TITLE Author

Practical Machine Learning Project: Predicting Active Quality Kumar Narasimhan

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**Summary**

The goal of this course project is to predict how well 6 different people performed barbell lifts utilizing data collected from activity monitoring devices. Each of the 6 people were asked to perform the barbell lifts correctly and in 5 different *incorrect* ways. Utilizing the activity monitor device data, a machine learning model is to be generated using a training set with class labels representing the 6 ways of performing the barbell lifts (supervised learning). Once the models are built, the generalization performance should be assessed, and then the training model is to be applied to a new set of testing data to make predictions. These predictions are later submitted for automated grading in a second component of the assignment.

**Input Data**

The input data consisted of various movement measurments including acceleration components of the arms and pitch and roll orientations of the dumbell. More information can be found at the original data authors website linked to below.

The data used here was downloaded from the course website, where the training and testing data were already partitioned:

[Training Data](https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

[Testing Data](https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The original data was taken from the originating study linked below. Please see the site and associated paper for more information. <http://groupware.les.inf.puc-rio.br/har>

**Data Analysis and Predictions**

Before we start the analysis, a few packages must be loaded. Most importantly, the *caret* package is used here, which provides a streamlined interface into a variety of machine learning methods, making the entire analysis process *much* easier.

Also note the use of the *doMC* library and *registerDoMC* function call. This is needed to utilize multiple cores which will be used during the modeling building cross validation procedure to speed-up the calculations

library(caret)

library(knitr)

library(doMC)

registerDoMC(cores = 4)

set.seed(140819)

Now, we need to read in the data and perform some dataset filtering. Given the complexity of the underlying activity features and limited documentation on what they actually are, I will use a very simple approach: remove all features with missing values. In this data, these could be NA's or simply empty strings

# Read in the training and testing data data

dat.train <- read.csv("pml-training.csv", stringsAsFactors=FALSE)

dat.test <- read.csv("pml-testing.csv", stringsAsFactors=FALSE)

# Function to filter the features

# Here, we just remove the features with any missing data

filterData <- function(idf) {

# Since we have lots of variables, remove any with NA's

# or have empty strings

idx.keep <- !sapply(idf, function(x) any(is.na(x)))

idf <- idf[, idx.keep]

idx.keep <- !sapply(idf, function(x) any(x==""))

idf <- idf[, idx.keep]

# Remove the columns that aren't the predictor variables

col.rm <- c("X", "user\_name", "raw\_timestamp\_part\_1", "raw\_timestamp\_part\_2",

"cvtd\_timestamp", "new\_window", "num\_window")

idx.rm <- which(colnames(idf) %in% col.rm)

idf <- idf[, -idx.rm]

return(idf)

}

# Perform the filtering on the datasets

# The training data has classe so we need to

# convert it to a factor for the model building

dat.train <- filterData(dat.train)

dat.train$classe <- factor(dat.train$classe)

dat.test <- filterData(dat.test)

This filtering process has reduced the total number of features to r ncol(dat.train)-1. All of these remaining features have complete data, eliminating the need for imputation.

Now that we have the training and testing sets prepared, it's time to build the prediction models on the training data. Here, I'll build 3 models via the following classifier algorithms: Random Forest, SVM (radial kernel), and KNN. Parameters will be tuned via 5-fold cross validation.

# Now create some prediction models on the training data

# Here, we'll use cross validation with trainControl to help optimize

# the model parameters

# Here, we'll do 5-fold cross validation

cvCtrl <- trainControl(method = "cv", number = 5, allowParallel = TRUE, verboseIter = TRUE)

# We'll make 3 models that use different approaches and use a voting mechanism for the class predictions

m1 <- train(classe ~ ., data = dat.train, method = "rf", trControl = cvCtrl)

m2 <- train(classe ~ ., data = dat.train, method = "svmRadial", trControl = cvCtrl)

m3 <- train(classe ~ ., data = dat.train, method = "knn", trControl = cvCtrl)

Now that we have the 3 models built, let's investigate the cross-validation performance accuracy.

# Make a data frame with the maximum accuracy values from the models obtained

# via the cross validation on the training data

acc.tab <- data.frame(Model=c("Random Forest", "SVM (radial)", "KNN"),

Accuracy=c(round(max(head(m1$results)$Accuracy), 3),

round(max(head(m2$results)$Accuracy), 3),

round(max(head(m3$results)$Accuracy), 3)))

kable(acc.tab)

From the table, the Random Forest model appears to have the highest cross-validation accuracy, with the SVM and KNN slightly lower.

Now, let's do predictions on the test set data. Here, we'll do predictions using the 3 models and look for concordance in the classifications.

# Do the predictions

test.pred.1 <- predict(m1, dat.test)

test.pred.2 <- predict(m2, dat.test)

test.pred.3 <- predict(m3, dat.test)

# Make a table and check if they all agree

pred.df <- data.frame(rf.pred = test.pred.1, svm.pred = test.pred.2, knn.pred = test.pred.3)

pred.df$agree <- with(pred.df, rf.pred == svm.pred && rf.pred == knn.pred)

all.agree <- all(pred.df$agree)

Here are the classifications predictions for the 3 models:

colnames(pred.df) <- c("Random Forest", "SVM", "KNN", "All Agree?")

kable(pred.df)

From the table, we can see that the results agree for all the models. Combined with the high accuracy results from the cross-validation procedure, it appears as though we have good prediction models.

The last step of the assignment is to write out the results to test results files to be uploaded for automated grading. The code below was reused from the course website as suggested for use during the prediction answer submission process.

# Looks like they all do; let's write out the prediction files to submit

# This uses the code supplied by the class instructions

answers <- pred.df$rf.pred

pml\_write\_files = function(x){

n = length(x)

for(i in 1:n){

filename = paste0("problem\_id\_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)

}

}

pml\_write\_files(answers)

**Assessment of Final Predictions**

The predictions above were submitted to for automated grading, and all were found to be correct