The order of this document:

- EDA
  - Financial Data
  - Student static data
  - Student progress data

- Comprehensive report of data analysis and predictive modeling along with the code documentation

Notes:

- This project was done using Python Data Science libraries
- The EDA was performed using Pandas-Profiling which does the entire process automatically.
- I have included the complete EDA results in HTML format inside the EDA folder. You can open it with any browser
- The Python codes are included so that you can execute them in Jupyter notebook
- To test your own set of Student IDs and verify my model, please read the instructions that I have provided on the first page after the EDA figures.

EDA: Financial Data

Overview

Overview | Warnings 43 | Reproduction

Dataset statistics

Number of variables	33
Number of observations	13769
Missing cells	305867
Missing cells (%)	67.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	3.5 MiB
Average record size in memory	264.0 B

Variable types

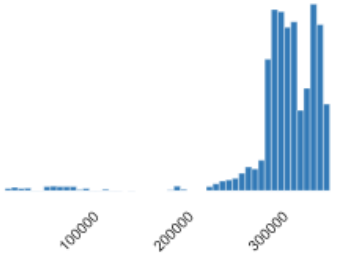
NUM	27
CAT	6

EDA: Financial Data

StudentID
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	13767
Distinct (%)	> 99.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

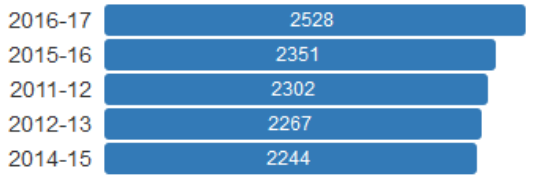
Mean	317095.175
Minimum	20932
Maximum	364184
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

cohort
Categorical

Distinct	6
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	107.6 KiB



Toggle details

cohort term
Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	107.6 KiB

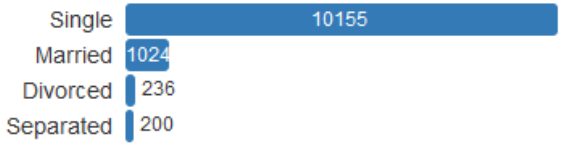
EDA: Financial Data

Marital Status

Categorical

MISSING

Distinct	4
Distinct (%)	< 0.1%
Missing	2154
Missing (%)	15.6%
Memory size	107.6 KiB



Toggle details

Adjusted Gross Income

Real number (ℝ)

MISSING
SKEWED
ZEROS

Distinct	5859
Distinct (%)	50.4%
Missing	2154
Missing (%)	15.6%
Infinite	0
Infinite (%)	0.0%

Mean	13124.92363
Minimum	-24326
Maximum	2576425
Zeros	5178
Zeros (%)	37.6%
Memory size	107.6 KiB

Toggle details

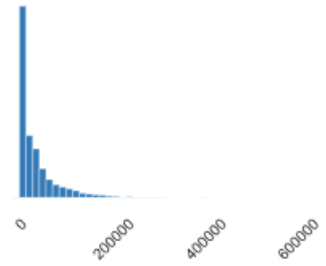
Parent Adjusted Gross Income

Real number (ℝ)

MISSING
ZEROS

Distinct	6053
Distinct (%)	52.1%
Missing	2154
Missing (%)	15.6%
Infinite	0
Infinite (%)	0.0%

Mean	28101.52165
Minimum	-62979
Maximum	657631
Zeros	5145
Zeros (%)	37.4%
Memory size	107.6 KiB



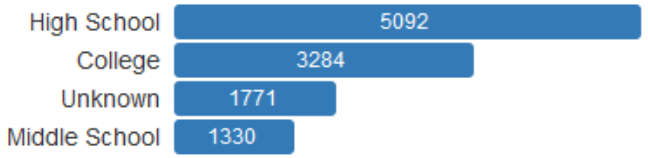
EDA: Financial Data

Father's Highest Grade Level

Categorical

MISSING

Distinct	4
Distinct (%)	< 0.1%
Missing	2292
Missing (%)	16.6%
Memory size	107.6 KiB



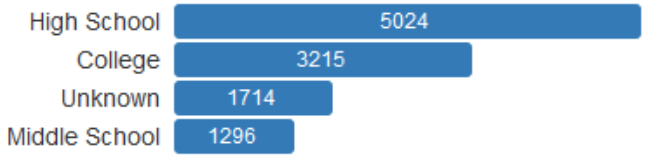
Toggle details

Mother's Highest Grade Level

Categorical

MISSING

Distinct	4
Distinct (%)	< 0.1%
Missing	2520
Missing (%)	18.3%
Memory size	107.6 KiB



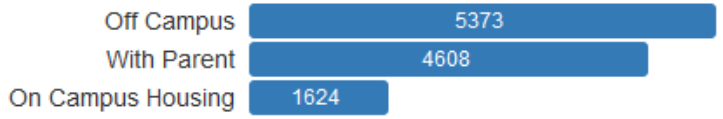
Toggle details

Housing

Categorical

MISSING

Distinct	3
Distinct (%)	< 0.1%
Missing	2164
Missing (%)	15.7%



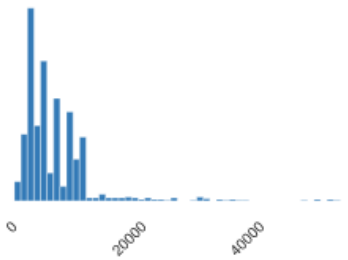
EDA: Financial Data

2012 Loan

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	219	**Mean**	7169.025869
Distinct (%)	17.7%	**Minimum**	337
Missing	12532	**Maximum**	55626
Missing (%)	91.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	107.6 KiB



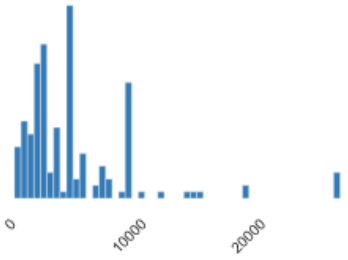
Toggle details

2012 Scholarship

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	57	**Mean**	5224.741813
Distinct (%)	33.3%	**Minimum**	283
Missing	13598	**Maximum**	27631.9
Missing (%)	98.8%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	107.6 KiB



Toggle details

2012 Work/Study

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	39	**Mean**	1872.995146
Distinct (%)	37.9%	**Minimum**	200
Missing	13666	**Maximum**	3000
Missing (%)	99.3%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	107.6 KiB

EDA: Financial Data

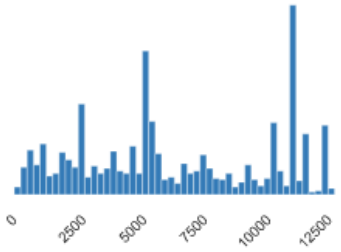
2012 Grant

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	450
Distinct (%)	33.2%
Missing	12415
Missing (%)	90.2%
Infinite	0
Infinite (%)	0.0%

Mean	6660.931617
Minimum	79.09
Maximum	13263
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

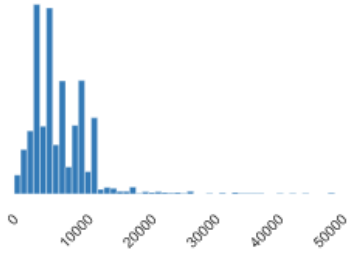
2013 Loan

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	339
Distinct (%)	15.5%
Missing	11582
Missing (%)	84.1%
Infinite	0
Infinite (%)	0.0%

Mean	7156.096278
Minimum	103
Maximum	50555
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

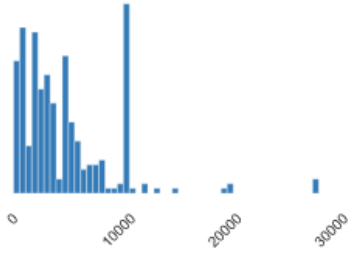
2013 Scholarship

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	90
Distinct (%)	29.0%
Missing	13459
Missing (%)	97.7%
Infinite	0
Infinite (%)	0.0%

Mean	4792.646903
Minimum	23
Maximum	28737.1
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



EDA: Financial Data

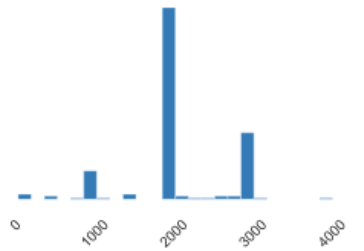
2013 Work/Study

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	24
Distinct (%)	13.4%
Missing	13590
Missing (%)	98.7%
Infinite	0
Infinite (%)	0.0%

Mean	2084.268156
Minimum	25
Maximum	4000
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

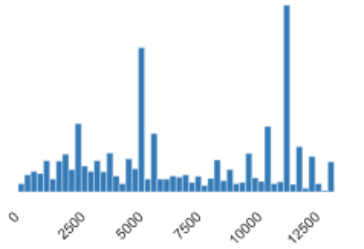
2013 Grant

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	629
Distinct (%)	27.1%
Missing	11450
Missing (%)	83.2%
Infinite	0
Infinite (%)	0.0%

Mean	7094.158098
Minimum	162
Maximum	13790
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

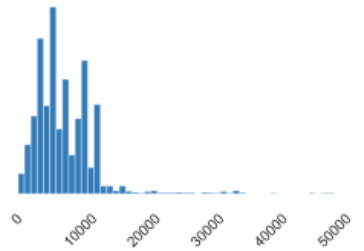
2014 Loan

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	510
Distinct (%)	18.6%
Missing	11028
Missing (%)	80.1%
Infinite	0
Infinite (%)	0.0%

Mean	7279.854064
Minimum	128
Maximum	49845
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



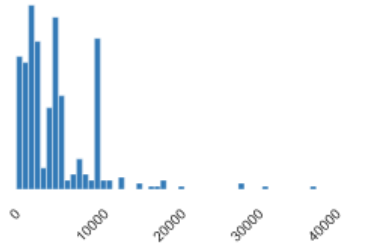
EDA: Financial Data

2014 Scholarship

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	109	**Mean**	4998.862332
Distinct (%)	26.2%	**Minimum**	100
Missing	13353	**Maximum**	38850.57
Missing (%)	97.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	107.6 KiB



