Assignment

##Loading in Packages and Data

```
library(data.table)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.6.3
```

```
## corrplot 0.84 loaded
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.6.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
testing  <- read.csv("C:/Users/soomen/OneDrive - Capgemini/Desktop/Ontwikkeling/Coursera R/MachineLearn
training <- read.csv("C:/Users/soomen/OneDrive - Capgemini/Desktop/Ontwikkeling/Coursera R/MachineLearn
```

##Creating training and testing datasets

```
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
traindata <- training[inTrain,]
testdata <- training[-inTrain,]
```

##Data Cleaning

If more than 75% of the rows are NA, remove the row

```
traindata <- traindata[, colSums(is.na(traindata)) < nrow(traindata) * 0.75]
testdata <- testdata[, colSums(is.na(testdata)) < nrow(testdata) * 0.75]
```

Check for NearZeroVariance columns and delete them

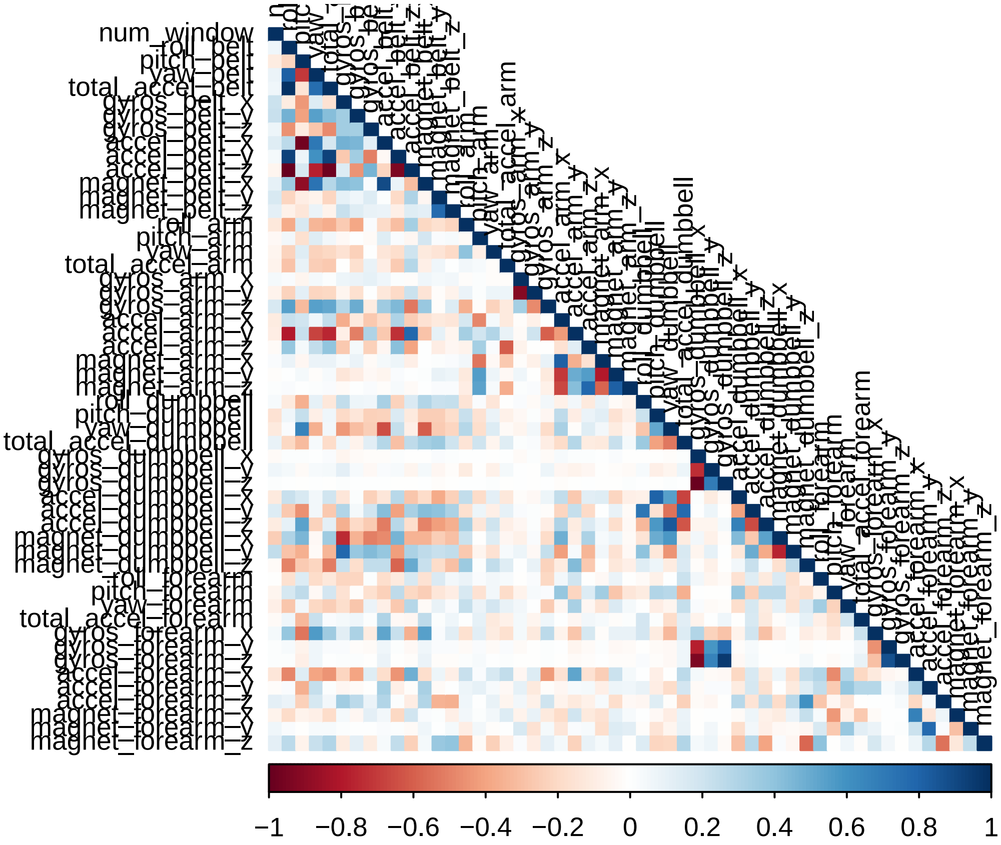
```
NearZeroVariance <- nearZeroVar(traindata)
traindata <- traindata[ ,-NearZeroVariance]
testdata <- testdata[ ,-NearZeroVariance]
```

Delete first columns since they contain unusable formats

```
traindata <- traindata[,-c(1:5)]
testdata <- testdata[,-c(1:5)]
```

Classe is column number 54 so it is left out of the correlation matrix

```
corrMat <- cor(traindata[,-54])
corrplot(corrMat, method = "color", type = "lower", tl.cex = 0.8, tl.col = rgb(0,0,0))
```



##Random Forest

```
modelrf <- train(classe~., method= "rf", data=traindata, trControl = trainControl(method="cv",number = 
modelrf$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: 0.28%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    8 2645    4    1    0 0.0048908954
## C    0    4 2391    1    0 0.0020868114
## D    0    0   10 2241    1 0.0048845471
## E    0    1    0    7 2517 0.0031683168
```

```
predictrf <- predict(modelrf, newdata=testdata)
confusionrf <- confusionMatrix(predictrf, testdata$classe)
confusionrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1136    1    0    0
##          C    0    1 1025   11    0
##          D    0    0    0  953    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9990   0.9886   0.9963
## Specificity            0.9995   0.9998   0.9975   0.9992   1.0000
## Pos Pred Value         0.9988   0.9991   0.9884   0.9958   1.0000
## Neg Pred Value         1.0000   0.9994   0.9998   0.9978   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1930   0.1742   0.1619   0.1832
## Detection Prevalence   0.2848   0.1932   0.1762   0.1626   0.1832
## Balanced Accuracy      0.9998   0.9986   0.9983   0.9939   0.9982
```

