



# Using Machine Learning to Analyze CSULB Master's Students' Time to Degree

Report Prepared by: Noppakan Sirikul

May 1, 2024

## Table of Contents

<b>Introduction .....</b>	<b>2</b>
<b>Background .....</b>	<b>3</b>
<b>Data .....</b>	<b>3</b>
<b>Results.....</b>	<b>4</b>
<b>Conclusion.....</b>	<b>6</b>
<b>Appendix .....</b>	<b>7</b>
<b>A. Gradient Boosting.....</b>	<b>7</b>
a. SAS Code and Results .....	7
b. R Code and Results .....	9
c. Python Code and Results .....	10
<b>B. Random Forest.....</b>	<b>11</b>
a. SAS Code and Results .....	11
b. R Code and Results .....	13
c. Python Code and Results .....	14
<b>C. Decision Tree.....</b>	<b>15</b>
a. SAS Code and Results .....	15
b. R Code and Results .....	17
c. Python Code and Results .....	19
<b>D. kNN.....</b>	<b>20</b>
a. SAS Code and Results .....	20
b. R Code and Results .....	22
c. Python Code and Results .....	23
<b>References .....</b>	<b>24</b>

## Introduction

One way that institutions often measure their student success is by analyzing graduation, or time-to-degrees rates. As a CSULB graduate student myself, understanding what factors impact my time to degree can help me better navigate my graduate school experience. By training models using Gradient Boosting, Random Forest, Decision Tree, and kNN, on student demographics, academic background, and academic performance, I hope to see what contributors most impact Master's students' graduation rates.

## Background

This past year, I have had the opportunity to be a part of a program on campus called Data Fellows. Data Fellows is composed of teams from different colleges and units within CSULB that work with institutional data to drive change at the university and college levels. My team members included Dr. Seungjoon Lee, Associate Professor of Applied Statistics, Dr. Dina Perrone, Associate Dean of Graduate Studies, Dr. Nancy Hall, Chair and Professor of Linguistics, and myself. Our project focused on Graduate Studies and our research question was “What factors impact graduate student time to degree?”. Because our undergraduate population is larger than our graduate, CSULB efforts are mostly centered around undergraduate students. Since all the other Data Fellows teams were focused on undergraduate studies, our team wanted to shed light on the graduate studies experience. We hope that by identifying areas where graduate students struggle, we can provide additional support and resources. We also hope that advisors can use these findings to improve their program structure, possibly adopting practices from other programs. Lastly, while time to degree doesn’t entirely measure student success, it can be a factor that students consider when looking back on their experience and can help institutions identify ways to improve their graduation rates.

## Data

My dataset was pulled from IR&A (Institutional Research and Analytics) for use in the Data Fellows project I was a part of about Graduate Studies at CSULB. The original dataset had 16,513 rows and 170 columns. I decided to limit my dataset to only Master’s students in stateside programs who entered between Fall 2016-Fall 2021 and have already graduated. In addition, I created new columns by combining certain ones together, like gender identity and gender expression as gender, as well as college, department, and major as master’s program to limit

multicollinearity. I also decided to only keep columns about academic performance in the first and second semester since the majority of students do not spend beyond four semesters in their Master's program. After omitting rows with any N/A values, I was left with 5,026 rows and 28 columns. I then built Gradient Boosting, Random Forest, Decision Tree, and kNN regression models with years to degree as the target variable and the following input variables: race/ethnicity, domestic/international status, first gen status, age, income level, family size, gender, sexual orientation, tuition type, military status, master's program, cohort term, pre-pandemic/pandemic cohort term, culminating activity, Bachelor institution, first and second semester academic standings, first and second semester GPAs, units taken during the first and second semesters, units passed during the first and second semesters, and first and second semester pass rates.

## Results

My Gradient Boosting (XGBoost) model in SAS resulted in prediction accuracies of 34.74%, 50.90%, and 70.53% within 10%, 15%, and 20% of the testing data, respectively. In R, my model reached prediction accuracies of 47.04%, 61.56%, and 72.83%, which performed slightly better than SAS. In Python, my prediction accuracies were 53.98%, 65.81%, and 74.75%, which were the highest I was able to achieve with the Gradient Boosting model.

In terms of variable importance, the top five in SAS were master's program, units taken during the first semester, cohort term, culminating activity, and admission GPA. Similarly, the top five in R were master's program, units taken during the second semester, culminating activity, units taken during the first semester, and cohort term. Lastly, the top five in Python were master's program, units taken during the second semester, culminating activity, cohort term, and units taken during the first semester. Amongst all three programs, there were almost identical

factors that came out on top, with just a slight variation in order, for our Gradient Boosting model.

My Random Forest model in SAS achieved prediction accuracies of 50.23%, 63%, and 71.58% within 10%, 15%, and 20% of the testing data, respectively. In R, my model reached prediction accuracies of 18.2%, 26.98%, and 54.5%, which performed significantly worse than SAS. In Python, my prediction accuracies were 50.1%, 61.53%, and 72.47%, close to our results in SAS. Like with Gradient Boosting, Python was able to produce the highest prediction accuracies for our Random Forest model.

In terms of variable importance, the top five in SAS were master's program, culminating activity, units taken during the second semester, units passed during the second semester, and units passed during the first semester. In R, the top five were master's program, cohort term, second semester GPA, second semester pass rate, and first semester GPA. Lastly, the top five in Python were master's program, culminating activity, units taken during the second semester, units passed during the second semester, and admission GPA. Unlike with Gradient Boosting, we see that units passed, pass rate, and GPA all emerge as some of the most important features in our Random Forest model.

My Decision Tree model in SAS achieved prediction accuracies of 71.25%, 74.59%, and 78.87% within 10%, 15%, and 20% of the testing data, respectively. This was surprising to me, since none of my other models consistently had above 70%. In R, my model reached prediction accuracies of 47.13%, 51.28%, and 68.48%, which performed significantly worse than SAS. In Python, my prediction accuracies were 34.49%, 58.25%, and 64.81%. This means that SAS was able to produce the highest prediction accuracies with my Decision Tree model. This is the only model where SAS performed the best since the others were all in Python. One issue I ran into

with the Decision Tree diagrams in R, however, were that they were nearly impossible to interpret due to the number of categorical variables. They are the most readable in Python.

For my kNN model, I used 26 as the number of neighbors which resulted in an MSE of 0.369. SAS achieved prediction accuracies of 32.78%, 47.37%, and 62.63% within 10%, 15%, and 20% of the testing data, respectively. In R, my model reached prediction accuracies of 20.06%, 37.55%, and 53.26%, which performed worse than SAS. In Python, my prediction accuracies were 43.94%, 56.66%, and 66.8%, which were the highest accuracies I was able to achieve with kNN model. Something I liked about the kNN diagrams is how you can clearly see that when an actual value goes up, the predicted value also goes up. The trend is interpretable.

## Conclusion

Overall, when using machine learning models on graduate student data, Python seems to perform the best given that it often had the highest prediction accuracies. From the results, it seems that the most important features that predict time to degree are master's program, culminating activity, cohort term, and the number of units taken during the first semester. Since most master's programs vary between one to three years long, if a program is only intended to be one year long, it makes sense that students would graduate within one to two years. Versus if a program is intended to be two years long, it makes sense that students would graduate between two to three years. In terms of culminating activity, a master's thesis or a graduate project span over multiple semesters, so students who do either of these may take longer to graduate versus students who take the comprehensive exams. Cohort term could also be important for two reasons 1) some programs only accept incoming students during the fall semester and 2) some courses are only offered during certain semesters, so students may take longer to graduate because they are waiting to take a required course. Lastly, since master's programs are short,

every semester is significant, and the choices we make in the beginning can impact the rest of our time during our studies. This may be why the number of units taken during the first semester is important.

Regarding future work that I would potentially do on this project, I would first look into each Master's program individually. All our programs have different structures and varying curriculums, so naturally this would be an important predictor for time to degree. I would also want to investigate the students who entered between Fall 2016-Fall 2021 but have not yet graduated. Since I only focused my project on students who already graduated, it would be interesting to see if those students were all in a specific program/college or were from a similar demographic. Aside from demographics and academics, it would also be great to have data on what else graduate students are involved in besides their program, like jobs, internships, and TAs. These involvements could impact their time to degree, but make for a more fulfilling graduate school experience, which is why graduation rates cannot entirely define student success. Lastly, conducting a binary model of graduating within three years or not could result in different findings. I only focused on doing regression models, but doing a binary classification could be a way of looking at "traditional" vs. "non-traditional" time to degree.

