INTERNSHIP REPORT

WATER QUALITY MONITORING USING IOT & DATA ANALYTICS

prepared for:

Dr. Munesh Singh
Asst. Professor
Indian Institute of Information Technology Design & Manufacturing,
Kancheepuram

prepared by:

[Rishu Raj]
[Manendra Nath Shukla]
[Soumya Michael]
[V. Suriyaa]
B.Tech CSE, IIIrd year
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi

TABLE OF CONTENTS

CHAPTER 1

1. ABSTRACT

Water Quality measurement has become a vital aspect for sustainability of human beings, for a healthier life. A good quality water may have several health benefits for humans as well as cultivation, aquaculture and many more. Our country is having a good growth statistics in recent years in case of aquaculture which are bringing a goof profit to economy. The major issue in this rearing of aquatic organisms, comes up at the monitoring phase. In rearing of fishes we require mainly, altering of the minerals for proper growth and sustainability of fishes in adequate environment. As for fishes, the habitat for them matters mainly on the temperature, pH value, ammonia concentrations etc. Measured using different sensors, they yield results which give us the stats of the quality of water. Our challenge is, in many cases the sensors may give fault occurrences or may be damaged. For those critical situations, our rearing could be led in undesired way.

Using IoT and Data Analytics, we can predict the amount of NH_3 , DO, pH value and temperature of water. Consequently, we could reduce the manual monitoring of water in case of Aquaculture like cultivation of fishes, oysters, measles etc. They require an appropriate amount of nutrients in water which requires regular monitoring of water quality to prevent their irregular growth. Implementing this underwater, regular monitoring manually will be reduced, and can get the required results of water remotely, using cloud technology. In case of irregular functioning of components, calibration is also possible remotely.

2. INTRODUCTION

Aquaculture is becoming India’s one of the flourishing sectors for cultivation. Globally, our India ranks as 3rd in Fisheries and 2nd in Aquaculture. As per growing population, the need for fishes and other sea food is increasing proportionally. A good habitat for rearing of fishes is required with proper amount of nutrients such as pH value, DO, Ammonia, temperature value. People has started shifting to inland fisheries, where the water quality is needed to be maintained at appropriate level for healthy growth of fisheries as well as other aquatic organisms. The unbalanced proportion of any one mineral in water could led to irregular growth as well as death of fisheries.

In aquaculture, important aspects are considered such as regular monitoring of temperature, pH value, ammonia concentrations, DO in the water. These factors could yield us desired results for the best survival of fishes. Any parameter change could lead us to loss of growth affecting the cost and time. Like, pH value alters as per change in the amount of CO_2 due to respiration of fishes in water, or maybe due to nitrification which changes the ammonia content in water. DO contents are changed due to excessive algae growth caused by phosphorus. As the algae dies they consume dissolved oxygen underwater which causes change in the DO concentration.

In our project, we are doing the regular monitoring of nutrients such as DO, pH value(salinity or acidity), Ammonia, temperature and concentration. It is implemented using sensor based system where we are setting up the device underwater in aquarium or fish habitat. The sensors would acquire the water pH value, temperature, Ammonia value. All the resulting value of these sensors will be received by the database stored in Cloud server. In our case, we are using Heroku Cloud Service. The simultaneous transmission of this data is sent and received by CloudMQTT which acts as a broker between the transmission to Cloud server.

All the magintude data fetched by the sensors, gathered at the cloud will act as input for data analytics. Data analytics is performed at the cloud server where we apply regression algorithms using machine learning such as Random forest, linear regression, multilayer perceptron regression model, Gradient boosting regression. Using these algorithms, analysis of the data is performed and we discover what could be the predicted value upon the given test value. In cases like remote areas, where internet could be a mere possibilty, there too the device would function properly wth some pre-defined data sets and algorithms fed into the database with a local base station.

Our project is a solution to the problems where we cannot breed a rich quality of aquatic breeds due to lack of regular monitoring and unpredicted behaviours, occuring in water habitat which leads to the improper growth. For eg. Our country, India cannot produce a good quality of pearl gems due to lack of information of the required concentrations of minerals to be added at desired intervals of time.

