Decision Trees

Predict the college course dropouts

Eddie Lin

2018/03/13

Project Description

College course dropouts could be an administrative and a financial issue for universities. As there may be different reasons that students drop a course, it is useful to see what are the characteristics of the students and the nature of the course that are related to dropouts. In this project, I will use 3 different decision tree models to predict college dropouts. Specifically, my interested questions are as follows:

1. Which tree model works the best in this current case to predict student dropouts?

2. How to increase tree model performance to catch more true dropouts(true posive)?

3. What are the important features about students that could lead to dropouts? *(though this can also be done w/ logistic regresion)

Tools & Data

- R
- R packages: caret, CART, C5.0, RandomForest

Data Description

The data comes from a university registrar's office. The definitions of variables in this data set are as follows:

- **student_id**: Student ID
- **years**: Number of years the student has been enrolled in their program of study
- **entrance_test_score**: Entrance exam test score
- **courses_taken**: Number of courses a student has taken during their program
- **complete**: Whether or not a student completed a course or dropped out (yes = completed)
- **enroll_data_time**: Date and time student enrolled in POSIXct format
- **course_id**: Course ID
- **international**: Is the student from overseas
- **online**: Is the student only taking online courses
- **gender**: One of five possible gender identities

Data split

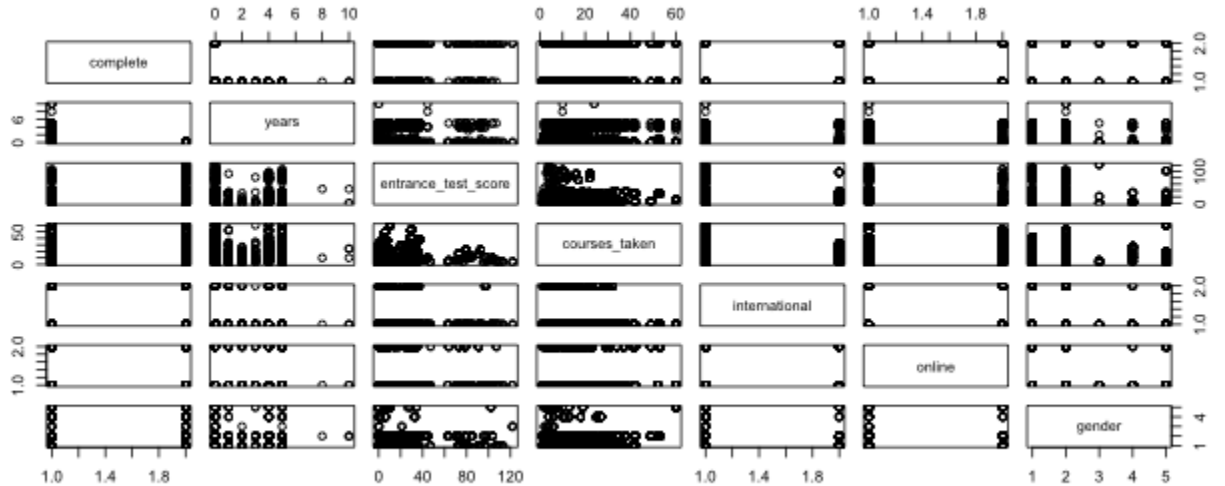
We will start by splitting the train and test data. In the current data set, the same students can take multiple courses (multiple rows associated with the same student ID), so we will randomly draw 25% of the students based on their IDs but not the number of rows

```
library(dplyr)
library(tidyr)
library(ggplot2)

set.seed(12345)
D1 <- as.data.frame(read.csv("drop-out.csv"))
train <- D1 %>% filter(student_id %in% sample(unique(student_id),cei
test <- D1[!(D1$student_id %in% train$student_id),]
D1 <- as.data.frame(read.csv("drop-out.csv"))
```

