Machine Learning: Build a baseline model

The LASER TEAM

Introduction

In this session, we will replicate the experiment in the paper (Predicting STEM and Non-STEM College Major Enrollment from Middle School Interaction with Mathematics Educational Software (https://educationaldatamining.org/EDM2014/uploads/procs2014/short%20papers/276_EDM-2014-Short.pdf)). We first need to understand the learning context and then build a machine learning model with the dataset. There is another critical step between these two steps, exploring variables in the dataset to gain an in-depth view of it. Given time constraints, we will skip this step in this session.

Understand the learning context

First, let's read the brief (four page) paper that will help to set the context for this session: Predicting STEM and Non-STEM College Major Enrollment from Middle School Interaction with Mathematics Educational Software (https://educationaldatamining.org/EDM2014/uploads/procs2014/short%20papers/276_EDM-2014-Short.pdf).

Use the following questions to guide your reading of the paper:

- What kinds of learning activities are available in the ASSISTment system?
- What's the research question?
- What's the prediction task?
- What are the variables used in the prediction task?
- How might these variables be informative for the prediction task?
- What kinds of new knowledge does this study add to the field of STEM education?

Your Turn ↘

- Which feature (variable) is a good predictor of the classification task? (poll)

Access the dataset

After reading the paper, we can see that the task is to predict whether a student will choose STEM majors and the authors used the following variables to predict STEM major enrollment: carelessness, knowledge, correctness, boredom, engaged concentration, confusion, frustration, off-task, gaming, and number of actions.

We will use a portion of the ASSISTment data from a data mining competition (https://sites.google.com/view/assistmentsdatamining/home?authuser=0) to replicate the experiment in the paper, but the prediction task is not whether a student will choose STEM majors, instead, it is predicting whether the students (who have now finished college) pursue a career in STEM fields (1) or not (0).

We will use the dataset (i.e, `dat_csv_combine_final.csv`) to run a logistic regression (i.e, a supervised Machine Learning algorithm which is used for classification problems) experiment.

Load the packages

We'll load four packages for this experiment. {tidymodels} consists of several core packages, including {rsample} (for sample splitting (e.g. train/test or cross-validation)), recipes (for pre-processing), {parsnip} (for specifying the model), and yardstick (for evaluating the model).

```
library(tidymodels)
```

```
## ── Attaching packages ─────────────────────────────────────────────────────
──────────────────────────────────── tidymodels 0.1.3 ──
```

```
## ✓ broom        0.7.6      ✓ recipes      0.1.16
## ✓ dials        0.0.9      ✓ rsample      0.0.9
## ✓ dplyr        1.0.6      ✓ tibble       3.1.1
## ✓ ggplot2      3.3.3      ✓ tidyr        1.1.3
## ✓ infer        0.5.4      ✓ tune         0.1.5
## ✓ modeldata    0.1.0      ✓ workflows    0.2.2
## ✓ parsnip      0.1.5      ✓ workflowsets 0.0.2
## ✓ purrr        0.3.4      ✓ yardstick    0.0.8
```

```
## ── Conflicts ────────────────────────────────────────────────────────────
──────────────────────────────── tidymodels_conflicts() ──
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## • Use tidymodels_prefer() to resolve common conflicts.
```

```
library(readr)
```

```
##
## Attaching package: 'readr'
```

```
## The following object is masked from 'package:yardstick':
##
##     spec
```

```
## The following object is masked from 'package:scales':
##
##     col_factor
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-2
```

```
library(here)
```

```
## here() starts at /Users/joshuarosenberg/aera-workshop
```

Load the dataset

```
dat_csv_combine <- read_csv(here("data", "dat_csv_combine_final.csv"))
```

```
##
## ── Column specification ────────────────────────────────────────────────────────
──────────────────────────────────────────────────────
## cols(
##   AveCarelessness = col_double(),
##   AveKnow = col_double(),
##   AveCorrect.x = col_double(),
##   AveResBored = col_double(),
##   AveResEngcon = col_double(),
##   AveResConf = col_double(),
##   AveResFrust = col_double(),
##   AveResOfftask = col_double(),
##   AveResGaming = col_double(),
##   NumActions = col_double(),
##   isSTEM = col_double()
## )
```

```
dat_csv_combine <- dat_csv_combine %>%
  mutate(isSTEM = as.character(isSTEM))
```

