Model For Predicting Exercise Quality

1.0 Summary: In a study, 6 test subjects were asked to perform barbell lifts correctly and incorrectly in 5 different ways while wearing accelerometers on the belt, forearms, arms and the dumbells. The data from accelerometers was recorded. The goal of this project is to build a supervised machine learning model that uses the recorded data to predict the manner in which the exercise was performed on a test data set.

2.0 Exploratory Data Analysis

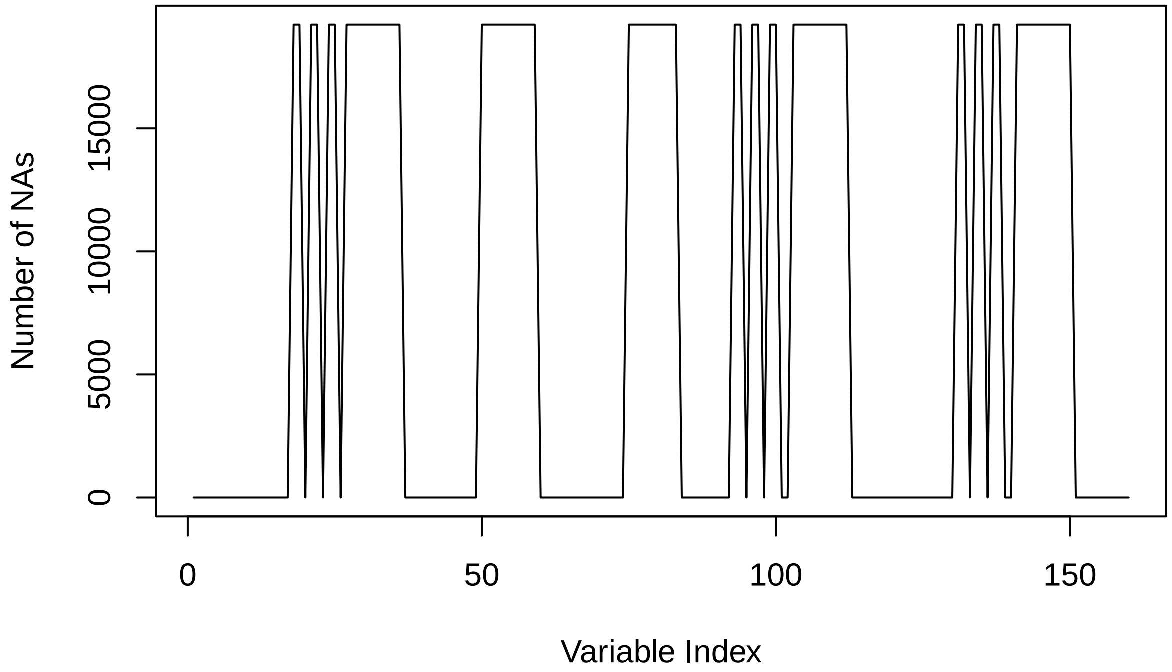
```r
if (!file.exists("./pml-training.csv")) {
    download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
        destfile = "./pml-training.csv")
}
if (!file.exists("./pml-testing.csv")) {
    download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
        destfile = "./pml-testing.csv")
}
```

2.1 Downloading The Data Reading the data into a data frame.

```r
input_data <- read.csv("./pml-training.csv")
test_data <- read.csv("./pml-testing.csv")
str(input_data)
```

2.2 Working With Missing Values Now that we have the data in a data frame, it is time to explore the columns in the data set. We know that the data set has 19622 records with 160 columns each. We first try to look at the missing values in the data set.

```r
df <- colSums(is.na(input_data))
plot(df, xlab = "Variable Index", ylab="Number of NAs", type='l')
```



The plot shows us that a lot of fields contain a significant number of NAs (close to 19000 NAs out of 19622 observations). We can remove these values safely because they do not affect the outcome of the experiment.

```
input_data <- read.csv("./pml-training.csv", na.strings=c("NA",""))
input_data <- input_data[, - which(as.numeric(colSums(is.na(input_data))) > 19000)]
```

