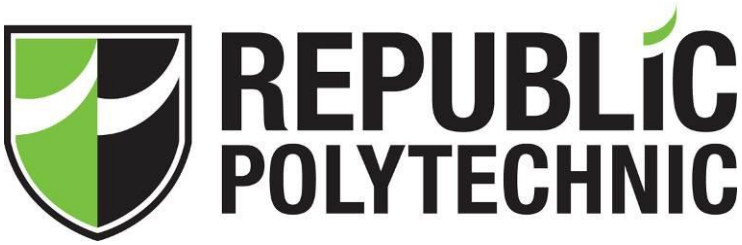


SMART WATER QUALITY MONITORING SYSTEM

Date of Submission: 30-JUL-2018

Submitted By:

LIM YUAN HER

School of Infocomm (SOI)

Republic Polytechnic

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank Mr. Tan Wee Siong for his guidance and advice through the course of this project.

In addition, I would like to thank my team mate, Mr. Lim Hong Loon for his co-operation and valuable inputs, without which, we would not have been able to complete this project on time.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

ABSTRACT

This report documents the design of an IoT-enabled Smart Water Quality Monitoring System using the Libelium Smart Water Sensor to monitor, collect and analyse data (temperature, pH, conductivity, dissolved oxygen(DO), oxidation reduction potential(ORP)) from various remote water catchment reservoirs to analyze the quality of water undergoing further treatment processes to make it suitable for human consumption. Data is transmitted periodically to the Libelium Meshlium gateway from the Libelium Smart Water sensor units, which in turn transmits the sensor data to the Microsoft Azure Cloud platform for data processing and analysis using machine learning techniques to perform classification of various water types as well as real-time visualization using Microsoft PowerBI dashboards and SMS/ email notification via Microsoft Logic App service. With the successful proof-of-concept implementation of this smart water monitoring system, an early warning system for water pollution based on the data analytic results can be designed to detect abnormalities so that preventive and corrective measures can be taken pre-emptively.

1 Introduction

Water is a precious resource which all organisms depend upon for survival. The process of treating raw water to make it suitable for human consumption is a complex and expensive operation. Traditionally, water quality monitoring is a 3-step process involving the collection of water samples, analysing it in the lab and conduct follow-up investigation, which is a manual process done by lab technicians and scientists. This process is error-prone and does not provide an indication of water quality beforehand. With the advent of wireless sensor technologies and Internet of Things (IoT) platform connectivity, it has become possible to perform real-time analysis of the water quality which can be used to optimize the treatment process parameters, resulting in total reduced costs and increasing the efficiency and effectiveness of the treatment process.

Based on this, a Smart Water Quality Monitoring System utilizing IoT and machine learning technologies is proposed to automate the water quality monitoring process, provide real-time data visualization via various platforms e.g. mobile, web etc. and notification via SMS and email to responsible parties of critical abnormalities so that pre-emptive measures can be taken accordingly.

2 Project Specification and Plan

2.1 Project Overview

2.1.1 Objective

The main objective of this project is to develop a cloud based real-time data analytic water quality monitoring system for Singapore water catchment reservoirs.

2.1.2 Scope

In order to ensure the timeliness of the project deliverables due to the short duration of this project (less than 2 months), the focus will be on developing a prototype to evaluate the feasibility of the project objectives identified in order to achieve a minimum viable product. Thus, the emphasis will not be on developing a ready for commercial production system, but on system proofing the viability of the system.

2.1.3 Assumptions

It is assumed that the system will adopt a cloud platform (Microsoft Azure) for data storage and processing. An Azure trial account will be used for the platform development in order minimize the costs involved.

The Libelium Smart Water kit will be used for data sensing and collection as it is readily available for temporary loan from the IoT lab to reduce the costs involved. Due to resource constraints, only one Waspmote Plug&Sense unit will be used for testing and development.

As there is only one Meshlium gateway unit available, the limited communication range restrictions and the need to share this unit with other project teams, the data collection activities will be carried out within the RPIC building itself. The water quality will be simulated using various materials with the sensor data measurements collected in the lab.

2.2 Functional Requirements

The diagram below illustrates the overall requirement for the Smart Water Quality Monitoring System:

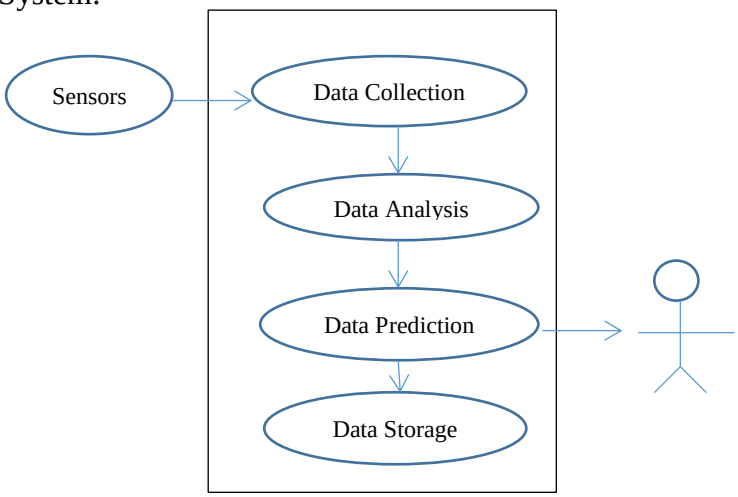


Figure 1 – Functional Requirement Use Caase

The subsequent sections below list the specific functional requirements of the Smart Water Quality Monitoring System.

2.2.1 Functional Requirement 1

The first requirement is that the system shall provide real-time monitoring of the water quality parameters i.e. pH, Dissolved Oxygen, Oxidation Reduction Potential, Conductivity and temperature for each sensor unit deployment location (the deployment locations are designated as North, South, East and West for the purpose of this project). This should include a visualization dashboard accessible via desktop, web and mobile devices.

2.2.2 Functional Requirement 2

The second requirement is that the system shall provide alert messaging via SMS and email notifications for abnormal water quality parameter detection and equipment malfunction.

2.2.3 Functional Requirement 3

The third requirement is that the system shall possess analytic capabilities e.g. using data analytics, machine learning etc. to provide the following metrics:

- Grading classification of the water quality e.g. Grade 1, Grade 2 etc.
- Overall System Alert Status (0 – Normal, 1 – Abnormal)

2.3 Project Plan

The figure below lists the project tasks and time allocated for each including the project milestone deliverables:

s/n	Task	June																						July			
		9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4
	Proposal																										
1	Prepare Proposal																										
2	Submit Project Proposal																										
	Setup/ Configuration																										
1	Waspmote Plug&Sense unit Programming																										
2	Azure Configuration (IoT Platform)																										
3	PowerBI Configuration (Visualization)																										
4	Logic App Configuration (Notification Services)																										
	Test Setup/ Data Collection																										
1	Data Collection (Water Measurement)																										
	Data Analysis																										
1	Analytics																										
2	Machine Learning																										
	System Testing																										
1	Test Data Generation/ Functional Check																										
	System Documentation																										
1	Report Writing																										
2	Powerpoint Slides Preparation																										
3	Project Presentation Preparation																										
4	Submit Project Report																										
5	Project Presentation																										

s/n	Task	July																												Aug
		5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1
	Proposal																												
1	Prepare Proposal																												
2	Submit Project Proposal																												
	Setup/ Configuration																												
1	Waspmote Plug&Sense unit Programming																												
2	Azure Configuration (IoT Platform)																												
3	PowerBI Configuration (Visualization)																												
4	Logic App Configuration (Notification Services)																												
	Test Setup/ Data Collection																												
1	Data Collection (Water Measurement)																												
	Data Analysis																												
1	Analytics																												
2	Machine Learning																												
	System Testing																												
1	Test Data Generation/ Functional Check																												
	System Documentation																												
1	Report Writing																												
2	Powerpoint Slides Preparation																												
3	Project Presentation Preparation																												
4	Submit Project Report																												
5	Project Presentation																												

Milestone

Figure 2 – Project Schedule

3 Business Analysis

3.1 Business Issues

The treatment of water from the reservoirs and water catchment areas is a multi-barrier one to comply with water quality standards specified in the Schedule of the Environmental Public Health (Quality of Piped Drinking Water) Regulations and international World Health Organisation (WHO) Guidelines for drinking water quality [2].

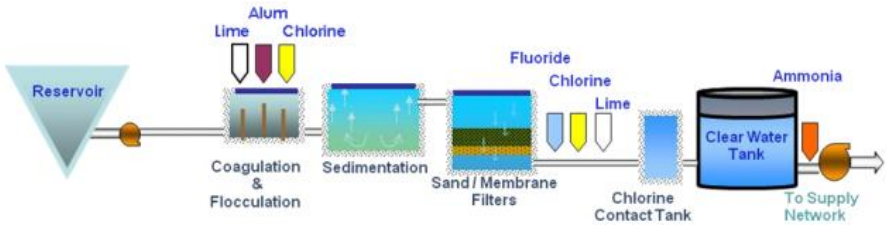


Figure 3 – Overall flow of Treatment Process

In the traditional water treatment process, the water quality at each stage is monitored and controlled through the Supervisory Control and Data Acquisition (SCADA) system, with online analytical equipment data verified with laboratory testing once every 8 hours. This is an expensive and time-consuming task. Furthermore, the quality and effectiveness are heavily dependent on the experience of the laboratory technicians involved. Any misdiagnosis can have detrimental impact on the resultant water quality, resulting in negative consequences on human health due to treated water that is not fit for human consumption. A more pre-emptive rather than a reactive approach needs to be adopted.

In addition, the use of traditional on-premise SCADA systems in traditional water treatment process requires a significant upfront investment to build and ongoing costs to maintain a fairly complicated infrastructure for operations. The on-site servers/ workstations need to be set up and maintained, and changes or repairs can cause costly and inconvenient delays. Furthermore, there is high risk of data loss if the servers become damaged in any way, unless significant investments and effort is made to build in redundancy and off-site storage of critical data.

These are 2 major issues that the Smart Water Quality Monitoring System will address in its system design and implementation.

3.2 Market Analysis

Singapore has a small population of 5.8 million and lacks natural fresh water sources, and as such, it relies on four main sources to meet its water demand: 40% imported from Malaysia, 30% reclaimed by NEWater, 20% rainfall collected in reservoirs or water catchment areas and 10% through seawater desalination. The Singapore government aims to acquire 50% reclaimed from NEWater, 30% from salt water desalination and 20% from rainfall collected in water catchment areas and reservoirs by 2060 [3]. Major national water projects such as NEWater (recycled water), the Deep Tunnel Sewerage System (DTSS), desalination and rainfall storage like the Marina Barrage has helped Singapore towards its goal of water independence. As a result, a thriving water-related industry has developed with more than 180 international and local companies involved.

The water industry is growing and is projected to expand by 10-15% annually. The current size of the market is estimated at US$1 billion, with growth drivers coming from strong demand from government-initiated projects e.g. construction of new desalination plants, NEWater facilities etc. [3]

Although Singapore has already put in place a sustainable water supply system through diversification of its water resources, investments in advanced water technologies and R&D work to look for even more efficient treatment processes is an ongoing endeavour. S$200 million has also been allocated by the National Research Foundation (NRF) under the Research, Innovation and Enterprise (RIE) 2020 Plan, thus raising total R&D funding for water to S$670 million over 15 years, to achieve S$2.85 billion of annual value-added contribution and 15,000 jobs in the water industry by 2020 [4].

Water conservation will become increasingly important in Singapore as water prices for household and industrial use continue to rise with cost-based inflation. Hence, there is room for significant growth in the water conservation and recycling equipment market in Singapore.

The local water catchments in Singapore comprise of 17 reservoirs and a network of storm water collection ponds which runoff into these 17 reservoirs throughout the island. To increase Singapore’s water catchment area to 90% by 2060, major

estuaries are dammed to create reservoirs to collect water from remaining streams near the shoreline [1].

There is a need to maintain water quality and minimize treatment costs for drinking water for better management of resources and effort.

3.3 Business Solutions

To address the issues identified in Section 3.1, the Smart Water Quality Monitoring System detailed in this report incorporates 2 major features that target to reduce the inefficiencies in the current process workflow and infrastructure:

1. Pre-classify reservoir water quality into different grades before the start of the treatment process to identify the treatment parameters required in advance to achieve optimization for maximum effectiveness and quality of the entire treatment process. With better visibility of the process required, the amount of sample testing required can be significantly reduced.

2. Employ messaging systems e.g. SMS, email etc. to notify relevant personnel of abnormalities in the water quality so that pre-emptive actions can be taken to address the issues involved. This can help to reduce the downstream treatment effort, thus reducing costs further.

3. Use a cloud-based system e.g. Microsoft Azure for data storage and processing. Microsoft Azure offers a tiered pay-as-you-go usage model, thus eliminating the upfront costs involved as compared to using traditional on-premise SCADA systems. In addition, it also offers scalable backup options without needing to incur significant infrastructural equipment costs and expenses from ongoing maintenance and upkeep.

4. Use dashboard visualization e.g. Microsoft PowerBI as it offers a develop once, multiple access via desktop/ web/ mobile devices out-of-the-box without requiring significant development efforts. This multiple access model equips the responsible personnel with readily available information at their fingertips for fast response and action to any situation.

4 System Design and Implementation

4.1 System Architecture

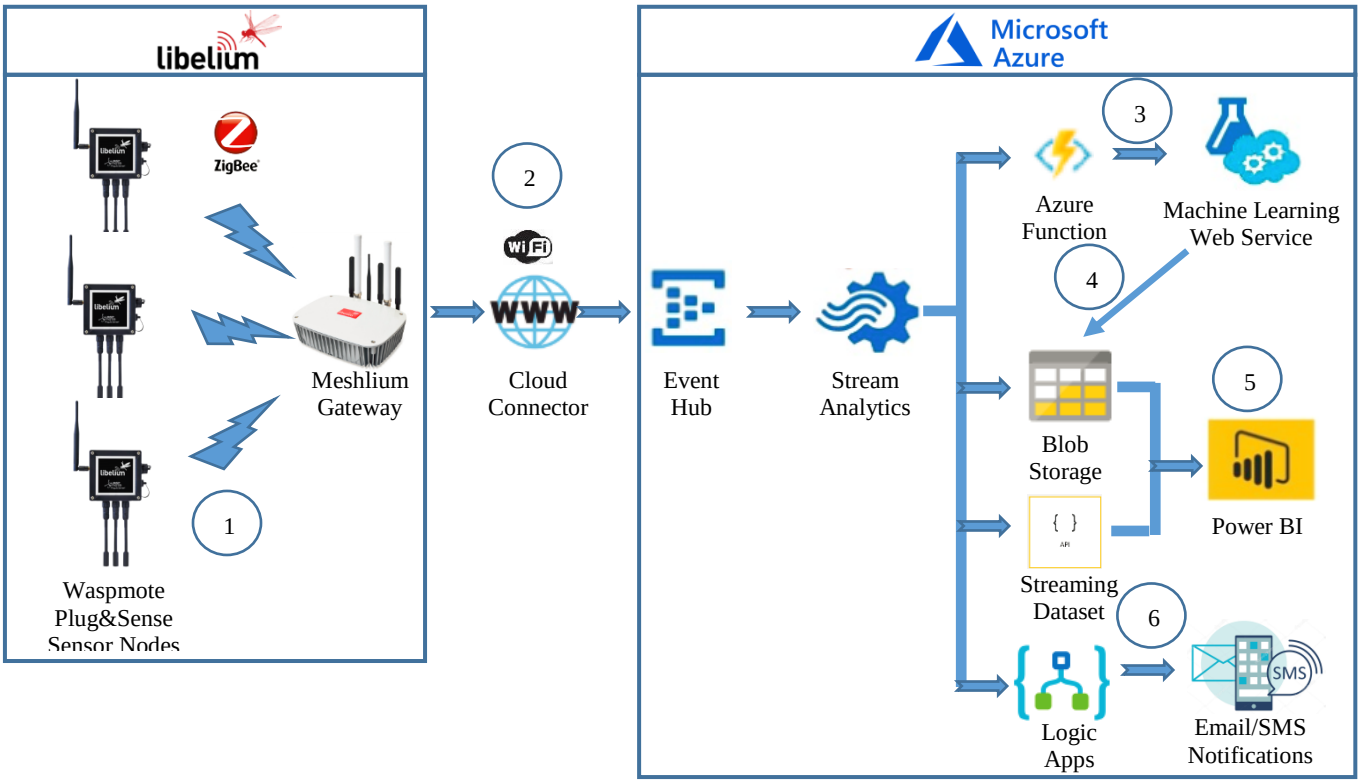


Figure 4 – Overall System Architecture

The figure above illustrates the overall system architecture.

1. Individual Waspmote Plug&Sense sensor units placed at strategic locations around the water catchment reservoirs transmit sensor data to the Meshlium Gateway via ZigBee communications.
2. Meshlium Gateway sends the sensor data via the cloud connector to the Microsoft Azure Event Hub.
3. Upon receipt of new data in the Azure Event Hub, the Stream Analytics job triggers an azure function to initiate the machine learning prediction via web service calls to the trained machine learning model.
4. The incoming and prediction data is also stored in the configured Blob Storage area for archival purposes.
5. The prediction and real-time streaming data is visualized using pre-configured Power BI dashboards easily accessible via desktop web browser, tablet or mobile devices.
6. Email/ SMS notifications are achieved using Azure Logic Apps.

4.2 Detailed System Design

4.2.1 IoT Data Metrics

The table below illustrates the data metrics collected from the Waspmote Plug&Sense sensor unit:

s/n	Data	Description
1	Temperature	Measures the hotness/coldness of the water (Degrees Celsius)
2	pH Level	Measures the acidity/ alkalinity of the water (scale from 0 - 14 with neutral point of 7)
3	Dissolved Oxygen (DO)	Measures the amount of gaseous oxygen (O2) dissolved in the water (mg/L)
4	Oxygen Reduction Potential (ORP)	Measures the water's oxidizing power and its potential ability to sanitize itself (mV)
5	Conductivity	Measures the ionic strength of the water and its ability to conduct electricity (µS/cm)

Table 1 – IoT Data Metrics

4.2.2 Information Flow

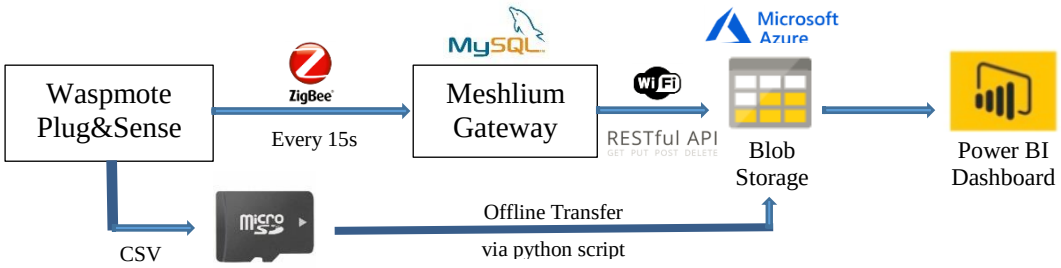


Figure 5 – Information Flow Diagram

The sensor data is periodically sent (every 15s) from each Waspmote Plug&Sense sensor unit to the Meshlium gateway via ZigBee communication. To accommodate network communication loss scenarios (either between the sensor unit and Meshlium Gateway via ZigBee communication or between Meshlium Gateway to Microsoft Azure via wireless communication), the sensor data is also stored in the micro-SD card inserted in each sensor unit in comma-separated variable (CSV) format. This data can be transferred offline later to Microsoft Azure using custom python script (see Appendix A.2).

Figure 6 – Offline Transfer from microSD storage CSV file to Microsoft Azure Blob Storage

The Meshlium gateway sends received data to Microsoft Azure using RESTful API messages where the data is stored in Blob storage. Machine learning techniques are performed on the collected data for analysis purposes to perform classification of the various water types.

Data is visualized using dashboards created in the Microsoft Power BI platform.

4.2.3 Waspmote Plug&Sense Unit Programming/Configuration

The:Waspmote Plug&Sense unit is programmed using the Waspmote IDE as provided by the manufacturer. It is essentially a customized Arduino Sketch IDE and thus adopts similar programming conventions (C-based).

