Group Project: Early Alert with LMS Data

INFO 4100 Learning Analytics

[[Jacob Bee Ho Brown (jdb393), Emma Condie (erc97), Ellie Garell (emg223), Sanithia Edwards (se273), Yue Ji (yj359), Julia Lehman (jsl357), Heitung Sun (hs835)]]

# Introduction

**Goals:** The goal of this project is to learn how to work with raw Learning Management System (LMS) data and apply some of the prediction skills you have learned so far. You will develop a one-day early warning system for students who miss a graded submission. I am sharing with you an export of the class’s edX log data thus far. I have anonymized the dataset and performed minimal data cleaning, leaving plenty of real-world messiness for you to tackle here. As always, you should start by getting to know the datasets. In this case, you should be able to really understand what is going on because it is YOUR data. In fact, you can navigate to the relevant pages on edX to see what page/action the data refers to.

**Group Project:** This is a group project and I expect you to work as a team to come up with the best possible prediction accuracy. Your team will submit one common solution (note that EACH team member will need to submit the knitted Word doc on edx to get credit like with the first group project).

**Try Your Best:** All members of the TWO teams that achieve the highest F1 scores will receive an extra credit point, and their solutions will be featured. To be eligible, your prediction problem needs to be set up correctly (i.e. everything else needs to be correct).

# Step 1: Understand the data

There are three datasets which can be connected using the hash\_id column (a hashed version of the user id) and I am giving you links to the official documentation which you should read to understand the data better:

1. Clickstream data (1 row per student per action): [click for documentation](https://edx.readthedocs.io/projects/devdata/en/stable/internal_data_formats/tracking_logs.html#tracking-logs)
2. Module States (1 row per student per accessed content): original name [courseware-studentmodule (click for doumentation)](https://edx.readthedocs.io/projects/devdata/en/stable/internal_data_formats/sql_schema.html#courseware-studentmodule)
3. Assessment grades (1 row per assessment per student)

I have already converted date-time objects into a numeric timestamp for you.

To look up what pages URLs refer to (works for browser events, not server events), you can paste the URL into your browser. This should work for most URLs. I recommend doing this to be able to engineer more meaningful features.

*Question 1:* In the space below, explore each dataset using head(), n\_distinct(data$some\_id), summary(), table(data$column). You can also plot the distribution of variables with histograms or boxplots. Check out the data documentation linked above to understand the meaning of each column.

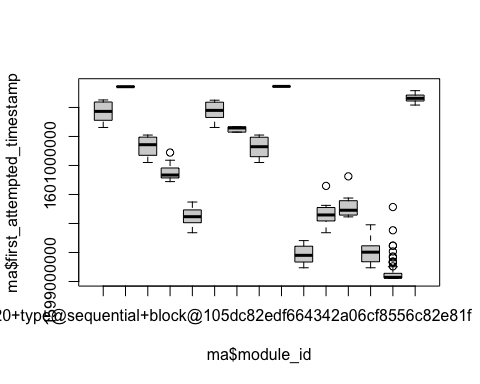
###############################################   
###### BEGIN INPUT: Explore each dataset ######   
###############################################  
  
# Exploring Clickstreams  
head(cl)  
n\_distinct(cl$hash\_id)  
n\_distinct(cl$survey\_id)  
n\_distinct(cl$time)  
n\_distinct(cl$name)  
n\_distinct(cl$event\_type)  
n\_distinct(cl$referer)  
n\_distinct(cl$page)  
n\_distinct(cl$event\_source)  
n\_distinct(cl$event)  
n\_distinct(cl$timestamp)  
#summary(cl)  
table(cl$hash\_id)  
#table(cl$survey\_id)  
#table(cl$time)  
table(cl$name)  
table(cl$event\_type)  
table(cl$referer)  
table(cl$page)  
table(cl$event\_source)  
cl$event  
table(cl$timestamp)  
  
# Exploring Module States  
head(m)  
n\_distinct(m$hash\_id)  
n\_distinct(m$module\_type)  
n\_distinct(m$grade)  
n\_distinct(m$created)  
n\_distinct(m$modified)  
n\_distinct(m$max\_grade)  
n\_distinct(m$module\_id)  
n\_distinct(m$created\_timestamp)  
n\_distinct(m$modified\_timestamp)  
#summary(m)  
#table(cl$hash\_id)  
table(m$module\_type)  
table(m$grade)  
table(m$created)  
table(m$modified)  
table(m$max\_grade)  
table(m$module\_id)  
table(m$created\_timestamp)  
table(m$modified\_timestamp)  
  
# Exploring Assessment grades  
# add code here  
head(a)  
n\_distinct(a$hash\_id)  
n\_distinct(a$usage\_key)  
n\_distinct(a$earned\_graded)  
n\_distinct(a$first\_attempted)  
n\_distinct(a$created)  
n\_distinct(a$created\_timestamp)  
n\_distinct(a$modified\_timestamp)  
n\_distinct(a$first\_attempted\_timestamp)  
#summary(a)  
#table(a$hash\_id)  
#table(a$usage\_key)  
table(a$earned\_graded)  
table(a$first\_attempted)  
table(a$created)  
table(a$created\_timestamp)  
table(a$modified\_timestamp)  
table(a$first\_attempted\_timestamp)  
  
###############################################  
###############################################

You may notice that it would be helpful to combine the information about grades and time of first attempt with the module state data. Below I make this join for you. See that only ‘sequential’ modules have grade data associated with them. The boxplot shows when the different sequentials (containing problems) were attempted. This gives you an idea of the order of problems in the course.

ma = m %>% left\_join(  
 a %>% select(hash\_id:possible\_graded, first\_attempted\_timestamp),   
 by = c("hash\_id"="hash\_id", "module\_id"="usage\_key")  
)  
  
# Only sequential modules have a grade associated with them  
table(ma$module\_type, ma$first\_attempted\_timestamp>0)

##   
## TRUE  
## chapter 0  
## course 0  
## openassessment 0  
## problem 0  
## sequential 1163  
## video 0

# We see that assignments were due (submitted) at different times  
boxplot(ma$first\_attempted\_timestamp ~ ma$module\_id)



# Step 2: Define a prediction task

Recall the guidelines for defining a good prediction problem covered in the Handbook chapter on prediction. You are looking for something actionable (an opportunity to intervene) and a situation that repeats (so the prediction can be useful in the future). The tradeoff with the dataset you have here is that on the one hand it is very relevant to you but on the other hand it is relatively small. Still, the data is fine-grained and sufficiently messy to give you a taste of LMS data analysis.