3. LITERATURE SURVEY

The present water quality manually monitoring system is very tedious process because it takes lot of time.The architecture will have the data monitoring station,nodes and remte station,all station areconnected together using wireless connection link, and the data send from to station which contain arm controller which is used forcompact space application and the data which is collected is send to the base station such as ph value,turbidity,conductivity etc..and can visulized remotely in server pc[1]

In this paper water quality monitoring system is based on bp neural network,where is going to consider all aparameters and factors of the water and surrounding environment and make a analysis based on bp neural network multi-sensor information fusion algorithm is used.[2]

Fixed node on line detection and regular sampling are the methods of water quality monitoring in water at current stage.this paper introuced the unmanned ship for watr quality monitoring and texting,which can be in any off the location in real time by means of dynamic and static nodes[3]

In order to solve the existing traditional rural drinking water, introduced a wsn based drinking water monitoring system and it consist of node,gate server,node-node.monitoring node send data to the gateway through wireliss module send directly,or it willsend through routing gate way to gate way node and it will check periodicaly the quality of the water and amkes the report in form of graphical representation to find the composition of chemicals present in water[4]

CHAPTER 2

1. PROBLEM STATEMENT

Water quality monitoring system is a continuous process but there is no continuous and remote monitoring in existing system, human labour is required to operate the system for measurement of water nutrients and also less reliable, no monitoring at the source of waters i.e. no on field monitoring and the frequency of testing is very low which results the health problem for life of human being, fish, pearl and so on. Due to these disadvantages of the existing system it is required to develop a system that will allow real time and continuous monitoring of water quality nutrients for cultivation of pearl, fish using IoT with Data Analytics

2. PROPOSED SYSTEM

Here we proposed a robust and efficient system for monitoring of water quality at remote places like ponds, aquarium, fish and pearl cultivating field using high quality and robust sensors with low power consumption, low cost and high detection accuracy. pH, conductivity, dissolved oxygen, turbidity level, ammonia, etc. are the parameters that are measured and analysed to improve the water quality. To make this system more cost effective we replaced the ammonia sensor with our own developed sensor called reagent based sensor for measurement of ammonia quantity water.

Following are the main objectives of idea implementation:

- To measure water parameters such as pH, dissolved oxygen, turbidity, ammonia content, etc. using own developed and available sensors at remote place.
- To collect data from various sensor nodes and send it to cloud server by using cloud MQTT broker.
- To analyse quality parameters for quality control through machine learning using python

- To send alert to an authorized person automatically when there is a fluctuation is the measurement of sensor value, so that, recalibration and necessary actions can be taken through our developed user interface as web app.
- To detect and identify the fault and failure of sensor through the concept of predictive maintenance.

Software Design:

A user interfaced is developed to monitor the data coming from raspberry pi to database stored in server through cloud mqtt broker using mqtt protocol. All the analytics is done on the server side program which includes the prediction of all parameters.

3. METHODS

i. DATASET

Dataset is a collection of data, which is used for computer processing such as in Machine learning dataset taken as an input for prediction, data analysis. Therefore, dataset acts as an input for performing various computations. In our project, dataset is an input for data analysis, training and predicting parameters, which would be helpful in case of any sensor failure. Dataset contains following parameters

- Temperature (Float type)
- Salinity (Float type)
- Dissolved oxygen percent (Float type)
- pH value (Float type)
- Turbidity (Float type)

All this parameters plays a crucial role in the aquaculture, which affects growth and production of aquatic organisms such as fish, crustaceans, mollusca, aquatic plant, algae etc.

Temperature effects the growth and survival of shrimp, fish and other aquatic animals. Mainly the aquatic organisms are poikilothermic(cold blooded), therefore they depend upon the temperature of surrounding water. In dataset Temperature ranges from 5.17 to 30.98. Temperature varies due to climate condition.

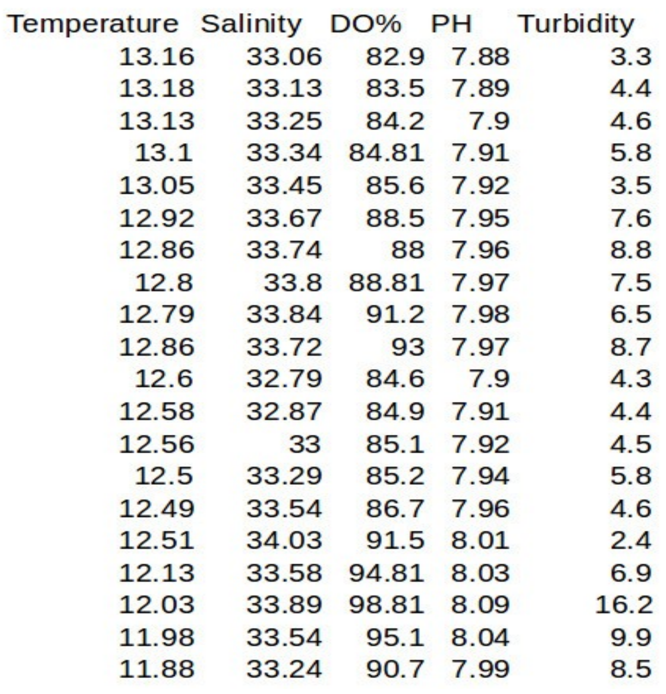
Salinity is the concentration of dissolved salt in given volume of water. Egg fertilization and incubation, yolk sac resorption, early embryogenic, swimbladder inflation, larval growth depend upon salinity. Salinity ranges from 12 to 35 in dataset.

