Human Activity Recognition: Data From Wearable Devices

jymaze

Executive Summary

Materials

Human Activity Recognition - HAR - has emerged as a key research area in the last years. Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. There are many potential applications for HAR, such as: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.This small project uses a dataset with 5 classes (sitting-down, standing-up, standing, walking, and sitting) collected on 8 hours of activities of 4 healthy subjects. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Methods

The dataset was split into a training set and a test set of size 70% and 30% of the original dataset, respectively. A random forest model (300 trees) was then fitted to the training set using a 3-fold cross-validation. The variable importance in the resulting model was also calculated in term of accuracy.

Results

The random forest model yielded an out-of-bag estimate of error rate of 0.76%. Analysis of variables importance showed that the variables roll_belt, pitch_forearm, and yaw_belt, were the most important. The existence of significant correlation between some variables may have rendered the measure of importance inaccurate though. Prediction accuracy on the test set was excellent, at 0.9944. The random Forest model was highly performant on this dataset.

A. Getting and Cleaning the Data

```
## Loading required package: lattice
## Loading required package: ggplot2
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

We load the data from the csv file.

```
d <- read.csv(url_train, stringsAsFactors=FALSE)
dim(d)
```

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

The dataset comprises 19622 rows and 160 columns (variables).

We suppress the 7 first columns as they are not relevant to our analysis. We then isolate the vector of values to be predicted (‘classe’). In the rest of the dataset, we keep only the columns with numerical values and for which no data is missing.

```
d <- d[ ,8:ncol(d)]
d[,'classe'] <- as.factor(d[ ,'classe'])
classe <- d[ ,'classe'] # save the classe column
d <- d[ ,sapply(d, is.numeric)] # as side-effect, this deletes the class column

# helper function finding the proportion of missing value for a vector
find_na_columns <- function(x) {
    mean(is.na(x))
}

d <- d[ ,sapply(d, find_na_columns) == 0] # keep columns with no NA
```

Data cleaning is complete. We now split the dataset into a training set (70%) and a test set (30%). We apply the same partition to the ‘classe’ vector.

```
set.seed(3456) # for reproducibility
inTrain <- createDataPartition(y=classe, p=0.7, list=FALSE)
train_d <- d[inTrain, ]
test_d <- d[-inTrain, ]
train_classe <- classe[inTrain]
test_classe <- classe[-inTrain]
```

B. Fitting a Random Forest Model

We fit a 300 tree random forest model using a 3-fold cross validation on the training set. The predicted value is the activity quality (‘classe’ vector), and the predictors are the activity monitors in the training set. A random forest model was chosen for its excellent accuracy among current algorithms, its capacity to handle thousands of input variables without variable deletion, and it capacity to estimate what variables are important in the classification. It also generates an internal unbiased estimate of the out-of-sample error rate as the forest building progresses.

