Prediction Assignment Writeup

Andrew Golus

November 12, 2017

Introduction

The goal of this project is to use data from accelerometers on the belt, forearm, arm and dumbell of six participants and predict the manner in which they did the exercise.

Prepare Data Set

The attached R code achieves the following: - Loads the pml-training and pml-testing data - Eliminates 100 columns with NA values only in pml_testing from both data sets - Selects 52 predictors (continuous variables) and one response (categorical Variables)

```
pml_training <- read.csv("pml-training.csv")
pml_testing <- read.csv("pml-testing.csv")
na_perc <-sapply(pml_testing, function(x)
sum(length(which(is.na(x))))/length(x))
table(na_perc)
```

```
## na_perc
##   0   1
##  60 100
```

```
features <- cbind(1:160, na_perc)
features <- subset(features, na_perc == 0)[,1]
pml_training <- select(pml_training, features)
names(pml_training)[60] <- "classe"
pml_testing <- select(pml_testing, features)
pml_training <- select(pml_training, 8:60)
pml_testing <- select(pml_testing, 8:60)
```

Split Data

The attached R code achieves the following: - Sets seed to guarantee reproducibility of analysis - Splits the pml_training data into t_training (70% of observations) and t_testing (30% of observations) data sets on 'Classe' - pml_testing will serve as a validation set

```
set.seed(7777)
inTrain <- createDataPartition(y = pml_training$classe, p = 0.7, list = FALSE)
t_training <- pml_training[inTrain,]
t_testing <- pml_training[-inTrain,]
```

