MachineLearning_Week4_Project

```
setwd("~/Desktop")
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

Download Datasets

```
training<-read.csv(file="pml-training.csv")
testing<-read.csv(file="pml-testing.csv")
```

#Remove irrelevant variables for prediction

```
training<-training[,-(1:7)]
testing<-testing[,-(1:7)]
```

#Remove variables with too many missing datapoints

```
training[training==""]<- NA
naFraction<-colSums(is.na(training))/(dim(training)[1])
training<-training[,(naFraction<0.95)]
```

#Create a validation set

```r
inBuild<-createDataPartition(y=training$classe, p=0.7, list=FALSE)
validation<-training[-inBuild,]
buildData<-training[inBuild,]
```

#Use PCA to collapse number of variables to only the ones that are needed to differentiate between outcomes

```r
preProc<-preProcess(buildData,method="pca",thresh=.95)
trainPC<-predict(preProc,buildData)
```

#Use random forest method to create a model

```r
mod1<-randomForest(buildData$classe ~ .,data=trainPC,do.trace=F)
print(mod1)
```

```
##
## Call:
##  randomForest(formula = buildData$classe ~ ., data = trainPC,      do.trace = F)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 2.53%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3867   12   19    7    1 0.009984639
## B   49 2566   38    1    4 0.034612491
## C    4   34 2329   25    4 0.027963272
## D    3    1   95 2149    4 0.045737123
## E    0   11   19   16 2479 0.018217822
```

```r
importance(mod1)
```

```
##      MeanDecreaseGini
## PC1          572.1296
## PC2          466.3768
## PC3          504.9358
## PC4          374.7280
## PC5          555.5898
## PC6          435.1290
## PC7          383.0410
## PC8          733.7232
## PC9          537.8621
## PC10         383.2485
## PC11         344.0391
## PC12         578.6651
## PC13         361.5936
## PC14         687.4345
## PC15         472.3457
## PC16         432.2449
## PC17         423.6527
## PC18         297.3672
```

```
## PC19         333.5268
## PC20         357.2353
## PC21         392.4038
## PC22         402.0485
## PC23         234.7490
## PC24         252.8322
## PC25         343.1069
```

#Predict on the validation data

```
validation[validation==""]<- NA
naFractionTest<-colSums(is.na(validation))/(dim(validation)[1])
validation<-validation[,(naFractionTest<0.95)]

validPC<-predict(preProc,validation)
confusionMatrix(as.factor(validation$classe), predict(mod1,validPC))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1659    6    5    3    1
##          B   15 1109   14    1    0
##          C    0   14  996   12    4
##          D    2    0   47  912    3
##          E    0    1   18    3 1060
##
## Overall Statistics
##
##                Accuracy : 0.9747
##                  95% CI : (0.9703, 0.9785)
##     No Information Rate : 0.2848
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.968
##
##  Mcnemar's Test P-Value : 8.379e-06
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9899   0.9814   0.9222   0.9796   0.9925
## Specificity            0.9964   0.9937   0.9938   0.9895   0.9954
## Pos Pred Value         0.9910   0.9737   0.9708   0.9461   0.9797
## Neg Pred Value         0.9960   0.9956   0.9827   0.9961   0.9983
## Prevalence             0.2848   0.1920   0.1835   0.1582   0.1815
## Detection Rate         0.2819   0.1884   0.1692   0.1550   0.1801
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9931   0.9876   0.9580   0.9845   0.9940
```

#Predict classe of testing dataset using model created from training dataset

```
testing[testing==""]<- NA
naFractionFinal<-colSums(is.na(testing))/(dim(testing)[1])
testing<-testing[,(naFractionFinal<0.95)]

testPC <- predict(preProc,testing)
testing$classe <- predict(mod1,testPC)
```

#Conclusion ###The goal of this course project was to quantify how well participants completed a weight-lifting exercise. Data was collected from accelerometers on the belt, forearm, arm, and dumbell from 6 parcipants. Variables with too many missing values (over 95% missing) were not used to build the prediction model. 70% of the training dataset was used to build the model using random forest and the remaining observations (30%) were used for validation of the model. PCA was used to reduce the predictor set into a smaller set of variables that would differentiate between outcomes. When the random forest model was used on the validation set, there was 97.3% accuracy, sensitivity level between 92.4% - 99.1%, and specificity between 99%-99.8%. The model was used to predict the performance of 20 different test cases. Potential limitations of this study is that there was data from only 6 participants, with limited variety in background, meaning that it may not be a good predicting model for participants who are very different in terms of age and fitness levels.