Group Project: Early Alert with LMS Data

INFO 4100/5101 Learning Analytics

[[Omar Ahmed(owa2), Chris Chung(cc2299), Luke Cura(lbc83), Selina Lin(sxl7)]]

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 an early warning system for students who miss an elaboration activity submission. I am sharing with you an export of the class's HITA 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.

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.

Grading and Rubric: This group project counts to your final grade as specified in the syllabus. Grading will be done using the following rubrics with 0, 1, or 2 points in each rubric: 0 if your submission didn't do it or got it wrong; 1 for a partially correct answer; and 2 for a correct answer. 1. Understanding the Data: Does the student exhibit an understanding of the dataset? 2. Preparing the Data: Does the student adequately prepare the dataset for analysis (outcome, features, timing consideration)? 3. Splitting the Data: Does the student split the data into a training and test set? 4. Training Prediction Models: Does the student train a model and report the accuracy on the training set? 5. Testing Prediction Models: Does the student test the trained model on the hold-out set and report accuracy? 6. Summarizing Results: Does the student provide a coherent and informative summary about the feasibility and accuracy of the early warning system?

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 two datasets which can be connected using the student_id column (a hashed version of the user email) and in some cases the activity_step_id column (an id to connect conversations to activities):

1. Conversation data (1 row per student per message): this includes all messages sent in the general chat and in the activities, with information about the message time (created_at), message length (length_char), and whether it was sent by the AI vs. student (system: 1=AI, 0=student); conversations that occur within an Activity (reading elaboration or homework help) have an activity_step_id, otherwise this shows an NA value; you can trace what system message were sent in response to a student message using the src_id and reply_to_id columns.

2. Activities data (1 row per activity per student): this includes binary started and completed indicator for all activities and students who at least started them.

You can convert any date-time column `X` into a numeric `timestamp` which may be helpful (but optional): `as.numeric(as.POSIXct(X, tz = "UTC"))`. Just note that the timezone is UTC not EST.

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.

```
###############################################
###### BEGIN INPUT: Explore each dataset ######
###############################################

# Exploring Conversations data
# add code here
head(con)
```

```
## # A tibble: 6 x 8
##   student_id                  conversation_id activity_step_id src_id reply_to_id
##   <chr>                       <chr>           <chr>             <dbl>       <dbl>
## 1 37298e1e-f8e8-4c1a-8aab-d~ 001151fc-1149-~ <NA>             484548          NA
## 2 37298e1e-f8e8-4c1a-8aab-d~ 001151fc-1149-~ <NA>             484549      484548
## 3 66902053-8ab2-4ae5-af06-e~ 00149902-b2be-~ c5334167-f212-4~ 534050          NA
## 4 66902053-8ab2-4ae5-af06-e~ 00149902-b2be-~ c5334167-f212-4~ 534013          NA
## 5 66902053-8ab2-4ae5-af06-e~ 00149902-b2be-~ c5334167-f212-4~ 534076          NA
## 6 66902053-8ab2-4ae5-af06-e~ 00149902-b2be-~ c5334167-f212-4~ 534051      534050
## # i 3 more variables: created_at <dttm>, system <lgl>, length_char <dbl>
```

```
n_distinct(con$student_id)
```

```
## [1] 270
```

```
summary(con)
```

```
##   student_id         conversation_id    activity_step_id       src_id
##  Length:50953       Length:50953       Length:50953       Min.   :474987
##  Class :character   Class :character   Class :character   1st Qu.:507784
##  Mode  :character   Mode  :character   Mode  :character   Median :601952
##                                                           Mean   :576830
##                                                           3rd Qu.:624926
##                                                           Max.   :650041
##
##   reply_to_id       created_at                        system
##  Min.   :474987   Min.   :2025-01-18 21:36:09.99   Mode :logical
##  1st Qu.:507620   1st Qu.:2025-02-02 19:22:53.44   FALSE:24653
##  Median :602260   Median :2025-02-27 20:31:07.49   TRUE :26300
##  Mean   :577164   Mean   :2025-02-20 06:46:08.77
##  3rd Qu.:624936   3rd Qu.:2025-03-03 17:48:18.92
##  Max.   :650040   Max.   :2025-03-09 18:56:41.63
##  NA's   :26307
##   length_char
##  Min.   :   0.0
##  1st Qu.:  62.0
##  Median : 399.0
##  Mean   : 481.7
##  3rd Qu.: 656.0
##  Max.   :4000.0
##
```

```
table(con$system)
```

```
##
## FALSE  TRUE
## 24653 26300
```

```
# Exploring Activities data
# add code here
head(a)
```

```
## # A tibble: 6 x 6
##   id                          student_id activity_step_id name  started completed
##   <chr>                       <chr>      <chr>            <chr> <lgl>   <lgl>
## 1 7ee28996-fb62-4dcc-83e2-8~ eec1223d-~ 7ee28996-fb62-4~ Week~ TRUE    TRUE
## 2 7ee28996-fb62-4dcc-83e2-8~ aacb24b7-~ 7ee28996-fb62-4~ Week~ TRUE    TRUE
## 3 7ee28996-fb62-4dcc-83e2-8~ 63103119-~ 7ee28996-fb62-4~ Week~ TRUE    TRUE
## 4 7ee28996-fb62-4dcc-83e2-8~ de3d339f-~ 7ee28996-fb62-4~ Week~ TRUE    TRUE
## 5 7ee28996-fb62-4dcc-83e2-8~ ce437628-~ 7ee28996-fb62-4~ Week~ TRUE    TRUE
## 6 7ee28996-fb62-4dcc-83e2-8~ b100be8d-~ 7ee28996-fb62-4~ Week~ TRUE    TRUE
```

```
n_distinct(a$student_id)
```

```
## [1] 259
```

```
summary(a)
```

```
##       id             student_id        activity_step_id       name
##  Length:1909        Length:1909        Length:1909        Length:1909
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##  started         completed
##  Mode:logical   Mode :logical
##  TRUE:1909      FALSE:271
##                 TRUE :1638
```

```
table(a$completed)
```

```
##
## FALSE  TRUE
##   271  1638
```

```
# Exploring connections between datasets
# add code here

# students in both datasets
common_students <- intersect(unique(con$student_id), unique(a$student_id))
length(common_students)
```

```
## [1] 258
```

```
###############################################
###############################################
```

