Course 8 Project

Nupur Sinha

5/30/2020

Practical Machine Learning: Human Activity Recognition

One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which they did the exercise. This is the "classe" variable in the training set.

```
# Loading all required libraries
library(AppliedPredictiveModeling)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.5.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.5.2
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.2
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.5.2

## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.4
```

Reading the data

The data for this project comes from this source: http://groupware.les.inf.puc-rio.br/har.

Train set

```r
train_file_name <- "pml-training.csv"
train_fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

if(!file.exists(train_file_name)) {
  download.file(train_fileUrl, train_file_name, method = "curl")
}

training <- read.csv(file = train_file_name, header = T)

dim(training)
```

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

```r
unique(training$classe)
```

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

Validation set

```r
validation_set_file_name <- "pml-testing.csv"
validation_set_fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if(!file.exists(validation_set_file_name)) {
  download.file(validation_set_fileUrl, validation_set_file_name, method = "curl")
}
```

```
validation_set  <- read.csv(file = validation_set_file_name, header = T)
dim(validation_set)
```

```
## [1]  20 160
```

Cleaning the data

Before we build models, we need to clean the data.

1. There are some identification columns (username, etc., columns 1 to 7) that we can exclude from our analysis.
2. We will then exclude any variables that have near zero variance.
3. We will also exclude any variables that has zero to very low fill rate.

Whatever preprocessing and transformation we apply to train set, we will apply the same to the validation set.

```
set.seed(123)

# 1. Exclude identification variables i.e. columns 1 to 7

training <- training[, -(1:7)]
validation_set  <- validation_set[, -(1:7)]

dim(training)
```

```
## [1] 19622   153
```

```
dim(validation_set)
```

```
## [1]  20 153
```

```
# 2. Exclude variables with near zero variation

nzv_var <- nearZeroVar(training)
training <- training[, -nzv_var]
validation_set  <- validation_set[, -nzv_var]

dim(training)
```

```
## [1] 19622    94
```

```
dim(validation_set)
```

```
## [1] 20 94
```

```
# 3. Exclude variables with very zero to very low fill rate

training<- training[, colSums(is.na(training)) == 0]
validation_set <- validation_set[, colSums(is.na(validation_set)) == 0]
dim(training)
```

```
## [1] 19622    53
```

```
dim(validation_set)
```

```
## [1] 20 53
```

Create train and test sets

```
inTrain <- createDataPartition(training$classe, p = 0.7, list = FALSE)
trainData <- training[inTrain, ]
testData <- training[-inTrain, ]
dim(trainData)
```

```
## [1] 13737    53
```

```
dim(testData)
```

```
## [1] 5885   53
```

Building models

We will build three models along with cross validation method using the train data and predict on the test set. We will compare the accuracy of the three models to pick the best one to predict on the validation set.

