Ethics|Final Project|Part I

Aldo Adriazola, Avisek Choudhury, Kait Arnold

East Section

Contents

Introduction 1
Data Import and Cleaning 1
Model Building 3
Summary & Conclusion 5

Introduction

In 2012, Minerva High School in Pittsburgh, Pennsylvania was struggling with its high drop out rates at 9% and low (on-time) graduation rates of 55%. The school's principal and board will respond to the problem with a data-driven solution aimed at getting their students back on track and gaining insight into the why behind the high dropout rates. This solution will use machine learning to identify predictors of student disengagement as a proxy for potential drop out. Flagging at risk students will enable the teachers, counselors, administrators to respond accordingly with new interventions, meetings with school counselors and/or parents, new incentive structures, etc in the hopes of reversing the high dropout rates.

Data Import and Cleaning

Upon import of the data, we converted all string columns into factors utilizing R's stringAsFactors argument in the read.csv function. We then removed the student ID field because it will not be used in this analysis. We converted all factors into numeric and converted the dropped field back into a factor for use as the response variable in our modeling efforts.

In an effort to normalize our data. we rescaled our resulting data using the following logic:

`z = (x-min(x)) / (max(x) - min(x))`

```r
#Load the libraries
library(dplyr)
library(readr)
library(tidyverse)
library(class)
library(caret)
library(rpart)
library(partykit)
library(randomForest)
library(e1071)
#Read the csv file
case3data <- read.csv("case3data.csv",stringsAsFactors = TRUE)
# remove studentID and move dropped to the first column
case3data <- case3data %>%
    select(-studentID) %>% select(dropped,everything())
```

```
# convert all factors to numeric
case3data <-case3data %>% mutate_if(is.factor,as.numeric)
# convert dropped to a factor
case3data$dropped <- as.factor(case3data$dropped)
#Print the summary
#case3data %>% summary
```

Below, we rescale the data using the normalization method described above and we create the training and testing data as 80/20.

```
# First, rescale the data
# create the rescaling function we have been using thus far
rescale_x <- function(x){(x-min(x))/(max(x)-min(x))}
# create a copy of the df
rescaled_df <- case3data
# apply the rescale function to all columns except dropped
rescaled_df[2:14] <- sapply(rescaled_df[2:14],rescale_x)
# confirm rescaling worked correctly
# all rescaled vars should be within [0,1]
summary(rescaled_df)
```

```
##  dropped       grade             year              zip            ethnicity
##  0:16837   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1: 1048   1st Qu.:0.0000   1st Qu.:0.2000   1st Qu.:0.1613   1st Qu.:0.0000
##            Median :0.3333   Median :0.6000   Median :0.1613   Median :0.0000
##            Mean   :0.5013   Mean   :0.5123   Mean   :0.3589   Mean   :0.2584
##            3rd Qu.:1.0000   3rd Qu.:0.8000   3rd Qu.:0.7419   3rd Qu.:0.7500
##            Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##       sex              gpa         subsidizedLunches employmentHours
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.5020   1st Qu.:0.0000    1st Qu.:0.0000
##  Median :0.0000   Median :0.6160   Median :0.5000    Median :0.0000
##  Mean   :0.4711   Mean   :0.6023   Mean   :0.3905    Mean   :0.1309
##  3rd Qu.:1.0000   3rd Qu.:0.7100   3rd Qu.:0.5000    3rd Qu.:0.2500
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000    Max.   :1.0000
##  hrsWifiPerWeek     sanctions      librarySwipesPerWeek   apClasses
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000       Min.   :0.00000
##  1st Qu.:0.1818   1st Qu.:0.5000   1st Qu.:0.1034       1st Qu.:0.00000
##  Median :0.3182   Median :0.5000   Median :0.1379       Median :0.00000
##  Mean   :0.3391   Mean   :0.4263   Mean   :0.1765       Mean   :0.07556
##  3rd Qu.:0.4545   3rd Qu.:0.5000   3rd Qu.:0.2414       3rd Qu.:0.14286
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000       Max.   :1.00000
##  athleticSeasons
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.2000
##  Mean   :0.2137
##  3rd Qu.:0.4000
##  Max.   :1.0000
```

```
# Now split the data
# set the seed to Notre Dame's founding year
set.seed(1842)
# determine the number of rows in the dataframe
n <- nrow(rescaled_df)
```

```r
# get a list of 20% of the rows in combined to use as indices
test_idx <- sample.int(n, size = round(0.2 * n))
# set the the training data to be those rows not matching the index list
training <- rescaled_df[-test_idx,]
# set the the test data to be those rows matching the index list
testing <- rescaled_df[test_idx,]
```