Preprocessing

The attached R code achieves the following: - Checks for near zero variance variables - Checks for highly correlated variables - Runs a principal component analysis (pca) on the t_training data set as there are highly correlated variables - Applies the pca to the training, testing and validation sets (pml_testing)

```
nzv <- nearZeroVar(t_training, saveMetrics = TRUE)
nzv
```

```
##                       freqRatio percentUnique zeroVar   nzv
## roll_belt              1.083728    8.01485040   FALSE FALSE
## pitch_belt             1.097015   12.36805707   FALSE FALSE
## yaw_belt               1.108571   13.10329766   FALSE FALSE
## total_accel_belt       1.063486    0.21110868   FALSE FALSE
## gyros_belt_x           1.064963    0.90995123   FALSE FALSE
## gyros_belt_y           1.125258    0.48045425   FALSE FALSE
## gyros_belt_z           1.071197    1.20113562   FALSE FALSE
## accel_belt_x           1.083799    1.15745796   FALSE FALSE
## accel_belt_y           1.141075    1.00458615   FALSE FALSE
## accel_belt_z           1.107794    2.08924802   FALSE FALSE
## magnet_belt_x          1.131148    2.18388294   FALSE FALSE
## magnet_belt_y          1.133333    2.10380724   FALSE FALSE
## magnet_belt_z          1.050157    3.19574871   FALSE FALSE
## roll_arm              50.468085   17.51474121   FALSE FALSE
## pitch_arm             79.100000   20.45570357   FALSE FALSE
## yaw_arm               34.376812   19.18177186   FALSE FALSE
## total_accel_arm        1.054662    0.47317464   FALSE FALSE
## gyros_arm_x            1.094395    4.59343379   FALSE FALSE
## gyros_arm_y            1.574850    2.67161680   FALSE FALSE
## gyros_arm_z            1.112637    1.71798792   FALSE FALSE
## accel_arm_x            1.083333    5.60529956   FALSE FALSE
## accel_arm_y            1.220779    3.83635437   FALSE FALSE
## accel_arm_z            1.088889    5.55434229   FALSE FALSE
## magnet_arm_x           1.031746    9.64548300   FALSE FALSE
## magnet_arm_y           1.034483    6.20222756   FALSE FALSE
## magnet_arm_z           1.040000    9.08495305   FALSE FALSE
## roll_dumbbell          1.056818   86.96949843   FALSE FALSE
## pitch_dumbbell         2.215054   84.74193783   FALSE FALSE
## yaw_dumbbell           1.094118   86.46720536   FALSE FALSE
## total_accel_dumbbell  1.053553    0.30574361   FALSE FALSE
## gyros_dumbbell_x       1.012107    1.69614909   FALSE FALSE
## gyros_dumbbell_y       1.272727    1.95821504   FALSE FALSE
## gyros_dumbbell_z       1.126551    1.38312586   FALSE FALSE
## accel_dumbbell_x       1.073276    3.01375846   FALSE FALSE
## accel_dumbbell_y       1.047904    3.28310403   FALSE FALSE
## accel_dumbbell_z       1.163636    2.90456432   FALSE FALSE
## magnet_dumbbell_x      1.065041    7.82558055   FALSE FALSE
## magnet_dumbbell_y      1.218487    5.99111888   FALSE FALSE
## magnet_dumbbell_z      1.029851    4.80454248   FALSE FALSE
## roll_forearm          11.835498   13.54735386   FALSE FALSE
## pitch_forearm         70.076923   19.11625537   FALSE FALSE
## yaw_forearm           15.093923   13.00866274   FALSE FALSE
## total_accel_forearm    1.109481    0.47317464   FALSE FALSE
## gyros_forearm_x        1.074380    2.01645192   FALSE FALSE
## gyros_forearm_y        1.104247    5.22675985   FALSE FALSE
## gyros_forearm_z        1.196141    2.09652763   FALSE FALSE
## accel_forearm_x        1.220339    5.67809565   FALSE FALSE
## accel_forearm_y        1.123077    7.07578074   FALSE FALSE
## accel_forearm_z        1.017857    4.04018345   FALSE FALSE
## magnet_forearm_x       1.071429   10.48263813   FALSE FALSE
## magnet_forearm_y       1.084746   13.36536362   FALSE FALSE
## magnet_forearm_z       1.069767   11.79296790   FALSE FALSE
## classe                 1.469526    0.03639805   FALSE FALSE
```

```
M <- abs(cor(t_training[,-53]))
diag(M) <- 0
which(M > 0.8, arr.ind = TRUE)
```

```
##                   row col
## yaw_belt            3   1
## total_accel_belt    4   1
## accel_belt_y        9   1
## accel_belt_z       10   1
## accel_belt_x        8   2
## magnet_belt_x      11   2
## roll_belt           1   3
## roll_belt           1   4
## accel_belt_y        9   4
## accel_belt_z       10   4
## pitch_belt          2   8
## magnet_belt_x      11   8
## roll_belt           1   9
## total_accel_belt    4   9
## accel_belt_z       10   9
## roll_belt           1  10
## total_accel_belt    4  10
## accel_belt_y        9  10
## pitch_belt          2  11
## accel_belt_x        8  11
## gyros_arm_y        19  18
## gyros_arm_x        18  19
## magnet_arm_x       24  21
## accel_arm_x        21  24
## magnet_arm_z       26  25
## magnet_arm_y       25  26
## accel_dumbbell_x   34  28
## accel_dumbbell_z   36  29
## pitch_dumbbell     28  34
## yaw_dumbbell       29  36
```

```
preProc <- preProcess(t_training[,-53], method = "pca", thres = 0.8)
t_training <- predict(preProc, t_training)
t_testing <- predict(preProc, t_testing)
pml_testing <- predict(preProc, pml_testing)
```

