The roles of each team member are described below including a description of each script that they wrote.

Name	Module	Tasks
Gabe	data_compilation scrapers/ county_care	ccare_scraper.py: • First function, **scrape_ccare**, runs 4122 initial queries on County Care's find a provider tool. Each query is a combination of one of the 18 provider type options on their website and one of 229 zip codes around Cook County. • Next function, **gen_re_scrape_list**, checks if any of those searches are "packed" that is all responses are in that zip code and returns a list of search parameters to replicate those searches. • The final function, **re_scrape**, runs two sets of additional searches on the packed zip code-provider type pairs: (1) first specifying the gender of the provider and (2) if the search is still packed, querying by each specific sub-specialty. scrape_cleaner.py: • The function **ccare_scrape_unpacker** goes through the two saved JSON files of scraped county care data and compiles them both into a list of dictionaries in "scraped_data/ccare_results_clean.json." • The function **scrape_to_merge** compiles the whole scraping-unpacking process. It is written to be executed from the command line. utils.py: • Only function is **ccare_post**. This function makes a post to county care's find a provider tool. It randomly sets a sleep and selects a different browsing agent for each post.
	Data_compilation/ __main__	__main__: • Executes all of the web scraping, cleaning and merge functions and returns usable data sets in the data_output folder.
Paula	Data_compilation clean/	clean_impact.py: • This script includes the function **clean_impact** that cleans and transforms raw IMPACT data, in text form, and transform it into a structured DataFrame. This process is done by reading the file by rows and identifying columns, creating a dictionary by provider, and saving all information. After that, it continues separating name and address in more distinct columns. It also shortens the zip code to 5 digits. This script finishes by dropping possible duplicates, and extra columns. clean_npi.py: • This script includes two functions. The **clean_npi** function cleans and preprocess NPI data from a CSV file based on selected columns, contained in another data frame. It renames columns to simplify, checks for atypical phone numbers (and fills in with NaN values), and fills missing phone numbers in cases where is a second non atypical option. The **match_npi** function extract to match unique phone numbers from preprocessed NPI data and creates additional rows to multiply match options based on different phone numbers. This script finishes by dropping possible duplicates, and observations with NaN values. clean_scrap.py:

		• This file includes two functions. The **clean_scrap** function cleans and preprocess scrape data (CountyCare) from a JSON file based on selected columns, contained in another data frame. It renames columns to simplify, check for atypical phone numbers, and fills in with NaN values. It saves the total number of scrap results. It finishes by dropping possible duplicates. The **match_scrap** function, extracts and matches relevant columns from preprocessed scrape data. It drops rows with NaN values and possible duplicates.
	Data_compilation linkage/	match.py: • This script contains one main function and three helper functions. The match function performs matching operations on the scrape (CountyCare) dataset using official information to identify health providers. It takes all the clean data bases and performs three approaches. Finishes by dropping extra columns, and dropping duplicates based on first name, last name, and phone number (original identifiers). - (1) matching approach to fill missing 'npi' values in df_scrape using df_npi based on exact matches of last name and phone number. - (2): matching approach to fill missing 'npi' values in df_scrape with df_impact, exact zip codes, last name, and similar first name. - (3): matching approach to fill missing 'npi values in df_scrape with df_impact, subsetting by zip code and looking for similarities in first name, last name, and address. final_data.py: • This script produces data frames used by the dashboard, taking paths according to the user (in case of files outside of the app).
Magdalena	data_viz/	data_analysis.py: • Contains two helper functions one that uses the data obtained in the linkage to create the dataset of all unique observations considering only primary care providers and specialists. The other makes calculations and creates the dataset for visualizations in the dashboard. viz_graphs.py: • Contains three functions that create each of the Dash Graphs used in the dashboard. One graph for the total matches and non-matches for each provider type, the other with the percentage of matches for each provider type, and the last create a choropleth graph with the non-matches for each zip code.
	dashboard/	dashboard.py: • contains one function that creates the cards with the main numbers of the scraping and linkage process, and the context of Medicaid patients. Then integrate all the functions from *data_analysis.py* and *viz_graphs.py* into an application that creates a dashboard with the context all the whole project, uses a markdown for the abstract, and cards with the main numbers and shows the graph that represents the data in a more disaggregated way to make a more detailed analysis of the findings after the scraping and linkage with the NPI and Impact databases.

.