First Year & First Year Transfer Enrollment Predictive Analytic Project

Meaghan Wetherell and Olabode Makinde

March 20, 2017

Overview

This project was conducted in 2017, with two specific research goals in mind:

1. Find out which factors are the most important influencers on a student's decision to enroll
2. Design a model capable of predicting enrollment with >90% accuracy.

Unlike studies on retention, persistence, and graduation, there are very few predictive model articles on higher education student enrollment. Those that do exist are mostly prior to the 2000's, and are very out of date with regards to new statistical methodology. There are many powerpoints produced by enrollment departments but these modesl are not statistical models, but business models: they attempt to tell when a university will know its enrollment numbers for the next year, but rarely attempt to identify which applicants will enroll.

Given the limited literature, I moved ahead with a new approach: use random forest modelling with application-relevant data to predict students enrollment behavior on an individual level. This method is similar to that used by Nandeshwar and Chaudhair (2009), who used decision trees with self-reported demographics variables to predict enrollment, with an average accuracy of 83-84%. They found that financial aid was the most important variable, and that students given aid would enroll regardless of their High School GPA or ACT/SAT scores.

Random Forest models are essentially aggregations and averages of thousands of decision trees (Liaw and Weiner 2002). They are therefore far more robust to errors than decision trees. One of the problems with both decision trees and random forest models is that they have higher average accuracy when groups are highly imbalanced: that is, if 85% of people enrolled, both models will be very good at predicting who enrolled, but not neccessarily which students did not enroll. A model run on such an uneven dataset could get 85% accuracy by predicting that 100% of the students enrolled, so it's important to look at the classifications and see which groups are being underestimated. The following document describes the methodology I used in 2017 to predict enrollment at the university, including random forest models, confusion matrices (to see who's classified as what), and a number of graphics and subsequent tests to evaluate the differences between demographic groupings. This methodology has been applied to retention studies and outperforms logistic regression and other machine learning methods (Delen 2010), but to my knowledge it has not yet been applied to enrollment prediction.

A warning to others who attempt to do this study: there were several setbacks in the aquisition of data, because current data was overwriting original data. For example, students in *financialaidaward_rpt* who have an offered_balance of 0 may have been offered money, they just chose not to come and so it was 0'd out. This happened with demographics as well, which would read as "Not reported" for students who didn't come, and then once a student enrolled and edited their ethnicity/race, it would update. Make sure that the data you're using dates to the time of application - and double check the warehouse SQL code, because *applicationadmission_rpt* was supposed to date to the time of admission and did not for all of the demographics data.

References

Delan, D. 2010. A comparative analysis of machine learning techniques for student retention management. Decision Support Systems 49:498-506.

Liaw and Weiner 2002. Classification and Regression by randomForest. R News Vol. 2/3.

McPherson et al 1994. Predicting Higher Education Enrollment in the United States: An Evaluation of Different Modeling Approaches. Discussion paper No. 26, Williams Project on the Economics of Higher Education.

Nandeshwar and Chaudhair, 2009. Enrollment Prediction Models Using Data Mining.

Analyses

Packages

In R, different groups of analyses and functions are created as packages, which are free to download and install. To begin with, you'll need to remind R that it should look for function names in certain packages (otherwise, R will not understand what you're asking it to do). If these packages are not installed, use the `install.packages()` function to create them.

```
library(dplyr)
library(randomForest)
library(sjPlot)
library(e1071)
library(caret)
library(reshape2)
```

Upload Data

Now, we will upload and rename our data files as follows.

1. Data on student admission and enrollment (**data**) from *application_admission_rpt*
2. Financial aid data (**faid**) pulled from *ps_stdnt_awrd_actv_v*
3. A document detailing the grant and loan names (**rosetta**) pulled from *financialaidaward_rpt*
4. Birthdate information (**dob**) from *person_details_v*
5. Distance between the university and student hometown (**distance**) calculated using a seperate R Markdown file. Because we may recruit students from different areas in the

future, it should be re-run to get all possible distance options before proceeding in the future.

```
data <- read.csv("N:/Institutional Effectiveness/Student 
Enrollment/data14.csv")
faid <- read.csv("N:/Institutional Effectiveness/Student 
Enrollment/financialaid.csv")
rosetta <- read.csv("N:/Institutional Effectiveness/Student 
Enrollment/financialaidRosetta.csv")
dob <- read.csv("N:/Institutional Effectiveness/Student Enrollment/dob.csv")
distances <- read.csv("N:/Institutional Effectiveness/Student 
Enrollment/distances.csv")
```

Prune and Merge financial data.

