Prediction Assignment

Devin Dinwiddie

June 04, 2022

Contents

0.1 Introduction

The rise of fitness tracking software has exploded in recent years. It is now possible to keep track of many kinds of data related to fitness activity in a wearable device or even a exercise machine. This has led to the generation of a vast amounts of fitness data that can be used to gain insight into the health and well being of the user. Often these devices quantify how much of an activity a person did; steps, jumps, lifts etc. but rarely do they determine how well an exercise was preformed. Here we will use prediction based modeling to predict if an exercise was preformed correctly or incorrectly. The data comes from 6 participants who were asked to preform barbell lifts in one of two ways; incorrectly and correctly. Various data was collected during their performance and using the collected metrics we will build a prediction model that can be used to determine if the exerciser preformed the exercise correctly or not. We can in-vision a similar model being used to give real time output of exercise performance.

0.2 Methods

0.2.1 Data Cleaning and Model Selection

Training and Test data files were obtained from Here and Here respectively. Columns with missing values were removed from both training and test sets. Columns un-related to exercise performance were also removed. These included user names, time stamp data and window data. After initial cleaning a total of 53 variables relating to exercise performance remained including the identifier column classe. Training data was further split to include a validation data set. Initially three models were run, a random forest model(RF) with ten trees, a gradient boosting machine model(GBM), and linear discriminate analyses model(LDA). All models were run with five fold cross validation. Based on model accuracy the RF was deemed to preform the best followed closely by GBM and then LDA. From the results of initial model testing RF was picked as the best model for prediction, and further tuning and training was preformed to optimize prediction.

0.2.2 Random Forest Model Tuning and Validation

To contend with model over fitting, cross validated prediction was applied with sequentially reduced predictors ranked by variable importance and cross validated errors were used to assess feature selection of the model. Training of RF model was preformed using 10,100,1000 and 2000 trees holding the hyper-parameter mtry constant at 22. While we contend tuning the number of trees is not necessary due to the asymptomatic behavior of the law of large numbers and identical distribution of trees in an RF model, this step was preformed strictly to determine the minimum number of trees needed to gain a boost in prediction without sacrificing computational performance. Further tuning of the number of candidate variables sampled at each split was preformed using a range from 2 to 42 (total number of variables in the optimized training data). The final model was tested on the validation data to determine the expected out of sample error, and test data for accuracy of predictions. All models used five fold cross validation. Code can be found in the Appendix.

0.3 Results

0.3.1 Model Selection

Of the three models initially tested the LDA model preformed the least well at predicting if the barbell exercise was preformed correctly, with an average accuracy of 70.194%. The GBM was next with an average accuracy of 96.085% and the RF model preformed the best with an average accuracy of 98.157%. The out of bag classification error of the RF model was 4.364%. The most important features for our RF model were roll_belt,pitch_forearm,and yaw_belt(fig1). Based on these results the RF model was selected for further tuning.

