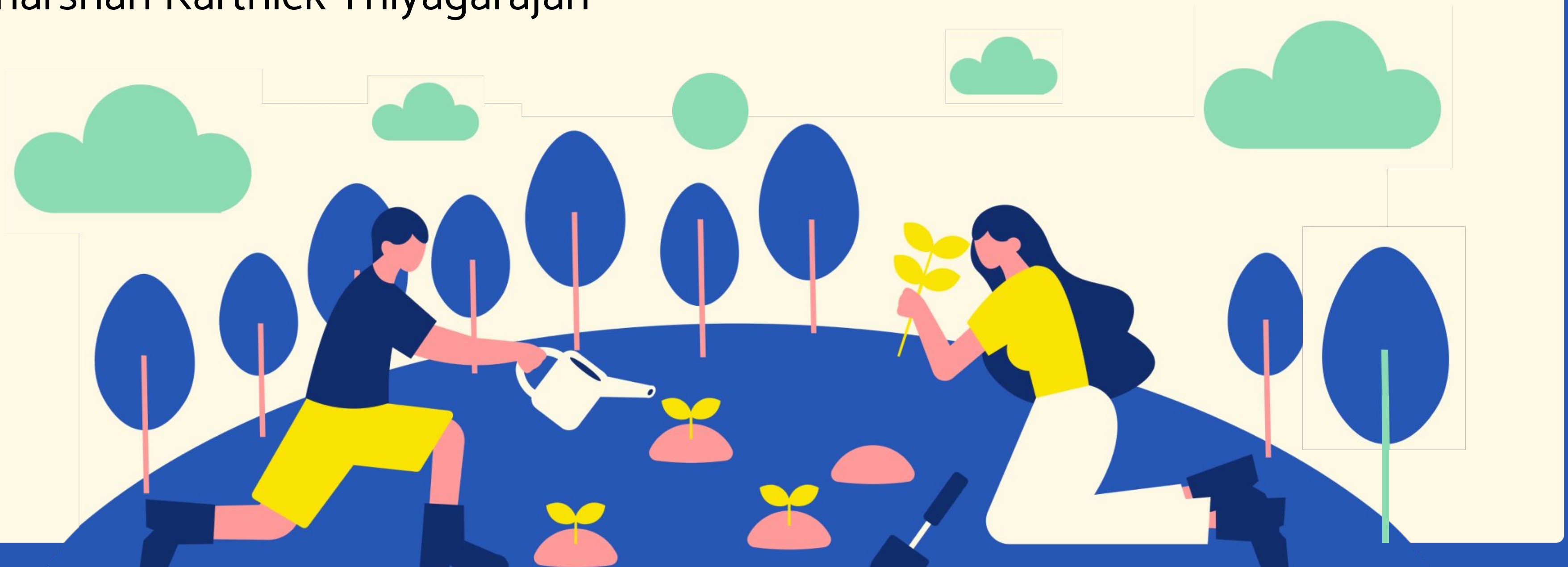
CS 2316 Final Project

Dharshan Karthick Thiyagarajan

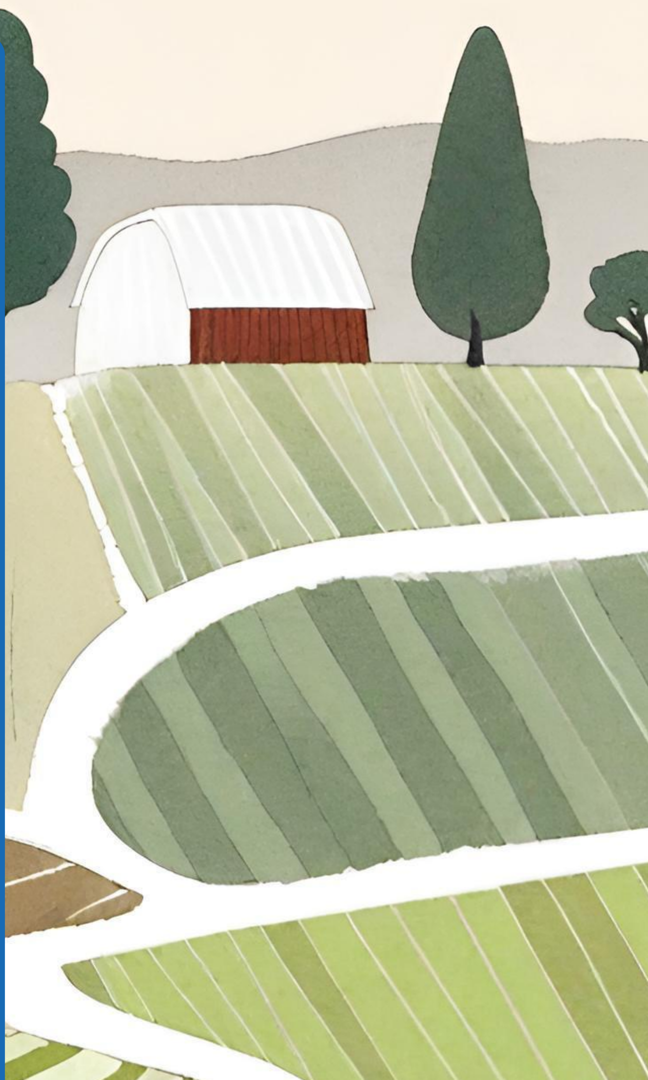
An analysis of the US economy through the environment.



Interest

- Began by trying to locate topics that directly affected me
- I chose global warming as that is a topic that is only increasing in relevancy.

