ADSDB - Analysis of Chronic Diseases and Alzheimer's Disease in Texas (USA)

Andreja Andrejic, Mateja Zatezalo
UPC Barcelona

October 31, 2024

Contents

Platform Access

Google Drive

Google Drive is used as the platform for organizing the development of the project. Python scripts, databases and metadata are stored in different folders in our shared folder.

Google Colab notebooks are divided into different zones as described in the project statement. Along with those, there are notebooks profiling each database and data quality notebooks.

Access the Google Drive through the following link:
https://drive.google.com/drive/folders/1KUk3pbdo3_kJhazxgLJ4aDoZKU17gofu?usp=share_link

GitHub

We used GitHub for code versioning. After the core of the project was written in Google Colab notebooks, we diverted to GitHub in order to provision code orchestration. The access to the GitHub repository is through the following link:
https://github.com/andrmrr/ADSDB24.git

1 Introduction

The project of the ADSDB course entails delving into challenges of integrating Data Science into operational frameworks, which has a goal of offering competitive advantages to companies and organizations. With this system, we aim to help create a useful data-centric system in the healthcare field.

This project consists of two parts. First part consists of creating a data management backbone that integrates two different data sources. Operations and data management is automated and organized in various zones. Additionally, generic data quality and exploitation is implemented to facilitate the extraction of useful insights in the second part.

This part of this project entails the implementation of the Data Management Backbone, which is divided into 4 different zones: *Landing*, *Formatted*, *Trusted*, and *Exploitation*. Finally, the *Operations* Zone is implemented to orchestrate the whole system architecture.

2 Data Selection

For this project, we selected two datasets from two different data sources. One regards **"Alzheimer's Disease and Healthy Aging Data"** from the US Department of Health and Human Services, while the other regards **"U.S. Chronic Disease Indicators"** from the same US department.

These datasets were chosen for this project since they hold interesting data about various health problems in the US. They are practical regarding the successful management of data. They both have *Year* columns and *Stratification* columns, which can be joined. On the other hand, this data can be analyzed and used to retrieve meaningful insights. Insights in this field could improve patient outcomes and healthcare strategies.

This data was extracted from the website 'data.gov' [2] which is a well-known website that contains hundreds of thousands of data sources for various topics in United States. This website holds data from specific US states, but also data collected across the whole country.

The chosen datasets hold information for multiple US states and span across several years. Every row contains information about the cause of disease, information about demographics of patients such as their state of residency, age group, ethnicity etc. These datasets are very large and hence could slow the analysis process of this project, we decided to limit the domain to the state of Texas. Information within this state is plentiful, and would hold enough data for the second part of this project.

2.1 Data Sources

2.1.1 Alzheimer's Disease and Healthy Aging Data

This dataset is derived from BRFSS (Behavioral Risk Factor Surveillance System) [1] survey which were conducted from 2015 to 2022. BRFSS collects data via annual telephone surveys where they emerge public health issues, health conditions, risk factors and behaviors across the United States.

This dataset will be used to analyze different causes of Alzheimer's disease and correlation within different demographics of people in this state.

- Location: The data we use for this project is filtered for the state of Texas.
- Version: Data is taken for years 2017 and 2021, where the similarities and differences will be analyzed in the second part of the project.
- Attributes: Year, Location, Data Source, type of health problems, including Class, Topic, Question, values of data, Stratifications which group people in different age groups, races, ethnicities etc.

2.1.2 U.S. Chronic Disease Indicators

This dataset is derived from the chronic disease indicators outlined in the Morbidity and Mortality Weekly Report titled "Indicators for Chronic Disease Surveillance — United States, 2013" available prior to the 2024 CDI update. It includes a set of 124 indicators, facilitating data collection and reporting across states and territories for key public health metrics. The Chronic Disease Indicators (CDI) website offers access to specific state data, the most recent data releases, and additional resources and information on these indicators.

This dataset will be used to examine different chronic diseases in different demographics, and how they relate to Alzheimer's disease causes within the state of Texas.

- Location: This dataset is very large as the original contains more than one million rows, hence the data we use for this project is filtered for the state of Texas.
- Version: Data is taken for years 2017 and 2021, the same as for Alzheimer's dataset.
- Attributes: Year, Location, different Data Sources, type of health problems, including Topic of disease, Question, values of data, Stratifications which group people in different age groups, races, ethnicities etc.

2.2 Dataset Integration

For further analysis of these two datasets, their integration will be necessary. The main goal of this is to enrich our understanding of health outcomes and risk factors associated with chronic diseases, including Alzheimer's disease.

By joining these datasets by the attribute "YearStart" we ensure that the analysis is temporally aligned, which provides analysis over trends or patterns. This allows us to investigate temporal correlations and potential relationships between Alzheimer's prevalence and other chronic disease causes and metrics.

