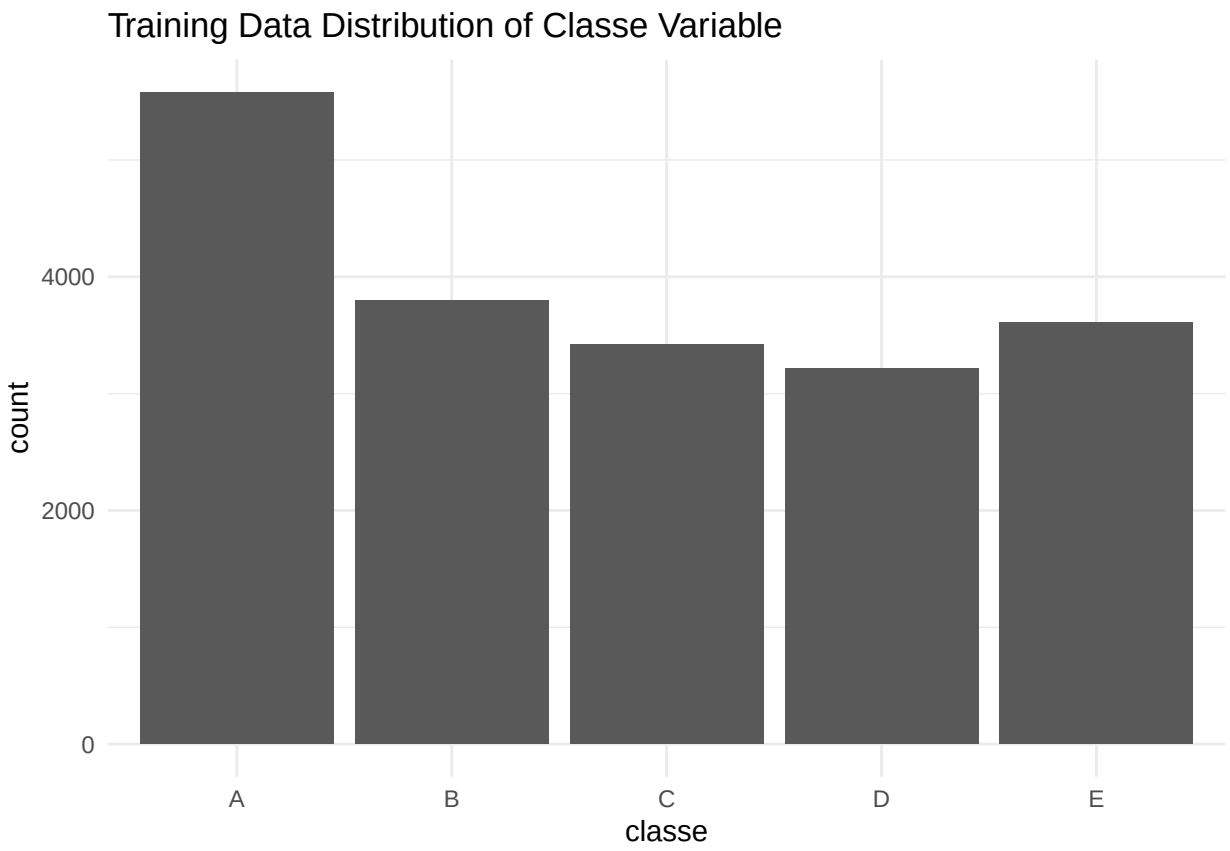
```r
sum(is.na(testing_data))
```

```
## [1] 2000
```

```r
head(unique(testing_data$columnName))
```

```
## NULL
```

```r
ggplot(training_data, aes(classe)) + geom_bar() + theme_minimal() + labs(title="Training Data Distributi
```



Data Processing Summary

Data preprocessing involved several crucial steps to prepare the data for modeling. Columns with excessive missing values were removed, ensuring data quality and integrity. Irrelevant columns, particularly the first seven, were discarded to focus on more pertinent features. The transformation of categorical variables into factors and the standardization of numerical columns were also integral to the preprocessing phase. This process ensured a consistent and analytically suitable dataset for training machine learning models.

```r
# Removing columns with too many missing values
threshold <- 0.6 * nrow(training_data)
training_data <- training_data[, colSums(is.na(training_data)) < threshold]
testing_data <- testing_data[, colnames(testing_data) %in% colnames(training_data)]

# Removing irrelevant columns (first 7 columns in this case)
training_data <- select(training_data, -(1:7))
```

```r
testing_data <- select(testing_data, -(1:7))

# Converting categorical variables to factors
training_data <- mutate_if(training_data, is.character, as.factor)
testing_data <- mutate_if(testing_data, is.character, as.factor)

# Standardizing numerical columns
num_cols <- sapply(training_data, is.numeric)
preProcValues <- preProcess(training_data[, num_cols], method = c("center", "scale"))
training_data[, num_cols] <- predict(preProcValues, training_data[, num_cols])
testing_data[, num_cols] <- predict(preProcValues, testing_data[, num_cols])
```

Data Modeling

Two models were explored for this classification task: a Decision Tree (CART) and a Random Forest. Both models were trained on a subset of the processed training data, with cross-validation (5-fold) employed to ensure robustness and generalizability. The Decision Tree served as a baseline, offering interpretability, while the Random Forest model, known for its accuracy and ability to handle complex interactions, was the primary focus.

```r
# Splitting the dataset into training and testing subsets
set.seed(123)
splitIndex <- createDataPartition(training_data$classe, p = 0.75, list = FALSE)
training_subset <- training_data[splitIndex, ]
testing_subset <- training_data[-splitIndex, ]
```

```r
# Decision Tree Model
fitControl <- trainControl(method = "cv", number = 5)
decisionTreeModel <- train(classe ~ ., data = training_subset, method = "rpart", trControl = fitControl
```

```r
# Random Forest Model
randomForestModel <- randomForest(classe ~ ., data = training_subset)
```

```r
# Printing model summaries
print(decisionTreeModel)
```

```
## CART
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11775, 11774, 11775, 11774
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03598215  0.5016312  0.34888168
##   0.05949555  0.4411591  0.25097067
##   0.11497199  0.3489572  0.09871325
```

```
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03598215.
```

```
print(randomForestModel)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training_subset)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4180    3    1    0    1 0.001194743
## B    9 2832    7    0    0 0.005617978
## C    0    8 2556    3    0 0.004285158
## D    0    0   30 2378    4 0.014096186
## E    0    0    3    8 2695 0.004065041
```

Analysis of Model Findings

The Decision Tree model provided moderate accuracy, suggesting some complexity in data that a single tree could not fully capture. In contrast, the Random Forest model demonstrated exceptional performance, with a notably low Out-Of-Bag error rate and high accuracy across all classes, as evidenced by the confusion matrix. This stark difference in performance highlighted the Random Forest's superiority in handling this dataset's intricacies.

```
# Making predictions using the Random Forest model
test_predictions <- predict(randomForestModel, newdata = testing_data)

# Display the predictions
print(test_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
```

Prediction Results and Conclusion

The final stage involved deploying the Random Forest model on the test data, resulting in predictions for 20 different cases. The model's predictions showcased its ability to effectively classify the exercise quality across various categories. The success of these predictions underscores the potential of machine learning in enhancing our understanding and assessment of physical activity through wearable technology.

This project not only demonstrates the practical application of machine learning in a health and fitness context but also sets the stage for further exploration into the optimization of training algorithms and the integration of such models into wearable technology for real-time feedback and analysis.