Software Design Document

for

Comorbidity and Genetic Factors and their Impacts on Patients with COVID-19

Version 1.1.4 approved

Prepared by

Rohan Chatterjee	Project Lead
Riese Atianzar	User Interface Designer
Francisco Contreras	User Experience (UX) Engineer
Luis Gonzalez	User Experience (UX) Engineer
Emily Gonzalez	Data Visualization Engineer
Ting Fung Ha	Technical Writer
Juan Hernandez	Data Analyst
Carlos Hernandez	Data Gatherer/Developer
Jimuel Cedrick Julaton	Data Visualization Engineer
Chen-Ching Lin	Data Visualization Engineer

Faculty Advisor: Dr. Navid Amini

Sponsor : Vodafone

Project Liaison : Haley Kirk

21 November 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Ting Fung Ha	21 November 2022	Document created	1.1.1
Ting Fung Ha	27 November 2022	First draft completed	1.1.2
All Members	28 November 2022	First Group Revision	1.1.3
Rohan Chatterjee, Ting Fung Ha	1 December 2022	Revision with Project Lead	1.1.4
Ting Fung Ha	1 March 2023	Part II of the project, First Draft	1.2
Ting Fung Ha	7 March 2023	Part II of the project, Second Draft	1.2.1
All Members	8 April 2023 - 21 April 2023	Group Evaluation and reviews	1.2.2
Ting Fung Ha	1 May 2023	Revision I	1.3
Ting Fung Ha	2 May 2023	Revision II	1.3.1

1. Introduction

1.1 Purpose

The document will provide the users with a general explanation of the COVID-19 data visualization application and Machine learning analysis of the relationship between COVID-19 and blood type. Each part will include the components involved, its functionalities, and how the

components work together. The document will also cover some of the design strategies, concerns, challenges and various restrictions before using each portion of the project.

1.2 Document Conventions

The document is written in a minimalistic and a straightforward style. Therefore, no typical conventions were followed.

1.3 Intended Audience and Reading Suggestions

Analysts: It is recommended for analysts to go through Section 2 to 3 and 5 to 7. These sections will introduce some of the origin of the datasets, along with tools and technologies applied to this project.

Data Scientists: Data scientists should also read Section 2 to 7 of this document.

Medical Professionals: Medical professionals can refer to Section 2 and 5 for more relevant information.

For anyone with an interest in the project, it is recommended to read through the entire document to get a full understanding of this project.

1.4 System Overview

Part I:

The application will create visualizations based on the data obtained from trusted organizations. After that, users will have the option to focus on one of the visualizations in depth and see its relationship with other categories from the same dataset..

Part II:

The second part of the project involves using machine learning, and a feature selection algorithm along with statistical tests to analyze the associations between blood type and COVID-19 mortality and severity along with identifying relevant comorbidities.

2. Design Considerations

2.1 Assumptions and Dependencies

Part I:

The first portion of this section mainly covered the experiences our team had with Tableau products (Tableau Public, Desktop and Portal).

The second portion of the project is an application that is capable of turning real-time data into graphical visualizations. The user can see specific aspects of the data they want to see based on a variety of predefined filters such as date, color, state, etc. To achieve this, a number of JavaScript libraries were used. Please refer to Section 5, subsection 1 for a list of tools and technologies used.

All products in this part can be installed on computers with the following specifications:

- Windows OS: 8 / 8.1 / 10(x64)
- Macintosh OS: Mojave 10.14 / Catalina 10.15 / Big Sur 11.4+ / Monterey 12.6+ (for Tableau 2022.3+)

Part II:

This part of the project will depend heavily on Python 3+ and some of its libraries. As a result, the user is assumed to have a basic understanding of running Python scripts through Jupyter notebook.

Some of these Python libraries include, but not limited to the following:

- Matplotlib
- Numpy
- Python Data Analysis Library (PANDAS)
- Sklearn

2.2 General Constraints

· Given the majority of the project is based on software, there will be very little hardware limitations. However, the device should have enough computing power to view online contents

with short delays. In order to access the Javascript application and Tableau portal, users must have a stable internet connection to access all of its functions.

2.3 Goals and Guidelines

One of the main goals of the project is to understand the COVID-19 virus based on a selection of datasets with a variety of attributes. The team will take these datasets and analyze them, hoping to gain some insights on this new kind of virus. These insights will aid us in the second part of the project, where a machine learning algorithm will determine if someone is more susceptible to COVID-19 based on genetic factors and comorbidities. In order to achieve this, the team was aided with valuable datasets from our sponsor, Vodafone. These datasets contain confidential medical information from actual patients, which means each member from the team must maintain its integrity and confidentiality.

