Practical Machine Learning - Course Project

Rodrigo

21/08/2019

Introduction

Data from the Human Activity Recognition (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) was used for this machine learning assignment. A model was derived from a training set, which was split into two subsets, used for training and for cross validation. Then, the model was applied to test a set 20 samples and predict the outcome variable.

The code used when creating the model is available belw, estimating the out-of-sample error, and making predictions. A description of each step of the process was also included.

Data Preparation

The caret package was loaded, and read in the training and testing data:

```
library(caret)
```

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

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

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

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

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

On this step, the number of features was reduced by removing variables with nearly zero variance, variables that were almost always NA, and variables that did not make intuitive sense for prediction. Note that the decision about which ones would be removed was made by analyzing ptrain1, and performing the identical removals on ptrain2:

```
# remove variables with nearly zero variance
nzv <- nearZeroVar(ptrain1)
ptrain1 <- ptrain1[, -nzv]
ptrain2 <- ptrain2[, -nzv]
# remove variables that are almost always NA
mostlyNA <- sapply(ptrain1, function(x) mean(is.na(x))) > 0.95
ptrain1 <- ptrain1[, mostlyNA==F]
ptrain2 <- ptrain2[, mostlyNA==F]
# remove variables that don't make intuitive sense for prediction (X, user_name, raw_timestam
p_part_1, raw_timestamp_part_2, cvtd_timestamp), which happen to be the first five variables
ptrain1 <- ptrain1[, -(1:5)]
ptrain2 <- ptrain2[, -(1:5)]
```

Model Definition

It was decided to start with a Random Forest model, to see what would be its performance. The model was fit on ptrain1, and instructed the “train” function to use 3-fold cross-validation to select optimal tuning parameters for the model.

```
# instruct train to use 3-fold CV to select optimal tuning parameters
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
# fit model on ptrain1
fit <- train(classe ~ ., data=ptrain1, method="rf", trControl=fitControl)
# print final model to see tuning parameters it chose
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: 0.24%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    7 2648    3    0    0 0.0037622272
## C    0    4 2392    0    0 0.0016694491
## D    0    0    9 2242    1 0.0044404973
## E    0    1    1    5 2518 0.0027722772
```

It was decided to use 500 trees and 27 variables were tried at each split.

Model Evaluation and Selection

The fitted model was used to predict the label (“classe”) in ptrain2, and to show the confusion matrix to compare the predicted versus the actual labels:

```
# use model to predict classe in validation set (ptrain2)
preds <- predict(fit, newdata=ptrain2)
# show confusion matrix to get estimate of out-of-sample error
confusionMatrix(ptrain2$classe, preds)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    4 1134    1    0    0
##          C    0    1 1025    0    0
##          D    0    0    7  957    0
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9976
##                  95% CI : (0.996, 0.9987)
##     No Information Rate : 0.2851
##     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            0.9976   0.9991   0.9923   0.9990   1.0000
## Specificity            1.0000   0.9989   0.9998   0.9986   0.9998
## Pos Pred Value         1.0000   0.9956   0.9990   0.9927   0.9991
## Neg Pred Value         0.9991   0.9998   0.9984   0.9998   1.0000
## Prevalence             0.2851   0.1929   0.1755   0.1628   0.1837
## Detection Rate         0.2845   0.1927   0.1742   0.1626   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9988   0.9990   0.9960   0.9988   0.9999
```

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

That may be considered a very good result, and, thus, the Random Forests Model was selected to predict on the test set, rather than trying additional algorithms.

Selected Model Re-training

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, everything that was performed previously on ptrain and ptest was repeated on this step:

```
# remove variables with nearly zero variance
nzv <- nearZeroVar(ptrain)
ptrain <- ptrain[, -nzv]
ptest <- ptest[, -nzv]
# remove variables that are almost always NA
mostlyNA <- sapply(ptrain, function(x) mean(is.na(x))) > 0.95
ptrain <- ptrain[, mostlyNA==F]
ptest <- ptest[, mostlyNA==F]
# remove variables that don't make intuitive sense for prediction (X, user_name, raw_timestam
p_part_1, raw_timestamp_part_2, cvtd_timestamp), which happen to be the first five variables
ptrain <- ptrain[, -(1:5)]
ptest <- ptest[, -(1:5)]
# re-fit model using full training set (ptrain)
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=ptrain, method="rf", trControl=fitControl)
```

Making Test Set Predictions

On this step, the model fit was used on ptrain to predict the label for the observations in ptest, and write those predictions to individual files:

```
# predict on test set
preds <- predict(fit, newdata=ptest)
# convert predictions to character vector
preds <- as.character(preds)
# create function to write predictions to files
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)
    }
}
# create prediction files to submit
pml_write_files(preds)
```

Conclusion

The dataset was downloaded, cleaned and split into training, and testing sets. A random forest model was chosen, since there was a large amount of possible predictors. The model had a 99.8% accuracy, out-of-sample error estimated in 0.2%, and was able to predict correctly the class for 20 test data samples.