Predictive Analysis of Graduation Rates

Data Preparation

Table 1. Data Fields

Field Attribute	Field Name	Data Type
Demographic Data	Marital Status	Numeric
Demographic Data	Nationality	Numeric
Demographic Data	Displaced	Numeric
Demographic Data	Gender	Numeric
Demographic Data	Age at Enrollment	Numeric
Demographic Data	International	Numeric
Socieconomic Data	Mother's Qualification	Numeric
Socieconomic Data	Father's Qualification	Numeric
Socieconomic Data	Mother's Occupation	Numeric
Socieconomic Data	Father's Occupation	Numeric
Socieconomic Data	Educational Special Needs	Numeric
Socieconomic Data	Debtor	Numeric
Socieconomic Data	Tuition Fees Up to Date	Numeric
Socieconomic Data	Scholarship Holder	Numeric
Macroeconomic Data	Unemployment Rate	Numeric
Macroeconomic Data	Inflation Rate	Numeric
Macroeconomic Data	GDP	Numeric
Enrollment Data	Application Mode	Numeric
Enrollment Data	Application Order	Numeric
Enrollment Data	Course	Numeric
Enrollment Data	Daytime/Evening Attendance	Numeric
Enrollment Data	Previous Qualification	Numeric
Enrollment Data (End of 1st Semester)	Curricular Units 1st Sem (Credited)	Numeric
Enrollment Data (End of 1st Semester)	Curricular Units 1st Sem (Enrolled)	Numeric
Enrollment Data (End of 1st Semester)	Curricular Units 1st Sem (Evaluations)	Numeric
Enrollment Data (End of 1st Semester)	Curricular Units 1st Sem (Approved)	Numeric

Enrollment Data (End of 1st Semester)	Curricular Units 1st Sem (Grade)	Numeric
Enrollment Data (End of 1st Semester)	Curricular Units 1st Sem (Without Evaluations)	Numeric
Enrollment Data (End of 2nd Semester)	Curricular Units 1st Sem (Credited)	Numeric
Enrollment Data (End of 2nd Semester)	Curricular Units 1st Sem (Enrolled)	Numeric
Enrollment Data (End of 2nd Semester)	Curricular Units 1st Sem (Evaluations)	Numeric
Enrollment Data (End of 2nd Semester)	Curricular Units 1st Sem (Approved)	Numeric
Enrollment Data (End of 2nd Semester)	Curricular Units 1st Sem (Grade)	Numeric
Enrollment Data (End of 2nd Semester)	Curricular Units 1st Sem (Without Evaluations)	Numeric

Data preparation was primarily handled in previous milestones. Full steps include:

1. Validated dataset for any NaN/Null values (no adjustments needed)
2. Desciprtive analysis was performed in order to better understand the basic statistics of the attributes (distribution, mean, median, min, and max)
3. One of the issues identified early on was the possible imbalance of data within the "Target" Field, which described whether or not the student was a "Graduate", "Dropout", or "Enrolled." The dataset had a higher density of graduates (2,209) with dropouts next (1,421), and lastly enrolled (794). Graduates represented nearly 50% of the population, which could cause the model to skew towards a positive result. No adjustments were made in the preparation steps but kept in mind for model building.
4. Early visualizations were created to look at field correlation and multi-collinearity. A heatmap using Pearson correlation coefficient was created to look at the different field attributes. Collinearity was determined to be the strongest within the same field attribute, but it could also be seen between groups. This also helped inform feature selection.
5. To combat the earlier mentioned imbalance, the "Target" column was converted to a binary column where "Graduate" and "Enrolled" represented a value of 1 and "Dropout" represented a value of 0. This helped balance the representation imbalance and simplified the nuance of academic success between "Graduate" and "Enrolled."

