Practical Machine Learning Course 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. In this project, I will 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).

Step 1: Data Loading

The data for this project come from this source: (http://web.archive.org/web/20161224072740/http://groupware.les.inf.puc-rio.br/har).

```
library(lattice)
library(ggplot2)
library(caret)

if (!file.exists("pml-training.csv")) {
    fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
    download.file(fileUrl, destfile="./pml-training.csv")
}
if (!file.exists("pml-testing.csv")) {
    fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
    download.file(fileUrl, destfile="./pml-testing.csv")
}
pmlData <- read.csv("./pml-training.csv", na.strings=c("NA", ""), header=TRUE)
validation <- read.csv("./pml-testing.csv", na.strings=c("NA", ""), header=TRUE)
```

Step 2: Data Preparation

Notice the large amount of missing values in the many columns, we're removing columns with more than 95% NAs, as well as descriptive columns which are not relevant to model fitting.

```
if(anyNA(pmlData)){
    # Removing columns with more than 95% NAs
    keep <- names(pmlData)[apply(pmlData, 2, function(x) mean(is.na(x))<0.05)]
    pmlData <- subset(pmlData, select=keep)
}
# Remove columns not relevant to modeling
pmlData <- subset(pmlData, select=-grep("X|user_name|_timestamp|_window", names(pmlData)))
```

