**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).

**Data Location:**

The training data for this project are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv>

The test data are available here:

<https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

**Steps Followed:**

**Install frequently used packages:**

install.packages("AppliedPredictiveModeling")

install.packages("caret")

install.packages("e1071")

install.packages("rattle")

install.packages("pgmm")

install.packages("ElemStatLearn")

install.packages("randomForest")

install.packages("gbm")

install.packages("elasticnet")

install.packages("lubridate")

install.packages("forecast")

install.packages("dplyr")

library(AppliedPredictiveModeling)

library(caret)

library(e1071)

library(rattle)

library(pgmm)

library(ElemStatLearn)

library(randomForest)

library(gbm)

library(elasticnet)

library(lubridate)

library(forecast)

library(dplyr)

**#Read csv files into R**

setwd("C:/Users/User/Desktop/R/ML Project")

base <- read.csv("pml-training.csv")

test <- read.csv("pml-testing.csv")

**#Create Training & Validation with 60:40 split**

inTrain = createDataPartition(base$classe, p = 0.6)[[1]]

training = base[ inTrain,]

validation = base[-inTrain,]

**Data Cleaning:**

The excel file had text fields like “#DIV0!” which would be meaningless in terms of predicting the “Classe” variable. Hence, such values were replaced by missing values. Thereafter, variables with limited information were removed – namely, ones with more than 90% missing, and the ones with near zero variance. Some variables like Serial Number, Name and Timestamp were also removed since they would not make a meaningful prediction of “classe”.

**#Replace #DIV0! by blank/missing**

varnames <- c("kurtosis\_picth\_belt", "kurtosis\_yaw\_belt", "skewness\_roll\_belt", "skewness\_roll\_belt.1", "skewness\_yaw\_belt",

"skewness\_roll\_arm", "skewness\_pitch\_arm", "skewness\_yaw\_arm",

"kurtosis\_roll\_arm", "kurtosis\_picth\_arm", "kurtosis\_yaw\_arm",

"kurtosis\_yaw\_dumbbell", "skewness\_yaw\_dumbbell",

"kurtosis\_yaw\_forearm", "skewness\_yaw\_forearm")

varnames

clean<-function(x){

x<-gsub("#DIV/0!","",x)

}

training[,c(varnames)] <- apply(training[,c(varnames)],2,clean)

head(training,100)

validation[,c(varnames)] <- apply(validation[,c(varnames)],2,clean)

head(validation,100)

**#Remove ones with > 90% missing :**

training.na <- sapply(training, function(x) mean(is.na(x))) > 0.90

training.na

training <- training[, training.na==FALSE]

validation <- validation[, training.na==FALSE]

**#Remove Serial No., Name, Timestamp:**

training <- training[, -(1:5)]

validation <- validation[, -(1:5)]

**#Remove variables with very low or zero variance:**

lowVariance <- nearZeroVar(training)

lowVariance

training <- training[, -lowVariance]

validation <- validation[, -lowVariance]

**#Compare Different Models:**

Three different prediction techniques were used to compare the results on validation in order to choose the most accurate one. Here we have tried Random Forest, Gradient Boosting Machines and Linear Discriminant Analysis.

**Random Forest:**

modelFit.rf <- train(classe~.,method="rf",data=training)

ValidPred.rf <- predict(modelFit.rf,validation)

confusionMatrix(validation$classe,ValidPred.rf)

*Confusion Matrix and Statistics*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction | A | B | C | D | E |
| A | 2,231 | - | - | - | 1 |
| B | 2 | 1,511 | 4 | 1 | - |
| C | - | 6 | 1,362 | - | - |
| D | - | - | 9 | 1,277 | - |
| E | - | - | - | 2 | 1,440 |

*Overall Statistics*

Accuracy : 0.9968

95% CI : (0.9953, 0.9979)

No Information Rate : 0.2846

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.996

Mcnemar's Test P-Value : NA

*Statistics by Class:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Class: A | Class: B | Class: C | Class: D | Class: E |
| Sensitivity | 0.9991 | 0.996 | 0.9905 | 0.9977 | 0.9993 |
| Specificity | 0.9998 | 0.9989 | 0.9991 | 0.9986 | 0.9997 |
| Pos Pred Value | 0.9996 | 0.9954 | 0.9956 | 0.993 | 0.9986 |
| Neg Pred Value | 0.9996 | 0.9991 | 0.998 | 0.9995 | 0.9998 |
| Prevalence | 0.2846 | 0.1933 | 0.1752 | 0.1631 | 0.1837 |
| Detection Rate | 0.2843 | 0.1926 | 0.1736 | 0.1628 | 0.1835 |
| Detection Prevalence | 0.2845 | 0.1935 | 0.1744 | 0.1639 | 0.1838 |
| Balanced Accuracy | 0.9995 | 0.9975 | 0.9948 | 0.9981 | 0.9995 |