Dissolved oxygen (DO) is an important indicator of water quality. It is the amount of oxygen dissolved in given volume of water. It is essential for the survival of fish and other aquatic organism. Diffusion, aeration, photosynthesis, respiration and decomposition affects the Dissolved oxygen concentration. DO % ranges from 17.62 to 308.2 in dataset.

pH Aquaculture organisms usually die if the pH is less than 4 or greater than 11 and optimum pH for most of the aquatic is between 6 to 9. pH value fluctuate due to photosynthesis and respiration. In dataset pH ranges from 7.31 to 8.83.

Turbidity is an indicator of amount of suspended particles in water, which block light to aquatic plants for performing photosynthesis. This, in turn affects organisms that depend on these plant for food and oxygen. Ranges from 0.1 to 149 in dataset.

Dataset conclude various relationship between parameters, as we can predict when temperature increase pH decreases and when pH decreases results in decrease in amount of dissolved oxygen. As turbidity increases results in decrease in amount of dissolved oxygen in water.



Temperature	Salinity	DO%	PH	Turbidity
13.16	33.06	82.9	7.88	3.3
13.18	33.13	83.5	7.89	4.4
13.13	33.25	84.2	7.9	4.6
13.1	33.34	84.81	7.91	5.8
13.05	33.45	85.6	7.92	3.5
12.92	33.67	88.5	7.95	7.6
12.86	33.74	88	7.96	8.8
12.8	33.8	88.81	7.97	7.5
12.79	33.84	91.2	7.98	6.5
12.86	33.72	93	7.97	8.7
12.6	32.79	84.6	7.9	4.3
12.58	32.87	84.9	7.91	4.4
12.56	33	85.1	7.92	4.5
12.5	33.29	85.2	7.94	5.8
12.49	33.54	86.7	7.96	4.6
12.51	34.03	91.5	8.01	2.4
12.13	33.58	94.81	8.03	6.9
12.03	33.89	98.81	8.09	16.2
11.98	33.54	95.1	8.04	9.9
11.88	33.24	90.7	7.99	8.5

Fig. 2.3.1 Dataset

ANALYSIS OF DATASET

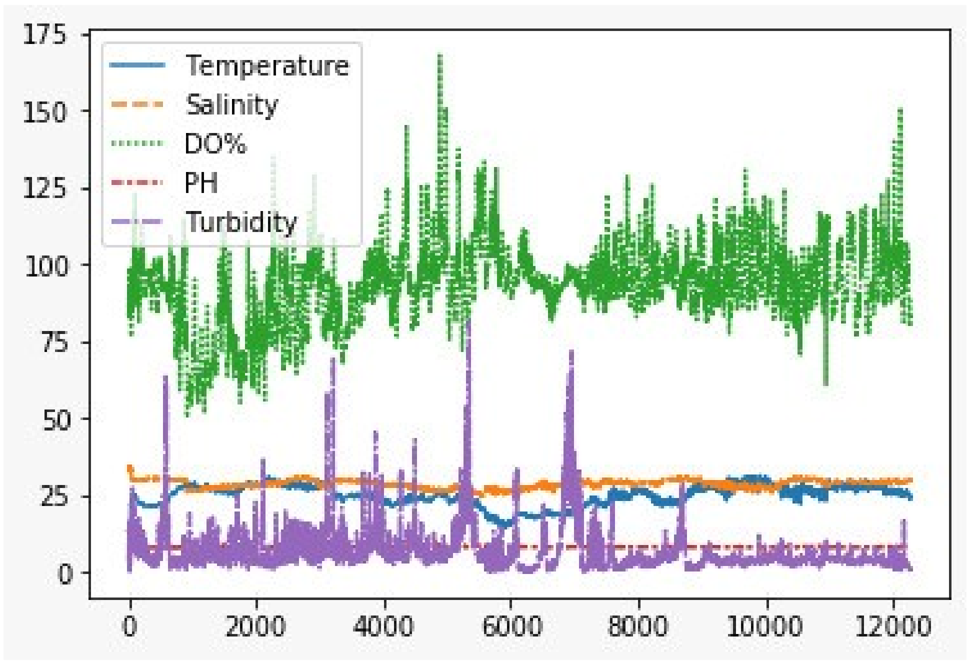
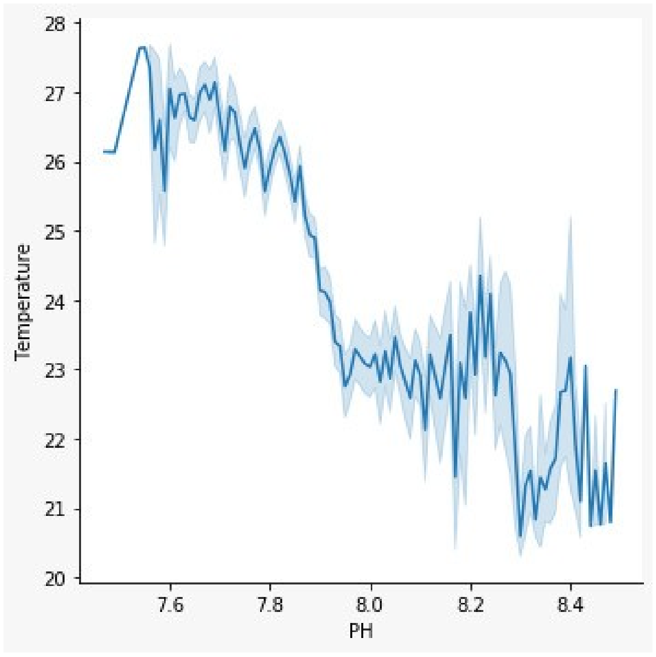


