

Michael Perrine

DSC 550 Data Mining

Professor Werner

Final Assignment

Milestone 1

Analysis of Population Data

Milestone 1

What do demographics, economic growth and school ratings have in common? These factors are important to developers. The cost of building in the wrong location can cost millions in losses. The goal of this analysis is to determine locations of interest for land development. We believe Texas is an attractive place to develop land for the purpose of economic growth. To validate this assumption, I will explore several factors that make a location attractive. These factors are the quality of schools, economic development, and population growth.

To perform my analysis, I will use several libraries found in the python program. Some of the libraries are Pandas, Matplotlib, NumPy, Scikit learn, and Seaborn among others. These libraries will give me the ability to perform my preliminary analysis. The data will come from the Texas Open Data Portal: State of Texas | Open Data Portal | Open Data Portal. This website has csv files for all the data of interest. To begin the project, I will upload the files into my Python program. I will need to perform some data analysis to uncover relationships such as what factors stimulate population growth? What factors drive business? What impact does a good school play in migration decisions? How can I use this information to predict areas of growth?

Some of the work that I will perform will be exploratory data analysis. I will need to transform and clean the data sets. Once I clean the data it will be ready for the visualization process. Visualizing the data is another important tool to analyze and interpret data. This includes building graphs such as histograms, scatter plots, and box plots. These are just some of the visualization tools available.

After I build my graphs, I can determine the relationships involved. The next step is to build a model. Scikit Learn is the library that will build the model. This is probably the most

important step in the process because it gives me the ability to make predictions and take actionable steps in determining what areas are prime investment opportunities.

There are challenges to my analysis. I don't know what relationships if any I will find. If I do find relationships in the data, I'm not sure how strong they are or if they will lead to a viable investment decision.

```
In [1]: # import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
In [2]: pd.options.display.max_columns = None
```

1. The first step is to upload the data and to build some graphs to visualize the data

```
In [3]: # This code imports the school data and displays the first 10 rows
school = pd.read_csv(r"texas school data.csv")
school.head(3)
```

Out[3]:

	District\nNumber	District	Campus\nNumber	Campus	Region	County	School\n'
0	1902	CAYUGA ISD	NaN	NaN	REGION 07: KILGORE	ANDERSON	Di:
1	1902	CAYUGA ISD	1902001.0	CAYUGA H S	REGION 07: KILGORE	ANDERSON	High Sc
2	1902	CAYUGA ISD	1902041.0	CAYUGA MIDDLE	REGION 07: KILGORE	ANDERSON	Middle Sc

```
In [4]: # This code drops rows with NaN values
school_1 = school[(~school.isnull()).all(axis=1)]
school_1.head(3)
```

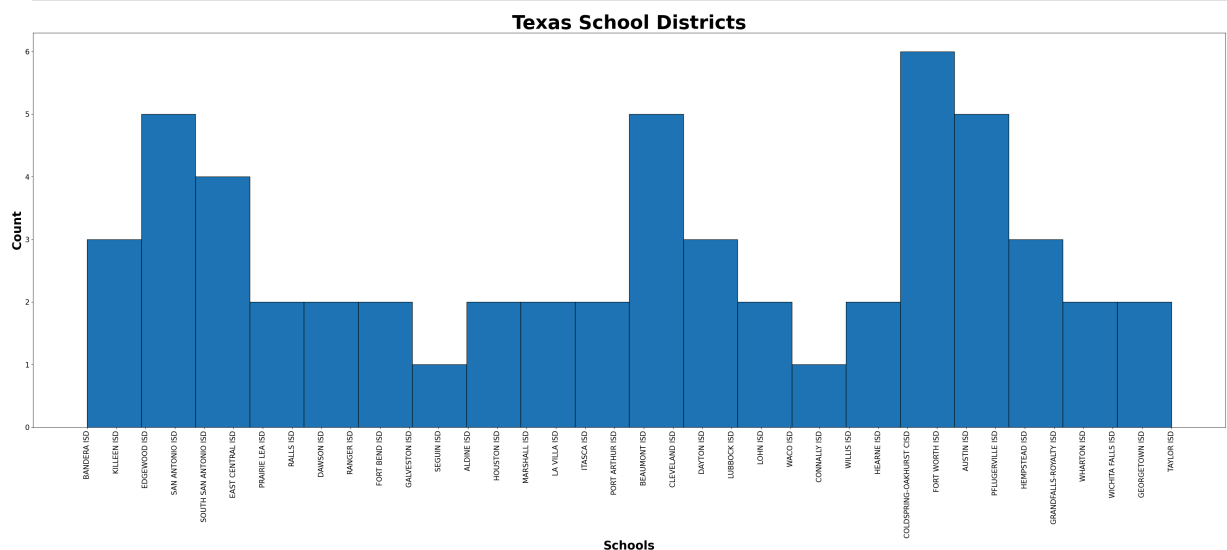
Out[4]:

District\nNumber	District	Campus\nNumber	Campus	Region	County	Scho
159	10902 BANDERA ISD	10902001.0	BANDERA H S	REGION 20: SAN ANTONIO	BANDERA	Hi
278	14906 KILLEEN ISD	14906044.0	MANOR MIDDLE	REGION 12: WACO	BELL	Midc
280	14906 KILLEEN ISD	14906048.0	PALO ALTO MIDDLE	REGION 12: WACO	BELL	Midc

In [5]: `# This code shows the dimensions of the school data`
`school_1.shape`

Out[5]: (56, 41)

In [6]: `# This code displays a histogram of the school data`
`plt.figure(figsize=(45,15))`
`plt.hist(x = school_1["District"], bins=20, edgecolor = "black")`
`plt.title("Texas School Districts", fontsize=40, weight='bold')`
`plt.xlabel("Schools", fontsize=25, weight='bold')`
`plt.ylabel("Count", fontsize=25, weight='bold')`
`plt.xticks(rotation = 90, fontsize= 15)`
`plt.yticks(fontsize= 15)`
`plt.show()`



In this first graph I want to visualize the total schools in each district. This is important because larger school districts may have a correlation to population growth and I expect this will play a role in the decision to move to a certain area. After viewing this data I noticed that

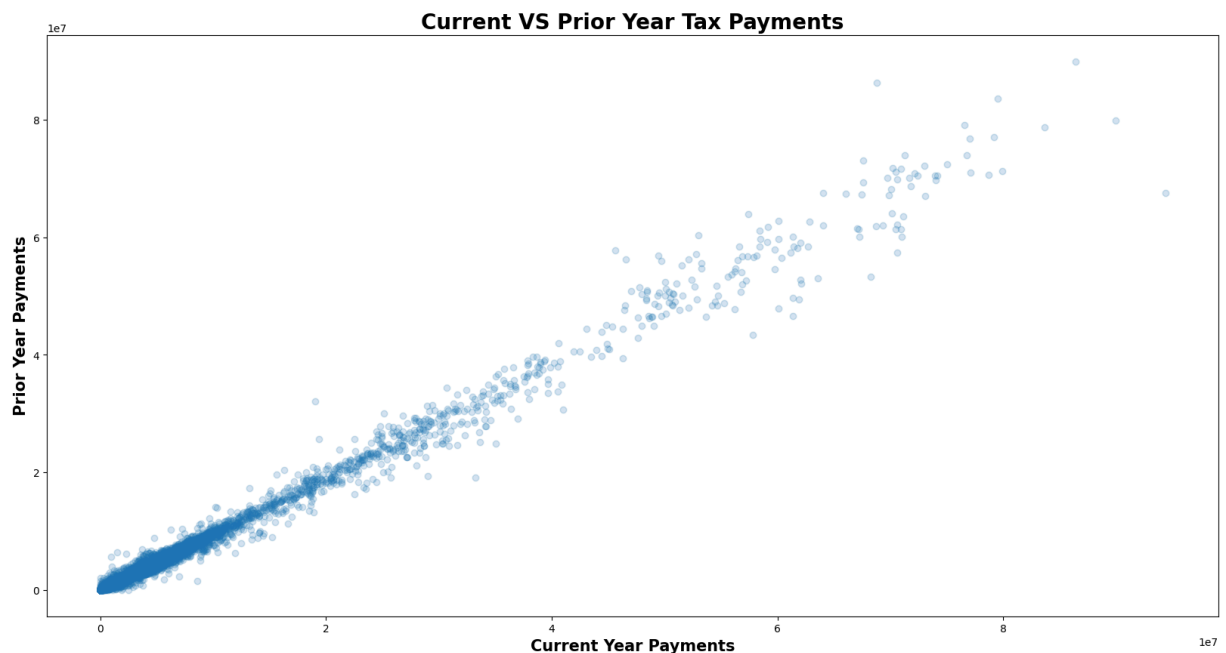
the Cold Spring/Oakhurst and Ft Worth school districts are the largest districts in Texas. These districts are coming in at approximately 120 schools respectively.

```
In [7]: # This code imports the sales tax allocation data set and displays the first 5 rows
sales_tax = pd.read_csv(r"Sales_Tax_Allocation_City_20250420.csv")
sales_tax.head(3)
```

Out[7]:

	City	Net Payment This Period	Comparable Payment Prior Year	Percent Change From Prior Year	Payment to Date	Previous Payments to Date	Percent Change To Date	Report Month
0	Abbott	14301.88	15919.41	-10.16	146889.53	152146.26	-3.45	11
1	Abernathy	23516.10	23987.13	-1.96	270303.69	256849.99	5.23	11
2	Abilene	5526055.82	5477084.93	0.89	57014074.04	53887697.25	5.80	11

```
In [8]: # This code creates a scatter plot showing the comparison of the current year and p
plt.figure(figsize = (20,10))
plt.scatter(x = 'Net Payment This Period',y = 'Comparable Payment Prior Year', data
plt.title("Current VS Prior Year Tax Payments", fontsize=20, weight='bold')
plt.xlabel("Current Year Payments", fontsize=15, weight='bold' )
plt.ylabel("Prior Year Payments", fontsize=15, weight='bold')
plt.show()
```



