

**BACHELOR**

**PyWash 2**

**An Accessible and Automated Data Cleaning Assistant**

de Vos, Dave

*Award date:*  
2023

*Awarding institution:*  
Tilburg University  
Jheronimus Academy of Data Science

[Link to publication](#)

**Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

# Final Bachelor Project

PyWash 2: An Accessible and Automated Data Cleaning Assistant

*Dave de Vos*

*0910744*

d.d.Vos@student.tue.nl

Technische Universiteit Eindhoven,  
Department of Mathematics and Computer Science  
Tilburg University, School of Law  
Jheronimus Academy of Data Science

Supervisor:

Dr. ir. Joaquin Vanschoren



# Table of Contents

<b>Chapter 1: Introduction</b>	4
1.1 Problem Statement	4
1.2 Scope	4
1.4 Outline	4
<b>Chapter 2: Pywash - Functionality Update</b>	5
2.1 the SharedDataFrame	5
2.2 Importing & Parsing	5
2.3 Data Types & Missing Values	5
2.4 Outlier Detection and Standardization/ Normalization	5
2.5 Dataset Export	6
<b>Chapter 3: Pywash – Interface</b>	7
3.1 General Design Strategy	7
3.2 Uploading the Data	7
3.3 Cleaning the Data	8
3.3.1 Column Types and Anomaly Detection	8
3.3.2 Outlier Handling, Missing Value Detection & Duplicate Row Handling	8
3.3.3 Standardization & Normalization	9
3.3.4 Data Preview Table	9
3.4 Visualizing the data	10
3.4.1 Data Export	10
3.4.2 Data Summary Table	10
3.4.3 Visualization	10
3.4.4 Full Dataset Table	11
3.4 Technical Implementation	11
3.5.1 Components	11
3.5.2 Callbacks	11
3.5.3 Storage	12
3.5.4 Applied Dash Strategies	12
3.5.5 Using the Pywash application	13
<b>Chapter 4: Conclusion</b>	15
4.1 Future Work	15
<b>Bibliography</b>	16
<b>Appendix A: Callback Graph</b>	17



# Chapter 1: Introduction

This report contains a showcase of the functionality of the new PyWash interface, as well as the decision-making process behind it. This Final Bachelor Project is not just a continuation of the original PyWash app of Yuri Maas and Laurens Castelijns, but also a collaboration between myself, Dave de Vos, and the other people of my Bachelor Project Circle for this project: Joshua Kalisvaart, Jonas Niederle, and Thomas Quadt. This document can be read without having to cross-reference their work, but it may sometimes be referenced.

## 1.1 Problem Statement

Last year, the initial version of PyWash was created by Yuri Maas and Laurens Castelijns. They added a lot of functionality for cleaning a wide variety of datasets and even some methods to visualize them, but the design of the UI was very much secondary in their huge efforts to create a working data cleaning assistant from scratch. As a result, the UI leaves some things to be desired: Between constantly switching tabs and manually choosing how to apply the data cleaning, it requires a lot of effort to actually clean the imported datasets. Furthermore, whilst the visualizations that were added were well-implemented, the existing visualizations are insufficient to get a good first impression of what the data looks like on a wide range of datasets. Therefore, the project this year was centered around two main questions:

- How can the data cleaning features of the existing PyWash application be further improved on and automated?
- How can the usability of the PyWash program be further improved on?

## 1.2 Scope

Whilst the others all decided on improving various data cleaning features of the existing PyWash implementation, I decided to work on improving the usability of the PyWash program. What this means in practice, is that I focused on three main things:

- Building a new user interface focused on automation and user-friendliness.
- Creating a selection of visualizations, easily accessible yet also comprehensive of a wide variety of data, to check the cleaned dataset.
- Implementing the newly made features into the general PyWash application.

## 1.4 Outline

Firstly, in Chapter 2 a quick refresher of the old PyWash functionality is given, together with some of the changes that have been made by my circle members. Secondly, in Chapter 3 the work on the PyWash Interface will be described in four different parts: A general outline and the underlying main design decisions, the 'Data Cleaning' window, the 'Visualization' window, and aspects of the technical implementation. Lastly, in Chapter 4, the project is summarized and some possible future work is presented.

