PA1

alanpoon

Tuesday, March 31, 2015

Investigation of barbell lifts analysis

Synopsis

In this analysis, machine learning algorithm is employed to the prediction of the manner the test subjects used in the exercise. This report will also cover how the model is being built and how the sample error is estimated. The prediction model will be used to predict 20 different test cases.

Getting the training data

```r
setwd("C:/Users/alanpoon/Desktop/coursera/Practical Machine Learning 8/machineLearningGit")

if (!"trainingData.csv" %in% dir("../courseProjectData")  ) {

    print("trainingData.csv is there")
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                  destfile = "../courseProjectData/trainingData.csv")

  }

if (!"trainingData" %in% ls()) {
    trainingData <- read.csv("../courseProjectData/trainingData.csv", sep = ",")
}
```

Getting the test data

```r
if (!"testData.csv" %in% dir("../courseProjectData")  ) {

    print("testData.csv is there")
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                  destfile = "../courseProjectData/testData.csv")


}
if (!"testData" %in% ls()) {
    testData <- read.csv("../courseProjectData/testData.csv", sep = ",")
}
```

Prediction Study Design

*Design Methodology +Define Error Rate ++The Type of error Rate used to evaluate the prediction is accuracy because we want to weigh false positives/negatives equally. +Split Data into Training, Testing and validation (optional) +On the Training set pick features- use cross-validation +On the Training set pick prediction function- use cross validation +If no validation - apply 1x to test set +If validation - apply to test set and refine, apply 1x to validation

Designing the prediction model for the manner of lifting barbell

Since there are several methods of lifting the barbell, prediction model of binary variable is not chosen. Since, there are many variables, principal component analysis may be used. Since there are more than 2 classes, generalized linear model cannot be used. Since, there are alot missing values in the data, decision tree may be the easiest machine learning algorithm to apply.

Data Preprocessing

```
testData[is.na(testData)]<-0
trainingData[is.na(trainingData)]<-0

jf<-trainingData
jf<-jf[,colSums(is.na(jf)) == 0]
drops <- c("cvtd_timestamp")
tf<-jf[,!(names(jf) %in% drops)]
```

All the NA values are replaced by 0. The columns that contain blanks and #Div/0! are dropped. ### DownSample the imbalanced Data We noted that Classe C is the minority class with 3422 records, while the Classe A is the majority class with 5580 records. We will perform down sample on the dataset to resolve the problem of imbalanced Data.

```
library(caret)
tf<-downSample(tf,as.factor(tf$classe))
```

Create Data Partition

We will create data partition to split the training set data into data to be used by model fitting and sample test data for validation and statistical measure for our model.

```
set.seed(32323)
folds<-createFolds(y=tf,k=5,list=FALSE,returnTrain=FALSE)
##summary(tf[folds[[2]],])
```

```
set.seed(32323)
inTrain<-createDataPartition(y=tf$classe,p=0.75,list=FALSE)
trainModelData<-tf[inTrain,]
sampleTestData<-tf[-inTrain,]
```

Table for the predictors

We use nearZeroVar() in caret package to remove variables that are near zero variates.

```r
nsv<-nearZeroVar(trainModelData,saveMetrics=TRUE)
nsvToBeDrop<-nsv[nsv$nzv==TRUE,]
drops<-row.names(nsvToBeDrop)
yf<-trainModelData[,!(names(trainModelData) %in% drops)]
```

'yf' is the data frame after the near zero variables are removed. Next, we want to perform (PCA), Principal Component Analysis to weigh the variables. First, we need to remove the class variable and the user name, which is the 57th and 1st variables in the yf.

```r
pf<-yf[,c(-58,-59,-2)]

uf<-abs(cor(pf))
uf[upper.tri(uf)]<-0
diag(uf)<-0
er<-which(uf>0.8,arr.ind=T)
```

We obtain a list of variables having high correlation with each other. Next, we want to plot their correlation to spot trends with ggplot2 package.

```r
library(ggplot2)
xName<-names(pf)[c(8)]
yName<-names(pf)[c(5)]
qq<- qplot(get(xName),get(yName),colour=classe,data=yf) + geom_smooth(method='lm',formula=y~x)
```

For the 8th and 5th variable, we do not see any importance of considering both 8th and 5th variables together in predicting the classe attribute. Next, we further reduce the number of variables if they are correlated. From 'er' table, we can accept 5th,6th,12th,22th,25th,29th,32th and 33rd variables.

