Practical Machine Learning

Course Project: CP_template

Introduction

This document presents the results of the Course Project for the Coursera course: Practical Machine Learning. This assessment required the student to explore personal activity data in order to predict the manner in which they did the exercise.

Data

This assignment makes use of data collected from accelerometers on the belt, forearm, arm, and dumbell of six participants. More information is available from the website here: raw data (see the section on the Weight Lifting Exercise Dataset).

- Training Dataset: training data
- Testing Dataset: test data

1. Loading Packages/ Data

```
for (package in c('caret', 'randomForest', 'rpart', 'rpart.plot')) {
  if (!require(package, character.only = TRUE, quietly = FALSE)) {
    install.packages(package)
    library(package, character.only = TRUE)
  }
}

val_dfpath <- paste(getwd(), "data", sep = "/")
val_dfchk <- c("pmltraining.raw", "pmltesting.raw")
val_dfname <- c("pml-training.csv", "pml-testing.csv")
val_dfdlink <- "https://d396qusza40orc.cloudfront.net/predmachlearn"
```

2. Pre-process the Data

Read data

```
data_pmltraining.raw <- read.csv(paste(val_dfpath, val_dfname[1], sep = "/"), na.strings = c("NA", "#DI
data_pmltesting.raw <- read.csv(paste(val_dfpath, val_dfname[2], sep = "/"), na.strings = c("NA", "#DIV
```

Check the original data

```
dim(data_pmltraining.raw)
```

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

```
## str(data_pmltraining.raw)
## summary(data_pmltraining.raw)
```

Remove near zero covariates

```
data_pmltraining.nzv <- nearZeroVar(data_pmltraining.raw, saveMetrics = TRUE)
data_pmltraining <- data_pmltraining.raw[, !data_pmltraining.nzv$nzv]
remove(data_pmltraining.nzv)
```

Remove first column of the training dataset

```
data_pmltraining <- data_pmltraining[, -1]
```

Remove variables with more than 60% NA values

```
val_pmltraining.nav <- which((colSums(!is.na(data_pmltraining)) >= 0.6 * nrow(data_pmltraining)))
data_pmltraining <- data_pmltraining[, val_pmltraining.nav]
remove(val_pmltraining.nav)
```

Split into training and testing datasets

```
data_pmltraining.inTrain <- createDataPartition(data_pmltraining$classe, p = 0.6, list = FALSE)
data_pmltraining.train <- data_pmltraining[data_pmltraining.inTrain, ]
data_pmltraining.test <- data_pmltraining[-data_pmltraining.inTrain, ]
remove(data_pmltraining.inTrain)
```

Transform training and testing datasets

```
val_pmltraining.allcol <- colnames(data_pmltraining.train)
val_pmltraining.datacol <- colnames(data_pmltraining.train[, -ncol(data_pmltraining.train)])
data_pmltraining.test <- data_pmltraining.test[val_pmltraining.allcol]
data_pmltesting <- data_pmltesting.raw[val_pmltraining.datacol]
remove(val_pmltraining.allcol, val_pmltraining.datacol)
```

Coerce data classes between training and final testing datasets

```
data_pmltesting <- rbind(data_pmltraining.train[2, -ncol(data_pmltraining.train)] , data_pmltesting)
data_pmltesting <- data_pmltesting[-1, ]
```

Check the processed data

```
dim(data_pmltraining.train)
```

```
## [1] 11776    58
```

```
## str(data_pmltraining.train)
## summary(data_pmltraining.train)
```

```
dim(data_pmltraining.test)
```

```
## [1] 7846   58
```

```
## str(data_pmltraining.test)
## summary(data_pmltraining.test)
```

```
dim(data_pmltesting)
```

```
## [1] 20 57
```

```
## str(data_pmltesting)
## summary(data_pmltesting)
```

3. Prediction Modelling

```
set.seed(12345)
val_dtmodel <- rpart(classe ~ ., data = data_pmltraining.train, method = "class")
val_dtmodel.predict <- predict(val_dtmodel, data_pmltraining.test, type = "class")
val_dtcm <- confusionMatrix(val_dtmodel.predict, data_pmltraining.test$classe)
val_dtcm
```

