Approach to modeling:

After importing the datasets into the R session, and inspecting its structure, I observed the following:

- The dataset consists of data from mainly 5 users
  - Since we want our model to be generalizable, usernames should not be included in the dataset
- A lot of variables are not from accelerometers and I initially focused my attention on them:
  - The variable 'X' is the index number that is unique for each row
  - Raw timestamp 1 is a zero-variance variable
  - Raw timestamp 2 is has value variation, but no variation for each of the response class
  - 'New_window' and 'num_window' also relate to the response in some form, but these certainly are not readings from the accelerometer

It was decided to exclude all these variables (columns from 1 to 7) from the datasets.

Post deletion of these, I next focused on the numeric variables that were coded as factors. These were mostly skewness and kurtosis variables, and their classes were changed to 'numeric'. After this, a summary analysis of the numeric variables revealed a 100 of these had over 98% of the data missing. Since imputing data on such a large scale would jeopardize the generalizability of our model, I would prefer to eliminate these variables from the datasets.

Hence, after this cleaning, I was left with 53 variables – 52 predictors and one response variable.

Since out training dataset has a large number of rows (19622), I decided to split the data into training and validation sets, in a 75:25 ratio. Validating the model before predicting the test data on it seemed like a safe approach to me.

For the exploratory data analysis, I used a custom package 'lolcat' which I have previously used in my college courses. I particularly like its functions since we can output common EDA statistics like the five-number summary as a data-frame. For graphical interpretation, since the response is a categorical variable and the predictors are continuous, the best representation would be a boxplot, with a linear model smoother depicting the correlation between the classes and the predictor. I haven't implemented this in this model but it is a suggestion.

Lastly, I did not use any cross validation or pre-processing prior to running the model. I wanted to first assess the accuracy achieved without any extra work. If the resulting accuracy was low, other improvement methods would be tried. Further, I used the random forest algorithm (from the package 'random forest') to create my model (the caret implementation had some issues, and was too slow to perform). I used the algorithm with all default settings.

Predicting the outcome on the validation data, and creating its confusion matrix provided with the model statistics. The accuracy achieved was 99.53% (the out of sample error rate is 1 – accuracy; or 1 – 0.9953 = 0.0047). Hence, I did not intend to improve the model any further and directly used it to predict the test data outcomes.