In this graph we are comparing current and prior year sales taxes paid. This is an important metric. Texans don't pay state income tax. All their revenue is derived from sales tax. This will indicate the areas where more spending takes place. The thought is areas that are spending more will have more affluent residents and will be more attractive to individuals when they decide to move.

```
In [9]: # This code imports the population data set and displays the first 5 rows
pop = pd.read_csv(r"population data.csv")
pop.head()
```

```
Out[9]:
```

	migration_scenario	year	year_month	FIPS	area_name	age_in_yrs_num	age_in_yrs_cha
0	Mid	2020	202004	0	State of Texas	-1	All Age
1	Mid	2020	202004	0	State of Texas	0	< 1 y
2	Mid	2020	202004	0	State of Texas	1	1 y
3	Mid	2020	202004	0	State of Texas	2	2 yr
4	Mid	2020	202004	0	State of Texas	3	3 yr

```
In [10]: # This code creates a subset of the population data and removes the first row which
# respective columns. It also displays the first 5 rows of the data set
pop_sub = pop[['year', 'total_male', 'total_female', 'total' ]]
pop_sub = pop_sub.iloc[1: ]
pop_sub.head()
```

```
Out[10]:
```

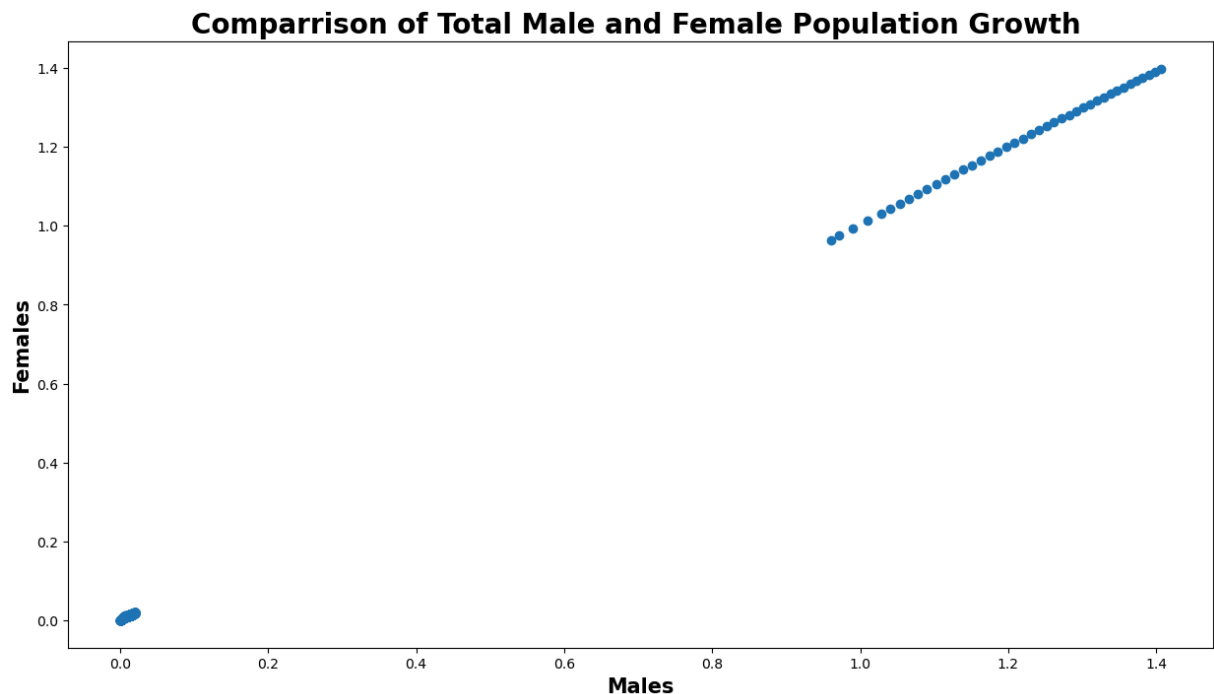
	year	total_male	total_female	total
1	2020	173553	167009	340562
2	2020	176404	170602	347006
3	2020	183499	177691	361190
4	2020	192237	185157	377394
5	2020	199902	193206	393108

```
In [11]: # This code adds two new columns to show percentage change
# between the respective columns
pop_sub['total_male_Percentage'] = pop_sub['total_male'].apply(lambda x: (x / pop_s
pop_sub['total_female_Percentage'] = pop_sub['total_female'].apply(lambda x: (x / p
pop_sub.head()
```

```
Out[11]:
```

	year	total_male	total_female	total	total_male_Percentage	total_female_Percentage
1	2020	173553	167009	340562	0.011549	0.010881
2	2020	176404	170602	347006	0.011738	0.011115
3	2020	183499	177691	361190	0.012211	0.011577
4	2020	192237	185157	377394	0.012792	0.012063
5	2020	199902	193206	393108	0.013302	0.012588

```
In [12]: # This code creates a scatter plot comparing total male and female populations
plt.figure(figsize = (15,8))
plt.scatter(x = 'total_male_Percentage',y = 'total_female_Percentage', data = pop_s
plt.title("Comparrison of Total Male and Female Population Growth ", fontsize=20, w
plt.xlabel("Males", fontsize=15, weight='bold' )
plt.ylabel("Females", fontsize=15, weight='bold')
plt.show()
```