In [29]:

```
## import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

## library versions
print('pandas version:', pd.__version__)
print('numpy version:', np.__version__)
print('seaborn version:', sns.__version__)
```

```
pandas version: 1.4.2
numpy version: 1.21.5
seaborn version: 0.11.2
```

In [3]:

```
## ignore warnings
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

In [8]:

```
## print options
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 25)

## load dataset and view preview
df = pd.read_csv("DSC630_Project.csv")
df.head()
```

Out[8]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Previous qualification (grade)	Na
0	1	17	5	171	1	1	122.0	
1	1	15	1	9254	1	1	160.0	
2	1	1	5	9070	1	1	122.0	
3	1	17	2	9773	1	1	122.0	
4	2	39	1	8014	0	1	100.0	

In [10]:

```
## (1) check for null values
null = df.isnull().values.any()
print(null)
```

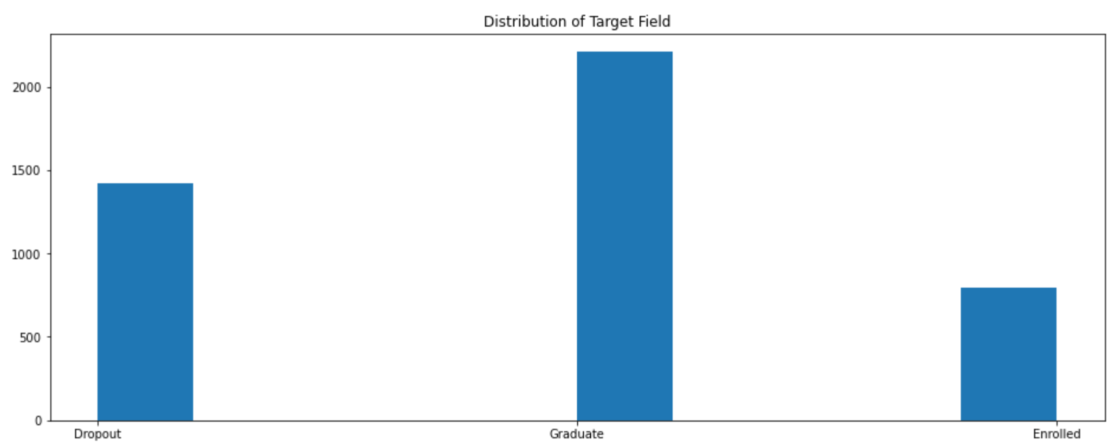
```
False
```

In [11]:
```python
## (2) descriptive analysis
df.describe()
```

Out[11]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previou qualificatio
count	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.00000
mean	1.178571	18.669078	1.727848	8856.642631	0.890823	4.57775
std	0.605747	17.484682	1.313793	2063.566416	0.311897	10.21659
min	1.000000	1.000000	0.000000	33.000000	0.000000	1.00000
25%	1.000000	1.000000	1.000000	9085.000000	1.000000	1.00000
50%	1.000000	17.000000	1.000000	9238.000000	1.000000	1.00000
75%	1.000000	39.000000	2.000000	9556.000000	1.000000	1.00000
max	6.000000	57.000000	9.000000	9991.000000	1.000000	43.00000

In [16]:
```python
## (3) Distribution of "Target" field
fig = plt.figure(figsize =(16, 6))
plt.hist(df['Target'])
plt.title('Distribution of Target Field', loc = 'center', fontsize = 1
plt.show()
```



In [17]:

