Predicting the Probability of Need for Student Intervention

Amanda: Project Lead & SME (Business Objectives)

Dylan: Data Scientist// Statistician (Technical/Data Understanding)

Bridge: Data Engineer/ Statistician (Preparing and Building the Model)

Brianne: Business Analyst (Model Accuracy)

Dr. Janice Carrillo

Dr. James Hoover

August 3rd, 2020

Model Selection Section

Model Description: Predicted Variable(Target Variable)

The model utilized three dependent variables to predict the likelihood of student success. The first, as indicated by one throughout the report, was for students who were unlikely to graduate. The second, as noted by number two throughout the report, identifies students who were likely to graduate in four years. The third, as indicated by the number three throughout the report, indicates students who were expected to graduate after four years and before five years. Please see the Appendix 1 about how we create these target variables.

Value	description	New Value
no	didn't Graduated	1
4yrs	graduate in four years	2
5yrs	graduate in five years	3

We gave the target variable some new values so that we can use these numerical values to build our model.

Model Description: Predictor Variables

TERM_SID
AGE_YEARS
AGE_MONTHS
AGE_DAYS
TERM_BEG_DT_SID
TERM_END_DT_SID
JUNIOR_SENIOR_FLAGNO
TOT_TRNSFR
TOT_TEST_CREDIT
UNT_TAKEN_PRGRSS
CUM_GPA
ENRL_CNT
MAJORCOUNT
UF_CLASSNO
RESIDENCYNO
TOT_TAKEN_GPA
ACAD_PROG_PRIMARYNO
UF_CLASS_EOTNO
UNT_TAKEN_GPA
TERM_END_DT_SID_CATGRYNO
TERM_BEG_DT_CATGRYNO
STUDENT_SUCCESSNO
LOW_TERM_GPA_IND
PARTTIME_TERM_IND
NOT_REG_TERM_IND
WITHDRWL_TERM_IND
FULLTIME_TERM_IND
OVR_12HR_TERM_IND

We connected the table UF_F_CPPS, table UF_D_ACAD_PLAN, and table UF_B_STDNT_TERM together and did the data processing and data filtering, then we

got these 28 predictor variables. Please see the Appendix 2 for the detail of how we create these predictor variables

For the meanings of these variables, please see the Appendix 3.

On the final KNN Model, on the UF_R3_ANALYSIS_LASTMAJOR0714BE table, one-hot encoding was utilized to transform 13 categorical variables into integers. There is no ordinal relationship in our categorical variables. In these cases, One-hot encoding allows the representation of categorical data to be more expressive. This can help in both making the problem easier for the network to model. When a one hot encoding is used for the output variable, it may offer a more nuanced set of predictions than a single label.

Categorical Variables Transformed Through One-Hot Encoding:

‘JUNIOR_SENIOT_FLAGNO’
'ENRL_CNT'
'MAJORCOUNT'
'UF_CLASSNO'
'RESIDENCYNO'
‘TERM_END_DT_SID_CATGRYNO'
'TERM_BEG_DT_CATGRYNO'
‘LOW_TERM_GPA_IND'
'PARTTIME_TERM_IND'
'NOT_REG_TERM_IND'
'WITHDRWL_TERM_IND'
'FULLTIME_TERM_IND'
'OVR_12HR_TERM_IND

Please reference Appendix: KNN Model and Lime to see the full code creating the one-hot encoding variables.

Model Types Reviewed

Throughout the Summer, numerous model types were considered including Linear Regression, Logistical Regression, Decision Trees, Random Forest, and K Means Clustering. For an overview of the model iterations attempted please see the section “**Key model iteration discussion**”.

In the end, it was settled to analyze two model types, a Random Forest and KNN Model.

As you can see from the chart below the Random Forest model had an overall accuracy score of 91%. For students not predicted to graduate, and students predicted to graduate in the traditional four-year timeline the Random Forest Model was accurate 94% of the time.

Random Forest

	precision	recall	f1-score	support
1	0.98	0.91	0.94	3403
2	0.90	0.98	0.94	9567
3	0.71	0.36	0.48	1269
accuracy			0.91	14239
macro avg	0.86	0.75	0.79	14239
weighted avg	0.90	0.91	0.90	14239

For the KNN Model, the overall accuracy score is lower at 85%. This is in large part due to the low predictability score, 26%, of students who are expected to graduate after 4 years but within 5 years of enrollment. The predictability score for students not expected to graduate, 91%, and students expected to graduate within 4 years, 89%, is still relatively high.

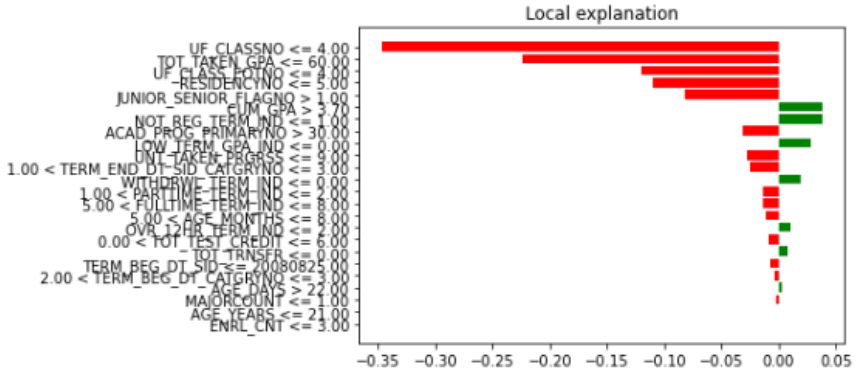
KNN Model

	precision	recall	f1-score	support
1	0.96	0.86	0.91	3495
2	0.85	0.94	0.89	9417
3	0.35	0.20	0.26	1327
accuracy			0.85	14239
macro avg	0.72	0.67	0.69	14239
weighted avg	0.83	0.85	0.84	14239

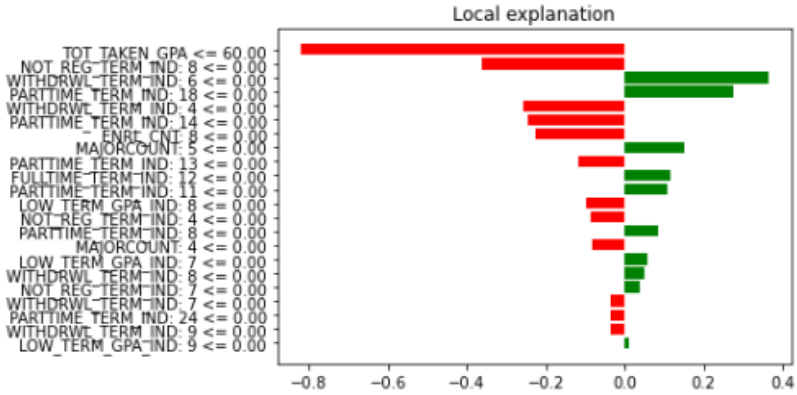
Reasoning for Selection of Final Model Types

We use LIME library to solve for model interpretability by producing locally faithful explanations and test many students' results. As indicated in the charts below for one instance, the top 5 absolute value of a variable contribution in the Random Forest model was -0.35, -0.23, -0.13.-0.12, -0.09, while the top 5 absolute value of a variable contribution in the KNN model was -0.83, -0.4, 0.35, 0.3, -0.28.

Random Forest Lime Report on One-Hot Encoded Variables Not Graduated



KNN Model Lime Report on One-Hot Encoded Variables Not Graduated



While the Random Forest model overall was more accurate, the KNN Model proved to have variables with a higher relative contribution to the prediction. Therefore the KNN Model was utilized for further analysis.

Model Iteration Section: SQL Tables

There were two stages of model iterations, the first was iterations in the SQL data tables utilized and the second was in with the Python models that were built. Please see the Model Iteration Section of this report for a discussion of the process for updating the python tables utilized.

Description of Versions and Functions of SQL Tables in Model

Student Success Analysis

Over the course of the summer dozens of SQL tables were created before analyzing them using a python model for undergraduate student success as measured by 4 or 5 year graduation rates.

Below is a list of the key SQL table iterations, with links to the Github source code.

Tables for Modeling:

UF_R1_SUCCESS_ANALYSIS_UDGD&0709&1719: Created three tables consisting of all undergrad students (UF_R1_SUCCESS_ANALYSIS_UGRD), undergraduates enrolled from 2007-2009 (UF_R1_SUCCESS_ANALYSIS_0709) and undergraduates enrolled from 2017 to 2019 (UF_R1_SUCCESS_ANALYSIS_1719).

UF_R2_SUCCESS_UNDERGRAND: Used in Python Description. Utilized all undergraduate class data from the UF_B_Term table and UF_B_PERSON_STDNT_GRP table.

UF_R3_ANALYSIS_LASTMAJOR0714: Used in Python Description. The Table was updated to reflect a single major per student, and narrowed to undergraduate students from 2007 to 2014.

Tables to Determine Graduation Status:

UF_R3_PROG_STA0719: Create three new variables: MAXSTATUS, MAXEFF, MINEFF. MAXSTATUS stands for the final graduation status of each student who has multiple graduation
status' in their whole college life because they have "graduation status" every semester. For example, There are two graduation status named "activate" and "completion", The former means this student is still in college, and the latter means this student is graduated successfully. The value of MAXEFF tracks graduation time, and the value of MINEFF tracks enrollment time.

TABLE UF_R3_PROG_STA0719v2: Use the variables we created above to create the target variable: "STUDENT_SUCCESS";

Tables to create the other important variables:

UF_SUCCESS_TARGET: Created the variables: LOW_TERM_GPA_IND, PARTTIME_TERM_IND, NOT_REG_TERM_IND, WITHDRWL_TERM_IND, FULLTIME_TERM_IND, OVR_12HR_TERM_IND.

UF_R3_SUCCESS_ANALYSIS_UDINT: Created the variables: TERM_END_DT_SID_CATGRY, TERM_BEG_DT_CATGRY, TERM_LENGTH_CATEGORY, TERM_SEASON

UF_R3_ANALYSIS_MAJORCOUNT2007: Created the variables: MAJORCOUNT

The other functional tables:

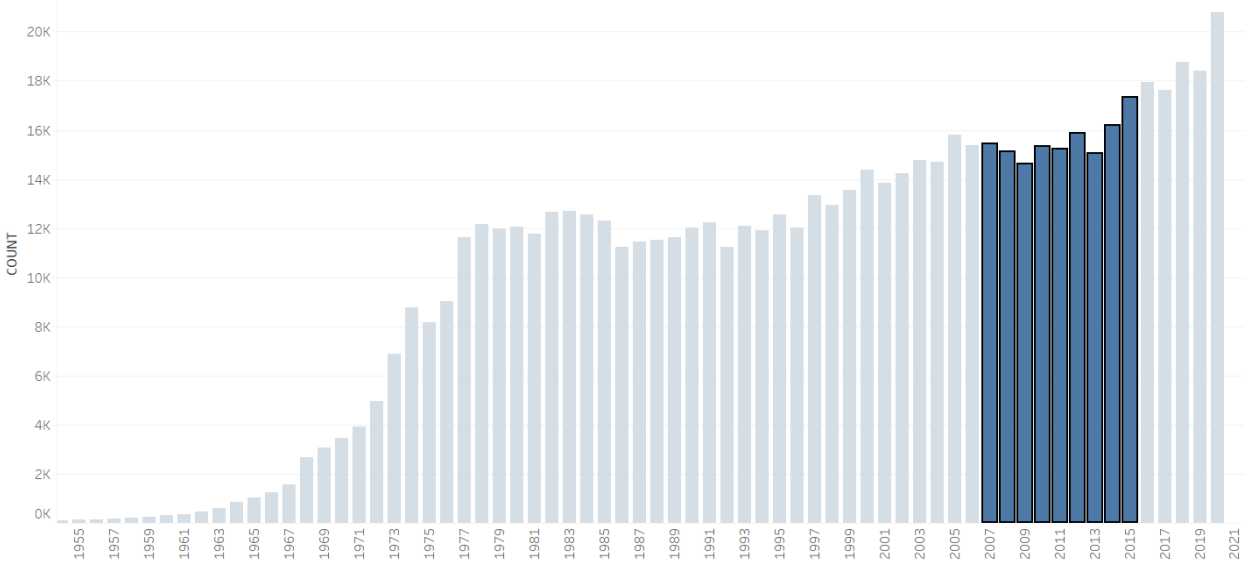
UF_R3_ANALYSIS_DATALISTAGG: Create this table to combine all the information of each student in one row.