2.3 KPIs

The analytical backbone of this project will be focused on providing valuable insights into health trends, risk factors and outcomes. Some potential KPIs or metrics we can consider for the analysis include:

- Prevalence Rates: Annual prevalence rate of chronic diseases and Alzheimer's disease per 100 000 citizens. This could help in understanding of commonality of these diseases within Texas.
- Demographic Trends: Examination of age, gender, ethnic, and racial demographics trends over time to identify if some subgroups contain outliers or similar results. For example, percentage of elderly people (aged 65+) with Alzheimer's compared to ones with chronic heart conditions.
- Risk Factor Analysis: Measure of correlation of various risk factors (smoking, high blood pressure, alcohol use) with the prevalence of these diseases. Identifying correlations can provide insights regarding potential policies in healthcare.
- Hospitalization Rates: Hospitalization rates and healthcare utilization for Alzheimer's compared to other chronic diseases per 100 diagnosed cases annually.

When the project reaches the analytical backbone stage, these KPIs could potentially be changed and additional KPIs could be added.

3 Data Management Backbone

This section further explains each zone of the Data Management Backbone, as described in the Introduction.

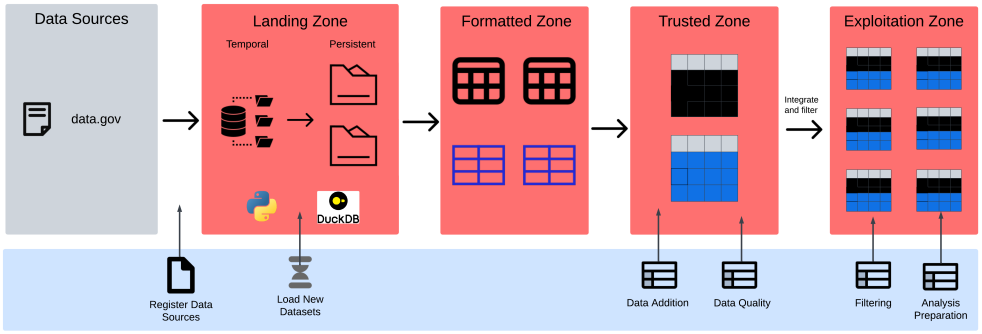

Figure 1: Simplified Data Management Backbone diagram

3.1 Data Ingestion

The "raw" data is stored in csv files locally. The original dataset include only one version, so for the sake of the project, we separated it. The separation was done with respect to the year the data is gathered. This way we have multiple versions of the dataset that can be loaded individually.

3.2 Landing Zone

This zone consists of two zones: *Temporal* and *Persistent* landing zones.

Temporal landing zone is a single directory where datasets are ingested into from external sources. It is emptied whenever its datasets migrate to the persistent landing zone.

Persistent landing zone is a directory with subdirectories for every dataset. This way, multiple versions of the data can be stored and easily accessed. Therefore, every loaded dataset gets a timestamp appended to the file name for simple version checking. Datasets saved in the persistent landing zone are never deleted, unless a major change in the backbone takes place.

3.3 Formatted Zone

In Formatted Zone, we move on from Landing Zone by creating a DuckDB database that stores needed datasets for further analysis. In this zone new datasets from the Landing Zone are checked and loaded into the database. Following processes ensure the execution of this stage:

- Whenever the *formatted loader* is executed, it checks for new versions of the existing dataset and for new datasets in the persistent landing zone.
- New versions of dataset and new datasets are inserted into the database, so that each version is an individual table. DuckDB automatically infers the schema from the dataset. If there is no new data, no action is executed.

This process can be automated, by running a thread that checks periodically for new datasets, hence no user input is required. For the requirements of the project, these actions are manually performed from the user interface. In case of errors or no new datasets found, the database stays the same and no changes are executed. For the purpose of keeping track of any new tables, each time a new table is created, its identifier is saved a special JSON file. This will be necessary for the following Trusted Zone.

3.4 Trusted Zone

The *trusted loader* handles migration from the Formatted Zone to the Trusted Zone. For each dataset, different versions of the dataset (tables in the formatted zone) are merged into one, creating larger tables containing information from different versions. This is accomplished in the following way:

- Only new tables from the Formatted Zone are loaded to be appended to the existing tables in the Trusted Zone. We keep track of the new tables via a special JSON file.
- Since there could be multiple versions of incoming datasets, they are formatted, so each of them has the same schema (superset of their original schemas). They are then combined to be added to the database of the Trusted Zone
- Each incoming dataset goes through a list of data quality processes.
- Finally, the prepared data is added to the existing tables, where the existing data has already been prepared. Once the data is combined, any duplicate rows are removed.