- I was also always interested into the US economy, and its overall fluctuations.
- I wanted to see if there was a correlation between an increase in emissions that cause global warming and the GDP of the US.



Expectations/Predictions

- ***I expected to see:***
  - A correlation between the GDP of the United States and increased emissions.
  - An overall increase in the GDP of the United States.
  - An overall increase in all types of emissions, but especially CO_2 emissions.

I decided to use emissions, with a focus in CO_2, as a metric for global warming as they contribute to the greenhouse gases and they are also primarily human generated.

Data Collection Process

01
Government Database
Found a comprehensive downloadable XLSX file on emissions data by year and state

This data met the size requirements and was from data.gov

02
Wikipedia HTML
This had a massive table containing a lot of economic factors of the US

I cross referenced this data with other sources to verify it's accuracy.

03
API EIA
Found an API from EIA(key required for access)

This contained data on emissions by each sector of the economy in a JSON-like structure.

Data Cleaning and its Challenges

Downloaded Dataset Cleaning

- **Converted the xlsx file to a df by reading it using Pandas.**
  - State Names were inconsistent across sources, so I used the map function to map the names to a consistent pattern.
  - The totals were incorrectly summed so I manually recalculated the sum to ensure accuracy.
  - I learned about using the apply and map functions in Pandas through this portion of cleaning.

```python
#  Data Source: https://catalog.data.gov/dataset/annual-u-s-electric-power-industry-estimated-emissions-by-state-fro
# Resources used:
#     Pandas Documentation (Class notes)- https://docs.google.com/document/d/1djwz0iMq4weNDKBMZw71bGMnpgphU4h0/edit
#     Mapping(advanced requirement): https://www.geeksforgeeks.org/using-dictionary-to-remap-values-in-pandas-datafr
#     Display Function documentation https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html
import pandas as pd
def data_parser():
#     print("Hi")

    # Made a dictionary to quickly map everything to handle my first inconsistency
    state_territory_mapping = {
        'AL': 'Alabama', 'KY': 'Kentucky', 'OH': 'Ohio',
        'AK': 'Alaska', 'LA': 'Louisiana', 'OK': 'Oklahoma',
        'AZ': 'Arizona', 'ME': 'Maine', 'OR': 'Oregon',
        'AR': 'Arkansas', 'MD': 'Maryland', 'PA': 'Pennsylvania',
        'AS': 'American Samoa', 'MA': 'Massachusetts', 'PR': 'Puerto Rico',
        'CA': 'California', 'MI': 'Michigan', 'RI': 'Rhode Island',
        'CO': 'Colorado', 'MN': 'Minnesota', 'SC': 'South Carolina',
        'CT': 'Connecticut', 'MS': 'Mississippi', 'SD': 'South Dakota',
        'DE': 'Delaware', 'MO': 'Missouri', 'TN': 'Tennessee',
        'DC': 'District of Columbia', 'MT': 'Montana', 'TX': 'Texas',
        'FL': 'Florida', 'NE': 'Nebraska', 'TT': 'Trust Territories',
        'GA': 'Georgia', 'NV': 'Nevada', 'UT': 'Utah',
        'GU': 'Guam', 'NH': 'New Hampshire', 'VT': 'Vermont',
        'HI': 'Hawaii', 'NJ': 'New Jersey', 'VA': 'Virginia',
        'ID': 'Idaho', 'NM': 'New Mexico', 'VI': 'Virgin Islands',
        'IL': 'Illinois', 'NY': 'New York', 'WA': 'Washington',
        'IN': 'Indiana', 'NC': 'North Carolina', 'WV': 'West Virginia',
        'IA': 'Iowa', 'ND': 'North Dakota', 'WI': 'Wisconsin',
        'KS': 'Kansas', 'MP': 'Northern Mariana Islands', 'WY': 'Wyoming'
    }

    emission_data = pd.read_excel('emission_annual.xls')

    # Remove '\n' from column names
    emission_data.columns = [col.replace('\n', ' ') for col in emission_data.columns]

    # This ensures the I only consider the data where all sources == 'All Sources'
    # This is because the data is formatted in such a way where it has varying types of energy sources. As my goal
    #is to address a broader perspective,
    ## I will only consider the data where the Energy Source is listed as all sources.
    emission_data = emission_data[emission_data['Energy Source'] == 'All Sources']

    # Uses map values to map the abbreviation to the expanded name (inconsistency)
    emission_data['State'] = emission_data['State'].map(state_territory_mapping)

    # Grouped by Year, State, Producer and sum the values to recalculate the proper totals of C02 S02 and NOX.

    emission_data = emission_data.groupby(['Year', 'State', 'Producer Type']).sum().reset_index()

   ## displayed the final df (documentation above)
    display(emission_data)

  ## also wrote the file to an excel file
    emission_data.to_excel("emission_data.xlsx", index=False)
## Initial Data: 43,260 rows, cleaned it to 9280 rows.

############ Function Call ############
data_parser()
```

Web Requirement 1 (HTML)