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 trade-off 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 an elaboration activity submission. Specifically, **your goal is to predict one day before the submission deadline, if a student will forget to complete the elaboration activity**, 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 elaboration activities and some students who started but did not complete theirs.

We define an **incomplete submission** as having a FALSE for `completed` or no activity record at all (meaning the student did not even start the activity).

Instructions

Important note about the setup: The final prediction target (i.e. the test case) will be "Week 7 Reading Elaboration: Multimedia Learning". You should train your model to predict for all preceding elaboration activities (i.e., one in Week 2; two in Week 3; one in Week 6). Omit any Week 8 activities because they were not due when the data was extracted. You can use Homework Help activities to create features, but do not use them as training targets because these activities are optional.

1. Treat each elaboration activity assignment as a prediction task (thus there are x*n prediction opportunities where x = number of elaboration activities and n = number of students who have had at least one conversation)
2. Create a dataset that has 1 row per student per elaboration activity 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. You should assume due dates are Saturdays at midnight EST (which is 5am UTC the same day). I provide the deadlines in UTC below.
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

Reading Elaboration Deadlines (in UTC): - Week 2: 2025-02-01 05:00:00 - Week 3: 2025-02-08 05:00:00 - Week 6: 2025-03-01 05:00:00 - Week 7: 2025-03-08 05:00:00

Step 3: Getting you started

Create the outcome variable

Identify the target activities and whether a student did NOT complete it. Recall that we want to have a *warning* system, so the outcome should be the negative action (i.e. missing it).

Get the missing outcome for each elaboration activity, associate the deadline for each one, and then compute the timestamp for 24hrs prior to its deadline.

Now you know which elaboration activities to target. **Be sure to kick out the ones from Week 8**; They were not due yet when the export was created.