# Chapter 2: PyWash - Functionality Update

The PyWash application we are currently developing did not come out of nowhere; Last year, the original version of PyWash was developed by Yuri Maas and Laurens Castelijns. Although the original version of PyWash was already functional, several parts of it have been improved by the members of this year's circle. Of particular note are new methods for missing value detection (Jonas Niederle), column type prediction (Thomas Quadt), and outlier detection (Thomas Quadt).

This chapter will quickly explain the various functions the new UI is built upon as of this year. Keep in mind that most of this work detailed in this chapter is done by either the PyWash team of last year, or my other circle members of this year, and is only included for completeness' sake.

## 2.1 the SharedDataFrame

As the nervous center of the PyWash application, an object of the SharedDataFrame class stores an imported dataset and its properties, and is able to call any and all functions that are supposed to be used during the cleaning process of PyWash.

## 2.2 Importing & Parsing

PyWash has functionality for importing two types of data: CSV and ARFF. CSV was chosen for its popularity, and ARFF for its use in OpenML, a website with a comprehensive list of tens of thousands of freely accessible datasets, together with information about and tasks executed with said datasets (OpenML, n.d.). It is able to load a single dataset in the form of a file of the previously mentioned types on your computer, using your Operating System's standard file selection procedure.

It is then decoded with cp-1252 as a default, but PyWash will try to find the encoding with the "chardet" Python package (Chardet - PyPI, 2017) if the default method fails. Afterwards, it tries to correctly load the dataset using "CleverCSV", a Python Package for analyzing CSV files (CleverCSV - PyPI, 2020), or uses a set method used for ARFF files. Lastly, once the dataset has been correctly parsed, it is put into a Pandas DataFrame (pandas - Python Data Analysis Library, 2020) as the data is imported into the SharedDataFrame object.

## 2.3 Data Types & Missing Values

Using a custom method named Ptype, Thomas Quadt created a new way to detect column types and store possible 'anomalies', values that did not get detected as the same datatype as the rest of their column.

Missing values are detected and handled using code by Jonas Niederle, using a whole host of methods from the scikit-learn package (scikit-learn: Machine Learning in Python, 2020).

## 2.4 Outlier Detection and Standardization/ Normalization

In the case of outlier detection, there are now two methods of detecting outliers: Last year's method, which used a host of outlier detection algorithms, selected three algorithms to use as a 'base' to detect outliers: Isolation Forest, Principal Component Analysis (PCA), and K Nearest Neighbours (KNN). This decision was based on their own results and earlier work of Ji Zhang (Zhang, December 2018). To add

onto this Thomas Quadt has added a slower, but more thorough ensemble method based on the work of Sylvie Ratté (Ratté, 2016) as an alternative.

Basic methods of standardization and normalization for individual columns were also implemented last year, with normalization taking range of [0,1] and standardization having a mean of 0 and unit variance.

## 2.5 Dataset Export

Of course, after all the cleaning actions have been applied, the user can export the dataset back to their computer. This feature also has support for CSV and ARFF, although CSV files cannot be exported as ARFF files due to the extra data required for ARFF files.

# Chapter 3: PyWash – Interface

Of course, the main focus in this report is the part I mainly worked on, which is the implementation of the new user interface and maintaining the PyWash SharedDataFrame. This chapter will cover everything concerning my work on the interface, from my general strategy to individual parts of the interface, as well as giving some information about the technical implementation of the interface itself.

## 3.1 General Design Strategy

The main issue with the UI design of the previous year, as discussed during our initial meeting, was that in order to successfully clean their dataset, the user needed to do a lot of busywork: Switching between tabs, looking through dozens of options and choosing from a comprehensive list for many of them is a good idea in theory, since it gives the user complete control over their way of data cleaning. However, in practice, the user gets lost between all those options, some of which might not even be known to them. An example is the Outlier Detection: The user might not know how the ten algorithms specified in the Outlier Detection method work, either on their own or when combined with other algorithms.

Therefore, I focused on two main objectives when creating the new user interface:

- Separate the PyWash process into three distinct steps: Uploading the data, cleaning the data, and visualizing/Exporting the data
- Automate data cleaning functionalities wherever possible.