The following functions are used to program the Waspmote Plug&Sense unit:

s/n	Function	Purpose	Requirement
1	SW 08: Frame Class (v12)	Used for reading sensor values	http://www.libelium.com/v12/development/waspmote/examples/sw-08-frame-class-v12/
2	ZB 03: Send packets	Used for sending sensor output values to meshlium gateway	http://www.libelium.com/v12/development/waspmote/examples/zb-03-send-packets/
3	USB 01: USB functions	Used for sending sensor output values to serial terminal for monitoring	http://www.libelium.com/v12/development/waspmote/examples/usb-01-usb-functions/
4	SD 07: Datalogger	Used for saving sensor output values to SD memory card	http://www.libelium.com/v12/development/waspmote/examples/sd-07-datalogger/
5	RTC 01: Setting and reading time	Used for getting Real-time Clock information	http://www.libelium.com/v12/development/waspmote/examples/rtc-01-setting-and-reading-time/
6	Power 04: Getting battery level	Used for getting Waspmote Battery Level	http://www.libelium.com/v12/development/waspmote/examples/power-04-getting-battery-level/
7	Power 02: Deep sleep mode	Used for putting Waspmote in deep sleep mode in between sensor	http://www.libelium.com/v12/development/waspmote/examples/power-02-deep-sleep-mode/

s/n	Function	Purpose	Requirement
		polling	
8	UT 07: Formatted strings	Used for formatting sensor output value string	http://www.libelium.com/v12/development/waspmote/examples/ut-07-formatted-strings/

Table 2 – Waspmote Function List

In order to conserve power, the Waspmote Plug&Sense unit is put in deep sleep mode in between the polling cycle of 15 seconds.

Refer to Appendix A-1 for the full program source code.

4.2.4 Meshlium Gateway Configuration Details

The Meshlium Gateway acts as the IoT Gateway that aggregates sensor data received via ZigBee communication from the various Waspmote Plug&Sense units and transmits this to the Azure IoT Plaform via web-based REST API calls in JSON format for data storage/ processing. A Libelium-Azure development kit [5] is available from the manufacturer to perform the necessary configuration. The gateway contains a web-based configuration interface (Meshlium Manager System) accessed via http://192.168.1.106/ManagerSystem/ from a web browser. The figure below shows the details used to connect to the "sw-ph" event hub configured in the Microsoft Azure platform.

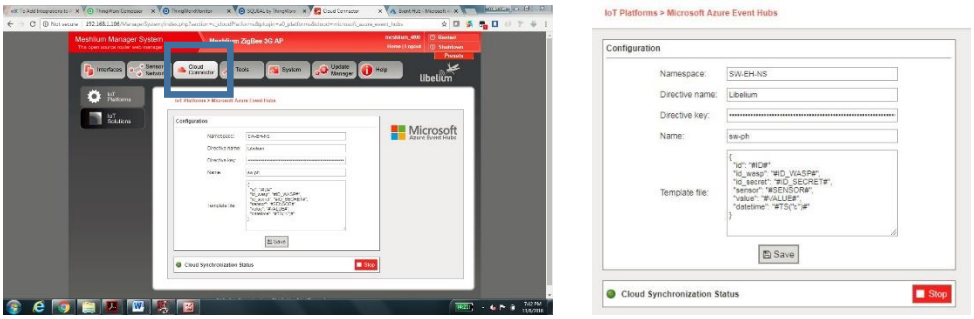


Figure 7 – Meshlium Gateway – Azure Configuration Details

The sensor data is transmitted at 15-second intervals from the Meshlium gateway to the Microsoft Azure platform for storage and processing.

4.2.5 IoT Platform Selection/ Implementation Details

4.2.5.1 Selection

In selecting a suitable IoT platform, our team shortlisted the following platforms for consideration: Amazon AWS and Microsoft Azure. The selection criteria used to decide on the final choice are as follows:

1. Industrial Adoption

   Based on the IoT Developer Survey conducted by the Eclipse Foundation [6], the top two IoT platforms that are gaining wider adoption over the past three years are Amazon AWS and Microsoft Azure (see figure below).

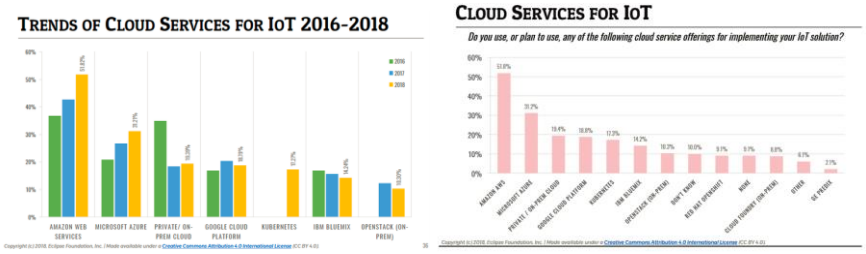


Figure 8 – IoT Cloud Platform Adoption Rate

2. Costs – there is no upfront licensing costs involved for access, which is favourable for prototyping type projects where proof of concept has yet to be established. Between Microsoft Azure and Amazon AWS, a study [8] found that AWS costs more as per-message charging was based on 1KB message size as compared to Azure which is 4KB i.e. 8KB message will be treated as 8 messages in AWS and 2 in Azure.

Based on the above considerations, Microsoft Azure is selected for this project.

4.2.5.2 Implementation

4.2.5.2.1 Data Ingestion/ Visualization

Incoming sensor data from the Meshlium gateway is received by an Azure Event Hub (***sw-ph***), an event ingestion service. Azure Stream Analytic Job (***SW-SA***) is used to process the event hub messages for data storage using Azure Blob Storage (***smartwaterstorage***) and transmission to the PowerBI service as streaming datasets for real-time dashboard visualization. Stream Analytic Jobs (***SW-SA-2*** and ***SW-SA-3***) are used for water type classification

(by calling Azure function ***SW_PredictWaterType***) and for equipment anomaly detection (by calling Azure function ***SW_AnomalyDetectio***n) respectively. Output messages from ***SW-SA-2*** and ***SW-SA-3*** are processed by Azure Logic Apps (***SW-LA*** and ***SW-LA-2***) for SMS/email notifications using built-in connectors for Google Mail and Twilio SMS services.

The table below lists the various entities developed for ingestion/ processing of sensor data from the Libelium gateway:

s/n	Entity	Description
Event Hub		
1	SW-EH-NS	Container namespace for event hubs
2	sw-ph	Data storage buffer for sensor data
3	notificationhub	Data storage buffer for SMS/email services
4	notificationhub2	Data storage buffer for SMS/email services
Stream Analytics Job		
1	SW-SA	• Stores incoming sensor data in ***smartwaterstorage/Data*** blob storage • Sends streaming data to Power BI
2	SW-SA-2	• Triggers azure function ***SW_PredictWaterType*** for classification prediction • Stores data in ***smartwaterstorage/Predictions*** blob storage • Sends data to ***notificationhub2*** event hub for SMS/email notification by ***SW-LA-2*** logic app
3	SW-SA-3	• Triggers azure function ***SW_AnomalyDetection*** for anomaly detection • Stores data in ***smartwaterstorage/Predictions*** blob storage • Sends detection data to ***notificationhub*** event hub for SMS/email notification by ***SW-LA*** logic app
Blob Storage		
1	smartwaterstorage	Container Storage for Data/Predictions
2	Data	Stores incoming sensor data
3	Predictions	Stores classification/ anomaly detection prediction data
Logic App		
1	SW-LA	Sends SMS/email notification for equipment anomaly events
2	SW-LA-2	Sends SMS/email notification for sensor parameter anomaly events
Power BI		
1	North/ South/ East/West	Dashboards for displaying streaming sensor real-time/prediction data

Table 3 – Azure Data Ingestion/ Processing/ Notification resources

4.2.5.2.2 Data Analytics/ Machine Learning

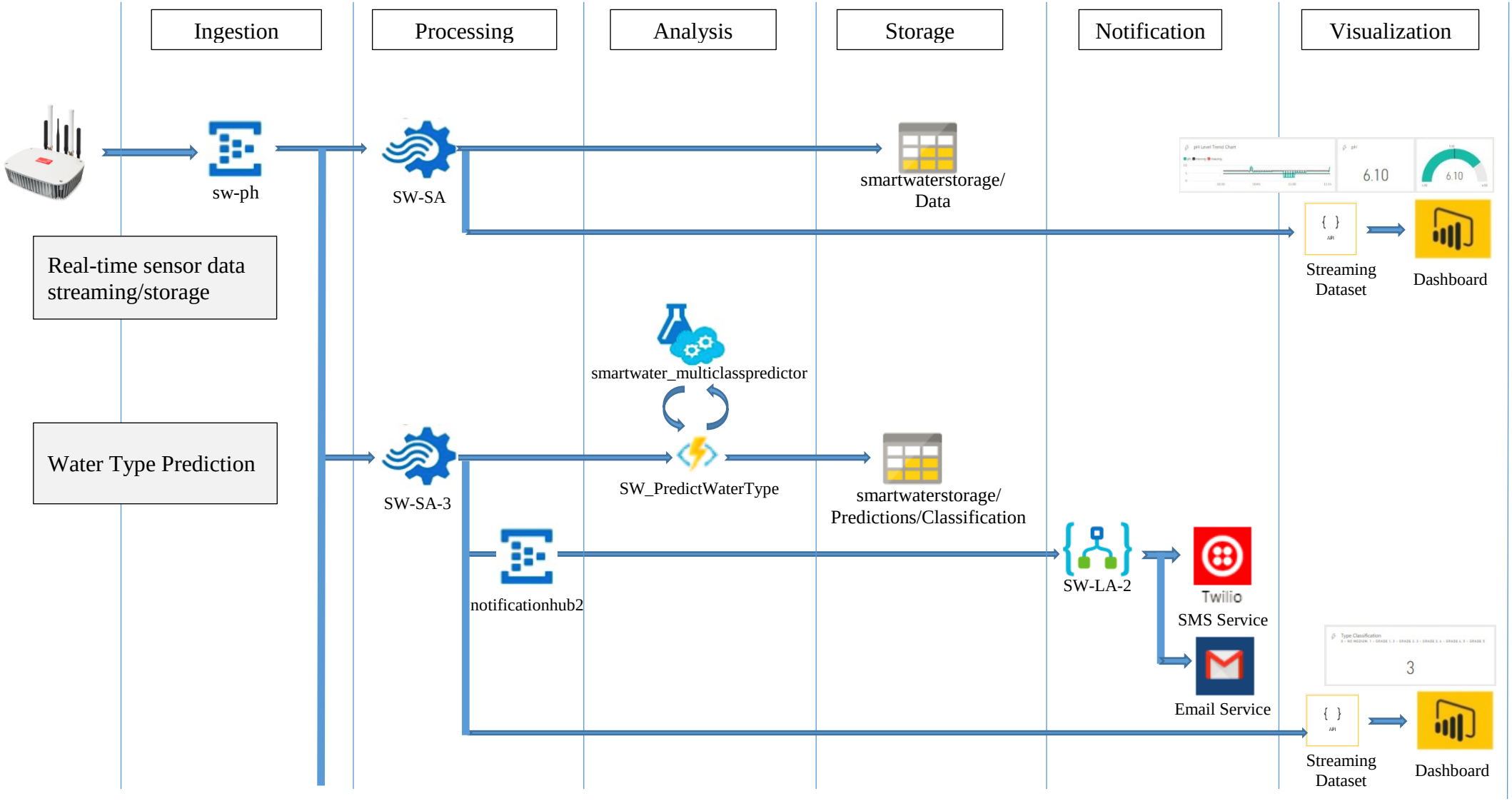
Azure Machine Learning Studio was used for data exploratory analysis and to build, test, and deploy the water type classification and equipment anomaly detection trained models as web services. Azure Machine Learning Studio experiments were used to evaluate the Azure multi-class classification (Multiclass Decision Forest, Multiclass Decision Jungle, Multiclass Logistic Regression, One-vs-All Multiclass and Multiclass Neural Network) and anomaly detection (One-Class Support Vector Machine, PCA-Based Anomaly Detection) algorithms, while Azure-hosted Jupyter Notebooks were used to evaluate the Scikit-Learn algorithms. Below is a list of the Machine Learning Studio resources (refer to Appendix A-2 for screenshots):

s/n	Entity	Description
Experiments		
1	SmartWaterTest_MultiClass	Used for evaluating model performance on dataset using Azure-specific multi-class algorithms
2	SmartWaterTest_MultiClass_CV	Used for evaluating model performance on dataset using Azure-specific multi-class algorithms with cross-validation
3	SmartWaterTest_AnomalyDetection	Used for evaluating model performance on dataset using Azure-specific anomaly detection algorithms
4	SmartWater_MultiClassAnalysis/ SmartWater_MultiClassPredictor	Used for building multi-class learning model using Decision Forest algorithm and deployment as ***SmartWater_MultiClassPredictor*** web service
5	SmartWater_AnomalyDetection/ SmartWater_AnomalyDetector	Used for building anomaly detection model using One-Class Support Vector Machine algorithm and deployment as ***SmartWater_AnomalyDetector*** web service
Notebooks		
1	SmartWater_EDA_ MulticlassClassificationAnalysis	Used for dataset exploratory analysis and evaluating model performance on dataset using Scikit-Learn multi-class classification algorithms
2	SmartWater_AnomalyDetectionAnalysis	Used for evaluating model performance on dataset using Scikit-Learn anomaly detection algorithms
3	Smart Water_ClusteringAnalysis	Used for evaluating model performance on dataset using Scikit-Learn clustering algorithms
4	SmartWater_Deploy	Used for building multi-class classification model using Random Forest algorithm and deployment as

s/n	Entity	Description
		SmartWater_MultiClassPredictor_2 web service Used for building anomaly detection model using OneClassSVM algorithm and deployment as ***SmartWater_AnomalyDetector_2*** web service
Web Services		
1	SmartWater_MultiClassPredictor	Deployed web service for Water Type Multi-Class Classification using Multiclass Decision Forest algorithm
2	SmartWater_AnomalyDetector	Deployed web service for Equipment Anomaly Detection using One-Class Support Vector Machine algorithm
3	SmartWater_MultiClassPredictor_2	Deployed web service for Water Type Multi-Class Classification using Random Forest algorithm
4	SmartWater_AnomalyDetector_2	Deployed web service for Equipment Anomaly Detection using OneClassSVM algorithm

Table 4 – Azure Analytics/ Machine Learning resources

The figure below illustrates the flow of information between the various configured entities from Meshlium Gateway to Azure/ PowerBI:



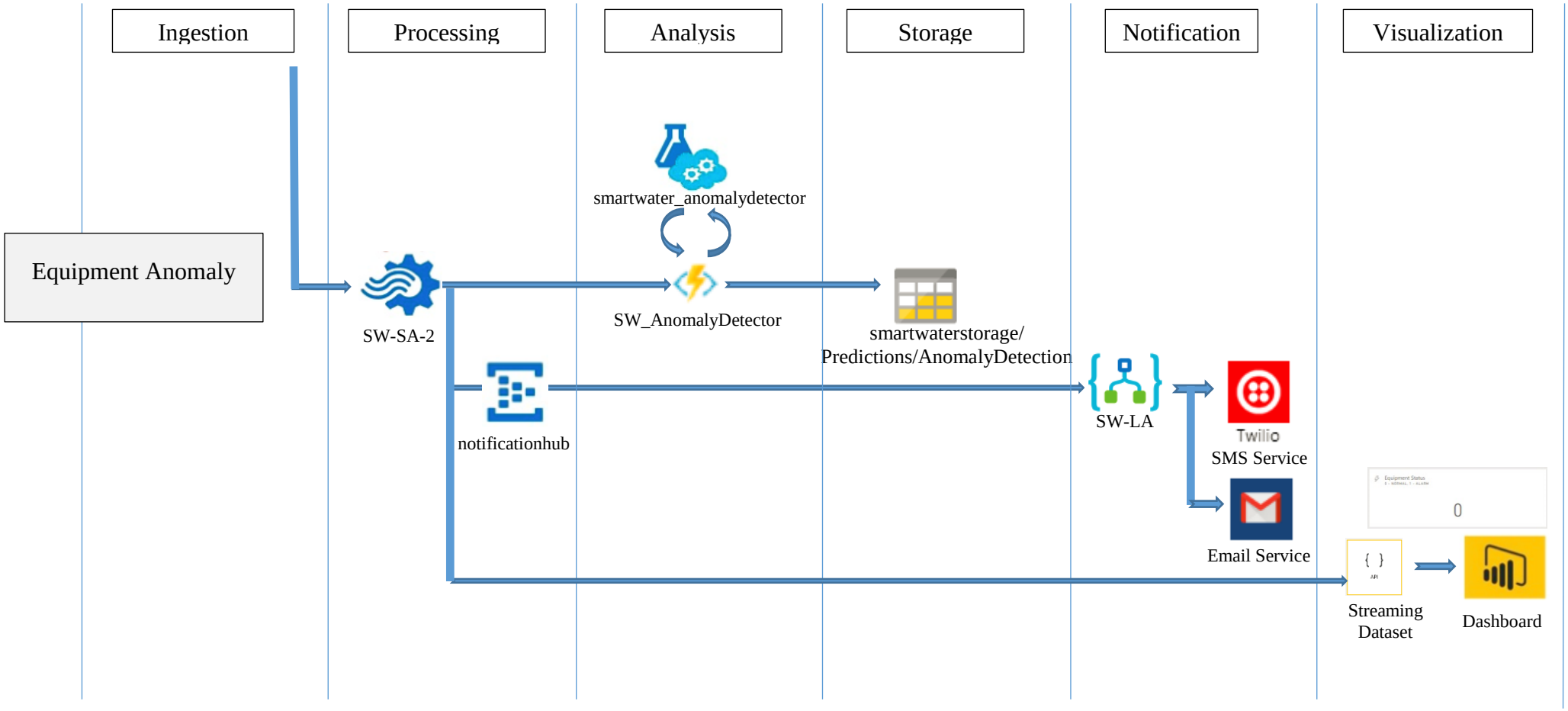


Figure 9 –Data Flow Diagram

Refer to Appendix A-3 for the Azure configuration screens.

4.2.5.3 Capacity Limit Analysis

The table below summarizes the system capacity requirements as compared to the Microsoft Azure platform limits:

s/n	Item	Requirement	Azure Limitation
1	Ingress Throughput	1 event per 15s i.e. 0.067events/s	1000events/s
2	Event size	1 event = 5kb	256kB
3	Storage Capacity	The Meshlium gateway transmits sensor data at 15-second intervals from each Waspmote Plug&Sense unit i.e. 5760 events per day (4 events/min x 60mins x 24 hours). Each sensor node requires 5760 x 5kB/event = 28.8MB/daily or 10.512GB/year.	4.77TB

Table 5 – Capacity Limit Analysis

From the table above, it is observed that the Azure platform can comfortably accommodate the ingress, storage and event data size requirements from each sensor unit.

