PML-CourseProject

Ariel

Summary

We use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Our objective is to predict the manner in which they did the exercise (i.e., he **classe** variable in the data).

Reading data

```
trainingRaw <- read.csv("training.csv")
testingRaw <- read.csv("testing.csv")
```

Cleaning data

Removing variables with too many NA's

```
tooManyNAs <- function(x) sum(is.na(x))>(.1*length(x))
badvarsTrain <- which(sapply(trainingRaw, tooManyNAs))
badvarsTest <- which(sapply(testingRaw, tooManyNAs))
# note that badvarsTest contains the variables to exclude
any(intersect(badvarsTest, badvarsTrain) != badvarsTrain)
```

```
## [1] FALSE
```

```
# we keep just the variables with not too many NAs
training <- trainingRaw[, -badvarsTest]
testing <- testingRaw[, -badvarsTest]
# there is no more NAs in our data
c(anyNA(training), anyNA(testing))
```

```
## [1] FALSE FALSE
```

Partitioning training data to validate (i.e., to estimate out of sample error using the validation data)

```
library(caret)
inTrain <- createDataPartition(training$classe, p = .75, list = F)
validation <- training[-inTrain,]
training <- training[inTrain,]
```

We fit a classification tree (named: **mdl1**) plaining to use varImp function to filter the most influential variables. This first model makes us note that variable **X** (the order of the samples) is enough to predict the outcome (because the outcome is ordered!!).

```
mdl1 <- train(classe ~ ., method = "rpart", data = training)
mdl1$finalModel
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

```
## 1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##   2) X< 5580.5 4185     0 A (1 0 0 0 0) *
##   3) X>=5580.5 10533  7685 B (0 0.27 0.24 0.23 0.26)
##     6) X< 9378.5 2848     0 B (0 1 0 0 0) *
##     7) X>=9378.5 7685  4979 E (0 0 0.33 0.31 0.35) *
```

So we ignore X and fit a new classification tree (named: **mdl2**) Finaly, we just keep the predictors signaled by **varImp(mdl2)**.

```
mdl2 <- train(classe ~ ., method = "rpart", data = training[,-1])
impList2 <- varImp(mdl2)
# names of influential variables (or levels of factor variables)
impVars <- rownames(impList2$importance)[which(impList2$importance[,1] > 0)]
# indexes of influential variables
finalvarsIndex <- c(which(names(training) %in% impVars), 5, 60)
# keeping just the variables that "matter"
training <- training[, finalvarsIndex]
validation <- validation[, finalvarsIndex]
testing <- testing[, finalvarsIndex]
```

Constructing our final models

First we fit three models (using methods: **gbm**, **rf**, and **lda**) on the training data. We measure the out of sample error of each model and the blending of the three. The results are that the first two models have excellent accuracy, leaving almost no room for improvements made with the blending.

```
library(gbm)
gbmmdl <- train(classe ~ ., data = training, method = "gbm", verbose = F,
                trControl = trainControl(method = "cv", number = 3))
rfmdl <- train(classe ~ ., data = training, method = "rf",
               trControl = trainControl(method = "cv", number = 3))
ldamdl <-  train(classe ~ ., method = "lda", data = training)
# to make the blending of the three models
predDF <- data.frame(predict(rfmdl, validation),
                     predict(gbmmdl, validation),
                     predict(ldamdl, validation),
                     classe=validation$classe)
# combined model
combModFit <- train(classe ~.,method="rf",data=predDF,
                    trControl = trainControl(method = "cv", number = 2))
combPred <- predict(combModFit,predDF)
# showing accuracy of the three models
for(i in 1:3) print(confusionMatrix(validation$classe, predDF[,i])[[3]])
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9987765      0.9984523      0.9973389      0.9995509      0.2850734
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9971452      0.9963889      0.9952147      0.9984384      0.2848695
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.7228793      0.6500773      0.7101203      0.7353724      0.2689641
## AccuracyPValue  McnemarPValue
```

```
##      0.0000000            NaN
```

```
# showing accuracy of blended model
confusionMatrix(validation$classe, combPred)[[3]]
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9991843      0.9989683      0.9979129      0.9997777      0.2846656
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Predicting on the Test data

Finally, we predict the outcome on the test data. As the first two models agree perfectly on the prediction it is not necessary to use the third model (our original idea was to use majority vote to obtain the prediction on the test data). Any of **rfpred** and **gbmpred** predictions will provide our answer to the Quiz portion of the course project.

```
rfpred <- predict(rfmdl, testing)
gbmpred <- predict(gbmmdl, testing)
any(rfpred != gbmpred)
```

```
## [1] FALSE
```

Final observations

Note that we did not re-trained our models merging training and validation data, this is because the accuracy obtained with the training data was large enough. For the same reason we didn't performed any preprocessing, creation of new predictors, or any kind of tuning of our models to improve prediction accuracy.