Question 2: Now build a dataset with an indicator for each person and each elaboration activity with 1=incomplete/not started, 0=complete. 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). Be sure to use all students in the `con` dataset as the basis, not just those who are in the `a` dataset because some students in the course may not have started any activity.

```
###############################################
####### BEGIN INPUT: Define outcome ###########
###############################################

student_list<-data.frame(student_id=unique(con$student_id))

a_names<-data.frame(name = unique(a$name))|>
  filter(str_detect(name, "Elaboration"))|>
  filter(name!="Week 8 Reading Elaboration: What's Wrong?")

deadlines <- data.frame(
  week = c(2, 6, 3, 7, 3),
  deadline = as.POSIXct(c("2025-02-01 05:00:00", "2025-03-01 05:00:00",
                          "2025-02-08 05:00:00", "2025-03-08 05:00:00",
                          "2025-02-08 05:00:00"), tz = "UTC"),
  names = a_names
)

joined<-inner_join(a, deadlines)
```

```
## Joining with ‘by = join_by(name)‘
```

```
all_combinations <- expand.grid(
  student_id = student_list$student_id,
  name = a_names$name,
  stringsAsFactors = FALSE
)


outcome <- left_join(all_combinations, joined) |>
    mutate(
    deadline_ts = as.numeric(deadline),
    cutoff_ts = deadline_ts - (24 * 60 * 60),
    missed = ifelse(is.na(completed) | !completed, 1, 0)
  )
```

```
## Joining with 'by = join_by(student_id, name)'
```

```
head(outcome)
```

```
##                             student_id
## 1 37298e1e-f8e8-4c1a-8aab-d01f79f51aec
## 2 66902053-8ab2-4ae5-af06-eab0dc0e0a3f
## 3 931f9101-341d-4250-bbc4-6e173dcec6f6
## 4 0b2d5e11-85a2-450c-aadc-4b59d647e0ce
## 5 516d8364-65aa-44e1-a6be-4e80b625eebe
## 6 4e6d36e2-b9b9-48e3-9495-b52a298f5c61
##                                                   name
## 1 Week 2 Reading Elaboration: Mining Big Data in Education
## 2 Week 2 Reading Elaboration: Mining Big Data in Education
## 3 Week 2 Reading Elaboration: Mining Big Data in Education
## 4 Week 2 Reading Elaboration: Mining Big Data in Education
## 5 Week 2 Reading Elaboration: Mining Big Data in Education
## 6 Week 2 Reading Elaboration: Mining Big Data in Education
##                                     id                       activity_step_id
## 1 7877f09d-5905-4c41-ae01-490a170b2ed0 7877f09d-5905-4c41-ae01-490a170b2ed0
## 2 7877f09d-5905-4c41-ae01-490a170b2ed0 7877f09d-5905-4c41-ae01-490a170b2ed0
## 3 7877f09d-5905-4c41-ae01-490a170b2ed0 7877f09d-5905-4c41-ae01-490a170b2ed0
## 4 7877f09d-5905-4c41-ae01-490a170b2ed0 7877f09d-5905-4c41-ae01-490a170b2ed0
## 5 7877f09d-5905-4c41-ae01-490a170b2ed0 7877f09d-5905-4c41-ae01-490a170b2ed0
## 6 7877f09d-5905-4c41-ae01-490a170b2ed0 7877f09d-5905-4c41-ae01-490a170b2ed0
##   started completed week            deadline deadline_ts  cutoff_ts missed
## 1    TRUE      TRUE    2 2025-02-01 05:00:00  1738386000 1738299600      0
## 2    TRUE      TRUE    2 2025-02-01 05:00:00  1738386000 1738299600      0
## 3    TRUE      TRUE    2 2025-02-01 05:00:00  1738386000 1738299600      0
## 4    TRUE      TRUE    2 2025-02-01 05:00:00  1738386000 1738299600      0
## 5    TRUE      TRUE    2 2025-02-01 05:00:00  1738386000 1738299600      0
## 6    TRUE      TRUE    2 2025-02-01 05:00:00  1738386000 1738299600      0
```

```
###############################################
###############################################
```

Feature Engineering

For each elaboration activity, identify what data is appropriate for feature engineering

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

Remember that the dataset we are aiming for has 1 row per student and activity 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 a specific deadline and create two basic features from the conversation. You should try to create a lot more features, including complex ones, that can use the conversation and activity data (but remember the timing constraint).

```
secs_in_day = 60 * 60 * 24
example_deadline = as.numeric(as.POSIXct("2025-03-01 05:00:00", tz = "UTC"))

example_features = con %>%
    filter(as.numeric(as.POSIXct(created_at, tz = "UTC")) < example_deadline - secs_in_day) %>%
    group_by(student_id) %>%
    summarise(
        num_chat_conversations = n_distinct(conversation_id[is.na(activity_step_id)]),
        avg_student_msg_len = mean(length_char[system==FALSE])
    )

head(example_features)
```

```
## # A tibble: 6 x 3
##   student_id                           num_chat_conversations avg_student_msg_len
##   <chr>                                                  <int>               <dbl>
## 1 008cc018-56f3-4d73-943a-e3323cd77a~                        1               346.
## 2 00e250b8-ac46-404e-9e97-a9bfc31e9c~                        0               212.
## 3 0123dccc-c0e5-47e6-8851-2c43d3cb84~                        5               124.
## 4 0223bad6-7402-488e-88db-6c2434ac60~                       25                44.2
## 5 03ae23d4-e564-4470-b5a4-3e3d5825c5~                        2               267.
## 6 0547e759-3e7c-42e4-b118-d348e67be2~                        6               162.
```