**Gradient Boosting Machines:**

modelFit.gbm <- train(classe~.,method="gbm",data=training)

ValidPred.gbm <- predict(modelFit.gbm,validation)

confusionMatrix(validation$classe,ValidPred.gbm)

*Confusion Matrix and Statistics*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction | A | B | C | D | E |
| A | 2,221 | 11 | - | - | - |
| B | 18 | 1,481 | 15 | 4 | - |
| C | - | 20 | 1,344 | 4 | - |
| D | - | 5 | 21 | 1,260 | - |
| E | - | 3 | 1 | 13 | 1,425 |

*Overall Statistics*

Accuracy : 0.9853

95% CI : (0.9824, 0.9879)

No Information Rate : 0.2854

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9815

Mcnemar's Test P-Value : NA

*Statistics by Class:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Class:A | Class:B | Class:C | Class:D | Class:E |
| Sensitivity | 0.992 | 0.9743 | 0.9732 | 0.9836 | 1 |
| Specificity | 0.998 | 0.9942 | 0.9963 | 0.996 | 0.9974 |
| Pos Pred Value | 0.9951 | 0.9756 | 0.9825 | 0.9798 | 0.9882 |
| Neg Pred Value | 0.9968 | 0.9938 | 0.9943 | 0.9968 | 1 |
| Prevalence | 0.2854 | 0.1937 | 0.176 | 0.1633 | 0.1816 |
| Detection Rate | 0.2831 | 0.1888 | 0.1713 | 0.1606 | 0.1816 |
| Detection Prevalence | 0.2845 | 0.1935 | 0.1744 | 0.1639 | 0.1838 |
| Balanced Accuracy | 0.995 | 0.9842 | 0.9847 | 0.9898 | 0.9987 |

**Linear Discriminant Analysis:**

modelFit.lda <- train(classe ~ .,method="lda",data=training,preProcess=c("center","scale"))

ValidPred.lda <- predict(modelFit.lda,validation)

confusionMatrix(validation$classe,ValidPred.lda)

*Confusion Matrix and Statistic:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Prediction | A | B | C | D | E |
| A | 1,847 | 48 | 150 | 178 | 9 |
| B | 242 | 965 | 191 | 60 | 60 |
| C | 127 | 132 | 912 | 162 | 35 |
| D | 68 | 47 | 157 | 966 | 48 |
| E | 57 | 199 | 124 | 148 | 914 |

*Overall Statistics:*

Overall Statistics

Accuracy : 0.7142

95% CI : (0.7041, 0.7242)

No Information Rate : 0.2984

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6384

Mcnemar's Test P-Value : < 2.2e-16

*Statistics by Class:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Class:A | Class:B | Class:C | Class:D | Class:E |
| Sensitivity | 0.789 | 0.6937 | 0.5945 | 0.638 | 0.8574 |
| Specificity | 0.9301 | 0.9143 | 0.9278 | 0.9495 | 0.9221 |
| Pos Pred Value | 0.8275 | 0.6357 | 0.6667 | 0.7512 | 0.6338 |
| Neg Pred Value | 0.912 | 0.9327 | 0.904 | 0.9165 | 0.9763 |
| Prevalence | 0.2984 | 0.1773 | 0.1955 | 0.193 | 0.1359 |
| Detection Rate | 0.2354 | 0.123 | 0.1162 | 0.1231 | 0.1165 |
| Detection Prevalence | 0.2845 | 0.1935 | 0.1744 | 0.1639 | 0.1838 |
| Balanced Accuracy | 0.8595 | 0.804 | 0.7611 | 0.7938 | 0.8898 |

**Final Prediction Used:**

Random Forest and GBM both showed good accuracy. However, Random Forest had a better accuracy. Also, the balanced accuracy was >99% for all the classes in Random Forest. With an accuracy of 99.68%, this model should predict the “classe” correctly almost all the times (~299 out of 300 times)

**Result on Test Data:**

The Random Forest model was used to predict the 20 test cases in the test dataset since it was the most accurate model among the three that were tried out. Below are the results:

Test.rf <- predict(modelFit.rf,test)

Test.rf

*Output:*

[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