Group Project: Early Alert with LMS Data

INFO 4100 Learning Analytics

[Caitlin Lau (CJL236), Destiny Malloy (DAM359), Jeremy Clerc (JC2267), Esuvat Bomani (EKB72), Asma Khan (AK2355), Trevor Warner (TW448) ]]

# 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 ######   
###############################################  
#WE need to add more here  
# Exploring Clickstreams  
# add code here  
head(cl)

## hash\_id survey\_id  
## 1 798c07d80b47c627c00fea7273b2a80f 27fa9f9752fc942b5c93987b36c2cb62  
## 2 e7564593ff45d5b6f4c0c5f3e862d918 63d10018951e18e6c0f471730163eed5  
## 3 7b870ab030f6e55920eebfae7cd39386 3dfeb94120f2ee77670d5efcfcd1a530  
## 4 0fefde5cc3b1c65dba071888ce9e522e b31b520886ee96ca3aea2eafb59acb9a  
## 5 <NA> <NA>  
## 6 <NA> <NA>  
## time  
## 1 2020-09-01T22:35:25.507026+00:00  
## 2 2020-09-01T23:16:27.334846+00:00  
## 3 2020-09-01T23:08:48.403250+00:00  
## 4 2020-09-01T23:16:42.966905+00:00  
## 5 2020-09-01T23:23:55.093422+00:00  
## 6 2020-09-01T23:24:00.095250+00:00  
## name  
## 1 edx.course.enrollment.activated  
## 2 /courses/course-v1:Cornellx+INFO4100+Fall2020/course/  
## 3 edx.course.enrollment.activated  
## 4 edx.course.enrollment.activated  
## 5 /courses/course-v1:Cornellx+INFO4100+Fall2020/xblock/block-v1:Cornellx+INFO4100+Fall2020+type@html+block@ab730c96210b4e7c8055853e92e944f7/handler/publish\_completion  
## 6 /courses/course-v1:Cornellx+INFO4100+Fall2020/xblock/block-v1:Cornellx+INFO4100+Fall2020+type@openassessment+block@a204b7a5387446cfb7cdfa55e3608a0c/handler/render\_student\_training  
## event\_type  
## 1 edx.course.enrollment.activated  
## 2 /courses/course-v1:Cornellx+INFO4100+Fall2020/course/  
## 3 edx.course.enrollment.activated  
## 4 edx.course.enrollment.activated  
## 5 /courses/course-v1:Cornellx+INFO4100+Fall2020/xblock/block-v1:Cornellx+INFO4100+Fall2020+type@html+block@ab730c96210b4e7c8055853e92e944f7/handler/publish\_completion  
## 6 /courses/course-v1:Cornellx+INFO4100+Fall2020/xblock/block-v1:Cornellx+INFO4100+Fall2020+type@openassessment+block@a204b7a5387446cfb7cdfa55e3608a0c/handler/render\_student\_training  
## referer  
## 1   
## 2 https://preview.edge.edx.org/dashboard  
## 3   
## 4   
## 5 https://edge.edx.org/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/6502c5233b274e2fa8797269fb79e890/af7025c78a4a46a88efe118e496259f9/?activate\_block\_id=block-v1%3ACornellx%2BINFO4100%2BFall2020%2Btype%40sequential%2Bblock%40af7025c78a4a46a88efe118e496259f9  
## 6 https://edge.edx.org/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/6502c5233b274e2fa8797269fb79e890/af7025c78a4a46a88efe118e496259f9/?activate\_block\_id=block-v1%3ACornellx%2BINFO4100%2BFall2020%2Btype%40sequential%2Bblock%40af7025c78a4a46a88efe118e496259f9  
## page event\_source event  
## 1 <NA> server 463675, course-v1:Cornellx+INFO4100+Fall2020, audit  
## 2 <NA> server {"POST": {}, "GET": {}}  
## 3 <NA> server 463681, course-v1:Cornellx+INFO4100+Fall2020, audit  
## 4 <NA> server 463682, course-v1:Cornellx+INFO4100+Fall2020, audit  
## 5 <NA> server {"POST": {"{\\"completion\\":1}": [""]}, "GET": {}}  
## 6 <NA> server {"POST": {}, "GET": {}}  
## timestamp  
## 1 1598999726  
## 2 1599002187  
## 3 1599001728  
## 4 1599002203  
## 5 1599002635  
## 6 1599002640

# print("distinct hash id")  
n\_distinct(cl$hash\_id)

## [1] 96

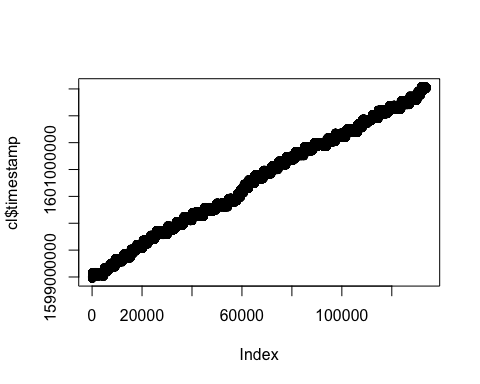
# n\_distinct(cl$survey\_id)  
# n\_distinct(cl$time)  
# print("distinct name")  
# n\_distinct(cl$name)  
 summary(cl$timestamp)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.599e+09 1.600e+09 1.601e+09 1.601e+09 1.602e+09 1.603e+09

table(cl$event\_source)

##   
## browser server   
## 44772 89049

#boxplot(cl$timestamp ~ cl$module\_id)  
 plot(cl$timestamp)