Feature visualization

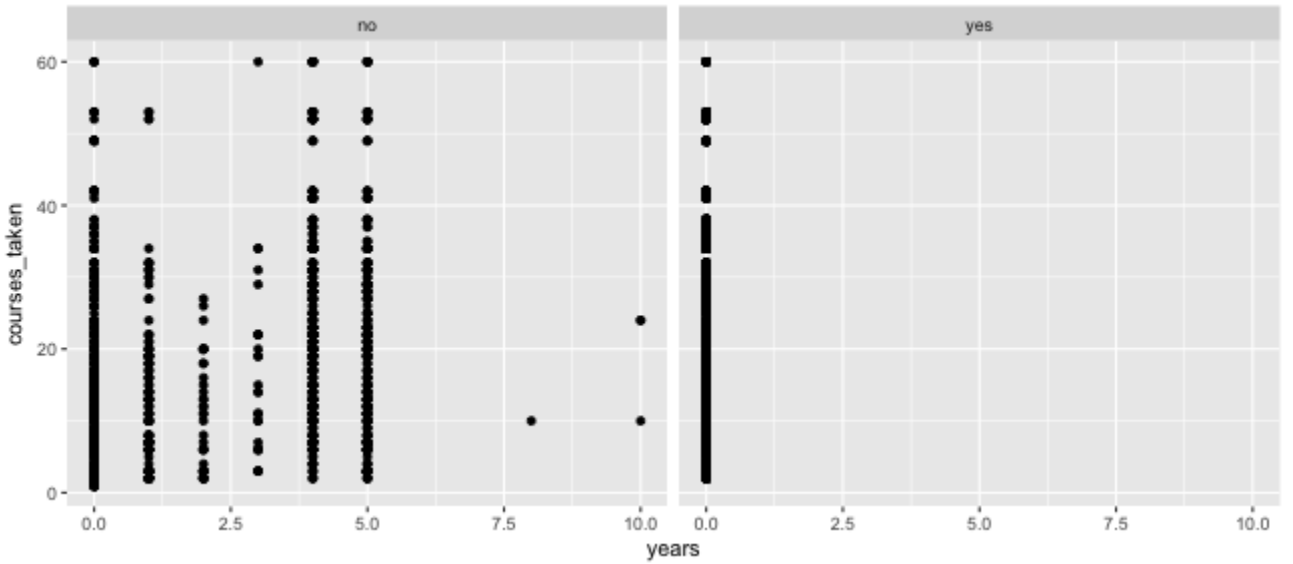
let's visualize the relatins between features to get a sense of stuff

```
D.viz <- select(D1, complete, years, entrance_test_score, courses_tak

pairs(D.viz)
```



Feature visualization (continued)

scatterplot is not very useful to see about catagorical or discrete variables as dependent varibale. So I want to seperate the features and into catagorical and continuous features



- it seems like the newly-enrolled students are more likely to complete a course

See if **Entrance_test_score** is any different between not-/ complete students

```
D.continuous <- select(D1, entrance_test_score, complete)
avg.score <- D.continuous %>% group_by(complete) %>% summarise(avg =
arrange(avg)
avg.score
```

```
## # A tibble: 2 x 2
##   complete   avg
##   <fct>    <dbl>
## 1 yes       11.0
## 2 no        12.4
```

- no complete students avg score = 12.4 while complete student is 11.0. Entrance_test_score may not be a valid feature

Use **prob.table** to see if **international student** is an influential feature

```
D.factoral <- select(D1, international, online, gender)
print("internatonal")
```

```
## [1] "internatonal"
```

```
prop.table(table(D1$international, D1$complete))
```

```
##
##                no       yes
##   no  0.2535404 0.5990445
##   yes 0.0406074 0.1068077
```

see about **gender** feature

```
print("gender")
```

```
## [1] "gender"
```

```
prop.table(table(D1$gender, D1$complete))
```

```
##
##                no          yes
##   1 0.1337655690 0.3347551612
##   2 0.1455383040 0.3330489678
##   3 0.0008530967 0.0017061935
##   4 0.0066541546 0.0134789285
##   5 0.0073366320 0.0228629927
```

see if how are students only taking online courses doing

```
print("take online course only")
```

```
## [1] "take online course only"
```

```
prop.table(table(D1$online, D1$complete))
```

```
##
##                no        yes
##    no  0.22931240 0.53472104
##    yes 0.06483535 0.17113121
```

some observations of categorical variables

- non-international students are more likely to complete
- gender coded as 1 & 2 have better odds to complete
- those who don't only take online class have better odds to complete

Why am I doing this so far?

- There may not be a standard way to preprocess the data, but I like to see how each feature works with the target variable just to get a sense
- For tree models, things get pretty overwhelming to read after features are thrown into a model, so it is useful to see about which could be an important feature at this point
- Yet, as I was plotting individual features against the target variable, this does not exclude issues such collinearity or the influence of outliers, etc.

Use **CART** tree to train & test a model

```
library(caret)

train2 <- train[,c(2:10)] #Remove the student_id variable that we do

# I also removed the enroll_data_time since it is just time stamp and
train3 <- train2[,c('years', 'entrance_test_score', 'courses_taken',

train3$gender <- as.factor(train3$gender)

#Define the control elements we would like to use
ctrl <- trainControl(method = "repeatedcv", #Tell caret to perform 10
                repeats = 3, #Tell caret to repeat each fold three t
                classProbs = TRUE, #Calculate class probabilities for
                summaryFunction = twoClassSummary)

#Define the model
cartFit <- train(complete ~ ., #Define which variable to predict
                data = train3, #Define the data set to train the mode
                trControl = ctrl, #Tell caret the control elements
                method = "rpart", #Define the model type
                metric = "ROC", #Tell caret to calculate the ROC curv
                preProc = c("center", "scale")) #Center and scale the
```

