ML Assignment

Human Activity Recognition

Then loading the relevant packages required. For me, this includes tidyverse, caret and randomForest. I have decided that I will be using randomforest early on due to the relatively lack of computational power it requires. I will go further into the details of selection later on.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library(tidyverse)
```

```
## ── Attaching packages ────────────────────────────────────────────────────────────
────────── tidyverse 1.2.1 ──
```

```
## ✔ tibble  1.4.2     ✔ purrr   0.2.4
## ✔ tidyr   0.8.0     ✔ dplyr   0.7.4
## ✔ readr   1.1.1     ✔ stringr 1.3.0
## ✔ tibble  1.4.2     ✔ forcats 0.3.0
```

```
## ── Conflicts ─────────────────────────────────────────────────────────────────────
──── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

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

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

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

This assignment begins with loading both training and testing sets of the data.

```
pml_training <- read_csv("/Users/steve/Downloads/pml-training.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X1 = col_integer(),
##   user_name = col_character(),
##   raw_timestamp_part_1 = col_integer(),
##   raw_timestamp_part_2 = col_integer(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   num_window = col_integer(),
##   total_accel_belt = col_integer(),
##   kurtosis_roll_belt = col_character(),
##   kurtosis_picth_belt = col_character(),
##   kurtosis_yaw_belt = col_character(),
##   skewness_roll_belt = col_character(),
##   skewness_roll_belt.1 = col_character(),
##   skewness_yaw_belt = col_character(),
##   max_picth_belt = col_integer(),
##   max_yaw_belt = col_character(),
##   min_pitch_belt = col_integer(),
##   min_yaw_belt = col_character(),
##   amplitude_pitch_belt = col_integer(),
##   amplitude_yaw_belt = col_character()
##   # ... with 46 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
## Warning: 185 parsing failures.
## row # A tibble: 5 x 5 col     row col               expected actual   file
                   expected   <int> <chr>            <chr>    <chr>   <chr>

                  actual 1  2231 kurtosis_roll_arm a double #DIV/0! '/Users/steve/Dow
nloads/pml-tr… file 2  2231 skewness_roll_arm a double #DIV/0! '/Users/steve/Download
s/pml-tr… row 3  2255 kurtosis_roll_arm a double #DIV/0! '/Users/steve/Downloads/pml-
tr… col 4  2255 skewness_roll_arm a double #DIV/0! '/Users/steve/Downloads/pml-tr… ex
pected 5  2282 kurtosis_roll_arm a double #DIV/0! '/Users/steve/Downloads/pml-tr…
## ... ................. ...
.......................................................................... ........
.......................................................................... ......
.......................................................................... ....
.......................................................................... ...
.......................................................................... ...
.......................................................................... ........
..........................................................................
## See problems(...) for more details.
```

```
pml_testing <- read_csv("/Users/steve/Downloads/pml-testing.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   X1 = col_integer(),
##   raw_timestamp_part_1 = col_integer(),
##   raw_timestamp_part_2 = col_integer(),
##   num_window = col_integer(),
##   roll_belt = col_double(),
##   pitch_belt = col_double(),
##   yaw_belt = col_double(),
##   total_accel_belt = col_integer(),
##   gyros_belt_x = col_double(),
##   gyros_belt_y = col_double(),
##   gyros_belt_z = col_double(),
##   accel_belt_x = col_integer(),
##   accel_belt_y = col_integer(),
##   accel_belt_z = col_integer(),
##   magnet_belt_x = col_integer(),
##   magnet_belt_y = col_integer(),
##   magnet_belt_z = col_integer(),
##   roll_arm = col_double(),
##   pitch_arm = col_double(),
##   yaw_arm = col_double()
##   # ... with 37 more columns
## )
## See spec(...) for full column specifications.
```