Testing Section

Train, Test, Validation Data: Descriptions and Code

Please reference Appendix 9 Testing and Cross Validation to see the python training and testing code created.

The data we used to build and test the model are belong to the students who enrolled in UF undergraduate program between 2007 and 2014.



We have used two methods to ensure model accuracy including:

1) Splitting into Training & Testing Dataset
2) K-Fold Cross Validation

Testing Criteria

Two ways were used to test the criteria. The first way was cutting the dataset directly and using 80% of them as a training dataset and 20% as a testing dataset. The second way was K-Fold Cross Validation: Briefly, Cross-Validation is another way to split the dataset. The training set is split into k smaller sets. A model is trained using k-1 of the folds as training data; the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy). This will result in different k results.

Testing methodology

The dataset used was one to fit a model built by the group. This model was then used to predict the results. After that, the results from the model are compared to the actual result.

Testing Outcomes

1) Test result:
   We split the dataset into two pieces: a training set and a testing set. 80% of observations to training set, 20% of observations to test set. We train the model on the training set and test the model on the testing set.

```
print(x_train.shape)
print(x_test.shape)
```

```
(56957, 161)
(14239, 161)
```

   The dimension of x_train is 56957*161.
   The dimension of x_test is 14239*161.

```
print(y_train.shape)
print(y_test.shape)
```

```
(56957,)
(14239,)
```

The dimension of y_train is 56957*1.
The dimension of y_test is 14239*1.

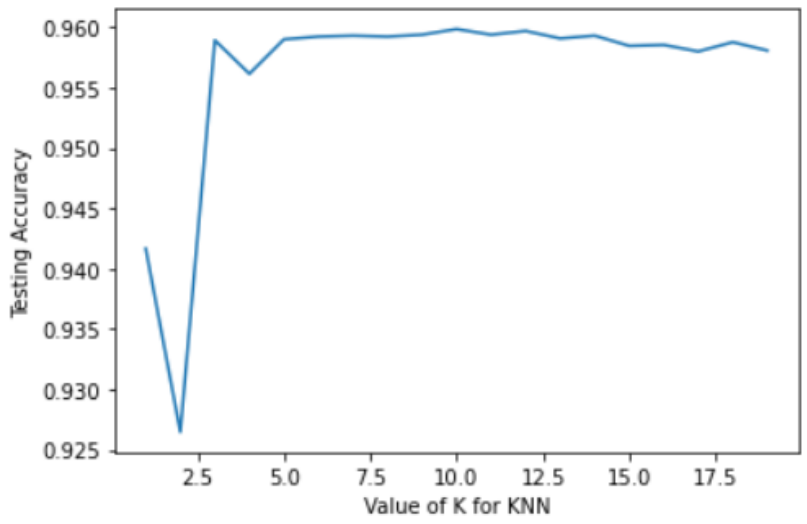
```
k_range = range(1, 20)

scores = []

for k in k_range:
    knn = kNN(n_neighbors=k)
    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    scores.append(metrics.accuracy_score(y_test, y_pred))

print(scores)
```

```
[0.9416621652905317, 0.9264603750289374, 0.9589474496488927, 0.9561694575198704, 0.959024616096921, 0.9592561154410062, 0.9593332818890347, 0.9592561154410062, 0.959410448337063, 0.9598734470252335, 0.959410448337063, 0.9597191141291767, 0.9591017825449495, 0.9593332818890347, 0.9584844509607223, 0.9585616174087507, 0.9580214522725519, 0.9587931167528359, 0.9580986187205803]
```

We created the list “scores” to locate an best value for K, then we created a plot to show the result:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(k_range, scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing Accuracy')
```

```
Text(0, 0.5, 'Testing Accuracy')
```



According to the plot shown above, we chose the value of K = 3.

	precision	recall	f1-score	support
1	0.98	0.91	0.94	3403
2	0.90	0.98	0.94	9567
3	0.71	0.36	0.48	1269
accuracy			0.91	14239
macro avg	0.86	0.75	0.79	14239
weighted avg	0.90	0.91	0.90	14239

Each score of the group of students whose student_success is 3 is significantly lower than other groups.

Our guess: 1) The predictor variables we used is not suitable for analyzing this value. 2) The number of students in group 3 is much smaller than the other two groups.

So we narrowed our target variables from three values (1,2,3) to two values(1,2), and recreated the test result:

score on the testdata: 0.958638783856779

	precision	recall	f1-score	support
1	0.98	0.87	0.92	3459
2	0.95	0.99	0.97	9500
accuracy			0.96	12959
macro avg	0.96	0.93	0.95	12959
weighted avg	0.96	0.96	0.96	12959

2) K-Fold Cross Validation

