Contents

1 Business Problem:

1.1 Background

Every year, the federal government releases large amounts of data on U.S. schools, schoo multiple datasets that are often difficult to access, and changes in data structure complica (https://educationdata.urban.org/documentation/index.html (https://educationdata.urban.(this data to make it easier to combine data from different reporting sources.

Using the Urban institutes consolidated data platform we will be combining datasets desc describing characteristics at the school district level.

We will use this data to build a classification model which classifies high schools as either low and high graduation rates is based on the federal government's standard that those s graduation rate schools.

1.2 Limitations

- Year: For our examination we only used data from 2015 as this year was the most dat
- Our data is limited to only those schools which reported their graduation rates

1.3 Problem Statement

Predict which schools have high and which schools have low high school graduation rate and low graduation rates so that school districts know where to focus resources when att

2 Import Libraries

Contents

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtic
import seaborn as sns
import numpy as np
import scipy.stats as stats
import statsmodels.api as sm
import catboost
import time
import warnings
warnings.filterwarnings('ignore')

from sklearn.utils import class_weight
from sklearn.metrics import accuracy_score, confusion_matrix, class
from catboost import Pool, sum_models
from catboost import CatBoostClassifier
from statsmodels.formula.api import ols
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absc
from sklearn.model_selection import KFold, cross_val_score, Stratif
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder,  OneHotEncoder, Mir
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, RandomForestCla
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer, MissingIndicator
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import FeatureUnion
from sklearn.pipeline import Pipeline
from scipy.stats import uniform, truncnorm, randint
```

executed in 2.99s, finished 14:34:07 2021-06-17

3 Data Exploration and Prep

Contents

Our school level data contains information on the school location, degree of urbanization, assessments, number of allegations for harassment/bullying, number of students enrolled number of students participation in ACT/SAT tests.

Our district level data contains financial information for each school district. This includes of the fiscal year, district expenditures, and revenue. District data also includes number of instruction.

3.1 Data Load

In [2]:
```
#Northeastern United States school and district data
df_school_northeast = pd.read_csv('data/EducationDataPortal_schools
df_district_northeast = pd.read_csv('data/EducationDataPortal_distr
```
executed in 125ms, finished 14:34:07 2021-06-17

In [3]:
```
#Western United States school and district data
df_school_west = pd.read_csv('data/EducationDataPortal_schools_west
df_district_west = pd.read_csv('data/EducationDataPortal_districts_
```
executed in 123ms, finished 14:34:07 2021-06-17

In [4]:
```
#Midwest United States school and district data
df_school_midwest = pd.read_csv('data/EducationDataPortal_schools_m
df_district_midwest = pd.read_csv('data/EducationDataPortal_distric
```
executed in 183ms, finished 14:34:08 2021-06-17

In [5]:
```
#Southern Atlantic United States school and district data
df_school_south_atl = pd.read_csv('data/EducationDataPortal_schools
df_district_south_atl = pd.read_csv('data/EducationDataPortal_distr
```
executed in 61ms, finished 14:34:08 2021-06-17

In [6]:
```
#Southern Central United States school and district data
df_school_south_central = pd.read_csv('data/EducationDataPortal_sch
df_district_south_central = pd.read_csv('data/EducationDataPortal_c
```
executed in 87ms, finished 14:34:08 2021-06-17

Contents

In [7]:
```
dataframes_school  = ['df_school_northeast','df_school_west', 'df_s
                      'df_school_south_atl', 'df_school_south']
```
executed in 3ms, finished 14:34:08 2021-06-17

In [8]:
```
dataframes_district  = ['df_district_northeast','df_district_west',
                        'df_district_south_atl', 'df_district_south
```
executed in 2ms, finished 14:34:08 2021-06-17

3.2 Functions

In [9]:
```
# Returns the percentage of values in a dataframe that are nulls
def percent_null_df (df,str_name):
    count_null = df.isnull().sum().sum()
    count_non_null = df.count().sum()
    return 'percent of nulls in '+ str_name +' is '+str(int((count_
```
executed in 3ms, finished 14:34:08 2021-06-17

3.3 df_school

In [10]:
```
#Concatenate school info for all three states into one dataframe
df_school = pd.concat([df_school_northeast, df_school_south_atl,
                       df_school_midwest, df_school_west, df_school
```
executed in 28ms, finished 14:34:08 2021-06-17

Contents

In [11]:
```
df_school['state_name'].value_counts(normalize=True, dropna=False)
```

executed in 19ms, finished 14:34:08 2021-06-17

```
South Carolina          0.012596
Oregon                  0.012206
Louisiana               0.011815
Maryland                0.010741
New Mexico              0.009423
Idaho                   0.008935
South Dakota            0.008886
Utah                    0.008691
Montana                 0.008349
North Dakota            0.008154
West Virginia           0.005761
Nevada                  0.005517
Maine                   0.005371
New Hampshire           0.004589
Wyoming                 0.004101
Vermont                 0.002929
Rhode Island            0.002734
Delaware                0.001855
District of Columbia    0.001660
Name: state_name, dtype: float64
```

In [12]:
```
df_school['grad_rate_midpt'].value_counts(normalize=True, dropna=Fa
```

executed in 8ms, finished 14:34:08 2021-06-17

Out[12]:
```
97     0.106288
NaN    0.104726
92     0.091593
95     0.082219
90     0.061664
         ...
35     0.000098
23     0.000049
26     0.000049
16     0.000049
20     0.000049
Name: grad_rate_midpt, Length: 99, dtype: float64
```

Contents

In [13]:
```
df_school
```

executed in 38ms, finished 14:34:08 2021-06-17

Out[13]:

	year	ncessch	school_name	state_name	lea_name	zip_location	urban_cer
0	2015	90000201136	A. I. Prince Technical High School	Connecticut	Connecticut Technical High Sc	6106	C
1	2015	90000201137	Bullard-Havens Technical High School	Connecticut	Connecticut Technical High Sc	6610	C
2	2015	90000201138	E. C. Goodwin Technical High School	Connecticut	Connecticut Technical High Sc	6053	S
3	2015	90000201139	Ella T. Grasso Southeastern Technical High School	Connecticut	Connecticut Technical High Sc	6340	
4	2015	90000201140	Eli Whitney Technical	Connecticut	Connecticut Technical	6514	S

3.3.1 Data Fields

Contents

In [14]:
```
df_school.head()
```

executed in 20ms, finished 14:34:08 2021-06-17

0	2015	90000201136	A. I. Prince Technical High School	Connecticut	Connecticut Technical High Sc	6106	City, r
1	2015	90000201137	Bullard-Havens Technical High School	Connecticut	Connecticut Technical High Sc	6610	City, r
2	2015	90000201138	E. C. Goodwin Technical High School	Connecticut	Connecticut Technical High Sc	6053	Suburb
3	2015	90000201139	Ella T. Grasso Southeastern Technical High School	Connecticut	Connecticut Technical High Sc	6340	Rural
4	2015	90000201140	Eli Whitney Technical High School	Connecticut	Connecticut Technical High Sc	6514	Suburb

5 rows × 64 columns

Contents

In [15]:
```
df_school.info()
```

executed in 52ms, finished 14:34:08 2021-06-17

```
 28  transfers_alt_sch_disc            18961 non-null  float64
 29  days_suspended                    18549 non-null  float64
 30  suspensions_instances_preschool   18549 non-null  float64
 31  suspensions_instances             18549 non-null  float64
 32  corpinstances_preschool           18549 non-null  float64
 33  corpinstances                     18549 non-null  float64
 34  salaries_teachers                 18549 non-null  object
 35  cohort_num                        18337 non-null  float64
 36  grad_rate_high                    18337 non-null  object
 37  grad_rate_low                     18337 non-null  object
 38  grad_rate_midpt                   18337 non-null  object
 39  allegations_harass_sex            18961 non-null  object
 40  allegations_harass_race           18961 non-null  object
 41  allegations_harass_disability     18961 non-null  object
 42  allegations_harass_orientation    18961 non-null  object
 43  allegations_harass_religion       18961 non-null  object
 44  students_disc_harass_dis          18961 non-null  float64
 45  students_disc_harass_race         18961 non-null  float64
 46  students_disc_harass_sex          18961 non-null  float64
 47  students report harass dis        18961 non-null  float64
```

In [16]:
```
df_school.shape
```

executed in 3ms, finished 14:34:08 2021-06-17

Out[16]:
```
(20482, 64)
```

3.3.1.1 Data Field Cleanup

In [17]:
```
column_names = list(df_school.columns)
```

executed in 1ms, finished 14:34:08 2021-06-17

In [18]:
```
for c in column_names:
    df_school[c].replace('Suppressed data', np.NaN,inplace =True)
```

executed in 31ms, finished 14:34:08 2021-06-17

Contents

In [19]:
```
for c in column_names:
    df_school[c].replace('Not applicable', np.NaN,inplace =True)
```

executed in 28ms, finished 14:34:08 2021-06-17

In [20]:
```
for c in column_names:
    df_school[c].replace('Missing/not reported', np.NaN,inplace =Tr
```

executed in 29ms, finished 14:34:08 2021-06-17

3.3.2 Columns to drop

Contents

In [21]:
```
df_school.info()
```

executed in 31ms, finished 14:34:08 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20482 entries, 0 to 3966
Data columns (total 64 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   year                                  20482 non-null  int64
 1   ncessch                               20482 non-null  int64
 2   school_name                           20482 non-null  object
 3   state_name                            20482 non-null  object
 4   lea_name                              20482 non-null  object
 5   zip_location                          19473 non-null  object
 6   urban_centric_locale                  20457 non-null  object
 7   school_level                          20482 non-null  object
 8   school_type                           20482 non-null  object
 9   charter                               19368 non-null  object
 10  enrollment                            20001 non-null  object
 11  read_test_num_valid                   18073 non-null  float64
 12  read_test_pct_prof_low                17434 non-null  object
 13  read_test_pct_prof_high               17434 non-null  object
 14  read_test_pct_prof_midpt              17434 non-null  object
 15  math_test_num_valid                   18132 non-null  float64
 16  math_test_pct_prof_low                17432 non-null  object
 17  math_test_pct_prof_high               17432 non-null  object
 18  math_test_pct_prof_midpt              17432 non-null  object
 19  students_susp_in_sch                  18732 non-null  object
 20  students_susp_out_sch_single          18898 non-null  object
 21  students_susp_out_sch_multiple        18898 non-null  object
 22  expulsions_no_ed_serv                 18898 non-null  object
 23  expulsions_with_ed_serv               18732 non-null  object
 24  expulsions_zero_tolerance             18898 non-null  object
 25  students_corporal_punish              954 non-null    object
 26  students_arrested                     18305 non-null  object
 27  students_referred_law_enforce         18728 non-null  object
 28  transfers_alt_sch_disc                18961 non-null  float64
 29  days_suspended                        18549 non-null  float64
 30  suspensions_instances_preschool       18549 non-null  float64
 31  suspensions_instances                 18549 non-null  float64
 32  corpinstances_preschool               18549 non-null  float64
 33  corpinstances                         18549 non-null  float64
 34  salaries_teachers                     18049 non-null  object
```

Contents


```
 35  cohort_num                          18337 non-null  float64
 36  grad_rate_high                      17812 non-null  object
 37  grad_rate_low                       17812 non-null  object
 38  grad_rate_midpt                     17812 non-null  object
 39  allegations_harass_sex              18836 non-null  object
 40  allegations_harass_race             18836 non-null  object
 41  allegations_harass_disability       18836 non-null  object
 42  allegations_harass_orientation      18817 non-null  object
 43  allegations_harass_religion         18817 non-null  object
 44  students_disc_harass_dis            18961 non-null  float64
 45  students_disc_harass_race           18961 non-null  float64
 46  students_disc_harass_sex            18961 non-null  float64
 47  students_report_harass_dis          18961 non-null  float64
 48  students_report_harass_race         18961 non-null  float64
 49  students_report_harass_sex          18961 non-null  float64
 50  enrl_biology                        17067 non-null  float64
 51  enrl_chemistry                      15532 non-null  float64
 52  enrl_advanced_math                  14161 non-null  float64
 53  enrl_calculus                       11670 non-null  float64
 54  enrl_algebra2                       16321 non-null  float64
 55  enrl_physics                        13303 non-null  float64
 56  enrl_geometry                       16722 non-null  float64
 57  instances_mech_restraint            18424 non-null  float64
 58  instances_phys_restraint            18827 non-null  float64
 59  instances_seclusion                 18827 non-null  float64
 60  students_mech_restraint             18429 non-null  float64
 61  students_phys_restraint             18813 non-null  float64
 62  students_seclusion                  18828 non-null  float64
 63  students_SAT_ACT                    18961 non-null  float64
dtypes: float64(29), int64(2), object(33)
memory usage: 10.2+ MB
```

In [22]:
```
df_school['corpinstances_preschool'].value_counts(normalize=True, d
```

executed in 7ms, finished 14:34:08 2021-06-17

Out[22]:
```
-1.0    0.904355
 NaN    0.094376
 0.0    0.001221
 3.0    0.000049
Name: corpinstances_preschool, dtype: float64
```

In [23]:
```
df_school['students_corporal_punish'].value_counts(normalize=True,
```
executed in 7ms, finished 14:34:08 2021-06-17

Out[23]:
```
NaN     0.953423
0       0.007714
2       0.005908
4       0.003955
6       0.002246
          ...
78      0.000049
171     0.000049
1653    0.000049
111     0.000049
120     0.000049
Name: students_corporal_punish, Length: 116, dtype: float64
```

- Drop year as this is the same for all rows
- Drop corpinstances_preschool this column is mostly -1 which is not interpretable

In [24]:
```
df_school = df_school.drop(['year','corpinstances_preschool','stude
```
executed in 11ms, finished 14:34:08 2021-06-17

3.3.3 school_type

In [25]:
```
df_school['school_type'].value_counts(normalize=True, dropna=False)
```
executed in 7ms, finished 14:34:08 2021-06-17

Out[25]:
```
Regular school              0.798994
Other/alternative school    0.136559
Vocational school           0.050337
Special education school    0.014110
Name: school_type, dtype: float64
```

In [26]:
```
df_school_regular = df_school.loc[df_school['school_type'] == 'Regu
df_school_regular
```

executed in 39ms, finished 14:34:08 2021-06-17

38	90001800834	Explorations	Connecticut	EXPLORATIONS DISTRICT	Town, fringe
...	...	...	...	...	...
3960	484668005347	YSLETA H S	Texas	YSLETA ISD	City, large
3962	484668006496	DEL VALLE H S	Texas	YSLETA ISD	City, large
3964	484668011982	VALLE VERDE EARLY COLLEGE H S	Texas	YSLETA ISD	City, large
3965	484671005351	ZAPATA H S	Texas	ZAPATA COUNTY ISD	Rural, fringe
3966	489913005627	BENAVIDES SECONDARY	Texas	BENAVIDES ISD	Rural, remote

16365 rows × 60 columns

Contents

Contents

In [27]:
```
df_school_alternative = df_school.loc[df_school['school_type'] == '
df_school_alternative
```

executed in 28ms, finished 14:34:08 2021-06-17

Out[27]:

	ncessch	school_name	state_name	lea_name	urban_centric_locale	school_l
16	90000300343	Bridgeport Correctional Center	Connecticut	UNIFIED SCHOOL DISTRICT #1	City, midsize	
17	90000300344	Brooklyn Correctional Institution	Connecticut	UNIFIED SCHOOL DISTRICT #1	Rural, fringe	
18	90000300347	Cheshire Correctional Institution	Connecticut	UNIFIED SCHOOL DISTRICT #1	Suburb, large	
19	90000300361	Hartford Correctional Center	Connecticut	UNIFIED SCHOOL DISTRICT #1	City, midsize	
20	90000300374	New Haven Correctional Center	Connecticut	UNIFIED SCHOOL DISTRICT #1	City, midsize	
...	...	...	...	...	...	
3932	484578000736	HARRELL ACCELERATED LEARNING CENTER	Texas	WICHITA FALLS ISD	City, midsize	
3936	484578009422	WICHITA COUNTY JUVENILE JUSTICE AEP	Texas	WICHITA FALLS ISD	City, midsize	
3954	484668003897	CESAR CHAVEZ ACADEMY	Texas	YSLETA ISD	City, large	
3961	484668005938	TEJAS SCHOOL OF CHOICE	Texas	YSLETA ISD	City, large	

	ncessch	school_name	state_name	lea_name	urban_centric_locale	school_l
3963	484668008548	PLATO ACADEMY	Texas	YSLETA ISD	City, large	