4.2.6 Data Visualization

4.2.6.1 Power BI Dashboard

To visualize the real-time sensor values streamed from the Waspmote Plug&Sense sensor units, dashboards are created in Microsoft Power BI. This dashboard provides the following functions:

1. Displays the real-time values of each sensor parameter reading,
2. Provides historical 1-hour trend value displays for each sensor parameter reading.
3. Visual Gauge showing the current value for each sensor type in relation to its normal operating range.
4. Analytic outputs:
    - Water type classification
    - Equipment fault status indicator – 1 (Alarm), 0 (Normal),

- Overall system status indicator (alert category) – 1 (Abnormal), 0 (Normal)

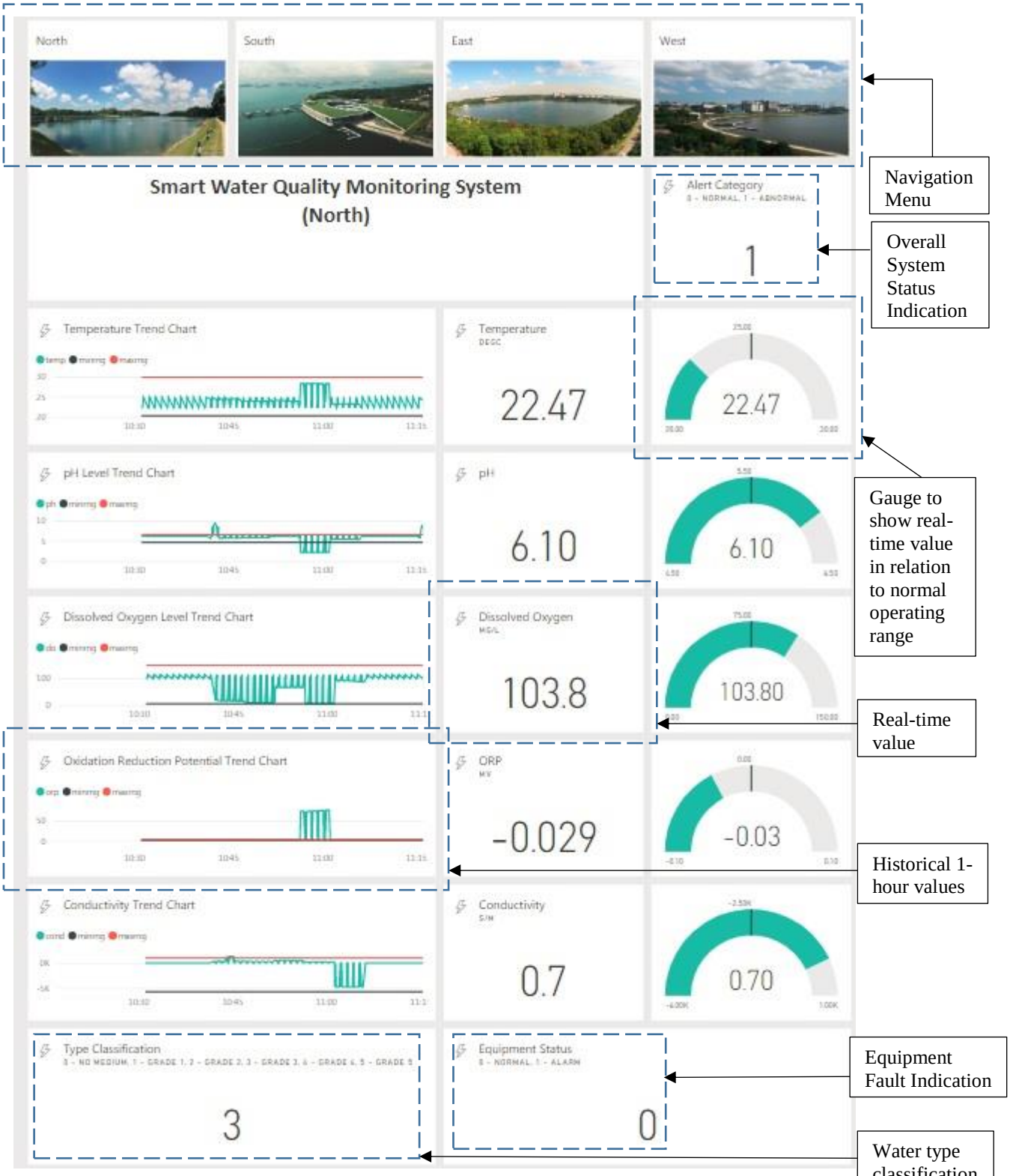


Figure 10 – PowerBI Dashboard - Web

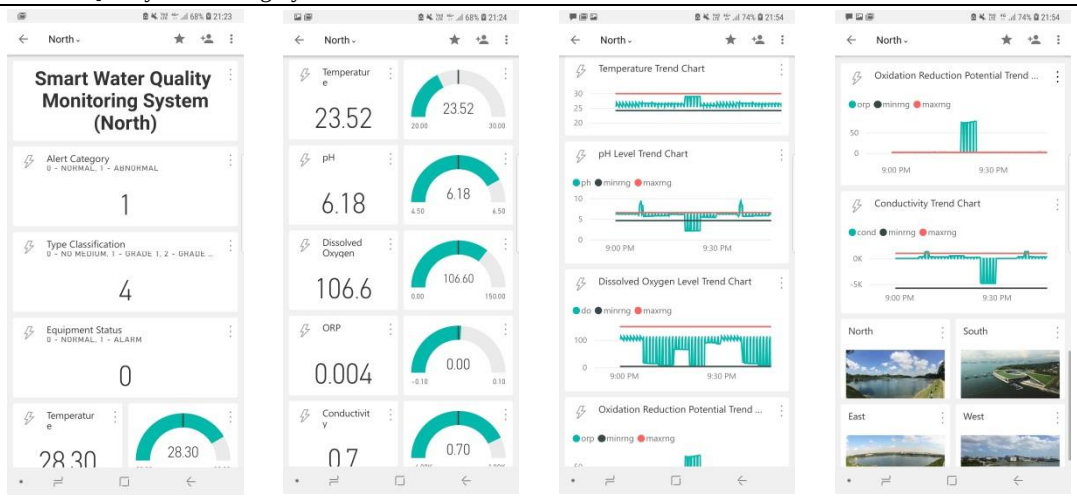


Figure 11 – PowerBI Dashboard - Mobile

On mobile devices e.g. smartphone, tablet etc., the same dashboard information is available as on desktop and web, by installing the PowerBI app available from the Google PlayStore (for Android devices) and App Store (for Apple iOS devices).

4.2.6.2 Notification Services

In addition to the dashboard display, the system also provides SMS/ email notifications about anomalous events occurring:

- Sensor parameter value out of normal operating range
- Equipment alarm e.g. dislodged from mounting location, battery low etc.

In

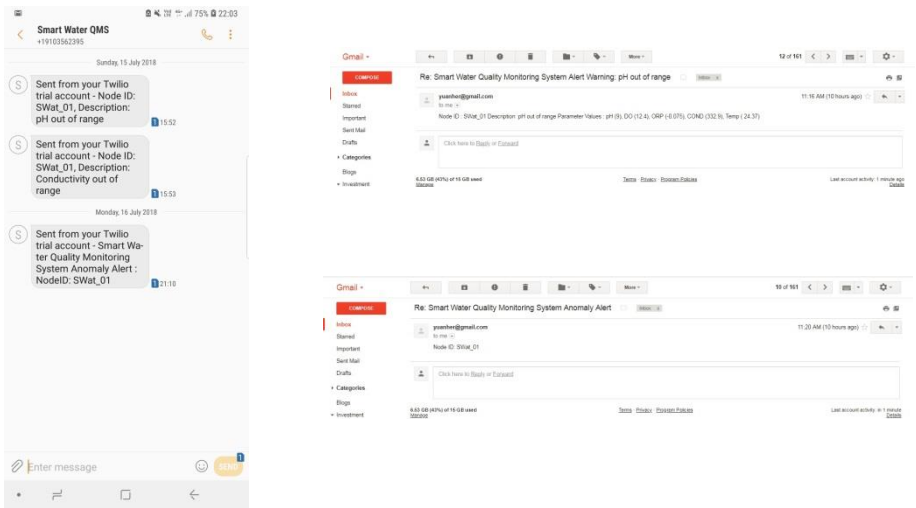


Figure 12 – SMS/ Email Notifications

4.3 Experimental Setup

4.3.1 Introduction

To simulate different water quality conditions, various media are used as listed below:

s/n	Media	Description	Remark
1	Tap Water	-	Grade 1
2	Soap		Grade 2
3	Flour		Grade 3
4	Shampoo		Grade 4
5	Sea Water		- Grade 5

Table 6 – Material Type Classification

Data is collected for one-hour periods for each type of media used.

4.3.2 Test Setup

The figure below shows the test equipment setup for data collection:

Figure 13 – Test Setup for Data Collection

The 5 sensor probes (temperature, pH, DO, ORP, conductivity) are immersed in a plastic container filled with water mixed with the media identified in the table above. Sensor data collected are transmitted from the Waspmote Plug&Sense unit to be processed by the Azure platform via the Meshlium Gateway.

4.3.3 Data Analytics

4.3.3.1 Problem Definition

4.3.3.1.1 Introduction

There are two main problem statements (Operational/Maintenance) as follows:

1. Operational - Develop a machine learning model which could be used to classify the water quality from which the process operator/ engineer can then use this information to tune the parameters used in the subsequent water treatment process.
2. Maintenance - Generate notifications about anomalous conditions associated with the Waspmote Plug&Sense unit so that relevant corrective actions can be taken by the maintenance personnel

The sensor data recorded by the Waspmote Plug&Sense unit are labelled time-series data and appropriate labels are applied to the sensor data based on the media used during the data collection. With the labelled data, multi-class classification (supervised learning) and Clustering (unsupervised learning) techniques are used to analyze the sensor data and to classify the water type e.g. Grade 1, 2 etc.

The figure below illustrates the process as described above:

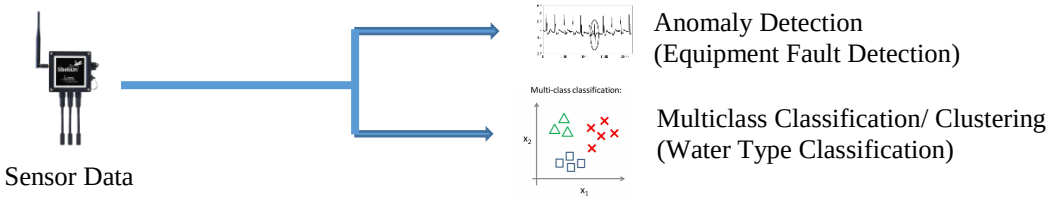


Figure 14 – Data Classification Process

4.3.3.2 Data Summarization

4.3.3.2.1 Define Dataset

The table below summarizes the source of the data used for the subsequent machine learning model development/testing:

1	NoMedium.csv	4	Flour_Water,csv
2	Tap_Water.csv	5	Shampoo_Water.csv
3	Soap_Water,csv	6	Sea_Water.csv

Table 7 – Classification Datasets

Data from each collection run will be combined into a single dataset for model training and validation.

4.3.3.2.2 Descriptive Statistics

The dataset consists of sensor data recorded by the Waspmote Plug&Sense unit.

The following is a listing of the features of the dataset:

```
for col in df_all.columns:
    print(df_all[col].name, ':\t', df_all[col].dtype)
```

```
ACC_X :  int32
ACC_Y :  int32
ACC_Z :  int32
BAT :    int32
WT :     float64
PH :     float64
DO :     float64
ORP :    float64
COND :   float64
class :  int32
```

Dataset consists of 9 features and 1 output label

Figure 15 – Dataset Structure

Below shows an extract from the dataset:

sensor	ACC_X	ACC_Y	ACC_Z	BAT	WT	PH	DO	ORP	COND	class
0	-2	-5	1023	7	26.42	5.51	138.5	0.011	0.7	0
1	0	-4	1020	7	26.42	5.46	138.1	0.013	0.7	0
2	-5	-8	1023	7	26.42	5.45	138.1	0.016	0.7	0
3	-6	-5	1019	7	26.41	5.43	138.1	0.017	0.7	0
4	-5	-4	1019	7	26.41	5.46	137.7	0.017	0.7	0

Figure 16 – Dataset Extract

4.3.3.2.3 Uni- & Multi-variate Plots

4.3.3.2.3.1 Line Plots

Below are line plots of each water type in the dataset:

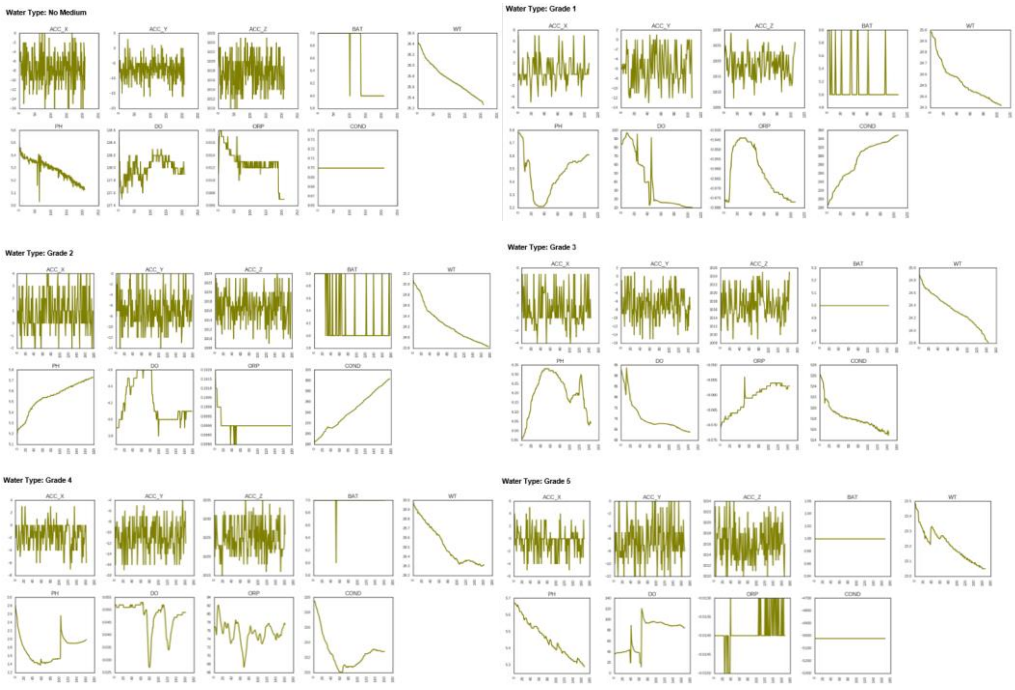


Figure 17 – Line Plots

From the line plots above, it can be observed that most features exhibit quite different data profiles over the collection period.

4.3.3.2.3.2 Histogram Plots

Below are histogram plots of the features in the dataset:

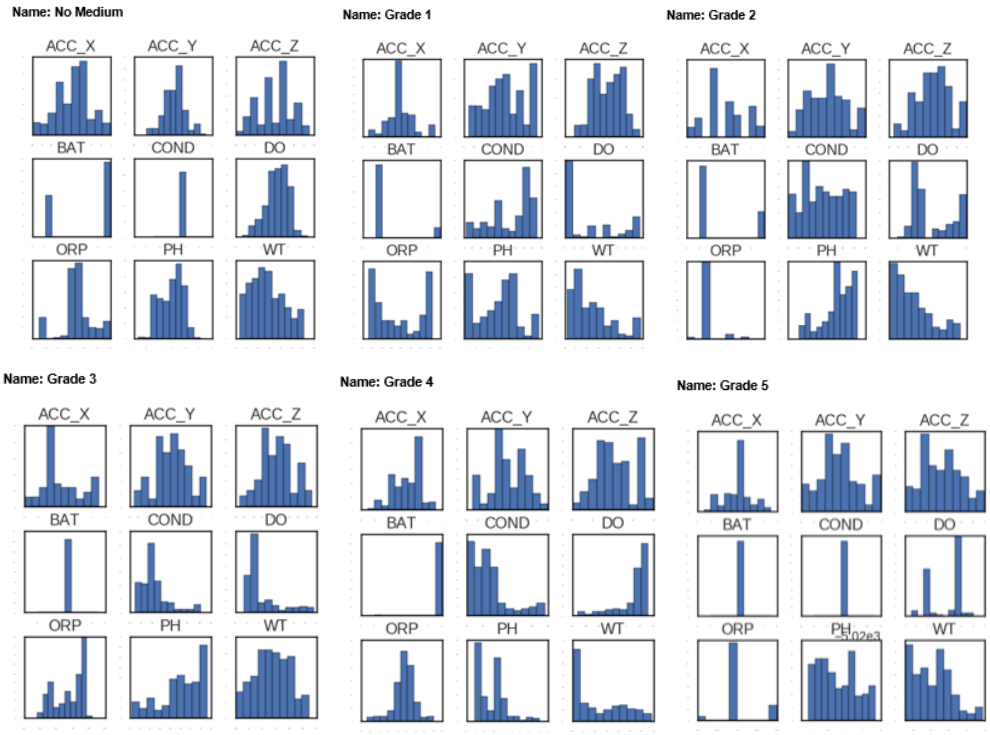


Figure 18 – Histogram Plots

From the histogram plots above, it can be observed that some features e.g. “ACC_X”, “ACC_Y”, “ACC_Z” exhibit normal distribution profile regardless of water type.

4.3.3.2.3.3 Box Plots

Below are histogram plots of each feature for all water types in the dataset:

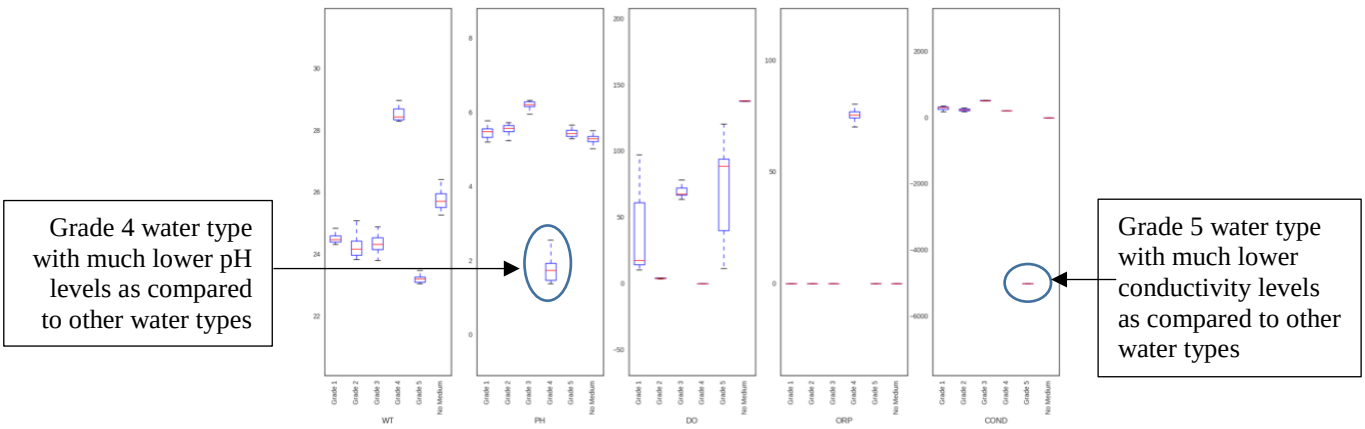


Figure 19 – Box Plots

From the figures above, the following observations can be recorded:

- Grade 4 water type has much lower pH levels and higher ORP levels.
- Grade 5 water type has much lower conductivity levels.

4.3.3.2.3.4 Class Distribution Profile

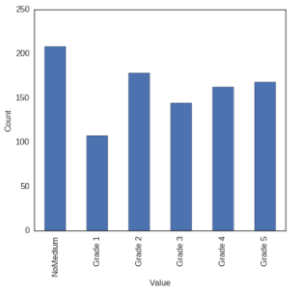


Figure 20 – Water Type Distribution Profile

From the figure above, it is observed that each water type contains sufficient data samples to form a representative proportion of the entire dataset.

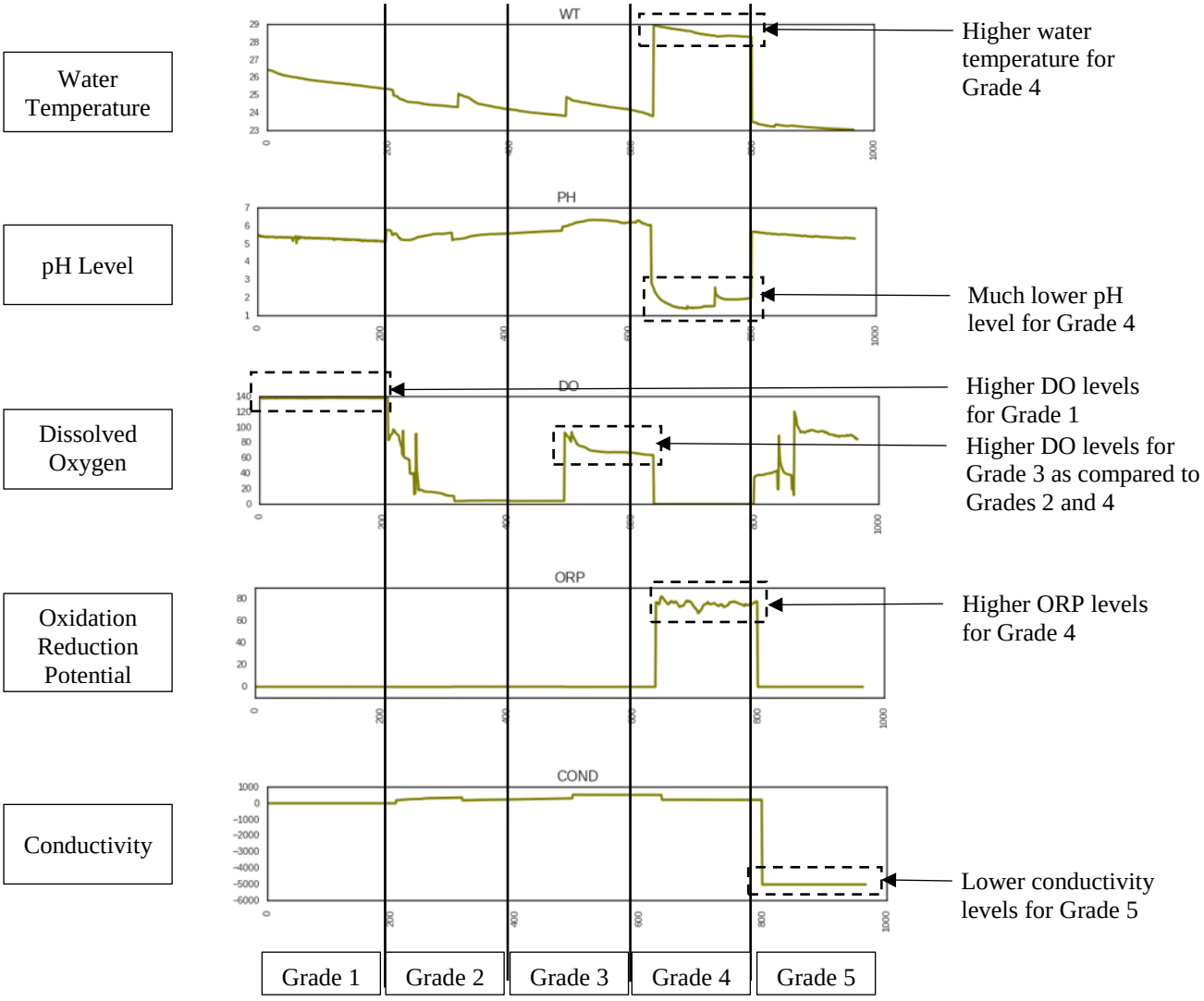


Figure 21 – Line Plot of combined dataset by feature

From the figure above, it can be observed that each water type has unique characteristics that could be used for identification and classification.

