Practical Machine Learning Course Project

Celeste Falcon

2020/04/17

Set up, load packages, and record session info

```
# load packages here
library(tidyverse)
```

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

```
## ✓ ggplot2 3.3.0     ✓ purrr   0.3.3
## ✓ tibble  3.0.0     ✓ dplyr   0.8.3
## ✓ tidyr   1.0.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────────────────────────────
────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(knitr)
library(conflicted)
library(caret)
```

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

```
# load global functions here
`%nin%` = Negate(`%in%`)

conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package
```

```
conflict_prefer("select", "dplyr")
```

```
## [conflicted] Will prefer dplyr::select over any other package
```

```
# load Session Info here
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Red Hat Enterprise Linux
##
## Matrix products: default
## BLAS: /gpfs1/blxgrid/tools/bioinfo/app/R-3.5.1/lib64/R/lib/libRblas.so
## LAPACK: /gpfs1/blxgrid/tools/bioinfo/app/R-3.5.1/lib64/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] caret_6.0-47     lattice_0.20-38  conflicted_1.0.4 knitr_1.21
##  [5] forcats_0.5.0    stringr_1.4.0    dplyr_0.8.3      purrr_0.3.3
##  [9] readr_1.3.1      tidyr_1.0.0      tibble_3.0.0     ggplot2_3.3.0
## [13] tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.0          jsonlite_1.6        splines_3.5.1
##  [4] foreach_1.5.0       carData_3.0-3       modelr_0.1.2
##  [7] gtools_3.8.2        assertthat_0.2.1    statmod_1.4.34
## [10] cellranger_1.1.0    yaml_2.2.0          pillar_1.4.3
## [13] backports_1.1.6     glue_1.4.0          digest_0.6.25
## [16] rvest_0.3.2         minqa_1.2.4         colorspace_1.4-1
## [19] htmltools_0.3.6     Matrix_1.2-15       plyr_1.8.6
## [22] pkgconfig_2.0.3     broom_0.5.1         BradleyTerry2_1.1-2
## [25] haven_2.2.0         scales_1.1.0        openxlsx_4.1.4
## [28] rio_0.5.16          brglm_0.6.2         lme4_1.1-23
## [31] generics_0.0.2      car_3.0-7           ellipsis_0.3.0
## [34] withr_2.1.2         cli_2.0.2           magrittr_1.5
## [37] crayon_1.3.4        readxl_1.3.1        memoise_1.1.0
## [40] evaluate_0.14       fansi_0.4.1         nlme_3.1-137
## [43] MASS_7.3-51.1       xml2_1.2.0          foreign_0.8-71
## [46] tools_3.5.1         data.table_1.12.8   hms_0.5.3
## [49] formatR_1.5         lifecycle_0.2.0     munsell_0.5.0
## [52] zip_2.0.4           compiler_3.5.1      profileModel_0.6.0
## [55] rlang_0.4.5         grid_3.5.1          nloptr_1.2.2.1
## [58] iterators_1.0.12    rstudioapi_0.11     qvcalc_1.0.2
## [61] rmarkdown_1.11      boot_1.3-20         gtable_0.3.0
## [64] codetools_0.2-16    abind_1.4-5         curl_4.3
## [67] reshape2_1.4.4      R6_2.4.1            lubridate_1.7.4
## [70] stringi_1.4.6       Rcpp_1.0.4.6        vctrs_0.2.4
## [73] tidyselect_1.0.0    xfun_0.4
```

Data Input

```
training <- read_csv("/gpfs2/duds/giywg/PracticalMachineLearning/project/pml-training.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   user_name = col_character(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   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_yaw_belt = col_character(),
##   min_yaw_belt = col_character(),
##   amplitude_yaw_belt = col_character(),
##   kurtosis_picth_arm = col_character(),
##   kurtosis_yaw_arm = col_character(),
##   skewness_pitch_arm = col_character(),
##   skewness_yaw_arm = col_character(),
##   kurtosis_yaw_dumbbell = col_character(),
##   skewness_yaw_dumbbell = col_character(),
##   kurtosis_roll_forearm = col_character(),
##   kurtosis_picth_forearm = col_character()
##   # ... with 8 more columns
## )
```