```
: array([0.73712049, 0.73087909, 0.73415847, 0.73074482, 0.73529412])
```

The result of K-Fold Cross Validation is good, but not perfect.

3) Table:

	STUDENT_SUCCESSNO	Predict
17539	2	2
48990	1	1
16953	2	2
14271	2	2
8958	1	1
...	...	...
19420	2	2
26371	2	2
15094	2	2
61950	2	2
60332	2	2

[12959 rows x 2 columns]

The amount of students whose predict value are different from the “STUDENT_SUCCESSNO” values:

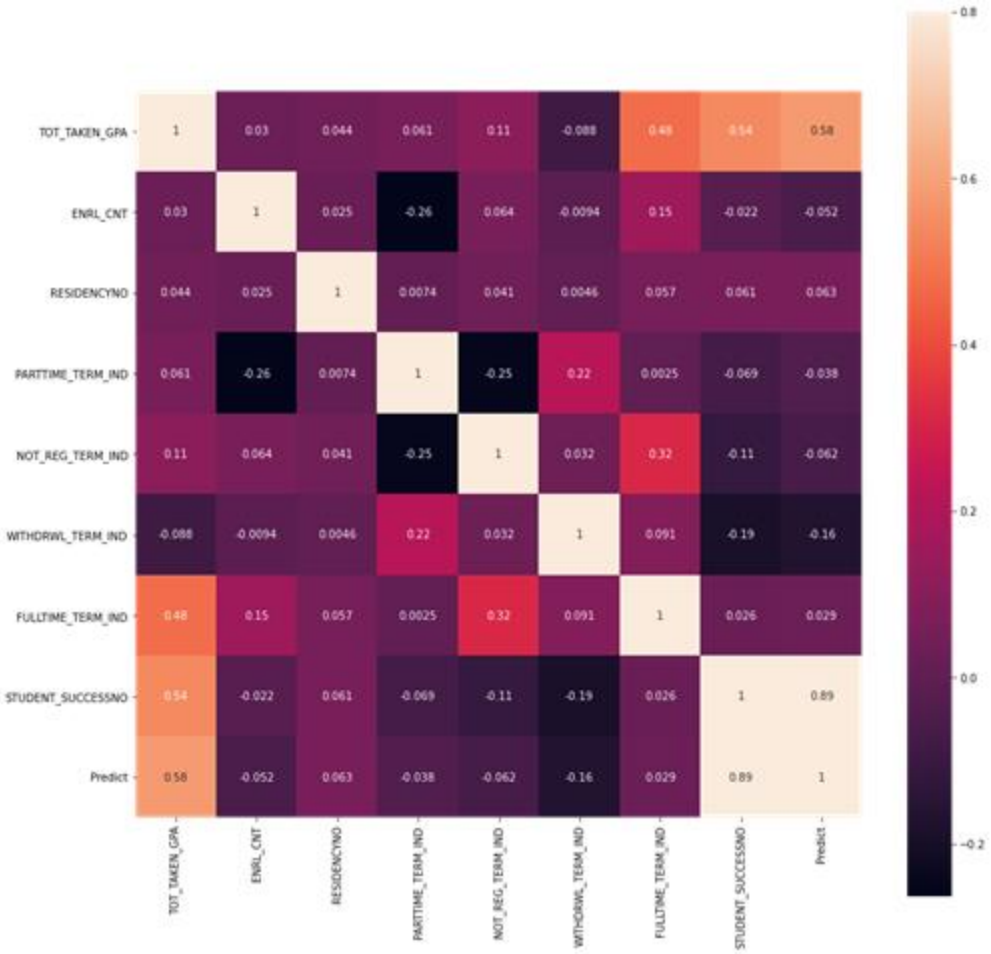
1012	2	1
1013		

The total amount of students in our table:

71196	4952	21
71197		

We use the KNN model to generate predict value for each student enrolled in college between 2007 and 2014, And we compared these values to the “STUDENT_SUCCESSNO” for each student. There are only 1.4% students whose predict value are different from the “STUDENT_SUCCESSNO” values.

4) Confusion Matrix



As we can see from this Confusion Matrix, the correlations of each predictor variables and target variable “STUDENT_SUCCESSNO” are similar to the correlations of each predictor variables and variable “Predict”.

The python code for creating these test outcomes can be accessed on our github page named “Train and Test” and “KNN Model”.

Testing Assessment Criteria

To reduce the size of the data, we narrowed the student records from the years 2007 to 2014. These results were used to test the model. Anything after 2015 was not included as students could be currently taking classes and were not expected to graduate.

Since the overall goal was to help find students in danger of not graduating, it was decided to focus on those who should have finished their undergraduate degree. From here, we ran five instances each of the KNN model for students who graduated and students who did not graduate to find the most influential variables.

Overall, we could have used more outside data to combine with the data. For example; from discussions with advisors and coaches, it was questioned if co-curricular activities were included in the research. These co-curricular could help students with their classes and likewise affect their GPA.

Assessment Criteria & Business Outcome Criteria

The original project goal was to utilize data to build an analytic model to predict the probability of student success given academic characteristics. The plan was for the model to be used to provide further insights for coaches working directly with students.

Assessment Outcomes: Python Model Iteration

Key Model Iteration Discussion

There were two stages of model iterations, the first was iterations in the SQL data tables utilized and the second was in with the Python models that were built.

List of Models Developed and Assessed

Over the course of the summer we assessed dozens of models for their ability to predict undergraduate student success as measured by 4 or 5 year graduation rates.

Below is a list of the key model iterations, with links to the Github source code. Under the name given to each model iteration, is a list of the types of models assessed. The naming convention for model names was generally Date_Table_Name_ModelType.

7.14.20_Model_Linear&Tree&NearestNeighbors
Linear Regression Model
Decision Tree Model
KNN Model

7.14.20_UF_R2_UGRDKMEANREDO2007_KMeans
K Means Clustering

7.17.20_Principal_Component_Analysis_Reduction_Technique
Random Forest

7.18.20_UF_R3_ANALYSIS_LASTMAJOR0714_Random_Forest_Model&LIME
Random Forest
LIME

7.22.20_UF_R3_ANALYSIS_LASTMAJOR0714_KNNModel&LIME
KNN Model
LIME

Reasons for Moving to Another Iteration of the Model

For our model iterations before July 18th we had yet to find a model whose result did not predict a perfect hundred percent success rate. Realizing that a perfect success rate was hard to achieve we experimented with dozens of changes, with the key updates listed below.

7.14.20_Broken_Model_Linear_Tree_NearestNeighbors: Baseline model, model types were chosen based on their perceived ability to work with categorical dependent variables.

7.14.20_UF_R2_UGRDKMEANREDO2007_KMeans:Believing that our success rate had to do with the model type we switched to a K Means Clustering model.

7.17.20_Principal_Component_Analysis_Reduction_Technique: Based on the assumption that our data size was contributing to a faulty model we conducted a Principal Component Analysis.

Starting on July 18th we began to incorporate the use of the Lime package into our model iterations. Typically the categorical models we were utilizing provide a general prediction of student X will graduate or will not graduate. They do not provide information on why specific students are determined likely or unlikely to graduate.

Lime is a package used in Python to explain the relative contribution of a variable to the prediction of a SINGLE student's likelihood to graduate. While insightful on a individual student basis, results from Lime should not be read as variable X is a significant predictor to every student's likelihood to graduate.

7.18.20_UF_R3_ANALYSIS_LASTMAJOR0714_Random_Forest_Model&LIME:
Utilized the Lime package to indicate which variables contributed to an individual student's prediction

7.22.20_UF_R3_ANALYSIS_LASTMAJOR0714_KNNModel&LIME:
Utilized One-hot Encoding and the Lime package to indicate which variables contributed to an individual student's prediction

Parameter Changes Made in Each Iteration

7.14.20_Model_Linear&Tree&NearestNeighbors:
None, Baseline Model.

7.14.20_UF_R2_UGRDKMEANREDO2007_KMeans:
New model type, K Means Clustering Utilized.

7.17.20_Principal_Component_Analysis_Reduction_Technique:
Conducted a Principal Component Analysis (PCA), a variable reduction method used on large datasets that keeps the integrity of the information. For example if the age of the student was calculated using three variables, Years, Months, and Days, a PCA may find that only the age of the student in years is relevant to the overall modeling assumptions.

7.18.20_UF_R3_ANALYSIS_LASTMAJOR0714_Random_Forest_Model&LIME:
Utilized the Lime package to indicate which variables contributed to an individual student's prediction.

7.22.20_UF_R3_ANALYSIS_LASTMAJOR0714_KNNModel&LIME: Utilized One-hot Encoding and the Lime package to indicate which variables contributed to an individual student's prediction.

Final Model Description Section

Data Requirements

Please reference the data selection, data cleaning, and data mining steps in the Data Description Report published on our github page.

Model Type

As shown in the Model Selection Section part in this report and our github page, we have created four model: Linear Regression, Decision Tree, Random Forest and KNN model. Considering the accuracy of these models and the model interpretability of the variables in these models, we finally chose the KNN model as the optimal model for our project to predict whether a 4-year undergraduate student will graduate successfully.

The code to create the KNN model and the usage of LIME package is available on Github.

Required Variables and Sources

The variables are from many different tables. Some of them are created by us, the other are collected from the data lake. The source of each variable is shown as below:

Table UF_B_STDNT_TERM:
PERSON_SID, TERM_SID, AGE_YEARS, AGE_MONTHS, AGE_DAYS, TOT_TRNSFR, TOT_TEST_CREDIT, UNT_TAKEN_PRGRSS, CUM_GPA, ENRL_CNT, ACAD_PROG_PRIMARY, TERM_BEG_DT_SID, TOT_TAKEN_GPA, UNT_TAKEN_GPA.

Table UF_R3_ANALYSIS_NODATE2007:
JUNIOR_SENIOR_FLAGNO, UF_CLASSNO, RESIDENCYNO, ACAD_PROG_PRIMARYNO, UF_CLASS_EOTNO.

Table UF_R3_ANALYSIS_MAJORCOUNT2007:
MAJORCOUNT.

Table UF_SUCCESS_TARGET:
LOW_TERM_GPA_IND,PARTTIME_TERM_IND,NOT_REG_TERM_IND,
WITHDRWL_TERM_IND, FULLTIME_TERM_IND, OVR_12HR_TERM_IND.

Table UF_R3_SUCCESS_ANALYSIS_UDINT:
TERM_END_DT_SID_CATGRYNO, TERM_BEG_DT_CATGRYNO,
STUDENT_SUCCESSNO.

Model Documentation

Tool Utilized: The model for student success to predict which undergraduate students are likely to graduate was built using the following Tools:
SQL, R: data cleaning, data mining and table creating.
Python: Use the packages including Pandas, SkLearn, Numpy, MatPlotlib, Seaborn, and Lime to build the model.
Github: code saving and sharing.
Tableau: data visualization.

Data Sources: UF_B_STDNT_TERM, UF_D_ACAD_PROG, UF_F_CPPS in the University of Florida Data Lake environment created by the UF IT staff on behalf of the University of Florida Provost office.

Preparation Steps: For a full report on the data preparation steps utilized to create the SQL tables in our Python Model please reference the latest version of the Data Description report on Github.

Modeling&Test Code: The final modeling code utilized is in 7.22.20_UF_R3_ANALYSIS_LASTMAJOR0714_KNNModel&LIME. The final test code utilized is in Train and Test. For additional insights into the iterations of modeling code please reference Appendixes 1-10 or the main Python Github folder.

Version Number: Based on our naming convention our third versioning of our model was utilized. While we had dozens of sub-versions all of the first versions focused on narrowing our SQL tables to the correct universe. Our second versions experimented with different types of python models including Logistical Regression, Decision Trees, Random Forest and final a KNN model.

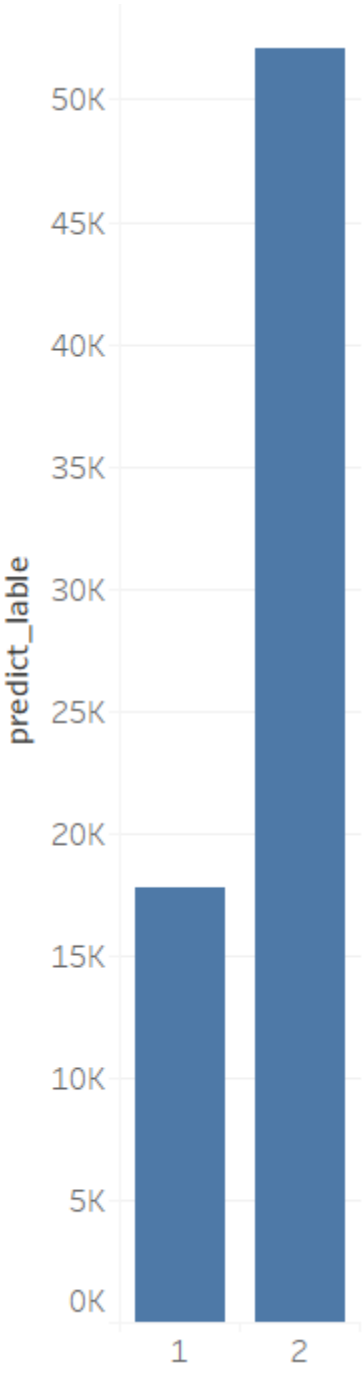
Model Discussion

After we built the KNN model and finished testing its accuracy, we used this model to try to make prediction for the students who enrolled in UF undergraduate program after 2014.

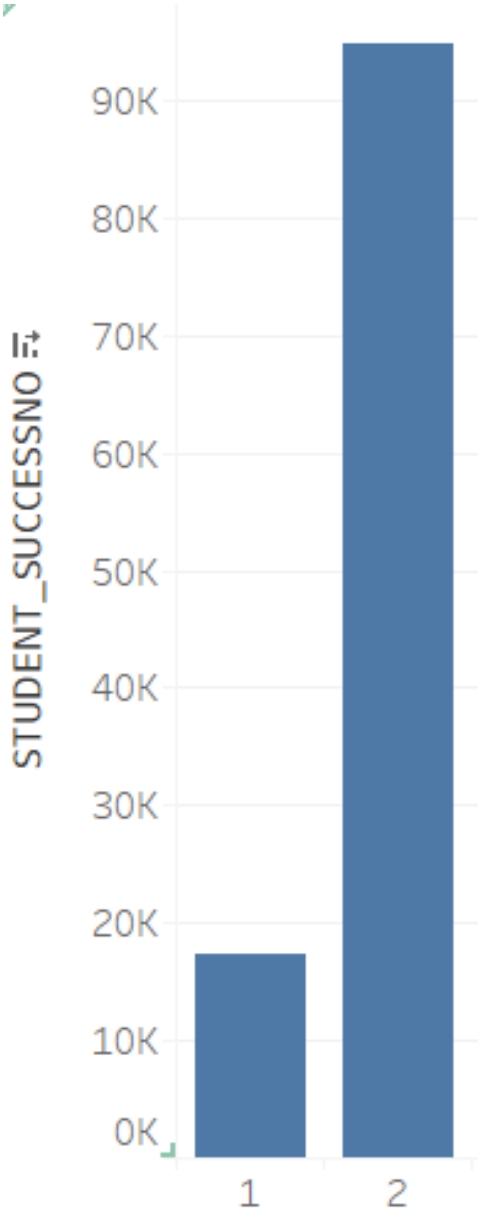
Please refer to this github file for the python code completing this prediction.

Please see the prediction result as shown in Predict Result.

We used Tableau to see the number of students who predicted graduate successfully and the number of students who predicted not graduate successfully. The result is shown below:



2015-



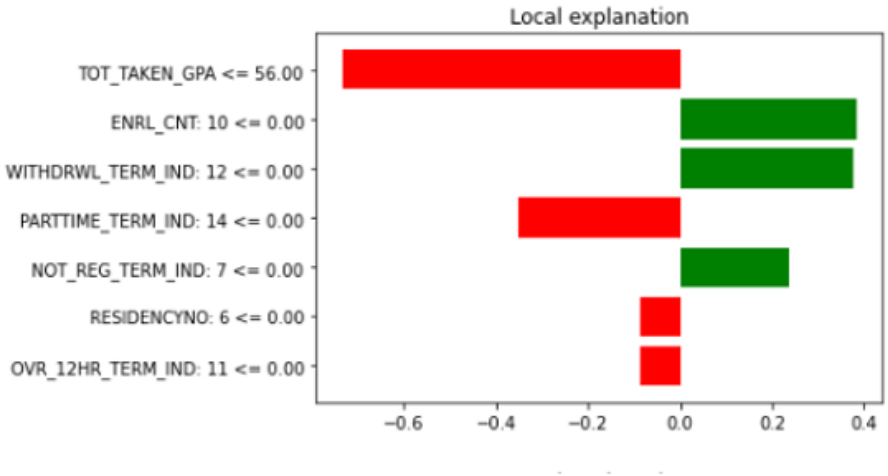
2007-2014

As shown in the left Histogram, The number of students in group 2 is much larger than the number of students in group1, which is consistent with common sense and is similar to the values of target variable belong to the students enrolled between 2007 and 2014(shown in the right Histogram).

Student Success Analysis

We selected one student from each group as examples, and use the LIME package to analysis the top seven variables which have higher model interpretability.

For the student who predicted not graduated successfully:

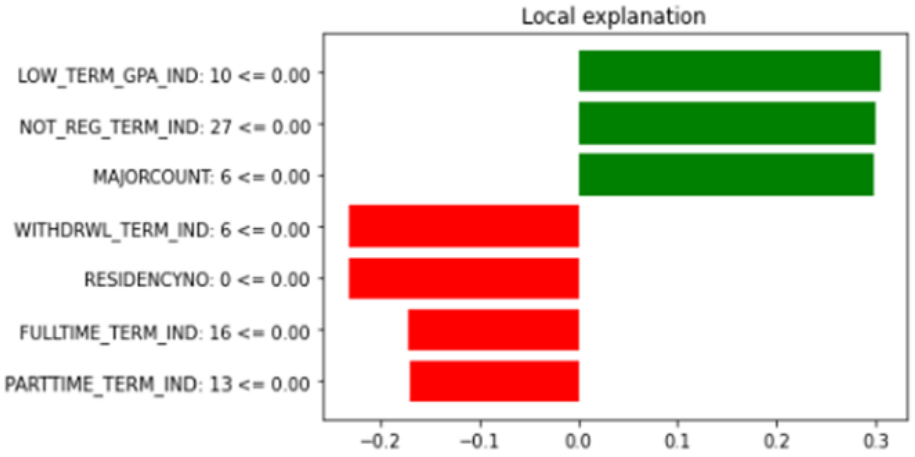


The meaning of the variables in the chart above:
Enrollment count 10: The Number of Enrolled Classes for the last term is 4.
Withdrawal Term ind 12: The number of semesters this student withdrew is 0.
Part Time Term ind 14: The number of Part Time semesters is 3.
Not register term in 7: The number of semesters not registered is 1
Residencyno 6: The number of Residency no is 2, which is Temporary FL residents.
Ovr 12hr term ind 11: The number of semesters which selected more than 12 credit hours is 3.

As we can see from this chart, Enrollment count 10, Withdrawal Term ind 12, and Not register term ind 7 have positive effects on this student's success graduation.
This student's number of enrolled classes for the last term is normal, which may indicate that he/she didn't have to register more classes than normal to get enough credits or register less class than normal because of the poor learning ability. Also, this student did not withdraw any semester, indicating that he has confidence in his learning ability. There is one semester this student didn't register any class. It is good for him/her to use this semester to build other abilities, which may improve his/her academic performance.
The other variables have negative effects on this student's success graduation.
The number of Part Time semesters indicates that this student may didn't have enough time to focus on study. However, there are 3 semesters this student registered more than 12 credit hours. This may increase the pressure of his studies. Be a temporary FL resident also adds some challenges to his studies. This may be related to the living cost, the living habits of different regions, etc.

Student Success Analysis

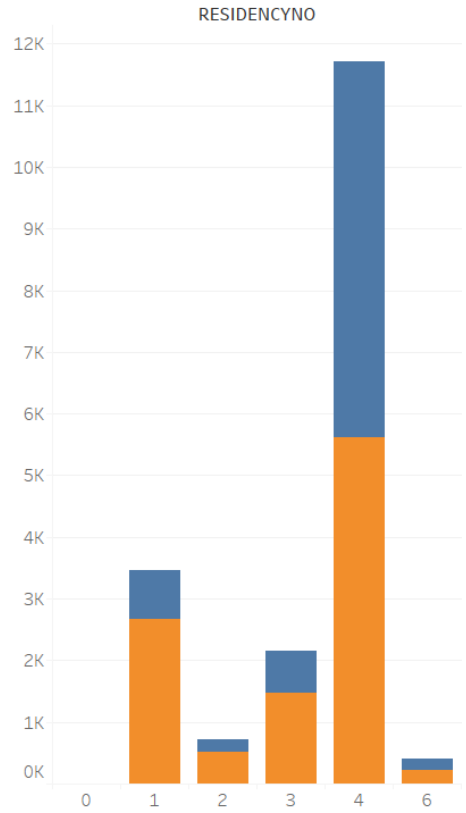
For the student who predicted graduated successfully:



The meaning of the variables in the chart above:
Low_term_GPA:10: The number of semesters the student has enrolled and didn't withdrew, but the GPA is less than 2.2 is 0.
The number of semesters not registered is 2
Major Count 6: This student has taken 2 majors.
Withdrawal Term: The number of semesters withdrew is 0.
Residency: The number of Residency no is 4, which is Non-FL Resident.
FULLTIME_TERM_IND 16: # of semesters the student was enrolled full-time is 7.
Part Time Term 13: The number of Part Time semesters is 2.

As we can see from this chart, this student never got low GPA in any of his semester. He/she also didn't register any class for two semesters. This experience not only has no negative impact on study performance, but also enables students to have a more reasonable schedule plan. This student is studying for a double degree, which may indicate that he/she has a strong learning ability.
Although he/she didn't withdraw any semester, this has a negative effect on his/her graduation. This student is Non-FL Resident, which may cause the same problem as previous student. The last two variables indicate that he/she does not have enough time to focus on learning.

We check the impact of the variable “residencyno” of students enrolled after 2015 on graduation, and created a chart:



The meaning of the variables in the chart above:
1: Resident International FL VISA
2: Temporary Florida Resident Tuition Exemption
3: Non-Resident International Non-Florida
4: Non-Florida Resident
6: Legal Resident International Non-Florida
We took the value 5 out of this Chart because it is much larger than others, and most of the students in this category predicted graduated successfully.
As we can see, the proportion of people in the category 4 who are predicted can’t graduate is much larger than the other groups. A deeper dive is worth to be done for this variable.

Model Recommendations Section

We recommend that a future team takes the following steps before deploying our model on a student graduation dashboard.

Additional Data

Both within the UF Data Lake and from publicly available sources there are additional datasets that can be joined with the UF_B_Term data that was utilized for our model prediction.

The UF student coaches suggested that data ideniting which schools students transferred from, information on High Schools students attended, and information on what extra curricular activities students participated once at the University would be valuable.

Other datasets that we discussed adding were for socio-economic indicators of the geographic areas students were from, large scale disasters such as hurricanes that could impact a student's ability to remain in Gainesville and following students who were admitted to the University in the year following a football championship.

Any one of these data sources could be added to the data in our model to update the prediction ability of students who are more or less likely to graduate successfully in four years.

Appendixes

Appendix 1: Target Variable Created

Github Link

First, we created three new variables, which are MAXSTATUS, MAXEFF and MINEFF. MAXSTATUS stands for the final graduation status of each student who has multiple graduation status' in their whole college life because they have "graduation status" every semester. For example, There are two graduation status named "activate" and "completion", The former means this student is still in college, and the latter means this student is graduated successfully. The value of MAXEFF tracks graduation time, and the value of MINEFF tracks enrollment time. The SQL code for creating these variables are shown as below:

```
(SELECT MAX(B_ED.PROG_STATUS_SID)
        FROM UF_F_CPPS B_ED
                WHERE A.PERSON_SID = B_ED.PERSON_SID
                  AND A.ACAD_PLAN = B_ED.ACAD_PLAN) as MAXSTATUS,
(SELECT MAX(B_ED.EFFDT_SID)
        FROM UF_F_CPPS B_ED
                WHERE A.PERSON_SID = B_ED.PERSON_SID
                  AND A.ACAD_PLAN = B_ED.ACAD_PLAN) as MAXEFF,
(SELECT MIN(B_ED.EFFDT_SID)
        FROM UF_F_CPPS B_ED
                WHERE A.PERSON_SID = B_ED.PERSON_SID
                  AND A.ACAD_PLAN = B_ED.ACAD_PLAN) AS MINEFF
```

Then we use these variables to create the target variable "STUDENT_SUCCESS". The SQL code is shown as below:

```
(case when A.MAXSTATUS = 4 AND (A.MAXEFF - A.MINEFF) >=39900 then ('5yrs')
when A.MAXSTATUS = 4 AND (A.MAXEFF - A.MINEFF <39900) then ('4yrs')
else ('no')
end)AS "STUDENT_SUCCESS"
```

Appendix 2: Predictor Variables Created

Github Link

Please see the explanation of each table in github link above.

```
CREATE TABLE UF_SUCCESS_TARGET as (SELECT
   PERSON_SID,
   SUM(CASE WHEN "CUR_GPA" <= 2.2 AND "ACADEMIC_LOAD" <> 'N' AND
UNT_TAKEN_GPA<> 0 THEN 1 ELSE 0 END) AS "LOW_TERM_GPA_IND",
   SUM(CASE WHEN "ACADEMIC_LOAD" in ('H','L','T') THEN 1 ELSE 0 END) AS
"PARTTIME_TERM_IND",
   SUM(CASE WHEN "ACADEMIC_LOAD" = 'N' THEN 1 ELSE 0 END) AS
"NOT_REG_TERM_IND",
   SUM(CASE WHEN "ACADEMIC_LOAD" <> 'N' AND UNT_TAKEN_GPA=0  THEN 1 ELSE 0
END) AS "WITHDRWL_TERM_IND",
   SUM(CASE WHEN "ACADEMIC_LOAD" = 'F' THEN 1 ELSE 0 END) AS
"FULLTIME_TERM_IND",
   SUM(CASE WHEN "UNT_TAKEN_GPA" > 12 THEN 1 ELSE 0 END) AS
"OVR_12HR_TERM_IND"
FROM UF_B_STDNT_TERM where ACAD_CAREER ='UGRD'
GROUP BY PERSON_SID)

CREATE TABLE UF_R3_PROG_STA0719 AS(
select distinct
A.PERSON_SID,
A.ACAD_PROG,
A.ACAD_PLAN,
B.ACAD_PLAN_TYPE_LD,
(SELECT MAX(B_ED.PROG_STATUS_SID)
        FROM UF_F_CPPS B_ED
                WHERE A.PERSON_SID = B_ED.PERSON_SID
                  AND A.ACAD_PLAN = B_ED.ACAD_PLAN) as MAXSTATUS,
(SELECT MAX(B_ED.EFFDT_SID)
        FROM UF_F_CPPS B_ED
                WHERE A.PERSON_SID = B_ED.PERSON_SID
                  AND A.ACAD_PLAN = B_ED.ACAD_PLAN) as MAXEFF,
(SELECT MIN(B_ED.EFFDT_SID)
        FROM UF_F_CPPS B_ED
                WHERE A.PERSON_SID = B_ED.PERSON_SID
                  AND A.ACAD_PLAN = B_ED.ACAD_PLAN) AS MINEFF
FROM UF_F_CPPS A INNER JOIN UFDL_PWCBA_DATA.uf_d_acad_plan B

ON (A.ACAD_PLAN = B.ACAD_PLAN_CD
```

```
AND A.ACAD_CAR_SID =8)
)

CREATE TABLE UF_R3_PROG_STA0719v2 AS(
SELECT
A.PERSON_SID,
A.ACAD_PROG,
A.ACAD_PLAN,
A.ACAD_PLAN_TYPE_LD,
A.MAXSTATUS,
(case when A.MAXSTATUS = 4 then (1)
else (0)
end)AS "SUCCESS_GRADUATES",
A.MAXEFF,
A.MINEFF,
(CASE WHEN substr(A.MAXEFF,-4)<=0508 THEN (substr(A.MAXEFF,0,4)||'01')
WHEN substr(A.MAXEFF,-4)>0508 AND substr(A.MAXEFF,-4)<0819 THEN
(substr(A.MAXEFF,0,4)||'05')
WHEN substr(A.MAXEFF,-4)>= 0819 THEN (substr(A.MAXEFF,0,4)||'08')
END) AS BEGDATE,
(CASE WHEN substr(A.MAXEFF,-4)<=0508 THEN (substr(A.MAXEFF,0,4)||'05')
WHEN substr(A.MAXEFF,-4)>0508 AND substr(A.MAXEFF,-4)<0819 THEN
(substr(A.MAXEFF,0,4)||'08')
WHEN substr(A.MAXEFF,-4)>= 0819 THEN (substr(A.MAXEFF,0,4)||'12')
END) AS ENDDATE,
(case when A.MAXSTATUS = 4 AND (A.MAXEFF - A.MINEFF) >=39900 then ('5yrs')
when A.MAXSTATUS = 4 AND (A.MAXEFF - A.MINEFF <39900) then ('4yrs')
else ('no')
end)AS "STUDENT_SUCCESS"
FROM UF_R3_PROG_STA0719 A
)