The prediction problem for this project is to build a one-day early warning system for missing a graded submission. Specifically, **your goal is to predict one day before the submission deadline, if a student will forget to submit an assignment**, so that the system can send a reminder. As you may have noticed during the data exploration phase above (if not, you should go back and examine this), there are several graded submissions and some students missed one or more of them. We define **missing a submission** as having an NA for first\_attempted\_timestamp but of course only for those that are past due.

### Instructions

1. Treat each graded assignment as a prediction task (thus there are x\*n prediction opportunities where x = number of graded assignments and n = 31 students).
2. Create a dataset that has 1 row per student per graded assessment with the binary outcome (did they MISS it? yes/no) and several predictors (see next tip)
3. Predictors (i.e. features) need to be engineered with data from **24hrs before each assignment is due**, which of course varies across assignments; that means you have much more information to predict later assignments than earlier ones
4. Once your dataset is ready, split it into a training and a test set
5. Train a prediction model on the training data; you can try out any of the ones we have covered in the prediction homework and Random Forest
6. Keep tuning your model choice, model parameters (if any), and feature engineering
7. Finally, test your prediction accuracy on the test set

# Step 3: Getting you started

## Create the outcome variable

**Identify the graded assessments and whether a student did NOT submit**. Recall we want to have a *warning* system, so the outcome should be the negative action.

Get the outcome for each graded assignment. Figure out the deadline for each and compute the timestamp for 24hrs prior to the deadline. You probably want to use the ma dataset I created for you above.

The following table helps you see the various graded assignments to consider. We keep only those where possible\_graded > 0. **I define the deadline as the 90th percentile of submissions (you may use this simplification).**

ma %>%   
 filter(possible\_graded > 0) %>%  
 group\_by(module\_id) %>%   
 summarise(  
 deadline = quantile(first\_attempted\_timestamp, probs = .9, na.rm=T),  
 p\_unsubmitted = mean(is.na(first\_attempted\_timestamp))  
 ) %>%   
 arrange(deadline)

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 17 x 3  
## module\_id deadline p\_unsubmitted  
## <chr> <dbl> <dbl>  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0638  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0215  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0753  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0   
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0526  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0211  
## 7 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0957  
## 8 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0426  
## 9 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0753  
## 10 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0737  
## 11 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0326  
## 12 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0761  
## 13 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0426  
## 14 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.571   
## 15 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.909   
## 16 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… NA 1   
## 17 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… NA 1

ma %>%   
 filter((possible\_graded > 0) &(!is.na(first\_attempted\_timestamp))) %>%  
 group\_by(module\_id) %>%   
 summarise(  
 deadline = quantile(first\_attempted\_timestamp, probs = .9, na.rm=T),  
 p\_unsubmitted = sum(first\_attempted\_timestamp<deadline-60 \* 60 \* 24)/n()  
 ) %>%   
 arrange(deadline)

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 15 x 3  
## module\_id deadline p\_unsubmitted  
## <chr> <dbl> <dbl>  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.841  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.725  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.709  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.716  
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.656  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.656  
## 7 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.682  
## 8 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.622  
## 9 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.593  
## 10 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.352  
## 11 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.607  
## 12 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.576  
## 13 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.644  
## 14 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0   
## 15 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0

Now you know which assessments (module\_ids) to target. **Be sure to kick out the one with p\_unsubmitted > 0.5**; They were not due yet when the export was created.

*Question 2:* Now build a dataset with an indicator for each person and each of these module\_ids with 1=unsubmitted, 0=submitted. Keep track of the deadline: you only want to use features based on data up to 24hrs before it (i.e. 24 \* 60 \* 60 seconds).

###############################################   
####### BEGIN INPUT: Define outcome ###########  
###############################################  
  
# edit ma to include a column with the module deadline  
# and a binary column for the probability for whether the student submits on time  
ma\_edited = ma %>%   
 filter(possible\_graded > 0) %>%  
 group\_by(module\_id) %>%   
 summarise(  
 deadline = quantile(first\_attempted\_timestamp, probs = .9, na.rm=T),  
 p\_unsubmitted = mean(is.na(first\_attempted\_timestamp))  
 ) %>%   
 arrange(deadline) %>%   
 filter(p\_unsubmitted <=0.5)

## `summarise()` ungrouping output (override with `.groups` argument)

print(ma\_edited)

## # A tibble: 13 x 3  
## module\_id deadline p\_unsubmitted  
## <chr> <dbl> <dbl>  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0638  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0215  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0753  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0   
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0526  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0211  
## 7 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0957  
## 8 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0426  
## 9 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0753  
## 10 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0737  
## 11 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0326  
## 12 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0761  
## 13 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+… 1.60e9 0.0426

# combine these new columns with ma   
ma\_combined = ma %>% left\_join(  
 ma\_edited %>% select(module\_id:deadline,p\_unsubmitted),   
 by = c("module\_id"="module\_id")  
)  
  
# create outcome dataset  
outcome = ma\_combined %>%   
 filter((possible\_graded > 0)&(p\_unsubmitted <=0.5)) %>%  
 group\_by(hash\_id, module\_id) %>%   
 summarise(  
 unsubmitted = (is.na(first\_attempted\_timestamp))\*1)

## `summarise()` regrouping output by 'hash\_id' (override with `.groups` argument)

# add deadline column from ma\_edited  
outcome = outcome %>% left\_join(  
 ma\_edited %>% select(module\_id:deadline),   
 by = c("module\_id"="module\_id")  
 )  
outcome

## # A tibble: 1,219 x 4  
## # Groups: hash\_id [95]  
## hash\_id module\_id unsubmitted deadline  
## <chr> <chr> <dbl> <dbl>  
## 1 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 2 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 3 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 4 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 5 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 6 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 7 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 8 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 9 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## 10 046b9f96c0155a63c… block-v1:Cornellx+INFO4100+Fall2020… 0 1.60e9  
## # … with 1,209 more rows

###############################################   
###############################################

## Feature Engineering

**For each graded assessment, identify what data is appropriate for feature engineering**

Before you start feature engineering, you need to constrain the data for **each** assessment.