- **Used Beautifulsoup to parse through the data**
  - Found the table using the apt class and tag. then, used the tags to separate out the column names and data.
  - Used a for loop to append each row of data and then appended each row to a list of lists
  - Converted this into Pandas df and then cleaned/dropped based on relevancy.
  - Filled the null values with np.nan and then filled them with the mean of the column.

```python
def web_parser1():
    # initalized the beautiful soup process by making a request and initializing the soup object
    website_url = 'https://en.wikipedia.org/wiki/Economy_of_the_United_States'
    response = requests.get(website_url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Found the HTML table with class attribute and then found the tr tag
    table = soup.find('table', attrs={'class': 'wikitable'})
    first_row = table.find('tr')

    # Extract the column names from each 'th' element in the first row
    column_names = []
    for th in first_row.find_all('th'):
        column_name = th.get_text(strip=True)
#         print("the column name is ", column_name)
        column_names.append(column_name)

    # This code loops through each row and column of the HTML table and extracts the text
    # from each cell. It then appends the row data to a list of data
    data = []
    for row in table.find_all('tr'):
        row_data = []
        for cell in row.find_all('td'):
            row_data.append(cell.text)
        data.append(row_data)

    # Converted the data to a Pandas DataFrame, removed the first row as it is having None Values
    df = pd.DataFrame(data[1:], columns= column_names)

    # Used map to replace the \n vals
    df = df.map(lambda x: x.replace('\n', ''))

    # Replace n/a with NaN (an intermediate step for the calculations to come)(only col with n/a)
    df['Government debt(in % of GDP)'] = df['Government debt(in % of GDP)'].replace('n/a', np.nan)

    # Remove % sign and converted to numeric (needed for future data analysis) (inconsistency)
    # rstrip() removes the trailing whitespace characters from the right end of a string.
    df['Government debt(in % of GDP)'] = df['Government debt(in % of GDP)'].str.rstrip('%').astype(float)
    df['Inflation rate(in Percent)'] = df['Inflation rate(in Percent)'].str.rstrip('%').astype(float)
    df['GDP growth(real)'] = df['GDP growth(real)'].str.rstrip('%').astype(float)

    # Calculate the mean of the column
    ## when looking at methods to fill the data, the mean seems to be the one that provided values of the same trend
    ## other methods that I considered were using the median and manually calculating the debt.
    mean_value = df['Government debt(in % of GDP)'].mean()
    # Replace null values of the column with the mean (inconsistency handled)
    df['Government debt(in % of GDP)'] = df['Government debt(in % of GDP)'].fillna(mean_value)

    # Dropped the Unemployment(in Percent) column using inplace because I do not plan on using this col for my
    # calculations
    df.drop(columns=['Unemployment(in Percent)'], inplace=True)
```

Web Requirement 2 (API)

- **Used Requests module to access the API using personal key**
  - The API had a JSON (dictionary-type) Structure and was very large.
  - Found out the proper indices to access the Data I needed and converted that to a dataframe .
  - Learned a lot about how APIs work and are relevant to industry

```python
import requests
import pandas as pd
## Source:
#     https://www.eia.gov/opendata/browser/co2-emissions/co2-emissions-aggregates?frequency=annual&data=value;&start
## Additional Resources:
#     How to use API keys: https://coding-boot-camp.github.io/full-stack/apis/how-to-use-api-keys

def web_parser2():
    ## Personal API Key generated by signing up on the website
    api_key = 'evjniXiTBaTS6FClaMhueeCdcAdvGEz3EqV52bUW'
    url = 'https://api.eia.gov/v2/co2-emissions/co2-emissions-aggregates/data/?frequency=annual&data[0]=value&start=
    params = {'api_key': api_key}

    response = requests.get(url, params=params)

    if response.status_code == 200:
        ## parsed through the json (dictionary-like structure) to get the valid_data and then converted it to a df
        valid_data = response.json()["response"]["data"]
        valid_df = pd.DataFrame(valid_data)

        # Convert the 'period' column to integer type and sorted/filtered the values based on period
        ## used data from 2019-onwards as this had a massive amount of data and I needed to filter by such to
        ## reduce my load
        valid_df['period'] = valid_df['period'].astype(int)
        valid_df = valid_df[valid_df['period'] >= 2019]

        # Dropped columns that were irrelevant
        valid_df.drop(columns=['fuelId', 'stateId', 'value-units'], inplace=True)

        # Rename the columns (in millions as every data point was in millions)
        valid_df.rename(columns={'value': 'value (in Millions)'}, inplace=True)
#         valid_df = valid_df[valid_df['value (in Millions)'] != '0'] ## not needed
#         valid_df['value (in Millions)'] = valid_df['value (in Millions)'].astype(int)


#         print("Data retrieved successfully.")
## displays output df and write it to a file
        display(valid_df)
        valid_df.to_excel('valid_data.xlsx', index=False)
    else:
        print(f"Error: {response.status_code}")




############ Function Call ############
web_parser2()
```

Insights and Visualizations

Insight 1: Correlation between US GDP and Emissions Increase/CO2

- **Correlation Coefficient of 0.02 and -0.2:**
  - Called the two functions and received two different Dataframes
  - Used pandas .corr() method to drive a correlation between the GDP per Capita and total emissions value (in millions) and Between GDP and CO2 emissions (actual)

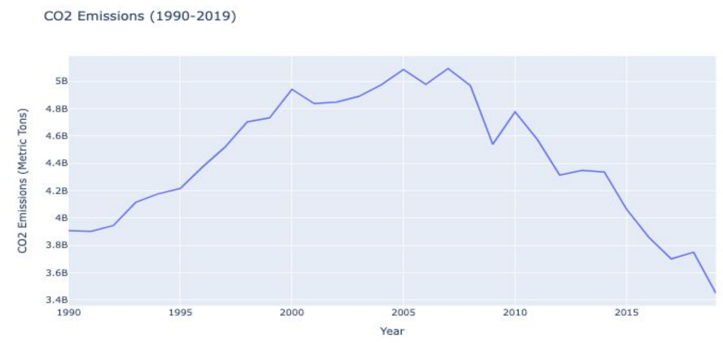
- **Conclusions:**
  - There is little to no correlation between the GDP and total Emissions (0.02), and there is a slightly negative correlation between GDP and CO2 emissions (-0.2) This was the opposite of what I predicted.
  - The value being on the negative side is actually a good sign for The environment.