# Exploring Assessment grades  
# add code here  
head(a)

## hash\_id  
## 1 4f766411b43e4664c4308d5fb3dd50d8  
## 2 4f766411b43e4664c4308d5fb3dd50d8  
## 3 4f766411b43e4664c4308d5fb3dd50d8  
## 4 4f766411b43e4664c4308d5fb3dd50d8  
## 5 4f766411b43e4664c4308d5fb3dd50d8  
## 6 4f766411b43e4664c4308d5fb3dd50d8  
## usage\_key  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@a9e732d36e2847f2995147ec3c902962  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@556b2a247fe1409bbd91458085949051  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@6802dda692df48999551d42b4f8a338e  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@a36f35c0e0d84ce2a795d228f3d50b61  
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@0de90422009d4d92a2de261e22f30e9f  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@9c13103d27894d52b90c77082a714d04  
## earned\_graded possible\_graded first\_attempted created  
## 1 0 0 NULL 2020-10-03 19:10:26.939392  
## 2 0 0 NULL 2020-10-03 19:10:26.954574  
## 3 0 1 NULL 2020-10-03 19:10:26.988185  
## 4 0 8 NULL 2020-10-03 19:10:27.004273  
## 5 0 0 NULL 2020-10-10 15:21:17.119873  
## 6 0 0 NULL 2020-09-17 03:25:53.301765  
## modified created\_timestamp modified\_timestamp  
## 1 2020-10-10 16:53:12.315667 1601752227 1602348792  
## 2 2020-10-10 16:53:12.326251 1601752227 1602348792  
## 3 2020-10-10 16:53:12.352428 1601752227 1602348792  
## 4 2020-10-10 16:53:12.364465 1601752227 1602348792  
## 5 2020-10-10 16:53:12.376822 1602343277 1602348792  
## 6 2020-10-10 16:53:12.181680 1600313153 1602348792  
## first\_attempted\_timestamp  
## 1 NA  
## 2 NA  
## 3 NA  
## 4 NA  
## 5 NA  
## 6 NA

# print("distinct hash\_id")  
n\_distinct(a$hash\_id)

## [1] 95

# print("distinct created timestamp")  
n\_distinct(a$created\_timestamp)

## [1] 3135

summary(a)

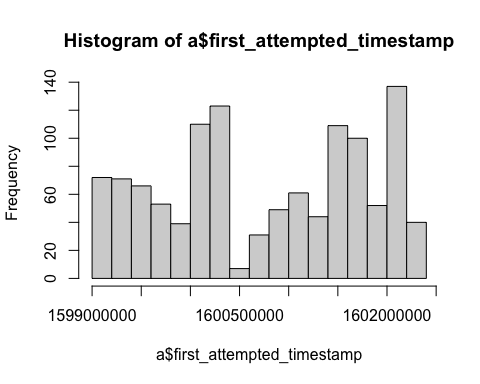
## hash\_id usage\_key earned\_graded possible\_graded   
## Length:3135 Length:3135 Min. : 0.000 Min. : 0.000   
## Class :character Class :character 1st Qu.: 0.000 1st Qu.: 0.000   
## Mode :character Mode :character Median : 0.000 Median : 0.500   
## Mean : 1.368 Mean : 2.136   
## 3rd Qu.: 1.000 3rd Qu.: 1.000   
## Max. :19.000 Max. :19.000   
##   
## first\_attempted created modified created\_timestamp   
## Length:3135 Length:3135 Length:3135 Min. :1.599e+09   
## Class :character Class :character Class :character 1st Qu.:1.600e+09   
## Mode :character Mode :character Mode :character Median :1.601e+09   
## Mean :1.601e+09   
## 3rd Qu.:1.602e+09   
## Max. :1.602e+09   
##   
## modified\_timestamp first\_attempted\_timestamp  
## Min. :1.602e+09 Min. :1.599e+09   
## 1st Qu.:1.602e+09 1st Qu.:1.600e+09   
## Median :1.602e+09 Median :1.601e+09   
## Mean :1.602e+09 Mean :1.601e+09   
## 3rd Qu.:1.602e+09 3rd Qu.:1.602e+09   
## Max. :1.602e+09 Max. :1.602e+09   
## NA's :1971

table(a$earned\_graded)

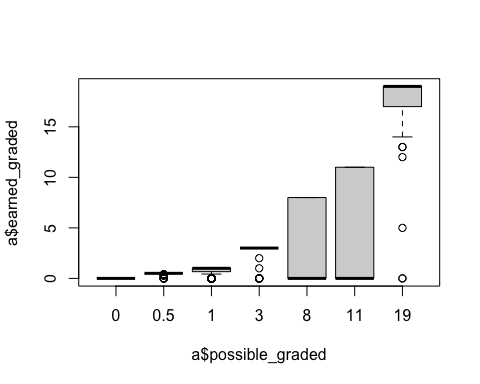
##   
## 0 0.25 0.375 0.428571428571 0.5   
## 1971 1 12 1 77   
## 0.571428571429 0.666666666667 0.714285714286 0.75 0.833333333333   
## 1 17 9 1 42   
## 0.857142857143 1 2 3 5   
## 29 616 1 87 4   
## 6 7 8 11 12   
## 3 25 58 88 1   
## 13 14 15 16 17   
## 2 2 5 7 9   
## 18 19   
## 17 49

hist(a$first\_attempted\_timestamp)

## Warning in breaks[-1L] + breaks[-nB]: NAs produced by integer overflow



boxplot(a$earned\_graded~ a$possible\_graded)