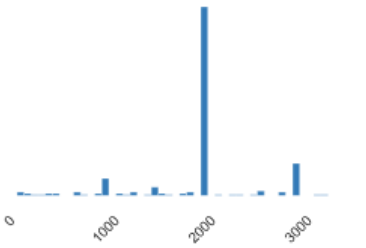
Toggle details

2014 Work/Study

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	44	**Mean**	1932.888889
Distinct (%)	18.1%	**Minimum**	70
Missing	13526	**Maximum**	3300
Missing (%)	98.2%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	107.6 KiB



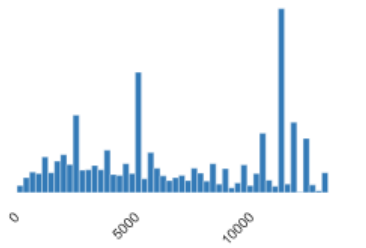
Toggle details

2014 Grant

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	856	**Mean**	7208.105896
Distinct (%)	29.2%	**Minimum**	97.24
Missing	10840	**Maximum**	14001
Missing (%)	78.7%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	107.6 KiB



EDA: Financial Data

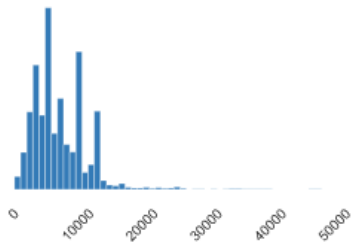
2015 Loan

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	566
Distinct (%)	18.6%
Missing	10718
Missing (%)	77.8%
Infinite	0
Infinite (%)	0.0%

Mean	7240.876106
Minimum	25
Maximum	47824
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

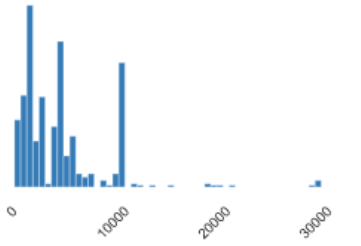
2015 Scholarship

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	141
Distinct (%)	23.7%
Missing	13174
Missing (%)	95.7%
Infinite	0
Infinite (%)	0.0%

Mean	4754.907479
Minimum	200
Maximum	30477.91
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

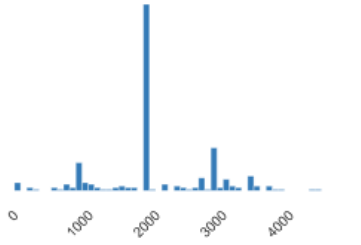
2015 Work/Study

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	71
Distinct (%)	28.5%
Missing	13520
Missing (%)	98.2%
Infinite	0
Infinite (%)	0.0%

Mean	2127.195783
Minimum	10
Maximum	4600
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



EDA: Financial Data

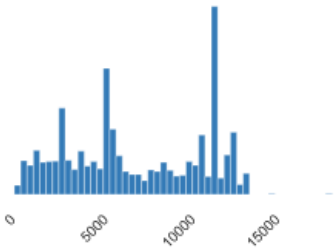
2015 Grant

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	917
Distinct (%)	26.9%
Missing	10365
Missing (%)	75.3%
Infinite	0
Infinite (%)	0.0%

Mean	7369.863796
Minimum	209
Maximum	19038
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

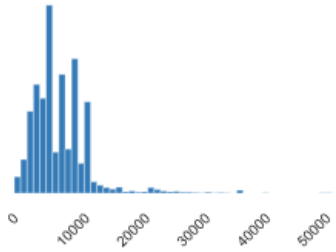
2016 Loan

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	574
Distinct (%)	18.1%
Missing	10594
Missing (%)	76.9%
Infinite	0
Infinite (%)	0.0%

Mean	7625.026822
Minimum	103
Maximum	52880
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

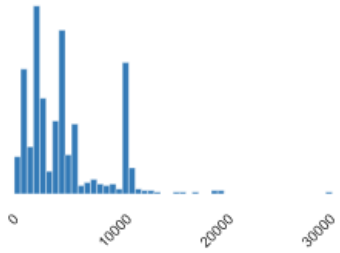
2016 Scholarship

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	159
Distinct (%)	23.2%
Missing	13084
Missing (%)	95.0%
Infinite	0
Infinite (%)	0.0%

Mean	4897.317679
Minimum	28.3
Maximum	31265.5
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



EDA: Financial Data

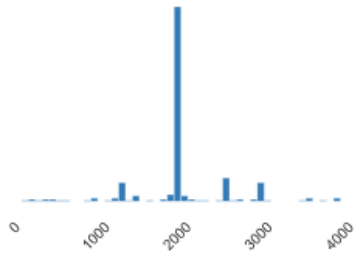
2016 Work/Study

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	46
Distinct (%)	16.9%
Missing	13497
Missing (%)	98.0%
Infinite	0
Infinite (%)	0.0%

Mean	2036.352941
Minimum	75
Maximum	4000
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

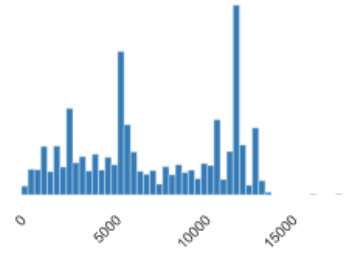
2016 Grant

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	1094
Distinct (%)	29.6%
Missing	10075
Missing (%)	73.2%
Infinite	0
Infinite (%)	0.0%

Mean	7458.963519
Minimum	9.69
Maximum	18505
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

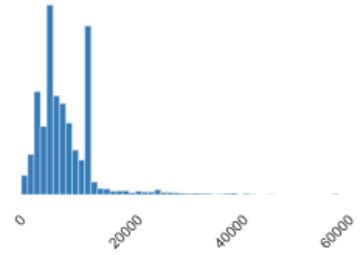
2017 Loan

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	690
Distinct (%)	20.8%
Missing	10445
Missing (%)	75.9%
Infinite	0
Infinite (%)	0.0%

Mean	8256.243983
Minimum	103
Maximum	60118
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



EDA: Financial Data

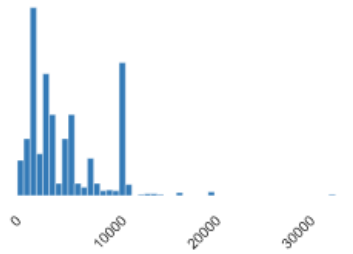
2017 Scholarship

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	230
Distinct (%)	23.4%
Missing	12784
Missing (%)	92.8%
Infinite	0
Infinite (%)	0.0%

Mean	5024.025198
Minimum	100
Maximum	33847.9
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

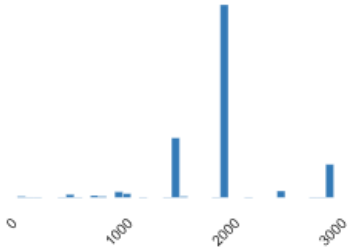
2017 Work/Study

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
MISSING

Distinct	39
Distinct (%)	10.6%
Missing	13402
Missing (%)	97.3%
Infinite	0
Infinite (%)	0.0%

Mean	1928.544441
Minimum	45
Maximum	3000
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



Toggle details

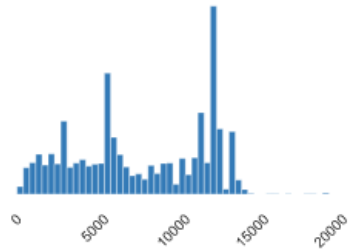
2017 Grant

Real number ($\mathbb{R}_{\geq 0}$)

MISSING

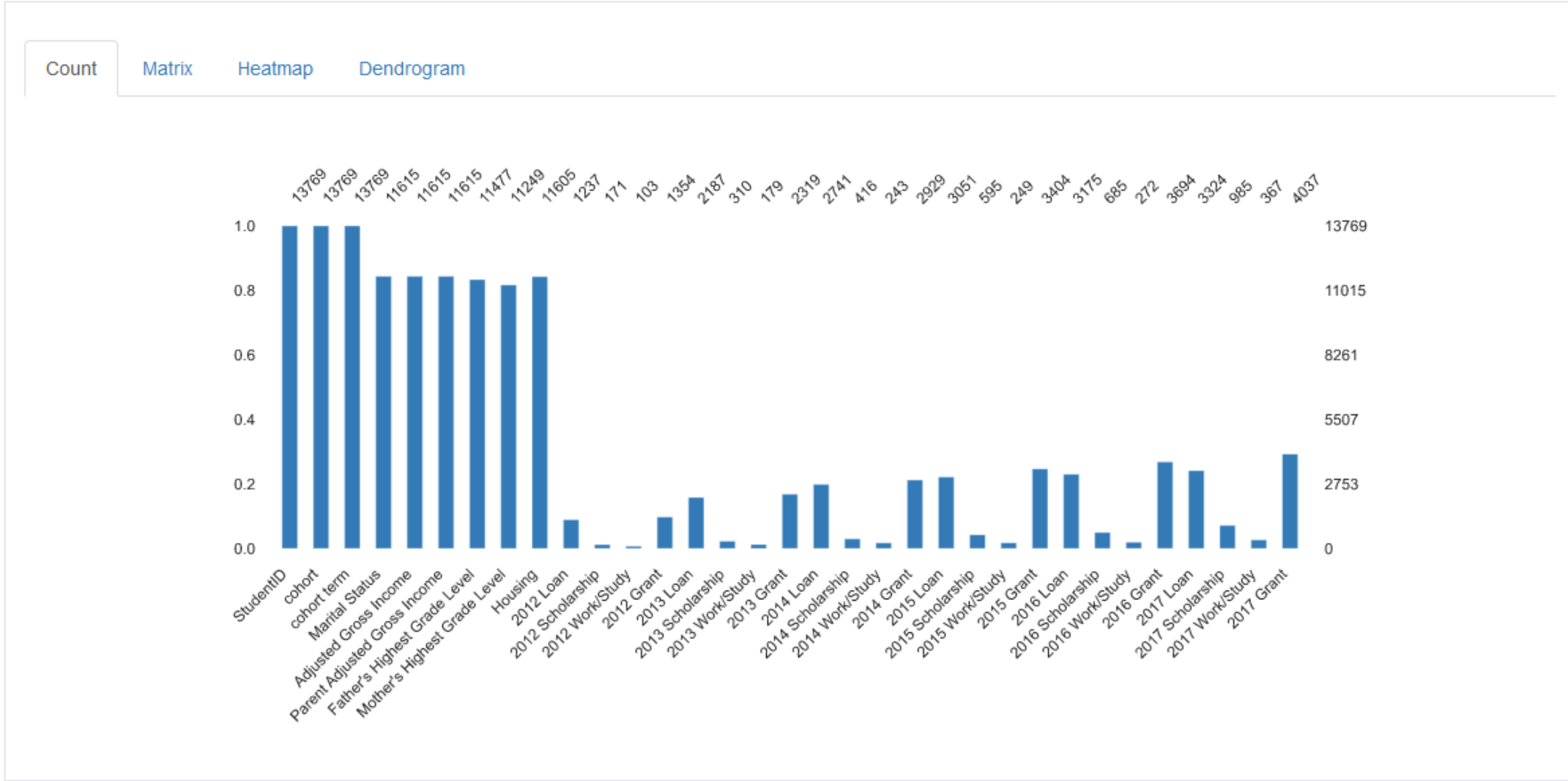
Distinct	1104
Distinct (%)	27.3%
Missing	9732
Missing (%)	70.7%
Infinite	0
Infinite (%)	0.0%