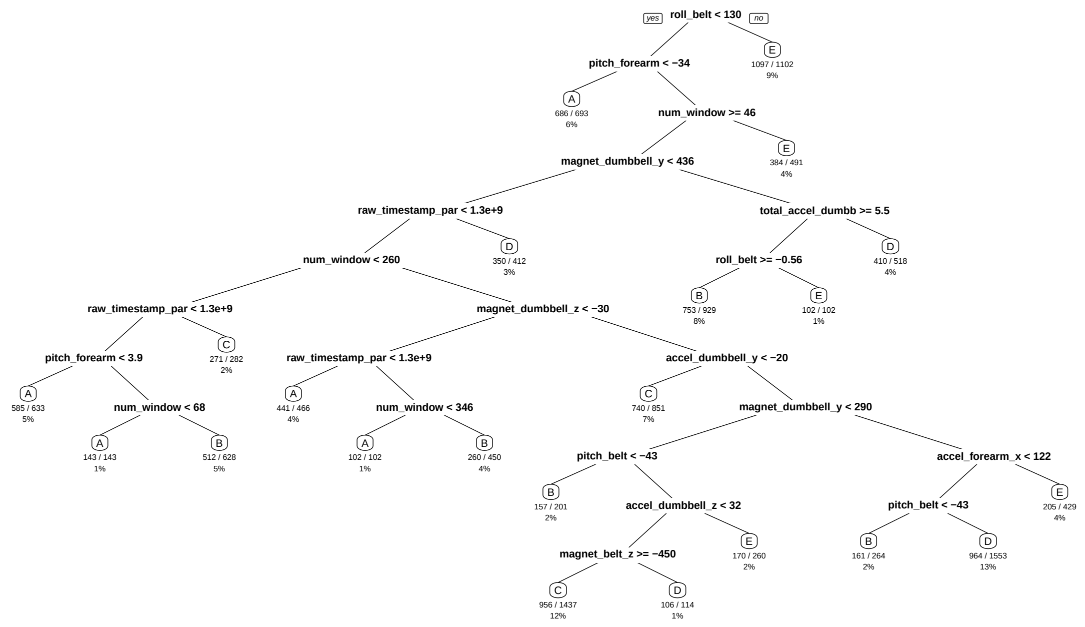
```r
acceptableVar<-pf[,!apply(uf,2,function(x) any(x > 0.80)) ]
acceptableVar1<-pf[,sapply(as.vector(row.names(uf)),function(x) match(x,row.names(uf)) %in% c(1,5,6,12,
data.new<-merge(acceptableVar,acceptableVar1,by='X')

drops <- c("X")
data.new<-data.new[,!(names(data.new) %in% drops)]
### insert back the classe attribute
data.new$classe<-yf$classe
data.new$user_name<-yf$user_name
```

