Predict Activity

Hui Yi Lee

4/3/2020

Executive Summary

This report aims to predict the classe of a given observation, given various other variables in the data set, such as the timestamp and other activity indicators.

Load and Process the Data

We first load the training data set, assign it to an object called "activity" and further partition it into training and test sets for cross validation purposes.

```r
setwd("C:/Users/lee_h/Desktop/datasciencecoursera/Course 8_W4_Prog_Assignment")
activity <- read.csv("pml-training.csv", sep = ",", header = TRUE)
library(lattice); library(ggplot2); library(caret)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

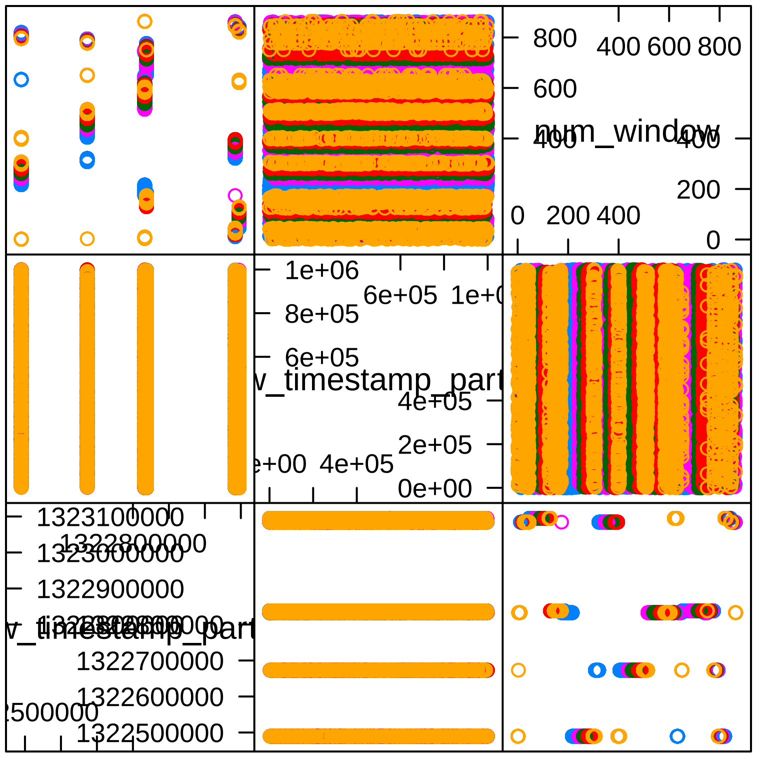
```r
inTrain <- createDataPartition(y = activity$classe, p = 0.8, list = FALSE)
train <- activity[inTrain, ]
test <- activity[-inTrain, ]
```

Exploratory Analysis

We perform some exploratory analysis.

Plot 1: First create a feature plot to see how pairs of the first few variables are related to each other.

