Practical Machine Learning Project

Juan Jose Suarez

27/6/2020

Clean Data

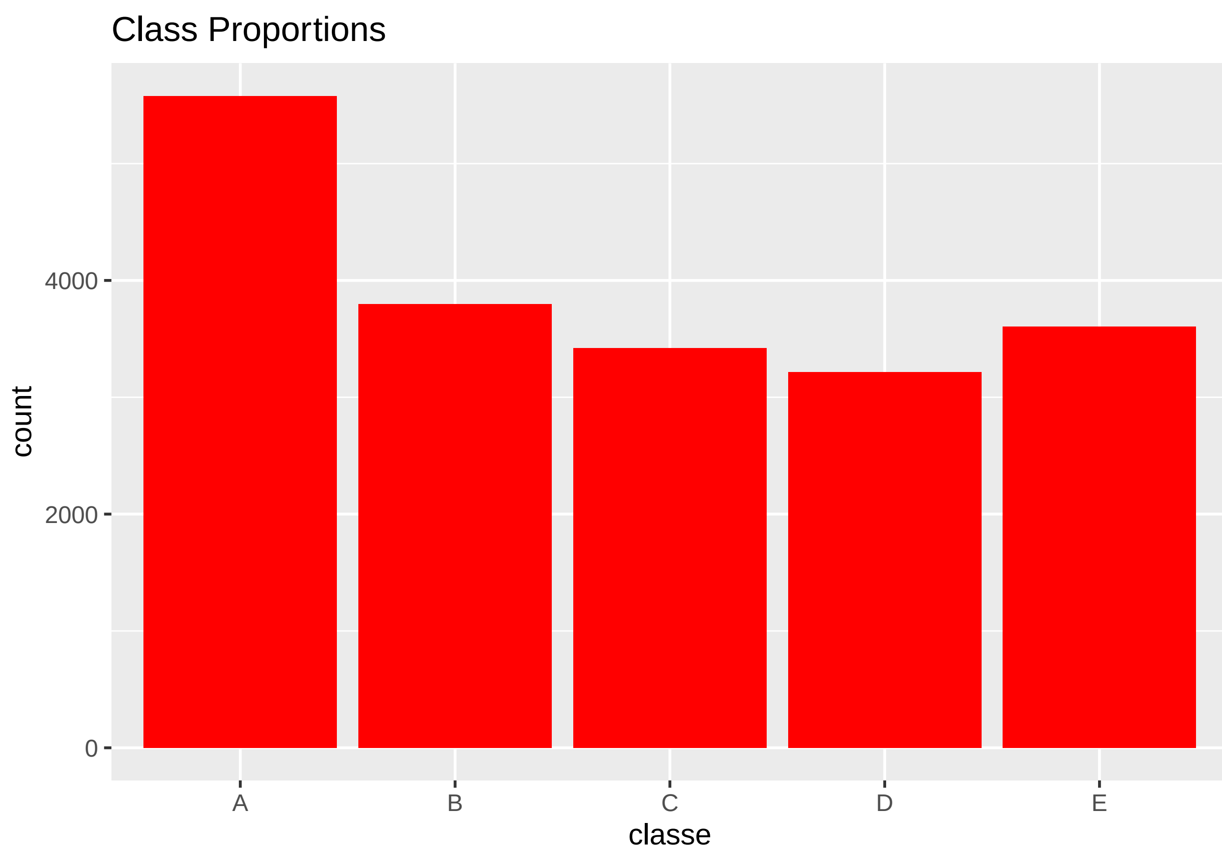
We have 19.622 samples. Drop 100 columns with mostly null data, 1 date column and the row number.

```
## [1] 19622   160
```

```
## [1] 19622    58
```

Data Exploration

```
ggplot(pml.training, aes(x = classe)) + geom_bar(fill="red") + ggtitle("Class Proportions")
```