## Appendix

### A. Gradient Boosting

#### a. SAS Code and Results

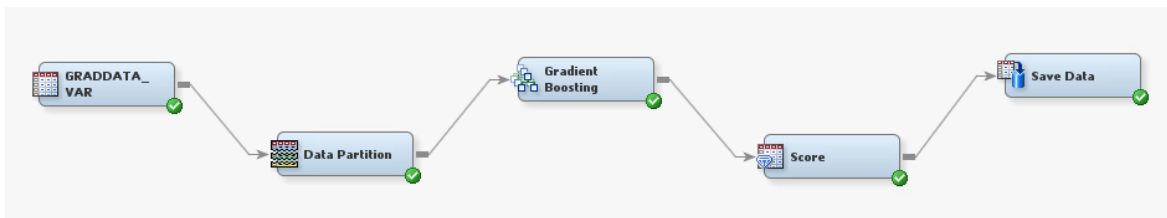


Figure 1: SAS Gradient Boosting Enterprise Miner Diagram

```

/*GRADIENT BOOSTING*/

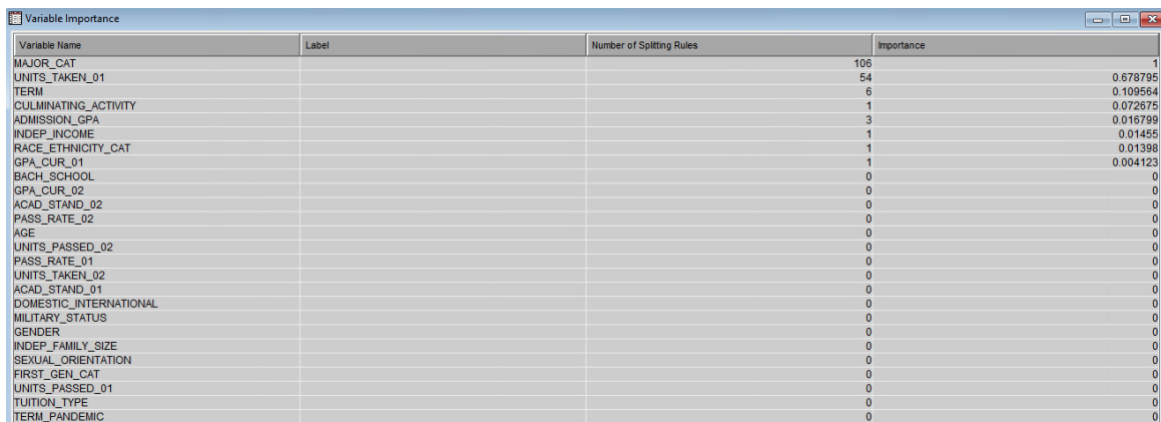
/*IMPORT DATAFILE*/
proc import out=sasuser.graddata_var
datafile="//vdi-filesshare02/UEMprofiles/032283262/Desktop/STAT 574/Project/Graduate Student Data_VAR.csv"
dbms=csv replace;
run;

/*DETERMINE 10%, 15%, AND 20% ACCURACY*/
data accuracy;
set tmp1.em_save_test;
ind10=(abs(R_years_to_degree)<0.10*(years_to_degree));
ind15=(abs(R_years_to_degree)<0.15*(years_to_degree));
ind20=(abs(R_years_to_degree)<0.20*(years_to_degree));
run;

proc sql;
select sum(ind10)/count(*) as accuracy10, sum(ind15)/count(*) as accuracy15, sum(ind20)/count(*) as accuracy20
from accuracy;
quit;

```

Figure 2: SAS Gradient Boosting Code



Variable Name	Label	Number of Splitting Rules	Importance
MAJOR_CAT		106	1
UNITS_TAKEN_01		54	0.678795
TERM		6	0.109564
CULMINATING_ACTIVITY		1	0.072675
ADMISSION_GPA		3	0.016799
INDEP_INCOME		1	0.01455
RACE_ETHNICITY_CAT		1	0.01398
GPA_CUR_01		1	0.004123
BACH_SCHOOL		0	0
GPA_CUR_02		0	0
ACAD_STAND_02		0	0
PASS_RATE_02		0	0
AGE		0	0
UNITS_PASSED_02		0	0
PASS_RATE_01		0	0
UNITS_TAKEN_02		0	0
ACAD_STAND_01		0	0
DOMESTIC_INTERNATIONAL		0	0
MILITARY_STATUS		0	0
GENDER		0	0
INDEP_FAMILY_SIZE		0	0
SEXUAL_ORIENTATION		0	0
FIRST_GEN_CAT		0	0
UNITS_PASSED_01		0	0
TUITION_TYPE		0	0
TERM_PANDEMIC		0	0

Figure 3: SAS Gradient Boosting Variable Importance

accuracy10	accuracy15	accuracy20
34.73684	50.90226	70.52632

Figure 4: SAS Gradient Boosting Prediction Accuracies



## b. R Code and Results

```

1  #IMPORT DATAFILE
2  grad_student.data<- read.csv(file="/Users/nappakansirikul/Desktop/STAT 574/PROJECT/DATA/Graduate Student Data_VAR.csv", header=TRUE, sep=",")
3  grad_student.data<-na.omit(grad_student.data)
4
5  #SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
6  set.seed(902847)
7  sample <- sample(c(TRUE, FALSE), nrow(grad_student.data), replace=TRUE, prob=c(0.8,0.2))
8  train<- grad_student.data[sample,]
9  test<- grad_student.data[!sample,]
10
11  train.x<- data.matrix(train[,-28])
12  train.y<- data.matrix(train[,28])
13  test.x<- data.matrix(test[,-28])
14  test.y<- data.matrix(test[,28])
15
16  #INSTALL PACKAGES
17  #install.packages("xgboost")
18  library(xgboost)
19
20  #FITTING EXTREME GRADIENT BOOSTED REGRESSION TREE
21  xgb.reg<- xgboost(data=train.x, label=train.y,
22                  max.depth=6, eta=0.01, subsample=0.8, colsample_bytree=0.5,
23                  nrounds=1000, objective="reg:linear")
24  #eta=learning rate
25  #colsample_bytree defines what percentage of features (columns)
26  #will be used for building each tree
27
28  #DISPLAYING FEATURE IMPORTANCE
29  print(xgb.importance(colnames(train.x), model=xgb.reg))
30
31  #COMPUTING PREDICTION ACCURACY FOR TESTING DATA
32  pred.y<- predict(xgb.reg, test.x)
33
34  #accuracy within 10%
35  accuracy10<- ifelse(abs(test.y-pred.y)<0.1*test.y,1,0)
36  accuracy10_round<-round((mean(accuracy10)*100),2))
37  cat("accuracy10=",accuracy10_round,"%")
38
39  #accuracy within 15%
40  accuracy15<- ifelse(abs(test.y-pred.y)<0.15*test.y,1,0)
41  accuracy15_round<-round((mean(accuracy15)*100),2))
42  cat("accuracy15=",accuracy15_round,"%")
43
44  #accuracy within 20%
45  accuracy20<- ifelse(abs(test.y-pred.y)<0.2*test.y,1,0)
46  accuracy20_round<-round((mean(accuracy20)*100),2))
47  cat("accuracy20=",accuracy20_round,"%")
48
49

```

Figure 5: R Gradient Boosting Code

	Feature	Gain	Cover	Frequency
1:	MAJOR_CAT	0.240547975	0.230755277	0.123681020
2:	UNITS_TAKEN_02	0.109978197	0.065273634	0.043891159
3:	CULMINATING_ACTIVITY	0.082952088	0.037629748	0.032688249
4:	UNITS_TAKEN_01	0.076049613	0.057622283	0.046158682
5:	TERM	0.062904502	0.057433064	0.063827399
6:	UNITS_PASSED_02	0.062753590	0.040644987	0.034574110
7:	STUDENT_ID	0.058315205	0.102379067	0.136926945
8:	UNITS_PASSED_01	0.055060648	0.037267844	0.033227067
9:	ADMISSION_GPA	0.047974269	0.092453965	0.099815904
10:	AGE	0.030052734	0.060307641	0.070248305
11:	GPA_CUR_01	0.027165905	0.025036186	0.034596561
12:	GPA_CUR_02	0.026929784	0.027850263	0.034596561
13:	BACH_SCHOOL	0.019469024	0.021490114	0.029634951
14:	INDEP_INCOME	0.014426169	0.020729381	0.035427237
15:	PASS_RATE_02	0.013075937	0.014087594	0.011719276
16:	PASS_RATE_01	0.011497255	0.014405636	0.014054151
17:	RACE_ETHNICITY_CAT	0.010343912	0.010444253	0.031655516
18:	INDEP_FAMILY_SIZE	0.008375298	0.016854992	0.025863230
19:	GENDER	0.008180059	0.012764421	0.015895110
20:	TERM_PANDEMIC	0.007608177	0.005050276	0.014929729
21:	FIRST_GEN_CAT	0.007391783	0.007352764	0.022405819
22:	SEXUAL_ORIENTATION	0.006987934	0.008902289	0.017040097
23:	ACAD_STAND_02	0.004386486	0.011591142	0.007723048
24:	MILITARY_STATUS	0.002580927	0.005129812	0.006061695
25:	ACAD_STAND_01	0.002095713	0.006319155	0.005186116
26:	TUITION_TYPE	0.001501538	0.007210641	0.004175834
27:	DOMESTIC_INTERNATIONAL	0.001395276	0.003013570	0.003996228
	Feature	Gain	Cover	Frequency

Figure 6: R Gradient Boosting Variable Importance

```

> cat("accuracy10=",accuracy10_round,"%")
accuracy10= 47.04 %
> cat("accuracy15=",accuracy15_round,"%")
accuracy15= 61.56 %
> cat("accuracy20=",accuracy20_round,"%")
accuracy20= 72.83 %