2797 rows × 60 columns

Contents

Contents

In [28]:
```
df_school_vocational = df_school.loc[df_school['school_type'] == 'V
df_school_vocational.head(10)
```

executed in 76ms, finished 14:34:08 2021-06-17

Out[28]:

	ncessch	school_name	state_name	lea_name	urban_centric_locale	school_level
0	90000201136	A. I. Prince Technical High School	Connecticut	Connecticut Technical High Sc	City, midsize	High
1	90000201137	Bullard-Havens Technical High School	Connecticut	Connecticut Technical High Sc	City, midsize	High
2	90000201138	E. C. Goodwin Technical High School	Connecticut	Connecticut Technical High Sc	Suburb, large	High
3	90000201139	Ella T. Grasso Southeastern Technical High School	Connecticut	Connecticut Technical High Sc	Rural, fringe	High
4	90000201140	Eli Whitney Technical High School	Connecticut	Connecticut Technical High Sc	Suburb, large	High
5	90000201141	Emmett OBrien Technical High School	Connecticut	Connecticut Technical High Sc	Suburb, large	High
6	90000201142	H. C. Wilcox Technical High School	Connecticut	Connecticut Technical High Sc	Suburb, large	High
7	90000201143	H. H. Ellis Technical High School	Connecticut	Connecticut Technical High Sc	Suburb, large	High
8	90000201144	Henry Abbott Technical High School	Connecticut	Connecticut Technical High Sc	City, small	High
9	90000201145	Howell Cheney Technical High School	Connecticut	Connecticut Technical High Sc	Suburb, large	High

10 rows × 60 columns

Contents

In [29]:
```
df_school_special = df_school.loc[df_school['school_type'] == 'Spec
df_school_special.head(10)
```

executed in 25ms, finished 14:34:08 2021-06-17

Out[29]:

	ncessch	school_name	state_name	lea_name	urban_centric_locale
46	90007001588	Center for Autism Spectrum and Development Dis...	Connecticut	AREA COOPERATIVE EDUCATIONAL	Suburb, large
64	90051001515	Community/Vocational Program	Connecticut	BRISTOL SCHOOL DISTRICT	Suburb, large
71	90070001567	Lincoln Academy Regional Interdistrict Collabo...	Connecticut	CAPITOL REGION EDUCATION COUN	City, small
73	90070001693	Soundbridge @ Wethersfield High School	Connecticut	CAPITOL REGION EDUCATION COUN	Suburb, large
74	90070001802	STRIVE (Southern Transition Real-World and Ind...	Connecticut	CAPITOL REGION EDUCATION COUN	Suburb, large
77	90075001766	Cheshire Quinnipiac University Transition Coll...	Connecticut	CHESHIRE SCHOOL DISTRICT	Suburb, large
92	90123001514	The Learning Center at East Hampton	Connecticut	EAST HAMPTON SCHOOL DISTRICT	Town, fringe
98	90132001730	Post High School Transition Program	Connecticut	EAST LYME SCHOOL DISTRICT	Suburb, midsize
100	90132001843	Medically Fragile Program (WAVES)	Connecticut	EAST LYME SCHOOL DISTRICT	NaN
108	90147001523	Enfield Transitional Learning Academy	Connecticut	ENFIELD SCHOOL DISTRICT	Suburb, large

10 rows × 60 columns

In [30]:
```
#Proportion of nulls regular school type
percent_null_df(df_school_regular,'df_school_regular')
```

executed in 39ms, finished 14:34:08 2021-06-17

Out[30]: 'percent of nulls in df_school_regular is 4%'

In [31]:
```
#Proportion of nulls vocational school type
percent_null_df(df_school_vocational,'df_school_vocational')
```

executed in 8ms, finished 14:34:08 2021-06-17

Out[31]: 'percent of nulls in df_school_vocational is 61%'

In [32]:
```
#Proportion of nulls alternative school type
percent_null_df(df_school_alternative,'df_school_alternative')
```

executed in 12ms, finished 14:34:08 2021-06-17

Out[32]: 'percent of nulls in df_school_alternative is 24%'

In [33]:
```
#Proportion of nulls special education school type
percent_null_df(df_school_special,'df_school_special')
```

executed in 6ms, finished 14:34:08 2021-06-17

Out[33]: 'percent of nulls in df_school_special is 38%'

Our Other/alternative school in the school_type field includes jails and detention centers. I other data fields. We will remove all rows with alternative schools since there seems to be students in these types of schools are experiencing circumstances and educational exper

We will also remove vocational and special education school types as these school types nulls

In [34]:
```
df_school.shape[0]
```

executed in 3ms, finished 14:34:08 2021-06-17

Out[34]: 20482

In [35]:
```
df_school = df_school.loc[df_school['school_type'] == 'Regular scho
df_school.shape[0]
```

executed in 13ms, finished 14:34:08 2021-06-17

Out[35]: 16365

3.3.4 Graduation Rate - format the target

- The graduation rates are numerical values separated into three columns.
  - grad_rate_high
  - grad_rate_midpt
  - grad_rate_low
- Looking at the data documentation from EDFacts https://www2.ed.gov/about/inits/ed (https://www2.ed.gov/about/inits/ed/edfacts/index.htmlThe) we see that the reason fo graduation rates in a range so that student privacy is protected. The range creates a l middle of the low and high values.
- We will use grad_rate_midpt to measure graduation rates as this is the most balanced

- The federal government defines graduation rates as low when less than 2/3 of a coho
- We will classify high schools with midpoint graduation rates 66 and below as low and high.

In [36]:
```
#Let's see a few examples of the grad rate reporting range before w
grad_rate_cols = ['school_name','cohort_num','grad_rate_high','grad
                  'grad_rate_low']
```

executed in 2ms, finished 14:34:08 2021-06-17

Contents

In [37]:
```
df_school[grad_rate_cols].head(10)
```

executed in 9ms, finished 14:34:08 2021-06-17

Out[37]:

	school_name	cohort_num	grad_rate_high	grad_rate_midpt	grad_rate_l
32	Walter G. Cady School	NaN	NaN	NaN	N
33	E. O. Smith High School	255.0	95	95	
36	Common Ground High School	34.0	100	95	
37	The Bridge Academy	34.0	79	74	
38	Explorations	27.0	79	69	
39	Connecticut Valley Hospital	NaN	NaN	NaN	N
40	Stamford Academy	53.0	29	24	
42	Ansonia High School	125.0	89	87	
47	Avon High School	239.0	97	97	
48	Berlin High School	248.0	95	95	

In [38]:
```
df_school = df_school.drop(['grad_rate_high','grad_rate_low'], axis
```

executed in 7ms, finished 14:34:08 2021-06-17

Contents

In [39]:
```
df_school.info()
```

executed in 25ms, finished 14:34:08 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16365 entries, 32 to 3966
Data columns (total 58 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ncessch                         16365 non-null  int64
 1   school_name                     16365 non-null  object
 2   state_name                      16365 non-null  object
 3   lea_name                        16365 non-null  object
 4   urban_centric_locale            16364 non-null  object
 5   school_level                    16365 non-null  object
 6   school_type                     16365 non-null  object
 7   charter                         15404 non-null  object
 8   enrollment                      16328 non-null  object
 9   read_test_num_valid             15747 non-null  float64
 10  read_test_pct_prof_low          15500 non-null  object
 11  read_test_pct_prof_high         15500 non-null  object
 12  read_test_pct_prof_midpt        15500 non-null  object
 13  math_test_num_valid             15819 non-null  float64
```

In [40]:
```
df_school.update(df_school[['grad_rate_midpt']].fillna(0))
```

executed in 17ms, finished 14:34:08 2021-06-17

In [41]:
```
df_school['grad_rate_midpt'].value_counts(normalize=True,dropna=Fal
```

executed in 7ms, finished 14:34:09 2021-06-17

Out[41]:
```
97    0.127345
92    0.112679
95    0.100214
90    0.073816
87    0.071983
        ...
25    0.000061
38    0.000061
0     0.000061
18    0.000061
23    0.000061
Name: grad_rate_midpt, Length: 92, dtype: float64
```

Contents

In [42]:
```
df_school.shape
```
executed in 3ms, finished 14:34:09 2021-06-17

Out[42]:
```
(16365, 58)
```

In [43]:
```
19198*(1-.43)
```
executed in 4ms, finished 14:34:09 2021-06-17

Out[43]:
```
10942.86
```

In [44]:
```
df_school = df_school[df_school.grad_rate_midpt != 0]
```
executed in 12ms, finished 14:34:09 2021-06-17

In [45]:
```
df_school.shape[0]
```
executed in 3ms, finished 14:34:09 2021-06-17

Out[45]:
```
15830
```

In [46]:
```
# convert grad rate columns to numeric
df_school[['grad_rate_midpt']] = df_school[['grad_rate_midpt']].ast
```
executed in 9ms, finished 14:34:09 2021-06-17

In [47]:
```
# Let's take a look at the summary statistics for the grad_rate_mid

print(df_school['grad_rate_midpt'].describe())
```
executed in 6ms, finished 14:34:09 2021-06-17

```
count    15830.000000
mean        85.590587
std         15.379323
min          0.000000
25%         82.000000
50%         90.000000
75%         95.000000
max         99.000000
Name: grad_rate_midpt, dtype: float64
```

Contents

In [48]:
```
print('grad_rate_midpt', '\n', df_school['grad_rate_midpt'].value_
```

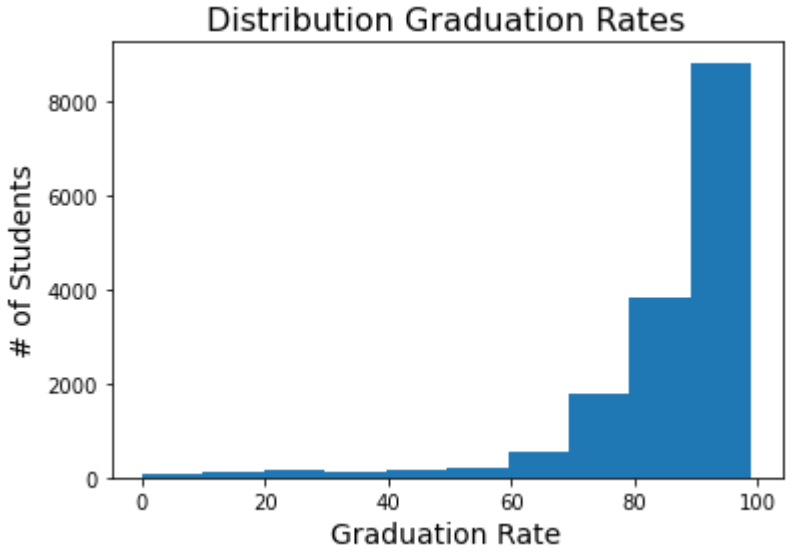
executed in 4ms, finished 14:34:09 2021-06-17

```
grad_rate_midpt
 97    0.131649
92    0.116488
95    0.103601
90    0.076311
87    0.074416
Name: grad_rate_midpt, dtype: float64
```

In [49]:
```
# Plot of the target status group

plt.hist(df_school['grad_rate_midpt'])
plt.xlabel('Graduation Rate', fontsize=14)
plt.ylabel('# of Students', fontsize=14)
plt.title("Distribution Graduation Rates", fontsize=16)
plt.show()
```

executed in 142ms, finished 14:34:09 2021-06-17



It looks like there is a strong skew towards higher performing high schools. Lets go ahead again.

3.3.4.1 Bin the target

As stated above based on how the federal government defines low graduation rates we w

- Low: 66 and below
- High: 67 and above

In [50]:
```
#Based on the federal government's definition we will define the bi
grad_rate_bins = [0,66,100]
```
executed in 3ms, finished 14:34:09 2021-06-17

In [51]:
```
# now lets convert the grad_rate_midpt column to binned categories
# note that when binning these, the default setting is that the val

df_school['grad_rate_midpt'] = pd.cut(df_school['grad_rate_midpt'],
```
executed in 7ms, finished 14:34:09 2021-06-17

In [52]:
```
# and we will also take a look at the distribution across bins
print(df_school['grad_rate_midpt'].value_counts(normalize=True, dro
```
executed in 7ms, finished 14:34:09 2021-06-17

```
(66.0, 100.0]    0.931522
(0.0, 66.0]      0.068414
NaN              0.000063
Name: grad_rate_midpt, dtype: float64
```

In [53]:
```
df_school = df_school.dropna(subset=['grad_rate_midpt'])
df_school.shape
```
executed in 11ms, finished 14:34:09 2021-06-17

Out[53]:
```
(15829, 58)
```

Contents

In [54]:
```
print(df_school['grad_rate_midpt'].value_counts(normalize=True, dro
```
executed in 5ms, finished 14:34:09 2021-06-17

```
(66, 100]    0.931581
(0, 66]      0.068419
Name: grad_rate_midpt, dtype: float64
```

In [55]:
```
# We will need to address this class imbalance
```
executed in 2ms, finished 14:34:09 2021-06-17

3.3.4.2 Encode the target

In [56]:
```
df_school['grad_rate_midpt'].value_counts(normalize=True, dropna=Fa
```
executed in 6ms, finished 14:34:09 2021-06-17

Out[56]:
```
(66, 100]    0.931581
(0, 66]      0.068419
Name: grad_rate_midpt, dtype: float64
```

In [57]:
```
df_school['grad_rate_midpt'] = df_school['grad_rate_midpt'].astype
```
executed in 51ms, finished 14:34:09 2021-06-17

In [58]:
```
df_school['grad_rate_midpt'].value_counts(normalize=True, dropna=Fa
```
executed in 7ms, finished 14:34:09 2021-06-17

Out[58]:
```
(66, 100]    0.931581
(0, 66]      0.068419
Name: grad_rate_midpt, dtype: float64
```

In [59]:
```
df_school['grad_rate_midpt'] = df_school['grad_rate_midpt'].replac
```
executed in 4ms, finished 14:34:09 2021-06-17

In [60]:
```
df_school['grad_rate_midpt'] = df_school['grad_rate_midpt'].replac
```
executed in 6ms, finished 14:34:09 2021-06-17

3.3.5 subject enrollment - drop all enrl columns

In [61]:
```
#Let's look at how many nulls we are dealing with now
percent_null_df(df_school,'df_school')
```

executed in 36ms, finished 14:34:09 2021-06-17

Out[61]:
```
'percent of nulls in df_school is 3%'
```

Contents

Contents

In [62]:
```
df_school.info()
```

executed in 25ms, finished 14:34:09 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15829 entries, 33 to 3966
Data columns (total 58 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   ncessch                               15829 non-null  int64
 1   school_name                           15829 non-null  object
 2   state_name                            15829 non-null  object
 3   lea_name                              15829 non-null  object
 4   urban_centric_locale                  15829 non-null  object
 5   school_level                          15829 non-null  object
 6   school_type                           15829 non-null  object
 7   charter                               14910 non-null  object
 8   enrollment                            15822 non-null  object
 9   read_test_num_valid                   15478 non-null  float64
 10  read_test_pct_prof_low                15305 non-null  object
 11  read_test_pct_prof_high               15305 non-null  object
 12  read_test_pct_prof_midpt              15305 non-null  object
 13  math_test_num_valid                   15552 non-null  float64
 14  math_test_pct_prof_low                15343 non-null  object
 15  math_test_pct_prof_high               15343 non-null  object
 16  math_test_pct_prof_midpt              15343 non-null  object
 17  students_susp_in_sch                  15583 non-null  object
 18  students_susp_out_sch_single          15598 non-null  object
 19  students_susp_out_sch_multiple        15598 non-null  object
 20  expulsions_no_ed_serv                 15598 non-null  object
 21  expulsions_with_ed_serv               15583 non-null  object
 22  expulsions_zero_tolerance             15598 non-null  object
 23  students_arrested                     15218 non-null  object
 24  students_referred_law_enforce         15579 non-null  object
 25  transfers_alt_sch_disc                15617 non-null  float64
 26  days_suspended                        15265 non-null  float64
 27  suspensions_instances_preschool       15265 non-null  float64
 28  suspensions_instances                 15265 non-null  float64
 29  corpinstances                         15265 non-null  float64
 30  salaries_teachers                     15107 non-null  object
 31  cohort_num                            15829 non-null  float64
 32  grad_rate_midpt                       15829 non-null  int64
 33  allegations_harass_sex                15547 non-null  object
 34  allegations_harass_race               15547 non-null  object
```

```
 35  allegations_harass_disability    15547 non-null  object
 36  allegations_harass_orientation   15533 non-null  object
 37  allegations_harass_religion      15533 non-null  object
 38  students_disc_harass_dis         15617 non-null  float64
 39  students_disc_harass_race        15617 non-null  float64
 40  students_disc_harass_sex         15617 non-null  float64
 41  students_report_harass_dis       15617 non-null  float64
 42  students_report_harass_race      15617 non-null  float64
 43  students_report_harass_sex       15617 non-null  float64
 44  enrl_biology                     14892 non-null  float64
 45  enrl_chemistry                   14262 non-null  float64
 46  enrl_advanced_math               13362 non-null  float64
 47  enrl_calculus                    11327 non-null  float64
 48  enrl_algebra2                    14580 non-null  float64
 49  enrl_physics                     12400 non-null  float64
 50  enrl_geometry                    14621 non-null  float64
 51  instances_mech_restraint         15196 non-null  float64
 52  instances_phys_restraint         15538 non-null  float64
 53  instances_seclusion              15538 non-null  float64
 54  students_mech_restraint          15197 non-null  float64
 55  students_phys_restraint          15526 non-null  float64
 56  students_seclusion               15539 non-null  float64
 57  students_SAT_ACT                 15617 non-null  float64
dtypes: float64(28), int64(2), object(28)
memory usage: 7.1+ MB
```