The purpose of this graph is to compare the total growth between male and females in Texas. The graph didn't turn out as expected. I tried different variations and inputs but with no success. I need to spend more time on this data set. It could be a factor of the age break down in the data.

```
In [13]: # This code imports the data set into a pandas data frame and displays the first 10
key_eco = pd.read_csv(r"Key_Economic_Indicators_20250420.csv")

key_eco.head(10)
```

Out[13]:

	Month	Year	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.	U.S. F En
0	1	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
1	2	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
2	3	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
3	4	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
4	5	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
5	6	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
6	7	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
7	8	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
8	9	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
9	10	2005	NaN	NaN	NaN	NaN	NaN	NaN	I

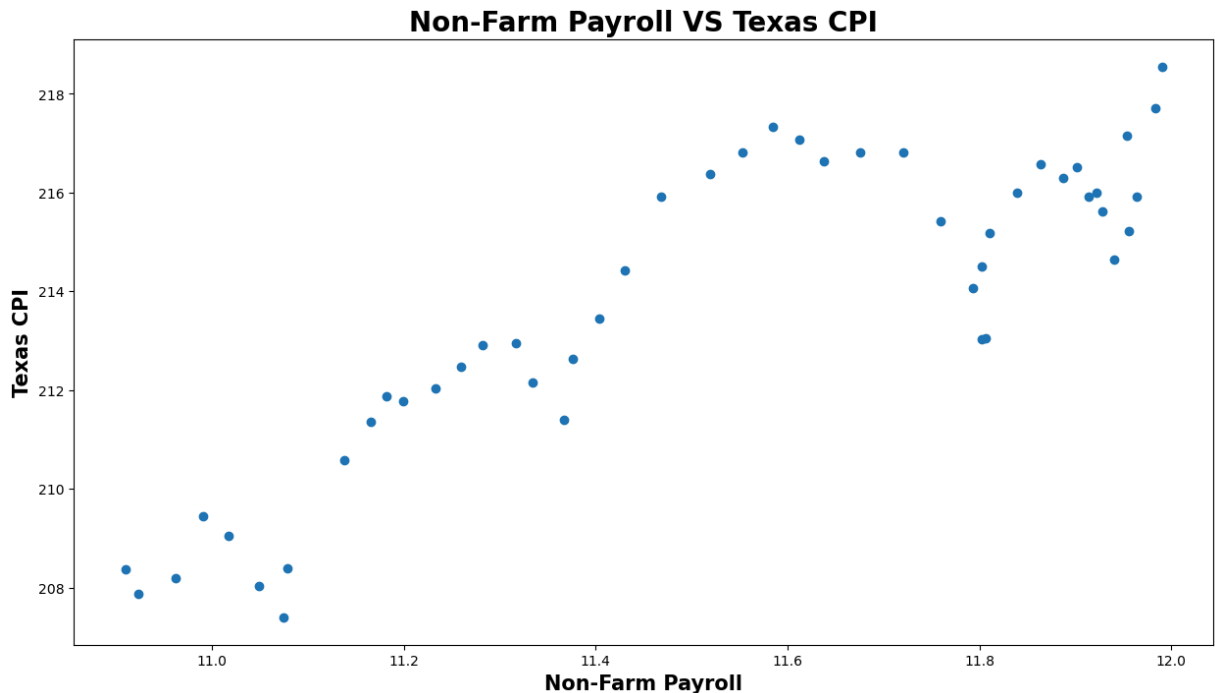
```
In [14]: # This code drops the NaN values in the data set
key_eco_1 = key_eco.dropna()
key_eco_1.head()
```

Out[14]:

	Month	Year	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.	U
89	6	2012	104.8	95.6	62.7	105.844	208.372	229.478	22
90	7	2012	93.7	92.6	65.4	105.880	207.881	229.104	22
91	8	2012	75.1	74.9	61.3	106.238	208.201	230.379	23
92	9	2012	83.7	78.5	68.4	106.576	209.451	231.407	23
93	10	2012	92.0	84.2	73.1	106.886	209.044	231.317	23

```
In [15]: #This code create a scatter plot for Non-farm Payroll and Texas CPI
plt.figure(figsize = (15,8))
plt.scatter(x = 'Nonfarm Employment TX',y = 'Consumer Price Index TX', data = key_e
plt.title("Non-Farm Payroll VS Texas CPI", fontsize=20, weight='bold')
plt.xlabel("Non-Farm Payroll", fontsize=15, weight='bold' )
```

