Predictive Analytics Tutorial

OpenSDP

September 9, 2017

Table of Contents

## Introduction

This file provides guidance and R syntax examples for the hands-on predictive analytics session during the Fall 2017 Cohort 8 Strategic Data Project Workshop in Philadelphia.

During the workshop, we'll ask you develop a predictive college-going indicator for the state of Faketucky using student data collected through the end of 11th grade. You can take any approach you like to do this. Your goal is to make the best predictions possible, and then think about how the predictive model would work in the real world, and then recommend an indicator. In the real world, the indicator you recommend might or might not be the most predictive one--you might argue for one that is more useful because it gives predictions sooner in a student's academic career, or you might argue for one that is slightly less accurate but simpler and easier to explain.

Logistic regression is one tool you can use, and we'll demonstrate it here. There are many other techniques of increasing complexity. (Many of the best predictive analytics packages are written in the R programming language.) But for a binary outcome variable, most data scientists start with logistic regressions, and those are very straightforward to do in R.

Here are the steps:

1. explore the data, especially college enrollment predictors and outcomes
2. examine the relationship between predictors and outcomes
3. evaluate the predictive power of different variables and select predictors for your model
4. make predictions using logistic regression
5. convert the predicted probabilities into a 0/1 indicator
6. look at the effect of different probability cutoffs on prediction accuracy (develop a "confusion matrix")

When you've been through those steps with your first model, you can submit it to Kaggle for scoring, and then iterate through the process again until you are satisfied with the results.

The commands in this script won't tell you everything you need to do to develop your model, but they will give you command syntax that you should be able to adjust and adapt to get the project done. You can also take an even simpler approach, outlined in the Chicago Consortium on School Research CRIS technical guide assigned in the workshop pre-reading. With that "checklist" approach, you experiment with different thresholds for your predictor variables, and combine them to directly predict 0/1 values without using the predict command after running a logistic regression. The CCSR approach has the advantage of being easy to explain and implement, but it might not yield the most accurate predictions. We won't demonstrate that approach here, but if you want to try it you can draw on the syntax examples here and follow the instructions in the CCSR technical guide.

Before you get started, you need to think about variables, time, and datasets. The sooner in a student's academic trajectory you can make a prediction, the sooner you can intervene--but the less accurate your predictions, and hence your intervention targeting, is likely to be. What data, and specifically which variables, do you have available to make predictions? What outcome are you trying to predict? It can be helpful to group the data you have available by time categories: pre-high school, early high school, late high school, and graduation/post-secondary. One fundamental rule is that you can't use data from the future to make predictions. If you're planning to use your model to make predictions for students at the end of 11th grade, for instance, and if most students take AP classes as seniors, you can't use data about AP coursetaking collected during senior year to predict the likelihood of college enrollment, even if you have that data available for past groups of students.

In terms of datasets, you can develop your model and then test its accuracy on the dataset you used to develop the model, but that is bad practice--in the real world, your model is only as good as its predictions on different, out of sample datasets. It's good practice to split your data into three parts: one part for developing your model, one for repeatedly testing different versions of your model, and a third to use for a final out of sample test.

We're using two cohorts of high-school students for the predictive analytics task--students who were ninth graders in 2009 and in 2010. In a production predictive analytics model for a school system, you might split data from the most recent cohort for which you have data into two parts for model development and testing, and then check the model against outcomes for the next year's cohort when it became available.

For the workshop, though, we're using the online Kaggle competition platform to evaluate model accuracy and the data is split somewhat differently. The 2009 data is available to you for model development. Kaggle has randomly split the 2010 data, which you'll use to make predictions with your model for scoring, into two parts. Kaggle will show scoring results for the first part on a public leaderboard, but final scores will depend on how the model performs on the second half of the data.

One last point--in the real world, you'll need to make predictions for every student, even if you're missing data for that student which your model needs in order to run. Just making predictions using a logistic regression won't be enough. You'll need to use decision rules based on good data exploration and your best judgment to predict and fill in outcomes for students where you have insufficient data.

If you're using the Rmd file version of these materials, start by saving a new version of the file, so you can edit it without worrying about overwriting the original. Then work through the file inRStudio by highlighting one or a few command lines at a time, clicking the "execute" icon (or pressing control-enter), and then looking at the results in the R console. Edit or add commands as you wish.