```

Figure 7: R Gradient Boosting Prediction Accuracies

### c. Python Code and Results

```
In [4]: #IMPORT PACKAGES
import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor

#IMPORT DATAFILE
graddata=pandas.read_csv('Graduate Student Data_MAJ2.csv')
graddata=graddata.dropna()

#RECODE CATEGORICAL VARIABLES
coding_term={'Fall 2016':0,'Fall 2017':1,'Fall 2018':2,'Fall 2019':3,'Fall 2020':4,'Fall 2021':5,
'Spring 2017':6,'Spring 2018':7,'Spring 2019':8,'Spring 2020':9,'Spring 2021':10}
graddata['TERM']=graddata['TERM'].map(coding_term)

coding_pandemic={'PRE-PANDEMIC':0,'PANDEMIC':1}
graddata['TERM_PANDEMIC']=graddata['TERM_PANDEMIC'].map(coding_pandemic)

coding_race={'Asian':0,'Black or African American':1,'Hispanic/Latino':2,'White':3,'Other':4}
graddata['RACE_ETHNICITY_CAT']=graddata['RACE_ETHNICITY_CAT'].map(coding_race)

coding_visa={'DOMESTIC':0,'INTERNATIONAL':1}
graddata['DOMESTIC_INTERNATIONAL']=graddata['DOMESTIC_INTERNATIONAL'].map(coding_visa)

coding_firstgen={'NOT FIRST GEN':0,'FIRST GEN':1,'UNKNOWN':2}
graddata['FIRST_GEN_CAT']=graddata['FIRST_GEN_CAT'].map(coding_firstgen)

coding_income={'LESS THAN $6K PER YEAR':0,$6,000 TO $ 11,999':1,$12,000 TO $ 23,999':2,$24,000 TO $ 35,999':3,
'$36,000 TO $ 47,999':4,$48,000 TO $ 59,999':5,$60,000 OR MORE PER YEAR':6,'NO RESPONSE':7}
graddata['INDEP_INCOME']=graddata['INDEP_INCOME'].map(coding_income)

coding_gender={'CIS FEMALE':0,'CIS MALE':1,'NON-BINARY/TRANS':2}
graddata['GENDER']=graddata['GENDER'].map(coding_gender)

coding_sexualorientation={'STRAIGN/HETEROSEXUAL':0,'QUEER':1,'DECLINE TO SAY':2}
graddata['SEXUAL_ORIENTATION']=graddata['SEXUAL_ORIENTATION'].map(coding_sexualorientation)

coding_tuition={'IN-STATE':0,'OUT-OF-STATE':1,'OTHER':2}
graddata['TUITION_TYPE']=graddata['TUITION_TYPE'].map(coding_tuition)

coding_military={'NOT A US MILITARY SERVICE MEMBER':0,'MILITARY SERVICE MEMBER':1}
graddata['MILITARY_STATUS']=graddata['MILITARY_STATUS'].map(coding_military)

coding_acad1={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3}
graddata['ACAD_STAND_01']=graddata['ACAD_STAND_01'].map(coding_acad1)

coding_acad2={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3,
'Disqualified':4}
graddata['ACAD_STAND_02']=graddata['ACAD_STAND_02'].map(coding_acad2)

coding_activity={'Master's Thesis':0,'Graduate Project':1,'Comprehensive Exams':2,'Unknown':3}
graddata['CULMINATING_ACTIVITY']=graddata['CULMINATING_ACTIVITY'].map(coding_activity)

coding_bachschool={'CSULB':0,'Other CSU':1,'Other College in the US':2,'UC':3,'International':4}
graddata['BACH_SCHOOL']=graddata['BACH_SCHOOL'].map(coding_bachschool)

x=graddata.iloc[:,1:27].values
y=graddata.iloc[:,27].values

#SPLIT DATA INTO 80% TRAINING AND 20% TESTING SETS
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.20, random_state=902487)

#FIT GRADIENT BOOSTED REGRESSION TREE
gbreg_params={'n_estimators':1000, 'max_depth':6, 'learning_rate':0.01, 'loss':'squared_error'}
gb_reg=GradientBoostingRegressor(**gbreg_params)
gb_reg.fit(x_train,y_train)

#DISPLAY VARIABLE IMPORTANCE
var_names=pandas.DataFrame(['TERM','TERM_PANDEMIC','MAJOR_CAT_NUM','RACE_ETHNICITY_CAT','DOMESTIC_INTERNATIONAL',
'FIRST_GEN_CAT','AGE','INDEP_INCOME','INDEP_FAMILY_SIZE','GENDER','SEXUAL_ORIENTATION',
'TUITION_TYPE','MILITARY_STATUS','ACAD_STAND_01','ACAD_STAND_02','ADMISSION_GPA',
'CULMINATING_ACTIVITY','GPA_CUR_01','GPA_CUR_02','UNITS_TAKEN_01','UNITS_TAKEN_02',
'UNITS_PASSED_01','UNITS_PASSED_02','PASS_RATE_01','PASS_RATE_02','BACH_SCHOOL'],
columns=['var_name'])

loss_reduction=pandas.DataFrame(gb_reg.feature_importances_, columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0, ascending=False)
print(var_importance)

#COMPUTE PREDICTION ACCURACY FOR TESTING DATA
y_pred=gb_reg.predict(x_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred,y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#ACCURACY WITHIN 10%
accuracy10=sum(ind10)/len(ind10)
print("accuracy10=",round(accuracy10*100,2),'%')

#ACCURACY WITHIN 15%
accuracy15=sum(ind15)/len(ind15)
print("accuracy15=",round(accuracy15*100,2),'%')

#ACCURACY WITHIN 20%
accuracy20=sum(ind20)/len(ind20)
print("accuracy20=",round(accuracy20*100,2),'%')
```

Figure 8: Python Gradient Boosting Code

	var_name	loss_reduction
2	MAJOR_CAT_NUM	0.264933
20	UNITS_TAKEN_02	0.207534
16	CULMINATING_ACTIVITY	0.123829
0	TERM	0.062221
19	UNITS_TAKEN_01	0.049247
15	ADMISSION_GPA	0.047168
18	GPA_CUR_02	0.029525
6	AGE	0.028925
22	UNITS_PASSED_02	0.024718
21	UNITS_PASSED_01	0.023412
25	BACH_SCHOOL	0.021272
17	GPA_CUR_01	0.019147
7	INDEP_INCOME	0.014609
24	PASS_RATE_02	0.012036
3	RACE_ETHNICITY_CAT	0.009698
1	TERM_PANDEMIC	0.009317
23	PASS_RATE_01	0.008981
9	GENDER	0.008936
5	FIRST_GEN_CAT	0.008561
8	INDEP_FAMILY_SIZE	0.006743
12	MILITARY_STATUS	0.004523
10	SEXUAL_ORIENTATION	0.004258
11	TUITION_TYPE	0.004034
14	ACAD_STAND_02	0.002911
13	ACAD_STAND_01	0.002552
4	DOMESTIC_INTERNATIONAL	0.000913

Figure 9: Python Gradient Boosting Variable Importance

```
accuracy10= 53.98 %
accuracy15= 65.81 %
accuracy20= 74.75 %
```

Figure 10: Python Gradient Boosting Prediction Accuracies

## B. Random Forest

### a. SAS Code and Results

```
/*RANDOM FOREST*/

/*IMPORT DATAFILE*/
%proc import out=graddata
datafile="//vdi-fileshare02/UEM/profiles/032283262/Desktop/STAT 574/Project/Graduate Student Data_VAR.csv"
dbsms=csv replace;
run;

/*SPLIT DATA INTO 80% TRAINING AND 20% TESTING*/
%proc surveyselect
data=graddata
rate=0.8
seed=902847
out=graddata
outall method=sts;
run;

/*BUILDING RANDOM FOREST REGRESSION*/
%proc hpforest
data=graddata
seed=902847
maxtrees=60
vars_to_try=4
trainfraction=0.7
maxdepth=50;
target years_to_degree/level=interval;
input term term_pandemic major_cat race_ethnicity_cat domestic_international first_gen_cat gender sexual_orientation tuition_type military_status
acad_stand_01 acad_stand_02 culminating_activity bach_school/level=nominal;
input age indep_family_size admission_gpa gpa_cur_01 gpa_cur_02 units_taken_01 units_taken_02 units_passed_01 units_passed_02 pass_rate_01 pass_rate_02/level=interval;
partition rolevar=selected(train='1');
save file="C:/Users/032283262/Desktop/random_forest.bin";
run;

/*COMPUTE PREDICTED VALUES FOR TESTING DATA*/
%data test;
set graddata;
if(selected='0');
run;

%proc hp4score data=test;
id years_to_degree;
score file="C:/Users/032283262/Desktop/random_forest.bin"
out=predicted;
run;

%proc print;
run;

/*DETERMINE 10%, 15%, AND 20% ACCURACY*/
%data accuracy;
set predicted;
if(abs(years_to_degree-P_years_to_degree)<0.10*years_to_degree) then ind10=1; else ind10=0;
if(abs(years_to_degree-P_years_to_degree)<0.15*years_to_degree) then ind15=1; else ind15=0;
if(abs(years_to_degree-P_years_to_degree)<0.20*years_to_degree) then ind20=1; else ind20=0;
run;

%proc sql;
select (sum(ind10)/count(*))*100 as accuracy10, (sum(ind15)/count(*))*100 as accuracy15, (sum(ind20)/count(*))*100 as accuracy20
from accuracy;
quit;
```

Figure 11: SAS Random Forest Code

Loss Reduction Variable Importance					
Variable	Number of Rules	MSE	OOB MSE	Absolute Error	OOB Absolute Error
MAJOR_CAT	1689	0.070576	0.05400	0.086620	0.076324
CULMINATING_ACTIVITY	1344	0.030377	0.02545	0.020462	0.017280
UNITS_TAKEN_02	3225	0.034052	0.02007	0.033736	0.023443
UNITS_PASSED_02	3379	0.035127	0.01800	0.024171	0.012241
UNITS_PASSED_01	3349	0.029229	0.01462	0.019324	0.008771
UNITS_TAKEN_01	3188	0.025712	0.00922	0.026058	0.013639
TERM	1254	0.015029	0.00879	0.009509	0.005319
PASS_RATE_01	1494	0.011291	0.00487	0.010022	0.005149
TUITION_TYPE	318	0.003231	0.00208	0.001960	0.001381
DOMESTIC_INTERNATIONAL	247	0.001644	0.00117	0.001013	0.000571
BACH_SCHOOL	1581	0.010532	0.00115	0.006080	0.000103
ACAD_STAND_02	117	0.001378	0.00062	0.000696	0.000192
TERM_PANDEMIC	892	0.003089	0.00024	0.001850	-0.000038741
ACAD_STAND_01	148	0.000594	0.00010	0.000297	-0.000111
MILITARY_STATUS	36	0.000221	-0.00012	0.000124	-0.000020743
GENDER	1300	0.004125	-0.00069	0.002607	-0.000349
PASS_RATE_02	1603	0.008119	-0.00084	0.007004	0.000733
FIRST_GEN_CAT	1493	0.004480	-0.00131	0.003328	-0.000259
GPA_CUR_01	3616	0.021004	-0.00152	0.016543	0.001572
SEXUAL_ORIENTATION	1494	0.004997	-0.00166	0.002830	-0.001340
RACE_ETHNICITY_CAT	1276	0.004534	-0.00251	0.002741	-0.001823
GPA_CUR_02	3398	0.016748	-0.00538	0.015519	0.001859
INDEP_FAMILY_SIZE	3319	0.012908	-0.01044	0.012293	-0.003190
AGE	5604	0.024364	-0.02073	0.022481	-0.006895
ADMISSION_GPA	5631	0.024526	-0.02234	0.024376	-0.007460

