Peer-graded Assignment- Prediction Assignment Writeup

Peer-graded Assignment - Prediction Assignment Writeup [Practical Machine Learning - Johns Hopkins University]

Rohith Mohan

March 30, 2024

Contents

Overview

This document serves as the conclusive summary of the Peer Assessment initiative within the Practical Machine Learning curriculum offered through the Coursera John's Hopkins University Data Science Specialization. Crafted and executed in RStudio, leveraging its knitr functionalities, the report is presented in both html and markdown formats. The primary objective of this endeavor is to forecast the performance of six participants in completing designated exercises. Employing a machine learning algorithm trained on the 'classe' variable within the training dataset, predictions are made on the performance of 20 test cases contained in the test data.

Introduction

In today's era, the accessibility of devices like Jawbone Up, Nike FuelBand, and Fitbit enables the collection of vast amounts of personal activity data at a relatively low cost. These devices are emblematic of the quantified self movement—a community of enthusiasts who regularly track various metrics about themselves to enhance their health, identify behavioral patterns, or simply due to their fascination with technology. While individuals often quantify the quantity of a particular activity they engage in, they seldom measure the quality of their performance.

This project aims to leverage data gathered from accelerometers placed on the belt, forearm, arm, and dumbbell of six participants. These individuals were tasked with executing barbell lifts both correctly and incorrectly in five distinct manners.

For further details, please refer to the following website: http://groupware.les.inf.puc-rio.br/har.

Data

The training and test datasets for this project can be accessed through the following links:

Training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

Test data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

These datasets originate from the following source: http://groupware.les.inf.puc-rio.br/har

The data's full reference is provided as:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. "Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13)". Stuttgart, Germany: ACM SIGCHI, 2013.

Data Loading and Cleaning

```
library(lattice)
library(caret)
library(corrplot)
library(randomForest)
library(rattle)
library(RColorBrewer)
library(ggplot2)
library(rpart)
library(rpart.plot)
set.seed(666)
```

Data loading

```
# Data loading
trainingdata <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingdata <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"


trainingdataset <- read.csv(url(trainingdata), na.strings = c("NA",""))
testingdataset  <- read.csv(url(testingdata), na.strings = c("NA",""))
```

Dataset Partitioning

After loading the data, we'll split the training set, using 75% for model training and the remaining 25% for validation.

```
# Dataset Partitioning
TrainingPart <- createDataPartition(trainingdataset$classe, p=0.75, list=FALSE)
trainingdata <- trainingdataset[TrainingPart, ]
testingdata <- trainingdataset[-TrainingPart, ]
dim(trainingdata)
```

```
## [1] 14718   160
```

```
dim(testingdata)
```

```
## [1] 4904  160
```

Filtering to the 95% threshhold and removing Nulls/Near-Zero-Variance

```
# Filtering to the 95% threshold and removing Nulls/Near-Zero-Variance
NearZeroVariables <- nearZeroVar(trainingdata)
trainingdata <- trainingdata[, -NearZeroVariables]
testingdata <- testingdata[, -NearZeroVariables]

Nulls <- sapply(trainingdata, function(x) mean(is.na(x))) > 0.95
trainingdata <- trainingdata[, Nulls == FALSE]
testingdata <- testingdata[, Nulls == FALSE]

# Remove Id Variables
trainingdata <- trainingdata[, -(1:5)]
testingdata <- testingdata[, -(1:5)]

dim(trainingdata)
```

```
## [1] 14718    54
```

```
dim(testingdata)
```

```
## [1] 4904   54
```

The number of variables has been reduced from 160 to 54.

