Practical Machine Learning Course Project

Luca Caramellino

Synopsis

Use data from accelerometers wore by volunteer participants executing barbell to identify the quality of the exercise. Some volunteer executed the exercise correctly while other purposedely committed common mistakes. The predictive software must be able to identify the correct exercises with reasonable accuracy.

Load libraries

```
library(AppliedPredictiveModeling)
library(caret)
library(rattle)
library(rpart.plot)
library(randomForest)
library(knitr)
library(e1071)
```

Load Data

```
training <- read.csv("pml-training.csv", na.strings=c("NA",""), header=TRUE)
colnames_train <- colnames(training)
testing <- read.csv("pml-testing.csv", na.strings=c("NA",""), header=TRUE)
colnames_test <- colnames(testing)
```

Filter Data

Data sets are filtered to remove NA values and near zero variables.

```
nonNAs <- function(x) {
    as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}

colcnts <- nonNAs(training)
drops <- c()
for (cnt in 1:length(colcnts)) {
    if (colcnts[cnt] < nrow(training)) {
        drops <- c(drops, colnames_train[cnt])
    }
}

training <- training[,!(names(training) %in% drops)]
training <- training[,8:length(colnames(training))]

testing <- testing[,!(names(testing) %in% drops)]
testing <- testing[,8:length(colnames(testing))]
```

The training set has **19622** samples and **52** potential predictors after filtering. The testing set result instead with **20** samples and **52** predictors.

Check for Covariates

```
nsv <- nearZeroVar(training, saveMetrics=TRUE)
nsv
```

```
##                      freqRatio percentUnique zeroVar   nzv
## roll_belt             1.101904     6.7781062   FALSE FALSE
## pitch_belt            1.036082     9.3772296   FALSE FALSE
## yaw_belt              1.058480     9.9734991   FALSE FALSE
## total_accel_belt      1.063160     0.1477933   FALSE FALSE
## gyros_belt_x          1.058651     0.7134849   FALSE FALSE
## gyros_belt_y          1.144000     0.3516461   FALSE FALSE
## gyros_belt_z          1.066214     0.8612782   FALSE FALSE
## accel_belt_x          1.055412     0.8357966   FALSE FALSE
## accel_belt_y          1.113725     0.7287738   FALSE FALSE
## accel_belt_z          1.078767     1.5237998   FALSE FALSE
## magnet_belt_x         1.090141     1.6664968   FALSE FALSE
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## magnet_belt_z         1.006369     2.3290184   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_x           1.015504     3.2769341   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## accel_arm_x           1.017341     3.9598410   FALSE FALSE
## accel_arm_y           1.140187     2.7367241   FALSE FALSE
## accel_arm_z           1.128000     4.0362858   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
## magnet_arm_y          1.056818     4.4439914   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_x      1.003268     1.2282132   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## gyros_dumbbell_z      1.060100     1.0498420   FALSE FALSE
## accel_dumbbell_x      1.018018     2.1659362   FALSE FALSE
## accel_dumbbell_y      1.053061     2.3748853   FALSE FALSE
## accel_dumbbell_z      1.133333     2.0894914   FALSE FALSE
## magnet_dumbbell_x     1.098266     5.7486495   FALSE FALSE
## magnet_dumbbell_y     1.197740     4.3012945   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
```