CREATE TABLE UF_R3_SUCCESS_ANALYSIS_UDINT as (SELECT
A.ACAD_CAR_SID,
A.INSTITUTION_SID,
A.STDNT_CAR_NBR,
A.TERM_SID,
A.STRM,
A.TERM_BEG_DT_SID,
A.TERM_END_DT_SID,
A.INSTITUTION,
A.ACAD_CAREER,
A.AGE_YEARS,
A.AGE_MONTHS,
```

```
A.AGE_DAYS,
A.TOT_CUMULATIVE,
A.JUNIOR_SENIOR_FLAG,
A.TOT_TAKEN_PRGRSS,
A.TOT_TRNSFR,
A.TOT_TEST_CREDIT,
A.TOT_OTHER,
A.TOT_PASSD_PRGRSS,
A.UNT_TAKEN_PRGRSS,
A.CUR_GPA,
A.CUM_GPA,
A.ENRL_CNT,
A.ENRL_FLG,
A.SSR_TRMAC_LAST_DT,
A.ACAD_LEVEL_BOT,
A.ACAD_LEVEL_EOT,
A.UF_CLASS,
A.RESIDENCY,
A.LASTUPD_EW_DTTM,
A.PROF_GRAD_FLAG,
A.ACADEMIC_LOAD,
A.TOT_GRADE_POINTS,
A.TOT_TAKEN_GPA,
A.ENRL_SUMMER_A_FLAG,
A.ENRL_SUMMER_B_FLAG,
A.ENRL_SUMMER_C_FLAG,
A.ACAD_PROG_PRIMARY,
A.UF_CLASS_EOT,
A.UNT_TAKEN_GPA,
A.UNT_INPROG_GPA,
A.TERM_FIRST_APPT_TIME,
   (CASE WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 AND
substr(A.TERM_END_DT_SID,-4)<=0430 THEN( 'EARLY')
   WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 AND substr(A.TERM_END_DT_SID,-
4)>0430 AND  substr(A.TERM_END_DT_SID,-4)<=0502 THEN( 'AVERAGE')
   WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 AND substr(A.TERM_END_DT_SID,-
4)>0502 THEN( 'LATE')
   WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 AND substr(A.TERM_END_DT_SID,-
4)<=1214 THEN( 'EARLY')
   WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 AND substr(A.TERM_END_DT_SID,-
4)>1214 AND substr(A.TERM_END_DT_SID,-4)<=1217 THEN( 'AVERAGE')
   WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 AND substr(A.TERM_END_DT_SID,-
4)>1220 THEN( 'LATE')
   ELSE NULL
```

```
   END )AS "TERM_END_DT_SID_CATGRY",
   (CASE WHEN substr(A.TERM_BEG_DT_SID,-4)<=0105 THEN ('EARLY')
   WHEN substr(A.TERM_BEG_DT_SID,-4)>0105 AND substr(A.TERM_BEG_DT_SID,-
4)<=0107 THEN ('AVERAGE')
    WHEN substr(A.TERM_BEG_DT_SID,-4)>0107 AND substr(A.TERM_BEG_DT_SID,-
4)<=0230 THEN ( 'LATE')
    WHEN substr(A.TERM_BEG_DT_SID,-4)>0800 AND substr(A.TERM_BEG_DT_SID,-
4)<=0822 THEN ('EARLY')
   WHEN substr(A.TERM_BEG_DT_SID,-4)>0822 AND substr(A.TERM_BEG_DT_SID,-
4)<=0825 THEN ('AVERAGE')
   WHEN substr(A.TERM_BEG_DT_SID,-4)>0825 THEN ( 'LATE')
    ELSE NULL
    END) AS "TERM_BEG_DT_CATGRY",
    ( CASE WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) <=115 THEN('SHORT')
    WHEN WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) >115 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) <=116 THEN ('AVERAGE')
    WHEN WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) >116 THEN('LONG')
    WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) <=109 THEN('SHORT')
    WHEN WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) >109 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) <=114 THEN ('AVERAGE')
    WHEN WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 AND
(TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) >114 THEN('LONG')
    END) AS "TERM_LENGTH_CATEGORY",
    (TO_DATE(A.TERM_END_DT_SID,'YYYYMMDD')-
TO_DATE(A.TERM_BEG_DT_SID,'YYYYMMDD')) AS "TERM_LENGTH_DAYS",
    (CASE WHEN substr(A.TERM_BEG_DT_SID,-4)>=0800 THEN ('fall')
     WHEN substr(A.TERM_BEG_DT_SID,-4)<=0230 THEN ('spring')
     ELSE ('summer')
     END) AS "TERM_SEASON",
     B.PERSON_SID,
B.LOW_TERM_GPA_IND,
```

```
B.PARTTIME_TERM_IND,
B.NOT_REG_TERM_IND,
B.WITHDRWL_TERM_IND,
B.FULLTIME_TERM_IND,
B.OVR_12HR_TERM_IND
   FROM UF_SUCCESS_TARGET B INNER JOIN UF_B_STDNT_TERM A
   ON A.PERSON_SID = B.PERSON_SID
   WHERE A.ACAD_CAR_SID = 8
   AND A.CUR_GPA<>0
)

CREATE TABLE UF_R3_ANALYSIS_GRASUSDATE2007 AS(
SELECT distinct
A.PERSON_SID,
A.ACAD_CAR_SID,
A.INSTITUTION_SID,
A.STDNT_CAR_NBR,
A.TERM_SID,
A.STRM,
A.TERM_BEG_DT_SID,
A.TERM_END_DT_SID,
A.INSTITUTION,
A.ACAD_CAREER,
A.AGE_YEARS,
A.AGE_MONTHS,
A.AGE_DAYS,
A.TOT_CUMULATIVE,
A.JUNIOR_SENIOR_FLAG,
A.TOT_TAKEN_PRGRSS,
A.TOT_TRNSFR,
A.TOT_TEST_CREDIT,
A.TOT_OTHER,
A.TOT_PASSD_PRGRSS,
A.UNT_TAKEN_PRGRSS,
A.CUR_GPA,
A.CUM_GPA,
A.ENRL_CNT,
A.ENRL_FLG,
A.SSR_TRMAC_LAST_DT,
A.ACAD_LEVEL_BOT,
A.ACAD_LEVEL_EOT,
A.UF_CLASS,
A.RESIDENCY,
A.LASTUPD_EW_DTTM,
```

```
A.ACADEMIC_LOAD,
A.TOT_GRADE_POINTS,
A.TOT_TAKEN_GPA,
A.ENRL_SUMMER_A_FLAG,
A.ENRL_SUMMER_B_FLAG,
A.ENRL_SUMMER_C_FLAG,
A.ACAD_PROG_PRIMARY,
A.UF_CLASS_EOT,
A.UNT_TAKEN_GPA,
A.UNT_INPROG_GPA,
A.TERM_FIRST_APPT_TIME,
A.TERM_END_DT_SID_CATGRY,
A.TERM_BEG_DT_CATGRY,
A.TERM_LENGTH_CATEGORY,
A.TERM_LENGTH_DAYS,
A.TERM_SEASON,
A.LOW_TERM_GPA_IND,
A.PARTTIME_TERM_IND,
A.NOT_REG_TERM_IND,
A.WITHDRWL_TERM_IND,
A.FULLTIME_TERM_IND,
A.OVR_12HR_TERM_IND,
B.SUCCESS_GRADUATES,
B.STUDENT_SUCCESS
FROM UF_R3_SUCCESS_ANALYSIS_UDINT A
INNER JOIN UF_R3_PROG_STA0719V2 B
ON A.PERSON_SID = B.PERSON_SID
AND substr(A.TERM_BEG_DT_SID,0,6) = B.BEGDATE
AND substr(A.TERM_END_DT_SID,0,6) = B.ENDDATE
AND A.ACAD_PROG_PRIMARY = B.ACAD_PROG
AND substr(A.TERM_BEG_DT_SID,0,4) >=2007
where B.ACAD_PLAN_TYPE_LD = 'Major'
)
create table UF_R3_ANALYSIS_N0DATE2007 AS
(
SELECT
PERSON_SID,
TERM_SID,
AGE_YEARS,
AGE_MONTHS,
AGE_DAYS,
(CASE WHEN JUNIOR_SENIOR_FLAG = 'Y' THEN (1)
ELSE(2)
END)AS JUNIOR_SENIOR_FLAGNO,
```

```
TOT_TRNSFR,
TOT_TEST_CREDIT,
UNT_TAKEN_PRGRSS,
CUM_GPA,
ENRL_CNT,
ACAD_PROG_PRIMARY,
TERM_BEG_DT_SID,
(
CASE WHEN UF_CLASS = '0' THEN(0)
WHEN UF_CLASS = '1' THEN(1)
WHEN UF_CLASS = '2' THEN(2)
WHEN UF_CLASS = '3' THEN(3)
WHEN UF_CLASS = '4' THEN(4)
WHEN UF_CLASS = '5' THEN(5)
WHEN UF_CLASS = '6' THEN(6)
ELSE(7)
END
)AS
UF_CLASSNO,
(CASE WHEN RESIDENCY='R' THEN(1)
WHEN RESIDENCY='T' THEN(2)
WHEN RESIDENCY='A'THEN(3)
WHEN RESIDENCY = 'N'THEN(4)
WHEN RESIDENCY='F' THEN(5)
WHEN RESIDENCY='E' THEN(6)
ELSE(0)
END) AS RESIDENCYNO,
TOT_TAKEN_GPA,
(CASE WHEN ACAD_PROG_PRIMARY='UNVEM'THEN(1)
WHEN ACAD_PROG_PRIMARY='UNENG' THEN(2)
WHEN ACAD_PROG_PRIMARY='UGJRC' THEN(3)
WHEN ACAD_PROG_PRIMARY='UGLAS' THEN(4)
WHEN ACAD_PROG_PRIMARY='UGEDU' THEN(5)
WHEN ACAD_PROG_PRIMARY='UNLAS' THEN(6)
WHEN ACAD_PROG_PRIMARY='UGCMN' THEN(7)
WHEN ACAD_PROG_PRIMARY='UGACT' THEN(8)
WHEN ACAD_PROG_PRIMARY='UGPHM' THEN(9)
WHEN ACAD_PROG_PRIMARY='UGAGL' THEN(10)
WHEN ACAD_PROG_PRIMARY='UGHHU' THEN(11)
WHEN ACAD_PROG_PRIMARY='UNAGL' THEN(12)
WHEN ACAD_PROG_PRIMARY='UGNUR' THEN(13)
WHEN ACAD_PROG_PRIMARY='UGART' THEN(14)
WHEN ACAD_PROG_PRIMARY='UNMED' THEN(15)
WHEN ACAD_PROG_PRIMARY='UGNTR' THEN(16)
```