Remember that the dataset we are aiming for has 1 row per person and assessment with several feature variables and one outcome variable. You created the outcome above. Now you need to create the appropriate features to join. I’m giving you an example for using deadline = 1600304996 and creating 2 basic features from the clickstream. You should try to create a lot more features, including complex ones, that can use the clickstream or other datasets (but remember the timing constraint).

# PROF CODE EXAMPLE FEATURE  
secs\_day = 60 \* 60 \* 24  
example\_deadline = 1600304996  
  
example\_features = cl %>%   
 filter(timestamp < example\_deadline - secs\_day) %>%  
 group\_by(hash\_id) %>%  
 summarise(  
 num\_events = n(),  
 num\_seq\_goto = sum(event\_type=="seq\_goto")  
 )

## `summarise()` ungrouping output (override with `.groups` argument)

head(example\_features)

## # A tibble: 6 x 3  
## hash\_id num\_events num\_seq\_goto  
## <chr> <int> <int>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 287 10  
## 2 051123be58fb0985c3578ad867ec22e2 407 6  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 190 2  
## 4 08258abf7826f7c758ffe1ecfc468f17 695 27  
## 5 0ca99849369b2ed808ee3060afe52eb6 829 64  
## 6 0df790f3a775479a223f91ff3d888994 344 0

*Question 3:* Engineer features for each student and assessment, subject to the timing constraint.

###############################################   
###### BEGIN INPUT: Engineer features #########  
###############################################  
  
# want dataset with 1 row per person and assessment, with several feature variables, and 1 outcome variable  
features = outcome  
ma\_features = data.frame()  
cl\_features = data.frame()  
# for each graded assessment, select data to use for feature eng based on module ddl-1 day  
for (row in 1:nrow(ma\_edited)) #nrow(ma\_edited)==13  
{  
 module\_ddl = as.numeric(ma\_edited[row,"deadline"])  
 moduleid = as.character(ma\_edited[row,"module\_id"])  
 ### modules that are available before ddl + 1 day, including the current module  
 #filter out assignments that weren't due yet and data from the window 24hours before the deadline  
 example\_features = ma\_combined %>%   
 #filter((possible\_graded >0 ) & (deadline <= module\_ddl) & (first\_attempted\_timestamp < module\_ddl - secs\_day)) %>%  
 filter((possible\_graded >0 ) & (deadline <= module\_ddl - secs\_day)) %>%  
 group\_by(hash\_id) %>%  
 summarise(  
 #add our features for assessments here!  
 #num\_assignments = n(),  
 diff = mean(deadline + secs\_day - first\_attempted\_timestamp),  
 #miss\_ddl = sum(is.na(first\_attempted\_timestamp)),  
 miss\_ddl = sum(deadline - first\_attempted\_timestamp<0),  
 #avg\_miss\_ddl = sum(deadline - first\_attempted\_timestamp<0)/n(),  
 avg\_grade = sum(earned\_graded)/sum(possible\_graded),  
 #perf\_score = sum(earned\_graded==possible\_graded),  
 avg\_perf\_score = sum(earned\_graded==possible\_graded)/n(),  
 #grade = earned\_graded/possible\_graded,  
 #time\_between\_modification = mean(modified\_timestamp-created\_timestamp),  
 # grade\_above\_avg = (earned\_graded/possible\_graded) > mean(earned\_graded)/possible\_graded, # earned\_graded/possible\_graded) == mean(earned\_graded)/possible\_graded  
 #more\_than\_1\_possible\_point = sum(possible\_graded>1.0),  
 #prev\_attemp\_last\_hr = sum(first\_attempted\_timestamp > deadline - 3600), # not sure about this one  
 #submitted\_latest = !is.na(first\_attempted\_timestamp)  
 )  
 #feature checking if students submitted aldready the current assignment one day before the calculated deadline  
   
 problem = ma\_combined %>%   
 filter((module\_type=="problem") & (modified\_timestamp <= module\_ddl - secs\_day))%>%  
 group\_by(hash\_id) %>%  
 summarise(  
 #full\_grade = sum(grade==max\_grade),  
 #time\_spent = mean(modified\_timestamp - created\_timestamp)  
 )  
 video\_info = ma\_combined %>%   
 filter((module\_type=="video") & (modified\_timestamp <= (module\_ddl - secs\_day))) %>%  
 group\_by(hash\_id) %>%  
 summarise(  
 #num\_videos = n(),  
 #time\_on\_videos = sum(modified\_timestamp-created\_timestamp),  
 #time\_per\_video\_avg = (sum(modified\_timestamp-created\_timestamp))/n()  
 )  
 #video\_info  
 features\_per\_module = features[c(-3,-4)] %>% filter(module\_id == moduleid) %>% left\_join(  
 example\_features,   
 by = c("hash\_id"="hash\_id")  
 )  
 # features\_per\_module = features\_per\_module %>% left\_join(  
 # video\_info,   
 # by = c("hash\_id"="hash\_id")  
 # )  
 # features\_per\_module = features\_per\_module %>% left\_join(  
 # problem,   
 # by = c("hash\_id"="hash\_id")  
 # )  
   
