Data_Cleaning

August 7, 2024

1 Data Preprocessing

The below blocks of code generate the csv's necessary to conduct the analysis.

It compiles the spend per student, total budget, mean scale test scores, and % meets or exceeds expectations for all students grades 3-8 from 2019 - 2023 (omitting 2020).

1. The first block generates the csv "schools_budget_data.csv". It scrapes the public database for the spend data.
2. The second block generates the csv "all_school_data_2019_2023.csv". It compiles all data from the 4 spreadsheets and the web scraper into one csv.

Of note, the first block takes several minutes to run, and should be done sparingly. The second block also takes a few minutes to run.

[1]:
```
import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup
```

[9]:
```
## THIS CODE BLOCK GENERATES THE SPEND DATA FOR EACH SCHOOL

## This is a web scraper that consolidates the historical spend data for␣
 ↪Colorado schools at the school level
## This takes forever to run, so just use the csv.

## Not all school codes are 4 digits, so this takes any digit under four and␣
 ↪appends 00s to the front as needed
def fourDigitSchoolCode(code):
    codeString = str(code)
    fill = '0000'
    if (len(codeString) == 4):
        return codeString
    return fill[0 : (4 - len(codeString))] + codeString

## Historical spend data per Student/School
def getPricing(districtCode, schoolCode):
    DCode = fourDigitSchoolCode(districtCode)
    SCode = fourDigitSchoolCode(schoolCode)
```

```
    cost_url = 'https://www.cde.state.co.us/schoolview/financialtransparency/historical/'+str(DCode)+'/'+str(SCode)
    page = requests.get(cost_url)
    soup = BeautifulSoup(page.content, "html.parser")
    results = soup.find(id="maincontent")
    infoLinks = results.findAll("div", class_="infoLink")
    perStudentPricing = []
    for i in infoLinks:
        data = i.find("strong", class_="txt-large")
        if data:
            perStudentPricing.append(data.text.strip().replace('$','').replace(',',''))
    ## Only the first four entries are specific to the school. The others are state averages
    perStudentPricing = perStudentPricing[0:4]

    ## Now get the school budget
    totalSpendingDivs = results.findAll("div", class_="insetCta-inner theme-bg-0 theme-before-bg-0 theme-after-bg-0")
    totalSpending = []
    for i in totalSpendingDivs:
        data = i.find("div", class_="dataPoint-value").text.strip().replace('$','').replace(',','')
        totalSpending.append(data)
    ## Returns 2 arrays. The first is the per student for 2023, 2022, 2021, and 2020
    ## The second is the total school budget, for the same years
    return [*perStudentPricing, *totalSpending]

Results_2023 = pd.read_csv('csvs/2023 CMAS ELA and Math District and School Summary Achievement Results.csv', sep=",", header=0)
Results_2023 = Results_2023[(Results_2023['Level'] == 'SCHOOL') & (Results_2023['Content'] == 'Mathematics') & (Results_2023['Grade'] == 'All Grades')]

df = pd.DataFrame()

School_Budgets = pd.read_csv('schools_budget_data.csv', sep=",", header=0)
print(School_Budgets[School_Budgets.columns[0]])
# df[['2023 Spend Per Student', '2022 Spend Per Student', '2021 Spend Per Student', '2020 Spend Per Student', 'Total Spending 2023', 'Total Spending 2022', 'Total Spending 2021', 'Total Spending 2020']]=Results_2023.apply(lambda x: getPricing(x['District Code'], x['School Code']),axis=1,result_type='expand')
# df.to_csv('schools_budget_data.csv')
```

```
0          2690
```

```
1         2699
2         2716
3         2732
4         2746
         ...
1513     16927
1514     16937
1515     16945
1516     16956
1517     16970
Name: Unnamed: 0, Length: 1518, dtype: int64
```