```
## (4) Heatmap of Pearson Correlation

## correlation matrix
df.corr()
```

Out[17]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previou qualificatic
Marital status	1.000000	0.264006	-0.125854	0.046365	-0.274939	0.06252
Application mode	0.264006	1.000000	-0.286357	0.065385	-0.304092	0.4224
Application order	-0.125854	-0.286357	1.000000	0.059507	0.158657	-0.1843
Course	0.046365	0.065385	0.059507	1.000000	-0.043151	0.0066
Daytime/evening attendance	-0.274939	-0.304092	0.158657	-0.043151	1.000000	-0.0718
...	...	...	...	...	...	
Curricular units 2nd sem (grade)	-0.071506	-0.115424	0.055517	0.348728	0.050493	0.0009
Curricular units 2nd sem (without evaluations)	0.020426	0.047983	-0.015757	0.030816	-0.004229	0.0051
Unemployment rate	-0.020338	0.089080	-0.098419	0.007153	0.061974	0.1119
Inflation rate	0.008761	-0.016375	-0.011133	0.017710	-0.024043	-0.0637
GDP	-0.027003	-0.022743	0.030201	-0.020265	0.022929	0.0640

36 rows × 36 columns

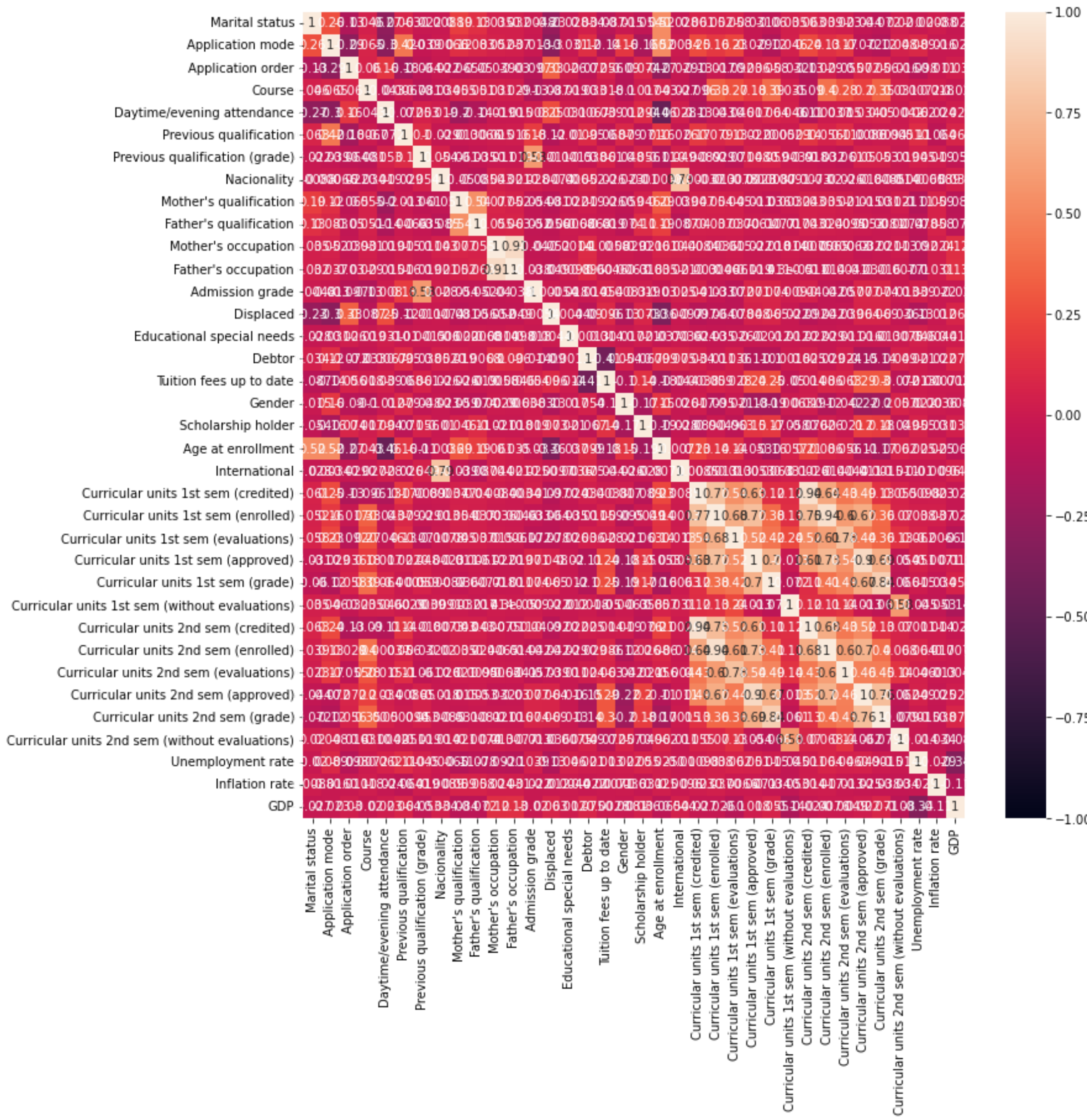
Additional Notes:

While it was earlier indicated that the highest collinearity was between fields within the same field attribute, specifics include fields such as "Nationality" and "Internaional" and "Mother's Occupation" and "Father's Occupation."

In [26]:

```
plt.figure(figsize = (12,12))
sns.heatmap(df.corr(), vmin= -1, vmax = 1, annot = True)
```

Out[26]: <AxesSubplot:>



In [34]:

```
## (5) Adjust column- "Target"
df['Target'] = df['Target'].replace(['Graduate', 'Enrolled', 'Dropout'
df.head()
```

Out[34]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Previous qualification (grade)	Na
0	1	17	5	171	1	1	122.0	
1	1	15	1	9254	1	1	160.0	
2	1	1	5	9070	1	1	122.0	
3	1	17	2	9773	1	1	122.0	
4	2	39	1	8014	0	1	100.0	

In [40]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4424 entries, 0 to 4423
Data columns (total 37 columns):
 #   Column                                          Non-Null Count
Dtype
---  ------                                          --------------
-----
 0   Marital status                                  4424 non-null
int64
 1   Application mode                                4424 non-null
int64
 2   Application order                               4424 non-null
int64
 3   Course                                          4424 non-null
int64
 4   Daytime/evening attendance                      4424 non-null
int64
 5   Previous qualification                          4424 non-null
int64
 6   Previous qualification (grade)                  4424 non-null
float64
 7   Nacionality                                     4424 non-null
int64
 8   Mother's qualification                          4424 non-null
int64
 9   Father's qualification                          4424 non-null
int64
 10  Mother's occupation                             4424 non-null
int64
 11  Father's occupation                             4424 non-null
int64
 12  Admission grade                                 4424 non-null
float64
 13  Displaced                                       4424 non-null
int64
 14  Educational special needs                       4424 non-null
int64
 15  Debtor                                          4424 non-null
int64
 16  Tuition fees up to date                         4424 non-null
int64
 17  Gender                                          4424 non-null
int64
 18  Scholarship holder                              4424 non-null
int64
 19  Age at enrollment                               4424 non-null
int64
 20  International                                   4424 non-null
int64
 21  Curricular units 1st sem (credited)             4424 non-null
int64
 22  Curricular units 1st sem (enrolled)             4424 non-null
int64
 23  Curricular units 1st sem (evaluations)          4424 non-null
int64
 24  Curricular units 1st sem (approved)             4424 non-null
int64
```

```
 25  Curricular units 1st sem (grade)                4424 non-null
float64
 26  Curricular units 1st sem (without evaluations)  4424 non-null
int64
 27  Curricular units 2nd sem (credited)             4424 non-null
int64
 28  Curricular units 2nd sem (enrolled)             4424 non-null
int64
 29  Curricular units 2nd sem (evaluations)          4424 non-null
int64
 30  Curricular units 2nd sem (approved)             4424 non-null
int64
 31  Curricular units 2nd sem (grade)                4424 non-null
float64
 32  Curricular units 2nd sem (without evaluations)  4424 non-null
int64
 33  Unemployment rate                               4424 non-null
float64
 34  Inflation rate                                  4424 non-null
float64
 35  GDP                                             4424 non-null
float64
 36  Target                                          4424 non-null
object
dtypes: float64(7), int64(29), object(1)
memory usage: 1.2+ MB
```