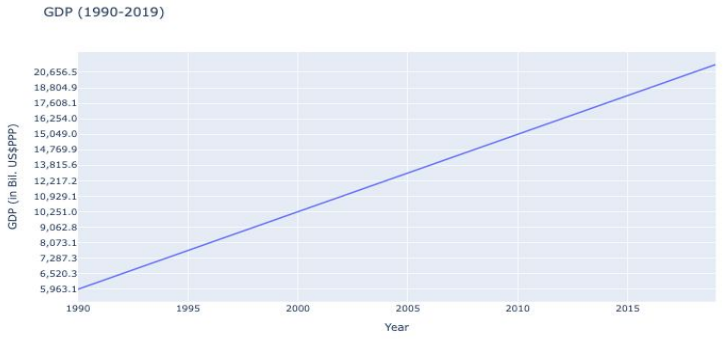
```
    valid_df=web_parser1()
    tempy=web_parser2()
    anotherdf=data_parser()
#     df['GDP per capita(in US$ PPP)'] = df['GDP per capita(in US$ PPP)'].str.replace(',', '').astype(float)
    ## converted the value into float (as for some reason it was a string originally)
    tempy['value (in Millions)'] = tempy['value (in Millions)'].astype(float)
    ## used corr to find the correlation and rounded it to two points
    correlation_coefficient = round(valid_df['GDP per capita(in US$ PPP)'].corr(tempy['value (in Millions)']),2)
    corr_coeff_c02=round(valid_df['GDP per capita(in US$ PPP)'].corr(anotherdf['CO2 (Metric Tons)']),2)
    line1 = "Correlation between the Emissions Value (in Millions) and the GDP Per Capita of the United States: " +
    line2 = "Correlation between Metric Tons of CO2 emissions and GDP Per Capita of the United States: " + str(corr_

    print(line1)
```

Insight 2: Linear Regression Model for the GDP of the United States

- **Used sk.Learn's linear model to create a function**
  - Ensured that the GDP dataframe had the correct values Needed to compute said analysis.
  - Used sk.learn()'s documentation on linear regression to Compute a model of the US GDP over time. This model can also be used to predict the GDP of future years.
- **Conclusions:**
  - I was able to figure out that our GDP modeled a positive linear Graph, and generated a visualization to depict this relationship .
  - I developed a GUI interface that incorporates this model And enables the user to predict the GDP for future years.

```python
# Called the web_parser1 function to get the DataFrame
df = web_parser1()

# Converted the GDP column data types from string to an int (faced an issue with commas so replaced that
df['GDP(in Bil. US$PPP)'] = df['GDP(in Bil. US$PPP)'].str.replace(',', '').astype(float)

# I then set the independent and dependent variables for the model
# -(independent variables): Year column
# -(dependent variable): GDP(in Bil. US$PPP) column
X = df[['Year']]
y = df['GDP(in Bil. US$PPP)']
# initalized the Linear Regression model and used fit function to fit x and y in
model = LinearRegression()
model.fit(X, y)

# Printed the intercept and coefficient of the linear regression model
print("Intercept:", model.intercept_)
print("Coefficient:", model.coef_[0])


# Tested this model to predict the GDP for a certain year.
## This will come in handy for the GUI which is going to be a GDP calculator.
year = 2024
predict_gdp = model.predict([[year]])
print("Predicted GDP for year", year, ":", predict_gdp[0])
```






```python
# got the dfs and filtered out to get relevant data
    df = data_parser()
    otherdf=web_parser1()
    otherdf['Year'] = otherdf['Year'].astype(int)
    df_selected = df[(df['Year'] >= 1990) & (df['Year'] <= 2019)]
    otherdf_selected = otherdf[(otherdf['Year'] >= 1990) & (otherdf['Year'] <= 2019)]

    # Calculating for the total c02 emissions per year
    total_co2_emissions = df_selected.groupby('Year')['CO2 (Metric Tons)'].sum().reset_index()

    # merging the two dfs
    merge = pd.merge(total_co2_emissions, otherdf_selected, on='Year')

    # line plot for CO2 emissions
    fig_co2 = px.line(merge, x='Year', y='CO2 (Metric Tons)',
                      title='CO2 Emissions (1990-2019)',
                      labels={'CO2 (Metric Tons)': 'CO2 Emissions (Metric Tons)'})

    #  line plot for GDP
    fig_gdp = px.line(merge, x='Year', y='GDP(in Bil. US$PPP)',
                      title='GDP (1990-2019)',
                      labels={'GDP(in Bil. US$PPP)': 'GDP (in Bil. US$PPP)'})


    fig_co2.show()
    fig_gdp.show()


## original plan of just gdp
```

Visualization:

- This are two line graphs showing the trends of CO2 emissions and GDP from the years of (1990-2019)
  The GDP is constantly/linearly increasing, whereas CO2 fluctuates.
  I used Plotly express's line chart documentation to help create this graph.

Insight 3: How have we developed?