There are many models where predictors with a single unique value (also known as zero variance predictors“) will cause the model to fail. Since we will be tuning models using resampling methods, a random sample of the training set may result in some predictors with more than one unique value to become a zero-variance predictor (in our data, the simple split of the data into a test and training set caused three descriptors to have a single unique value in the training set). These so-called near zero-variance predictors” can cause numerical problems during resampling for some models, such as linear regression.

```r
# Remove zero covariates
nsv <- nearZeroVar(pmlData[, -grep("classe", names(pmlData))], saveMetrics=TRUE)
pmlData <- pmlData[, nsv$zeroVar==FALSE & nsv$nzv==FALSE]

summary(pmlData)
```

```
##    roll_belt        pitch_belt          yaw_belt       total_accel_belt
##  Min.   :-28.90   Min.   :-55.8000   Min.   :-180.00   Min.   : 0.00
##  1st Qu.:  1.10   1st Qu.:  1.7600   1st Qu.: -88.30   1st Qu.: 3.00
##  Median :113.00   Median :  5.2800   Median : -13.00   Median :17.00
##  Mean   : 64.41   Mean   :  0.3053   Mean   : -11.21   Mean   :11.31
##  3rd Qu.:123.00   3rd Qu.: 14.9000   3rd Qu.:  12.90   3rd Qu.:18.00
##  Max.   :162.00   Max.   : 60.3000   Max.   : 179.00   Max.   :29.00
##   gyros_belt_x        gyros_belt_y       gyros_belt_z
##  Min.   :-1.040000   Min.   :-0.64000   Min.   :-1.4600
##  1st Qu.:-0.030000   1st Qu.: 0.00000   1st Qu.:-0.2000
##  Median : 0.030000   Median : 0.02000   Median :-0.1000
##  Mean   :-0.005592   Mean   : 0.03959   Mean   :-0.1305
##  3rd Qu.: 0.110000   3rd Qu.: 0.11000   3rd Qu.:-0.0200
##  Max.   : 2.220000   Max.   : 0.64000   Max.   : 1.6200
##   accel_belt_x       accel_belt_y     accel_belt_z     magnet_belt_x
##  Min.   :-120.000   Min.   :-69.00   Min.   :-275.00   Min.   :-52.0
##  1st Qu.: -21.000   1st Qu.:  3.00   1st Qu.:-162.00   1st Qu.:  9.0
##  Median : -15.000   Median : 35.00   Median :-152.00   Median : 35.0
##  Mean   :  -5.595   Mean   : 30.15   Mean   : -72.59   Mean   : 55.6
##  3rd Qu.:  -5.000   3rd Qu.: 61.00   3rd Qu.:  27.00   3rd Qu.: 59.0
##  Max.   :  85.000   Max.   :164.00   Max.   : 105.00   Max.   :485.0
##  magnet_belt_y   magnet_belt_z       roll_arm         pitch_arm
##  Min.   :354.0   Min.   :-623.0   Min.   :-180.00   Min.   :-88.800
##  1st Qu.:581.0   1st Qu.:-375.0   1st Qu.: -31.77   1st Qu.:-25.900
##  Median :601.0   Median :-320.0   Median :   0.00   Median :  0.000
##  Mean   :593.7   Mean   :-345.5   Mean   :  17.83   Mean   : -4.612
##  3rd Qu.:610.0   3rd Qu.:-306.0   3rd Qu.:  77.30   3rd Qu.: 11.200
##  Max.   :673.0   Max.   : 293.0   Max.   : 180.00   Max.   : 88.500
##     yaw_arm          total_accel_arm  gyros_arm_x        gyros_arm_y
##  Min.   :-180.0000   Min.   : 1.00   Min.   :-6.37000   Min.   :-3.4400
##  1st Qu.: -43.1000   1st Qu.:17.00   1st Qu.:-1.33000   1st Qu.:-0.8000
##  Median :   0.0000   Median :27.00   Median : 0.08000   Median :-0.2400
##  Mean   :  -0.6188   Mean   :25.51   Mean   : 0.04277   Mean   :-0.2571
##  3rd Qu.:  45.8750   3rd Qu.:33.00   3rd Qu.: 1.57000   3rd Qu.: 0.1400
##  Max.   : 180.0000   Max.   :66.00   Max.   : 4.87000   Max.   : 2.8400
##   gyros_arm_z       accel_arm_x       accel_arm_y      accel_arm_z
##  Min.   :-2.3300   Min.   :-404.00   Min.   :-318.0   Min.   :-636.00
##  1st Qu.:-0.0700   1st Qu.:-242.00   1st Qu.: -54.0   1st Qu.:-143.00
##  Median : 0.2300   Median : -44.00   Median :  14.0   Median : -47.00
##  Mean   : 0.2695   Mean   : -60.24   Mean   :  32.6   Mean   : -71.25
##  3rd Qu.: 0.7200   3rd Qu.:  84.00   3rd Qu.: 139.0   3rd Qu.:  23.00
##  Max.   : 3.0200   Max.   : 437.00   Max.   : 308.0   Max.   : 292.00
##   magnet_arm_x     magnet_arm_y     magnet_arm_z    roll_dumbbell
##  Min.   :-584.0   Min.   :-392.0   Min.   :-597.0   Min.   :-153.71
##  1st Qu.:-300.0   1st Qu.:  -9.0   1st Qu.: 131.2   1st Qu.: -18.49
##  Median : 289.0   Median : 202.0   Median : 444.0   Median :  48.17
##  Mean   : 191.7   Mean   : 156.6   Mean   : 306.5   Mean   :  23.84
##  3rd Qu.: 637.0   3rd Qu.: 323.0   3rd Qu.: 545.0   3rd Qu.:  67.61
```