Mean	7794.156061
Minimum	0.1
Maximum	19823
Zeros	0
Zeros (%)	0.0%
Memory size	107.6 KiB



EDA: Financial Data

Missing values



EDA: Student Static Data

Overview

Overview | Warnings 20 | Reproduction

Dataset statistics

Number of variables	35
Number of observations	13261
Missing cells	26608
Missing cells (%)	5.7%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	3.5 MiB
Average record size in memory	280.0 B

Variable types

CAT	22
NUM	9
BOOL	3
UNSUPPORTED	1

EDA: Student Static Data

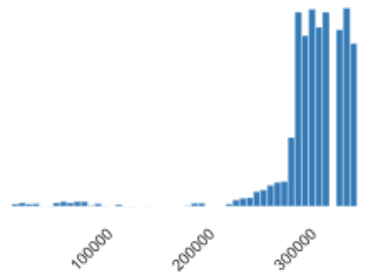
StudentID

Real number ($\mathbb{R}_{\geq 0}$)

UNIQUE

Distinct	13261
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	316150.6113
Minimum	20932
Maximum	359783
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB

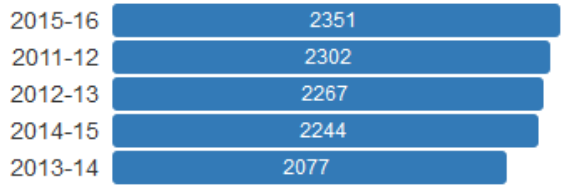


Toggle details

Cohort

Categorical

Distinct	6
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

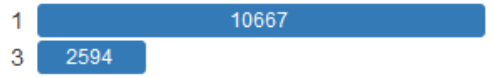


Toggle details

CohortTerm

Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



EDA: Student Static Data

~~Campus~~
~~Unsupported~~

MISSING
REJECTED
UNSUPPORTED

Missing	13261
Missing (%)	100.0%
Memory size	103.7 KiB

Address1
Categorical

HIGH CARDINALITY
UNIFORM

Distinct	12703
Distinct (%)	96.6%
Missing	113
Missing (%)	0.9%
Memory size	103.6 KiB

NJCU-Registrar's Office	6
Westview Towers	5
John F	5
Summit Apts	5
Jackson Garden Apt	4
Other values (12698)	13123

Toggle details

Address2
Categorical

HIGH CARDINALITY
MISSING

Distinct	290
Distinct (%)	74.6%
Missing	12872
Missing (%)	97.1%
Memory size	103.6 KiB

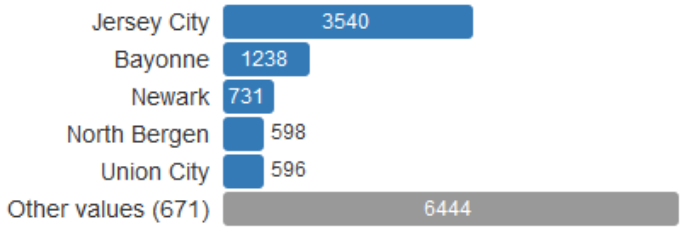
1	14
2	13
Apt 2	12
2039 John F Kennedy Blvd	6
2nd Floor	5
Other values (285)	339

EDA: Student Static Data

City
Categorical

HIGH CARDINALITY

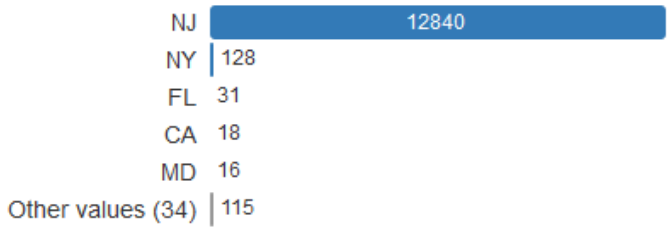
Distinct	676
Distinct (%)	5.1%
Missing	114
Missing (%)	0.9%
Memory size	103.6 KiB



Toggle details

State
Categorical

Distinct	39
Distinct (%)	0.3%
Missing	113
Missing (%)	0.9%
Memory size	103.6 KiB



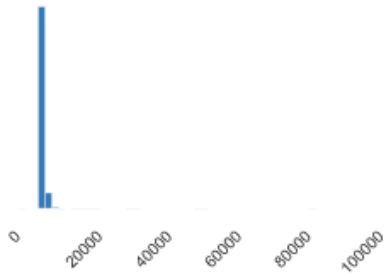
Toggle details

Zip
Real number ($\mathbb{R}_{\geq 0}$)

MISSING

Distinct	662
Distinct (%)	5.0%
Missing	134
Missing (%)	1.0%
Infinite	0
Infinite (%)	0.0%

Mean	7790.287042
Minimum	747
Maximum	98118
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB



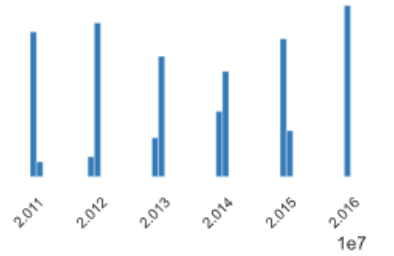
EDA: Student Static Data

RegistrationDate

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	1045
Distinct (%)	7.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	20136109.45
Minimum	20110111
Maximum	20160912
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB

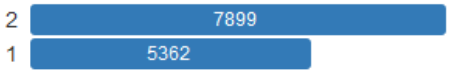


Toggle details

Gender

Categorical

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



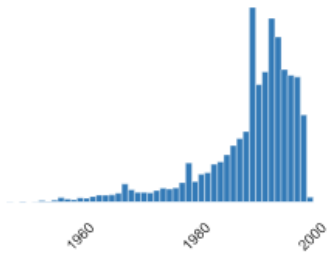
Toggle details

BirthYear

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	55
Distinct (%)	0.4%
Missing	1
Missing (%)	< 0.1%
Infinite	0
Infinite (%)	0.0%

Mean	1988.916591
Minimum	1945
Maximum	2000
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB



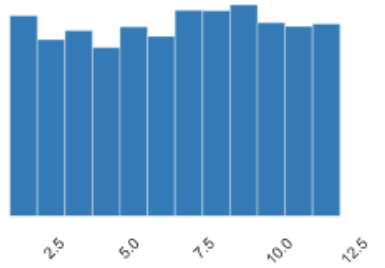
EDA: Student Static Data

BirthMonth

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	12
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	6.580876254
Minimum	1
Maximum	12
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB

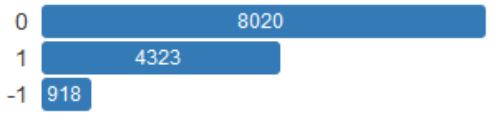


Toggle details

Hispanic

Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



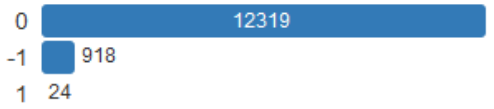
Toggle details

AmericanIndian

Categorical

HIGH CORRELATION

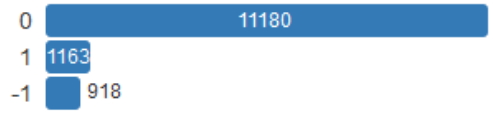
Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



EDA: Student Static Data

Asian
Categorical

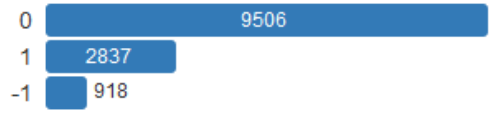
Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



Toggle details

Black
Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

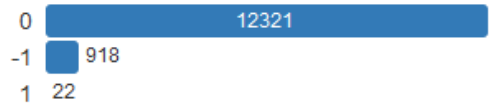


Toggle details

NativeHawaiian
Categorical

HIGH CORRELATION

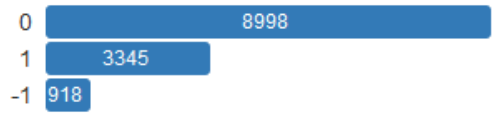
Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



EDA: Student Static Data

White
Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



Toggle details

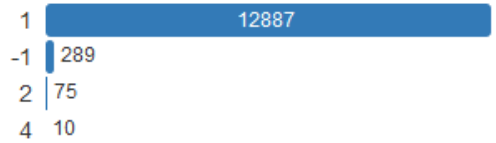
TwoOrMoreRace
Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

Toggle details

HSDip
Categorical

Distinct	4
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



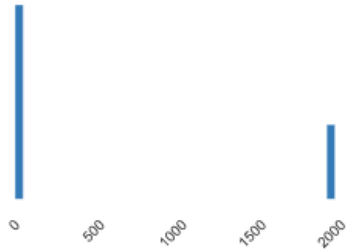
EDA: Student Static Data

HSDipYr

Real number (ℝ)

Distinct	39
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	557.8221854
Minimum	-1
Maximum	2016
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB



Toggle details

HSGPAUnwtd

Real number (ℝ)