# Exploring Module States  
# add code here  
head(m)

## hash\_id module\_type grade created  
## 1 a985c4dca8d15841c664d9886cd4ab7d sequential NULL 2020-09-30 03:27:44  
## 2 a985c4dca8d15841c664d9886cd4ab7d problem 5 2020-09-30 03:27:44  
## 3 6845955f12d207d926c34a67931ca8f0 sequential NULL 2020-09-30 12:10:04  
## 4 6845955f12d207d926c34a67931ca8f0 problem 6 2020-09-30 12:10:04  
## 5 87d62aa8e9634fb91f7c49c2b53e3529 video NULL 2020-10-01 03:37:22  
## 6 e433299a8a6496f2b680ceb9b8369f73 sequential NULL 2020-10-01 01:50:56  
## modified max\_grade  
## 1 2020-09-30 03:27:44 NULL  
## 2 2020-09-30 03:37:36 6  
## 3 2020-09-30 12:10:04 NULL  
## 4 2020-09-30 13:36:08 6  
## 5 2020-10-01 03:47:53 NULL  
## 6 2020-10-01 01:50:56 NULL  
## module\_id  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@82057a6174ff4557821474a735fe8ba2  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@problem+block@ce70fd29003341acbd813ec54ef8f425  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@82057a6174ff4557821474a735fe8ba2  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@problem+block@ce70fd29003341acbd813ec54ef8f425  
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@video+block@a50ba7f5804d43fea4b073cb7ec127e7  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@82057a6174ff4557821474a735fe8ba2  
## created\_timestamp modified\_timestamp  
## 1 1601436464 1601436464  
## 2 1601436464 1601437056  
## 3 1601467804 1601467804  
## 4 1601467804 1601472968  
## 5 1601523442 1601524073  
## 6 1601517056 1601517056

summary(m)

## hash\_id module\_type grade created   
## Length:5232 Length:5232 Length:5232 Length:5232   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## modified max\_grade module\_id created\_timestamp   
## Length:5232 Length:5232 Length:5232 Min. :1.599e+09   
## Class :character Class :character Class :character 1st Qu.:1.600e+09   
## Mode :character Mode :character Mode :character Median :1.600e+09   
## Mean :1.601e+09   
## 3rd Qu.:1.601e+09   
## Max. :1.602e+09   
## modified\_timestamp   
## Min. :1.599e+09   
## 1st Qu.:1.600e+09   
## Median :1.601e+09   
## Mean :1.601e+09   
## 3rd Qu.:1.602e+09   
## Max. :1.602e+09

table(m$module\_type)

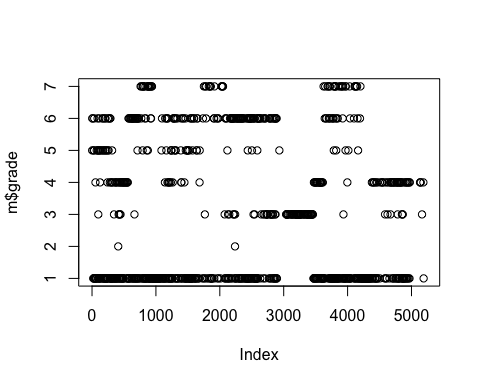
##   
## chapter course openassessment problem sequential   
## 613 97 361 981 2642   
## video   
## 538

n\_distinct(m$module\_type)

## [1] 6

plot(m$grade)

## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs introduced by coercion



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

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.

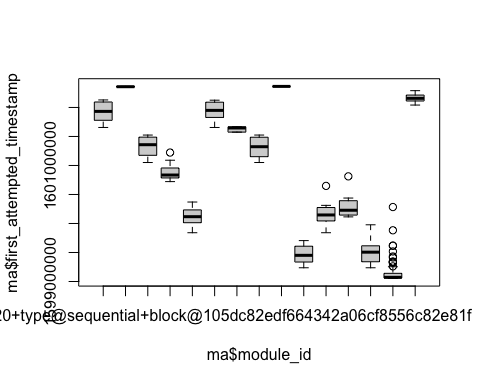
ma = m %>% left\_join(  
 a %>% select(hash\_id:possible\_graded, first\_attempted\_timestamp),   
 by = c("hash\_id"="hash\_id", "module\_id"="usage\_key")  
)  
head(ma)

## hash\_id module\_type grade created  
## 1 a985c4dca8d15841c664d9886cd4ab7d sequential NULL 2020-09-30 03:27:44  
## 2 a985c4dca8d15841c664d9886cd4ab7d problem 5 2020-09-30 03:27:44  
## 3 6845955f12d207d926c34a67931ca8f0 sequential NULL 2020-09-30 12:10:04  
## 4 6845955f12d207d926c34a67931ca8f0 problem 6 2020-09-30 12:10:04  
## 5 87d62aa8e9634fb91f7c49c2b53e3529 video NULL 2020-10-01 03:37:22  
## 6 e433299a8a6496f2b680ceb9b8369f73 sequential NULL 2020-10-01 01:50:56  
## modified max\_grade  
## 1 2020-09-30 03:27:44 NULL  
## 2 2020-09-30 03:37:36 6  
## 3 2020-09-30 12:10:04 NULL  
## 4 2020-09-30 13:36:08 6  
## 5 2020-10-01 03:47:53 NULL  
## 6 2020-10-01 01:50:56 NULL  
## module\_id  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@82057a6174ff4557821474a735fe8ba2  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@problem+block@ce70fd29003341acbd813ec54ef8f425  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@82057a6174ff4557821474a735fe8ba2  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@problem+block@ce70fd29003341acbd813ec54ef8f425  
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@video+block@a50ba7f5804d43fea4b073cb7ec127e7  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@82057a6174ff4557821474a735fe8ba2  
## created\_timestamp modified\_timestamp earned\_graded possible\_graded  
## 1 1601436464 1601436464 0.8333333 1  
## 2 1601436464 1601437056 NA NA  
## 3 1601467804 1601467804 1.0000000 1  
## 4 1601467804 1601472968 NA NA  
## 5 1601523442 1601524073 NA NA  
## 6 1601517056 1601517056 1.0000000 1  
## first\_attempted\_timestamp  
## 1 1601436464  
## 2 NA  
## 3 1601467804  
## 4 NA  
## 5 NA  
## 6 1601517056

