Predictive Machine Learning Project

Aaron Kelly

10/14/2018

Predicting Exercise Manner from Spatial Orientation Data

A random forest model, controlled to prevent overfitting is applied to spatial orientation data to categorize exercise manner. The data are first cleaned and then subset to relevant variables (the spatial orientation variables, without the timeseries and indexing variables). A chunk of the data is broken off and split into a training and testing subset. The random forest is trained and tested for accuracy. Finally, the model is applied to a new set of data, generating predictions used in answering questions on a quiz.

```
#caret library is loaded in order to avail myself of the machine learning training functions it contain
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018e.
## 1.0/zoneinfo/America/Los_Angeles'
```

```
#The relevant data is loaded from its source into a data frame object.
data<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),na.strings=c


#The loaded data are cleaned up.

##An empty data frame object is created, having the same number of rows as that of the loaded data, but
  Data=as.data.frame(matrix(nrow=length(data[,1]),ncol=0))

##An object is created in which will be stored the names of columns from the loaded that are clean.
  Names<-c()

##Finally, a bit of action.  A loop that uses the index, "i", and goes from 1 to the number of columns i
  for (i in 1:length(data[1,])){

    ###This is just a ratio or decimal that measures what percentage of a particular column has NAs.
    incompleteness<-sum(is.na(data[,i]))/length(data[,1])

    ###If the column is completely clean, then we put the name of column into the Names object, and we p
    if (incompleteness==0){
      Names<-c(Names,colnames(data)[i])
      Data<-data.frame(Data,data[,i])
      names(Data)<-Names
    }}

#Now that we have a clean version of the data, the indexing and timestamp columns are removed, because
  Data<-Data[,-(1:7)]
```

```
#Cutting off a random chunk of the data for training.

##Because the number of observations in the cleaned dataframe is quite large and the machine learning i

  ###Partition is created, marking off 1/4th of the data.
  chunkPart<-createDataPartition(Data$classe,p=1/4)[[1]]

  ###The partition created above is applied the clean datafram resulting in a random subset that is 1/4
  chunk<-Data[chunkPart,]

  ###The chunk is further partitioned into a training set consisting of 3/4 of the chunk = 3/16 of the
  trainPart<-createDataPartition(chunk$classe, p=3/4)[[1]]
  training<-chunk[trainPart,]

  ###The complement of the training set within chunk is stored as a testing set, which amounts to about
  testing<-chunk[-trainPart,]


#Creating Model on Training Subset

##First, a parameter is set that will force the machine learning to use cross-validation, which will spl
governer<-trainControl(method="cv",number=50)

##Now, a random forest is trained because it is fundamentally a process for sorting into categories, whi
rfModel<-train(classe~.,method="rf",data=training, trControl=governer)

##Once the model has been trained, it is applied to the testing set.
rfPredict<-predict(rfModel,subset(testing, select=-c(classe)))

##A logical vector is created that effectively counts the number of predictions that are correct.
rfTrue<-rfPredict==testing$classe

##The count of correct predictions is converted into a percentage which tells the accuracy of the predi
rfEval<-sum(rfTrue)/length(testing$classe)*100

##The measurement of the accuracy of the model is printed.
print(rfEval)
```

```
## [1] 97.06122
```

```
#Now that the model has been created and tested, it is applied to the important data (the quiz data).

##The data to be used for final prediction are loaded into file.
quizTest<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),na.string

##The data are cleaned, as before.
Names<-c()
cleanquizTest<-as.data.frame(matrix(nrow=length(quizTest[,1]),ncol=0))

for (i in 1:length(quizTest[1,])){

  incompleteness<-(sum(is.na(quizTest[,i]))/length(quizTest[,1]))
  if (incompleteness==0){
    Names<-c(Names,colnames(quizTest)[i])
    cleanquizTest<-data.frame(cleanquizTest,quizTest[,i])
    names(cleanquizTest)<-Names
  }
}

##The previously created model is applied to the quiz data, generating predictions, which are then prin
rfPredictQuiz<-predict(rfModel,cleanquizTest[,-60])
print(rfPredictQuiz)
```

```
##  [1] B A A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```