Distinct	214
Distinct (%)	1.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	0.1624319433
Minimum	-1
Maximum	4
Zeros	0
Zeros (%)	0.0%
Memory size	103.6 KiB

Toggle details

~~HSGPAWtd~~

~~Categorical~~

CONSTANT
REJECTED

Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

EDA: Student Static Data

~~FirstGen~~
~~Categorical~~

CONSTANT
REJECTED

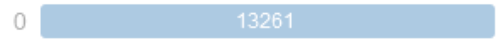
Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

Toggle details

~~DualHSSummerEnroll~~
~~Boolean~~

CONSTANT
REJECTED

Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

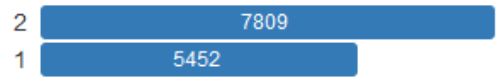


Toggle details

EnrollmentStatus
Categorical

HIGH CORRELATION
HIGH CORRELATION

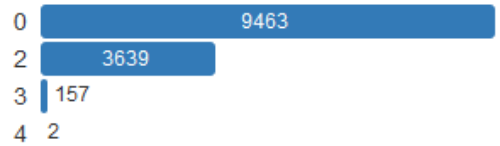
Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



EDA: Student Static Data

HighDeg
Categorical

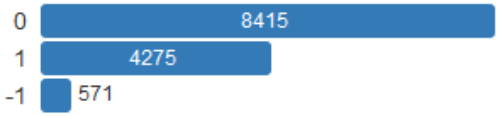
Distinct	4
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



Toggle details

MathPlacement
Categorical

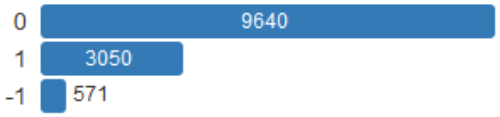
Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



Toggle details

EngPlacement
Categorical

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

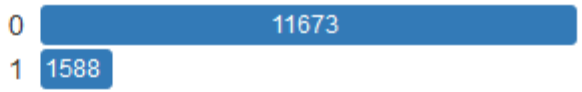


EDA: Student Static Data

GatewayMathStatus

Boolean

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB

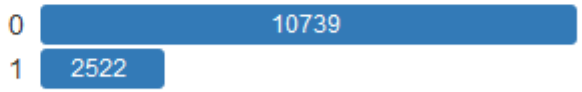


Toggle details

GatewayEnglishStatus

Boolean

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	103.6 KiB



EDA: Student Progress Data

Overview

Overview | Warnings 9 | Reproduction

Dataset statistics

Number of variables	17
Number of observations	57945
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	7.5 MiB
Average record size in memory	136.0 B

Variable types

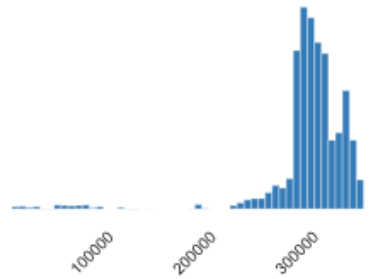
CAT	10
NUM	6
BOOL	1

EDA: Student Progress Data

StudentID
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	13767
Distinct (%)	23.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

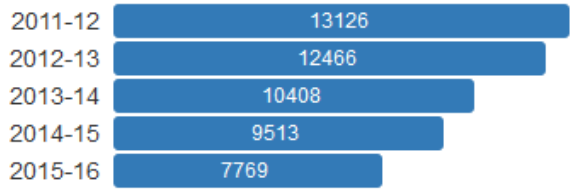
Mean	310884.0867
Minimum	20932
Maximum	364184
Zeros	0
Zeros (%)	0.0%
Memory size	452.7 KiB



Toggle details

Cohort
Categorical

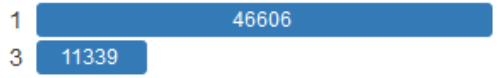
Distinct	6
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



Toggle details

CohortTerm
Categorical

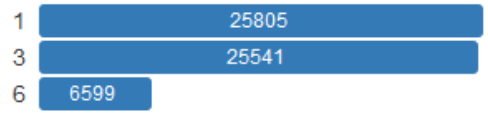
Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



EDA: Student Progress Data

Term
Categorical

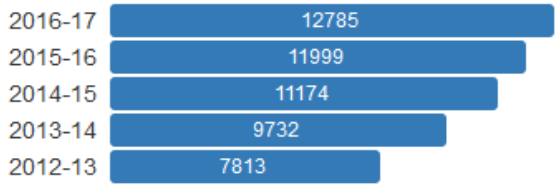
Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



Toggle details

AcademicYear
Categorical

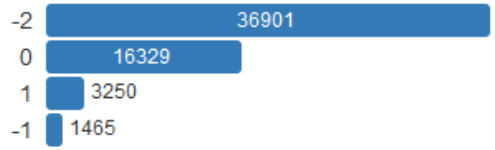
Distinct	6
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



Toggle details

CompleteDevMath
Categorical

Distinct	4
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB

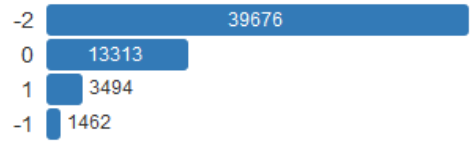


EDA: Student Progress Data

CompleteDevEnglish

Categorical

Distinct	4
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



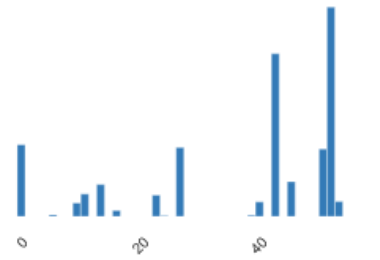
Toggle details

Major1

Real number (ℝ)

ZEROS

Distinct	41	**Mean**	36.86532498
Distinct (%)	0.1%	**Minimum**	-1
Missing	0	**Maximum**	54.0101
Missing (%)	0.0%	**Zeros**	5341
Infinite	0	**Zeros (%)**	9.2%
Infinite (%)	0.0%	**Memory size**	452.7 KiB

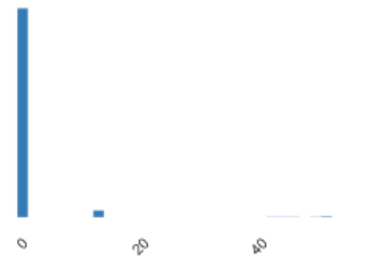


Toggle details

Major2

Real number (ℝ)

Distinct	30	**Mean**	0.188326998
Distinct (%)	0.1%	**Minimum**	-1
Missing	0	**Maximum**	54.0101
Missing (%)	0.0%	**Zeros**	0
Infinite	0	**Zeros (%)**	0.0%
Infinite (%)	0.0%	**Memory size**	452.7 KiB



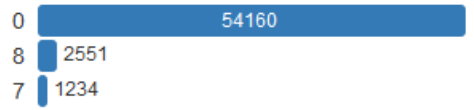
EDA: Student Progress Data

Complete1

Categorical

HIGH CORRELATION

Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



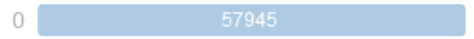
Toggle details

Complete2

Boolean

CONSTANT
REJECTED

Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB



Toggle details

CompleteCIP1

Real number (ℝ)

HIGH CORRELATION

Distinct	32
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	1.017689335
Minimum	-2
Maximum	54.0101
Zeros	0
Zeros (%)	0.0%
Memory size	452.7 KiB

EDA: Student Progress Data

~~CompleteCIP2~~
~~Categorical~~

CONSTANT
REJECTED

Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB

-2 57945

Toggle details

~~TransferIntent~~
~~Categorical~~

CONSTANT
REJECTED

Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB

-1 57945

Toggle details

~~DegreeTypeSought~~
~~Categorical~~

CONSTANT
REJECTED

Distinct	1
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	452.7 KiB

6 57945

EDA: Student Progress Data

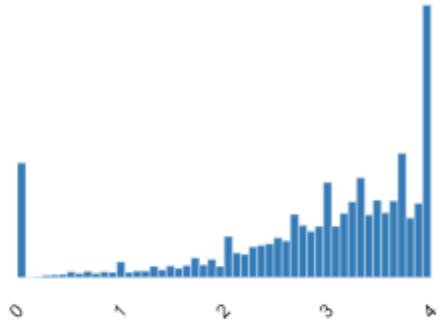
TermGPA

Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Distinct	382
Distinct (%)	0.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	2.893633273
Minimum	0
Maximum	4
Zeros	3339
Zeros (%)	5.8%
Memory size	452.7 KiB



Toggle details

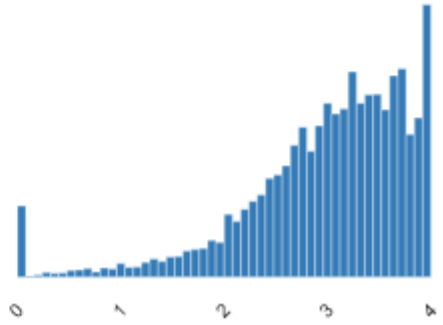
CumGPA

Real number ($\mathbb{R}_{\geq 0}$)

ZEROS

Distinct	395
Distinct (%)	0.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	2.973698162
Minimum	0
Maximum	4
Zeros	1018
Zeros (%)	1.8%
Memory size	452.7 KiB



DataAnalysisCode_MJ_Sarfi

November 30, 2020

This code implements several different types of Machine Learning models, after cleaning the provided data sets, to perform a binary classification. The outcome of the code is a model with a reasonable accuracy that predicts whether a student will drop out or not given their student ID.

For Promazo evaluator: In order to test your own set of StudentIDs to verify my model, please insert your list of StudentIDs in this .*csv file:

'Student Dropout Prediction Data/Test Data/test_student_IDs.csv'

The final outcome of the model will be saved in:

'Student Dropout Prediction Data/Test Data/FinalOutcome.csv'

It will have the same format as the sample submission file that you provided.

