# Fitness Prediction Assignment

## Background

The goal of this report is to predict how well people do a particular activity. Data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants was used to build the prediction model.

## Analysis

### Data preprocessing

First, load the caret and ggplot2 packages in R, and read the training and testing files.

```
library(caret)
```

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

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

```
library(ggplot2)
```

```
finaltest <- read.csv(file = "./pml-testing.csv",header = TRUE)
traindata <- read.csv(file = "./pml-training.csv",header = TRUE)
```

Then, clean up the data before building the model.It includes removing variables with all NA values, constant and near zero variables, and variables that have high correlations.

```
na <- apply(is.na(traindata),2,sum)
traindata2 <- traindata[,!na==19216]

zerovar <- nearZeroVar(traindata2)
newdata1 <- traindata2[,-zerovar]

deCorr <-  cor(newdata1[,6:58])
highCorr <- findCorrelation(deCorr,cutoff=0.9,exact = FALSE)
newdata2 <-  newdata1[, -highCorr]
newdata3 <- newdata2[,6:52]
```

### Cross validation

Use caret package to split the data to the training and the testing set.

```
set.seed(123456)
inTrain <- createDataPartition(y=newdata3$classe,p=0.7,list=FALSE)
training <- newdata3[inTrain,]
testing <- newdata3[-inTrain,]
```

# Build the model

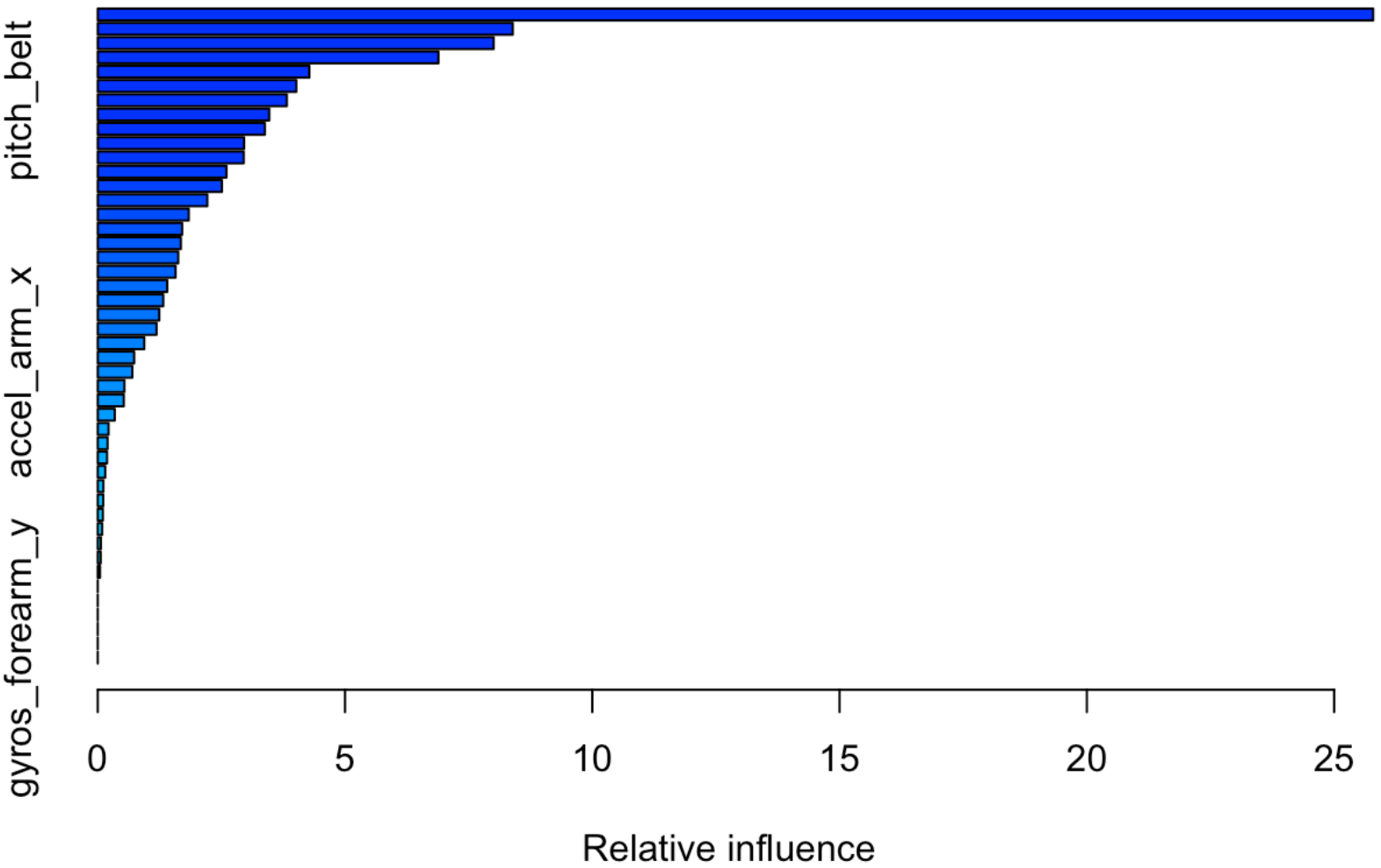
Use gradient boosting method to build up the model.

```
set.seed(123456)
fitall<- train(classe~.,method="gbm",verbose=FALSE,na.action = na.omit,data=traini
ng)

fitall
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    46 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7177215  0.6400952
##   1                  100      0.7896509  0.7328423
##   1                  150      0.8318507  0.7866375
##   2                   50      0.8402195  0.7972603
##   2                  100      0.8929296  0.8644165
##   2                  150      0.9185761  0.8969493
##   3                   50      0.8835112  0.8524087
##   3                  100      0.9300122  0.9114448
##   3                  150      0.9487861  0.9352117
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
summary(fitall)
```