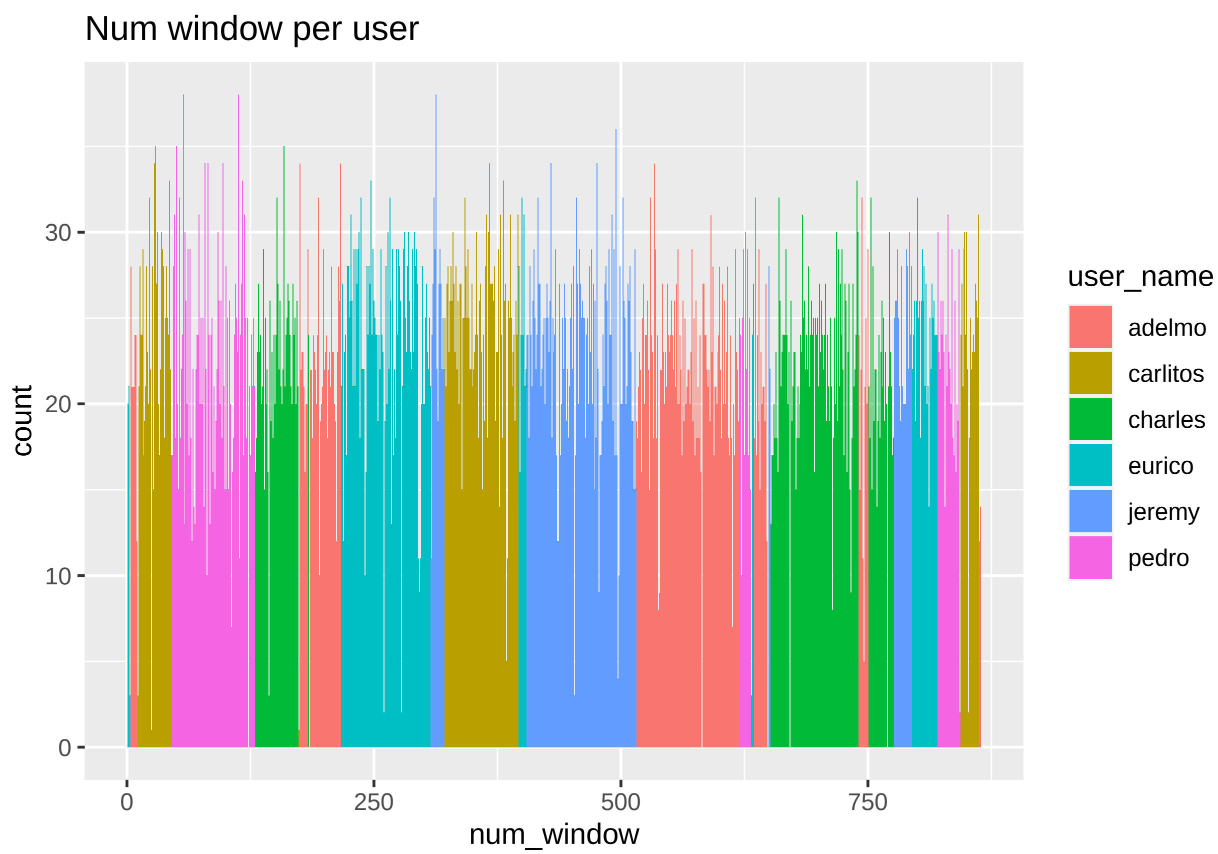


Variables with high correlation.

```
##                     A                 B          R        R2
## 178      accel_belt_z        roll_belt -0.9920085 0.9840809
## 172         roll_belt total_accel_belt  0.9809241 0.9622122
## 1851 gyros_dumbbell_x gyros_dumbbell_z -0.9789507 0.9583445
## 343      accel_belt_z total_accel_belt -0.9749317 0.9504919
## 231      accel_belt_x       pitch_belt -0.9657334 0.9326410
## 618      accel_belt_y     accel_belt_z -0.9333854 0.8712083
```