- **Aggregated through each of the emissions factors in the DF**
  - Used the .agg() functionality of Pandas and calculated the sum Of each of the emission factors, then used this aggregated df To calculate the percent change from 2017 to 2019
  - Also, utilized the GDP table to calculate the average growth in GDP in the same time period .
- **Conclusions:**
  - Overall, all emission factors went down from their levels In 2017, with SO2 levels dropping up to 23.5%. This is a great Sign for the environment.
  - The average GDP growth in this time period was 2.63, which Shows that the health of the economy was increasing.

	Pollutant	Total Emissions (2017)	Total Emissions (2019)	Percent Change	Average GDP Growth
0	CO2	3699499944	3448792210	-6.776801	2.633333
1	SO2	3313698	2533380	-23.548253	2.633333
2	NOx	3011524	2683092	-10.905840	2.633333

```python
df_filtered = df[df['Year'].isin([2017, 2019])]

# Calculated total emissions for each pollutant for each year using the .agg() module in Pandas
total_emissions = df_filtered.groupby(['Year', 'State', 'Producer Type']).agg({
    'CO2 (Metric Tons)': 'sum',
    'SO2 (Metric Tons)': 'sum',
    'NOx (Metric Tons)': 'sum'
}).reset_index()

# Calculate percent change for each pollutant from 2017 to 2019 by calculating 2019-2017/2017 (percent change fo
percent_change_CO2 = ((total_emissions.loc[total_emissions['Year'] == 2019, 'CO2 (Metric Tons)'].sum() -
                       total_emissions.loc[total_emissions['Year'] == 2017, 'CO2 (Metric Tons)'].sum()) /
                      total_emissions.loc[total_emissions['Year'] == 2017, 'CO2 (Metric Tons)'].sum()) * 100

percent_change_SO2 = ((total_emissions.loc[total_emissions['Year'] == 2019, 'SO2 (Metric Tons)'].sum() -
                       total_emissions.loc[total_emissions['Year'] == 2017, 'SO2 (Metric Tons)'].sum()) /
                      total_emissions.loc[total_emissions['Year'] == 2017, 'SO2 (Metric Tons)'].sum()) * 100

percent_change_NOx = ((total_emissions.loc[total_emissions['Year'] == 2019, 'NOx (Metric Tons)'].sum() -
                       total_emissions.loc[total_emissions['Year'] == 2017, 'NOx (Metric Tons)'].sum()) /
                      total_emissions.loc[total_emissions['Year'] == 2017, 'NOx (Metric Tons)'].sum()) * 100

# Made new df to contain the results of both the totals and the percent change
result_df = pd.DataFrame({
    'Pollutant': ['CO2', 'SO2', 'NOx'],
    'Total Emissions (2017)': [total_emissions.loc[total_emissions['Year'] == 2017, 'CO2 (Metric Tons)'].sum(),
                               total_emissions.loc[total_emissions['Year'] == 2017, 'SO2 (Metric Tons)'].sum(),
                               total_emissions.loc[total_emissions['Year'] == 2017, 'NOx (Metric Tons)'].sum()]
    'Total Emissions (2019)': [total_emissions.loc[total_emissions['Year'] == 2019, 'CO2 (Metric Tons)'].sum(),
                               total_emissions.loc[total_emissions['Year'] == 2019, 'SO2 (Metric Tons)'].sum(),
                               total_emissions.loc[total_emissions['Year'] == 2019, 'NOx (Metric Tons)'].sum()]
    'Percent Change': [percent_change_CO2, percent_change_SO2, percent_change_NOx]
})

I wanted to see if there is a relationship between the average growth in GPD so I calculated date using the
apt column from the web_parser1() df
other_df['Year'] = other_df['Year'].astype(int)
gdp_growth = other_df[(other_df['Year'] >= 2017) & (other_df['Year'] <= 2019)]

# Calculate the average GDP growth (real) from 1990 to 2019
average_gdp_growth = gdp_growth['GDP growth(real)'].mean()

# Added the average gdp growth to the result_df for the
  print(average_gdp_growth)
result_df['Average GDP Growth'] = average_gdp_growth

return result_df
```

Insight 4: How do we compare to the rest of the US?

