DS 2002 Final Project Reflection Paper

Benjamin Yeh, Jaelyn Do, Jonah Lee, Kevin Lam

Data Selection

Finding a good dataset to use was a challenging process that required more exploration and consideration than we expected. Initially, we had to determine the type of data that would be both meaningful and feasible to analyze, which proved to be more difficult than expected. It was hard to come across a dataset that was not only in megabytes, but also aligned with the type of quantitative data we wanted to work with for our project. We initially experimented with various datasets, such as tennis rankings and COVID-19, but soon found that they either lacked depth or posed challenges in terms of data structure and usability. Another major consideration was ensuring that the dataset we chose would allow us to derive statistical insights that were relevant to societal trends and issues. We wanted our findings to have broader implications that could connect to real-world challenges. After some trial and error, we eventually settled on two datasets related to carbon emissions. These datasets stood out to us because they not only provided quantitative data, but we also found that they were relevant to societal trends on the environmental impacts of urbanization.

ETL Setup and Implementation

While designing the Extraction Transformation Loading (ETL) pipeline, there were a couple challenges that arose. First was examining the data itself to figure out how to approach designing the ETL pipeline. The data is on CO2 and Greenhouse Gas Emissions in the world and contains many attributes of different types of emissions, many of which we were not aware of. Because of this, we conducted some minor research on the types of emissions and decided whether we wanted to keep them in our dataset for observation. Secondly, it was difficult to decide what method of cloud storage we wanted to use. Originally, we decided to use SQL through Google Cloud Platform because we were all familiar with relational databases. However, after analyzing the data further, we found that it made more sense to use a NoSQL database like MongoDB instead. This is because our data did not heavily rely on structure and it didn't have defined relationships where using a relational database would be beneficial. Overall, designing the ETL pipeline did not present many really tough challenges, but there were several things that needed to be addressed in order to ensure our pipeline was efficient and adaptable. Through the process of designing our ETL pipeline, we were able to learn lessons of data considerations and the situations to use NoSQL databases vs SQL databases.

The implementation of the ETL pipeline presented several challenges. The first was deciding how we wanted to retrieve the data itself. Our data was found on GitHub as a CSV file. Our original thought was manually to download the CSV file and place it on our repo to be read in. However, this required extra steps and created potential for human error. After discussing and looking into it, we found a way to directly download the CSV file in our Python code using a raw GitHub link. This made things simpler as we were able to automatically download the file and read it using a GET request in Python, rather than manually downloading and uploading it ourselves. Another challenge we faced was writing the code to transform the data. This required further research on the attributes of our datasets in order to clean our data properly. After examining our data further, we also found a lot of missing values. However, most of these missing values came from early years, where less emissions data was being recorded. Due to these missing values, we decided to filter the data in our transform step, keeping only entries from 1950 - 2023. Other missing values were handled by dropping the entry entirely or filling them in with the median. Another challenge we faced was splitting the data accordingly. Our dataset was a compilation of many datasets. The 'country' column includes territories, continents, and socioeconomic classes. This means we had to figure out a way to split this dataset accordingly into the 4 datasets we wanted to use. We tackled this by creating arrays with the names of countries, names of continents, etc, where Python code could split the data and create the separate datasets. Finally, while implementing our ETL pipeline, we had to learn how to use MongoDB to properly store our data. This is discussed further in cloud storage. Some lessons we learned through the ETL implementation was how to use API requests to get data on GitHub, how to wrangle data, specifically how to handle missing values, and how to use MongoDB for cloud storage.

Analysis

Some of us in the group had a background in statistics while some of us have never taken a statistics class before. We settled on doing bootstrapping to find confidence intervals for the two different R squared values, the relationship between CO2 emissions and countries' population and the relationship between CO2 emissions with the countries' GDP. We thought this was the best way because we wanted to look closer at the correlation between these two sets of variables, and bootstrapping is a method that allows you to find the confidence interval for a measurement that might not traditionally be possible. The code was quite simple for calculating the interval since it was not very technical and some of us had experience working with bootstrapping before.

Cloud Storage

The cloud storage portion of our project presented some challenges. First, as discussed above, was which database type to use, SQL vs NoSQL. Ultimately, we decided to go with NoSQL via MongoDB Atlas. The main challenge when it came to cloud storage was learning how to use MongoDB because none of us had experience working with it outside of this class. In particular, setting up the database credentials and Python drivers was not very smooth. After creating a database user, we attempted to ping the database using the Atlas URI but ran into connection and permission issues. After some debugging, we discovered there was an issue with the permissions set on the database user. After setting the proper permissions and fixing the URI, we were able to successfully ping and connect to our database on MongoDB. Another challenge we ran into was not wanting to put our URI with credentials in our code in the repo itself because of security reasons. Luckily, environment variables were an easy fix and using them in Python is also straightforward. Through the implementation of cloud storage in our project, we learned how to use MongoDB effectively.