Figure 12: SAS Random Forest Variable Importance

accuracy10	accuracy15	accuracy20
50.22556	63.00752	71.57895

Figure 13: SAS Random Forest Prediction Accuracies

## b. R Code and Results

```

1 #IMPORT DATAFILE
2 grad_student.data<- read.csv(file="/Users/noppakansirikul/Desktop/STAT 574/PROJECT/DATA/Graduate Student Data_VAR.csv", header=TRUE, sep=",")
3 grad_student.data<-na.omit(grad_student.data)
4
5 #SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
6 set.seed(902487)
7 sample<- sample(c(TRUE, FALSE), nrow(grad_student.data), replace=TRUE, prob=c(0.8,0.2))
8 train<- grad_student.data[sample,]
9 test<- grad_student.data[!sample,]
10
11 #BUILDING RANDOM FOREST REGRESSION
12 #install.packages("randomForest")
13 library(randomForest)
14 rf.reg<- randomForest(YEARS_TO_DEGREE ~ TERM + TERM_PANDEMIC + MAJOR_CAT + RACE_ETHNICITY_CAT + DOMESTIC_INTERNATIONAL + FIRST_GEN_CAT + AGE + INDEP_INCOME + INDEP_FAMILY_SIZE + GENDER
15 + SEXUAL_ORIENTATION + TUITION_TYPE + MILITARY_STATUS + ACAD_STAND_01 + ACAD_STAND_02 + ADMISSION_GPA + CULMINATING_ACTIVITY + GPA_CUR_01 + GPA_CUR_02 + BACH_SCHOOL + PASS_RATE_01 + PASS_RATE_02,
16 data=train, ntree=150, mtry=5, maxnodes=30, na.action=na.omit)
17
18 #DISPLAYING FEATURE IMPORTANCE
19 print(importance(rf.reg,type=2))
20
21 #COMPUTING PREDICTION ACCURACY FOR TESTING DATA
22 P_YEARS_TO_DEGREE<- predict(rf.reg, newdata=test)
23
24 #accuracy within 10%
25 accuracy10<- ifelse(abs(test$YEARS_TO_DEGREE - P_YEARS_TO_DEGREE)<0.10*test$YEARS_TO_DEGREE,1,0)
26 accuracy10_round<-round((mean(accuracy10)*100),2)
27 cat("accuracy10=",accuracy10_round, "%")
28
29 #accuracy within 15%
30 accuracy15<- ifelse(abs(test$YEARS_TO_DEGREE - P_YEARS_TO_DEGREE)<0.15*test$YEARS_TO_DEGREE,1,0)
31 accuracy15_round<-round((mean(accuracy15)*100),2)
32 cat("accuracy15=",accuracy15_round, "%")
33
34 #accuracy within 20%
35 accuracy20<- ifelse(abs(test$YEARS_TO_DEGREE - P_YEARS_TO_DEGREE)<0.20*test$YEARS_TO_DEGREE,1,0)
36 accuracy20_round<-round((mean(accuracy20)*100),2)
37 cat("accuracy20=",accuracy20_round, "%")
38
39

```

Figure 14: R Random Forest Code

	IncNodePurity
TERM	38.0876319
TERM_PANDEMIC	4.8101514
MAJOR_CAT	155.1249554
RACE_ETHNICITY_CAT	1.8652250
DOMESTIC_INTERNATIONAL	1.2214095
FIRST_GEN_CAT	1.4192189
AGE	11.3024413
INDEP_INCOME	3.4616321
INDEP_FAMILY_SIZE	1.4291953
GENDER	3.7522562
SEXUAL_ORIENTATION	3.2208295
TUITION_TYPE	0.1532751
MILITARY_STATUS	0.7686973
ACAD_STAND_01	1.8254124
ACAD_STAND_02	4.3470757
ADMISSION_GPA	10.5742999
CULMINATING_ACTIVITY	22.4408973
GPA_CUR_01	32.7221292
GPA_CUR_02	37.2007501
BACH_SCHOOL	10.4779953
PASS_RATE_01	21.7124568
PASS_RATE_02	34.4944273

Figure 15: R Random Forest Variable Importance

```

> cat("accuracy10=",accuracy10_round,"%")
accuracy10= 18.2 %
> cat("accuracy15=",accuracy15_round,"%")
accuracy15= 26.98 %
> cat("accuracy20=",accuracy20_round,"%")
accuracy20= 54.5 %

```

Figure 16: R Random Forest Prediction Accuracies

### c. Python Code and Results

```
In [1]: #IMPORT PACKAGES
import pandas
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

#IMPORT DATAFILE
graddata=pandas.read_csv('Graduate Student Data_MAJ2.csv')
graddata=graddata.dropna()

#RECODE CATEGORICAL VARIABLES
coding_term={'Fall 2016':0,'Fall 2017':1,'Fall 2018':2,'Fall 2019':3,'Fall 2020':4,'Fall 2021':5,
'Spring 2017':6,'Spring 2018':7,'Spring 2019':8,'Spring 2020':9,'Spring 2021':10}
graddata['TERM']=graddata['TERM'].map(coding_term)

coding_pandemic={'PRE-PANDEMIC':0,'PANDEMIC':1}
graddata['TERM_PANDEMIC']=graddata['TERM_PANDEMIC'].map(coding_pandemic)

coding_race={'Asian':0,'Black or African American':1,'Hispanic/Latino':2,'White':3,'Other':4}
graddata['RACE_ETHNICITY_CAT']=graddata['RACE_ETHNICITY_CAT'].map(coding_race)

coding_visa={'DOMESTIC':0,'INTERNATIONAL':1}
graddata['DOMESTIC_INTERNATIONAL']=graddata['DOMESTIC_INTERNATIONAL'].map(coding_visa)

coding_firstgen={'NOT FIRST GEN':0,'FIRST GEN':1,'UNKNOWN':2}
graddata['FIRST_GEN_CAT']=graddata['FIRST_GEN_CAT'].map(coding_firstgen)

coding_income={'LESS THAN $6K PER YEAR':0,'$6,000 TO $ 11,999':1,'$12,000 TO $ 23,999':2,'$24,000 TO $ 35,999':3,
'$36,000 TO $ 47,999':4,'$48,000 TO $ 59,999':5,'$60,000 OR MORE PER YEAR':6,'NO RESPONSE':7}
graddata['INDEP_INCOME']=graddata['INDEP_INCOME'].map(coding_income)

coding_gender={'CIS FEMALE':0,'CIS MALE':1,'NON-BINARY/TRANS':2}
graddata['GENDER']=graddata['GENDER'].map(coding_gender)

coding_sexualorientation={'STRAIGHT/HETEROSEXUAL':0,'QUEER':1,'DECLINE TO SAY':2}
graddata['SEXUAL_ORIENTATION']=graddata['SEXUAL_ORIENTATION'].map(coding_sexualorientation)

coding_tuition={'IN-STATE':0,'OUT-OF-STATE':1,'OTHER':2}
graddata['TUITION_TYPE']=graddata['TUITION_TYPE'].map(coding_tuition)

coding_military={'NOT A US MILITARY SERVICE MEMBER':0,'MILITARY SERVICE MEMBER':1}
graddata['MILITARY_STATUS']=graddata['MILITARY_STATUS'].map(coding_military)

coding_acad1={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3}
graddata['ACAD_STAND_01']=graddata['ACAD_STAND_01'].map(coding_acad1)

coding_acad2={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3,
'Disqualified':4}
graddata['ACAD_STAND_02']=graddata['ACAD_STAND_02'].map(coding_acad2)

coding_activity={'Master's Thesis':0,'Graduate Project':1,'Comprehensive Exams':2,'Unknown':3}
graddata['CULMINATING_ACTIVITY']=graddata['CULMINATING_ACTIVITY'].map(coding_activity)

coding_bachschool={'CSULB':0,'Other CSU':1,'Other College in the US':2,'UC':3,'International':4}
graddata['BACH_SCHOOL']=graddata['BACH_SCHOOL'].map(coding_bachschool)

x=graddata.iloc[:,1:27].values
y=graddata.iloc[:,27].values

#SPLIT DATA INTO 80% TRAINING AND 20% TESTING SETS
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.20, random_state=902487)

#FIT RANDOM FOREST REGRESSION TREE
rf_reg=RandomForestRegressor(n_estimators=100, random_state=323445,
max_depth=50, max_features=4)
rf_reg.fit(x_train, y_train)

#DISPLAY VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['TERM', 'TERM_PANDEMIC', 'MAJOR_CAT_NUM', 'RACE_ETHNICITY_CAT', 'DOMESTIC_INTERNATIONAL',
'FIRST_GEN_CAT', 'AGE', 'INDEP_INCOME', 'INDEP_FAMILY_SIZE', 'GENDER', 'SEXUAL_ORIENTATION',
'TUITION_TYPE', 'MILITARY_STATUS', 'ACAD_STAND_01', 'ACAD_STAND_02', 'ADMISSION_GPA',
'CULMINATING_ACTIVITY', 'GPA_CUR_01', 'GPA_CUR_02', 'UNITS_TAKEN_01', 'UNITS_TAKEN_02',
'UNITS_PASSED_01', 'UNITS_PASSED_02', 'PASS_RATE_01', 'PASS_RATE_02', 'BACH_SCHOOL'],
columns=['var_name'])

loss_reduction=pandas.DataFrame(rf_reg.feature_importances_, columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0, ascending=False)
print(var_importance)

#COMPUTE PREDICTION ACCURACY FOR TESTING DATA
y_pred=rf_reg.predict(x_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1 if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1 if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1 if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#ACCURACY WITHIN 10%
accuracy10=sum(ind10)/len(ind10)
print("accuracy10=",round(accuracy10*100,2),"%")

#ACCURACY WITHIN 15%
accuracy15=sum(ind15)/len(ind15)
print("accuracy15=",round(accuracy15*100,2),"%")

#ACCURACY WITHIN 20%
accuracy20=sum(ind20)/len(ind20)
print("accuracy20=",round(accuracy20*100,2),"%")
```

Figure 17: Python Random Forest Code



	var_name	loss_reduction
2	MAJOR_CAT_NUM	0.134927
16	CULMINATING_ACTIVITY	0.081902
20	UNITS_TAKEN_02	0.078490
22	UNITS_PASSED_02	0.075284
15	ADMISSION_GPA	0.067634
21	UNITS_PASSED_01	0.067560
19	UNITS_TAKEN_01	0.063140
0	TERM	0.056618
6	AGE	0.056328
18	GPA_CUR_02	0.041528
7	INDEP_INCOME	0.038673
17	GPA_CUR_01	0.037144
25	BACH_SCHOOL	0.034463
3	RACE_ETHNICITY_CAT	0.028763
8	INDEP_FAMILY_SIZE	0.022114
5	FIRST_GEN_CAT	0.018915
23	PASS_RATE_01	0.018524
24	PASS_RATE_02	0.016878
10	SEXUAL_ORIENTATION	0.015417
9	GENDER	0.014856
1	TERM_PANDEMIC	0.011291
11	TUITION_TYPE	0.005311
14	ACAD_STAND_02	0.004771
13	ACAD_STAND_01	0.003460
12	MILITARY_STATUS	0.003338
4	DOMESTIC_INTERNATIONAL	0.002670

Figure 18: Python Random Forest Variable Importance

```
accuracy10= 50.1 %
accuracy15= 61.53 %
accuracy20= 72.47 %
```