Fig. 3.1.1 Analysis of parameters



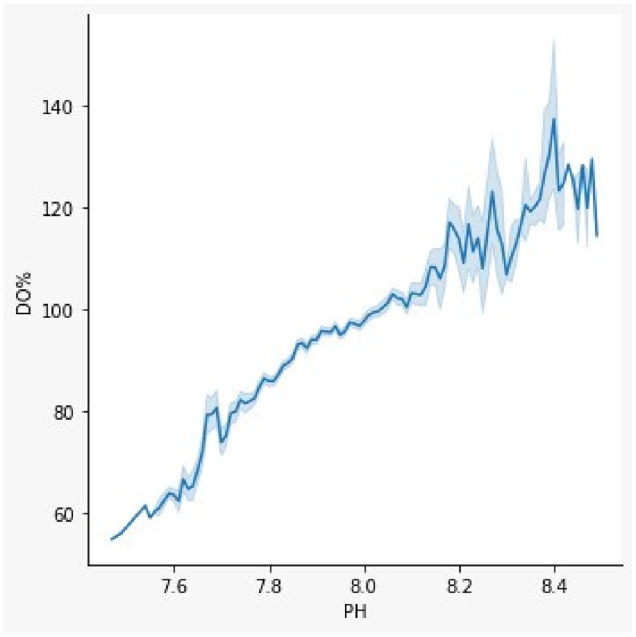


Fig. 3.1.2 Relation between pH and Temperature

Fig. 3.1.3 Relation between pH and DO

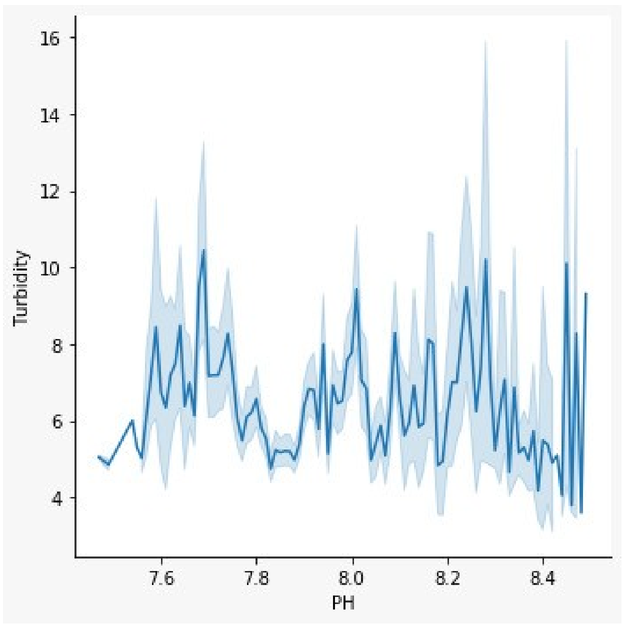


Fig. 3.1.5 Relation between pH and Turbidity

ii. FLASK

Flask is a light weight and micro framework provided by python programming language for web application and baseband on the WSGI toolkit and jinja2 template engine. It takes the flexible python programming language and provides a simple template for webapp development. Once imported into python, Flask can be used to save time building web applications. It keeps the core simple but extensible. It has no database abstraction layer, form validation, or any other components. Flask supports extensions. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and more.
The overall programing is done with python, the sensors data is extracted using this software and stored. The HTML and CSS is used to design the web page of the webapp. Flask is the framework that connect Python and HTML.