Check the dimension of the dataset using glimpse:

```
glimpse(dat_csv_combine)
```

```
## Rows: 514
## Columns: 11
## $ AveCarelessness <dbl> 0.15884823, 0.09419537, 0.11115940, 0.12640124, 0.09322404,
0.08986537, 0.06911221, 0.04433382, 0.06904095, 0.0748813…
## $ AveKnow         <dbl> 0.25516410, 0.19194831, 0.25083836, 0.22998826, 0.14635207,
0.15090116, 0.11683027, 0.08263669, 0.10922893, 0.1331100…
## $ AveCorrect.x    <dbl> 0.3796576, 0.4132492, 0.5000000, 0.4454073, 0.3096614, 0.3279
110, 0.2910906, 0.2044199, 0.2937595, 0.3171472, 0.43893…
## $ AveResBored     <dbl> 0.2227958, 0.2614552, 0.2731885, 0.2426905, 0.2114054, 0.2213
441, 0.1994639, 0.1975080, 0.1955164, 0.2302150, 0.24180…
## $ AveResEngcon    <dbl> 0.6500791, 0.6386360, 0.6469445, 0.6588631, 0.6557109, 0.6582
559, 0.6645131, 0.6865811, 0.6876323, 0.6702865, 0.64011…
## $ AveResConf      <dbl> 0.06998741, 0.06397125, 0.10460163, 0.11235571, 0.07930752,
0.10194228, 0.08489538, 0.02904180, 0.08965089, 0.0992465…
## $ AveResFrust     <dbl> 0.16434746, 0.12985820, 0.13163183, 0.13222999, 0.10291970,
0.11264519, 0.13348763, 0.12744520, 0.12416391, 0.1160137…
## $ AveResOfftask   <dbl> 0.1531466, 0.2343312, 0.2300051, 0.1853302, 0.1268604, 0.1376
487, 0.1139429, 0.1329352, 0.1120463, 0.1848881, 0.17698…
## $ AveResGaming    <dbl> 0.23679964, 0.04509638, 0.04023338, 0.07782716, 0.24548949,
0.17318927, 0.34492697, 0.42362468, 0.34898425, 0.2904710…
## $ NumActions      <dbl> 993, 317, 406, 577, 1211, 1168, 1302, 724, 1314, 659, 565, 75
4, 883, 852, 661, 1490, 1766, 640, 1119, 999, 1052, 506,…
## $ isSTEM          <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", "0", "0", "0", "0", "0", "1", "1", "1…
```

You will see that there are 514 students and 11 variables in this dataset.

Next, let's check details about this dataset using the handy `count()` function, which works as follows

```
dat_csv_combine %>%
  count(isSTEM)
```

```
## # A tibble: 2 x 2
##   isSTEM     n
##   <chr>  <int>
## 1 0        350
## 2 1        164
```

We can see, in this dataset, 164 students chose STEM and 350 students chose non-STEM.

Your Turn ⤵

- Without a model, just with random guess, how likely you would make a correct prediction of whether a student would choose STEM?

This is an imbalanced dataset (164:350). While there are several methods for combating this issue using recipes (search for steps to upsample or downsample) or other more specialized packages like themis, the analyses shown below analyze the data as-is.

Now, look closely at the values of features and you will notice that the range of "number of actions" is not the same as the other variables. We should scale it so that they have the same range. We won't use tidyverse code (and the `mutate()` function) below, as it's a bit easier to write this code this way, using the `$` operator to directly change the variable.

```
dat_csv_combine$NumActions <- rescale(dat_csv_combine$NumActions,
                                      to = c(0, 1),
                                      from = range(dat_csv_combine$NumActions,
                                                   na.rm = TRUE,
                                                   finite = TRUE))
```

Data splitting & resampling

For a data splitting strategy, let's reserve 25% of the data to the test set. Since we have an imbalanced dataset, we'll use a stratified random sample:

```
set.seed(123) #set.seed is used so that we can reproduce the result since splitting migh
t be different every time we use initial_split function

splits <- initial_split(dat_csv_combine, strata = isSTEM)

data_other <- training(splits) #this dataset will be used to build models

data_test  <- testing(splits) #we will never use this dataset in the process of building
models. This dataset is only for final test!
```

The initial_split() function is specially built to separate the dataset into a training and testing set. By default, it holds 3/4 of the data for training and the rest for testing. That can be changed by passing the prop argument. For instance, `splits <- initial_split(dat_csv_combine, prop = 0.6)`.

Here is training set proportions by `isSTEM`:

```
data_other %>%
  count(isSTEM)
```

```
## # A tibble: 2 x 2
##   isSTEM     n
##   <chr>  <int>
## 1 0        263
## 2 1        123
```

Here is test set proportions by `isSTEM`:

```
data_test  %>%
  count(isSTEM)
```

```
## # A tibble: 2 x 2
##   isSTEM     n
##   <chr>  <int>
## 1 0         87
## 2 1         41
```

Single resample

To build the model, let's create a single resample called a validation set. In {tidymodels}, a validation set is treated as a single iteration of resampling. This will be a split from the 386 students that were not used for testing, which we called data_other. This split creates two new datasets:

- the set held out for the purpose of measuring performance, called the validation set, and
- the remaining data used to fit the model, called the training set.