The interface itself was created using the Dash python package (Dash Overview - Plotly, 2020) and its subsidiary packages. Dash is a Python framework built on top of Flask, Plotly, React, and React Js.

## 3.2 Uploading the Data

The uploading part of the user interface, as seen in Figure 1, has deliberately been kept as simple as possible, as to not overwhelm new users with a plethora of buttons and information that they would not have been able to use until the data has been uploaded anyway. Once the user has uploaded a file either by dragging a file onto the designated space or clicking on it and selecting their dataset using their Operating System’s standard file selection procedure, the program waits until the dataset is fully loaded in before switching the user to the ‘Data Cleaning’ part of the user interface.

### Pywash browser interface

! Drag & drop csv or click here to get started !

Keep in mind that processing can take a while depending on the imported dataset

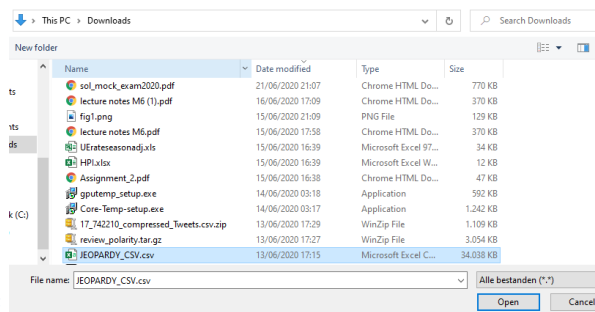


Figure 1. Importing the data. Left: Initial PyWash screen. Right: Importing data with the default Windows ‘Open’ feature.



### 3.3 Cleaning the Data

The data cleaning part of the user interface, as can be seen in Figure 2, has been divided into compartments using blank space.

#### 3.3.1 Column Types and Anomaly Detection

The first compartment is about column types and anomaly detection. These are the highest in priority, as the other cleaning methods depend on these being correct. By clicking on the primary column type list, it displays all columns in the data, together with their type. The user can then use the secondary list to change the data type. For anomalies, the user can click on the primary anomaly list to check if there are any columns with anomalies, and if so, select one or more of them using the secondary anomaly list. Then, the user can either dismiss them as non-anomalies, or confirm them as actual anomalies, removing them from the dataset. This is also done automatically with any leftover anomalies once cleaning starts.

#### 3.3.2 Outlier Handling, Missing Value Detection & Duplicate Row Handling

The second compartment is about handling the outliers, choosing a column for missing value detection (if desired), and testing for duplicate rows. These are the other primary data cleaning methods, which are second highest in priority: Still required to successfully clean a dataset, but about the contents of the dataset instead of possible structural/parsing errors. For the outlier detection, the application sets the outlier detection method by default, but the user can change the method if necessary (see Chapter 2.4), including not doing outlier detection at all. The way that missing value detection currently works unfortunately means it cannot currently be automated, but if the user knows or suspects that there could be missing values in a column, they can set it to be cleaned. Lastly, there is a simple Yes/No toggle for detecting and removing duplicate rows.

**Pywash Browser Interface**

---

Check or change column types

Inspect found anomalies per column, change column type or delete columns that are not anomalies

Choose preferred method for handling outliers

Select column to clean missing values, only for regression or classification target variable

Test for duplicated rows?

 No  Yes

Normalize column(s)?

Standardize column(s)?

**Data Preview**

Married	Age	Years_of_education	Male	Religious	Sex_partners	Income	Drug_use	Same_sex_relations	AIDS_know
1	41	19	1	0	1	27500	0	0	0
1	29	14	0	1	1	27500	0	0	0
0	45	12	1	1	1	16250	0	0	0
1	35	12	1	1	1	55000	0	0	0
1	47	12	0	1	1	500	0	0	0

Figure 2. Cleaning the `analcata_data_gsssexsurvey.csv` Dataset in PyWash (`analcata_data_gsssexsurvey`, 2014).

### 3.3.3 Standardization & Normalization

The third compartment is about standardizing or normalizing certain columns: These operations are not strictly necessary for a cleaned dataset, but might be helpful depending on the user's intention. Therefore, they can choose any number of columns to be normalized or standardized.

