machine learning project

Machine Learning Project

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.

The goal of the project is to use the available data to predict the manner in which they did the exercise. The report demonstrates the process of predction modeling, cross validation, and accuracy. Finally, the prediction model is applied to predict 20 different test cases.

Data loading and cleaning

Load the downloaded datasets and conduct data cleaning by removing columns that contain majority of NA missing values or empty values and columns that are unrelated to accelerometer measurements.

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

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

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## corrplot 0.84 loaded
```

dimension of the clean train dataset

```
dim(trainset)
```

```
## [1] 19622    53
```

dimension of test dataset

```
dim(testset)
```

```
## [1] 20 53
```

The clean training data contains 19622 observations and 53 variables, while the testing dataset contains 20 observations and 53 variables.

Split training data into a training dataset (70%) and a validation set (30%)

Split the clean training dataset into two parts: real training and validation sets. The validation set will be examined for cross-validation in future steps.

```
set.seed(12345) # For reproducibile purpose
inTrain <- createDataPartition(trainset$classe, p=0.70, list=F)
traindf <- trainset[inTrain, ]
inValid <- trainset[-inTrain, ]
```

Predict Modeling

Random Forest Modeling

Random Forest algorithm is used because it automatically selects important variables and is robust to correlate covariates and outliers in general.

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10989, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9911916  0.9888574
##   27    0.9900268  0.9873839
##   52    0.9842756  0.9801059
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Cross Validation

```
pred_rf <- predict(modFit_rf, newdata=inValid)
cm_rf <- confusionMatrix(pred_rf,inValid$classe)
cm_rf$overall["Accuracy"]
```

```
##  Accuracy
## 0.9892948
```

```
out_of_sample_error <- 1-as.numeric (cm_rf$overall[1])
out_of_sample_error
```

```
## [1] 0.01070518
```

With random forest, we reach an accuracy of 98.93% and the estimated out-of-sample error is (1- 0.9893) *100 = 1.07% by using cross-validation with 5 steps.

prediction of 20 testing cases

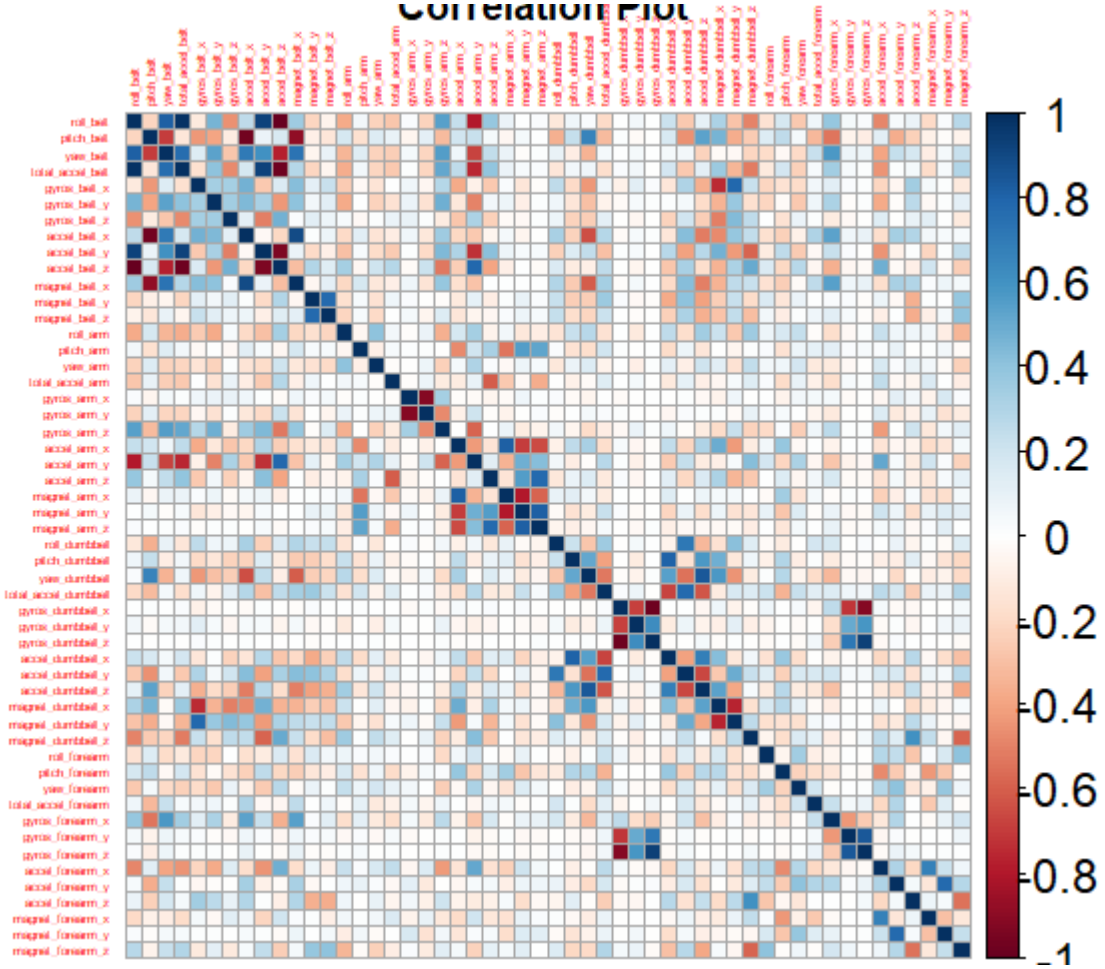
```
pred_test <- predict(modFit_rf, newdata=testset)
pred_test
```

