Coursera Practical Machine Learning Course Project

Armin Kakas

Sunday, May 24, 2015

Contents

Analytical Intent

For my final project, I performed a holistic **predictive modeling** *(classification)* exercise with the **Weight Lifting Exercises Dataset** (both training and test data available publicly, online). My work will include the following elements:
1. Data pre-processing, including variable reduction methods (missing data, correlations, variable importance)
2. Exploratory analysis via correlation matrix
3. Classification model tuning using cross-validation for resampling (at least KNN, CART and Random Forest)
4. Model ensembles for better predictive accuracy (i.e. stacking KNN and Random Forest)
5. Predictions on test data

Dataset Description

Below is a link to a more detailed description of the data and the underlying study:
- PUC: Human Activity Recognition

The original data comes from two researchers from the *Pontifical Catholic University of Rio de Janeiro*, and is available on their website or via UCI's *machine learning repository*. The underlying study that comprises the data is best described by their research page:

> "Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E)."

The participants of the project were fitted with **accelerometers** (think of a Jawbone or Fitbit like instrument) on the belt, forearm, arm and dumbell.

As part of my classification exercise, I will attempt to correctly predict the manner in which the participants did the bicep-curl exercise. In other words, I will want to correctly classify **classes A through E**.

Data Cleanup and Visualization

In this first phase of our classification exercise, I read in the data and immediately remove some variables that serve no purpose for our classification*(user name and time stamp elements)*. Subsequently, I remove any variables whose proportion of **missing values (NAs)** is at least 75%. Variables that have that many NAs are likely not very useful explanatory variables. Finally, I removed a few other numerical variables that were decoded as factors, but that otherwise had a **large majority blank** data elements.

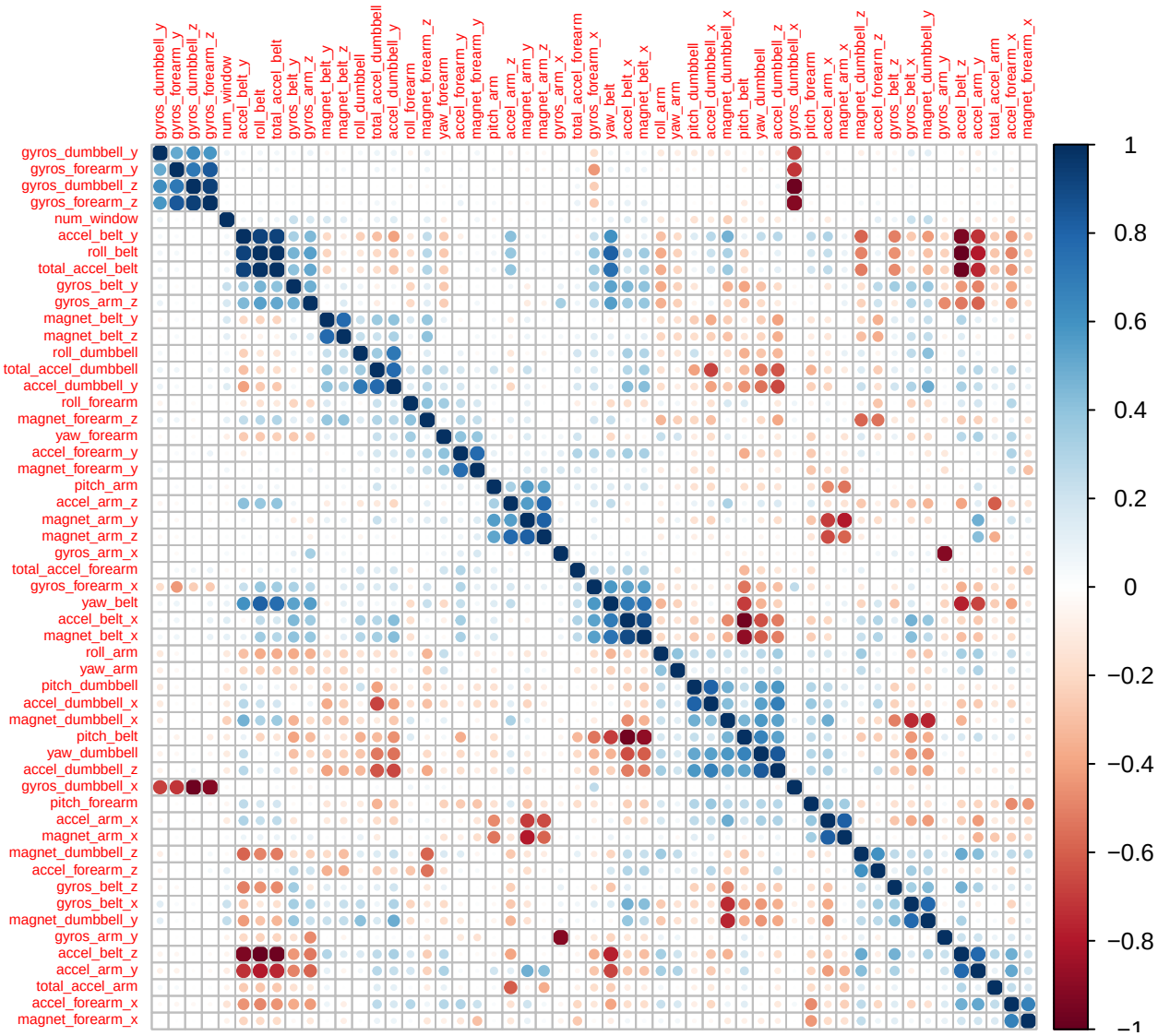
Thus, before doing any more robust variable reduction methods, we *reduced our predictors from 159 to 53*.

```
data <- tbl_df(read.csv('pml-training.csv'))
data <- data %>% rename(class = classe) %>% select(-X,-user_name, -raw_timestamp_part_1, -raw_timestamp_

#Let's eliminate variables that have too many missing values
dataNAs <- data.frame(apply(data, 2, function(x) length(which(is.na(x)))) / dim(data)[1])
dataNAs$variable = row.names(dataNAs)
dataNAs$portion.NA <- dataNAs[,1]
#We will filter out variables whose proportion of 'NA' records are at least 75%
dataNAs <- tbl_df(dataNAs) %>% select(variable, portion.NA) %>% arrange(desc(portion.NA)) %>%
                          filter(portion.NA <= 0.75)
vars.to.keep1 <- dataNAs$variable
data <- data[, vars.to.keep1]

#Upon visual inspection (using dplyr's 'glimpse' function), I decided to delete additional columns
data <- data %>% select(-starts_with('kurtosis')) %>%
                    select(-starts_with('skewness')) %>%
                        select(-starts_with('max_yaw')) %>%
                                select(-starts_with('min_yaw')) %>%
                                    select(-starts_with('amplitude_yaw')) %>% select(-new_window)
```