If you're using a paper or PDF version of these materials, just read on--the R output appears below each section of commands. This script uses the 2009 cohort data, which has one observation (row) per student. Each observation contains data about demographics, academic performance, school and district enrollment, and high school and post-secondary outcomes. It also has information about the characteristics of the colleges that students attended. To work through this script, you need to put the training\_2009.csv data file on your computer in a working folder of your choice, and then edit the commands below to tell R where to look for the data. If you have trouble doing this, ask for help from other members of your group.

## Setup

To prepare for this project you will need to ensure that your R installation has the necessary add-on packages and that you can read in the training data.

# Install add-on packages needed  
install.packages("dplyr") # this will update your installed version to align with   
install.packages("pROC") # those in the tutorial   
install.packages("devtools")

# Load the packages you need  
library(dplyr)  
library(pROC)  
library(devtools)  
  
# Load the helper functions not in packages  
devtools::source\_gist("ed47cd156462a9900df1f77a000f4a52",   
 filename = "helper\_funcs.R")  
  
# Read in the data  
# This command assumes that the data is in a folder called data, below your   
# current working directory. You can check your working directory with the   
# getwd() command, and you can set your working directory using the RStudio   
# environment, or the setwd() command.  
  
train\_data <- read.csv("data/training\_2009.csv", stringsAsFactors = FALSE)

### Validate the data

Ensure that the data imported correctly.

First, ensure that the data is unique by student ID.

nrow(train\_data) == n\_distinct(train\_data$sid)

## Explore the Data

Verify that the data includes just students who were ninth-graders in 2009

table(train\_data$chrt\_ninth, useNA = "always")

When did these students graduate (if they did)?

table(train\_data$chrt\_grad, useNA = "always")

Now that we have a sense of the time structure of the data, let's look at geography. How many high schools and how many districts are? What are those regional education services coops?

length(unique(train\_data$first\_hs\_name))  
length(unique(train\_data$first\_dist\_name))  
table(train\_data$first\_coop\_code, useNA = "always")

Which districts are part of the coop region you have been assigned to, and how many students do they have? Find out the abbreviation code for your coop and then replace the my\_coop variable below.

my\_coop <- "NKCES"  
table(train\_data$first\_dist\_name[train\_data$first\_coop\_code == my\_coop],   
 useNA = "always")

What are outcome variables, and what are potential predictor variables? What student subgroups are we interested in? Let's start by looking at student subgroups. Here's gender.

table(train\_data$male, useNA="always")

Here's a short for-loop to look at one-way tabs of a lot of variables at once.

for(i in c("male", "race\_ethnicity", "frpl\_11", "sped\_11", "lep\_11",   
 "gifted\_11")){  
 print(i)  
 print(table(train\_data[, i], useNA="always"))  
}

Note that when we read the data in, race\_ethnicity contains blank values that are not marked as NA by R. Let's change that here:

train\_data$race\_ethnicity[train\_data$race\_ethnicity == ""] <- NA  
table(train\_data$race\_ethnicity, useNA = "always")

Let's examine the distribution of student subgroups by geography. For this command, we'll use the same looping syntax from above, which lets you avoid repetition by applying commands to multiple variables at once. You can type ?for into the R console if you want to learn more about how to use loops in R.

for(var in c("male", "race\_ethnicity", "frpl\_11", "sped\_11", "lep\_11", "gifted\_11")){  
 print(var)  
 print( # have to call print inside a loop  
 round( # round the result  
 prop.table( # convert table to percentages  
 table(train\_data$first\_coop\_code, train\_data[, var], # build the table  
 useNA = "always"),   
 margin = 2), # calculate percentages by column, change to 1 for row  
 digits = 3) # round off at 3 digits  
 \*100 ) # put on percentage instead of proportion scale  
}

Now, let's look at outcomes. We won't examine them all, but you should. Here's a high school graduation outcome variable:

table(train\_data$ontime\_grad, useNA = "always")

Wait! What if the data includes students who transferred out of state? That might bias the graduation rate and make it too low, because those ninth graders might show up as having dropped out.

table(train\_data$transferout, useNA = "always")

It looks like the data has been cleaned to include only students who did not transfer out.