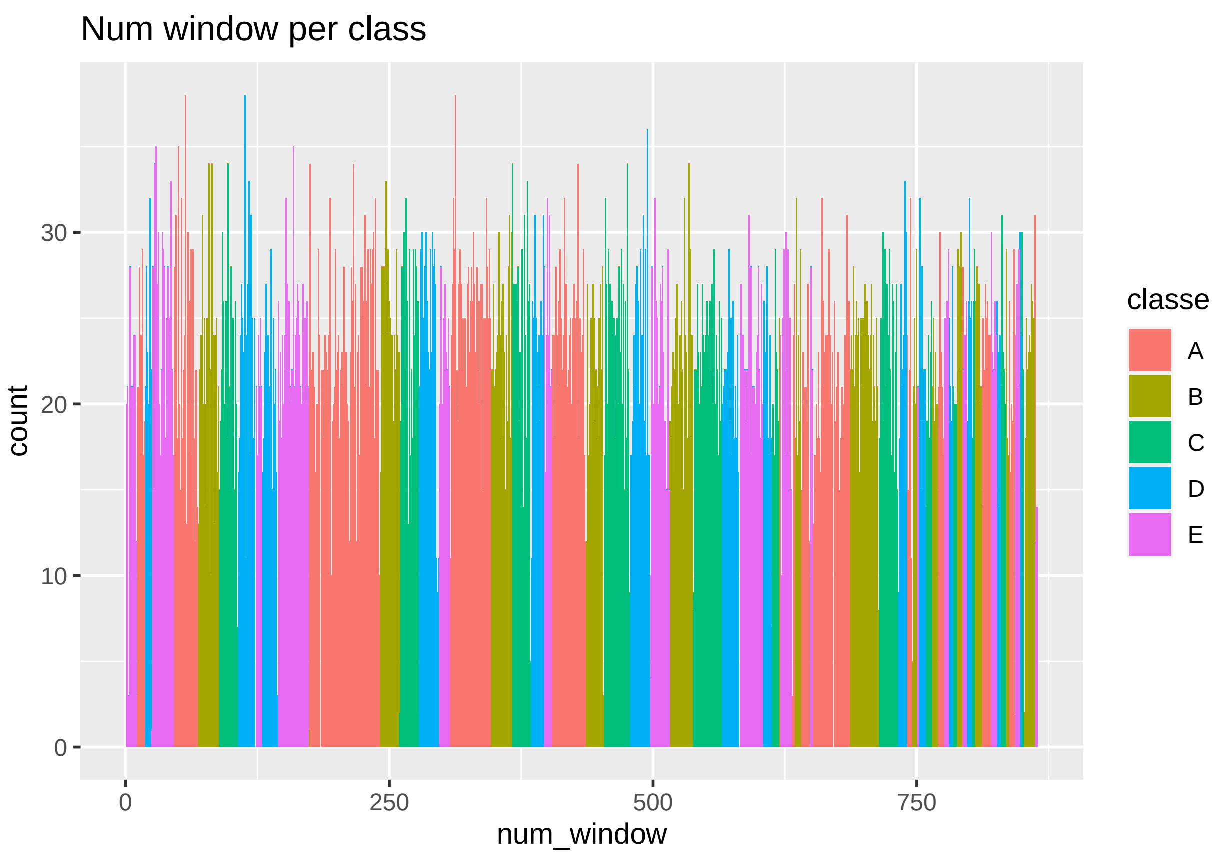
Num window seems to explain perfectly the user.

```
ggplot(pml.training, aes(x=num_window, fill=user_name)) + geom_histogram(binwidth = 1) + ggtitle("Num wi
```