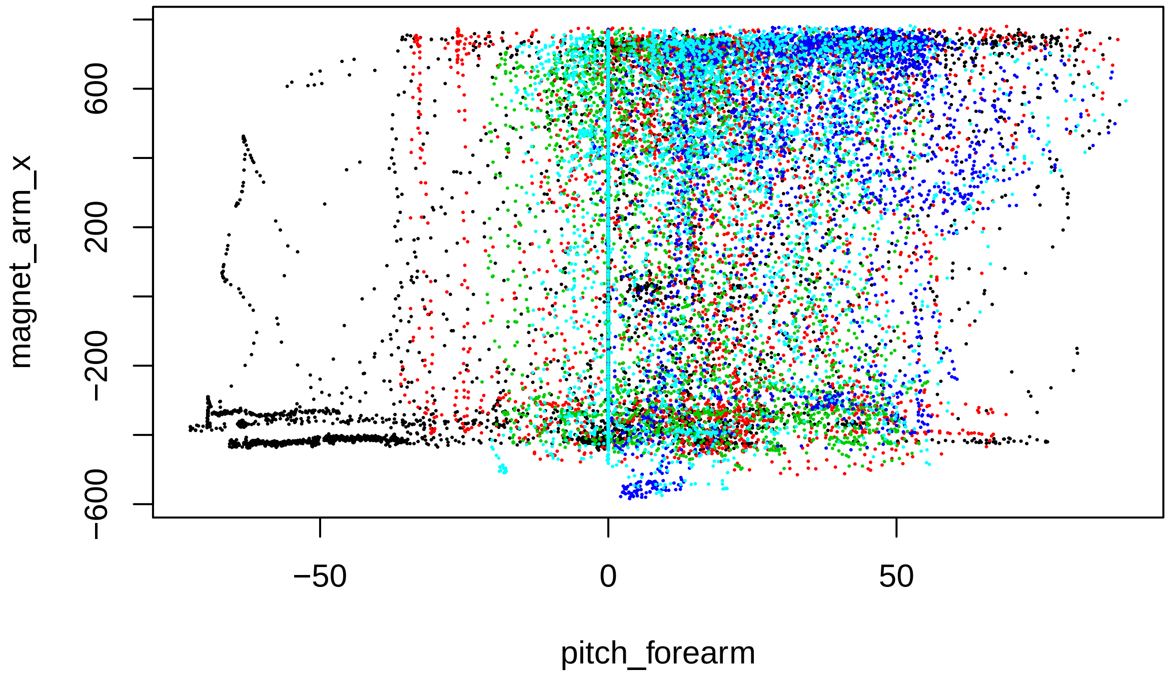
```
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_y       1.036554     3.7763735   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_y       1.059406     5.1116094   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

No covariates was identified so there is no need to further filtering the data set.

Plot features with highest correlation with classe

```
cor <- abs(sapply(colnames(training[, -ncol(training)]), function(x) cor(as.numeric(training[, x]), as.n

plot(training[, names(which.max(cor))], training[, names(which.max(cor[-which.max(cor)]))], col = train
```



There isn't any strong predictors that correlates with `classe` therteofre linear regression is not suitable. Random forests algorithm may generate more robust predictions for our data and is therefore selected.

Algorithm (Random Forest)

Creating smaller dataset from the original one

The training dataset is divided into smaller sets both to avoid overfitting and to allow predictive algorithm to run faster.

```
set.seed(3)
ids_small <- createDataPartition(y=training$classe, p=0.25, list=FALSE)
small1 <- training[ids_small,]
remainder <- training[-ids_small,]

set.seed(333)
ids_small <- createDataPartition(y=remainder$classe, p=0.33, list=FALSE)
small2 <- remainder[ids_small,]
remainder <- remainder[-ids_small,]

set.seed(333)
ids_small <- createDataPartition(y=remainder$classe, p=0.5, list=FALSE)
small3 <- remainder[ids_small,]
small4 <- remainder[-ids_small,]

set.seed(333)
inTrain <- createDataPartition(y=small1$classe, p=0.6, list=FALSE)
small_training1 <- small1[inTrain,]
small_testing1 <- small1[-inTrain,]

set.seed(333)
inTrain <- createDataPartition(y=small2$classe, p=0.6, list=FALSE)
small_training2 <- small2[inTrain,]
small_testing2 <- small2[-inTrain,]

set.seed(333)
inTrain <- createDataPartition(y=small3$classe, p=0.6, list=FALSE)
small_training3 <- small3[inTrain,]
small_testing3 <- small3[-inTrain,]