3.4.1 Data Quality

Data quality processes ensure the data that the analysts will be using is prepared for all kinds of algorithms and manipulations. Since different datasets have completely different contents, each dataset goes through a specific set of processes that clean and prepare its data. For the purpose of the project, we have two distinct datasets and they each have their preprocessing files. If another dataset was to be added to the data management backbone, it would be necessary to define its own data quality processes. Nonetheless, they all go through a same set of steps:

- The first step was to handle missing values. For our case, with the nature of the data and its features taken into consideration, missing data was sometimes imputed and other times rows or columns were removed.
- Next, features without any information or with redundant information were removed. Since, we are using a sample of the original data, i.e. data specific to the state of Texas, columns in our dataframes regarding location have a single value throughout. These columns were removed, but these ubiquitous values were saved in a separate JSON file as metadata.
- In some cases type conversion was necessary, specifically in our case, features regarding years were converted to categorical
- Finally, rows with identical values in all columns were removed

Additional steps were considered, but were not applicable in our datasets, such as scaling, normalization and spellchecking.

3.5 Exploitation Zone

The Exploitation Zone contains a new DuckDB database with tables organized with semantical meaning. Datasets from the Trusted Zone are transformed by the *exploitation loader* and placed in the Exploitation Zone.

As mentioned before, the most logical approach is to integrate our datasets by the column 'YearStart'. The content of the datasets are quite similar and can be horizontally combined (JOIN). Each row in our two datasets is stratified with regard to two different categories, Gender and Race/Ethnicity. We used this as context for the Exploitation Zone and divided all the data by applying these filters.

An additional data quality process was necessary for this part, since the stratification values were not identical in both datasets. Thus, an extra column *Standardized_Race_Strat* was added with the reconciled values of the stratification feature.

Finally, the database contains seven tables, namely female, male, American Indian or Alaska Native, Asian or Pacific Islander , Black, White and Other. Each table contains data form the Alzheimer dataset, Chronic Disease Indicators dataset or both.

3.6 Profiling

Profiling is done in a general way, so that it can be applied to any dataset, regardless of its structure. It includes:

- Basic feature information and statistics, such as quartiles and mean variance for numerical features, count and most common for categorical features and number of missing values.
- Plots showing frequency/count of each feature.

3.7 Drawbacks of the solution

Limiting ourselves to the scope of the project (putting aside big data and distributed systems), we identified different segments of our project that can be improved. These include:

- The user interface could have a text-field to choose which version(s) of the data will be ingested.
- Profiling should not be completely general, but have some specific aspects for each dataset.
- The Data Quality processes can be improved through more trial and error.
- With some refactoring, better code reusability can be accomplished.

Overall, given the time and the scope of the project, we are fairly satisfied with our solution. Our focus was on error handling and automating as much as possible, which we feel are executed well.

4 Operations Stage

This section describes the operations zone of the project, orchestration, GUI and performance monitoring.

4.1 Orchestration

As a part of project requirements, the Python notebooks from Google Colab have been migrated to our GitHub repository, as mentioned in Section 1.

Within the GitHub repository, we set a critical script named *orchestration.py*, which plays the main role in the pipeline. This script is designed to orchestrate the entire Data Management Backbone, while coordinating between all zones and scripts described in Section 4.

Propagation of data from each zone is done manually by the user. The whole pipeline can be executed by running *orchestration.py* or the transfer to each zone can be done individually using the GUI we created. It is run by the *gui.py* script. These processes could be done automatically, by creating a background process in the background that executes the pipeline at given intervals or when it detects a new dataset.

An important factor to mention is that the reproducibility of our system is guaranteed as the backbone does not contain any random processes.

4.2 GUI

A Graphical User Interface (GUI) was built to provide user interaction and display of the Data Management Backbone of this project.

The GUI was developed using Python's *tkinter* library, which provides tools for creating graphical user interfaces. This interface is created in the file "gui.py".

User Interface contains buttons devoted to executing every zone of the Data Management Backbone. By clicking the buttons, a label informs the user if the action was successful and what is the current status of the database. The interface also shows performance metrics, described in the following subsection.

4.3 Performance Monitoring

The script for the GUI from previous section also contains a functionality to monitor and display some system performance metrics such as CPU and memory usage, which is important for efficient processing of the Data Management Backbone.

A separate thread is created to handle system monitoring without blocking the main GUI thread. This ensures that the GUI remains responsive even while fetching system stats. The *sys_monitor* function continuously fetches system performance data using the *subprocess* module to execute the *top* command, which is a tool for monitoring system processes and resource usage.

Metrics are displayed on labels in the interface and they are updated every 5 seconds to reflect on the current performance.

By using threading to separate performance monitoring from the main GUI loop, responsiveness is remained in the application. With these elements, the GUI allows for user interactions, while also informs the user of the system's performance, to ensure optimal operation of the Data Management Backbone.

References

[1] Behavioral Risk Factor Surveillance System — cdc.gov. https://www.cdc.gov/brfss/index.html.

[2] Data.gov Home - Data.gov — data.gov. https://data.gov/.