Practical Machine Learning: Project Writeup

Introduction

The goal of this project was to take almost 20,000 rows of tesing data from wearable devices with known exercies of their recordings, to build a machine learning algorithm to use in predicting 20 test cases. Several models were used. In the end a random forest model had the highest accuracy and was used to corectly predic 20 out of the 20 test cases.

Loading the Data

The data was downloaded and loaded into R. Set seed was set to 19662 for reproducability.

```
set.seed(19622)
training<-read.csv("C:/Users/Chris/ML/pml-training.csv")
testing<-read.csv("C:/Users/Chris/ML/pml-testing.csv")
```

Preprocessing the Data

In looking at the data, I noticed there were a lot of NA values. I felt that if there were too many, it could screw up the results of the model building. I decided that if a column had more thean 70% NA values, it should not be used in the training data. Many columns didn't have NA values because they were in a text form, so the function, converted everthing to a number and then looked to see which columns had more than 70% NAs. It then took those columns out. In the end the training data went from 160 columns of data to 60 columns of more complete data.

```
Num<-training[,7:159]
for (i in 1:length(names(training[,7:159]))){
        Num[,i]<-as.numeric(as.character(Num[,i]))
}
training[,7:159]<-Num
NAthreshold<-sapply(training,function(x)
        (sum(is.na(x))/length(x)>.70))
training<-training[,NAthreshold==FALSE]
```

Testing for Correlations

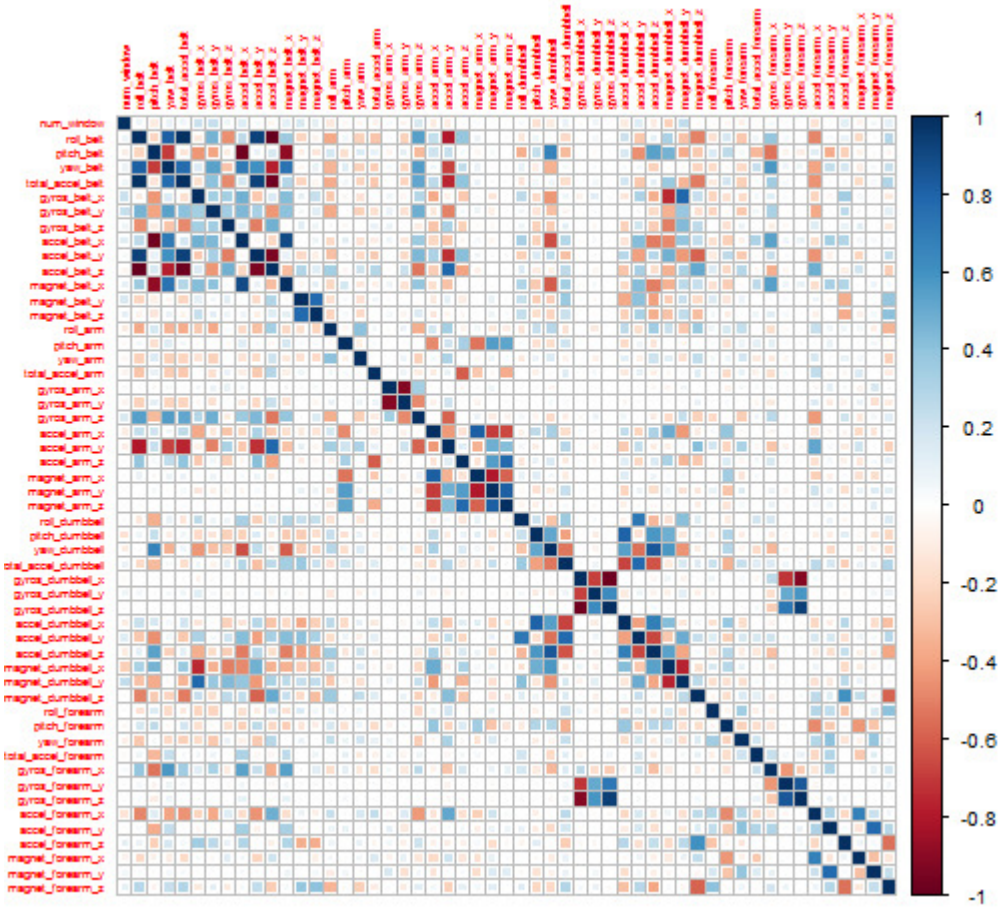
If we were to use a linear model, it would be imporotant to make sure that the variables do not

have a very stron correlation with eachoher. A correlation matrix was created and charted out. In the end, I decided that the the correlations weren't too bad and that I decided not to preprocess any further with PCA.