4.3.3.2.4 Correlation

Next, we check the correlation between the dataset features:

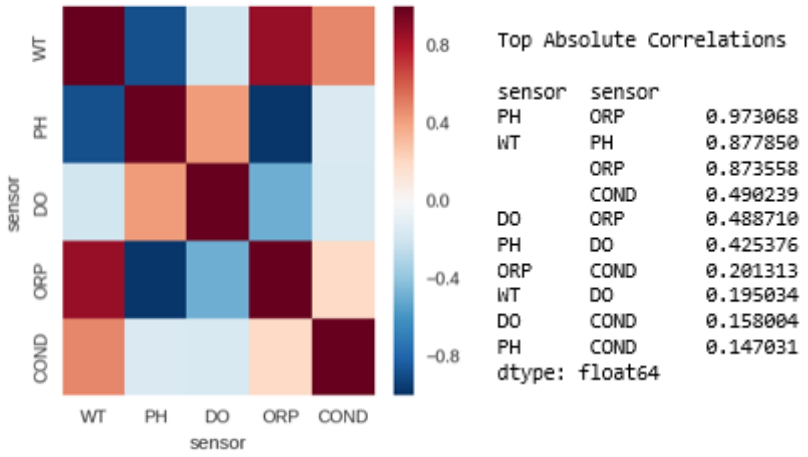


Figure 22 – Correlation Matrix & Top absolute correlations

From the figure above, it can be observed that some features exhibit high correlation with each other e.g. pH and ORP/WT.

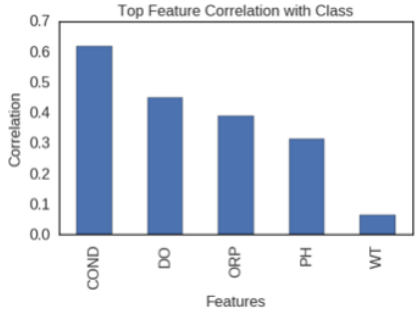


Figure 23 – Top Feature Correlation with class attribute

From the figure above, it can be observed that conductivity (COND) has the highest correlation with the class attribute, followed by Dissolved Oxygen (DO) and Oxidation Reduction Potential (ORP).

4.3.4 Machine Learning

4.3.4.1 Clustering

4.3.4.1.1 Select Optimal K

The raw time series data collected are analysed using unsupervised machine learning techniques for automatic cluster identification. In all, 8 clustering techniques were evaluated to select the most suitable to be used.

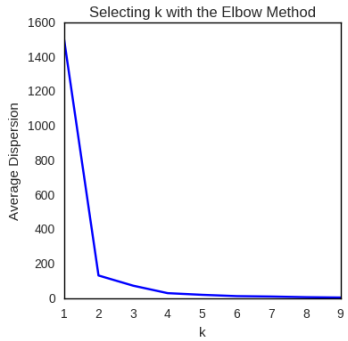
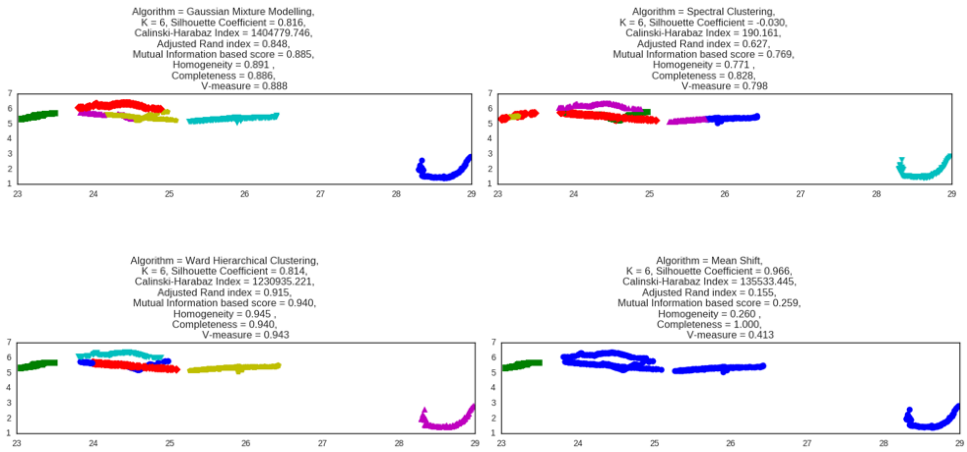


Figure 24 – Number of clusters K selection using Elbow Method

To determine the number of clusters to be used, the elbow method is used. From the figure above, the optimal value for K where the improvements (centroids are closest to the clusters centroids) decline most rapidly is K=5. This is used to evaluate the various clustering algorithms used.

4.3.4.1.2 Evaluate Algorithm Performance

The figure below shows the scatter plot of Water Temperature Vs pH Level showing cluster identification using different learning algorithms



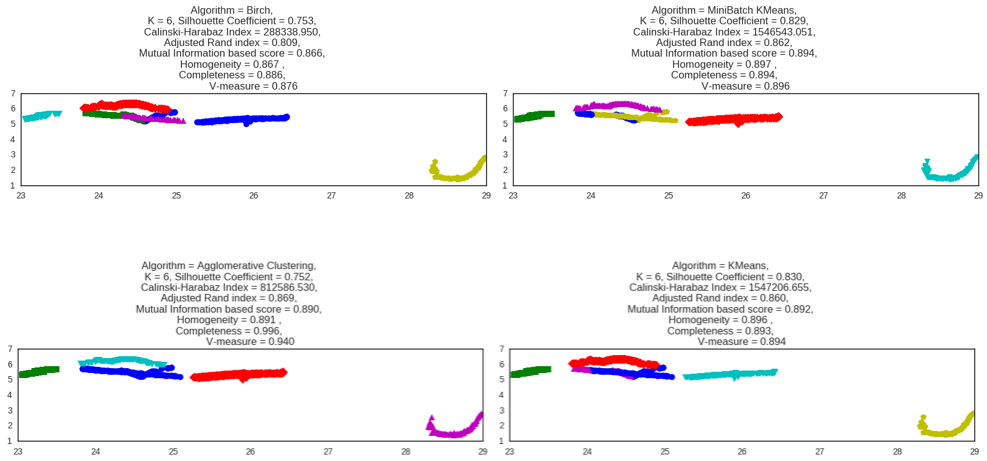


Figure 25 –Data Classification Algorithm Evaluation from scatter plots

As the ground truth labels are known in the "class" attribute, the figures below compare the actual class information with the clusters identified using the different clustering algorithms:

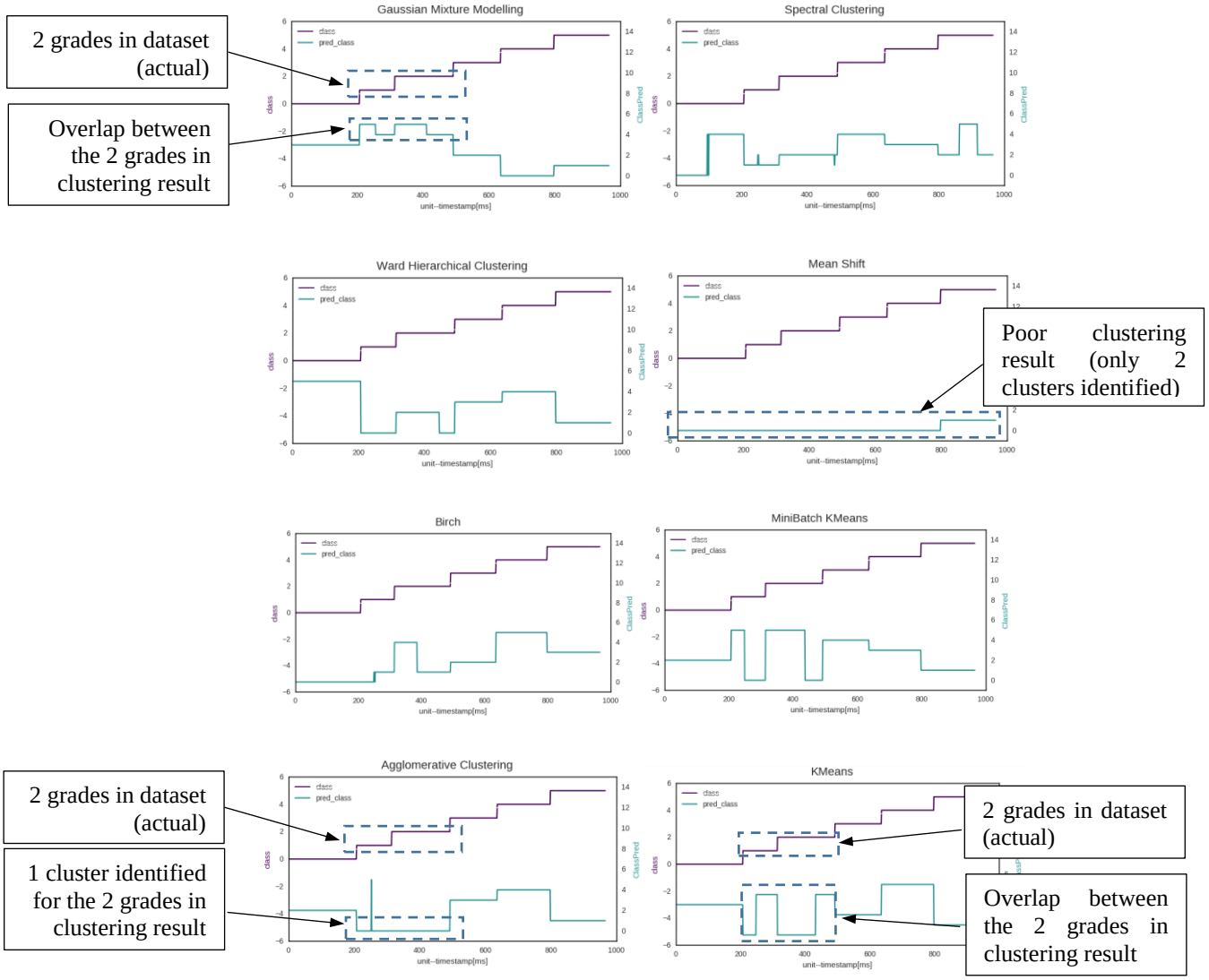


Figure 26 – Ground Truth Labels ("class") vs Identified Clusters Comparison

From the figure above, it can be observed that the identified clusters using Ward Hierarchical Clustering algorithm performs the best amongst all compared except for Grade 2 wherein there is slight overlap with Grade 1 i.e. mis-identified as Grade 1.

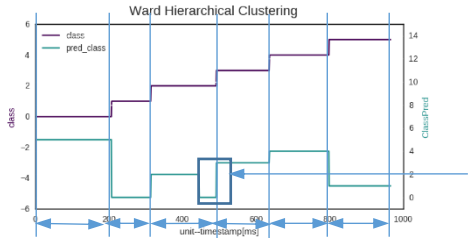


Figure 27 – Ward Hierarchical Clustering Actual Class vs Identified Clusters Comparison

The performance of the clustering results is measured using the following evaluation metrics (number closer to 1 indicates better clustering result):

1. Adjusted Rand Score
2. Adjusted Mutual Information Score
3. Homogeneity/Completeness/V-Measure scores

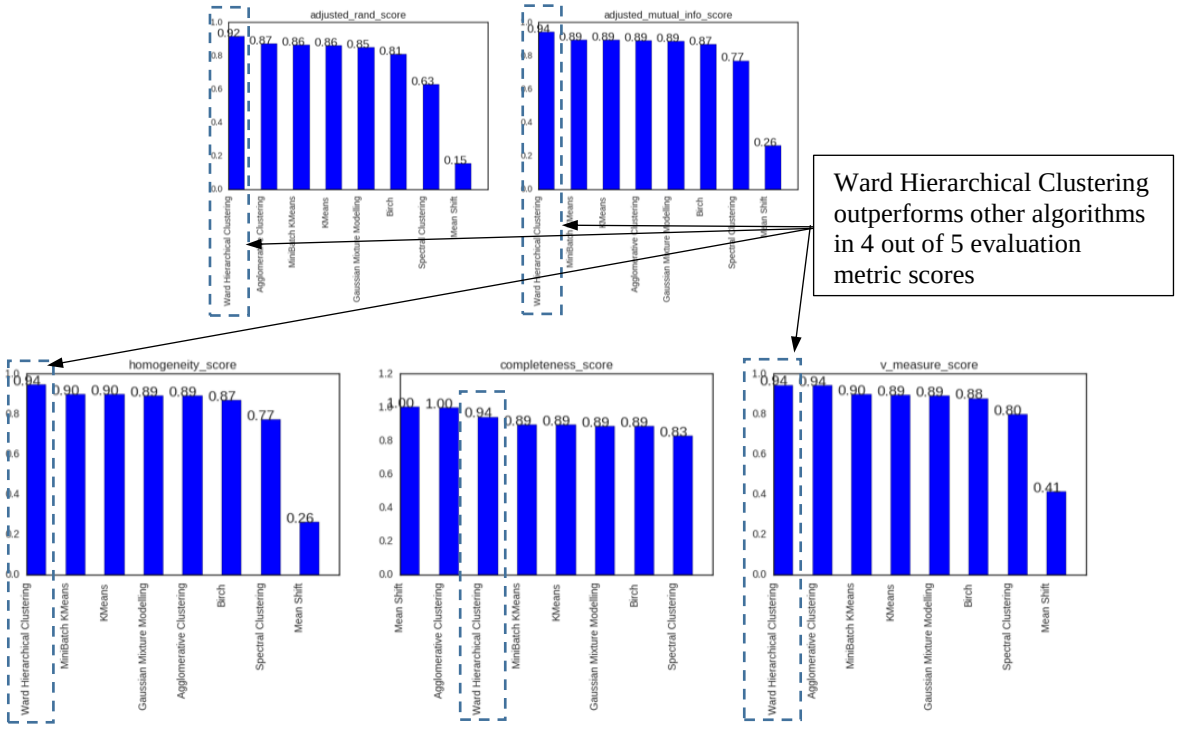


Figure 28 –Evaluation Criteria

From the figure above, it can be observed that Ward Hierarchical Clustering algorithm performs the best in 4 out of 5 metric scores, thus making it suitable for data classification.

4.3.4.2 Multi-Class Classification

4.3.4.2.1 Feature Selection

Feature selection is used to select features that contribute most to the prediction variable to improve accuracy, reduce overfitting and reduces training time. In this project, 3 automatic feature selection techniques are used, namely Reduced Feature Elimination (RFE), Model Feature Importance and Factor analysis. The results are illustrated as below:

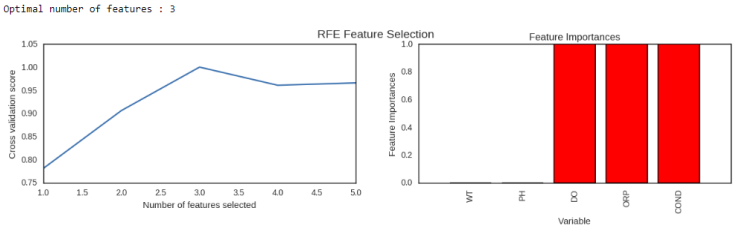


Figure 29 – Reduced Feature Elimination results

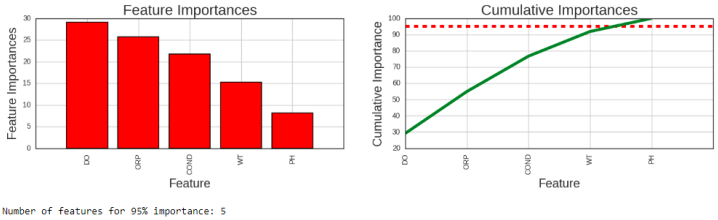


Figure 30 – Model Feature Importance results

	WT	PH	DO	ORP	COND
0	-0.8498	0.2151	8.3927	-5.6838	-1995.6224
1	-0.4055	0.7676	51.8879	-16.3169	0.2669
2	1.4047	-1.1852	7.2845	22.2898	-0.0383
3	0.1875	-0.1530	-0.0159	-0.1494	-0.0003
4	0.0000	0.0000	-0.0000	0.0000	-0.0000

```
Number of important features:  5
```

Figure 31 Factor Analysis results

From the figures above, it can be observed that all 5 features contribute to the variability in the dataset (from Factor Analysis results), contribute to >95% feature importance (from Model Feature Importance results) and the 3 most important features are DO, ORP, COND (from Reduced Feature Elimination and Model Feature Importance results).

4.3.4.2.2 Training/Validation Dataset and Evaluation Metric

80% of the data from the dataset is used to train the learning model and the remaining 20% of data is used to validate the developed model. Accuracy metric is used to evaluate the performance of the model.

4.3.4.2.3 Algorithm Performance Check

We evaluate 11 classifier algorithms (6 ensemble classifiers and 5 Azure machine learning classifiers) using default tuning parameters with the following accuracy, precision and recall classification scores:

Name	Accuracy	Precision	Recall
ExtraTrees	100.0000	1.0000	1.0000
RandomForest	100.0000	1.0000	1.0000
GradientBoosting	100.0000	1.0000	1.0000
Votting	100.0000	1.0000	1.0000
Neural Network	100.0000	1.0000	1.0000
Decision Jungle	100.0000	1.0000	1.0000
Decision Forest	100.0000	1.0000	1.0000
Bagging	99.4845	0.9958	0.9907
SVM (One vs All)	98.6183	0.9691	0.9420
Logistic Regression	97.0639	0.9408	0.8768
AdaBoost	90.7216	0.7807	0.8333

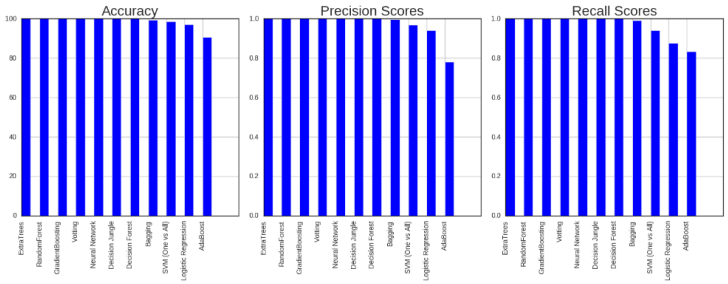


Figure 32 –Algorithm Spot Check Results

From the results above, it can be observed that except for SVM (One vs All), Bagging, Logistic Regression and AdaBoost classifiers, the rest of the classifiers were able to achieve very good performance.

4.3.4.2.4 Cross-Validation Check

Next, we evaluate the effect of performing cross-validation on the performance of the 11 classifier algorithms and obtained the following performance classification scores:

Name	Accuracy	Logloss	Precision	Recall
ExtraTrees	100.0000	-0.0014	1.0000	1.0000
RandomForest	100.0000	-0.0004	1.0000	1.0000
GradientBoosting	100.0000	-0.0007	1.0000	1.0000
Votting	100.0000	-0.0201	1.0000	1.0000
DecisionForest	100.0000	0.0125	0.9988	0.9988
DecisionJungle	100.0000	0.0061	0.9966	0.9962
LogisticRegression	100.0000	0.0515	1.0000	1.0000
NeuralNetwork	100.0000	0.0361	1.0000	1.0000
OneVsAll	98.9659	0.1447	0.9826	0.9733
Bagging	98.4515	-0.0879	0.9891	0.9763
AdaBoost	88.4882	-0.3575	0.7693	0.8333

Logistic Regression	97.0639	0.9408	0.8768

Original result
(without cross validation)

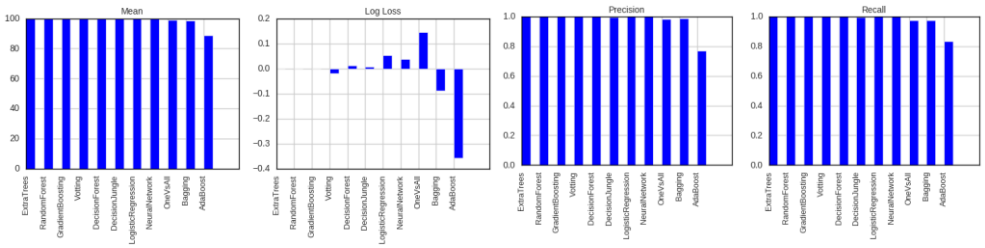


Figure 33 –Algorithm Comparison on Transformed Data Results

From the results above, it is observed that using cross-validation improves the performance of Logistic Regression classifier.