Training model

```r
   library(rattle)
library(rpart)
library(rpart.plot)
  set.seed(125)
model1<-rpart(classe ~ ., data=data.new, method="class")
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```

Classification Tree



Predict the sample Test Data using the model in rpart

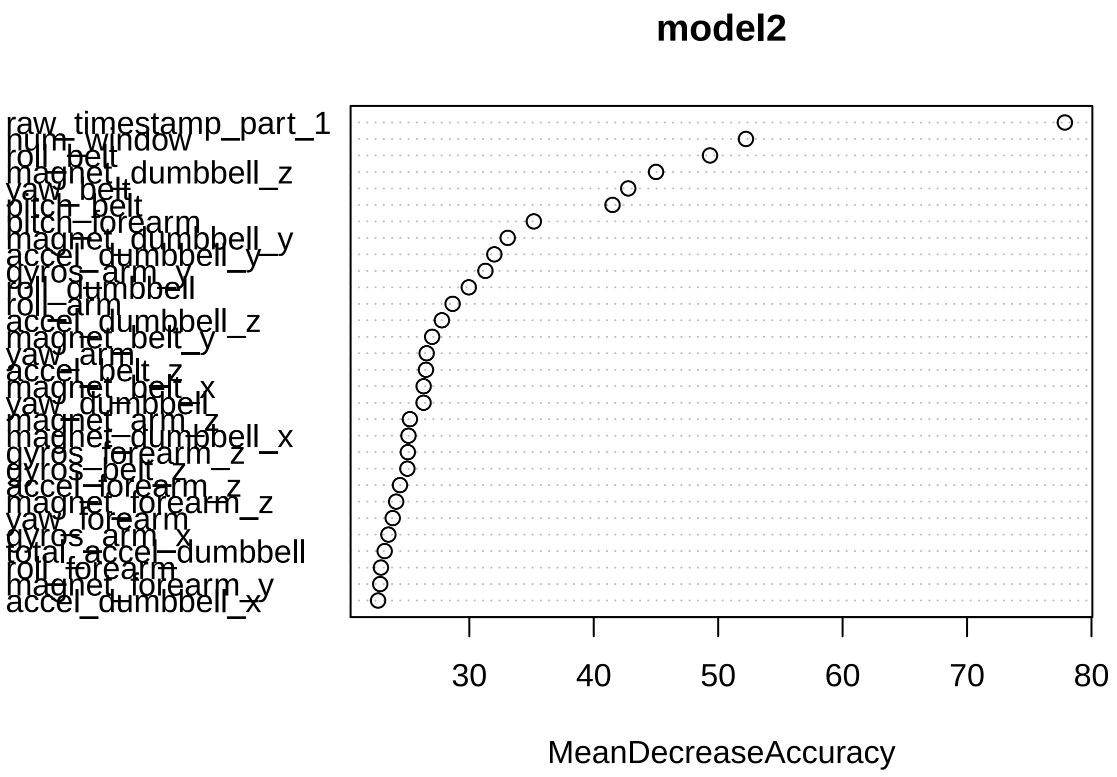
```r
ptraining<-predict(model1,sampleTestData,type='class')
confusionMatrix(ptraining, sampleTestData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 663  24   2   0   3
##          B  85 608  53  36  35
##          C  14  49 660 123  31
##          D  25  87  56 592 105
##          E  17  36  33  53 630
##
## Overall Statistics
##
##                Accuracy : 0.7843
##                  95% CI : (0.7713, 0.797)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7304
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8246   0.7562   0.8209   0.7363   0.7836
## Specificity            0.9910   0.9350   0.9325   0.9151   0.9568
## Pos Pred Value         0.9581   0.7442   0.7526   0.6844   0.8192
## Neg Pred Value         0.9576   0.9388   0.9542   0.9328   0.9465
## Prevalence             0.2000   0.2000   0.2000   0.2000   0.2000
## Detection Rate         0.1649   0.1512   0.1642   0.1473   0.1567
## Detection Prevalence   0.1721   0.2032   0.2182   0.2152   0.1913
## Balanced Accuracy      0.9078   0.8456   0.8767   0.8257   0.8702
```

The cross-validation accuracy for Rpart is 80.82% ### Training model with random Forest

```
library(randomForest)
 set.seed(125)
model2<-randomForest(classe ~ ., data = data.new, importance = TRUE, ntrees = 10)
#what are the important variables (via permutation)
varImpPlot(model2, type=1)
```



```
ptraining2 <- predict(model2, sampleTestData)
confusionMatrix(ptraining2, sampleTestData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 802   0   0   0   0
##          B   2 804   1   0   0
##          C   0   0 803   4   0
##          D   0   0   0 799   3
##          E   0   0   0   1 801
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9951, 0.9986)
##     No Information Rate : 0.2
##     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            0.9975   1.0000   0.9988   0.9938   0.9963
## Specificity            1.0000   0.9991   0.9988   0.9991   0.9997
## Pos Pred Value         1.0000   0.9963   0.9950   0.9963   0.9988
## Neg Pred Value         0.9994   1.0000   0.9997   0.9984   0.9991
## Prevalence             0.2000   0.2000   0.2000   0.2000   0.2000
## Detection Rate         0.1995   0.2000   0.1998   0.1988   0.1993
## Detection Prevalence   0.1995   0.2007   0.2007   0.1995   0.1995
## Balanced Accuracy      0.9988   0.9995   0.9988   0.9964   0.9980
```

Predict the sample Test Data using the model in random Forest The cross-validation accuracy is 99.88%.

Applying Model to the testData

Since randomForest accuracy is better, randomForest is selected predict the testData.

```r
ptest2<-predict(model2,testData)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```