PML course project - Qualitative activity recognition:

dnp

Friday, March 20, 2015

Executive summary

In this project, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, that were asked to perform barbell lifts correctly and incorrectly in 5 different ways, in order to predict the manner in which they did the exercise. This is the **classe** variable in the training set.

We train a predictive model with the **Random Forest** method and **Repeated Cross Validation** and use it to predict 20 different test cases.

Loading and cleaning data

We clean the dataset from features that are: - irrelevant to our predictive task (first 8 columns) - predominantly NAs - highly correlated

Load data sets

```
validset <- read.csv("pml-testing.csv")
dt <- read.csv("pml-training.csv", na.strings = "#DIV/0!")
dim(dt)
```

```
## [1] 19622   160
```

Check and filter irrelevant featuress

```
dt <- dt[, 8:length(dt)] # date, timestamp, etc
```

Check and filter features with NAs

```
classes <- sapply(dt, class)
table(classes) # check classes of variables
```

```
## classes
##  factor integer logical numeric
##      68      25       6      54
```

```
index_logical <- which(classes == "logical")
index_factor <- which(classes == "factor")
check_logical <- sapply(dt[ ,index_logical], summary) # check for NAs
check_factor <- sapply(dt[ ,index_factor], summary) # check for NAs
index_factor <- index_factor[-68] # classe variable should be factor
dt <- dt[, -c(index_logical, index_factor)] # delete NA variables
na_cols <- sapply(dt, anyNA) #check more columns for NA values
index_na <- which(na_cols)
check_NAs <- sapply(dt[ ,index_na], summary)
dt <- dt[, -index_na] # delete more NA variables
```

Check and filter correlated features

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
descrCor <- cor(dt[, -53])
highlyCorDescr <- findCorrelation(descrCor, cutoff = .75)
```

Final clean dataset

```
fdt <- dt[, -highlyCorDescr]
dim(fdt)
```

```
## [1] 19622    33
```

Built predictive model

Because we have a large dataset, we will built a Random Forest model, which is characterised by its high predictive accuracy and minimal parameter tuning. In summary, we devide the data into a training and a testing set, fit the model with the training set and evaluate performance with the testing set.

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
## Loading required package: iterators
```

Data partition

```
set.seed(222)
inTrain <- createDataPartition(y=fdt$classe, p=0.6, list=FALSE)
training <- fdt[inTrain,]
testing <- fdt[-inTrain,]
```

Model fit

- **Random Forest**
- Train control method: **Repeated CV**
- **Repeats: 3**

```
ctrl <- trainControl(method = "repeatedcv", repeats = 3)

registerDoParallel(clust <- makeCluster(detectCores()))
set.seed(222)
fit <- train(classe ~ ., method = "rf", data = training,
                 trControl = ctrl, tuneLength = 6)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
stopCluster(clust)

fit
```

```
## Random Forest
##
## 11776 samples
##    32 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
##
## Summary of sample sizes: 10598, 10598, 10599, 10597, 10598, 10599, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9887910  0.9858193  0.003478605  0.004402979
##    8    0.9897531  0.9870380  0.003154607  0.003990346
##   14    0.9881964  0.9850681  0.002894141  0.003662336
##   20    0.9860169  0.9823106  0.003143602  0.003977179
##   26    0.9839506  0.9796964  0.002849597  0.003606035
##   32    0.9803274  0.9751123  0.004225011  0.005345461
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 8.
```

```
predictions <- predict(fit, newdata = testing)
```