iii. HEROKU

Heroku is a cloud platform that supports several programming languages in which you can deploy your application tool. It provide Platform as a Service (PaaS), which is flexible, elegant and easy to use. So, here we are using Heroku to deploy our flask app. Heroku uses postgresql database to persist data. Using Git, source code is deployed in Heroku. For starting the application it looks for a Procfile in the application’s root directory. When application source received by Heroku platform, it initialize the building of the source application. The output are assembled into a slug. Through Heroku CLI you can interact with their service.

iv. CLOUD MQTT

CloudMQTT is a cloud based platform for creating and deploying managed cloud hosted MQTT brokers to your applications. MQTT stands for Message Queuing Telemetry Transport. We are using CloudMQTT as broker for sending and receiving message between client and server. It provide an asynchronous communications protocol. MQTT uses set of control packets. Publisher or subscriber sends CONNECT packets to the broker and for each CONNECT packet broker response with CONNACK packet. CONNACK packet response, whether the connection established or not. Through PUBLISH packet we can send message to broker and server response back with PUBACK packet. SUBSCRIBE packet sends by client to broker indicating that they want to subscribe to particular topic. By sending the PINGREQ we can keep the connection alive between server and client, and server response with PINGRESP. Subscriber is the listener and publisher is the sender. Broker receive the message from publisher and sends to all subscriber who subscribed for particular topic

v. REGRESSION MODELS:

For Data Analysis between temperature and pH value, we can use our dataset as input. Thereby, we can train the model by taking a ratio of data as test data and trained data. Therefore, we can train our system and use it for predicting the data in cases, when we are unable to use any of sensor.

It is a statistical analysis of data used to predict one variable by using another variable. It is used to estimate how the independent variable have relationship between dependent variables, and using it we explore the various forms of relationships among them. Using this we can determine which factors affect the most, and which all factors can be ignored.

In our project, regression analysis is performed using six regression models as follows:

A) Linear Regression Model
B) Multilayer Perceptron Regression Model
C) Support Vector Machine Regression Model
D) Decision Tree Regression Model

E) Gradient Boosting Regression Model
F) Random Forest Regression Model

Using these, we predict the relation among the pH value and temperature value.

A. Linear Regression Model

Linear Regression is a statistical model that attempt to shows the relationship between two variables with linear equation. It predict a dependent variable value based on the given independent variable. As we plot dependent variable on y-axis and independent variable on x-axis, it gives straight line that best fit the data points. It uses the formula

$$\mathbf{Y = mx + b}$$

here, b = intercept
m= slope of line

$$m = \frac{\sum(x - mean(x))(y - mean(y))}{sum(x - mean(x))}$$

x and y are variables, where y is the dependent variable and x is an independent variable.

B. Multilayer Perceptron Model

The ANN is abbreviated as artificial neural network, as initial stage we used this algorithm to process the data for predicting the quality of water. On first stage helps us to understand impact of decreasing and increasing the dataset horizontally and vertically on computation time and on second stage it helps us to understand the cases an situations at where the model fits best and on third stage it also helps us why some of the model works better in certain situation and environment. Multilayer perceptron denotes the n number of hidden layer present between input layer and output layer, the layers are dense layer activation layer etc.., it can have n number of dense layer.

NEURAL NETWORK FORMULATION

The link between nodes are the most cruces finding in an ANN. We have to find the weight between the linkages after broadcasting framework. The known value will the inputs. Let’s consider the inputs as n1,n2, and n3, Hidden states defined as h1,h2,h3... hn, and output has o1,o2. The weight between the node is lined can be denoted as follows

W(n1h1) is the weight of link between n1 and h1 nodes and assign the weight randomly between link to start the algorithm, find the activation function using the linkages and inputs, using the activation values of nodes and hidden layers to find the output activation value, finally find the error rate and back propagate or recalibrate it

formula

LOGIT(H1)=W(N1H1)*N1+W(N2H1))*N2+W(N3H1)*N3+CONSTANT=F=>P(H1)=1/(1+E^(-F))

by using this algorithm we got 70% accuracy for the prediction model and text data accuracy, which is not actually enough to say the prediction value is correct

C. Support Vector Machine (SVM):

Support Vector Machine is known as another discriminative model like Linear Regression. SVM is a supervised learning model for analysing the data used for classification, regression, and outliers detection. In our project we used it for regression purpose to predict the water quality parameters like pH, temperature, ammonia, DO and so on. SVM is most applicable in the case of Non-Linear data. But in our case we have random and linear data so that it did not worked well to use it as a best machine learning model. Given Input *x*, Class or Label *c* and LaGrange multipliers *α*; weight vector Θ can be calculated by following equation:

$$\Theta = \sum_{i=1}^{m} \alpha_i c_i x_i$$

D. Decision Tree Regression Model

Decision Tree is nothing but it is a map of all the possible outcome of a series of related choices. One of the best thing about Decision Tree is that it allows an individual or organization to weigh possible actions against based on their costs, probabilities and benefits. Normally, a decision tree start with a single node, which branches into as much as possible outcomes. Each of those output leads to additional nodes. So, at last it will become tree like structure.