4.3.4.2.5 Estimate accuracy on validation dataset

Next, we estimate the accuracy classification score metric of the Random Forest Classifier using transformed data (standardization, normalization, scaling and binarization) on the validation dataset and obtained the following accuracy classification scores:

```
Accuracy Estimation of RandomForest on validation dataset
Standardized: 0.90206185567
Normalized: 1.0
Rescaled: 1.0
Binarized: 0.536082474227
```

Figure 34 –Validation Dataset accuracy estimation results

From the results above, it is observed that the Random Forest Classifier produces the best score with a normalized or rescaled copy of the training dataset.

4.3.4.2.6 Algorithm Tuning

Next, hyperparameter tuning was performed on the Random Forest Classifier by using grid search using a set of parameter values, with each value evaluated using 10-fold cross-validation on a standardized copy of the training dataset:

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
 'max_features': [4],
 'min_samples_leaf': [1, 3, 5],
 'min_samples_split': [2, 4, 6],
 'n_estimators': [100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]}
```

Figure 35 –Grid Search parameters on Random Forest Classifier

```
Best Params: {'min_samples_split': 2, 'bootstrap': True, 'min_samples_leaf': 1, 'n_estimators': 100, 'max_features': 4, 'max_de
pth': 10}
Best Score: 1.0
Best Estimator: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=10, max_features=4, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=100, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

Figure 36 –Algorithm Tuning on Random Forest Classifier Results

From the results above, the best value for "max_depth" parameter for the Random Forest Classifier is 10, with 100% accuracy score.

4.3.4.2.7 Deployment

The AzureML python client library and Azure Machine Learning Experiment is used to deploy the trained Random Forest and Decision Forest models respectively as machine learning web services for water type classification.

The table below lists the machine learning web services deployed for water type classification:

s/n	Name	Description
1	SmartWater_MultiClassPredictor	Web service deployed from trained Azure Decision Forest model
2	SmartWater_MultiClassPredictor_2	Web service deployed from trained Scikit-Learn Random Forest model

Table 8 – Machine Learning Web Services for Water Type Classification

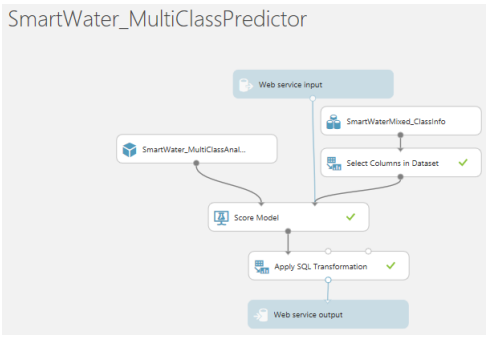


Figure 37 – SmartWater_MultiClassPredictor Deployment

Figure 38 –SmartWater_MultiClassPredictor_2 Deployment

4.3.4.2.8 Summary

In this section, various learning algorithms were evaluated to perform multi-class classification on the sensor data. Random Forest Classifier and Decision Forest Classifier were found to be suitable for water type classification. Tuning was performed using Grid Search to select the optimal combination of parameters to produce the best accuracy estimation result.

4.3.4.3 Anomaly Detection

4.3.4.3.1 Introduction

To detect any equipment abnormal events, anomaly detection is performed on the accelerometer data. 5 anomaly detection algorithms are evaluated, 3 from Scikit-Learn library (OneClassSVM, Isolation Forest and Robust Covariance) and 2 Azure Specific (One-Class Support Vector Machine and PCA-Based Anomaly Detection). For the Scikit-Learn algorithms, Azure hosted Jupyter Notebook is used to perform the analysis whereas for the Azure specific ones, Azure machine learning experiment is used to perform the analysis.

4.3.4.3.2 Multivariate Plot

The data profile plot of the Accelerometer data is shown below:

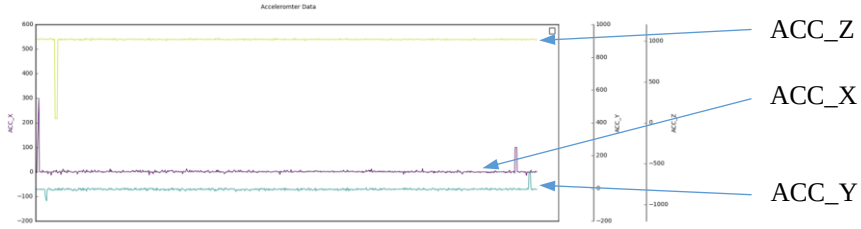


Figure 39 –Accelerometer Data Profile

4.3.4.3.3 Algorithm Evaluation

The algorithm evaluation results are shown below:

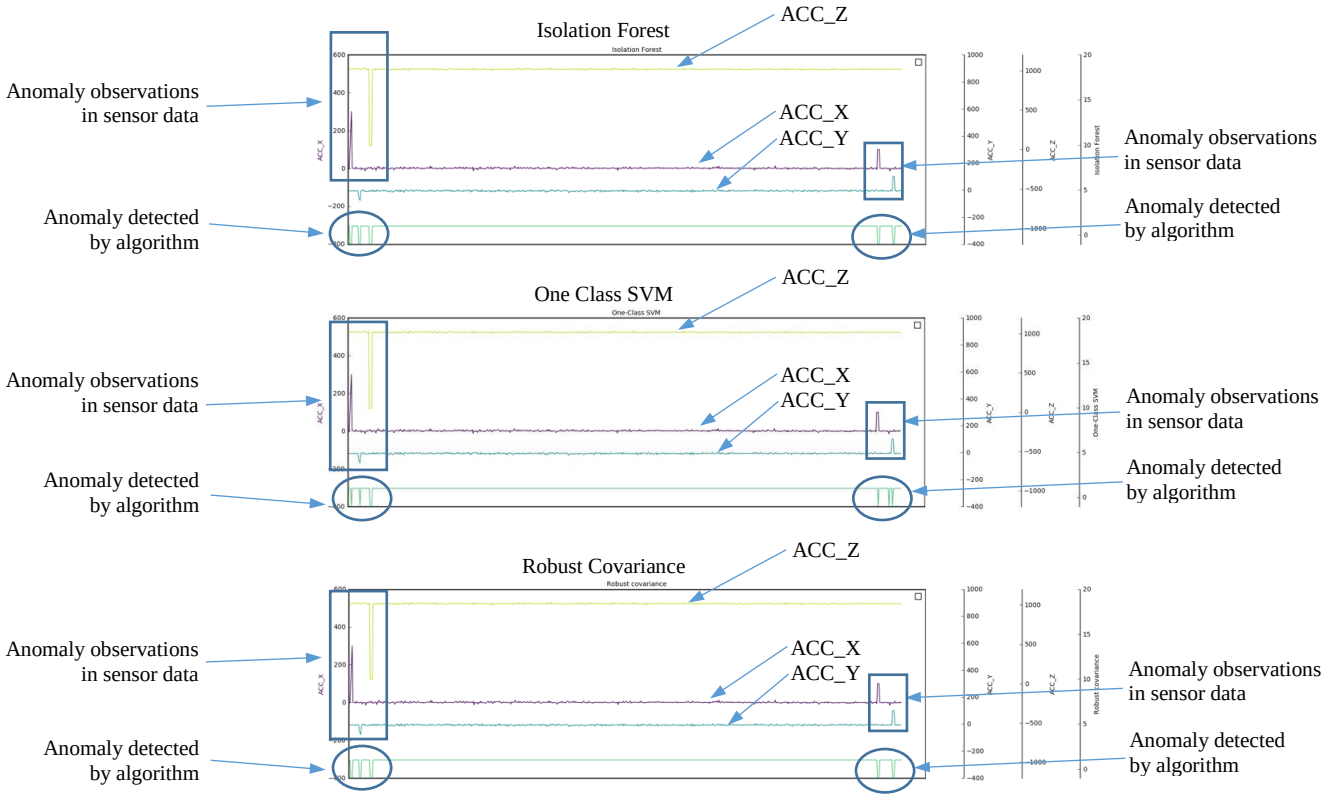


Figure 40 –Anomaly Detection Algorithm Evaluation Results

From the figure above, it can be observed that all 3 Scikit-Learn algorithms are able to detect the anomaly occurrence points accurately.

4.3.4.3.4 Deployment

The AzureML python client library and Azure Machine Learning Experiment is used to deploy the trained OneClassSVM and One-Class Support Vector Machine models respectively as machine learning web services for equipment

anomaly detection. The table below lists the machine learning web services deployed for equipment anomaly detection:

s/n	Name	Description
1	SmartWater_AnomalyDetector	Web service deployed from trained Azure One-Class Support Vector Machine model
2	SmartWater_ AnomalyDetector _2	Web service deployed from trained Scikit-Learn OneClassSVM model

Table 9 – Machine Learning Web Services for Equipment Anomaly Detection

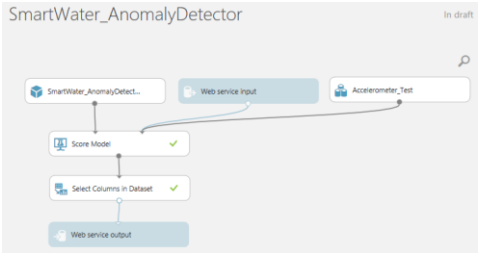


Figure 41 – SmartWater_AnomalyDetector Deployment

Figure 42 –SmartWater_AnomalyDetector _2 Deployment

4.3.4.3.5 Summary

In this section, various learning algorithms were evaluated to perform anomaly detection on the accelerometer data. OneClassSVM and One-Class Support Vector Machine algorithms were found to be suitable and deployed as machine learning web services.

5 System Testing

5.1 Functional Testing

The following lists the functional tests conducted to verify the functional requirements of the Smart Water Quality Monitoring System as identified in section 2.2.

Test Specification ID : TS1 Name of Tester : Lim Yuan Her / Lim Hong Loon Use Case ID : UC1 Date of Test : 29 July 2018 Description of Test : Verify system functional requirements				
S/No	**Test Case**	**Expected Result**	**Pass/ Fail**	**Remarks**
Functional Requirement 1 (Section 2.2.1)				
1.	Check dashboards availability	Four dashboards are available and designated as North, South, East and West from desktop web browser and mobile device (smartphone)	Pass	
2.	Check water quality parameters displayed on visualization dashboard	Dashboard displays the pH, Dissolved Oxygen, Oxidation Reduction Potential, Conductivity and temperature for each sensor unit deployment location	Pass	
3.	Check dashboards on desktop/web and mobile devices	Dashboard on mobile device display the same information as that on desktop/web devices	Pass	
Functional Requirement 2 (Section 2.2.2)				
1.	Simulate water quality parameter e.g. pH out of normal operating range	SMS and email notification messages received for abnormal water quality parameter detection	Pass	
2.	Simulate equipment fault condition	SMS and email notification messages received for equipment fault status	Pass	

Functional Requirement 3 (Section 2.2.3)				
1.	Simulate different water type condition	Dashboard displays the classified water type e.g. Grade 1, 2 etc. accordingly.	Pass	
2.	Simulate at least one abnormal water quality parameter i.e. out of normal operating range	Dashboard displays the correct overall system alert status i.e. 1 for abnormal system condition	Pass	

6 User and Technical Documentations

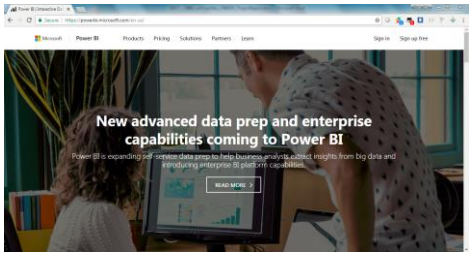
6.1 User Documentation/Guide/Manual

6.1.1 Introduction

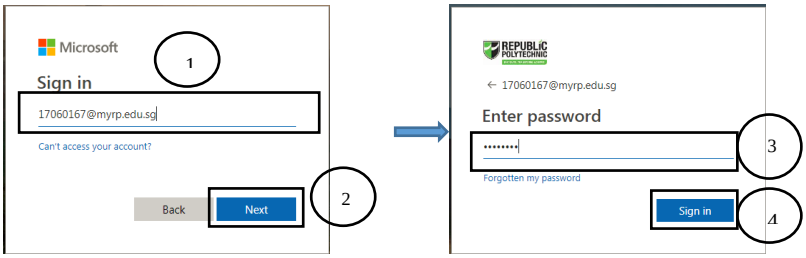
This section details the steps to launch the program/app for accessing the dashboards. The user should have basic knowledge of using a web browser before proceeding.

6.1.2 Desktop/Web Access

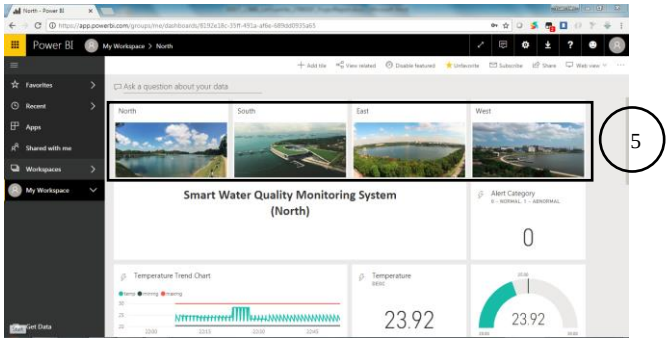
1. Launch the internet web browser and navigate to http://www.powerbi.com



2. Click the “Login” link on the top right corner of the screen and log in using designated username and password:

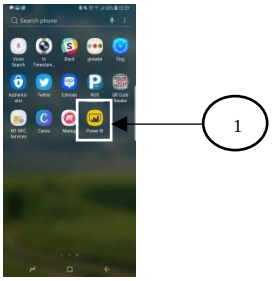


3. The PowerBI Dashboard will be displayed. Click on a dashboard icon to navigate to the desired dashboard.

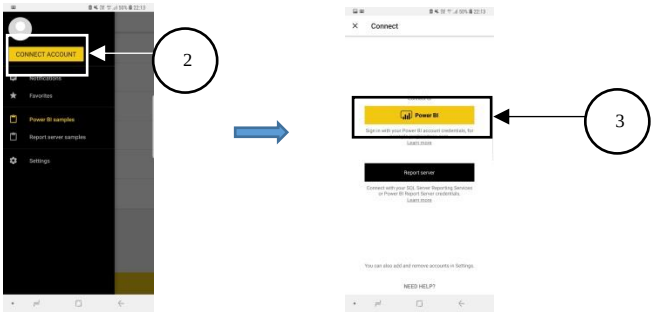


6.1.3 Mobile Access

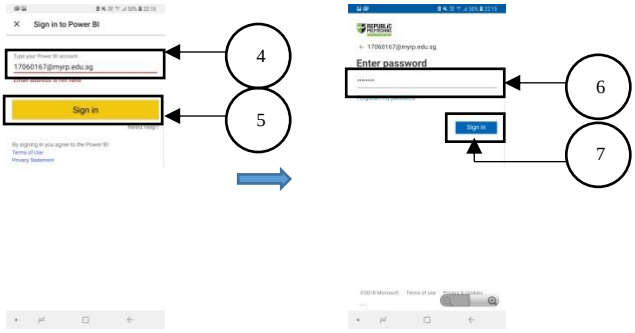
1. Click on the PowerBI app icon.



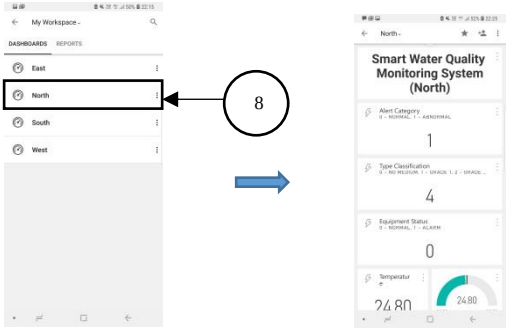
2. Click on “CONNECT ACCOUNT” button, and on the “Connect” screen, click o the “Power BI” button.



3. Enter username on the “Sign in to Power BI” screen, and click “Sign In”. Key in your password in the “Enter password” screen and click “Sign In”.



4. The “Workspace” screen will be displayed, showing the list of dashboards available for viewing. Click on a dashboard selection from the list to view.



6.2 Technical Documentation (Installation guide/Manual)

6.2.1 Introduction

This section details the installation steps to install the required program/ app to access the dashboards.

6.2.2 Desktop/Tablet Access

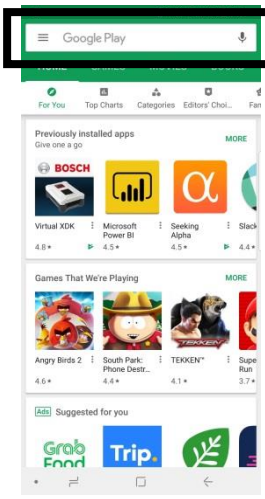
Basic web browser e.g. Internet Explorer is installed by default in desktop/ tablet. No additional installation is required.

6.2.3 Mobile App Installation (PowerBI)

Follow the steps below to install the PowerBI App from the Google Playstore:

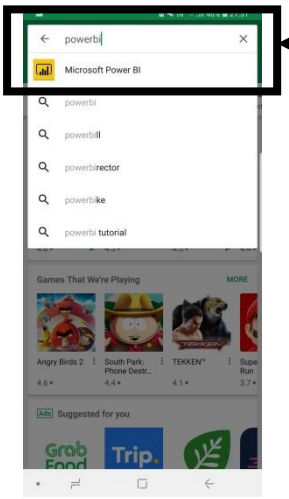
1

Click on the Google Playstore



2

Click in the Search Box



3

Type "PowerBI" in the search box

Click "Microsoft Power BI" from the search results

4

Click "Install" to install the app on the mobile device

6.2.4 Meshlium Gateway Setup

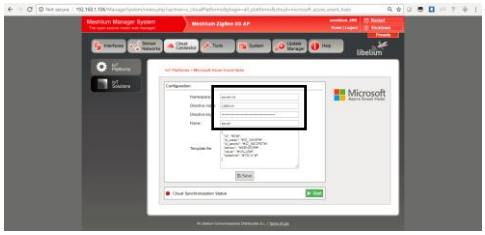
Follow the steps below to configure the Meshlium gateway's connection to Microsoft Azure:

a) Navigate to http://192.168.1.106/ManagerSystem from any web browser.

b) Click the "Cloud Connector" button.

c) Click the "Microsoft Azure Event Hubs' button.

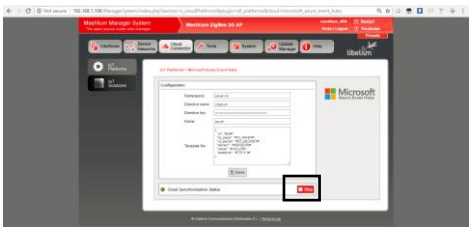
d) Enter the Azure connection details in the entry fields as per the table below:



s/n	Field	Value	Remarks
1	Namespace	sw-eh-ns	Event Hub Namespace
2	Directive Name	Libelium	Event Hub Shared Access Key Name
3	Directive Key	011wja2uBU9UsbvHZpBnfR/ pX64Sh3S+/iyZ17FlkAc=	Event Hub Shared Access Key
4	Name	sw-ph	Event Hub Name

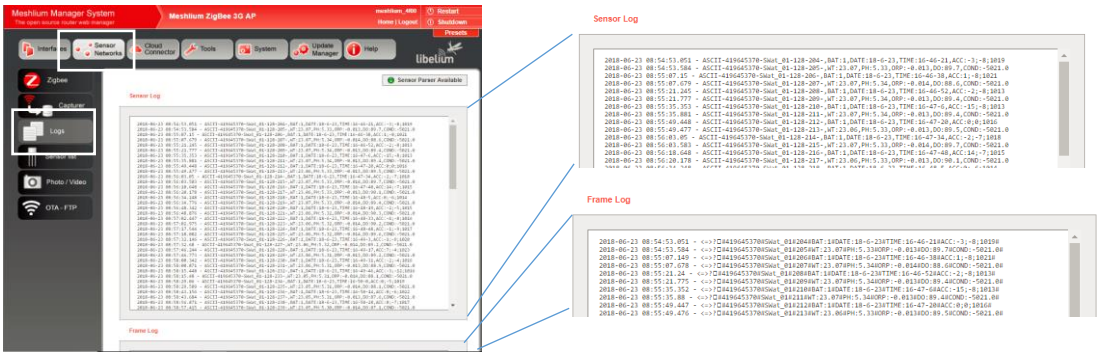
Table 10 – Meshlium Gateway Cloud Connector Configuration

e) Click the “Start” Button to initiate the gateway synchronization to Azure.