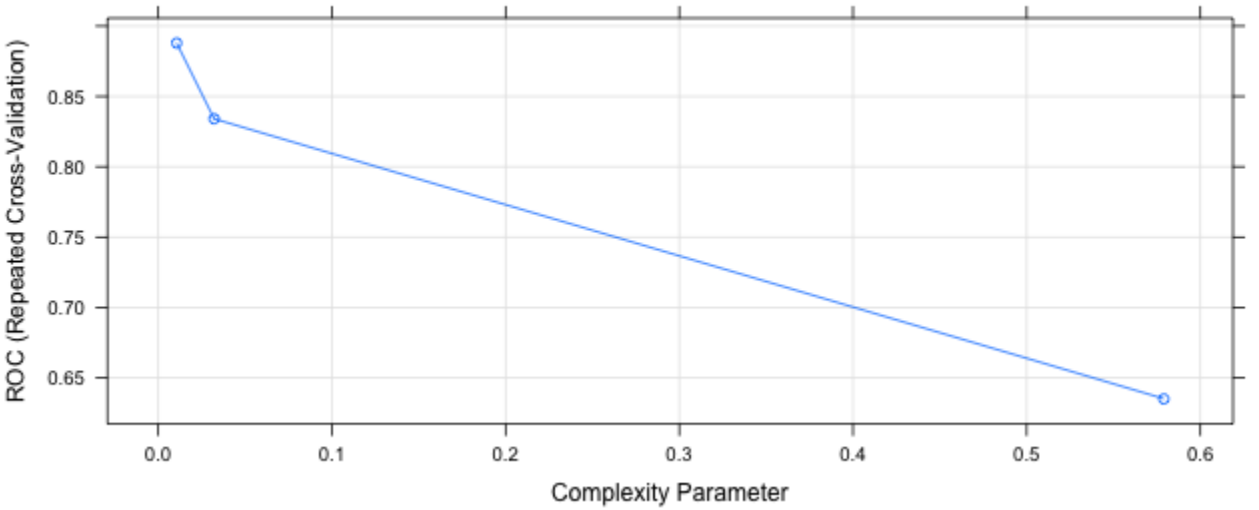
Check the performance of **CART** tree trained model

```
cartFit
```

```
## CART
##
## 4284 samples
##    7 predictor
##    2 classes: 'no', 'yes'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 3855, 3855, 3855, 3855, 3856, 3856, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens       Spec
##   0.01058632  0.8878884  0.6585321  0.9950920
##   0.03216612  0.8340590  0.6113199  0.9969467
##   0.57899023  0.6349727  0.2699454  1.0000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01058632.
```

Plot ROC against complexity

```
plot(cartFit)
```



About **Sensitivity** and **Specificity**

- Sensitivity(true positive): is the true positive rate also called the recall. It is the number instances from the positive (first) class that actually predicted correctly.
- Specificity(true negative): is also called the true negative rate. Is the number of instances from the negative class (second) class that were actually predicted correctly.

See **CART** model performance on test data

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no   805   15
##        yes  423 3041
##
##                Accuracy : 0.8978
##                  95% CI : (0.8883, 0.9067)
##     No Information Rate : 0.7134
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7224
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6555
##             Specificity : 0.9951
##          Pos Pred Value : 0.9817
##          Neg Pred Value : 0.8779
##              Prevalence : 0.2866
##          Detection Rate : 0.1879
##    Detection Prevalence : 0.1914
##       Balanced Accuracy : 0.8253
```

CART tree model performance on new data needs improvement

- The overall accuracy is .90 but the true positive(sensitivity) that represents successful prediction of "no complete" student is not very good(.66). This current tree model is better at predicitng the "complete" students (.995) but not so much so at predicting "no complete" students.We should try some other models to improve this predictive performance.

Use C5.0 tree to predict dropout students

- C5.0 is a improved version of C4.5 algorithm by Kuhn and Johnson (2013)
- C5.0 has some other hyperparameters that we can set in the tree model compared to C4.5. Things like boosting method and the selection of rules and trees methods.
- While boosting, the tree combines some poor performance features and supply into the whole trees model and iterate to improve the overall perfomance. The rules learning, compared to tree, take advantage of reusing features in the model to split instances till it reaches an optimal point for ROC value.

Fit the tree model by using C5.0

```
library(C50)
c50Fit <- train(complete ~ ., #Define which variable to predict
                data = train3, #Define the data set to train the mode
                trControl = ctrl, #Tell caret the control elements
                method = "C5.0", #Define the model type
                metric = "ROC", #Tell caret to calculate the ROC curv
                preProc = c("center", "scale")) #Center and scale the
```

Check out trained model performance

