Comparing Different Methods To Predict Student Retention

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Mark Van Coutren

Dr. Scott Thatcher, Thesis Advisor

Statistics Department

School of Science & Mathematics

2024

TRUMAN STATE UNIVERSITY
Kirksville, Missouri

Dedication

I dedicate this thesis to all the people who have helped and supported me throughout my college career. I'd like to thank my mother and late father who always supported me and were there to lend an ear. I'd also like to thank my older brother who passed down all his hard-earned wisdom to me. Finally, I'd like to thank Barney and Ally Hartman who first made Mountain Dew which provided the extra energy I needed on many late nights.

Acknowledgments

This thesis would not have been possible without the guidance and support of Dr. Scott Alberts, AVP Jonathan Vieker, and Dr. Shanshan Lv. I would also like to thank all the professors and advisors who helped me both in graduate school and my entire college career. I also wish to thank Steve Willott, my high school AP statistics teacher, who sparked my love of math and statistics that led me down this path.

TABLE OF CONTENTS

Abstract

Student retention is a key area of study for universities so they can reach out to students at risk of dropping out to offer support and try to prevent it. While each college will have different reasons for student attrition and each individual student will have their own circumstances as well, is there a type of model that generally works better than others without requiring too much technical knowledge that can be a starting point for universities trying to predict at-risk students?

Three commonly implemented models for predicting students at risk for drop-out include random forest, logistic regression, and neural networks. Based on the results, each model can be tuned to be accurate. However, the complexity of using a neural network is prohibitive, random forests are relatively easy enough to make but not quite as beginner-friendly as logistic regression, which is well-known, powerful, and easy enough to use. It should be noted though there are areas that universities can look into that could not be included in this paper, such as behavioral variables and obtaining more data than is accessible to the general public.

Introduction

Over half of American high school graduates enroll in College, almost 62% in October 2021. Though it's trended downward since 2018, it's been over 61% since at least 1993 which is the first year the United States Bureau of Labor Statistics collected data (Bureau of Labor Statistics, 2022). Higher education is a pathway many take to get better paying jobs, become more well-informed citizens, and generally improve their quality of life. Many parents of low-income families save for years to send their children to college in the hope that they will graduate and have a better life than they did. Though many students attend higher education, 32.9% of undergraduate students will drop out before graduating. Furthermore, 24.1% of college freshmen drop out during their first year (What to Become, 2021). Student retention, or how many students re-enroll from year to year is an area of great interest to universities to both maintain their reputation and enrollment. Most universities live and die on the number of students paying tuition and a student who leaves will not only not pay any more tuition but also could warn other prospective students about their negative experience at the university, making them less likely to enroll. Reaching out to the students who may drop out and offering assistance could sway their decision to drop out so many universities use various statistical methods to attempt to predict students who are likely to drop out before they do.

Predicting retention has been a subject of study since at least the early 70s. Vincent Tinto was one of the first to study and write about retention. Since this was before computer hardware and software had evolved to allow for complex models to be run on each student, his suggestions are more institutional and wide-sweeping than determining if each student is at risk of dropping out. However, his models form the basis of most research

today. His model of student departure (not a model in the statistical sense) focuses on 4 key components: formal and informal academic performance and formal and informal social systems. He found that students not only need to do well in classes but also need social support networks to remain enrolled. Another observation he had that still rings true today is that student retention is a very complex subject with each institution having its specific problems and each student will likely have a unique blend of reasons for dropping out. As he said in one of his later works *Leaving College: Rethinking the Causes and Cures of Student Attrition*, "Despite the extensive body of literature which speaks to the question [why students leave college], there is still much we do not know about its longitudinal character and the complex interplay of forces which give rise to it."

Professional software, such as Civitas and Navigate360, are common tools used to predict students at risk of dropping out, but these can be expensive and could be out of reach for some universities that are already struggling financially due to students leaving. These will have access to the entirety of the university's data giving the model a very large training set. Though these companies keep their models secret, some studies have posted their results publicly and these can provide insight into what variables are expected to be important. Most of the best predictors for whether a student will drop out are things the university cannot affect like high school GPA and ACT scores, gender, race, variables that represent socioeconomic status, and whether they're a first-generation college student or not to note a few examples. The newest research focuses on student behavior since it is both very predictive and something the university can influence with programs to help students who are at risk of dropping out.

There most likely is not a single best model to predict which students are at risk of dropping out across all universities since the biggest reasons students drop out can vary widely from university to university. Despite this, there could be types of models that are more accurate and easier to make, and seeing which models work better on a small sample of data could provide some insight to universities that are looking into building a model to predict at risk students.

Methods

The ideal data for this type of analysis would be a large sample of American college students, with a lot of demographic and behavioral data. Unfortunately, that type of data is not often publically available due to student privacy laws such as the Family Educational Rights and Privacy Act (FERPA) and the Protection of Pupil Rights Amendment (PPRA). Hence, the data selection is very limited. The data used for this comparison is a combination of multiple institutions and databases in Europe compiled for a paper that had similar goals in mind (Realinho & Machado & Baptista & Martins 2022), and uploaded to Kaggle by user "The Devastator". The data is 4,424 observations of 35 variables, mostly demographic data, with a Target variable that describes whether the student graduated, dropped out, or stayed enrolled. Notably, all of the students in the data attended European schools. This does mean that all of the models could be biased for European students and European students could leave secondary education for different reasons than US students. Numerous political, social, and economic differences between the US and Europe could affect a student's decision to stay or drop out such as college being free/heavily subsidized in many European countries for example. There still could be useful information gained

from this paper but care should be used when applying these results to students outside of the scope of the data.

Three general models will be compared in this paper: a Random Forest, a Neural Network, and a Logistic Regression, and they will be individually tuned to get the best result. All of these models will use the tidyverse and caret R packages with the Random Forest and Neural Network models using the randomForest and neuralnet packages respectively. A Random Forest is a set of decision trees with a group of randomly selected variables that are combined to get a single result. This is commonly used in regression problems and classification. A Neural Network is a machine learning algorithm that attempts to replicate the physiology of the human brain. In this situation, three sets of layers are used, comprised of nodes with weighted connections between the nodes that determine how much the following node will activate, meaning a higher number will be passed onto the next layer. First, the data is put in an input layer, each node in the input layer will "light up" based on how large the value is relative to all the other data points. Then one or more hidden layers that connect the input layer to the output layer. This is where what's called deep learning happens. The number of layers and nodes within each layer can be chosen and adjusted to optimize performance. Lastly, there is an output layer that provides the result, in this case, dropout or not. The specific weights of the connections are where the training happens. The neural network is given inputs and guesses the answer it then calculates the difference between that guess and the correct answer and it adjusts the weights accordingly. It does this for all the data and after many iterations, the connections have weights that ideally accurately map the relationship between the input and output layer. A visualization of a Neural Network will be provided later. The final model

will be a Logistic Regression. The concept is the same as normal linear regression, using a set of independent data to predict a dependent variable. The difference between this and normal regression is the variable predicted is the probability of being a certain factor so it is restricted between 1 and 0.

The data cleaning was straightforward since the original authors had already cleaned it. The user who uploaded the dataset to Kaggle added a few variables that seemed to be about the county that the student was from but since it was not relevant to this use case, they were removed. Originally, the goal was to make models that predicted if a student was going to drop out, graduate on time, or stay enrolled past the expected time but predicting students that were going to stay longer was cut back for reasons discussed later, so any student that stayed enrolled past the typical time of the degree completion was counted as a non-drop-out. After that, the factor variables needed to be converted from integers to factors for use in the Random Forest and Logistic Regression. The Neural Network requires special consideration that will be discussed later. The factor variables include Marital.status, Daytime.evening.attendance, Application.mode, Course, Previous.qualification, Mother.s.qualification, Father.s.qualification, Mother.s.occupation, Father.s.occupation, International, and Target, and from there, the lappy() and factor() functions were used to turn them into factors. To see the specific code used, please refer to the appendix. Finally, some of the variables measured were very granular, so quite a few had less than 10 observations in this dataset. To satisfy the assumptions of Logistic Regression, they were combined into a general “other” value for each variable. This data was used for all models to maintain consistency. There are only 6 variables that need this consideration, Application.mode, Previous.qualification, Mother.s.qualification,

Father.s.qualification, Mother.s.occupation, and Father.s.occupation. Once the variables are bucketed, the subset function was used to select the variables considered for the models, Application.mode, Application.order, Gender, Displaced, Tuition.fees.up.to.date, Scholarship.holder, Marital.status, Daytime.evening.attendance, Course, Previous.qualification, Mother.s.qualification, Father.s.qualification, Mother.s.occupation, Father.s.occupation, Age.at.enrollment, International, and Target. The rest were left out because they either had an unclear meaning or were irrelevant to the study at this time. Taking a second look at these variables could be an area of further study.

