Machine Learning

Introduction, Data Splitting and Data Cleansing

This assignment is concerned with applying machine learning techniques to classify the manner in which excercise was performed. The raw data set has 160 columns, and the goal is to be able to predict the manner in which the excercise was performed. The variable 'classe' categorizes the manner into A,B,C,D and E.

A training and test set are provided, however the test set is to be used to validate our model, so the training set should be split into a training and test set. A 75:25 training:test split is used after setting the seed to ensure the method is reproducable. The splitting will allow cross validation of the final model.

```
set.seed(1234)
inTrain = createDataPartition(data$classe, p = 3/4)[[1]]
training = data[ inTrain,]
testing = data[-inTrain,]
```

Many columns contain predominantly NA's. I chose to remove these, as imputing this many missing values will not provide good results. Further, the first seven columns contain labels, not predictors, so they should be removed. Additionally, I want to ensure that there are no colinear predictors or near zero variance predictors. This can be tested with nearZeroVar() and findCorrelation(), and it turns out that there are no predictors that should be removed for this reason.

```
table(sapply(as.list(training), function(x){ sum(is.na(x)) }))
```

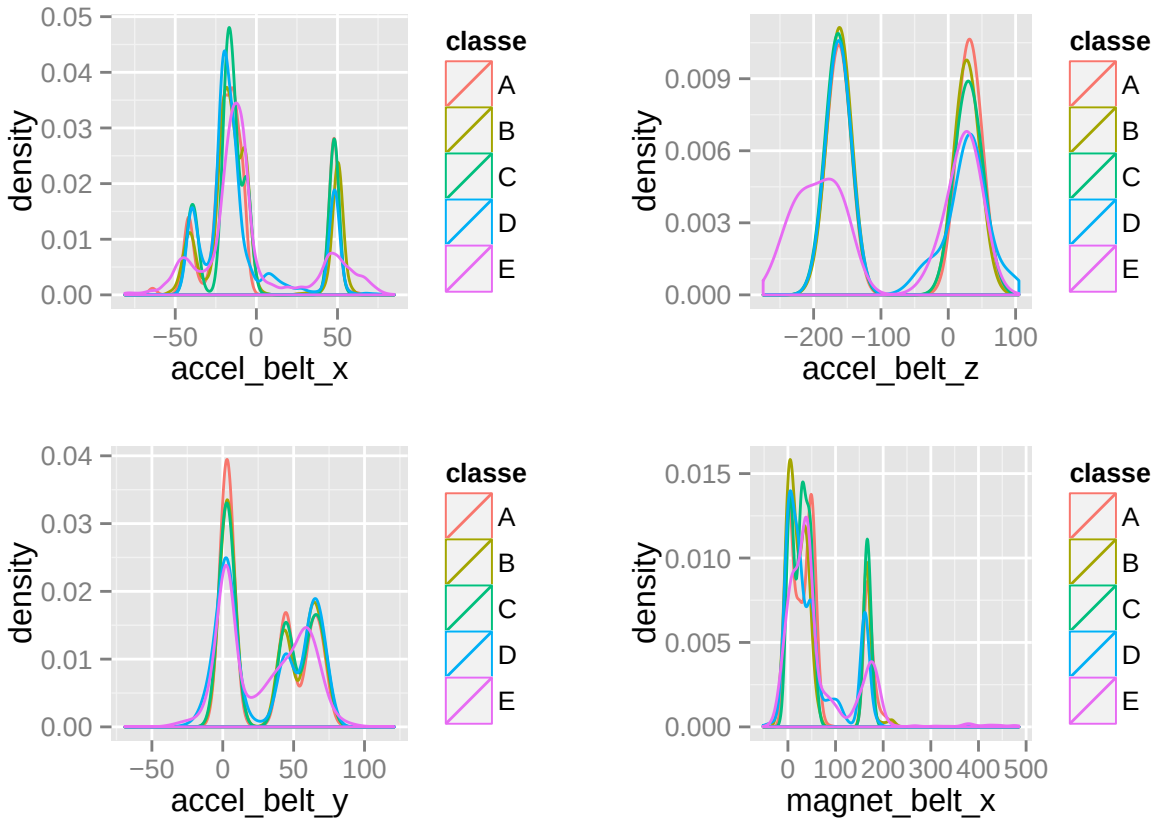
```
##
##     0 14409
##    60   100
```

```
training <- training[ , colSums(is.na(training)) == 0 ]
#can also remove columns 1:7 as they are labels
training <- select(training, -c(1:7))
```

So overall, we are left with 52 predictors to determine the outcome 'classe'.