# 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).**

#We assigned table to something  
unsubmit\_tbl <- 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)

#head(unsubmit\_tbl)  
  
#the assessment table has a lot of empty records in there even if no questions associated   
#p unsubmit looks up if most people have not submitted anything for it it is not a real assignment. And why it needs to be kicked out  
#the ma table is missing the deadlines initially.   
#the ma table tries to figure out the deadline when ppl submit

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 ###########  
###############################################  
#should this be > 0.5  
unsubmit\_tbl = subset(unsubmit\_tbl, p\_unsubmitted <= 0.5)  
unsubmit\_tbl$deadline\_minus\_24 = (unsubmit\_tbl$deadline-(60 \* 60 \* 24))  
#head(unsubmit\_tbl)  
  
new\_unsubmit\_tbl=merge(x = unsubmit\_tbl, y = ma, by = "module\_id", all.x = TRUE)  
  
#new column: if earned\_grade=0 and first attempted timestamp in ma is NA then it is unsubmitted.  
new\_unsubmit\_tbl$is\_submitted=(ifelse(is.na(new\_unsubmit\_tbl$first\_attempted\_timestamp), 1, 0 ))  
head(new\_unsubmit\_tbl)

## module\_id  
## 1 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@105dc82edf664342a06cf8556c82e81f  
## 2 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@105dc82edf664342a06cf8556c82e81f  
## 3 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@105dc82edf664342a06cf8556c82e81f  
## 4 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@105dc82edf664342a06cf8556c82e81f  
## 5 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@105dc82edf664342a06cf8556c82e81f  
## 6 block-v1:Cornellx+INFO4100+Fall2020+type@sequential+block@105dc82edf664342a06cf8556c82e81f  
## deadline p\_unsubmitted deadline\_minus\_24 hash\_id  
## 1 1602114378 0.0326087 1602027978 798c07d80b47c627c00fea7273b2a80f  
## 2 1602114378 0.0326087 1602027978 4f766411b43e4664c4308d5fb3dd50d8  
## 3 1602114378 0.0326087 1602027978 61ad52da61fce560fb9e5890b4cb73f1  
## 4 1602114378 0.0326087 1602027978 7977646ac56ba5d46566bc39a1c5afa1  
## 5 1602114378 0.0326087 1602027978 1cf5f50fe94bc1143c0b3e1cd0f48ca0  
## 6 1602114378 0.0326087 1602027978 bf41a298801ead675bf27cc7ed969c6e  
## module\_type grade created modified max\_grade  
## 1 sequential NULL 2020-10-08 02:41:35 2020-10-08 02:41:35 NULL  
## 2 sequential NULL 2020-10-03 00:38:39 2020-10-03 00:38:39 NULL  
## 3 sequential NULL 2020-10-07 23:36:08 2020-10-07 23:36:08 NULL  
## 4 sequential NULL 2020-10-07 00:34:45 2020-10-07 00:34:45 NULL  
## 5 sequential NULL 2020-10-04 23:17:37 2020-10-04 23:17:37 NULL  
## 6 sequential NULL 2020-10-04 02:34:40 2020-10-04 02:34:40 NULL  
## created\_timestamp modified\_timestamp earned\_graded possible\_graded  
## 1 1602124895 1602124895 0.500 0.5  
## 2 1601685519 1601685519 0.375 0.5  
## 3 1602113768 1602113768 0.500 0.5  
## 4 1602030885 1602030885 0.500 0.5  
## 5 1601853457 1601853457 0.500 0.5  
## 6 1601778880 1601778880 0.500 0.5  
## first\_attempted\_timestamp is\_submitted  
## 1 1602124895 0  
## 2 1601685519 0  
## 3 1602113768 0  
## 4 1602030885 0  
## 5 1601853457 0  
## 6 1601778880 0

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

## 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).

#just alwat remember to minus sec\_day for example\_deadline  
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)

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

cl = cl[!is.na(cl$hash\_id), ]  
  
mod\_and\_deadline = unsubmit\_tbl[,c("module\_id", "deadline\_minus\_24")]   
mod\_and\_deadline = mod\_and\_deadline[order(mod\_and\_deadline$deadline\_minus\_24, decreasing = FALSE),]  
#head(mod\_and\_deadline)  
#FIX make it so that it is not hard coded  
#Add minimum module that clickstream information can be used in  
cl$module\_id <- ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[1], mod\_and\_deadline$module\_id[1],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[2], mod\_and\_deadline$module\_id[2],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[3], mod\_and\_deadline$module\_id[3],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[4], mod\_and\_deadline$module\_id[4],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[5], mod\_and\_deadline$module\_id[5],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[6], mod\_and\_deadline$module\_id[6],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[7], mod\_and\_deadline$module\_id[7],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[8], mod\_and\_deadline$module\_id[8],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[9], mod\_and\_deadline$module\_id[9],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[10], mod\_and\_deadline$module\_id[10],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[11], mod\_and\_deadline$module\_id[11],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[12], mod\_and\_deadline$module\_id[12],  
 ifelse(cl$timestamp <=mod\_and\_deadline$deadline\_minus\_24[13], mod\_and\_deadline$module\_id[13],  
 NA )))))))))))))  
#cl\_mod\_combined = left\_join(cl, unsubmit\_tbl, by = "module\_id")  
cl\_mod\_combined = merge(x = cl, y = unsubmit\_tbl, by = "module\_id", all.x = TRUE)  
cl\_mod\_combined = cl\_mod\_combined[!is.na(cl\_mod\_combined$module\_id), ]  
#cl  
#head(cl\_mod\_combined)  
#tail(cl)  
#unsubmit\_tbl  
#tail(cl\_mod\_combined)

