April 2020

Technical Report

Digital Water

About the project

The project was born from a university challenge in which students were tasked to solve one of the problems that "Agua y Drenaje de Monterrey" currently faces, the company in charge of supplying water to the state of Nuevo Léon, México. Our team excelled at this challenge, and we were selected to present our idea to the CEO and the board of directors. Our team saw a great problem and an opportunity to solve it and wanted to further develop our project since we are convinced that it has potential to solve the current water crisis. As of today the city has 10 months of water left for their citizens.We have received mentorship from ATOS and Saturdays.AI (NGO)

Summary

In this technical report, you can find out about the problem we are tackling and the solution that we identify: AI to revolutionize the water industry. This report contains all the details regarding the technical approach taken to solve the problem. Please note that since this project may be subject for improvement in the future or subsequent Atos IT Challenge phases. For any doubts, please contact us via email and we will be happy to responde to the best of our capacity.

Problem Definition

Every problem needs a solution

There is a water crisis in Mexico and all over the world, the core problem of the water crisis is the low value that water has in the cities, this has created a lack of effective water management, especially in the 2 main cities of Mexico, Monterrey and Mexico City. This, in turn, has caused many cities to get close to reach "Day Zero", the point at which there is no water to meet the needs of citizens.

Our solution will tackle this problem.The scope is to solve the problem first for the city of Monterrey as the city has 10 months left of water for its citizens. Using Artificial Intelligence (AI), we first provide daily and monthly water demand predictions, and then segment demand into volumetric flowrate of water that will be needed to extract from particular resources, such as dams, wells or rivers. The goal is to provide water companies with insightful information to change operations to immediate actions rather than reactive responses to the crisis.

Our project will be expanded to a water leakage detection system as 32% of water in the distribution pipelines is lost and automatic control of water pumps. These 2 expansions are still in early development state, and will not be presented as part of the App Development Phase.

Lastly, we are taking all the actions required to solve the water crisis in our city, they include 2 local and 1 international players:

- Agua y Drenaje de Monterrey: We already have the interest of the agency. A meeting with Gerardo Garza González (CEO of Agua y Drenaje) and the board of directors was postponed because of COVID-19.
- Iniciativa Nuevo Leon 4.0: We are in the process of meeting with Eduardo Garza T Junco (CEO of the initiative) to see the possibility of including our project in the state initiative goals.
- AI for Good (United Nations): Our project was invited to present at the AI for Good Summit by Ahmed Rashad Riad (Development manager of the initiative)

Deliverables

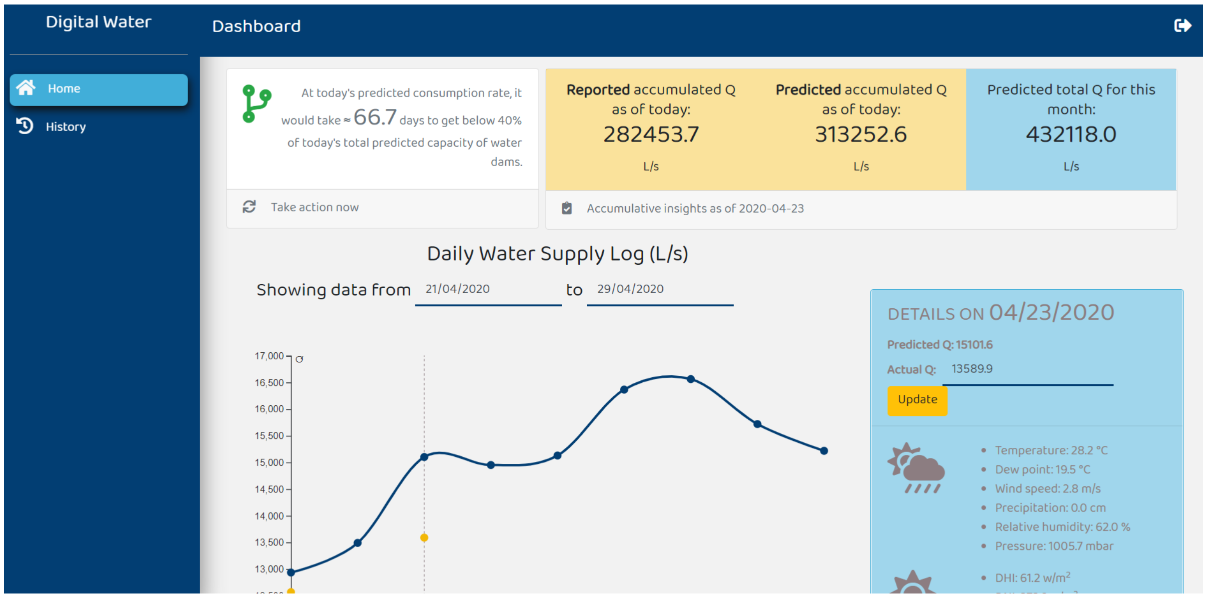
Below are the required deliverables. If something is not clear, please don't hesitate to email us.