E. Gradient Boosting Regression Model

Gradient Boosting Algorithm is used in machine learning for regression as well as classification, as it produces a prediction model in the form of an ensemble of weak prediction models. It is a kind of supervised learning algorithm in which we derive a loss function and we need to minimize it. For which we calculate Mean Square Error.

Mean Square Error

Mean Square Error = $\sum (y - y_i{}^p)^2$

Where, y_i = ith target value, $y_i{}^p$=ith prediction, $L(y_i, y_i{}^p)$ is a Loss function.

For predictions to have, the MSE value as minimum, we use gradient descent and update our predictions based on a learning rate.

$$y_i{}^p = y_i{}^p + \alpha * \delta \sum (y^i - y_i{}^p)_2 / \delta y_i{}^p$$

Where α is learning rate and,

$\sum (y - y_i{}^p)^2$ is sum of residuals.

F. Random Forest (RF):

Random Forest is an ensemble machine learning technique capable of performing both regression and classification tasks using multiple decision trees and a statistical technique called bagging. Bagging along with boosting are two of the most popular ensemble techniques which aim to tackle high variance and high bias. A RF instead of just averaging the prediction of trees it uses two key concepts that give it the name *random*:

1. Random sampling of training observations when building trees
2. Random subsets of features for splitting nodes

Clearly saying, Random forest builds multiple decision trees and merge their predictions together to get a more accurate and stable prediction rather than relying on individual decision trees.

Why it is better model?

The basic concept behind a random forest is to group the predictions made by many decision trees into a single model. Individually, predictions made by decision trees may not be accurate but combined together, the predictions will be closer to the true value on average. Each individual tree brings their own information sources to the problem as they consider a random subset of features when forming questions and they have access to a random set of the training data points. If we only build one tree we would only take advantage of their limited scope of information, but by combining many trees' predictions together, our net information would be much greater. If instead, each tree used the same dataset every tree would be greatly affected by the same way by an anomaly or an outlier. This increased diversity in the forest leading to more robust overall predictions and the name ‘random forest.’ When it comes time to make a prediction, the random forest regression model takes the average of all the individual decision tree estimates.

Evaluation Criteria:

All the regression models are tested and used for making prediction of required parameters. The following metrics were calculated for evaluating the performance of the developed system. Using all these metrices, we can decide that which one model is best suited for this project.

Mean Absolute Error (MAE):

It is used to measure the average magnitude of the errors in a set of predictions, without considering their direction. So, it gain the knowledge over the test sample of the absolute differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j|$$

Mean Squared Error (MSE):

It is knows as an estimator which measures the average of error squares. It is a risk function, respectively to the expected value of the squared error loss.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Root Mean Squared Error(RMSE):

Root mean squared error is nothing but scoring rule that also measures the average magnitude of the error. It is the square root of the average of squared differences between prediction and actual observation

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

R2 Score:

It is closely related to the **MSE.** It is defined the proportion of the variance in the dependent variable that is predictable from the independent variable(s).” A low value would show a low level of correlation, meaning a regression model that is not valid, but not in all cases.

R^2 is defined as follows,

$$SS_t = \sum_{i=1}^{m}(y_i - \bar{y})^2$$

$$SS_r = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

$$R^2 \equiv 1 - \frac{SS_r}{SS_t}$$

4. FAULT DIAGNOSIS

Fault Identification:

Sensor is a electronic device which is usually equipped with system to sense our surrounding environment. Sensor which is connected with IoT device for a special pupose to collect required information to perform a better action.Although,sensors are performing well but they are susceptible to faults too. It results as task failure in IoT devices. Indentification and detection of the faults is a major task in IoT and IIoT 4.0 should be performed on the system.

Here, we present a method that uses a fault tolerence with respect of model to successfully detect. We experiment our method by introducing our own algorithm with dht11 sensor which is used to measure the temperature and humidity of the environment uisng python programming and with this algorithm we determined the difference between each and every sequnce value of the temperature in real time using the temperature sensor.