Results

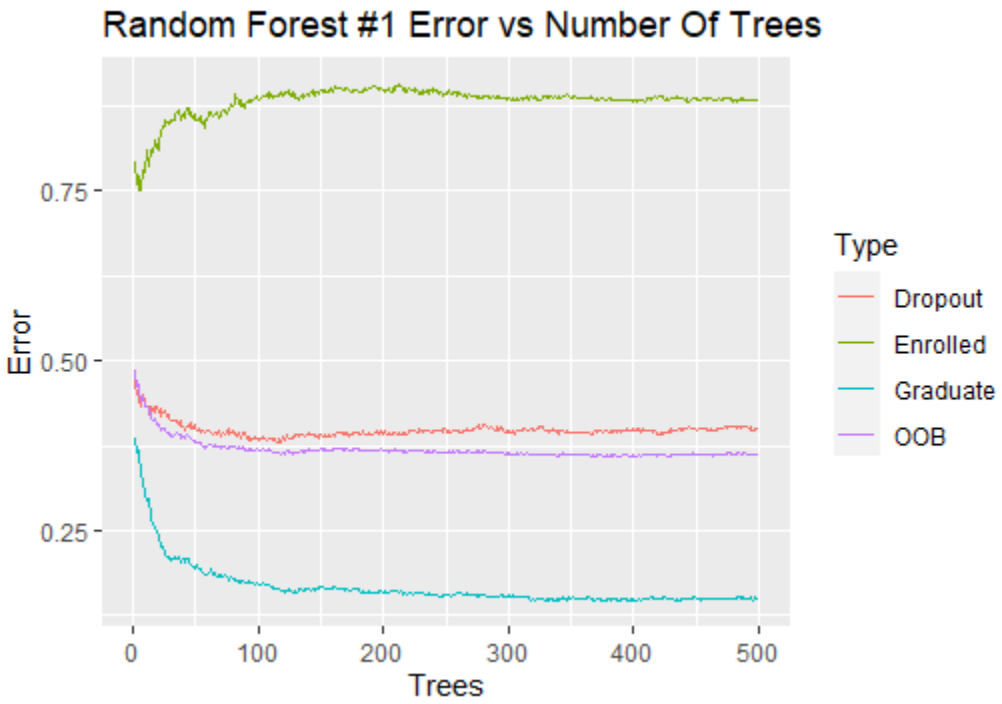
The first model is a Random Forest model. For the first attempt, the model will try to predict students who dropped out, graduated, and stayed enrolled. This means the target variable didn't need to be mutated, and therefore the data could be used without further cleaning. 85% of the data was then split into a training set and the other 15% into a test set for later. The results of running a random forest without changing the default arguments are below.

```
 randomForest(formula = Target ~ ., data = df.train1, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 36.17%
Confusion matrix:
         Dropout Enrolled Graduate class.error
Dropout      726       54      430   0.4000000
Enrolled     158       80      437   0.8814815
Graduate     222       59     1594   0.1498667
```

```
Accuracy
0.622084
```

From this run, there was an Out-Of-the-Box error rate (OOB) of 36.17%, and the model's prediction on the test set had an accuracy of 62.21%. While this was better than a random guess, it can be improved. There were a few things to do to see where improvements to the model can be made. The first thing was to see if the model was allotted enough trees to improve. For random forests, sometimes the OOB will still fluctuate until the model reaches the limit of trees and could still get better with more trees. To check the number of trees, the error rates can be loaded into a data frame and plotted against the number of trees using ggplot. The idea and code used to do this came from "StatQuest: Random Forests in R." by StatQuest with Josh Starmer on YouTube. The resulting graph is below.



Based on the graph, the error rates were stabilizing long before 500 trees. This means the number of trees could have been reduced to save computing time but was left at 500. Something else to try is optimizing the number of variables at each split. Random Forests have a certain number of variables to try at each tree. The default value is 3 but this

can be changed with the mtry argument in the randomForest() function. To see which value is best, a model was run at different values for mtry, ranging from 1 to 10 and the final OOB error rates were saved to see which one was the lowest. For this model, the lowest error rate was 3 with 2 being a close second. Since they were always very close either can be chosen and the results should be similar. Since the default value was the best, other improvements were explored. Another option would be to reduce the scope of the prediction. From the class.error and the graph, it was clear that the model struggled most with predicting the students who stayed enrolled. This may have been able to be reconciled with the data already available or with more data but that is best left for future analysis in the interest of being able to test multiple methods. For this paper, the analysis was reduced to dropout vs not dropout. To do so, all the "Graduate" and "Enrolled" values were mutated into "Did Not Dropout.". The variable split was tested again with the mutated data and a mtry of 2 was found to be slightly better than 3. The same analysis was run with a mtry of 2 on the mutated data. The following results were obtained.

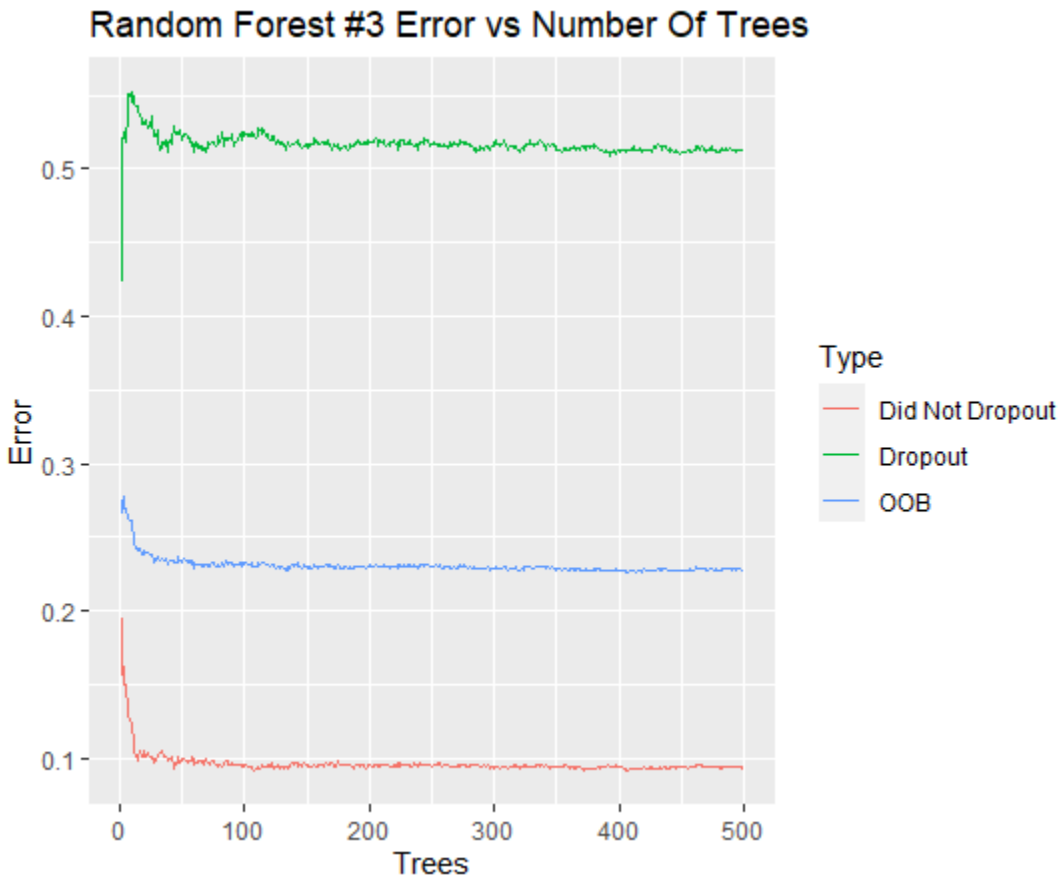
```
 randomForest(formula = Target ~ ., data = df.train3, importance = TRUE,      mtry = 2)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 22.77%
Confusion matrix:
                Did Not Dropout Dropout class.error
Did Not Dropout            2319     239  0.09343237
Dropout                     617     585  0.51331115
```

```
 Accuracy
0.7981221
```

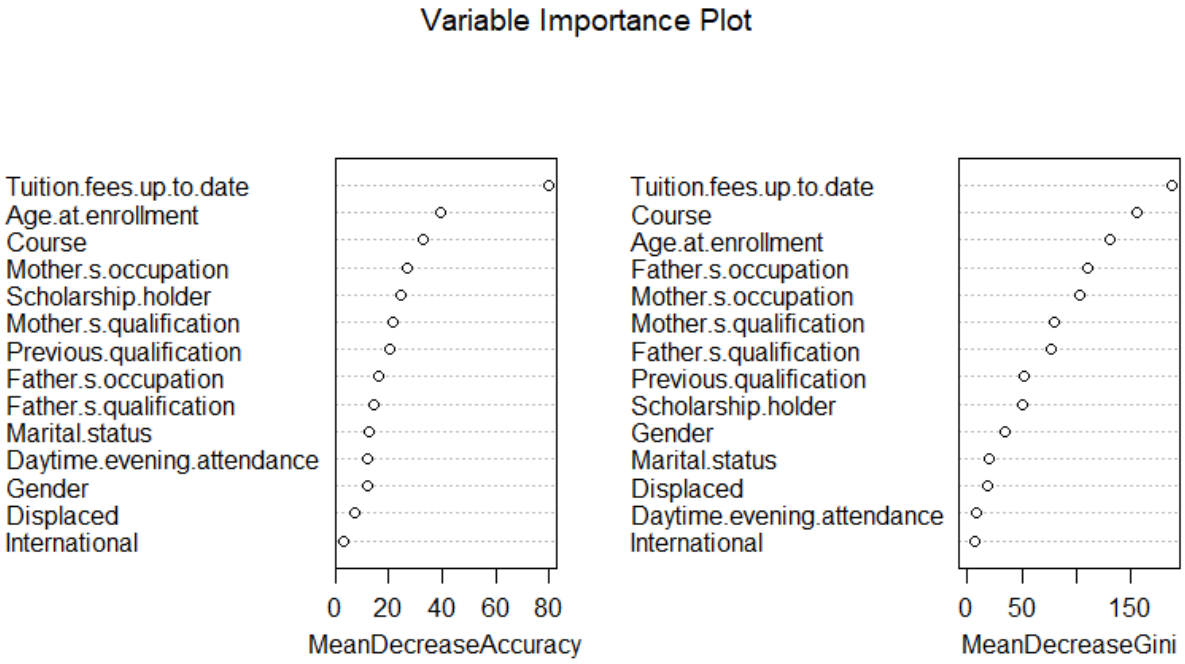
A much better error rate, 22.77%, and accuracy, 79.81%, were reported. The class error for dropout was still not the best but it wasn't as bad as it was for enrolled. Then the model was checked to determine if it had enough time to converge. As seen from the graph

below, this model converges long before 500 trees as well, though the model still struggles with predicting dropout. With an accuracy in the high 70s, this model was used in the final comparison.