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

```
testing <- read_csv("/gpfs2/duds/giywg/PracticalMachineLearning/project/pml-testing.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_logical(),
##   X1 = col_double(),
##   user_name = col_character(),
##   raw_timestamp_part_1 = col_double(),
##   raw_timestamp_part_2 = col_double(),
##   cvtd_timestamp = col_character(),
##   new_window = col_character(),
##   num_window = col_double(),
##   roll_belt = col_double(),
##   pitch_belt = col_double(),
##   yaw_belt = col_double(),
##   total_accel_belt = col_double(),
##   gyros_belt_x = col_double(),
##   gyros_belt_y = col_double(),
##   gyros_belt_z = col_double(),
##   accel_belt_x = col_double(),
##   accel_belt_y = col_double(),
##   accel_belt_z = col_double(),
##   magnet_belt_x = col_double(),
##   magnet_belt_y = col_double(),
##   magnet_belt_z = col_double()
##   # ... with 40 more columns
## )
## See spec(...) for full column specifications.
```

Exploratory analysis

I took several steps to better understand the data set itself.

Explore data set

These steps included examining the y-variable (classe) and viewing the column names and first 10 observations for each variable (I have commented this out since it creates a very long print out in the html file). I could see that several columns of the test data set were completely missing data (all NA). These will not be useful as predictors and could lead to overfitting of the training data set, so I removed these columns from both the testing and training data.

```
# Explore y-variable to be predicted
unique(training$classe)
```

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

```
class(training$classe)
```

```
## [1] "character"
```

```
## Set classe variable to factor in both training data
training$classe <- as.factor(training$classe)

# Examine data variables print(training, width = Inf) print(testing, width =
# Inf)

# remove columns that are all NA in testing data set as these will not be
# useful as predictors and could lead to overfitting in training data set
testing <- janitor::remove_empty(testing, which = "cols") %>% select(-X1)

cols_in_testing <- colnames(testing)

# remove those same columns from training data set so they are not used to
# create prediction function
training <- training %>% select_if(names(.) %in% cols_in_testing | names(.) ==
    "classe")
```

Create subset for internal testing from training data set to be able to estimate out of sample error and accuracy

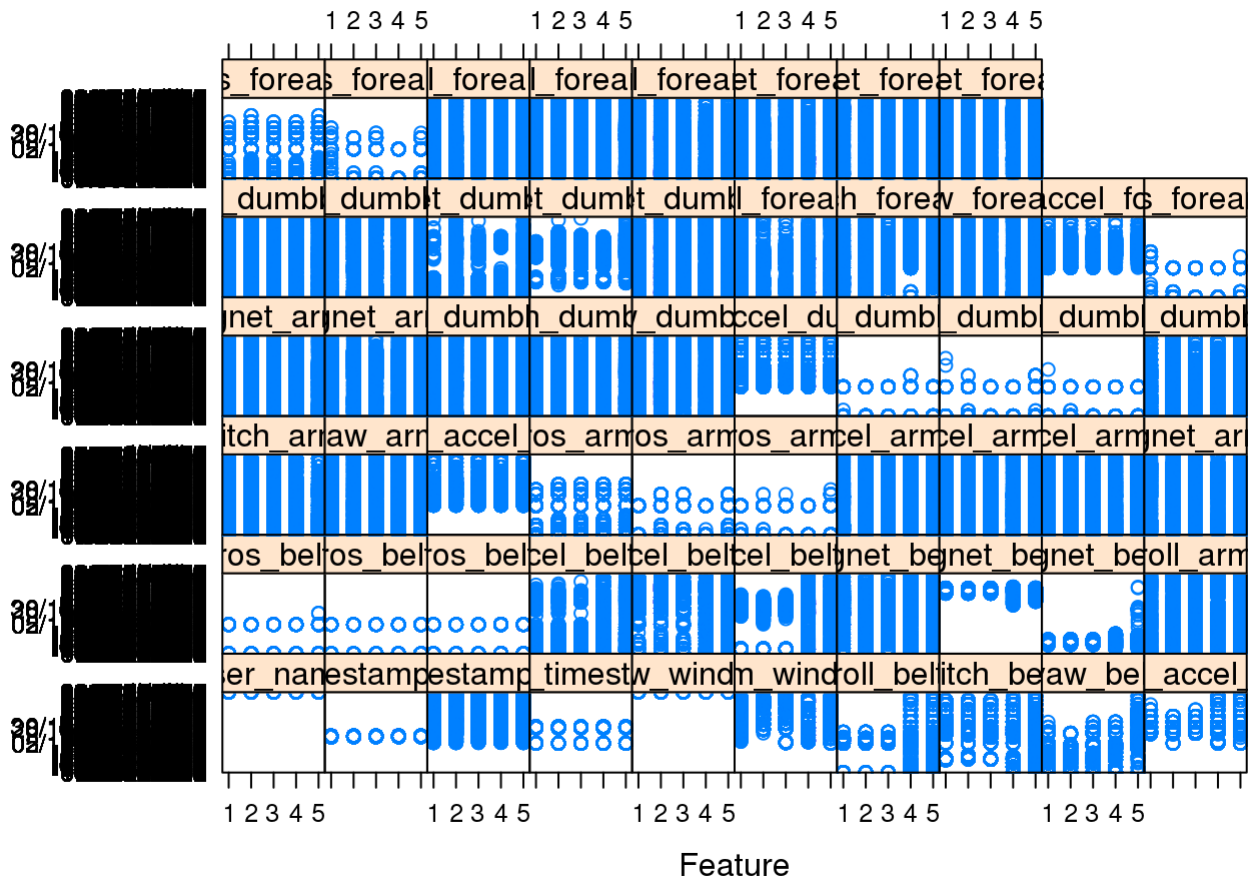
In order to estimate my out of sample error, I decided to further subset the original training data into two groups that I called “internal training” and “internal testing”. This allowed me to test prediction models created in “internal training” on the “internal testing” data set and calculate the accuarcy of the method.

```
in_internal_training <- as.vector(createDataPartition(y = training$classe, p = 0.85,
    list = FALSE))

internal_training <- training[in_internal_training, ]
internal_testing <- training[-in_internal_training, ]

internal_testing2 <- internal_testing[, -59]
```