[31]:
```
# importing all the essential libraries and ML packages
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
import seaborn as sns
from random import sample
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from IPython.display import Image
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import mutual_info_classif
from sklearn.model_selection import train_test_split,␣
↪StratifiedKFold,cross_val_score
from sklearn.metrics import classification_report, accuracy_score,␣
↪confusion_matrix
```

0.1 Importing the Data and cleaning it

[32]:
```
# First, import the financial data provided by Promazo
# please note that I spotted some inconsistencies between the names of these␣
 ↪three columns in the financial
# dataset and the other data sets. ['ID with leading', 'cohort term', 'cohort'].
 ↪ So, I had to rename them.
financialData = pd.read_csv('Student Dropout Prediction Data/Student Financial␣
 ↪Aid Data/2011-2017_Cohorts_Financial_Aid_and_Fafsa_Data.csv')
financialData = financialData.rename(columns={'ID with leading': 'StudentID'})
financialData = financialData.rename(columns={'cohort term': 'CohortTerm'})
financialData = financialData.rename(columns={'cohort': 'Cohort'})
```

[33]:
```
# Now, let's import the student static data that is stored in multiple files
directory = 'Student Dropout Prediction Data/Student Static Data/'

stFall2011 = pd.read_csv(directory + 'Fall 2011_ST.csv')
stFall2012 = pd.read_csv(directory + 'Fall 2012.csv')
stFall2013 = pd.read_csv(directory + 'Fall 2013.csv')
stFall2014 = pd.read_csv(directory + 'Fall 2014.csv')
stFall2015 = pd.read_csv(directory + 'Fall 2015.csv')
stFall2016 = pd.read_csv(directory + 'Fall 2016.csv')

stSpring2012 = pd.read_csv(directory + 'Spring 2012_ST.csv')
stSpring2013 = pd.read_csv(directory + 'Spring 2013.csv')
stSpring2014 = pd.read_csv(directory + 'Spring 2014.csv')
stSpring2015 = pd.read_csv(directory + 'Spring 2015.csv')
stSpring2016 = pd.read_csv(directory + 'Spring 2016.csv')
```

[34]:
```
# Then, we import the student progress data that is stored in multiple files
directory = 'Student Dropout Prediction Data/Student Progress Data/'

spFall2011 = pd.read_csv(directory + 'Fall 2011_SP.csv')
spFall2012 = pd.read_csv(directory + 'Fall 2012_SP.csv')
spFall2013 = pd.read_csv(directory + 'Fall 2013_SP.csv')
spFall2014 = pd.read_csv(directory + 'Fall 2014_SP.csv')
spFall2015 = pd.read_csv(directory + 'Fall 2015_SP.csv')
spFall2016 = pd.read_csv(directory + 'Fall 2016_SP.csv')
spSpring2012 = pd.read_csv(directory + 'Spring 2012_SP.csv')
spSpring2013 = pd.read_csv(directory + 'Spring 2013_SP.csv')
spSpring2014 = pd.read_csv(directory + 'Spring 2014_SP.csv')
spSpring2015 = pd.read_csv(directory + 'Spring 2015_SP.csv')
spSpring2016 = pd.read_csv(directory + 'Spring 2016_SP.csv')
spSpring2017 = pd.read_csv(directory + 'Spring 2017_SP.csv')
spSum2012 = pd.read_csv(directory + 'Sum 2012.csv')
spSum2013 = pd.read_csv(directory + 'Sum 2013.csv')
spSum2014 = pd.read_csv(directory + 'Sum 2014.csv')
```

```
spSum2015 = pd.read_csv(directory + 'Sum 2015.csv')
spSum2016 = pd.read_csv(directory + 'Sum 2016.csv')
spSum2017 = pd.read_csv(directory + 'Sum 2017.csv')
```

[35]:
```
# importing the student IDs that were provided by Promazo in order to be used␣
↪for model training
# these IDs will be used to extract the relevant data for training
TrainLabels = pd.read_csv('Student Dropout Prediction Data/DropoutTrainLabels.
↪csv')
```

Then, I concatenate multiple pieces of student static and progress data to create one dataframes for each. This will make the future steps easier

[36]:
```
studentStaticData = pd.concat([stFall2011, stFall2012, stFall2013, stFall2014,␣
↪stFall2015, stFall2016,
                               stSpring2012, stSpring2013, stSpring2014,␣
↪stSpring2015,
                               stSpring2016], ignore_index=True)
```

[37]:
```
studentProgressData = pd.concat([spFall2011, spFall2012, spFall2013,␣
↪spFall2014, spFall2015,
                                 spFall2016, spSpring2012, spSpring2013,␣
↪spSpring2014, spSpring2015,
                                 spSpring2016, spSpring2017, spSum2012,␣
↪spSum2013, spSum2014, spSum2015,
                                 spSum2016, spSum2017], ignore_index=True)
```

0.1.1 Imputing the missing values in each data set

The exploratory data analysis shows that there are many missing values in the financial dataset. Since most of the students are single, I will replace the missing values in the marital status column with "Single". Also, the missing values in the Housing column are replaced with "Off Campus" and the NaN values in the Father and Mother Highest Grade Level are filled with "Unknown". The rest of the missing values in the financial data are related to the loan and grant data which I am going to assume they are all zero.

[38]:
```
# After replacing the empty values (missing) in marital status and income with␣
↪appropriate entries
# we can securely replace all other missing values with 0. Because the rest are␣
↪numbers and not objects
financialData['Marital Status'] = financialData['Marital Status'].
↪fillna("Single")
financialData['Housing'] = financialData['Housing'].fillna("Off Campus")
financialData["Father's Highest Grade Level"] = financialData["Father's Highest␣
↪Grade Level"].fillna("Unknown")
financialData["Mother's Highest Grade Level"] = financialData["Mother's Highest␣
↪Grade Level"].fillna("Unknown")
```

```
financialData = financialData.fillna(0)
```

According to the EDA for the student static data, there are NaN values as well as irrelevant data that need to be accounted for as explained in the comments below:

[39]:
```
# All campus data is missing for all students -> let's remove the entire campus␣
 ↪column not significant
# All values of HSGPAWtd, FirstGen, DualHSSummerEnroll aremissing. will drop␣
 ↪the columns
# 97% of the data from Address2 is missing. Address1 has no correlation with␣
 ↪student dropou. let's drop them
# state - zip code and city are part of the address too. So, we will drop them
# CumLoanAtEntry is either -1 or -2. remove it entirely
studentStaticData = studentStaticData.drop(['Campus', 'HSDip', 'HSGPAWtd',␣
 ↪'FirstGen',
                                            'DualHSSummerEnroll',␣
 ↪'CumLoanAtEntry',
                                            'Address1', 'Address2', 'Zip',␣
 ↪'State', 'City'], 1)
# replace -1 values in HSGPAUnwtd with 0
studentStaticData['HSGPAUnwtd'] = studentStaticData['HSGPAUnwtd'].apply(lambda␣
 ↪x: 0 if x==-1 else x)

# replace missing birth year with the most frequent value (mode)
studentStaticData['BirthYear'] = studentStaticData['BirthYear'].fillna(1993)

# negative values of NumColCredAttemptTransfer, NumColCredAcceptTransfer,␣
 ↪MathPlacement are replaced with zero
studentStaticData['NumColCredAttemptTransfer'] =␣
 ↪studentStaticData['NumColCredAttemptTransfer'].apply(lambda x: 0 if x<0 else␣
 ↪x)
studentStaticData['NumColCredAcceptTransfer'] =␣
 ↪studentStaticData['NumColCredAcceptTransfer'].apply(lambda x: 0 if x<0 else␣
 ↪x)
studentStaticData['MathPlacement'] = studentStaticData['MathPlacement'].
 ↪apply(lambda x: 0 if x<0 else x)

# Let's put all racial data into one column (Ehtnicity) so that it is easier to␣
 ↪handle
studentStaticData['Ethnicity'] = ['Unknown']*len(studentStaticData)

for i in range(len(studentStaticData)):
    if (studentStaticData['Hispanic'][i] == 1):
        studentStaticData['Ethnicity'][i] = "Hispanic"
    elif (studentStaticData['AmericanIndian'][i] == 1):
        studentStaticData['Ethnicity'][i] = "AmericanIndian"
```

```
    elif (studentStaticData['Asian'][i] == 1):
        studentStaticData['Ethnicity'][i] = "Asian"
    elif (studentStaticData['Black'][i] == 1):
        studentStaticData['Ethnicity'][i] = "Black"
    elif (studentStaticData['NativeHawaiian'][i] == 1):
        studentStaticData['Ethnicity'][i] = "NativeHawaiian"
    elif (studentStaticData['White'][i] == 1):
        studentStaticData['Ethnicity'][i] = "White"
    elif (studentStaticData['TwoOrMoreRace'][i] == 1):
        studentStaticData['Ethnicity'][i] = "TwoOrMoreRace"
    else :
        studentStaticData['Ethnicity'][i] = "Unknown"

# Remove all kinds of racial data after incorporating them into Ethnicity
studentStaticData = studentStaticData.drop(['Hispanic', 'AmericanIndian',␣
↪'Asian', 'Black',
                                            'White', 'TwoOrMoreRace',␣
↪"NativeHawaiian"], 1)
```

```
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:35:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:25:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:31:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:39:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:29:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:37:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:27:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
C:\Users\mjsar\anaconda3\lib\site-packages\ipykernel_launcher.py:33:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

According to the EDA for the student progress data, there are NaN values as well as irrelevant data that need to be accounted for as explained in the comments below:

[40]:
```
# values of Major1 and Major2 that are negative are imputed to 0
studentProgressData['Major1'] = studentProgressData['Major1'].apply(lambda x: 0␣
↪if x<0 else x)
studentProgressData['Major2'] = studentProgressData['Major2'].apply(lambda x: 0␣
↪if x<0 else x)

# removing a bunch of columns because eiter they are missing or they all have a␣
↪constant value
studentProgressData = studentProgressData.drop(['Complete2', 'CompleteCIP2',
                                                'TransferIntent',␣
↪'DegreeTypeSought'], 1)
```

Now, we need to merge all the input data sets together. The column that joins them all together is the student ID.

[41]:
```
# Join student static and student progress data and financial data
static_progressData = pd.merge(studentStaticData, studentProgressData, how =␣
↪'inner',
                               on=["StudentID", "Cohort", "CohortTerm"])
static_progress_financialData = pd.merge(financialData, static_progressData,
                                         on=["StudentID", "Cohort",␣
↪"CohortTerm"], how='inner')
# only use the student IDs that are provided for training. So, let's merge the␣
↪training labels with
```