2.4 Development Methods

The project overall consists of two development methods, Unplanned mad Scramble Development, and the process of trial and error. Unplanned mad scramble was mostly used during part one of the project, where the filters and customizations were distributed between some of the group members.

Our team adopted the trial and error method during part one of the visualization creation stage, where we had to select the categories with the highest relevant and significant values. This same method was reused during the second part of the project, notably using machine learning to highlight the most important features that will determine if a person is more susceptible to COVID-19 or not..

3. Architectural Strategies

3.1 Product Usage

3.1.1 Part I – Visualizations

3.1.1.1 – Tableau Portal

3.1.1.1.1 – Tableau Public: Tableau Public was used because of its functionality on rendering different types of visualizations with a dataset. The abundant customization options enlightened the team with various new insights, which aided us in the decision making and content creation stage.

3.1.1.1.2 – Tableau Desktop: The team decided to use Tableau Desktop because it is an upgrade of Tableau Public with additional enhanced abilities. A number of the datasets were so large in file size where Tableau Public cannot handle it. As a result, Tableau Desktop was chosen as a replacement for Tableau Public.

3.1.1.2 – An Application to Create Visualizations from Datasets

3.1.1.2.1 – D3.js: D3.js allowed the team to turn datasets of csv files into interactive visualizations. Also, the D3 library was preferred over other libraries due to its documentation and ease of practical examples.

3.1.1.2.2 – GitHub: GitHub was used to group all the files together in an organized and centralized form. Allowed us to utilize various functions Github offers to split the workload individually over all our team members.

3.1.1.2.3 – JavaScript and its libraries, React JS and Material UI: The application was created with JavaScript with one framework and a front end library for styling. ReactJS framework and Material UI library for its various components. Overall, our front end visuals are delivered using both Library and framework.

3.1.1.2.4 – Jupyter Notebook: This product is part of Project Jupyter, and it is used for interactive computing across multiple programming languages.

3.1.1.2.5 – Node.js: It is used for server-side programming, and primarily deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but was originally designed with real-time, push-based architectures in mind.

3.1.2 Part II – Clinical Analysis with Machine Learning

A feature selection algorithm known as Boruta will be incorporated with machine learning and try to determine some comorbidities to have an effect on the severity and mortality of COVID-19. The results generated from the algorithm will be checked with a statistical test, known as Chi-Square for validation.

3.2 Reuse of Existing Software Components

A large portion of our codes were reused within multiple sections during part one of our project. Multiple design templates from the two JavaScript libraries, React and Material UI were reused throughout the application pages and customization options panel. Next, D3.js was widely reused for creating graphical representations. In the end, Node.js and Jupyter Notebook were used constantly to get the most up to date data, and some of the user interaction functions.

3.3 Future Plans for Software Enhancement

Some of the plans listed on our agenda may include, but limit to the following:

- To have more visualizations based on other trustworthy and updated sources
- More customization options for visualizations
- Include verification/validation component to check the datasets
- A component to handle overflow errors

