Practical ML Course Project

Ian Robinson

9/12/2020

Practical Machine Learning - Course Project

We'll be using Human Activity Recognition data from Groupware to train a model to recognize the type of exercise performed by a subject based on 160 variables collected by biometric sensors. The training set classifies exercises into 5 categories, represented in the "classe" variable.

Load Data and set seed

```
set.seed(500)
training <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv')
quiz <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv')
```

Split training data into training and test sets

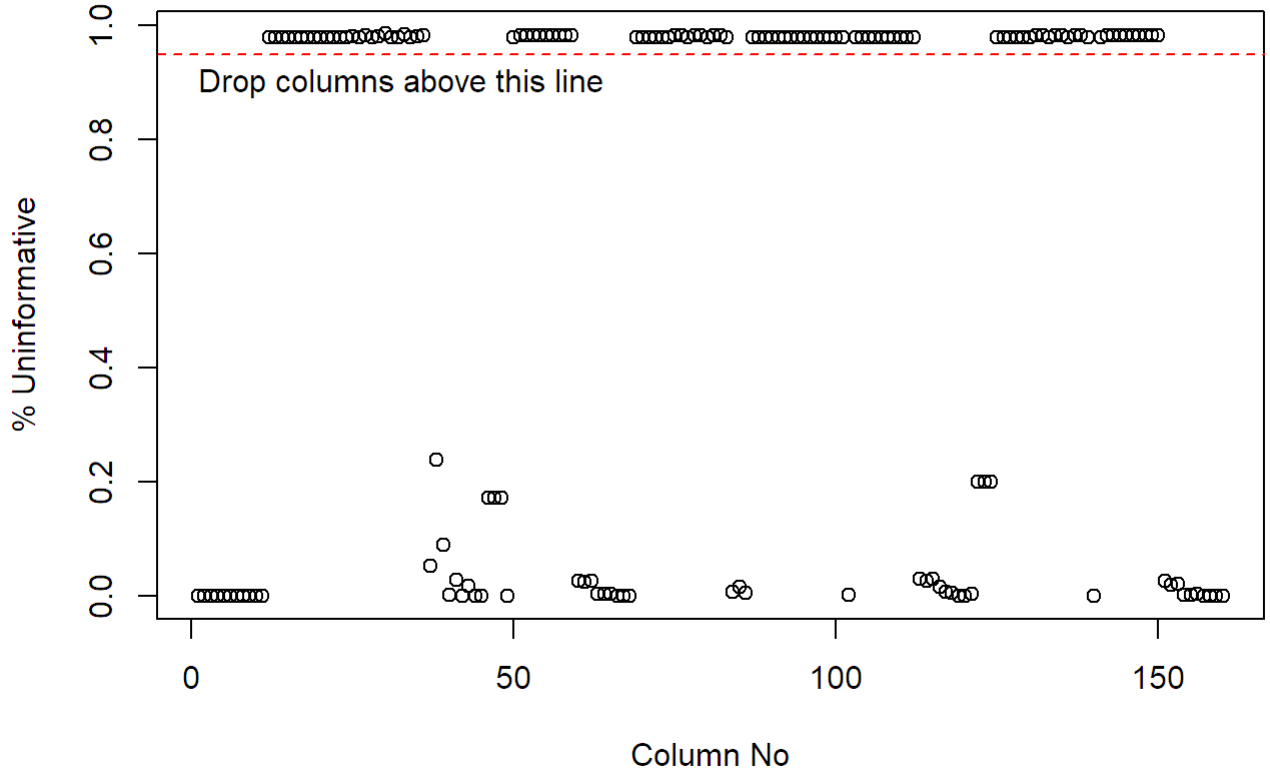
For validation purposes, we'll split our training data into two sets. We'll use 85% of the data for training (performing some testing and estimating out-of-sample accuracy using k-fold validation), and we'll reserve 15% of our data purely for validation at the end.

```
library(caret)
inTrain <- createDataPartition(y=training$classe,p=.85,list=FALSE)
training2 = training[inTrain,]
testing2 = training[-inTrain,]
```

