Practical Machine Learning Course Project

Cindy Neo

21-04-2024

Overview

In this project, we will use the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which the exercise was done. This is the "classe" variable in the training set. We will train 4 different models - **Decision Trees**, **Random Forest**, **Gradient Boosted Trees**, and **Support Vector Machine** - using k-folds cross validation on the training set. We will then predict using a validation set which is randomly selected from the training set, to obtain the **accuracy** and **out of sample error rate**. Based on these metrics, we will then decide on the final model to be used to predict the 20 cases on the test set.

Background

Using devices such as *Jawbone Up*, *Nike FuelBand*, and *Fitbit*, it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, my goal will be to 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. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Load Data

First, let's load the training data and test data.

```
## Check if the file exists
if (!file.exists("Dataset")) {
    dir.create("Dataset")
}

## Load the training data
trainUrl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
```

```r
download.file(trainUrl, destfile = "./Dataset/pml-training.csv")
trainData = read.csv("./Dataset/pml-training.csv")

## Load the test data
testUrl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testUrl, destfile = "./Dataset/pml-testing.csv")
testData = read.csv("./Dataset/pml-testing.csv")

dim(trainData)
```

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

```r
dim(testData)
```

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

We see that there are 160 variables in both training and test data, with 19622 observations in the training data and 20 observations in the test data.

Clean Data

Let's remove the unnecessary variables.

```r
## Removing variables which are mostly NA values
trainData <- trainData[, colMeans(is.na(trainData)) < 0.90]

## Removing metadata which are irrelevant to the outcome
trainData <- trainData[, -c(1: 7)]

## Removing near zero variance variables
library(caret)
nzv <- nearZeroVar(trainData)
trainData <- trainData[, -nzv]
dim(trainData)
```

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

Training and Validation Set

Now that we have removed the unnecessary variables, we can now split the training data into a training set and a validation set. The test data will be left alone, and used for prediction after we have selected our final model.

```r
set.seed(1234)
inTrain <- createDataPartition(y = trainData$classe,
                               p = 0.7, list = FALSE)

## Create training set
training <- trainData[inTrain, ]
```

```
## Create validation set
validation <- trainData[-inTrain, ]

dim(training)
```

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

```
dim(validation)
```

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

Create and Test the Models

We will train 4 different models - **Decision Trees**, **Random Forest**, **Gradient Boosted Trees**, and **Support Vector Machine** - using k-folds cross validation on the training set. We will then predict using the validation set to obtain the **accuracy** and **out of sample error rate**.

Set up control to use 3-fold cross validation on the training set.

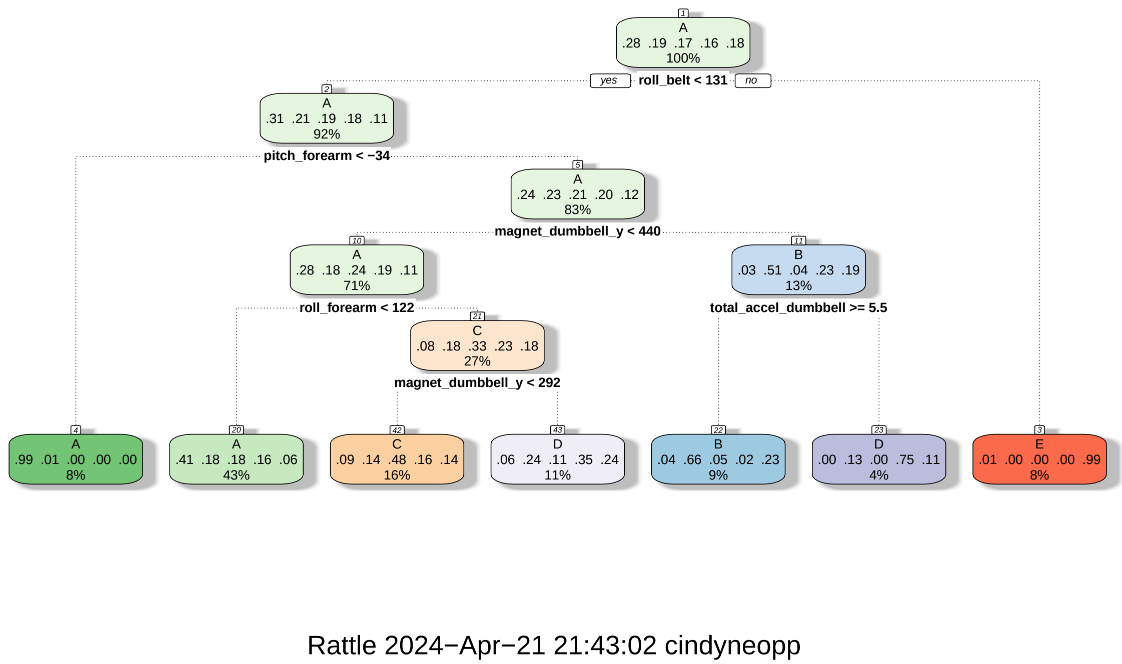
```
control <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
```

Decision Trees

Model:

```
DT <- train(classe ~ ., data = training, method = "rpart", trControl = control, tuneLength = 5)

library(rattle)
fancyRpartPlot(DT$finalModel)
```