```
WHEN ACAD_PROG_PRIMARY='UNPHM' THEN(17)
WHEN ACAD_PROG_PRIMARY='UGMED' THEN(18)
WHEN ACAD_PROG_PRIMARY='UGDCP' THEN(19)
WHEN ACAD_PROG_PRIMARY='UNEDU' THEN(20)
WHEN ACAD_PROG_PRIMARY='UNDCP' THEN(21)
WHEN ACAD_PROG_PRIMARY='UNPBH' THEN(22)
WHEN ACAD_PROG_PRIMARY='UNNUR' THEN(23)
WHEN ACAD_PROG_PRIMARY='UGPBH' THEN(24)
WHEN ACAD_PROG_PRIMARY='UNDEN' THEN(25)
WHEN ACAD_PROG_PRIMARY='UNACT' THEN(26)
WHEN ACAD_PROG_PRIMARY='UGNUS' THEN(27)
WHEN ACAD_PROG_PRIMARY='UNBUS' THEN(28)
WHEN ACAD_PROG_PRIMARY='UNCMN' THEN(29)
WHEN ACAD_PROG_PRIMARY='UGENG' THEN(30)
WHEN ACAD_PROG_PRIMARY='UNART' THEN(31)
ELSE(32)
END) AS ACAD_PROG_PRIMARYNO,
(
CASE WHEN UF_CLASS_EOT = '0' THEN(0)
WHEN UF_CLASS_EOT = '1' THEN(1)
WHEN UF_CLASS_EOT = '2' THEN(2)
WHEN UF_CLASS_EOT ='3' THEN(3)
WHEN UF_CLASS_EOT = '4' THEN(4)
WHEN UF_CLASS_EOT = '5' THEN(5)
WHEN UF_CLASS_EOT = '6' THEN(6)
ELSE(7)
END
)AS
UF_CLASS_EOTNO,
UNT_TAKEN_GPA,
(CASE WHEN TERM_END_DT_SID_CATGRY='LATE' THEN(1)
WHEN TERM_END_DT_SID_CATGRY ='EARLY' THEN(2)
ELSE(3)
END)AS
TERM_END_DT_SID_CATGRYNO,
(CASE WHEN TERM_BEG_DT_CATGRY='LATE' THEN(1)
WHEN TERM_BEG_DT_CATGRY ='EARLY' THEN(2)
ELSE(3)
END)AS
TERM_BEG_DT_CATGRYNO,
(CASE WHEN TERM_LENGTH_CATEGORY='LATE' THEN(1)
WHEN TERM_LENGTH_CATEGORY ='EARLY' THEN(2)
ELSE(3)
END)AS
```

```
TERM_LENGTH_CATEGORYNO,
TERM_LENGTH_DAYS,
SUCCESS_GRADUATES,
(CASE WHEN STUDENT_SUCCESS = 'no' then(1)
when STUDENT_SUCCESS ='4yrs' then(2)
else(3)
end)as
STUDENT_SUCCESSNO,
LOW_TERM_GPA_IND,
PARTTIME_TERM_IND,
NOT_REG_TERM_IND,
WITHDRWL_TERM_IND,
FULLTIME_TERM_IND,
OVR_12HR_TERM_IND
 FROM UF_R3_ANALYSIS_GRASUSDATE2007
)

create table UF_R3_ANALYSIS_MAXSUCCESS2007 AS(
select *
from uf_r3_analysis_date2007 A
WHERE A.STUDENT_SUCCESSNO = (SELECT MAX(B.STUDENT_SUCCESSNO)
FROM uf_r3_analysis_date2007 B
WHERE A.PERSON_SID = B.PERSON_SID
)
)

CREATE TABLE UF_R3_ANALYSIS_MAJORCOUNT2007 AS(
select
A.PERSON_SID,
count(A.ACAD_PROG_PRIMARY) AS MAJORCOUNT
from UFDL_PWCBA_DATA.uf_r3_analysis_date2007 A
GROUP BY person_sid
)

CREATE TABLE UF_R3_ANALYSIS_LASTMAJOR0714 AS(
select
A.PERSON_SID,
A.TERM_SID,
A.AGE_YEARS,
A.AGE_MONTHS,
A.AGE_DAYS,
A.JUNIOR_SENIOR_FLAGNO,
A.TOT_TRNSFR,
A.TOT_TEST_CREDIT,
```

```
A.UNT_TAKEN_PRGRSS,
A.CUM_GPA,
A.ENRL_CNT,
A.ACAD_PROG_PRIMARY,
B.MAJORCOUNT,
A.TERM_BEG_DT_SID,
A.UF_CLASSNO,
A.RESIDENCYNO,
A.TOT_TAKEN_GPA,
A.ACAD_PROG_PRIMARYNO,
A.UF_CLASS_EOTNO,
A.UNT_TAKEN_GPA,
A.TERM_END_DT_SID_CATGRYNO,
A.TERM_BEG_DT_CATGRYNO,
A.STUDENT_SUCCESSNO,
A.LOW_TERM_GPA_IND,
A.PARTTIME_TERM_IND,
A.NOT_REG_TERM_IND,
A.WITHDRWL_TERM_IND,
A.FULLTIME_TERM_IND,
A.OVR_12HR_TERM_IND
FROM UFDL_PWCBA_DATA.uf_r3_analysis_majorcount2007 B INNER JOIN
UFDL_PWCBA_DATA.uf_r3_analysis_maxsuccess2007 A
ON A.PERSON_SID = B.PERSON_SID
WHERE A.TERM_BEG_DT_SID<20150000
)
```

Appendix 3: Meanings of Predictor Variables

TERM_SID ----------Term surrogate identification
AGE_YEARS----------Age in Years of the person for the term
AGE_MONTHS----------Age in Months of the person for the term
AGE_DAYS----------Age in Days of the person for the term
TERM_BEG_DT_SID----------The start date of this term
TERM_END_DT_SID----------The end date of this term
JUNIOR_SENIOR_FLAGNO---------- Whether this student is Junior or Senior
TOT_TRNSFR----------The total number of transfer credits
TOT_TEST_CREDIT----------The total number of credits a student tested out of
UNT_TAKEN_PRGRSs----------The total number of credits a student haven't taken yet
CUM_GPA----------GPA of a student
ENRL_CNT----------# of classes a student taken for the last semester
MAJORCOUNT----------# of undergraduate degree a student has or working for
UF_CLASSNO----------The types of class
RESIDENCYNO----------Where a student from(e.g., local, international)
TOT_TAKEN_GPA---------- Sum of GPA for each class a student token.
ACAD_PROG_PRIMARYNO---------- Major of a student
UF_CLASS_EOTNO----------The types of class
UNT_TAKEN_GPA----------Sum of GPA for each class a student didn't token
TERM_END_DT_SID_CATGRYNO--------Categorize "ends dates" as Early, Average, or Late compared to other years. By using K Means Clustering on "ends dates"
TERM_BEG_DT_CATGRYNO----------Categorize "begin dates" as Early, Average, or Late compared to other years. By using K Means Clustering on "begin dates"
LOW_TERM_GPA_IND----------# of semesters the student has enrolled and didn't withdrew, but the gpa is less than 2.2.
PARTTIME_TERM_IND----------# of semesters a student is is half-time, less than half time, or 3/4 time
NOT_REG_TERM_IND----------# of semesters the student didn't register.

WITHDRWL_TERM_IND----------# of semesters the student withdrew from classes
FULLTIME_TERM_IND----------# of semesters the student was enrolled full-time
OVR_12HR_TERM_IND----------# of semesters the student was enrolled for more then 12 credits hours.

Appendix 4: Model Linear&Tree& NearestNeighbors

Github Link

```
df.isnull().sum()
df.dtypes
data_types = df.dtypes
cat_cols = list(data_types[data_types=='object'].index)
con_cols = list(data_types[data_types=='int64'].index) +
list(data_types[data_types=='float64'].index)
con_cols.remove('STUDENT_SUCCESSNO')

df=df.drop(['TOT_TEST_CREDIT'], axis=1)

corrmat = df[con_cols].corr()
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(15,15))
# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=.8, annot=True, square=True)
plt.show()

df.loc[df['STUDENT_SUCCESSNO'] !=1]=0

df['STUDENT_SUCCESSNO'].value_counts()

from sklearn.model_selection import train_test_split
Y = df['STUDENT_SUCCESSNO']
X = df.drop(['STUDENT_SUCCESSNO'], axis= 1)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=1234)

fig = plt.figure(figsize=(20,10))
plt.bar(df['STUDENT_SUCCESSNO'].value_counts().index,
df['STUDENT_SUCCESSNO'].value_counts().values)
plt.xticks(df['STUDENT_SUCCESSNO'].value_counts().index,fontsize=15)
plt.show()

from sklearn import linear_model
from sklearn.metrics import accuracy_score, recall_score, f1_score
lm_lr = linear_model.LogisticRegression()
lm_lr.fit(x_train,y_train)
```

```
y_pred=lm_lr.predict(x_test)
print('Logistic Regression')
print('acc:', accuracy_score(y_test, y_pred))
print('rec:', recall_score(y_test, y_pred))
print('F1:', f1_score(y_test, y_pred))

from sklearn import tree
t_dtc=tree.DecisionTreeClassifier(random_state=1,max_depth=2)
t_dtc.fit(x_train,y_train)
Ynew=t_dtc.predict(x_test)
metrics.classification .confusion_matrix(y_test,Ynew)