3.4 User Interface Paradigms

3.4.1 None applicable at this time.

3.5 Hardware and/or Software Interface Paradigms

3.5.1 Users on mobile devices must have a touchscreen display for user to data interactions.

3.5.2 Personal computer must have a pointer device for user interactions and selections.

3.5.3 A stable internet connection is required in order to access all of the contents.

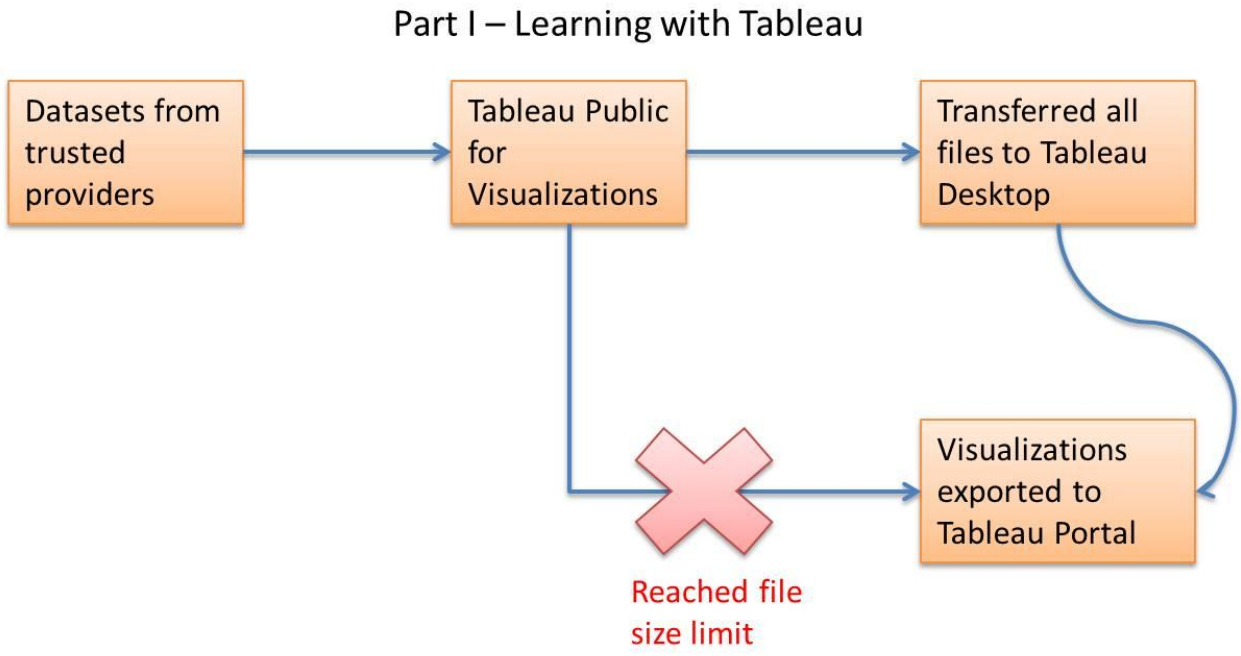
3.6 Error Detection and Recovery Options

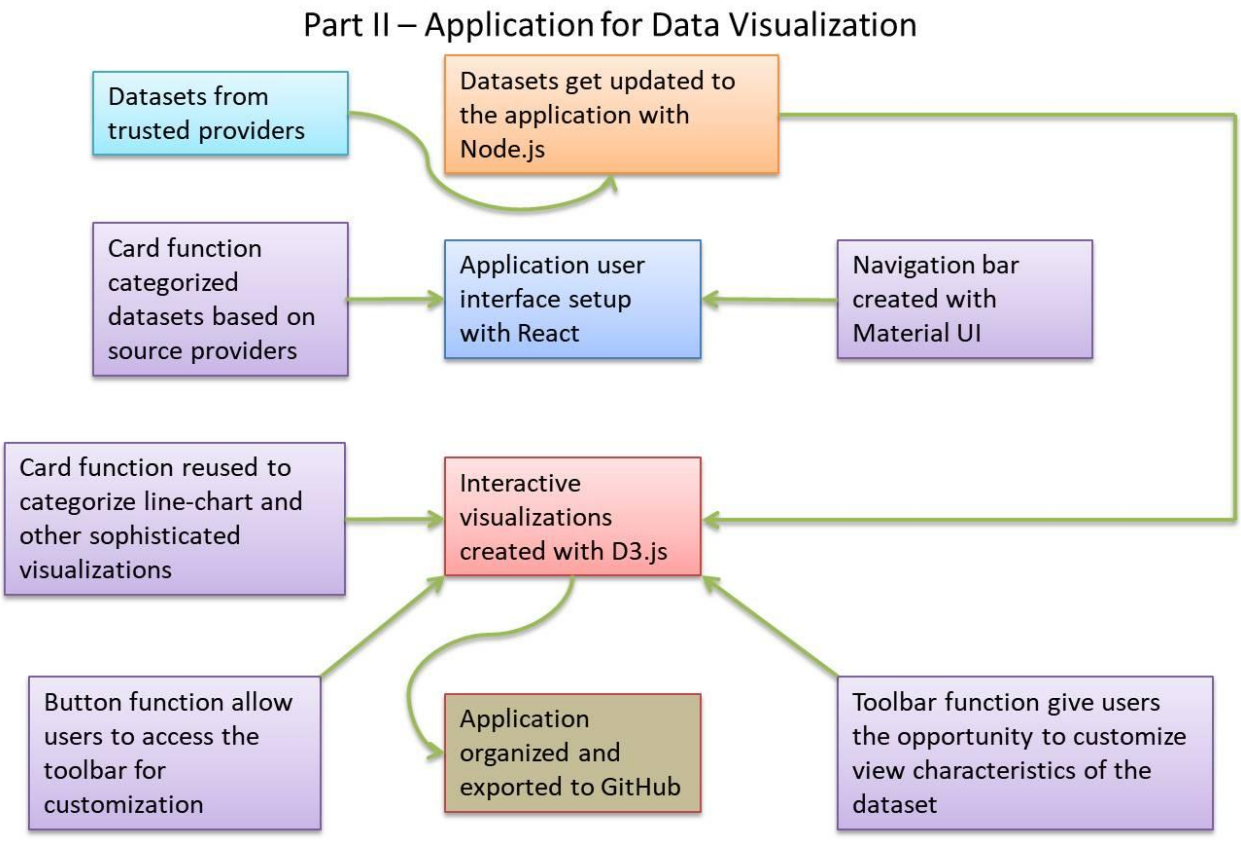
Error detection for dataset is provided by React.js