##Descision Tree

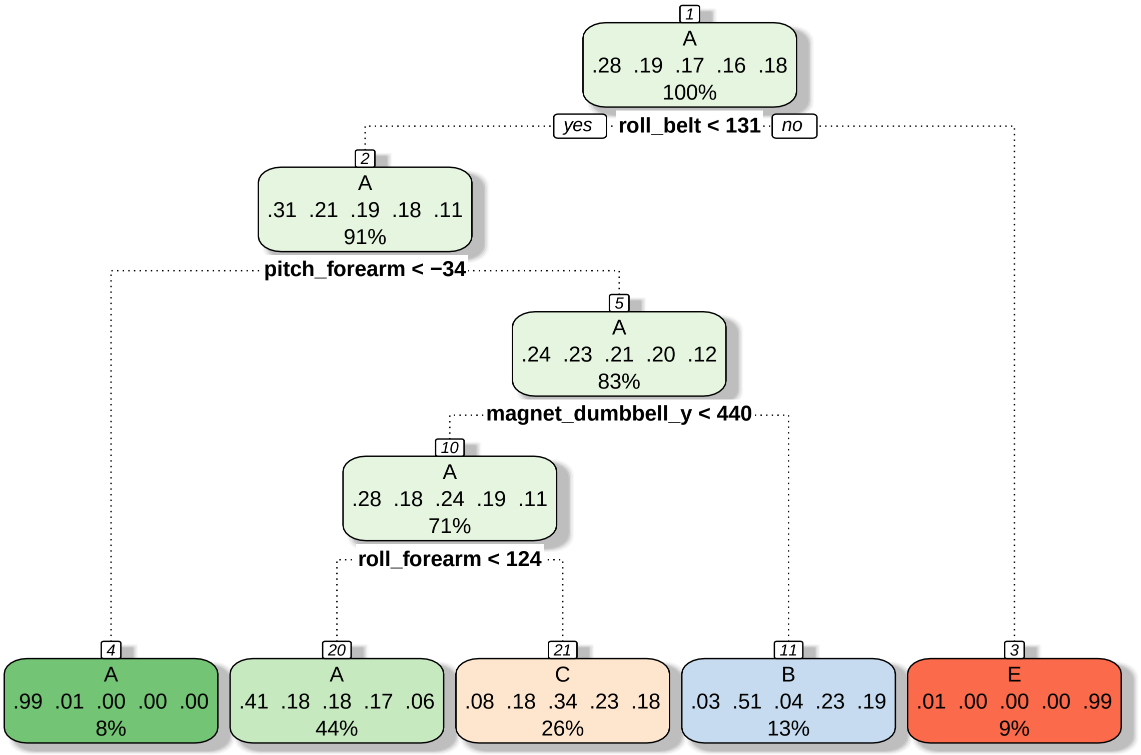
```
modeldt <- train(classe~., method= "rpart", data = traindata)
modeldt$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12563 8668 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1097    6 A (0.99 0.0055 0 0 0) *
##      5) pitch_forearm>=-33.95 11466 8662 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 9686 6938 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 6049 3581 A (0.41 0.18 0.18 0.17 0.059) *
##         21) roll_forearm>=123.5 3637 2416 C (0.077 0.18 0.34 0.23 0.18) *
##       11) magnet_dumbbell_y>=439.5 1780  878 B (0.031 0.51 0.041 0.23 0.19) *
##    3) roll_belt>=130.5 1174   11 E (0.0094 0 0 0 0.99) *
```

```
predictdt <- predict(modeldt, newdata=testdata)
confusiondt <- confusionMatrix(predictdt, testdata$classe)
confusiondt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1521  470  485  432  167
##          B   25  384   35  158  147
##          C  125  285  506  374  300
##          D    0    0    0    0    0
##          E    3    0    0    0  468
##
## Overall Statistics
##
##                Accuracy : 0.4892
##                  95% CI : (0.4764, 0.5021)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3322
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9086  0.33714  0.49318   0.0000  0.43253
## Specificity            0.6310  0.92309  0.77691   1.0000  0.99938
## Pos Pred Value         0.4946  0.51268  0.31824      NaN  0.99363
## Neg Pred Value         0.9456  0.85300  0.87893   0.8362  0.88659
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2585  0.06525  0.08598   0.0000  0.07952
## Detection Prevalence   0.5225  0.12727  0.27018   0.0000  0.08003
## Balanced Accuracy      0.7698  0.63012  0.63504   0.5000  0.71595
```

```
fancyRpartPlot(modeldt$finalModel)
```



Rattle 2020−mei−08 16:34:53 SOOMEN

##Boosted model

```
controls <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelbm <- train(classe ~ ., data = traindata, trControl = controls, method = "gbm", verbose = FALSE)
modelbm$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predictbm <- predict(modelbm, newdata=testdata)
confbm <- confusionMatrix(predictbm, testdata$classe)
confbm
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1671    7    0    0    0
##          B    3 1123    8    0    9
##          C    0    9 1010   15    2
##          D    0    0    7  948   10
##          E    0    0    1    1 1061
##
## Overall Statistics
##
##                Accuracy : 0.9878
##                  95% CI : (0.9846, 0.9904)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9845
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9860   0.9844   0.9834   0.9806
## Specificity            0.9983   0.9958   0.9946   0.9965   0.9996
## Pos Pred Value         0.9958   0.9825   0.9749   0.9824   0.9981
## Neg Pred Value         0.9993   0.9966   0.9967   0.9967   0.9956
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1908   0.1716   0.1611   0.1803
## Detection Prevalence   0.2851   0.1942   0.1760   0.1640   0.1806
## Balanced Accuracy      0.9983   0.9909   0.9895   0.9900   0.9901
```

##Final Result with RandomForest Random forest has the highest accuracy as can be seen above, hence it is used for the final prediction.

```
predictresult <- predict(modelrf, newdata=testing)
predictresult
```

```
##  [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
```