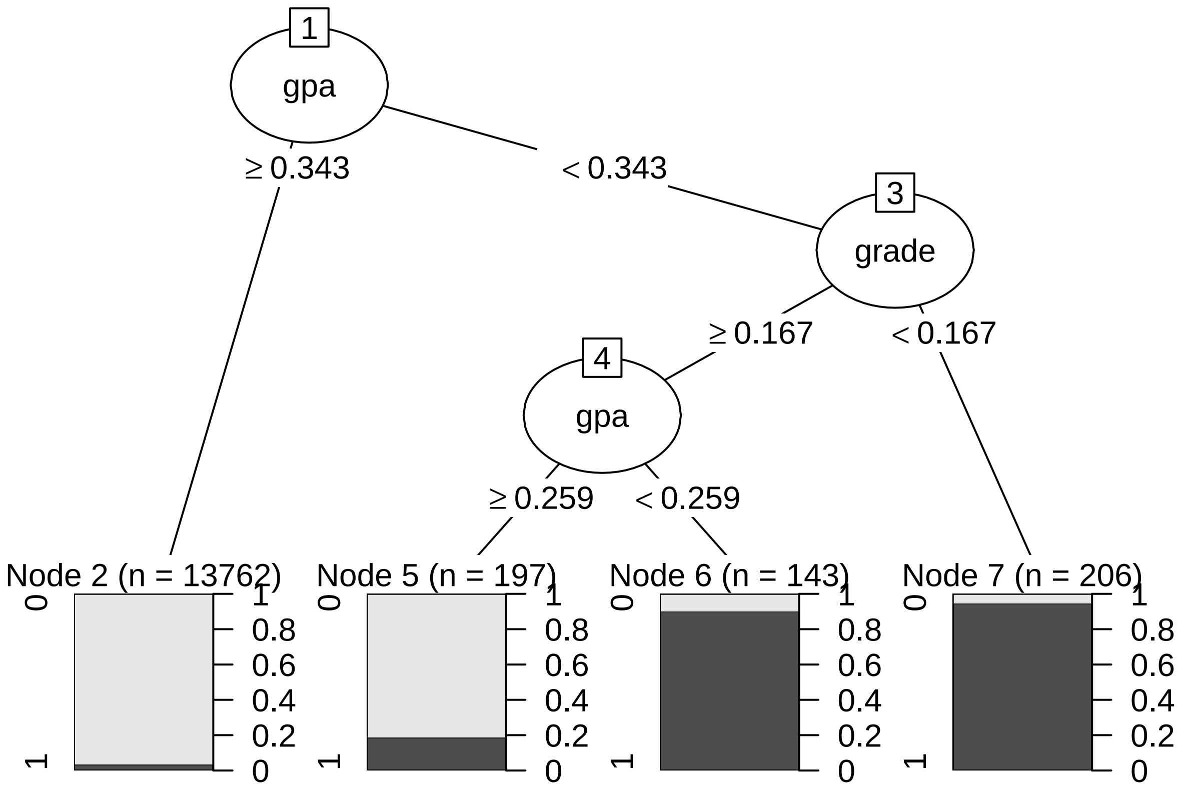
Model Building

Our response variable `dropped` is a categorical variable. We have tried the Logistic Regression, Decision Tree, Random Forest and K-Nearest Neighbor to build the model and predict the response variable. Although all the approach yielded excellent accuracy, we ended up choosing Decision Tree to build our final model.

