week 4 assignment

Manoj

4/15/2021

BackGround

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement ??? a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Creating training and test sets.

```
trainingSet<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))

testSet<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))
```

Dimension of training dataset

```
dim(trainingSet)
```

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

Training dataset has 160 columns 19622 rows, we likely to overfit the data using 160 columns. First goal is to reduce the no of columns.

checking columns having most NA's and removing them from dataset.

```
trainingSet[trainingSet==""]<-NA

length(which(sapply(trainingSet,function(x) sum(is.na(x))/nrow(trainingSet))>0.97))
```

```
## [1] 100
```

```
testSet[testSet==""]<-NA

trainingSet$classe<-as.factor(trainingSet$classe)
```

There are 100 columns which have more than 97 of missing values, removing them from dataset.

```
trainingSetRemovedColumns<-trainingSet[,-c(which(sapply(trainingSet,function(x) sum(is.na(x))/nrow(trai

testSetRemovedColumns<-testSet[,-c(which(sapply(testSet,function(x) sum(is.na(x))/nrow(testSet))>0.97))]
```

Removing first 7 variables as they are windows generated.

```
trainingSetRemovedColumns<-trainingSetRemovedColumns[,-c(1:7)]
testSetRemovedColumns<-testSetRemovedColumns[,-c(1:7)]
```

checkimg and removing zero covariates if there are any from dataset

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
nsv<-nearZeroVar(trainingSetRemovedColumns,saveMetrics = T)
nsv[nsv$nzv==T,]
```

```
## [1] freqRatio     percentUnique zeroVar       nzv
## <0 rows> (or 0-length row.names)
```

there no columns that has true non zero variance.

Creating training and validation data sets from training set

```
inTrian<-createDataPartition(trainingSetRemovedColumns$classe,p=0.8,list = FALSE)
newTrainingSet<-trainingSetRemovedColumns[inTrian,]
crossValSet<-trainingSetRemovedColumns[-inTrian,]
```

applying preprocessing and pca to the data.

```
ctrl=trainControl(preProcOptions = list(thresh=0.80))
modelRpart<-train(classe~.,data =newTrainingSet,method="rpart",preProcess=c("center","scale","pca"),trC
```

plotting tree generated by rpart.

```
library(rattle)
```

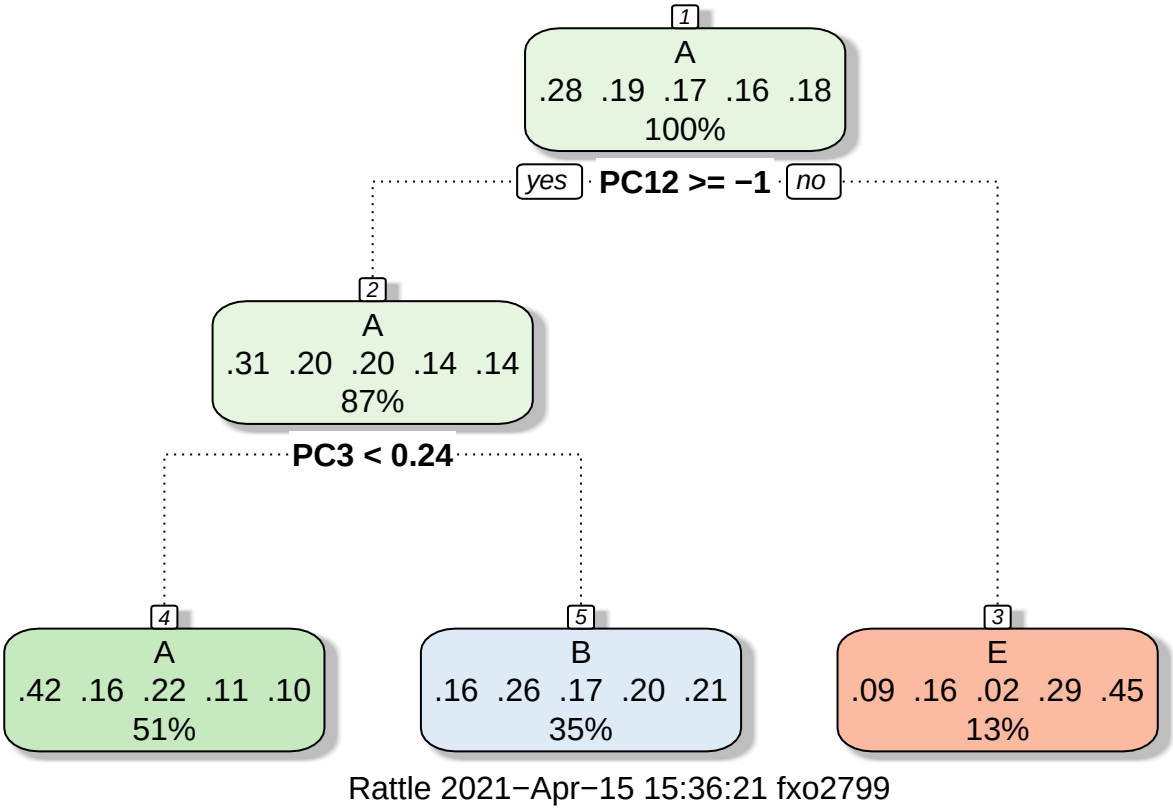
```
## Warning: package 'rattle' was built under R version 4.0.5
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(modelRpart$finalModel)
```



predicting classe using validation data