ALGORITHM WORKING:

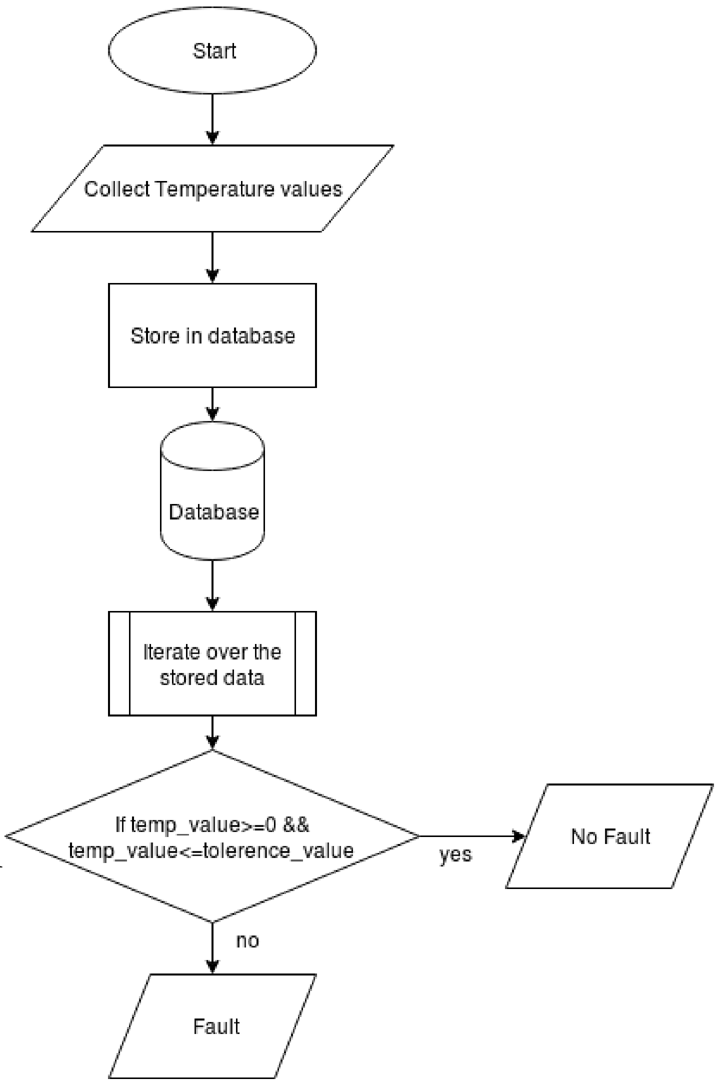

Start
Collect Temperature values
Store in database
Database
Iterate over the stored data
If temp_value>=0 &&
temp_value<=tolerence_value
yes
No Fault
no
Fault

Fig.2.4.1.FaultDetection Algorithm

IoT Security :

Security is one of the most important consideration that has to taken in care to protect the IoT devices from the attackers and hackers the main objective to preserve data privacy, confidentiality,users data should be safe and secure which is collected from various IoT devices and to make ensure the availablity of the service offered by the IoT ecosystem.

We experimented different types of attack over the IoT device(Raspberry Pi) by doing this experimentation on the devices we came to know that Iot devices are very vunerable.

COMPARISION OF DIFFRENT ATTACKS ON DEVICES:

There are different types of attacks which can be performed on IoT devices easily, the different types of attacks are :

- Worm Attack
- Malicious Attack
- DoS Attack
- MITM Attack
- SQL Injection

In this types of attack the most common attacks are:

1.Worm Attack:

Worm Attack is a type of attack which makes an adversary damage to the system by using malicious code or script.This code has ability to replicate to itself with any human action. Due to continuous execution of the code inside the IoT device which force the RAM(memory) to use 100% utilization, so that the other process which is executed in the IoT device will be stopped providing service because the RAM is fully trapped by the malicious script, so this pushed into the condition that there will be no availability of memory for the execution of the other process which has been requested before.
Then there is a other parameter called Central Processing Unit(CPU).Which is directly proportional to the RAM usage by analysing this two parameters.

In a nutshell, It clearly depicts that the particular IoT device has got attacked by worm attack.
We represented the graph below which shows the worm attack done and analysed by the python script.

2.Man in the Middle Attacks:

Man in middle Attack is a type of attack and biggest threaten in iiot 4.0,where iot device easily subjected towards this attack.which intercept between the two communicating device as a listener and mediater, who acts as a middle hidden server between the client and the server. This attack does not allow the actual client and actual server to connect with each other. And the hidden system will make a communication with the client and the server without there knowledge and act has a transmitter between them,it transmit the data packet from the client to actual server and transmit the data packet from the server to the client as doing this process along the side hidden system also saves the data which is getting transmitted between the actual and server,so it hijacks the data without there prior knowledge and permission,it just transmitte the data that the client and server should smell that some one prsent between the communication.

This attack is experimented in the IOT device(raseberry pi). This attack is done by simple logic called snoffing program(designed), it simply make the client ip requested address and server ip address to connect with the hidden computer and force to connect with it, by that hidden system by that we collected the data packet saved,official.