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

```
###############################################
###### BEGIN INPUT: Engineer features #########
###############################################

# using the week 7 deadline
week_7_deadline = as.numeric(as.POSIXct("2025-03-08 05:00:00", tz = "UTC"))

# Set the time window to the 24 hours before the due date
time_window_start = week_7_deadline - (60 * 60 * 24)

# features
features_week_7 = con %>%
  filter(as.numeric(as.POSIXct(created_at, tz = "UTC")) < time_window_start) %>%
  group_by(student_id) %>%
  summarise(

    num_chat_conversations = n_distinct(conversation_id[is.na(activity_step_id)]),  # Number of distinc

    num_student_msgs = sum(system == 0),
    avg_student_msg_len = mean(length_char[system == 0]),
    num_messages_ai = sum(system == 1),
    avg_ai_msg_len = mean(length_char[system == 1] ) ,
 # Number of messages by AI and Student, along with the lengths of the messages on  average

    total_time_spent = sum(length_char),
 # Total time spent in conversations (based on message length)

    prop_ai_msgs = num_messages_ai / (num_student_msgs + num_messages_ai) ,
 # Proportion of AI messages in all messages

    total_attempts = sum(!is.na(activity_step_id)),
 # Total number of attempts at the activity
```

```
  )

# Check the features
head(features_week_7)
```

```
## # A tibble: 6 x 9
##   student_id         num_chat_conversations num_student_msgs avg_student_msg_len
##   <chr>                               <int>            <int>               <dbl>
## 1 008cc018-56f3-4d7~                      1               59               254.
## 2 00e250b8-ac46-404~                      0                8               212.
## 3 0123dccc-c0e5-47e~                      6               51               143.
## 4 0223bad6-7402-488~                     25               32                44.2
## 5 03ae23d4-e564-447~                      2               52               288.
## 6 0547e759-3e7c-42e~                      8              243               168.
## # i 5 more variables: num_messages_ai <int>, avg_ai_msg_len <dbl>,
## #   total_time_spent <dbl>, prop_ai_msgs <dbl>, total_attempts <int>
```

```
################################################
################################################
```

Step 4: Split your dataset

Question 4: We would like to train the model on earlier assessments in order to make early alert predictions for later ones. As the hold-out test set, designate the most recently due elaboration activity (i.e. the one for Week 7). 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 activity_id of the Week 7 activity and store data associated with that period in the `test` dataset. Take all the remaining data (earlier periods for prior weeks) and store them in the `train` dataset.

```
################################################
######## BEGIN INPUT: Split dataset ###########
################################################

# Identify last due elaboration activity for testing
# add code here
week_7_activity = "Week 7 Reading Elaboration: Multimedia Learning"

# Split the dataset into train and test based on the activity_ids or periods
test <- outcome %>%
  filter(name == week_7_activity)
train <- outcome %>%
  filter(name != week_7_activity)

################################################
################################################
```

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

# Install packages if needed
if (!require("rpart")) install.packages("rpart")
```

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

```
if (!require("rpart.plot")) install.packages("rpart.plot")
```

```
## Loading required package: rpart.plot
```

```
if (!require("dplyr")) install.packages("dplyr")
if (!require("e1071")) install.packages("e1071")
```

```
## Loading required package: e1071
```

```
# Load libraries
library(rpart)
library(rpart.plot)
library(dplyr)
library(e1071)

# Merge training outcomes with features (using student_id)
train_data <- train %>%
  left_join(features_week_7, by = "student_id") %>%
  na.omit()

# Convert outcome to factor (0 = complete, 1 = missed)
train_data$missed <- as.factor(train_data$missed)

# --- CART Model ---
cart_model <- rpart(missed ~ num_chat_conversations + num_student_msgs + avg_student_msg_len +
                      num_messages_ai + avg_ai_msg_len + total_time_spent + prop_ai_msgs +
                      total_attempts,
                    data = train_data, method = "class")