3.7 Memory Management Policies

Memory management should be handled by the operating system and the browser installed on each device.

4. System Architecture





5. Policies and Tactics

5.1 Choice of which specific products used

5.1.1 Part I: Tableau Products and an Application Creating Visualizations based on Real-time Data

5.1.1.1. D3.js: Create visualizations in the application

5.1.1.2. GitHub: File organization

5.1.1.3. JavaScript and React: User interface

5.1.1.4. Jupyter Notebook: Data visualization between multiple languages

5.1.1.5. Material UI: User functions and design

5.1.1.6. Node.js: Fetch data automaticallyTableau Public: Data analysis

5.1.1.7. Tableau Desktop: Data analysis and export

5.1.1.8. Tableau Public: Data analysis

5.1.2 Part II: Machine Learning with Boruta Algorithm

5.1.2.1. Jupyter Notebook: All of the Python codes and the libraries were done on Jupyter Notebook.

5.1.2.2. NumPy: This library was used to group the data into data structures for other operations

5.1.2.3. Python 3+: Backbone of the project, required to use the Python libraries

5.1.2.4. Python Analysis Data Library (PANDAS): PANDAS was used for data analysis and manipulations

5.1.2.5. SciKit-Learn (SKLEARN): A simple yet efficient tool for predictive data analysis. It is built on top of NumPy, SciPy and matplotlib.

5.2 Plans for ensuring requirements traceability

5.2.1 Most of the products are maintained by third-party organizations.

5.3 Plans for testing the software

5.3.1 – Part I: Visualizations

The first step was to test whether the dataset from the trusted site can be read by the function which creates the visualization. Once successful, the window will be switched to the other visualizations to make sure the graphs are displayed correctly.

The second step will test the filter functions, where certain attributes from the dataset can be changed based on the user. If successful, the visualizations will be modified and updated.

The third step will test the Your Dashboard functions, where visualizations can be added to your dashboard and when clicking on your dashboard the visualization appears there and loads the visual.

5.3.2 – Part II: Clinical Analysis with Machine Learning

The dataset obtained from our sponsor, Vodafone, will be applied here. The group will feed the data into a trained machine learning model to determine if there are any relations between the severity and mortality of COVID-19 and patient blood types. All the results will be checked by a Chi-Square statistical test.

5.4 Coding guidelines and conventions

5.4.1. The application was broken down into individual components, then assigned to group members to make it functional. In the end, all components were put together to have our application completed.

5.4.2. All the related files were loaded into a .JSON file for ease of access and organizing.

6. Detailed System Design

6.1 Part I – Tableau Products and Visualizations

6.1.1. User Interface

6.1.1.1. Responsibilities

The layout of the application is written with React, a JavaScript library specifically meant for user interfaces. This component sets the framework and defines the general layout of the application.

6.1.1.2. Constraints

Because React is a JavaScript library, the browser will need to have JavaScript enabled in order for this component to function properly. Secondly, a stable internet connection is required in order to access the library.

6.1.1.3. Composition

Each customization option will change the view of the current visualization to the browser they are viewing it.

6.1.1.4. Uses/Interactions

The user interface is the first component users will see during their access. It will remain the same even if the user decides to use the other functions of the applications, or to view the different visualizations. Therefore, the UI will interact with all the components listed in this document.

6.1.1.5. Resources

To load the interface, the computer will utilize the CPU, memory and the React library itself. The allocation of these resources will be handled by the operating system located on the user's device.