This data is from *ps_stdnt_awrd_actv_v* without much editing, and represents the whole financial aid history rather than the snapshot offered in *financialaidawdrpt*. I want the amount of money they were offered after their application, but before their current enrolled term. I'm going to create a file containing current admit terms and turn that into a date, then take all the `ACTION_DTTM` data that is before that date, that way I have an accurate picture for each person of their offer package, for each unique time they enrolled.

```
moneydate <- data %>%
  filter(CURRENT_ADMIT_TYPE_CODE %in% c("FYR", "FYT")) %>%
  mutate(AppDate = ifelse(grepl("Fall", 
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), paste(CURRENT_ADMIT_TERM_YEAR_NAME, 
"0901", sep=""), ifelse(grepl("Winter", 
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), paste(CURRENT_ADMIT_TERM_YEAR_NAME, 
"0101", sep=""), ifelse(grepl("Spring", 
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), paste(CURRENT_ADMIT_TERM_YEAR_NAME, 
"0301", sep=""), paste(CURRENT_ADMIT_TERM_YEAR_NAME, "0601", sep=""))))) %>%
  mutate(AppDate = as.Date(AppDate, "%Y%m%d")) %>%
  dplyr::select(APPLICANT_ID, AppDate)

faid$ActualYear <- faid$AID_YEAR - 1
data$ActualYear <- ifelse(grepl("Spring|Winter|Summer", 
data$CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), 
data$CURRENT_ADMIT_TERM_YEAR_NAME - 1, data$CURRENT_ADMIT_TERM_YEAR_NAME)

faid[faid=="."] <- "0"

faid1 <- merge(faid, rosetta, by.x="ITEM_TYPE", 
by.y="FINANCIAL_AID_ITEM_TYPE_CODE", all.x=TRUE, sort=FALSE)
faid1 <- merge(faid1, moneydate,  by.x="EMPLID", by.y="APPLICANT_ID", 
all.x=TRUE, sort=FALSE)

#This ensures you've correctly merged to duplicate, turn on if needed to 
check
#test <- faid1 %>% group_by(EMPLID) %>% mutate(howmany = 
length(unique(AppDate))) %>% filter(howmany>1)
```

```r
faid1 <- faid1 %>%
  mutate(ACTION_DTTM = as.Date(ACTION_DTTM, "%Y/%m/%d")) %>%
  mutate(STUDENT_ID = EMPLID) %>%
  arrange(EMPLID, ACTION_DTTM) %>%
  dplyr::select(STUDENT_ID, ACTION_DTTM, OFFER_AMOUNT, AID_YEAR, ActualYear,
FINANCIAL_AID_TYPE_NAME, FINANCIAL_AID_SOURCE_DESCRIPTION, AppDate)

faid2 <- faid1 %>%
  group_by(STUDENT_ID) %>%
  filter(ACTION_DTTM < AppDate)

 faid3 <- dcast(faid2,
STUDENT_ID+ActualYear~FINANCIAL_AID_SOURCE_DESCRIPTION+FINANCIAL_AID_TYPE_NAM
E,  value.var="OFFER_AMOUNT", fun.aggregate=sum, na.rm=TRUE)

 faid4 <- faid3 %>%
   mutate(LoansOffered = Federal_Loan+Private_Loan) %>%
   mutate(OtherMoneyOffered = Federal_Grant + Institutional_Grant +
Institutional_Scholarship + Institutional_Waiver + Other_Grant  +
Private_Grant + Private_Scholarship + State_Grant + State_Scholarship) %>%
   mutate(TotalMoneyOffered = Federal_Loan+Private_Loan + Federal_Grant +
Institutional_Grant + Institutional_Scholarship + Institutional_Waiver +
Other_Grant +  Private_Grant + Private_Scholarship + State_Grant +
State_Scholarship) %>%
   dplyr::select(STUDENT_ID, ActualYear, LoansOffered, OtherMoneyOffered,
TotalMoneyOffered)

data <- merge(data, faid4, by.x=c("APPLICANT_ID", "ActualYear"),
by.y=c("STUDENT_ID", "ActualYear"), all.x=TRUE, all.y=FALSE)
data[is.na(data)] <- 0
```

Edit & Merge Data

As exported from WebFocus, much of this dataset does join in quite the manner that we want. For example, Aid Year and Academic Year are not always the same thing. To facillitate an accurate merge, we need to do some maintance. When merging datasets, sometimes you will be merging columns that have different names. In this case, use by.x and by.y to specify the columns in the first and second dataframe that will be used for merging. In this case, I wanted a left-join: I wanted to retain all of the **data** information but there was extra in **dob** that I did not need (students that weren't FYR or FYT). If you do not specific all.x=TRUE, then you will get an inner join and retain the information in **dob** that does not have a match.

```r
distances$address <- paste(distances$City, distances$State, sep=", ")
data$address <- paste(data$ORIGIN_CITY, data$ORIGIN_STATE, sep=", ")

data <- merge(data, dob, by.x=c("APPLICANT_ID"), by.y=c("PERSONID"),
all.x=TRUE)
data <- merge(data, distances, by="address", all.x=TRUE)
```

Alter Data and Create New Fields

Because we are going to be conducting a random forest analysis on this data, we need to some data reorganization. Specifically, we need to remove NAs, periods, and blanks from our data - random forest analyses don't like those things. For that, I've used the package `dplyr` and the command `mutate`. You'll notice that there are several examples of what are called "nested ifelse" commands. The best way to read these are "ifelse this is true, put this, if not then put this." In the first line of the code, I'm changing all of the blanks in the Waiver field to be 0. This line reads "if Waiver is an NA, put 0; if waiver is either blank or a period, put 0; otherwise, put Waiver."