```r
featurePlot(x = train[, c("raw_timestamp_part_1", "raw_timestamp_part_2", "num_window")], y = train$cla
```

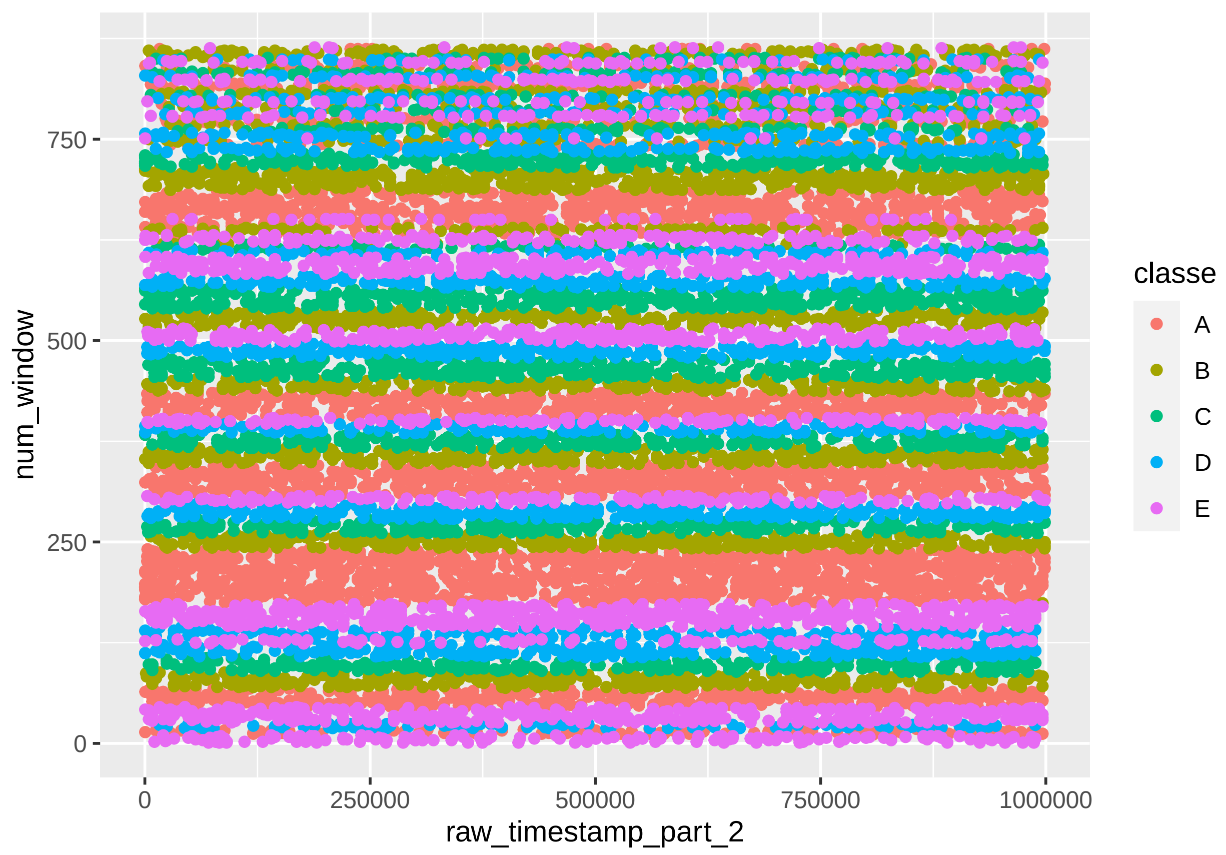


Scatter Plot Matrix

This suggests that num_window may be important in explaining variation in classe, since there is a clear pattern in classe along the y axis, while raw_timestamp_part_2 may be less important.

Plot 2: Focus on plot of timestamp part 2 against num_window, and colour by the classe variable .

```
qplot(raw_timestamp_part_2, num_window, colour = classe, data = train)
```



Pre-processing

Generate variables which do not have much variability in them, so not likely to be good predictors. Extract variables with more than 2% unique values.

```
nsv <- nearZeroVar(train, saveMetrics = TRUE)
colwant1 <- which(nsv$percentUnique >= 2)
colwant1 <- c(colwant1)
colwant1 <- append(colwant1, 160)
colwant1 <- colwant1[!colwant1 %in% 1]
train1 <- train[, colwant1]
nsv1 <- nearZeroVar(train1, saveMetrics = TRUE)
dim(train1)
```

```
## [1] 15699    44
```

Building the Model

Model 1

First relevel the classe variable. Then fit a multinominal model on all predictors in “train” data set. Generate predictions on the subsetted train data set. This gives accuracy of 28.4%.

```r
library(nnet)
```

```
## Warning: package 'nnet' was built under R version 3.6.3
```

```r
train1$classe <- relevel(train1$classe, ref = "A")
train.mn1 <- multinom(classe ~ ., data = train1, na.action = na.omit, maxit = 10000, MaxNWts = 10000000
```

```
## # weights:  6215 (4968 variable)
## initial  value 510.191818
## iter  10 value 503.499260
## iter  10 value 503.499260
## iter  20 value 503.443746
## iter  20 value 503.443743
## iter  20 value 503.443738
## final  value 503.443738
## converged
```

```r
probability.table <- fitted(train.mn1)
train.mn1$pred <- predict(train.mn1, newdata = train1, "class")
ctable <- table(train1$classe, train.mn1$pred)
round((sum(diag(ctable))/sum(ctable))*100, 2)
```

```
## [1] 28.08
```

Transform the test set in the same way, and predict on the test set using the "train.mn1" model. This gives accuracy of only 20%, lower than that in the training set. Indicates that there may be an issue of overfitting.

```r
test1 <- test[, colwant1]
test1$pred <- predict(train.mn1, newdata = test1, "class")
ctable <- table(test1$classe, test1$pred)
round((sum(diag(ctable))/sum(ctable))*100, 2)
```

```
## [1] 22.47
```