```
# the combined data set
combinedData_trainLabels = pd.merge(static_progress_financialData, TrainLabels,␣
↪on=["StudentID"])
```

Our data contains many columns that have categorical data in form of string or objects, such as marital status. We need to convert this data to a numerical data type because many machine learning models perform poorly with non-numerical categorical data. This process is known as label encoding, which is done using the following function. We will basically associate each Marital status condition, such Single, with an integer.

[42]:
```
# objects first need to be converted from object type to categoty type for␣
↪Label Encoding
def label_encoding(inputDF):
    # only perform label encoding on "object" data types not the int/float␣
↪numerical data
    inputDF['Cohort'] = inputDF['Cohort'].astype('category')
    inputDF['Marital Status'] = inputDF['Marital Status'].astype('category')
    inputDF["Father's Highest Grade Level"] = inputDF["Father's Highest Grade␣
↪Level"].astype('category')
    inputDF["Mother's Highest Grade Level"] = inputDF["Mother's Highest Grade␣
↪Level"].astype('category')
    inputDF['Housing'] = inputDF['Housing'].astype('category')
    inputDF['AcademicYear'] = inputDF['AcademicYear'].astype('category')
    inputDF['Ethnicity'] = inputDF['Ethnicity'].astype('category')

    inputDF["Cohort"] = inputDF["Cohort"].cat.codes
    inputDF["Marital Status"] = inputDF["Marital Status"].cat.codes
    inputDF["Father's Highest Grade Level"] = inputDF["Father's Highest Grade␣
↪Level"].cat.codes
    inputDF["Mother's Highest Grade Level"] = inputDF["Mother's Highest Grade␣
↪Level"].cat.codes
    inputDF["Housing"] = inputDF["Housing"].cat.codes
    inputDF["AcademicYear"] = inputDF["AcademicYear"].cat.codes
    inputDF["Ethnicity"] = inputDF["Ethnicity"].cat.codes

    return inputDF
```

After label encoding the categorical data, we attempt to split the data into train and test portions. Also, the target variable (Dropout) is separated from the input variables.

[43]:
```
labelEncoded_combinedData = label_encoding(combinedData_trainLabels)
# remove 'StudentID', 'Dropout', 'BirthMonth' because they are extraneous as␣
↪predicting variables
X = labelEncoded_combinedData.drop({'StudentID', 'Dropout', 'BirthMonth'}, axis␣
↪= 1)
Y = labelEncoded_combinedData['Dropout']
```

```
# 75% of data as training and 25% of the data will be used for validating our␣
 ↪ML models
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, train_size = 0.75,␣
 ↪random_state = 7)
```

0.2 Extracting preliminary insight based on Feature Importance analysis

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

Because we are dealing with a simple binary classification problem here, we will use univariate feature selection methods. Univariate feature selection works by selecting the best features based on univariate statistical tests. We compare each feature to the target variable, to see whether there is any statistically significant relationship between them. When we analyze the relationship between one feature and the target variable, we ignore the other features. That is why it is called 'univariate'. Each feature has its test score.Finally, all the test scores are compared, and the features with top scores will be selected.

There three main methods for performing Univariate feature selection. One if Chi-square method which works best when both input and the target variables are categorical. Also, all values must be positive. This is not a suitable method for our case clearly. The other methods are 1) ANOVA and 2)Mutual information test. Both of these methods are agnostic with respect to the data type. Hence, I demonstrate results based on both methods here.

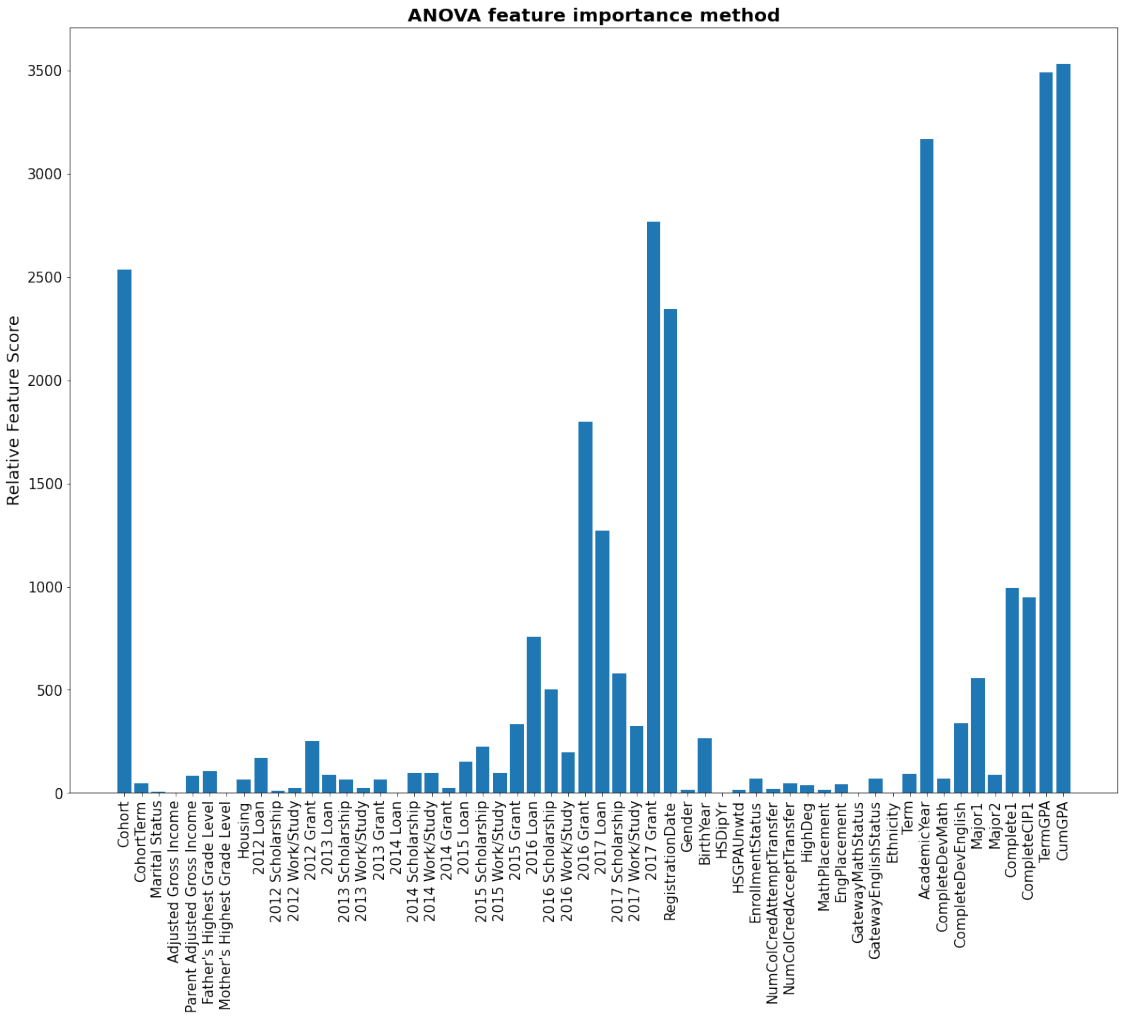
[44]:
```
# This function performs Mutual information test for feature selection
def mutual_select_features(X_train, y_train, X_test):
        # configure to select all features
        fs = SelectKBest(score_func=mutual_info_classif, k='all')
        # learn relationship from training data
        fs.fit(X_train, y_train)
        # transform train input data
        X_train_fs = fs.transform(X_train)
        # transform test input data
        X_test_fs = fs.transform(X_test)
        return X_train_fs, X_test_fs, fs

# This function performs ANOVA method for feature selection
def ANOVA_select_features(X_train, y_train, X_test):
        # configure to select all features
        fs = SelectKBest(score_func=f_classif, k='all')
        # learn relationship from training data
        fs.fit(X_train, y_train)
        # transform train input data
        X_train_fs = fs.transform(X_train)
        # transform test input data
        X_test_fs = fs.transform(X_test)
```