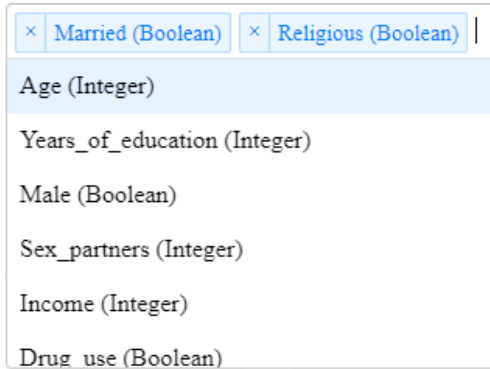


Figure 3. Selecting multiple columns for standardization.

### 3.3.4 Data Preview Table

Below these compartments, all the way at the bottom is a small preview of the actual data. While the DataTable of the previous year was an interesting centerpiece of their user interface, it took up too much space and distracted too much from the actual cleaning functions. Therefore, it has been replaced by a small 5-row data preview. This table also has the properties of being able to delete and rename columns if necessary through icons to the left of the column name, and these changes will be applied to the dataset before the main data cleaning processes start.

Above it is a button to start processing the data and to move onto the final phase of the application, visualization and data export.

#### Data Preview











  Married	  Age	  Years_of_educatio	  Male	  Religious
1	41	19	1	0
1	29	14	0	1
0	45	12	1	1
1	35	12	1	1
1	47	12	0	1

Figure 4. The Preview DataTable, with some columns from *analcata\_data\_gsssexsurvey.csv* (*analcata\_data\_gsssexsurvey*, 2014).

### 3.4 Visualizing the data

The visualization part of the user interface has also been cleaned up and has some new features, as can be seen in Figure 5 and 6.

#### Pywash Browser Interface

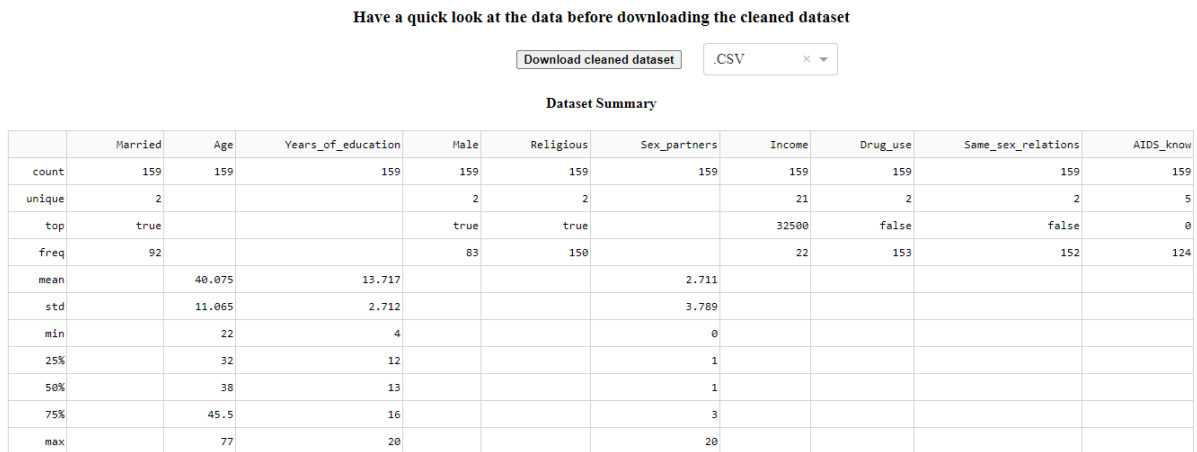


Figure 5. Exporting the data & Data Summary

#### 3.4.1 Data Export

Squarely centered, at the top of the page, is the export button to download the cleaned dataset. The main function of the application is to import, clean, and export a dataset, and as such the download button should be the first thing the user sees when entering this stage. Next to it is an option to switch the type of file to download, which is automatically set to the type of file the user imported.