Figure 19: Python Random Forest Prediction Accuracies

## C. Decision Tree

### a. SAS Code and Results

```
/*DECISION TREE*/

/*IMPORT DATAFILE*/
proc import out=graddata
datafile="/vdi-fileshare02/UEMprofiles/032283262/Desktop/STAT 574/Project/Graduate Student Data_VAR.csv"
dbsms=csv replace;
run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING*/
proc surveyselect
data=graddata
rate=0.8
seed=902847
out=graddata
outall method=srs;
run;

/*RSS SPLITTING CRITERION - FULL TREE*/
proc hpsplit
data=graddata
seed=902847;
class term term_pandemic major_cat race_ethnicity_cat domestic_international first_gen_cat indep_income gender sexual_orientation tuition_type military_status
acad_stand_01 acad_stand_02 culminating_activity bach_school;
model years_to_degree=term term_pandemic major_cat race_ethnicity_cat domestic_international first_gen_cat age indep_income indep_family_size gender sexual_orientation tuition_type
military_status acad_stand_01 acad_stand_02 admission_gpa culminating_activity gpa_cur_01 gpa_cur_02 units_taken_01 units_taken_02 pass_rate_01 pass_rate_02 bach_school;
grow RSS;
partition rolevar=selected(train="1");
run;

/*RSS SPLITTING AND COST-COMPLEXITY PRUNING*/
proc hpsplit data=graddata;
class term term_pandemic major_cat race_ethnicity_cat domestic_international first_gen_cat indep_income gender sexual_orientation tuition_type military_status
acad_stand_01 acad_stand_02 culminating_activity bach_school;
model years_to_degree=term term_pandemic major_cat race_ethnicity_cat domestic_international first_gen_cat age indep_income indep_family_size gender sexual_orientation tuition_type
military_status acad_stand_01 acad_stand_02 admission_gpa culminating_activity gpa_cur_01 gpa_cur_02 units_taken_01 units_taken_02 pass_rate_01 pass_rate_02 bach_school;
grow RSS;
prune costcomplexity(leaves=5);
partition rolevar=selected(train="1");
output out=predicted;
id selected;
run;

/*COMPUTING PREDICTION ACCURACY FOR TESTING DATA*/
data test;
set predicted;
if(selected="0");
keep _leaf_ years_to_degree P_years_to_degree;
run;

data accuracy;
set predicted;
if(abs(years_to_degree-P_years_to_degree)<0.10*years_to_degree) then ind10=1; else ind10=0;
if(abs(years_to_degree-P_years_to_degree)<0.15*years_to_degree) then ind15=1; else ind15=0;
if(abs(years_to_degree-P_years_to_degree)<0.20*years_to_degree) then ind20=1; else ind20=0;
run;

proc sql;
select mean(ind10) as accuracy10, mean(ind15) as accuracy15, mean(ind20) as accuracy20
from accuracy;
quit;
```

Figure 20: SAS Decision Tree Code

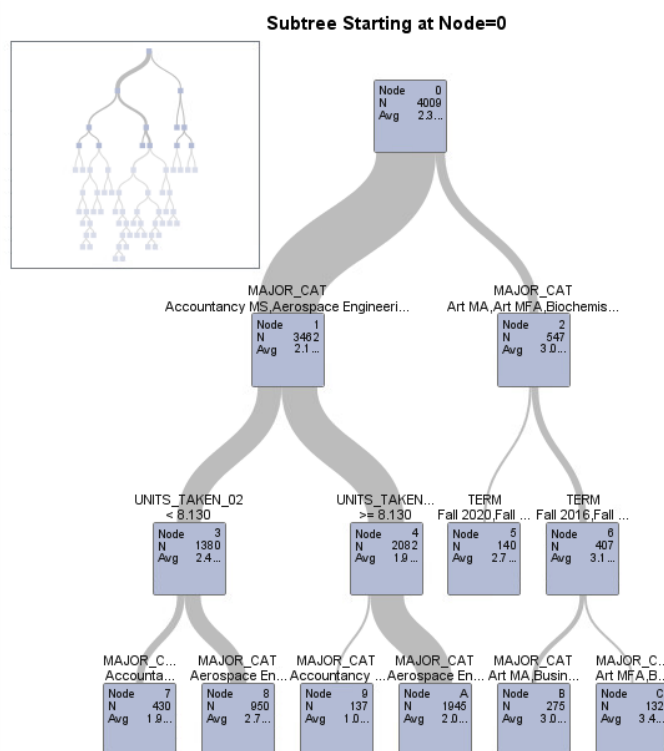


Figure 21: SAS Full Decision Tree with RSS Splitting

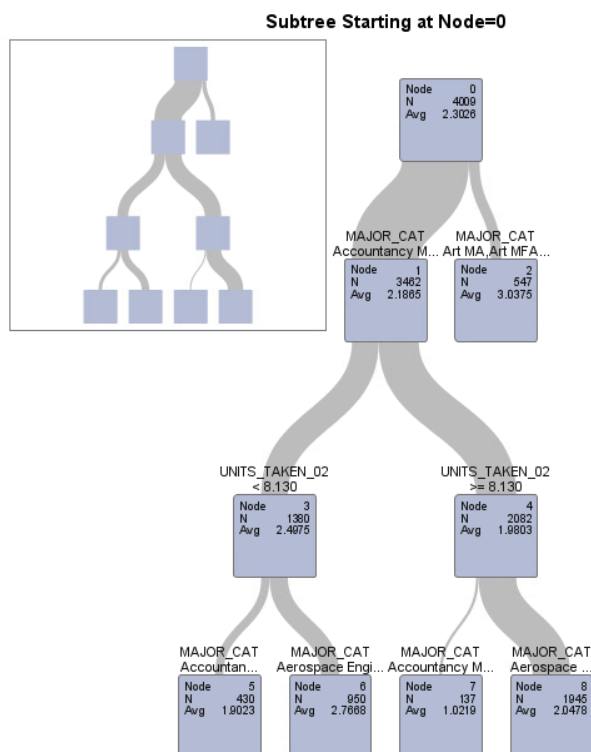


Figure 22: SAS Decision Tree with RSS Splitting and Cost-Complexity Pruning



accuracy10	accuracy15	accuracy20
0.712481	0.745865	0.788722

Figure 23: SAS Decision Tree Prediction Accuracies

## b. R Code and Results

```

2 #IMPORT DATAFILE
3 grad_student.data<- read.csv(file="/Users/noppakansirikul/Desktop/STAT 574/PROJECT/DATA/Graduate Student Data_VAR.csv", header=TRUE, sep=",")
4 grad_student.data<-na.omit(grad_student.data)
5
6 #SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
7 set.seed(902847)
8 sample <- sample(c(TRUE, FALSE), nrow(grad_student.data), replace=TRUE, prob=c(0.8,0.2))
9 train<- grad_student.data[sample,]
10 test<- grad_student.data[!sample,]
11
12 #FITTING FULL REGRESSION TREE WITH RSS SPLITTING
13 #install.packages("rpart")
14 #install.packages("rpart.plot")
15 library(rpart) #recursive partitioning and regression trees
16 reg.tree.full<- rpart(YEARS_TO_DEGREE ~ TERM + TERM_PANDMIC + MAJOR_CAT + RACE_ETHNICITY_CAT + DOMESTIC_INTERNATIONAL + FIRST_GEN_CAT + AGE + INDEP_INCOME + INDEP_FAMILY_SIZE + GENDER
17 + SEXUAL_ORIENTATION + TUITION_TYPE + MILITARY_STATUS + ACAD_STAND_01 + ACAD_STAND_02 + ADMISSION_GPA + CULMINATING_ACTIVITY + GPA_CUR_01 + GPA_CUR_02 + BACH_SCHOOL + PASS_RATE_01 + PASS_RATE_02,
18 data=train, method="anova", xval=10, cp=0) #xval=# of cross-validations
19 printcp(reg.tree.full)
20
21 #FITTING REGRESSION TREE WITH RSS SPLITTING AND COST-COMPLEXITY PRUNING
22 reg.tree.RSS<- rpart(YEARS_TO_DEGREE ~ TERM + TERM_PANDMIC + MAJOR_CAT + RACE_ETHNICITY_CAT + DOMESTIC_INTERNATIONAL + FIRST_GEN_CAT + AGE + INDEP_INCOME + INDEP_FAMILY_SIZE + GENDER
23 + SEXUAL_ORIENTATION + TUITION_TYPE + MILITARY_STATUS + ACAD_STAND_01 + ACAD_STAND_02 + ADMISSION_GPA + CULMINATING_ACTIVITY + GPA_CUR_01 + GPA_CUR_02 + BACH_SCHOOL + PASS_RATE_01 + PASS_RATE_02,
24 data=train, method="anova", cp=0.01) #pruned tree with 7 leaves
25 library("rpart.plot")
26 rpart.plot(reg.tree.RSS, type=3)
27
28 #FITTING REGRESSION TREE WITH RSS SPLITTING AND COST-COMPLEXITY PRUNING
29 reg.tree.RSS<- rpart(YEARS_TO_DEGREE ~ TERM + TERM_PANDMIC + MAJOR_CAT + RACE_ETHNICITY_CAT + DOMESTIC_INTERNATIONAL + FIRST_GEN_CAT + AGE + INDEP_INCOME + INDEP_FAMILY_SIZE + GENDER
30 + SEXUAL_ORIENTATION + TUITION_TYPE + MILITARY_STATUS + ACAD_STAND_01 + ACAD_STAND_02 + ADMISSION_GPA + CULMINATING_ACTIVITY + GPA_CUR_01 + GPA_CUR_02 + BACH_SCHOOL + PASS_RATE_01 + PASS_RATE_02,
31 data=train, method="anova", cp=0.01) #pruned tree with 7 leaves
32 library("rpart.plot")
33 rpart.plot(reg.tree.RSS, type=3)
34
35 #PROBLEM 1B
36
37 #COMPUTING PREDICTION ACCURACY FOR TESTING DATA
38 P_YEARS_TO_DEGREE <- predict(reg.tree.RSS, newdata=test)
39
40 #accuracy within 10% = 0.4634146
41 accuracy10 <- ifelse(abs(test$YEARS_TO_DEGREE-P_YEARS_TO_DEGREE)<0.10*test$YEARS_TO_DEGREE,1,0)
42 accuracy10_round<-round((mean(accuracy10)*100),2))
43 cat("accuracy10=",accuracy10_round, "%")
44
45 #accuracy within 15% = 0.6857963
46 accuracy15 <- ifelse(abs(test$YEARS_TO_DEGREE-P_YEARS_TO_DEGREE)<0.15*test$YEARS_TO_DEGREE,1,0)
47 accuracy15_round<-round((mean(accuracy15)*100),2))
48 cat("accuracy15=",accuracy15_round, "%")
49
50 #accuracy within 20% = 0.8034433
51 accuracy20 <- ifelse(abs(test$YEARS_TO_DEGREE-P_YEARS_TO_DEGREE)<0.20*test$YEARS_TO_DEGREE,1,0)
52 accuracy20_round<-round((mean(accuracy20)*100),2))
53 cat("accuracy20=",accuracy20_round, "%")