```
        return X_train_fs, X_test_fs, fs
```

ANOVA Feature Selection The feature selection results show a significant correlation between [‘Cohot, cumGPA, TermGPA, Academic Year’] and dropping out. On the other hand, parameters such as Marital status and Gender are not that significant.

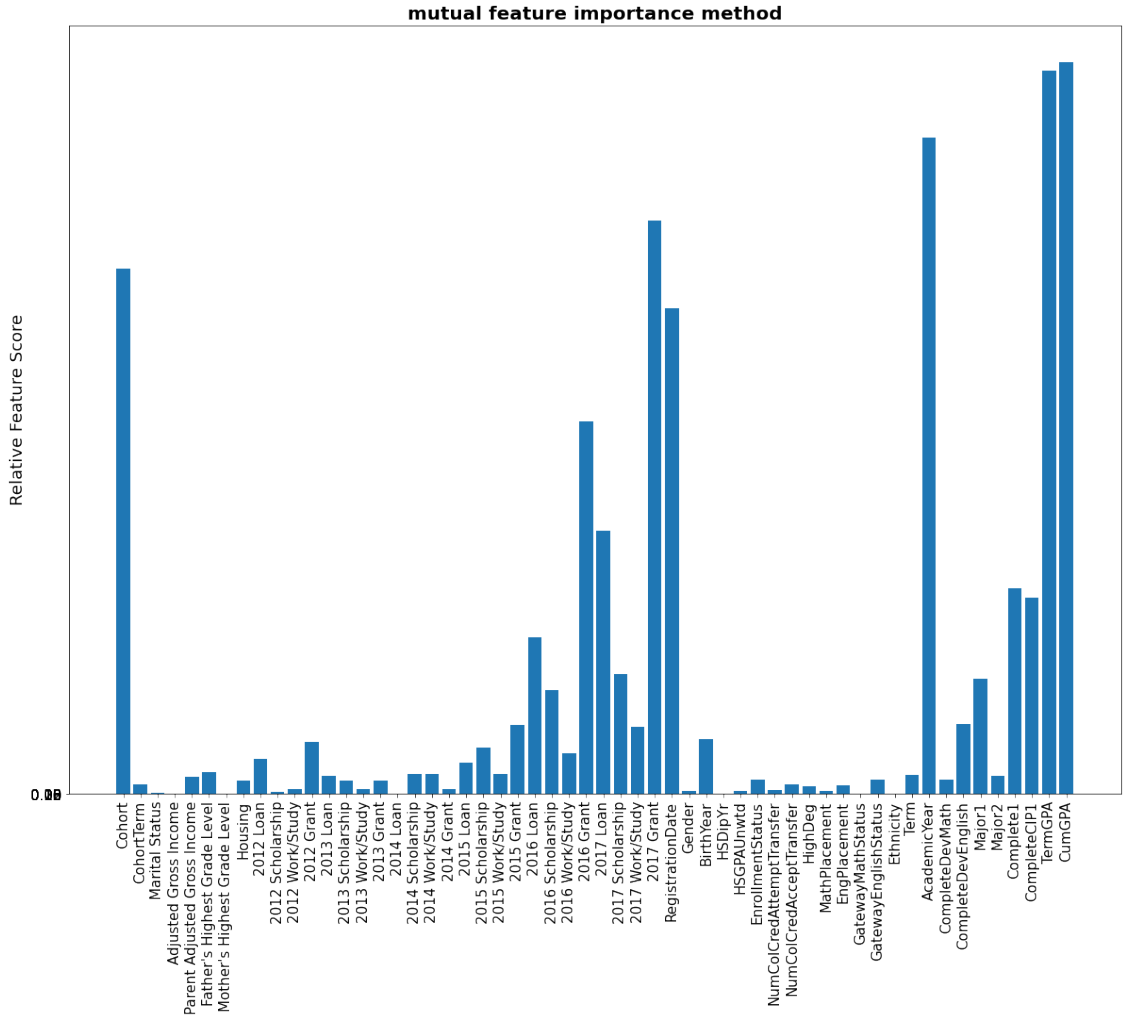
[45]:
```
X_train_fs, X_test_fs, fs = ANOVA_select_features(X_train, Y_train, X_test)
plt.figure(figsize=(20,15))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
ax = plt.gca()
ax.set_ylabel('Relative Feature Score', color='black', fontsize=18)
ax.set_title('ANOVA feature importance method', color='black', fontsize=20,␣
 ↪fontweight='bold')
ax.set_xticks(np.arange(0, len(X.columns), 1.0))
ax.set_yticks(np.arange(0, 3600, 500))
ax.set_yticklabels(np.arange(0, 3600, 500), color='black', fontsize=15)
ax.set_xticklabels(list(X.columns.values), rotation='vertical', color='black',␣
 ↪fontsize=15)
plt.show()
```



Mutual information Feature Selection This feature selaction backs up the previous results from ANOVA to some extent. For example, GPA and cohort remain important factors.

[46]:
```
plt.figure(figsize=(20,15))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
ax = plt.gca()
ax.set_ylabel('Relative Feature Score', color='black', fontsize=18)
ax.set_title('mutual feature importance method', color='black', fontsize=20,␣
↪fontweight='bold')
ax.set_xticks(np.arange(0, len(X.columns), 1.0))
ax.set_yticks(np.arange(0, 0.22, 0.025))
ax.set_yticklabels(np.arange(0, 0.22, 0.025), color='black', fontsize=15)
plt.gca().yaxis.set_major_formatter(StrMethodFormatter('{x:,.2f}')) # 2 decimal␣
↪places
```

```
ax.set_xticklabels(list(X.columns.values), rotation='vertical', color='black',␣
 ↪fontsize=15)
plt.show()
```



0.3 Applying classification ML models

I tested 5 different ML classifiers and evaluated the model performance using the test data.

1) Random forest Classifier

[47]:
```
# create the Random forest Classifier
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
y_pred_rf = rf.predict(X_test)
probs = pd.DataFrame(rf.predict_proba(X_test))
```

```
# print verification metrics data
print('Accuracy\n')
print(accuracy_score(Y_test, y_pred_rf))
print('Confusion Matrix\n')
print(metrics.confusion_matrix(Y_test, y_pred_rf))
print(classification_report(Y_test, y_pred_rf))

# Store metrics
rf_accuracy = metrics.accuracy_score(Y_test, y_pred_rf)
rf_roc_auc = metrics.roc_auc_score(Y_test, probs[1])
rf_fpr, rf_tpr, _ = metrics.roc_curve(Y_test, probs[1])
rf_confus_matrix = metrics.confusion_matrix(Y_test, y_pred_rf)
rf_classification_report = metrics.classification_report(Y_test, y_pred_rf)
rf_precision = metrics.precision_score(Y_test, y_pred_rf, pos_label=1)
rf_recall = metrics.recall_score(Y_test, y_pred_rf, pos_label=1)
rf_f1 = metrics.f1_score(Y_test, y_pred_rf, pos_label=1)

# Evaluate the model using 10-fold cross-validation
rf_cv_scores = cross_val_score(rf, X_test, Y_test, scoring='precision', cv=10)
rf_cv_mean = np.mean(rf_cv_scores)
```

```
Accuracy

0.9756834315058148
Confusion Matrix

[[9202  101]
 [ 221 3718]]
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      9303
           1       0.97      0.94      0.96      3939

    accuracy                           0.98     13242
   macro avg       0.98      0.97      0.97     13242
weighted avg       0.98      0.98      0.98     13242
```

2) Logestic Regression

[48]:
```
# create Logestic Regression classifier. I had to increase the max iteration to␣
 ↪make it converge
logr = LogisticRegression(solver='lbfgs',max_iter=100000)
logr.fit(X_train,Y_train)
y_pred_logr = logr.predict(X_test)
probs = pd.DataFrame(logr.predict_proba(X_test))

# print verification metrics data
```

```
print('Accuracy\n')
print(accuracy_score(Y_test, y_pred_logr))
print('Confusion Matrix\n')
print(metrics.confusion_matrix(Y_test, y_pred_logr))
print(classification_report(Y_test, y_pred_logr))

# Store metrics
logit_accuracy = metrics.accuracy_score(Y_test, y_pred_logr)
logit_roc_auc = metrics.roc_auc_score(Y_test, probs[1])
logit_fpr, logit_tpr, _ = metrics.roc_curve(Y_test, probs[1])
logit_confus_matrix = metrics.confusion_matrix(Y_test, y_pred_logr)
logit_classification_report = metrics.classification_report(Y_test, y_pred_logr)
logit_precision = metrics.precision_score(Y_test, y_pred_logr, pos_label=1)
logit_recall = metrics.recall_score(Y_test, y_pred_logr, pos_label=1)
logit_f1 = metrics.f1_score(Y_test, y_pred_logr, pos_label=1)

# Evaluate the model using 10-fold cross-validation
logit_cv_scores = cross_val_score(logr, X_test, Y_test, scoring='precision',␣
↪cv=10)
logit_cv_mean = np.mean(logit_cv_scores)
```

```
Accuracy

0.7161304938830992
Confusion Matrix

[[8928  375]
 [3384  555]]
              precision    recall  f1-score   support

           0       0.73      0.96      0.83      9303
           1       0.60      0.14      0.23      3939

    accuracy                           0.72     13242
   macro avg       0.66      0.55      0.53     13242
weighted avg       0.69      0.72      0.65     13242
```

3) Decision Tree

[49]:
```
# Instantiate with a max depth of 3
tree_model = tree.DecisionTreeClassifier(max_depth=40)
# Fit a decision tree
tree_model = tree_model.fit(X_train, Y_train)
y_pred_tree = tree_model.predict(X_test)
probs = pd.DataFrame(tree_model.predict_proba(X_test))

# print verification metrics data
```

```
print('Accuracy\n')
print(accuracy_score(Y_test, y_pred_tree))
print('Confusion Matrix\n')
print(metrics.confusion_matrix(Y_test, y_pred_tree))
print(classification_report(Y_test, y_pred_tree))

# Store metrics
tree_accuracy = metrics.accuracy_score(Y_test, y_pred_tree)
tree_roc_auc = metrics.roc_auc_score(Y_test, probs[1])
tree_fpr, tree_tpr, _ = metrics.roc_curve(Y_test, probs[1])
tree_confus_matrix = metrics.confusion_matrix(Y_test, y_pred_tree)
tree_classification_report = metrics.classification_report(Y_test, y_pred_tree)
tree_precision = metrics.precision_score(Y_test, y_pred_tree, pos_label=1)
tree_recall = metrics.recall_score(Y_test, y_pred_tree, pos_label=1)
tree_f1 = metrics.f1_score(Y_test, y_pred_tree, pos_label=1)

# Evaluate the model using 10-fold cross-validation
tree_cv_scores = cross_val_score(tree_model, X_test, Y_test,␣
 ↪scoring='precision', cv=10)
tree_cv_mean = np.mean(tree_cv_scores)
```

```
Accuracy

0.9386799577103156
Confusion Matrix

[[8902  401]
 [ 411 3528]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      9303
           1       0.90      0.90      0.90      3939

    accuracy                           0.94     13242
   macro avg       0.93      0.93      0.93     13242
weighted avg       0.94      0.94      0.94     13242
```

4) Naive Bayes classifier

[50]:
```
# create Naive Bayes classifier
bayes_model = GaussianNB()
# Fit the model
bayes_model.fit(X_train, Y_train)
y_pred_bayes = bayes_model.predict(X_test)
probs = pd.DataFrame(bayes_model.predict_proba(X_test))

# print verification metrics data
```

```
print('Accuracy\n')
print(accuracy_score(Y_test, y_pred_bayes))
print('Confusion Matrix\n')
print(metrics.confusion_matrix(Y_test, y_pred_bayes))
print(classification_report(Y_test, y_pred_bayes))

# Store metrics
bayes_accuracy = metrics.accuracy_score(Y_test, y_pred_bayes)
bayes_roc_auc = metrics.roc_auc_score(Y_test, probs[1])
bayes_fpr, bayes_tpr, _ = metrics.roc_curve(Y_test, probs[1])
bayes_confus_matrix = metrics.confusion_matrix(Y_test, y_pred_bayes)
bayes_classification_report = metrics.classification_report(Y_test,␣
 ↪y_pred_bayes)
bayes_precision = metrics.precision_score(Y_test, y_pred_bayes, pos_label=1)
bayes_recall = metrics.recall_score(Y_test, y_pred_bayes, pos_label=1)
bayes_f1 = metrics.f1_score(Y_test, y_pred_bayes, pos_label=1)