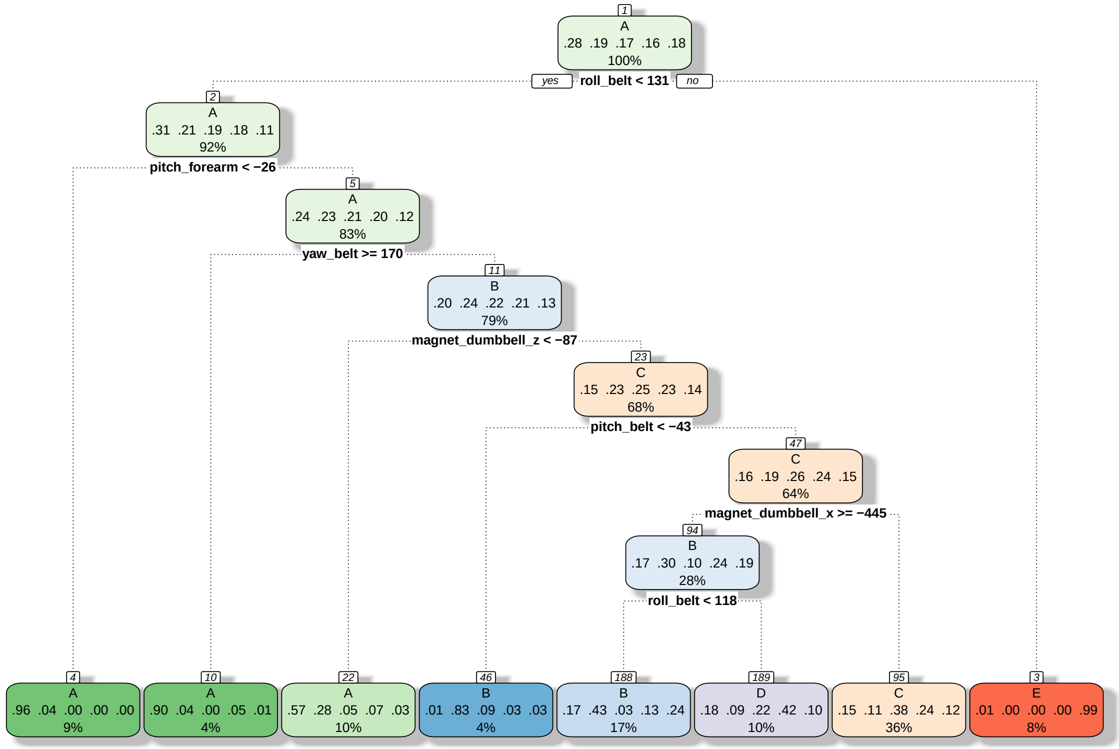
1. Decision tree model

```
# Cross validation
fitControl <- trainControl(method='cv', number = 3)

# Decision Tree model
modFit_decision_tree <- train(classe~., data=trainData, method="rpart", trControl=fitControl)
modFit_decision_tree$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##     2) roll_belt< 130.5 12580 8681 A (0.31 0.21 0.19 0.18 0.11)
##       4) pitch_forearm< -26.45 1238   45 A (0.96 0.036 0 0 0) *
##       5) pitch_forearm>=-26.45 11342 8636 A (0.24 0.23 0.21 0.2 0.12)
##        10) yaw_belt>=169.5 550   56 A (0.9 0.044 0 0.049 0.0091) *
##        11) yaw_belt< 169.5 10792 8203 B (0.2 0.24 0.22 0.21 0.13)
##          22) magnet_dumbbell_z< -87.5 1420  615 A (0.57 0.28 0.051 0.075 0.025) *
##          23) magnet_dumbbell_z>=-87.5 9372 7048 C (0.15 0.23 0.25 0.23 0.14)
##            46) pitch_belt< -42.95 576   96 B (0.014 0.83 0.092 0.033 0.028) *
##            47) pitch_belt>=-42.95 8796 6525 C (0.16 0.19 0.26 0.24 0.15)
##              94) magnet_dumbbell_x>=-445.5 3783 2642 B (0.17 0.3 0.097 0.24 0.19)
##               188) roll_belt< 117.5 2357 1340 B (0.17 0.43 0.025 0.13 0.24) *
##               189) roll_belt>=117.5 1426  830 D (0.18 0.087 0.22 0.42 0.097) *
##              95) magnet_dumbbell_x< -445.5 5013 3110 C (0.15 0.11 0.38 0.24 0.12) *
##     3) roll_belt>=130.5 1157    7 E (0.0061 0 0 0 0.99) *
```

```
# Plot
fancyRpartPlot(modFit_decision_tree$finalModel)
```



Rattle 2020−May−31 18:57:45 nupursinha

```r
# Predict on the test data
pred_decision_tree <- predict(modFit_decision_tree, newdata = testData)

# Accuracy of the decision tree model
decision_tree_cm <- confusionMatrix(testData$classe, pred_decision_tree)
decision_tree_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061  163  341  102    7
##          B  235  631  230   43    0
##          C   27   42  819  138    0
##          D   64  133  509  258    0
##          E   13  281  247   60  481
##
## Overall Statistics
##
##                Accuracy : 0.5523
##                  95% CI : (0.5394, 0.565)
##     No Information Rate : 0.3647
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4373
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7579   0.5048   0.3816  0.42928  0.98566
## Specificity            0.8633   0.8904   0.9446  0.86639  0.88864
## Pos Pred Value         0.6338   0.5540   0.7982  0.26763  0.44455
## Neg Pred Value         0.9195   0.8696   0.7269  0.93030  0.99854
## Prevalence             0.2379   0.2124   0.3647  0.10212  0.08292
## Detection Rate         0.1803   0.1072   0.1392  0.04384  0.08173
## Detection Prevalence   0.2845   0.1935   0.1743  0.16381  0.18386
## Balanced Accuracy      0.8106   0.6976   0.6631  0.64784  0.93715
```

```r
decision_tree_accuracy <- decision_tree_cm$overall[1]
decision_tree_accuracy
```

```
##  Accuracy
## 0.5522515
```