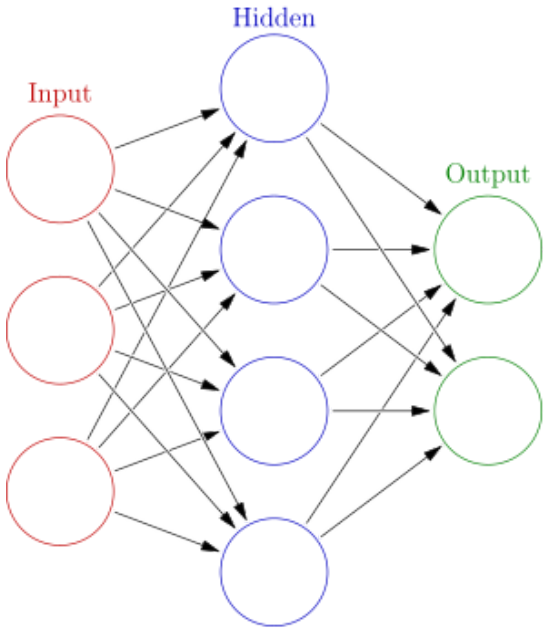


There are two other metrics to consider when analyzing a random forest. The average decrease of accuracy when each variable is taken out of the random forest and the average decrease of Gini impurity when each variable is added into the random forest. Gini impurity is a measurement of how likely an incorrect classification is. Both of these can be examined at the same time with a variable importance plot. The status of a student's tuition seems to be the most important variable. Beyond that, course and age are the next most

important, after that mother's/father's occupation which can indicate socioeconomic status. This will be discussed further later.



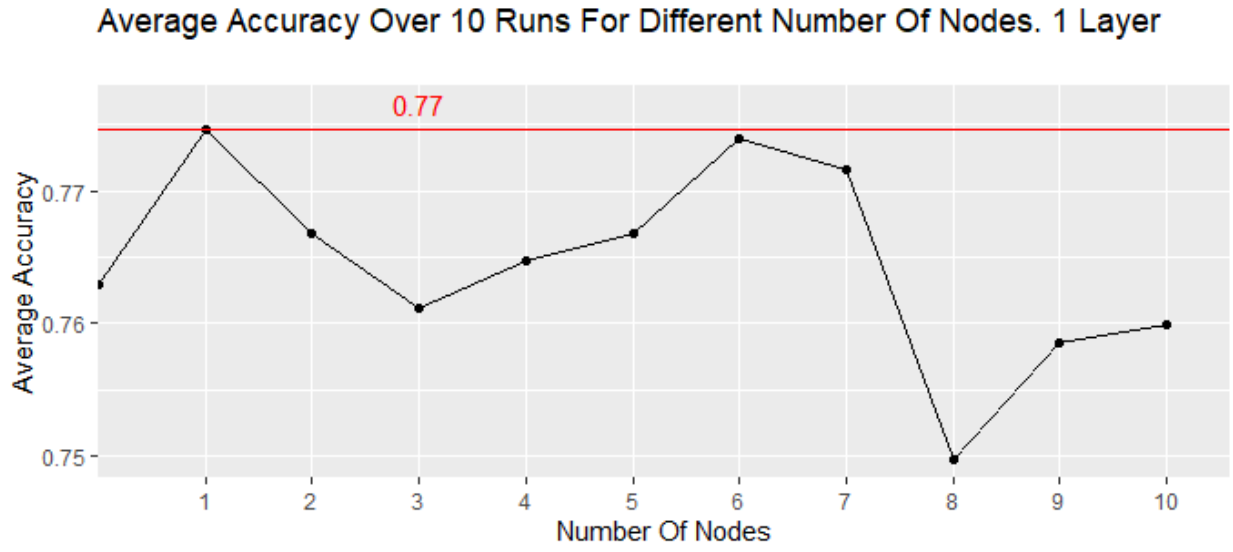
For the next model, a Neural Network was used. As a reminder, this consists of three layers, an input, hidden, and output layer. A visualization of a generic neural network is below (Wikimedia Commons, 2023) though it is slightly different from the one used in this model.



In this model, there is an input layer consisting of each of the variables and an output layer of how likely a student is to not drop out. Choosing the specific number of nodes in a hidden layer and how many hidden layers are needed can be tricky. The general rule is that the more complex a problem is, the more layers and nodes are required. Tuning the neural network to a specific model requires checking many different configurations and seeing what the accuracy is. It is important to note that while neural networks themselves are deterministic, meaning that given the same inputs are used, the model will be the same every time, the training and test splits used, which is still 85% train and 15% test, are not so a few different configurations were tested multiple times to see which one tended to perform better. A for loop was used to go through each number of nodes and another loop to run a neural network a few times and store the accuracy. To prepare the data for the neural network, it needs to first be normalized. This means transforming the data so all values fall between 0 and 1 with the higher numbers closer to 1 and the lower numbers

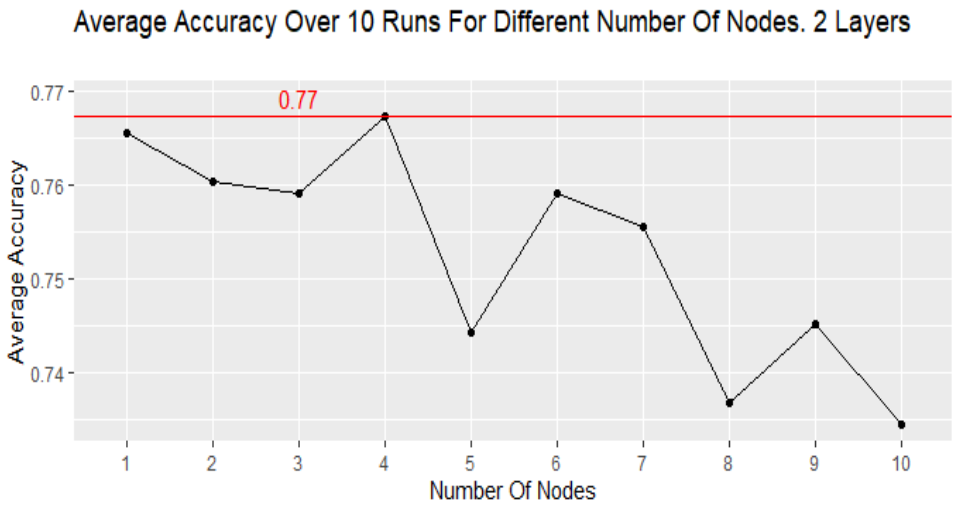
closer to 0. This is done easily by subtracting the minimum of each variable from each data point in that variable and dividing that result by the range of the data. This means that the variables with higher values will not light up a node more than they should be relative to the rest of the data. It should be noted that neural networks are not the best for data that has factor variables with a lot of different levels because the neural network cannot accept factors directly so the factors would need to be converted to indicator variables. That would cause the computing time to skyrocket. For now, they will be treated as numbers.

One hidden layer was done first, and then each number of nodes was checked between 0 and 10 ten times each, the accuracy of each run was saved, the results for each node configuration averaged, and the result charted. This does take a while, especially getting into the larger numbers of nodes. The entire code chunk took over 13 hours. The graph of the average accuracy is below.



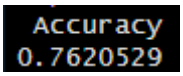
As from the graph, the optimal number of nodes maxed out at just 1. 6 nodes were almost as accurate but 1 node was used for this paper. The same process was used for neural networks with 2 layers with the same number of nodes in each layer. It should be

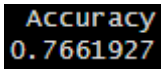
noted that this code often errored out before completing. Raising the threshold from the default of .01 to .1 allowed it to run but further tweaking could be done. The graph of the average accuracy for 2 layers is below.