Dates are not easy for R to automatically read, and so I must also remind R using the `as.Date` command that certain fields are dates, not weird number strings. Similarly, R doesn't always read numbers in correctly, and so I've changed the ENROLLED_COUNT field into a factor (with two levels, 0 or 1), as well as the new SEASON field and the Specialty field.

By turning both the applied date and the date of birth into dates, I can also subtract the two, divde by the number of days in a year, remind R that it's numeric, and create a new field: AGE2, the age at application.

Some fields were modified then not used in the final study; you could try adding them later.

```
data <- data %>%
  mutate(year = CURRENT_ADMIT_TERM_YEAR_NAME) %>%
  mutate(CGPA = as.numeric(ifelse(LAST_EXTERNAL_ORGANIZATION_GPA == ".",
paste(HIGH_SCHOOL_GPA), paste(LAST_EXTERNAL_ORGANIZATION_GPA)))) %>%
  mutate(APPLIED_DATE = as.Date(APPLIED_DATE, "%Y/%m/%d")) %>%
  mutate(DATE_OF_BIRTH = as.Date(DATE_OF_BIRTH, "%Y/%m/%d")) %>%
  mutate(ACADEMIC_INTEREST_DESCRIPTION_COMBO =
paste(ACADEMIC_INTEREST_DESCRIPTION1, ACADEMIC_INTEREST_DESCRIPTION2,
ACADEMIC_INTEREST_DESCRIPTION3, sep=", ")) %>%
  mutate(Specialty =
as.factor(ifelse(grepl("Education|Teach|Geology|Wine|Craft|Music|Composition|
Theatre|Performance|Paramedic|Info Tech|Information
Technology|Business|Marketing|Administrator|Human Resource Management|Supply
Chain|Accounting|Aviation|Pilot", ACADEMIC_INTEREST_DESCRIPTION_COMBO),
"Yes","No"))) %>%
  mutate(SEASON = as.factor(ifelse(grepl("Fall",
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), "Fall", ifelse(grepl("Winter",
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), "Winter", ifelse(grepl("Summer",
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), "Summer", ifelse(grepl("Spring",
CURRENT_ADMIT_TERM_ACADEMIC_YEAR_NAME), "Spring", "Unknown"))))) ) %>%
  mutate(address = paste(ORIGIN_CITY, ORIGIN_STATE, sep=", ")) %>%
  mutate(ENROLLED_COUNT = factor(ENROLLED_COUNT), CURRENT_ADMIT_TYPE_CODE =
factor(CURRENT_ADMIT_TYPE_CODE), ETHNICITY_RACE = factor(ETHNICITY_RACE), EWO
= factor(EWO)) %>%
  mutate(AGE2 = as.numeric((APPLIED_DATE - DATE_OF_BIRTH)/365))

## Warning in eval(substitute(expr), envir, enclos): NAs introduced by
## coercion
```

Don't worry about that NA warning - you'll deal with it in a second.

Filter Data

Now that my data is cleaned and modified, I can proceed to filter it. In this study, I'm only looking at first years and first year transfers from 2010-2016, and I want them to have been admitted - but I want to study them seperately. **data2** will contain only first year freshmen, while **data3** will contain all of them. They need to have a GPA and a reported Age, as these are important characteristics, so I use a series of filters to remove individuals with these missing fields. You can, if needed, also use different code to fill in blank data with best guesses (means, medians, etc), but our dataset was large enough I chose to simply omit missing data, not estimate.

I also add and edit different subfields here. In some cases, I wanted unknown Pell eligibility to be recorded as a third category *unknown* but in other analyses I wanted it to be recorded as *no*, so I edited it in this second dataset to make sure I wasn't overwriting the original import. It makes it easier to go back and get the original data by mutating and renaming it here as **data2**.

```
data2 <- data %>%
  filter(CURRENT_ADMIT_TYPE_CODE == "FYR") %>%
  filter(CURRENT_ADMIT_TERM_YEAR_NAME %in% c(2010, 2011, 2012, 2013, 2014,
2015, 2016))%>%
  filter(ADMITTED_COUNT == "1") %>%
   filter(WA_RESIDENCY_FLAG != "." | WA_RESIDENCY_FLAG != "") %>%
  filter(PERSON_SELF_REPORTED_DISABILITY_FLAG != "." |
PERSON_SELF_REPORTED_DISABILITY_FLAG != ".") %>%
  filter(!is.na(CGPA)) %>%
  filter(!is.na(AGE2)) %>%
  filter(CGPA != ".") %>%
  filter(CGPA != "0") %>%
  filter(DENIED_COUNT != "1")%>%
  filter(Miles != "NA") %>%
  mutate(ROYALL_APPLICATION_COUNT = ifelse(ROYALL_APPLICATION_COUNT == "1",
"Royall Application", "Not Royall Application")) %>%
  mutate(ROYALL_APPLICATION_COUNT =
as.factor(ifelse(is.na(ROYALL_APPLICATION_COUNT), "Not Royall Application",
paste(ROYALL_APPLICATION_COUNT)))) %>%
  mutate(Specialty =
as.factor(ifelse(grepl("Education|Teach|Geology|Wine|Craft|Music|Composition|
Theatre|Performance|Paramedic|Info Tech|Information
Technology|Business|Marketing|Administrator|Human Resource Management|Supply
Chain|Accounting|Aviation|Pilot", ACADEMIC_INTEREST_DESCRIPTION_COMBO),
"Yes","No"))) %>%
  mutate(duplicated = as.factor(ifelse(duplicated(APPLICANT_ID),
"Duplicated", "Not Duplicated")))
```