Exploratory visualizations

I also used visualizations to explore the data. I started by looking at a feature plot. However, with so many predictor variables in this data set, it was difficult to see whether there are any patterns. I then created violin plots for each variable by the classe variable. These plots allowed me to see the distribution of each variable within each classe for quantitative variables, and for qualitative variables it at least showed me whether each level of the variable existed within each classe. For the sake of brevity for my graders, I have not reproduced the violin plots in this html file, but the code for them can be seen in this code chunk. Finally, I created a table of classe by the user_name variable to see whether these were related (plot not shown here to reduce the number of figures graders must review, but code is shown below).

```
featurePlot(x = internal_training[, -59], y = internal_training$classe)
```




```
# for (p in 1:(ncol(internal_training) - 1)) { # p=6 boxplot <-
# ggplot(internal_training, aes_string(x = 'classe', y =
# colnames(internal_training)[p])) + geom_violin(draw_quantiles =
# c(0.25,0.50,0.75)) + ylab(colnames(internal_training)[p]) print(boxplot) }

t1 <- as.data.frame(table(internal_training$classe, internal_training$user_name))
colnames(t1)[1] <- "classe"

# ggplot(t1, aes(x = classe, y = Freq)) + geom_bar(stat = 'identity') +
# facet_grid(.~Var2)
```

Exploratory modeling

With so many variables in the data set, I decided to do some exploratory modeling steps to get to know the data better. I started by fitting a classification tree since that would be somewhat interpretable. Timestamp related variables showed up in this tree but my inuition was that these were perhaps not very explanatory and could cause issues as predictors for the final test set. I decided to create several visuals of these timestamp variables to look for patterns, or the absence thereof. Not seeing a pattern between the timestamp variables and classe, I decided to remove these predictors so that more interpretable variables would be used in the prediction model. I then fitted a new classification tree model without the timestamp variables. The dendogram plot was more interpretable, but the accuracy ot the method was very low. So, my next step was to try a random forest model to combine by voting across many tree models. In this model, I used 10-fold cross validation to minimize overfitting.

```
# Start by fitting a classification tree for initial model exploration since
# it is fairly interpretable
mod_rpart <- train(classe ~ ., method = "rpart", data = internal_training)
```

```
## Loading required package: rpart
```