```
##  Max.   : 782.0   Max.   : 583.0   Max.   : 694.0   Max.   : 153.55
##  pitch_dumbbell     yaw_dumbbell      total_accel_dumbbell
##  Min.   :-149.59   Min.   :-150.871   Min.   : 0.00
##  1st Qu.: -40.89   1st Qu.: -77.644   1st Qu.: 4.00
##  Median : -20.96   Median :  -3.324   Median :10.00
##  Mean   : -10.78   Mean   :   1.674   Mean   :13.72
##  3rd Qu.:  17.50   3rd Qu.:  79.643   3rd Qu.:19.00
##  Max.   : 149.40   Max.   : 154.952   Max.   :58.00
##  gyros_dumbbell_x    gyros_dumbbell_y   gyros_dumbbell_z
##  Min.   :-204.0000   Min.   :-2.10000   Min.   : -2.380
##  1st Qu.:  -0.0300   1st Qu.:-0.14000   1st Qu.: -0.310
##  Median :   0.1300   Median : 0.03000   Median : -0.130
##  Mean   :   0.1611   Mean   : 0.04606   Mean   : -0.129
##  3rd Qu.:   0.3500   3rd Qu.: 0.21000   3rd Qu.:  0.030
##  Max.   :   2.2200   Max.   :52.00000   Max.   :317.000
##  accel_dumbbell_x  accel_dumbbell_y  accel_dumbbell_z  magnet_dumbbell_x
##  Min.   :-419.00   Min.   :-189.00   Min.   :-334.00   Min.   :-643.0
##  1st Qu.: -50.00   1st Qu.:  -8.00   1st Qu.:-142.00   1st Qu.:-535.0
##  Median :  -8.00   Median :  41.50   Median :  -1.00   Median :-479.0
##  Mean   : -28.62   Mean   :  52.63   Mean   : -38.32   Mean   :-328.5
##  3rd Qu.:  11.00   3rd Qu.: 111.00   3rd Qu.:  38.00   3rd Qu.:-304.0
##  Max.   : 235.00   Max.   : 315.00   Max.   : 318.00   Max.   : 592.0
##  magnet_dumbbell_y magnet_dumbbell_z  roll_forearm       pitch_forearm
##  Min.   :-3600     Min.   :-262.00   Min.   :-180.0000   Min.   :-72.50
##  1st Qu.:  231     1st Qu.: -45.00   1st Qu.:  -0.7375   1st Qu.:  0.00
##  Median :  311     Median :  13.00   Median :  21.7000   Median :  9.24
##  Mean   :  221     Mean   :  46.05   Mean   :  33.8265   Mean   : 10.71
##  3rd Qu.:  390     3rd Qu.:  95.00   3rd Qu.: 140.0000   3rd Qu.: 28.40
##  Max.   :  633     Max.   : 452.00   Max.   : 180.0000   Max.   : 89.80
##   yaw_forearm      total_accel_forearm gyros_forearm_x
##  Min.   :-180.00   Min.   :  0.00      Min.   :-22.000
##  1st Qu.: -68.60   1st Qu.: 29.00      1st Qu.: -0.220
##  Median :   0.00   Median : 36.00      Median :  0.050
##  Mean   :  19.21   Mean   : 34.72      Mean   :  0.158
##  3rd Qu.: 110.00   3rd Qu.: 41.00      3rd Qu.:  0.560
##  Max.   : 180.00   Max.   :108.00      Max.   :  3.970
##  gyros_forearm_y     gyros_forearm_z    accel_forearm_x   accel_forearm_y
##  Min.   : -7.02000   Min.   : -8.0900   Min.   :-498.00   Min.   :-632.0
##  1st Qu.: -1.46000   1st Qu.: -0.1800   1st Qu.:-178.00   1st Qu.:  57.0
##  Median :  0.03000   Median :  0.0800   Median : -57.00   Median : 201.0
##  Mean   :  0.07517   Mean   :  0.1512   Mean   : -61.65   Mean   : 163.7
##  3rd Qu.:  1.62000   3rd Qu.:  0.4900   3rd Qu.:  76.00   3rd Qu.: 312.0
##  Max.   :311.00000   Max.   :231.0000   Max.   : 477.00   Max.   : 923.0
##  accel_forearm_z   magnet_forearm_x  magnet_forearm_y magnet_forearm_z
##  Min.   :-446.00   Min.   :-1280.0   Min.   :-896.0   Min.   :-973.0
##  1st Qu.:-182.00   1st Qu.: -616.0   1st Qu.:   2.0   1st Qu.: 191.0
##  Median : -39.00   Median : -378.0   Median : 591.0   Median : 511.0
##  Mean   : -55.29   Mean   : -312.6   Mean   : 380.1   Mean   : 393.6
##  3rd Qu.:  26.00   3rd Qu.:  -73.0   3rd Qu.: 737.0   3rd Qu.: 653.0
##  Max.   : 291.00   Max.   :  672.0   Max.   :1480.0   Max.   :1090.0
##  classe
##  A:5580
##  B:3797
##  C:3422
```

```
##  D:3216
##  E:3607
##
```

Step 3: Data Slicing

In this case, 75% of the data will be used for model training and the remainder will be used for evaluating model performance. The function creates the random splits within each classe so that the overall classe distribution is preserved as well as possible.

```
set.seed(2018)

# Create a building data set and validation set
inTrain <- createDataPartition(y=pmlData$classe,
                               p=.75, list=FALSE)
training <- pmlData[inTrain, ]
testing <- pmlData[-inTrain, ]
```

Step 4: Fitting Models

Three algorithms (Generalized Boosted Model, Support Vector Machine, and Random Forests) are applied to categorize classe based on participants' exercise in the training set. Confusion matrix below characterizes the various summaries for fitting model's performance.

```
### Model 1: Generalized Boosted Model
gbmFit <- train(classe ~ ., data=training, method="gbm",
                trControl=trainControl(method='cv', number=5),
                verbose = FALSE)
pred_gbm <- predict(gbmFit, newdata=testing)
confusionMatrix(pred_gbm, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1375   26    0    1    3
##          B   11  909   23    1    5
##          C    5   11  826   30   14
##          D    2    2    5  769   15
##          E    2    1    1    3  864
##
## Overall Statistics
##
##                Accuracy : 0.9672
##                  95% CI : (0.9618, 0.972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9585
##  Mcnemar's Test P-Value : 2.076e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9857   0.9579   0.9661   0.9565   0.9589
```

```
## Specificity            0.9915   0.9899   0.9852   0.9941   0.9983
## Pos Pred Value         0.9786   0.9579   0.9323   0.9697   0.9920
## Neg Pred Value         0.9943   0.9899   0.9928   0.9915   0.9908
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2804   0.1854   0.1684   0.1568   0.1762
## Detection Prevalence   0.2865   0.1935   0.1807   0.1617   0.1776
## Balanced Accuracy      0.9886   0.9739   0.9756   0.9753   0.9786
```

```
### Model 2: Support Vector Machine
svmFit <- train(classe ~ ., data=training, method="svmRadial",
                trControl=trainControl(method='cv', number=5),
                preProcess=c("center", "scale"),
                allowParallel=TRUE)
pred_svm <- predict(svmFit, newdata=testing)
confusionMatrix(pred_svm, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1386   86    6    3    1
##          B    3  835   34    0    9
##          C    5   25  807   91   40
##          D    1    1    7  710   26
##          E    0    2    1    0  825
##
## Overall Statistics
##
##                Accuracy : 0.9305
##                  95% CI : (0.923, 0.9374)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9119
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9935   0.8799   0.9439   0.8831   0.9156
## Specificity            0.9726   0.9884   0.9602   0.9915   0.9993
## Pos Pred Value         0.9352   0.9478   0.8337   0.9530   0.9964
## Neg Pred Value         0.9974   0.9717   0.9878   0.9774   0.9814
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2826   0.1703   0.1646   0.1448   0.1682
## Detection Prevalence   0.3022   0.1796   0.1974   0.1519   0.1688
## Balanced Accuracy      0.9831   0.9341   0.9520   0.9373   0.9574
```

```
### Model 3: Random Forest
rfFit <- train(classe ~ ., data=training, method="rf",
                trControl=trainControl(method='cv', number=5),
                allowParallel=TRUE)
pred_rf <- predict(rfFit, newdata=testing)
confusionMatrix(pred_rf, testing$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1391    9    0    0    0
##          B    4  940    2    0    1
##          C    0    0  852    8    1
##          D    0    0    1  796    1
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9945
##                  95% CI : (0.992, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.993
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9971   0.9905   0.9965   0.9900   0.9967
## Specificity            0.9974   0.9982   0.9978   0.9995   1.0000
## Pos Pred Value         0.9936   0.9926   0.9895   0.9975   1.0000
## Neg Pred Value         0.9989   0.9977   0.9993   0.9981   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2836   0.1917   0.1737   0.1623   0.1831
## Detection Prevalence   0.2855   0.1931   0.1756   0.1627   0.1831
## Balanced Accuracy      0.9973   0.9944   0.9971   0.9948   0.9983
```

All three models got over 90% accuracy when applied to the testing dataset. Among all, the Random Forest model seems to be performing the best. Next let's evaluate all three models collectively.

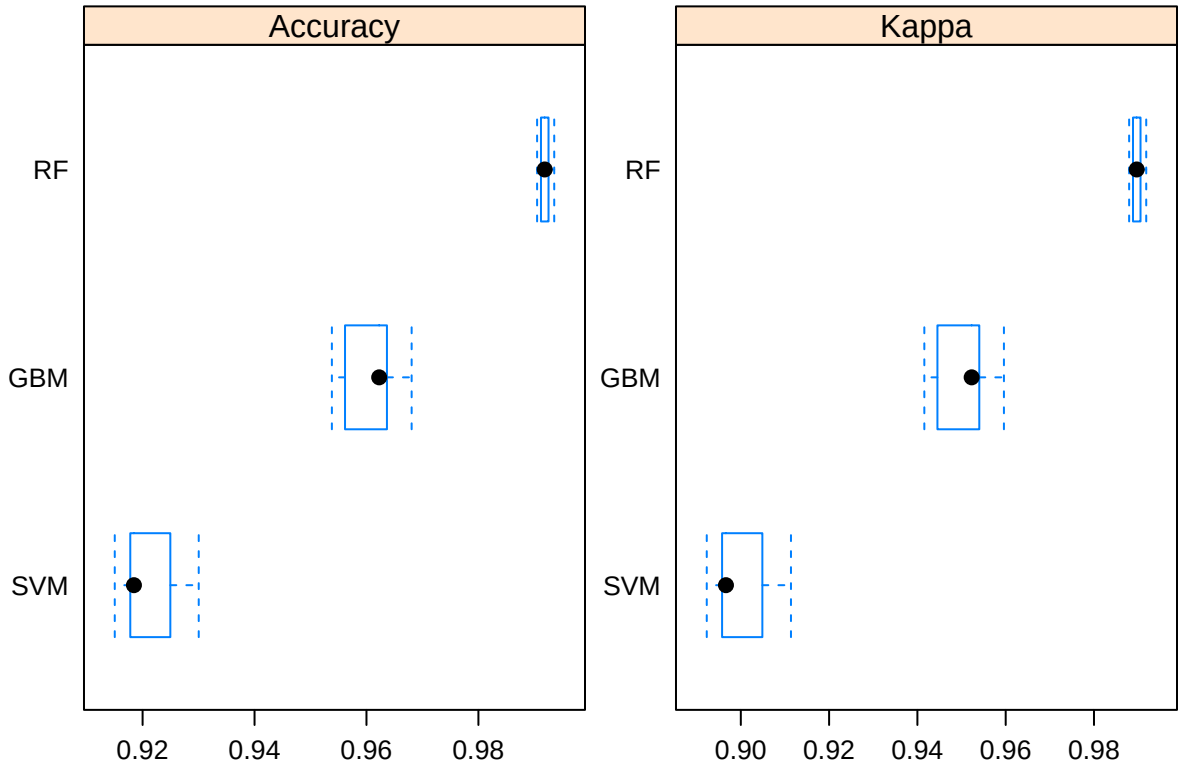
```
# Compare model performances using resample()
models_compare <- resamples(list(GBM=gbmFit, SVM=svmFit, RF=rfFit))

# Summary of the models performances
summary(models_compare)
```

```
##
## Call:
## summary.resamples(object = models_compare)
##
## Models: GBM, SVM, RF
## Number of resamples: 5
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GBM 0.9538200 0.9561672 0.9622706 0.9607967 0.9636549 0.9680707    0
## SVM 0.9150238 0.9177989 0.9184506 0.9212516 0.9249576 0.9300272    0
## RF  0.9904859 0.9911745 0.9918423 0.9919147 0.9925246 0.9935462    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
## GBM 0.9415721 0.9445333 0.9522897 0.9504039 0.9540200 0.9596044    0
## SVM 0.8922813 0.8957570 0.8966332 0.9001821 0.9048802 0.9113587    0
## RF  0.9879629 0.9888373 0.9896784 0.9897716 0.9905426 0.9918368    0
```

```
# Draw box plots to compare models
scales <- list(x=list(relation="free"), y=list(relation="free"))
bwplot(models_compare, scales=scales)
```



As the plot suggested, the Random Forest model has the best overall performance due to its high accuracy and kappa.

Step 5: Apply Selected Model to Validation Data

Lastly, applying the Random Forest model to validation dataset.

```
pred_rf_v <- predict(rfFit, newdata=validation)
pred_rf_v
```

```
##  [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
```