Build and Evaluate Models

In [41]:
```
## split the data

## import train_test_split
from sklearn.model_selection import train_test_split

## define X and y
X = df.drop('Target', axis=1)
y = df['Target']

## split data using 89/20 split
X_train, X_test, y_train, y_test = train_test_split(
  X,y , random_state = 42, test_size = 0.2, shuffle=True)

print('X_train : ')
print(X_train.shape)

print('X_test : ')
print(X_test.shape)

print('y_train : ')
print(y_train.shape)

print('y_test : ')
print(y_test.shape)
```

```
X_train :
(3539, 36)
X_test :
(885, 36)
y_train :
(3539,)
y_test :
(885,)
```

Random Forest Classifier

During initial analysis and discovery, there was a preference for moving forward with a decision tree classification model. However, during the descriptive analysis, it was difficult to conclude what fields contibuted towards a student's academic success or not. Random Forest Classifiers take the averages of decision trees, which cancels out the biases. This reduces overfitting. While it can be more challenging to interpret a random forest classifier, I felt that it would be the better model to move forward with understanding that different approaches can be made later on.

```
In [43]:
# import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(random_state = 0) ## instantiate classifi
rfc.fit(X_train, y_train) ## fit model
y_pred = rfc.predict(X_test) ## predict Test set results

## import accuracy_score
from sklearn.metrics import accuracy_score

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format
```

```
Model accuracy score with 10 decision-trees : 0.8554
```

```
In [46]:
## test and adjust for model accuracy, n_estimators = 100

rfc_100 = RandomForestClassifier(n_estimators = 100, random_state = 0)
rfc_100.fit(X_train, y_train) ## fit model
y_pred_100 = rfc_100.predict(X_test) ## predict Test set results

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. forma
```

```
Model accuracy score with 100 decision-trees : 0.8554
```

Results Interpretation

There is no discernable difference between 10 decision-trees and 100-decision trees. Unexpectedlty, the accuracy neither increased nor decreased with a shift of the number of decision-trees. As all work thus far has included all attributes of the dataset, further work should be done to select only the important attributes. Once feature selection has been done, the model can be re-evaluated for accuracy.

Looking at the feature selection scores below, we can see that fields like "Marital Status", "Daytime/Evening Attendance", "Nationality", "International", and "Educational Special Needs" are the least important features in contrast to academic attributes such as "Curricular units 2nd sem (approved)", "Curricular units 2nd sem (grade)", and "Curricular units 1st sem (approved)."

Out of an abundance of caution, only the lowest 5 attributes will be dropped from the model. If minimal improvements to the model accuracy are found, further research and reductions will be made.

In [47]:

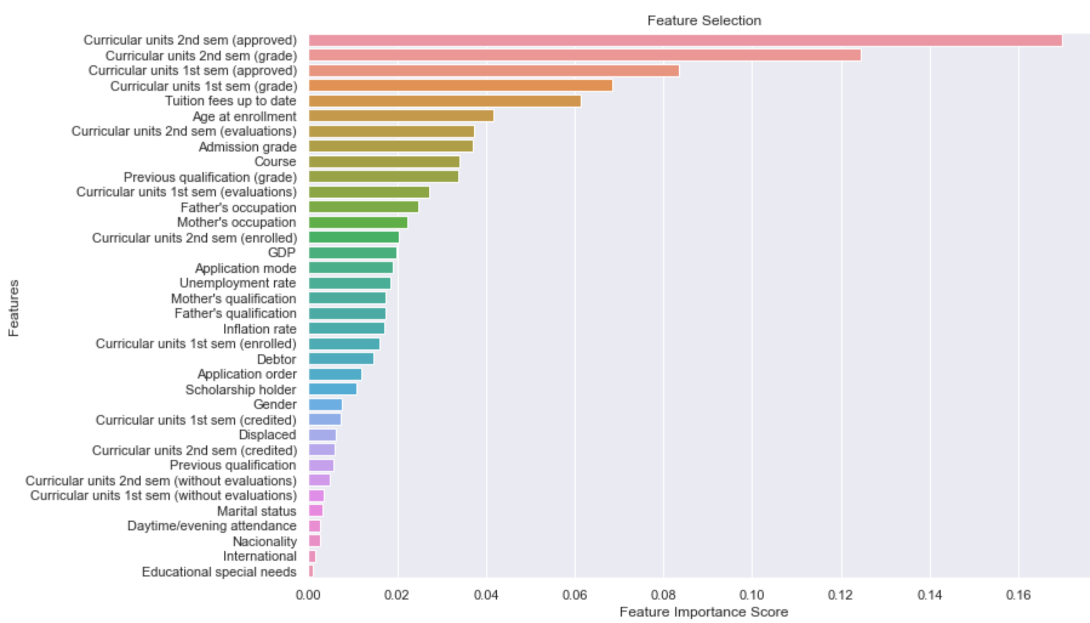
```python
clf = RandomForestClassifier(n_estimators = 100, random_state = 0) ##
clf.fit(X_train, y_train) ## fit the model to the training set

# view the feature scores
feature_scores = pd.Series(clf.feature_importances_, index = X_train.c
feature_scores
```