###############################################   
###### BEGIN INPUT: Engineer features #########  
###############################################  
  
  
#get more features and then reduce them on performance   
# feature 1: num load\_video with a hash\_ID   
# feature 2: num pause\_video with a hash\_ID  
# feature 3: speed\_change\_video with hashID  
# feature 4: stop\_video with hashID  
#feature 5: how many times they clicked on sllyabus (/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/2d815b2e787344838a1509c7a5861d2d/6fffcdccb3b84a8cbc79c173cbbe20e8/)  
# feature 6: did they go to progress tab?  
#/courses/course-v1:Cornellx+INFO4100+Fall2020/progress  
#feature 7: seek\_video  
#feature 8: edx.grades.problem.submitted  
#feature 9: edx.ui.lms.link\_clicked  
#feature 10: Did they click the slack tab  
 #/courses/course-v1:Cornellx+INFO4100+Fall2020/f7c1480e105a4999839847b62fd7c19e/  
#feature 11: num clicked on the week 2 reading  
 #/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/6502c5233b274e2fa8797269fb79e890/8f630d1feb054691ac467f5c8c4bf68a/  
#feature 12: num clicked on the week 5 reading  
#feature 12: num clicked on the week 7 reading  
  
cl\_mod\_combined\_features = cl\_mod\_combined %>%   
 group\_by(hash\_id, module\_id,deadline\_minus\_24) %>%  
 summarise(  
 num\_load\_video = sum(name=="load\_video"),  
 num\_pause\_video = sum(name=="pause\_video"),  
 num\_speed\_change\_video = sum(name=="speed\_change\_video"),  
 num\_stop\_video = sum(name=="stop\_video"),  
 num\_click\_syllabus = sum(name=="/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/2d815b2e787344838a1509c7a5861d2d/6fffcdccb3b84a8cbc79c173cbbe20e8/"),  
 num\_click\_progress = sum(name=="/courses/course-v1:Cornellx+INFO4100+Fall2020/progress"),  
 num\_seek\_video = sum(name=="seek\_video"),  
 num\_name\_prob\_submit = sum(name=="edx.grades.problem.submitted"),  
 num\_slack\_tab= sum(name=="/courses/course-v1:Cornellx+INFO4100+Fall2020/f7c1480e105a4999839847b62fd7c19e/"),  
 num\_week\_two\_reading= sum( name=="/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/6502c5233b274e2fa8797269fb79e890/8f630d1feb054691ac467f5c8c4bf68a/"),  
 num\_week\_five\_reading=sum(name=="/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/9c0661a25c0c4649a4971fb1a7d94c20/47161f5dde62497d9dd8f5b532de69d2/"),  
 num\_week\_seven\_reading= sum(name=="/courses/course-v1:Cornellx+INFO4100+Fall2020/courseware/7692a90992fe43acbe8caea218a2a595/bb26fa8b1f7648768c9ba9076f54e786/")  
 )

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

#cl\_mod\_combined\_features  
cl\_mod\_combined\_features = cl\_mod\_combined\_features[order(cl\_mod\_combined\_features$deadline\_minus\_24, decreasing = FALSE),]  
cl\_mod\_combined\_features = cl\_mod\_combined\_features[order(cl\_mod\_combined\_features$hash\_id, decreasing = FALSE),]  
#head(cl\_mod\_combined\_features)  
  
#Aggregate data so that you have only use ceratain data for certain modules (time)  
for(i in 2:nrow(cl\_mod\_combined\_features)) {  
 if(cl\_mod\_combined\_features$hash\_id[i-1] == cl\_mod\_combined\_features$hash\_id[i] && cl\_mod\_combined\_features$deadline\_minus\_24[i-1] < cl\_mod\_combined\_features$deadline\_minus\_24[i]){  
 cl\_mod\_combined\_features$num\_load\_video[i] = cl\_mod\_combined\_features$num\_load\_video[i] + cl\_mod\_combined\_features$num\_load\_video[i-1]  
 cl\_mod\_combined\_features$num\_pause\_video[i] = cl\_mod\_combined\_features$num\_pause\_video[i] + cl\_mod\_combined\_features$num\_pause\_video[i-1]  
 cl\_mod\_combined\_features$num\_speed\_change\_video[i] = cl\_mod\_combined\_features$num\_speed\_change\_video[i] + cl\_mod\_combined\_features$num\_speed\_change\_video[i-1]  
 cl\_mod\_combined\_features$num\_stop\_video[i] = cl\_mod\_combined\_features$num\_stop\_video[i] + cl\_mod\_combined\_features$num\_stop\_video[i-1]  
 cl\_mod\_combined\_features$num\_click\_syllabus[i] = cl\_mod\_combined\_features$num\_click\_syllabus[i] + cl\_mod\_combined\_features$num\_click\_syllabus[i-1]  
 cl\_mod\_combined\_features$num\_click\_progress[i] = cl\_mod\_combined\_features$num\_click\_progress[i] + cl\_mod\_combined\_features$num\_click\_progress[i-1]  
 cl\_mod\_combined\_features$num\_seek\_video[i] = cl\_mod\_combined\_features$num\_seek\_video[i] + cl\_mod\_combined\_features$num\_seek\_video[i-1]  
 cl\_mod\_combined\_features$num\_name\_prob\_submit[i] = cl\_mod\_combined\_features$num\_name\_prob\_submit[i] + cl\_mod\_combined\_features$num\_name\_prob\_submit[i-1]  
 cl\_mod\_combined\_features$num\_slack\_tab[i] = cl\_mod\_combined\_features$num\_slack\_tab[i] + cl\_mod\_combined\_features$num\_slack\_tab[i-1]  
 }  
}  
cl\_features = cl\_mod\_combined\_features  
  