```

Figure 24: R Decision Tree Code

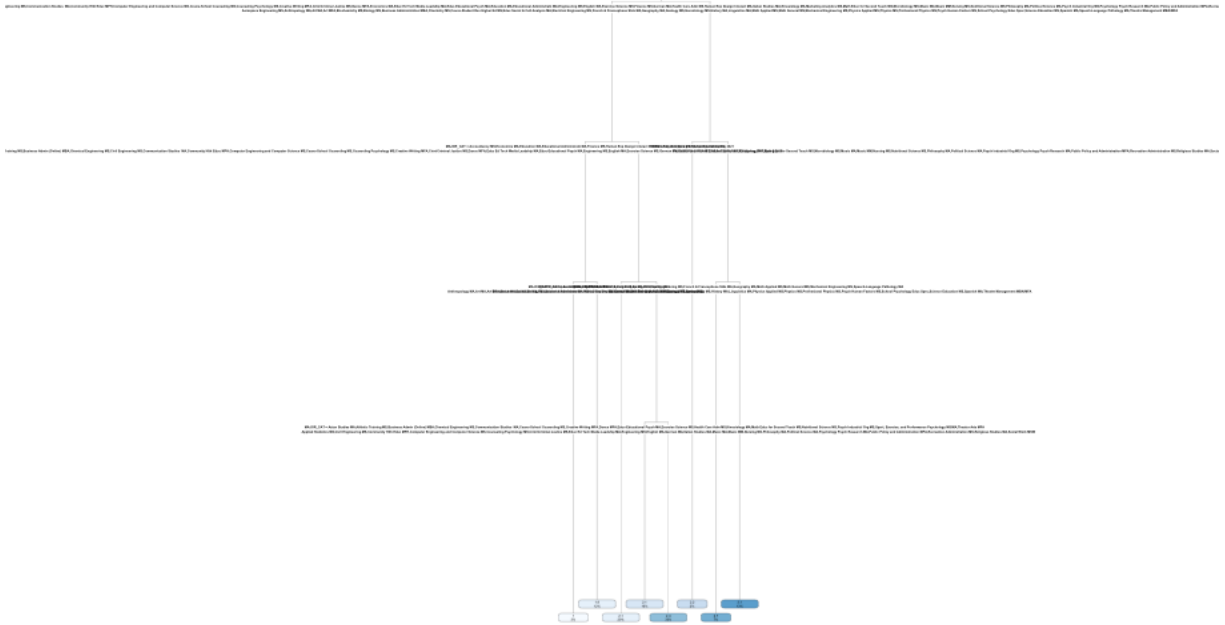


Figure 25: R Full Decision Tree with RSS Splitting

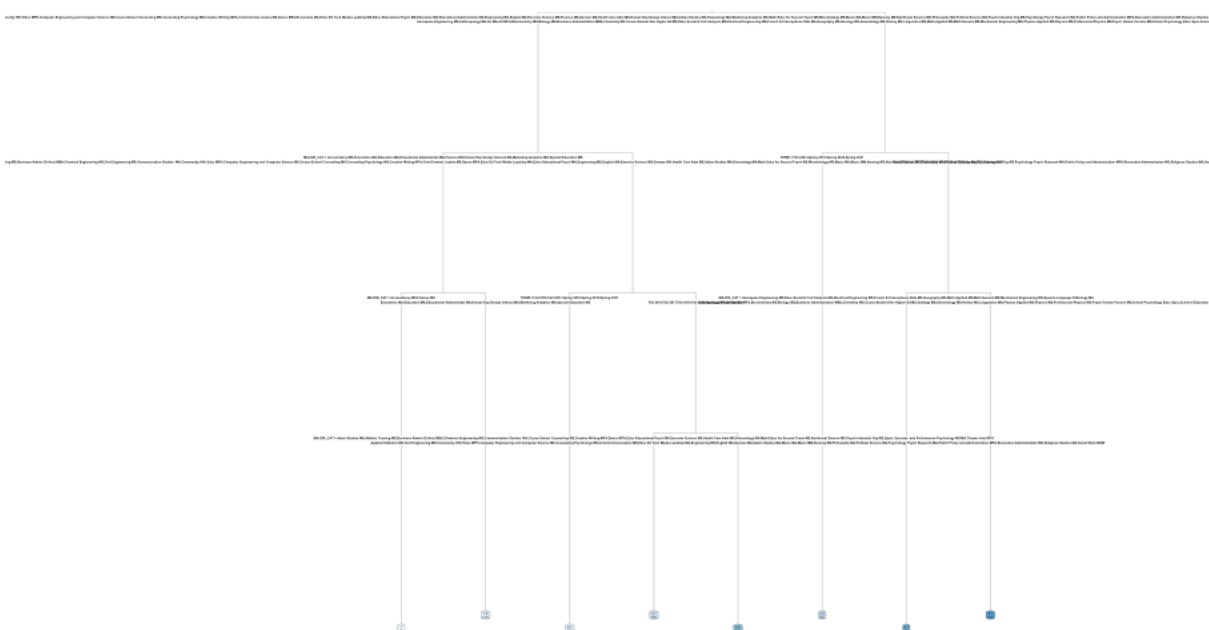


Figure 26: R Decision Tree with RSS Splitting and Cost-Complexity Pruning

```
> cat("accuracy10=",accuracy10_round,"%")
accuracy10= 47.13 %
> cat("accuracy15=",accuracy15_round,"%")
accuracy15= 51.28 %
> cat("accuracy20=",accuracy20_round,"%")
accuracy20= 68.48 %
```

Figure 27: R Decision Tree Prediction Accuracies

### c. Python Code and Results

```
In [2]: import numpy
import pandas
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn import tree
from sklearn.model_selection import train_test_split

#IMPORT DATAFILE
graddata=pandas.read_csv('Graduate Student Data_MAJ2.csv')
graddata=graddata.dropna()

#RECODE CATEGORICAL VARIABLES
coding_term={'Fall 2016':0,'Fall 2017':1,'Fall 2018':2,'Fall 2019':3,'Fall 2020':4,'Fall 2021':5,
'Spring 2017':6,'Spring 2018':7,'Spring 2019':8,'Spring 2020':9,'Spring 2021':10}
graddata['TERM']=graddata['TERM'].map(coding_term)

coding_pandemic={'PRE-PANDEMIC':0,'PANDEMIC':1}
graddata['TERM_PANDEMIC']=graddata['TERM_PANDEMIC'].map(coding_pandemic)

coding_race={'Asian':0,'Black or African American':1,'Hispanic/Latino':2,'White':3,'Other':4}
graddata['RACE_ETHNICITY_CAT']=graddata['RACE_ETHNICITY_CAT'].map(coding_race)

coding_visa={'DOMESTIC':0,'INTERNATIONAL':1}
graddata['DOMESTIC_INTERNATIONAL']=graddata['DOMESTIC_INTERNATIONAL'].map(coding_visa)

coding_firstgen={'NOT FIRST GEN':0,'FIRST GEN':1,'UNKNOWN':2}
graddata['FIRST_GEN_CAT']=graddata['FIRST_GEN_CAT'].map(coding_firstgen)

coding_income={'LESS THAN $6K PER YEAR':0,'$6,000 TO $ 11,999':1,'$12,000 TO $ 23,999':2,'$24,000 TO $ 35,999':3,
'$36,000 TO $ 47,999':4,'$48,000 TO $ 59,999':5,'$60,000 OR MORE PER YEAR':6,'NO RESPONSE':7}
graddata['INDEP_INCOME']=graddata['INDEP_INCOME'].map(coding_income)

coding_gender={'CIS FEMALE':0,'CIS MALE':1,'NON-BINARY/TRANS':2}
graddata['GENDER']=graddata['GENDER'].map(coding_gender)

coding_sexualorientation={'STRAIGN/HETEROSEXUAL':0,'QUEER':1,'DECLINE TO SAY':2}
graddata['SEXUAL_ORIENTATION']=graddata['SEXUAL_ORIENTATION'].map(coding_sexualorientation)

coding_tuition={'IN-STATE':0,'OUT-OF-STATE':1,'OTHER':2}
graddata['TUITION_TYPE']=graddata['TUITION_TYPE'].map(coding_tuition)

coding_military={'NOT A US MILITARY SERVICE MEMBER':0,'MILITARY SERVICE MEMBER':1}
graddata['MILITARY_STATUS']=graddata['MILITARY_STATUS'].map(coding_military)

coding_acad1={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3}
graddata['ACAD_STAND_01']=graddata['ACAD_STAND_01'].map(coding_acad1)

coding_acad2={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3,
'Disqualified':4}
graddata['ACAD_STAND_02']=graddata['ACAD_STAND_02'].map(coding_acad2)

coding_activity={'Master's Thesis':0,'Graduate Project':1,'Comprehensive Exams':2,'Unknown':3}
graddata['CULMINATING_ACTIVITY']=graddata['CULMINATING_ACTIVITY'].map(coding_activity)

coding_bachschool={'CSULB':0,'Other CSU':1,'Other College in the US':2,'UC':3,'International':4}
graddata['BACH_SCHOOL']=graddata['BACH_SCHOOL'].map(coding_bachschool)

x=graddata.iloc[:,1:27].values
y=graddata.iloc[:,27].values

#SPLIT DATA INTO 80% TRAINING AND 20% TESTING SETS
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.20, random_state=902487)

#FITTING REGRESSION TREE WITH RSS SPLITTING CRITERION
rtree = DecisionTreeRegressor(random_state=902487,
criterion="squared_error", max_leaf_nodes=5)
reg_tree_RSS = rtree.fit(x_train, y_train)

#PLOTING FITTED TREE
fig=plt.figure(figsize=(15,10))
fn=['TERM','TERM_PANDEMIC','MAJOR_CAT_NUM','RACE_ETHNICITY_CAT','DOMESTIC_INTERNATIONAL',
'FIRST_GEN_CAT','AGE','INDEP_INCOME','INDEP_FAMILY_SIZE','GENDER','SEXUAL_ORIENTATION',
'TUITION_TYPE','MILITARY_STATUS','ACAD_STAND_01','ACAD_STAND_02','ADMISSION_GPA','CULMINATING_ACTIVITY',
'GPA_CUR_01','GPA_CUR_02','UNITS_TAKEN_01','UNITS_TAKEN_02','UNITS_PASSED_01','UNITS_PASSED_02',
'PASS_RATE_01','PASS_RATE_02','BACH_SCHOOL']
tree.plot_tree(reg_tree_RSS, feature_names=fn, filled=True)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=reg_tree_RSS.predict(x_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred,y_test):
    ind10.append(1 if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1 if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1 if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#ACCURACY WITHIN 10%
accuracy10=sum(ind10)/len(ind10)
print("accuracy10=",round(accuracy10*100,2),'%')

#ACCURACY WITHIN 15%
accuracy15=sum(ind15)/len(ind15)
print("accuracy15=",round(accuracy15*100,2),'%')

#ACCURACY WITHIN 20%
accuracy20=sum(ind20)/len(ind20)
print("accuracy20=",round(accuracy20*100,2),'%')
```