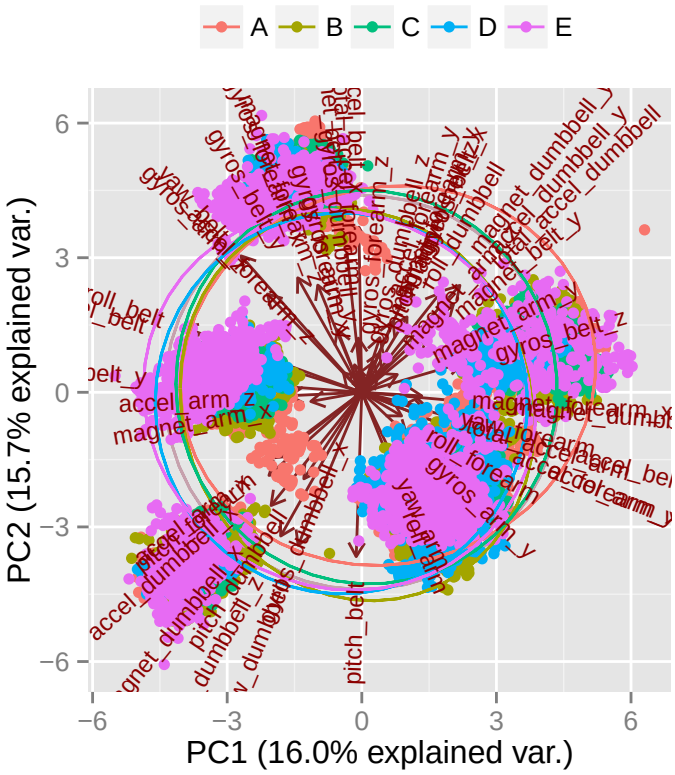
Exploratory Data Analysis

The aim of the exploratory data analysis is to determine what type of model might be suitable.



The sample plots demonstrate that many of the variables do not look normally distributed, so linear models might struggle without preprocessing. A PCA decomposition is shown below

```
training.pca <- prcomp(training[,-53], center = TRUE, scale. = TRUE)
g <- ggbiplot(training.pca, obs.scale = 1, var.scale = 1,
              groups = training$classe, ellipse = TRUE,
              circle = TRUE)
g <- g + scale_color_discrete(name = '')
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
print(g)
```



This shows us that after the PCA, which we do to create a new set of variables that explains the most variance of the data set, when using all the variable, unique clusters do not form for each classe. This suggests that linear models may struggle without considerable effort preprocessing or carefully selecting and testing which features produce clear well seperated clusters. Therefore, a non-linear model will be applied.