Filter For FYT

Same process, different dataset.

```
data3 <- data %>%
  filter(CURRENT_ADMIT_TYPE_CODE == "FYT") %>%
  filter(CURRENT_ADMIT_TERM_YEAR_NAME %in% c(2010, 2011, 2012, 2013, 2014,
2015, 2016))%>%
  filter(ADMITTED_COUNT == "1") %>%
   filter(WA_RESIDENCY_FLAG != "." | WA_RESIDENCY_FLAG != "") %>%
  filter(PERSON_SELF_REPORTED_DISABILITY_FLAG != "." |
PERSON_SELF_REPORTED_DISABILITY_FLAG != ".") %>%
  filter(!is.na(CGPA)) %>%
  filter(!is.na(AGE2)) %>%
  filter(CGPA != ".") %>%
  filter(CGPA != "0") %>%
  filter(DENIED_COUNT != "1")%>%
  filter(Miles != "NA") %>%
  mutate(ROYALL_APPLICATION_COUNT = ifelse(ROYALL_APPLICATION_COUNT == "1",
"Royall Application", "Not Royall Application")) %>%
  mutate(ROYALL_APPLICATION_COUNT =
as.factor(ifelse(is.na(ROYALL_APPLICATION_COUNT), "Not Royall Application",
paste(ROYALL_APPLICATION_COUNT)))) %>%
  mutate(Specialty =
as.factor(ifelse(grepl("Education|Teach|Geology|Wine|Craft|Music|Composition|
Theatre|Performance|Paramedic|Info Tech|Information
Technology|Business|Marketing|Administrator|Human Resource Management|Supply
Chain|Accounting|Aviation|Pilot", ACADEMIC_INTEREST_DESCRIPTION_COMBO),
"Yes","No"))) %>%
  mutate(duplicated = as.factor(ifelse(duplicated(APPLICANT_ID),
"Duplicated", "Not Duplicated")))
```

Set Up Data For Prediction

Before we proceed, it's important to also get your test data setup BEFORE running the model - else, the levels of the factors won't match, and random forest won't be able to run. In this case, we're going to look at data for 2017 and 2018.

```
PredictFYR <- data %>%
  filter(CURRENT_ADMIT_TYPE_CODE == "FYR") %>%
  filter(CURRENT_ADMIT_TERM_YEAR_NAME %in% c(2017,2018))%>%
  filter(ADMITTED_COUNT == "1") %>%
  filter(WA_RESIDENCY_FLAG != "." | WA_RESIDENCY_FLAG != "") %>%
  filter(PERSON_SELF_REPORTED_DISABILITY_FLAG != "." |
PERSON_SELF_REPORTED_DISABILITY_FLAG != ".") %>%
  filter(!is.na(CGPA)) %>%
  filter(!is.na(AGE2)) %>%
  filter(CGPA != ".") %>%
  filter(CGPA != "0") %>%
  filter(DENIED_COUNT != "1")%>%
  filter(Miles != "NA") %>%
  mutate(ROYALL_APPLICATION_COUNT = ifelse(ROYALL_APPLICATION_COUNT == "1",
"Royall Application", "Not Royall Application")) %>%
  mutate(ROYALL_APPLICATION_COUNT =
```

```
as.factor(ifelse(is.na(ROYALL_APPLICATION_COUNT), "Not Royall Application", 
paste(ROYALL_APPLICATION_COUNT)))) %>%
  mutate(Specialty = 
as.factor(ifelse(grepl("Education|Teach|Geology|Wine|Craft|Music|Composition|
Theatre|Performance|Paramedic|Info Tech|Information 
Technology|Business|Marketing|Administrator|Human Resource Management|Supply 
Chain|Accounting|Aviation|Pilot", ACADEMIC_INTEREST_DESCRIPTION_COMBO), 
"Yes","No"))) %>%
  mutate(duplicated = as.factor(ifelse(duplicated(APPLICANT_ID), 
"Duplicated", "Not Duplicated")))

levels(PredictFYR$CAMPUS_CODE) <- levels(data2$CAMPUS_CODE)
levels(PredictFYR$ENROLLED_COUNT ) <- levels(data2$ENROLLED_COUNT )
levels(PredictFYR$PERSON_GENDER) <- levels(data2$PERSON_GENDER)
levels(PredictFYR$FIRST_GENERATION_FLAG) <- 
levels(data2$FIRST_GENERATION_FLAG)
levels(PredictFYR$WA_RESIDENCY_FLAG) <- levels(data2$WA_RESIDENCY_FLAG)
levels(PredictFYR$SEASON) <- levels(data2$SEASON)
levels(PredictFYR$PERSON_VETERAN_FLAG) <- levels(data2$PERSON_VETERAN_FLAG)
levels(PredictFYR$PERSON_SELF_REPORTED_DISABILITY_FLAG) <- 
levels(data2$PERSON_SELF_REPORTED_DISABILITY_FLAG)
levels(PredictFYR$ETHNICITY_RACE) <- levels(data2$ETHNICITY_RACE)
```