Figure 28: Python Decision Tree Code

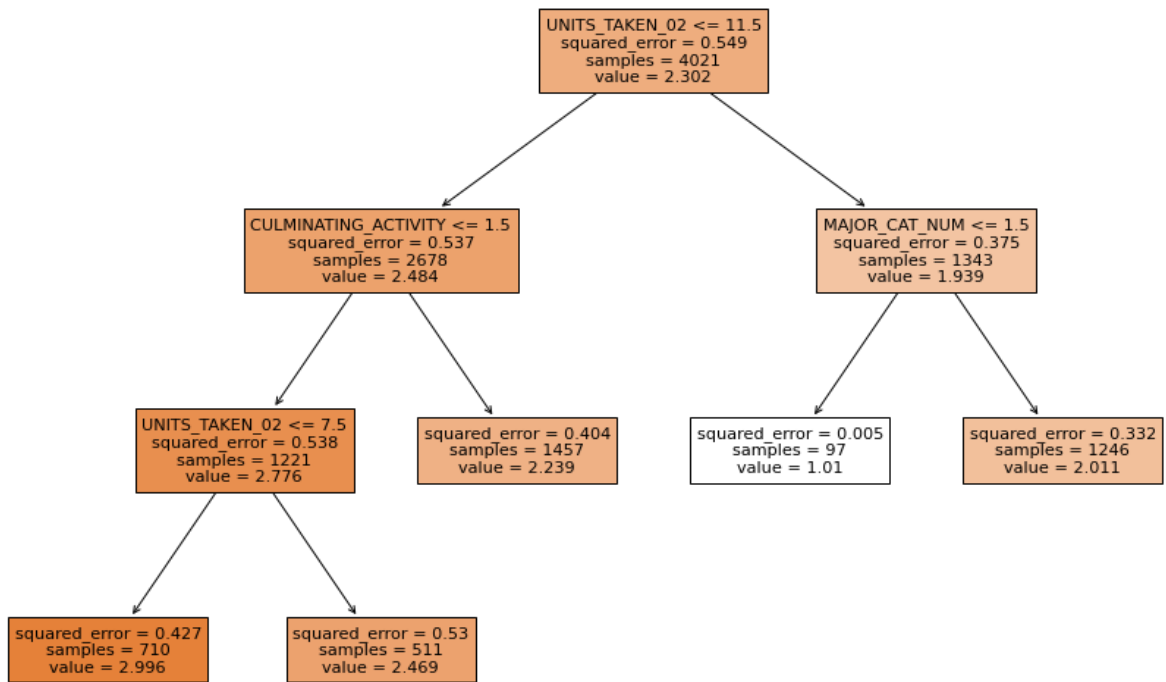


Figure 29: Python Full Decision Tree with RSS Splitting

```

accuracy10= 34.49 %
accuracy15= 58.25 %
accuracy20= 64.81 %
  
```

Figure 30: Python Decision Tree Prediction Accuracies

## D. kNN

### a. SAS Code and Results

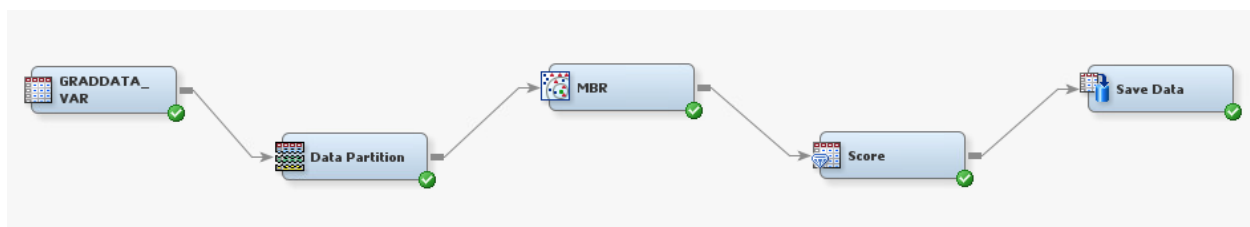


Figure 31: SAS kNN Enterprise Miner Diagram

```

/*KNN*/

/*IMPORT DATAFILE*/
proc import out=sasuser.graddata_var
  datafile="//vdi-fileshare02/UEMprofiles/032283262/Desktop/STAT 574/Project/Graduate Student Data_VAR.csv"
  dbms=csv replace;
run;

/*Running Memory Based Reasoning (MBR) (or kNN regression)
in Enterprise Miner*/

/*COMPUTING ACCURACY WITHIN 10%, 15%, AND 20%*/
data accuracy;
  set tmp1.em_save_test;
  ind10=(abs(R_median_house_value)<0.10*median_house_value);
  ind15=(abs(R_median_house_value)<0.15*median_house_value);
  ind20=(abs(R_median_house_value)<0.20*median_house_value);
  obs_n=_N_;
run;

proc sql;
  select mean(ind10) as accuracy10, mean(ind15) as accuracy15, mean(ind20) as accuracy20
  from accuracy;
quit;

/*PLOTTING ACTUAL AND PREDICTED VALUES FOR TESTING DATA*/;
goptions reset=all border;
title1 "k-Nearest Neighbor (KNN) Regression";
symbol1 interpol=join value=dot color=limegreen;
symbol2 interpol=join value=dot color=hotpink;
legend1 value=("actual" "predicted")
position=(top right inside) label=none;
proc gplot data=accuracy;
  plot years_to_degree*obs_n
  EM_PREDICTION*obs_n/ overlay legend=legend1;
run;

```

Figure 32: SAS kNN Code



Figure 33: SAS kNN Actual vs. Predicted Values

accuracy10	accuracy15	accuracy20
0.32782	0.473684	0.626316

Figure 34: SAS kNN Prediction Accuracies

## b. R Code and Results

```

1
2 #IMPORT DATAFILE
3 grad_student.data<- read.csv(file="/Users/nappakansirikul/Desktop/STAT 574/PROJECT/DATA/Graduate Student Data_VAR.csv", header=TRUE, sep=",")
4 grad_student.data<-na.omit(grad_student.data)
5
6 #SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
7 set.seed(902847)
8 sample <- sample(c(TRUE, FALSE), nrow(grad_student.data), replace=TRUE, prob=c(0.8,0.2))
9 train<- grad_student.data[sample,]
10 test<- grad_student.data[!sample,]
11
12 train.x<- data.matrix(train[1:28])
13 train.y<- data.matrix(train[29])
14 test.x<- data.matrix(test[1:28])
15 test.y<- data.matrix(test[29])
16
17 #TRAINING K-NEAREST NEIGHBOR REGRESSION
18 #install.packages("caret")
19 #Classification and Regression Training
20 library(caret)
21 print(train[YEARS_TO_DEGREE ~ TERM + TERM_PANDemic + MAJOR_CAT + RACE_ETHNICITY_CAT + DOMESTIC_INTERNATIONAL + FIRST_GEN_CAT + AGE + INDEP_INCOME + INDEP_FAMILY_SIZE + GENDER
22 + SEXUAL_ORIENTATION + TUITION_TYPE + MILITARY_STATUS + ACAD_STAND_01 + ACAD_STAND_02 + ADMISSION_GPA + CULMINATING_ACTIVITY + GPA_CUR_01 + GPA_CUR_02 + BACH_SCHOOL, data=train, method="knn"))
23
24 #FITTING OPTIMAL KNN REGRESSION (K=50)
25 knn.reg<- knnreg(train.x, train.y, k=50)
26
27 #COMPUTING PREDICTION ACCURACY FOR TESTING DATA
28 pred.y<- predict(knn.reg, test.x)
29
30 #accuracy within 10%
31 accuracy10<- ifelse(abs(test.y-pred.y)<0.10*test.y,1,0)
32 accuracy10_round<-round((mean(accuracy10)*100),2)
33 cat("accuracy10=",accuracy10_round,"%")
34
35 #accuracy within 15%
36 accuracy15<- ifelse(abs(test.y-pred.y)<0.15*test.y,1,0)
37 accuracy15_round<-round((mean(accuracy15)*100),2)
38 cat("accuracy15=",accuracy15_round,"%")
39
40 #accuracy within 20%
41 accuracy20<- ifelse(abs(test.y-pred.y)<0.20*test.y,1,0)
42 accuracy20_round<-round((mean(accuracy20)*100),2)
43 cat("accuracy20=",accuracy20_round,"%")
44
45 #PLOTING ACTUAL AND PREDICTED VALUES FOR TESTING DATA
46 x<- 1:length(test.y)
47 plot(x, test.y, type="l", lwd=2, col= "limegreen", main="KNN Regression",
48 panel.first=grid())
49 lines(x, pred.y, lwd=2, col= "hotpink")
50 legend("topright", c("actual", "predicted"), lty=1, lwd=2,
51 col=c( "limegreen", "hotpink"))
52

