The type of doing exercise

In this paper is presented the model to determine a type of performing a physical exercise with a dumbbell. The model is based on information collected by the sensors during the exercise.

For the construction of the model used training data set and a test data set of 20 observations for final check of the model.

After the download, an exploratory data analysis was performed, which showed that for each observation there are 160 variables, on of which is target.

The task is to predict the value of the target variable. The target variable (classe) can take only five different values: A B C D E, therefore, suppose that in this case we have to solve the classification task. To solve such tasks, the method of learning trees is usually used.

Also, an exploratory analysis showed that 160 variables contain many missing (NA) values. In the end it will be necessary to predict target variable in test set of 20 observation, therefore, the exploratory analysis was performed for this set too.

Analysis showed that in the test sample, 100 variables contain only NA values, therefore these variables will not help in predicting the values of the target variable in test data set, so it can be removed from the training set.

I also removed time and window type variables, because the meaning of the first one is not clear, and the second one obviously does not affect a value of the target variable.

As a result, 54 variables remained, on of which is target.

Loading data and removing empty variables

```
data <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training
.csv")
test <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.
csv")
cnames <- colnames(test[,is.na(test[1,])])
test_clear <- test[,!names(test) %in% cnames]
test_clear <- test_clear[,-c(1)]
test_clear <- test_clear[,-c(2,3,4,5,6)]
names(test_clear)[54] <- "classe"
data1 <- data[,names(test_clear)]
```

Separation of training data set on training and testing part

```
library(caret)
```

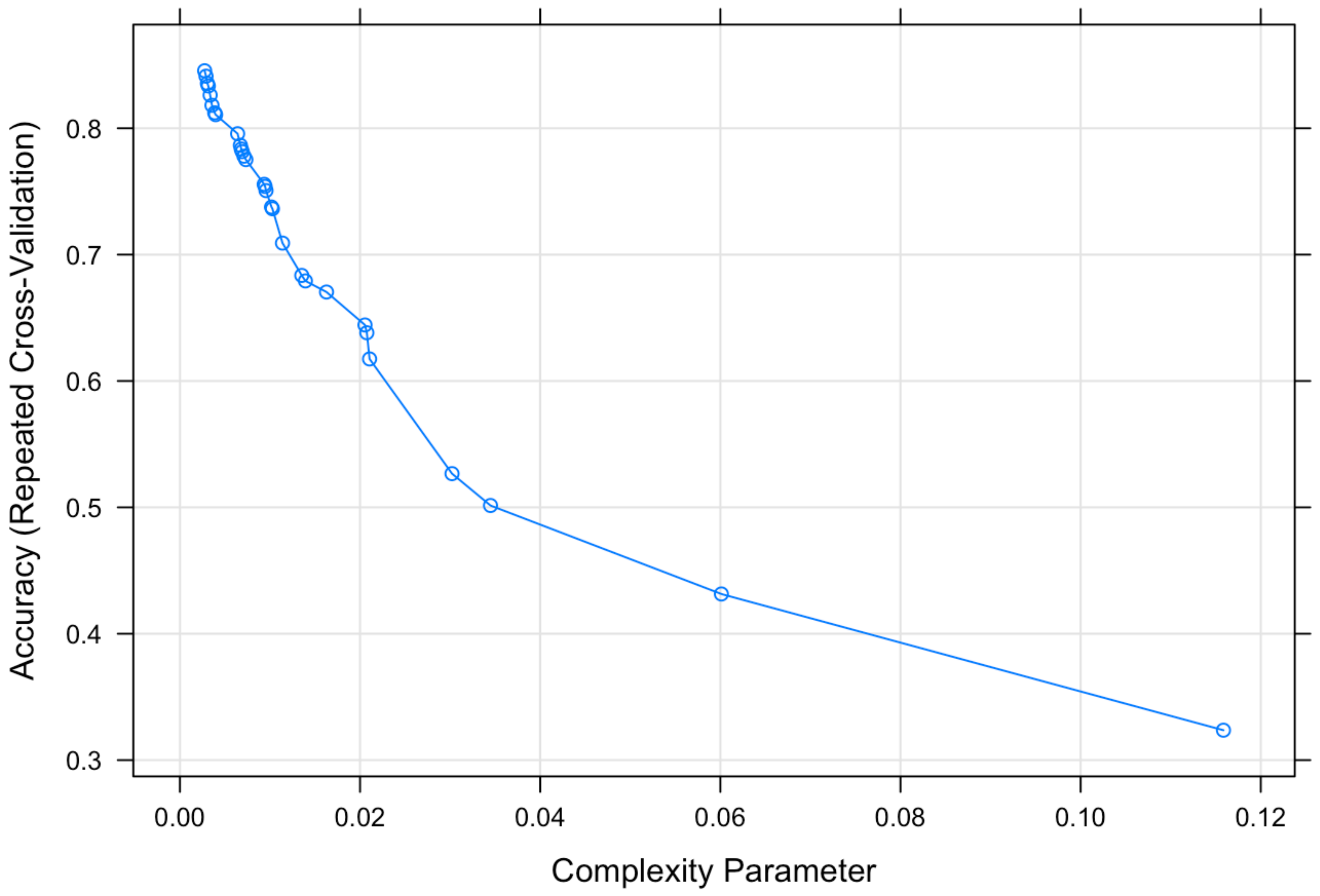
```
## Loading required package: lattice
```