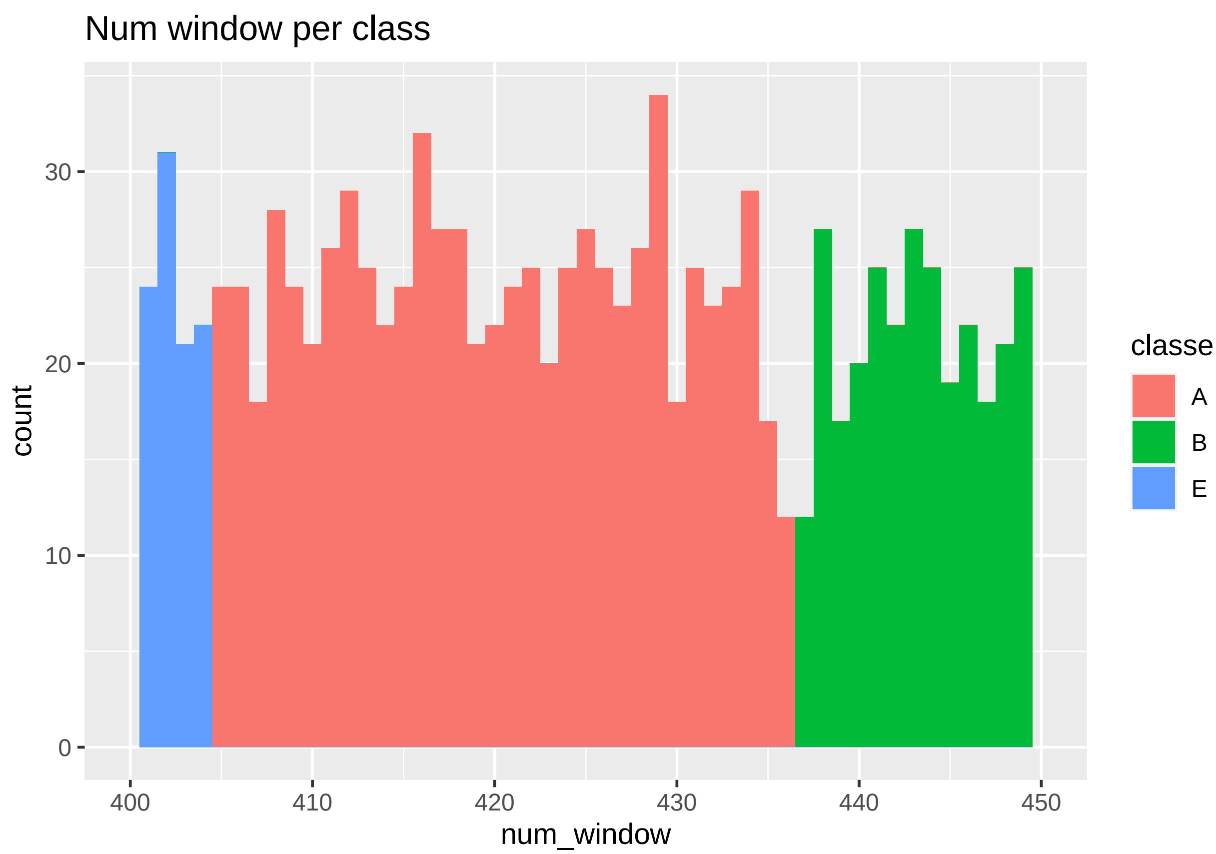
Num window also seems to explain perfectly the class. This correlations seem to represent the structure of the experimental settings.

```
ggplot(pml.training, aes(x=num_window, fill=classe)) + geom_histogram(binwidth = 1) + ggtitle("Num windo
```



Actually if you zoom in you can see that the classes correspond exactly to certain intervals of the num_window variable. Is it possible to perfectly predict the class by training a Decision Tree only with the num_window variable?

```
ggplot(pml.training, aes(x=num_window, fill=classe)) + geom_histogram(binwidth = 1) + ggtitle("Num windo
```



Model

Divide the data 80% for training and 20% for testing.

```
set.seed(1234)
inTrain <- createDataPartition(pml.training$classe, p = 0.8)[[1]]
training <- pml.training[inTrain, ]
testing <- pml.training[-inTrain, ]
```