Decision Tree

The Decision Tree algorithm performed very well and is easy to visualize and understand. It algorithm belongs to the family of supervised learning algorithms; it can be used for solving both regression and classification problems. The general motivation for using a decision tree is to predict the class or value of target variables by learning decision rules inferred from prior data (the training data). The decision tree algorithm tries to solve the problem by using a tree representation. Each internal node of the tree corresponds to an attribute and each leaf node corresponds to a class label. In R there are several packages available to run the decision tree algorithm. Here, we'll use `rpart` package and also `caret` package to implement the decision tree with cross-validation.

Next, we might need to prune the decision tree in an effort to reduce the size of the tree by removing sections of the tree that provide little predictive power. This will reduce complexity and over fitting of the final model leading to better interpretability for our school board and principle Vulcani.



The tree looks simple and doesn't need pruning. Now let's use this Decision Tree to predict the response.

```
# use the decision tree created above to predict values in the test data
# and then store the results
testing$tree_predict <- predict(diag.tree,
                                newdata=testing,
                                type="class")
# create the confusion matrix using the table function
confusion_tree <- table(Predicted =testing$tree_predict,
                        Actual=testing$dropped)
# Print a legend
cat("0 = Not Dropped Out, 1 = Dropped Out\n\n")
```

```
## 0 = Not Dropped Out, 1 = Dropped Out
```

```
# Print the confusion matrix
confusion_tree
```

```
##          Actual
## Predicted    0    1
##         0 3365  125
##         1    4   83
```

```
# show the accuracy of the decision tree
cat("Overall accuracy of prediction:\t",
    sum(diag(confusion_tree)/nrow(testing)) %>%
      round(4),"\n")
```

```
## Overall accuracy of prediction:   0.9639
```

```r
# show the percentage of M misclassified as B
cat("Rate of misclassifying Not Dropped Out as Dropped Out:\t",
    (confusion_tree[1,2] /
       (confusion_tree[1,1] + confusion_tree[1,2])) %>%
      round(4),"\n")
```

```
## Rate of misclassifying Not Dropped Out as Dropped Out:    0.0358
```

```r
# show the percentage of B misclassified as M
cat("Rate of misclassifying Dropped Out as Not Dropped Out:\t",
    (confusion_tree[2,1] /
       (confusion_tree[2,1] + confusion_tree[2,2])) %>%
      round(4),"\n")
```

```
## Rate of misclassifying Dropped Out as Not Dropped Out:    0.046
```

Summary & Conclusion

Decision trees are easy for people to understand, because the visualization is relatively simple. Our decision tree proved to be quite accurate, yielding the correct response 96.39% of the time. Other models that were considered included:

- Logistic Regression with 95.4% accuracy
- Random Forest with 96.95% accuracy
- KNN with 94.72% accuracy

The decision tree proved to be the second most accurate method tested. It is far easier to comprehend than the most accurate method - the random forest - and is only slightly less accurate. The decision tree was able to predict the output using only used two variables: GPA and grade. This shows us that students' performance at each stage in their high school career is a key predictor in determining who might drop out. In the school's action items from this model, the primary focus would be to improve students' GPA performance.

Even though the random forest had the highest accuracy, we will use the decision tree as our final model for better interpretability. In comparison to Hephaestats' final model with eight key indicators, our final accuracy used fewer variables and was over four percentage points higher than their model. We were not given a list of the variables that Hephaestats used, and the specific modeling efforts used by Hepaestats were not revealed in the case study. Following the principle of Occam's razor, we chose to use a less complex model, especially since it yielded better results.