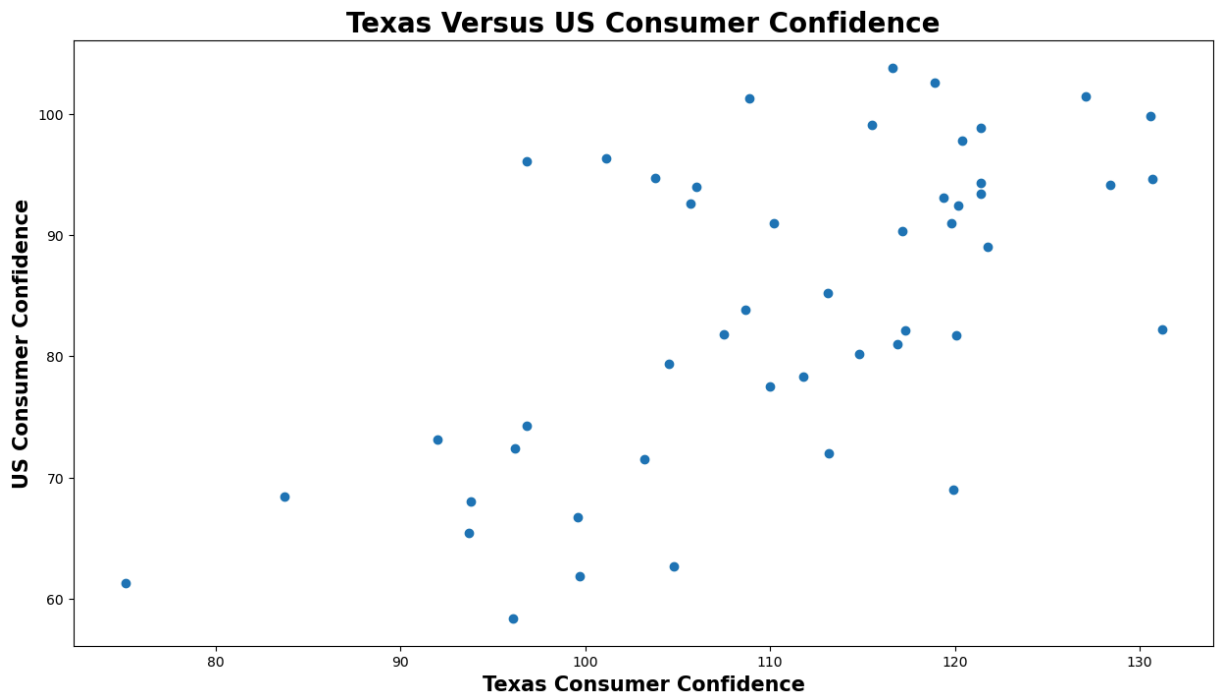
```
plt.ylabel("Texas CPI", fontsize=15, weight='bold')
plt.show()
```



This graph shows a comparison between non-farm payroll and the Texas CPI. Understanding the correlation between these two features is important because it will help to uncover the economic condition of the cities within Texas. How well do these respective economies respond to changes in price. Is there a correlation between population growth and the elasticity of the economy

```
In [16]: sub_eco= key_eco_1[['Year', 'Consumer Confidence Index TX', 'Consumer Confidence In
```

```
In [17]: # This code creates the scatter plot for US and Texas CCI
plt.figure(figsize = (15,8))
plt.scatter(x = 'Consumer Confidence Index TX',y = 'Consumer Confidence Index US',
plt.title("Texas Versus US Consumer Confidence", fontsize=20, weight='bold')
plt.xlabel("Texas Consumer Confidence", fontsize=15, weight='bold' )
plt.ylabel("US Consumer Confidence", fontsize=15, weight='bold')
plt.show()
```

This graph shows a comparison between the Texas CCI and the US CCI. This graph does show a correlation between the US and Texas CCI markers. The correlation is relatively weak and I see a few outliers. However there is a positive relationship between the two features and I will explore this further.

Milestone 2

I want to prepare the school data set for a linear regression. My first step is to evaluate the features of the data. I removed Grades\nServed, Campus Enrollment\nType, District\nNumber, Campus\nNumber. I removed these features because they don't add value to the analysis. The district and campus numbers are just campus identification numbers. Grades served and campus enrollment don't offer useful data for the analysis.