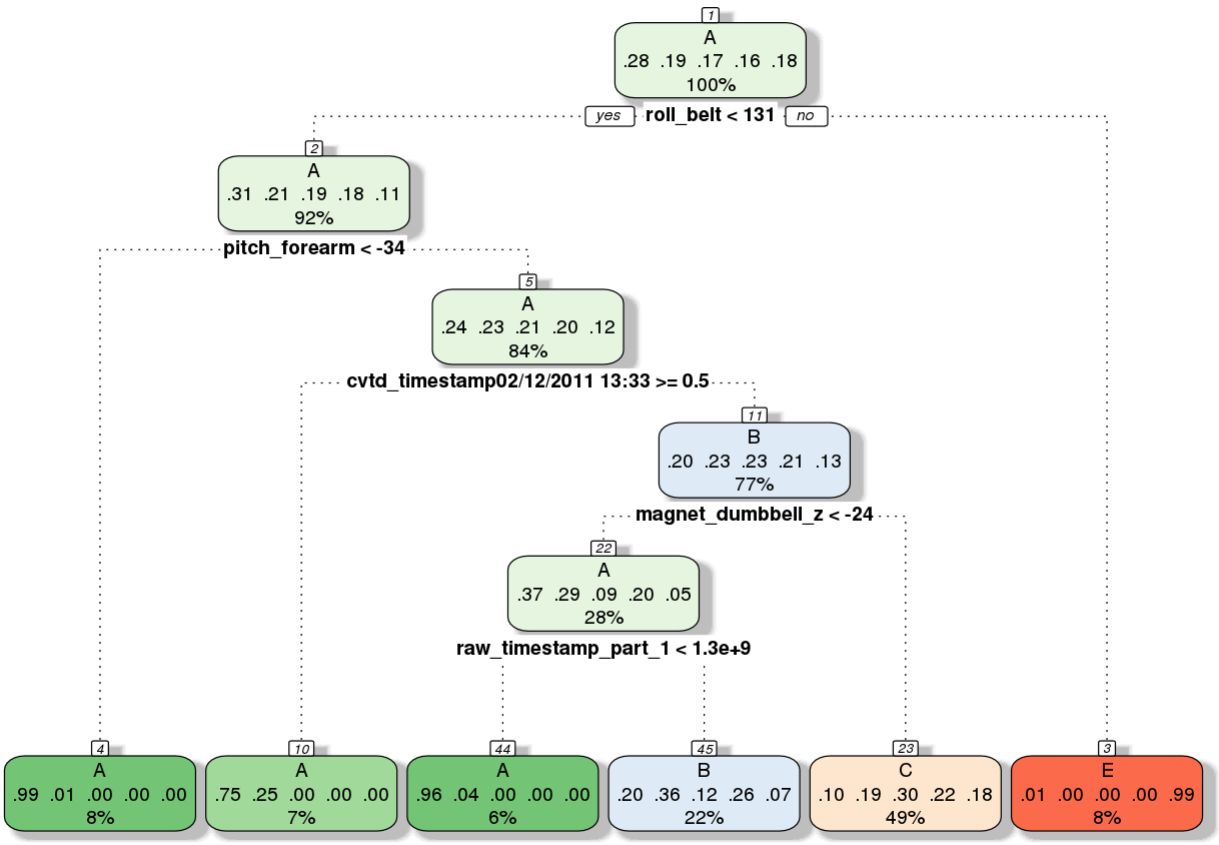
```
print(mod_rpart$finalModel)
```

```
## n= 16680
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 16680 11937 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 15295 10560 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1334    10 A (0.99 0.0075 0 0 0) *
##      5) pitch_forearm>=-33.95 13961 10550 A (0.24 0.23 0.21 0.2 0.12)
##       10) cvtd_timestamp02/12/2011 13:33>=0.5 1123   284 A (0.75 0.25 0 0 0) *
##       11) cvtd_timestamp02/12/2011 13:33< 0.5 12838  9904 B (0.2 0.23 0.23 0.21 0.13)
##         22) magnet_dumbbell_z< -24.5 4718  2988 A (0.37 0.29 0.093 0.2 0.052)
##           44) raw_timestamp_part_1< 1.3228e+09 1031    44 A (0.96 0.039 0.0039 0 0) *
##           45) raw_timestamp_part_1>=1.3228e+09 3687  2376 B (0.2 0.36 0.12 0.26 0.066) *
##         23) magnet_dumbbell_z>=-24.5 8120  5649 C (0.1 0.19 0.3 0.22 0.18) *
##    3) roll_belt>=130.5 1385     8 E (0.0058 0 0 0 0.99) *
```

```
library(rattle)
```

