ADSDB - Analysis of Chronic Diseases and Alzheimer's Disease (USA) - Part II

Andreja Andrejic, Mateja Zatezalo
UPC Barcelona

January 10, 2025

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. The second part consists of creating a data analysis backbone that uses data from the data management backbone, processes it and trains an ML model in order to create predictions based on the data. Specifically, we want to predict "Percentage of older adults who report having a disability (includes limitations related to sensory or mobility impairments or a physical, mental, or emotional condition)" based on health data in the USA. Data analysis is organized in various zones and operations automates everything including data management from the previous part of the project.

This part of this project entails the implementation of the Data Analysis Backbone, which is divided into 4 different zones: *Analytical Sandbox*, *Feature Engineering*, *Model Generation*, and *PCA*. Finally, the *Operations* part is implemented to orchestrate the whole system architecture.

2 Context

We had made serious changes in the data management backbone. The biggest issue was not having enough individual cases with the same set of features (including missing data), so that we could properly train the model. Therefore, we altered the subset of the data we ingest into the landing zone. Note: a row in the original datasets represents a question regarding health on the subset of the population of a US state in a specific year. The subset relates to race/gender and there are total of 7 such corresponding stratificators.
Since the original datasets are too big to be efficiently processed in this project, we previously took only data regarding the US state of Texas and years 2017 and 2021. That left us with only startificators and two different years to represent our cases. This is not enough examples to train an ML model.
Hence we decided to include all US states and instead remove around 70% of the questions, which left us with around one hundred different questions. Finally, the data is of manageable size and we had enough cases to properly train a model.
This change required minor changes in the Landing, Formatted and Trusted zone, but warranted a complete redo of the exploitation zone. Previously, in the exploitation zone we split data into tables based on the stratificators (race/gender). However, this was not aligned with the shape of data we wanted for the analysis. Instead, in this zone we split the trusted database into tables based on the topic of the question. A topic contains one or more different questions and there are a total of 25 topics and tables generated in the exploitation zone.
Finally, we changed the file structure of the data management notebooks. We placed each notebook in the corresponding (sub)folder and placed repeatable code from the trusted zone into distinct notebooks in order to improve reusability.

3 Data Analysis Backbone

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

3.1 Analytical sandbox

Analytical sandbox is used to transfer a subset of the data from the data management backbone (specifically exploitation zone) to be used for analysis. Here we took relevant columns from the tables of the exploitation zone and renamed some of them in order to more easily process the data in the feature engineering zone. We did not remove any rows.

3.2 Feature Engineering Zone

This zone consists of two zones: *Feature Generation* and *Data Preparation* zones. *Labeling* was not necessary in our case, as we had a clear output variable.

3.2.1 Feature Generation Zone

In this zone, we created and formatted some features and also combined the tables divided by topic into one single table. All processes are defined as separate functions and kept in a single notebook, so that they can be iteratively executed in a single loop for all tables.
Firstly, we noted the stratificators that existed in both of our datasets and reconciled those categories, so that we can combine them. Then we set the precision of float values for latitude and longitude, so that they would match properly for tables from both datasets.
When the tables were ready, we combined them in a single one. Each row of the combined table has latitude, longitude, year and stratificator as variables on which they are joined. And each row of the original 25 tables has two specific values, namely the question and the value (answer). All the different questions are vertically combined for all combinations of the join variables. A new column is generated for each question containing its value (the answer of the question). Since the questions themselves are long, making them annoying to deal with in the dataframe, the name of each column is the ordinal number of the question (Q1, Q2, Q3 etc.) as well as QTarget for the target column. The actual questions are irrelevant for further data preparation and model training. The actual questions and their order is kept in a separate JSON file to be used for the final analysis of the prediction.