The class subject enrollment columns stand out as having the most null values let's drop t

In [63]:
```
df_school = df_school.drop(['enrl_biology', 'enrl_chemistry','enrl_
                            'enrl_calculus','enrl_algebra2','enrl_p
                            'enrl_geometry'], axis=1)
```

executed in 11ms, finished 14:34:09 2021-06-17

In [64]:
```
#How do our nulls look
percent_null_df(df_school,'df_school')
```

executed in 36ms, finished 14:34:09 2021-06-17

Out[64]:
```
'percent of nulls in df_school is 1%'
```

Contents

3.3.6 Encode Charter

In [65]:
```
df_school['charter'].value_counts(normalize=True, dropna=False)
```

executed in 6ms, finished 14:34:09 2021-06-17

Out[65]:
```
No     0.872386
Yes    0.069556
NaN    0.058058
Name: charter, dtype: float64
```

In [66]:
```
df_school['charter'] = df_school['charter'].replace('No', 0)
```

executed in 3ms, finished 14:34:09 2021-06-17

In [67]:
```
df_school['charter'] = df_school['charter'].replace('Yes', 1)
```

executed in 8ms, finished 14:34:09 2021-06-17

In [68]:
```
df_school['charter'].value_counts(normalize=True, dropna=False)
```

executed in 6ms, finished 14:34:09 2021-06-17

Out[68]:
```
0.0    0.872386
1.0    0.069556
NaN    0.058058
Name: charter, dtype: float64
```

We will group the small proportion of NaN values with the majority class which is zero or a

In [69]:
```
df_school.update(df_school[['charter']].fillna(0))
```

executed in 13ms, finished 14:34:09 2021-06-17

3.3.7 Reading and Math Tests

Contents

In [70]:
```
df_school.info()
```

executed in 22ms, finished 14:34:09 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15829 entries, 33 to 3966
Data columns (total 51 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   ncessch                               15829 non-null  int64
 1   school_name                           15829 non-null  object
 2   state_name                            15829 non-null  object
 3   lea_name                              15829 non-null  object
 4   urban_centric_locale                  15829 non-null  object
 5   school_level                          15829 non-null  object
 6   school_type                           15829 non-null  object
 7   charter                               15829 non-null  float64
 8   enrollment                            15822 non-null  object
 9   read_test_num_valid                   15478 non-null  float64
 10  read_test_pct_prof_low                15305 non-null  object
 11  read_test_pct_prof_high               15305 non-null  object
 12  read_test_pct_prof_midpt              15305 non-null  object
 13  math_test_num_valid                   15552 non-null  float64
 14  math_test_pct_prof_low                15343 non-null  object
 15  math_test_pct_prof_high               15343 non-null  object
 16  math_test_pct_prof_midpt              15343 non-null  object
 17  students_susp_in_sch                  15583 non-null  object
 18  students_susp_out_sch_single          15598 non-null  object
 19  students_susp_out_sch_multiple        15598 non-null  object
 20  expulsions_no_ed_serv                 15598 non-null  object
 21  expulsions_with_ed_serv               15583 non-null  object
 22  expulsions_zero_tolerance             15598 non-null  object
 23  students_arrested                     15218 non-null  object
 24  students_referred_law_enforce         15579 non-null  object
 25  transfers_alt_sch_disc                15617 non-null  float64
 26  days_suspended                        15265 non-null  float64
 27  suspensions_instances_preschool       15265 non-null  float64
 28  suspensions_instances                 15265 non-null  float64
 29  corpinstances                         15265 non-null  float64
 30  salaries_teachers                     15107 non-null  object
 31  cohort_num                            15829 non-null  float64
 32  grad_rate_midpt                       15829 non-null  int64
 33  allegations_harass_sex                15547 non-null  object
 34  allegations_harass_race               15547 non-null  object
```

Contents


```
 35  allegations_harass_disability    15547 non-null  object
 36  allegations_harass_orientation   15533 non-null  object
 37  allegations_harass_religion      15533 non-null  object
 38  students_disc_harass_dis         15617 non-null  float64
 39  students_disc_harass_race        15617 non-null  float64
 40  students_disc_harass_sex         15617 non-null  float64
 41  students_report_harass_dis       15617 non-null  float64
 42  students_report_harass_race      15617 non-null  float64
 43  students_report_harass_sex       15617 non-null  float64
 44  instances_mech_restraint         15196 non-null  float64
 45  instances_phys_restraint         15538 non-null  float64
 46  instances_seclusion              15538 non-null  float64
 47  students_mech_restraint          15197 non-null  float64
 48  students_phys_restraint          15526 non-null  float64
 49  students_seclusion               15539 non-null  float64
 50  students_SAT_ACT                 15617 non-null  float64
dtypes: float64(22), int64(2), object(27)
memory usage: 6.3+ MB
```

In [71]:
```
read_test = ['school_name', 'read_test_num_valid', 'read_test_pct_p
             'read_test_pct_prof_high', 'read_test_pct_prof_midpt']
```

executed in 2ms, finished 14:34:09 2021-06-17

Contents

In [72]:
```
df_school[read_test].head(20)
```

executed in 10ms, finished 14:34:09 2021-06-17

Out[72]:

	school_name	read_test_num_valid	read_test_pct_prof
33	E. O. Smith High School	260.0	
36	Common Ground High School	39.0	
37	The Bridge Academy	139.0	
38	Explorations	21.0	
40	Stamford Academy	25.0	
42	Ansonia High School	137.0	
47	Avon High School	244.0	
48	Berlin High School	231.0	
49	Path Academy	6.0	
50	Bethel High School	199.0	
51	Bloomfield High School	143.0	

In [73]:
```
math_test = ['school_name', 'math_test_num_valid' , 'math_test_pct_
             'math_test_pct_prof_high', 'math_test_pct_prof_midpt']
```

executed in 2ms, finished 14:34:09 2021-06-17

Contents

In [74]:
```
df_school[math_test].head(20)
```

executed in 11ms, finished 14:34:09 2021-06-17

Out[74]:

	school_name	math_test_num_valid	math_test_pct_prc
33	E. O. Smith High School	260.0	
36	Common Ground High School	39.0	
37	The Bridge Academy	139.0	
38	Explorations	21.0	
40	Stamford Academy	25.0	
42	Ansonia High School	137.0	
47	Avon High School	246.0	
48	Berlin High School	231.0	
49	Path Academy	6.0	
50	Bethel High School	199.0	
51	Bloomfield High School	143.0	
52	Learning Academy at Bloomfield	1.0	
53	Bolton High School	67.0	
54	Branford High School	233.0	
56	Bassick High School	119.0	
57	Central High School	249.0	
58	Harding High School	176.0	
59	Information Technology and Software Engineerin...	132.0	
60	Biotechnology Research and Zoological Studies ...	134.0	
61	Aerospace/Hydrospace Engineering and Physical ...	130.0	

The read_test_num_valid column and the math_test_num_valid column desribe the Numb and for whom a proficiency level was assigned

The low, high, and midpt columns describe the low, high, and midpoint of the range used language arts assessment (0–100 scale)

In [75]:
```
# The midpoint column is the most balanced measure so we will drop
df_school = df_school.drop(['math_test_pct_prof_low','math_test_pct
                            'read_test_pct_prof_low','read_test_pct_
```

executed in 8ms, finished 14:34:09 2021-06-17

3.3.8 Numeric Columns - nulls

Let's take a look at our columns and check back in on the proportion of nulls. If our propo columns with medians.

In [76]:
```
df_school.info()
```

executed in 23ms, finished 14:34:09 2021-06-17

```
 8   enrollment                      15822 non-null  object
 9   read_test_num_valid             15478 non-null  float64
 10  read_test_pct_prof_midpt        15305 non-null  object
 11  math_test_num_valid             15552 non-null  float64
 12  math_test_pct_prof_midpt        15343 non-null  object
 13  students_susp_in_sch            15583 non-null  object
 14  students_susp_out_sch_single    15598 non-null  object
 15  students_susp_out_sch_multiple  15598 non-null  object
 16  expulsions_no_ed_serv           15598 non-null  object
 17  expulsions_with_ed_serv         15583 non-null  object

 18  expulsions_zero_tolerance       15598 non-null  object
 19  students_arrested               15218 non-null  object
 20  students_referred_law_enforce   15579 non-null  object
 21  transfers_alt_sch_disc          15617 non-null  float64
 22  days_suspended                  15265 non-null  float64
 23  suspensions_instances_preschool 15265 non-null  float64
 24  suspensions_instances           15265 non-null  float64
 25  corpinstances                   15265 non-null  float64
```

Contents

Contents

In [77]:
```
#How do our nulls look?
percent_null_df(df_school,'df_school')
```

executed in 30ms, finished 14:34:09 2021-06-17

Out[77]:
```
'percent of nulls in df_school is 1%'
```

In [78]:
```
column_names = list(df_school.columns)
```

executed in 2ms, finished 14:34:09 2021-06-17

In [79]:
```
categorical = ['year','ncessch','school_name','state_name','lea_nam
               'urban_centric_locale','school_level','school_type',
               'corpinstances_preschool']
```

executed in 2ms, finished 14:34:09 2021-06-17

Contents

In [80]:
```
# We've dropped some of our numerical columns let's update our nume
numerical = set(column_names) - set(categorical)
numerical = list(numerical)
numerical
```

executed in 4ms, finished 14:34:09 2021-06-17

Out[80]:
```
['days_suspended',
 'students_disc_harass_sex',
 'expulsions_with_ed_serv',
 'students_arrested',
 'salaries_teachers',
 'students_seclusion',
 'instances_phys_restraint',
 'math_test_num_valid',
 'students_disc_harass_dis',
 'instances_mech_restraint',
 'math_test_pct_prof_midpt',
 'read_test_pct_prof_midpt',
 'corpinstances',
 'read_test_num_valid',
 'allegations_harass_race',
 'students_susp_out_sch_single',
 'students_report_harass_race',
 'allegations_harass_disability',
 'students_susp_out_sch_multiple',
 'enrollment',
 'suspensions_instances',
 'suspensions_instances_preschool',
 'expulsions_zero_tolerance',
 'students_mech_restraint',
 'transfers_alt_sch_disc',
 'cohort_num',
 'allegations_harass_orientation',
 'students_report_harass_sex',
 'allegations_harass_religion',
 'students_phys_restraint',
 'students_SAT_ACT',
 'students_disc_harass_race',
 'instances_seclusion',
 'grad_rate_midpt',
 'students_report_harass_dis',
 'students_susp_in_sch',
 'expulsions_no_ed_serv',
```

```
 'allegations_harass_sex',
 'students_referred_law_enforce']
```

In [81]:
```
numerical.remove('grad_rate_midpt')
```
executed in 2ms, finished 14:34:09 2021-06-17

In [82]:
```
nan_values = df_school.isna()
nan_columns = nan_values.any()
columns_with_nan = df_school.columns[nan_columns].tolist()
print(columns_with_nan)
```
executed in 16ms, finished 14:34:09 2021-06-17

```
['enrollment', 'read_test_num_valid', 'read_test_pct_prof_midpt', '
t', 'students_susp_in_sch', 'students_susp_out_sch_single', 'studer
rv', 'expulsions_with_ed_serv', 'expulsions_zero_tolerance', 'stude
'transfers_alt_sch_disc', 'days_suspended', 'suspensions_instances_
nces', 'salaries_teachers', 'allegations_harass_sex', 'allegations_
'allegations_harass_orientation', 'allegations_harass_religion', 's
_race', 'students_disc_harass_sex', 'students_report_harass_dis', '
harass_sex', 'instances_mech_restraint', 'instances_phys_restraint'
nt', 'students_phys_restraint', 'students_seclusion', 'students_SAT
```

In [83]:
```
#Replace NaNs with median for each column
df_school[numerical] = df_school[numerical].fillna(df_school[numeri
```
executed in 70ms, finished 14:34:09 2021-06-17

In [84]:
```
nan_values = df_school.isna()
nan_columns = nan_values.any()
columns_with_nan = df_school.columns[nan_columns].tolist()
print(columns_with_nan)
```
executed in 16ms, finished 14:34:09 2021-06-17

```
[]
```

In [85]:
```
df_school[numerical] = df_school[numerical].astype(str).astype(floa
```
executed in 242ms, finished 14:34:09 2021-06-17

In [86]:
```
#How do our nulls look?
percent_null_df(df_school,'df_school')
```

executed in 26ms, finished 14:34:10 2021-06-17

Out[86]:
```
'percent of nulls in df_school is 0%'
```

3.3.9 Visualizations

Contents

Contents

In [87]:
```
df_school.info()
```

executed in 13ms, finished 14:34:10 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15829 entries, 33 to 3966
Data columns (total 47 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   ncessch                               15829 non-null  int64
 1   school_name                           15829 non-null  object
 2   state_name                            15829 non-null  object
 3   lea_name                              15829 non-null  object
 4   urban_centric_locale                  15829 non-null  object
 5   school_level                          15829 non-null  object
 6   school_type                           15829 non-null  object
 7   charter                               15829 non-null  float64
 8   enrollment                            15829 non-null  int64
 9   read_test_num_valid                   15829 non-null  int64
 10  read_test_pct_prof_midpt              15829 non-null  int64
 11  math_test_num_valid                   15829 non-null  int64
 12  math_test_pct_prof_midpt              15829 non-null  int64
 13  students_susp_in_sch                  15829 non-null  int64
 14  students_susp_out_sch_single          15829 non-null  int64
 15  students_susp_out_sch_multiple        15829 non-null  int64
 16  expulsions_no_ed_serv                 15829 non-null  int64
 17  expulsions_with_ed_serv               15829 non-null  int64
 18  expulsions_zero_tolerance             15829 non-null  int64
 19  students_arrested                     15829 non-null  int64
 20  students_referred_law_enforce         15829 non-null  int64
 21  transfers_alt_sch_disc                15829 non-null  int64
 22  days_suspended                        15829 non-null  int64
 23  suspensions_instances_preschool       15829 non-null  int64
 24  suspensions_instances                 15829 non-null  int64
 25  corpinstances                         15829 non-null  int64
 26  salaries_teachers                     15829 non-null  int64
 27  cohort_num                            15829 non-null  int64
 28  grad_rate_midpt                       15829 non-null  int64
 29  allegations_harass_sex                15829 non-null  int64
 30  allegations_harass_race               15829 non-null  int64
 31  allegations_harass_disability         15829 non-null  int64
 32  allegations_harass_orientation        15829 non-null  int64
 33  allegations_harass_religion           15829 non-null  int64
 34  students_disc_harass_dis              15829 non-null  int64
```

Contents


```
 35  students_disc_harass_race        15829 non-null  int64
 36  students_disc_harass_sex         15829 non-null  int64
 37  students_report_harass_dis       15829 non-null  int64
 38  students_report_harass_race      15829 non-null  int64
 39  students_report_harass_sex       15829 non-null  int64
 40  instances_mech_restraint         15829 non-null  int64
 41  instances_phys_restraint         15829 non-null  int64
 42  instances_seclusion              15829 non-null  int64
 43  students_mech_restraint          15829 non-null  int64
 44  students_phys_restraint          15829 non-null  int64
 45  students_seclusion               15829 non-null  int64
 46  students_SAT_ACT                 15829 non-null  int64
dtypes: float64(1), int64(40), object(6)
memory usage: 5.8+ MB
```