```
##  [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
```

Appendix: Figures

1. Correlation Matrix Visualization (examination of multilineration)

```
cor_mat <- cor(trainset[,-53])
corrplot(cor_mat,method="color",addgrid.col = "darkgray",title="Correlation Plot",tl.cex=0.4,cl.ce
x = 1.5)
```




```
dev.copy(png,"plot 1.png",width=480, height=480)
```

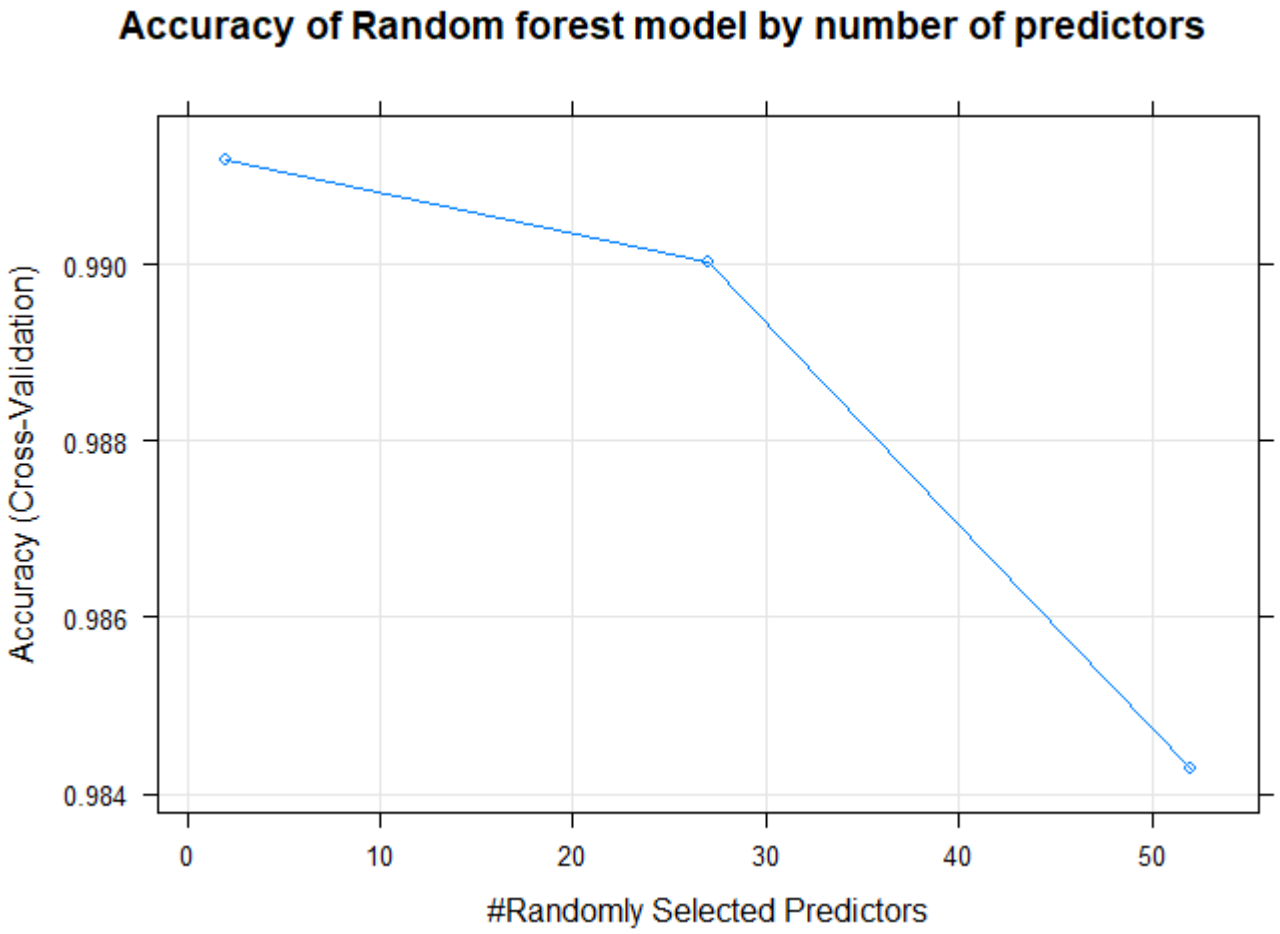
```
## png
##   3
```

```
dev.off()
```

```
## png
##   2
```

2. Accuracy Visulization of the random forest model

```
#png("plot 2.png",width=480, height=480)
plot(modFit_rf,main="Accuracy of Random forest model by number of predictors")
```




```
dev.copy(png,"plot 2.png",width=480, height=480)
```

```
## png
##   3
```

```
dev.off()
```

```
## png
##   2
```