  
# adding MA Feature #####  
  
new\_unsubmit\_tbl = new\_unsubmit\_tbl[!is.na(new\_unsubmit\_tbl$hash\_id), ]  
new\_unsubmit\_tbl = new\_unsubmit\_tbl[order(new\_unsubmit\_tbl$deadline\_minus\_24, decreasing = FALSE),]  
new\_unsubmit\_tbl = new\_unsubmit\_tbl[order(new\_unsubmit\_tbl$hash\_id, decreasing = FALSE),]  
  
  
for(i in 2:nrow(new\_unsubmit\_tbl)) {  
 if (new\_unsubmit\_tbl$hash\_id[i] == new\_unsubmit\_tbl$hash\_id[i-1] && new\_unsubmit\_tbl$deadline\_minus\_24[i-1] < new\_unsubmit\_tbl$deadline\_minus\_24[i] ){  
 new\_unsubmit\_tbl$earned\_graded[i] = new\_unsubmit\_tbl$earned\_graded[i] + new\_unsubmit\_tbl$earned\_graded[i-1]  
 }  
}  
  
new\_unsubmit\_tbl\_t = new\_unsubmit\_tbl  
  
new\_unsubmit\_tbl\_t$first\_attempted\_timestamp[is.na(new\_unsubmit\_tbl$first\_attempted\_timestamp)] <- 0  
new\_unsubmit\_tbl\_t$time\_till\_deadline = new\_unsubmit\_tbl\_t$deadline - new\_unsubmit\_tbl\_t$first\_attempted\_timestamp  
temp\_df = new\_unsubmit\_tbl\_t %>%  
 group\_by(hash\_id) %>%  
 summarise(  
 maximum\_grade = max(earned\_graded),  
 points\_off = sum(possible\_graded) - max(earned\_graded),  
 avg\_grade = max(earned\_graded) / n(),  
 avg\_grade\_sq = (max(earned\_graded) / n())^2,  
 earliest\_time\_till\_deadline= max(time\_till\_deadline),  
 latest\_time\_till\_deadline = min(time\_till\_deadline),  
 mean\_time\_till\_deadline = mean(time\_till\_deadline),  
 mean\_time\_sq = mean(time\_till\_deadline)^2,  
 var\_time\_till\_deadline = var(time\_till\_deadline),  
 std\_time\_till\_deadline = sd(time\_till\_deadline),  
 num\_past\_time\_till\_deadline = sum(time\_till\_deadline<0),  
 modified\_minus\_created = modified\_timestamp -created\_timestamp  
 )

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

new\_unsubmit\_tbl\_t = left\_join(new\_unsubmit\_tbl\_t, temp\_df,  
 by ="hash\_id")  
  
  
###############################################  
###############################################

# 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  
  
#list of modules ranked by deadline  
first\_modules = unsubmit\_tbl %>%   
 group\_by(module\_id) %>%  
 summarise(  
 avg\_deadline = mean(deadline\_minus\_24)  
 ) %>%  
 mutate(i = rank(avg\_deadline))

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

#combining features and outcome  
combined = merge(cl\_mod\_combined\_features, new\_unsubmit\_tbl\_t, by.cl\_mod\_combined\_features=c("hash\_id", "module\_id"), by.new\_unsubmit\_tbl=c("hash\_id", "module\_id"))  
  
#combining module\_id rank based on deadline  
combined = left\_join(combined, first\_modules, by = 'module\_id')  
  
colnames(new\_unsubmit\_tbl\_t)

## [1] "module\_id" "deadline"   
## [3] "p\_unsubmitted" "deadline\_minus\_24"   
## [5] "hash\_id" "module\_type"   
## [7] "grade" "created"   
## [9] "modified" "max\_grade"   
## [11] "created\_timestamp" "modified\_timestamp"   
## [13] "earned\_graded" "possible\_graded"   
## [15] "first\_attempted\_timestamp" "is\_submitted"   
## [17] "time\_till\_deadline" "maximum\_grade"   
## [19] "points\_off" "avg\_grade"   
## [21] "avg\_grade\_sq" "earliest\_time\_till\_deadline"  
## [23] "latest\_time\_till\_deadline" "mean\_time\_till\_deadline"   
## [25] "mean\_time\_sq" "var\_time\_till\_deadline"   
## [27] "std\_time\_till\_deadline" "num\_past\_time\_till\_deadline"  
## [29] "modified\_minus\_created"

# Split the dataset into train and test based on the module\_ids or periods  
# test - last 4 modules  
test = combined %>%  
 filter(i >= 10) %>%  
 group\_by(hash\_id) %>%  
 subset(select = c(hash\_id, num\_load\_video, num\_pause\_video, num\_speed\_change\_video, num\_stop\_video,  
 num\_click\_syllabus, num\_click\_progress, num\_seek\_video, num\_name\_prob\_submit,  
 num\_slack\_tab, is\_submitted, earned\_graded, num\_stop\_video, maximum\_grade, avg\_grade, points\_off, earliest\_time\_till\_deadline, latest\_time\_till\_deadline, mean\_time\_till\_deadline, var\_time\_till\_deadline, std\_time\_till\_deadline, num\_past\_time\_till\_deadline, mean\_time\_sq, avg\_grade\_sq))  
  