Drop Columns

We're going to drop any columns that contain less than 5% informative values (defined as anything other than, NA, 0, or a blank string). We'll also drop the first five columns, which contain unique row identifiers and timestamp information. We'll also set the classe variable and the "new_window" variable as factors. We'll do this for the training, testing, and quiz sets as applicable.

```
#plot % of non-informative values in each column:
plot(colMeans(is.na(training2)|training2==0|training2==""),ylab='% Uninformative',xlab='Column N
o')
abline(h=.95,col='red',lty=2)
text(1,.9,labels='Drop columns above this line',adj=0)
```




```
# drop anything that's more than 95% NAs, 0s, and ""'s
filter <- colMeans(is.na(training2)|training2==0|training2=="")<.95
training2 <- training2[,filter]
testing2 <- testing2[,filter]
quiz2 <- quiz[,filter]

#drop initial 5 columns (individual row identifiers)
training2 <- training2[,-c(1:5)]
testing2 <- testing2[,-c(1:5)]
quiz2 <- quiz2[,-c(1:5)]

#set string columns as factors
training2$classe <- as.factor(training2$classe)
testing2$classe <- as.factor(testing2$classe)
training2$new_window <- as.factor(training2$new_window)
testing2$new_window <- as.factor(testing2$new_window)
quiz2$new_window <- as.factor(quiz2$new_window)
```

Create preprocess object

We'll create a preprocess object to center, scale, and apply principal component analysis to the columns we've selected. The reason we do this is to reduce the number of predictors for our random forest algorithm to handle without having to get too far into the weeds about the individual features of our data set. We'll then create new variables containing the transformed features for the training, testing, and quiz sets, which we'll use to train and predict our model.

```
preproc <- preProcess(training2,method=c('center','scale','pca'),thresh=.9)
trainingfeatures <- predict(preproc,training2)
testingfeatures <- predict(preproc,testing2)
quizfeatures <- predict(preproc,quiz2)
```

Train the model

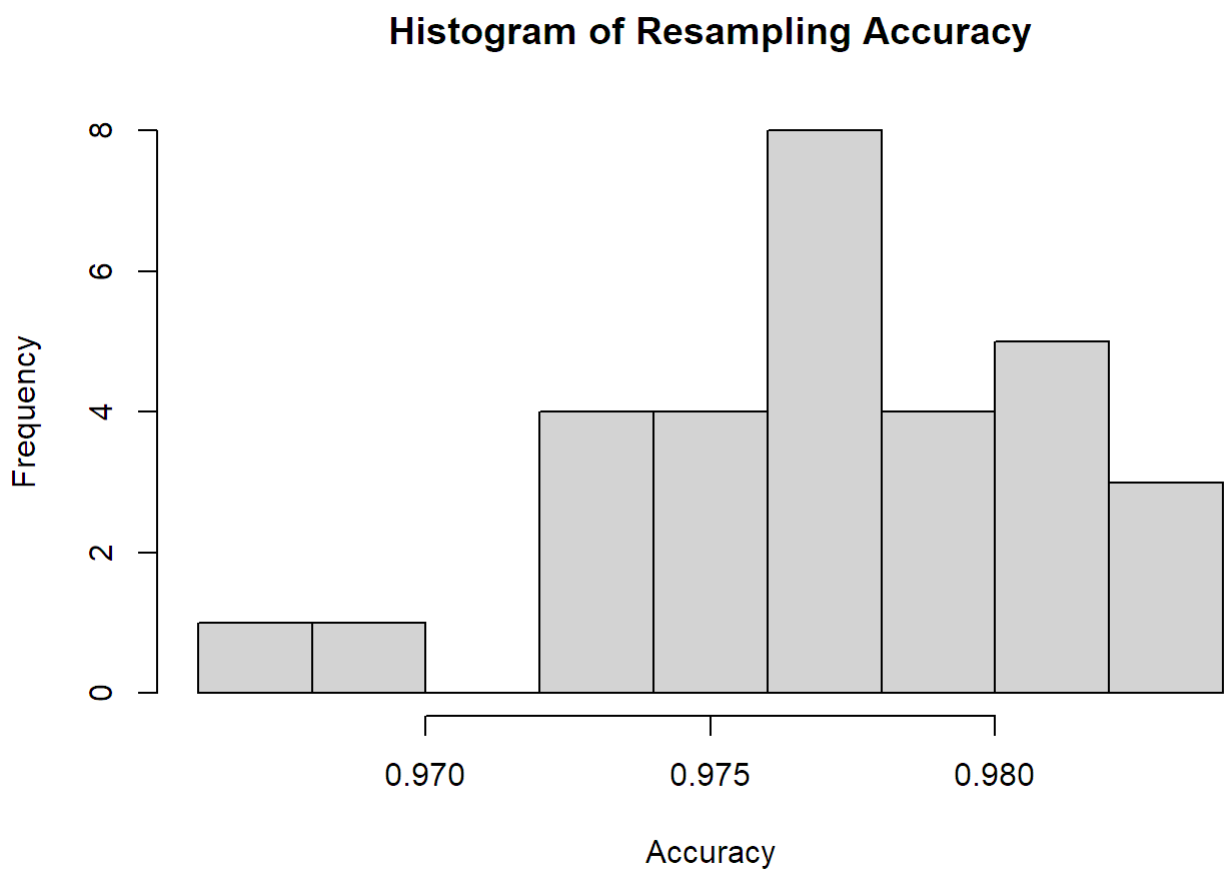
In this chunk we'll train our model. The model we've selected is a Random Forest model because that algorithm typically performs well on classification problems without requiring much tuning, although it is computationally intensive. To estimate out-of-sample error and reduce overfitting, we'll do 10-fold cross validation 3 times. To limit runtime, we'll limit the random forest to 100 trees.

```
#set 10-fold cross-validation, repeated 3 times
control <- trainControl(method='repeatedcv',number=10,repeats = 3)
#train model
mdl <- train(classe~.,data=trainingfeatures,method='rf',ntree=100,trControl=control)
print(mdl)
```

```
## Random Forest
##
## 16680 samples
##    20 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 15013, 15012, 15013, 15011, 15010, 15013, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9768587  0.9707266
##   11    0.9730220  0.9658742
##   20    0.9688649  0.9606146
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Check model in-sample accuracy and test set accuracy