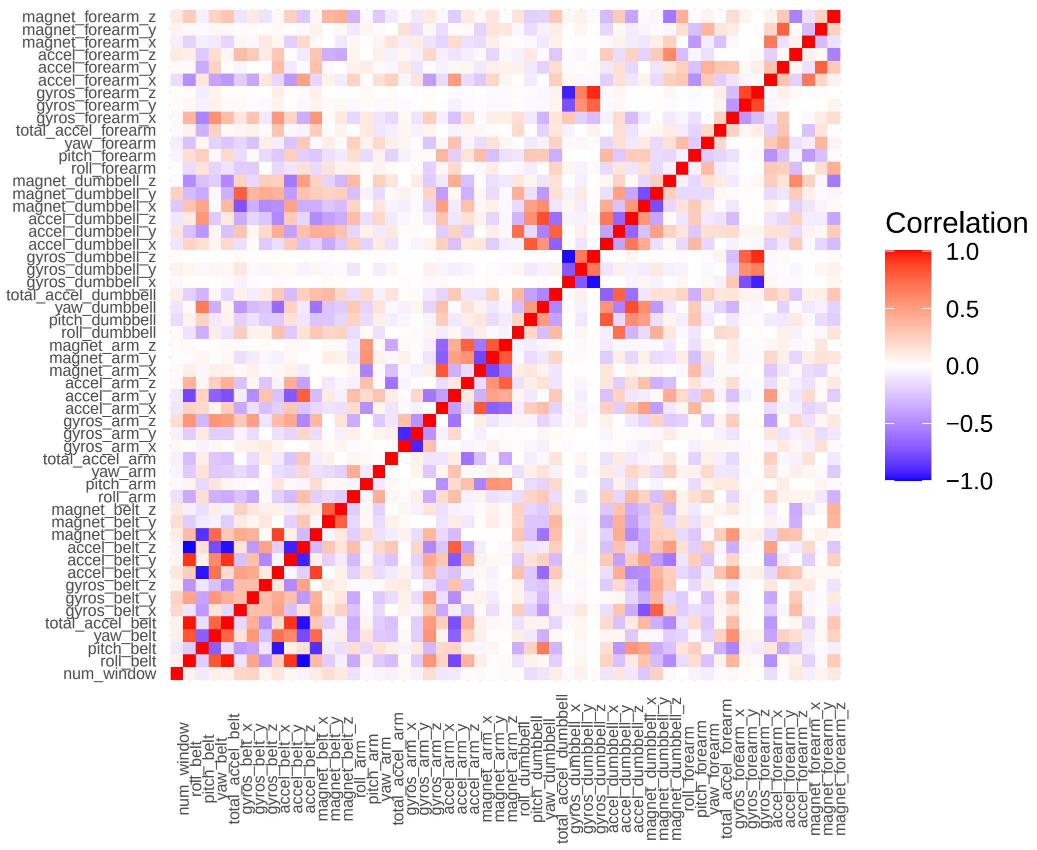
Model Analysis

Correlation Analysis

```
# Calculate correlation matrix
correlationmatrix <- cor(trainingdata[, -54])

# Convert correlation matrix to tidy format
correlationmatrix_tidy <- as.data.frame(as.table(correlationmatrix))
colnames(correlationmatrix_tidy) <- c("Variable1", "Variable2", "Correlation")

# Plot heatmap using ggplot2
ggplot(correlationmatrix_tidy, aes(x = Variable1, y = Variable2, fill = Correlation)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme_minimal() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x = element_text(angle = 90, vjust = 1, size = 6),
        axis.text.y = element_text(size = 6)) +
  coord_fixed()
```



The above correlation matrix shows each cell representing the correlation coefficient between two variables, with color intensity indicating the strength and direction of the correlation. Blue denotes negative correlation, red indicates positive correlation, and white suggests no correlation. Clusters of similarly colored cells highlight groups of correlated variables, facilitating the understanding of relationships between variables for data analysis and modeling.

Prediction Models

Random Forest Model

```
set.seed(666)
controlrandomforest <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fitrandomforest <- train(classe ~ ., data = trainingdata, method = "rf",
                          trControl = controlrandomforest, verbose = FALSE)
fitrandomforest$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.21%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    6 2839    2    1    0 0.0031601124
## C    0    4 2560    3    0 0.0027269186
## D    0    0   10 2402    0 0.0041459370
## E    0    0    0    3 2703 0.0011086475
```

```
predict_RF <- predict(fitrandomforest, newdata = testingdata)
confusionmatrixrf <- confusionMatrix(predict_RF, factor(testingdata$classe))
confusionmatrixrf
```

