Practical Machine Learning Course Project

Raihan Ahmed

29/07/2022

Project Overview

Personal fitness trackers are commonly used to track individual fitness data. Users can quantify how much of a particular movement they make, but not how well they did the movement. This analysis will use accelerometer data from six participants. Accelerometers were worn on the belt, forearm, arm or on a dumbell. Participants were instructed to perform dumbell bicep curls (10 reps) incorrectly and correctly, in five different ways. I will use the data to predict the manner in which they did the exercise.

Data Preparation

Load necessary packages

```
library(lattice)
library(ggplot2)
library(caret)
library(kernlab)
library(rattle)
library(reshape2)
library(corrplot)
library(ggcorrplot)
library(randomForest)
library(rpart)
set.seed(3232)
```

load the test and training data

```
trainData <- read.csv("./pml-training.csv")
testData <- read.csv("./pml-testing.csv")
dim(trainData)
```

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

```
dim(testData)
```

```
## [1]  20 160
```

Data cleaning - Here, variables with nearly zero variance or that are almost always NA, and the columns containing summary statistics or irrelevant data will be removed.

```
trainClean <- trainData[,colMeans(is.na(trainData))< .9]
trainClean <- trainClean[,-c(1:7)]
nvz <- nearZeroVar(trainClean)
trainClean <- trainClean[,-nvz]
dim(trainClean)
```

```
## [1] 19622    53
```

Performed correlation matrix, see appendix, fig 1. The correlation matrix shows there are variables that are correlated, which must be removed

```
c <- findCorrelation(corMat, cutoff = .90)
trainClean <- trainClean[,-c]
```

Split the data into training (70%) and validation (30%)

```
inTrain <- createDataPartition(y=trainClean$classe, p=0.7, list=FALSE)
train <- trainClean[inTrain,]
valid <- trainClean[-inTrain,]
# Create a control for 3 fold validation
control <- trainControl(method="cv", number=3, verboseIter = FALSE)
```

Building the models

Random Forests

```
# Fit the model on train using random forest
train$classe <- factor(train$classe)
RFfit <- randomForest(classe~., data=train, method="class", trControl = control, tuneLength = 5)
RFfit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train, method = "class",      trControl = control, tuneLe
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.58%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   15 2638    5    0    0 0.007524454
## C    0   21 2373    2    0 0.009599332
## D    0    0   25 2227    0 0.011101243
## E    0    0    2    6 2517 0.003168317
```

```
RFpred<- predict(RFfit, valid) # predict on the valid data set.
cmRF <- confusionMatrix(RFpred, as.factor(valid$classe))
cmRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1666    3    0    0    0
##          B    7 1136    7    0    0
##          C    0    0 1019   11    2
##          D    0    0    0  951    2
##          E    1    0    0    2 1078
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9974   0.9932   0.9865   0.9963
## Specificity            0.9993   0.9971   0.9973   0.9996   0.9994
## Pos Pred Value         0.9982   0.9878   0.9874   0.9979   0.9972
## Neg Pred Value         0.9981   0.9994   0.9986   0.9974   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2831   0.1930   0.1732   0.1616   0.1832
## Detection Prevalence   0.2836   0.1954   0.1754   0.1619   0.1837
## Balanced Accuracy      0.9973   0.9972   0.9953   0.9931   0.9978
```

```
table(RFpred, valid$classe)
```

```
##
## RFpred    A    B    C    D    E
##      A 1666    3    0    0    0
##      B    7 1136    7    0    0
##      C    0    0 1019   11    2
##      D    0    0    0  951    2
##      E    1    0    0    2 1078
```