3.2.2 Data Preparation Zone

This zone contains two processes: categorical column encoding and the handling of missing values. There are no multivariate outliers in our data, which is not a surprise since we have almost 100 columns at this point and the "curse of dimensionality" says that with high-dimensional feature space spreads the data making all data points far from each other. No scaling or normalization is required for ExtraTrees / Random Forest Regressor. Also all our data is in percentages (floats between 0 and 100).
The only categorical variable we have at this point is the stratificator, six in total. We used one-hot encoding to expand this column into six binary ones, one for each category: *Male*, *Female*, *White*, *Black*, *Hispanic* and *Overall*.
Missing values handling had two parts. In the first part we analyzed the number of missing values in each column. First we removed all rows with missing target variable. This brouht the number of rows from 600 to 500. At this point we had 80 predictor variables and we decided to remove all columns (questions) with more than 20 rows with missing values. This left us with the final shape of our dataframe: **(500, 51)**.
The rest of the missing data we imputed. For this we first had to split the data into train and test sets in order not to introduce a bias to the test set. We first tried MIMMI imputation algorithm, because it recognizes non-linear relationships well, but Google Colab could not handle it. So, we ended up using K-means imputation. Even though it cannot encode more complex relationships in the data, it is fast. Since the values for different examples do not differ greatly and we do not have a lot of missing data, we agreed that K-means with 3 neighbors will do a good enough job of imputing the missing values.

3.3 Model Generation

In this section we trained the model and simultaneously validated it using cross-validation. We decided on using the Extra Trees Regressor, which is a faster version of the Random Forest Regressor. We chose it for its performance and versatility and how well it handles high-dimensional data.
Because of the limited computation power of Google Colab, we ran the cross-validation multiple times with different parameter choices. The final version gave us the best model of the ones we have tried. The criteria of the grid search were MSE and R^2. The final model had the following parameters.

Parameter	**Value**
criterion	squared error
max depth	None
min samples leaf	1
min samples split	2
number of estimators	1000

Table 1: Extra Trees Parameter

The grid-search of parameters and training of the model took between 15 and 20 minutes depending on the parameters which were searched for.

The following table shows the MSE and R^2 on the test set.

MSE	$\mathbf{R^2}$
16.133625266470652	0.6233233267128164

Table 2: Model Performance Metrics

3.4 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.

4 Model Result Analysis

4.1 Main Model

In this project, we analyzed two datasets: one focusing on Alzheimer's disease and the other on chronic diseases. Our primary goal was to predict the percentage of people with disabilities caused by Alzheimer's disease and chronic diseases. These predictions were based on a variety of variables such as location, demographic stratifications (e.g., age, gender, race), and the specific questions/topics related to health indicators. Additionally, we aimed to explore how the variable Questions (representing different health topics) impacts the predictive power of our model and its ability to capture relationships within the data.

Main metrics of the model performance are shown in Section 3.3. Here we focus on result analysis.

Actual vs Predicted values

Figure 2 shows the relationship between the true data values and the predictions generated by our model. The alignment with the red line suggests that the model performs well in predicting the percentage of people affected by disabilities due to Alzheimer's and chronic diseases, with most predictions being close to the true values. There are some deviations which potentially suggest areas for improvement or limitation of the model.

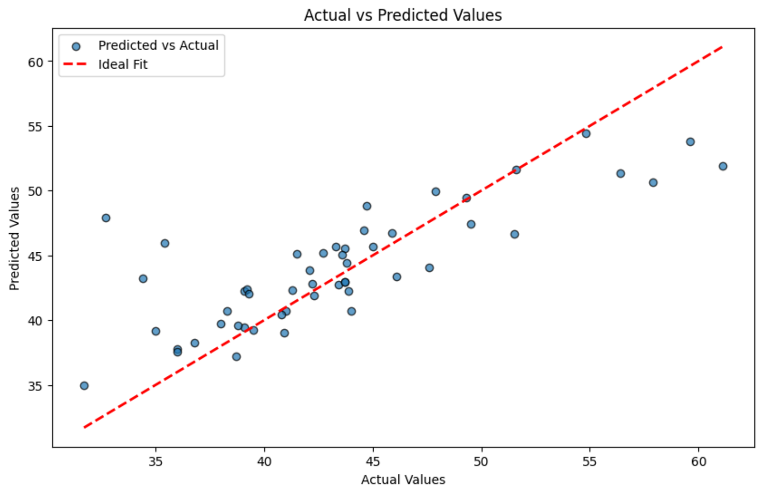
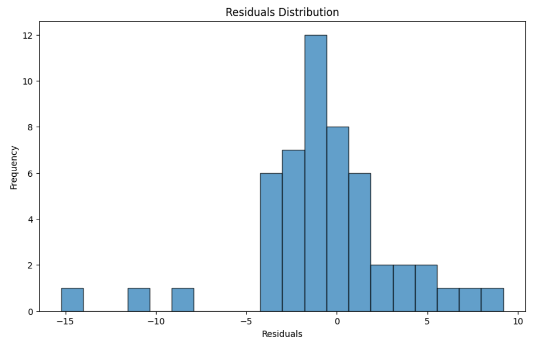


