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

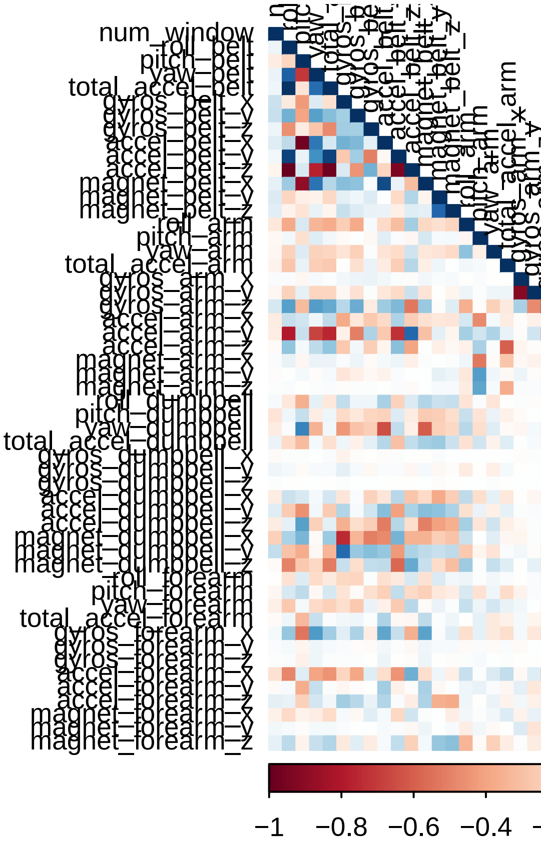
```
traindata <- traindata[, colSums(is.na(traindata)) < nrow(traindata) * 0.75]
testdata <- testdata[, colSums(is.na(testdata)) < nrow(testdata) * 0.75]
```

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

##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: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2649    2    1    0 0.0033860045
## C    0    5 2391    0    0 0.0020868114
## D    0    0   12 2240    0 0.0053285968
## E    0    0    0    4 2521 0.0015841584
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    2    0    0
##          C    0    0 1023    2    0
##          D    0    0    1  962    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9973, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9983
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9971   0.9979   0.9982
## Specificity            0.9998   0.9996   0.9996   0.9994   1.0000
## Pos Pred Value         0.9994   0.9982   0.9980   0.9969   1.0000
## Neg Pred Value         1.0000   0.9998   0.9994   0.9996   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1738   0.1635   0.1835
## Detection Prevalence   0.2846   0.1937   0.1742   0.1640   0.1835
## Balanced Accuracy      0.9999   0.9994   0.9983   0.9987   0.9991
```

##Descision Tree

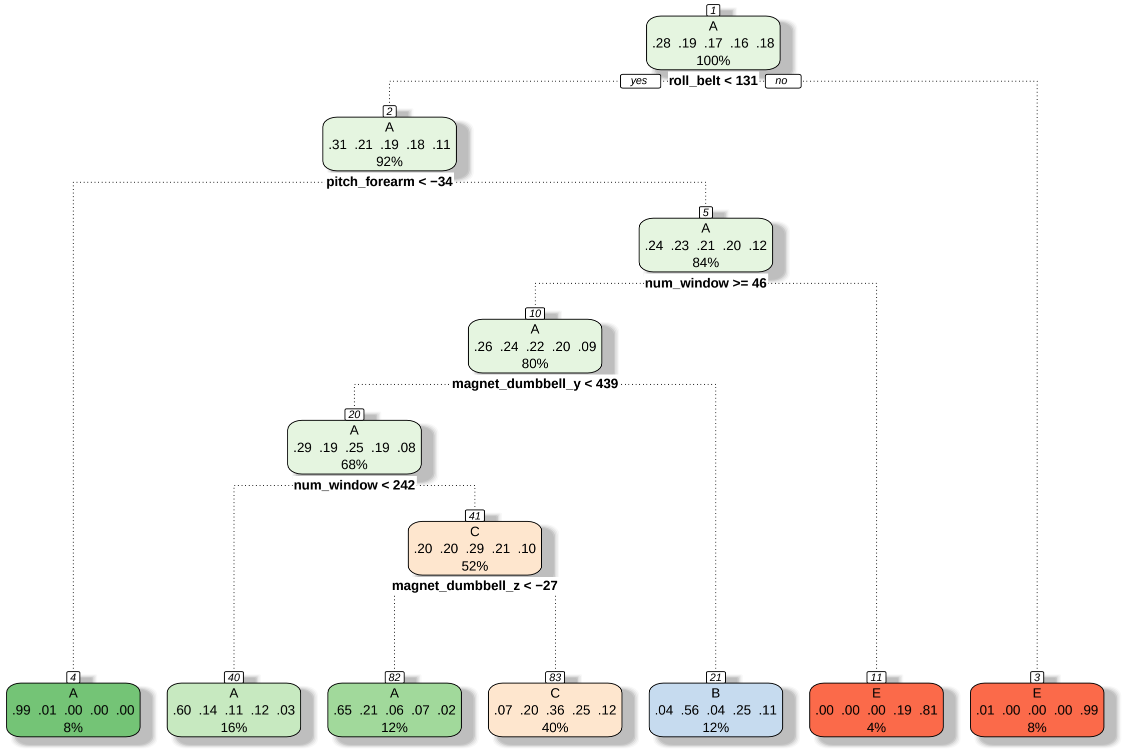
```
modeldt <- train(classe~., method= "rpart", data = traindata)
modeldt$finalModel
```