 submitted = ma\_combined %>%   
 filter((possible\_graded >0 ) & (module\_id == moduleid)) %>%  
 group\_by(hash\_id) %>%  
 summarise(  
 #add our features for assessments here!  
 #num\_assignments = n(),  
 #diff = mean(deadline + secs\_day - first\_attempted\_timestamp),  
 not\_submitted\_by\_ddl = (!(first\_attempted\_timestamp < module\_ddl - secs\_day))\*1,  
 #miss\_ddl = sum(deadline - first\_attempted\_timestamp<0),  
 #avg\_miss\_ddl = sum(deadline - first\_attempted\_timestamp<0)/n(),  
 #avg\_grade = sum(earned\_graded)/sum(possible\_graded),  
 #perf\_score = sum(earned\_graded==possible\_graded),  
 #avg\_perf\_score = sum(earned\_graded==possible\_graded)/n(),  
 #grade = earned\_graded/possible\_graded,  
 #time\_between\_modification = mean(modified\_timestamp-created\_timestamp),  
 # grade\_above\_avg = (earned\_graded/possible\_graded) > mean(earned\_graded)/possible\_graded, # earned\_graded/possible\_graded) == mean(earned\_graded)/possible\_graded  
 #more\_than\_1\_possible\_point = sum(possible\_graded>1.0),  
 # prev\_attemp\_last\_hr = sum(first\_attempted\_timestamp > deadline - 3600), # not sure about this one  
 )  
 print(submitted)  
 features\_per\_module = features\_per\_module %>% left\_join(  
 submitted,  
 by = c("hash\_id"="hash\_id")  
 )  
   
 ma\_features = rbind(ma\_features, features\_per\_module)  
   
   
 cl\_features\_before\_module\_ddl = cl %>%   
 filter(timestamp < module\_ddl - secs\_day) %>%  
 group\_by(hash\_id) %>%  
 summarise(  
 # add our features for clickstream data here!  
 num\_events = log2(n()),  
 #num\_seq\_goto = sum(event\_type=="seq\_goto"),  
 #num\_module = sum(str\_detect(page,moduleid)),  
 #pause\_play\_video = sum(event\_type=="play\_video"),  
 #num\_pause\_video = sum(event\_type=="pause\_video"),  
 num\_video\_interactions = sum(str\_detect(event\_type, "video")),  
 #num\_prob\_check = sum(event\_type=="problem\_check"),  
 #num\_show\_ans = sum(event\_type=="showanswer"),  
 num\_links\_clicked = sum(name=="edx.ui.lms.link\_clicked"),  
 #num\_ref\_from\_mail = sum(str\_detect(referer, "mail")),  
 #time\_spent = mean(diff(timestamp)),  
 num\_events\_past\_week = sum(timestamp > module\_ddl - secs\_day\*7),  
 num\_events\_past\_three\_day= sum(timestamp > module\_ddl - secs\_day\*4),  
 num\_events\_past\_day = sum(timestamp > module\_ddl - secs\_day\*2),  
 # add more event types (follow URL to figure out what they are)  
 # use page  
 # use name  
 )  
 #print(cl\_features\_before\_module\_ddl)  
 cl\_features\_per\_module = features[c(-3,-4)] %>% filter(module\_id == moduleid) %>% left\_join(  
 cl\_features\_before\_module\_ddl,   
 by = c("hash\_id"="hash\_id")  
 )  
 #print(features\_per\_module)  
 cl\_features = rbind(cl\_features, cl\_features\_per\_module)  
}

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 94 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 0  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 0  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 NA  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e NA  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 84 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 93 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 0  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 1  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 83 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 93 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 0  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 NA  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e NA  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 83 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 95 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 1  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 0  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 85 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 95 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 1  
## 7 0e951c83667194cc73f4610350b72945 NA  
## 8 0fefde5cc3b1c65dba071888ce9e522e 0  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 85 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 95 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 1  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 0  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 85 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 94 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 0  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 1  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 0  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 84 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 94 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 1  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 1  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 0  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe 1  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 84 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 93 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 1  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 1  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 0  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe NA  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 83 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 95 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 0  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 1  
## 5 0ca99849369b2ed808ee3060afe52eb6 1  
## 6 0df790f3a775479a223f91ff3d888994 NA  
## 7 0e951c83667194cc73f4610350b72945 1  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f NA  
## # … with 85 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 92 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 1  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 1  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 1  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 82 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 92 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 1  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 NA  
## 4 08258abf7826f7c758ffe1ecfc468f17 1  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 1  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 1  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 82 more rows

## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)  
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 94 x 2  
## hash\_id not\_submitted\_by\_ddl  
## <chr> <dbl>  
## 1 046b9f96c0155a63c1262c8bd0a327e0 1  
## 2 051123be58fb0985c3578ad867ec22e2 1  
## 3 06b0110dadf9c128fd7550a2cd8a2bd4 1  
## 4 08258abf7826f7c758ffe1ecfc468f17 0  
## 5 0ca99849369b2ed808ee3060afe52eb6 0  
## 6 0df790f3a775479a223f91ff3d888994 0  
## 7 0e951c83667194cc73f4610350b72945 0  
## 8 0fefde5cc3b1c65dba071888ce9e522e 0  
## 9 1403b390b640882382a0bb37983933fe 0  
## 10 14809cc5e83174cc0cb9d414bbb4fc9f 0  
## # … with 84 more rows

## `summarise()` ungrouping output (override with `.groups` argument)

#print(cl\_features)  
  
# how do we save each set of features for each module into one dataset?   
#print(ma\_features)  
all\_features = left\_join(ma\_features, cl\_features, by = c("hash\_id"="hash\_id", "module\_id"="module\_id"))  
all\_features

## # A tibble: 1,219 x 13  
## # Groups: hash\_id [95]  
## hash\_id module\_id diff miss\_ddl avg\_grade avg\_perf\_score not\_submitted\_b…  
## <chr> <chr> <dbl> <int> <dbl> <dbl> <dbl>  
## 1 046b9f… block-v1… NA NA NA NA 0  
## 2 051123… block-v1… NA NA NA NA 0  
## 3 06b011… block-v1… NA NA NA NA 0  
## 4 08258a… block-v1… NA NA NA NA 0  
## 5 0ca998… block-v1… NA NA NA NA 0  
## 6 0df790… block-v1… NA NA NA NA NA  
## 7 0e951c… block-v1… NA NA NA NA 0  
## 8 0fefde… block-v1… NA NA NA NA NA  
## 9 1403b3… block-v1… NA NA NA NA 0  
## 10 14809c… block-v1… NA NA NA NA 0  
## # … with 1,209 more rows, and 6 more variables: num\_events <dbl>,  
## # num\_video\_interactions <int>, num\_links\_clicked <int>,  
## # num\_events\_past\_week <int>, num\_events\_past\_three\_day <int>,  
## # num\_events\_past\_day <int>

###############################################  
###############################################

# Step 4: Split your dataset

*Question 4:* We would like train the model on earlier assessments in order to make early alert predictions for later ones. As the hold-out test set, designate the four (4) last assessments (i.e. with the 4 latest computed deadlines, or the last 4 periods; same thing). You will use all the remaining data to train. Note that this may not be the best setup for all applications (e.g. if we wanted to use the model at the start of the course next year, but it is a reasonable approach if we wanted to use the model for the rest of this course offering). Identify the module\_ids of the last four assignments, put data associated with their periods in the test dataset. Take all the remaining data (earlier periods excl the last 4) and put it in the train dataset.