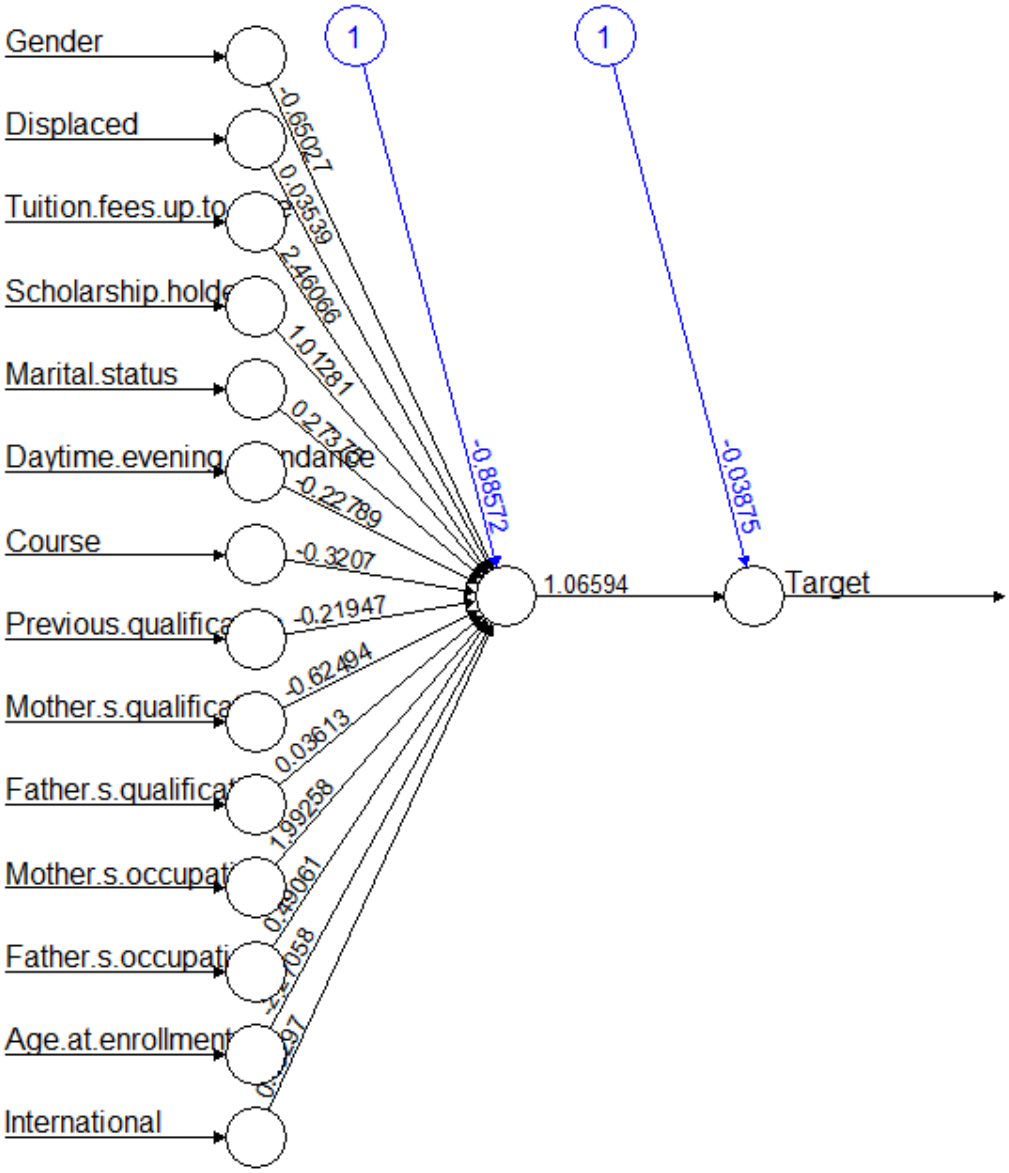


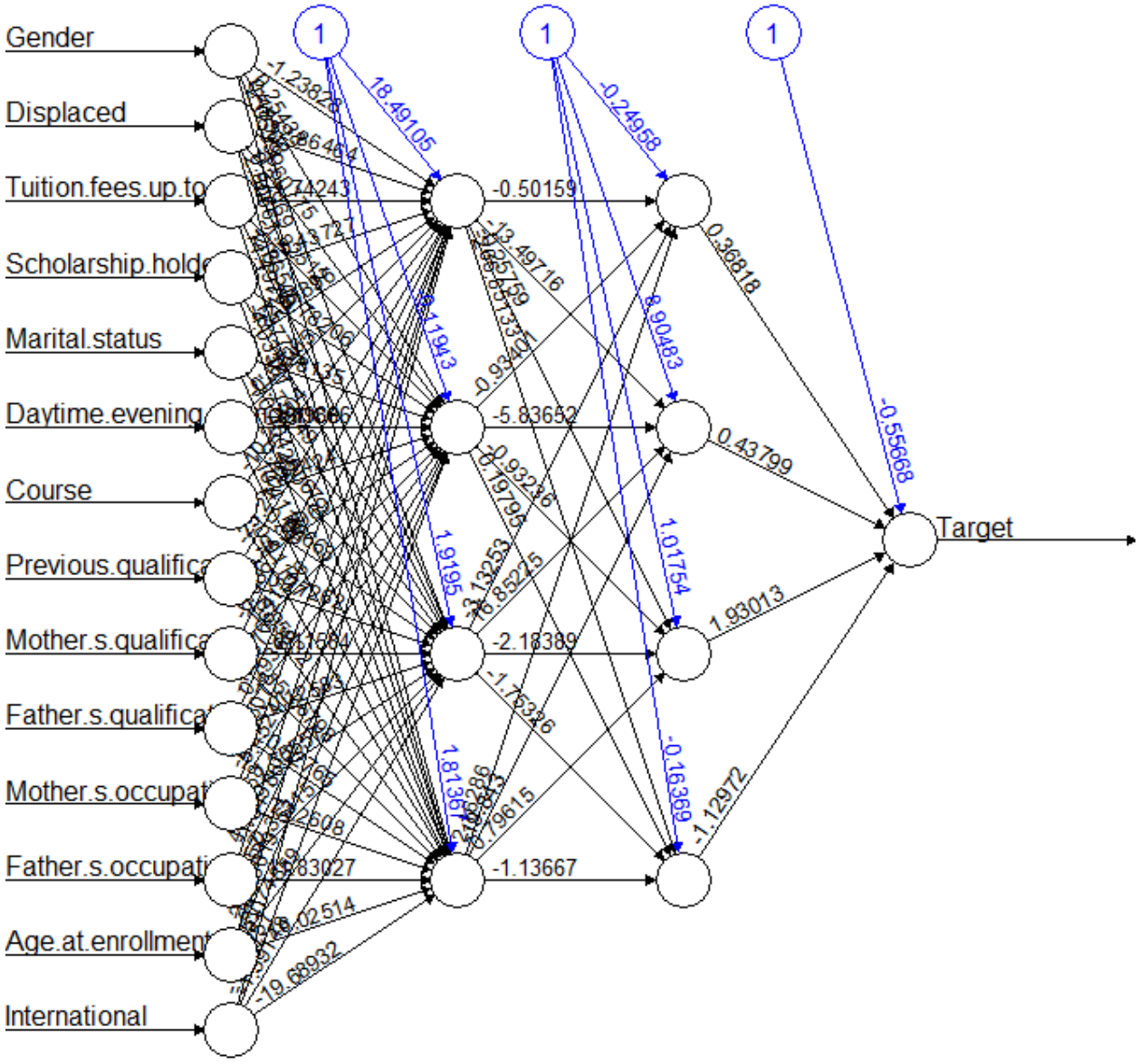
The 2 layer neural network got a very similar accuracy but with 4 nodes in each layer. It's worth noting that increasing the number of nodes seems to decrease the accuracy, possibly due to overfitting. A 3-layer neural network was attempted but it wouldn't run even with a threshold of .5 so that was left as an area of further study.

The output layer of the neural network gives the predicted likelihood of a student not dropping out. To change this from a decimal to a factor, anything higher than .5 will be set to 1, and anything else to 0. Both the 1-layer and 2-layer neural networks had an accuracy in the mid-70s. A visualization of both neural networks is below.

1 layer: 

2 layers: 


Gender
Displaced
Tuition.fees.up.to
Scholarship.holde
Marital.status
Daytime.evening
Course
Previous.qualifica
Mother.s.qualifica
Father.s.qualificat
Mother.s.occupati
Father.s.occupati
Age.at.enrollment
International
1
1
-0.88572
-0.03875
1.06594
Target



A logistic regression was used for the final model to test. While the underlying math behind changing from Linear Regression to Logistic Regression is fairly complex, it is very simple to do in software. All that needed to be done is use a glm() function and tell it to use logistic regression. The only other consideration is that when the test data is predicted, the model predicts the probability of not dropping out. To compare to the real target variable,

probability needs to be converted to a factor. This is done easily enough by labeling anything with a greater than .5 as a non-dropout, just as in the neural network. The accuracy of Logistic Regression can be measured in an additional way known as the Akaike information criterion or AIC, which measures the likelihood of seeing the observations given the model, adjusted for model complexity to combat overfitting. For AIC, the lower the value, the better. A logistic regression was run with all the variables as a baseline. The output was too large to include in this text so it will be in the appendix. The model had an accuracy of 77.07% and an AIC of 3531. While this is about as good as some of the other more complicated models, it can be improved by selecting the variables that the model uses. There are a few different methods of model selection but the most popular is stepwise model selection. Initially, all of the variables are included in the model, and at each step, a variable is selected to either be removed or added back into the model based on the Akaike Information Criterion (AIC). This way the model gets the most important variables without overfitting. When the stepwise regression was run, a final AIC of 3502 and an accuracy of 77.69% was reached. The specific model will also be included in the appendix. Looking at the coefficients for the variables, tuition fees being up to date, previous qualifications, and the mother's occupation seemed to be the most important variables.

Discussion

After analyzing all three of the models, it's notable that each one had an accuracy of in the mid to high 70s. Though this was affected by the randomness of the test train split, they all performed very well. The biggest factor that affects the decision of which model to use is the complexity of making it. Neural networks are generally more complex than the

other models and some compromises were made with some of the factors. That, along with the long time to check the different node configurations kills any recommendation to use it to anyone but technical experts. The decision between Logistic Regression and Random Forest is a lot closer. Either could feasibly be used and give great results. Regression is much more well-known and has fewer options to tweak before getting good results so that might be better for anyone without a lot of technical knowledge.