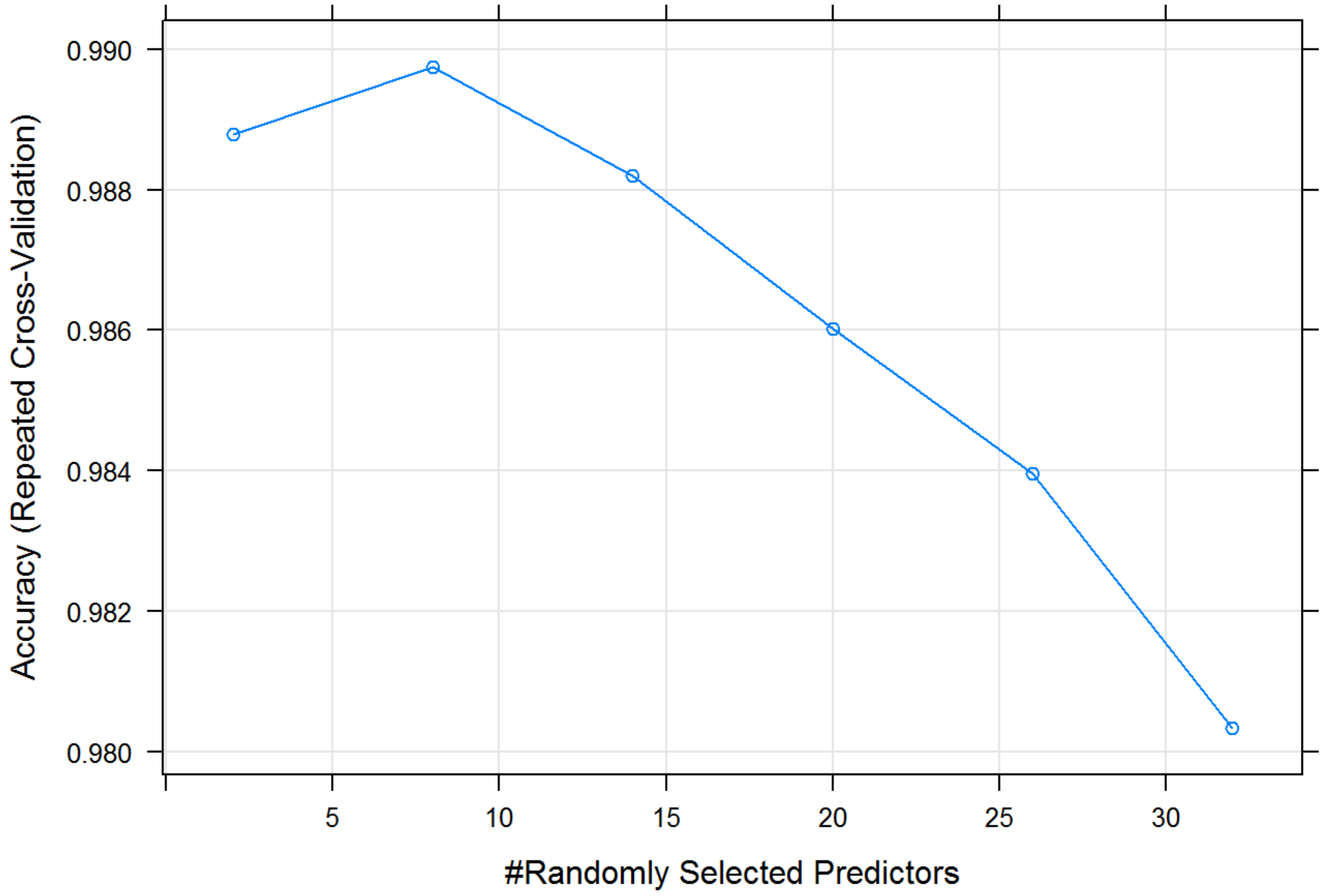
Model evaluation

We assess our model’s performance on the test set, using various statistics.

Evaluation metrics

Cross validation:

```
plot(fit)
```



Accuracy and error rates were estimated with **Repeated Cross-Validation**. Accuracy was used to select the optimal model using the largest value. The final value used for the model was with 8 selected predictors.

Accuracy and Out of sample error:

```
fit$finalModel
```

```
## 
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry) 
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 8
## 
##         OOB estimate of  error rate: 0.95%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3342    4    0    2    0 0.001792115
## B   15 2252    7    1    4 0.011847301
## C    0   30 2006   17    1 0.023369036
## D    0    0   23 1906    1 0.012435233
## E    0    1    0    6 2158 0.003233256
```

The above printout shows **the OOB estimate of error rate** and the class error rates of the model’s fit.

Confusion matrix and statistics:

```
confusionMatrix(predictions, testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   17    0    2    0
##          B    0 1497    6    0    0
##          C    0    4 1356   13    3
##          D    0    0    5 1267    2
##          E    1    0    1    4 1437
## 
## Overall Statistics
## 
##                Accuracy : 0.9926
##                  95% CI : (0.9905, 0.9944)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9906
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9862   0.9912   0.9852   0.9965
## Specificity            0.9966   0.9991   0.9969   0.9989   0.9991
## Pos Pred Value         0.9916   0.9960   0.9855   0.9945   0.9958
## Neg Pred Value         0.9998   0.9967   0.9981   0.9971   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1908   0.1728   0.1615   0.1832
## Detection Prevalence   0.2868   0.1916   0.1754   0.1624   0.1839
## Balanced Accuracy      0.9981   0.9926   0.9941   0.9921   0.9978
```

The overall **Accuracy rate** is 99.26%. The **Null or No Information rate**, which is the largest proportion of the observed classes, is 28.45%. The overall accuracy rate is greater than the rate of the largest class.