Create Training and Test Datasets

```
trainFYR <- data2%>% filter(ActualYear %in% c(2010, 2011, 2012, 2013, 2014))
testFYR <- data2 %>% filter(ActualYear %in% c(2015))
trainFYT <- data3 %>% filter(ActualYear %in% c(2010, 2011, 2012, 2013, 2014))
testFYT <- data3 %>% filter(ActualYear %in% c(2015))
```

Random Forest Analysis

Using the package `randomForest`, I created a random forest model using our dataset to predict and classify student enrollment. I have not edited any of the controls on this analysis, and since it is bootstrapped it can take several minutes to run. If you are in a time crunch, include the following code:`ntree=100, mtry=2, do.trace=100`

This will decrease your accuracy a little, but it will speed up your process a lot. For the following, **rf.modF** will be the random forest model for first year freshmen, and **rf.modT** is for first year transfer students. It is possible to run them together using their current admit type code as a variable, but I was specifically asked to run them seperately.

Right now, it looks like a lot of these flags don't correspond to the original application-listed ethnicity/etc. So that's a real problem. I'm storing this working model here, and then messing with it.

rf.modF = randomForest(ENROLLED_COUNT ~ CAMPUS_CODE+ PERSON_GENDER + FIRST_GENERATION_FLAG + WA_RESIDENCY_FLAG+ AGE2+ SEASON + Miles+ CGPA + PERSON_VETERAN_FLAG + PERSON_SELF_REPORTED_DISABILITY_FLAG + ETHNICITY_RACE + LoansOffered + OtherMoneyOffered, data=data2, importance=TRUE)

```r
rf.modF = randomForest(ENROLLED_COUNT ~ AGE2 + SEASON  + Miles + CGPA +
LoansOffered + OtherMoneyOffered, data=trainFYR, importance=TRUE)

rf.modT = randomForest(ENROLLED_COUNT ~ CAMPUS_CODE + AGE2 + SEASON  + Miles+
CGPA + LoansOffered + OtherMoneyOffered, data=trainFYT, importance=TRUE)
```

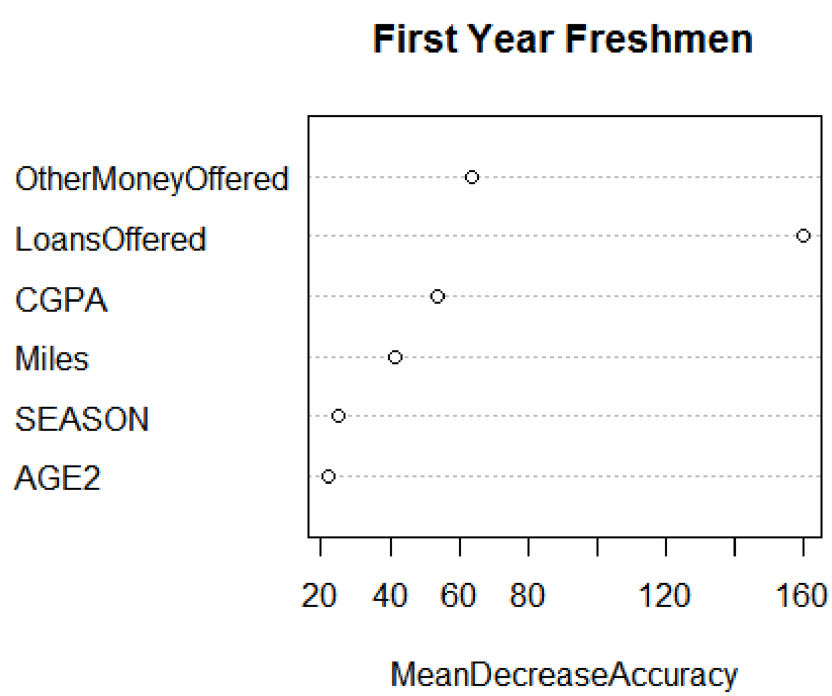
Now, you'll want to check these models for accuracy. To do so, create a confusion matrix. First, have the model predict enrollment on the original (or new) dataset.

```r
testFYR$Predicted <- predict(rf.modF, testFYR)
testFYT$Predicted <- predict(rf.modT, testFYT)
```