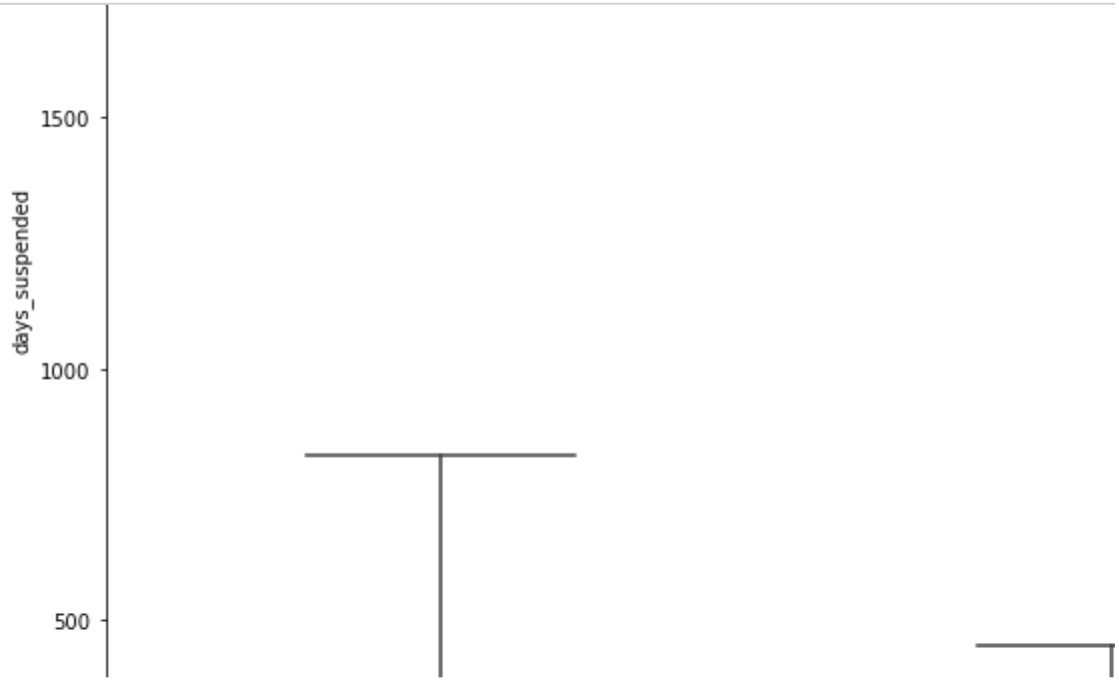
In [88]:
```
graph_feats = ['days_suspended','suspensions_instances','students_s
               'salaries_teachers','math_test_num_valid','read_test_
```

executed in 2ms, finished 14:34:10 2021-06-17

Contents

In [89]:
```
for variable in graph_feats:
    ax, figure = plt.subplots(1,1,figsize=(12,12))
    plt.ylim(-100,2500)
    sns.boxplot(x='grad_rate_midpt', y=variable, data=df_school, sh
    plt.title("{} vs. Graduation Rate".format(variable))
```

executed in 799ms, finished 14:34:10 2021-06-17



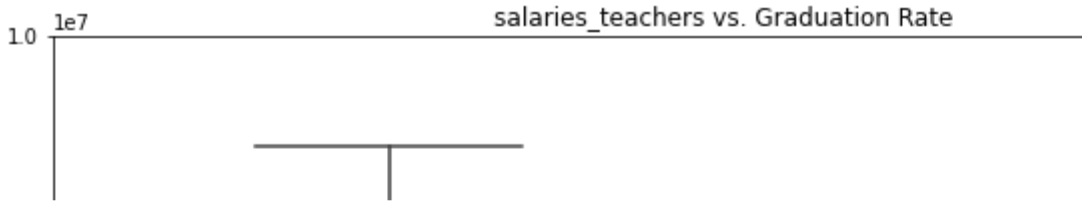
In [90]:
```
numerical_teacher = ['salaries_teachers']
```

executed in 3ms, finished 14:34:10 2021-06-17

Contents

In [91]:
```
for variable in numerical_teacher:
    ax, figure = plt.subplots(1,1,figsize=(12,12))
    plt.ylim(-100,10000000)
    sns.boxplot(x='grad_rate_midpt', y=variable, data=df_school, sh
    plt.title("{} vs. Graduation Rate".format(variable))
```

executed in 118ms, finished 14:34:10 2021-06-17



In [92]:
```
graph_feats = ['days_suspended','suspensions_instances','students_s
               'salaries_teachers','math_test_num_valid','read_test_
```

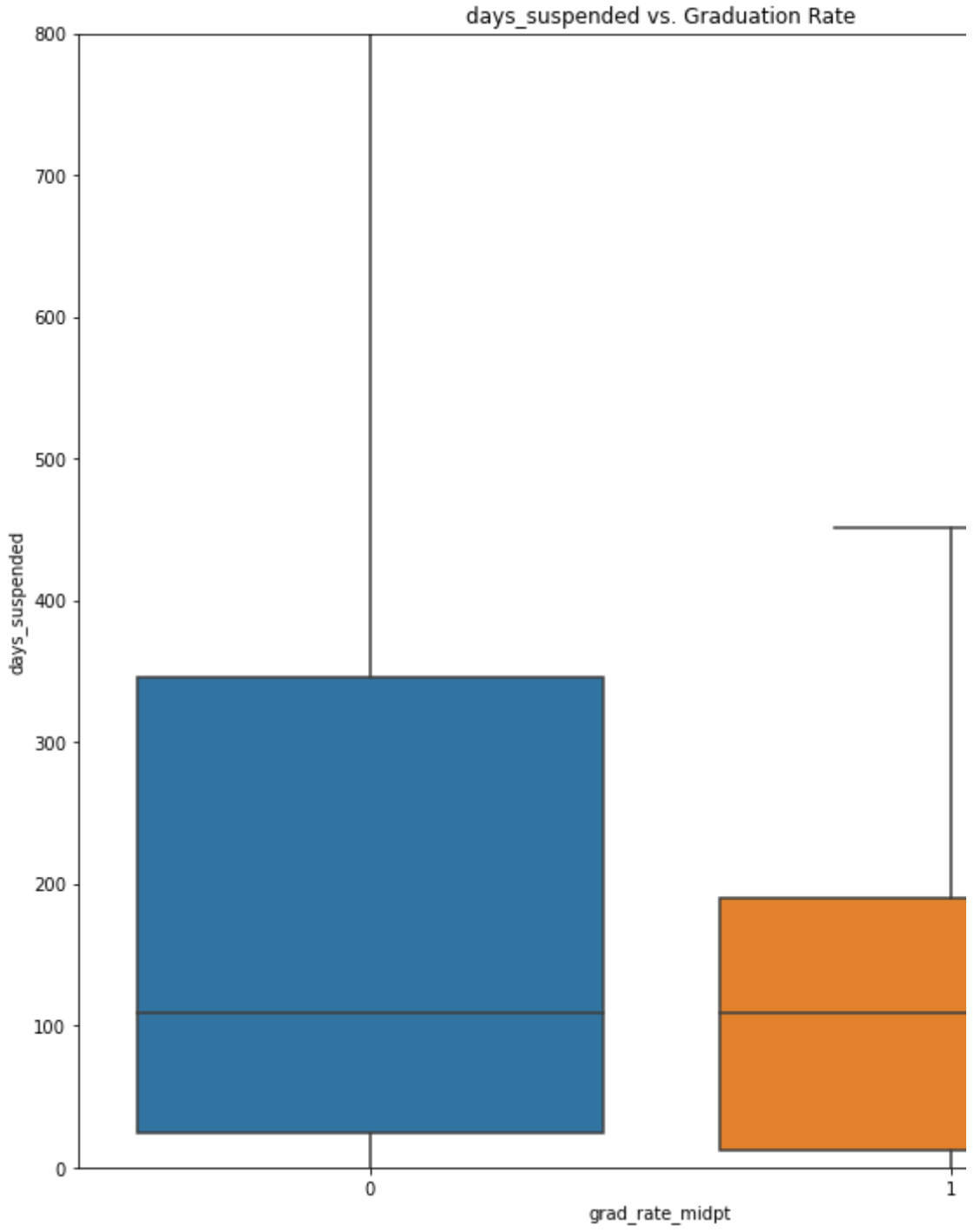
executed in 3ms, finished 14:34:10 2021-06-17

In [93]:
```
for variable in graph_feats:
    ax, figure = plt.subplots(1,1,figsize=(12,12))
    plt.ylim(0,800)
    sns.boxplot(x='grad_rate_midpt', y=variable, data=df_school, sh
    plt.title("{} vs. Graduation Rate".format(variable))
```

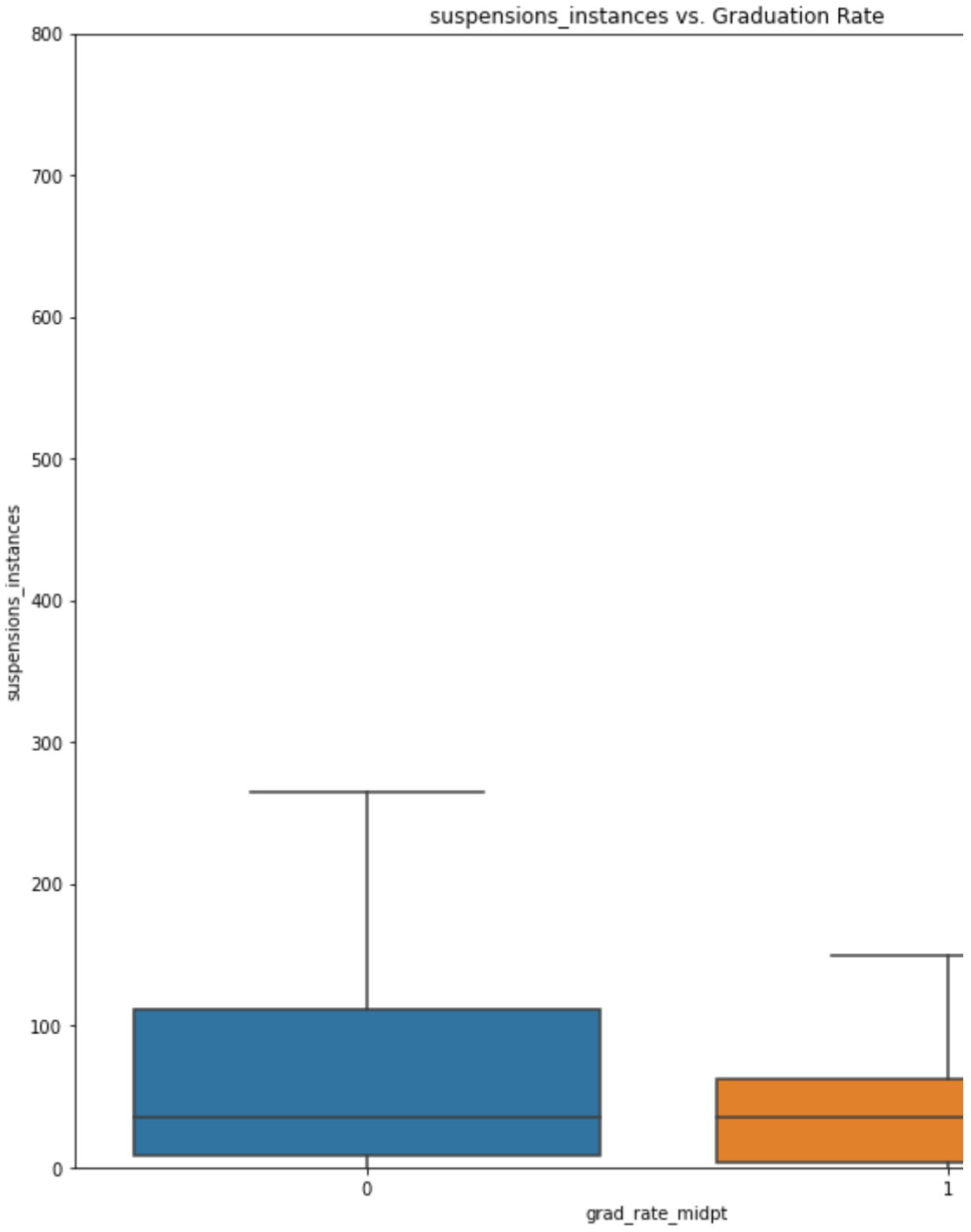
executed in 956ms, finished 14:34:11 2021-06-17