The first 7 columns in the data set are `X`, `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`, `new_window` and `num_window`. These are static values and do not directly impact the outcome of the experiment. They can be safely removed from the data set.

```
input_data <- input_data[, -c(1:7)]
dim(input_data)
```

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

```
names(input_data)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

3.0 Feature Selection The next step is to select the predictor variables to be used in the model.

3.1 Corelated Predictors We now check for Corelated predictors in the data set. If two variables are highly correlated they will impart nearly exactly the same information to the regression model. Including both variables will result in a weak model by infusing the model with noise.

```
library(caret)
```

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

```
set.seed(1016)
in_train <- createDataPartition(input_data$classe, p=0.70, list=FALSE)
training <- input_data[in_train,]
validation <- input_data[-in_train,]
```

The following code examines the correlation coefficient. In this model, the Pearson correlation coefficient was chosen to be 0.99 (Indicating a very high level of correlation)

```
M <- abs(cor(training[,-53]))
diag(M) <- 0
which(M > 0.99,arr.ind=T)
```

```
##              row col
## accel_belt_z  10   1
## roll_belt      1  10
```

3.2 PCA There are variables in the data set which have a high corelation coefficient. PCA can be used to reduce the number of variables. We can set the cutoff for the cumulative percent of variance to be retained by PCA to 0.99.

```
preProc=preProcess(training[,-53],method="pca",thresh=.99)
pca_train=predict(preProc,training[,-53])
pca_validation <- predict(preProc, validation[-53])
```

This reduces the number of predictors in the training set from 53 to 36.

4.0 Predictive Model For building the predictive model, we use the Random Forest algorithm.

```r
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
model_rf = randomForest(training$classe~., data=pca_train, ntree = 2048)
model_rf
```

```
## 
## Call:
##  randomForest(formula = training$classe ~ ., data = pca_train,      ntree = 2048) 
##                Type of random forest: classification
##                      Number of trees: 2048
## No. of variables tried at each split: 6
## 
##         OOB estimate of  error rate: 1.99%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3898    4    1    2    1    0.002048
## B   48 2580   23    2    5    0.029345
## C    4   31 2342   17    2    0.022538
## D    3    0   86 2155    8    0.043073
## E    4   10   14    8 2489    0.014257
```

Using cross validation, the accuracy of the model can be checked.

```r
confusionMatrix(validation$classe, predict(model_rf, pca_validation))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1667    5    0    0    2
##          B   20 1104   13    1    1
##          C    1   12 1006    7    0
##          D    1    1   26  932    4
##          E    0    1    8    2 1071
## 
## Overall Statistics
## 
##                Accuracy : 0.982
##                  95% CI : (0.978, 0.985)
##     No Information Rate : 0.287
##     P-Value [Acc > NIR] : < 2e-16
## 
##                   Kappa : 0.977
##  Mcnemar's Test P-Value : 0.000312
## 
## Statistics by Class:
## 
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.987    0.983    0.955    0.989    0.994
## Specificity             0.998    0.993    0.996    0.994    0.998
## Pos Pred Value          0.996    0.969    0.981    0.967    0.990
## Neg Pred Value          0.995    0.996    0.990    0.998    0.999
## Prevalence              0.287    0.191    0.179    0.160    0.183
## Detection Rate          0.283    0.188    0.171    0.158    0.182
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.993    0.988    0.976    0.991    0.996
```

The model has an accuracy of `0.982`.

5.0 Conclusion Now that we have a model, we can use it to predict the exercise quality over the test data set. We can do this with the following code.

```
test_data <- test_data[, names(test_data) %in% names(input_data)]
pca_test <- predict(preProc, test_data)
predicted_results <- predict(model_rf, pca_test)
```

6.0 References

1. Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.