```
crossValPredictions<-predict(modelRpart,crossValSet)
confusionMatrix(crossValSet$classe,crossValPredictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 832 242   0   0  42
##          B 366 302   0   0  91
##          C 423 250   0   0  11
##          D 206 299   0   0 138
##          E 196 296   0   0 229
##
## Overall Statistics
##
##                Accuracy : 0.3474
##                  95% CI : (0.3325, 0.3626)
##     No Information Rate : 0.5157
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1423
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4113  0.21742       NA       NA  0.44814
## Specificity            0.8505  0.81965   0.8256   0.8361  0.85580
## Pos Pred Value         0.7455  0.39789       NA       NA  0.31761
## Neg Pred Value         0.5757  0.65645       NA       NA  0.91193
## Prevalence             0.5157  0.35407   0.0000   0.0000  0.13026
## Detection Rate         0.2121  0.07698   0.0000   0.0000  0.05837
## Detection Prevalence   0.2845  0.19347   0.1744   0.1639  0.18379
## Balanced Accuracy      0.6309  0.51854       NA       NA  0.65197
```

rpart model has very low accuracy lets try random forest and predict validation data.

```
modelrf<-train(classe~.,data =newTrainingSet,method="rf",preProcess=c("center","scale"),ntree=100)
crossValPredictions<-predict(modelrf,crossValSet)
confusionMatrix(crossValSet$classe,crossValPredictions)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    1    1    0    0
##          B    6  753    0    0    0
```

```
##          C    0    3  678    3    0
##          D    0    1    8  632    2
##          E    0    2    2    4  713
##
## Overall Statistics
##
##                Accuracy : 0.9916
##                  95% CI : (0.9882, 0.9942)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9894
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9908   0.9840   0.9890   0.9972
## Specificity            0.9993   0.9981   0.9981   0.9967   0.9975
## Pos Pred Value         0.9982   0.9921   0.9912   0.9829   0.9889
## Neg Pred Value         0.9979   0.9978   0.9966   0.9979   0.9994
## Prevalence             0.2855   0.1937   0.1756   0.1629   0.1823
## Detection Rate         0.2840   0.1919   0.1728   0.1611   0.1817
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9970   0.9944   0.9911   0.9928   0.9974
```

random forest performs better than rpart with an accuracy of 99.5%

we will us the rf model to predict the the test set.

```
predict(modelrf,testSetRemovedColumns)
```

```
##  [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
```