###############################################   
######## BEGIN INPUT: Split dataset ###########  
###############################################  
  
# Identify last 4 periods for testing  
# add code here  
all\_features = left\_join(ma\_features, cl\_features, by = c("hash\_id"="hash\_id", "module\_id"="module\_id"))  
all\_features = left\_join(all\_features, outcome, by = c("hash\_id"="hash\_id", "module\_id"="module\_id"))  
sum(is.na(all\_features[["unsubmitted"]])) #==0

## [1] 0

sum(is.na(all\_features[["miss\_ddl"]])) # 96

## [1] 307

sum(is.na(all\_features[["num\_events"]])) # 11

## [1] 11

sum(all\_features[["unsubmitted"]]==1) #63

## [1] 63

#replace\_na(1) will result in classification tree not working  
for (col in colnames(all\_features)){  
 #if (col=="miss\_ddl" || col=="unsubmitted"){  
 # all\_features[[col]] = all\_features[[col]] %>% replace\_na(1)  
 #}  
 #else{  
 all\_features[[col]] = all\_features[[col]] %>% replace\_na(0)  
 #}  
}  
#all\_features = all\_features %>% drop\_na()  
#all\_features\_dropped\_na  
#all\_features = all\_features %>% dplyr::mutate(x = replace\_na(diff, 0))  
#$diff = all\_features$diff %>% replace\_na(0)  
#all\_features$avg\_perf\_score = all\_features$avg\_perf\_score %>% replace\_na(0)  
#all\_features$num\_events = all\_features$num\_events %>% replace\_na(0)  
all\_features

## # A tibble: 1,219 x 15  
## # Groups: hash\_id [95]  
## hash\_id module\_id diff miss\_ddl avg\_grade avg\_perf\_score not\_submitted\_b…  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 046b9f… block-v1… 0 0 0 0 0  
## 2 051123… block-v1… 0 0 0 0 0  
## 3 06b011… block-v1… 0 0 0 0 0  
## 4 08258a… block-v1… 0 0 0 0 0  
## 5 0ca998… block-v1… 0 0 0 0 0  
## 6 0df790… block-v1… 0 0 0 0 0  
## 7 0e951c… block-v1… 0 0 0 0 0  
## 8 0fefde… block-v1… 0 0 0 0 0  
## 9 1403b3… block-v1… 0 0 0 0 0  
## 10 14809c… block-v1… 0 0 0 0 0  
## # … with 1,209 more rows, and 8 more variables: num\_events <dbl>,  
## # num\_video\_interactions <dbl>, num\_links\_clicked <dbl>,  
## # num\_events\_past\_week <dbl>, num\_events\_past\_three\_day <dbl>,  
## # num\_events\_past\_day <dbl>, unsubmitted <dbl>, deadline <dbl>

last\_four = ma\_edited$module\_id[10:13]  
#print(last\_four)  
  
modelingTime = 1602288653 + 1000000   
K = 1000000  
T = 1/2  
#all\_features$weight = K\*exp(-(modelingTime - all\_features$deadline)/T)  
all\_features$weight = 1/((modelingTime - all\_features$deadline)/K)  
# Split the dataset into train and test based on the module\_ids or periods  
 test = all\_features %>% filter(module\_id %in% last\_four)  
 print(test)

## # A tibble: 373 x 16  
## # Groups: hash\_id [95]  
## hash\_id module\_id diff miss\_ddl avg\_grade avg\_perf\_score not\_submitted\_b…  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 046b9f… block-v1… 192892. 1 0.955 0.667 1  
## 2 051123… block-v1… 238216. 0 0.857 0.778 0  
## 3 06b011… block-v1… -3831. 5 0.904 0.556 1  
## 4 08258a… block-v1… 263935. 0 0.915 0.667 1  
## 5 0ca998… block-v1… 415542. 0 0.994 0.889 1  
## 6 0df790… block-v1… 0 0 0.644 0.556 0  
## 7 0e951c… block-v1… 0 0 0.960 0.778 1  
## 8 0fefde… block-v1… 0 0 0.782 0.556 1  
## 9 1403b3… block-v1… 0 0 0.915 0.556 0  
## 10 14809c… block-v1… 353794. 0 0.891 0.778 0  
## # … with 363 more rows, and 9 more variables: num\_events <dbl>,  
## # num\_video\_interactions <dbl>, num\_links\_clicked <dbl>,  
## # num\_events\_past\_week <dbl>, num\_events\_past\_three\_day <dbl>,  
## # num\_events\_past\_day <dbl>, unsubmitted <dbl>, deadline <dbl>, weight <dbl>

train = all\_features %>% filter(!module\_id %in% c(last\_four,"block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@bc80acbecf2d43f7b1d24704ff03fdf3"))  
 print(train)

## # A tibble: 752 x 16  
## # Groups: hash\_id [95]  
## hash\_id module\_id diff miss\_ddl avg\_grade avg\_perf\_score not\_submitted\_b…  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 046b9f… block-v1… 3.67e5 0 1 1 0  
## 2 051123… block-v1… 3.61e5 0 1 1 0  
## 3 06b011… block-v1… 3.59e5 0 1 1 1  
## 4 08258a… block-v1… 3.58e5 0 1 1 0  
## 5 0ca998… block-v1… 3.63e5 0 1 1 0  
## 6 0df790… block-v1… 0. 0 0 0 1  
## 7 0e951c… block-v1… 2.41e5 0 1 1 0  
## 8 0fefde… block-v1… 0. 0 0 0 1  
## 9 1403b3… block-v1… 3.25e5 0 1 1 0  
## 10 14809c… block-v1… 3.68e5 0 1 1 0  
## # … with 742 more rows, and 9 more variables: num\_events <dbl>,  
## # num\_video\_interactions <dbl>, num\_links\_clicked <dbl>,  
## # num\_events\_past\_week <dbl>, num\_events\_past\_three\_day <dbl>,  
## # num\_events\_past\_day <dbl>, unsubmitted <dbl>, deadline <dbl>, weight <dbl>

summary(all\_features)