```
c50Fit
```

```
## C5.0
##
## 4284 samples
##    7 predictor
##    2 classes: 'no', 'yes'
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 3856, 3855, 3856, 3855, 3855, 3856, ...
## Resampling results across tuning parameters:
##
##   model  winnow  trials  ROC        Sens       Spec
##   rules  FALSE    1      0.8313059  0.6658048  0.9939994
##   rules  FALSE   10      0.9223472  0.6742303  0.9908390
##   rules  FALSE   20      0.9250284  0.6826314  0.9893060
##   rules   TRUE    1      0.8315485  0.6671598  0.9933458
##   rules   TRUE   10      0.9222596  0.6774779  0.9905111
##   rules   TRUE   20      0.9261630  0.6848039  0.9897457
##   tree   FALSE    1      0.9019237  0.6869741  0.9913815
##   tree   FALSE   10      0.9235035  0.6774890  0.9900711
```

See about the model performance on the test data

```
c50Classes <- predict(c50Fit, test3)
confusionMatrix(data = c50Classes, test3$complete)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no   860   44
##        yes  368 3012
##
##                Accuracy : 0.9038
##                  95% CI : (0.8946, 0.9125)
##     No Information Rate : 0.7134
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7447
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7003
##             Specificity : 0.9856
##          Pos Pred Value : 0.9513
##          Neg Pred Value : 0.8911
```

Use **RandomForest** to train & predict

- Another useful and commonly used algorithm is `RandomForest` (RF)
- A brief intro. about RF can be found here: what's RandomForest

Fit and predict student dropouts with RF

```
library(randomForest)
set.seed(1234)
rfFit <- randomForest(complete ~ .,train3)
rfClasses <- predict(rfFit, test3)
confusionMatrix(data=rfClasses, test3$complete)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no   820    0
##        yes  408 3056
##
##                Accuracy : 0.9048
##                  95% CI : (0.8956, 0.9134)
##     No Information Rate : 0.7134
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7414
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6678
```

see feature importance w/ randomForest

- With C5.0, we can achieve a similar purpose by checking the percentage(%) that a feature is used
- (BE CAREFUL!!) If you are using `rules` method in C5.0, please mind that the order of feature percentage does not make the same sense as when we are using `trees` method. (Because features can be reused)
- According to mean decrease Gini, 'years', 'course_id'(which courses), 'course_taken'(# of course taken) are the 3 most important features. This coincides with top 3 attributes usage given by C5.0 tree.

```
varImpPlot(rfFit,type=2)
```

Wait, what are our research questions again ???

1. Which tree model works the best in this current case to predict student dropouts?

2. How to increase tree model performance to catch more true dropouts(true posive)?

3. What are the important features about students that could lead to dropouts? *(though this can also be done w/ logistic regresion)

Findings & Summary

- In our case, the C5.0 tree works the best as it could catch about 70% of the student dropouts
- There are other methods in C5.0 that we can tune with, such as boosting or using both *rules* and *trees* to try to get a better performance. I also tried another 2 tree models: CART and randomForest, but all of these attempts do not result in better model performance.
- Years of program study, (which) courses , number of courses students have taken seem to be important features to predict students' course dropouts in this case.
- If we put the these results into practical consideration, this could mean that there may be a few courses that which are more challenging to students (?). School administrators could take a look at the nature, level, or the course requirements, etc of those courses.

Limitations & Suggestions

No data analytics is perfect, I came up with a few thoughts and make some suggestions in the followings:

- Regardless of which tree model, the model perfomance in this project is not close to be perfect
- Perfection is a relative idea and it depends on the area of research. Although our models are not near perfect in predicting new data, we also have to be careful not to overfit it. Sommetimes the reason could be that the data is simply too noisy
- In general, decision trees are resilient to imbalanced data and can handle both categorical and continuous features at the same time pretty well. But each tree algorithm has its pros and cons, when to use what depends on the understanding of the data and trial & error

Limitations & Suggestions - (continued)

- For student data like this, we can try to aggregate different years of data and continue to tinker our tree model. Arguably, data from several years should be more robust than that from one year of student data.

- In practice, we can also create a **`cost matrix`** to give different weights to false postive/negative (for non-dropouts we predict dropouts/for the dropouts we predict non-dropouts). A cost matrix can be based on some school resources consideration. For example, if the cost for missing a course dropout (false negative) is 3 times the financial burden than catching a dropout when the student is not (false postive), cost matrix will come in pretty handy.

- We shoud also be careful of the data representativeness. Although there are > 5,000 rows of observations, the data is generated by only 682 students and that may not be a very representitive student body to conclude which student features will lead to dropouts