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

```
set.seed(3435)
inTrain <- createDataPartition(y=data1$classe, p=0.7, list = FALSE)
training <- data1[inTrain,]
testing <- data1[-inTrain,]
```

Try to build a learning tree using cross-validation method with 30 different values of the penalty for the complexity and 3 repetition.

```
control <- trainControl(method = "repeatedcv", repeats = 3)
modfit <- train(classe~., data=training, method="rpart", tuneLength=30, trControl=
control)
plot(modfit)
```



```
modfit
```

```
## CART
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 12363, 12363, 12365, 12362, 12362, 12363, ...
## Resampling results across tuning parameters:
##
##   cp           Accuracy   Kappa
##   0.002746414  0.8454817  0.80431080
##   0.002898993  0.8411855  0.79884958
##   0.003051572  0.8354829  0.79161452
##   0.003153291  0.8334693  0.78907168
##   0.003356729  0.8262152  0.77994122
##   0.003560167  0.8181344  0.76974031
##   0.003865324  0.8121889  0.76228302
##   0.003967043  0.8107331  0.76047434
##   0.006408300  0.7956649  0.74132062
##   0.006713457  0.7865158  0.72958321
##   0.006815176  0.7835562  0.72581857
##   0.006916896  0.7814695  0.72311243
##   0.007120334  0.7778531  0.71847804
##   0.007323772  0.7751605  0.71506652
##   0.009358153  0.7556246  0.69009573
##   0.009459872  0.7539259  0.68791833
##   0.009561591  0.7506500  0.68376081
##   0.010171905  0.7376197  0.66711352
##   0.010273624  0.7362855  0.66544808
##   0.011392534  0.7091118  0.63206902
##   0.013528634  0.6835293  0.60067697
##   0.013935510  0.6791854  0.59546882
##   0.016275048  0.6704018  0.58452445
##   0.020547248  0.6443205  0.55273789
##   0.020750687  0.6381593  0.54429931
##   0.021055844  0.6174346  0.51480185
##   0.030210558  0.5266769  0.38268501
##   0.034482759  0.5014700  0.34878376
##   0.060115960  0.4314565  0.23487811
##   0.115858000  0.3236561  0.06024662
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002746414.
```

By the constructed model, it can be seen that the best value value of the accuracy of the tree is slightly more than 80%

Check the resulting model on a testing set and estimate a prediction accuracy

```
pred <- predict(modfit, newdata = testing)
testing$predRight <- pred == testing$classe
table(pred, testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1551  106   29   36   21
##    B   71  850   57   33   42
##    C   13   97  851   41   41
##    D   29   46   77  801   82
##    E   10   40   12   53  896
```

```
table(testing$classe, testing$predRight)
```

```
##
##      FALSE TRUE
##   A    123 1551
##   B    289  850
##   C    175  851
##   D    163  801
##   E    186  896
```

```
table(testing$classe)
```

```
##
##    A    B    C    D    E
## 1674 1139 1026  964 1082
```

```
c<-table(testing$classe, testing$predRight)
x<-t(t(table(testing$classe)))
c <- cbind(c,x)
colnames(c)[3]<-"All"
true_percent <- c[,2]/c[,3]
c<-cbind(c, true_percent)
c
```

```
##   FALSE TRUE  All true_percent
## A   123 1551 1674    0.9265233
## B   289  850 1139    0.7462687
## C   175  851 1026    0.8294347
## D   163  801  964    0.8309129
## E   186  896 1082    0.8280961
```

It can be seen that the best result is for type A events - more than 90%, worst - for type B events - about 70%, and an overall accuracy is about 80%.