Predictions on Test Data

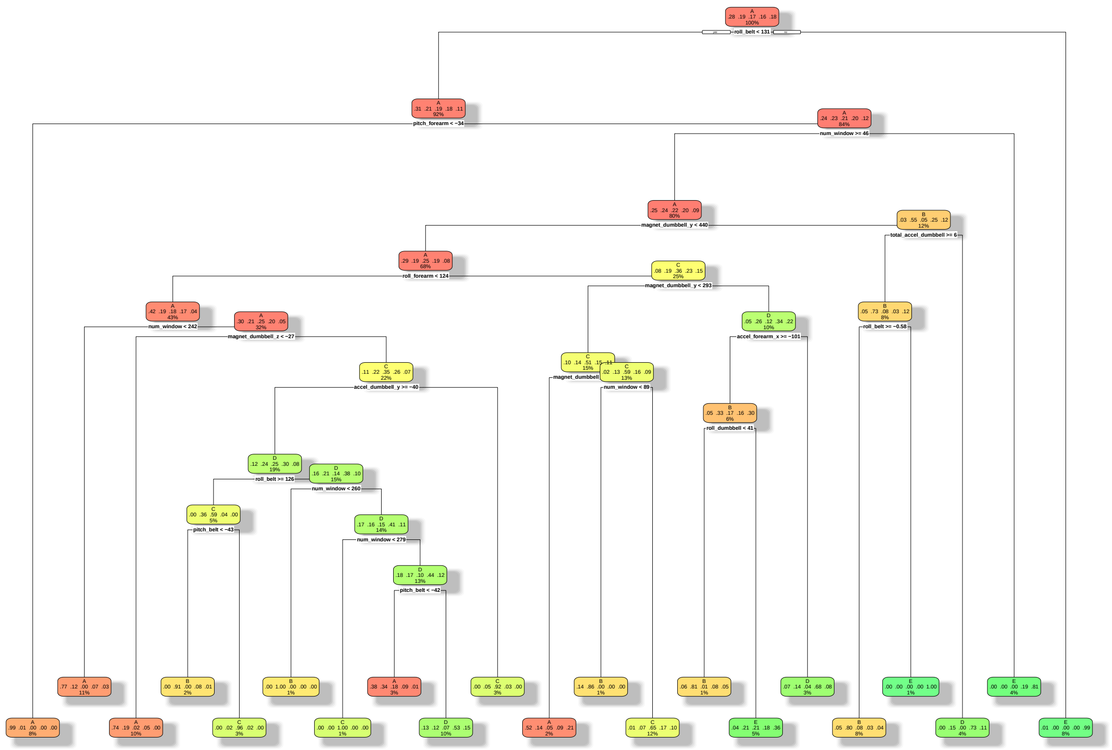
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  948    2    0    0
##          C    0    0  853    4    0
##          D    0    0    0  799    0
##          E    0    0    0    1  901
##
## Overall Statistics
##
##                Accuracy : 0.9984
##                  95% CI : (0.9968, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9977   0.9938   1.0000
## Specificity            0.9997   0.9995   0.9990   1.0000   0.9998
## Pos Pred Value         0.9993   0.9979   0.9953   1.0000   0.9989
## Neg Pred Value         1.0000   0.9997   0.9995   0.9988   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1739   0.1629   0.1837
## Detection Prevalence   0.2847   0.1937   0.1748   0.1629   0.1839
## Balanced Accuracy      0.9999   0.9992   0.9983   0.9969   0.9999
```

Decision Tree Model

```
set.seed(666)
fit_decision_tree <- rpart(classe ~ ., data = trainingdata, method="class")

rpart.plot(fit_decision_tree, box.palette = "RdYlGn", shadow.col = "gray")
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
predict_decision_tree <- predict(fit_decision_tree, newdata = testingdata, type="class")
conf_matrix_decision_tree <- confusionMatrix(predict_decision_tree, factor(testingdata$classe))
conf_matrix_decision_tree
```

Predictions on Test Data

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1253  203   33   72   45
##          B   35  513   17   13   15
##          C    7   54  694  118   56
##          D   77  130   57  515   94
##          E   23   49   54   86  691
##
## Overall Statistics
##
##                Accuracy : 0.7476
##                  95% CI : (0.7351, 0.7597)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6794
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8982   0.5406   0.8117   0.6405   0.7669
## Specificity            0.8994   0.9798   0.9420   0.9127   0.9470
## Pos Pred Value         0.7802   0.8651   0.7470   0.5899   0.7652
## Neg Pred Value         0.9569   0.8989   0.9595   0.9283   0.9475
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2555   0.1046   0.1415   0.1050   0.1409
## Detection Prevalence   0.3275   0.1209   0.1894   0.1780   0.1841
## Balanced Accuracy      0.8988   0.7602   0.8768   0.7766   0.8570
```

Model Accuracy

In this report, the Random Forest model demonstrates the highest accuracy, achieving a remarkable value of 99.84%. We can present the model's predictions confidently based on this performance.

```
# Get predictions for the 20 observations of the original pml-testing.csv

predictionmodel <- as.data.frame(predict(fitrandomforest, newdata = testingdataset))
predictionmodel
```

```
##    predict(fitrandomforest, newdata = testingdataset)
## 1                                                   B
## 2                                                   A
## 3                                                   B
## 4                                                   A
## 5                                                   A
## 6                                                   E
## 7                                                   D
## 8                                                   B
## 9                                                   A
## 10                                                  A
## 11                                                  B
## 12                                                  C
## 13                                                  B
## 14                                                  A
## 15                                                  E
## 16                                                     E
## 17                                                     A
## 18                                                     B
## 19                                                     B
## 20                                                     B
```