Fig 1: Feature Importance

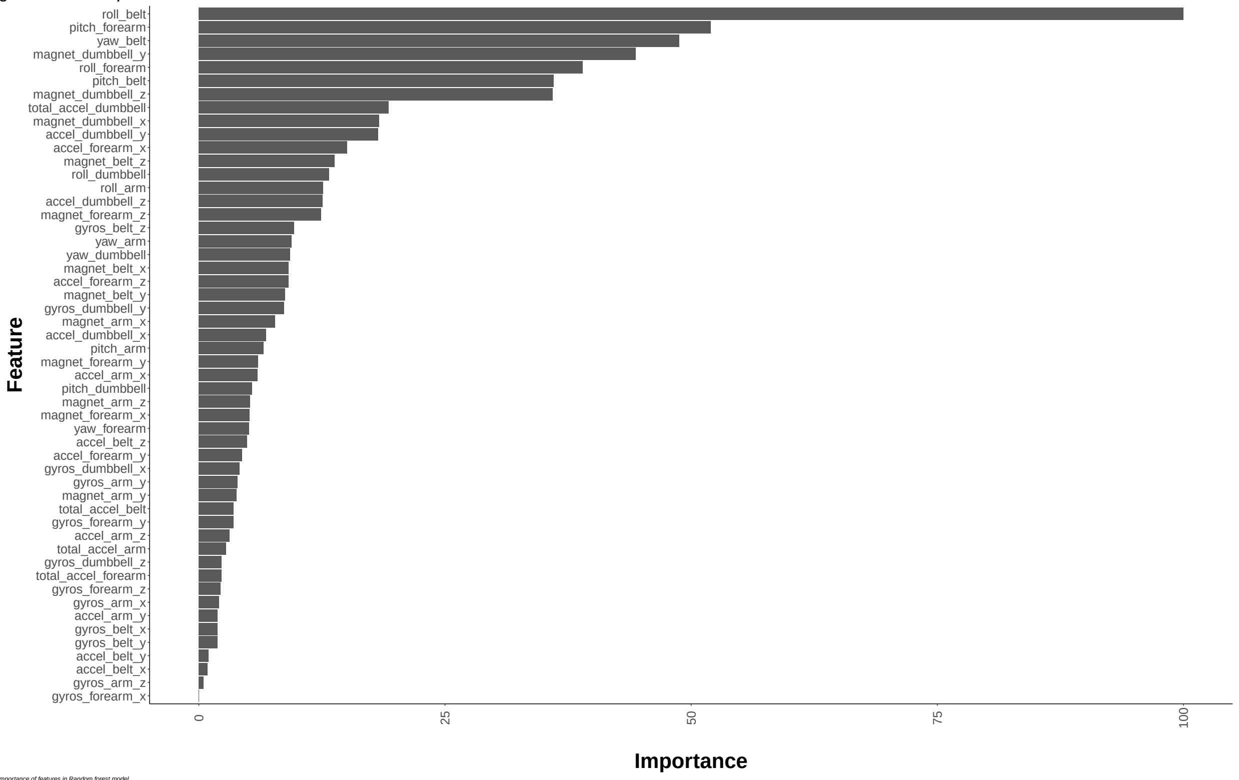
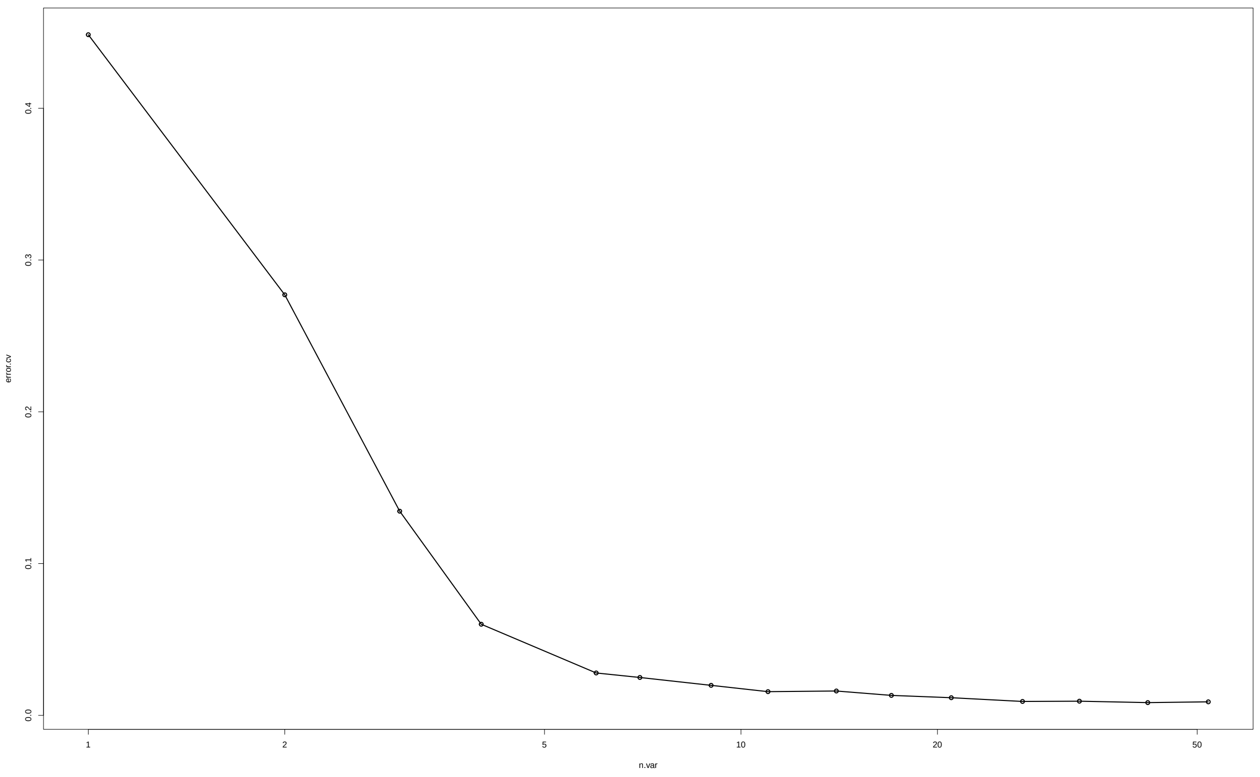