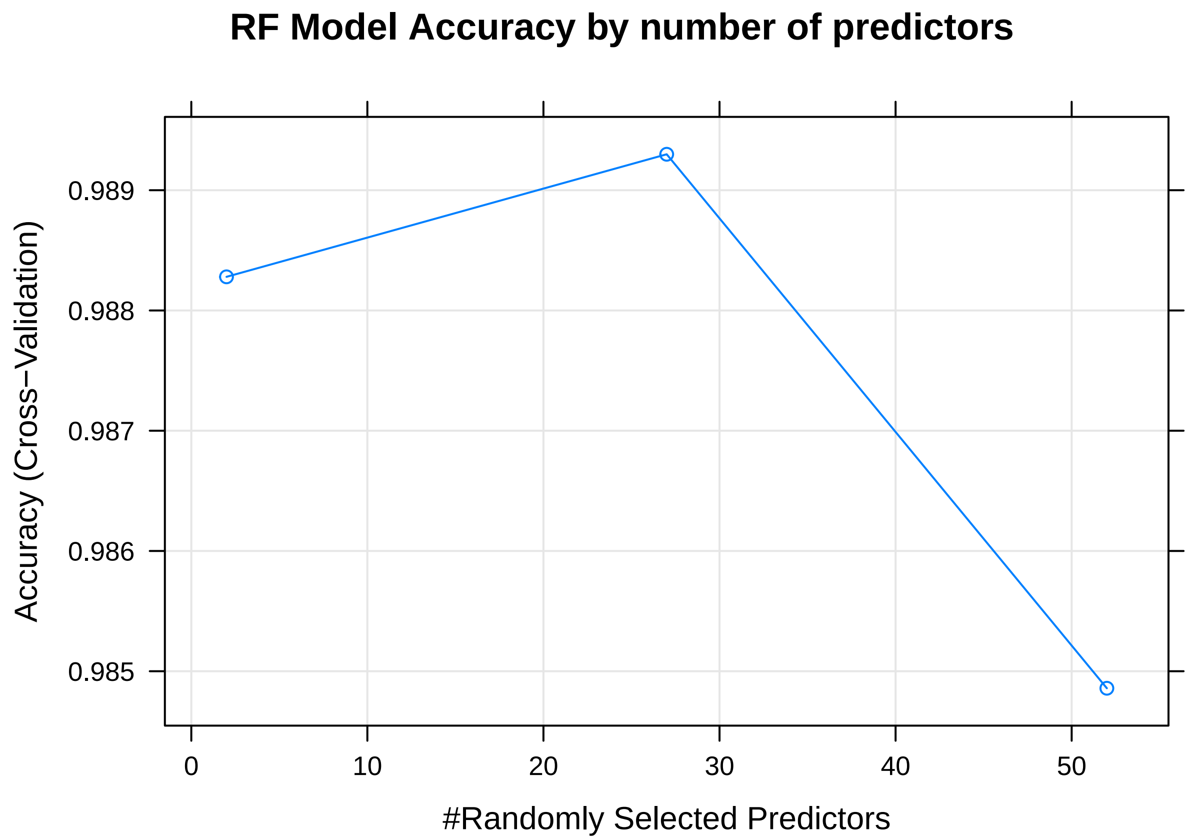
Accuracy of the decision tree model is only 0.55 => Out of sample error is 0.45 which is high

