Human Activity Recognition Study

Synopsis

Participants in study were asked to perform different excercises with measurements taken using accelerometers on the belt, forearm, arm, and dumbell. These measurements are then used to predict what excercises are performed.

Data Processing

The data is read in and split the data into the training and testing sets using a 60% split for training based on the "classe", which is the variable we are trying to predict. Models will be trained on the training set and tested on the testing set in order to prevent overfitting. The holdout set with unknown "classe" which we will later try to predict is also brought in and held until later.

```
data <- read.table("pmltraining.csv", header = TRUE, sep = ',', quote = '\"', dec='.', na.strings = c('
test <- read.table("pmltesting.csv", header = TRUE, sep = ',', quote = '\"', dec = '.', na.strings = c(

test[sapply(test, is.logical)] <- lapply(test[sapply(test, is.logical)], as.numeric)

inTrain <- createDataPartition(y=data$classe, p=0.6, list = FALSE)
train <- data[inTrain,]
val <- data[-inTrain,]
```

Next data cleaning steps are performed. Variables with near zero variance are removed and BoxCox transformation and k-nearest neighbors imputation are applied. BoxCox will attempt to normalize the data for PCA and knnImpute will fill in missing values which could be troublesome for some algorithms. The same cleaning and preprocess objects from the training set is applied on the test and holdout set.

```
keepVar <- row.names(subset(nearZeroVar(train, saveMetrics = TRUE), nzv == FALSE))

cleanTrain <- select(train, one_of(keepVar))
cleanVal <- select(val, one_of(keepVar))
cleanTest <- bind_cols(select(test, one_of(keepVar[1:124])), classe = rep('', times = 20))
```

```
## Warning: Unknown columns: `classe`, `NA`
```

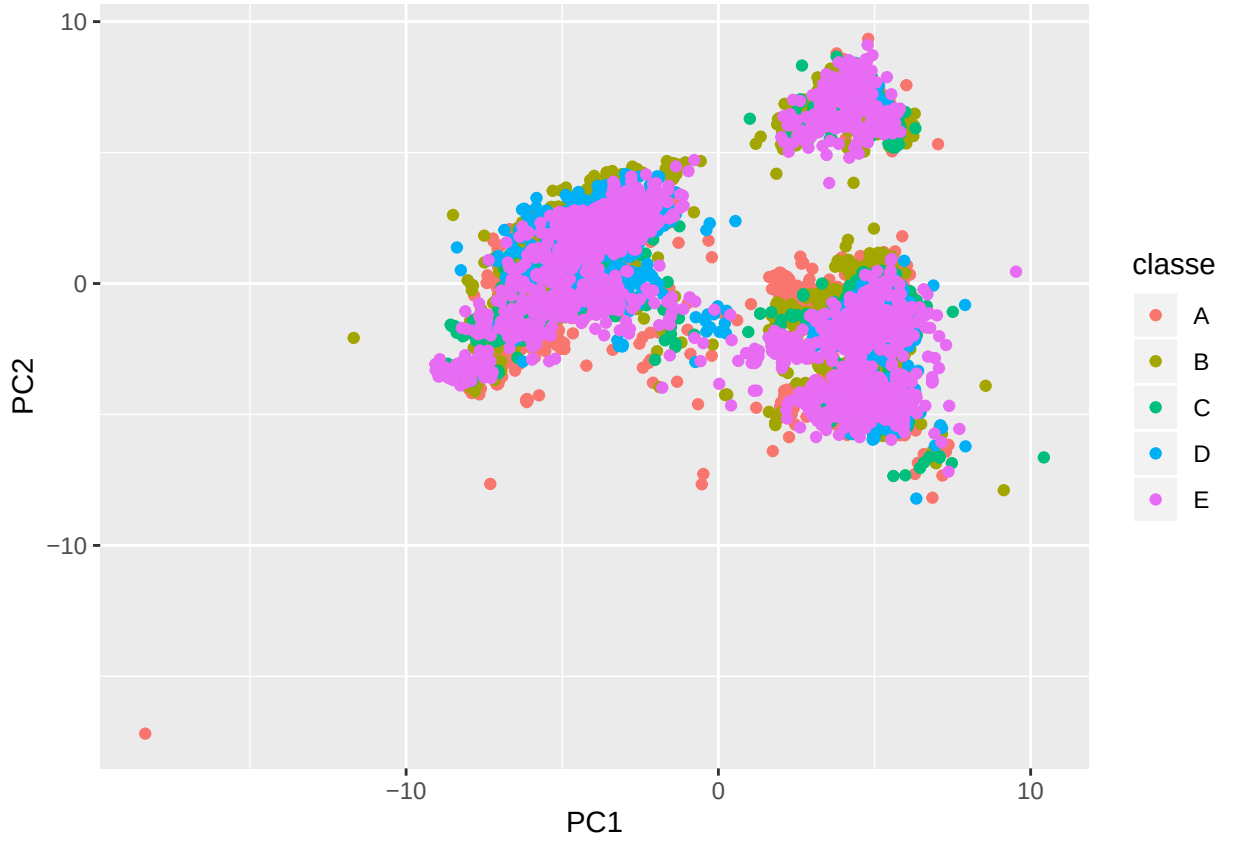
```
preProc1 <- preProcess(cleanTrain, method = c('BoxCox','knnImpute'))
trainPP1 <- predict(preProc1, cleanTrain)
valPP1 <- predict(preProc1, cleanVal)
testPP1 <- predict(preProc1, cleanTest)
```