```
fitCtrl <- trainControl(method = 'cv', number = 3)
model_rf <- train(x=train_d, y=train_classe, method = 'rf', ntree=300, importance=TRUE, t
rControl = fitCtrl)
model_rf$finalModel
```

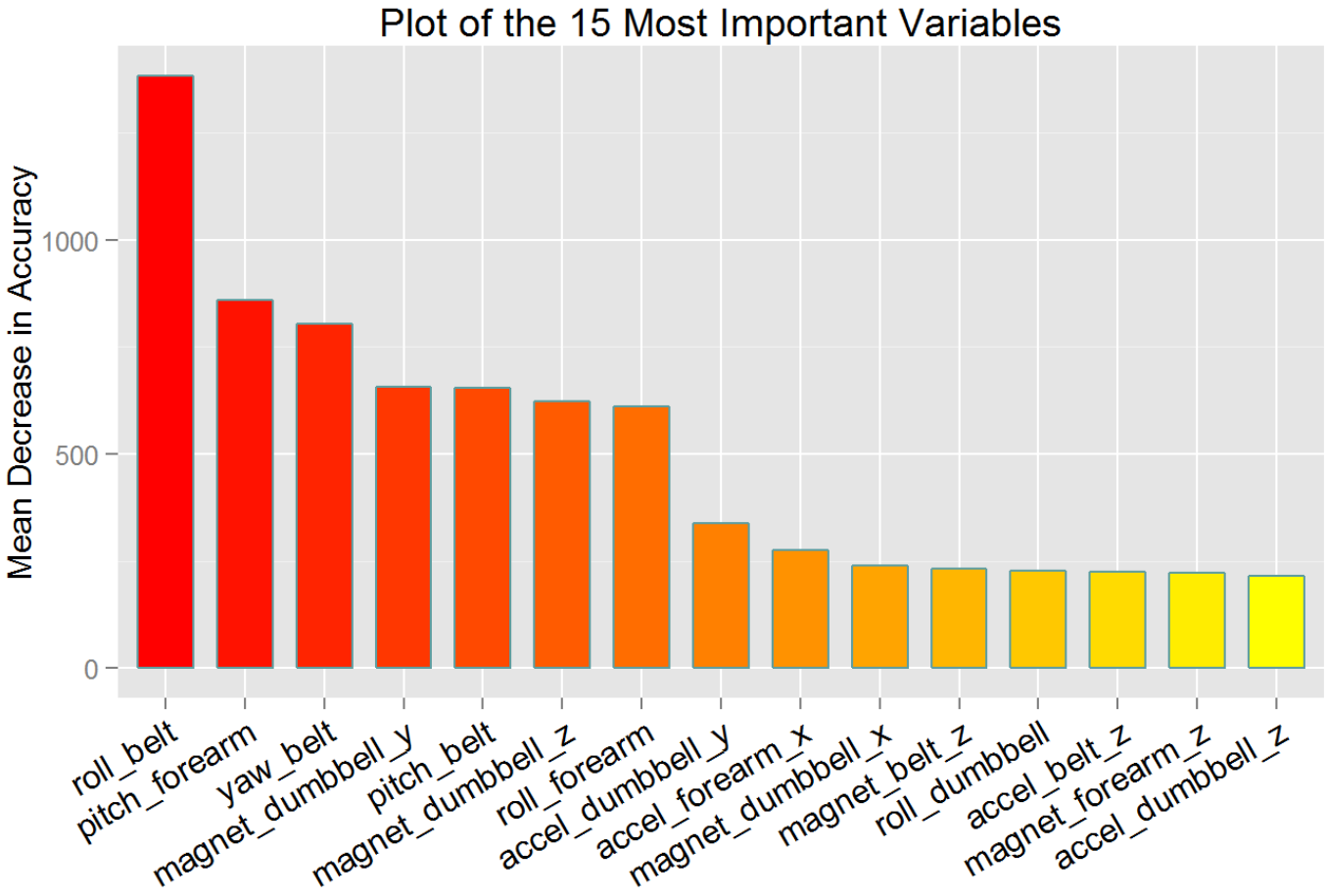
```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.76%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    2    3    0    1 0.001536098
## B   21 2625   11    1    0 0.012415350
## C    0   17 2370    9    0 0.010851419
## D    0    2   21 2227    2 0.011101243
## E    0    2    4    9 2510 0.005940594
```

The fit is excellent, with an very low out-of-bag estimate of error rate of 0.76%. Because of the characteristics of the random forest building process, this constitutes a good estimation of the real out-of-sample error, which will be assessed on the test set.

C. Analyzing the Variables Importance

Variables importance is extracted from the model and the 15 most important variables are represented in a bar graph.

```
importance_raw <- importance(model_rf$finalModel, type=2)
importance_df <- data.frame(cbind('names'=rownames(importance_raw),'mean_decrease_acc'=im
portance_raw[ ,1]),row.names=NULL)
importance_df$mean_decrease_acc <- as.numeric(as.character(importance_df$mean_decrease_ac
c))
importance_df <- importance_df[order(importance_df$mean_decrease_acc, decreasing=TRUE), ]
subset_importance <- importance_df[1:15, ]
ggplot(data=subset_importance, aes(x=reorder(names,mean_decrease_acc,FUN=function(x) -x),
y=mean_decrease_acc))+
    geom_bar(stat='identity', color='cadetblue', fill=heat.colors(20)[1:15], width=0.7)+
    xlab('')+ylab('Mean Decrease in Accuracy')+
    ggtitle('Plot of the 15 Most Important Variables')+
    theme(axis.text.x=element_text(angle=30, hjust=1, color='black', size=12), legend.pos
ition='none')
```



The variables roll_belt, pitch_forearm, and yaw_belt, are the 3 most important in this random forest model. It should be noted that in this dataset some of the variables seems to be relatively strongly correlated. While it does not impact the performance of the model, it may induce inaccuracies in the variable importance measures (see the correlation matrix plot for this dataset in appendix 1).

C. Predicting Classes on the Test Set

We use the model to predict classes on the test set.