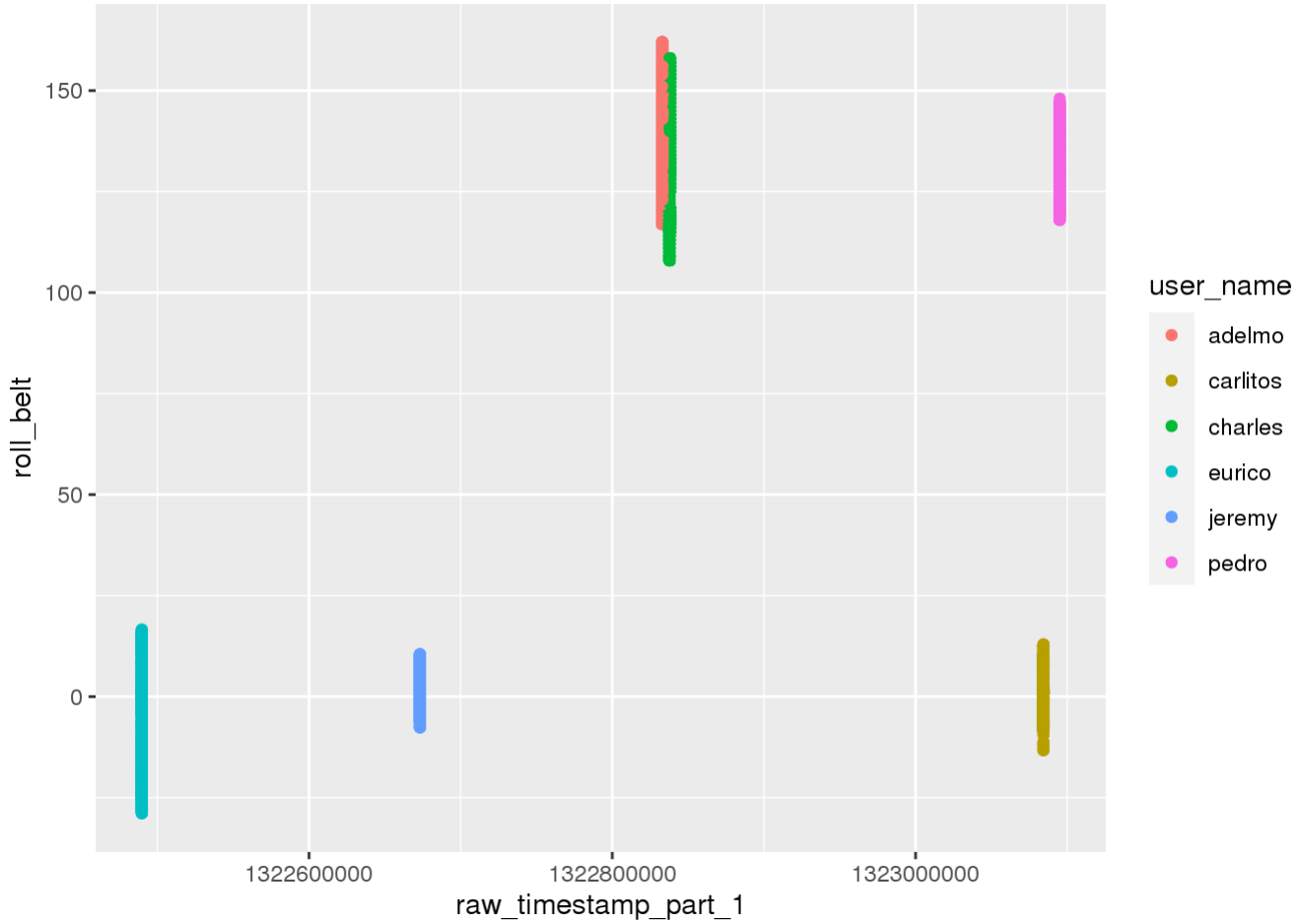
```
## Rattle: A free graphical interface for data science with R.
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(mod_rpart$finalModel)
```