2. Random forest model

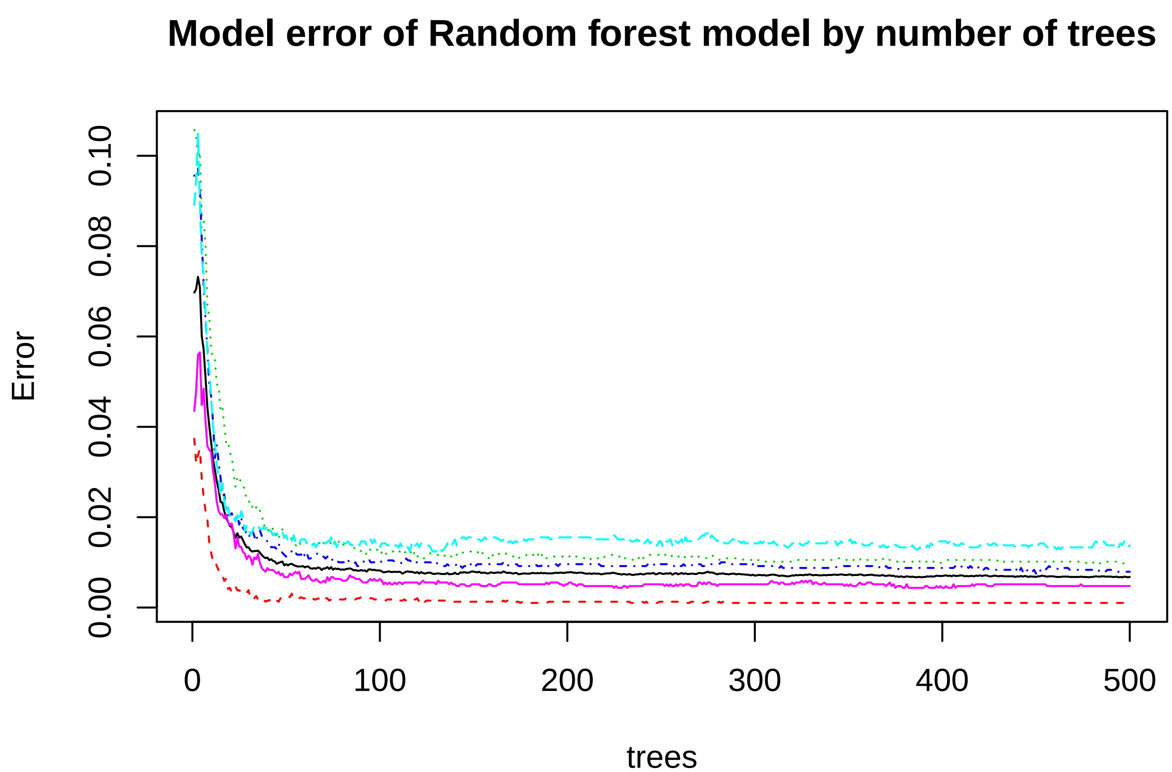
```r
# Cross validation
controlRF <- trainControl(method = "cv", number = 3, verboseIter = FALSE)

# Random forest model
modFit_rf  <- train(classe~., method = "rf", data = trainData, trControl=controlRF)

# Plot
plot(modFit_rf, main="RF Model Accuracy by number of predictors")
```




```r
plot(modFit_rf$finalModel, main="Model error of Random forest model by number of trees")
```



```r
# Predict on the test data
pred_rf <- predict(modFit_rf, newdata = testData)

# Accuracy of the random forest model
rf_cm <- confusionMatrix(testData$classe, pred_rf)
rf_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    6 1131    2    0    0
##          C    0    9 1014    3    0
##          D    0    0   17  946    1
##          E    0    0    1    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.9906, 0.995)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9912
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9912   0.9807   0.9958   0.9991
## Specificity            0.9998   0.9983   0.9975   0.9964   0.9996
## Pos Pred Value         0.9994   0.9930   0.9883   0.9813   0.9982
## Neg Pred Value         0.9986   0.9979   0.9959   0.9992   0.9998
## Prevalence             0.2853   0.1939   0.1757   0.1614   0.1837
## Detection Rate         0.2843   0.1922   0.1723   0.1607   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9981   0.9948   0.9891   0.9961   0.9993
```

```r
rf_accuracy <- rf_cm$overall[1]
rf_accuracy
```

```
##  Accuracy
## 0.9930331
```