Modelling Strategy

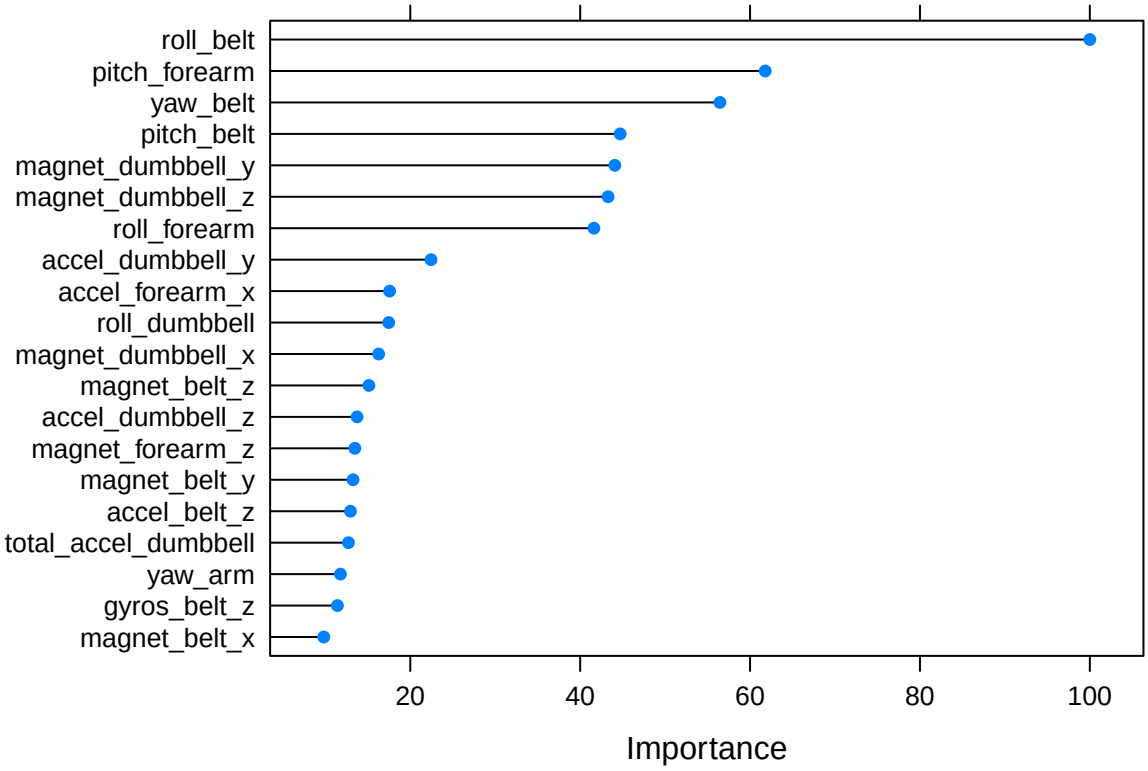
A random forest will be fitted to the training data. The model is well known for accuracy, but is prone to overfitting and can take a long time to fit. (In fact, the model took over an hour to fit to the training data set)

```
fitRF <- train(classe~., data=training, model='rf')
```

The caret defaults were used; it would only make sense to start adjusting them upon cross validation of the model if it did not perform well. For a random forest, the default resampling is 25 bootstrap resamples.

Results

From the fit, we can see the 20 most important predictors below (see ?randomForest for definition of importance).



Time to cross validate on our testing set. Select the same columns in testing as we did for training, and then predict using our random forest fit. We can look at the confusion matrix to pull out some useful statistics about the result.

```
keep <- names(training)
testing <- testing[, c(unlist(keep))]
predRF <- predict(fitRF, newdata=testing)
testing$RFpred <- predRF
confusionMatrix(table(predRF,testing$classe))
```

```
## Confusion Matrix and Statistics
##
##
## predRF    A    B    C    D    E
##      A 1395    7    0    0    0
##      B    0  939    6    1    0
##      C    0    3  846    8    2
##      D    0    0    3  794    1
##      E    0    0    0    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9935
##                  95% CI : (0.9908, 0.9955)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9917
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9895   0.9876   0.9967
## Specificity            0.9980   0.9982   0.9968   0.9990   0.9998
## Pos Pred Value         0.9950   0.9926   0.9849   0.9950   0.9989
## Neg Pred Value         1.0000   0.9975   0.9978   0.9976   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1725   0.1619   0.1831
## Detection Prevalence   0.2859   0.1929   0.1752   0.1627   0.1833
## Balanced Accuracy      0.9990   0.9938   0.9931   0.9933   0.9982
```

The model is over 99% accurate on the test set, so I will stop here. In other words, I expect the out of sample error to be less than 1% on the validation set (so I should score 20/20!)

Conclusion

A predictive model has been built to determine the manner in which an excercise is performed using 52 predictors, which are measurements from sensors attached to subjects. A random forest was chosen to build the model. The out of sample error is less than 1%. The model will now be applied to the validation set and submitted for grading.

```
keep <- names(training)
keep[53] <- 'problem_id'
validation <- validation[,c(unlist(keep))]
predRFV <- predict(fitRF, newdata=validation)
predRFV
```

```
##  [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
```