Fig1 importance of features in Random forest model

0.3.2 Model Tuning

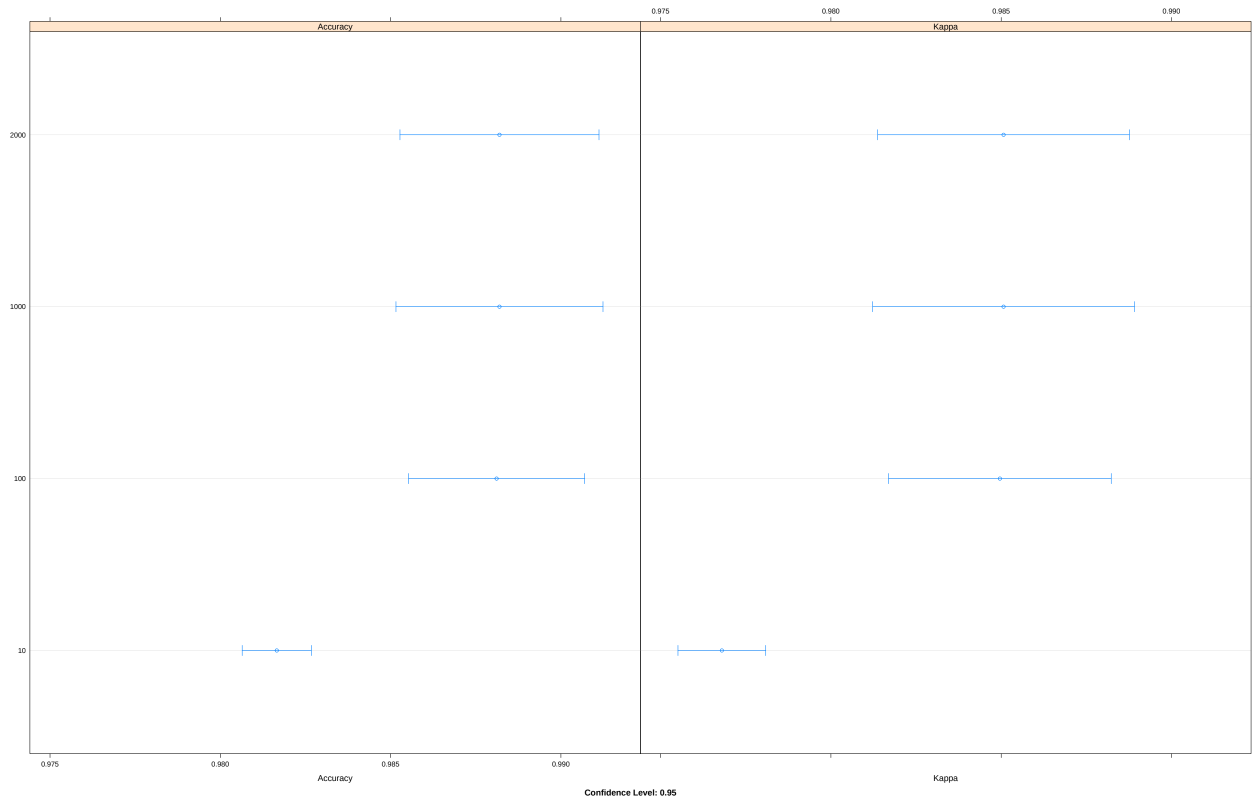
Using five fold cross validation to assess feature selection and over fitting of the model it was determined that the prediction error decrease moving from the full data set of 52 variables to 42 variables was minimal 0.0089 and 0.0084 respectively(fig2). Based on this we decided to only keep the top 42 most important variables in the training, validation and test data sets to avoid over fitting and save computational resources. In doing this we see only a slight dip in accuracy(98.021%) when our model is re-trained on the subset training data set, indicating that the variables removed where not influencing the prediction outcome as we surmised. Tuning for the minimum number of trees we could use to still give a boost in prediction it was determined that 100 trees did a good job increasing accuracy to 98.8%. Any increase in the number of trees beyond 100 did not increase accuracy by much (Fig3). Further tuning of the number of candidate variables sampled at each split(mtry) found the optimum to be 9 and increased the accuracy of the training data set to 99.04%. When applying this final model to our validation data set we achieve a prediction accuracy of 99.006%, Therefore we expect our expected out of sample error to be around 0.0099414%. Applying our model to the test data-set preformed prediction with 100% accuracy.