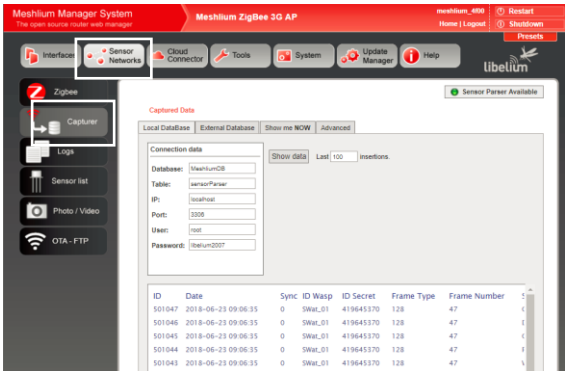


The “Start” button text will change to “Stop”, indicating the synchronization process was started successfully.

f) Check the Meshlium gateway is receiving sensor data from the Waspmote Plug&Sense units from the “*Sensor Networks -> Logs*” page:



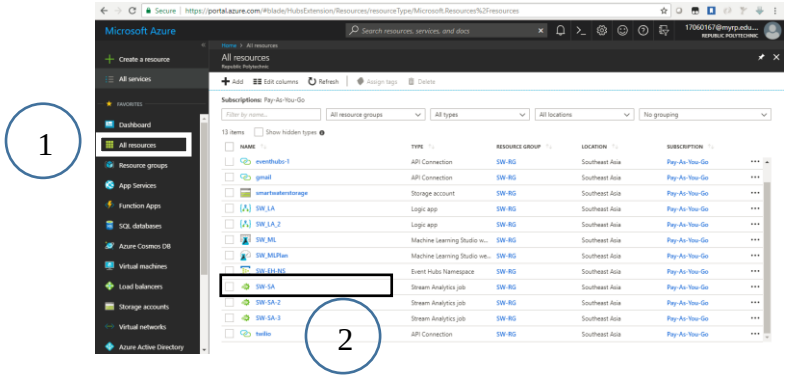
g) Check the Meshlium gateway local database (MySQL) is storing the received sensor data from the Waspmote Plug&Sense units from the “*Sensor Networks -> Capturer*” page:



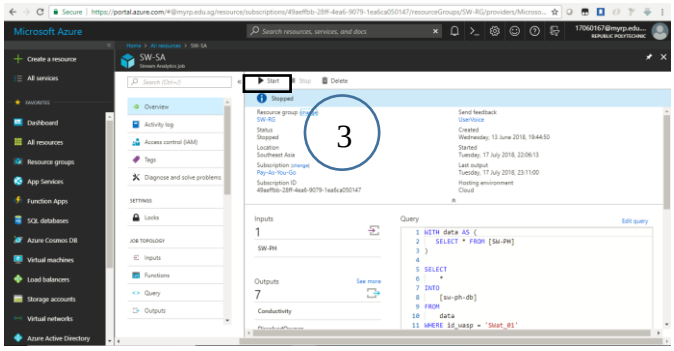
6.2.5 Azure Setup

Follow the steps below to start the required resources in Microsoft Azure:

a) Navigate to “All Resources” from the left navigation bar.



b) Select the “SW-SA” stream analytics job from the list. The “SW-SA” configuration screen will be displayed.



c) Click the “Start” button to initiate the stream analytics job.

d) Repeat steps (b) and (c) to initiate the “SW-SA-2” and “SW-SA-3” stream analytics jobs.

7 Conclusions

7.1 Introduction

This report detailed the design and implementation results of a Smart Water Quality Monitoring System using the Libelium Smart Water Kit using Microsoft Azure as the IoT platform for data processing/storage. Microsoft PowerBI visualization dashboards used to display the sensor data are accessed via desktop web browsers and mobile devices (smartphone/ tablet). Analytic capabilities for water type classification and equipment fault detection are built into the system using machine learning techniques. Alerting mechanism for water quality anomaly/ equipment fault detection are achieved using SMS/email notification.

7.2 Further Enhancements

Some further system enhancements that could be considered for future work are as listed below:

- Currently, the Azure Event Hub Connector is used to connect the Meshlium Gateway to the Microsoft Azure platform. Using the Azure IoT Hub Connector instead will provide additional support for Cloud-to-Device messaging and enhanced security (individual device connection authentication as opposed to device collective Shared Access Keys).

- Extend visualization capabilities using chatbots to provide a natural language interface for querying sensor data information on mobile devices/ social media platforms.

- Use solar power to power the outdoor installed Libelium Waspmote Plug&Sense unit by using solar panels or portable USB solar power bank e.g. Vander life 300000mAh Portable Solar Power Bank (https://shopee.sg/Vander-life-Waterproof-300000mAh-Portable-Solar-Charger-Dual-USB-Battery-Power-Bank-i.10134820.1223248070) to extend the unit operating time.

8 References

- [1] Irvine, Kim, Lloyd Chua and Hans S. Eikass (2014). *The Four National Taps of Singapore - A Holistic Approach to Water Resources Management from Drainage to Drinking Water*. Journal of Water Management Modelling, 2014.
- [2] Public Utilities Board, Singapore. *Treatment and Water Quality Monitoring Process from Reservoir to Tap,* https://www.pub.gov.sg/Documents/advisory_2015Oct01.pdf
- [3] *Global Market Insights: Singapore Water Treatment & Wastewater Recycling Systems,* https://www.export.gov/article?id=Global-Market-Insights-Singapore-Water-Treatment-Wastewater-Recycling-Systems
- [4] *S$200 million funding boost for Singapore's water industry over the next five years,* https://www.gov.sg/~/sgpcmedia/media_releases/pub/press_release/P-20160711-2/attachment/Water%20Industry%20-%20RIE2020%20-%20Press%20Release.pdf
- [5] Libelium-Azure Development Kit Quick Start Guide
- [6] *Developers favoring AWS, Microsoft Azure for cloud IoT platforms,* https://www.zdnet.com/article/developers-favoring-aws-microsoft-azure-for-cloud-iot-platforms/
- [7] *IoT Platform Selection: Azure vs. AWS vs. GCP,* https://dzone.com/articles/iot-platform-selection-azure-vs-aws-vs-gcp-compari
- [8] *What Cloud Platform Should You Pick for IoT? A Head to Head Between AWS and Azure,* https://medium.com/the-why-and-how/what-cloud-platform-should-you-pick-for-iot-a-head-to-head-between-aws-and-azure-43cf8918fea6
- [9] *PTC Thingworx picks Microsoft's Azure as preferred IIOT cloud,* http://www.ioti.com/industrial-iot-iiot/ptc-thingworx-picks-microsofts-azure-preferred-iiot-cloud

9 Appendices

9.1 A.1 - Source code for Waspmote Plug&Sense unit

```
#include <WaspSensorSW.h>
#include <WaspGPS.h>
#include <WaspXBeeZB.h>
#include <WaspFrame.h>

long  sequenceNumber = 0;
char* sleepTime = "00:00:00:05";
char data[100];

// define buffer to store the message
char message[100];
// define local buffer for float to string conversion
char Temp_str[10];
char pH_str[10];
char ORP_str[10];
char DO_str[10];
char COND_str[10];
char ACC_str[20];

int accelerometerX;
int accelerometerY;
int accelerometerZ;

// define folder and file to store data
char path[]="/data";
char filename[]="/data/log";

// define variable
uint8_t sd_answer;

//unsigned long timestamp;
//timestamp_t time;

bool  gpsStatus;

int   batteryLevel;

// Destination MAC address
char RX_ADDRESS[] = "0013A20041030D54";

uint8_t error;

float valuePH;
float valueTemp;
float valuePHCalculated;
float valueORP;
float valueORPCalculated;
float valueDO;
float valueDOCalculated;
float valueCond;
float valueCondCalculated;

// pH Sensor Calibration values
#define CAL_POINT_10 1.985
#define CAL_POINT_7 2.070
#define CAL_POINT_4 2.227
// Temperature at which pH calibration was carried out
#define CAL_TEMP 23.7
// Offset obtained from ORP sensor calibration
#define CALIBRATION_OFFSET 0.0
// Calibration of the DO sensor in normal air
#define AIR_CALIBRATION 2.65
// Calibration of the DO sensor under 0% solution
#define ZERO_CALIBRATION 0.0
// Value 1 used to calibrate the Conductivity sensor
#define POINT1_COND 10500
// Value 2 used to calibrate the Conductivity sensor
```

```
#define POINT2_COND 40000
// Point 1 of the Conductivity calibration
#define POINT1_CAL 197.00
// Point 2 of the Conductivity calibration
#define POINT2_CAL 150.00

char nodeID[] = "SWat_01";

pHClass pHSensor;
ORPClass ORPSensor;
DOClass DOSensor;
conductivityClass ConductivitySensor;
pt1000Class TemperatureSensor;

void setup()
{
  // init USB port
  USB.ON();
  //USB.println(F("Sending packets example"));

  // Set the Waspmote ID
  frame.setID(nodeID);

  // Switch on the board
  SensorSW.ON();

  // init XBee
  xbeeZB.ON();

  // Set SD ON
  SD.ON();

  // create path
  sd_answer = SD.mkdir(path);

  // Create file for Waspmote Frames
  sd_answer = SD.create(filename);

  delay(1000);

  ////////////////////////////
  // check XBee's network parameters
  ////////////////////////////
  //checkNetworkParams();

  // Configure the calibration values
  pHSensor.setCalibrationPoints(CAL_POINT_10, CAL_POINT_7, CAL_POINT_4, CAL_TEMP);
  DOSensor.setCalibrationPoints(AIR_CALIBRATION, ZERO_CALIBRATION);
  ConductivitySensor.setCalibrationPoints(POINT1_COND,   POINT1_CAL,   POINT2_COND,
POINT2_CAL);
}

void loop()
{
  /////////////////////////////////////////
  // Turn on the board
  /////////////////////////////////////////
  // init XBee
  xbeeZB.ON();
  SensorSW.ON();
  delay(1000);

  //Turn on the RTC
  RTC.ON();

  //Turn on the accelerometer
  ACC.ON();

  /////////////////////////////////////////
  // Read sensors
  /////////////////////////////////////////

  accelerometerX = ACC.getX();

  //Reading acceleration in Y axis
  accelerometerY = ACC.getY();

  //Reading acceleration in Z axis
```

```
accelerometerZ = ACC.getZ();

// Getting Time
//GPS.getPosition();

// First dummy reading for analog-to-digital converter channel selection
PWR.getBatteryLevel();
// Getting Battery Level
batteryLevel = PWR.getBatteryLevel();

// Read the ph sensor
valuePH = pHSensor.readpH();
// Read the temperature sensor
valueTemp = TemperatureSensor.readTemperature();
// Convert the value read with the information obtained in calibration
valuePHCalculated = pHSensor.pHConversion(valuePH,valueTemp);
// Reading of the ORP sensor
valueORP = ORPSensor.readORP();
// Apply the calibration offset
valueORPCalculated = valueORP - CALIBRATION_OFFSET;
// Reading of the ORP sensor
valueDO = DOSensor.readDO();
// Conversion from volts into dissolved oxygen percentage
valueDOCalculated = DOSensor.DOConversion(valueDO);
// Reading of the Conductivity sensor
valueCond = ConductivitySensor.readConductivity();
// Conversion from resistance into ms/cm
valueCondCalculated = ConductivitySensor.conductivityConversion(valueCond);

// Create new frame (ASCII)
frame.createFrame(ASCII);
// Add Data values

// Add battery value
frame.addSensor(SENSOR_BAT, batteryLevel);
// Add Date values
frame.addSensor(SENSOR_DATE, RTC.year, RTC.month, RTC.date);
// Add Time values
frame.addSensor(SENSOR_TIME, RTC.hour, RTC.minute, RTC.second);
// Add Accelerometer values
frame.addSensor(SENSOR_ACC, accelerometerX, accelerometerY, accelerometerZ);

// Display sent frame
//frame.showFrame();

// Create New Frame
sendPacket();

frame.createFrame(ASCII);
// Add temperature
frame.addSensor(SENSOR_WT, valueTemp);
// Add PH
frame.addSensor(SENSOR_PH, valuePHCalculated);
// Add ORP value
frame.addSensor(SENSOR_ORP, valueORP);
// Add DO value
frame.addSensor(SENSOR_DO, valueDOCalculated);
// Add conductivity value
frame.addSensor(SENSOR_COND, valueCondCalculated);

dtostrf( valueTemp, 1, 2, Temp_str);
dtostrf( valuePHCalculated, 1, 2, pH_str);
dtostrf( valueORP, 1, 3, ORP_str);
dtostrf( valueDOCalculated, 1, 1, DO_str);
dtostrf( valueCondCalculated, 1, 1, COND_str);
sprintf(ACC_str, "%d;%d;%d", accelerometerX, accelerometerY, accelerometerZ);

// Display sent frame
//frame.showFrame();

sendPacket();

//snprintf(  message,  sizeof(message),  "%d/%d/%d,%d:%d:%d,%d,%s,%s,%s,%s,%s",
RTC.date,  RTC.month,  RTC.year,  RTC.hour,  RTC.minute,  RTC.second,  batteryLevel,
Temp_str, pH_str, ORP_str, DO_str, COND_str );
snprintf( message, sizeof(message), "%d/%d/%d,%d:%d:%d,%d,%s,%s,%s,%s,%s,%s,%i",
RTC.date,  RTC.month,  RTC.year,  RTC.hour,  RTC.minute,  RTC.second,  batteryLevel,
ACC_str, Temp_str, pH_str, ORP_str, DO_str, COND_str, error );
```

```
  USB.println( message );

  ////////////////////////////////////////////////////
  // Append data into file
  ////////////////////////////////////////////////////
  sd_answer = SD.appendln(filename, message);

  // Turn off the sensors
  SensorSW.OFF();

  // Sleep
  PWR.deepSleep(sleepTime,RTC_OFFSET,RTC_ALM1_MODE1,ALL_OFF);
  //Increase the sequence number after wake up
  sequenceNumber++;
}

void sendPacket()
{
  // send XBee packet
  error = xbeeZB.send( RX_ADDRESS, frame.buffer, frame.length );

  // check TX flag
  if( error == 0 )
  {
    //USB.println(F("send ok"));

    // blink green LED
    Utils.blinkGreenLED();

  }
  else
  {
    // Print error message:
    /*
     * '7' : Buffer full. Not enough memory space
     * '6' : Error escaping character within payload bytes
     * '5' : Error escaping character in checksum byte
     * '4' : Checksum is not correct
     * '3' : Checksum byte is not available
     * '2' : Frame Type is not valid
     * '1' : Timeout when receiving answer
    */
    //USB.print(F("Error Code: "));
    //USB.println(error,DEC);

    // blink red LED
    Utils.blinkRedLED();
  }
}
```

9.2 A.2 - Source code for offline sensor data transfer to Azure

```
import uuid
from datetime import datetime, timedelta
import random
import json
import time
import pandas as pd
import numpy as np
from azure.servicebus import ServiceBusService

def follow(thefile):
    thefile.seek(0,2)
    while True:
        line = thefile.readline()
        if not line:
            time.sleep(0.1)
            continue
        yield line

sbs = ServiceBusService(service_namespace='sw-eh-ns', shared_access_key_name='Libelium', shared_access_key_value='01lwja2uBU9UsbvHZpBnfR/pX64Sh3S+/iyZ17FlkAc=')
i = 492255

df_WMData = pd.read_csv("SW_TestData.csv", header=None, names=["date","time","bat","acc","wt","ph","do","orp","cond","conncode"])

for index, row in df_WMData.iterrows():
    sensor_date = datetime.strptime(row["date"], '%d/%m/%y').strftime('%y-%m-%d')
    sensor_time = datetime.strptime(row["time"], '%H:%M:%S').strftime('%H-%M-%S')
    event_time = datetime.strptime(row["date"] + " " + row["time"], '%d/%m/%y %H:%M:%S').strftime('%Y-%m-%dT%H:%M:%S.%fZ')
    datetime_fld = datetime.strptime(row["date"] + " " + row["time"], '%d/%m/%y %H:%M:%S') - timedelta(hours=8, minutes=0)
    datetime_fld = datetime_fld.strftime('%Y-%m-%dT%H:%M:%S.%f')

    val = [row["bat"], row["acc"], row["wt"], row["ph"], row["do"], row["orp"], row["cond"], sensor_time, sensor_date]

    for i, sensortype in enumerate(['BAT', 'ACC', 'WT', 'PH', 'DO', 'ORP', 'COND', 'TIME', 'DATE']):
        reading = {'id': i, 'id_wasp': 'SWat_01', 'id_secret': 419645370, 'sensor' : sensortype, 'value' : val[i],
                   'datetime' : datetime_fld, 'EventProcessedUtcTime' : event_time, 'PartitionId' : random.randint(0, 1), 'EventEnqueuedUtcTime' : event_time}
        s = json.dumps(reading)
        sbs.send_event('sw-ph', s)
        print(str(s))
    time.sleep(14)
```

9.3 A.3 – Microsoft Azure Configuration Screens

9.3.1 Azure Overview

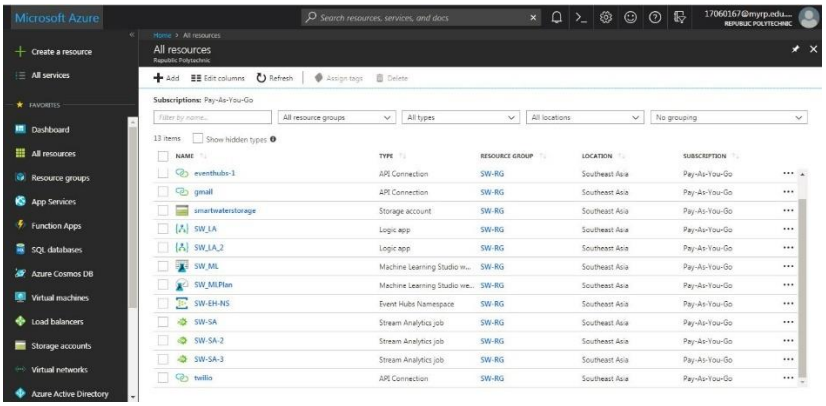


Figure 43 – Azure Configuration

9.3.2 Azure Event Hubs

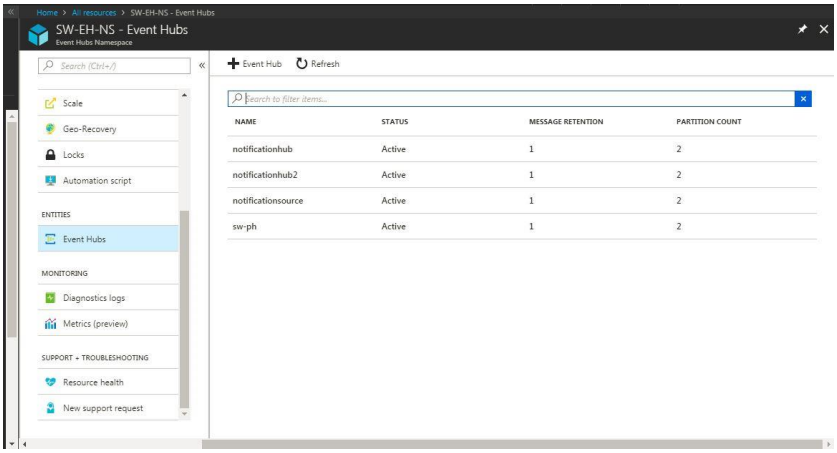


Figure 44 – Azure Event Hubs Configuration

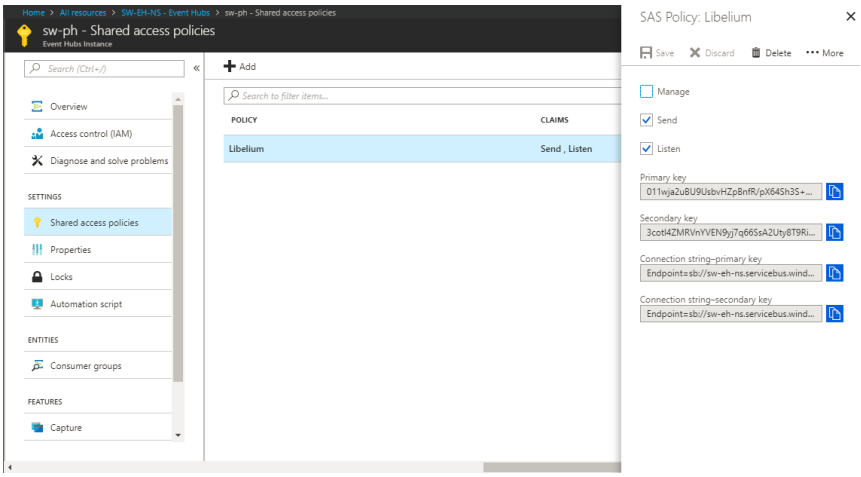


Figure 45 – sw-ph Event Hub Configuration

9.3.3 Azure Stream Analytics Job

9.3.3.1 SW-SA

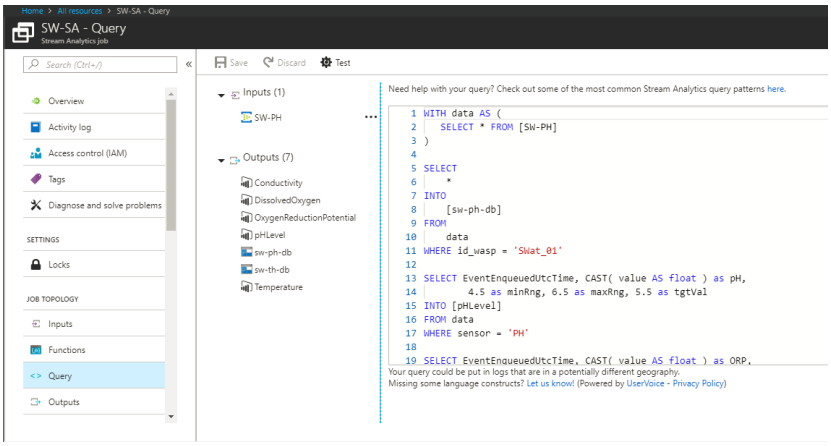


Figure 46 – SW-SA Query Configuration

```
WITH data AS (
   SELECT * FROM [SW-PH]
)

SELECT
    *
INTO
    [sw-ph-db]
FROM
    data
WHERE id_wasp = 'SWat_01'

SELECT EventEnqueuedUtcTime, CAST( value AS float ) as pH,
       4.5 as minRng, 6.5 as maxRng, 5.5 as tgtVal
INTO [pHLevel]
FROM data
WHERE sensor = 'PH'

SELECT EventEnqueuedUtcTime, CAST( value AS float ) as ORP,
       -0.1 as minRng, 0.1 as maxRng, 0.0 as tgtVal
INTO [OxygenReductionPotential]
FROM data
WHERE sensor = 'ORP'

SELECT EventEnqueuedUtcTime, CAST( value AS float ) as DO,
       0 as minRng, 150 as maxRng, 75 as tgtVal
INTO [DissolvedOxygen]
FROM data
WHERE sensor = 'DO'

SELECT EventEnqueuedUtcTime, CAST( value AS float ) as COND,
       -6000 as minRng, 1000 as maxRng, -2500 as tgtVal
INTO [Conductivity]
FROM data
WHERE sensor = 'COND'

SELECT EventEnqueuedUtcTime, CAST( value AS float ) as TEMP,
       20 as minRng, 30 as maxRng, 25 as tgtVal
INTO [Temperature]
FROM data
WHERE sensor = 'WT'
```