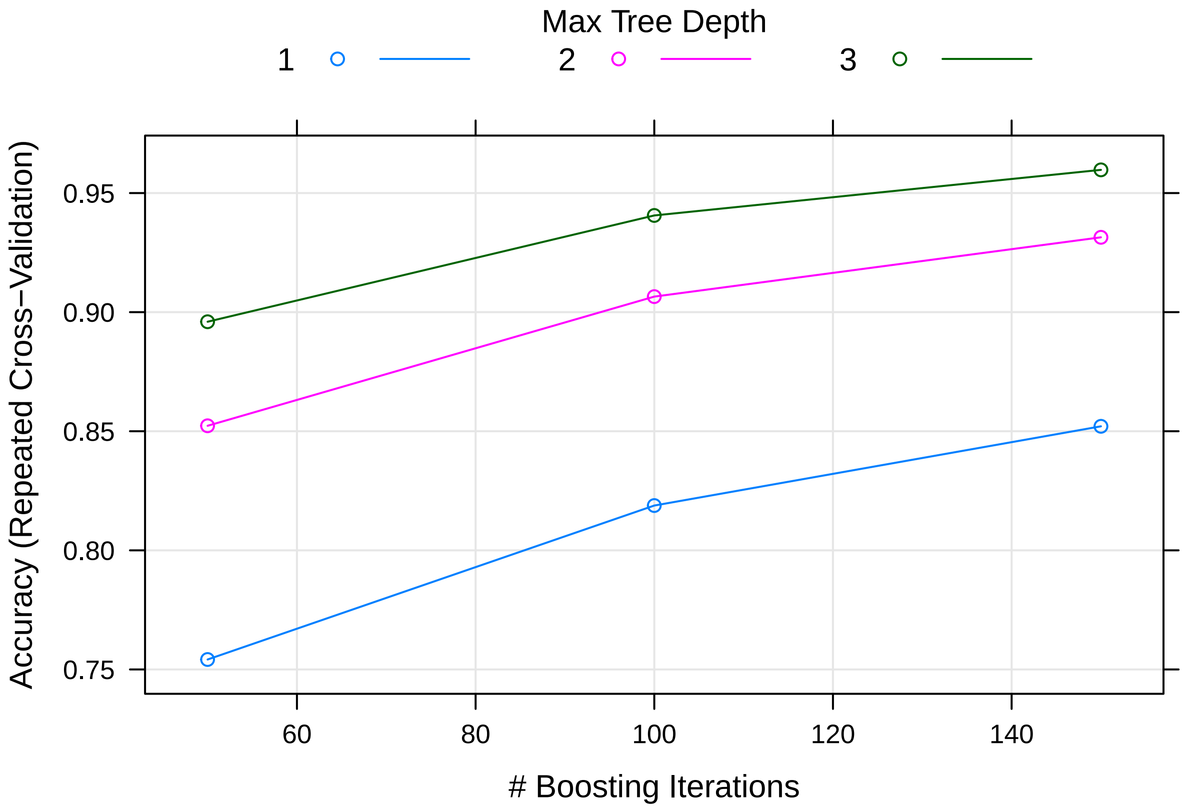
Accuracy of the random forest model is 0.9930331 => Out of sample error is 0.0069

3. GBM

```r
# Cross validation
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)

# GBM
modFit_gbm <- train(classe ~ ., method = "gbm", data = trainData, trControl=controlGBM, verbose = F)
```

```r
# Plot
plot(modFit_gbm)
```




```r
# Predict on the test data
pred_gbm <- predict(modFit_gbm, newdata = testData)

# Accuracy of GBM
gbm_cm <- confusionMatrix(testData$classe, pred_gbm)
gbm_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1649   15    5    4    1
##          B   49 1062   27    1    0
##          C    0   39  972   15    0
##          D    0    1   40  920    3
##          E    6    4    7   16 1049
##
## Overall Statistics
##
##                Accuracy : 0.9604
##                  95% CI : (0.9551, 0.9652)
##     No Information Rate : 0.2895
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9499
```

```
##
##  Mcnemar's Test P-Value : 6.092e-10
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9677   0.9474   0.9248   0.9623   0.9962
## Specificity            0.9940   0.9838   0.9888   0.9911   0.9932
## Pos Pred Value         0.9851   0.9324   0.9474   0.9544   0.9695
## Neg Pred Value         0.9869   0.9876   0.9837   0.9927   0.9992
## Prevalence             0.2895   0.1905   0.1786   0.1624   0.1789
## Detection Rate         0.2802   0.1805   0.1652   0.1563   0.1782
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9809   0.9656   0.9568   0.9767   0.9947
```

```r
gbm_accuracy <- gbm_cm$overall[1]
gbm_accuracy
```

```
##  Accuracy
## 0.9604078
```

Accuracy of GBM is 0.9604078 => Out of sample error is 0.039

Conclusion

The accuracy of the 3 modeling methods used are as follows:

1. Decision Tree : 0.7368
2. Random Forest : 0.9963
3. GBM : 0.9839

Picking random forest model as it had the best accuracy and applying the best model to the validation set to get the predictions

```r
pred_validation_set <- predict(modFit_rf, newdata = validation_set)
pred_validation_set
```

```
##  [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
```