Contents

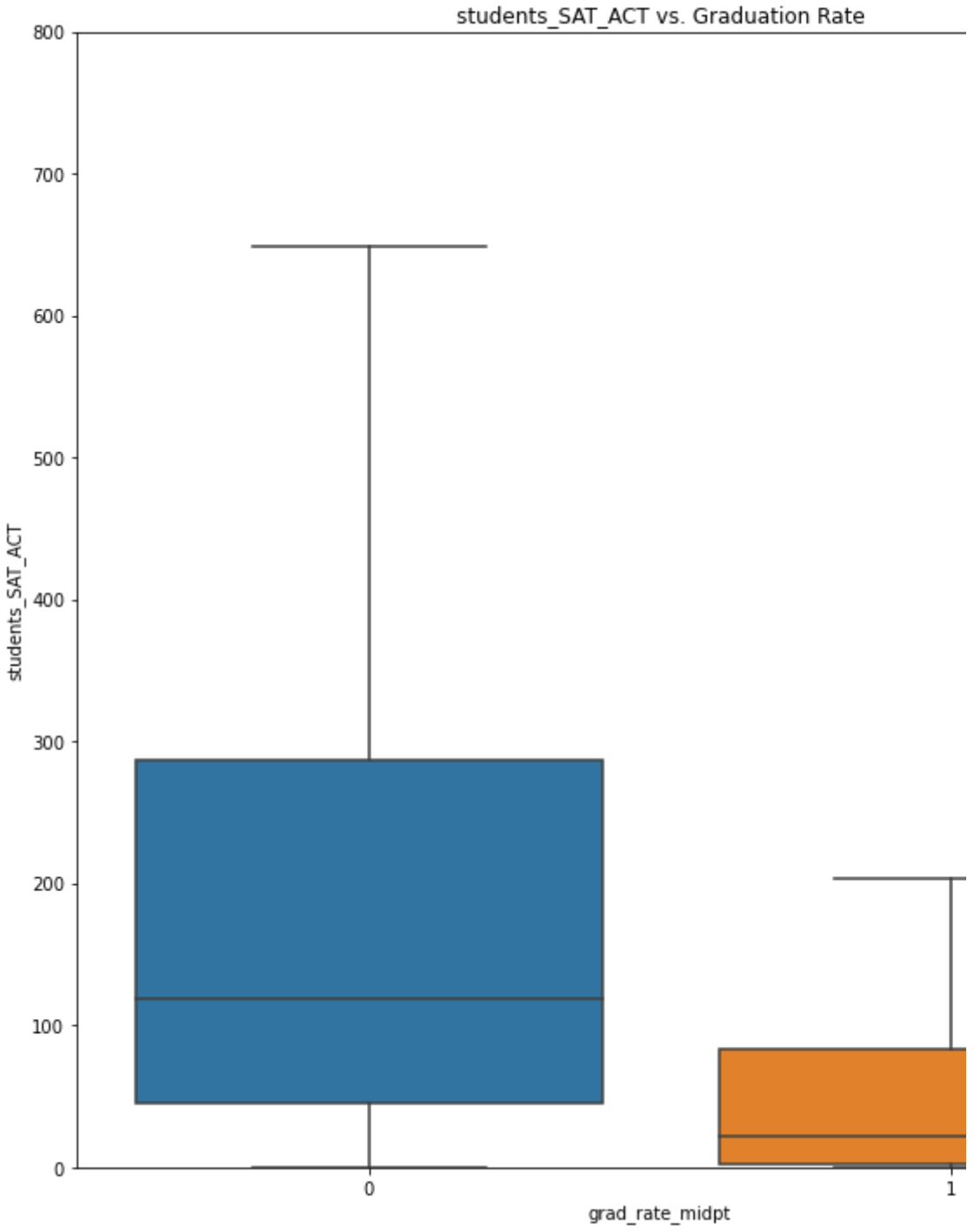
Contents



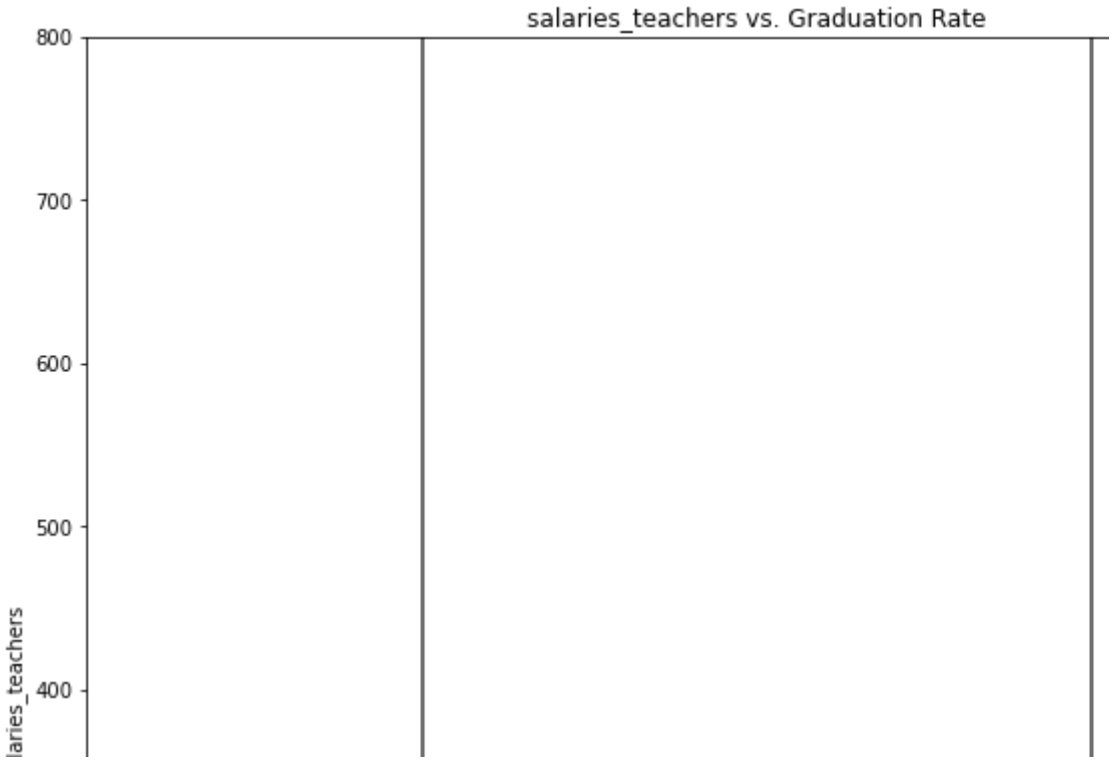
Contents



Contents

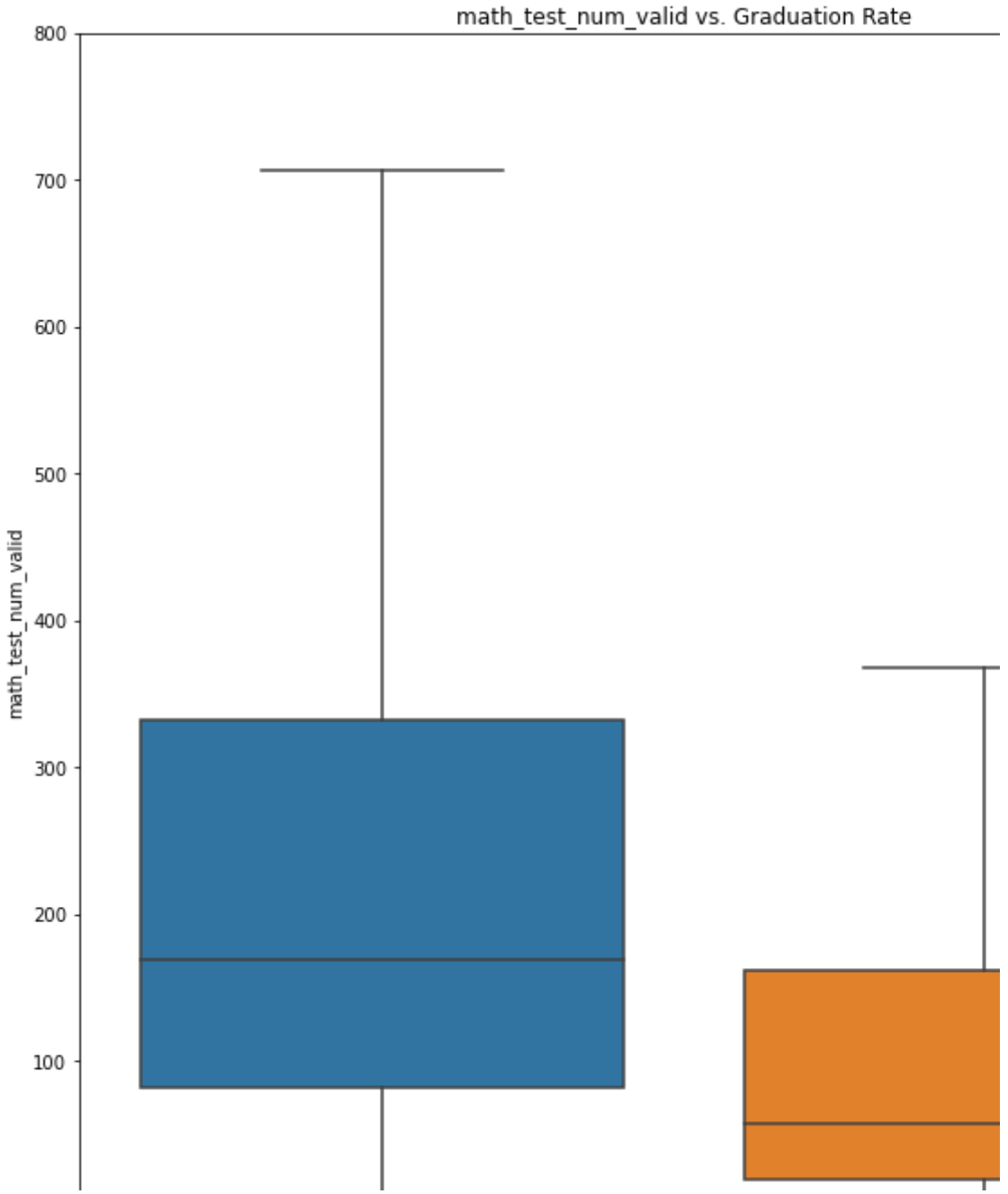


Contents

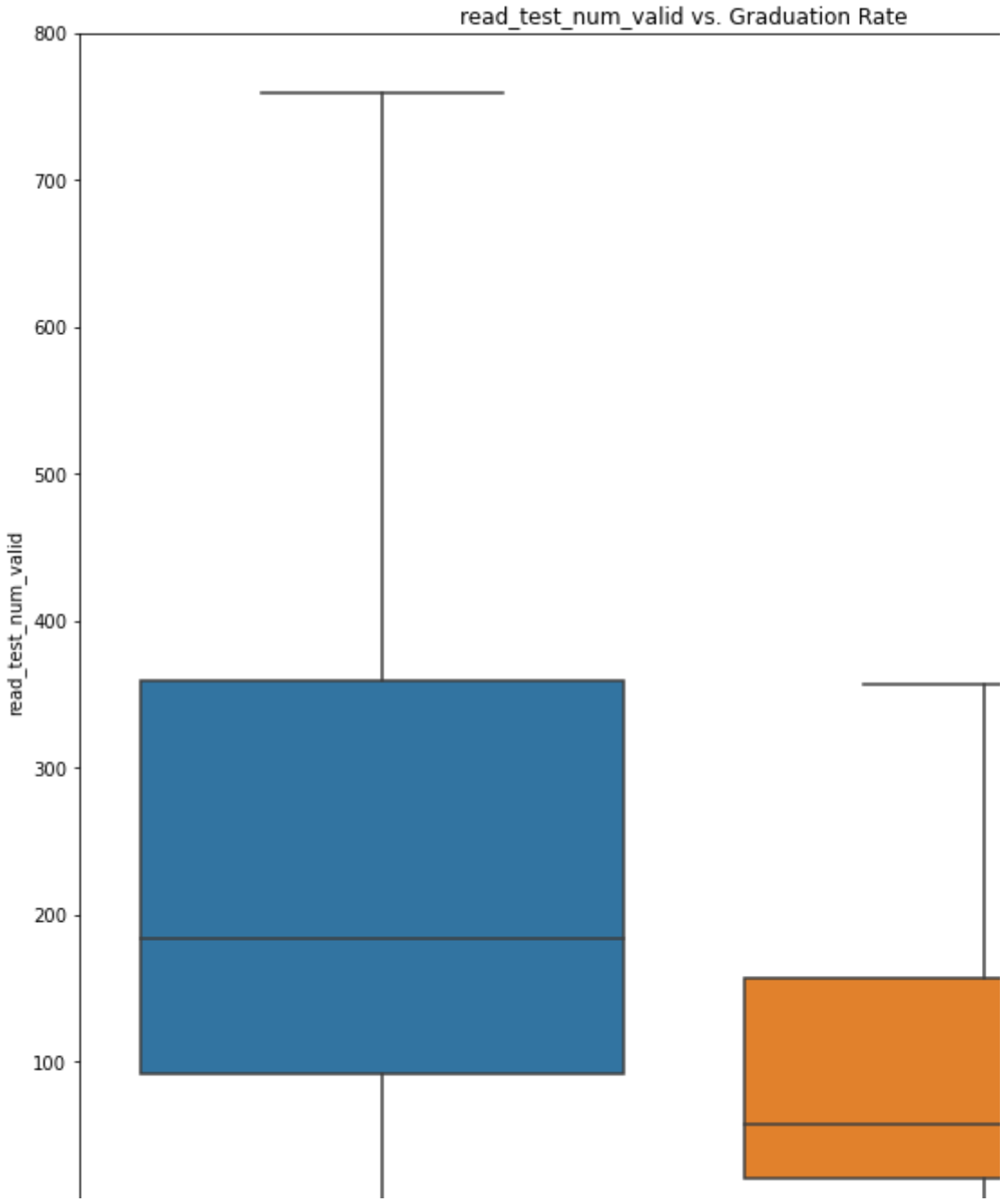


Contents

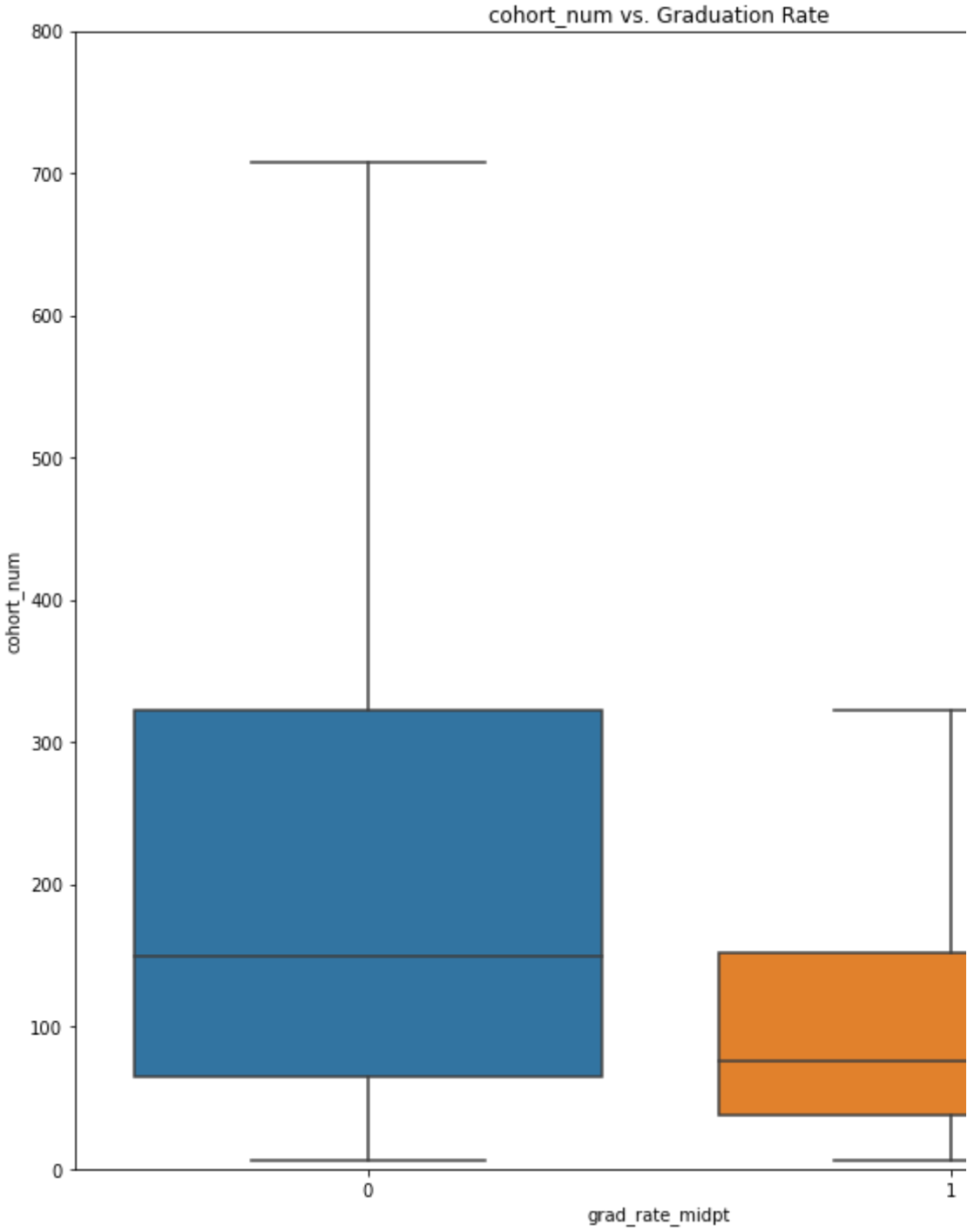
Contents



Contents



Contents



Contents

Contents

3.4 df_district

In [94]:
```
#Concatenate district info for all three states into one dataframe
df_district = pd.concat([df_district_northeast, df_district_south_a
                         df_district_midwest, df_district_west, df_
```

executed in 24ms, finished 14:34:11 2021-06-17

In [95]:
```
df_district['state_name'].value_counts(normalize=True, dropna=False
```

executed in 6ms, finished 14:34:11 2021-06-17

Out[95]:

Texas	0.067443
California	0.062879
Ohio	0.060570
Illinois	0.057134
New York	0.055308
Michigan	0.049992
Pennsylvania	0.042958
Arizona	0.038125
New Jersey	0.037427
Oklahoma	0.032594
Minnesota	0.031305
Missouri	0.030768
Montana	0.026687
Wisconsin	0.025345
Indiana	0.023090
Massachusetts	0.021962
Vermont	0.019492
Iowa	0.018633
Washington	0.017720
Kansas	0.017022
North Carolina	0.016968
New Hampshire	0.016109
Arkansas	0.015626
Nebraska	0.015250
Maine	0.014391
Colorado	0.014230
North Dakota	0.012189
Georgia	0.011974
Virginia	0.011974
Louisiana	0.011974
Oregon	0.011867
Connecticut	0.011115
Kentucky	0.009988
Alabama	0.009665
South Dakota	0.009182
Utah	0.008914
Mississippi	0.008645
Idaho	0.008592
New Mexico	0.008538
Tennessee	0.007840

```
South Carolina           0.005477
Florida                  0.004081
District of Columbia     0.003974
Rhode Island             0.003437
Wyoming                  0.003276
West Virginia            0.003061
Delaware                 0.002846
Maryland                 0.001342
Nevada                   0.001020
Name: state_name, dtype: float64
```

3.4.1 Data Fields

In [96]:
```
df_district.head()
```

executed in 19ms, finished 14:34:11 2021-06-17

Out[96]:

	year	leaid	lea_name	state_name	state_leaid	city_location	urban_centric_loca
0	2015	900001	UNIFIED SCHOOL DISTRICT #3	Connecticut	349	HARTFORD	City, midsi
1	2015	900002	Connecticut Technical High Sc	Connecticut	900	MIDDLETOWN	Suburb, lar
2	2015	900003	UNIFIED SCHOOL DISTRICT #1	Connecticut	336	WETHERSFIELD	Suburb, lar
3	2015	900004	UNIFIED SCHOOL DISTRICT #2	Connecticut	347	HARTFORD	City, sm
4	2015	900005	REGIONAL SCHOOL DISTRICT 19	Connecticut	219	STORRS	Suburb, lar

5 rows × 55 columns

Contents

In [97]:
```
df_district.info()
```

executed in 56ms, finished 14:34:12 2021-06-17

```
 37  exp_nonelsec_other              18623 non-null  object
 38  exp_textbooks                   18623 non-null  object
 39  exp_utilities_energy            18623 non-null  object
 40  exp_tech_supplies_services      18623 non-null  object
 41  exp_tech_equipment              18623 non-null  object
 42  outlay_capital_total            18623 non-null  object
 43  outlay_capital_construction     18623 non-null  object
 44  outlay_capital_land_structures  18623 non-null  object
 45  outlay_capital_instruc_equip    18623 non-null  object
 46  outlay_capital_other_equip      18623 non-null  object
 47  outlay_capital_nonspec_equip    18623 non-null  object
 48  salaries_total                  18623 non-null  object
 49  salaries_instruction            18623 non-null  object
 50  benefits_employee_total         18623 non-null  object
 51  cohort_num                      12235 non-null  float64
 52  grad_rate_high                  12235 non-null  object
 53  grad_rate_low                   12235 non-null  object
 54  grad_rate_midpt                 12235 non-null  object
dtypes: float64(4), int64(2), object(49)
memory usage: 8.0+ MB
```