The estimated accuracy is 0.9937, and oos error is 0.0063

Decision Tree

```
DTfit <- train(classe~., data=train, method='rpart', trControl=control, tuneLength=5)
DTfit
```

```
## CART
##
## 13737 samples
##    45 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9159, 9158, 9157
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01680907  0.5994069  0.4954366
##   0.02064897  0.5829538  0.4741508
##   0.02949853  0.5483727  0.4231955
##   0.03336385  0.5199125  0.3862614
##   0.06593938  0.3467870  0.1029266
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01680907.
```

```
DTpred <- predict(DTfit, valid)
cmDT <- confusionMatrix(DTpred, factor(valid$classe))
cmDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1221  202    8   96   47
##          B   28  475   47   19  158
##          C  393  389  866  368  433
##          D   32   71  104  403   90
##          E    0    2    1   78  354
##
## Overall Statistics
##
##                Accuracy : 0.564
##                  95% CI : (0.5512, 0.5767)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4511
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.7294  0.41703   0.8441  0.41805  0.32717
## Specificity            0.9162  0.94690   0.6742  0.93965  0.98314
## Pos Pred Value         0.7757  0.65337   0.3536  0.57571  0.81379
## Neg Pred Value         0.8949  0.87127   0.9534  0.89180  0.86642
## Prevalence             0.2845  0.19354   0.1743  0.16381  0.18386
## Detection Rate         0.2075  0.08071   0.1472  0.06848  0.06015
## Detection Prevalence   0.2675  0.12353   0.4161  0.11895  0.07392
## Balanced Accuracy      0.8228  0.68197   0.7591  0.67885  0.65515
```

```
table(DTpred, valid$classe)
```

```
##
## DTpred    A    B    C    D    E
##      A 1221  202    8   96   47
##      B   28  475   47   19  158
##      C  393  389  866  368  433
##      D   32   71  104  403   90
##      E    0    2    1   78  354
```

The estimated accuracy is 0.5869, and oos error is 0.4131 The decision tree figure is available in the appendix, fig2.

Support Vector Machine

```
SVMfit <- train(classe~., data=train, method="svmLinear", trControl = control, tuneLength = 5, verbose =
SVMfit
```

```
## Support Vector Machines with Linear Kernel
##
## 13737 samples
##    45 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9158, 9157, 9159
## Resampling results:
##
##   Accuracy   Kappa
##   0.7527856  0.6855331
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
SVMpred <- predict(SVMfit, valid)
SVMcm <- confusionMatrix(SVMpred, factor(valid$classe))
SVMcm
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    A    B    C    D    E
##          A 1503  177   87   82   55
##          B   55  813  107   52  162
##          C   61   49  753  116   77
##          D   50   28   48  667  108
##          E    5   72   31   47  680
##
## Overall Statistics
##
##                Accuracy : 0.7504
##                  95% CI : (0.7391, 0.7614)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6826
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8978   0.7138   0.7339   0.6919   0.6285
## Specificity            0.9048   0.9208   0.9376   0.9524   0.9677
## Pos Pred Value         0.7894   0.6838   0.7131   0.7403   0.8144
## Neg Pred Value         0.9570   0.9306   0.9435   0.9404   0.9204
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2554   0.1381   0.1280   0.1133   0.1155
## Detection Prevalence   0.3235   0.2020   0.1794   0.1531   0.1419
## Balanced Accuracy      0.9013   0.8173   0.8358   0.8222   0.7981
```

```
table(SVMpred,valid$classe)
```

```
##
## SVMpred    A    B    C    D    E
##       A 1503  177   87   82   55
##       B   55  813  107   52  162
##       C   61   49  753  116   77
##       D   50   28   48  667  108
##       E    5   72   31   47  680
```

The estimated accuracy is 0.7499, and oos error is 0.2501

Selecting the most accurate method to use on the test set.

Random forests method was 99.37% accurate. Decision trees method was 58.69% accurate, and Support Vector Machine method was 74.99% accurate. In this case the data indicates that the most accurate model for predicting our test set is Random Forest

Predicting the test data set

```
TestPred <- predict(RFfit, testData)
TestPred
```

```
##  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
```

#Appendix fig. 1 correlation matrix

```
corMat <- cor(trainClean[sapply(trainClean, is.numeric)])
melt <- melt(corMat, as.is=TRUE)
View(melt)
corPlot <- ggplot(data=melt, aes(x=Var1, y=Var2, fill=value)) +
        geom_tile(color="white") +
        scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
         theme(axis.text.x = element_text(angle = 90, vjust = 1,
    size = 6, hjust = 1), axis.text.y=element_text(size=6))+
 coord_fixed()
corPlot
```