```

Figure 35: R kNN Code

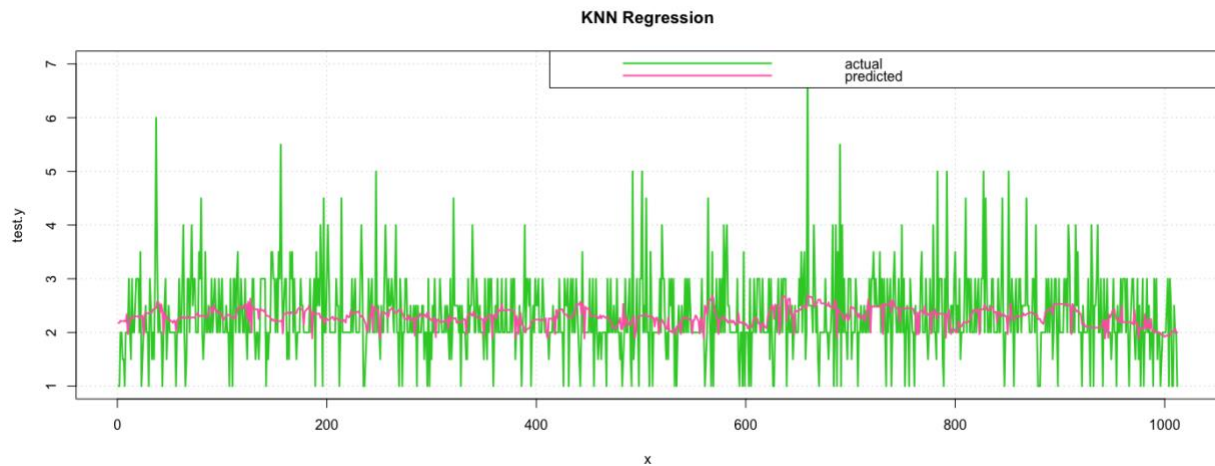


Figure 36: R kNN Actual vs. Predicted Values

```

> cat("accuracy10=",accuracy10_round,"%")
accuracy10= 20.06 %
> cat("accuracy15=",accuracy15_round,"%")
accuracy15= 37.55 %
> cat("accuracy20=",accuracy20_round,"%")
accuracy20= 53.26 %

```

Figure 37: R kNN Prediction Accuracies

### c. Python Code and Results

```
In [1]: import pandas
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from statistics import mean

#IMPORT DATAFILE
graddata=pandas.read_csv('Graduate Student Data_MAJ2.csv')
graddata=graddata.dropna()

#RECODE CATEGORICAL VARIABLES
coding_term={'Fall 2016':0,'Fall 2017':1,'Fall 2018':2,'Fall 2019':3,'Fall 2020':4,'Fall 2021':5,
            'Spring 2017':6,'Spring 2018':7,'Spring 2019':8,'Spring 2020':9,'Spring 2021':10}
graddata['TERM']=graddata['TERM'].map(coding_term)

coding_pandemic={'PRE-PANDEMIC':0,'PANDEMIC':1}
graddata['TERM_PANDEMIC']=graddata['TERM_PANDEMIC'].map(coding_pandemic)

coding_race={'Asian':0,'Black or African American':1,'Hispanic/Latino':2,'White':3,'Other':4}
graddata['RACE_ETHNICITY_CAT']=graddata['RACE_ETHNICITY_CAT'].map(coding_race)

coding_visa={'DOMESTIC':0,'INTERNATIONAL':1}
graddata['DOMESTIC_INTERNATIONAL']=graddata['DOMESTIC_INTERNATIONAL'].map(coding_visa)

coding_firstgen={'NOT FIRST GEN':0,'FIRST GEN':1,'UNKNOWN':2}
graddata['FIRST_GEN_CAT']=graddata['FIRST_GEN_CAT'].map(coding_firstgen)

coding_income={'LESS THAN $6K PER YEAR':0,'$6,000 TO $ 11,999':1,'$12,000 TO $ 23,999':2,'$24,000 TO $ 35,999':3,
              '$36,000 TO $ 47,999':4,'$48,000 TO $ 59,999':5,'$60,000 OR MORE PER YEAR':6,'NO RESPONSE':7}
graddata['INDEP_INCOME']=graddata['INDEP_INCOME'].map(coding_income)

coding_gender={'CIS FEMALE':0,'CIS MALE':1,'NON-BINARY/TRANS':2}
graddata['GENDER']=graddata['GENDER'].map(coding_gender)

coding_sexualorientation={'STRAIGHT/HETEROSEXUAL':0,'QUEER':1,'DECLINE TO SAY':2}
graddata['SEXUAL_ORIENTATION']=graddata['SEXUAL_ORIENTATION'].map(coding_sexualorientation)

coding_tuition={'IN-STATE':0,'OUT-OF-STATE':1,'OTHER':2}
graddata['TUITION_TYPE']=graddata['TUITION_TYPE'].map(coding_tuition)

coding_military={'NOT A US MILITARY SERVICE MEMBER':0,'MILITARY SERVICE MEMBER':1}
graddata['MILITARY_STATUS']=graddata['MILITARY_STATUS'].map(coding_military)

coding_acad1={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3}
graddata['ACAD_STAND_01']=graddata['ACAD_STAND_01'].map(coding_acad1)

coding_acad2={'Good Academic Standing':0,'Probation - First Term':1,'Probation - Second Term':2,'Missing':3,
              'Disqualified':4}
graddata['ACAD_STAND_02']=graddata['ACAD_STAND_02'].map(coding_acad2)

coding_activity={'Master's Thesis':0,'Graduate Project':1,'Comprehensive Exams':2,'Unknown':3}
graddata['CULMINATING_ACTIVITY']=graddata['CULMINATING_ACTIVITY'].map(coding_activity)

coding_bachschool={'CSULB':0,'Other CSU':1,'Other College in the US':2,'UC':3,'International':4}
graddata['BACH_SCHOOL']=graddata['BACH_SCHOOL'].map(coding_bachschool)

x=graddata.iloc[:,1:27].values
y=graddata.iloc[:,27].values

#SPLIT DATA INTO 80% TRAINING AND 20% TESTING SETS
x_train, x_test, y_train, y_test=train_test_split(x, y, test_size=0.20, random_state=902487)

#FITTING KNN REGRESSION
reg=KNeighborsRegressor(n_neighbors=26)
knn_reg=reg.fit(x_train, y_train)

#COMPUTE PREDICTION ACCURACY FOR TESTING DATA
y_pred=knn_reg.predict(x_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred,y_test):
    ind10.append(1 if abs(sub1-sub2)<0.10*sub2 else ind10.append(0))
    ind15.append(1 if abs(sub1-sub2)<0.15*sub2 else ind15.append(0))
    ind20.append(1 if abs(sub1-sub2)<0.20*sub2 else ind20.append(0))

#ACCURACY WITHIN 10%
accuracy10=sum(ind10)/len(ind10)
print("accuracy10=",round(accuracy10*100,2),"%")

#ACCURACY WITHIN 15%
accuracy15=sum(ind15)/len(ind15)
print("accuracy15=",round(accuracy15*100,2),"%")

#ACCURACY WITHIN 20%
accuracy20=sum(ind20)/len(ind20)
print("accuracy20=",round(accuracy20*100,2),"%")

#plotting actual and predicted observations vs. observation number
import matplotlib.pyplot as plt

n_obs=list(range(0,len(y_test)))
plt.plot(n_obs, y_test, label="actual")
plt.plot(n_obs, y_pred, label="predicted")
plt.xlabel('n_obs')
plt.ylabel('YEARS_TO_DEGREE')
plt.title('KNN Regression')
plt.legend()
plt.show()
```

Figure 38: Python kNN Code



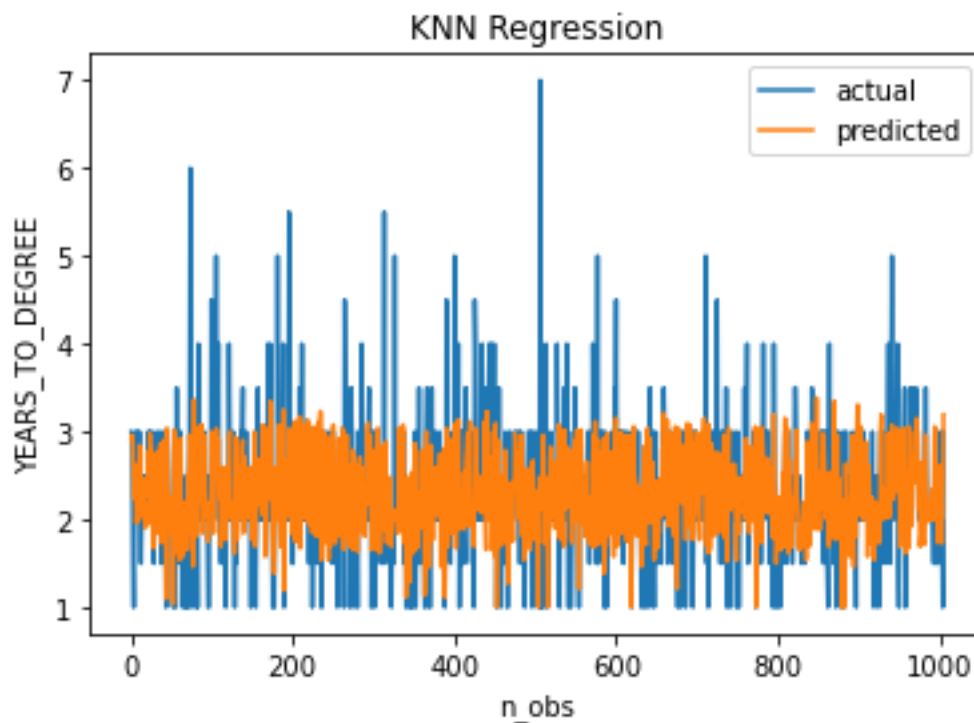


Figure 39: Python kNN Actual vs. Predicted Values

```
accuracy10= 43.94 %
accuracy15= 56.66 %
accuracy20= 66.8 %
```

Figure 40: Python kNN Prediction Accuracies

## References

Darrow, Melanie. "Measuring Student Success beyond Completion." *AGB*, 13 Mar. 2024, [agb.org/trusteeship-article/measuring-student-success-beyond-completion/#:~:text=Three%20major%20sets%20of%20student,terms%20of%20student%20success%20among.](https://agb.org/trusteeship-article/measuring-student-success-beyond-completion/#:~:text=Three%20major%20sets%20of%20student,terms%20of%20student%20success%20among.)

"Data Fellows." *Data Fellows / California State University, Long Beach*, 23 Feb. 2024, [www.csulb.edu/data-fellows](http://www.csulb.edu/data-fellows).