[17]:
```
## This code block creates the full csv with all data from 2019 - 2023␣
 ↪(omitting 2020)
def remove_nulls(dataItem):
    # check the data type. If a string, remove the dollar sign
    # and any N/As
    if type(dataItem) is str:
        return(dataItem.replace('- -', 'N/A'))
    return(dataItem)

def getGradePerformance(districtCode, schoolCode, grade, Results_2023,␣
 ↪Results_2022, Results_2021, Results_2019):
    performance_2023 = 'N/A'
    performance_2022 = 'N/A'
    performance_2021 = 'N/A'
    performance_2019 = 'N/A'
    mean_scale_score_2023 = 'N/A'
    mean_scale_score_2022 = 'N/A'
    mean_scale_score_2021 = 'N/A'
    mean_scale_score_2019 = 'N/A'
    result_2023 = Results_2023[(Results_2023['District Code'] == districtCode)␣
 ↪& (Results_2023['School Code'] == schoolCode) & (Results_2023['Grade'] ==␣
 ↪grade)]
    result_2022 = Results_2022[(Results_2022['District Code'] == districtCode)␣
 ↪& (Results_2022['School Code'] == schoolCode) & (Results_2022['Grade'] ==␣
 ↪grade)]
    result_2021 = Results_2021[(Results_2021['District Code'] ==␣
 ↪int(districtCode)) & (Results_2021['School Code'] == int(schoolCode)) &␣
 ↪(Results_2021['Grade'] == int(grade))]
    result_2019 = Results_2019[(Results_2019['District Code'] == districtCode)␣
 ↪& (Results_2019['School Code'] == schoolCode) & (Results_2019['Grade'] ==␣
 ↪grade)]
    if not result_2023.empty:
        ## If the Number of Valid Scores does not meet the threshold, we have␣
 ↪to ignore it
        validScores = result_2023['Number of Valid Scores'].values[0]
```

```
        value = result_2023['2023'].apply(remove_nulls)
        mean_val = result_2023['Mean Scale Score'].apply(remove_nulls)
        if validScores != '- -' and validScores != '< 16' and (value.values[0] !
↪= 'N/A'):
            performance_2023 = (value.astype('float')).array[0]
        if mean_val.values[0] != 'N/A':
            mean_scale_score_2023 = (mean_val.astype('float')).array[0]

    if not result_2022.empty:
        validScores = result_2022['Number of Valid Scores'].values[0]
        value = result_2022['2022'].apply(remove_nulls)
        mean_val = result_2022['Mean Scale Score'].apply(remove_nulls)
        if validScores != '- -' and validScores != '< 16' and (value.values[0] !
↪= 'N/A'):
            performance_2022 = (value.astype('float')).array[0]
        if mean_val.values[0] != 'N/A':
            mean_scale_score_2022 = (mean_val.astype('float')).array[0]

    if not result_2021.empty:
        validScores = result_2021['Number of Valid Scores'].values[0]
        value = result_2021['Percent Met or Exceeded Expectations'].
↪apply(remove_nulls)
        mean_val = result_2021['Mean Scale Score'].apply(remove_nulls)
        if validScores != '- -' and validScores != '< 16' and (value.values[0] !
↪= 'N/A'):
            performance_2021 = (value.astype('float')).array[0]
        if mean_val.values[0] != 'N/A':
            mean_scale_score_2021 = (mean_val.astype('float')).array[0]

    if not result_2019.empty:
        validScores = result_2019['Number of Valid Scores'].values[0]
        value = result_2019['Percent Met or Exceeded Expectations'].
↪apply(remove_nulls)
        mean_val = result_2019['Mean Scale Score'].apply(remove_nulls)
        if validScores != '- -' and validScores != '< 16' and (value.values[0] !
↪= 'N/A'):
            performance_2019 = (value.astype('float')).array[0]
        if mean_val.values[0] != 'N/A':
            mean_scale_score_2019 = (mean_val.astype('float')).array[0]

    return [performance_2023, performance_2022, performance_2021,␣
↪performance_2019, mean_scale_score_2023, mean_scale_score_2022,␣
↪mean_scale_score_2021, mean_scale_score_2019]

## Now we need to get all the data from 2019 - 2022 and add it all in
## Need to extend the getGradePerformance function to do this
```

```
Results_2019 = pd.read_csv('csvs/2019 CMAS ELA MATH District and School␣
 ↪Achievement Results.csv', sep=",", header=0)
Results_2019 = Results_2019[(Results_2019['Level'] == 'SCHOOL') &␣
 ↪(Results_2019['Subject'] == 'Mathematics')]

Results_2021 = pd.read_csv('csvs/2021 CMAS ELA and Math District and School␣
 ↪Summary Achievement Results - Required Tests.csv', sep=",", header=0)
Results_2021 = Results_2021[(Results_2021['Level'] == 'SCHOOL') &␣
 ↪(Results_2021['Content'] == 'Mathematics')]

Results_2022 = pd.read_csv('csvs/2022 CMAS ELA and Math District and School␣
 ↪Summary Achievement Results.csv', sep=",", header=0)
Results_2022 = Results_2022[(Results_2022['Level'] == 'SCHOOL') &␣
 ↪(Results_2022['Content'] == 'Mathematics')]

Results_2023 = pd.read_csv('csvs/2023 CMAS ELA and Math District and School␣
 ↪Summary Achievement Results.csv', sep=",", header=0)
Results_2023 = Results_2023[(Results_2023['Level'] == 'SCHOOL') &␣
 ↪(Results_2023['Content'] == 'Mathematics')]

School_Budgets = pd.read_csv('schools_budget_data.csv', sep=",", header=0)

def getMyPrice(index, df):
    result = df[(df[df.columns[0]] == index)].values.flatten().tolist()[1:]
    if len(result) != 0:
        return result
    return [0,0,0,0,0,0,0,0]

## Now append the spend data to the 2023 results csv
Results_2023[[
    '2023 Spend Per Student',
    '2022 Spend Per Student',
    '2021 Spend Per Student',
    '2019 Spend Per Student',
    '2023 Total Budget',
    '2022 Total Budget',
    '2021 Total Budget',
    '2019 Total Budget']] = Results_2023.apply(lambda x: getMyPrice(x.name,␣
 ↪School_Budgets), axis=1,result_type="expand")

## Now, iterate through all the grades and add them to the master list
grades = ['3','4','5','6','7','8']
for grade in grades:
    Results_2023[[
        '2023 Grade '+grade+' meets or exceeds expectations',
        '2022 Grade '+grade+' meets or exceeds expectations',
```

```
        '2021 Grade '+grade+' meets or exceeds expectations',
        '2019 Grade '+grade+' meets or exceeds expectations',
        '2023 Grade '+grade+' Mean Scale Score',
        '2022 Grade '+grade+' Mean Scale Score',
        '2021 Grade '+grade+' Mean Scale Score',
        '2019 Grade '+grade+' Mean Scale Score']] = Results_2023.apply(lambda x:
 ↪ getGradePerformance(
            x['District Code'],
            x['School Code'],
            grade,
            Results_2023,
            Results_2022,
            Results_2021,
            Results_2019), axis=1,result_type='expand')
Finals_2023 = Results_2023[(Results_2023['Level'] == 'SCHOOL') &␣
 ↪(Results_2023['Content'] == 'Mathematics') & (Results_2023['Grade'] == 'All␣
 ↪Grades')]
Finals_2023.to_csv('all_school_data_2019_2023.csv')
```