Next up is the selection of variables to be used as part of the training the model. The following are the principles I used in selection of variable. - The variable must not have more than 50% of the data missing. i.e they cannot have NA as their majority input. This is because the missing values would need to be imputed, and if a large number is imputed then the data may potentiall be skewed. - The variable must not be overly specific, otherwise this will overfit the training set but do poorly on the test set. However, I did include the username, as I can see that the users in training and testing sets are the same, doing this allows the model to adjust for individual differences that could potentially change the prediction. - The variable must contribute to the predicted variable in a likely fashion. i.e, must not be noise. These include the window and the timestamp variables, which does not seem to have a known direct correlation with the type of movement the users make, and may instead create additional noise and reduce the accuracy of the variable.

```
pml_training1 <- pml_training %>% select(user_name, roll_belt, pitch_belt, yaw_belt,
total_accel_belt, gyros_belt_x, gyros_belt_y, gyros_belt_z, accel_belt_x, accel_belt_
y, accel_belt_z, magnet_belt_x, magnet_belt_y, magnet_belt_z, roll_arm, pitch_arm, ya
w_arm, total_accel_arm, gyros_arm_x, gyros_arm_y, gyros_arm_z, accel_arm_x, accel_arm
_y, accel_arm_z, magnet_arm_x, magnet_arm_y, magnet_arm_z, roll_dumbbell, pitch_dumbb
ell, yaw_dumbbell, total_accel_dumbbell, gyros_dumbbell_x, gyros_dumbbell_y, gyros_du
mbbell_z, accel_dumbbell_x, accel_belt_y, accel_belt_z, magnet_dumbbell_x, magnet_dum
bbell_y, magnet_dumbbell_z, roll_forearm, pitch_forearm, yaw_forearm, gyros_forearm_
x, gyros_forearm_y, gyros_forearm_z, accel_forearm_x, accel_forearm_y, accel_forearm_
z, magnet_forearm_x, magnet_forearm_y, magnet_forearm_z, classe)
```

This resulted in me selecting only the variables associated with movement and has minimal missing data. The code is dataframe is re-saved, selecting only the variables to ease later codes. The “classe” variable which is the target variable we are trying to predict is converted to a factor class to ensure that the later model will be a categorical prediction.

```
pml_training1$classe <- as.factor(pml_training1$classe)
```

I have not gone through the entirety of the dataset, however just incase that there is some missing data I would like to impute them using K nearest neighbour. As part of the process I would also like to standardise the data to reduce the large effect of outliers from the dataset. The train data is split into the training and validation sets by 80:20 rule to estimte my out of sample error.

```
train_process <- pml_training1[,-51]
preprocess_train <- preProcess(train_process, method = "knnImpute")
pml_train_f <- predict(preprocess_train, train_process)
pml_train_f1 <- pml_train_f %>% cbind(pml_training1[,51])
partition <- createDataPartition(pml_train_f1$classe, p=0.8, list=FALSE)
train <- pml_train_f1[partition,]
valid <- pml_train_f1[-partition,]
pml_train_f <- predict(preprocess_train, train)
```

The next step is to set up the training parameters via the trainControl function. In this assignment I will use the cross validation only as this is less computational demanding but produces relative good results. Cross validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The data is split into ‘k’ number of groups, and each group is treated as test data, while the remaining groups are treated like a training data set. For the purpose of this exercise k=10

```
train1 <- trainControl(method = "cv", number = 10)
```

Ideally, I would have preferred to use the leave out out cross validation would produce the most amount of data for training to occur. However, it is also very computational demanding, and my current laptop may not have sufficient computation power for.

Once the training parameters are completed and the data is well set up for the training, we would decide on the training method used. Decision tree learning is a method commonly used in data mining and machine learning. The goal is to create a model that predicts the value of a target variable based on several input variables. Hence, for a simple, yet accurate algorithm, I have chose randomforest, a sub class of decision tree that aggregrates the results from many trees. If there was sufficient computational power, I would have loved to attempt gradient boosting, especially given the large number of variables used within the model. I will save this model as “model1”.

```
model1 <- train(classe ~ ., data = train, method = "rf", trControl = train1)
```

The model can then be used to predict the classe of the testing data. I will then print out these predictions.

```
predict1 <- predict(model1, pml_testing)
predict1
```

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

As part of predicting the out of sample error, I will do so on the validation set.

```
predicted_valid <- predict(model1, valid)
confusionMatrix(predicted_valid, valid$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    3    0    0    0
##          B    2  754    6    0    0
##          C    0    2  676    7    1
##          D    0    0    2  636    2
##          E    0    0    0    0  718
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9936          
##                  95% CI : (0.9906, 0.9959)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9919          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9934   0.9883   0.9891   0.9958
## Specificity            0.9989   0.9975   0.9969   0.9988   1.0000
## Pos Pred Value         0.9973   0.9895   0.9854   0.9938   1.0000
## Neg Pred Value         0.9993   0.9984   0.9975   0.9979   0.9991
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1922   0.1723   0.1621   0.1830
## Detection Prevalence   0.2847   0.1942   0.1749   0.1631   0.1830
## Balanced Accuracy      0.9986   0.9954   0.9926   0.9939   0.9979
```

And that gives my accuracy and the estimated out of sample error can be calculated by 1 - accuracy.