Intro:

the aim of this project is to fit a predictive model that predicts what type of movement is an athlete doing based on the values given by multiple sensors attached in the athlete's bodies while they were performing these movements.

My appraoch:

as shown in the source code file I started by loading the data which was in the form of ".csv" files. At the first glance to this raw data we can see that it has lots of empty columns or "NA" columns, along with some non numeric data like time series which is of no use for our prediction. Which means we should perform some preprocessing on it, for it to be of any use for us.

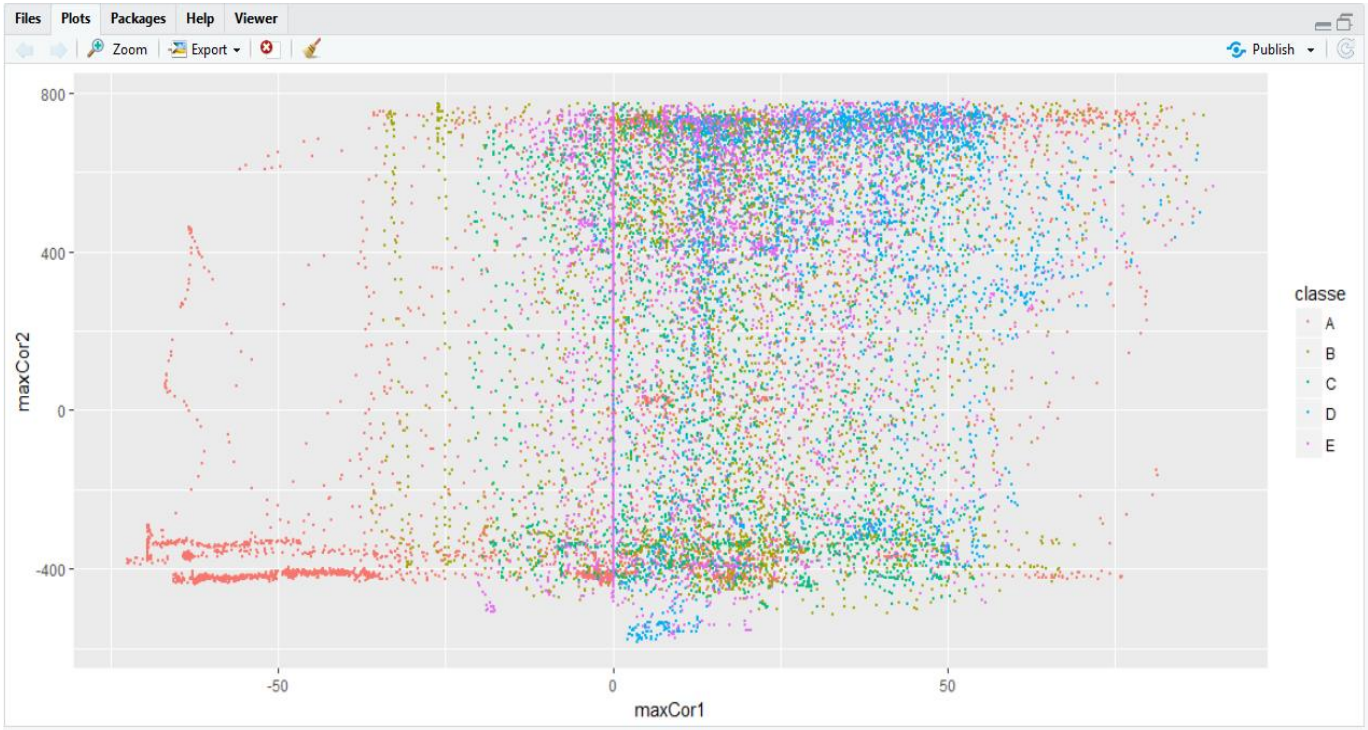
Data cleaning:

we start by saving the" classe" column so that it dose not get affected by the preprocessing. Then we remove the "X" variable - which is numeric yet it only carries the order of the samples and holds no relevant information for the the prediction. Then we remove the "NA" columns, empty columns, and non-numeric columns respectively, which leaves us with only 56 predictors out of 160. After that we reattach the "classe" column to our data frame.

Data exploring :

In order to select the best model type we need to know how correlated are our predictors with our outcome. So we perfomed a calculation of "Spearman" rank based correlation because we are calculating the correlation with a categorical variable . Then we save the correlation of each predictor with the "classe" variable in a object called "M".

After that we save the two predictors with the most correlation with the outcome in two lists called "maxCor1" and "maxCor2". Then we plot them against each other and color the "classe" variable using "qplot":



the plot shows no particular trend in which the outcome is distributed, So we can only conclude that we have a set of "weak" predictors, and therefore we need to fit a boosted model instead of a general linear regression model. And use repeated cross validation which would compromise for the "weakness" of our data set. And thereby increase accuracy on the expanse of scalability .

Model fitting:

so we start by partitioning our data into a training and testing sets, then we set the trainControl() to use repeated cross validation repeated 3 times and using 5 folds.Then to fit a general boosted model we use the train() function from the "caret" package.

Confusion matrix:

After the model is finally trained which took a few minutes we now calculate the confusion matrix using "confusionMatrix()" in order to predict the out of sample error :

```
          Reference
Prediction    A    B    C    D    E
         A 2231    5    0    0    0
         B    1 1510    2    0    0
         C    0    3 1364    5    0
         D    0    0    2 1273    5
         E    0    0    0    8 1437
```

```
Overall Statistics

               Accuracy : 0.996
                 95% CI : (0.9944, 0.9973)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.995
 Mcnemar's Test P-Value : NA
```

We can see that the overall accuracy of our model is 0.996 which means that the expected out of sample error is

E = 1 - 0.996 = 0.004 = 0.4% which is a pretty satisfying value.

We can also see that the easiest classe of moves to predict was "A" as it had the highest accuracy metrics values as shown in the confusion matrix:

```
Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9996   0.9947   0.9971   0.9899   0.9965
Specificity            0.9991   0.9995   0.9988   0.9989   0.9988
Pos Pred Value         0.9978   0.9980   0.9942   0.9945   0.9945
Neg Pred Value         0.9998   0.9987   0.9994   0.9980   0.9992
Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
Detection Rate         0.2843   0.1925   0.1738   0.1622   0.1832
Detection Prevalence   0.2850   0.1928   0.1749   0.1631   0.1842
Balanced Accuracy      0.9993   0.9971   0.9979   0.9944   0.9976
```

Predicting for the 20 test samples:

Finally we predict on the testing samples which gives the following :

```
[1] B A B A A E D B A A B C B A E E A B B B
```