Correlation Matrix of 53 Remaining Explanatory Variables

As we can see, there are some highly correlated variables (both positively and negatively), which we will account for during our **variable reduction** phase. Once our variables are reduced based on both the **missing values** and a high degree of collinearity, then we will perform a random forest ensemble to help us exclude additional features that are not that informative.

```
corrdata <- data[, -54]
dataCorr <- cor(corrdata)
library(corrplot)
corrplot(dataCorr, order = "hclust", tl.cex = .50)
```



Variable Reduction: High Collinearity and Variable Importance

Using the **caret** package's *findCorrelation* function, we identify and eliminate variables with at least a 90% pairwise correlation. The logic in *findCorrelation* is smart enough to identify pairwise variables meeting the specified threshold of correlation, and removes the one with the largest mean absolute correlation (vs. all other variables). Below is a list of variables that were removed from the data set due to high collinearity.

```
highCorr <- findCorrelation(dataCorr, .90) #which columns to remove due to high collinearity
names(corrdata[, highCorr])
```

[1] "accel_belt_z" "roll_belt" "accel_belt_y"
[4] "accel_belt_x" "gyros_arm_y" "gyros_forearm_z" [7] "gyros_dumbbell_x"

```
data <- cbind(data[,54], corrdata[, -highCorr]) #revise my data set to exclude the variables
```

After eliminating the variables that exhibited a high degree of collinearity, we further reduced the remaining 46 variables leveraging *random forest variable importance* methods. For this, we performed a stratified 60 /

40 partition of our data set, and trained a **random forest** model on it. Our **random forest** model was fitted using **2000** trees, with explanatory variables both *scaled and centered*. Our resampling method during the tuning process was carried out using a **5-fold cross validation**.

To speed up the tuning process, we leveraged the **doMC** package to run things in parallel with 5 cores (out of a maximum of 8 on my machine). Due to timing concerns, the model tuning will not be evaluated as part of the output of this predictive analytics write-up.