PCA is performed. Since there are 150+ variables in the dataset, the selected model will likely overfit the data if the variables are used as is. PCA will create weighted combinations of the variables which will make overfitting less likely. Again the same PCA from the training set is applied on the test and holdout sets in order to get good out of sample accuracy metrics.

```
preProc2 <- preProcess(trainPP1, method = 'pca', thresh = 0.9)
trainPP2 <- predict(preProc2, trainPP1)
trainPP2 <- bind_cols(trainPP2[,-1], classe = train$classe)
valPP2 <- predict(preProc2, valPP1)
valPP2 <- bind_cols(valPP2[,-1], classe = val$classe)
testPP2 <- predict(preProc2, testPP1)
```

Plot shows the separation of the different "classe" by the first two principle components.

```
qplot(x=PC1, y=PC2, color=classe, data=trainPP2, geom="point")
```



Prediction

Random forest and boosting algorithms are applied on the training set. k-fold cross validation is used within the training set with 10 folds so that multiple models can be fitted then tested and the best candidate model chosen.

```
trainContrl <- trainControl(method = 'cv', number = 10)

modFitRF <- train(classe ~ ., method = 'rf', trControl = trainContrl, data = trainPP2)
modFitBo <- train(classe ~ ., method = 'gbm', trControl = trainContrl, data = trainPP2, verbose = FALSE)
```

The candidate random forest and boosting models are then applied on the testing set to determine the out of sample error. Comparison of the accuracy metrics indicates that the random forest model performs much better than the boosting model.

```
rfPredVal <- predict(modFitRF, valPP2)
boPredVal <- predict(modFitBo, valPP2)

confusionMatrix(val$classe, rfPredVal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2104   50   34   33   11
##          B   65 1366   55   13   19
##          C   40   44 1239   29   16
##          D   27   21   72 1149   17
##          E    9   24   20   14 1375
##
## Overall Statistics
##
##                Accuracy : 0.9219
##                  95% CI : (0.9157, 0.9277)
##     No Information Rate : 0.2861
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9012
##
##  Mcnemar's Test P-Value : 0.00378
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9372   0.9076   0.8725   0.9281   0.9562
## Specificity            0.9771   0.9760   0.9799   0.9793   0.9895
## Pos Pred Value         0.9427   0.8999   0.9057   0.8935   0.9535
## Neg Pred Value         0.9749   0.9780   0.9721   0.9864   0.9902
## Prevalence             0.2861   0.1918   0.1810   0.1578   0.1833
## Detection Rate         0.2682   0.1741   0.1579   0.1464   0.1752
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9572   0.9418   0.9262   0.9537   0.9729
```