Decision tree prediction

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2159   63    7    3    0
##          B   55 1264   79   59    0
##          C   18  181 1260  194   59
##          D    0   10   11  836   86
##          E    0    0   11  194 1297
##
## Overall Statistics
##
##                Accuracy : 0.8687
##                  95% CI : (0.861, 0.8761)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8339
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9673   0.8327   0.9211   0.6501   0.8994
```

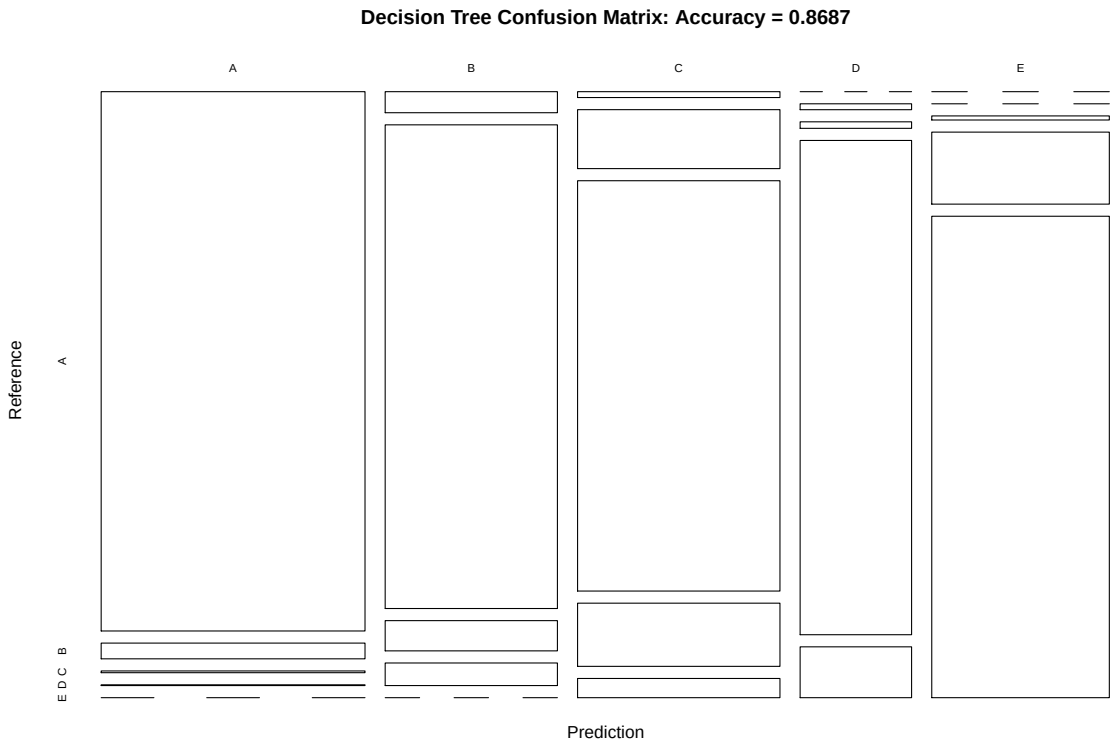
```
## Specificity            0.9870   0.9695   0.9302   0.9837   0.9680
## Pos Pred Value         0.9673   0.8675   0.7360   0.8865   0.8635
## Neg Pred Value         0.9870   0.9602   0.9824   0.9348   0.9771
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2752   0.1611   0.1606   0.1066   0.1653
## Detection Prevalence   0.2845   0.1857   0.2182   0.1202   0.1914
## Balanced Accuracy      0.9771   0.9011   0.9256   0.8169   0.9337
```

Decision tree prediction has a reported accuracy against the training dataset:

```
round(val_dtcm$overall['Accuracy'], 4)
```

```
## Accuracy
##   0.8687
```

```
plot(val_dtcm$table, col = val_dtcm$byClass, main = paste("Decision Tree Confusion Matrix: Accuracy =",
```