#### 3.4.2 Data Summary Table

Below this is a basic summary of the data. The relevant parts of this change based on the type of the column, so that it shows information like the amount of unique values and frequencies for categorical variables, and information like mean, standard deviation, minimum and maximum for numerical values. This helps the user to get a quick overview of the dataset as a whole, and to see if there have been any major mistakes during the cleaning process.

#### 3.4.3 Visualization

If the user still wants to know more about the dataset before downloading it, next up is the actual visualization: A list from which the user can select columns, based on which the program will show an appropriate visualization. These visualizations are from a selection of one-column, two-column and multi-column visualizations in order to visualize a broad range of datasets. For one-column visualizations, numerical data is displayed as a histogram, whilst categorical data is displayed as a bar chart. For two-column visualization, both purely numerical data and mixed (categorical/numerical) data is displayed as a scatterplot, but for mixed data jitter is included, stylized as a violin plot in order to

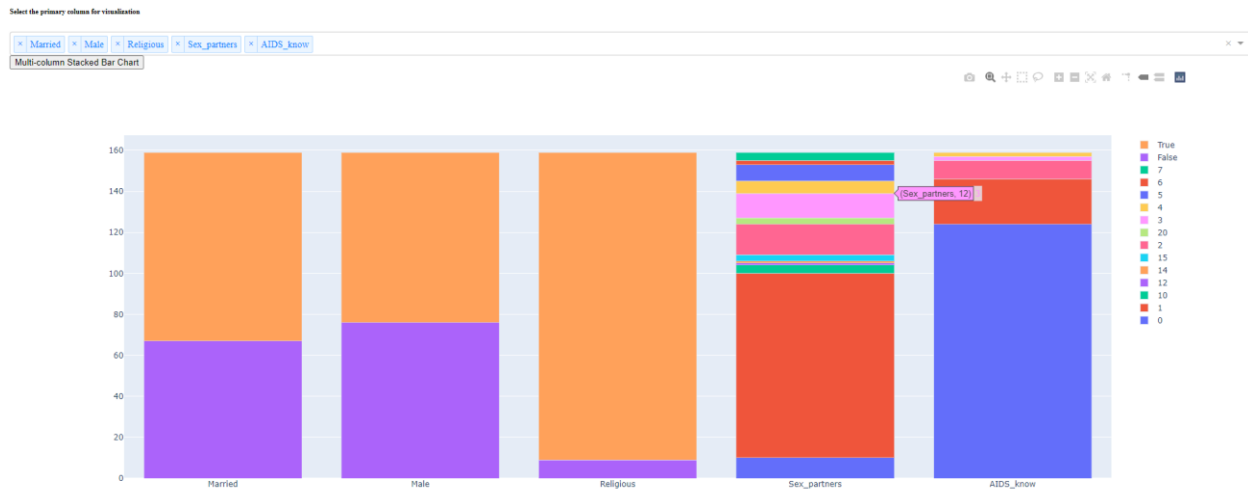


Figure 6. Multi-column Stacked Bar Chart of *analcatdata\_gsssexsurvey.csv* (*analcatdata\_gsssexsurvey*, 2014). The *sex\_partners* column was manually changed to Categorical earlier on using the *Column Type* functionality.

visualize the amount of datapoints per category better. For multi-column data, any number of categorical columns can be displayed as a stacked bar chart, where each ‘bar’ represents a column.

### 3.4.4 Full Dataset Table

At the bottom of the page is the full dataset in the form of a table, if the user wants to manually inspect individual values. The individual columns of the table are sortable, and every odd-numbered row is shaded slightly darker to aid the user in visual inspection.

## 3.4 Technical Implementation

As mentioned before, the entire interface has been put together using the Dash Python package. Below, the functionality of the main tools used in Dash, as well as some of the strategies employed while creating the user interface are outlined.

### 3.5.1 Components

The actual layout of the app, which consist of all the elements the user can and cannot interact with and their location, is created by creating a diverse set of components and layering these components next to and on top of each other. These components are already user-interactive in that the user can open a dropdown list and select an item, but are also static in that this will not have an effect on any other component in the interface.