We'll use the `validation_split()` function to allocate 20% of the data_other dataset to the validation set and the remaining 80% to the training set. This means that our model performance metrics will be computed on a single set of 76 (or 78) instances (20%*386). This is fairly small, we usually would not do single resample with such as small dataset. Here, we are doing it so that we know how to run a single resample experiment.

```
set.seed(234)
# Create data split for train and test
data_other_split <- initial_split(data_other,
                                  prop = 0.8,
                                  strata = isSTEM)

# Create training data
data_other_train <- data_other_split %>%
                    training()

# Create testing data
data_other_validation <- data_other_split %>%
                    testing()

# Checking the number of rows in train and test dataset
nrow(data_other_train)
```

```
## [1] 310
```

```
nrow(data_other_validation)
```

```
## [1] 76
```

This function, like initial_split(), has the same strata argument, which uses stratified sampling to create the resample. This means that we'll have roughly the same proportions of students who chose and did not choose STEM in our new validation and training sets, as compared to the original data_other proportions.

A first model

Since our outcome variable `isSTEM` is categorical, logistic regression would be a good first model to start. This is also the algorithm in the paper. For logistic regression, the predicted label should be factor.

```
data_other_train <- data_other_train %>%
  mutate(isSTEM = as.factor(isSTEM))

data_other_validation <- data_other_validation %>%
  mutate(isSTEM = as.factor(isSTEM))
```

Next, let's create the model with data_other_train.

```
fitted_logistic_model<- logistic_reg() %>%
        # Set the engine
        set_engine("glm") %>%
        # Set the mode
        set_mode("classification") %>%
        # Fit the model
        fit(isSTEM~., data = data_other_train) #the training data is data_other_train an
d the predicted label is isSTEM
```

Now, let's take a look at the model

```
temp <- tidy(fitted_logistic_model)    # Generate Summary Table
temp
```

```
## # A tibble: 11 x 5
##    term            estimate std.error statistic p.value
##    <chr>              <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)      -1.79        6.92   -0.259    0.796
##  2 AveCarelessness   5.39        8.68    0.622    0.534
##  3 AveKnow           0.200       6.22    0.0321   0.974
##  4 AveCorrect.x      1.86        3.85    0.482    0.630
##  5 AveResBored       2.44       14.1     0.173    0.863
##  6 AveResEngcon     -2.34        7.68   -0.304    0.761
##  7 AveResConf        5.51        4.88    1.13     0.259
##  8 AveResFrust      -0.694       2.96   -0.235    0.814
##  9 AveResOfftask    -1.15        4.41   -0.260    0.795
## 10 AveResGaming      0.351       3.01    0.117    0.907
## 11 NumActions        0.0999      1.63    0.0612   0.951
```

Your Turn ⤵

- Which feature is a good predictor of the classification task? (poll)

You might notice that not all features are significant predictors as we have a really small dataset for training the data. It means that with this model, our prediction might not be accurate. Let's take a look at the accuracy. We will use data_other_validation to get the model performance.

```
# Class prediction
pred_class <- predict(fitted_logistic_model,
                      new_data = data_other_validation,
                      type = "class")

pred_class[1:5,] # this gives us the first 5 predicted results
```

```
## # A tibble: 5 x 1
##   .pred_class
##   <fct>
## 1 0
## 2 0
## 3 0
## 4 0
## 5 0
```

Let's compare the predicted result and the true value:

```
prediction_results <- data_other_validation %>%
  select(isSTEM) %>%
  bind_cols(pred_class)

prediction_results[1:5, ] # this gives us the first 5 true values versus predicted resul
ts
```

```
## # A tibble: 5 x 2
##   isSTEM .pred_class
##   <fct>  <fct>
## 1 0      0
## 2 0      0
## 3 1      0
## 4 0      0
## 5 0      0
```

Next, let's take a look at the accuracy of the model with function accuracy. We can use percent accuracy and chance corrected accuracy (i.e., Kappa) to evaluate the model. We can get the percent accuracy:

```
accuracy(prediction_results, truth = isSTEM,
         estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.671
```

We can get the Kappa:

```
kap(prediction_results, truth = isSTEM,
         estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 kap     binary        0.0326
```

We notice that the model is not doing well. Next, we want to closely look at the confusion matrix to analyze the model performance.

```
conf_mat(prediction_results, truth = isSTEM,
         estimate = .pred_class)
```

```
##           Truth
## Prediction  0  1
##          0 49 22
##          1  3  2
```

Your Turn ⤵

- What do you notice?