```
set.seed(12345)
val_rfmodel <- randomForest(classe ~ ., data = data_pmltraining.train)
val_rfmodel.predict <- predict(val_rfmodel, data_pmltraining.test, type = "class")
val_rfcm <- confusionMatrix(val_rfmodel.predict, data_pmltraining.test$classe)
val_rfcm
```

Random forest prediction

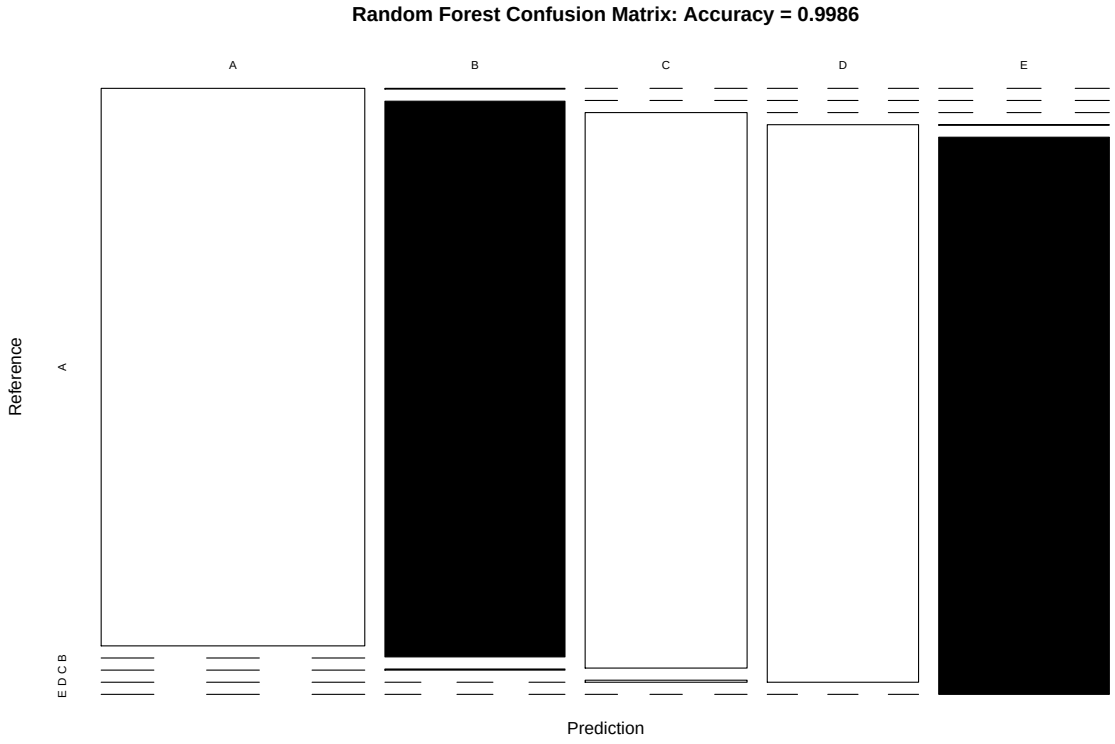
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    0    0    0    0
##          B    2 1518    3    0    0
##          C    0    0 1365    5    0
##          D    0    0    0 1280    0
##          E    0    0    0    1 1442
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   1.0000   0.9978   0.9953   1.0000
## Specificity            1.0000   0.9992   0.9992   1.0000   0.9998
## Pos Pred Value         1.0000   0.9967   0.9964   1.0000   0.9993
## Neg Pred Value         0.9996   1.0000   0.9995   0.9991   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1935   0.1740   0.1631   0.1838
## Detection Prevalence   0.2842   0.1941   0.1746   0.1631   0.1839
## Balanced Accuracy      0.9996   0.9996   0.9985   0.9977   0.9999
```

Random forest prediction has a reported accuracy against the training dataset:

```
round(val_rfcm$overall['Accuracy'], 4)
```

```
## Accuracy
##   0.9986
```

```
plot(val_rfcm$table, col = val_rfcm$byClass, main = paste("Random Forest Confusion Matrix: Accuracy =",
```




```
set.seed(12345)
val_fitControl <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
val_gbmmodel <- train(classe ~ ., data = data_pmltraining.train, method = "gbm", trControl = val_fitCon
val_gbmmodel.predict <- predict(val_gbmmodel, newdata = data_pmltraining.test)
val_gbmcm <- confusionMatrix(val_gbmmodel.predict, data_pmltraining.test$classe)
val_gbmcm
```