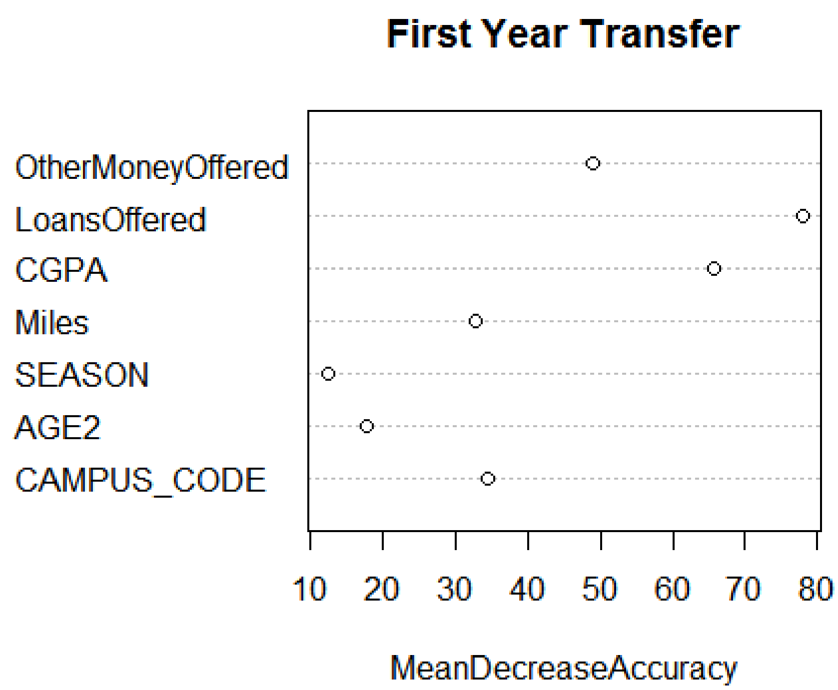
Then, create a confusion matrix to determine accuracy and whether the model is conservative or not in its estimates.

```r
confusionMatrix(data=testFYR$Predicted, reference=testFYR$ENROLLED_COUNT,
positive='1')

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1204  455
##          1  542  776
##
##                Accuracy : 0.6651
##                  95% CI : (0.6478, 0.6821)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3167
##  Mcnemar's Test P-Value : 0.006457
##
##             Sensitivity : 0.6304
##             Specificity : 0.6896
##          Pos Pred Value : 0.5888
##          Neg Pred Value : 0.7257
##              Prevalence : 0.4135
##          Detection Rate : 0.2607
##    Detection Prevalence : 0.4427
##       Balanced Accuracy : 0.6600
##
##        'Positive' Class : 1
##

confusionMatrix(data=testFYT$Predicted, reference=testFYT$ENROLLED_COUNT,
positive='1')

## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction   0   1
##          0 427 172
##          1 158 261
##
##                Accuracy : 0.6758
##                  95% CI : (0.6461, 0.7045)
##     No Information Rate : 0.5747
##     P-Value [Acc > NIR] : 2.221e-11
##
##                   Kappa : 0.3341
##  Mcnemar's Test P-Value : 0.4742
##
##             Sensitivity : 0.6028
##             Specificity : 0.7299
##          Pos Pred Value : 0.6229
##          Neg Pred Value : 0.7129
##              Prevalence : 0.4253
##          Detection Rate : 0.2564
##    Detection Prevalence : 0.4116
##       Balanced Accuracy : 0.6663
##
##        'Positive' Class : 1
##

varImpPlot(rf.modF, sort=F, type=1, main="First Year Freshmen")
```




```
varImpPlot(rf.modT, sort=F, type=1, main = "First Year Transfer")
```



Predict New Enrollment

Use this model to predict on 2017-2018 data.You can get both response (enrolled or not), and liklihood.

```
PredictFYR$Predicted <- predict(rf.modF, PredictFYR, type="response")
PredictFYR$predictionprob <- predict(rf.modF, PredictFYR, type="prob")
```

If you are confident enrollment is finished, you can also test the predictions using the confusion matrix.

```
confusionMatrix(data=PredictFYR$Predicted,
reference=PredictFYR$ENROLLED_COUNT, positive='1')

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1676    3
##          1  682   21
##
##                Accuracy : 0.7124
##                  95% CI : (0.6938, 0.7305)
##     No Information Rate : 0.9899
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.039
```

```
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.875000
##             Specificity : 0.710772
##          Pos Pred Value : 0.029872
##          Neg Pred Value : 0.998213
##              Prevalence : 0.010076
##          Detection Rate : 0.008816
##    Detection Prevalence : 0.295130
##       Balanced Accuracy : 0.792886
##
##        'Positive' Class : 1
##
```

It's important to note that financial aid data was incredibly important in predicting enrollment. If a student hasn't been given their financial aid package yet, then those values will default to 0, and a student will be predicted as less likely to come. So if you want accuracy on predictions, you have two options:

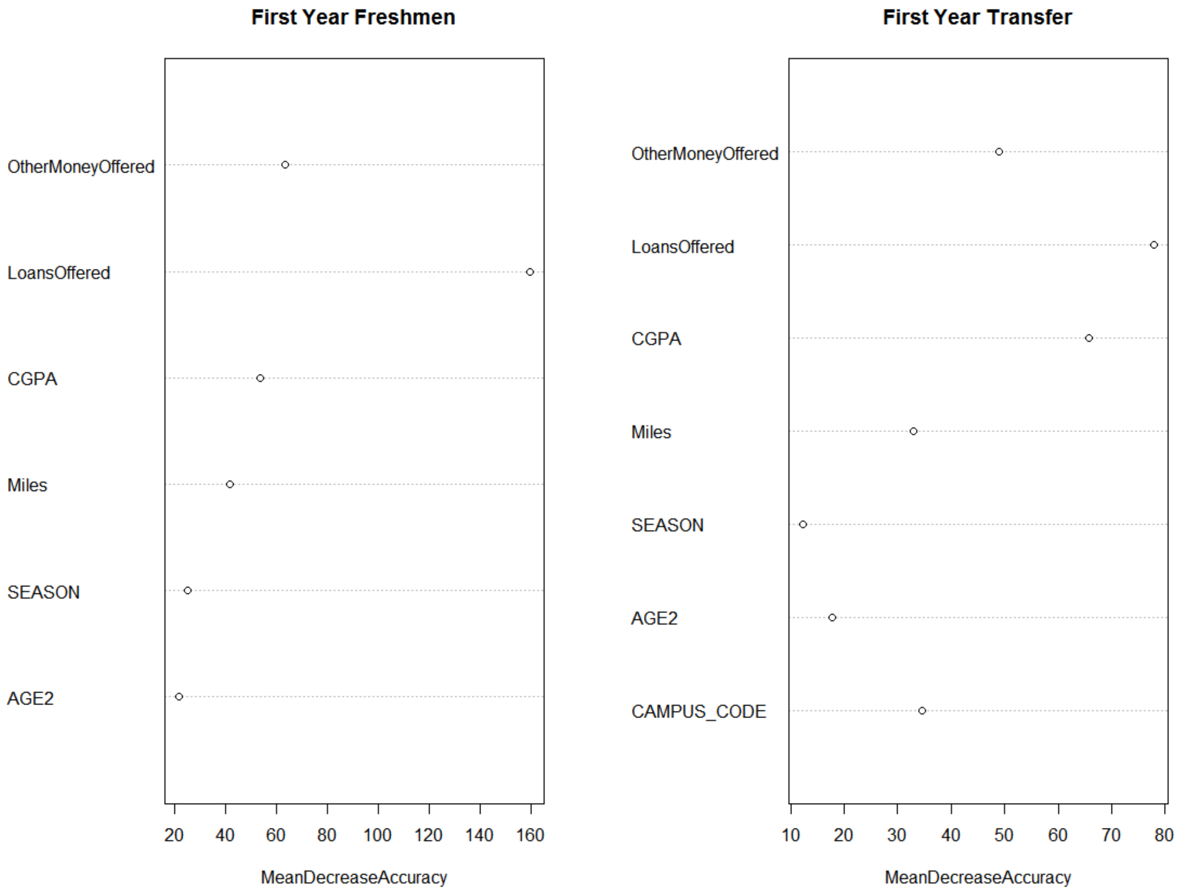
1. Create a model that relies on only application data (which I found to lose about 10-15% accuracy)
2. Wait until financial aid data is in.

Plot Important Variables

RandomForests are so large that it's really difficult to look at the decision trees and come up with meaningful interpretations. However, variable importance plots can give you a good idea of which things are the most important. First, let's look at mean decrease in accuracy - that is, if you remove an item from a decision tree, how much less accurate are those decision trees?

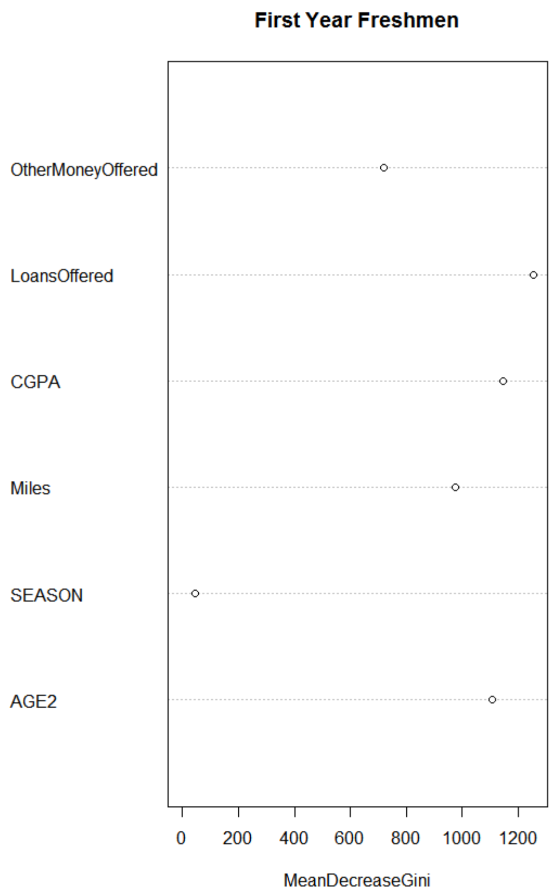
To make our plots pretty, I've done some additional setup. I've changed the rownames of the random forest objects to be easier to read, then used the `par` function to tell it to put two plots next to each other.

