Course Project for Practical Machine Learning

This project reads in the training data from people who did the weight lifting excercise and try to predict if people are doing excercise correctly in the test data.

1. Load the training data. As we believe that information like the data index and the time of recording is not relevant to if people are doing actions correctly, we discard them. Additionally, there are a lot of features with mostly NA or blank values, we discard them as well. So we manually select the 54 features to use below.

```
rm(list=ls())
library(caret)
```

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

```
data <- read.csv('pml-training.csv')

# select the non-NaN features
colIndex <- c(7,8,9,10,11,37:49,60:68,84:86,102,113:124,140,151:160)
data <- data[,colIndex]
```

2. For training the model and cross validation, we randomly split the data into 70% training and 30% validation data.

```
# split the training data into training and validation for cross validation
set.seed(98123)
inTrain <- createDataPartition(y=data$classe, p=0.7, list=FALSE)
trainData <- data[inTrain,]
validationData <- data[-inTrain,]

dims <- dim(trainData)
col <- dims[2]
```

3. Perform PCA to reduce the dimension of the training data and apply PCA model on

```
# do PCA to reduce the dimension
preProc <- preProcess(trainData[,-col], method="pca",thresh=0.9)
trainPC <- predict(preProc, trainData[,-col])
validationPC <- predict(preProc, validationData[,-col])

# add the class label to train and test data
dims <- dim(trainPC)
colPC <- dims[2]

newTrainPC <- cbind(trainPC,trainData[,col])
names(newTrainPC)[colPC+1] <- "classe"
newValidationPC <- cbind(validationPC, validationData[,col])
names(newValidationPC)[colPC+1] <- "classe"
```

4. Prepare the test data according the preprocess steps done for training data:

```
# parepare test data according to training data
testData <- read.csv('pml-testing.csv')
testData <- testData[,colIndex]
testPC <- predict(preProc, testData[,-col])
```

5. Train the model on training data and validate the model on validation data.

```
modelFit <- train(classe ~., data=newTrainPC, method="rf")
```

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

```
predTrain <- predict(modelFit, newTrainPC)
predValidation <- predict(modelFit, newValidationPC)

confTrain <- confusionMatrix(trainData[,col], predTrain)
confValidation <- confusionMatrix(validationData[,col], predValidation)
```

The accuracy of the model on the training dataset is:

```
train_accuracy <- as.numeric(confTrain$overall["Accuracy"])
```

The in sample error is:

```
in_sample_error <- (1-train_accuracy)
in_sample_error
```

```
## [1] 0
```

The accuracy of the model on the validation dataset is:

```
validation_accuracy <- as.numeric(confValidation$overall["Accuracy"])
```

The out of sample error is expected at:

```
out_of_sample_error <- (1-validation_accuracy)
out_of_sample_error
```

```
## [1] 0.01852167
```

So we can see that the in sample error is lower than the expected out of sample error, which is normal for prediction models.

6. Finally we use the model to predict the test data:

```
predTest <- predict(modelFit, testPC)
predTest
```

```
##  [1] B A A A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

From the submission results, our model got 19 out of 20 test cases correct, yielding a 95% accuracy on test data. So the real out of sample error is bigger than the expectation.