pitch_belt
accel_arm_x
gyros_forearm_y
0
5
10
15
20
25
Relative influence

```
##                                      var     rel.inf
## roll_belt                       roll_belt 25.78926628
## roll_forearm                 roll_forearm  8.39125049
## magnet_dumbbell_y       magnet_dumbbell_y  8.00626142
## magnet_dumbbell_z       magnet_dumbbell_z  6.88915395
## accel_forearm_x           accel_forearm_x  4.27810927
## magnet_belt_z               magnet_belt_z  4.01502133
## pitch_belt                     pitch_belt  3.82262519
## magnet_forearm_z         magnet_forearm_z  3.46824088
## roll_dumbbell               roll_dumbbell  3.37897800
## gyros_belt_z                 gyros_belt_z  2.95712369
## magnet_arm_x                 magnet_arm_x  2.95170690
## accel_dumbbell_y         accel_dumbbell_y  2.60215844
## yaw_arm                           yaw_arm  2.51030666
## gyros_dumbbell_y         gyros_dumbbell_y  2.21665595
## accel_dumbbell_z         accel_dumbbell_z  1.84060656
## magnet_dumbbell_x       magnet_dumbbell_x  1.70722005
## accel_forearm_z           accel_forearm_z  1.68085864
## accel_dumbbell_x         accel_dumbbell_x  1.62853706
## magnet_belt_y               magnet_belt_y  1.57419283
## pitch_arm                       pitch_arm  1.40457919
## magnet_forearm_x         magnet_forearm_x  1.32517609
## magnet_belt_x               magnet_belt_x  1.24389303
## magnet_arm_y                 magnet_arm_y  1.18929221
## total_accel_dumbbell total_accel_dumbbell  0.94016130
## gyros_arm_y                   gyros_arm_y  0.73673677
## accel_arm_x                   accel_arm_x  0.70035535
## gyros_belt_y                 gyros_belt_y  0.53975078
## gyros_dumbbell_x         gyros_dumbbell_x  0.52667619
## gyros_forearm_z           gyros_forearm_z  0.34325568
## gyros_dumbbell_z         gyros_dumbbell_z  0.21799134
## accel_arm_z                   accel_arm_z  0.19849137
## total_accel_forearm   total_accel_forearm  0.18864001
## yaw_forearm                   yaw_forearm  0.15120635
## accel_belt_z                 accel_belt_z  0.11399101
## gyros_arm_x                   gyros_arm_x  0.11127082
## magnet_forearm_y         magnet_forearm_y  0.10339630
## accel_forearm_y           accel_forearm_y  0.08890921
## accel_belt_y                 accel_belt_y  0.06267026
## pitch_dumbbell             pitch_dumbbell  0.05872902
## gyros_forearm_x           gyros_forearm_x  0.04655412
## accel_belt_x                 accel_belt_x  0.00000000
## total_accel_arm           total_accel_arm  0.00000000
## gyros_arm_z                   gyros_arm_z  0.00000000
## accel_arm_y                   accel_arm_y  0.00000000
## yaw_dumbbell                 yaw_dumbbell  0.00000000
## gyros_forearm_y           gyros_forearm_y  0.00000000
```

As is shown, the final model's accuracy is 0.95 and kappa is 0.94. So the model fits quite well.Among these variables, roll_belt has the largest influence on the model.

# Out of sample error

Apply the model to the testing set, and compare the result from prediction and actual classe.

```
predictionall <- predict(fitall,testing[,-52])
confusionMatrix(predictionall, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1647   50    1    1    0
##          B    9 1042   38    5    9
##          C   12   40  977   48   16
##          D    4    3   10  907   20
##          E    2    4    0    3 1037
##
## Overall Statistics
##
##                Accuracy : 0.9533
##                  95% CI : (0.9476, 0.9585)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9409
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9839   0.9148   0.9522   0.9409   0.9584
## Specificity            0.9877   0.9871   0.9761   0.9925   0.9981
## Pos Pred Value         0.9694   0.9447   0.8939   0.9608   0.9914
## Neg Pred Value         0.9935   0.9797   0.9898   0.9885   0.9907
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2799   0.1771   0.1660   0.1541   0.1762
## Detection Prevalence   0.2887   0.1874   0.1857   0.1604   0.1777
## Balanced Accuracy      0.9858   0.9510   0.9642   0.9667   0.9783
```

As is shown, the prediction accuracy is 0.96, which is a bit higher than the in sample accuracy. In general, this model fits well.

# Prediction

Finally, predict the 20 different test cases.

```
finalpredict <- predict(fitall,finaltest)
finalpredict
```

```
##  [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
```