set.seed(333)
inTrain <- createDataPartition(y=small4$classe, p=0.6, list=FALSE)
small_training4 <- small4[inTrain,]
small_testing4 <- small4[-inTrain,]
```

Random Forest (Test 1)

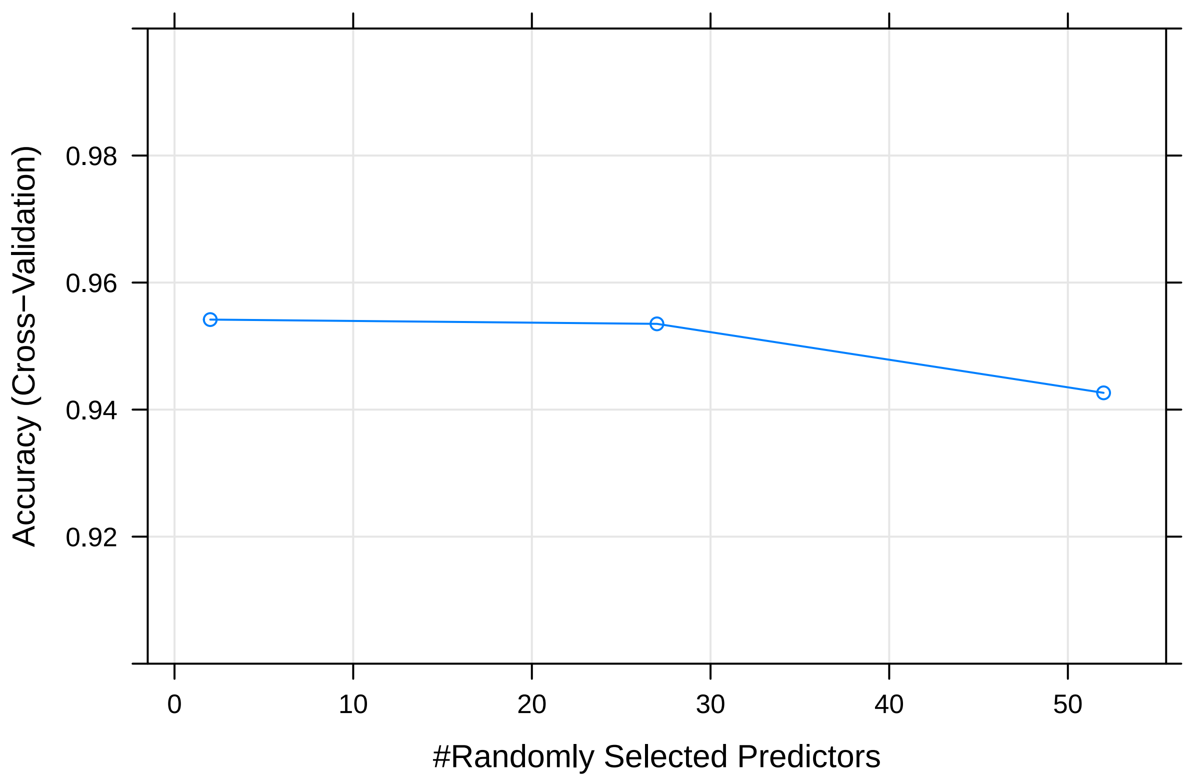
Selected random forest and run it on the first train data set using cross validation:

```
set.seed(2)
##Train
modFit <- train(small_training1$classe ~ ., method="rf", trControl=trainControl(method = "cv", number =
print(modFit, digits=3)
```

```
## Random Forest
```

```
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2209, 2211, 2209, 2209
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.954     0.942  0.00527      0.00665
##   27    0.954     0.941  0.00554      0.00701
##   52    0.943     0.927  0.00782      0.00989
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(modFit, ylim = c(0.9, 1))
```




```
##Test Set
predictions <- predict(modFit, newdata=small_testing1)
cf <- confusionMatrix(predictions, small_testing1$classe)
print(cf, digits=4)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 555  15   1   1   5
##          B   2 349  11   0   1
##          C   0  14 329  29   7
##          D   1   2   1 290   1
##          E   0   0   0   1 346
##
## Overall Statistics
##
##                Accuracy : 0.9531
##                  95% CI : (0.9428, 0.962)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9406
##  Mcnemar's Test P-Value : 9.481e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9184   0.9620   0.9034   0.9611
## Specificity            0.9843   0.9911   0.9691   0.9970   0.9994
## Pos Pred Value         0.9619   0.9614   0.8681   0.9831   0.9971
## Neg Pred Value         0.9978   0.9806   0.9918   0.9814   0.9913
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2830   0.1780   0.1678   0.1479   0.1764
## Detection Prevalence   0.2942   0.1851   0.1933   0.1504   0.1770
## Balanced Accuracy      0.9895   0.9548   0.9656   0.9502   0.9802
```