Generalized boosted regression prediction

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1510    2    0    0
##          C    0    3 1360    3    0
##          D    0    4    6 1282    1
##          E    0    0    0    1 1441
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9959, 0.9983)
##     No Information Rate : 0.2845
```

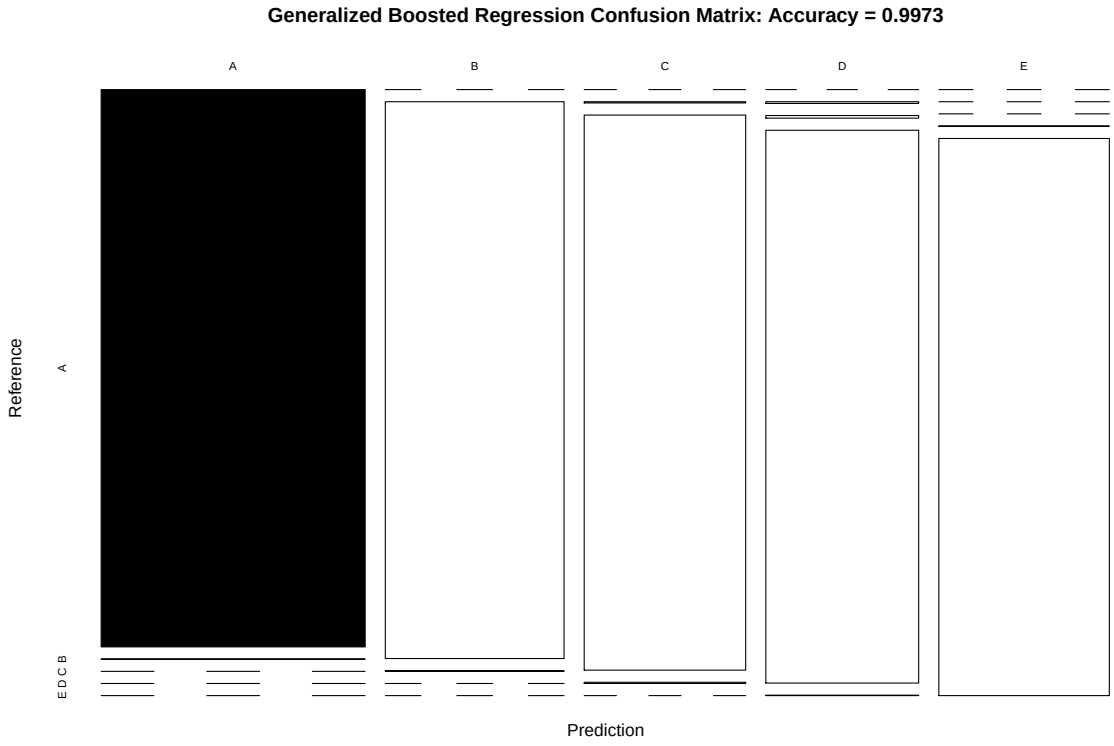
```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9942   0.9969   0.9993
## Specificity            0.9998   0.9997   0.9991   0.9983   0.9998
## Pos Pred Value         0.9996   0.9987   0.9956   0.9915   0.9993
## Neg Pred Value         1.0000   0.9987   0.9988   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1925   0.1733   0.1634   0.1837
## Detection Prevalence   0.2846   0.1927   0.1741   0.1648   0.1838
## Balanced Accuracy      0.9999   0.9972   0.9966   0.9976   0.9996
```

Generalized boosted regression prediction has a reported accuracy against the training dataset:

```
round(val_gbmcm$overall['Accuracy'], 4)
```

```
## Accuracy
##   0.9973
```

```
plot(val_gbmcm$table, col = val_gbmcm$byClass, main = paste("Generalized Boosted Regression Confusion Ma
```



4. Model Selection

Random forest prediction model is selected due to its superior accuracy. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set:

```
val_ooserror <- 1 - round(val_rfcm$overall['Accuracy'], 4)
val_ooserror
```

```
## Accuracy
##   0.0014
```

With an accuracy above 99% on the cross-validation data, it is expected that few or none of the test samples will be missclassified.

```
val_rfmodel.final <- predict(val_rfmodel, data_pmltesting, type = "class")
val_rfmodel.final
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

5. Coursera Submission

```
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(val_rfmodel.final)
```