## hash\_id module\_id diff miss\_ddl   
## Length:1219 Length:1219 Min. :-858902 Min. :0.0000   
## Class :character Class :character 1st Qu.: 0 1st Qu.:0.0000   
## Mode :character Mode :character Median : 235871 Median :0.0000   
## Mean : 209399 Mean :0.3314   
## 3rd Qu.: 352805 3rd Qu.:0.0000   
## Max. : 502947 Max. :7.0000   
## avg\_grade avg\_perf\_score not\_submitted\_by\_ddl num\_events   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. : 0.000   
## 1st Qu.:0.9144 1st Qu.:0.7143 1st Qu.:0.0000 1st Qu.: 8.234   
## Median :0.9796 Median :0.8571 Median :0.0000 Median : 9.236   
## Mean :0.8523 Mean :0.7777 Mean :0.3363 Mean : 8.851   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.: 9.870   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :11.185   
## num\_video\_interactions num\_links\_clicked num\_events\_past\_week  
## Min. : 0.00 Min. : 0.00 Min. : 0.0   
## 1st Qu.: 32.00 1st Qu.: 23.00 1st Qu.: 101.0   
## Median : 58.00 Median : 44.00 Median : 177.0   
## Mean : 87.44 Mean : 50.06 Mean : 188.2   
## 3rd Qu.:110.50 3rd Qu.: 69.00 3rd Qu.: 251.0   
## Max. :771.00 Max. :183.00 Max. :1026.0   
## num\_events\_past\_three\_day num\_events\_past\_day unsubmitted   
## Min. : 0.0 Min. : 0.00 Min. :0.00000   
## 1st Qu.: 32.5 1st Qu.: 0.00 1st Qu.:0.00000   
## Median : 80.0 Median : 15.00 Median :0.00000   
## Mean :100.6 Mean : 35.57 Mean :0.05168   
## 3rd Qu.:150.0 3rd Qu.: 48.00 3rd Qu.:0.00000   
## Max. :763.0 Max. :698.00 Max. :1.00000   
## deadline weight   
## Min. :1.599e+09 Min. :0.2531   
## 1st Qu.:1.600e+09 1st Qu.:0.3352   
## Median :1.601e+09 Median :0.4360   
## Mean :1.601e+09 Mean :0.5149   
## 3rd Qu.:1.602e+09 3rd Qu.:0.6164   
## Max. :1.602e+09 Max. :1.0000

cor(all\_features[! (names(all\_features) %in% c("hash\_id","module\_id","deadline"))])[,"unsubmitted"]

## diff miss\_ddl avg\_grade   
## -0.18346631 0.03308926 -0.14086105   
## avg\_perf\_score not\_submitted\_by\_ddl num\_events   
## -0.14933670 -0.16619158 -0.11003210   
## num\_video\_interactions num\_links\_clicked num\_events\_past\_week   
## -0.02873552 -0.05731649 -0.10713750   
## num\_events\_past\_three\_day num\_events\_past\_day unsubmitted   
## -0.07804353 -0.04961481 1.00000000   
## weight   
## 0.01401373

###############################################  
###############################################

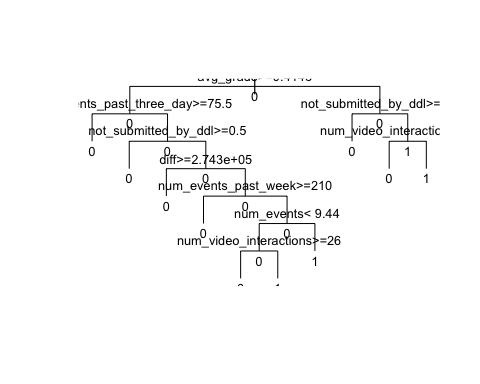
# Step 5: Train your models

*Question 5:* Train a prediction model and iterate on it. You should try out different algorithms that you have learned so far. You can go back and check your features and refine them to get better performance. To check how well you are doing, you should focus on your training data and compute the F1 score: F1 = 2/[(1/recall)+(1/precision)]. Report your F1 score on the training data below (don’t forget this!).

###############################################   
####### BEGIN INPUT: Train and report #########  
###############################################  
  
# Fit model to training data  
# add code here  
### logistic regression  
m\_logreg = glm(as.factor(train$unsubmitted) ~ . - hash\_id - module\_id - deadline - weight, data = train, family = "binomial")  
# the output are the coefficients:  
m\_logreg

##   
## Call: glm(formula = as.factor(train$unsubmitted) ~ . - hash\_id - module\_id -   
## deadline - weight, family = "binomial", data = train)  
##   
## Coefficients:  
## (Intercept) diff   
## 1.007e+00 -5.423e-06   
## miss\_ddl avg\_grade   
## 2.615e-01 -1.225e+00   
## avg\_perf\_score not\_submitted\_by\_ddl   
## -1.456e-01 -1.982e+01   
## num\_events num\_video\_interactions   
## 4.336e-02 4.619e-03   
## num\_links\_clicked num\_events\_past\_week   
## 4.689e-03 -9.409e-03   
## num\_events\_past\_three\_day num\_events\_past\_day   
## -1.692e-03 -9.592e-03   
##   
## Degrees of Freedom: 751 Total (i.e. Null); 740 Residual  
## Null Deviance: 289.1   
## Residual Deviance: 180.2 AIC: 204.2

### knn  
library(class)  
#predictors = c("avg\_perf\_score","diff","num\_events")  
#m\_knn = knn(train = train[,predictors], test = train[,predictors], cl = train$unsubmitted, k = 3)  
m\_knn = knn(train[,3:(ncol(train)-3)], train[,3:(ncol(train)-3)], train$unsubmitted, k = 2)  
# the output are the predictions:  
#m\_knn  
  