To re-emphasize, the below random forest was used only to identify the variables with the largest predictive power. The ones whose *variable importance was less than 10 (relative to the variable with the largest predictive relevance)* were eliminated from the data set.

```
#only run this section in R if we need to change the threshold level on
#"Variable Importance"

#create a 60% "training data" on which I can tune a Random Forest
#we will use the RF model to figure out additional variables we can remove

set.seed(1234)
trainIndex <- createDataPartition(y = data$class, p = .6,
                                  list = FALSE,
                                  times = 1)
data.train <- data[trainIndex,]

ctrl1 <- trainControl(method = "cv",
                      number = 5,
                      classProbs = TRUE,
                      summaryFunction = defaultSummary)

#running the Random Forest in parallel
registerDoMC(cores = 5)

RF.littleFit <- train(class ~., data = data.train,
                 method = 'rf',
                 metric = "Accuracy",
                 preProc = c("center", "scale"),
                 ntree = 2000,
                 strata = data.train$class,
                 tuneLength = 5,
                 trControl = ctrl1)

variable.importance <- varImp(RF.littleFit)

lowest.importance <- variable.importance$importance
lowest.importance$variable <- row.names(lowest.importance)
lowest.importance <- tbl_df(lowest.importance) %>% rename(importance = Overall) %>%
                         arrange(desc(importance)) %>% filter(importance > 10)

#writing out the list of variables to keep as I do not want to run this RF
#tuning again

write.csv(lowest.importance, file = 'varstokeep.csv', row.names = FALSE)
```

The variables that were kept in the analytical data set can be found below. All other variables were deleted from the data set, and will not be used for predictive purposes.

```
vars.to.keep2 <- fread('varstokeep.csv')
vars.to.keep2 <- vars.to.keep2$variable
vars.to.keep2
```

[1] "num_window" "pitch_forearm" "magnet_dumbbell_z" [4] "yaw_belt" "pitch_belt" "magnet_dumbbell_y" [7] "magnet_belt_y" "roll_forearm" "magnet_dumbbell_x" [10] "accel_dumbbell_y" "roll_dumbbell" "gyros_belt_z"

```
data <- cbind(class = data[,1],data[, vars.to.keep2])
```

After the initial **159 predictors**, we are now left with 12 predictors: **that is quite a significant reduction!**.

Random Forest Tuning for Final Model

Now, let's tune our model one last time - this time using a **70 / 30** training and test data split, a 5-fold cross validation, along with parallelization for speed.

```
set.seed(1234)
trainIndex <- createDataPartition(y = data$class, p = .7,
                                  list = FALSE,
                                  times = 1)
data.train <- data[trainIndex,]
data.test <- data[-trainIndex,]


ctrl2 <- trainControl(method = "cv",
                      number = 5,
                      classProbs = TRUE,
                      summaryFunction = defaultSummary)

#running the Random Forest in parallel
registerDoMC(cores = 6)

RF.bestFit <- train(class ~., data = data.train,
                method = 'rf',
                metric = "Accuracy",
                preProc = c("center", "scale"),
                strata = data.train$class,
                tuneLength = 5,
                trControl = ctrl2)

RFPredict <- predict(RF.bestFit,newdata = data.test)

y <- confusionMatrix(RFPredict, data.test$class)
key.metrics <- data.frame(y$byClass)[,c(1,2,8)]
pander::pander(y$table)
```

	A	B	C	D	E
A	1674	1	0	0	0
B	0	1138	4	2	0
C	0	0	1022	1	0

	A	B	C	D	E
D	0	0	0	961	0
E	0	0	0	0	1082

```
pander::pander(key.metrics)
```

	Sensitivity	Specificity	Balanced.Accuracy
Class: A	1	0.9998	0.9999
Class: B	0.9991	0.9987	0.9989
Class: C	0.9961	0.9998	0.9979
Class: D	0.9969	1	0.9984
Class: E	1	1	1

After tuning one more time with a random forest model, our **mtry** *(the number of variables randomly sampled at each tree split)* was selected to be 4.
As we can see, the model comes up quite strong, with an average specificity of **0.9996585** and a sensitivity of **0.9984223**. In addition to the classification accuracy statistics, the above table displays the **confusion matrix** for our predicted (row-wise) and actual (column-wise) classes.

Final Predictions

```
#Let's see which ones perform the best
traindata <- data
testdata <- tbl_df(fread('pml-testing.csv'))
#eliminating variables from the test data that we know we are not going to need
#for our predictive model
testdata <- testdata[, vars.to.keep2]

#running the Random Forest in parallel
registerDoMC(cores = 6)

RF.bestModel <- train(class ~., data = traindata,
                      method = 'rf',
                      metric = "Accuracy",
                      preProc = c("center", "scale"),
                      strata = traindata$class,
                      tuneLength = 5,
                      trControl = ctrl2)

RFPredict.final <- predict(RF.bestModel, newdata = testdata)

answers = as.character(RFPredict.final)
```

Exporting Final Predictions for Answer Submissions

```
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)
```