PML_Final_Project

Harish

Wednesday, November 18, 2015

Executive Summary

For this project, we are given data from accelerometers on the belt, forearm, arm, and dumbell of 6 research study participants. Our training data consists of accelerometer data and a label identifying the quality of the activity the participant was doing. Our testing data consists of accelerometer data without the identifying label. Our goal is to predict the labels for the test set observations.

Below is the code I used when creating the model. I have also included a description of each step of the process.

Reading the dataset

Load the caret package, and read in the training and testing data:

```r
library(caret)
```

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

```r
ptrain <- read.csv("pml-training.csv")
ptest <- read.csv("pml-testing.csv")
```

Because I want to be able to estimate the out-of-sample error, I randomly split the full training data (ptrain) into a smaller training set (ptrain1) and a validation set (ptrain2):

```r
set.seed(10)
inTrain <- createDataPartition(y=ptrain$classe, p=0.7, list=F)
ptrain1 <- ptrain[inTrain, ]
ptrain2 <- ptrain[-inTrain, ]
```

I am now going to reduce the number of features by removing variables with nearly zero variance, variables that are almost always NA, and variables that don't make intuitive sense for prediction. Note that I decide which ones to remove by analyzing ptrain1, and perform the identical removals on ptrain2:

```r
nzv <- nearZeroVar(ptrain1)
ptrain1 <- ptrain1[, -nzv]
ptrain2 <- ptrain2[, -nzv]


mostlyNA <- sapply(ptrain1, function(x) mean(is.na(x))) > 0.95
ptrain1 <- ptrain1[, mostlyNA==F]
ptrain2 <- ptrain2[, mostlyNA==F]


ptrain1 <- ptrain1[, -(1:5)]
ptrain2 <- ptrain2[, -(1:5)]
```

Model Building

I decided to start with a Random Forest model, to see if it would have acceptable performance. I fit the model on ptrain1, and instruct the "train" function to use 3-fold cross-validation to select optimal tuning parameters for the model.

```
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=ptrain1, method="rf", trControl=fitControl)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
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: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2649    3    0    0 0.0033860045
## C    0    8 2388    0    0 0.0033388982
## D    0    0    7 2244    1 0.0035523979
## E    0    0    0    5 2520 0.0019801980
```

I see that it decided to use 500 trees and try 27 variables at each split.

Model Evaluation and Selection

Now, I use the fitted model to predict the label ("classe") in ptrain2, and show the confusion matrix to compare the predicted versus the actual labels:

```
preds <- predict(fit, newdata=ptrain2)
confusionMatrix(ptrain2$classe, preds)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    3 1132    3    1    0
##          C    0    4 1022    0    0
##          D    0    0    2  962    0
##          E    0    0    0    1 1081
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9965   0.9951   0.9979   1.0000
## Specificity            1.0000   0.9985   0.9992   0.9996   0.9998
## Pos Pred Value         1.0000   0.9939   0.9961   0.9979   0.9991
## Neg Pred Value         0.9993   0.9992   0.9990   0.9996   1.0000
## Prevalence             0.2850   0.1930   0.1745   0.1638   0.1837
## Detection Rate         0.2845   0.1924   0.1737   0.1635   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9991   0.9975   0.9972   0.9988   0.9999
```

The accuracy is 99.8%, thus my predicted accuracy for the out-of-sample error is 0.2%.

This is an excellent result, so rather than trying additional algorithms, I will use Random Forests to predict on the test set.

Retraining the selected model

Before predicting on the test set, it is important to train the model on the full training set (ptrain), rather than using a model trained on a reduced training set (ptrain1), in order to produce the most accurate predictions. Therefore, I now repeat everything I did above on ptrain and ptest:

```r
nzv <- nearZeroVar(ptrain)
ptrain <- ptrain[, -nzv]
ptest <- ptest[, -nzv]
mostlyNA <- sapply(ptrain, function(x) mean(is.na(x))) > 0.95
ptrain <- ptrain[, mostlyNA==F]
ptest <- ptest[, mostlyNA==F]
ptrain <- ptrain[, -(1:5)]
ptest <- ptest[, -(1:5)]
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=ptrain, method="rf", trControl=fitControl)
```

Making Test Set Predictions

```r
preds <- predict(fit, newdata=ptest)
preds <- as.character(preds)

pml_write_files <- function(x) {
    n <- length(x)
    for(i in 1:n) {
        filename <- paste0("problem_id_", i, ".txt")
        write.table(x[i], file=filename, quote=F, row.names=F, col.names=F)
    }
}

pml_write_files(preds)
```

Conclusion

In this assignment, we accurately predicted the classification of 20 observations using a Random Forest algorithm trained on a subset of data using less than 20% of the covariates.

The accuracy obtained (accuracy = 99.8%, and out-of-sample error = 0.2%) is obviously highly suspicious as it is never the case that machine learning algorithms are that accurate, and a mere 85% if often a good accuracy result.