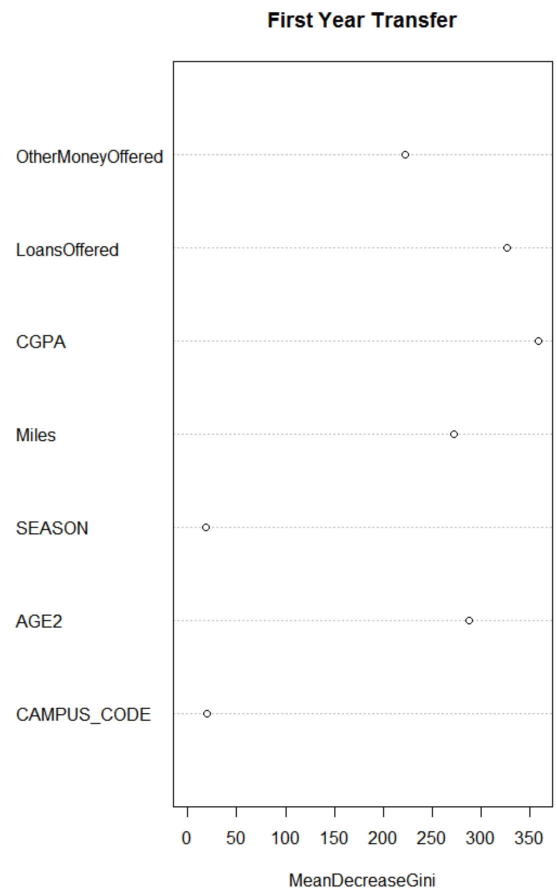
```
#rownames(rf.modF$importance)<- c("Service Campus","Gender", "First Generation","Residency","Age","Season","Distance from the university","GPA","Veteran","Disability","Ethnicity/Race", "Loans Offered", "Scholarships & Grants & Waivers")
#rownames(rf.modT$importance)<- c("Service Campus","Gender", "First Generation","Residency","Age","Season","Distance from the university","GPA","Veteran","Disability","Ethnicity/Race", "Loans Offered", "Scholarships & Grants & Waivers")

par(mfrow=c(1,2))
varImpPlot(rf.modF, sort=F, type=1, main="First Year Freshmen")
varImpPlot(rf.modT, sort=F, type=1, main = "First Year Transfer")
```



But there's also a reduction in "Gini" to be concerned about. Essentially, how many more splits must a tree make if this character is removed? Higher numbers mean these characters greatly simplify trees, even if they don't increase accuracy necessarily.

```
par(mfrow=c(1,2))
varImpPlot(rf.modF, sort=F, type=2, main="First Year Freshmen")
varImpPlot(rf.modT, sort=F, type=2, main = "First Year Transfer")
```





Racial makeup Analysis

Before we go anywhere, I want to take a look at our application admission data and look at people who changed their races in different application years, to see who our "not reported" group really is. 72% of people who reported their race as Not Reported and changed it on another application called themselves white european on at least one of those other applications. 11% called themselves latino.

```
r.change <- data %>%
  group_by(APPLICANT_ID, ETHNICITY_RACE) %>%
  tally()
r.change2 <- dcast(r.change, APPLICANT_ID~ETHNICITY_RACE, value.var = "n")
r.change2[is.na(r.change2)] <- 0
r.change3 <- r.change2 %>%
  mutate(differentgroups = `African American/Black`+`Alaskan/Native
American`+Asian+`European/Middle Eastern/White`+`Hawaiian/Pacific
Islander`+`Latino/Hispanic`+Multiracial+`NonResident Alien`+`Not Reported`)
%>%
  filter(differentgroups > 1) %>%
  filter(`Not Reported` != differentgroups) %>%
  filter(`Not Reported` != 0)
```

```
r.change4 <- data.frame(colSums(r.change3 !=0))
r.change4$labels <- rownames(r.change4)
rownames(r.change4) <- NULL
r.change4 <- r.change4[c(3:11),]
colnames(r.change4) <- c("count", "Ethnicity/Race")
r.change5 <- r.change4 %>% mutate(percentage = count/count[9])
knitr::kable(r.change5, caption="Individuals whose Race/Ethnicity was Not
Reported on one or more applications, and how they reported on a different
applications")
```

Individuals whose Race/Ethnicity was Not Reported on one or more applications, and how they reported on a different applications

count	Ethnicity/Race	percentage
219	African American/Black	0.0387748
61	Alaskan/Native American	0.0108003
279	Asian	0.0493980
4087	European/Middle Eastern/White	0.7236190
27	Hawaiian/Pacific Islander	0.0047805
618	Latino/Hispanic	0.1094193
308	Multiracial	0.0545326
82	NonResident Alien	0.0145184
5648	Not Reported	1.0000000