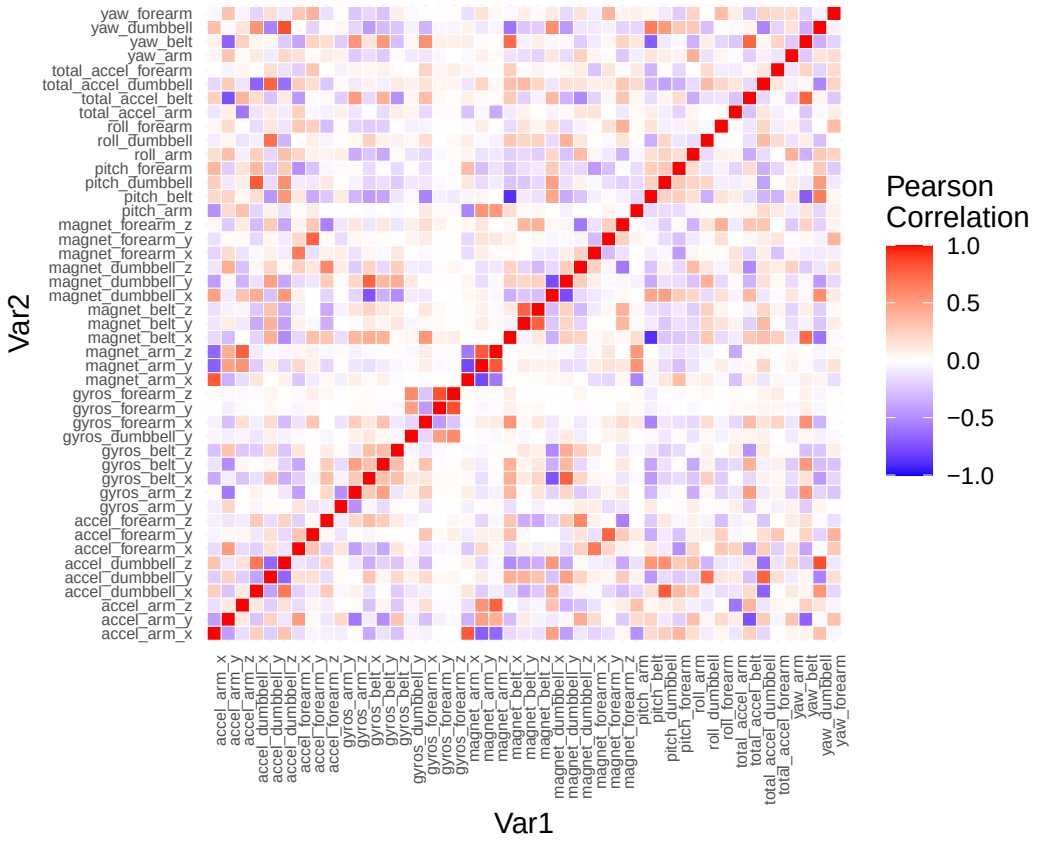
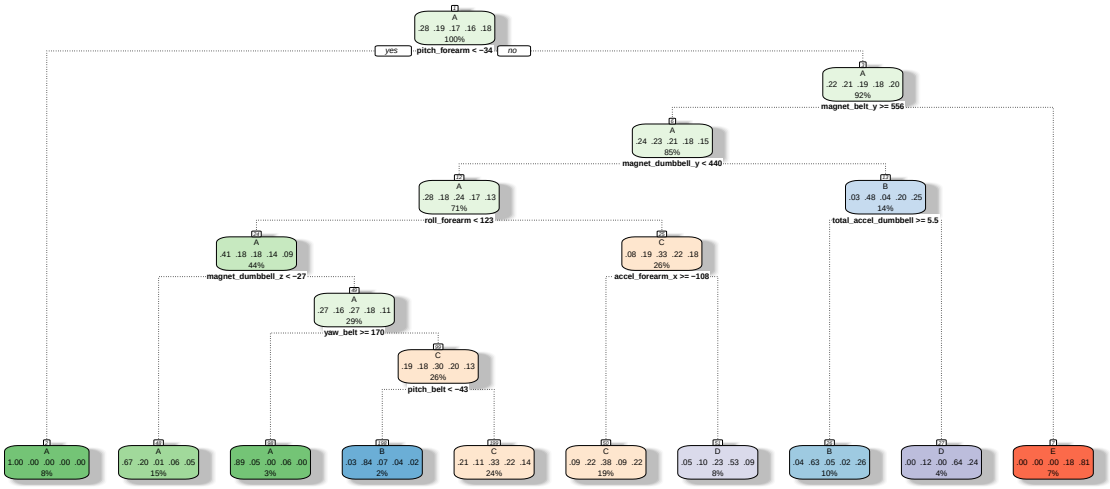


fig. 2 decision tree

```
fancyRpartPlot(DTfit$finalModel)
```



Rattle 2022−Jul−27 15:45:03 daniellec