### 3.5.2 Callbacks

In order to make components respond to each other and relay information to any other parts of an application, callbacks have to be specified. Any callback has, at the very least, one part of a component as an Input, and one as an Output. The Input not only gives information about the component part to the callback to be used, but also acts as a tripwire: As soon as anything changes in said component part, the callback will be activated. A callback can also have multiple Inputs and Outputs, as well as one or more States, which are able to relay information about additional component parts to the callback without triggering them like Inputs. Of course, since an Output component part for one callback can also be an Input for another, this can lead to extensive chains of callbacks, with many callbacks firing every time the user adjusts a component. There is, however, one limitation: There may not be a loop of any

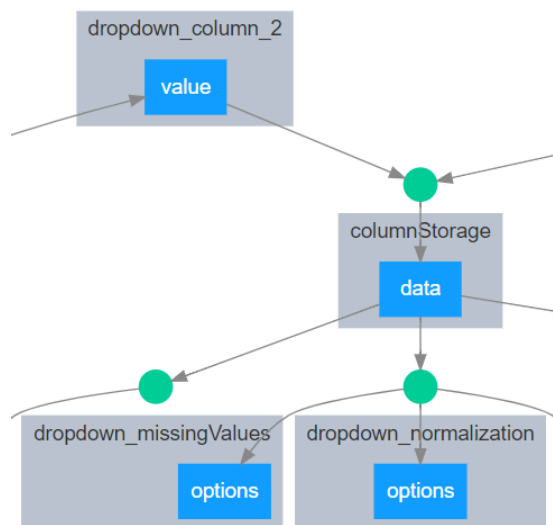


Figure 7. Small part of Dash Callback Graph.

size in the list of callbacks. This is disabled by Dash, as every callback would instantly and endlessly trigger another callback, freezing the program and potentially even the user's computer. As an example of the use of callbacks in PyWash, in Figure 7 two component parts (the 'value' part of 'dropdown\_column\_2' and an unknown component part) serve as input for a callback which outputs to the 'data' part of 'columnStorage'. Once this callback has fired, three different callbacks (referenced as green circles) that the 'data' part of 'columnStorage' is input for will also immediately fire if anything has changed in the component part. One of these callbacks, in turn, Outputs to three different components. It is clearly visible that the number of callbacks fired could rapidly increase depending on the complexity of the user interface and callback structure.

### 3.5.3 Storage

Some of these Dash components have a special function: To store any type of data. These can be used in place of global variables, as it can otherwise be wasteful to continuously re-calculate certain values. For example, in the PyWash code this is used to mark whether data has been uploaded or processed, as well as information about column data of the dataset ('columnStorage', as seen in Figure 7). Unfortunately, the storage components can only take JSON-serializable data, so the SharedDataFrames themselves could not be integrated into the PyWash user interface itself, instead being used as a global variable.

### 3.5.4 Applied Dash Strategies

First of all, to maintain oversight in how different components are layered, both the Dash layout and callbacks typically take up a lot of room in terms of lines of code. In order to not lose focus of everything, I separated the Dash code into four main files: the 'DataCleaning' file, which housed the part of the layout used for data cleaning (as seen in Figure 2), the 'Visualization' file, which housed the part of the layout used for visualization (as seen in Figure 5 and 6), the 'Callbacks' file, which housed all the callbacks used for the user interface, and the 'Main' file, which housed the main PyWash code that called the other three files, as well as the code for importing data (as seen in Figure 1) and some general code to make sure the various files work well with each other. I would have liked to do the same to the callbacks, but they start behaving oddly when put in separate files, so I had to keep them in a single file.

Secondly, when creating the user interface, I was careful to keep the layout ‘stretchy’. By this I mean that the user interface should stay as usable as possible when used on smaller resolutions, as not everyone has a full 1920x1080 screen available for PyWash (as seen in Figure 8). Although Dash DataFrames do not work well with this feature (as in, do not compress), it is not a major issue as they are still accessible at the bottom of the page if necessary.

### 3.5.5 Using the PyWash application

The current PyWash application is freely available on GitHub (Dave de Vos, 2020) , just like the previous year’s iteration (Yuri Maas, 2019). In order to use the current PyWash application, it is required to download (Clone) the application from GitHub. Then, using the ‘requirements.txt’ file, make sure that all packages listed have the right versions installed (It is assumed that the user already has a working installation of Python). After which, either run App.py yourself, or edit the locations used in Run\_PyWash.bat for easy access when running the application multiple times. Either of these methods will deploy PyWash on http://127.0.0.1:8050 on any browser. To access it, type or copy/paste the previously mentioned address into the address bar of said browser.