# Evaluate the model using 10-fold cross-validation
bayes_cv_scores = cross_val_score(bayes_model, X_test, Y_test,␣
 ↪scoring='precision', cv=10)
bayes_cv_mean = np.mean(bayes_cv_scores)
```

```
Accuracy

0.5736293611236973
Confusion Matrix

[[4039 5264]
 [ 382 3557]]
              precision    recall  f1-score   support

           0       0.91      0.43      0.59      9303
           1       0.40      0.90      0.56      3939

    accuracy                           0.57     13242
   macro avg       0.66      0.67      0.57     13242
weighted avg       0.76      0.57      0.58     13242
```

5) Kth Nearest Neighbor

[51]:
```
# instantiate learning model (k = 3)
knn_model = KNeighborsClassifier(n_neighbors=3)
# fit the model
knn_model.fit(X_train, Y_train)
y_pred_KNN = knn_model.predict(X_test)
probs = pd.DataFrame(knn_model.predict_proba(X_test))
```

```
# print verification metrics data
print('Accuracy\n')
print(accuracy_score(Y_test, y_pred_KNN))
print('Confusion Matrix\n')
print(metrics.confusion_matrix(Y_test, y_pred_KNN))
print(classification_report(Y_test, y_pred_KNN))

# Store metrics
knn_accuracy = metrics.accuracy_score(Y_test, y_pred_KNN)
knn_roc_auc = metrics.roc_auc_score(Y_test, probs[1])
knn_fpr, knn_tpr, _ = metrics.roc_curve(Y_test, probs[1])
knn_confus_matrix = metrics.confusion_matrix(Y_test, y_pred_KNN)
knn_classification_report = metrics.classification_report(Y_test, y_pred_KNN)
knn_precision = metrics.precision_score(Y_test, y_pred_KNN, pos_label=1)
knn_recall = metrics.recall_score(Y_test, y_pred_KNN, pos_label=1)
knn_f1 = metrics.f1_score(Y_test, y_pred_KNN, pos_label=1)

# Evaluate the model using 10-fold cross-validation
knn_cv_scores = cross_val_score(knn_model, X_test, Y_test, scoring='precision',␣
↪cv=10)
knn_cv_mean = np.mean(knn_cv_scores)
```

```
Accuracy

0.9658661833559885
Confusion Matrix

[[9160  143]
 [ 309 3630]]
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      9303
           1       0.96      0.92      0.94      3939

    accuracy                           0.97     13242
   macro avg       0.96      0.95      0.96     13242
weighted avg       0.97      0.97      0.97     13242
```

0.3.1 Evaluate Model Performance

I evaluated the performance metrics as shown below and it is evident that the Random Forest Classifier does a decent job compared to the other methods

[52]:
```
# Model comparison
models = pd.DataFrame({
```

```
    'Model'    : ['Logistic Regression', 'Decision Tree', 'Random Forest',␣
↪'kNN', 'Bayes'],
    'Accuracy' : [logit_accuracy, tree_accuracy, rf_accuracy, knn_accuracy,␣
↪bayes_accuracy],
    'Precision': [logit_precision, tree_precision, rf_precision, knn_precision,␣
↪bayes_precision],
    'recall'   : [logit_recall, tree_recall, rf_recall, knn_recall,␣
↪bayes_recall],
    'F1'       : [logit_f1, tree_f1, rf_f1, knn_f1, bayes_f1],
    'cv_precision' : [logit_cv_mean, tree_cv_mean, rf_cv_mean, knn_cv_mean,␣
↪bayes_cv_mean]
    })
models.sort_values(by='Precision', ascending=False, ignore_index=True)
```

[52]:

	Model	Accuracy	Precision	recall	F1	cv_precision
0	Random Forest	0.975683	0.973553	0.943894	0.958494	0.924456
1	kNN	0.965866	0.962099	0.921554	0.941390	0.728532
2	Decision Tree	0.938680	0.897938	0.895659	0.896797	0.752195
3	Logistic Regression	0.716130	0.596774	0.140899	0.227973	0.598625
4	Bayes	0.573629	0.403242	0.903021	0.557524	0.401779

ROC-AUC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, Higher the AUC, better the model is at distinguishing between patients with disease and no disease.

The results again confirm the superiority of the Random Forest model because ot has the largest AUC values. AUC is essentially the area under the ROC curve.

[56]:
```
# create the dataframe for plotting the ROC curves
AUCs = pd.DataFrame({
    'Model'    : ['Logistic Regression', 'Decision Tree', 'Random Forest',␣
↪'kNN', 'Bayes'],
    'FPR' : [logit_fpr, tree_fpr, rf_fpr, knn_fpr, bayes_fpr],
    'TPR': [logit_tpr, tree_tpr, rf_tpr, knn_tpr, bayes_tpr],
    'AUC'   : [logit_roc_auc, tree_roc_auc, rf_roc_auc, knn_roc_auc,␣
↪bayes_roc_auc],
    })

plt.figure(figsize=(14,10))
for i in AUCs.index:
    plt.plot(AUCs.loc[i]['FPR'],
             AUCs.loc[i]['TPR'],
             label="{}, AUC={:.3f}".format(AUCs.loc[i]['Model'], AUCs.
↪loc[i]['AUC']))
```

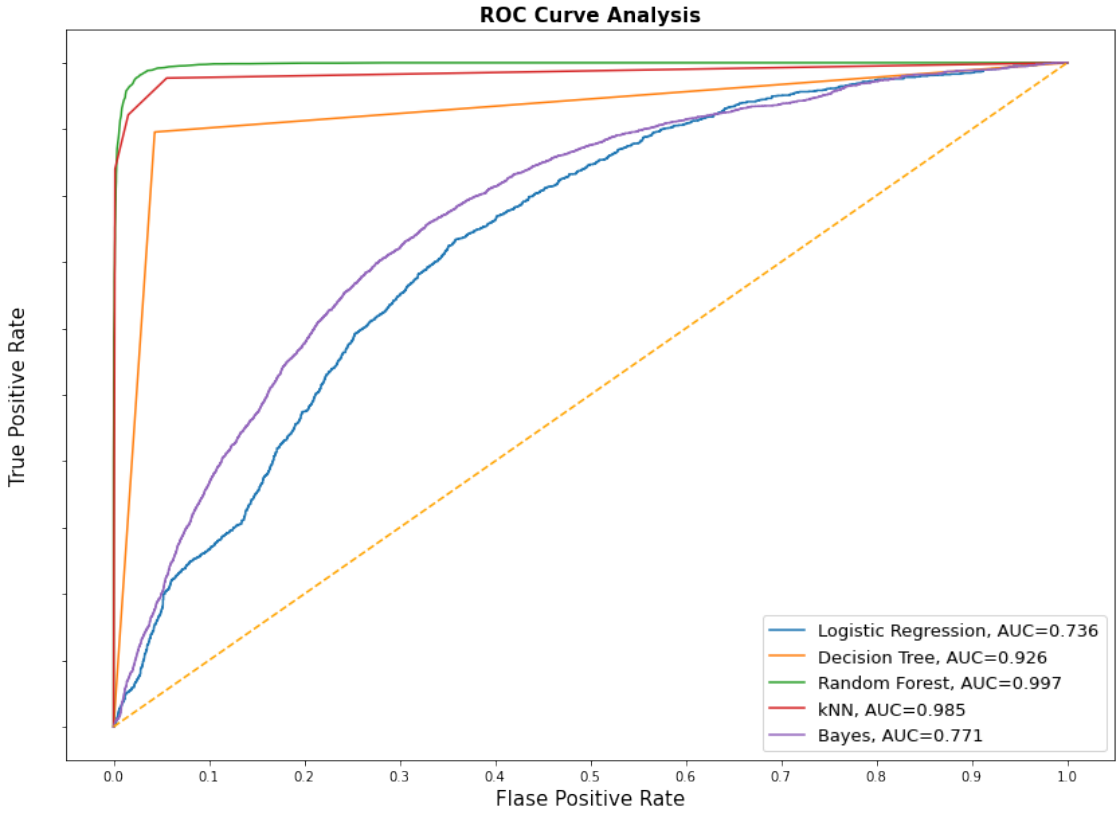
```
plt.plot([0,1], [0,1], color='orange', linestyle='--')

plt.xticks(np.arange(0.0, 1.1, step=0.1),  color='black')
plt.xlabel("Flase Positive Rate", fontsize=15, color='black')

plt.yticks(np.arange(0.0, 1.1, step=0.1), color='white')
plt.ylabel("True Positive Rate", fontsize=15, color='black')

plt.title('ROC Curve Analysis', fontweight='bold', fontsize=15, color='black')
plt.legend(prop={'size':13}, loc='lower right')

plt.show()
```



0.3.2 Validation against Promazo test data

Promazo will provide a list of student IDs in the following file:

'Student Dropout Prediction Data/Test Data/test_student_IDs.csv'

[54]:
```
# read in the test student IDs from the user
testStudentID = pd.read_csv('Student Dropout Prediction Data/Test Data/
 ↪test_student_IDs.csv')
```

```
# extract the input parametrs associated with these students from the cleaned␣
 ↪static, progress and
# financial data
merged_test_data = pd.merge(static_progress_financialData, testStudentID,␣
 ↪on=["StudentID"])
# do lavel encoding for categorical data
labelEncoded_mergedData = label_encoding(merged_test_data)
# drop 'StudentID', 'BirthMonth' columns
input_data = labelEncoded_mergedData.drop({'StudentID', 'BirthMonth'}, axis = 1)
# run the random forest classifier to predict if student will drop out or not
Dropout_prediction = rf.predict(input_data)
```

[55]:
```
# save the output of the model in the required format
output = pd.DataFrame({'StudentID':[],'Dropout':[]})
output['StudentID'] = labelEncoded_mergedData['StudentID']
output['Dropout'] = Dropout_prediction
output = output.drop_duplicates(subset=['StudentID'])
savePath = 'Student Dropout Prediction Data/Test Data/FinalOutcome.csv'
output.to_csv(savePath, index = False)
```