from sklearn.neighbors import KNeighborsClassifier as kNN
model = kNN(n_neighbors = 3, algorithm='auto', weights='distance')
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
print('KNN')
print('acc:', accuracy_score(y_test, y_pred))
print('rec:', recall_score(y_test, y_pred))
print('F1:', f1_score(y_test, y_pred))
```

Appendix 5: K Means Code

Github Link

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

data = pd.read_csv("F:/projectFlie/TABLE/UF_R2_UGRDKMEANREDO2007.csv")
datafall = data[data.TERM_SEASON == 'FALL']
dataspring = data[data.TERM_SEASON =='SPRING']

#fallbegmd
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3)
refall=datafall['BEGMD'].values.reshape(-1,1)
kmeans.fit(refall)

clusters = kmeans.cluster_centers_
print(clusters)
y_km = kmeans.fit_predict(refall)

y=[]
for i in range(len(refall[y_km==0])):
    y.append(0)
f=[]
for i in range(len(refall[y_km==1])):
    f.append(0)
q=[]
for i in range(len(refall[y_km==2])):
    q.append(0)

plt.scatter(refall[y_km ==0], y,color ='red')
plt.scatter(refall[y_km ==1], f,color ='green')
plt.scatter(refall[y_km ==2], f,color ='yellow')
plt.show()

#springbegmd
kmeans2 = KMeans(n_clusters = 3)
respring=dataspring['BEGMD'].values.reshape(-1,1)
kmeans2.fit(respring)

clusters2 = kmeans2.cluster_centers_
print(clusters2)
y_km2 = kmeans2.fit_predict(respring)
```

```
z=[]
for i in range(len(respring[y_km2==0])):
    z.append(0)
c=[]
for i in range(len(respring[y_km2==1])):
    c.append(0)
v=[]
for i in range(len(respring[y_km2==2])):
    v.append(0)

plt.scatter(respring[y_km2 ==0], z,color ='red')
plt.scatter(respring[y_km2 ==1], c,color ='green')
plt.scatter(respring[y_km2 ==2], v,color ='yellow')
plt.show()

#springend
kmeans3 = KMeans(n_clusters = 3)
respring2=dataspring['ENDMD'].values.reshape(-1,1)
kmeans3.fit(respring2)

clusters3 = kmeans3.cluster_centers_
print(clusters3)
y_km3 = kmeans3.fit_predict(respring2)

W=[]
for i in range(len(respring2[y_km3==0])):
    W.append(0)
I=[]
for i in range(len(respring2[y_km3==1])):
    I.append(0)
P=[]
for i in range(len(respring2[y_km3==2])):
    P.append(0)

plt.scatter(respring2[y_km3 ==0], W,color ='red')
plt.scatter(respring2[y_km3 ==1], I,color ='green')
plt.scatter(respring2[y_km3 ==2], P,color ='yellow')
plt.show()

#fallendmd
from sklearn.cluster import KMeans
kmeans4 = KMeans(n_clusters = 3)
refall2=datafall['ENDMD'].values.reshape(-1,1)
```

```
kmeans4.fit(refall2)

clusters4 = kmeans4.cluster_centers_
print(clusters4)
y_km4 = kmeans4.fit_predict(refall2)

J=[]
for i in range(len(refall2[y_km4==0])):
    J.append(0)
K=[]
for i in range(len(refall2[y_km4==1])):
    K.append(0)
L=[]
for i in range(len(refall2[y_km4==2])):
    L.append(0)

plt.scatter(refall2[y_km4 ==0], J,color ='red')
plt.scatter(refall2[y_km4 ==1], K,color ='green')
plt.scatter(refall2[y_km4 ==2], L,color ='yellow')
plt.show()

#springlength
kmeans5 = KMeans(n_clusters = 3)
respring3=dataspring['TERM_LENGTH_DAYS'].values.reshape(-1,1)
kmeans5.fit(respring3)

clusters5 = kmeans5.cluster_centers_
print(clusters5)
y_km5 = kmeans5.fit_predict(respring3)

g=[]
for i in range(len(respring3[y_km5==0])):
    g.append(0)
h=[]
for i in range(len(respring3[y_km5==1])):
    h.append(0)
b=[]
for i in range(len(respring3[y_km5==2])):
    b.append(0)

plt.scatter(respring3[y_km5 ==0], g,color ='red')
plt.scatter(respring3[y_km5 ==1], h,color ='green')
plt.scatter(respring3[y_km5 ==2], b,color ='yellow')
plt.show()
```

```
#falllength
from sklearn.cluster import KMeans
kmeans6= KMeans(n_clusters = 3)
refall3=datafall['TERM_LENGTH_DAYS'].values.reshape(-1,1)
kmeans6.fit(refall3)

clusters6 = kmeans6.cluster_centers_
print(clusters6)
y_km6 = kmeans6.fit_predict(refall3)

n=[]
for i in range(len(refall3[y_km6==0])):
    n.append(0)
u=[]
for i in range(len(refall3[y_km6==1])):
    u.append(0)
m=[]
for i in range(len(refall3[y_km6==2])):
    m.append(0)

plt.scatter(refall3[y_km6==0], n,color ='red')
plt.scatter(refall3[y_km6 ==1],u,color ='green')
plt.scatter(refall3[y_km6 ==2],m,color ='yellow')
plt.show()
```

Appendix 6: Principal Component Analysis Reduction Technique

Github Link

```
#Author @Bridge1998

PCA
from pandas.plotting import scatter_matrix
import sklearn as sk

#Data Visualization
print(X.shape)
print(Y.shape)
print(np.unique(Y))
plt.scatter(X[:, 0], X[:, 1], c=Y)
plt.show()

_ = scatter_matrix(pd.DataFrame(X), alpha=0.5, figsize=(12, 10), c=Y)

pca = sk.decomposition.PCA(n_components=4)#components = dimensions
# PCA dimensionality reduction
Xnew = pca.fit_transform(X)
# Print new feature matrix Xnew shape
print("Xnew shape: ", Xnew.shape)
# Print pca explained_variance_ratio_
print("explained_variance_ratio: ", pca.explained_variance_ratio_)
plt.plot(pca.explained_variance_ratio_, '-o')

pca = sk.decomposition.PCA(n_components=2)
# PCA dimensionality reduction
Xnew = pca.fit_transform(X)
# Print new feature matrix Xnew shape
print("Xnew shape: ", Xnew.shape)
# Print pca explained_variance_ratio_
print("explained_variance_ratio: ", pca.explained_variance_ratio_)
print( pca.explained_variance_ratio_.sum())#How much information we save.

from sklearn.model_selection import train_test_split
Y = df['STUDENT_SUCCESSNO']
X = df.drop(['STUDENT_SUCCESSNO',
'TERM_END_DT_SID','TERM_SID','UNT_TAKEN_GPA'], axis= 1)
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=1234)

from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
RF = RandomForestClassifier()
RF = RF.fit(x_train, y_train)
print(sk.metrics.confusion_matrix(y_true=Y, y_pred=RF.predict(X)))
print(sk.metrics.f1_score(Y, RF.predict(X), average='macro'))
```

Appendix 7: UF R3 ANALYSIS Random Forest Model

Github Link

```
#Random Forest Model
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
RF = RandomForestClassifier()
RF = RF.fit(x_train, y_train)
y_pred = RF.predict(x_test)
print (metrics.classification_report(y_test, y_pred))
Y_score = RF.predict_proba(x_test)[:,1]

from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)

#Lime
dfcolumn = df.columns.values.tolist()
dfcolumn.remove('STUDENT_SUCCESSNO')
dfcolumn.remove('TERM_END_DT_SID')
dfcolumn.remove('TERM_SID')
dfcolumn.remove('UNT_TAKEN_GPA')

import lime
import lime.lime_tabular
xtrainnp = np.array(x_train)
ytrainnp = np.array(y_train)

explainer = lime.lime_tabular.LimeTabularExplainer(xtrainnp,
                feature_names=dfcolumn,
                class_names=['1','2','3'],
                verbose=True, mode='regression')
choosen_instance = x_test.iloc[[200]].values[0]
exp = explainer.explain_instance(choosen_instance, RF.predict, num_features=24)
exp.show_in_notebook(show_all=False)
```

Appendix 7: KNN Model and LIME

Github Link

```
Import packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets,model_selection,linear_model,metrics,decomposition


#Read document
df = pd.read_csv("UF_R3_ANALYSIS_LASTMAJOR0714BE.csv")

#Data Preprocessing
df.isnull().sum()
df.dropna(axis=0, how='any',inplace=True)
df['STUDENT_SUCCESSNO'].value_counts()

#We only need 'STUDENT_SUCCESSNO' equals to 1 or 2 to build the model:
df = df.loc[df['STUDENT_SUCCESSNO'].isin(['1','2'])]

#Draw the heatmap using seaborn
import seaborn as sns
cat = df[["JUNIOR_SENIOR_FLAGNO",
"ENRL_CNT",
"MAJORCOUNT",
"UF_CLASSNO",
"RESIDENCYNO",
"TERM_END_DT_SID_CATGRYNO",
"TERM_BEG_DT_CATGRYNO",
"LOW_TERM_GPA_IND",
"PARTTIME_TERM_IND",
"NOT_REG_TERM_IND",
"WITHDRWL_TERM_IND",
"FULLTIME_TERM_IND",
"OVR_12HR_TERM_IND"]]
corcat = cat.corr(method='kendall')
f, ax = plt.subplots(figsize=(15,15))
```

```
sns.heatmap(corcat, vmax=.8, annot=True, square=True)
plt.show()

con = df[[
"TERM_SID",
"TERM_BEG_DT_SID",
"TERM_END_DT_SID",
"AGE_YEARS",
"AGE_MONTHS",
"AGE_DAYS",
"TOT_TRNSFR",
"TOT_TEST_CREDIT",
"UNT_TAKEN_PRGRSS",
"CUM_GPA",
"TOT_TAKEN_GPA",
“UNT_TAKEN_GPA”
]]corcon = con.corr(method='pearson')
f, ax = plt.subplots(figsize=(15,15))
sns.heatmap(corcon, vmax=.8, annot=True, square=True)
plt.show()

#According to the plot above, we can delete the variable “TERM_END_DT_SID”, “TERM_SID” and “UNT_TAKEN_GPA”.

con = df[[
TERM_BEG_DT_SID
"AGE_YEARS",
"AGE_MONTHS",
"AGE_DAYS",
"TOT_TRNSFR",
"TOT_TEST_CREDIT",
"UNT_TAKEN_PRGRSS",
"CUM_GPA",
"TOT_TAKEN_GPA"
]]

#KNN Model

#One-hot Encoding
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets,model_selection,linear_model,metrics,decomposition
df=pd.read_csv("D:/UFL/ISOM/summer2020/QMB6930/codeForProject/UF_R3_ANALYSIS_LASTMAJOR0714BE.csv")
df.dropna(axis=0, how='any',inplace=True)
df = df.loc[df['STUDENT_SUCCESSNO'].isin(['1','2'])]
df_JUNIOR_SENIOR_FLAGNO=df['JUNIOR_SENIOR_FLAGNO'].apply(str).str.get_dummies().add_prefix('JUNIOR_SENIOR_FLAGNO: ')
df_ENRL_CNT = df['ENRL_CNT'].apply(str).str.get_dummies().add_prefix('ENRL_CNT: ')
df_MAJORCOUNT=df['MAJORCOUNT'].apply(str).str.get_dummies().add_prefix('MAJORCOUNT: ')
df_UF_CLASSNO=df['UF_CLASSNO'].apply(str).str.get_dummies().add_prefix('UF_CLASSNO: ')
df_RESIDENCYNO=df['RESIDENCYNO'].apply(str).str.get_dummies().add_prefix('RESIDENCYNO: ')
df_TERM_END_DT_SID_CATGRYNO=df['TERM_END_DT_SID_CATGRYNO'].apply(str).str.get_dummies().add_prefix('TERM_END_DT_SID_CATGRYNO: ')
df_TERM_BEG_DT_CATGRYNO=df['TERM_BEG_DT_CATGRYNO'].apply(str).str.get_dummies().add_prefix('TERM_BEG_DT_CATGRYNO: ')
df_LOW_TERM_GPA_IND=df['LOW_TERM_GPA_IND'].apply(str).str.get_dummies().add_prefix('LOW_TERM_GPA_IND: ')
df_PARTTIME_TERM_IND=df['PARTTIME_TERM_IND'].apply(str).str.get_dummies().add_prefix('PARTTIME_TERM_IND: ')
df_NOT_REG_TERM_IND=df['NOT_REG_TERM_IND'].apply(str).str.get_dummies().add_prefix('NOT_REG_TERM_IND: ')
df_WITHDRWL_TERM_IND=df['WITHDRWL_TERM_IND'].apply(str).str.get_dummies().add_prefix('WITHDRWL_TERM_IND: ')
df_FULLTIME_TERM_IND=df['FULLTIME_TERM_IND'].apply(str).str.get_dummies().add_prefix('FULLTIME_TERM_IND: ')
df_OVR_12HR_TERM_IND=df['OVR_12HR_TERM_IND'].apply(str).str.get_dummies().add_prefix('OVR_12HR_TERM_IND: ')
df=pd.concat([df,df_JUNIOR_SENIOR_FLAGNO,df_ENRL_CNT,df_MAJORCOUNT,df_UF_CLASSNO,df_RESIDENCYNO,df_TERM_END_DT_SID_CATGRYNO,df_TERM_BEG_DT_CATGRYNO,df_LOW_TERM_GPA_IND,df_PARTTIME_TERM_IND,df_NOT_REG_TERM_IND,df_WITHDRWL_TERM_IND,df_FULLTIME_TERM_IND,df_OVR_12HR_TERM_IND],axis=1)
df = df.drop([
```

```
    'JUNIOR_SENIOR_FLAGNO',
 'ENRL_CNT',
 'MAJORCOUNT',
 'UF_CLASSNO',
 'RESIDENCYNO',
 'TERM_END_DT_SID_CATGRYNO',
 'TERM_BEG_DT_CATGRYNO',
 'LOW_TERM_GPA_IND',
 'PARTTIME_TERM_IND',
 'NOT_REG_TERM_IND',
 'WITHDRWL_TERM_IND',
 'FULLTIME_TERM_IND',
 'OVR_12HR_TERM_IND'
],axis=1)

#Model Building
from sklearn.model_selection import train_test_split
Y = df['STUDENT_SUCCESSNO']
X=df.drop(['STUDENT_SUCCESSNO',
'TERM_END_DT_SID','TERM_SID','UNT_TAKEN_GPA'], axis= 1)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=3234)

from sklearn.neighbors import KNeighborsClassifier as kNN
model = kNN(n_neighbors = 3, algorithm='auto', weights='distance')
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
from sklearn.metrics import classification_report, confusion_matrix
print (metrics.classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

#Lime
xtrainnp = np.array(x_train)
ytrainnp = np.array(y_train)
explainer = lime.lime_tabular.LimeTabularExplainer(xtrainnp,
                    feature_names=dfcolumn,
                    class_names=['1','2'],
                    verbose=True, mode='regression')
choosen_instance = x_test.iloc[[700]].values[0]
exp = explainer.explain_instance(choosen_instance, model.predict,num_features=7)
```

```
exp.show_in_notebook(show_all=False)
exp.as_pyplot_figure()

df2=pd.read_csv("D:/UFL/ISOM/summer2020/QMB6930/codeForProject/UF_R3_ANAL
YSIS_LASTMAJOR0714BE.csv")
df2.dropna(axis=0, how='any',inplace=True)
df2 = df2.loc[df2['STUDENT_SUCCESSNO'].isin(['1','2'])]
print(x_test.iloc[[700]])

pd.set_option('display.max_columns', None)
print(df2.iloc[[46109]])

choosen_instance = x_test.iloc[[25]].values[0]
exp = explainer.explain_instance(choosen_instance, model.predict,num_features=22)
exp.show_in_notebook(show_all=False)
exp.as_pyplot_figure()

print(x_test.iloc[[25]])

print(df2.iloc[[34670]])

#Visualization

fig = plt.figure(figsize=(20,10))
plt.bar(df['STUDENT_SUCCESSNO'].value_counts().index,
df['STUDENT_SUCCESSNO'].value_counts().values)
plt.xticks(df['STUDENT_SUCCESSNO'].value_counts().index,fontsize=15)
plt.show()

plt.scatter(X.values[:, 0], X.values[:, 1], c=Y)
plt.show()
```