```
ISacc <- mean(predict(mdl,trainingfeatures)==trainingfeatures$classe)
OSacc <- mean(mdl$resample$Accuracy)
Testacc <- mean(predict(mdl,testingfeatures)==testingfeatures$classe)
hist(mdl$resample$Accuracy,main='Histogram of Resampling Accuracy',xlab='Accuracy')
```



The final model accuracy on the training set is 1. The estimated out-of-sample accuracy based on mean accuracy of the 30 sets (3 repetitions of 10-fold cross-validation) on the training set is 0.9768587. As can be seen in the chart above, the resample accuracies are clustered tightly around 0.975, with a standard deviation of 0.0038195 The accuracy on the held-out validation set is 0.9792658, further confirming our estimated out-of-sample accuracy of ~97%.

Below is a confusion matrix of the model results on the reserved testing set:

```
library(caret)
confusionMatrix(predict(mdl,testingfeatures),testingfeatures$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 830  10   0   2   0
##          B   0 547   7   0   1
##          C   3  11 503  15   2
##          D   3   1   1 464   0
##          E   1   0   2   1 538
##
## Overall Statistics
##
##                Accuracy : 0.9796
##                  95% CI : (0.9738, 0.9844)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9742
##
##  Mcnemar's Test P-Value : 0.0007538
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9916   0.9613   0.9805   0.9627   0.9945
## Specificity            0.9943   0.9966   0.9872   0.9980   0.9983
## Pos Pred Value         0.9857   0.9856   0.9419   0.9893   0.9926
## Neg Pred Value         0.9967   0.9908   0.9958   0.9927   0.9987
## Prevalence             0.2845   0.1934   0.1744   0.1638   0.1839
## Detection Rate         0.2821   0.1859   0.1710   0.1577   0.1829
## Detection Prevalence   0.2862   0.1886   0.1815   0.1594   0.1842
## Balanced Accuracy      0.9930   0.9790   0.9839   0.9803   0.9964
```

Predict on the quiz set

```
quizpredictions = predict(mdl,quizfeatures)
```

The predictions of the model on the 20 quiz questions are as follows: B, A, B, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B