Application

Our application is web based. As of today, no username or password is required to access, however, in the future, we will add this function to avoid unwanted visitors. We are preparing user integration with LDAP Corporate

You can visit our web application here:

https://digital-water.herokuapp.com



At today's predicted consumption rate, it would take ≈ 66.7 days to get below 40% of today's total predicted capacity of water dams.

Reported accumulated Q as of today: 282453.7 L/s

Predicted accumulated Q as of today: 313252.6 L/s

Predicted total Q for this month: 432118.0 L/s

Accumulative insights as of 2020-04-23

Daily Water Supply Log (L/s)

Showing data from 21/04/2020 to 29/04/2020

DETAILS ON 04/23/2020

Predicted Q: 15101.6

Actual Q: 13589.9

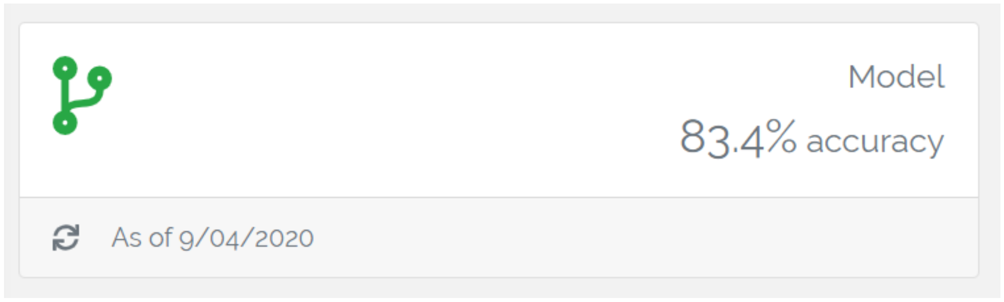
- Temperature: 28.2 °C
- Dew point: 19.5 °C
- Wind speed: 2.8 m/s
- Precipitation: 0.0 cm
- Relative humidity: 62.0 %
- Pressure: 1005.7 mbar
- DHI: 61.2 w/m^2

Use Cases

Understand usage of our application

The intended use of our application is to be used for better decision making regarding water management. Our AI models predict volumetric flowrate that needs to be extracted from natural resources such as dams or wells, to fulfill water demand in the city. This information is crucial for strategic planning and operations.

To test that our application works correctly, access our web application, and see how graphs are showing predicted and real values of water demand, as well as current dam capacity. In the right side corner you can see the features used for today's prediction. Also, an important aspect of our model is that it lets you see today's, accumulated of the month and monthly prediction of water demand. This is incredibly resourceful to strategic planning. Lastly, To further prove our predictions, a model accuracy indicator is shown above the first graph. Daily predictions are done automatically therefore no user action is needed to predict and display results.


Model

83.4% accuracy

As of 9/04/2020

Solution Architecture

Objectives of design architecture

This project has already attracted the attention of the local water state agency: Agua y drenaje, with this in mind; the design of our architecture has the objectives of being:

1. Scalable — With a modular architecture

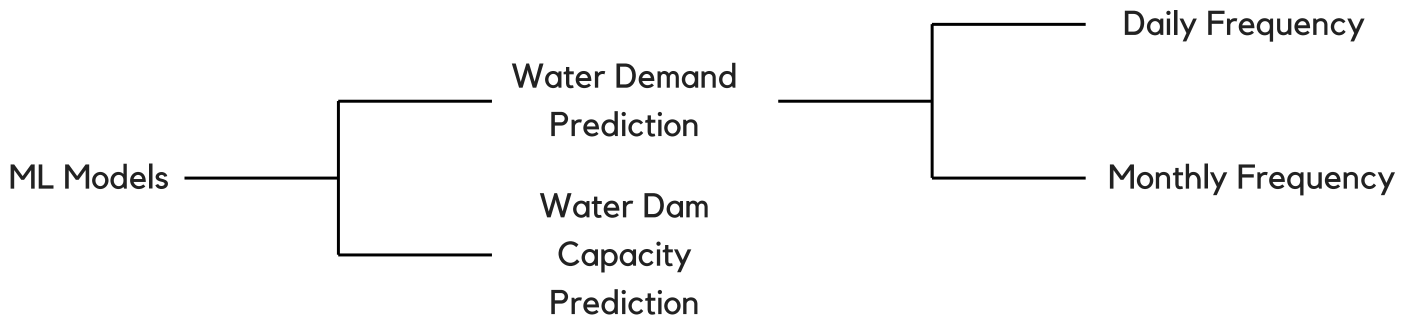
2. Interoperable — Through dynamic pipelines and instances