After I remove the unneeded columns I want to ensure that there are no null values. My next step is to create a subset to convert my categorical data into dummy variables. I also need to remove the original column data with the respective dummy variables for those columns. The final step to prep the data is to scale the data. For this I will use min max scaler. This will normalize the data.

```
In [18]: pd.options.display.max_columns = None
```

```
In [19]: # This code drops two columns grades served and campus enrollment type
school_1 = school.drop(['Grades\nServed', 'Campus Enrollment\nType',
                        'District\nNumber', 'Campus\nNumber'], axis=1)
```

```
In [20]: # This code removes the null values.
school_1 = school_1[(~school.isnull()).all(axis=1)]
```

```
In [21]: # This code displays the cleaned data
school_1.head()
```

Out[21]:

	District	Campus	Region	County	School\nType	Alternative\nEducation
159	BANDERA ISD	BANDERA H S	REGION 20: SAN ANTONIO	BANDERA	High School	
278	KILLEEN ISD	MANOR MIDDLE	REGION 12: WACO	BELL	Middle School	
280	KILLEEN ISD	PALO ALTO MIDDLE	REGION 12: WACO	BELL	Middle School	
517	EDGEWOOD ISD	BRENTWOOD MIDDLE	REGION 20: SAN ANTONIO	BEXAR	Middle School	
519	EDGEWOOD ISD	E T WRENN MIDDLE	REGION 20: SAN ANTONIO	BEXAR	Middle School	



```
In [22]: school_sub = school_1[['District', 'Campus', 'Region', 'County', 'School\nType', 'A
        'Charter', 'Distinction\nSoc Studies', 'Distinction\nProgress',
        'Distinction\nPostsecondary\nReadiness', 'Distinction\nClosing
        'Relative\nPerformance\nRating', 'Student\nAchievement\nRating'
        'School\nProgress\nRating', 'Academic\nGrowth\nRating', 'Closin
        'Support\nLabel', 'Public\nEducation\nGrant', 'Distinction\nELA
        'Distinction\nMathematics', 'Distinction\nScience']]
```

```
In [23]: school_sub_dummy = pd.get_dummies(school_sub).replace({True : 1, False : 0})
school_sub_dummy.head()
```

C:\Users\mchlp\AppData\Local\Temp\ipykernel_26804\2990287029.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`

```
school_sub_dummy = pd.get_dummies(school_sub).replace({True : 1, False : 0})
```

Out[23]:

	District_ALDINE ISD	District_AUSTIN ISD	District_BANDERA ISD	District_BEAUMONT ISD	District_CLEVER ISD
159	0	0	1	0	
278	0	0	0	0	
280	0	0	0	0	
517	0	0	0	0	
519	0	0	0	0	

```
In [24]: school_1.drop(['District', 'Campus', 'Region', 'County', 'School\nType',  
                        'Alternative\nEducation\nAccountability', 'Charter', 'Distinction\nSoc  
Distinction\nProgress', 'Distinction\nClosing the Gaps', 'Distinction\nRelative\nPerformance\nRating', 'Student\nAchievement\nRating',  
                        'Overall\nRating', 'Academic\nGrowth\nRating', 'School\nProgress\nRatin  
Closing\nthe Gaps Rating', 'Support\nLabel', 'Public\nEducation\nGrant  
Distinction\nELA/Reading', 'Distinction\nMathematics',  
                        'Distinction\nScience'], axis=1, inplace= True)
```

```
In [25]: lm_school= pd.concat([school_1, school_sub_dummy], axis=1)  
  
lm_school.head()
```

Out[25]:

	Number of\nStudents	%\nEconomically\nDisadvantaged	% EB/EL\nStudents	Overall\nScore	Stu
159	681	0.386	0.040	56	
278	659	0.912	0.199	57	
280	835	0.901	0.119	59	
517	750	0.855	0.161	59	
519	505	0.846	0.283	58	

```
In [26]: scaleMin_Max = MinMaxScaler()
```

```
In [27]: X_school = lm_school.iloc[:, 0:14]
```

```
In [28]: X_school = scaleMin_Max.fit_transform(X_school)
```

```
In [29]: X_school = pd.DataFrame(X_school, columns= ['Number of\nStudents', '%\nEconomically\nEB/EL\nStudents', 'Overall\nScore', 'Student\nAchievement\nScore', 'School\nProgress\nScore', 'Academic\nGrowth\nScore', 'Relative\nPerformance\nScore', 'Closing\nthe Gaps Score', 'US Congress\nDistrict', 'US\nSenate\nDistrict', 'TX House\nDistrict', 'TX House\nElection District', 'TX Se\nnate\nElection District'])
```

```
In [30]: lm_school = lm_school.drop(columns=['Number of\nStudents', '%\nEconomically\nDisadvantaged\nStudents', 'Overall\nScore', 'Student\nAchievement\nScore', 'School\nProgress\nScore', 'Academic\nGrowth\nScore', 'Relative\nPerformance\nScore', 'Closing\nthe Gaps Score', 'US Congress\nDistrict', 'US\nSenate\nElection District', 'TX House\nDistrict', 'TX House\nElection District', 'TX Senate\nDistrict', 'TX\nSenate\nElection District'])
```

```
In [31]: lm_school_fit = pd.concat([lm_school, X_school ], axis=1)
```

```
In [78]: lm_school_fit.head()
```