Figure 47 – SW-SA Query Script

9.3.3.2 SW-SA-2

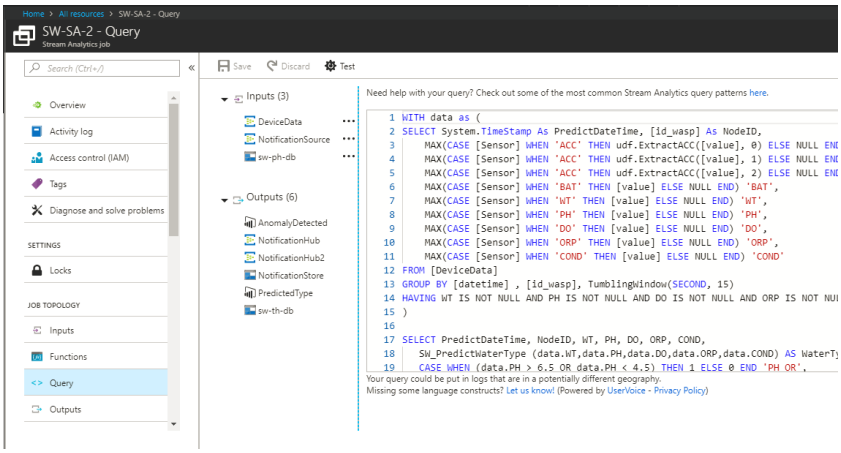


Figure 48 – SW-SA-2 Query Configuration

```
WITH data as (
SELECT System.TimeStamp As PredictDateTime, [id_wasp] As NodeID,
    MAX(CASE [Sensor] WHEN 'ACC' THEN udf.ExtractACC([value], 0) ELSE NULL END)
'ACC_X',
    MAX(CASE [Sensor] WHEN 'ACC' THEN udf.ExtractACC([value], 1) ELSE NULL END)
'ACC_Y',
    MAX(CASE [Sensor] WHEN 'ACC' THEN udf.ExtractACC([value], 2) ELSE NULL END)
'ACC_Z',
    MAX(CASE [Sensor] WHEN 'BAT' THEN [value] ELSE NULL END) 'BAT',
    MAX(CASE [Sensor] WHEN 'WT' THEN [value] ELSE NULL END) 'WT',
    MAX(CASE [Sensor] WHEN 'PH' THEN [value] ELSE NULL END) 'PH',
    MAX(CASE [Sensor] WHEN 'DO' THEN [value] ELSE NULL END) 'DO',
    MAX(CASE [Sensor] WHEN 'ORP' THEN [value] ELSE NULL END) 'ORP',
    MAX(CASE [Sensor] WHEN 'COND' THEN [value] ELSE NULL END) 'COND'
FROM [DeviceData]
GROUP BY [datetime] , [id_wasp], TumblingWindow(SECOND, 15)
HAVING WT IS NOT NULL AND PH IS NOT NULL AND DO IS NOT NULL AND ORP IS NOT NULL AND
COND IS NOT NULL
)

SELECT PredictDateTime, NodeID, WT, PH, DO, ORP, COND,
   SW_PredictWaterType (data.WT,data.PH,data.DO,data.ORP,data.COND) AS WaterType,
   CASE WHEN (data.PH > 6.5 OR data.PH < 4.5) THEN 1 ELSE 0 END 'PH_OR',
   CASE WHEN (data.DO > 150 OR data.DO < 0) THEN 1 ELSE 0 END 'DO_OR',
   CASE WHEN (data.ORP > 0.1 OR data.ORP < -0.1) THEN 1 ELSE 0 END 'ORP_OR',
   CASE WHEN (data.COND > 1000 OR data.COND < -6000) THEN 1 ELSE 0 END 'COND_OR',
   CASE WHEN (data.WT > 30 OR data.WT < 20) THEN 1 ELSE 0 END 'WT_OR',
   CASE WHEN
    (data.PH > 6.5 OR data.PH < 4.5) OR
    (data.DO > 150 OR data.DO < 0) OR
    (data.ORP > 0.1 OR data.ORP < -0.1) OR
    (data.COND > 1000 OR data.COND < -6000) OR
    (data.WT > 30 OR data.WT < 20)
    THEN 1 ELSE 0 END 'CRIT'
INTO [sw-th-db]
FROM data
WHERE SW_PredictWaterType (data.WT,data.PH,data.DO,data.ORP,data.COND) <> 0

SELECT PredictDateTime,
   SW_PredictWaterType (data.WT,data.PH,data.DO,data.ORP,data.COND) AS WaterType,
   CASE WHEN
    (data.PH > 6.5 OR data.PH < 4.5) OR
    (data.DO > 150 OR data.DO < 0) OR
    (data.ORP > 0.1 OR data.ORP < -0.1) OR
    (data.COND > 1000 OR data.COND < -6000) OR
    (data.WT > 30 OR data.WT < 20)
    THEN 1 ELSE 0 END 'CRIT'
INTO [PredictedType]
FROM data
WHERE SW_PredictWaterType (data.WT,data.PH,data.DO,data.ORP,data.COND) <> 0

SELECT PredictDateTime, NodeID, PH, DO, ORP, COND, WT,
   CASE WHEN (data.PH > 6.5 OR data.PH < 4.5) THEN 1 ELSE 0 END 'PH_OR',
```

```
    CASE WHEN (data.DO > 150 OR data.DO < 0) THEN 1 ELSE 0 END 'DO_OR',
    CASE WHEN (data.ORP > 0.1 OR data.ORP < -0.1) THEN 1 ELSE 0 END 'ORP_OR',
    CASE WHEN (data.COND > 1000 OR data.COND < -6000) THEN 1 ELSE 0 END 'COND_OR',
    CASE WHEN (data.WT > 30 OR data.WT < 20) THEN 1 ELSE 0 END 'WT_OR',
    CASE WHEN
     (data.PH > 6.5 OR data.PH < 4.5) OR
     (data.DO > 150 OR data.DO < 0) OR
     (data.ORP > 0.1 OR data.ORP < -0.1) OR
     (data.COND > 1000 OR data.COND < -6000) OR
     (data.WT > 30 OR data.WT < 20)
    THEN 1 ELSE 0 END 'CRIT'
INTO [NotificationHub2]
FROM data
WHERE IsFirst(SECOND, 60) = 1
```

Figure 49 – SW-SA-2 Query Script

9.3.3.3 SW-SA-3

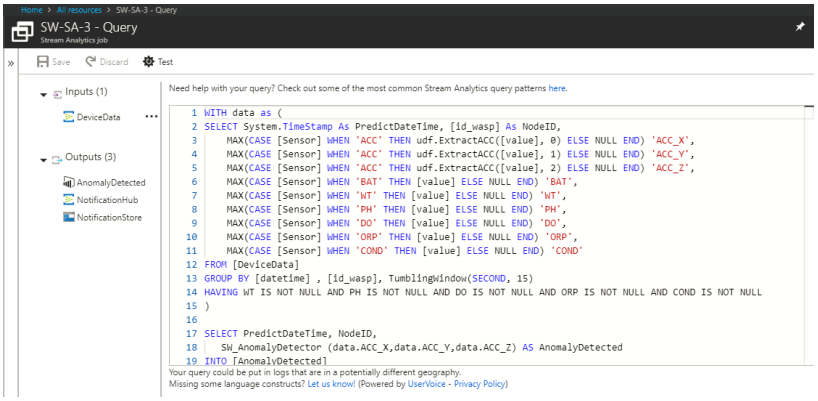


Figure 50 – SW-SA-3 Query Configuration

```
WITH data as (
SELECT System.TimeStamp As PredictDateTime, [id wasp] As NodeID,
    MAX(CASE [Sensor] WHEN 'ACC' THEN udf.ExtractACC([value], 0) ELSE NULL END)
'ACC_X',
    MAX(CASE [Sensor] WHEN 'ACC' THEN udf.ExtractACC([value], 1) ELSE NULL END)
'ACC_Y',
    MAX(CASE [Sensor] WHEN 'ACC' THEN udf.ExtractACC([value], 2) ELSE NULL END)
'ACC_Z',
    MAX(CASE [Sensor] WHEN 'BAT' THEN [value] ELSE NULL END) 'BAT',
    MAX(CASE [Sensor] WHEN 'WT' THEN [value] ELSE NULL END) 'WT',
    MAX(CASE [Sensor] WHEN 'PH' THEN [value] ELSE NULL END) 'PH',
    MAX(CASE [Sensor] WHEN 'DO' THEN [value] ELSE NULL END) 'DO',
    MAX(CASE [Sensor] WHEN 'ORP' THEN [value] ELSE NULL END) 'ORP',
    MAX(CASE [Sensor] WHEN 'COND' THEN [value] ELSE NULL END) 'COND'
FROM [DeviceData]
GROUP BY [datetime] , [id_wasp], TumblingWindow(SECOND, 15)
HAVING WT IS NOT NULL AND PH IS NOT NULL AND DO IS NOT NULL AND ORP IS NOT NULL AND
COND IS NOT NULL
)

SELECT PredictDateTime, NodeID,
   SW_AnomalyDetector (data.ACC_X,data.ACC_Y,data.ACC_Z) AS AnomalyDetected
INTO [AnomalyDetected]
FROM data
WHERE IsFirst(SECOND, 15) = 1

SELECT PredictDateTime,
   NodeID, ACC_X, ACC_Y, ACC_Z, SW_AnomalyDetector
(data.ACC_X,data.ACC_Y,data.ACC_Z) AS AnomalyDetected
INTO [NotificationStore]
FROM data
WHERE IsFirst(SECOND, 15) = 1

SELECT PredictDateTime, NodeID,
   SW_AnomalyDetector (data.ACC_X,data.ACC_Y,data.ACC_Z) AS AnomalyDetected
INTO [NotificationHub]
FROM data
WHERE IsFirst(MINUTE, 2) = 1
```