3. Solid & Ease to use — With extensive analysis of the problem, that gives us simple, easy to understand architecture and code

Solution Architecture

Our solution is segmented into three main modules:

1 Artificial Intelligence – The heart of our application. Each one has different scope and technical approach.

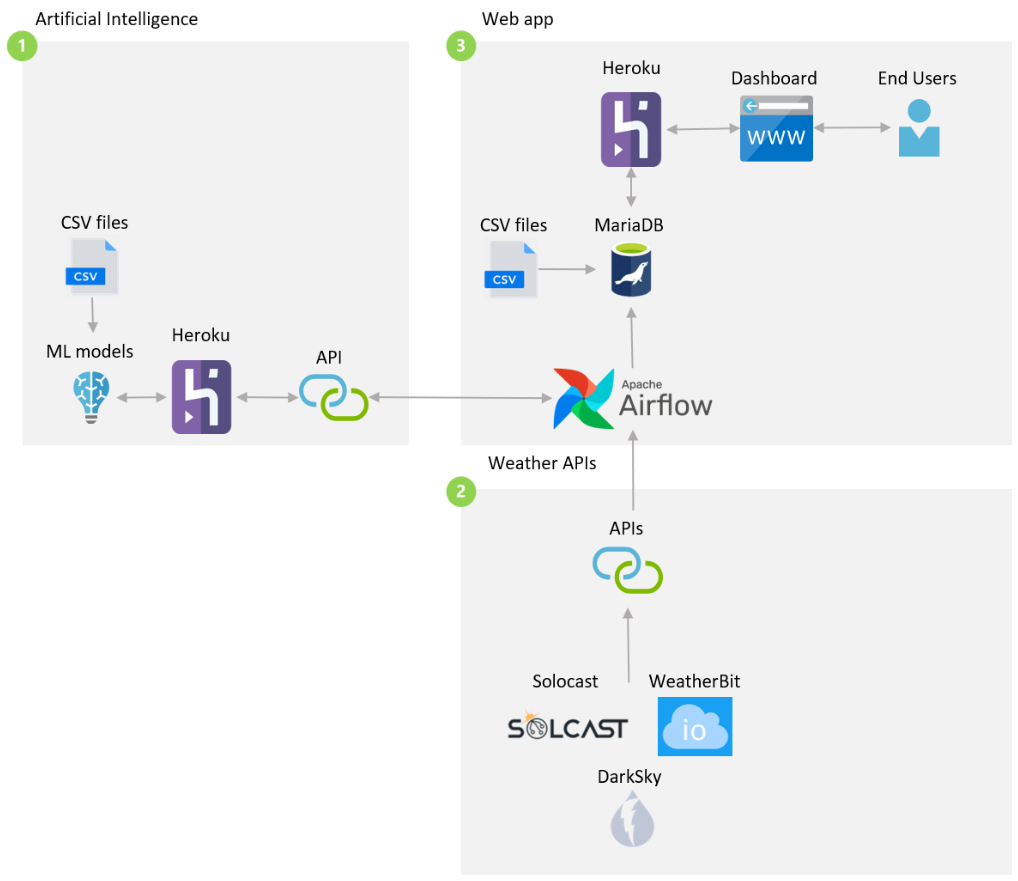


2 External APIs – Used for data extraction. Model requires constant meteorological features.

3 Web App – Apache Airflow is integrated into heroku from processes management, MariaDB for storage and Heroku to deploy the dashboard.

Module Interaction

Our modules interact in the following way as shown in the diagram:



Our current prototype uses Heroku, a container-based cloud Platform as a Service (PaaS). We used it to deploy, manage, and scale our apps. Additionally, we implemented Apache Airflow into Heroku. Airflow is a platform to programmatically author, schedule and monitor workflows. We used airflow to author workflows as directed acyclic graphs (DAGs) of tasks. The airflow scheduler executes our tasks of predictions on an array of workers while following the specified dependencies.