```
##Course Provided Test Set
print(predict(modFit, newdata=testing))
```

```
##  [1] B A C A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

```
oos1 <- 1 - cf$overall[1]
```

Where overall accuracy results **0.9530852** and out of sample error is **0.0469148**

Random Forest (Test 2)

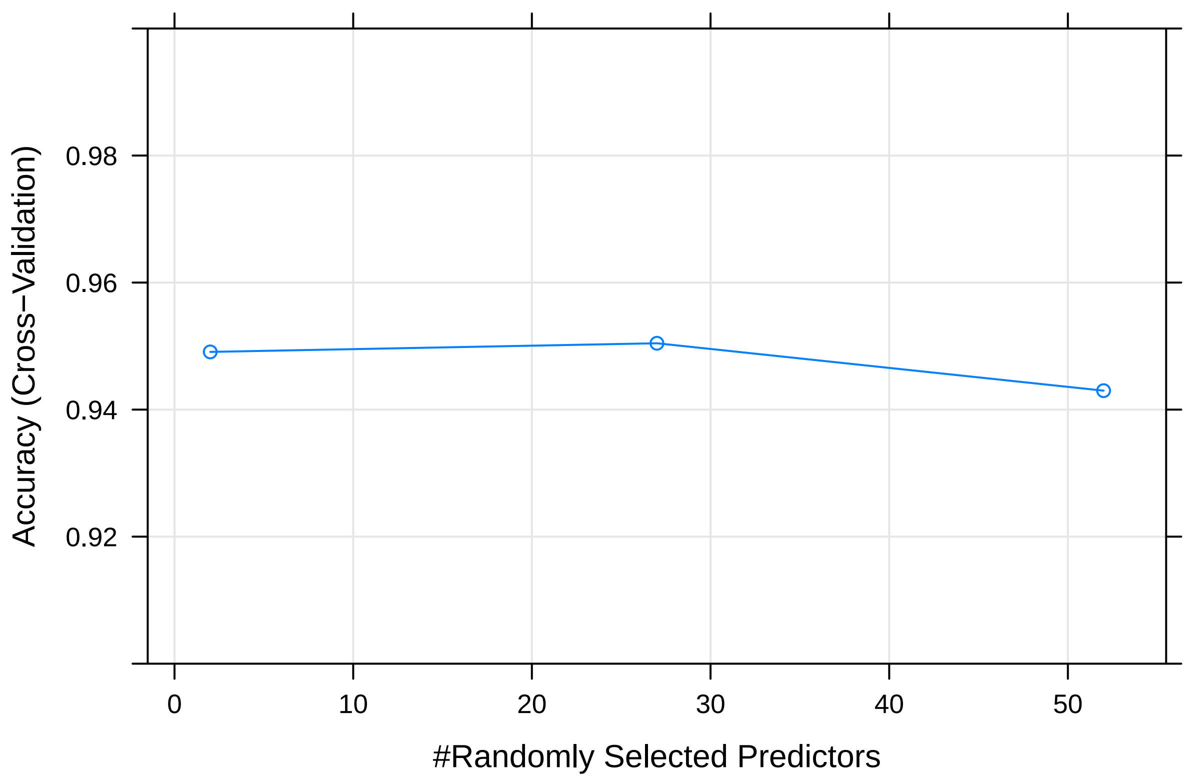
Second run adding reprocessing and cross validation:

```
set.seed(2)
##Train
modFit <- train(small_training1$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=trai
print(modFit, digits=3)
```

```
## Random Forest
```

```
##
## 2946 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2209, 2211, 2209, 2209
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.949     0.936  0.00600      0.00759
##   27    0.950     0.937  0.00701      0.00889
##   52    0.943     0.928  0.01046      0.01325
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```r
plot(modFit, ylim = c(0.9, 1))
```




```r
##Test Set
predictions <- predict(modFit, newdata=small_testing1)
cf <- confusionMatrix(predictions, small_testing1$classe)
print(cf, digits=4)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 555  11   0   1   0
##          B   3 354  15   0   4
##          C   0  13 323   9   3
##          D   0   1   3 309   1
##          E   0   1   1   2 352
##
## Overall Statistics
##
##                Accuracy : 0.9653
##                  95% CI : (0.9562, 0.973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9561
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9316   0.9444   0.9626   0.9778
## Specificity            0.9914   0.9861   0.9846   0.9970   0.9975
## Pos Pred Value         0.9788   0.9415   0.9282   0.9841   0.9888
## Neg Pred Value         0.9978   0.9836   0.9882   0.9927   0.9950
## Prevalence             0.2845   0.1938   0.1744   0.1637   0.1836
## Detection Rate         0.2830   0.1805   0.1647   0.1576   0.1795
## Detection Prevalence   0.2891   0.1917   0.1775   0.1601   0.1815
## Balanced Accuracy      0.9930   0.9588   0.9645   0.9798   0.9876
```

```
##Course Provided Test Set
print(predict(modFit, newdata=testing))
```

```
##  [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
```

```
oos2 <- 1 - cf$overall[1]
```

