Practical Machine Learning- Final Project

Jeffrey Uslan

Thursday, May 21, 2015

The Project used the following packages:

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(AppliedPredictiveModeling)
library(caret)
```

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

```
library(Hmisc)
```

```
## Loading required package: grid
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(Rcpp)
library(ggplot2)
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
```

```
##
## The following object is masked from 'package:Hmisc':
##
##     combine
```

First, let's read in our training and testing sets.

```
data=read.csv("./pml-training.csv")

final_test=read.csv("./pml-testing.csv")
```

Remove index variables

```
data=data[,-1]
final_test=final_test[,-1]
```

Identify which variables are viable as covariates. The code below identifies variables which are missing in the test set.

```
NA_cols=apply(final_test,2,function(x){sum(is.na(x))==length(x)})
NA_vars=names(final_test)[NA_cols]
```

Remove the missing variables from the training and test sets.

```
data=data[,!(names(data) %in% NA_vars)]
final_test=final_test[,!(names(final_test) %in% NA_vars)]
```

We can use the "na.approx" command from the "zoo" package to impute numeric variables.

```
n_nums=sapply(data,is.numeric)
data[,n_nums]=na.approx(data[,n_nums])

n_nums=sapply(final_test,is.numeric)
final_test[,n_nums]=na.approx(final_test[,n_nums])
```

Next we identify variables in the training set with near zero variance. These variables would enough value to our predictions to justify the computation power required to process them. We also omit the timestamp variable at this time.

```
NZV=nearZeroVar(data, saveMetrics=TRUE)
poor_var=which(NZV$nzv)
rm_vars=names(data)[c(which(names(data)=="cvtd_timestamp"),poor_var)]
```

Remove variables identified in the previous step from the training and test sets.

```
data=data[,!(names(data) %in% rm_vars)]
final_test=final_test[,!(names(final_test) %in% rm_vars)]
```

Keep only numeric variables and the dependent factor "classe" for the training set. Remove incomplete observations from the training set.

```
classe_ind=which(names(data)=="classe")
n_nums=which(sapply(data,is.numeric))
data=na.omit(data[,c(n_nums,classe_ind)])
```

Keep only numeric variables for the testing set.

```
n_nums=which(sapply(final_test,is.numeric))
final_test=final_test[,c(n_nums)]
```

Split the the training set for cross validation.

```
inTrain = createDataPartition(y=data$classe, p = .6)[[1]]
training = data[ inTrain,]
testing = data[-inTrain,]
```

Fit a random forest model on the sub-training set.

```
modFit <- randomForest(classe ~. , data=training)
# modFit<-train(classe~.,data=training,method="rf",prox=TRUE)
print(modFit)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.17%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3348    0    0    0    0 0.0000000000
## B    3 2276    0    0    0 0.0013163668
## C    0    4 2048    2    0 0.0029211295
## D    0    0    8 1920    2 0.0051813472
## E    0    0    0    1 2164 0.0004618938
```

The out-of-sample error rate estimate is 0.17%

Check the in-sample error rate of the model on the sub-training set

```
pred=predict(modFit,newdata= training, type = "class")
confusionMatrix(training$classe, pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
```

```
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Predict the out-of-sample error rate of the model on the sub-testing set

```
pred=predict(modFit, testing, type = "class")
confusionMatrix(testing$classe,pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    1    0    0    0
##          B    0 1517    1    0    0
##          C    0    2 1364    2    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9992
##                  95% CI : (0.9983, 0.9997)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.999
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            1.0000   0.9980   0.9993   0.9984   1.0000
## Specificity            0.9998   0.9998   0.9994   1.0000   1.0000
## Pos Pred Value         0.9996   0.9993   0.9971   1.0000   1.0000
## Neg Pred Value         1.0000   0.9995   0.9998   0.9997   1.0000
## Prevalence             0.2843   0.1937   0.1740   0.1642   0.1838
## Detection Rate         0.2843   0.1933   0.1738   0.1639   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9999   0.9989   0.9993   0.9992   1.0000
```

According to our cross-validation the out-of-sample error rate is 0.0008%

Apply the model to final test set.

```
pred=predict(modFit,newdata= final_test, type = "class")

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=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(pred)
```