## Pywash Browser Interface

The screenshot displays the PyWash browser interface with the following sections and controls:

- Check or change column types:** A dropdown menu labeled "Select to check or chang..." and a smaller dropdown menu.
- Inspect found anomalies per column, change column type or delete columns that are not anomalous:** A dropdown menu "Check which co...", a "Select ..." dropdown, and a "Select All" button. Below these are two buttons: "Selected items are not anomalies" and "Selected items are anomalies, handle them".
- Choose preferred method for handling outliers:** A dropdown menu showing "Slow & Precise: Mark in an... ×".
- Select column to clean missing values, only for regression or classification target variable:** A dropdown menu labeled "Select desired columns for missi...".
- Test for duplicated rows?:** Radio buttons for "No" and "Yes" (selected).
- Normalize column(s)?** A dropdown menu labeled "Select desired columns for normalizati...".
- Standardize column(s)?** A dropdown menu labeled "Select desired columns for standardiza...".
- Start Data Processing:** A prominent button at the bottom center.

Figure 8: PyWash retains its structure when squished to around a third of my screen



# Chapter 4: Conclusion

In this final bachelor project, we (consisting of me and the members of my Bachelor Project Circle) have identified issues with last year's PyWash version and developed methods of improving these issues. In doing so, I have found answers to both of the research questions, although my work focused on the usability of the PyWash application. It improves upon the previous PyWash with a new user interface completely rebuilt from scratch, and many features of the previous PyWash were adapted to adhere to a smoother user experience with as many automated features as possible.

## 4.1 Future Work

Whilst steps have been made to improve upon the initial iteration of the PyWash application, perfection can only be found through iteration.

The data quality detection method is still not fully integrated into the PyWash application and user interface, and there are almost certainly still bugs and edge cases present in the current application. Furthermore, there is still work to be done in automating parts of the cleaning operations, especially in missing value detection.

There is also always more work to be done in extending the types of datasets and files accepted by PyWash, as file types like Microsoft Excel (.xls and .xlsx), JSON and SQL-styled datasets are currently not supported.

Lastly, there is always more work to be done on the Dash side of PyWash, from maintaining and improving the user interface as new Dash functionality becomes available to adding more interactivity to changing around interface choices (little work has been done on the aspect of CSS styling, other than basic positioning) to improving on the list of usable visualizations.



# Bibliography

- analcata\_data\_gsssexsurvey*. (2014, 09 29). Retrieved from OpenML: <https://www.openml.org/d/506>
- Chardet - PyPI*. (2017, June 8). Retrieved from PyPI: <https://pypi.org/project/chardet/>
- CleverCSV - PyPI*. (2020, May 20). Retrieved from PyPI: <https://pypi.org/project/clevercsv/>
- Dash Overview - Plotly*. (2020, April 2). Retrieved from Plotly: <https://plotly.com/dash/>
- Dave de Vos, T. Q. (2020, 6 30). *PyWash2*. Retrieved from GitHub: <https://github.com/Pywash2/Pywash2>
- OpenML. (n.d.). *OpenML Home*. Retrieved from OpenML: <https://www.openml.org/>
- pandas - Python Data Analysis Library*. (2020, March 18). Retrieved from pandas: <https://pandas.pydata.org/>
- Ratté, S. (2016). An Unsupervised Approach for Combining Scores of Outlier Detection Techniques, Based on Similarity Measures. *Electronic Notes in Theoretical Computer Science*, 329:61-77.
- scikit-learn: Machine Learning in Python*. (2020, January). Retrieved from scikit-learn: <https://scikit-learn.org/stable>
- Yuri Maas, L. C. (2019, July 19). *Pywash*. Retrieved from GitHub: <https://github.com/Pywash/Pywash>
- Zhang, J. (December 2018). *Automatic Data Cleaning*. Technische Universiteit Eindhoven.