Where overall accuracy results **0.9653238** and out of sample error is **0.0346762**

Random Forest (Test 3)

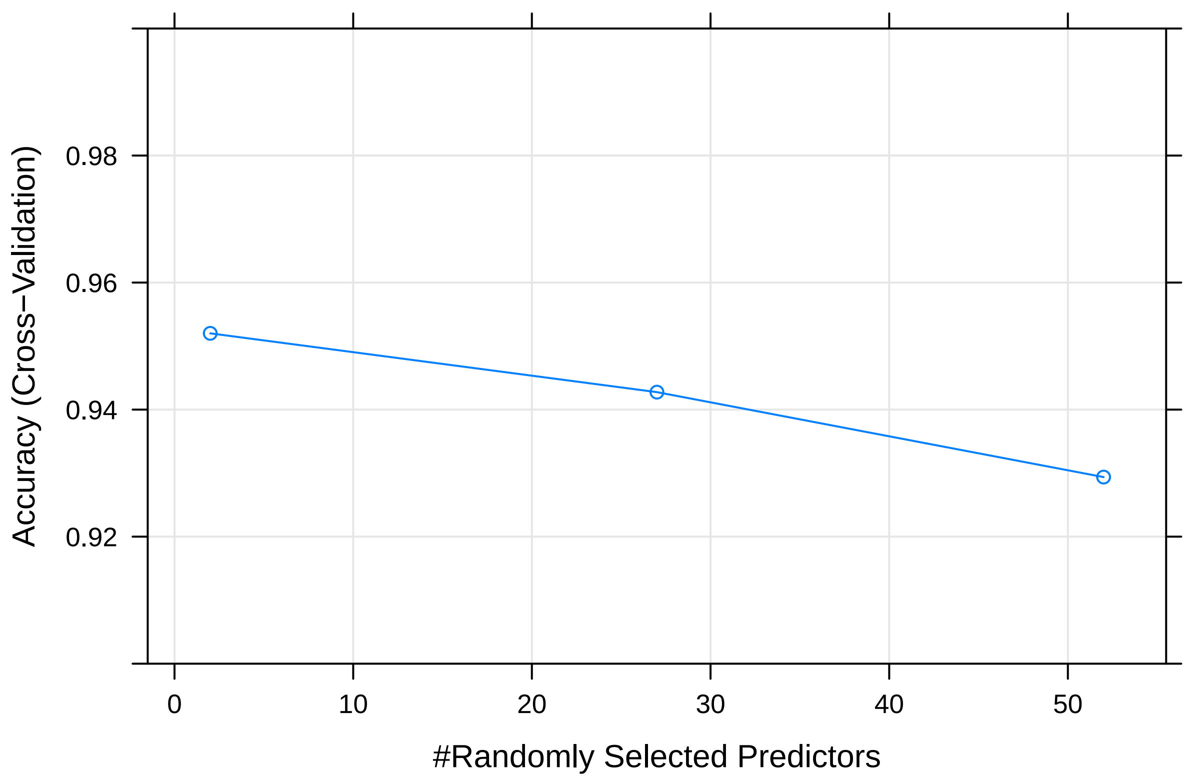
Verify results using the second training dataset:

```
set.seed(2)
##Train
modFit <- train(small_training2$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=tra
print(modFit, digits=3)
```

```
## Random Forest
```

```
##
## 2917 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2188, 2188, 2188, 2187
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.952     0.939  0.00829      0.0105
##   27    0.943     0.928  0.01212      0.0153
##   52    0.929     0.911  0.01823      0.0231
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```r
plot(modFit, ylim = c(0.9, 1))
```




```r
##Test Set
predictions <- predict(modFit, newdata=small_testing2)
cf <- confusionMatrix(predictions, small_testing2$classe)
print(cf, digits=4)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 548  26   0   1   0
##          B   1 337  18   2   1
##          C   0  11 316  20   6
##          D   3   0   4 294   5
##          E   0   2   0   1 345
##
## Overall Statistics
##
##                Accuracy : 0.948
##                  95% CI : (0.9371, 0.9574)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9341
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9928   0.8963   0.9349   0.9245   0.9664
## Specificity            0.9806   0.9859   0.9769   0.9926   0.9981
## Pos Pred Value         0.9530   0.9387   0.8952   0.9608   0.9914
## Neg Pred Value         0.9971   0.9753   0.9861   0.9853   0.9925
## Prevalence             0.2844   0.1937   0.1741   0.1638   0.1839
## Detection Rate         0.2823   0.1736   0.1628   0.1515   0.1777
## Detection Prevalence   0.2962   0.1850   0.1819   0.1577   0.1793
## Balanced Accuracy      0.9867   0.9411   0.9559   0.9586   0.9822
```

```
##Course Provided Test Set
print(predict(modFit, newdata=testing))
```

```
##  [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
```

```
oos3 <- 1 - cf$overall[1]
```