Fig2: cross validated error prediction with reduced variables



```
## NULL
```

Fig3: Accuracy vs nTree increase



0.4 discussion

Here we have demonstrated high accuracy prediction of exercise performance using data collected from fitness tracking software. Using a Random Forest prediction model that was tuned for feature selection and number of sampled variables we predicted with greater than 99% accuracy whether a barbell exercise was preformed correctly or incorrectly, our final model was tuned to be both highly accurate and computationally efficient. While we recognize our model tuning only slightly increased the accuracy compared to the original model and may not have been strictly necessary, one can imagine a situation were these tuning steps could help model performance immensely.

0.5 Appendix

0.5.1 Code

```
trainUrl ="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainFile = "./pml-training.csv" #may need to set path to file on your system
testFile  = "./pml-testing.csv"

if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile, method="curl") # get data if have not already
}
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile, method="curl")
}

trainDat <- read.csv("./pml-training.csv") # load data
testDat <- read.csv("./pml-testing.csv")
```

```r
non = complete.cases(t(trainDat)) & complete.cases(t(testDat)) # collect columns with missing values
trainDat = trainDat[,non] # remove missing values
testDat = testDat[,non]
trainDat = trainDat[,-c(1:7)] # remove unnecessary data
testDat = testDat[,-c(1:7)]

train = createDataPartition(trainDat$classe,p = 0.6,list = F) # create validation and trainging dat set
training = trainDat[train,]
validation = trainDat[-train,]

saveRDS(training,file = "./MachineLearningPredictionAssignment/RData/trainDat.RDS")
saveRDS(validation,file = "./MachineLearningPredictionAssignment/RData/validationDat.RDS")
saveRDS(testDat,file = "./MachineLearningPredictionAssignment/RData/testDat.RDS")
tc = trainControl(method = "cv",5) #five fold cross validation

set.seed(1234)
# initial models to test for accuracy
mod_rf <- train(classe ~ ., data = training, method = "rf",trControl = tc ,ntree = 10)
mod_gbm <- train(classe ~ ., data = training, method = "gbm",verbose = F,trControl = tc)
mod_lda <- train(classe ~ ., data = training, method = "lda",trControl = tc)

# save models for ease of kniting and computation
saveRDS(mod_rf,file = "./MachineLearningPredictionAssignment/RData/intialRFMod.RDS")
saveRDS(mod_gbm,file = "./MachineLearningPredictionAssignment/RData/intialGBMMod.RDS")
saveRDS(mod_lda,file = "./MachineLearningPredictionAssignment/RData/intialLDAMod.RDS")

mean(mod_rf$resample[[1]]) # accurarcy 0.9815725
mean(mod_rf$resample[[2]]) # kappa
mod_rf$finalModel$confusion # confusion matirx of training
mean(mod_rf$finalModel$confusion[1:5,6]) # classification error ~oob

mean(mod_gbm$resample[[1]]) # accurarcy 0.9608518
mean(mod_lda$resample[[1]]) # 0.7019375
```

```r
set.seed(1234)
t = training[,-(53:ncol(training))] # get variables - classe
t2 = as.factor(training$classe) # factor classe