In [98]:
```
#What proportion of our data frame is nulls?
percent_null_df(df_district,'df_district')
```

executed in 63ms, finished 14:34:12 2021-06-17

Out[98]:
```
'percent of nulls in df_district is 4%'
```

3.4.1.1 Data Field Cleanup

In [99]:
```
column_names = list(df_district.columns)
```

executed in 2ms, finished 14:34:12 2021-06-17

In [100]:
```
categorical = ['year','leaid','leaname','state_name','state_leaid',
               'urban_centric_locale','agency_type']
```

executed in 2ms, finished 14:34:12 2021-06-17

Contents

In [101]:
```
for ele in categorical:
    try:
        column_names.remove(ele)
    except ValueError:
        pass
```

executed in 2ms, finished 14:34:12 2021-06-17

In [102]:
```
numerical = column_names
```

executed in 2ms, finished 14:34:12 2021-06-17

In [103]:
```
column_names = list(df_district.columns)
```

executed in 2ms, finished 14:34:12 2021-06-17

In [104]:
```
for c in column_names:
    df_district[c].replace('Suppressed data', np.NaN,inplace =True)
```

executed in 35ms, finished 14:34:12 2021-06-17

In [105]:
```
for c in column_names:
    df_district[c].replace('Not applicable', np.NaN,inplace =True)
```

executed in 37ms, finished 14:34:12 2021-06-17

In [106]:
```
for c in column_names:
    df_district[c].replace('Missing/not reported', np.NaN,inplace =
```

executed in 35ms, finished 14:34:12 2021-06-17

3.4.2 Columns to drop

Contents

In [107]:
```
df_district.info()
```

executed in 37ms, finished 14:34:12 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18623 entries, 0 to 3049
Data columns (total 55 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   year                                  18623 non-null  int64  
 1   leaid                                 18623 non-null  int64  
 2   lea_name                              18623 non-null  object 
 3   state_name                            18623 non-null  object 
 4   state_leaid                           18623 non-null  object 
 5   city_location                         18623 non-null  object 
 6   urban_centric_locale                  18592 non-null  object 
 7   agency_type                           18623 non-null  object 
 8   enrollment                            17047 non-null  object 
 9   english_language_learners             12381 non-null  object 
 10  est_population_total                  13111 non-null  float64
 11  est_population_5_17_poverty           13111 non-null  float64
 12  est_population_5_17_poverty_pct       13107 non-null  float64
 13  rev_total                             16883 non-null  object 
 14  rev_fed_total                         16883 non-null  object 
 15  rev_state_total                       16883 non-null  object 
 16  rev_local_total                       16883 non-null  object 
 17  rev_local_prop_tax                    12577 non-null  object 
 18  exp_total                             16883 non-null  object 
 19  exp_current_elsec_total               16883 non-null  object 
 20  exp_current_instruction_total         16883 non-null  object 
 21  exp_current_supp_serve_total          16883 non-null  object 
 22  exp_current_pupils                    16883 non-null  object 
 23  exp_current_instruc_staff             16883 non-null  object 
 24  exp_current_general_admin             16883 non-null  object 
 25  exp_current_sch_admin                 16883 non-null  object 
 26  exp_current_operation_plant           16883 non-null  object 
 27  exp_current_student_transport         16883 non-null  object 
 28  exp_current_bco                       16883 non-null  object 
 29  exp_current_supp_serv_nonspec         16883 non-null  object 
 30  exp_current_other                     16883 non-null  object 
 31  exp_current_food_serv                 16883 non-null  object 
 32  exp_current_enterprise                16883 non-null  object 
 33  exp_current_other_elsec               16883 non-null  object 
 34  exp_nonelsec                          16883 non-null  object 
```

```
 35  exp_nonelsec_community_serv      16883 non-null  object
 36  exp_nonelsec_adult_education     16883 non-null  object
 37  exp_nonelsec_other               16883 non-null  object
 38  exp_textbooks                    16883 non-null  object
 39  exp_utilities_energy             16883 non-null  object
 40  exp_tech_supplies_services       16883 non-null  object
 41  exp_tech_equipment               16883 non-null  object
 42  outlay_capital_total             16883 non-null  object
 43  outlay_capital_construction      16883 non-null  object
 44  outlay_capital_land_structures   16883 non-null  object
 45  outlay_capital_instruc_equip     16883 non-null  object
 46  outlay_capital_other_equip       16883 non-null  object
 47  outlay_capital_nonspec_equip     16883 non-null  object
 48  salaries_total                   16883 non-null  object
 49  salaries_instruction             16883 non-null  object
 50  benefits_employee_total          16883 non-null  object
 51  cohort_num                       12235 non-null  float64
 52  grad_rate_high                   11972 non-null  object
 53  grad_rate_low                    11972 non-null  object
 54  grad_rate_midpt                  11972 non-null  object
dtypes: float64(4), int64(2), object(49)
memory usage: 8.0+ MB
```

In [108]:
```
# Our population columns have the highest proportion of nulls
df_district = df_district.drop(['est_population_total','est_populat
                                'est_population_5_17_poverty_pct'], axis
```

executed in 11ms, finished 14:34:12 2021-06-17

In [109]:
```
# Drop grad rates and urban centric locale:
# we don't want to delete any districts and are using grad rate and
df_district = df_district.drop(['grad_rate_high','grad_rate_low',
                                'grad_rate_midpt','cohort_num','urb
```

executed in 10ms, finished 14:34:12 2021-06-17

Contents

In [110]:
```
df_district['enrollment'].value_counts(normalize=True, dropna=False
```

executed in 11ms, finished 14:34:12 2021-06-17

Out[110]:
```
NaN      0.084627
0        0.020459
177      0.001557
202      0.001450
147      0.001289
           ...
83648    0.000054
4334     0.000054
8824     0.000054
23529    0.000054
4308     0.000054
Name: enrollment, Length: 5443, dtype: float64
```

In [111]:
```
# Enrollment is mostly zeros or NaNs
df_district = df_district.drop(['enrollment'], axis=1)
```

executed in 10ms, finished 14:34:12 2021-06-17

In [112]:
```
# year is the same for all rows, state_name same in school data, st
df_district = df_district.drop(['year','state_name','state_leaid'],
```

executed in 10ms, finished 14:34:12 2021-06-17

In [113]:
```
#What proportion of our data frame is nulls?
percent_null_df(df_district,'df_district')
```

executed in 54ms, finished 14:34:12 2021-06-17

Out[113]:
```
'percent of nulls in df_district is 9%'
```

3.4.3 english_language_learners

Contents

In [114]:
```
df_district['english_language_learners'].value_counts(normalize=Tru
```
executed in 8ms, finished 14:34:12 2021-06-17

Out[114]:
```
NaN      0.335177
3        0.102669
4        0.021694
0        0.020459
5        0.018955
           ...
5423     0.000054
1553     0.000054
1996     0.000054
13796    0.000054
675      0.000054
Name: english_language_learners, Length: 1581, dtype: float64
```

In [115]:
```
ell = ['english_language_learners']
```
executed in 2ms, finished 14:34:12 2021-06-17

In [116]:
```
for c in ell:
    df_district[c].replace('Missing/not reported', np.NaN,inplace =
```
executed in 4ms, finished 14:34:12 2021-06-17

In [117]:
```
df_district['english_language_learners'].value_counts(normalize=Tru
```
executed in 7ms, finished 14:34:12 2021-06-17

Out[117]:
```
NaN      0.335177
3        0.102669
4        0.021694
0        0.020459
5        0.018955
           ...
5423     0.000054
1553     0.000054
1996     0.000054
13796    0.000054
675      0.000054
Name: english_language_learners, Length: 1581, dtype: float64
```

Contents

In [118]:
```
# fill na with median
df_district['english_language_learners'] = df_district['english_lar
```

executed in 6ms, finished 14:34:12 2021-06-17

In [119]:
```
df_district['english_language_learners'].value_counts(normalize=Tru
```

executed in 8ms, finished 14:34:12 2021-06-17

Out[119]:
```
26.0     0.335177
3        0.102669
4        0.021694
0        0.020459
5        0.018955
           ...
735      0.000054
62575    0.000054
2085     0.000054
991      0.000054
1812     0.000054
Name: english_language_learners, Length: 1581, dtype: float64
```

Contents

In [120]:
```
df_district.info()
```

executed in 33ms, finished 14:34:12 2021-06-17

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18623 entries, 0 to 3049
Data columns (total 42 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   leaid                           18623 non-null  int64 
 1   lea_name                        18623 non-null  object
 2   agency_type                     18623 non-null  object
 3   english_language_learners       18623 non-null  object
 4   rev_total                       16883 non-null  object
 5   rev_fed_total                   16883 non-null  object
 6   rev_state_total                 16883 non-null  object
 7   rev_local_total                 16883 non-null  object
 8   rev_local_prop_tax              12577 non-null  object
 9   exp_total                       16883 non-null  object
 10  exp_current_elsec_total         16883 non-null  object
 11  exp_current_instruction_total   16883 non-null  object
 12  exp_current_supp_serve_total    16883 non-null  object
 13  exp_current_pupils              16883 non-null  object
```

In [121]:
```
df_district = df_district.drop_duplicates(subset=['lea_name'])
```

executed in 14ms, finished 14:34:12 2021-06-17

4 Join Datasets

In [122]:
```
df_school.shape
```

executed in 3ms, finished 14:34:12 2021-06-17

Out[122]: (15829, 47)

In [123]:
```
df_district.shape
```

executed in 3ms, finished 14:34:12 2021-06-17

Out[123]: (18257, 42)

Contents

In [124]:
```
# Columns to drop
df_school = df_school.drop(['ncessch','school_level','school_type',
```

executed in 5ms, finished 14:34:12 2021-06-17

In [125]:
```
df_train = df_school.merge(df_district, on = 'lea_name')
df_train.head()
```

executed in 48ms, finished 14:34:12 2021-06-17

Out[125]:

	state_name	lea_name	urban_centric_locale	charter	enrollment	read_test_num_v
0	Connecticut	REGIONAL SCHOOL DISTRICT 19	Suburb, large	0.0	1153	
1	Connecticut	COMMON GROUND HIGH SCHOOL DIS	City, midsize	1.0	186	
2	Connecticut	THE BRIDGE ACADEMY DISTRICT	City, midsize	1.0	279	
3	Connecticut	EXPLORATIONS DISTRICT	Town, fringe	1.0	92	
4	Connecticut	Stamford Academy	City, midsize	1.0	150	

5 rows × 83 columns

In [126]:
```
df_train.info()
```

executed in 48ms, finished 14:34:12 2021-06-17

```
 34  students_report_harass_sex       15785 non-null  int64
 35  instances_mech_restraint         15785 non-null  int64
 36  instances_phys_restraint         15785 non-null  int64
 37  instances_seclusion              15785 non-null  int64
 38  students_mech_restraint          15785 non-null  int64
 39  students_phys_restraint          15785 non-null  int64
 40  students_seclusion               15785 non-null  int64
 41  students_SAT_ACT                 15785 non-null  int64
 42  leaid                            15785 non-null  int64
 43  agency_type                      15785 non-null  object
 44  english_language_learners        15785 non-null  object
 45  rev_total                        15331 non-null  object
 46  rev_fed_total                    15331 non-null  object
 47  rev_state_total                  15331 non-null  object
 48  rev_local_total                  15331 non-null  object
 49  rev_local_prop_tax               13039 non-null  object
 50  exp_total                        15331 non-null  object
 51  exp_current_elsec_total          15331 non-null  object
 52  exp_current_instruction_total    15331 non-null  object
 53  exp_current_supp_serve_total     15331 non-null  object
```

In [127]:
```
# Columns to drop - we don't need these anymore
df_train = df_train.drop(['leaid','lea_name'], axis=1)
```

executed in 11ms, finished 14:34:12 2021-06-17

4.1 Check for NaNs

In [128]:
```
nan_values = df_train.isna()
nan_columns = nan_values.any()
```

executed in 25ms, finished 14:34:12 2021-06-17

In [129]:
```
columns_with_nan = df_train.columns[nan_columns].tolist()
print(columns_with_nan)
```

executed in 3ms, finished 14:34:12 2021-06-17

```
['rev_total', 'rev_fed_total', 'rev_state_total', 'rev_local_total'
nt_elsec_total', 'exp_current_instruction_total', 'exp_current_supp
nt_instruc_staff', 'exp_current_general_admin', 'exp_current_sch_ac
nt_student_transport', 'exp_current_bco', 'exp_current_supp_serv_nc
serv', 'exp_current_enterprise', 'exp_current_other_elsec', 'exp_nc
nelsec_adult_education', 'exp_nonelsec_other', 'exp_textbooks', 'ex
s', 'exp_tech_equipment', 'outlay_capital_total', 'outlay_capital_c
'outlay_capital_instruc_equip', 'outlay_capital_other_equip', 'outl
alaries_instruction', 'benefits_employee_total']
```

We still have NaN values in our numerical fields

5 Numeric Columns - cleanup

5.1 Cleanup

In [130]:
```
col_names_category = ['state_name','urban_centric_locale',
                      'agency_type','grad_rate_midpt']
```

executed in 2ms, finished 14:34:12 2021-06-17

In [131]:
```
col_names = list(df_train.columns)
```

executed in 2ms, finished 14:34:12 2021-06-17

In [132]:
```
num_col_names = set(col_names) - set(col_names_category)
num_col_names = list(num_col_names)
```

executed in 2ms, finished 14:34:12 2021-06-17

In [133]:
```
df_train[num_col_names] = df_train[num_col_names].fillna(df_train[r
```

executed in 158ms, finished 14:34:12 2021-06-17

In [134]:
```
df_train[num_col_names] = df_train[num_col_names].astype(str).astyp
```
executed in 664ms, finished 14:34:13 2021-06-17

5.2 Engineer % Columns

In [135]:
```
df_train['SAT_ACT_%_enrollment'] = df_train['students_SAT_ACT'] / d
```
executed in 7ms, finished 14:34:13 2021-06-17

In [136]:
```
df_train['teacher_%_total_salary'] =  df_train['salaries_instructio
df_train.head()
```
executed in 21ms, finished 14:34:13 2021-06-17

Out[136]:

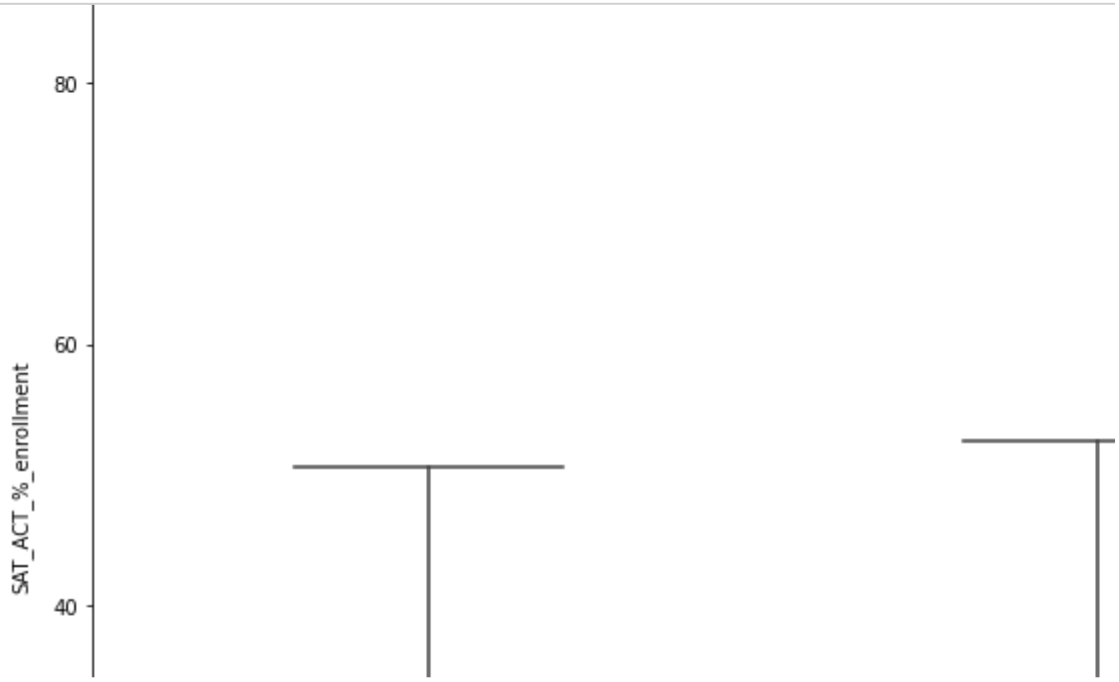
	state_name	urban_centric_locale	charter	enrollment	read_test_num_valid	read_test_p
0	Connecticut	Suburb, large	0	1153	260	
1	Connecticut	City, midsize	1	186	39	
2	Connecticut	City, midsize	1	279	139	
3	Connecticut	Town, fringe	1	92	21	
4	Connecticut	City, midsize	1	150	25	