The Random Forest and Logistic Regression gave the clearest picture of what variables affected dropout chance the most. Notably, the tuition fees were the most important variable for both Gini and accuracy in the random forest. This variable also had the biggest coefficient in the logistic regression and the biggest Z value, which means it affected the model more. This makes intuitive sense, as a student who is struggling to pay tuition would be more likely to drop out to both work to save up money and not incur more fees. The next variables in the importance plot were the age at enrollment and the course the student took. The harder courses would make students more likely to drop out but the regression coefficient was negative for age at enrollment which means that the younger someone is the less likely they are to drop out. It seems that traditional students seem to have an easier time graduating than those who start when they're older and have more responsibilities competing for their time.

There are also a lot of future areas of study that could prove fruitful that had to be left out of this paper due to data availability restrictions. As noted previously, all the data came from European colleges which might have different retention problems from different regions. The models also couldn't predict if students would stay longer than the expected time to obtain a degree, but with more data, it likely would have been able to more accurate.

It is also possible that key variables with high predictive power are missing from this analysis, as the availability of some of the useful variables is very limited. Student Retention is still a relatively new area of study and is constantly evolving. As mentioned earlier, the next frontier focuses on student behavior and, unlike demographic predictors, the university can influence student behavior. At-risk study groups and events for students who are demographically at risk of dropping out, like first-generation student dinners, have been used to help academically and socially integrate students who might otherwise feel isolated.

The predictive models based on this use case can also be tuned if this model is implemented in real life as a tool to predict which students were at risk of dropping out to offer them extra resources. It is more advantageous to increase the sensitivity, as most universities would rather include a student who isn't struggling than miss a student who is struggling. Even though this would hurt real-world accuracy, it would be much better for that use case. There are a few different ways to do this but the easiest way would be to randomly select students who dropped out and duplicate them in the training set.

While student retention as an area of study has matured from Tinto's days, it has remained a very complex issue with ever-evolving methods. With college enrollment decreasing year over year, universities will need to focus on keeping students enrolled rather than competing for the shrinking pool of new students. This could be increasingly difficult as remote learning becomes more and more common. How can a student feel socially integrated when they're still in the same place as they were before college? These challenges could force universities to focus more on student retention and the field could have a period of rapid growth.

References

Bureau of Labor Statistics, U.S. Department of Labor, The Economics Daily, 61.8 percent of recent high school graduates enrolled in college in October 2021 at https://www.bls.gov/opub/ted/2022/61-8-percent-of-recent-high-school-graduates-enrolled-in-college-in-october-2021.htm

What To Become. (2021). Everything You Need to Know About the College Dropout Rate. https://whattobecome.com/blog/college-dropout-rate/

Realinho, V.; Machado, J.; Baptista, L.; Martins, M.V. Predicting Student Dropout and Academic Success. Data 2022, 7, 146. https://doi.org/10.3390/data7110146

"Predict Students' Dropout and Academic Success." Kaggle, www.kaggle.com/datasets/thedevastator/higher-education-predictors-of-student-retention

Starmer, Josh. "StatQuest: Random Forests in R." YouTube, YouTube, 26 Feb. 2018, www.youtube.com/watch?v=6EXPYzbfLCE.

File:Colored neural network.svg. (2023, September 24). Wikimedia Commons. Retrieved 20:46, October 1, 2023 from https://commons.wikimedia.org/w/index.php?title=File:Colored_neural_network.svg&oldid=804347670.

Tinto, Vincent. Leaving College: Rethinking the Causes and Cures of Student Attrition. University of Chicago Press, 2022.

Appendix

Data Cleaning:

```
library(tidyverse)

library(caret)


dropout <- read.csv("C:/Users/Mark/Desktop/Grad School/PDAT630/dropout.csv")


dropout_clean=dropout

colnames(dropout_clean)[1]='Marital.status'


## Combining under represented factors into buckets of at least 10

table(dropout_clean$Marital.status)

table(dropout_clean$Application.mode)

table(dropout_clean$Previous.qualification)

table(dropout_clean$Mother.s.qualification)

table(dropout_clean$Father.s.qualification)

table(dropout_clean$Mother.s.occupation)

table(dropout_clean$Father.s.occupation)


dropout_clean=mutate(dropout_clean, Marital.status = ifelse((Marital.status == "3"|

                                        Marital.status == "6"), 7, Marital.status))


dropout_clean=mutate(dropout_clean, Application.mode = ifelse((Application.mode == "2"|
```

```
                                        Application.mode =="5"|
                                        Application.mode == "10"|
                                        Application.mode == "11"|
                                        Application.mode == "18"), 19, Application.mode))

dropout_clean=mutate(dropout_clean, Previous.qualification =
        ifelse((Previous.qualification == "4"|
                                        Previous.qualification == "5"|
                                        Previous.qualification == "8"|
                                        Previous.qualification == "10"|
                                        Previous.qualification == "11"|
                                        Previous.qualification == "13"|
                                        Previous.qualification == "17"), 18,
        Previous.qualification))

dropout_clean=mutate(dropout_clean, Mother.s.qualification =
        ifelse((Mother.s.qualification == "6"|
                                        Mother.s.qualification == "7"|
                                        Mother.s.qualification == "8"|
                                        Mother.s.qualification == "9"|
                                        Mother.s.qualification == "11"|
                                        Mother.s.qualification == "12"|
                                        Mother.s.qualification == "14"|
```

```
                                        Mother.s.qualification == "15"|
                                        Mother.s.qualification == "16"|
                                        Mother.s.qualification == "17"|
                                        Mother.s.qualification == "18"|
                                        Mother.s.qualification == "20"|
                                        Mother.s.qualification == "21"|
                                        Mother.s.qualification == "24"|
                                        Mother.s.qualification == "25"|
                                        Mother.s.qualification == "26"|
                                        Mother.s.qualification == "27"|
                                        Mother.s.qualification == "28"|
                                        Mother.s.qualification == "29"), 35,
    Mother.s.qualification))

dropout_clean=mutate(dropout_clean, Father.s.qualification = ifelse((Father.s.qualification
    == "6"|
                                        Father.s.qualification == "7"|
                                        Father.s.qualification == "8"|
                                        Father.s.qualification == "11"|
                                        Father.s.qualification == "12"|
                                        Father.s.qualification == "13"|
                                        Father.s.qualification == "15"|
                                        Father.s.qualification == "16"|
```

```
                                        Father.s.qualification == "17"|
                                        Father.s.qualification == "18"|
                                        Father.s.qualification == "19"|
                                        Father.s.qualification == "20"|
                                        Father.s.qualification == "21"|
                                        Father.s.qualification == "22"|
                                        Father.s.qualification == "23"|
                                        Father.s.qualification == "25"|
                                        Father.s.qualification == "26"|
                                        Father.s.qualification == "30"|
                                        Father.s.qualification == "31"|
                                        Father.s.qualification == "32"|
                                        Father.s.qualification == "33"|
                                        Father.s.qualification == "34"), 35,
    Father.s.qualification))

dropout_clean=mutate(dropout_clean, Mother.s.occupation = ifelse((Mother.s.occupation
    == "11"|
                                        Mother.s.occupation == "14"|
                                        Mother.s.occupation == "15"|
                                        Mother.s.occupation == "16"|
                                        Mother.s.occupation == "17"|
                                        Mother.s.occupation == "18"|
```

```
                                        Mother.s.occupation == "19"|
                                        Mother.s.occupation == "20"|
                                        Mother.s.occupation == "21"|
                                        Mother.s.occupation == "22"|
                                        Mother.s.occupation == "23"|
                                        Mother.s.occupation == "24"|
                                        Mother.s.occupation == "25"|
                                        Mother.s.occupation == "26"|
                                        Mother.s.occupation == "27"|
                                        Mother.s.occupation == "28"|
                                        Mother.s.occupation == "30"|
                                        Mother.s.occupation == "31"), 47, Mother.s.occupation))

dropout_clean=mutate(dropout_clean, Father.s.occupation = ifelse((Father.s.occupation ==
        "14"|
                                        Father.s.occupation == "15"|
                                        Father.s.occupation == "16"|
                                        Father.s.occupation == "17"|
                                        Father.s.occupation == "18"|
                                        Father.s.occupation == "19"|
                                        Father.s.occupation == "20"|
                                        Father.s.occupation == "21"|
                                        Father.s.occupation == "22"|
```

```
Father.s.occupation == "23"|
Father.s.occupation == "24"|
Father.s.occupation == "25"|
Father.s.occupation == "26"|
Father.s.occupation == "27"|
Father.s.occupation == "28"|
Father.s.occupation == "29"|
Father.s.occupation == "30"|
Father.s.occupation == "31"|
Father.s.occupation == "32"|
Father.s.occupation == "33"|
Father.s.occupation == "34"|
Father.s.occupation == "35"|
Father.s.occupation == "36"|
Father.s.occupation == "37"|
Father.s.occupation == "38"|
Father.s.occupation == "39"|
Father.s.occupation == "40"|
Father.s.occupation == "41"|
Father.s.occupation == "42"|
Father.s.occupation == "43"|
Father.s.occupation == "45"|
Father.s.occupation == "46"), 47, Father.s.occupation))
```

```
#factor_cols=c("Marital.status","Scholarship.holder","Tuition.fees.up.to.date","Gender","Disp
        laced","Daytime.evening.attendance","Application.mode",
        "Course","Previous.qualification","Mother.s.qualification","Father.s.qualification","Mo
        ther.s.occupation","Father.s.occupation","International","Target")
#dropout_clean[factor_cols] <- lapply(dropout_clean[factor_cols], factor)
#sapply(dropout_clean, table)

dropout_clean=subset(dropout_clean, select = c(Gender,Displaced,Tuition.fees.up.to.date,
                                Scholarship.holder,Marital.status, Daytime.evening.attendance,
        Course,
                                Previous.qualification,
        Mother.s.qualification,Father.s.qualification,
                                Mother.s.occupation, Father.s.occupation, Age.at.enrollment,
                                International, Target))

#str(dropout_clean)

write.csv(dropout_clean, "C:/Users/Mark/Desktop/Grad
        School/PDAT630/dropout_clean.csv", row.names=FALSE)
```