Where overall accuracy results **0.947965** and out of sample error is **0.052035**

Random Forest (Test 4)

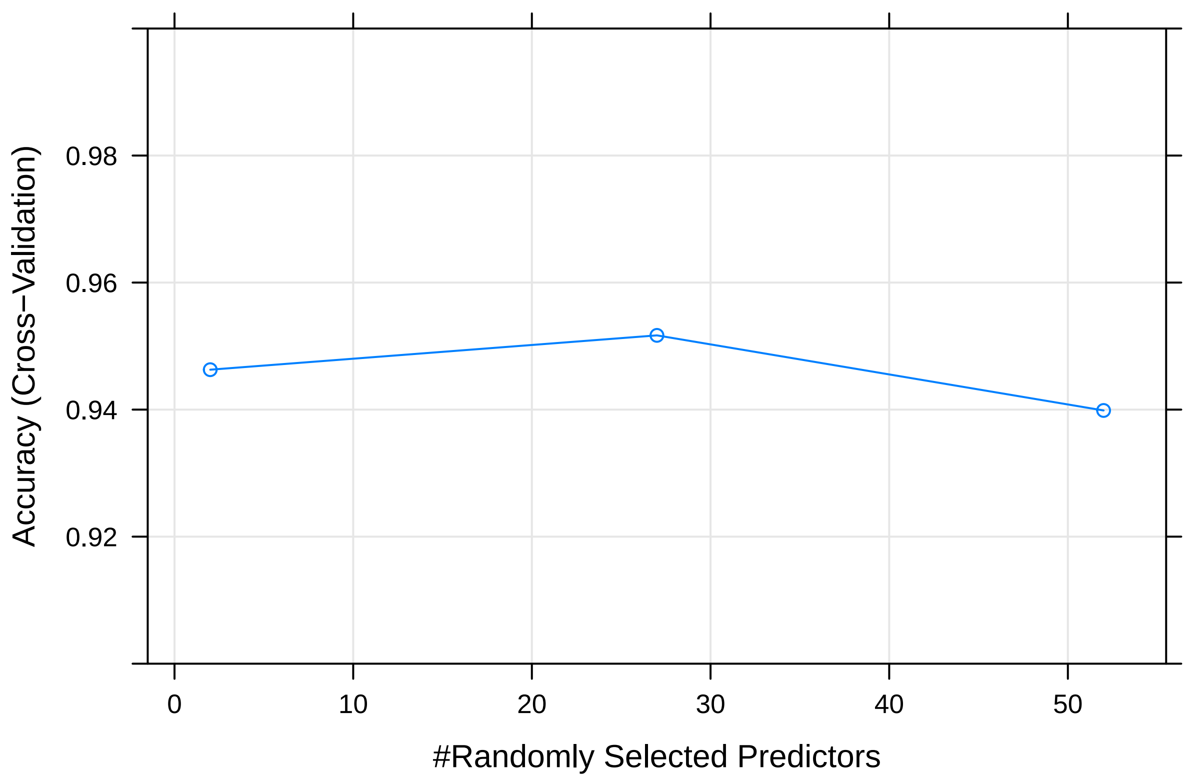
Last run on the third dataset as ulterior validation:

```
set.seed(2)
##Train
modFit <- train(small_training3$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=trai
print(modFit, digits=3)
```

```
## Random Forest
```

```
##
## 2960 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2220, 2220, 2220, 2220
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.946     0.932  0.0051       0.00642
##   27    0.952     0.939  0.0115       0.01453
##   52    0.940     0.924  0.0157       0.01987
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```r
plot(modFit, ylim = c(0.9, 1))
```




```r
##Test Set
predictions <- predict(modFit, newdata=small_testing3)
cf <- confusionMatrix(predictions, small_testing3$classe)
print(cf, digits=4)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 553  17   1   0   0
##          B   5 356   6   2   3
##          C   2   7 330   7   4
##          D   0   1   7 313   3
##          E   0   0   0   1 352
##
## Overall Statistics
##
##                Accuracy : 0.9665
##                  95% CI : (0.9576, 0.974)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9576
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9875   0.9344   0.9593   0.9690   0.9724
## Specificity            0.9872   0.9899   0.9877   0.9933   0.9994
## Pos Pred Value         0.9685   0.9570   0.9429   0.9660   0.9972
## Neg Pred Value         0.9950   0.9844   0.9914   0.9939   0.9938
## Prevalence             0.2843   0.1934   0.1746   0.1640   0.1838
## Detection Rate         0.2807   0.1807   0.1675   0.1589   0.1787
## Detection Prevalence   0.2898   0.1888   0.1777   0.1645   0.1792
## Balanced Accuracy      0.9874   0.9622   0.9735   0.9812   0.9859
```

```
##Course Provided Test Set
print(predict(modFit, newdata=testing))
```

```
##  [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
```

```
oos4 <- 1 - cf$overall[1]
```

Where overall accuracy results **0.9664975** and out of sample error is **0.0335025**

Random Forest (Test 5)