5 rows × 83 columns

In [137]:
```
pct_graph = ['SAT_ACT_%_enrollment','teacher_%_total_salary','read_
```
executed in 2ms, finished 14:34:13 2021-06-17

In [138]:
```
for variable in pct_graph:
    ax, figure = plt.subplots(1,1,figsize=(12,12))
    plt.ylim(0,100)
    sns.boxplot( x='grad_rate_midpt', y=variable, data=df_train, sh
    plt.title("{} vs. Graduation Rate".format(variable))
```

executed in 475ms, finished 14:34:13 2021-06-17



In [139]:
```
df_train = df_train.drop(['salaries_instruction','salaries_total','
```

executed in 18ms, finished 14:34:14 2021-06-17

5.3 Reset Num List

In [140]:
```
col_names_category = ['state_name','urban_centric_locale',
                      'agency_type','grad_rate_midpt']
```

executed in 2ms, finished 14:34:14 2021-06-17

In [141]:
```
col_names = list(df_train.columns)
```

executed in 2ms, finished 14:34:14 2021-06-17

Contents

In [142]:
```
num_col_names = set(col_names) - set(col_names_category)
num_col_names = list(num_col_names)
```
executed in 3ms, finished 14:34:14 2021-06-17

In [143]:
```
df_train[num_col_names] = df_train[num_col_names].fillna(df_train[r
```
executed in 42ms, finished 14:34:14 2021-06-17

In [144]:
```
df_train[num_col_names] = df_train[num_col_names].astype(str).astyp
```
executed in 889ms, finished 14:34:14 2021-06-17

6 Category Columns - cleanup

In [145]:
```
df_train[col_names_category] = df_train[col_names_category].astype(
```
executed in 21ms, finished 14:34:14 2021-06-17

In [146]:
```
col_names_ohe = ['state_name','urban_centric_locale',
                 'agency_type']
```
executed in 2ms, finished 14:34:14 2021-06-17

In [147]:
```
df_train[col_names_ohe] = df_train[col_names_ohe].astype(str)
```
executed in 20ms, finished 14:34:14 2021-06-17

In [148]:
```
df_train.info()
```

executed in 20ms, finished 14:34:15 2021-06-17

```
 29  students_disc_harass_race       15785 non-null  int64
 30  students_disc_harass_sex        15785 non-null  int64
 31  students_report_harass_dis      15785 non-null  int64
 32  students_report_harass_race     15785 non-null  int64
 33  students_report_harass_sex      15785 non-null  int64
 34  instances_mech_restraint        15785 non-null  int64
 35  instances_phys_restraint        15785 non-null  int64
 36  instances_seclusion             15785 non-null  int64
 37  students_mech_restraint         15785 non-null  int64
 38  students_phys_restraint         15785 non-null  int64
 39  students_seclusion              15785 non-null  int64
 40  agency_type                     15785 non-null  object
 41  english_language_learners       15785 non-null  int64
 42  rev_total                       15785 non-null  int64
 43  rev_fed_total                   15785 non-null  int64
 44  rev_state_total                 15785 non-null  int64
 45  rev_local_total                 15785 non-null  int64
 46  rev_local_prop_tax              15785 non-null  int64
 47  exp_total                       15785 non-null  int64
 48  exp current elsec total         15785 non-null  int64
```

6.1 Final Check for NaNs

In [149]:
```
nan_values = df_train.isna()
nan_columns = nan_values.any()
```

executed in 6ms, finished 14:34:15 2021-06-17

In [150]:
```
columns_with_nan = df_train.columns[nan_columns].tolist()
print(columns_with_nan)
```

executed in 4ms, finished 14:34:15 2021-06-17

```
[]
```

7 Train Test Split

Contents

In [151]:
```
X=df_train.drop(columns = ['grad_rate_midpt']) # Features
y=df_train['grad_rate_midpt'] #Target
```

executed in 7ms, finished 14:34:15 2021-06-17

In [152]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
```

executed in 13ms, finished 14:34:15 2021-06-17

In [153]:
```
X_train
```

executed in 22ms, finished 14:34:15 2021-06-17

Out[153]:

	state_name	urban_centric_locale	charter	enrollment	read_test_num_valid	read_t
7105	Minnesota	Rural, distant	0	183	96	
13034	Alabama	Town, distant	0	241	50	
6979	Michigan	Suburb, small	0	1265	300	
9785	Arizona	Town, fringe	1	121	7	
6744	Michigan	City, large	0	269	44	
...	...	...	...	...	...	
7404	Minnesota	Town, distant	0	992	243	
10737	California	Suburb, large	0	1377	312	
11490	Colorado	City, large	1	501	177	
12604	Washington	Town, distant	0	713	163	
5545	Illinois	City, midsize	0	1240	296	

11049 rows × 79 columns

In [154]:
```
X_train.shape
```

executed in 4ms, finished 14:34:15 2021-06-17

Out[154]:
```
(11049, 79)
```

In [155]:
```
X_test.shape
```
executed in 3ms, finished 14:34:15 2021-06-17

Out[155]:
```
(4736, 79)
```


8 Encode Features

In [156]:
```
# define transformers
ss = StandardScaler()
ohe = OneHotEncoder(handle_unknown='ignore', sparse=False,)
# set up pipelines for each column group
numeric_pipe = Pipeline([('ss', ss)])
categorical_pipe = Pipeline([('ohe', ohe)])
# set up columnTransformer
col_transformer = ColumnTransformer(
                    transformers=[
                        ('nums', numeric_pipe, num_col_names),
                        ('cats', categorical_pipe, col_names_ohe),
                    ],
                    remainder='drop',
                    n_jobs=-1
                    )
```
executed in 4ms, finished 14:34:15 2021-06-17

8.1 X_train Encode

In [157]:
```
X_train = col_transformer.fit_transform(X_train)
```
executed in 1.93s, finished 14:34:17 2021-06-17

Contents

In [158]:
```
col_transformer.named_transformers_['cats'].named_steps['ohe']\
    .get_feature_names()
```

executed in 5ms, finished 14:34:17 2021-06-17

Out[158]:
```
array(['x0_Alabama', 'x0_Arizona', 'x0_Arkansas', 'x0_California',
       'x0_Colorado', 'x0_Connecticut', 'x0_Delaware',
       'x0_District of Columbia', 'x0_Florida', 'x0_Georgia', 'x0_I
       'x0_Illinois', 'x0_Indiana', 'x0_Iowa', 'x0_Kansas', 'x0_Ker
       'x0_Louisiana', 'x0_Maine', 'x0_Maryland', 'x0_Massachusetts
       'x0_Michigan', 'x0_Minnesota', 'x0_Mississippi', 'x0_Missour
       'x0_Montana', 'x0_Nebraska', 'x0_Nevada', 'x0_New Hampshire'
       'x0_New Jersey', 'x0_New Mexico', 'x0_New York',
       'x0_North Carolina', 'x0_North Dakota', 'x0_Ohio', 'x0_Oklah
       'x0_Oregon', 'x0_Pennsylvania', 'x0_Rhode Island',
       'x0_South Carolina', 'x0_South Dakota', 'x0_Tennessee', 'x0_
       'x0_Utah', 'x0_Vermont', 'x0_Virginia', 'x0_Washington',
       'x0_West Virginia', 'x0_Wisconsin', 'x0_Wyoming', 'x1_City,
       'x1_City, midsize', 'x1_City, small', 'x1_Rural, distant',
       'x1_Rural, fringe', 'x1_Rural, remote', 'x1_Suburb, large',
       'x1_Suburb, midsize', 'x1_Suburb, small', 'x1_Town, distant'
       'x1_Town, fringe', 'x1_Town, remote', 'x2_Charter agency',
       'x2_Local school district that is a component of a supervisc
       'x2_Other education agency',
       'x2_Regional education service agency',
       'x2_Regular local school district', 'x2_State-operated agenc
       'x2_Supervisory union'], dtype=object)
```

In [159]:
```
X_train.shape
```

executed in 3ms, finished 14:34:17 2021-06-17

Out[159]:
```
(11049, 144)
```

8.2 X_test Encode

In [160]:
```
X_test = col_transformer.transform(X_test)
```

executed in 843ms, finished 14:34:17 2021-06-17

Contents

In [161]:
```
col_transformer.named_transformers_['cats'].named_steps['ohe']\
    .get_feature_names()
```

executed in 5ms, finished 14:34:17 2021-06-17

Out[161]:
```
array(['x0_Alabama', 'x0_Arizona', 'x0_Arkansas', 'x0_California',
       'x0_Colorado', 'x0_Connecticut', 'x0_Delaware',
       'x0_District of Columbia', 'x0_Florida', 'x0_Georgia', 'x0_I
       'x0_Illinois', 'x0_Indiana', 'x0_Iowa', 'x0_Kansas', 'x0_Ker
       'x0_Louisiana', 'x0_Maine', 'x0_Maryland', 'x0_Massachusetts
       'x0_Michigan', 'x0_Minnesota', 'x0_Mississippi', 'x0_Missour
       'x0_Montana', 'x0_Nebraska', 'x0_Nevada', 'x0_New Hampshire'
       'x0_New Jersey', 'x0_New Mexico', 'x0_New York',
       'x0_North Carolina', 'x0_North Dakota', 'x0_Ohio', 'x0_Oklah
       'x0_Oregon', 'x0_Pennsylvania', 'x0_Rhode Island',
       'x0_South Carolina', 'x0_South Dakota', 'x0_Tennessee', 'x0_
       'x0_Utah', 'x0_Vermont', 'x0_Virginia', 'x0_Washington',
       'x0_West Virginia', 'x0_Wisconsin', 'x0_Wyoming', 'x1_City,
       'x1_City, midsize', 'x1_City, small', 'x1_Rural, distant',
       'x1_Rural, fringe', 'x1_Rural, remote', 'x1_Suburb, large',
       'x1_Suburb, midsize', 'x1_Suburb, small', 'x1_Town, distant'
       'x1_Town, fringe', 'x1_Town, remote', 'x2_Charter agency',
       'x2_Local school district that is a component of a supervisc
       'x2_Other education agency',
       'x2_Regional education service agency',
       'x2_Regular local school district', 'x2_State-operated agenc
       'x2_Supervisory union'], dtype=object)
```

In [162]:
```
X_test.shape
```

executed in 4ms, finished 14:34:17 2021-06-17

Out[162]:
```
(4736, 144)
```

In [163]:
```
X_test.shape
```

executed in 4ms, finished 14:34:17 2021-06-17

Out[163]:
```
(4736, 144)
```

9 Model Development

9.1 Logistic Regression

In [164]:
```
# Logistic model
log_clf = LogisticRegression(class_weight='balanced')
log_model = log_clf.fit(X_train, y_train)
y_pred_train = log_model.predict(X_train)
y_pred = log_model.predict(X_test)
```

executed in 237ms, finished 14:34:18 2021-06-17

In [165]:
```
#Confusion matrix for Logistic Regression
log_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:\n', log_matrix)
```

executed in 10ms, finished 14:34:18 2021-06-17

```
Confusion Matrix:
 [[3926  466]
 [  54  290]]
```

In [166]:
```
print('The f1 score for the training model is:',f1_score(y_train, y
print('The f1 score for the model is:',f1_score(y_test, y_pred))
```

executed in 13ms, finished 14:34:18 2021-06-17

```
The f1 score for the training model is: 0.5368209255533198
The f1 score for the model is: 0.5272727272727273
```

In [167]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 13ms, finished 14:34:18 2021-06-17

```
                  precision    recall  f1-score   support

normal grad rate       0.99      0.89      0.94      4392
   low grad rate       0.38      0.84      0.53       344

        accuracy                           0.89      4736
       macro avg       0.69      0.87      0.73      4736
    weighted avg       0.94      0.89      0.91      4736
```

9.1.1 Check for Overfit

Contents

In [168]:
```
# View confusion matrix for train data and predictions
confusion_matrix(y_train, y_pred_train)
```

executed in 14ms, finished 14:34:18 2021-06-17

Out[168]:
```
array([[9231, 1082],
       [  69,  667]])
```

In [169]:
```
# View confusion matrix for test data and predictions
confusion_matrix(y_test, y_pred)
```

executed in 11ms, finished 14:34:18 2021-06-17

Out[169]:
```
array([[3926,  466],
       [  54,  290]])
```

In [170]:
```
print(metrics.classification_report(y_train, y_pred_train, labels=[
                                    target_names=['normal grad rate
```

executed in 19ms, finished 14:34:18 2021-06-17

```
                   precision    recall  f1-score   support

normal grad rate       0.99      0.90      0.94     10313
   low grad rate       0.38      0.91      0.54       736

        accuracy                           0.90     11049
       macro avg       0.69      0.90      0.74     11049
    weighted avg       0.95      0.90      0.91     11049
```

In [171]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 12ms, finished 14:34:18 2021-06-17

```
                   precision    recall  f1-score   support

normal grad rate       0.99      0.89      0.94      4392
   low grad rate       0.38      0.84      0.53       344

        accuracy                           0.89      4736
       macro avg       0.69      0.87      0.73      4736
    weighted avg       0.94      0.89      0.91      4736
```

9.1.2 Model reiteration - parameter tuning

In [172]:
```
# Logistic model
log_clf = LogisticRegression( class_weight='balanced')
log_model = log_clf.fit(X_train, y_train)
y_pred_train = log_model.predict(X_train)
y_pred = log_model.predict(X_test)
```

executed in 194ms, finished 14:34:18 2021-06-17

In [173]:
```
model_params = {
    #Algorithm to use in the optimization problem
    'solver': ['newton-cg', 'sag', 'saga','lbfgs'],
    # Maximum number of iterations taken for the solvers to converg
    'max_iter': [10,50,100],
    # random state
    'random_state': [1]
}
```

executed in 3ms, finished 14:34:18 2021-06-17

Contents

In [174]:
```
# Instantiate and fit the LogisticReg Model
log_clf = LogisticRegression(class_weight='balanced')
log_clf = GridSearchCV(log_clf,model_params,scoring = 'recall')
log_clf.fit(X_train,y_train)
```

executed in 25.8s, finished 14:34:44 2021-06-17

Out[174]:
```
GridSearchCV(estimator=LogisticRegression(class_weight='balanced'),
             param_grid={'max_iter': [10, 50, 100], 'random_state':
                         'solver': ['newton-cg', 'sag', 'saga', 'lb
             scoring='recall')
```

In [175]:
```
LogisticRegression(class_weight='balanced').get_params().keys()
```

executed in 4ms, finished 14:34:44 2021-06-17

Out[175]:
```
dict_keys(['C', 'class_weight', 'dual', 'fit_intercept', 'intercept
s', 'n_jobs', 'penalty', 'random_state', 'solver', 'tol', 'verbose'
```

In [176]:
```
print(log_clf.best_estimator_.get_params())
```

executed in 17ms, finished 14:34:44 2021-06-17

```
{'C': 1.0, 'class_weight': 'balanced', 'dual': False, 'fit_intercep
one, 'max_iter': 50, 'multi_class': 'auto', 'n_jobs': None, 'penalt
'tol': 0.0001, 'verbose': 0, 'warm_start': False}
```

In [177]:
```
print('The f1 score for the training model is:',f1_score(y_train, y
print('The f1 score for the model is:',f1_score(y_test, y_pred))
```

executed in 11ms, finished 14:34:44 2021-06-17

```
The f1 score for the training model is: 0.5368209255533198
The f1 score for the model is: 0.5272727272727273
```

Contents

In [178]:
```
print(metrics.classification_report(y_train, y_pred_train, labels=[
                                    target_names=['normal grad rate
```

executed in 17ms, finished 14:34:44 2021-06-17

```
                   precision    recall  f1-score   support

normal grad rate       0.99      0.90      0.94     10313
   low grad rate       0.38      0.91      0.54       736

        accuracy                           0.90     11049
       macro avg       0.69      0.90      0.74     11049
    weighted avg       0.95      0.90      0.91     11049
```

In [179]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 12ms, finished 14:34:44 2021-06-17