Let's look at the distribution of this outcome variable by geography and then by subgroup.

prop.table(  
 table(train\_data$first\_coop\_code, train\_data$ontime\_grad, useNA="always"),  
 margin = 1)  
  
for(var in c("male", "race\_ethnicity", "frpl\_11", "sped\_11", "lep\_11", "gifted\_11")){  
 print(var)  
 print(  
 prop.table(  
 table(grad = train\_data$ontime\_grad, var = train\_data[, var],   
 useNA = "always"),   
 margin = 1)  
 )  
}

What are other outcome variables? Can you identify and examine the college enrollment variables? For each outcome that you are interested in, you can copy and paste the commands below, change my\_var to your variable of interest, and then run the commands. If you don't have time to do this right now, skip forward to the next set of commands.

my\_var <- "enroll\_yr2\_any"  
prop.table(table(train\_data$first\_coop\_code, train\_data[, my\_var],   
 useNA = "always"), 1)

It looks like there is a college readiness indicator in the data, but it's zero except for a handful of students. In fact, the statewide college readiness indicator wasn't implemented until the 2010 cohort. You'll be able to compare your college readiness indicator to the Faketucky college readiness indicator when you score your model in Kaggle.

table(train\_data$collegeready\_ever\_in\_hs, useNA = "always")

Next, identify and examine the performance and behavioral variables that you can use as predictors. These are mostly numerical variables, so you should use the summary, histogram, and table commands to explore them. Here's some syntax for examining 8th grade math scores. You can replicate and edit it to examine other potential predictors and their distributions by different subgroups.

summary(train\_data$scale\_score\_8\_math)  
hist(train\_data$scale\_score\_8\_math)

by(train\_data$scale\_score\_8\_math, train\_data$first\_coop\_code, FUN = mean,   
 na.rm=TRUE)  
  
by(train\_data$scale\_score\_8\_math, train\_data$frpl, FUN = mean,   
 na.rm=TRUE)

Finally, here's some sample code you can use to look at missingness patterns in the data. Note we use the is.na() function to test whether a value is missing.

for(var in c("first\_coop\_code", "male", "race\_ethnicity")){  
 print(var)  
 print(  
 prop.table(table(train\_data[, var],   
 "missing\_math" = is.na(train\_data$scale\_score\_8\_math)), 1)  
 )  
}

Did you see any outlier or impossible values while you were exploring the data? If so, you might want to truncate them or change them to missing. Here's how you can replace a numeric variable with a missing value if it is larger than a certain number (in this case, 100 percent).

hist(train\_data$pct\_absent\_11)

train\_data$pct\_absent\_11[train\_data$pct\_absent\_11 > 100] <- NA  
hist(train\_data$pct\_absent\_11)

Now that you've explored the data, you can start to examine the relationship between predictor and outcome variables. Here we'll continue to look at the high school graduation outcome, and we'll restrict the predictors to just two: 8th grade math scores and percent of enrolled days absent through 11th grade. For your college-going model, you can of course use more and different predictor variables. First, check the correlation between outcome and predictors.

cor(train\_data[, c("ontime\_grad", "scale\_score\_8\_math", "pct\_absent\_11")],   
 use = "pairwise.complete.obs")

A correlation is just one number, and it would be nice to have a better idea of the overall relationship between outcomes and predictors. But you can't make a meaningful scatterplot when the independent, or y value, is a binary outcome variable (try it!). Here's some code to make plots that give you a clearer look at the relationship between our predictors and outcomes.

The idea behind this code is to show the mean of the outcome variable for each value of the predictor, or for categories of the predictor variable if it has too many values. First, define categories (in this case, round to the nearest percentage) of the percent absent variable, and then truncate the variable so that low-frequency values are grouped together.

train\_data$pct\_absent\_cat <- round(train\_data$pct\_absent\_11, digits = 0)  
table(train\_data$pct\_absent\_cat)  
train\_data$pct\_absent\_cat[train\_data$pct\_absenct\_cat >= 30] <- 30

Next, define a variable which is the average ontime graduation rate for each absence category, and then make a scatter plot of average graduation rates by absence percent.

train\_data <- train\_data %>%   
 group\_by(pct\_absent\_cat) %>% # perform the operation for each value   
 mutate(abs\_ontime\_grad = mean(ontime\_grad, na.rm=TRUE)) # add a new variable  
  
plot(train\_data$pct\_absent\_cat, train\_data$abs\_ontime\_grad)

You can do the same thing for 8th grade test scores. First look at the math test score and notice that some scores appear to be outliers.

hist(train\_data$scale\_score\_8\_math)

train\_data$scale\_score\_8\_math[train\_data$scale\_score\_8\_math > 80] <- NA  
hist(train\_data$scale\_score\_8\_math)