Rattle 2020-Apr-20 00:12:02 giywg

```
#### Look at classe by timestamp since the timestamp variable come up in the
#### first exploratory model but are difficult to interpret
ggplot(internal_training, aes(x = raw_timestamp_part_1, y = roll_belt, color = user_name)) +
    geom_point()
```

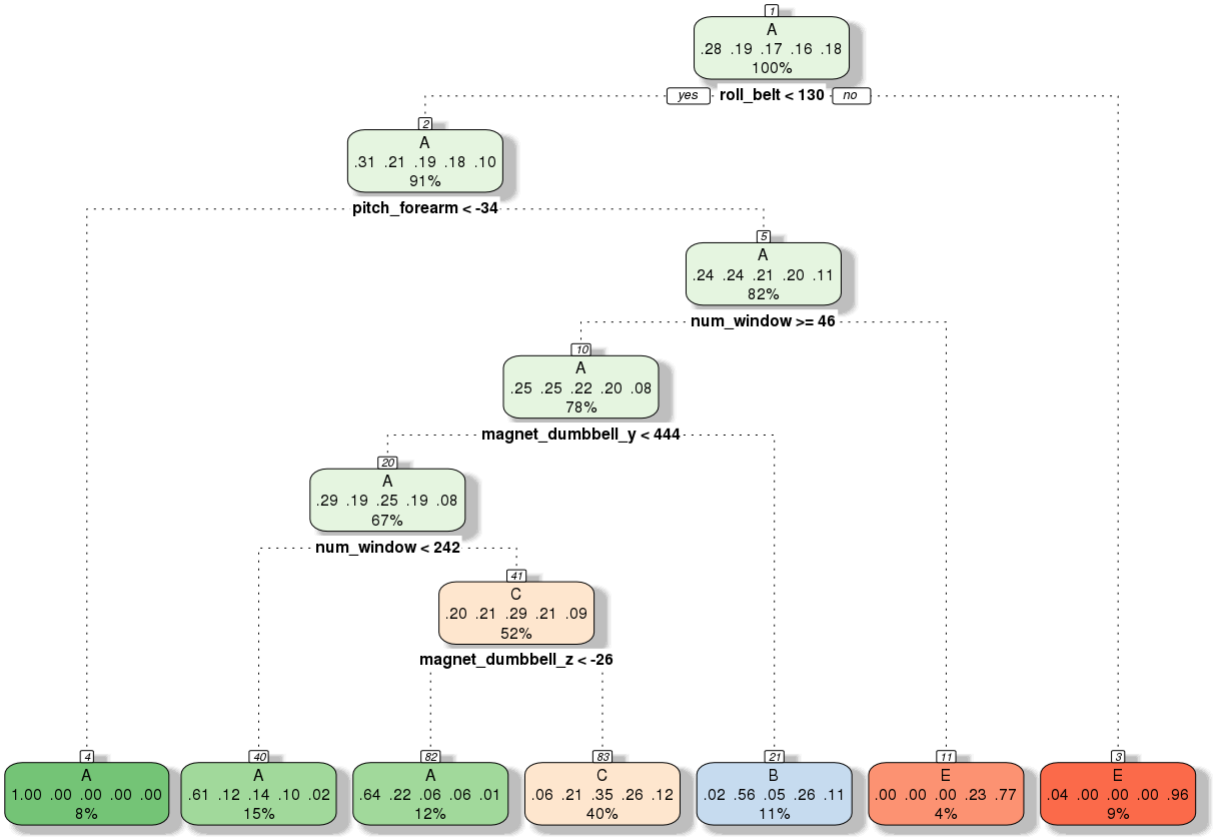



```
ggplot(internal_training, aes(x = raw_timestamp_part_2, y = roll_belt, color = user_name)) +
    geom_point()
```


```
# ggplot(internal_training, aes(x = raw_timestamp_part_2, y = roll_belt,
# color = classe)) + geom_point(alpha = 0.4) ggplot(internal_training, aes(x
# = cvtd_timestamp, y = roll_belt, color = user_name)) + geom_point() +
# theme(axis.text.x = element_text(angle = 45))

# Seeing that the timestamp variables do not seems to correlate with the
# classe variable, remove these so that more interpretable variables are
# used in prediction
predictors_to_exclude <- c("raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp")

internal_training2 <- internal_testing %>% select_if(names(.) %nin% predictors_to_exclude)

# Fit a new classification tree model without the timestamp variables as
# predictors, predict values in the internal testing set, and check accuracy
# value and confusion matrix
mod_rpart <- train(classe ~ ., method = "rpart", data = internal_training2)
print(mod_rpart$finalModel)
```

```
## n= 2942
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2942 2105 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 129.5 2664 1839 A (0.31 0.21 0.19 0.18 0.1)
##      4) pitch_forearm< -33.85 244    0 A (1 0 0 0 0) *
##      5) pitch_forearm>=-33.85 2420 1839 A (0.24 0.24 0.21 0.2 0.11)
##       10) num_window>=46 2306 1725 A (0.25 0.25 0.22 0.2 0.081)
##         20) magnet_dumbbell_y< 444 1971 1397 A (0.29 0.19 0.25 0.19 0.077)
##           40) num_window< 241.5 442  172 A (0.61 0.12 0.14 0.1 0.025) *
##           41) num_window>=241.5 1529 1093 C (0.2 0.21 0.29 0.21 0.092)
##             82) magnet_dumbbell_z< -26.5 360  131 A (0.64 0.22 0.064 0.064 0.014) *
##             83) magnet_dumbbell_z>=-26.5 1169  756 C (0.064 0.21 0.35 0.26 0.12) *
##         21) magnet_dumbbell_y>=444 335  146 B (0.021 0.56 0.051 0.26 0.11) *
##       11) num_window< 46 114   26 E (0 0 0 0.23 0.77) *
##    3) roll_belt>=129.5 278   12 E (0.043 0 0 0 0.96) *
```

```
fancyRpartPlot(mod_rpart$finalModel)
```