Using only num_window

Use 10 Fold Cross Validation repeated 3 times. Train a Decision Tree only using num_window trying multiple complexity (cp) parameters.

```
modelTree
```

```
## CART
##
## 15699 samples
##     1 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 14129, 14129, 14129, 14130, 14129, 14129, ...
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   0.001  0.9999363  0.9999194
##   0.005  0.9447114  0.9300623
##   0.010  0.8471454  0.8069234
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.001.
```

```
postResample(pred = predict(modelTree, newdata = testing), testing$classe)
```

```
## Accuracy    Kappa
##        1        1
```

```
table(predict(modelTree, newdata = testing), testing$classe)
```

```
##
##         A    B    C    D    E
##    A 1116    0    0    0    0
##    B    0  759    0    0    0
##    C    0    0  684    0    0
##    D    0    0    0  643    0
##    E    0    0    0    0  721
```

Given the estimation of the accuracy on the test data we can expect the model to have a near perfect prediction accuracy in the test data that is drawn from this same experiment. On the other hand this model would not extrapolate well to other datasets because exploits the specific experimental conditions of the data collection process.

Having said that, a model that generalizes well on real world data should not exploit this specific conditions, thus we will develop another model considering only sensor data.

Using only sensor data

Training a Decision Tree yields a cross-validation accuracy of 91.6%.

```
set.seed(8877)
# 10-fold CV
fitControl <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 3)

modelTree <- train(
  classe ~ .,
  training,
  method = "rpart",
  trControl = fitControl,
  tuneGrid = data.frame(cp = c(0.00001, 0.00005, 0.0001, 0.0005, 0.001))
)
```

```r
modelTree
```

```
## CART
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 14129, 14130, 14129, 14129, 14129, 14131, ...
## Resampling results across tuning parameters:
##
##   cp     Accuracy   Kappa
##   1e-05  0.9340514  0.9165635
##   5e-05  0.9342212  0.9167782
##   1e-04  0.9337545  0.9161888
##   5e-04  0.9239872  0.9038053
##   1e-03  0.9051328  0.8799379
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 5e-05.
```

A C5.0 tree with boosting yields a cross-validation accuracy of 99%.

```r
set.seed(9191)
# 5 fold CV
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

tuneGrid <- expand.grid(.trials = c(5, 7, 9),
                        .model = c("tree"),
                        .winnow = c(F))

modelC5 <- train(
  classe ~ .,
  training,
  method = "C5.0",
  trControl = fitControl,
  verbose=T,
  tuneGrid = tuneGrid
)

modelC5
```

```
## C5.0
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 12558, 12558, 12560, 12559, 12561
## Resampling results across tuning parameters:
##
##   trials  Accuracy   Kappa
##   5       0.9829924  0.9784866
##   7       0.9885340  0.9854971
##   9       0.9900630  0.9874300
##
## Tuning parameter 'model' was held constant at a value of tree
## Tuning
##  parameter 'winnow' was held constant at a value of FALSE
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were trials = 9, model = tree and winnow
##  = FALSE.
```

A Random Forest yields an out of bag accuracy of 99.46%.

```r
set.seed(2424)
modelRandomForest <-  randomForest(classe ~ ., data = training, ntree = 100)
modelRandomForest
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.54%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4459    4    0    0    1 0.001120072
## B   14 3020    4    0    0 0.005924951
## C    0   14 2717    7    0 0.007669832
## D    0    0   27 2544    2 0.011270890
## E    0    0    4    7 2875 0.003811504
```

We choose the best model and estimate its performance in the test data.

```r
postResample(predict(modelRandomForest, newdata = testing), testing$classe)
```

```
##  Accuracy     Kappa
## 0.9936273 0.9919386
```

This last model has a great accuracy (99.36%) on the testing data and is expected to generalize better in real world situations because it does not exploit specific conditions of the data collection process.