DarkSky, WeatherBit, SolCast are used to request values of features from different days, the system uses Airflow for daily requesting the features from the Weather APIs and posting a request to the AI's API to get the predictions, then storing the features and the prediction into the MariaDB database. Finally heroku is used for displaying the insights and workflows for the user in the Dashboard.

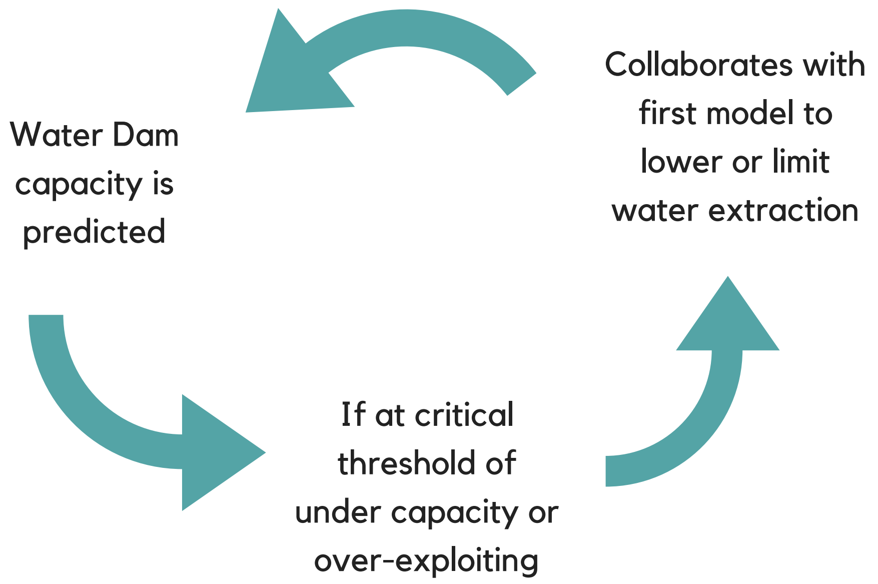
Collaborative AI

Two models interacting

Our main ML models - water demand prediction - forecasts the volumetric flowrate needed to extract from natural resources to fulfill demand.

An important resource for water extraction are water dams. Water dams provide around 50% of the total water.

Our secondary model forecasts water capacity that dams currently have. This model interacts with our main model, to ensure that only the needed water is extracted, avoiding over-exploiting of one resource.

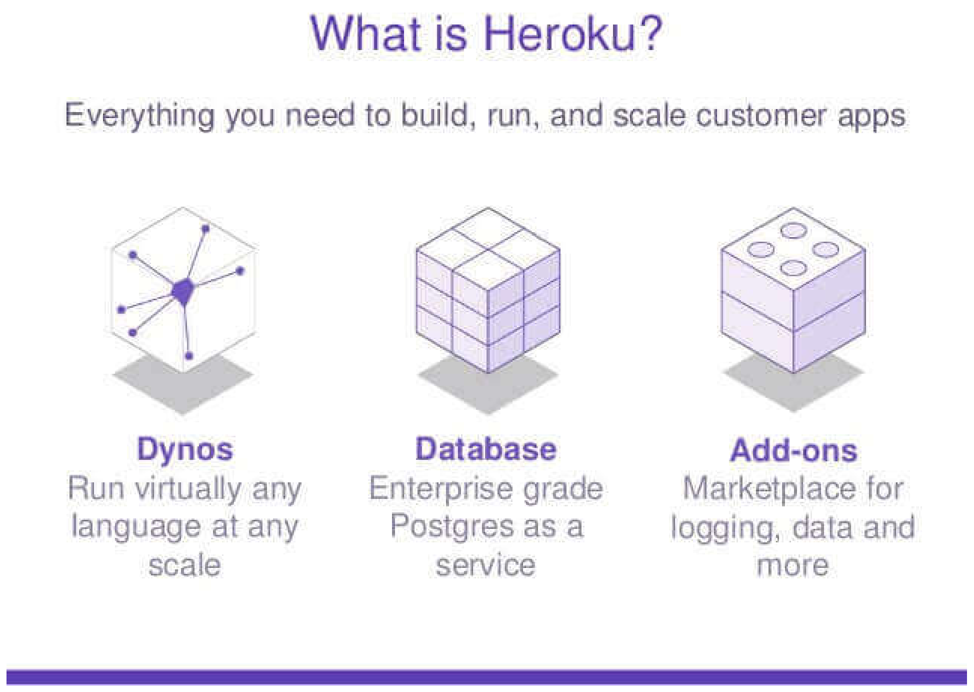

Water Dam capacity is predicted
If at critical threshold of under capacity or over-exploiting
Collaborates with first model to lower or limit water extraction

Cloud Solutions

Heroku to run them all!