6.1.6 Interface/Exports

6.1.2. Card Function

6.1.2.1. Responsibilities

This function organizes the datasets based on their source provider. It will then display it to the users. Similar function is used for displaying the line-chart and the relevant visualizations

6.1.2.2. Constraints

The function will require the browser to have JavaScript enabled, along with a stable internet connection

6.1.2.3. Composition

Each customization option will change the view of the current visualization to the browser they are viewing it.

6.1.2.4. Uses / Interactions

User interaction with the card function will lead to the graphical visualization function.

6.1.2.5. Resources

All resources allocation will be handled by the computer, the browser from the device, and the operating system.

6.1.3.6. Interfaces / Exports

This function will accept the dataset from the source provider, and generate a line-chart.

6.3.1. Navigation Bar

6.3.1.1. Responsibilities

Display as part of the user interface, which act as a shortcut for user to access some of the important pages.

6.3.1.2. Constraints

The function will require the browser to have JavaScript enabled, along with a stable internet connection

6.3.1.3. Compositions

Each navigation bar item is part of the navigation bar function.

6.3.1.4. Uses / Interactions

Interactions made here will redirect the user to the pages compiled with Card function.

6.3.1.5. Resources

All resources allocation will be handled by the computer, the browser from the device, and the operating system.

6.3.1.6. Interface / Exports

Any type of user interaction will be considered as a form of input. The output would be leading the user to the selected pages.

6.1.4 Toolbar

6.1.4.1. Responsibilities

This function will allow the users to customize some of the attributes from the dataset.

6.1.4.2. Constraints

The function will require the browser to have JavaScript enabled, along with a stable internet connection.

6.1.4.3. Composition

Each customization option will change the view of the current visualization to the browser they are viewing it.

6.1.4.4. Uses/Interactions

Toolbar will be an individual class, while each customization will be a subclass of the toolbar function.

6.1.4.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.4.6. Interface/Exports

This component will take users input on the options the toolbar gives.

6.1.5 Buttons

6.1.5.1. Responsibilities

Interaction with this design will lead to a customizable visualization panel, this is also used to confirm the user selection.

6.1.5.2. Constraints

The function will require the browser to have JavaScript enabled, along with a stable internet connection.

6.1.5.3. Composition

Each customization option will change the view of the current visualization that the user is viewing from so if it is a phone the browser will adapt to the screen.

6.1.5.4. Uses/Interactions

This component will interact with the graphical visualization component, which will update the user customization option to the current graph. If they would like to change the colors of a graph or the dates the graph is displaying they will be able to do that with the filters button.

6.1.5.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.5.6. Interface/Exports

This component will take user input and pass it to the graphical visualization component and apply the given changes.

6.1.6 Graphical Visualizations

6.1.6.1. Responsibilities

This component will create visualizations based on the given dataset and the graphs will be interactive.

6.1.6.2. Constraints

The functions will require the browser to have JavaScript enabled, along with a stable internet connection to view the application.

6.1.6.3. Composition

Each customization option will change the view of the current visualization depending on the browser the user is viewing from.

6.1.6.4. Uses/Interactions

Once the graph is created, it will be placed into a page created by the Card component. The card will also have some of the functionalities created by Buttons and Toolbar.

6.1.6.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.6.6. Interface/Exports

This component will reflect the data values based on the trace of the user's pointing device.

6.1.7 Data Fetching

6.1.7.1. Responsibilities

This component will fetch the latest data from the source provider needed to display visualizations.

6.1.7.2. Constraints

The function will require the browser to have JavaScript enabled, along with a stable internet connection for the application to run.

6.1.7.3. Composition

Each customization option will change the view of the current visualization. If the viewer is looking at the application from a phone browser the application will be formatted to the phone browser.

6.1.7.4. Uses/Interactions

The dataset obtained with this component will be passed directly to the graphical visualization component.

6.1.7.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.7.6. Interface/Exports

This component will take the dataset from source providers as input, and display itself on the webpage.

6.1.8 Dashboard

6.1.8.1. Responsibilities

This component will store a visualization selected by the user into another page for side-to-side comparison, or use them at another time.

6.1.8.2. Constraints

The functions will require the browser to have JavaScript enabled, along with a stable internet connection to view the application.

6.1.8.3. Composition

Visualizations saved onto the dashboard can be viewed through the toolbar

6.1.8.4. Uses/Interactions

This component can be accessed with the toolbar located on top of every page. The “saved to dashboard” option is housed by the Card function.

6.1.8.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.8.6. Interface/Exports

This component will take the visualization and export it to another page for other usages.