Figure 1: Actual vs Predicted values

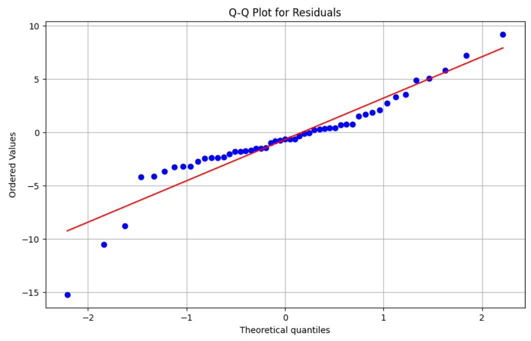
Residual analysis

The histogram of residuals (Figure 3a) provides insights into their overall distribution of residuals. The residuals are centered around 0, which is a good indication that the model does not exhibit significant bias. The slightly skewed distribution and the presence of extreme values at both ends suggest that the model struggles with certain data points, possibly due to noise or complex patterns in the data that were not captured by the model.

Q-Q plot (Figure 3b) compares the residuals' distribution to a normal distribution. The points closely follow the red diagonal line, indicating that the residuals approximately follow a normal distribution. However, deviations at the tails suggest potential outliers or non-normality in the data. This could indicate that the model does not perfectly capture certain aspects of the data or that the data has outliers affecting the predictions.



(a) Residual Distribution



(b) Q-Q Plot

Figure 2: Residual Analysis plots

Feature importances

The feature importance analysis provides valuable insights into which variables most strongly influence the prediction of the target variable — **the percentage of older adults who report having a disability caused by Alzheimer's or chronic diseases.**