```
prediction_test <- predict(model_rf, newdata=test_d)
confusionMatrix(prediction_test, test_classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    8    0    0    0
##          B    0 1130    0    0    0
##          C    1    1 1019   14    2
##          D    0    0    7  950    0
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9921   0.9932   0.9855   0.9982
## Specificity            0.9981   1.0000   0.9963   0.9986   1.0000
## Pos Pred Value         0.9952   1.0000   0.9826   0.9927   1.0000
## Neg Pred Value         0.9998   0.9981   0.9986   0.9972   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1920   0.1732   0.1614   0.1835
## Detection Prevalence   0.2856   0.1920   0.1762   0.1626   0.1835
## Balanced Accuracy      0.9988   0.9960   0.9947   0.9920   0.9991
```

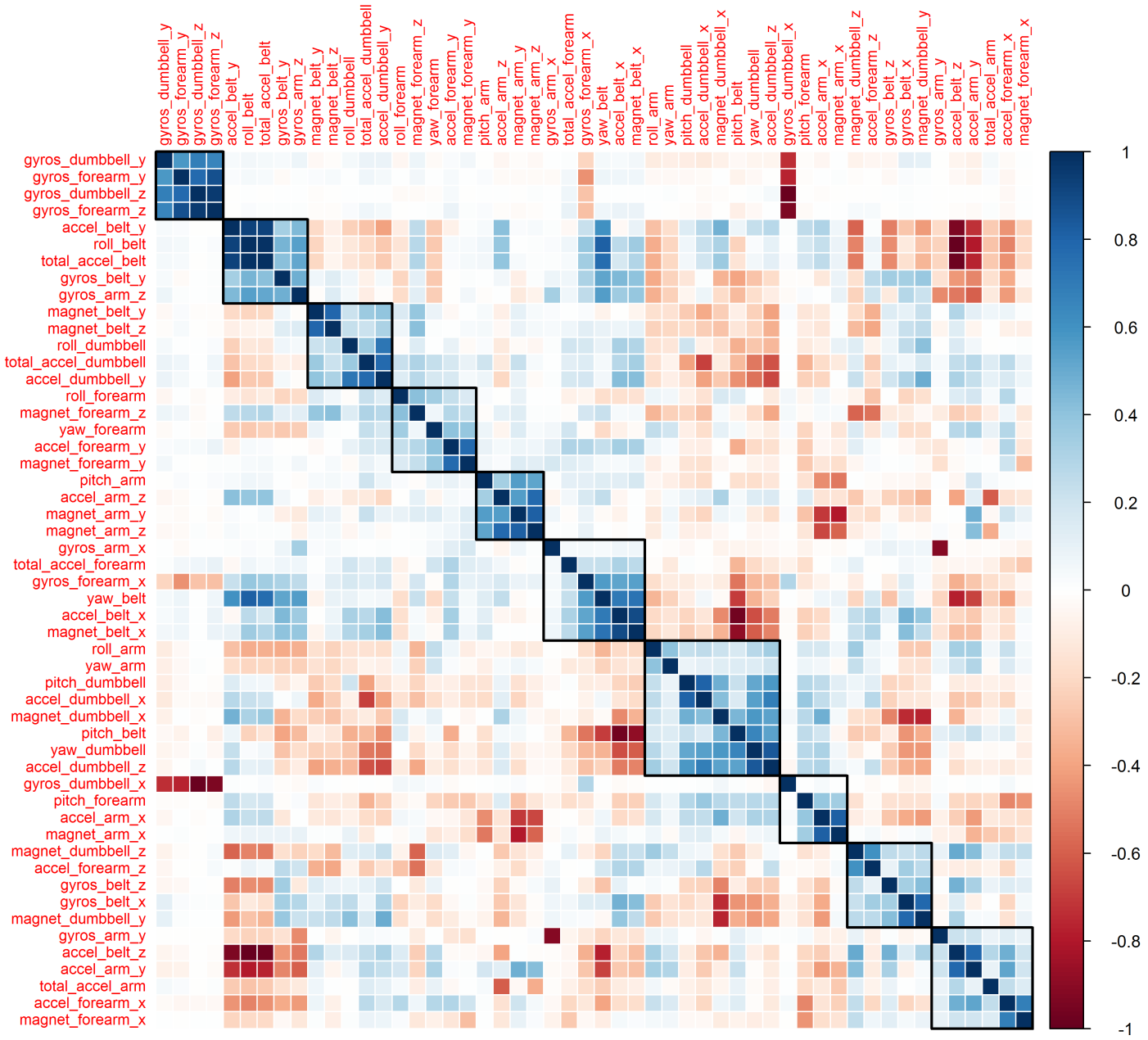
As expected, the prediction accuracy is excellent, at 0.9944. Despite its high accuracy on the training set, the random forest model shows its capacity not to overfit as it is just as accurate on an out-of-sample test set. As expected, the out-of-bag estimate of error rate calculated during the fitting of the model was a good estimate of the out-of-sample error rate. For this assignment, prediction was also applied to an external test set of 20 cases (see the prediction for this dataset in appendix 2)

D. Appendices

Appendix 1: Correlation Matrix

Some clusters of positive correlation between variables have been outlined around the diagonal of the plot.

```
corrplot(cor(train_d), method='color', tl.cex=0.7, order='hclust', addrec=10, insig='blan
k')
```



Appendix 2: Prediction on an External Test Set

The random forest model is applied to an external test set for which the cases classification is not known.

```
ext_test_d <- read.csv(url_ext_test, stringsAsFactors=FALSE)
ext_test_d <- ext_test_d[ ,8:ncol(ext_test_d)]
ext_test_d <- ext_test_d[ ,sapply(ext_test_d, is.numeric)]
ext_test_d <- ext_test_d[ ,sapply(ext_test_d, find_na_columns) == 0]
predict(model_rf, ext_test_d)
```

```
##  [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
```