Our cloud solution is Heroku is a widely used commercial PaaS. In addition to being designed as a container-based cloud PaaS, Heroku also supports a number of widely used programming languages. Also, the tools, services, and workflows provided by Heroku make it easier for developers to build, manage, deploy, and scale a variety of enterprise applications.

Visit https://www.heroku.com/ for further information


What is Heroku?
Everything you need to build, run, and scale customer apps
Dynos
Run virtually any language at any scale
Database
Enterprise grade Postgres as a service
Add-ons
Marketplace for logging, data and more

The PaaS further enable developers to focus extensively on various aspects of the application without creating or maintaining infrastructure. Ultimately Heroku uses AWS to deliver their services.

Database

To create our ML models we crafted and used a database containing three main categories:

Category	Source
Meteorological	National Solar Radiation Database https://nsrdb.nrel.gov/
Time-related	Made by us.
Real volumetric flowrate water extraction	Agua y Drenaje de Monterrey -Private Non-public Data
Total	Around 35 unique features

For detailed description, see appendix A.

Software

Programming Languages used

BackEnd	Python
FrontEnd	JavaScript Bootstrap D3.js

BackEnd includes the ML models, as well as model deployment. FrontEnd includes design (Bootstrap) and graph generation (D3).

Addtionally, some of the libraries used for the ML models are:

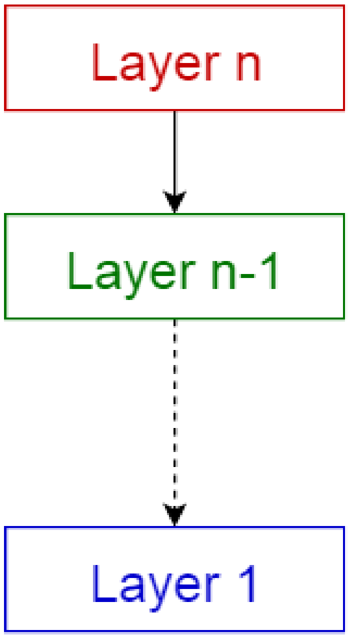
Python Libraries	Tensorflow: Deep Learning Sklearn: Pre-processing of data Pandas: Data Management Numpy: Arrays & Concatenate Datetime: Time frame Matplotlib.pyplot: Plots flask: Flask app creation json: Requests joblib: Model exportation

Software Architecture

We used two main patterns to structure our programs that can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer.

They are simplified as follows:

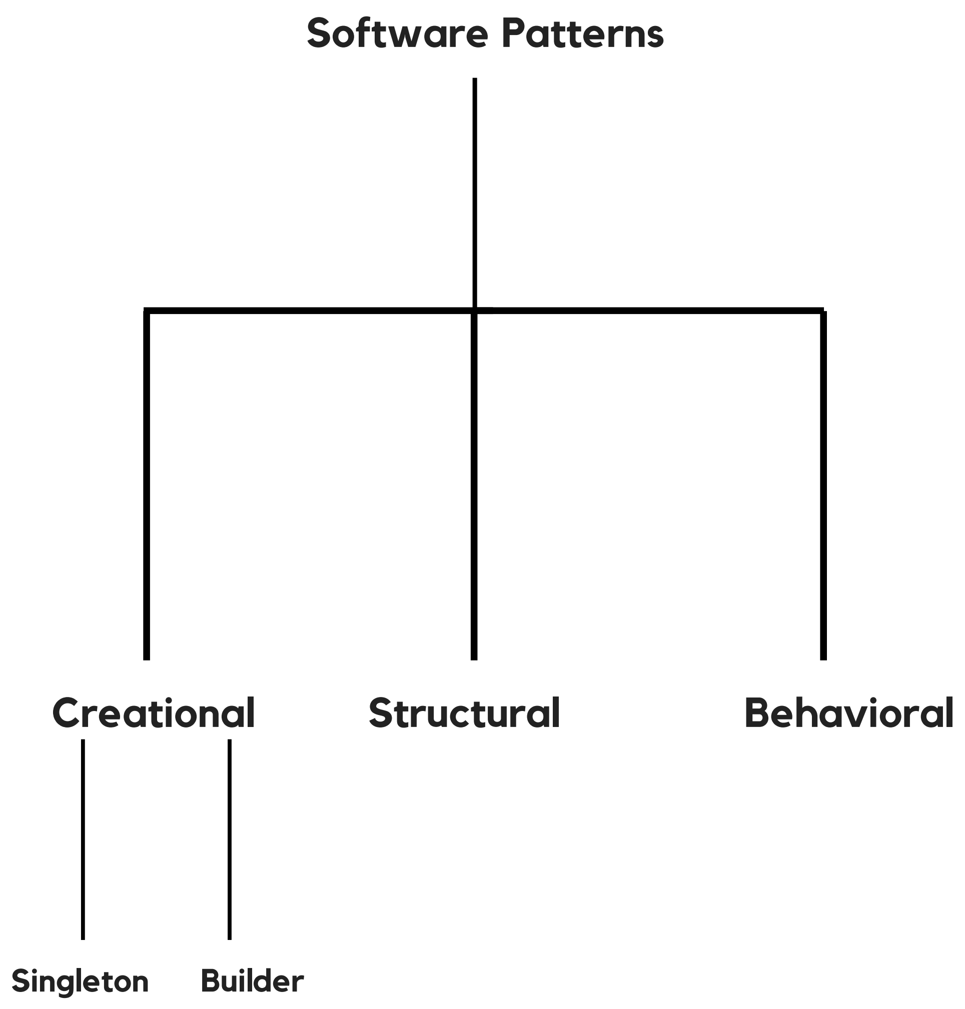
- Presentation layer (UI layer)
- Application layer (service layer)
- Business logic layer (domain layer)
- Data access layer (persistence layer)
- Data request layer



Please refer to the graph shown in the Module Interaction section to understand the used software layers

Software Patterns

A design pattern provides a general reusable solution for the common problems occurs in software design. The patterns typically show relationships and interactions between classes or objects. The idea is to speed up the development process by providing well tested, proven development/design paradigm. Design patterns are programming language independent strategies for solving a common problem. That means a design pattern represents an idea, not a particular implementation. By using the design patterns we can make our code more flexible, reusable and maintainable.



Creational

Singleton

- **Dashboard:** Only one Flask application object is instantiated and referenced throughout the code. This ensures that there is only one application running. Additionally, a single database connection object is used throughout the application.

- **Prediction models API:** As with Dashboard, the Flask application object is a singleton.

- **Airflow:** In the respective DAGs for daily and monthly predictions, there is no global database connection object. Instead, a MySQLHook (the Airflow way to set connection to a MariaDB or MySQL database) object is instantiated in the tasks that make use of the database. We decided that, given the nature of ETL processes that can fail and should be restarted at the point of failure (fault tolerance), a global connection object might become problematic. Having a local MySQLHook would better allow the encapsulation of each task in the ETL flow described in each DAG and provide a "cleaner slate" to restart a failed task (if needed).

Builder

Using the builder pattern was briefly considered to differentiate between the construction of empty objects and objects already populated with data for the classes MonthlyX and DailyX. Given that those two were the only possibilities (all objects would also have the same attributes) of object instantiation, we determined these classes were not complex enough to benefit from the Builder pattern.

Instead, we covered our building possibilities by defining for each class a constructor with default values. This way, if initialization values were provided, an object would be already populated with data upon creation. Otherwise, the constructor would produce an "empty" object with default values.

Behavioral

Chain of responsibility

Our ETL process is modeled in Airflow as a directed acyclic graph (DAG). We constructed the flow to follow a chain that forces an order and dependency in the tasks for extraction, transformation and loading. Below is the DAG created for the daily prediction ETL.