Out[47]:

```
Curricular units 2nd sem (approved)    0.169821
Curricular units 2nd sem (grade)       0.124344
Curricular units 1st sem (approved)    0.083448
Curricular units 1st sem (grade)       0.068416
Tuition fees up to date                0.061234
                                         ...
Marital status                         0.003309
Daytime/evening attendance             0.002624
Nacionality                            0.002515
International                          0.001437
Educational special needs              0.000895
Length: 36, dtype: float64
```

In [49]:

```python
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.barplot(x=feature_scores, y=feature_scores.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title('Feature Selection')
plt.show()
```



In [50]:

```
### Model Rebuild

## define X and y
X = df.drop(['Target', 'Marital status', 'Daytime/evening attendance',
y = df['Target']

## split data using 89/20 split
X_train, X_test, y_train, y_test = train_test_split(
  X,y , random_state = 42, test_size = 0.2, shuffle=True)

print('X_train : ')
print(X_train.shape)

print('X_test : ')
print(X_test.shape)

print('y_train : ')
print(y_train.shape)

print('y_test : ')
print(y_test.shape)
```

```
X_train :
(3539, 31)
X_test :
(885, 31)
y_train :
(3539,)
y_test :
(885,)
```

In [51]:

```
## model rebuild, n_estimators = 10
rfc = RandomForestClassifier(random_state = 0) ## instantiate classifi
rfc.fit(X_train, y_train) ## fit model
y_pred = rfc.predict(X_test) ## predict Test set results

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format
```

```
Model accuracy score with 10 decision-trees : 0.8621
```

In [52]:

```
## test and adjust for model accuracy, n_estimators = 100

rfc_100 = RandomForestClassifier(n_estimators = 100, random_state = 0)
rfc_100.fit(X_train, y_train) ## fit model
y_pred_100 = rfc_100.predict(X_test) ## predict Test set results

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. forma
```

```
Model accuracy score with 100 decision-trees : 0.8621
```

Results Interpretation (2)

The initial model accuracy score was .8554. Once 5 fields were removed, the model accuracy improved to .8621, which was a +.0067 improvement. While the model is improving in regards to accuracy, it is still challenging to fully understand the full impact of individual fields. To further summarize the performance of the random forest classifier model, a confusion matrix can assist in interpreting the overall performance and any errors that are being produced by the model.

Within the 2x2 confusion matrix below, we can interpret the four different results s such:

1. True Positive (Upper Left): 228, which indicates where the model correctly predicted the positive class
2. False Positive (Upper Right): 88, which indicates where the model incorrectly predicted the positive class when it was actually negative (type 1 error)
3. False Negative (Lower Left): 34, which indicates where the model incorrectly predicted the negative class when it was actually positive (type 2 error)
4. True Negative (Lower Right): 535, which indicates where the model correctly predicted a negative class.