6.1.9 Data Fetching

6.1.9.1. Responsibilities

This component will fetch the latest data from the source provider needed to display visualizations.

6.1.9.2. Constraints

The function will require the browser to have JavaScript enabled, along with a stable internet connection for the application to run.

6.1.9.3. Composition

Each customization option will change the view of the current visualization. If the viewer is looking at the application from a phone browser the application will be formatted to the phone browser.

6.1.9.4. Uses/Interactions

The dataset obtained with this component will be passed directly to the graphical visualization component.

6.1.9.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.9.6. Interface/Exports

This component will take the dataset from source providers as input, and display itself on the webpage.

6.1.10 Filters

6.1.10.1. Responsibilities

This component will provide some of the pre-defined customization options for the user to apply to the dataset.

6.1.10.2. Constraints

The functions will require the browser to have JavaScript enabled, along with a stable internet connection to view the application.

6.1.10.3. Composition

Each customization option will change the view of the current visualization depending on user's selection

6.1.10.4. Uses/Interactions

Any changes to the filter will change the appearance of the visualization

6.1.10.5. Resources

All resources allocation will be handled by the device operating system and browser.

6.1.10.6. Interface/Exports

Any filters selected by the user will get updated to the original visualization.

6.2 Clinical Analysis:

6.2.1. Boruta Algorithm (identified features)

The algorithm will take the original dataset and create a completely new dataset with double the amount of columns. Next, the algorithm will take the newly created dataset and train a random forest model. The COVID-19 features will be inputted into this machine to see if the importance level is greater than the importance level in the newly created dataset. After a number of iterations, the machine will output a selection of important features for analysis.

6.2.2. Stats test (results)

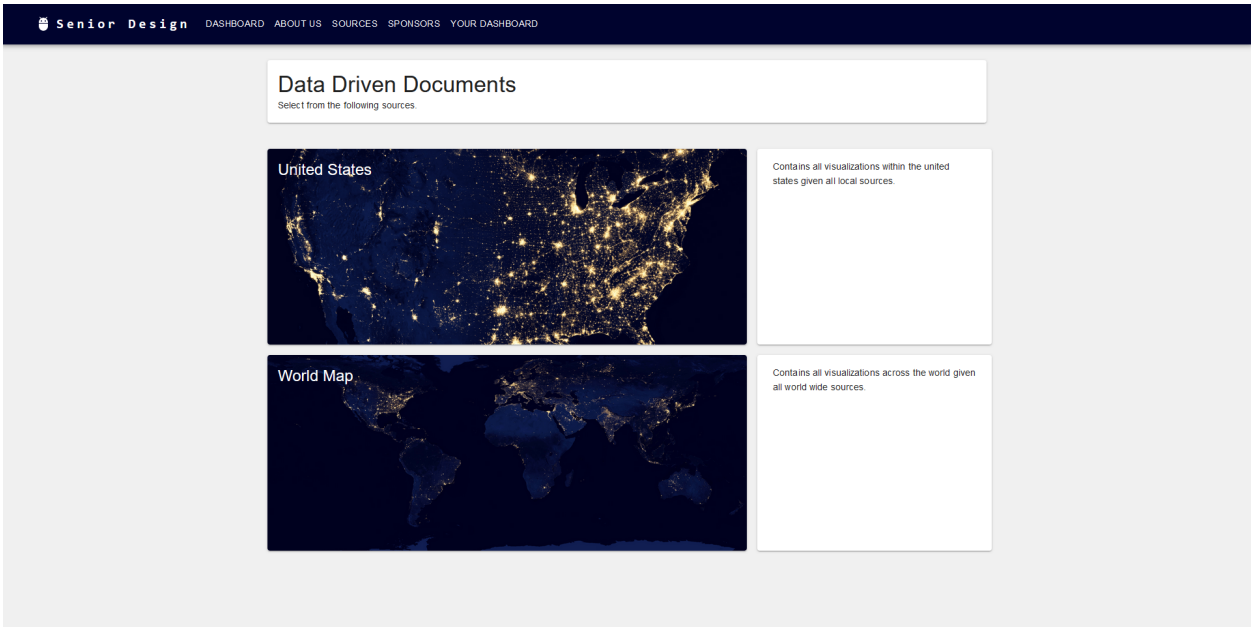
The core features identified by Boruta algorithm will be compared with a number of tests, Chi-Square and Fisher's Exact test. The first test will test the independence between categorical variables, while the second test returns a value if there is a significant association between the two features.