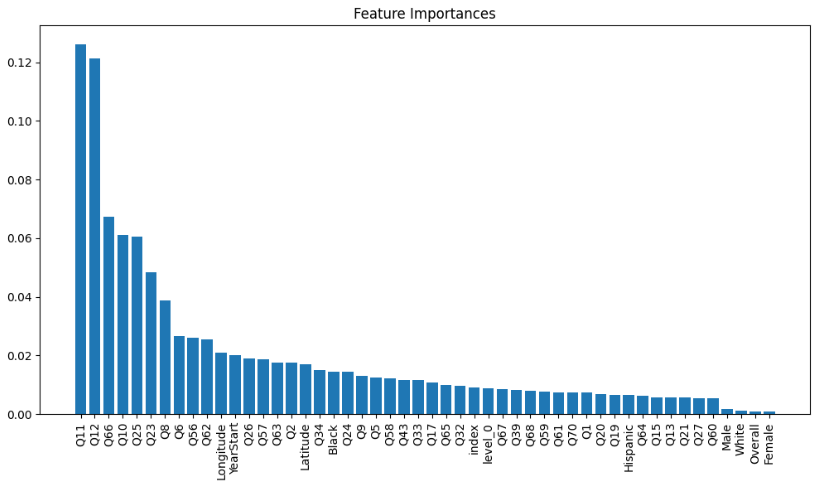


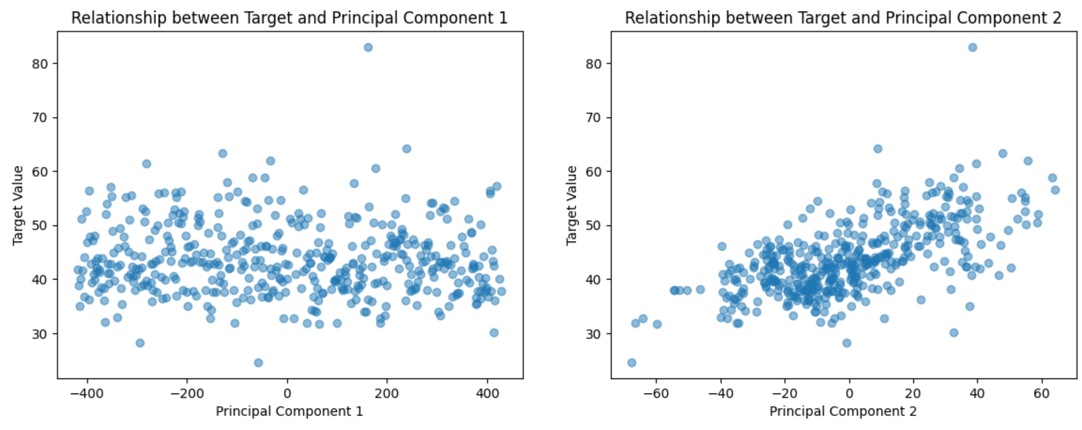
Figure 3: Feature importances

- **Top influential features include:**
  - Q11: Percentage of older adults who self-reported that their health is "fair" or "poor" is the most important predictor, with a feature importance score of 0.126. This highlights that individuals' self-perception of their overall health status plays a significant role in predicting disabilities.
  - Q12: Percentage of older adults who self-reported that their health is "good," "very good," or "excellent" ranks second with a score of 0.121. The dichotomy between "fair/poor" (Q11) and "good/excellent" (Q12) health likely provides complementary insights for prediction.

Geographic features such as Longitude and Latitude do not play significant roles, as their importance scores are lower (0.021 and 0.016, respectively). Year of collecting data also does not show a trend or pattern, with a score of 0.02. Stratificators, such as Male (0.0016), White (0.0012), and Female (0.0008), show very low importance scores, indicating that gender and race contribute minimally to predicting the target variable in this model.

4.2 Principal Component Analysis

- **First Plot: Relationship between Target and Principal Component 1**
  - The scatterplot shows a weak or no clear relationship between the target value and the first principal component (PC1). The points are spread almost uniformly across the range of PC1 values. This suggests that PC1 may not be significantly correlated with the target variable.
- **Second Plot: Relationship between Target and Principal Component 2**
  - The scatterplot shows a stronger relationship between the target value and the second principal component (PC2). There is a noticeable trend where higher PC2 values are associated with higher target values. PC2 appears to be more relevant for predicting the target variable compared to PC1. This suggests that the variance captured by PC2 in the input data is more closely related to the target variable, indicating its potential importance for the prediction task.



(a) Target and Principal Component 1 (b) Target and Principal Component 2

Figure 4: Relationship between Target and Principal Components

5 Operations Stage

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

5.1 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 both backbones.

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.

5.2 GUI

A Graphical User Interface (GUI) was built to provide user interaction and display of the Data Management and Data Analysis backbones 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 both backbones. By clicking the buttons, a label informs the user if the action was successful and what is the current status of the database. Finally, the interface shows output of our model. The interface also shows performance metrics, described in the following subsection.

5.3 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.

6 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:

- **Data Imbalance and Representativeness:** The subset of the data may not adequately represent the population due to the removal of around 70% of questions and limiting the dataset to specific years and states initially. This could lead to biases in the trained model.
- **Limited Handling of Missing Data:** The use of K-means imputation for missing data, while computationally efficient, may not capture complex relationships in the dataset. This might introduce inaccuracies in the imputed values, potentially affecting model performance.
- **Simplistic Feature Engineering:** The feature engineering approach focuses primarily on one-hot encoding and renaming columns but does not explore advanced techniques like interaction terms, feature selection, or dimensionality reduction, aside from PCA.
- **Model Dependency and Overfitting Risk:** The Extra Trees Regressor, while robust and high-performing, is sensitive to the quality and quantity of data. With a relatively small dataset (500 rows), the model might overfit, especially with no regularization.

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.

7 Conclusion

The completion of this project has demonstrated the successful integration of a robust data analysis pipeline to address complex healthcare data. By utilizing datasets focused on Alzheimer's disease and chronic disease indicators, our primary aim was to predict the percentage of individuals affected by disabilities stemming from these conditions. This goal was achieved through a comprehensive framework that incorporated modeling, feature engineering, and visualization of model results.