In [53]:

```
## Confusion Matrix

## import confusion matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix\n\n', cm)
```

```
Confusion matrix

 [[228  88]
 [ 34 535]]
```

In [71]:

```
## import metrics libraries
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
```

In [68]:

```
## recall
recall_score(y_test, y_pred, average = None)
```

Out[68]:

```
array([0.72151899, 0.94024605])
```

In [69]:

```
## precision
precision_score(y_test, y_pred, average = None)
```

Out[69]:

```
array([0.87022901, 0.85874799])
```

In [70]:
```python
f1_score(y_test, y_pred, average = None)
```

Out[70]:
```
array([0.78892734, 0.89765101])
```

In order to further finetune the model, "SelectFromModel" will be used in order to automatically select the features that have a greater importance than the mean importance of all of the features.

This features selection shows that 10 fields have an importance less than the mean of all of the features. However, once these fields have been dropped from the dataset, the model's accuracy drops to .7684, which is significantly worse than the previous accuracy score. My best hypothesis is that this is due to the high correlation/multicollinearity found during the descriptive analysis. Final recommendation would be to return the first model rebuild with a model accuracy of .8621.

In [73]:
```python
## import SelectFromModel
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(RandomForestClassifier(n_estimators = 100))
sel.fit(X_train, y_train)
```

Out[73]:
```
SelectFromModel(estimator=RandomForestClassifier())
```

In [76]:
```python
## True = importance is greater than the mean importance
## False = importance is lss than the mean importance
sel.get_support()
```

Out[76]:
```
array([False, False,  True, False,  True, False, False, False, False,
        True, False, False,  True, False, False,  True, False, False,
       False,  True,  True, False, False, False,  True,  True,  True,
       False, False, False, False])
```

In [78]:
```python
## get features column names
selected_feat= X_train.columns[(sel.get_support())]
len(selected_feat)
print(selected_feat)
```

```
Index(['Course', 'Previous qualification (grade)', 'Admission grade',
       'Tuition fees up to date', 'Age at enrollment',
       'Curricular units 1st sem (approved)',
       'Curricular units 1st sem (grade)',
       'Curricular units 2nd sem (evaluations)',
       'Curricular units 2nd sem (approved)',
       'Curricular units 2nd sem (grade)'],
      dtype='object')
```

In [81]:

```
### Model Rebuild 2

## define X and y
X = df.drop(['Target', 'Marital status', 'Daytime/evening attendance',
             'Educational special needs', 'Course', 'Previous qualific
       'Tuition fees up to date', 'Age at enrollment', 'Curricular uni
       'Curricular units 1st sem (grade)','Curricular units 2nd sem (e
       'Curricular units 2nd sem (grade)'], axis=1)
y = df['Target']

## split data using 89/20 split
X_train, X_test, y_train, y_test = train_test_split(
  X,y , random_state = 42, test_size = 0.2, shuffle=True)

print('X_train : ')
print(X_train.shape)

print('X_test : ')
print(X_test.shape)

print('y_train : ')
print(y_train.shape)

print('y_test : ')
print(y_test.shape)
```

```
X_train :
(3539, 21)
X_test :
(885, 21)
y_train :
(3539,)
y_test :
(885,)
```

In [82]:

```
## model rebuild, n_estimators = 10
rfc = RandomForestClassifier(random_state = 0) ## instantiate classifi
rfc.fit(X_train, y_train) ## fit model
y_pred = rfc.predict(X_test) ## predict Test set results

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format
```

```
Model accuracy score with 10 decision-trees : 0.7684
```

In [83]:

```
## test and adjust for model accuracy, n_estimators = 100

rfc_100 = RandomForestClassifier(n_estimators = 100, random_state = 0)
rfc_100.fit(X_train, y_train) ## fit model
y_pred_100 = rfc_100.predict(X_test) ## predict Test set results

print('Model accuracy score with 100 decision-trees : {0:0.4f}'. forma
```

```
Model accuracy score with 100 decision-trees : 0.7684
```

Decision Tree Classifier

As a comparison, the decision tree classifier model was built as a comparison to the original Random Forest Classifier. Accuracy was .8305, which was markedly lower than the Random Forest Classifier's performance at .8621.