Source Code

Notebooks & Github

1. Daily Water Demand – Visit our Google Collab notebook here: https://colab.research.google.com/drive/1ls6XuVbOJjJP1F9ZjJRd2hWSwQUKESRT

2. Monthly Water Demand – Visit our Google Collab notebook here: https://colab.research.google.com/drive/1umFnayEYQ3lDOoei0JMbrNqomOMuKOuO

3. Water dam capacity – Visit our Google Collab notebooks here:
   - Dam 1: https://colab.research.google.com/drive/1IuVJ6OCJHIrFRfkjmC7WHXrrNCGjTVJL
   - Dam 2: https://colab.research.google.com/drive/1ZQbLS_1wW4WTGFeVk4532oMwpsGqEYiJ
   - Dam 3: https://colab.research.google.com/drive/13AvXoikHzK4LCTasyd0hc9XOqqSbvzrw

4. Datasets used in previous notebooks: https://github.com/jarturoa/digitalwater-AI-api/tree/master/dbCSV

5. Github for all other source code: https://github.com/jarturoa/digitalwater-AI-api

AI Implementation

The heart of the project

The heart of the project lays on the Machine Learning models. Being able to work and understand a complex problem that deals with seasonality and time series as well as human behavior (demand) requires specialized and complex models.

Each model had a different approach:

- For the water demand prediction models, the goal was to predict the volumetric flowrate of water needed to extract from natural resources in order to fulfill demand.
- For the water dam capacity prediction model, the goal was to predict the current capacity that the dam holds.

For the water demand prediction in daily frequency, the machine learning algorithm applied was Long Short Term Memory (LSTM). The reason being that such models are perfect for complex models that are linked to time series and seasonality behavior.

This algorithm is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. LSTM has feedback connections, therefore can process entire sequences of data. LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

AI Implementation

The heart of the project

An interesting fact is that this models uses previous predictions to generate more accurate predictions. At first, we tried testing using Random Forest, however this approach does not take into account the importance of time series. Therefore, we tried several LSTM architectures.

Some of these changes were:

- Layers within the LSTM architecture
- Neurons
- Epochs
- Batch Size

Please refer to our Google Collab Notebook to see more explained detail

The first step was to clean and prepare the dataset. Also, we did feature engineering to create new features based on existing ones. Such as extract information from dates and create min/average/max feature columns.

The second step was to prepare the dataset into train, test and validation sets. In this case, our dataset contained 6 years of data, therefore we splited our dataset into 5 years of training and 1 year of test. LSTM require at least 1 year of testing data, because it needs to include all the seasons.

The models were fit using Tensorflow. For the water demand prediction - daily frequency, after several trials, decided to use a Sequential and LSTM input layer and a Dense output layer. For the parameters, 70 neurons and 50 epochs seemed to provide the best results.

AI Implementation

The heart of the project

For the water demand prediction in monthly frequency, the machine learning algorithm applied was DNN (deep neural network). After several tests, DNN showed the best results.

At first, we ran an XGBoost Random Forest with the initial 32 features but because of the complex feature engineering process needed to gradually improve the model and a recommendation from an expert, we moved to directly to test with neural networks, from this approach however we decided to only use 17 of the most feature important columns.

Our initial trials started with the following specs:

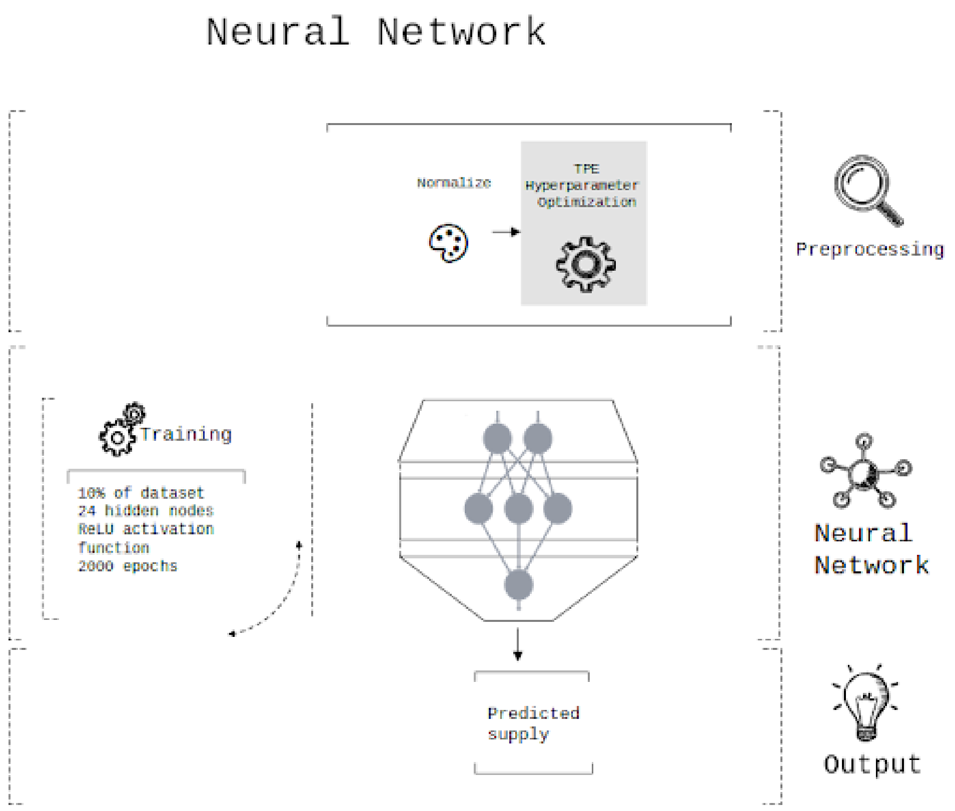
Trial	Nodes	Activation Function
1	240-120	ReLU
2	240-120	Sigmoid
3	240-120	Softmax
4	240-120	Hyperbolic Tangent

