**ECBS 5148 Data Architecture for Analysts**

**Individual Assignment - Report**

**Gerold Csendes**

# Pump it Up: Data Mining the Water Table

## Executive Summary

The context to my data product was to provide the data-related foundations to participating in DrivenData’s Pump It Up machine learning challenge. A wide-range of learning objectives were planned to be demonstrated but some of them turned out to be not relevant to my use-case. This was due to the real-life nature of this data product. Though the ‘base’ data provided by DrivenData turned out to be a lot messier than expected and also the entities present were different from my expectations, I still think that my final data product can be considered as a suitable solution for the problem at hand. In the rest of the report, I will walk you through my work during which I will highlight the demonstrated learning outcomes.

## Data Sources

The Pump it Up challenge provides a dataset of 57 seven attributes. I have already explored this dataset for my Data Visualization assignment and I concluded it to be very messy. The labels are provided separately but can be joined one-to-one to the train features. In addition to that, we also have the test attributes.

I want to do some advanced feature engineering, in this case calculating distances for a particular water pump station. I want to calculate what is the closest region center (capital) to a pump location. The reason for doing so is that I suspect water pumps may be harder to maintain further away from the capital of region due to the lack of available resources, mainly expertise and pump parts.

For this, I will need to have data on the regions and their centers. I am going to scrape this data from this website: [URL](https://www.citypopulation.de/en/tanzania/cities/) which lists the regions and their capitals along other -for this exercise irrelevant- attributes.

To calculate the distance between a water station and the closest city center, I am going to use the [HERE API](https://developer.here.com/). First, I will need to scrape the coordinates of the capitals and then using the two coordinates, I am going to be able to fetch distance data from the API. Since this API is super fast, and free subscribers are allowed to make 250,000 requests monthly, I will query distances between a water station and every capital. Then, filtering for the shortest distance will be the capital closest to a specific station.

To enable my model to also account for differences in income, I will again scrape data. In this case, I will scrape [Wikipedia](https://en.wikipedia.org/wiki/List_of_regions_of_Tanzania_by_GDP) to get the regional GDP per capita (of 2016) data.

## Table previews

### Features

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **owner** | **gps\_height** | **lon** | **lat** | **region** | **basin** | **water\_quality** | **…** |
| 1111 | XY | 1335 | 37.202 | -3.22 | Kilimanjaro | Pangani | soft |  |

The features data table contains the information of an inspection at a specific date. It records the date and the -mostly constant – attributes of the water station. This table is redundant, thus, it will be normalized.

### Labels

|  |  |
| --- | --- |
| **ID** | **status\_group** |
| 1111 | functional |
| 2222 | non functional |
| 3333 | functional but needs repair |

The labels record the inspection result.

### GDP

|  |  |
| --- | --- |
| **Region** | **GDP per capita in USD (PPP)** |
| Dar es Salaam | 4,415 |
| Mbeya | 4,236 |

GDP contains regional income date for 2016. Since inspections range between 2010 and 2012 I think this data will do good enough for my data product.

### Capitals and coordinates

|  |  |  |  |
| --- | --- | --- | --- |
| **Region** | **Capital** | **Latitude** | **Longitude** |
| Arusha | Arusha | -3.377580 | 36.687684 |
| Dar es Salaam | Dar es Salaam | -6.805026 | 39.219950 |

The coordinates for each capital will be scraped and joined to the capitals table.

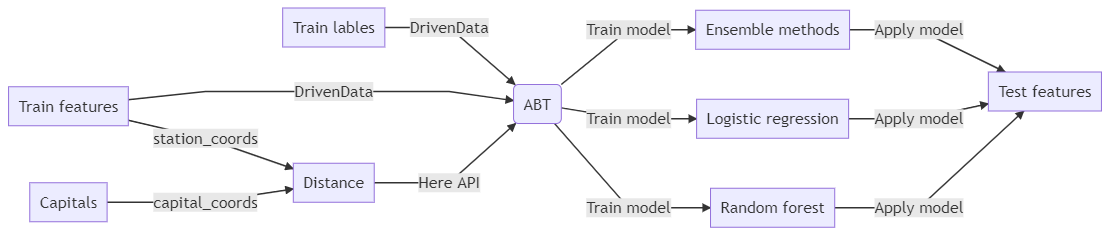
### Distances

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **wpt\_name** | **wpt\_lat** | **wpt\_long** | **capital** | **capital\_lat** | **capital\_long** | **dist(km)** |
| XY | -3.22 | 37.202 | Dar es Salaam | -6.805026 | 39.219950 | 569 |

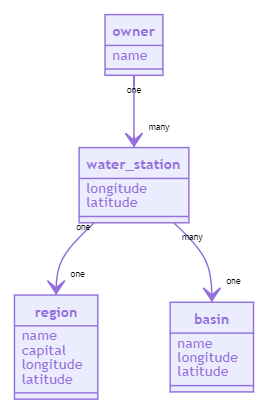
The distances table contains all water stations and capital combinations. There will be 5000 (No. station) \* 22 (No. region) = 110.000 records in this table.

## Conceptual model

ABT means Analytical Base Table which will be a single table ready-for-analysis. In other words, this will be my data product.



## Entity Relationship Model



## Technical and Legal Constraints

I will only use publicly available data and open source software tools. I will use, in particular, Python, SQLite, R and OpenRefine for data-related purposes and use Frictionless Data and Mermaid for documenting. I will work on my laptop, so data should fit into that.

Since the `core` dataset (DrivenData) is publicly available and I found no information indicating any legal constraint, I think I can use this dataset as it is. The robots.txt may prohibit me access to the material I am planning scrape, in which case I will have to look for other sources.

## Learning Outcomes Demonstrated

I am planning to demonstrate the acquired skills of this course through the following examples, grouped by class topic:

### Data Architecture

Separate important from unimportant features: the labels data contains also non-essential attributes which will be dropped

Represent a mental model visually: demonstrated in this document, at the ERD section

Create diagrams with Mermaid or other tool: demonstrated in this document, at the ERD section

### Data Modeling

Recognize tidy data: the data downloaded from DrivenData is not tidy but the final dataset will be

Create logical model for simple relational data and represent it with Entity-Relation Diagrams: demonstrated in this document, at the ERD section

Create and query a simple database in SQLite or other RDBMS: data will be stored in SQLite

Understand and apply normal forms 1-3 to simple relational data: normalization 1-3 will be applied on the data product

Model many-to-one relationships: conceptually demonstrated in this document, at the ERD section and will be implemented in SQLite

### Data Structures

Build a binary tree from simple ordered data: data tables will be indexed in SQLlite

Compare different data structures: TBA

### Data Serialization

Compare popular serialization formats fixed width, CSV, JSON, XML, YAML, JSONlines, Parquet: I/O and memory usage will be compared between CSV and Parquet for the biggest, labels dataset.

Explain the tradeoffs in data serialization: the results of the former will be elaborated

### Data Quality

Use string functions in OpenRefine or other tool to normalize text data: categorical data are messy, especially the name of the owners

Save, edit and replay changes in OpenRefine on different datasets: OpenRefine will be used for cleaning purposes and the steps implemented will be saved in json for transparency and reproducibility.

### Data Integration

Understand basic robots.txt structure: this will be inspected before scraping data

Use wget, curl or other programmatic tool to download data from the web: websites will be scraped

Create a Data Package to share data and metadata together: the final data product will be documented this way