```
## 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 12581 8685 A (0.31 0.21 0.19 0.18 0.11)  
##      4) pitch_forearm< -33.95 1089    8 A (0.99 0.0073 0 0 0) *
##      5) pitch_forearm>=-33.95 11492 8677 A (0.24 0.23 0.21 0.2 0.12)  
##       10) num_window>=45.5 10979 8164 A (0.26 0.24 0.22 0.2 0.088)  
##         20) magnet_dumbbell_y< 438.5 9325 6577 A (0.29 0.19 0.25 0.19 0.084)  
##           40) num_window< 241.5 2211  893 A (0.6 0.14 0.11 0.12 0.03) *
##           41) num_window>=241.5 7114 5040 C (0.2 0.2 0.29 0.21 0.1)  
##             82) magnet_dumbbell_z< -27.5 1596  562 A (0.65 0.21 0.056 0.068 0.018) *
##             83) magnet_dumbbell_z>=-27.5 5518 3533 C (0.072 0.2 0.36 0.25 0.12) *
##         21) magnet_dumbbell_y>=438.5 1654  734 B (0.041 0.56 0.044 0.25 0.11) *
##       11) num_window< 45.5 513  100 E (0 0 0 0.19 0.81) *
##    3) roll_belt>=130.5 1156   10 E (0.0087 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 1488  279  183  152   42
##          B   20  373   36  165   89
##          C  162  487  807  601  306
##          D    0    0    0    0    0
##          E    4    0    0   46  645
## 
## Overall Statistics
##                                           
##                Accuracy : 0.563           
##                  95% CI : (0.5502, 0.5757)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.4413          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8889  0.32748   0.7865   0.0000   0.5961
## Specificity            0.8442  0.93468   0.6798   1.0000   0.9896
## Pos Pred Value         0.6940  0.54612   0.3415      NaN   0.9281
## Neg Pred Value         0.9503  0.85275   0.9378   0.8362   0.9158
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2528  0.06338   0.1371   0.0000   0.1096
## Detection Prevalence   0.3643  0.11606   0.4015   0.0000   0.1181
## Balanced Accuracy      0.8666  0.63108   0.7332   0.5000   0.7929
```

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



Rattle 2020−mei−08 16:22:07 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 1667    7    0    1    0
##          B    6 1112    2    7    3
##          C    0   19 1021    8    2
##          D    1    1    2  945   11
##          E    0    0    1    3 1066
##
## Overall Statistics
```

```
##
##                Accuracy : 0.9874
##                  95% CI : (0.9842, 0.9901)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9841
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9763   0.9951   0.9803   0.9852
## Specificity            0.9981   0.9962   0.9940   0.9970   0.9992
## Pos Pred Value         0.9952   0.9841   0.9724   0.9844   0.9963
## Neg Pred Value         0.9983   0.9943   0.9990   0.9961   0.9967
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1890   0.1735   0.1606   0.1811
## Detection Prevalence   0.2846   0.1920   0.1784   0.1631   0.1818
## Balanced Accuracy      0.9970   0.9863   0.9946   0.9886   0.9922
```

##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
```