6.2.3. Severities (Mortality & Severity)

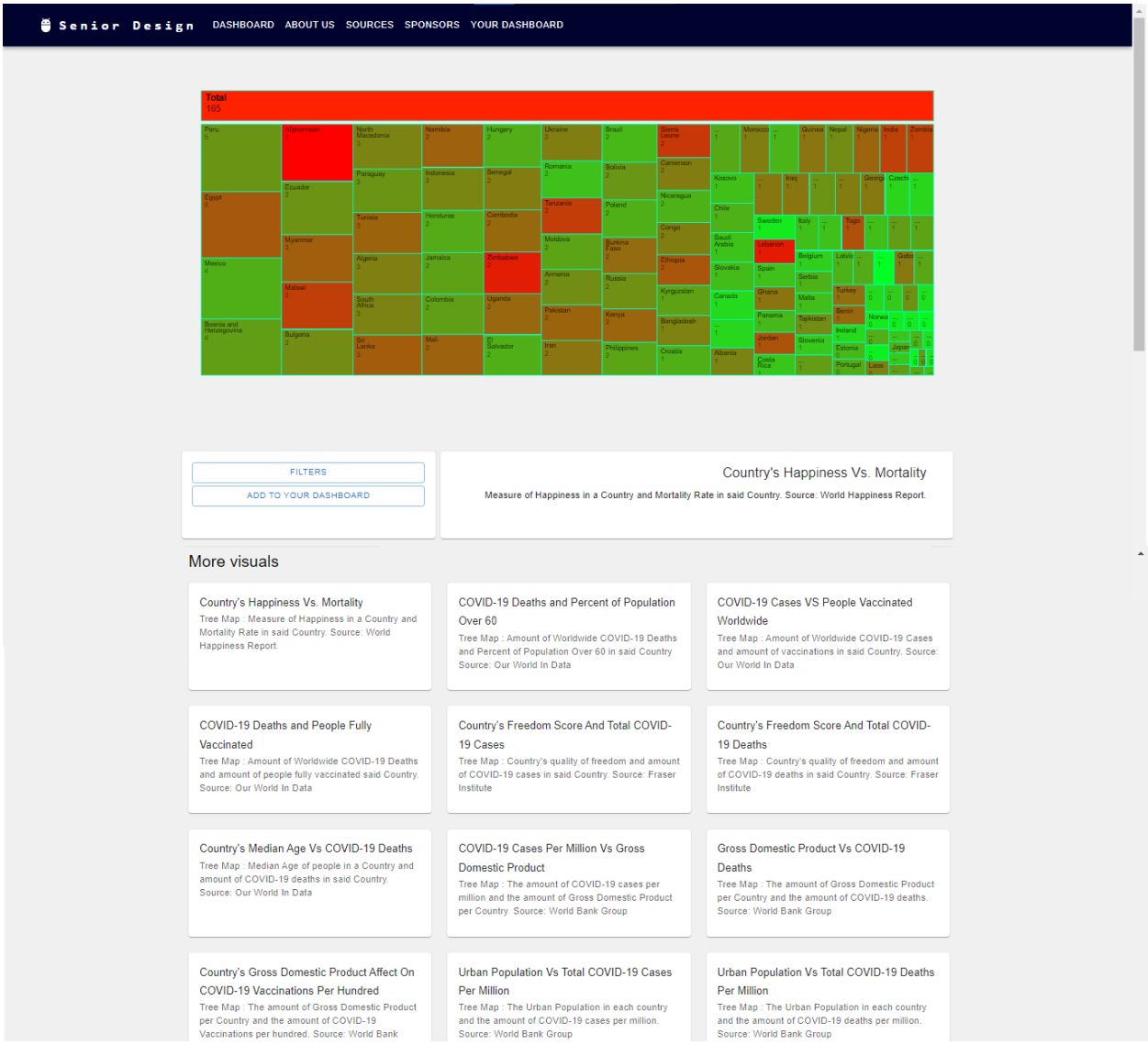
Our team entered the blood data from the patients, each with one vary feature into a trained machine for analysis. The machine learning algorithm includes the following:

- AdaBoost
- Logistic Regression
- Random Forest

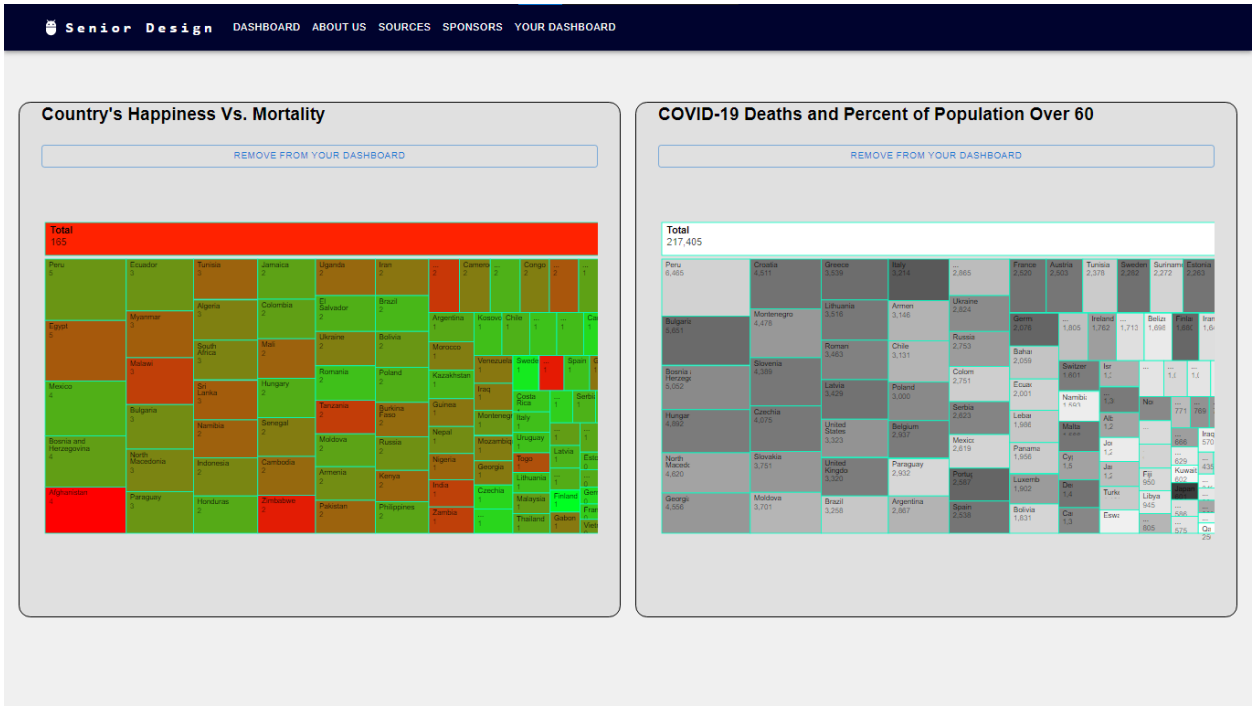
7. User Interface



This is the home page of our application, which contains the toolbar, and two datasets categorized into domestic and international.



Users will be directed to this page after selecting “World Map”, where the dataset is based on international statistics.



This is the dashboard, where user can save a visualization for future usage.

8. Glossary

React: A JavaScript library that is meant for designing user interfaces.

Material UI: A React component based on Material Design by Google in 2014.

Tableau Desktop: An application meant for organizing, studying and visualizing a dataset.

GitHub: An Internet hosting service for software development and control with the use of Git.

Node.js: A JavaScript runtime environment built on Google Chrome's V8 JavaScript engine.

Project Jupyter: A non-profit, open-sourced project born out of iPython Project in 2014. It is used to support interactive data science and scientific computing across all programming languages.

D3.js: A JavaScript library for manipulating documents based on the data.

9. References

12.1 Data-Driven Documents (D3) Overview

https://d3js.org/

12.2 Project Jupyter's Origins and Governance

https://jupyter.org/about

12.3 Nodejs

https://nodejs.org/

12.4 GitHub

https://github.com/

12.5 Material UI

https://v4.mui.com/

12.6 Title: Software Design Document_Draft

Authors: Antonio Campos, Amy Guttman, Alec Kaczmarek, Vincent Li, Saiyang Liu, Ricardo Marroquin, Miguel Nonoal-Garcia, Alexandra Strong, Jonathan Sum, Edwin Zapata Minero

Date: November 27, 2022

12.7 Title: A Mobile Assistive Technology for Peripheral Visual Field Loss

Authors: Daniel Esparza, Canhong Huang, Jonathan Kan, Abran Lezama Pastor, Brenden Mccabe, Ashley Munoz, Uchenna Onuigbo, Duy Pham, Nicolas Sandoval, Jacob Schultz

Date: November 27, 2022