Random Forest:

```
library(tidyverse)
```

```
library(caret)
library(randomForest)

dropout_clean <- read.csv("C:/Users/Mark/Desktop/Grad
        School/PDAT630/dropout_clean.csv")
factor_cols=c("Marital.status","Scholarship.holder","Tuition.fees.up.to.date","Gender","Displ
        aced","Daytime.evening.attendance","Course","Previous.qualification","Mother.s.quali
        fication","Father.s.qualification","Mother.s.occupation","Father.s.occupation","Internat
        ional","Target")
dropout_clean[factor_cols] <- lapply(dropout_clean[factor_cols], factor)
str(dropout_clean)

## Random Forest 1
dropout_clean_rf1=dropout_clean
str(dropout_clean_rf1)

df.train1 <- sample_frac(dropout_clean_rf1, size = .85)
df.test1 <- setdiff(dropout_clean_rf1, df.train1)

dropout.rf1 <- randomForest(Target ~ .,
                    importance = TRUE,
                    #proximity = TRUE,
                    data = df.train1)
```

```
print(dropout.rf1)
pred1=predict(dropout.rf1, df.test1, type="response")
print(confusionMatrix(as.factor(df.test1$Target), pred1)$overall[1])

oob.error.data <- data.frame(
  Trees=rep(1:nrow(dropout.rf1$err.rate), times=4),
  Type=rep(c("OOB", "Dropout", "Enrolled",
             "Graduate"), each=nrow(dropout.rf1$err.rate)),
  Error=c(dropout.rf1$err.rate[,"OOB"],
             dropout.rf1$err.rate[,"Dropout"],
             dropout.rf1$err.rate[,"Enrolled"],
             dropout.rf1$err.rate[,"Graduate"]))

ggplot(data=oob.error.data, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))+
  ggtitle("Random Forest #1 Error vs Number Of Trees")



## Random Forest 2
dropout_clean_rf2=dropout_clean
str(dropout_clean_rf2)

df.train2 <- sample_frac(dropout_clean_rf2, size = .85)
```

```
df.test2 <- setdiff(dropout_clean_rf2, df.train2)


dropout.rf2 <- randomForest(Target ~ .,
                    importance = TRUE,
                    #proximity = TRUE,
                    data = df.train2,
                    mtry=2)
print(dropout.rf2)
pred2=predict(dropout.rf2, df.test2, type="response")
print(confusionMatrix(as.factor(df.test2$Target), pred2)$overall[2])


oob.error.data <- data.frame(
  Trees=rep(1:nrow(dropout.rf2$err.rate), times=4),
  Type=rep(c("OOB", "Dropout", "Enrolled",
          "Graduate"), each=nrow(dropout.rf2$err.rate)),
  Error=c(dropout.rf2$err.rate[,"OOB"],
          dropout.rf2$err.rate[,"Dropout"],
          dropout.rf2$err.rate[,"Enrolled"],
          dropout.rf2$err.rate[,"Graduate"]))


ggplot(data=oob.error.data, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))+
  ggtitle("Random Forest #2 Error vs Number Of Trees")
```

```
## Random Forest 3
dropout_clean_rf3=mutate(dropout_clean, Target = factor(ifelse(Target == "Dropout",
        "Dropout", "Did Not Dropout")))
str(dropout_clean_rf3)

table(dropout_clean$Target)
table(dropout_clean_rf3$Target)

df.train3 <- sample_frac(dropout_clean_rf3, size = .85)
df.test3 <- setdiff(dropout_clean_rf3, df.train3)

dropout.rf3 <- randomForest(Target ~ .,
                    importance = TRUE,
                    #proximity = TRUE,
                    data = df.train3
                    #keep.forest=TRUE,
                    ,mtry=2
                    )
print(dropout.rf3)
pred3=predict(dropout.rf3, df.test3, type="response")
```

```
print(confusionMatrix(as.factor(df.test3$Target), pred3)$overall[1])


oob.error.data <- data.frame(
  Trees=rep(1:nrow(dropout.rf3$err.rate), times=3),
  Type=rep(c("OOB", "Dropout", #"Enrolled",
          "Did Not Dropout"), each=nrow(dropout.rf3$err.rate)),
  Error=c(dropout.rf3$err.rate[,"OOB"],
          dropout.rf3$err.rate[,"Dropout"],
          #dropout.rf3$err.rate[,"Enrolled"],
          dropout.rf3$err.rate[,"Did Not Dropout"]))

ggplot(data=oob.error.data, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))+
  ggtitle("Random Forest #3 Error vs Number Of Trees")



varImpPlot(dropout.rf3,
          main="Variable Importance Plot"
          )



oob.values <- vector(length=10)
for(i in 1:10) {
```

```
  temp.model <- randomForest(Target ~ ., data=df.train1,importance = TRUE, mtry=i)
  oob.values[i] <- temp.model$err.rate[nrow(temp.model$err.rate),1]
}
oob.values
## find the minimum error
min(oob.values)
## 2


oob.values <- vector(length=10)
for(i in 1:10) {
  temp.model <- randomForest(Target ~ ., data=df.train3, mtry=i)
  oob.values[i] <- temp.model$err.rate[nrow(temp.model$err.rate),1]
}
oob.values
## find the minimum error
min(oob.values)
## 2
```