Variation came from different number of nodes that we tried.At the end we decided for a Deep Neural Network using fewer nodes and a ReLu activation function as this was the best combination for our problem

AI Implementation

The heart of the project

We took 10% of the dataset for final testing and the rest for training and validation. All data was subject to analysis using WEKA that also helped for the TPE hyperparameter optimization (we found that this was the most efficient method to find the best hyperparameters for the model). For the model training the data was normalized, it was sent to the model architecture. After the initial 17 input layer for every feature the model has a 12 node hidden layer, then a 24 node hidden layer and finally an output layer of 1 node.

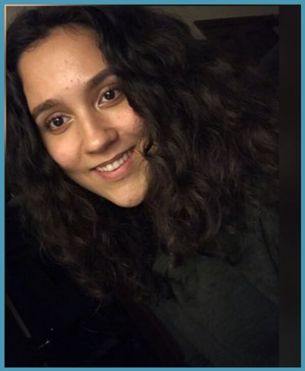


About the Authors

Ricardo Perez

Chemical Engineering

Spark Cognition Challenge,
ATOS IT Challenge Semifinalist



Alejandra Garza

Computer Science

ATOS IT Challenge Semifinalist
Global Goals Jams Monterrey 2017

Arturo Arellano

Informatics Engineering

Spark Cognition Challenge,
ATOS IT Challenge Semifinalist
1st Place Tsinghua University AIOps Challenge

Special Thanks

Francisco José Ruiz Jimenez

Atos Mentor

Jair Garza

Saturdays.AI Mentor

IBM Consultant

Agua y Drenaje de Monterrey

For providing valuable data

Appendix A.

The features used in the **water demand prediction - daily frequency** are:

- Solar Irradiance
  - Diffusive Horizontal Irradiance (DHI) [W/m2]
  - Direct Normal Irradiance (DNI) [W/m2]
  - Global Horizontal Irradiance (GHI) [W/m2]
- Meteorological
  - Dew Point (DP)
  - Wind Speed (WS)
  - Rain
  - Relative Humidity (RH)
  - Temperature (T)
  - Pressure (P)
- Time-related
  - Year
  - Month
  - Day
  - Weekday (1: Yes, 0: No)
  - Weekend (1: Yes, 0: No)
  - Festive (1: Yes, 0: No)
- Volumetric Flowrate
  - Water extracted (Q) [L/s]

Solar Irradiance and meteorological features take minimum, average, and maximum values of the day.

Total features = 32

Appendix A.

The features used in the water **demand prediction - monthly frequency** are:

- Economic
  - INPC
  - INPP
- Demographic
  - Population
- Solar Irradiance
  - Diffusive Horizontal Irradiance (DHI) [W/m2]
  - Direct Normal Irradiance (DNI) [W/m2]
  - Global Horizontal Irradiance (GHI) [W/m2]
- Meteorological
  - Dew Point (DP)
  - Wind Speed (WS)
  - Rain
  - Relative Humidity (RH)
  - Temperature (T)
  - Pressure (P)
- Time-related
  - Year
  - Month
- Volumetric Flowrate
  - Water extracted (Q) [L/s]

Solar Irradiance and meteorological features take minimum, average, and maximum values of the day.

Total features = 17

Appendix A.

The features used in the water dam capacity model are:

- Accumulated Rainfall [cm]
- Dam level [M ft3]

Total features = 2