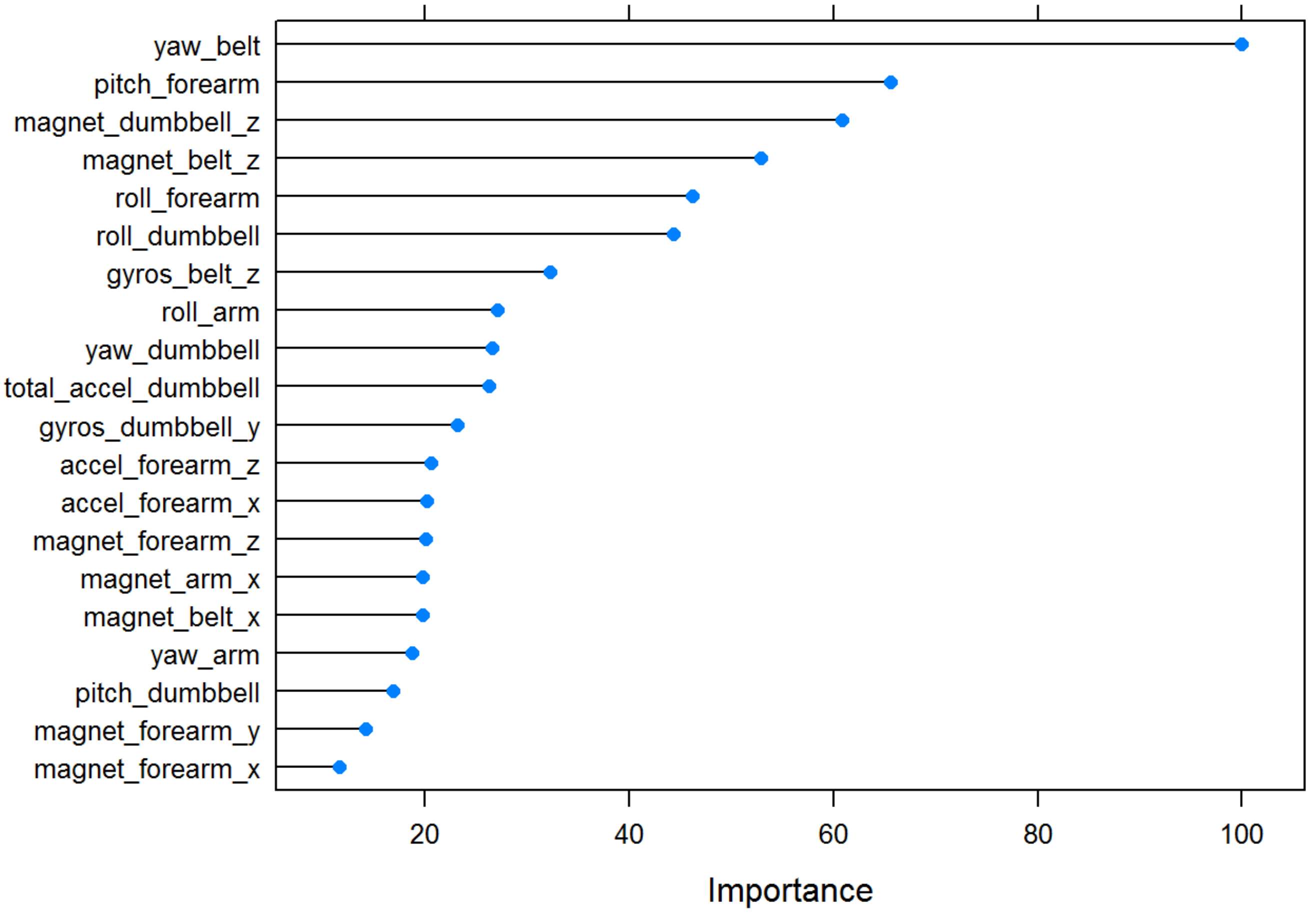
The **out of sample error rate** is 0.74%.

Variable importance:

```
varImp(fit, scale = F)
```

```
## rf variable importance
## 
##   only 20 most important variables shown (out of 32)
## 
##                       Overall
## yaw_belt              1050.1
## pitch_forearm          711.0
## magnet_dumbbell_z      663.9
## magnet_belt_z          585.5
## roll_forearm           519.1
## roll_dumbbell          501.1
## gyros_belt_z           381.5
## roll_arm               331.0
## yaw_dumbbell           326.0
## total_accel_dumbbell   323.2
## gyros_dumbbell_y       292.7
## accel_forearm_z        267.3
## accel_forearm_x        263.2
## magnet_forearm_z       262.0
## magnet_arm_x           258.3
## magnet_belt_x          258.3
## yaw_arm                248.6
## pitch_dumbbell         230.0
## magnet_forearm_y       203.5
## magnet_forearm_x       178.0
```

```
plot(varImp(fit), top = 20)
```



Prediction

Finally, we use our model to predict 20 different test cases.

```
predict(fit, newdata = validset)
```

```
##  [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
```

Conclusion

We trained a Random Forest predictive model with repeated cross-validation to our final 32-feature cleaned dataset. Our model has an accuracy on the testing set of 99.26%. We use our model to predict the variable **classe** for 20 different test cases. Classe represents the manner in which they did the exercise.

dnp