Neural Network:

```
library(tidyverse)
library(caret)
library(neuralnet)
```

```
normalize = function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

dropout_clean <- read.csv("C:/Users/Mark/Desktop/Grad
        School/PDAT630/dropout_clean.csv")

#factor_cols=c("Marital.status","Scholarship.holder","Tuition.fees.up.to.date","Gender","Disp
        laced","Daytime.evening.attendance","Course","Previous.qualification","Mother.s.qual
        ification","Father.s.qualification","Mother.s.occupation","Father.s.occupation","Interna
        tional","Target")
#dropout_clean[factor_cols] <- lapply(dropout_clean[factor_cols], factor)
#dropout_clean_nn=mutate(dropout_clean, Target = factor(ifelse(Target == "Dropout", 0,
        1)))
dropout_clean_nn=mutate(dropout_clean, Target = ifelse(Target == "Dropout", 0, 1))

str(dropout_clean_nn)

nor = as.data.frame(lapply(dropout_clean_nn, normalize))


## Normal 1 Layer Neural Network
df.train <- sample_frac(nor, size = .85)
```

```
df.test <- setdiff(nor, df.train)
hid=1

nn=neuralnet(Target~., data = df.train, stepmax=1e7, hidden=hid,threshold = .1)
plot(nn)
pred=predict(nn, df.test, type="response")
pred=as.factor(ifelse(pred>.5, 1, 0))
print(confusionMatrix(as.factor(df.test$Target), pred)$overall[1])


## Normal 2 Layer Neural Network
df.train <- sample_frac(nor, size = .85)
df.test <- setdiff(nor, df.train)
hid2=4

nn2=neuralnet(Target~., data = df.train, stepmax=1e7, hidden=c(hid2,hid2),threshold = .1)
plot(nn2)
pred2=predict(nn2, df.test, type="response")
pred2=as.factor(ifelse(pred2>.5, 1, 0))
print(confusionMatrix(as.factor(df.test$Target), pred2)$overall[1])


## Testing The Nodes 1st layer
```

```
max_nodes=10
max_runs=10
accuracy_runs_1=data.frame(id=c(1:max_runs))

total_time <- proc.time()

for (i in 0:max_nodes){
 i_time=proc.time()
 acc_in=c()
 for (j in 1:max_runs){
        df.train <- sample_frac(nor, size = .85)
        df.test <- setdiff(nor, df.train)
        nn=neuralnet(Target~.,
        data = df.train,
        stepmax=1e7,
        hidden=i,
        threshold = .5)
        pred=predict(nn, df.test, type="response")
        pred=as.factor(ifelse(pred>.5, 1, 0))
        #print(confusionMatrix(as.factor(df.test$Survived), pred)$overall[1])
        acc_in=c(acc_in,confusionMatrix(as.factor(df.test$Target), pred)$overall[1])
        print(paste("node", i, "run", j))
 }
```

```
  print(paste(i, "took", proc.time()[3] - i_time)[3])
  accuracy_runs_1[paste(i, "nodes")] = acc_in
}
print(paste("whole thing took", proc.time()[3] - total_time)[3])

scores_1=data.frame(sapply(subset(accuracy_runs_1, select = -c(id)), function(x) mean(x)))
sd_1=data.frame(sapply(subset(accuracy_runs_1, select = -c(id)), function(x) sd(x)))

accuracy_1=data.frame(nodes=c(0:max_nodes))
accuracy_1["mean"]=scores_1

ggplot(data=accuracy_1, aes(x=nodes, y=mean))+
  geom_point()+
  geom_line()+
  labs(title="Average Accuracy Over 10 Runs For Different Number Of Nodes. 1 Layer
            ")+
  xlab("Number Of Nodes")+
  ylab("Average Accuracy")+
  scale_x_discrete(limits=factor(c(1:max_nodes)))+
  geom_hline(yintercept = max(accuracy_1$mean), color="red")+
  annotate("text",x=3,y=max(accuracy_1$mean)+.002
           ,label=round(max(accuracy_1$mean),2)
           ,color="red")
```

```
std_1=data.frame(nodes=c(0:max_nodes))
std_1["std"]=sd_1

ggplot(data=std_1, aes(x=nodes, y=std))+
  geom_point()+
  geom_line()+
  labs(title=max_runs)


## Testing The Nodes 2nd layer
max_nodes=10
max_runs=10
accuracy_runs_2=data.frame(id=c(1:max_runs))

total_time <- proc.time()

for (i in 1:max_nodes){
  i_time=proc.time()
  acc_in=c()
  for (j in 1:max_runs){
          df.train <- sample_frac(nor, size = .85)
```

```
        df.test <- setdiff(nor, df.train)
        nn=neuralnet(Target~.,
        data = df.train,
        stepmax=1e7,
        hidden=c(i,i),
        threshold = .1)
        pred=predict(nn, df.test, type="response")
        pred=as.factor(ifelse(pred>.5, 1, 0))
        #print(confusionMatrix(as.factor(df.test$Survived), pred)$overall[1])
        acc_in=c(acc_in,confusionMatrix(as.factor(df.test$Target), pred)$overall[1])
        print(paste("node", i, "run", j))
 }
 print(paste(i, "took", proc.time()[3] - i_time)[3])
 accuracy_runs_2[paste(i, "nodes")] = acc_in
}
print(paste("whole thing took", proc.time()[3] - total_time)[3])

scores_2=data.frame(sapply(subset(accuracy_runs_2, select = -c(id)), function(x) mean(x)))
sd_2=data.frame(sapply(subset(accuracy_runs_2, select = -c(id)), function(x) sd(x)))

accuracy_2=data.frame(nodes=c(1:max_nodes))
accuracy_2["mean"]=scores_2
```

```
ggplot(data=accuracy_2, aes(x=nodes, y=mean))+
  geom_point()+
  geom_line()+
  labs(title="Average Accuracy Over 10 Runs For Different Number Of Nodes. 2 Layers
            ")+
  xlab("Number Of Nodes")+
  ylab("Average Accuracy")+
  scale_x_discrete(limits=factor(c(1:max_nodes)))+
  geom_hline(yintercept = max(accuracy_2$mean), color="red")+
  annotate("text",x=3,y=max(accuracy_2$mean)+.002
           ,label=round(max(accuracy_2$mean),2)
           ,color="red")


std_2=data.frame(nodes=c(1:max_nodes))
std_2["std"]=sd_2


ggplot(data=std_2, aes(x=nodes, y=std))+
  geom_point()+
  geom_line()+
  labs(title=max_runs)



# ## Testing The Nodes 3rd layer
```

```
# max_nodes=10
# max_runs=10
# accuracy_runs_3=data.frame(id=c(1:max_runs))
#
# total_time <- proc.time()
#
# for (i in 1:max_nodes){
#   i_time=proc.time()
#   acc_in=c()
#   for (j in 1:max_runs){
#         df.train <- sample_frac(nor, size = .85)
#         df.test <- setdiff(nor, df.train)
#         nn=neuralnet(Target~.,
#                      data = df.train,
#                      stepmax=1e7,
#                      hidden=c(i,i,i),
#                      threshold = .1)
#         pred=predict(nn, df.test, type="response")
#         pred=as.factor(ifelse(pred>.5, 1, 0))
#         #print(confusionMatrix(as.factor(df.test$Survived), pred)$overall[1])
#         acc_in=c(acc_in,confusionMatrix(as.factor(df.test$Target), pred)$overall[1])
#         print(paste("node", i, "run", j))
#   }
```

```
#   print(paste(i, "took", proc.time()[3] - i_time)[3])
#   accuracy_runs_3[paste(i, "nodes")] = acc_in
# }
# print(paste("whole thing took", proc.time()[3] - total_time)[3])
#
# scores_3=data.frame(sapply(subset(accuracy_runs_3, select = -c(id)), function(x)
          mean(x)))
# sd_3=data.frame(sapply(subset(accuracy_runs_3, select = -c(id)), function(x) sd(x)))
#
# accuracy_3=data.frame(nodes=c(1:max_nodes))
# accuracy_3["mean"]=scores_3
#
# ggplot(data=accuracy_3, aes(x=nodes, y=mean))+
#   geom_point()+
#   geom_line()+
#   labs(title="Average Accuracy Over 10 Runs For Different Number Of Nodes. 3 Layers
#         ")+
#   xlab("Number Of Nodes")+
#   ylab("Average Accuracy")+
#   scale_x_discrete(limits=factor(c(1:max_nodes)))
#
# std_3=data.frame(nodes=c(1:max_nodes))
# std_3["std"]=sd_3
```

```
#
# ggplot(data=std_3, aes(x=nodes, y=std))+
#   geom_point()+
#   geom_line()+
#   labs(title=max_runs)
```

Logistic Regression:

```
library(tidyverse)
library(caret)

dropout_clean <- read.csv("C:/Users/Mark/Desktop/Grad
        School/PDAT630/dropout_clean.csv")
factor_cols=c("Marital.status","Scholarship.holder","Tuition.fees.up.to.date","Gender","Displ
        aced","Daytime.evening.attendance","Course","Previous.qualification","Mother.s.quali
        fication","Father.s.qualification","Mother.s.occupation","Father.s.occupation","Internat
        ional","Target")
dropout_clean[factor_cols] <- lapply(dropout_clean[factor_cols], factor)
dropout_clean_lg=mutate(dropout_clean, Target = as.factor(ifelse(Target == "Dropout", 0,
        1)))
str(dropout_clean_lg)
```

```
df.train <- sample_frac(dropout_clean_lg, size = .85)

df.test <- setdiff(dropout_clean_lg, df.train)


## Logistic Regression

logreg = glm(Target~., family = "binomial", data = df.train)

coef(logreg)

summary(logreg)

pred=predict(logreg, df.test, type="response")

pred=as.factor(ifelse(pred>.5, 1, 0))

confusionMatrix(as.factor(df.test$Target), pred)$overall[1]



## Stepwise

logreg.step=step(logreg)

coef(logreg.step)

summary(logreg.step)

pred.step=predict(logreg.step, df.test, type="response")

pred.step=as.factor(ifelse(pred.step>.5, 1, 0))

confusionMatrix(as.factor(df.test$Target), pred.step)$overall[1]
```

Logistic Regression Models:

logistic regression

Call:

glm(formula = Target ~ ., family = "binomial", data = df.train)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.7209	-0.4930	0.4366	0.6866	2.6785

Coefficients: (1 not defined because of singularities)

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.877e+00	5.788e-01	-3.243	0.001183	**
Gender1	-6.462e-01	9.965e-02	-6.485	8.87e-11	***
Displaced1	9.118e-02	9.690e-02	0.941	0.346749	
Tuition.fees.up.to.date1	2.920e+00	1.647e-01	17.730	< 2e-16	***
Scholarship.holder1	1.078e+00	1.280e-01	8.421	< 2e-16	***
Marital.status2	5.999e-01	1.910e-01	3.142	0.001679	**
Marital.status4	4.588e-01	3.267e-01	1.404	0.160206	
Marital.status5	8.561e-01	5.790e-01	1.479	0.139225	
Marital.status7	-4.307e-01	8.998e-01	-0.479	0.632175	
Daytime.evening.attendance1	-6.151e-01	8.117e-01	-0.758	0.448593	

Course2	6.163e-01	8.186e-01	0.753	0.451528	
Course3	1.151e+00	2.683e-01	4.289	1.80e-05	***
Course4	1.240e+00	8.213e-01	1.510	0.130963	
Course5	1.140e+00	8.248e-01	1.383	0.166806	
Course6	1.130e+00	8.155e-01	1.386	0.165645	
Course7	2.330e-01	8.215e-01	0.284	0.776710	
Course8	-1.323e-01	8.261e-01	-0.160	0.872759	
Course9	6.842e-01	8.085e-01	0.846	0.397436	
Course10	1.370e+00	8.179e-01	1.674	0.094047	.
Course11	4.190e-01	8.149e-01	0.514	0.607148	
Course12	1.420e+00	8.069e-01	1.760	0.078432	.
Course13	9.607e-02	8.446e-01	0.114	0.909434	
Course14	8.072e-01	8.150e-01	0.990	0.322007	
Course15	1.039e+00	8.151e-01	1.275	0.202410	
Course16	-9.868e-02	8.213e-01	-0.120	0.904366	
Course17	NA	NA	NA	NA	
Previous.qualification2	-1.090e+00	5.780e-01	-1.886	0.059330	.
Previous.qualification3	-7.080e-01	2.429e-01	-2.915	0.003559	**
Previous.qualification6	-7.674e-01	7.814e-01	-0.982	0.326060	
Previous.qualification7	-1.435e+01	3.991e+02	-0.036	0.971307	
Previous.qualification9	-2.307e-01	4.121e-01	-0.560	0.575543	
Previous.qualification12	-1.050e+00	2.405e-01	-4.366	1.27e-05	***
Previous.qualification14	-3.045e-02	2.159e-01	-0.141	0.887860	