```
                   precision    recall  f1-score   support

normal grad rate       0.99      0.89      0.94      4392
   low grad rate       0.38      0.84      0.53       344

        accuracy                           0.89      4736
       macro avg       0.69      0.87      0.73      4736
    weighted avg       0.94      0.89      0.91      4736
```

9.2 Random Forest Classifier

In [180]:
```
# Instantiate and fit the RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=10,class_weight='balanced')
rfc.fit(X_train,y_train)
```

executed in 247ms, finished 14:34:44 2021-06-17

Out[180]:
```
RandomForestClassifier(class_weight='balanced', n_estimators=10)
```

Contents

In [181]:
```
training_start = time.perf_counter()
rfc.fit(X_train, y_train)
training_end = time.perf_counter()
prediction_start = time.perf_counter()
y_pred = rfc.predict(X_test)
y_pred_train = rfc.predict(X_train)
prediction_end = time.perf_counter()
acc_rfc = (y_pred == y_test).sum().astype(float) / len(y_pred)*100
rfc_train_time = training_end-training_start
rfc_prediction_time = prediction_end-prediction_start
print("Scikit-Learn's Random Forest Classifier's prediction accurac
print("Time consumed for training: %4.3f seconds" % (rfc_train_time
print("Time consumed for prediction: %6.5f seconds" % (rfc_predicti
```

executed in 265ms, finished 14:34:44 2021-06-17

```
Scikit-Learn's Random Forest Classifier's prediction accuracy is: 9
Time consumed for training: 0.239 seconds
Time consumed for prediction: 0.02154 seconds
```

In [182]:
```
print('The f1 score for the training model is:',f1_score(y_train, y
print('The f1 score for the model is:',f1_score(y_test, y_pred))
```

executed in 11ms, finished 14:34:44 2021-06-17

```
The f1 score for the training model is: 0.9560906515580737
The f1 score for the model is: 0.5279383429672447
```

In [183]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 10ms, finished 14:34:44 2021-06-17

```
                  precision    recall  f1-score   support

normal grad rate       0.95      0.99      0.97      4392
   low grad rate       0.78      0.40      0.53       344

        accuracy                           0.95      4736
       macro avg       0.87      0.69      0.75      4736
    weighted avg       0.94      0.95      0.94      4736
```

9.2.1 Check for Overfit

In [184]:
```
# View confusion matrix for train data and predictions
confusion_matrix(y_train, y_pred_train)
```

executed in 11ms, finished 14:34:44 2021-06-17

Out[184]:
```
array([[10312,     1],
       [   61,   675]])
```

In [185]:
```
# View confusion matrix for test data and predictions
confusion_matrix(y_test, y_pred)
```

executed in 8ms, finished 14:34:44 2021-06-17

Out[185]:
```
array([[4354,   38],
       [ 207,  137]])
```

In [186]:
```
rfc_training_preds = rfc.predict(X_train)
rfc_training_f1 = f1_score(y_train, rfc_training_preds)

rfc_val_preds = rfc.predict(X_test) # y_hat
rfc_val_f1 = f1_score(y_test, rfc_val_preds)
print(rfc_training_f1)
print(rfc_val_f1)
```

executed in 31ms, finished 14:34:44 2021-06-17

```
0.9560906515580737
0.5279383429672447
```

In [187]:
```
print(metrics.classification_report(y_train, y_pred_train, labels=[
                                    target_names=['normal grad rate
```

executed in 16ms, finished 14:34:44 2021-06-17

```
                  precision    recall  f1-score   support

normal grad rate       0.99      1.00      1.00     10313
   low grad rate       1.00      0.92      0.96       736

        accuracy                           0.99     11049
       macro avg       1.00      0.96      0.98     11049
    weighted avg       0.99      0.99      0.99     11049
```

In [188]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate'
```

executed in 12ms, finished 14:34:45 2021-06-17

```
                  precision    recall  f1-score   support

normal grad rate       0.95      0.99      0.97      4392
   low grad rate       0.78      0.40      0.53       344

        accuracy                           0.95      4736
       macro avg       0.87      0.69      0.75      4736
    weighted avg       0.94      0.95      0.94      4736
```

Definitely overfit we see this especially in the low grad rate class

9.2.2 Feature Importance

Contents

In [189]:
```
eed to make a list of feature names for feat importance graph
eed function since data is in numpy arrays
 get_column_names_from_ColumnTransformer(column_transformer):
 col_name = []
 for transformer_in_columns in column_transformer.transformers_[:-1
     raw_col_name = transformer_in_columns[2]
     if isinstance(transformer_in_columns[1],Pipeline):
         transformer = transformer_in_columns[1].steps[-1][1]
     else:
         transformer = transformer_in_columns[1]
     try:
         names = transformer.get_feature_names()
     except AttributeError: # if no 'get_feature_names' function, u
         names = raw_col_name
     if isinstance(names,np.ndarray): # eg.
         col_name += names.tolist()
     elif isinstance(names,list):
         col_name += names
     elif isinstance(names,str):
         col_name.append(names)
 return col_name
```

executed in 6ms, finished 14:34:45 2021-06-17

In [190]:
```
col_name_num = get_column_names_from_ColumnTransformer(col_transfor
```

executed in 3ms, finished 14:34:45 2021-06-17

In [191]:
```
col_name_cat = list(col_transformer.named_transformers_['cats'].nam
     .get_feature_names())
```

executed in 3ms, finished 14:34:45 2021-06-17

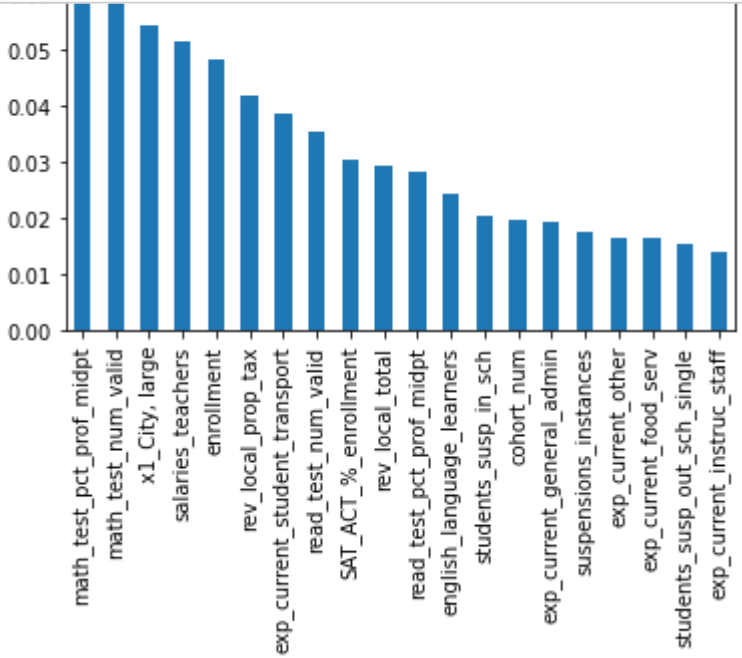
In [192]:
```
rfc_columns = col_name_num + col_name_cat
```

executed in 2ms, finished 14:34:45 2021-06-17

Contents

In [193]:
```
feat_importances = pd.Series(rfc.feature_importances_, index = rfc_
feat_importances.nlargest(20).plot(kind='bar')
```

executed in 228ms, finished 14:34:45 2021-06-17



9.2.3 Model reiteration - parameter tuning

Contents

In [194]:
```
model_params = {
    # number of
    'n_estimators': [5,10,20,50,100,200],
    # number of max features
    'max_features': [10,15,20,50,100],
    # max number of levels in each decision tree
    'max_depth': [5,10,15],
    # minimum amount of samples in a node
    'min_samples_leaf': [10],
    # minimum amount of samples to split
    'min_samples_split' : [100,1000],
    # random state
    'random_state': [1]

}
```

executed in 3ms, finished 14:34:45 2021-06-17

In [195]:
```
# Instantiate and fit the RandomForestClassifier
rfc=RandomForestClassifier(class_weight='balanced')
rfc = GridSearchCV(rfc, model_params, scoring = 'recall')
rfc.fit(X_train,y_train)
```

executed in 35m 3s, finished 15:09:48 2021-06-17

Out[195]:
```
GridSearchCV(estimator=RandomForestClassifier(class_weight='balance
             param_grid={'max_depth': [5, 10, 15],
                         'max_features': [10, 15, 20, 50, 100],
                         'min_samples_leaf': [10],
                         'min_samples_split': [100, 1000],
                         'n_estimators': [5, 10, 20, 50, 100, 200],
                         'random_state': [1]},
             scoring='recall')
```

In [196]:
```
print(rfc.best_estimator_.get_params())
```

executed in 6ms, finished 15:09:48 2021-06-17

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': 'balanced', '
res': 100, 'max_leaf_nodes': None, 'max_samples': None, 'min_impuri
'min_samples_leaf': 10, 'min_samples_split': 1000, 'min_weight_frac
None, 'oob_score': False, 'random_state': 1, 'verbose': 0, 'warm_st
```

Contents

In [197]:
```
print('The f1 score for the training model is:',f1_score(y_train, y
print('The f1 score for the model is:',f1_score(y_test, y_pred))
```

executed in 13ms, finished 15:09:48 2021-06-17

```
The f1 score for the training model is: 0.9560906515580737
The f1 score for the model is: 0.5279383429672447
```

In [198]:
```
print(metrics.classification_report(y_train, y_pred_train, labels=[
                                    target_names=['normal grad rate
```

executed in 16ms, finished 15:09:48 2021-06-17

	precision	recall	f1-score	support
normal grad rate	0.99	1.00	1.00	10313
low grad rate	1.00	0.92	0.96	736
accuracy			0.99	11049
macro avg	1.00	0.96	0.98	11049
weighted avg	0.99	0.99	0.99	11049

In [199]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 37ms, finished 15:09:48 2021-06-17

	precision	recall	f1-score	support
normal grad rate	0.95	0.99	0.97	4392
low grad rate	0.78	0.40	0.53	344
accuracy			0.95	4736
macro avg	0.87	0.69	0.75	4736
weighted avg	0.94	0.95	0.94	4736

9.3 Gradient Boosting Classifier

Contents

In [200]:
```
gb_clf = GradientBoostingClassifier()
gb_clf.fit(X_train, y_train)
y_pred = gb_clf.predict(X_test)
y_pred_train = gb_clf.predict(X_train)

# print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_
# print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(
```

executed in 10.6s, finished 15:09:59 2021-06-17

In [201]:
```
print('The f1 score for the training model is:',f1_score(y_train, y
print('The f1 score for the model is:',f1_score(y_test, y_pred))
```

executed in 23ms, finished 15:09:59 2021-06-17

```
The f1 score for the training model is: 0.7556596409055425
The f1 score for the model is: 0.6079447322970639
```

In [202]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 10ms, finished 15:09:59 2021-06-17

```
                  precision    recall  f1-score   support

normal grad rate       0.96      0.99      0.97      4392
   low grad rate       0.75      0.51      0.61       344

        accuracy                           0.95      4736
       macro avg       0.86      0.75      0.79      4736
    weighted avg       0.95      0.95      0.95      4736
```

9.3.1 Check for Overfit

In [203]:
```
# View confusion matrix for train data and predictions
confusion_matrix(y_train, y_pred_train)
```

executed in 13ms, finished 15:09:59 2021-06-17

Out[203]:
```
array([[10252,    61],
       [  252,   484]])
```

Contents

In [204]:
```
# View confusion matrix for test data and predictions
confusion_matrix(y_test, y_pred)
```

executed in 8ms, finished 15:09:59 2021-06-17

Out[204]:
```
array([[4333,   59],
       [ 168,  176]])
```

In [205]:
```
print(metrics.classification_report(y_train, y_pred_train, labels=[
                                    target_names=['normal grad rate
```

executed in 16ms, finished 15:09:59 2021-06-17

```
                   precision    recall  f1-score   support

normal grad rate       0.98      0.99      0.98     10313
   low grad rate       0.89      0.66      0.76       736

        accuracy                           0.97     11049
       macro avg       0.93      0.83      0.87     11049
    weighted avg       0.97      0.97      0.97     11049
```

In [206]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 11ms, finished 15:09:59 2021-06-17

```
                   precision    recall  f1-score   support

normal grad rate       0.96      0.99      0.97      4392
   low grad rate       0.75      0.51      0.61       344

        accuracy                           0.95      4736
       macro avg       0.86      0.75      0.79      4736
    weighted avg       0.95      0.95      0.95      4736
```

9.3.2 Model reiteration - parameter tuning

Contents

In [207]:
```
model_params = {
    # number of boosting stages
    'n_estimators': [5,10,20,50,100,200],
    # number of max features
    'max_features': [10,15,20,50,100],
     # minimum amount of samples in a node
    'min_samples_leaf': [10],
    # Learning rate
    'learning_rate':[.25,.5,.75,1],
    #The minimum number of samples required to split an internal no
    'min_samples_split' : [100,1000],
     # random state
    'random_state':[1]
}
```

executed in 3ms, finished 15:09:59 2021-06-17

In [208]:
```
gb_clf = GradientBoostingClassifier()
gb_clf = GridSearchCV(gb_clf,model_params, scoring = 'recall')
gb_clf.fit(X_train,y_train)
```

executed in 30m 25s, finished 15:40:24 2021-06-17

Out[208]:
```
GridSearchCV(estimator=GradientBoostingClassifier(),
             param_grid={'learning_rate': [0.25, 0.5, 0.75, 1],
                         'max_features': [10, 15, 20, 50, 100],
                         'min_samples_leaf': [10],
                         'min_samples_split': [100, 1000],
                         'n_estimators': [5, 10, 20, 50, 100, 200],
                         'random_state': [1]},
             scoring='recall')
```

Contents

In [209]:
```
print(metrics.classification_report(y_train, y_pred_train, labels=[
                                    target_names=['normal grad rate
```

executed in 27ms, finished 15:40:24 2021-06-17

```
                    precision    recall  f1-score   support

normal grad rate       0.98      0.99      0.98     10313
   low grad rate       0.89      0.66      0.76       736

        accuracy                           0.97     11049
       macro avg       0.93      0.83      0.87     11049
    weighted avg       0.97      0.97      0.97     11049
```

In [210]:
```
print(metrics.classification_report(y_test, y_pred, labels=[0,1],
                                    target_names=['normal grad rate
```

executed in 11ms, finished 15:40:24 2021-06-17

```
                    precision    recall  f1-score   support

normal grad rate       0.96      0.99      0.97      4392
   low grad rate       0.75      0.51      0.61       344

        accuracy                           0.95      4736
       macro avg       0.86      0.75      0.79      4736
    weighted avg       0.95      0.95      0.95      4736
```

Contents

In [212]:
```
lr_list = [0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_r
    gb_clf.fit(X_train, y_train)

    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.scor
    print('The f1 score for the training model is:',f1_score(y_trai
    print('The f1 score for the model is:',f1_score(y_test, y_pred)
```

executed in 16.4s, finished 15:42:51 2021-06-17

```
Learning rate:  0.05
Accuracy score (training): 0.942
Accuracy score (validation): 0.938
The f1 score for the training model is: 0.7556596409055425
The f1 score for the model is: 0.6079447322970639
Learning rate:  0.075
Accuracy score (training): 0.949
Accuracy score (validation): 0.942
The f1 score for the training model is: 0.7556596409055425
The f1 score for the model is: 0.6079447322970639
Learning rate:  0.1
Accuracy score (training): 0.953
Accuracy score (validation): 0.945
The f1 score for the training model is: 0.7556596409055425
The f1 score for the model is: 0.6079447322970639
Learning rate:  0.25
Accuracy score (training): 0.962
Accuracy score (validation): 0.948
The f1 score for the training model is: 0.7556596409055425
```

10 Conclusions

- Our modeling struggles with overfitting and is better at classifying the majority class (high enough performance that we feel confident in making recommendations based o our modeling.
- The number of students who completed math and reading tests and received a profic

- The number of students participating in ACT/SAT tests was a strong indicator for our normal graduation rate group.
- Higher % of total salary devoted to teacher salaries was an indicator of normal high s

11 Recommendations

- Mathematics and Reading: Focus resources on providing extra support for mathemat to improve these scores whether it be through offering remedial opportunities additio areas.
- ACT/SAT: As with many education metrics ACT/SAT participation likely also has a rela However, regardless of other factors ACT/SAT prep can be a motivating factor for stu goal for students in high school is to prepare themselves so they will be accepted int
- Teacher Salaries:
  - This feature may be confounded by other factors. For example a school with less workers if they are unable to receive funding for newer facilities.
  - We do not recommend making teacher salaries a higher proportion in budgeting factors.

12 Future Work

- Perform feature selection steps to improve overfit of the modeling.
- Investigate adding new features. Now that we have a better handle on which types of features from our reporting sources and also engineering new features with the data
- Investigate which types of schools did not report graduation rates. We need to know graduation rates. Is the type of schools not reporting graduation rates similar to those

Contents