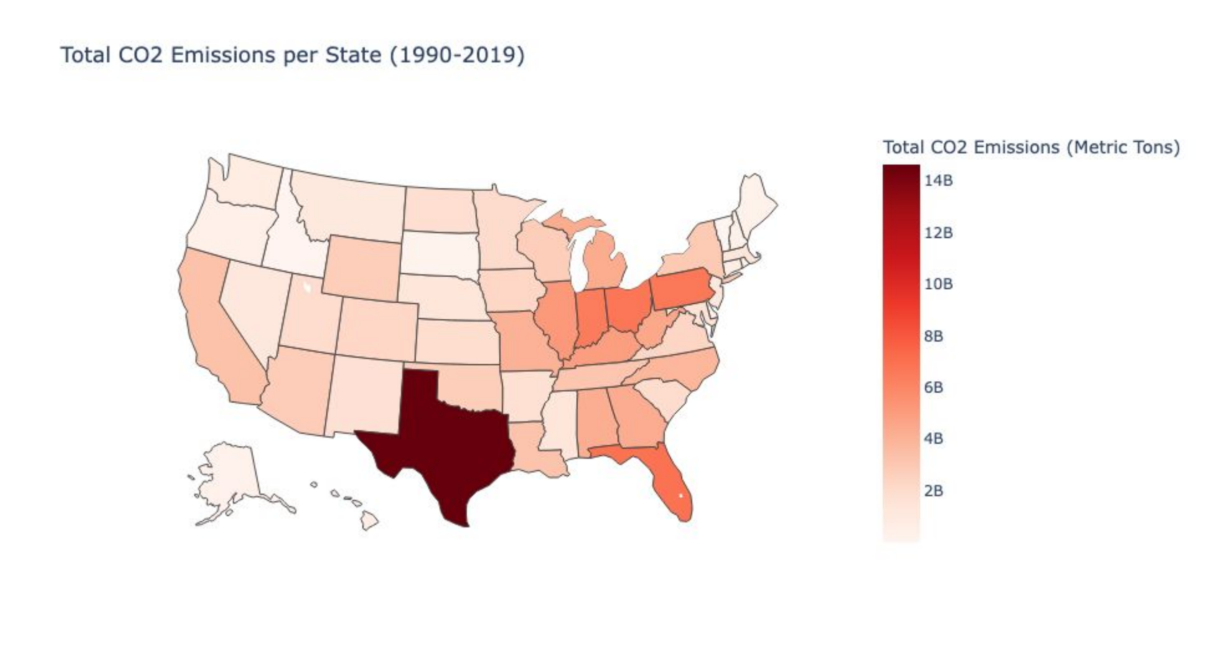
- **Used the government dataset to compare our average emissions to other states.**
  - Filtered the data frame for only Georgia and then calculated the Means of the emissions, and then did the same thing Excluding Georgia for the rest of the states.
  - Used f string formatting within the Pandas column creation to Make the resulting dataframe more appealing.
- **Conclusions:**
  - Overall, Georgia's (average emissions) are higher on all emission types . This could be attributed to our size, population, and many other factors.
  - This data shows, however, that we need to take action as a state To curb our environmental impact.

```python
df = data_parser()
other_df = web_parser1()

# filtered the df for only georgia
georgia = df[df['State'] == 'Georgia']

# Calculated the average CO2, SO2, and NOx emissions for Georgia and rounded to two decimals
avg_co2_ga = round(georgia['CO2 (Metric Tons)'].mean(), 2)
avg_so2_ga = round(georgia['SO2 (Metric Tons)'].mean(), 2)
average_nox_emissions_georgia = round(georgia['NOx (Metric Tons)'].mean(), 2)

# Calculated the average CO2, SO2, and NOx emissions for the rest of the states and round to two decimals
other_states_data = df[df['State'] != 'Georgia']
average_co2_emissions_other_states = round(other_states_data['CO2 (Metric Tons)'].mean(), 2)
average_so2_emissions_other_states = round(other_states_data['SO2 (Metric Tons)'].mean(), 2)
avg_nox_ga = round(other_states_data['NOx (Metric Tons)'].mean(), 2)

# new df to for the results
results_df = pd.DataFrame({
    'Emission Type': ['CO2 (Metric Tons)', 'SO2 (Metric Tons)', 'NOx (Metric Tons)'],
    'Georgia (Average Emissions)': [f'{avg_co2_ga}', f'{avg_so2_ga}', f'{average_nox_emissions_georgia}'],
    'Other States (Average Emissions)': [f'{average_co2_emissions_other_states}', f'{average_so2_emissions
})

return results_df
```

	Emission Type	Georgia (Average Emissions)	Other States (Average Emissions)
0	CO2 (Metric Tons)	19250661.36	14200426.57
1	SO2 (Metric Tons)	117729.47	56241.15
2	NOx (Metric Tons)	36730.02	28281.54

Visualization Code

```python
def visual3():
    df=data_parser()
    # getting the total emissions using .agg() and summing up all the emissions
    state_emissions = df.groupby('State').agg({
        'CO2 (Metric Tons)': 'sum',
        'SO2 (Metric Tons)': 'sum',
        'NOx (Metric Tons)': 'sum'
    }).reset_index()

   ## used the same logic to map the states to abbreviations as that is what is predominantly used in the example
    us_states = {
        'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ', 'Arkansas': 'AR', 'California': 'CA',
        'Colorado': 'CO', 'Connecticut': 'CT', 'Delaware': 'DE', 'Florida': 'FL', 'Georgia': 'GA',
        'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL', 'Indiana': 'IN', 'Iowa': 'IA',
        'Kansas': 'KS', 'Kentucky': 'KY', 'Louisiana': 'LA', 'Maine': 'ME', 'Maryland': 'MD',
        'Massachusetts': 'MA', 'Michigan': 'MI', 'Minnesota': 'MN', 'Mississippi': 'MS',
        'Missouri': 'MO', 'Montana': 'MT', 'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH',
        'New Jersey': 'NJ', 'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC',
        'North Dakota': 'ND', 'Ohio': 'OH', 'Oklahoma': 'OK', 'Oregon': 'OR', 'Pennsylvania': 'PA',
        'Rhode Island': 'RI', 'South Carolina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN',
        'Texas': 'TX', 'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA', 'Washington': 'WA',
        'West Virginia': 'WV', 'Wisconsin': 'WI', 'Wyoming': 'WY'
    }

    # used map to convert all the state names ensuring that theres consistency
    state_emissions['Abbreviation'] = state_emissions['State'].map(us_states)

    # choropleth map
    fig = go.Figure(data=go.Choropleth(
        locations=state_emissions['Abbreviation'],
        z=state_emissions['CO2 (Metric Tons)'],
        locationmode='USA-states',
        colorscale='Reds',
        colorbar_title="Total CO2 Emissions (Metric Tons)",
        hovertemplate='%{text}:<br>Total CO2 Emissions: %{z:.2f} Metric Tons<extra></extra>',
        text=state_emissions['State']
    ))
    ## setting title and setting geo_scope to the US (as that is the place where we are depicting the trend)
    fig.update_layout(
        title_text='Total CO2 Emissions per State (1990-2019)',
        geo_scope='usa',
    )

    fig.show()
```