### classification tree  
library(rpart)  
# = rpart(unsubmitted ~ avg\_grade + more\_than\_1\_possible\_point + num\_events + num\_video\_interactions + num\_events\_past\_week + num\_events\_past\_three\_day + num\_events\_past\_day, data = train, method = "class")  
m\_class\_tree = rpart(unsubmitted ~ diff+ miss\_ddl + avg\_grade + avg\_perf\_score +num\_events+num\_video\_interactions+ not\_submitted\_by\_ddl+ num\_links\_clicked + num\_events\_past\_week+ num\_events\_past\_three\_day + num\_events\_past\_day , data = train, method = "class")  
#m\_class\_tree = rpart(unsubmitted ~ diff + miss\_ddl + avg\_grade + num\_events + num\_events\_past\_week, data = train, method = "class")  
#m\_class\_tree = rpart(unsubmitted ~ avg\_perf\_score + diff + num\_events, data = train, method = "class")  
#m\_class\_tree = rpart(unsubmitted ~ . - hash\_id - module\_id - deadline - weight, data = train, method = "class")  
# the output are the decision trees  
#m\_class\_tree  
  
# you can even plot it!  
plot(m\_class\_tree, uniform = T)  
text(m\_class\_tree, use.n = F, all = TRUE, cex = .8)



# prune the trees to avoid overfitting by limiting tree complexity  
#cp\_class\_tree = m\_class\_tree$cptable[which.min(m\_class\_tree$cptable[,"xerror"]),"CP"]  
#m\_class\_tree\_pruned = prune(m\_class\_tree, cp = cp\_class\_tree)  
  
  
### naive bayes  
  
library(e1071)  
#m\_nb = naiveBayes(unsubmitted ~ diff + avg\_perf\_score + num\_events, data = train)  
m\_nb = naiveBayes(as.factor(unsubmitted) ~ . - hash\_id - module\_id - deadline - weight, data = train, method = "class")  
# the output are a-prior and conditional probabilities  
#m\_nb  
  
# Make predictions on the test dataset  
# logreg: this returns the probability of dropout, so you can set Prob > 0.5 to mean Dropout  
p\_logreg = predict(m\_logreg, newdata = train, type = "response") > 0.5  
#p\_logreg  
#  
# knn: this already has the prediction  
p\_knn = m\_knn  
# class tree  
p\_class\_tree = predict(m\_class\_tree, newdata = train, type = "class")  
# naive bayes  
p\_nb = predict(m\_nb, newdata = train, type = "class")  
  
cm\_logreg = table(true = train$unsubmitted, predicted = p\_logreg)  
cm\_logreg

## predicted  
## true FALSE TRUE  
## 0 711 5  
## 1 28 8

cm\_knn = table(true = train$unsubmitted, predicted = p\_knn)  
cm\_class\_tree = table(true = train$unsubmitted, predicted = p\_class\_tree)  
cm\_nb = table(true = train$unsubmitted, predicted = p\_nb)  
  
# convenience function for evaluation of confusion matrix  
cm\_eval = function(cm) {  
 list(  
 accur = sum(diag(cm)) / sum(cm),  
 recall = cm[2,2] / sum(cm[2,]),  
 precision = cm[2,2] / sum(cm[,2]),  
 F1 = 2 / (1/(cm[2,2] / sum(cm[2,])) + 1/(cm[2,2] / sum(cm[,2])))  
 )  
}  
  
cm\_eval(cm\_logreg)

## $accur  
## [1] 0.956117  
##   
## $recall  
## [1] 0.2222222  
##   
## $precision  
## [1] 0.6153846  
##   
## $F1  
## [1] 0.3265306

cm\_eval(cm\_knn)

## $accur  
## [1] 0.9601064  
##   
## $recall  
## [1] 0.5833333  
##   
## $precision  
## [1] 0.5833333  
##   
## $F1  
## [1] 0.5833333

cm\_eval(cm\_class\_tree)

## $accur  
## [1] 0.9694149  
##   
## $recall  
## [1] 0.5277778  
##   
## $precision  
## [1] 0.76  
##   
## $F1  
## [1] 0.6229508

cm\_eval(cm\_nb)

## $accur  
## [1] 0.650266  
##   
## $recall  
## [1] 0.9444444  
##   
## $precision  
## [1] 0.1152542  
##   
## $F1  
## [1] 0.2054381

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

#all\_features$weight  
fitControl <- trainControl(method = "none")  
  
#set.seed(825)  
knnFit4 <- train(as.factor(unsubmitted) ~ . , data = train[,!colnames(train) %in% c("hash\_id","module\_id","deadline","weight")],   
 method = "knn",   
 trControl = fitControl,   
 weights = train$weight  
 )  
knnFit4

## k-Nearest Neighbors   
##   
## 752 samples  
## 11 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: None

#predict(gbmFit4, newdata = as.factor(head(test)))  
p\_knn = predict(knnFit4, newdata = train)  
cm\_knn = table(true = train$unsubmitted, predicted = p\_knn)  
cm\_eval(cm\_knn)

## $accur  
## [1] 0.9547872  
##   
## $recall  
## [1] 0.08333333  
##   
## $precision  
## [1] 0.75  
##   
## $F1  
## [1] 0.15

p\_knn = predict(knnFit4, newdata = test)  
cm\_knn = table(true = test$unsubmitted, predicted = p\_knn)  
cm\_eval(cm\_knn)

## $accur  
## [1] 0.9088472  
##   
## $recall  
## [1] 0.0952381  
##   
## $precision  
## [1] 0.1176471  
##   
## $F1  
## [1] 0.1052632

###############################################  
###############################################

# Step 6: Test your model

*Question 6:* Using the model that you arrived at, predict on the held-out test data and report your final F1 score. Typically, you would only do this once at the very end, but for this project it is actually rather hard to do well on the test set, so you can try your model (sparingly to avoid overfitting too much) on the test data to compute the testing F1 score.