Appendix 8: Testing And Cross Validation

Import packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets,model_selection,linear_model,metrics,decomposition

df = pd.read_csv("UF_R3_ANALYSIS_LASTMAJOR0714.csv")

df.isnull().sum()
df.dropna(axis=0, how='any',inplace=True)
df['STUDENT_SUCCESSNO'].value_counts()
```

One Hot Encoding

```
df_JUNIOR_SENIOR_FLAGNO=df['JUNIOR_SENIOR_FLAGNO'].apply(str).str.get_dummies().
add_prefix('JUNIOR_SENIOR_FLAGNO: ')
df_ENRL_CNT = df['ENRL_CNT'].apply(str).str.get_dummies().add_prefix('ENRL_CNT: ')
df_MAJORCOUNT=df['MAJORCOUNT'].apply(str).str.get_dummies().add_prefix('MAJORCOU
NT: ')
df_UF_CLASSNO=df['UF_CLASSNO'].apply(str).str.get_dummies().add_prefix('UF_CLASSNO:
')
df_RESIDENCYNO=df['RESIDENCYNO'].apply(str).str.get_dummies().add_prefix('RESIDENCY
NO: ')
df_TERM_END_DT_SID_CATGRYNO=df['TERM_END_DT_SID_CATGRYNO'].apply(str).str.g
et_dummies().add_prefix('TERM_END_DT_SID_CATGRYNO: ')
df_TERM_BEG_DT_CATGRYNO=df['TERM_BEG_DT_CATGRYNO'].apply(str).str.get_dummi
es().add_prefix('TERM_BEG_DT_CATGRYNO: ')
df_LOW_TERM_GPA_IND=df['LOW_TERM_GPA_IND'].apply(str).str.get_dummies().add_prefi
x('LOW_TERM_GPA_IND: ')
df_PARTTIME_TERM_IND=df['PARTTIME_TERM_IND'].apply(str).str.get_dummies().add_prefi
x('PARTTIME_TERM_IND: ')
df_NOT_REG_TERM_IND=df['NOT_REG_TERM_IND'].apply(str).str.get_dummies().add_prefix
('NOT_REG_TERM_IND: ')
df_WITHDRWL_TERM_IND=df['WITHDRWL_TERM_IND'].apply(str).str.get_dummies().add_pr
efix('WITHDRWL_TERM_IND: ')
df_FULLTIME_TERM_IND=df['FULLTIME_TERM_IND'].apply(str).str.get_dummies().add_prefix
('FULLTIME_TERM_IND: ')
```

```
df_OVR_12HR_TERM_IND=df['OVR_12HR_TERM_IND'].apply(str).str.get_dummies().add_prefix('OVR_12HR_TERM_IND: ')
df=pd.concat([df,df_JUNIOR_SENIOR_FLAGNO,df_ENRL_CNT,df_MAJORCOUNT,df_UF_CLASSNO,df_RESIDENCYNO,df_TERM_END_DT_SID_CATGRYNO,df_TERM_BEG_DT_CATGRYNO,df_LOW_TERM_GPA_IND,df_PARTTIME_TERM_IND,df_NOT_REG_TERM_IND,df_WITHDRWL_TERM_IND,df_FULLTIME_TERM_IND,df_OVR_12HR_TERM_IND],axis=1)
df = df.drop([
    'JUNIOR_SENIOR_FLAGNO',
 'ENRL_CNT',
 'MAJORCOUNT',
 'UF_CLASSNO',
 'RESIDENCYNO',
 'TERM_END_DT_SID_CATGRYNO',
 'TERM_BEG_DT_CATGRYNO',
 'LOW_TERM_GPA_IND',
 'PARTTIME_TERM_IND',
 'NOT_REG_TERM_IND',
 'WITHDRWL_TERM_IND',
 'FULLTIME_TERM_IND',
 'OVR_12HR_TERM_IND'
],axis=1)
```

Create test dataset and train dataset

```
df2 = pd .read_csv("UF_R3_ANALYSIS_LASTMAJOR0714.csv")
df2.to_csv("train.csv", index=False)
df3 = df1[(df1['TERM_BEG_DT_SID'] >=20150101) & (df1['TERM_BEG_DT_SID'] <= 20191231)]
df3.to_csv("test.csv", index=False)


train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")


mid = train['STUDENT_SUCCESSNO']
train.drop(labels=['STUDENT_SUCCESSNO'], axis=1,inplace = True)
train.insert(0, 'STUDENT_SUCCESSNO', mid)

train_x = train.values[0:, 1:]
train_y = train.values[0:, 0]
result_x = test.values[0:, 0:]
```

Split the train dataset

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(train_x, train_y,test_size=0.1,random_state=33)
```

Fit the model

```
from numpy import ravel
from sklearn.neighbors import KNeighborsClassifier as kNN
model = kNN(n_neighbors = 3, algorithm='auto', weights='distance')
model.fit(x_train,ravel(y_train))
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='distance')
```

Model result

```
y_predict = model.predict(x_test)
print("score on the testdata:",model.score(x_test,y_test))
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           1       0.98      0.87      0.92      3459
           2       0.95      0.99      0.97      9500

    accuracy                           0.96     12959
   macro avg       0.96      0.93      0.95     12959
weighted avg       0.96      0.96      0.96     12959
```

List probability of predicting successfully, predicting results, and actual results. Create a csv file

```
probablity = model.predict_proba(x_test)
list_pro = []
for i in range(probablity.shape[0]):
    pro = max(list(probablity[i]))
    list_pro.append(pro)
index = np.array(id).reshape((-1,1))[:,0:1]
result = pd.DataFrame(np.column_stack((np.array(y_test).reshape(-
1,1),np.array(y_predict).reshape(-1,1),np.array(list_pro).reshape(-1,1))),
columns=['test_label','predict_lable','probablity'])
result.to_csv('knn_result.csv',index=False,header=True,encoding='gbk')
```

knn_result

test_label	predict_lable	probablity
2.0	2.0	0.6429143924679040
2.0	2.0	0.6439142920941770
2.0	2.0	0.607852756717332
2.0	2.0	0.6724350628467050
2.0	2.0	1.0
2.0	2.0	1.0
2.0	2.0	1.0
2.0	2.0	0.6971004747433420
2.0	2.0	1.0
2.0	2.0	1.0
2.0	2.0	1.0
2.0	2.0	1.0
2.0	3.0	0.683257252298769
3.0	2.0	0.6754797190775400
1.0	1.0	1.0

Find all the incorrect results and create a csv file

```
    diff_index = []
    for i in range(result.shape[0]):
        diff_index.append(result['test_label'][i] != result['predict_lable'][i])
    print(diff_index)
    diff = result[diff_index]
    diff_x = x_test[diff_index]
diff.to_csv('knn_result_diff.csv',index=False,header=True,encoding='gbk')
```

knn_result_diff

test_label	predict_lable	probablity
2.0	3.0	0.683257252298769
3.0	2.0	0.6754797190775400
3.0	2.0	0.5669753206982810
2.0	3.0	0.348665246621895
3.0	2.0	1.0
3.0	2.0	0.5786298503719970
3.0	2.0	0.7055139253496500
1.0	2.0	0.7357622900168210
3.0	2.0	1.0
2.0	1.0	0.7061090093432220
3.0	2.0	0.6788536924530730
3.0	1.0	0.6954391824647830
3.0	2.0	0.6467164326851390
1.0	2.0	1.0
1.0	2.0	1.0
1.0	2.0	1.0
1.0	2.0	1.0

Predict results of testing dataset and create a csv file

```
mid = test['STUDENT_SUCCESSNO']
test.drop(labels=['STUDENT_SUCCESSNO'], axis=1,inplace = True)
test.insert(0, 'STUDENT_SUCCESSNO', mid)

x_test = test.values[0:, 1:]
y_test = test.values[0:, 0]
```

```
y_predict = model.predict(x_test)
index = np.array(id).reshape((-1,1))[:,0:1]
result = pd.DataFrame(np.column_stack(np.array(y_predict).reshape(-1,1),)),
columns=['predict_lable'])
result.to_csv('predict_result.csv',index=False,header=True,encoding='gbk')
```

predict_resul

predict_lable
3.0
1.0
2.0
2.0
2.0
2.0
2.0
1.0
2.0
1.0
2.0
1.0
2.0
2.0
2.0
2.0
2.0
1.0

Appendix 9: K-Fold Cross Validation

Github Link

#Author @bridge1998

Cross Validation for Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import cross_val_score

Y = df['STUDENT_SUCCESSNO']
X = df.drop(['STUDENT_SUCCESSNO'], axis= 1)
RF = RandomForestClassifier()
scores = cross_val_score(RF, X, Y, cv=5)
scores
```

Cross Validation for KNN

```
from sklearn.metrics import accuracy_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier as kNN

model = kNN(n_neighbors = 3, algorithm='auto', weights='distance')
Y = df['STUDENT_SUCCESSNO']
X = df.drop(['STUDENT_SUCCESSNO'], axis= 1)
scores = cross_val_score(model, X, Y, cv=5)
scores
```