In [87]:

```
### taken from original mode

## define X and y
X = df.drop(['Target'], axis=1)
y = df['Target']

## split data using 89/20 split
X_train, X_test, y_train, y_test = train_test_split(
  X,y , random_state = 42, test_size = 0.2, shuffle=True)

print('X_train : ')
print(X_train.shape)

print('X_test : ')
print(X_test.shape)

print('y_train : ')
print(y_train.shape)

print('y_test : ')
print(y_test.shape)
```

```
X_train :
(3539, 36)
X_test :
(885, 36)
y_train :
(3539,)
y_test :
(885,)
```

In [ ]:

```
## import Decision Tree Classifier
import sklearn.tree import DecisionTreeClassifier
```

In [88]:

```
## create Decision Tree classifier object
clf = DecisionTreeClassifier(criterion = 'entropy', max_depth = 3)
clf = clf.fit(X, y) ## train decision tree classifier
y_pred = clf.predict(X_test)

print("Accuracy", accuracy_score(y_test, y_pred))
```

```
Accuracy 0.8305084745762712
```

In [90]:

```
## import classification report
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.64      0.73       316
           1       0.82      0.94      0.88       569

    accuracy                           0.83       885
   macro avg       0.84      0.79      0.80       885
weighted avg       0.83      0.83      0.82       885
```

Conclusions and Ethical Implications

Based on the models explored in this milestone, there is a higher desire to move forward with the random forest classification model. However, there is still interest in exploring an artificial neural network. Within the time constraints of this milestone, the artificial neural network was not explored, as the ultimate goal of this project is to be able to present the findings of this analysis and model building as if presenting to executives. From personal experience, it is better to ground the any work in tangible attributes and relationships, which I am concerned about doing successfully with an artificial neural network, especially considering the size of the dataset.

Both the random forest classifier and decision tree models performed in the mid-80s regarding model accuracy. I anticipated that the random forest classifier would perform the best, as this model is an amalgamation of decision tree classifiers. Improving the model accuracy is the main concern in the final stretch of the project with the ultimate goal to be at or above 90% accuracy. More work needs to be done to finetune field relevancy, as dropping too many fields (as seen in the 2nd model rebuild) resulted in a significant drop in model accuracy. However, dropping 5 of the lowest relevant fields only resulted in a minimal accuracy improvement.

Despite the goal being for the model to perform at or above 90% accuracy, it is important to understand that the data is incredibly nuanced and categorical in nature. Compared to real-world data, the size of the dataset is relatively small when considering the scale of a real-world university's population. Fields with high relevance are also ethically challenging to source, as the fields include demographic and socioeconomic data. While PII data was removed from the dataset, PII would need to be available in order to expand and collect the necessary data for any future work.

The majority of this data is sensitive in nature. Ethical data collection and analysis is important to prevent biasing the data, manipulating meaning and results, or influencing any interpretation in the final presentation. Extensive work has been done throughout this project in order to present the data as accurately and objectively as possible within the constraints of this dataset. Such work includes feature selection and correlation matrix to ensure no subjectivity is introduced in feature dropping.

Overall, this project acknowledges the limits to how well the data can portray people and their actions. While it is helpful to understand student behavior and the attributes that lead to successful completion of an educational journey, attributing dropout as the antithesis to

References

Hanson, M. (2022, June 17). College Dropout Rates. Retrieved from Education Data Initiative: https://educationdata.org/college-dropout-rates (https://educationdata.org/college-dropout-rates)

Hore, A. (2022, June). Predict Dropout or Academic Success. Retrieved from Kaggle: https://www.kaggle.com/datasets/ankanhore545/dropout-or-academic-success?select=Dropout_Academic+Success+-+Sheet1.csv (https://www.kaggle.com/datasets/ankanhore545/dropout-or-academic-success?select=Dropout_Academic+Success+-+Sheet1.csv)

National Forum on Education Statistics. (2010, February). The Forum Guide to Ethics. Retrieved from https://nces.ed.gov/pubs2010/2010801.pdf (https://nces.ed.gov/pubs2010/2010801.pdf)