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.2.2
```

```
M<-cor(training[,7:59],use="complete.obs",method="pearson")
corrplot(M,method="square",tl.cex=.5)
```



Cross Validation

Because of the large amount of data, I expected to have a hight amount of accuracy. For any model to be accepted I would want at least a 90% accuracy rate. Because the predicictions are factor varuables, I decided to evaluate my models based on accuracy rather than a different model evaluator. Given the amount data to train the models, I felt it was not unreasonable to epect that one of them could have less than 10% error rate. This means that I would most likely have no more than 2 wrong in the testing data set. To cross validate the results in the testing, I decided to

use Random subsampling. I took 20% of the training data and set it aside. I would create my models based on 80% of the training data and test the models on the 20% before using the models on the 20 cases of testing data.

```
library(caret);library(ggplot2)
```

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

```
inTrain<-createDataPartition(y=training$classe,p=.8,list=FALSE)
train.train<-training[inTrain,]
train.test<-training[-inTrain,]
```

Model Building

I decided to test several models and figure out which one had the best accuracy. In caret, these models took forever to run. The Niave Bayes, took two hours to build. I found that in caret, the random forest model took 90 minutes, but only took a few minutes if you used the randomForest function. The models I created were: *Regression Trees: mod_rpart *Linear Discriminant Analysis: mod_lda *Random Forest using caret: mod_rf *Random Forest direct: mod_rf2 *Niave Bayes: mod_nb *Generalized Boosted Regression Models: mod_gbm

```
mod_rpart<-train(classe~.,method="rpart",data=train.train[,-(1:7)])
mod_lda<-train(classe~.,method="lda",data=train.train[,-(1:7)])
mod_rf<-train(classe~.,method="rf",data=train.train[,-(1:7)])
mod_rf2<-randomForest(classe~.,data=train.train[,-(1:7)])
mod_nb<-train(classe~.,method="nb",data=train.train[,-(1:7)])
mod_gbm<-train(classe~.,method="gbm",data=train.train[,-
(1:7)],verbose=FALSE)
```

Evaluaing the results

To evaluate the results, I predicted the classe variable with the 20% of training data I set aside as train.test for each model.

```
test_rpart<-predict(mod_rpart,train.test[,-60])
test_lda<-predict(mod_lda,train.test[,-60])
test_rf<-predict(mod_rf,train.test[,-60])
test_rf2<-predict(mod_rf2,train.test[,-60])
test_nb<-predict(mod_nb,train.test[,-60])
test_gbm<-predict(mod_gbm,train.test[,-60])
```

I then looked at the accuracy from a confusion matrix based on each model.

```
cm_rpart<-confusionMatrix(train.test$classe,test_rpart)$overall[1]
cm_lda<-confusionMatrix(train.test$classe,test_lda)$overall[1]
cm_rf<-confusionMatrix(train.test$classe,test_rf)$overall[1]
cm_rf2<-confusionMatrix(train.test$classe,test_rf2)$overall[1]
cm_nb<-confusionMatrix(train.test$classe,test_nb)$overall[1]
cm_gbm<-confusionMatrix(train.test$classe,test_gbm)$overall[1]
```

The follwing accuracy was predicted from each model:

```
accuracytbl<-rbind(cm_rpart,cm_lda,cm_rf,cm_rf2,cm_nb,cm_gbm)
accuracytbl
```

```
##      model  accuracy
## 1 cm_rpart 0.4774407
## 2   cm_lda 0.7007392
## 3    cm_rf 0.9954117
## 4   cm_rf2 0.9961764
## 5    cm_nb 0.7412694
## 6   cm_gbm 0.9592149
```

The random forest model was the clear winner. The one that used randomForest appears to be slightly better than the caret package. It took a lot less time to run than the caret package, that's for sure.

Testing the final data

Preparing the testing data

I needed to reduce the test data set down to the same columns I used in my training data set.

```
finTest<-testing[,NAthreshold==FALSE]
```

Running it through the model

Last step was to run it through the model

```
finAnswers<-predict(mod_rf2,finTest)
finAnsers
```

```
##    prediction
## 1           B
## 2           A
## 3           B
## 4           A
## 5           A
## 6           E
## 7           D
## 8           B
## 9           A
## 10          A
## 11          B
## 12          C
## 13          B
## 14          A
## 15          E
## 16          E
## 17          A
## 18          B
## 19          B
## 20          B
```

Using the random forest model, I was able to get all 20 out of 20 test cases correct.