Out[78]:

	District_ALDINE ISD	District_AUSTIN ISD	District_BANDERA ISD	District_BEAUMONT ISD	District_CLEVERLAND ISD
159	0.0	0.0	1.0	0.0	0.0
278	0.0	0.0	0.0	0.0	0.0
280	0.0	0.0	0.0	0.0	0.0
517	0.0	0.0	0.0	0.0	0.0
519	0.0	0.0	0.0	0.0	0.0

The second data set shows sales tax data. I interested to see if this data shows areas of prosperity and will aid to predict areas of investment. The first step is to understand the data. Previously I created a scatter plot that showed a positive relationship with current and prior year sales tax. My next step is to prep the data for a regression analysis.

The first step is to remove data that will not add to the analysis. The columns that I am removing are city, report month, report year, and report period type. The values of importance is the actual taxes paid, both current and prior.

After removing the data, I want to scale the data for analysis. The scaling method that I want is standard scaler. I chose this because I have negative values and standard scaler plays nicely with negative numbers.

```
In [32]: # This code displays the first three rows
sales_tax.head(3)
```

Out[32]:

	City	Net Payment This Period	Comparable Payment Prior Year	Percent Change From Prior Year	Payment to Date	Previous Payments to Date	Percent Change To Date	Report Month
0	Abbott	14301.88	15919.41	-10.16	146889.53	152146.26	-3.45	11
1	Abernathy	23516.10	23987.13	-1.96	270303.69	256849.99	5.23	11
2	Abilene	5526055.82	5477084.93	0.89	57014074.04	53887697.25	5.80	11

```
In [33]: # This code that displays the dimension of the data
         sales_tax.shape
```

```
Out[33]: (171640, 10)
```

```
In [34]: # This code drops unneeded data
sales_tax = sales_tax.drop(['City', 'Report Month',
                           'Report Year', 'Report Period Type'], axis=1)
```

```
In [35]: sales_tax.head(5)
```

Out[35]:

	Net Payment This Period	Comparable Payment Prior Year	Percent Change From Prior Year	Payment to Date	Previous Payments to Date	Percent Change To Date
0	14301.88	15919.41	-10.16	146889.53	152146.26	-3.45
1	23516.10	23987.13	-1.96	270303.69	256849.99	5.23
2	5526055.82	5477084.93	0.89	57014074.04	53887697.25	5.80
3	8877.41	12792.00	-30.60	86833.38	139977.10	-37.96
4	1588208.70	1316545.16	20.63	15730647.58	14677334.92	7.17

```
In [36]: X = sales_tax.iloc[:,0:6]
```

```
In [37]: scale_standard = StandardScaler()
```

```
In [38]: X = scale_standard.fit_transform(X)
```

```
In [39]: sales_tax_fit = pd.DataFrame(X, columns=['Net Payment This Period',
        'Comparable Payment Prior Year', 'Percent Change From P
        'Payment to Date', 'Previous Payments to Date',
        'Percent Change To Date'])
```

```
In [40]: sales_tax_fit.head()
```

Out[40]:

	Net Payment This Period	Comparable Payment Prior Year	Percent Change From Prior Year	Payment to Date	Previous Payments to Date	Percent Change To Date
0	-0.178923	-0.176204	-0.035204	-0.154420	-0.152264	-0.013206
1	-0.175252	-0.172837	-0.024572	-0.147574	-0.146170	-0.007198
2	2.016969	2.102817	-0.020877	3.000263	2.975514	-0.006804
3	-0.181085	-0.177509	-0.061706	-0.157752	-0.152973	-0.037092
4	0.448124	0.366566	0.004718	0.710082	0.693201	-0.005855

The third data set is the population data. This data is relatively clean however there are some columns that are not needed in my analysis. I removed migration scenario, year month, fips, and area name. I removed these columns because I'm interested in the actual population numbers for the years specified. In addition I removed the first row because it contains a values that totals all ages of residents. After completing these data cleaning items, want to create a new data set that contains the years 2020-2025 I accomplished this by filter the year column.

```
In [41]: # This code displays the first three rows of the population data
pop.head(3)
```

Out[41]:

	migration_scenario	year	year_month	FIPS	area_name	age_in_yrs_num	age_in_yrs_cha
0	Mid	2020	202004	0	State of Texas	-1	All Age:
1	Mid	2020	202004	0	State of Texas	0	< 1 y
2	Mid	2020	202004	0	State of Texas	1	1 y



```
In [42]: # This code removes the first row
pop.drop(index=0, inplace=True)
```


```
In [43]: # This code drops the unneeded rows
pop_1= pop.drop(columns=['migration_scenario', 'year_month', 'FIPS', 'area_name', 'ag
'total'])
```

```
In [44]: # This code filters the data to give me the specified years for analysis
new_pop = pop_1.loc[(pop_1['year']>=2020) & (pop_1['year']<=2025)]
```

```
In [45]: # This code displays the first five rows of the new data set
new_pop.head()
```