Figure 51 – SW-SA-3 Query Script

9.3.4 Azure Functions

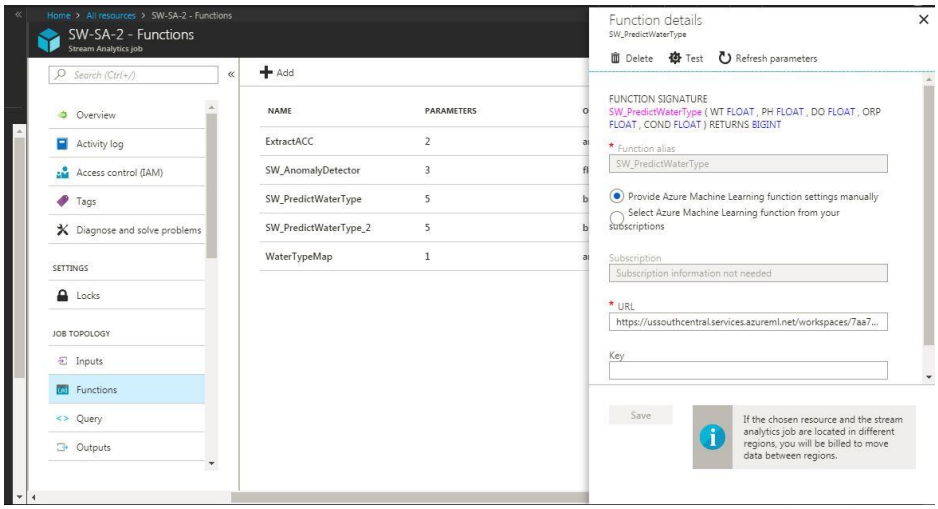


Figure 52 – SW_PredictWaterType Azure Function Configuration

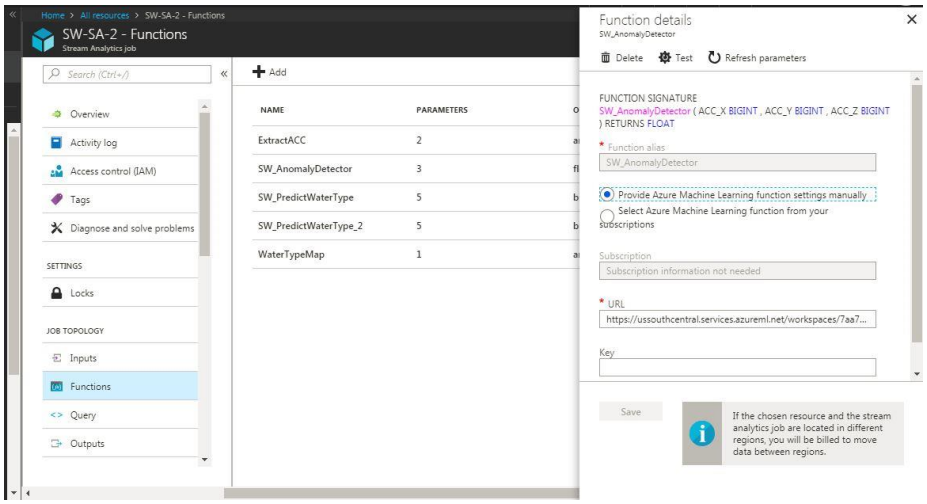


Figure 53 – SW_AnomalyDetector Azure Function Configuration

9.3.5 Azure Blob Storage

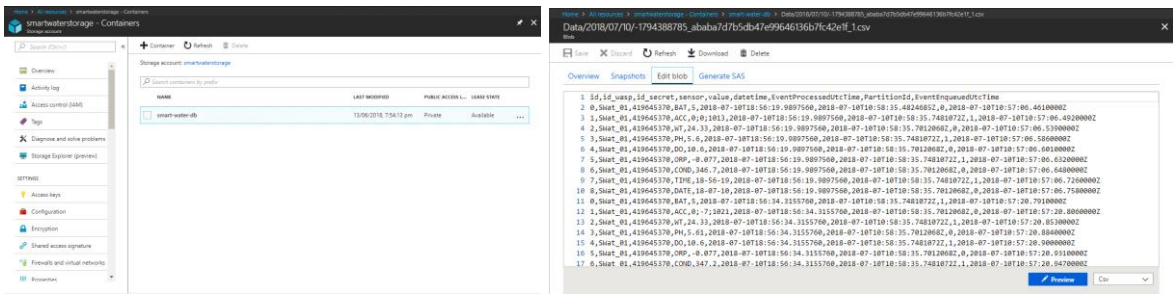


Figure 54 – smartwaterstorage Blob Container

9.3.6 Azure Logic App

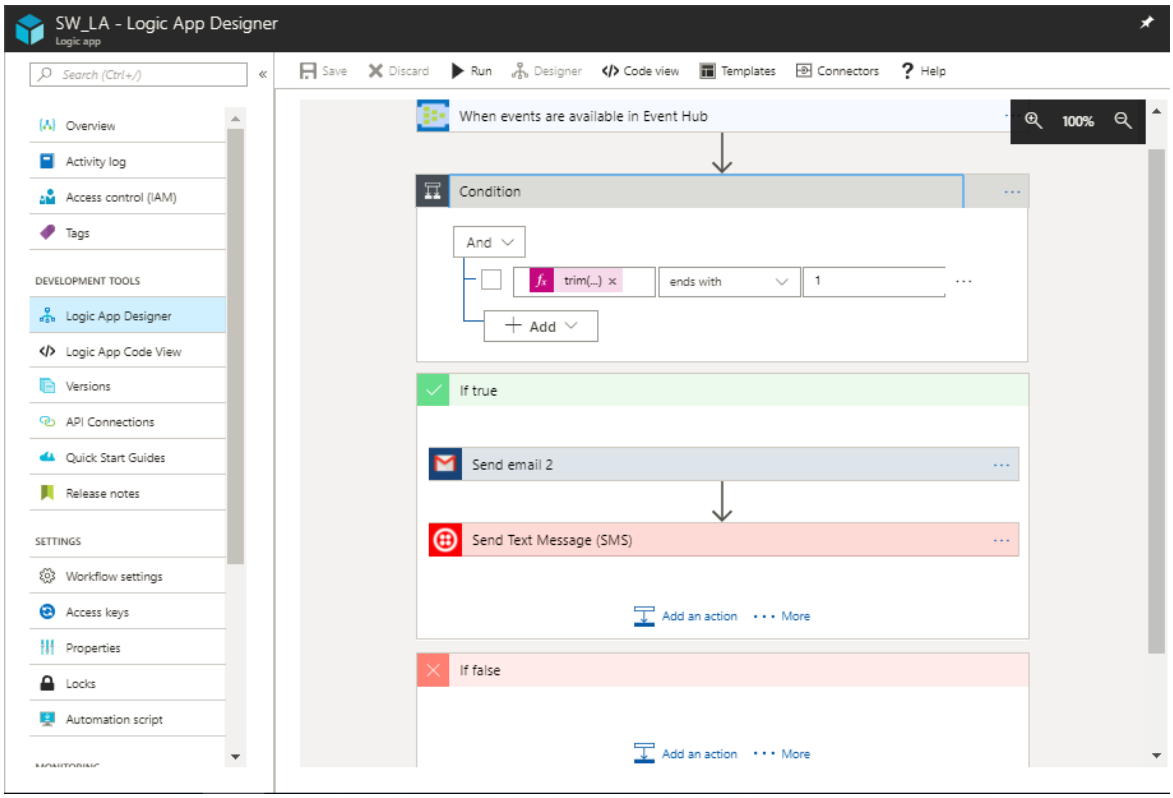


Figure 55 – SW_LA Azure Logic App Configuration

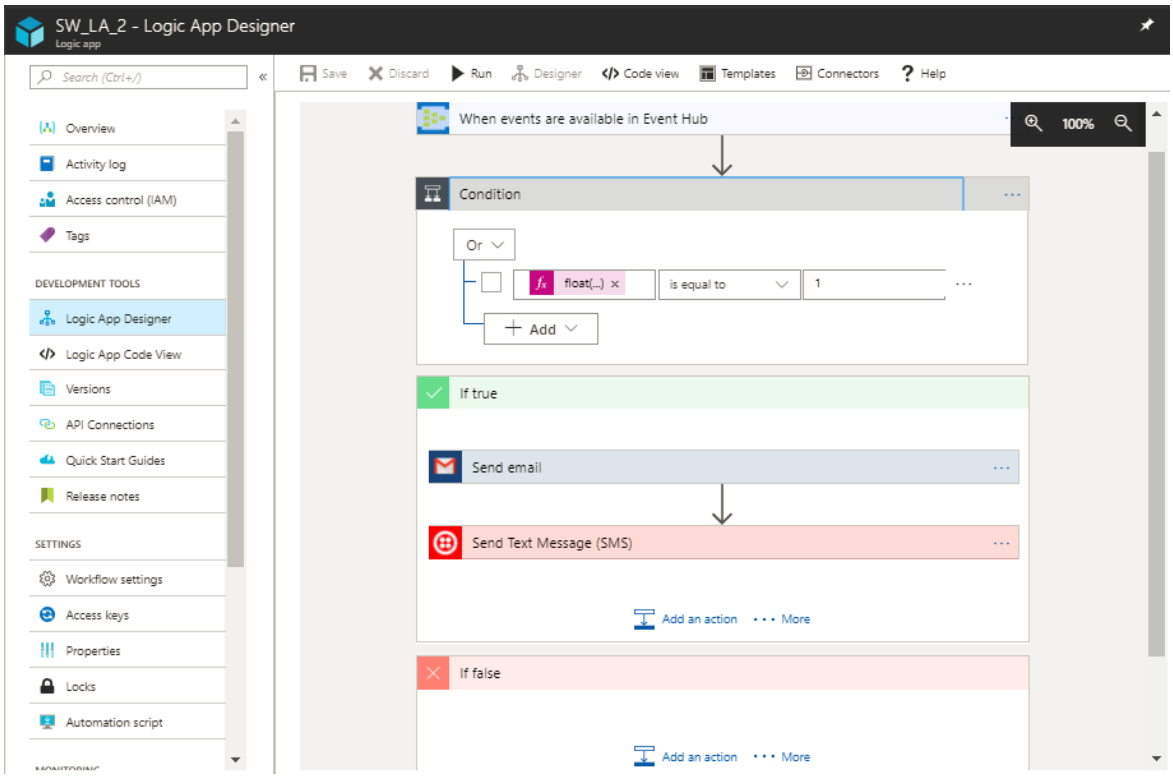


Figure 56 – SW_LA_2 Azure Logic App Configuration

9.3.7 Azure Machine Learning

9.3.7.1 General

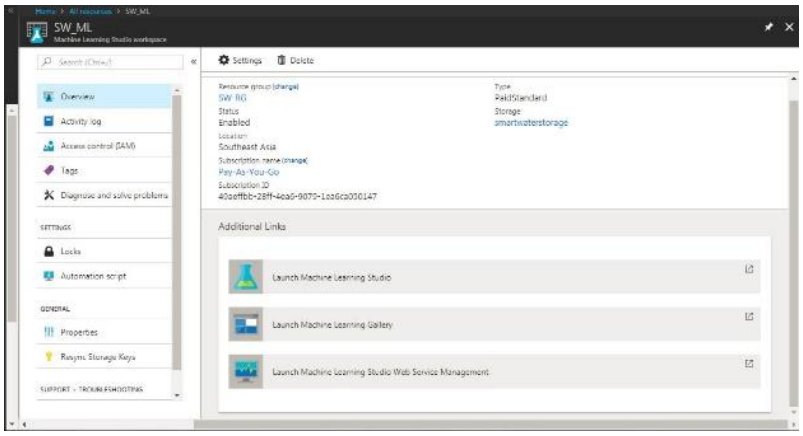


Figure 57 – SW_ML Azure Machine Learning Studio Workspace

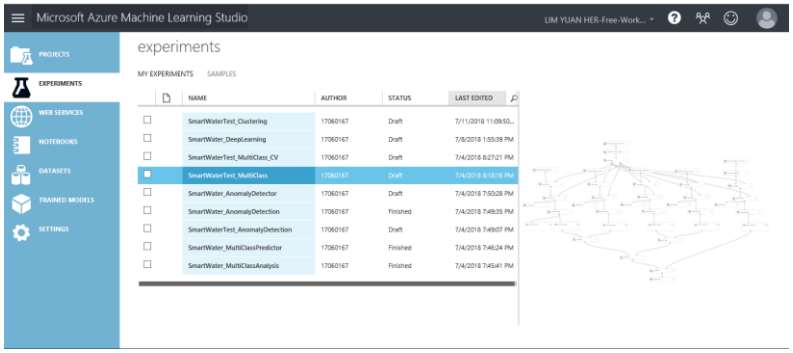


Figure 58 – SW_ML Azure Machine Learning Studio Experiments

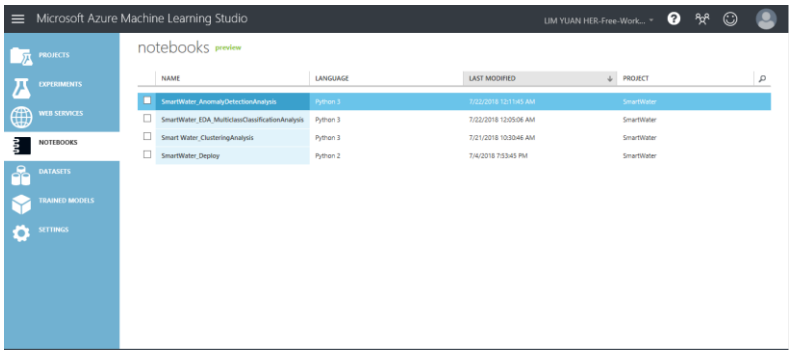


Figure 59 – SW_ML Azure Machine Learning Studio NoteBooks

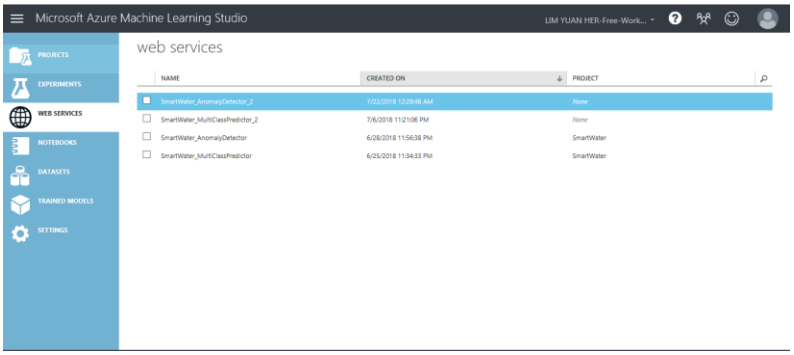


Figure 60 – SW_ML Azure Machine Learning Studio Web Services

9.3.7.2 Experiments

9.3.7.2.1 SmartWaterTest_MultiClass

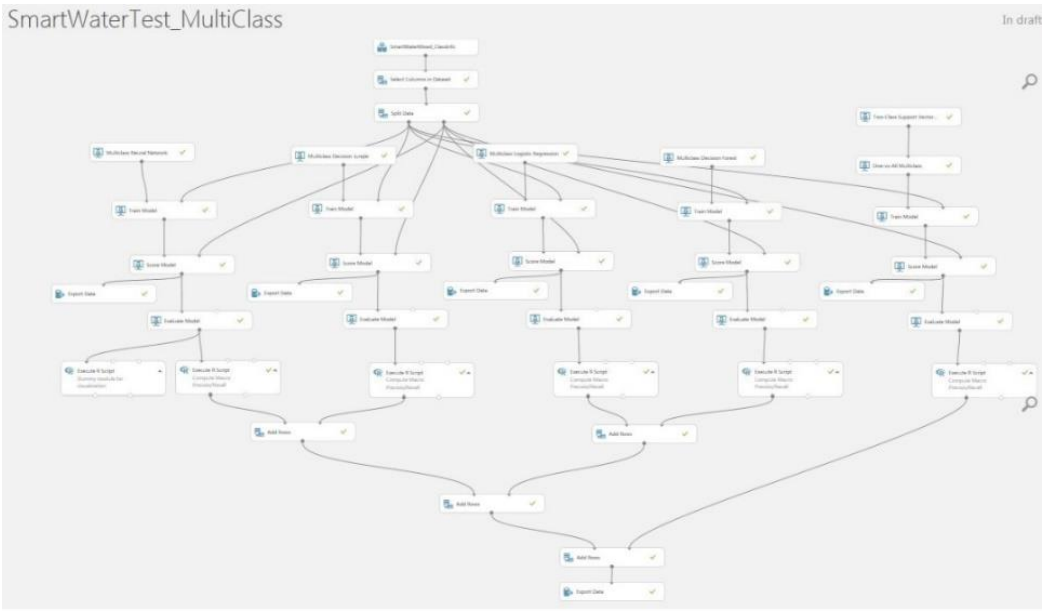


Figure 61 – SmartWaterTest_MultiClass Experiment Configuration

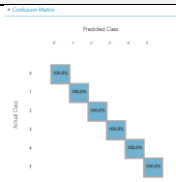
s/n	Algorithm	Metric Score	Confusion Matrix
1	Neural Network Decision Jungle Decision Forest	Metrics Overall accuracy 1 Average accuracy 1 Micro-averaged precision 1 Macro-averaged precision 1 Micro-averaged recall 1 Macro-averaged recall 1	
2	Multiclass Logistic Regression	Metrics Overall accuracy 0.911917 Average accuracy 0.970639 Micro-averaged precision 0.911917 Macro-averaged precision 0.940783 Micro-averaged recall 0.911917 Macro-averaged recall 0.876812	
3	OneVsAll Multiclass	Metrics Overall accuracy 0.958549 Average accuracy 0.986183 Micro-averaged precision 0.958549 Macro-averaged precision 0.969134 Micro-averaged recall 0.958549 Macro-averaged recall 0.942029	

Figure 62 – SmartWaterTest_MultiClass Algorithm Evaluation Individual Metric Scores

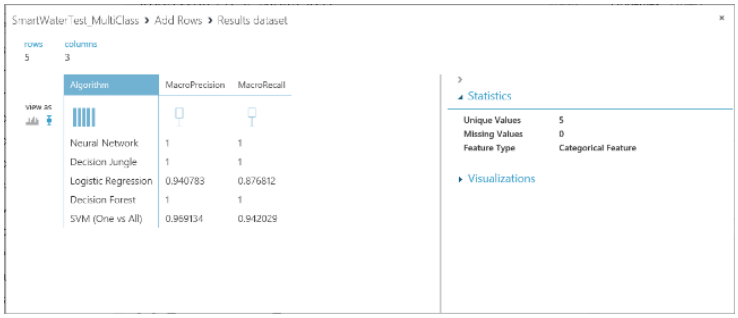


Figure 63 – SmartWaterTest_MultiClass Algorithm Evaluation Summary Metric Scores

9.3.7.2.2 SmartWaterTest_MultiClass_CV

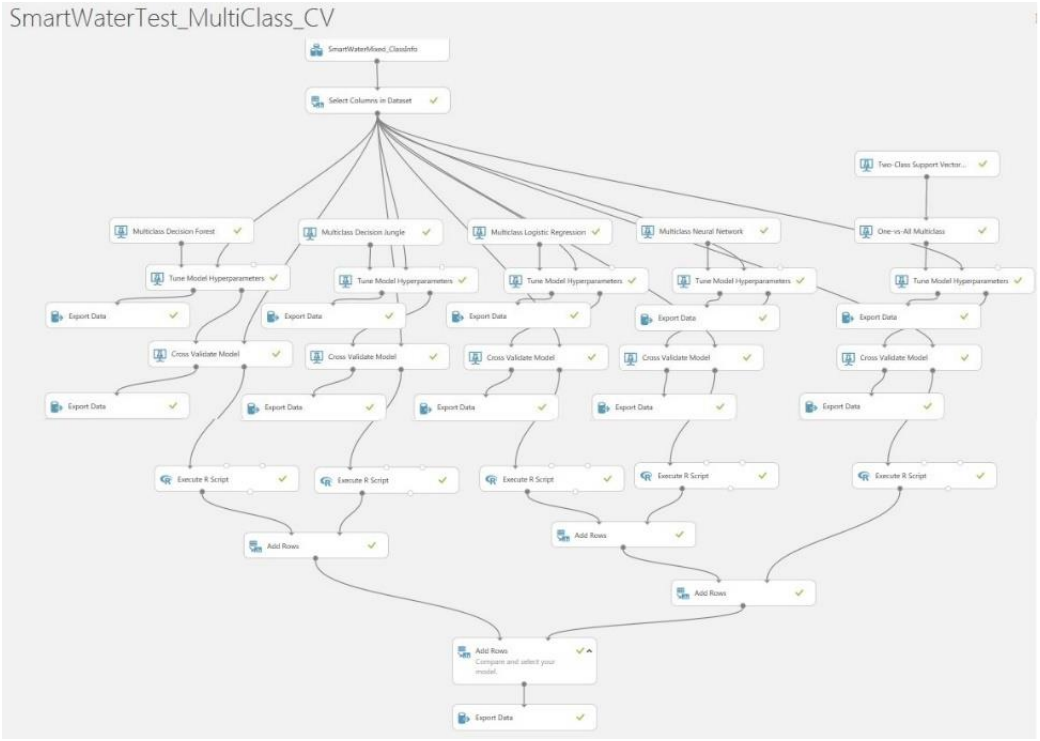


Figure 64 – SmartWaterTest_MultiClas_CVs Azure Machine Learning Experiment Configuration

Model		Average Log Loss for Class "0"	Precision for Class "0"	Recall for Class "0"	Average Log Loss for Class "1"	Precision for Class "1"	Recall for Class "1"	Average Log Loss for Class "2"	Precision for Class "2"	Recall for Class "2"	Average Log Loss for Class "3"	Precision for Class "3"	Recall for Class "3"	Average Log Loss for Class "4"	Precision for Class "4"	Recall for Class "4"	Average Log Loss for Class "5"	Precision for Class "5"	Recall for Class "5"
Microsoft.Analytics.Modules.Gemini.Dll.MulticlassGeminiDecisionForestClassifier	mean	0	1	1	0.029184	1	1	0.010362	1	0.994118	0.014467	1	1	0.008677	0.99375	1	0.000215	1	1
Microsoft.Analytics.Modules.Gemini.Dll.MulticlassGeminiDecisionForestClassifier	std	0	0	0	0.037308	0	0	0.014315	0	0.018602	0.019141	0	0	0.007262	0.019764	0	0.000679	0	0
Microsoft.Analytics.Modules.Gemini.Dll.MulticlassGeminiDecisionJungleClassifier	mean	0	1	1	0.019765	1	0.986667	0.008791	1	0.994118	0	0.989474	1	0.001695	0.99375	1	0	1	1
Microsoft.Analytics.Modules.Gemini.Dll.MulticlassGeminiDecisionJungleClassifier	std	0	0	0	0.049247	0	0.042164	0.025642	0	0.018602	0	0.033287	0	0.003672	0.019764	0	0	0	0
Multi-class Logistic Regression	mean	0.004737	1	1	0.163824	1	1	0.05993	1	1	0.02862	1	1	0.000635	1	1	0.004326	1	1
Multi-class Logistic Regression	std	0.0008	0	0	0.0683	0	0	0.006034	0	0	0.016167	0	0	0.00004	0	0	0.012144	0	0
Multi-class Neural Network	mean	0.001004	1	1	0.117199	1	1	0.027715	1	1	0.03429	1	1	0.00011	1	1	0.000464	1	1
Multi-class Neural Network	std	0.000157	0	0	0.066618	0	0	0.004832	0	0	0.041838	0	0	0.000013	0	0	0.000097	0	0
One-vs.-All Classifier(SVM (Pegasos-Linear))	mean	0.098563	1	1	0.397673	1	0.866506	0.174235	0.971202	1	0.026302	0.9416	1	0.026596	1	1	0.009923	1	1
One-vs.-All Classifier(SVM (Pegasos-Linear))	std	0.042744	0	0	0.223112	0	0.11918	0.018385	0.039025	0	0.013667	0.042831	0	0.007448	0	0	0.015238	0	0

Figure 65 – SmartWaterTest_MultiClass_CV Cross Validation Summary Metric Scores

9.3.7.2.3 SmartWaterTest_AnomalyDetection

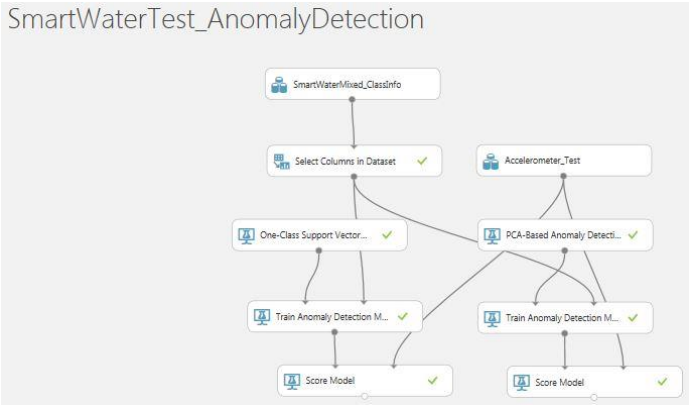


Figure 66 – SmartWaterTest_AnomalyDetection Azure Machine Learning Experiment Configuration

9.3.7.2.4 SmartWaterTest_Clustering

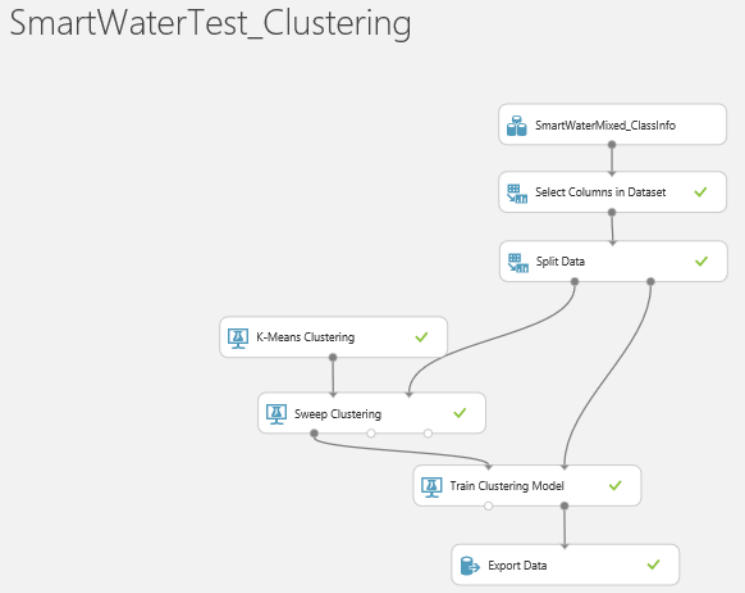


Figure 67 – SmartWaterTest_Clustering Azure Machine Learning Experiment Configuration

9.3.7.3 Web Services

9.3.7.3.1 SmartWater_MultiClassPredictor

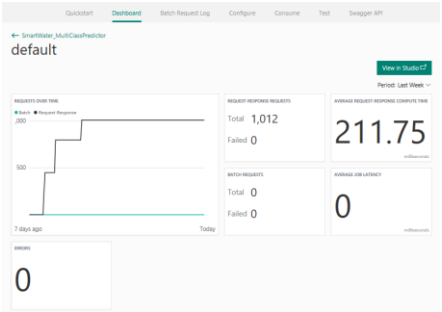


Figure 68 – SmartWater_MultiClassPredictor Web Service Dashboard

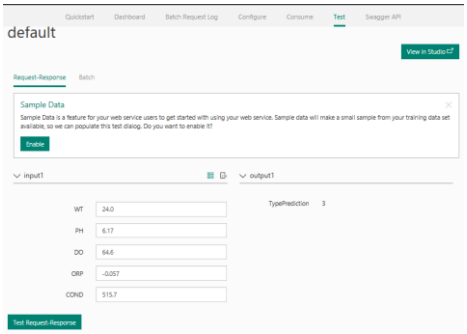


Figure 69 – SmartWater_MultiClassPredictor Web Service Test Screen

9.3.7.3.2 SmartWater_AnomalyDetector

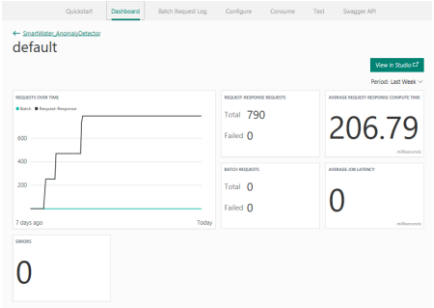


Figure 70 – SmartWater_MultiClassPredictor Web Service Dashboard

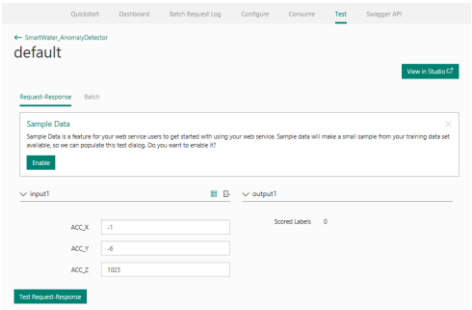


Figure 71 – SmartWater_MultiClassPredictor Web Service Test Screen

10 Project Poster

Section Explanation: *Pls create a poster write-up for your project. The file should be saved as a PPT file and embedded in this section as an object.*

Project Poster Template.ppt