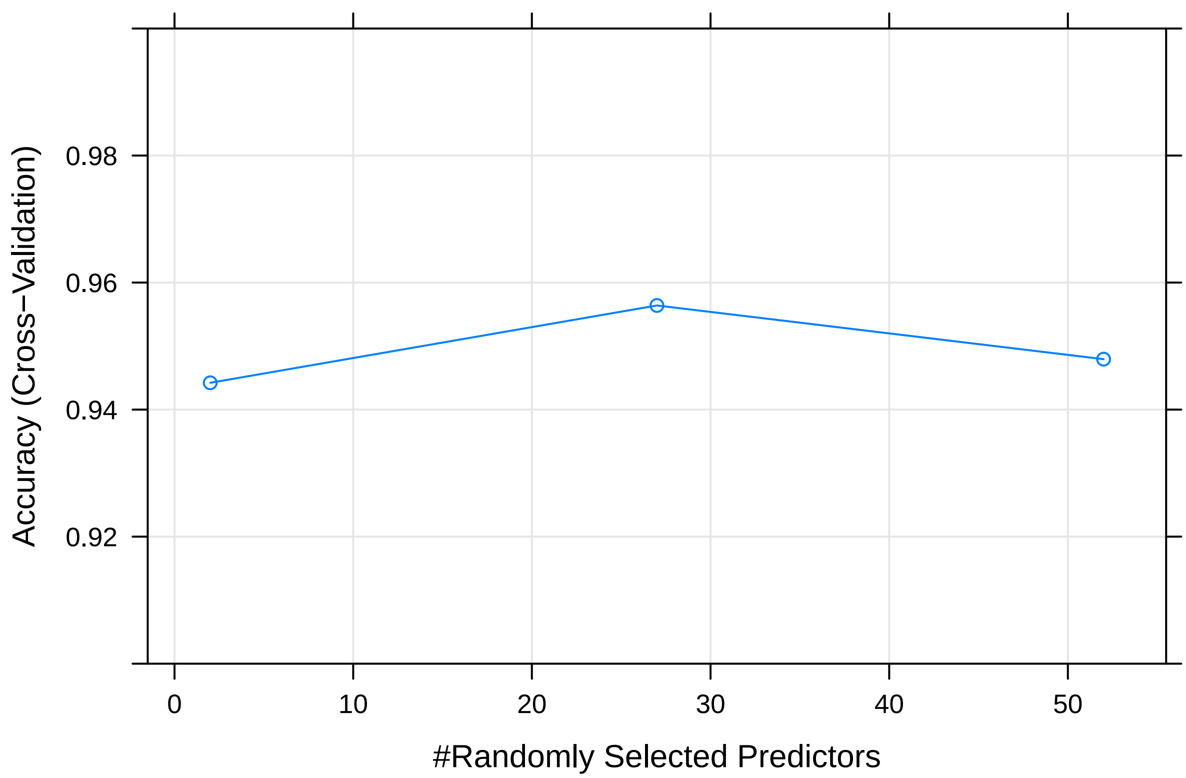
Final run on fourth dataset using preprocess and cross validation:

```
set.seed(2)
##Train
modFit <- train(small_training4$classe ~ ., method="rf", preProcess=c("center", "scale"), trControl=trai
print(modFit, digits=3)
```

```
## Random Forest
```

```
##
## 2958 samples
##   52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: centered (52), scaled (52)
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 2217, 2219, 2219, 2219
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.944     0.929  0.00712      0.00907
##   27    0.956     0.945  0.00756      0.00959
##   52    0.948     0.934  0.00810      0.01029
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
plot(modFit, ylim = c(0.9, 1))
```




```
##Test Set
predictions <- predict(modFit, newdata=small_testing4)
cf <- confusionMatrix(predictions, small_testing4$classe)
print(cf, digits=4)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   A   B   C   D   E
##          A 549  10   0   0   0
##          B   5 357  11   0   1
##          C   1  12 325   8   0
##          D   5   0   7 315  10
##          E   0   2   0   0 351
##
## Overall Statistics
##
##                Accuracy : 0.9634
##                  95% CI : (0.9542, 0.9713)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9538
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9804   0.9370   0.9475   0.9752   0.9696
## Specificity            0.9929   0.9893   0.9871   0.9866   0.9988
## Pos Pred Value         0.9821   0.9545   0.9393   0.9347   0.9943
## Neg Pred Value         0.9922   0.9850   0.9889   0.9951   0.9932
## Prevalence             0.2844   0.1935   0.1742   0.1640   0.1838
## Detection Rate         0.2788   0.1813   0.1651   0.1600   0.1783
## Detection Prevalence   0.2839   0.1899   0.1757   0.1712   0.1793
## Balanced Accuracy      0.9866   0.9632   0.9673   0.9809   0.9842
```

```r
## Course Provided Test Set
print(predict(modFit, newdata=testing))
```

```
##  [1] B A B A A E D B A A B C B A E E A B A B
## Levels: A B C D E
```

```r
oos5 <- 1 - cf$overall[1]
```

Where overall accuracy results **0.9634332** and out of sample error is **0.0365668**

Out of Sample Error Rate

The average of the sample error rates generated by the random forest method using pre-processing and cross validation on the 5 test sets provided a predicted out of sample rate of 0.0407391.