Visualization #3:

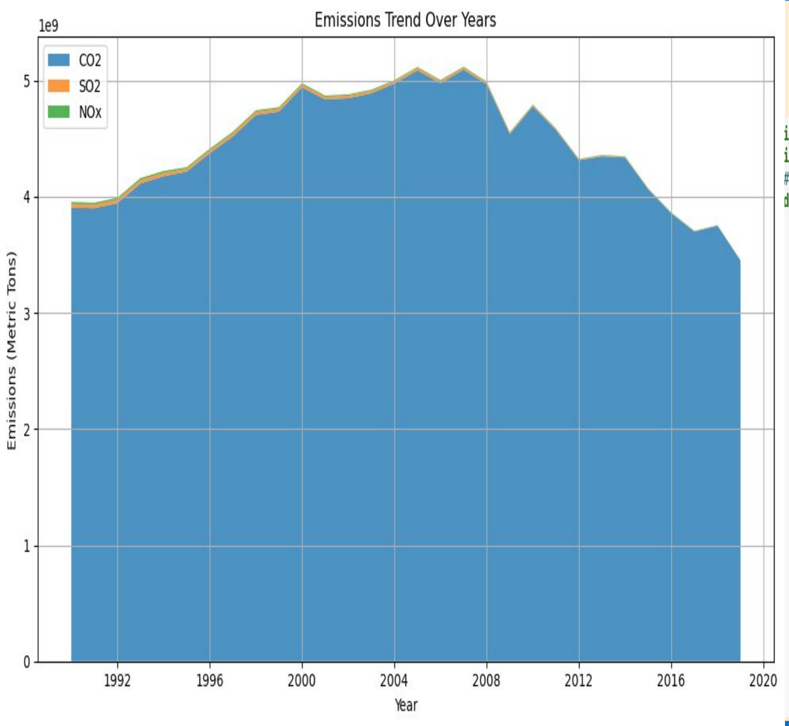
- This shows the total emissions of each state across the United States.
- As you can see, Texas is the greatest contributor followed by Florida.
- I aggregated emissions data per state and created a choropleth map using Plotly to visualize total CO_2 emissions per state from 1990 to 2019 in the United States.

Insight 5: What sector contributes the most to emissions?

- **Used information from the API (Energy Information Administration)**
  - Calculated the sum by grouping by the sector and then the Value that they contribute
  - Created a new dataframe and sorted it by highest percent Contributed to lowest.
- **Conclusions:**
  - Overall, transportation carbon dioxide emissions are the Highest. This is what I expected with the number of personal Vehicles in the US.
  - Learning from this data, we can choose to incorporate more public transport and overall safe energy practices.