Prediction:

```r
DT_pred <- predict(DT, newdata = validation)
DT_prediction <- confusionMatrix(DT_pred, factor(validation$classe))
DT_prediction
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1519  473  484  451  156
##          B   28  355   45   10  130
##          C   83  117  423  131  131
##          D   40  194   74  372  176
##          E    4    0    0    0  489
##
## Overall Statistics
##
##                Accuracy : 0.5366
##                  95% CI : (0.5238, 0.5494)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3957
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9074  0.31168  0.41228  0.38589  0.45194
## Specificity            0.6286  0.95512  0.90492  0.90165  0.99917
## Pos Pred Value         0.4927  0.62500  0.47797  0.43458  0.99189
## Neg Pred Value         0.9447  0.85255  0.87940  0.88228  0.89002
## Prevalence             0.2845  0.19354  0.17434  0.16381  0.18386
## Detection Rate         0.2581  0.06032  0.07188  0.06321  0.08309
## Detection Prevalence   0.5239  0.09652  0.15038  0.14545  0.08377
## Balanced Accuracy      0.7680  0.63340  0.65860  0.64377  0.72555
```

Random Forest

Model:

```
RF <- train(classe ~ ., data = training, method = "rf", trControl = control, tuneLength = 5)
```

Prediction:

```
RF_pred <- predict(RF, newdata = validation)
RF_prediction <- confusionMatrix(RF_pred, factor(validation$classe))
RF_prediction
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1132    8    0    0
##          C    0    3 1016    5    1
##          D    0    0    2  958    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9958
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9946
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9939   0.9903   0.9938   0.9991
## Specificity            0.9991   0.9981   0.9981   0.9996   0.9998
## Pos Pred Value         0.9976   0.9921   0.9912   0.9979   0.9991
```

```
## Neg Pred Value         0.9998   0.9985   0.9979   0.9988   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1924   0.1726   0.1628   0.1837
## Detection Prevalence   0.2850   0.1939   0.1742   0.1631   0.1839
## Balanced Accuracy      0.9992   0.9960   0.9942   0.9967   0.9994
```

Gradient Boosted Trees

Model:

```
GBM <- train(classe ~ ., data = training, method = "gbm", trControl = control, tuneLength = 5, verbose =
```

Prediction:

```
GBM_pred <- predict(GBM, newdata = validation)
GBM_prediction <- confusionMatrix(GBM_pred, factor(validation$classe))
GBM_prediction
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    5    0    0    0
##          B    1 1128   15    0    0
##          C    2    6 1007    8    4
##          D    0    0    4  953    1
##          E    0    0    0    3 1077
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.989, 0.9938)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9903   0.9815   0.9886   0.9954
## Specificity            0.9988   0.9966   0.9959   0.9990   0.9994
## Pos Pred Value         0.9970   0.9860   0.9805   0.9948   0.9972
## Neg Pred Value         0.9993   0.9977   0.9961   0.9978   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1917   0.1711   0.1619   0.1830
## Detection Prevalence   0.2848   0.1944   0.1745   0.1628   0.1835
## Balanced Accuracy      0.9985   0.9935   0.9887   0.9938   0.9974
```

Support Vector Machine

Model:

```
SVM <- train(classe ~ ., data = training, method = "svmLinear", trControl = control)
```

Prediction:

```
SVM_pred <- predict(SVM, newdata = validation)
SVM_prediction <- confusionMatrix(SVM_pred, factor(validation$classe))
SVM_prediction
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1537  154   79   69   50
##          B   29  806   90   46  152
##          C   40   81  797  114   69
##          D   61   22   32  697   50
##          E    7   76   28   38  761
##
## Overall Statistics
##
##                Accuracy : 0.7813
##                  95% CI : (0.7705, 0.7918)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.722
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9182   0.7076   0.7768   0.7230   0.7033
## Specificity            0.9164   0.9332   0.9374   0.9665   0.9690
## Pos Pred Value         0.8137   0.7177   0.7239   0.8086   0.8363
## Neg Pred Value         0.9657   0.9301   0.9521   0.9468   0.9355
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2612   0.1370   0.1354   0.1184   0.1293
## Detection Prevalence   0.3210   0.1908   0.1871   0.1465   0.1546
## Balanced Accuracy      0.9173   0.8204   0.8571   0.8447   0.8362
```

Results (Accuracy and out of sample error rate)

Let's look at the results (accuracy and out of sample error rate) of the 4 models.

```
## Create matrix with 2 columns and 4 rows
result = round(matrix(data = c(DT_prediction$overall[1], RF_prediction$overall[1], GBM_prediction$overal
```

```r
## Specify the column names and row names of matrix
colnames(result) = c('Accuracy', 'Out of Sample Error')
rownames(result) = c('Decision Trees', 'Random Forest', 'Gradient Boosted Trees', 'Support Vector Machin

## Assign and display the table
final = as.table(result)
final
```

```
##                        Accuracy Out of Sample Error
## Decision Trees           0.5366              0.4634
## Random Forest            0.9958              0.0042
## Gradient Boosted Trees   0.9917              0.0083
## Support Vector Machine   0.7813              0.2187
```

Based on the results, Random Forest is the best model, with an accuracy rate of 0.9958 and an out of sample error of 0.0042.

Prediction on the Test Data

We will now use the Random Forest model to predict the 20 cases on the test set.

```r
predict <- predict(RF, testData)
print(predict)
```

```
##  [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
```