  
# train - first 9 modules  
train = combined %>%   
 filter(i < 10) %>%  
 subset(select = c(hash\_id, num\_load\_video, num\_pause\_video, num\_speed\_change\_video, num\_stop\_video,  
 num\_click\_syllabus, num\_click\_progress, num\_seek\_video, num\_name\_prob\_submit,  
 num\_slack\_tab, is\_submitted, earned\_graded, num\_stop\_video, maximum\_grade, avg\_grade, points\_off, earliest\_time\_till\_deadline, latest\_time\_till\_deadline, mean\_time\_till\_deadline, var\_time\_till\_deadline, std\_time\_till\_deadline, num\_past\_time\_till\_deadline, mean\_time\_sq, avg\_grade\_sq ))  
  
  
#test the correlation   
correlation\_testing = combined %>%   
 subset(select = c( num\_load\_video, num\_pause\_video, num\_speed\_change\_video, num\_stop\_video,  
 num\_click\_syllabus, num\_click\_progress, num\_seek\_video, num\_name\_prob\_submit,  
 num\_slack\_tab, is\_submitted, earned\_graded, num\_stop\_video, maximum\_grade,avg\_grade, points\_off, earliest\_time\_till\_deadline, latest\_time\_till\_deadline, mean\_time\_till\_deadline, var\_time\_till\_deadline, std\_time\_till\_deadline, num\_past\_time\_till\_deadline, mean\_time\_sq, avg\_grade\_sq ))  
  
  
cor(correlation\_testing)[,"is\_submitted"]

## num\_load\_video num\_pause\_video   
## -0.02916594 0.02049402   
## num\_speed\_change\_video num\_stop\_video   
## -0.05194135 -0.06385537   
## num\_click\_syllabus num\_click\_progress   
## -0.04711751 -0.02584511   
## num\_seek\_video num\_name\_prob\_submit   
## 0.01402450 -0.05146771   
## num\_slack\_tab is\_submitted   
## -0.01397502 1.00000000   
## earned\_graded num\_stop\_video.1   
## -0.05543813 -0.06385537   
## maximum\_grade avg\_grade   
## -0.32097424 -0.31743474   
## points\_off earliest\_time\_till\_deadline   
## 0.32412576 0.27807075   
## latest\_time\_till\_deadline mean\_time\_till\_deadline   
## -0.22249288 0.40673040   
## var\_time\_till\_deadline std\_time\_till\_deadline   
## 0.38294969 0.34367104   
## num\_past\_time\_till\_deadline mean\_time\_sq   
## 0.15378037 0.38526888   
## avg\_grade\_sq   
## -0.30440643

# plot(combined$num\_load\_video, combined$is\_submitted)  
# plot(combined$num\_pause\_video, combined$is\_submitted)  
# plot(combined$num\_speed\_change\_video, combined$is\_submitted)  
# plot(combined$num\_click\_syllabus, combined$is\_submitted)  
# plot(combined$num\_click\_progress, combined$is\_submitted)  
# plot(combined$num\_seek\_video, combined$is\_submitted)  
# plot(combined$num\_name\_prob\_submit, combined$is\_submitted)  
# plot(combined$num\_slack\_tab, combined$is\_submitted)  
###############################################  
###############################################

# 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 #########  
###############################################  
# Logistic Regression  
#m\_logreg = glm(is\_submitted ~ num\_load\_video + num\_pause\_video + num\_speed\_change\_video + num\_stop\_video +  
# num\_click\_syllabus + num\_click\_progress + num\_seek\_video + num\_name\_prob\_submit +  
# num\_slack\_tab + earned\_graded, data = train, family = 'binomial')  
  
#m\_logreg  
  
#p\_logreg = predict(m\_logreg, newdata = test, type = "response") > 0.5  
#cm\_logreg = table(true = test$is\_submitted, predicted = p\_logreg )  
  
#cm\_logreg  
#cm\_eval(cm\_logreg)  
#### overfitting log\_reg  
  
  
library(class)  
m\_logreg = glm(is\_submitted ~ mean\_time\_till\_deadline + maximum\_grade + points\_off + avg\_grade + earliest\_time\_till\_deadline + latest\_time\_till\_deadline + mean\_time\_sq + avg\_grade\_sq, data = train, family = "binomial")  
m\_logreg

##   
## Call: glm(formula = is\_submitted ~ mean\_time\_till\_deadline + maximum\_grade +   
## points\_off + avg\_grade + earliest\_time\_till\_deadline + latest\_time\_till\_deadline +   
## mean\_time\_sq + avg\_grade\_sq, family = "binomial", data = train)  
##   
## Coefficients:  
## (Intercept) mean\_time\_till\_deadline   
## -1.511e+01 1.170e-08   
## maximum\_grade points\_off   
## 7.301e-02 3.427e-02   
## avg\_grade earliest\_time\_till\_deadline   
## 2.028e-01 4.418e-09   
## latest\_time\_till\_deadline mean\_time\_sq   
## -8.178e-07 -6.930e-18   
## avg\_grade\_sq   
## -5.417e-02   
##   
## Degrees of Freedom: 7381 Total (i.e. Null); 7373 Residual  
## Null Deviance: 2271   
## Residual Deviance: 1421 AIC: 1439