Train prediction algorithms

The attached R code achieves the following: - Sets seed to guarantee reproducibility of analysis - Sets cross validation with 10 sets to prevent high variance - Trains prediction algorithm on the t_training data set using three different models while centering and scaling the data - Predicts the ‘classe’ variable in the t_testing data set using the three algorithms - Inspects the accuracy of the preditions (see Appendix)

```
set.seed(8888)
ctrl <- trainControl(method = "cv", number = 10)
fit_lda <- train(classe ~ ., data = t_training, method = "lda", preProc = c("center", "scale"), trControl = ctr
l)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
pred_lda <- predict(fit_lda, t_testing)
result_lda <- confusionMatrix(pred_lda, t_testing$classe)
fit_rpart <- train(classe ~ ., data = t_training, method = "rpart", preProc = c("center", "scale"), trControl =
 ctrl)
pred_rpart <- predict(fit_rpart, t_testing)
result_rpart <- confusionMatrix(pred_rpart, t_testing$classe)
fit_rf <- train(classe ~ ., data = t_training, method = "rf")
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-12
```

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

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

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

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
pred_rf <- predict(fit_rf, t_testing)
result_rf <- confusionMatrix(pred_rf, t_testing$classe)
```

Conclusions

Of the three models trained and tested, the random forest has highest accuracy, with an estimated out-of-sample error at 96%.

```
predict(fit_rf, pml_testing)
```

```
##  [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 - Confusion Matrix and Statistics of the trained model on test data

Random Forest - Selected

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1624   18   16   11    4
##          B   11 1093   18    1    9
##          C   24   27  975   40    3
##          D   14    1   15  908    7
##          E    1    0    2    4 1059
##
## Overall Statistics
##
##                Accuracy : 0.9616
##                  95% CI : (0.9564, 0.9664)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9514
##  Mcnemar's Test P-Value : 0.001429
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9701   0.9596   0.9503   0.9419   0.9787
## Specificity            0.9884   0.9918   0.9807   0.9925   0.9985
## Pos Pred Value         0.9707   0.9655   0.9121   0.9608   0.9934
## Neg Pred Value         0.9881   0.9903   0.9894   0.9887   0.9952
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2760   0.1857   0.1657   0.1543   0.1799
## Detection Prevalence   0.2843   0.1924   0.1816   0.1606   0.1811
## Balanced Accuracy      0.9792   0.9757   0.9655   0.9672   0.9886
```

Linear Discriminant Analysis

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1141  303  444  110  160
##          B   95  380   82  207  190
##          C  150  151  342  120  178
##          D  209  159   78  423   82
##          E   79  146   80  104  472
##
## Overall Statistics
##
##                Accuracy : 0.4686
##                  95% CI : (0.4558, 0.4815)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3209
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6816  0.33363  0.33333  0.43880   0.4362
## Specificity            0.7585  0.87906  0.87672  0.89270   0.9148
## Pos Pred Value         0.5287  0.39832  0.36344  0.44479   0.5358
## Neg Pred Value         0.8570  0.84608  0.86165  0.89035   0.8781
## Prevalence             0.2845  0.19354  0.17434  0.16381   0.1839
## Detection Rate         0.1939  0.06457  0.05811  0.07188   0.0802
## Detection Prevalence   0.3667  0.16211  0.15990  0.16160   0.1497
## Balanced Accuracy      0.7200  0.60634  0.60503  0.66575   0.6755
```

Classification Tree

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1450  700  817  394  567
##          B    0    0    0    0    0
##          C    0    0    0    0    0
##          D  147  309  127  419  198
##          E   77  130   82  151  317
##
## Overall Statistics
##
##                Accuracy : 0.3715
##                  95% CI : (0.3591, 0.3839)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1654
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8662   0.0000   0.0000   0.4346  0.29298
## Specificity            0.4115   1.0000   1.0000   0.8413  0.90839
## Pos Pred Value         0.3691      NaN      NaN   0.3492  0.41876
## Neg Pred Value         0.8855   0.8065   0.8257   0.8837  0.85082
## Prevalence             0.2845   0.1935   0.1743   0.1638  0.18386
## Detection Rate         0.2464   0.0000   0.0000   0.0712  0.05387
## Detection Prevalence   0.6675   0.0000   0.0000   0.2039  0.12863
## Balanced Accuracy      0.6389   0.5000   0.5000   0.6380  0.60068
```