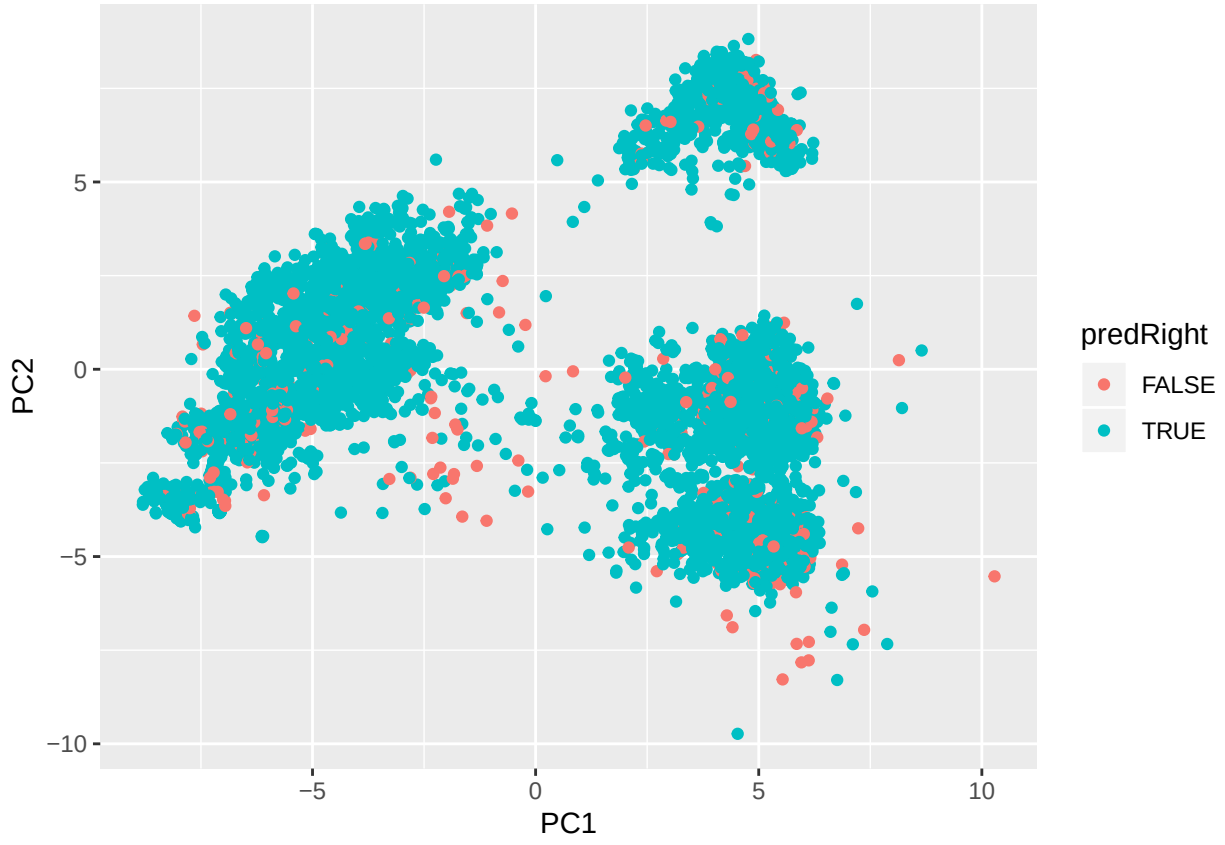
```
confusionMatrix(val$classe, boPredVal)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1820   98  153  116   45
##          B  176 1033  172   79   58
##          C  150   89 1017   66   46
##          D   74   59  141  971   41
##          E   75  158   96   66 1047
##
## Overall Statistics
##
##                Accuracy : 0.7504
```

```
##                  95% CI : (0.7407, 0.76)
##     No Information Rate : 0.2925
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6842
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7930   0.7189   0.6441   0.7481   0.8464
## Specificity            0.9258   0.9243   0.9440   0.9519   0.9402
## Pos Pred Value         0.8154   0.6805   0.7434   0.7551   0.7261
## Neg Pred Value         0.9154   0.9362   0.9132   0.9502   0.9703
## Prevalence             0.2925   0.1832   0.2012   0.1654   0.1577
## Detection Rate         0.2320   0.1317   0.1296   0.1238   0.1334
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8594   0.8216   0.7940   0.8500   0.8933
```

The below plot shows the spread of the random forest predictions by the first two principle components are whether the predictions are correct.

```
valPP2$predRight <- rfPredVal==valPP2$classe
qplot(PC1, PC2, colour=predRight, data=valPP2)
```



The final step attempts to predict the previosuly unknown "classe" on the 20 observation holdout set.

```
finalModFit <- predict(modFitRF, testPP2)
```

The predictions for the 20 observations are: B, A, A, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B. The out of sample error is estimated to be: