Practical Machine Learning

Analysis

Read in training set & Data cleaning

First we read in the dataset. When inspectiting via summary(origData), not shown due to volume of output, we see that there are many variables with a 98% NA values. For a more meaningful analysis we remove those variables, as well as username, timestamp and 'new_window'. This also takes care of any near zero variance variables. We use the remaining 53 variables for analysis plus the outcome variable.

```
library(caret)
```

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

```
origData <- read.csv("pml-training.csv", na.strings = c("NA", "") )
dim(origData)
```

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

```
cleanData <- origData[ , -grep("^X$|user_name|timestamp|new_window", names(origData))
]

NAVars <- apply( cleanData, 2, function(x) sum(is.na(x))  )
table(NAVars)
```

```
## NAVars
##     0 19216
##    54   100
```

```
cleanData <- cleanData[ , NAVars == 0 ]
dim(cleanData)
```

```
## [1] 19622    54
```

```
nearZeroVar(cleanData, saveMetrics = F)
```

```
## integer(0)
```

Partition into training and testsets

Next we partition the data into a training (70% of the data) and testset (30% of the data), to be used for crossvalidation.

```
set.seed(123)
inTrain <- createDataPartition( cleanData$classe, p = .7, list = F )
training <- cleanData[ inTrain, ]
testing <- cleanData[ -inTrain, ]
dim(training)
```

```
## [1] 13737    54
```

```
dim(testing)
```

```
## [1] 5885   54
```

Model Fit

```
modFit <- train( classe ~ ., data = training, method = "gbm", verbose = F)
#modFit

modFitRf <- train( classe ~ ., data = training, method = "rf")


confusionMatrix(training$classe, predict( modFit, training ))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3905    1    0    0    0
##          B   18 2616   24    0    0
##          C    0   11 2381    3    1
##          D    2    3   22 2224    1
##          E    0    3    2   12 2508
##
## Overall Statistics
##
##                Accuracy : 0.9925
##                  95% CI : (0.9909, 0.9939)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9905
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9949   0.9932   0.9802   0.9933   0.9992
## Specificity            0.9999   0.9962   0.9987   0.9976   0.9985
## Pos Pred Value         0.9997   0.9842   0.9937   0.9876   0.9933
## Neg Pred Value         0.9980   0.9984   0.9958   0.9987   0.9998
## Prevalence             0.2857   0.1917   0.1768   0.1630   0.1827
## Detection Rate         0.2843   0.1904   0.1733   0.1619   0.1826
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9974   0.9947   0.9895   0.9954   0.9988
```

```
confusionMatrix(training$classe, predict( modFitRf, training ))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## 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
```

First we try a stochastic gradient boosting method (gbm), to utilize boosting, which should deliver high accuracy. Results were satisfactory, with relatively good accuracy in the training set (99.25%). To see if we can get better results we run a second model, in this case a random forrest model (rf). Accuracy was even better, with 100% correct classification in the training set. The perfect prediction could indicate overfitting, however.

The error is expected to be higher out of sample, due to overfitting the data in the training set. For the GBM model we would expect an error greater than .75%. For the RF model we would expect an error somewhat greater than 0.

When crossvalidating on the testing set it turns out that the RF still provides the better accuracy (99.81% vs 98.56%). Both models provide very good predictions, but due to the near perfect accuracy we choose the RF model as the final model.

```
confusionMatrix(testing$classe, predict( modFit, testing ))
```

```
## Loading required package: gbm
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    0    0    1    0
##          B   13 1110   14    2    0
##          C    0   13 1010    3    0
##          D    0    3   18  943    0
##          E    1    3    4   10 1064
##
## Overall Statistics
##
##                Accuracy : 0.9856
##                  95% CI : (0.9822, 0.9884)
##     No Information Rate : 0.2867
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9817
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9917   0.9832   0.9656   0.9833   1.0000
## Specificity            0.9998   0.9939   0.9967   0.9957   0.9963
## Pos Pred Value         0.9994   0.9745   0.9844   0.9782   0.9834
## Neg Pred Value         0.9967   0.9960   0.9926   0.9967   1.0000
## Prevalence             0.2867   0.1918   0.1777   0.1630   0.1808
## Detection Rate         0.2843   0.1886   0.1716   0.1602   0.1808
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9957   0.9885   0.9811   0.9895   0.9981
```

```
confusionMatrix(testing$classe, predict( modFitRf, testing ))
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    4 1135    0    0    0
##          C    0    4 1022    0    0
##          D    0    0    2  961    1
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9965   0.9980   1.0000   0.9991
## Specificity            1.0000   0.9992   0.9992   0.9994   1.0000
## Pos Pred Value         1.0000   0.9965   0.9961   0.9969   1.0000
## Neg Pred Value         0.9991   0.9992   0.9996   1.0000   0.9998
## Prevalence             0.2851   0.1935   0.1740   0.1633   0.1840
## Detection Rate         0.2845   0.1929   0.1737   0.1633   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9988   0.9978   0.9986   0.9997   0.9995
```

Testing Dataset

First we read in the dataset and apply the same data cleaning functions as before.

```
origTestData <- read.csv("pml-testing.csv", na.strings = c("NA", "") )
dim(origTestData)
```

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

```
cleanTestData <- origTestData[ , -grep("^X$|user_name|timestamp|new_window", names(ori
gTestData)) ]

cleanTestData <- cleanTestData[ , NAVars == 0 ]
dim(cleanTestData)
```

```
## [1] 20 54
```

Then we predict the classes, using the random forrest model, for each of the test cases, and create text files via the supplied function. The model behaves perfectly, with 100% accuracy of predictions.

```
answers <- predict(modFitRf, cleanTestData)

print(answers)
```

```
##  [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
```

```
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(answers)
```