You can do the same plot as above now by modifying the group\_by() command.

train\_data <- train\_data %>%   
 mutate(math\_8\_cut = ntile(scale\_score\_8\_math, n = 100)) %>%  
 group\_by(math\_8\_cut) %>% # perform the operation for each value   
 mutate(math\_8\_ontime\_grad = mean(ontime\_grad, na.rm=TRUE)) # add a new variable  
  
plot(train\_data$math\_8\_cut, train\_data$math\_8\_ontime\_grad)

## Model

Now we're ready to call on the logit command to examine the relationship between our binary outcome variable and our predictor variables. When you run a logistic regression with the logit command, R calculates the parameters of an equation that fits the relationship between the predictor variables and the outcome. A regression model typically won't be able to explain all of the variation in an outcome variable--any variation that is left over is treated as unexplained noise in the data, or error, even if there are additional variables not in the model which could explain more of the variation.

Once you've run a logit regression, you can have R generate a variable with new, predicted outcomes for each observation in your data with the predict command. The predictions are calculated using the model equation and ignore the unexplained noise in the data. For logit regressions, the predicted outcomes take the form of a probability ranging 0 and 1. To start with, let's do a regession of ontime graduation on eighth grade math scores.

math\_model <- glm(ontime\_grad ~ scale\_score\_8\_math, data = train\_data,   
 family = "binomial") # family tells R we want to fit a logistic

The default summary output for logistic regression in R is not very helpful.

summary(math\_model)

Even before you use the predict command, you can use the logit output to learn something about the relationship between the predictor and the outcome variable. The Pseudo (read R-squared) is a proxy for the share of variation in the outcome variable that is explained by the predictor. Statisticians don't like it when you take the pseudo too seriously, but it can be useful in predictive exercises to quickly get a sense of the explanatory power of variables in a logit model.

logit\_rsquared(math\_model)

Does adding polynomial terms increase the pseudo ? You can use the formula interface in R to add functional transformations of predictors without generating new variables and find out.

math\_model2 <- glm(ontime\_grad ~ scale\_score\_8\_math +   
 I(scale\_score\_8\_math^2) + I(scale\_score\_8\_math^3),   
 data = train\_data,   
 family = "binomial") # family tells R we want to fit a logistic  
logit\_rsquared(math\_model2)

The model did not improve very much. Any time you add predictors to a model, the will increase, even if the variables are fairly meaningless, so it's best to focus on including predictors that add meaningful explanatory power.

Now take a look at the for the absence variable.

absence\_model <- glm(ontime\_grad ~ pct\_absent\_11, data = train\_data,   
 family = "binomial")   
summary(absence\_model)  
logit\_rsquared(absence\_model)

Let's combine our two predictors and test their combined power.

combined\_model <- glm(ontime\_grad ~ pct\_absent\_11 + scale\_score\_8\_math,   
 data = train\_data, family = "binomial")  
summary(combined\_model)  
logit\_rsquared(combined\_model)

Using this combined model, let's use the predict command to make our first predictions.

train\_data$grad\_pred <- predict(combined\_model, newdata = train\_data,  
 type = "response") # this tells R to give us a probability

This generates a new variable with the probability of ontime high school graduation, according to the model. But if you look at the number of observations with predictions, you'll see that it is smaller than the total number of students. This is because R doesn't use observations that have missing data for any of the variables in the model.

table(is.na(train\_data$grad\_pred))

Let's convert this probability to a 0/1 indicator for whether or not a student is likely to graduate ontime. If the probability in the model is equal to or greater than .5, or 50%, we'll say the student is likely to graduate.

train\_data$grad\_indicator <- ifelse(train\_data$grad\_pred > 0.5, 1, 0)  
table(train\_data$grad\_indicator, useNA = "always")

Lets evaluate the accuracy of the model by comparing the predictions to the actual graduation outcomes for the students for whom we have predictions. This type of crosstab is called a "confusion matrix." The observations in the upper right corner, where the indicator and the actual outcome are both 0, are true negatives. The observations in the lower right corner, where the indicator and the outcome are both 1, are true positives. The upper right corner contains false positives, and the lower left corner contains false negatives. Overall, if you add up the cell percentages for true positives and true negatives, the model got 84.5 percent of the predictions right.

prop.table(table(train\_data$ontime\_grad, train\_data$grad\_indicator)) %>% # shorthand way to round  
 round(3)