Previous.qualification15	-7.641e-02	4.059e-01	-0.188	0.850676
Previous.qualification16	1.269e+00	6.140e-01	2.067	0.038705 *
Previous.qualification18	6.699e-03	4.889e-01	0.014	0.989069
Mother.s.qualification2	7.030e-01	3.607e-01	1.949	0.051304 .
Mother.s.qualification3	6.127e-02	2.040e-01	0.300	0.763965
Mother.s.qualification4	1.098e+00	5.244e-01	2.093	0.036320 *
Mother.s.qualification5	3.464e-02	5.775e-01	0.060	0.952175
Mother.s.qualification10	-1.367e+00	3.985e-01	-3.429	0.000606 ***
Mother.s.qualification13	-1.181e-01	1.403e-01	-0.841	0.400184
Mother.s.qualification19	-9.642e-01	5.273e-01	-1.828	0.067476 .
Mother.s.qualification22	-1.231e-01	1.697e-01	-0.725	0.468420
Mother.s.qualification23	1.682e-02	1.763e-01	0.095	0.923968
Mother.s.qualification35	-5.140e-01	3.577e-01	-1.437	0.150734
Father.s.qualification2	9.739e-04	3.742e-01	0.003	0.997923
Father.s.qualification3	1.921e-02	2.281e-01	0.084	0.932893
Father.s.qualification4	-3.599e-01	4.425e-01	-0.813	0.416054
Father.s.qualification5	-9.603e-01	6.343e-01	-1.514	0.130032
Father.s.qualification9	1.582e+00	9.562e-01	1.654	0.098090 .
Father.s.qualification10	5.802e-01	4.824e-01	1.203	0.229076
Father.s.qualification14	1.534e-01	1.411e-01	1.088	0.276802
Father.s.qualification24	-2.441e-01	5.878e-01	-0.415	0.677997
Father.s.qualification27	-3.433e-02	1.599e-01	-0.215	0.829970
Father.s.qualification28	1.665e-01	1.667e-01	0.999	0.317876

Father.s.qualification29	1.414e-01	7.079e-01	0.200	0.841662
Father.s.qualification35	-5.384e-01	4.104e-01	-1.312	0.189603
Mother.s.occupation2	1.153e+00	5.094e-01	2.263	0.023641 *
Mother.s.occupation3	1.223e+00	4.735e-01	2.584	0.009776 **
Mother.s.occupation4	1.544e+00	4.529e-01	3.410	0.000650 ***
Mother.s.occupation5	1.222e+00	4.314e-01	2.832	0.004619 **
Mother.s.occupation6	1.239e+00	4.349e-01	2.849	0.004391 **
Mother.s.occupation7	1.351e+00	5.360e-01	2.522	0.011682 *
Mother.s.occupation8	1.036e+00	4.584e-01	2.260	0.023800 *
Mother.s.occupation9	1.319e-01	5.815e-01	0.227	0.820576
Mother.s.occupation10	1.449e+00	4.305e-01	3.366	0.000762 ***
Mother.s.occupation12	4.764e-01	5.791e-01	0.823	0.410736
Mother.s.occupation13	1.653e-01	1.247e+00	0.133	0.894549
Mother.s.occupation29	1.519e+01	2.653e+02	0.057	0.954340
Mother.s.occupation32	1.566e+00	1.948e+00	0.804	0.421514
Mother.s.occupation47	2.385e+00	1.289e+00	1.850	0.064343 .
Father.s.occupation2	-2.801e-01	5.132e-01	-0.546	0.585253
Father.s.occupation3	-9.526e-02	5.152e-01	-0.185	0.853313
Father.s.occupation4	-2.720e-01	4.718e-01	-0.577	0.564259
Father.s.occupation5	-6.000e-01	4.682e-01	-1.281	0.200034
Father.s.occupation6	-3.182e-01	4.625e-01	-0.688	0.491338
Father.s.occupation7	-2.307e-02	5.000e-01	-0.046	0.963202
Father.s.occupation8	-4.372e-01	4.635e-01	-0.943	0.345467

Father.s.occupation9 -3.052e-01 4.777e-01 -0.639 0.522882

Father.s.occupation10 -4.616e-01 4.631e-01 -0.997 0.318873

Father.s.occupation11 -3.309e-01 4.824e-01 -0.686 0.492769

Father.s.occupation12 -8.773e-01 6.140e-01 -1.429 0.153059

Father.s.occupation13 2.086e-02 1.101e+00 0.019 0.984888

Father.s.occupation44 -5.555e-02 1.920e+00 -0.029 0.976913

Father.s.occupation47 8.868e-01 1.527e+00 0.581 0.561512

Age.at.enrollment -4.652e-02 9.004e-03 -5.166 2.39e-07 ***

International1 6.253e-01 3.068e-01 2.038 0.041546 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 4727.2 on 3759 degrees of freedom

Residual deviance: 3356.8 on 3673 degrees of freedom

AIC: 3530.8

Number of Fisher Scoring iterations: 14

Stepwise logistic regression

```
Call:
glm(formula = Target ~ Gender + Tuition.fees.up.to.date + Scholarship.holder +
    Marital.status + Course + Previous.qualification + Mother.s.qualification +
    Mother.s.occupation + Age.at.enrollment + International,
    family = "binomial", data = df.train)

Deviance Residuals:
    Min      1Q  Median    3Q      Max
-2.6312  -0.5075   0.4519   0.6940   2.6948

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            -2.772932   0.931505  -2.977 0.002912 **
Gender1                -0.649904   0.098613  -6.590 4.39e-11 ***
Tuition.fees.up.to.date1  2.908970   0.163324  17.811  < 2e-16 ***
Scholarship.holder1     1.060209   0.126421   8.386  < 2e-16 ***
Marital.status2         0.578150   0.189673   3.048 0.002303 **
Marital.status4         0.466094   0.326154   1.429 0.152986
Marital.status5         0.855517   0.578019   1.480 0.138851
Marital.status7        -0.535709   0.894630  -0.599 0.549303
Course2                 0.795716   0.827549   0.962 0.336284
Course3                 1.914676   0.836789   2.288 0.022130 *
Course4                 1.439238   0.831950   1.730 0.083638 .
```

Course5	1.320567	0.834182	1.583	0.113406	
Course6	1.263031	0.825243	1.530	0.125894	
Course7	0.411988	0.831347	0.496	0.620200	
Course8	0.040810	0.835920	0.049	0.961062	
Course9	0.838188	0.819193	1.023	0.306219	
Course10	1.517627	0.828300	1.832	0.066919	.
Course11	0.601658	0.824882	0.729	0.465765	
Course12	1.580003	0.817831	1.932	0.053367	.
Course13	0.277921	0.853071	0.326	0.744584	
Course14	0.963694	0.826338	1.166	0.243525	
Course15	1.228296	0.825280	1.488	0.136662	
Course16	0.110882	0.831079	0.133	0.893862	
Course17	0.726624	0.823097	0.883	0.377348	
Previous.qualification2	-1.106402	0.581646	-1.902	0.057146	.
Previous.qualification3	-0.747084	0.239674	-3.117	0.001827	**
Previous.qualification6	-0.939690	0.780915	-1.203	0.228853	
Previous.qualification7	-14.382401	403.857130	-0.036	0.971591	
Previous.qualification9	-0.278032	0.411565	-0.676	0.499327	
Previous.qualification12	-1.081893	0.238335	-4.539	5.64e-06	***
Previous.qualification14	-0.038648	0.213061	-0.181	0.856057	
Previous.qualification15	-0.021711	0.404765	-0.054	0.957223	
Previous.qualification16	1.339443	0.608285	2.202	0.027665	*
Previous.qualification18	-0.038032	0.481562	-0.079	0.937051	

Mother.s.qualification2	0.671144	0.354828	1.891	0.058563	.
Mother.s.qualification3	0.074608	0.198175	0.376	0.706563	
Mother.s.qualification4	0.962836	0.504716	1.908	0.056433	.
Mother.s.qualification5	-0.231653	0.543469	-0.426	0.669927	
Mother.s.qualification10	-1.261405	0.381888	-3.303	0.000956	***
Mother.s.qualification13	-0.097214	0.135444	-0.718	0.472916	
Mother.s.qualification19	-1.028163	0.415938	-2.472	0.013439	*
Mother.s.qualification22	-0.171448	0.151925	-1.129	0.259106	
Mother.s.qualification23	0.038194	0.167603	0.228	0.819738	
Mother.s.qualification35	-0.594313	0.327120	-1.817	0.069247	.
Mother.s.occupation2	1.066261	0.447330	2.384	0.017144	*
Mother.s.occupation3	1.140298	0.411144	2.773	0.005546	**
Mother.s.occupation4	1.457368	0.390251	3.734	0.000188	***
Mother.s.occupation5	1.092941	0.369768	2.956	0.003119	**
Mother.s.occupation6	1.159236	0.372727	3.110	0.001870	**
Mother.s.occupation7	1.457640	0.469646	3.104	0.001911	**
Mother.s.occupation8	0.917585	0.394316	2.327	0.019964	*
Mother.s.occupation9	0.028117	0.529360	0.053	0.957641	
Mother.s.occupation10	1.328896	0.361097	3.680	0.000233	***
Mother.s.occupation12	0.134438	0.458410	0.293	0.769316	
Mother.s.occupation13	0.167364	0.725505	0.231	0.817558	
Mother.s.occupation29	16.237570	268.877060	0.060	0.951845	
Mother.s.occupation32	2.370206	1.177645	2.013	0.044150	*

Mother.s.occupation47 3.180370 0.717200 4.434 9.23e-06 ***

Age.at.enrollment -0.047632 0.008777 -5.427 5.74e-08 ***

International1 0.575598 0.301646 1.908 0.056367 .

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 4727.2 on 3759 degrees of freedom

Residual deviance: 3382.2 on 3700 degrees of freedom

AIC: 3502.2

Number of Fisher Scoring iterations: 14