# library(rpart)  
# # m\_class\_tree = rpart(is\_submitted ~ maximum\_grade, data = train, method = "class")  
# # m\_class\_tree  
# # m\_reg\_tree = rpart(is\_submitted ~ mean\_time\_till\_deadline, data = train, method = "anova")  
  
m\_knn = knn(train = train[,c( 'mean\_time\_till\_deadline', 'var\_time\_till\_deadline', 'std\_time\_till\_deadline', 'maximum\_grade','avg\_grade', 'points\_off')],  
 test = test[,c( 'mean\_time\_till\_deadline', 'var\_time\_till\_deadline', 'std\_time\_till\_deadline', 'maximum\_grade','avg\_grade', 'points\_off')],  
 cl = train$is\_submitted, k = 2 )  
  
  
# m\_knn = knn(train = train[,c( 'mean\_time\_till\_deadline')],   
# test = test[,c( 'mean\_time\_till\_deadline')],   
# cl = train$is\_submitted, k = 2 )  
####Kth NN  
#m\_knn = knn(train = train[,c('num\_load\_video','num\_pause\_video', 'num\_speed\_change\_video', 'num\_stop\_video',  
# 'num\_click\_syllabus', 'num\_name\_prob\_submit', 'time\_till\_deadline')],   
# test = test[,c('num\_load\_video','num\_pause\_video', 'num\_speed\_change\_video', 'num\_stop\_video',  
# 'num\_click\_syllabus', 'num\_name\_prob\_submit','time\_till\_deadline')], cl = train$is\_submitted, k = #2 )  
  
  
###############################################  
###############################################

# 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  
#p\_logreg = predict(m\_logreg, newdata = test, type = 'response')  
#p\_logreg = ifelse(p\_logreg > .5, 1, 0)  
#p\_logreg  
#cm\_logreg = table(true = test$is\_submitted, predicted = p\_logreg)  
#cm\_logreg  
  
# Compute F1  
cm\_eval = function(cm) {  
 list(  
 # recall = cm[2,2] / sum(cm[2,]),  
 F1 = 2 / (1/(cm[2,2] / sum(cm[2,])) + 1/(cm[2,2] / sum(cm[,2])))  
   
 )  
}  
  
p\_logreg = predict(m\_logreg, newdata= test, type ="response") > 0.5  
#p\_logreg  
cm\_log = table(test$is\_submitted, predicted = p\_logreg)  
print("the logistic regression model F1 score:")

## [1] "the logistic regression model F1 score:"

cm\_eval(cm\_log)

## $F1  
## [1] 0.2722513

# log reg: F1= 0.2722513  
  
# p\_class\_tree = predict(m\_class\_tree, newdata = test, type = "class")  
# cm\_class\_tree = table(true = test$is\_submitted, predicted = p\_class\_tree)  
# print("The class tree model F1: score")  
# cm\_eval(cm\_class\_tree)  
  
p\_knn = m\_knn  
  
cm\_knn = table(true = test$is\_submitted, predicted = p\_knn)  
cm\_knn

## predicted  
## true 0 1  
## 0 3324 25  
## 1 138 14

print("the knn modelF1 score")

## [1] "the knn modelF1 score"

cm\_eval(cm\_knn)

## $F1  
## [1] 0.1465969

#KNN: F1 = 0.2666667  
  
###############################################  
###############################################

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

The model that we used for this early warning system is a logistic regression model. Logistic regression is used to predict binary variable outcome, in this case we designed an early warning system to predict whether a student will submit an assignment 24 hours before the deadline. We used multiple features to predict whether or not a student will or will not submit an assignment. Over the course of this project, our team created 24 features but after filtering them down to see which features have the highest correlation in predicting the submission of an assignment we used the best 8 features with the highest correlation (0.38-0.4 in correlation). Some of these features like clicking on the week 5 reading or stopping the video had very low rates of predicting the submission of an assignment (0.05 corr) and therefore we filtered these out from our logistic model.

Instead, in our logistic model, we used variables that had a moderate to higher accuracy at predicting the submission. These features that we implemented into the system that are important are: the average time left until the deadline when they first submitted, the maximum grade they received in the course, the mean grade squared, how many points they got taken off their assignments, their average grade in the course, mean time they took squared (mean square) , the earliest time they submitted before the deadline, the latest time they submitted before the deadline.

Currently, this early warning system is not predicting the students who will dropout with sufficient precision and recall for its F1 score to be F1>0.7. Initially, our F1 score was only 0.09, but over the course of creating new features with higher correlations, we were able to increase our F1 score significantly to become 0.2722513. The click data in the model did a very poor job at predicting whether or not an assignment was submitted, this lead to an initially low F1 score, however in our final model, we mostly omitted click data in order to improve the recall and precision of our model. Our final model mainly uses features that focus on data gained from the assignments and submission times such as the earliest time they submitted before the deadline, how many points were taken off their assignments, and the mean grade they received in the course. We would not recommend implementing the system currently until we are able to gain more click data that can be utilized to generate more accurate features that we can base our model on since the 13 click features we discarded did a very poor job at predicting submissions of the students. However an important thing to note is that information about assignment submission, time taken, and grades received do a good job at predicting the submissions of the students and should definitely be used in a model used for an early warning system. If we could gather more data to increase the model’s accuracy, we would want to include features representing the amount of student engagement in the slack channels (posting in the office hours chat) and the level of participation in office hours. Our features mainly summarize the results of student’s previous assignments as opposed to measuring how engaged a student has been with their readings, lectures, instructors, and videos. However, we feel it would be interesting to understand more about how a student is interacting with the course materials both on and off screen since the clickstream data gives very minimal insight into a student’s overall attitude towards 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.