However, almost all of the wrong predictions are false positives--these are students who would not have been flagged as dropout risks even though they didn't graduate ontime. If you want your indicator system to be have fewer false positives, you can change the probability cutoff. This cutoff has a lower share of false positives and a higher share of false negatives, with a somewhat lower share of correct predictions.

prop.table(table(Observed = train\_data$ontime\_grad,   
 Predicted = train\_data$grad\_pred > 0.75)) %>%   
 round(3)

Note that this table only includes the complete cases. To look at missing values as well:

prop.table(table(Observed = train\_data$ontime\_grad,   
 Predicted = train\_data$grad\_pred > 0.75,   
 useNA="always")) %>% round(3)

## Missing Data

How should we handle the students with missing data? A predictive analytics system is more useful if it makes an actionable prediction for every student. And, the students missing 8th grade test scores are likely to be higher mobility students; you can check and see that they have a much lower graduation rate than the students who do have test scores.

table(Grad = train\_data$ontime\_grad,   
 miss\_math = is.na(train\_data$scale\_score\_8\_math))

There are a number of options. One is to run a model with fewer variables for only those students, and then use that model to fill in the missing indicators.

absence\_model <- glm(ontime\_grad ~ pct\_absent\_11,   
 data = train\_data[is.na(train\_data$scale\_score\_8\_math),],   
 family = "binomial")

train\_data$grad\_pred\_2 <- predict(absence\_model, newdata = train\_data,   
 type = "response")  
summary(absence\_model)

train\_data$grad\_indicator[is.na(train\_data$grad\_pred) &   
 train\_data$grad\_pred\_2 < 0.75] <- 0  
train\_data$grad\_indicator[is.na(train\_data$grad\_pred) &   
 train\_data$grad\_pred\_2 >= 0.75] <- 1

We now have predictions for all but a very small share of students, and those students are split between graduates and non-graduates. We have to apply a rule or a model to make predictions for them--we can't use information from the future, except to develop the prediction system. We'll arbitrarily decide to flag them as potential non-graduates, since students with lots of missing data might merit some extra attention.

table(train\_data$grad\_indicator, useNA = "always")  
train\_data$grad\_indicator[is.na(train\_data$grad\_indicator)] <- 0

## Evaluate Fit

Now we have a complete set of predictions from our simple models. How well does the prediction system work? Can we do better?

table(Observed = train\_data$ontime\_grad, Predicted = train\_data$grad\_indicator) %>%   
 prop.table %>% round(4)

A confusion matrix is one way to evaluate the success of a model and evaluate tradeoffs as you are developing prediction systems, but there are others. The metric used in the Kaggle competition is AUC, which stands for "area under the curve." You'll learn more about ways to evaluate a prediction system, including the AUC metric, during Day 2 of the workshop, but here's a sneak peak. First, look at row percentages instead of cell percentages in the confusion matrix.

table(Observed = train\_data$ontime\_grad, Predicted = train\_data$grad\_indicator) %>%   
 prop.table(margin = 1) %>% round(4)

Next, use the "roctab" command to plot the true positive rate (sensitivity in the graph) against the false positive rate (1-specificity in the graph). You can see these percentages match the row percentages in the last table. The AUC is the "area under ROC curve" in this graph, and it is a useful single-number summary of predictive accuracy.

roc(train\_data$ontime\_grad, train\_data$grad\_indicator) %>% plot

You can also calculate ROC on the continuouse predictor as well, to help you determine the threshold:

roc(train\_data$ontime\_grad, train\_data$grad\_pred) %>% plot

A couple of last thoughts and notes. First, note that so far we haven't done any out-of-sample testing. If you wanted to develop the best model you could to predict ontime high school graduation with just this data, you should subdivide the dataset so that you would have out of sample data to use for testing. You'll be able to test your models for the college enrollment Kaggle competition using 2010 cohort data.

Second, should we use subgroup membership variables to make predictions, if they improve the accuracy of predictions? This is more a policy question than a technical question, and you should consider it when you are developing your models. You'll also want to check to see how accurate your model is for different subgroups.

## Export to Kaggle

Finally, once you've made your college outcome predictions, you'll want to export them to a text file for uploading into Kaggle. Here's the syntax for exporting a list of student IDs and outcomes in text format for Kaggle.

The file will be exported to your current working directory which you can check with getwd().

write.csv(train\_data[, c("sid", "grad\_indicator")],   
 "prediction\_1.csv", row.names=FALSE)