Out[45]:

	year	age_in_yrs_num	total_male	total_female	nh_white_total	nh_white_male	nh_white
1	2020	0	173553	167009	99694	50985	
2	2020	1	176404	170602	103308	52866	
3	2020	2	183499	177691	106386	54360	
4	2020	3	192237	185157	112133	57341	
5	2020	4	199902	193206	117552	60108	




In my final data set key eco I have a lot of null values. I can't remove the null values due to how it will affect my dataset. It appears that data was not collected for the columns in question. The best way to deal with these values is to fill them with zeros. I believe that this will create the least negative impact especially since I want to normalize the data. I also plan on dropping the month column since I am primarily interested in the annual totals.

```
In [96]: key_eco.head(3)
```

Out[96]:

	Month	Year	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.	U.S. F En
0	1	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
1	2	2005	NaN	NaN	NaN	NaN	NaN	NaN	I
2	3	2005	NaN	NaN	NaN	NaN	NaN	NaN	I



```
In [97]: key_eco.tail(5)
```

Out[97]:

	Month	Year	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.
247	8	2025	NaN	NaN	NaN	NaN	NaN	NaN
248	9	2025	NaN	NaN	NaN	NaN	NaN	NaN
249	10	2025	NaN	NaN	NaN	NaN	NaN	NaN
250	11	2025	NaN	NaN	NaN	NaN	NaN	NaN
251	12	2025	NaN	NaN	NaN	NaN	NaN	NaN

In [98]: `key_eco_1 = key_eco.fillna(0)`

In [99]: `key_eco_1.head()`

Out[99]:

	Month	Year	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.	U.S. F En
0	1	2005	0.0	0.0	0.0	0.0	0.0	0.0	
1	2	2005	0.0	0.0	0.0	0.0	0.0	0.0	
2	3	2005	0.0	0.0	0.0	0.0	0.0	0.0	
3	4	2005	0.0	0.0	0.0	0.0	0.0	0.0	
4	5	2005	0.0	0.0	0.0	0.0	0.0	0.0	

In [100...]: `scale = MinMaxScaler()`

In [101...]: `X_key_eco_1 = key_eco_1.iloc[:, 2:31]`

In [102...]: `X_key_eco_1 = scale.fit_transform(X_key_eco_1)`

In [105...]: `X_key_eco_1 = pd.DataFrame(X_key_eco_1, columns= ['Consumer Confidence Index TX', 'C
Consumer Confidence Index US', 'PCE Deflator', 'Co
'CPI U.S. Ex Food and Energy', 'Nonfarm Employment TX',
'Unemployment TX', 'Unemployment U.S.', 'Single Family
'Multi Family Building Permits TX', 'Existing Single Fa
'Existing Single Family Home Price TX', 'Non Reside
'Total Sales Tax Collections Retail TX', 'Total Sales T
'Retail Gasoline Price TX', 'Retail Diesel Price TX', '`


```
'Nonfarm Employment Florida', 'Nonfarm Employment New Y  
'Nonfarm Employment California', 'Gross Value Crude Oil  
'Gross Value Natural Gas Production', 'Motor Fuel Taxed
```

```
In [106... X_key_eco_1.head()
```

Out[106...

	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.	CPI U.S. Ex Food and Energy	Nonfa Employment
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	



```
In [107... key_eco_1.drop(columns= ['Month', 'Consumer Confidence Index TX', 'Consumer Confiden  
                        'Consumer Confidence Index US', 'PCE Deflator', 'Co  
                        'CPI U.S. Ex Food and Energy', 'Nonfarm Employment TX',  
                        'Unemployment TX', 'Unemployment U.S.', 'Single Family  
                        'Multi Family Building Permits TX', 'Existing Single Fa  
                        'Existing Single Family Home Price TX', 'Non Reside  
                        'Total Sales Tax Collections Retail TX', 'Total Sales T  
                        'Retail Gasoline Price TX', 'Retail Diesel Price TX', '  
                        'Nonfarm Employment Florida', 'Nonfarm Employment New Y  
                        'Nonfarm Employment California', 'Gross Value Crude Oil  
                        'Gross Value Natural Gas Production', 'Motor Fuel Taxed  
                        'Motor Fuel Taxed Diesel'], inplace=True)
```

```
In [108... key_eco_clean = pd.concat([key_eco_1, X_key_eco_1], axis = 1)
```

```
In [109... key_eco_clean.head()
```

Out[109...

	Year	Consumer Confidence Index TX	Consumer Confidence West South Central	Consumer Confidence Index US	PCE Deflator	Consumer Price Index TX	Consumer Price Index U.S.	CPI U.S. Ex Food and Energy	Em
0	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	2005	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

During this milestone I prepared my data sets for regression analysis. I performed minor transformations because I want test my data sets with minimal disruptions. I may need to perform additional transformations after my initial regression run.