# --- Naive Bayes Model ---
nb_model <- naiveBayes(missed ~ num_chat_conversations + num_student_msgs + avg_student_msg_len +
                         num_messages_ai + avg_ai_msg_len + total_time_spent + prop_ai_msgs +
                         total_attempts,
                       data = train_data)
```

```r
# Get predictions on training data
cart_train_preds <- predict(cart_model, train_data, type = "class")
nb_train_preds <- predict(nb_model, train_data, type = "class")

# Function to compute precision, recall, and F1
compute_metrics <- function(actual, predicted) {
  cm <- table(factor(actual, levels = c(0,1)), factor(predicted, levels = c(0,1)))
  TP <- cm["1","1"]
  FP <- cm["0","1"]
  FN <- cm["1","0"]
  precision <- ifelse((TP+FP)==0, 0, TP/(TP+FP))
  recall <- ifelse((TP+FN)==0, 0, TP/(TP+FN))
  F1 <- ifelse((precision+recall)==0, 0, 2/((1/recall)+(1/precision)))
  list(precision = precision, recall = recall, F1 = F1)
}

# Compute training F1 scores
cart_train_metrics <- compute_metrics(train_data$missed, cart_train_preds)
nb_train_metrics <- compute_metrics(train_data$missed, nb_train_preds)

cat("CART Training F1 Score:", round(cart_train_metrics$F1, 3), "\n")
```

```
## CART Training F1 Score: 0.64
```

```r
cat("Naive Bayes Training F1 Score:", round(nb_train_metrics$F1, 3), "\n")
```

```
## Naive Bayes Training F1 Score: 0.223
```

```r
################################################
################################################
```

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.

```r
################################################
####### BEGIN INPUT: Test and report ##########
################################################

# Merge test outcomes with features (using student_id)
test_data <- test %>%
  left_join(features_week_7, by = "student_id") %>%
  na.omit()

# Convert outcome to factor
test_data$missed <- as.factor(test_data$missed)
```

```r
# Get predictions on test data using the CART model
cart_test_preds <- predict(cart_model, test_data, type = "class")

# Get predictions on test data using the Naive Bayes model
nb_test_preds <- predict(nb_model, test_data, type = "class")

# Compute F1 scores for the test data
cart_test_metrics <- compute_metrics(test_data$missed, cart_test_preds)
nb_test_metrics <- compute_metrics(test_data$missed, nb_test_preds)

cat("CART Testing F1 Score:", round(cart_test_metrics$F1, 3), "\n")
```

```
## CART Testing F1 Score: 0.3
```

```r
cat("Naive Bayes Testing F1 Score:", round(nb_test_metrics$F1, 3), "\n")
```

```
## Naive Bayes Testing F1 Score: 0.143
```

```r
###############################################
###############################################
```

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 ###########

Our analysis evaluated CART and Naive Bayes models to predict students at risk of missing elaboration activity deadlines using LMS interaction data. Key predictive features used included the number of general chat conversations, average student message length, and total activity attempts. The CART model achieved a training F1 score of 0.64 and a test F1 score of 0.3. The drop in performance between the training and testing score indicates overfitting. The Naive Bayes models achieved a training F1 score of 0.223 and a test F1 score of 0.143. So the CART model did slightly better than the Naive Bayes model at predicting. The F1 score gives a balance of the accuracy and recall, and for both models, the score is low. Given low scores, we don't recommend implementing this system without further refinement. This could include exploring more features or looking at more advanced predictive models.

%##

Estimate time spent

We want to give students an estimate of how much time this project will take. Please indicate how many hours you spent as a team to complete this project here.

- We spent 5 hours.

Generative AI usage

As stated in the course syllabus, using generative AI is allowed to help you as you complete this project. We are interested in how it is being used and whether it is helpful for you.

- How much did you use generative AI (e.g., not at all, some, most, or all the questions) and which one did you use?
- If you used generative AI, how did you use it and was it helpful?

We used generative AI for question 2. It was helpful to check that we were building the dataset correctly and to check the syntax of functions like inner_join and left_join.

Submit Project

This is the end of the project. Please **Knit a Word doc report** that shows both the R code and R output (be sure to check the Word doc) and upload it on Canvas. One upload for the team before the deadline is sufficient.