result = rfcv(t,t2,cv.fold = 5,step = 0.8) # cross validate feature selection
with(result, plot(n.var, error.cv, log="x", type="o", lwd=2,main = "Fig2: cross validated error predict
result$error.cv # look at errors
#          52          42          33          27          21          17          14          11
# 0.008916440 0.008406929 0.009341033 0.009171196 0.011633832 0.013162364 0.016049592 0.015625000 0.019
#           7           6           4           3           2           1
# 0.024966033 0.027938179 0.060037364 0.134510870 0.277088995 0.448454484

saveRDS(result,file = "./MachineLearningPredictionAssignment/RData/rfcv.RDS") # save rfcv results

NameList = rownames(varImp(mod_rf)$importance)[1:42] # get 42 most important variables in model predict
NameList = c(NameList,"classe") # add classe variable
idx = match(NameList,names(training)) # get index of column names in list

# create new test,validation and training data with selected variables
trainingNew = training[,idx]
```

```
validationNew = validation[,idx]
testNew = testDat[,idx]

set.seed(1234)
new_rf <- train(classe ~ ., data = trainingNew, method = "rf",trControl = tc ,ntree = 10)# test model o
# slight accuracy dip  27    0.9815725

saveRDS(new_rf,file = "./MachineLearningPredictionAssignment/RData/new_rf.RDS")

# find minimum number of trees needed to get good accuracy results
tunegrid <- expand.grid(.mtry = 22) # hold mtry constant
modellist <- list() # storage

#train with different ntree parameters
for (ntree in c(10,100,1000,2000)){
  set.seed(123)
  fit <- train(classe~.,
               data = trainingNew,
               method = 'rf',
               metric = 'Accuracy',
               tuneGrid = tunegrid,
               trControl = tc,
               ntree = ntree)
  key <- toString(ntree)
  modellist[[key]] <- fit
}

#Compare results
results <- resamples(modellist)
summary(results)

# add title to lattice plot
my.settings <- list(par.main.text = list(font = 2, # make it bold
                       just = "left",
                       x = grid::unit(5, "mm")))

# dot plot of accuracy results from tree tuning
P = dotplot(results,main = "Fig2: Accuracy vs nTree increase",par.settings = my.settings)
P

saveRDS(results,file = "./MachineLearningPredictionAssignment/RData/TreeTraining.RDS")

tuneGrid = expand.grid(.mtry=c(2:42))
set.seed(1234)
mod_rf2 <- train(classe ~ ., data = trainingNew, method = "rf",tuneGrid = tuneGrid,trControl = tc,ntree

saveRDS(mod_rf2,file = "./MachineLearningPredictionAssignment/RData/modRF2.RDS")

mean(mod_rf2$resample[[1]]) # 0.9904045

#predict on the validation data
pred = predict(mod_rf2,validationNew)
```

```
saveRDS(pred,file = "./MachineLearningPredictionAssignment/RData/validationPred.RDS")

fa = as.factor(validationNew$classe)
postResample(pred,fa) #accuracy on validation 0.9900586

1 - postResample(pred,fa) # expected out of sample error rate

#   Accuracy       Kappa
# 0.009941371 0.012577427

# predict on test set
pred2 = predict(mod_rf2,testNew)

saveRDS(pred2,file = "./MachineLearningPredictionAssignment/RData/TestPred.RDS")
```