###############################################   
####### BEGIN INPUT: Test and report ##########  
###############################################  
  
# Make predictions on the test dataset  
# logreg: this returns the probability of dropout, so you can set Prob > 0.5 to mean Dropout  
#m\_logreg = glm(unsubmitted ~ . - hash\_id - module\_id - deadline, data = train, family = "binomial")  
#p\_logreg = predict(m\_logreg, newdata = test, type = "response") > 0.5  
m\_logreg = glm(unsubmitted ~ . , data = train[,!colnames(train) %in% c("hash\_id","module\_id","deadline","weight")], family = "binomial")  
m\_logreg$xlevels[["module\_id"]] = union(m\_logreg$xlevels[["module\_id"]], levels(test$module\_id))  
p\_logreg = predict(m\_logreg, newdata = test, type = "response") > 0.5  
  
#  
# knn: this already has the prediction  
p\_knn = knn(train[,3:(ncol(train)-3)], test[,3:(ncol(train)-3)], train$unsubmitted, k = 5)  
# class tree  
p\_class\_tree = predict(m\_class\_tree, newdata = test, type = "class")  
# naive bayes  
p\_nb = predict(m\_nb, newdata = test, type = "class")  
# add code here  
  
# here is the confusion matrix for the logreg model:  
cm\_logreg = table(true = test$unsubmitted, predicted = p\_logreg)  
cm\_knn = table(true = test$unsubmitted, predicted = p\_knn)  
cm\_class\_tree = table(true = test$unsubmitted, predicted = p\_class\_tree)  
  
cm\_nb = table(true = test$unsubmitted, predicted = p\_nb)  
  
cm\_logreg

## predicted  
## true FALSE TRUE  
## 0 343 9  
## 1 14 7

cm\_eval(cm\_logreg)

## $accur  
## [1] 0.9383378  
##   
## $recall  
## [1] 0.3333333  
##   
## $precision  
## [1] 0.4375  
##   
## $F1  
## [1] 0.3783784

cm\_knn

## predicted  
## true 0 1  
## 0 337 15  
## 1 19 2

cm\_eval(cm\_knn)

## $accur  
## [1] 0.9088472  
##   
## $recall  
## [1] 0.0952381  
##   
## $precision  
## [1] 0.1176471  
##   
## $F1  
## [1] 0.1052632

cm\_class\_tree

## predicted  
## true 0 1  
## 0 343 9  
## 1 16 5

cm\_eval(cm\_class\_tree)

## $accur  
## [1] 0.9329759  
##   
## $recall  
## [1] 0.2380952  
##   
## $precision  
## [1] 0.3571429  
##   
## $F1  
## [1] 0.2857143

cm\_nb

## predicted  
## true 0 1  
## 0 271 81  
## 1 4 17

cm\_eval(cm\_nb)

## $accur  
## [1] 0.772118  
##   
## $recall  
## [1] 0.8095238  
##   
## $precision  
## [1] 0.1734694  
##   
## $F1  
## [1] 0.2857143

###############################################  
###############################################

# Step 7: Report

*Question 7:* As a team, write a brief report. Imagine your supervisor asked you to investigate the possibility of an early warning system. She would like to know what model to use, what features are important, and most importantly how well it would work. Given what you’ve learned, would you recommend implementing the system? Write your report answering the above questions here:

%######## BEGIN INPUT: Summarize findings ############

In question 2, we created the outcome indicator based on is.na(first\_attempted\_timestamp), that is, if the students actually submitted the homework before the actual deadline or not since the edX system does not accept submissions after the deadline. There are 13 assignments that are used for this analysis. There are two types of assignments: 9 of which are reading comprehension, reading reflection, and video reflection, which all have a possible grade of 1, and 4 of which are the individual and group homework with possible grades varying from 8 to 19 points.

After testing many different factors, we included the ones with the highest correlation values in our model. Despite these features being the most important to our model compared to the many others that we engineered, they still had low correlation values, the highest only about 0.2. We learned that predicting which students will not submit an assignment through this system is difficult because overall, the possible factors have low correlation values, resulting in F1 values below 0.4 for the models. This is true even for the logistic regression model, our best performing model with an F1 score of 0.3783784.

Given our results, we would in fact not recommend implementing the system. Our most important features included the average difference in time between when a student submitted and the calculated deadline, the number of clickstream events they generated, their average grade, how often on average they got a perfect score on an assignment, and whether or not they submitted assignments on time in the past.

One possible reason that our system did not work well might be the NA values in the original datasets and in the features we engineered. The NA values in first\_attempted\_timestamp were used to determine the outcome variable. In some of our features, we calculated the difference between the deadline and first\_attempted\_timestamp, which sometimes resulted in NA values for those features. Some models, such as the Naive Bayes, are able to still make predictions even when missing attributes are present. Thus, the impact of missing data largely relates to the choice of learning algorithm. We attempted to work around this issue by manually changing the null values to 0 where we thought it made sense to do so, but this altered the fit of the models to the data. We also considered replacing the missing values with a “normalized” value (i.e. mean of known values), but this would not have worked for binary features where we were seeing the NA values.

In evaluating the obstacles we faced when creating this model, we feel we had a particularly difficult time relating the data back to the corresponding due dates. Particularly because we had no basis on which events were related to which assignment, we were making these classifications purely based on due date (which was an assumed value) and the timestamp of the event. While these methods were useful, future models may be more successful with more clear data relating to the due date and the relevant events to each assignment.

While creating the models, we made some other observations we believe may be helpful if someone were to look into designing a similar system in the future. When we generated the percentage of unsubmitted assignments for each module we noticed one assignment, the only video reflection assignment, has the highest percentage of being unsubmitted relative to other assignments. Upon examining it more closely, we concluded that this might be because this assignment was due on a Friday followed by the first group project, an unusual time that students might forget. It is possible that our model is less effective because unique circumstances such as this may be influencing why a student doesn’t submit more often, as opposed to other more predictable situations such as a student consistently struggling in the course.

%###############################################

# Submit Project

This is the end of the project. Please **Knit a Word doc report** that shows both the R code and R output and upload it on the EdX platform. EACH TEAM MEMBER NEEDS TO SUBMIT THE REPORT ON EDX TO GET CREDIT.