Rattle 2020-Apr-20 00:12:06 giywg

```
pred_rpart <- predict(mod_rpart, newdata = internal_testing)
confusionMatrix(pred_rpart, internal_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 743 135  83  69  16
##          B   7 189  17  86  36
##          C  75 245 413 301 135
##          D   0   0   0   0   0
##          E  12   0   0  26 354
##
## Overall Statistics
##
##                Accuracy : 0.577
##                  95% CI : (0.559, 0.595)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.46
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.888   0.3322    0.805    0.000    0.654
## Specificity             0.856   0.9385    0.689    1.000    0.984
## Pos Pred Value          0.710   0.5642    0.353      NaN    0.903
## Neg Pred Value          0.950   0.8542    0.944    0.836    0.927
## Prevalence              0.285   0.1934    0.174    0.164    0.184
## Detection Rate          0.253   0.0642    0.140    0.000    0.120
## Detection Prevalence    0.356   0.1139    0.397    0.000    0.133
## Balanced Accuracy       0.872   0.6353    0.747    0.500    0.819
```

```
# The classification tree model had rather low accuracy, so try random
# forest model which will combine by voting across many tree models, predict
# values in the internal testing set, and check accuracy value and confusion
# matrix I chose to implement cross validation with 10 folds to mitigate
# overfitting in the training data thereby setting up for a better result in
# the testing data.
mod_rf <- train(classe ~ ., method = "rf", data = internal_training, trControl = trainControl(me
thod = "cv",
    number = 10))
```

```
## Loading required package: randomForest
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

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

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

```
pred_rf <- predict(mod_rf, newdata = internal_testing2)
confusionMatrix(pred_rf, internal_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 837   0   0   0   0
##          B   0 568   0   0   0
##          C   0   1 513   0   0
##          D   0   0   0 482   0
##          E   0   0   0   0 541
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.998, 1)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.998    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    0.998    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.285    0.193    0.174    0.164    0.184
## Detection Rate          0.285    0.193    0.174    0.164    0.184
## Detection Prevalence    0.285    0.193    0.175    0.164    0.184
## Balanced Accuracy       1.000    0.999    1.000    1.000    1.000
```

This random forest model predicts very well in the internal testing set, so I will use it to predict the classe variable in the final testing set to answer the quiz. Based on the fact that a) I used cross validation to reduce overfitting and b) I had very high accuracy in my internal testing data set, I expect that I will have very low error in predicting in the testing set for the quiz.

Final model

```
pred_rf_final <- predict(mod_rf, newdata = testing)
pred_rf_final
```

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

Turning in my answers for the quiz, I found that I got 100% right, verifying that my machine learning prediction method has performed well.