	Sector Name	Total Value (in millions)	Percent Contributing to Total Value of Emissions
4	Transportation carbon dioxide emissions	21615.559579	36.85
1	Electric Power carbon dioxide emissions	18362.436669	31.30
2	Industrial carbon dioxide emissions	11743.563994	20.02
3	Residential carbon dioxide emissions	3976.861466	6.78
0	Commercial carbon dioxide emissions	2964.260859	5.05

```
def insight5():
    df=web_parser2()
    # Created a new df to store the total value per sector
    total_per_sector_df = df.groupby('sector-name')['value (in Millions)'].sum().reset_index()

    # Renaming the cols for visual appeal and then calculated the percent contribution for each type of emi
    total_per_sector_df.columns = ['Sector Name', 'Total Value (in millions)']
    total_per_sector_df['Percent Contributing to Total Value of Emissions'] = round((total_per_sector_df['T
    total_per_sector_df = total_per_sector_df.sort_values(by='Percent Contributing to Total Value of Emissi
    return total_per_sector_df
```




```
import pandas as pd
import matplotlib.pyplot as plt
## used matplotlib as an adv req
def visual1():
    df=data_parser()
    # Converted the Year column to datetime, which will later allow me to group by
    df['Year'] = pd.to_datetime(df['Year'], format='%Y')

    # Grouped by 'Year' and calculated the sum of emissions for each year
    yearly_emissions = df.groupby('Year').sum()
    plt.figure(figsize=(10, 6))

    # Stacked area plot
    plt.stackplot(yearly_emissions.index, yearly_emissions['CO2 (Metric Tons)'],
                  yearly_emissions['SO2 (Metric Tons)'],
                  yearly_emissions['NOx (Metric Tons)'],
                  labels=['CO2', 'SO2', 'NOx'],
                  baseline='zero', alpha=0.8)

    plt.title('Emissions Trend Over Years')
    plt.xlabel('Year')
    plt.ylabel('Emissions (Metric Tons)')
    plt.legend(loc='upper left')
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

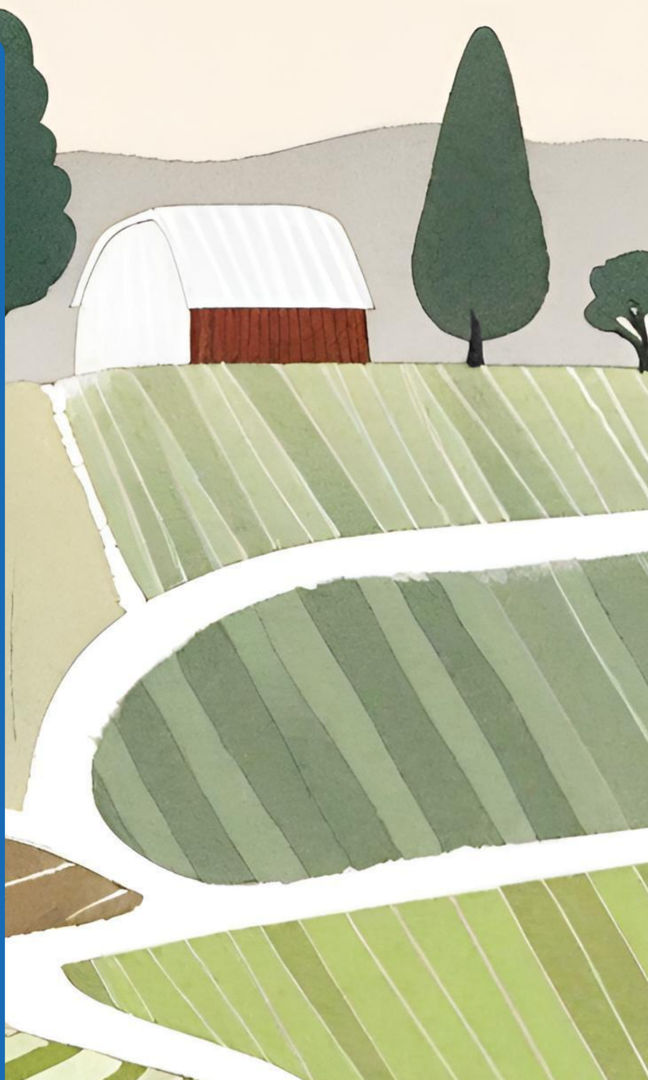
Visualization:

- This stacked area plot depicts the cumulative distribution of the three pollutants.
  The massive abundance of blue shows that CO2 is the leading air pollutant, followed by the sliver of SO2 and then a tiny bit of NOx.
  Used Matplotlib's stacked area plot documentation to derive this plot.

So.. What did I learn?

Technical Skills/Challenges

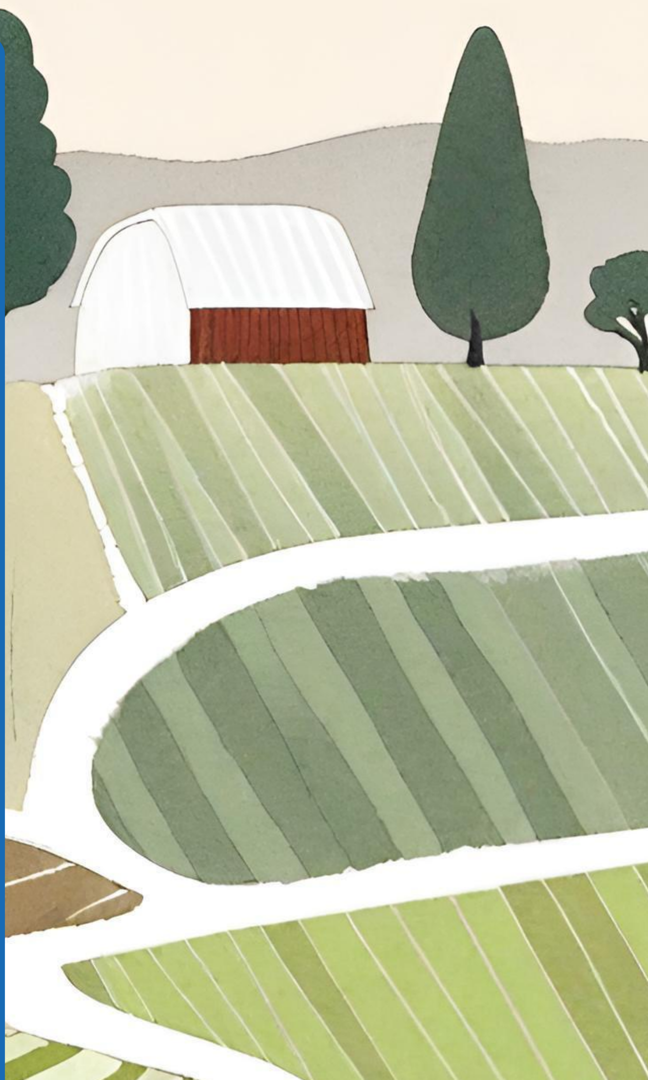
- **I learned a lot of new technical skills and modules within Pandas and Python overall:**
  - Within Pandas, the new modules I learned were .corr(), .merge(), .map(), and more. These modules were challenging to learn by looking the the documentation but they made the implementations of my ideas a lot more efficient.
  - I solidified my knowledge of parsing through APIs and JSONs. Also, I learned how to perform linear regression using sk.learn, which was really interesting. I also learned how to use plotly effectively to create great visualizations.
- **Biggest Challenge:**
  - My biggest challenge in this was visualizing the dataframes In performing large operations. An example of this is my Insight 2 in which I had to find the sum of specific columns across different dataframes and corroborate all the new data onto one new dataframe.
  - Another challenge was also learning the sk.learn module through the documentation. The documentation was pretty in-depth, and I had to take my time to determine the relevance of each line of code.



Overall Project Results

1. Overall, I found out that there is little to no correlation Between the growth of the economy and my perceived increase in emission levels.
2. The emission levels have been decreasing as time Progresses, which suggests that we have been making progress .
3. Georgia, as a state, still has relatively high emissions, so That is something we can control.

Future Project Idea: I plan on continuing this research into correlating economy and nature. However, I think there might be an increased correlation between the two in developing nations, and that will be my future research .



Thanks for a Great Experience