5. CONTRIBUTION

Water Quality Monitoring using Data Analytics, is a project which measures and monitors the dissolved contents in water using different IoT devices, which results us for discovering the parameters that affect the growth rate vital for aquatic lives.

The contribution made by us, Rishu Raj, Manendra Nath Shukla, Soumya Michael, V. Suriya is in the following direction.

We conceived of the presented idea by Dr. Munesh Singh. We designed the architecture model for reading of data from sensor and calculating the parameters as well as predicting the other values in cases such as damage of any sensor, or system malfunction. We also performed some experiments on Fault diagnosis.

We developed the theory for reading of data from DHT Sensor and storing it in the database. We investigated and supervised the findings of the work which resulted into ADAFruit package from which we can read the data from sensor.

We conceived of the presented idea, sending and receiving messages via CloudMQTT broker which works as the hosted publisher-receiver broker for IoT devices.

We wrote the codes for the following and deployed a database into cloud platform.

We, developed the flask web server application to publish the data online to database cloud server through which the access of the data could be made possible online from anywhere.

Formerly, we worked on cloud server, Pythonanywhere which offered the paid service. Later, Heroku cloud server was opted, due to the free service offered by them. We deployed the dataset received by the sensor and uploaded on server.

We designed a dataset from sources and came up with a dataset consisting of upto 40,000 sets of values for temperature, humidity, salinity, turbidity.

We conceived and planned the experiments for predicting the parameters which are not read by the data in case of system failure or malfunction. We experimented by applying different machine learning algorithms to perform regression. For prediction, different models were opted and accuracy rate was checked by each individual algorithms such as Linear regression, random forest etc.

We performed data analysis and summed up all the results, coming to the decision as Random Forest was giving the highest accuracy rate in our case.

We contributed to the fault tolerance section of the research, by evaluating the faults suffered by the system in cases of garbage values.

We performed the experiment and designed an algorithm which can derive a relation between time and temperature sensed by the DHT sensor.

We, contributed on the threats on IoT devices, in which we focused on the physical attacks implemented on the IoT devices. We designed an algorithm and implemented a worm attack on the raspberry pi. We discussed the results and tried several more attacks. However, bypassing the security of IoT devices was a challenging task for us.

We designed the algorithm as mentioned above and analysed the data, developed the theoretical framework.

CHAPTER 3

1. OBJECTIVES

- As we all know, water is contaminated with many chemicals and polluted particles. So, it is very important to continuously monitor the quality of the water at regular interval of time.
- Drinking contaminated water may cause many water-borne diseases like dysentery, typhoid fever cholera, trachoma, hepatitis etc. Therefore the water should be monitored.
- Abundant growth of algae and plant reduces the dissolved oxygen and other contain of water which cause death of fishes and other aquatic organism. So, water contents need to have appropriate amount of pH value, temperature, turbidity, DO, NH_3 content in it.
- In aquaculture, by introducing this system we can increase the production of healthy and good fishes. According to each aquatic organism we can provide proper environment for them.
- Also, we can provide the proper automatic system to monitor water for the production of gem quality pearls.
- 70% of India population depend upon agriculture, contaminated or polluted water may cause crop failure. So, introducing this system may lead to proper monitoring of water quality, to produce healthy crops.
- Therefore, it is crucial to measure and monitor the water quality for examining if water we are using is safe for human for good healthy life due to minerals content present in water, for aquatic organisms and for ecosystem.
- Through cloud we can easily store data and, can sends and receive message to and from client and server. Remotely we can control whole system, we can even calibrate our system remotely.
- We are using datasets, so in case of any sensor fault or failure we can predict value of that particular sensor. For this we trained our datasets and through machine learning predicted test data.
- Fault diagnosis helps to predict if sensor is giving correct value or fault value.

2. FUTURE SCOPE

Observing the current scenario, in future the aquaculture may undergo significant changes and may emerge into a fast growing industry. Freshwater aquaculture has resulted into tremendous growth in the past 15 years. As, we can see the statistics, many aquaculturists were practising extensive aquaculture but aquaculturists with semi-intensive operations benefited most from adoption of technology. The benefit-cost ratio for different systems of aquaculture varied between 1.22 to 1.86. The return to capital was much higher than the return to labor, due to the low labor input. As per growing population and demands, the need is also increasing proportionally, so aquaculture is yielding a much to the indusrtry.

In India, we are less likely to produce a gem quality pearl due to lack of monitoring and requirements needed. With the regular monitoring, we can produce a good number of gem quality pearls, by introducing this technology.

Water quality as always is a major concern for variety of fields whether health, industry, production, hospitals, etc. With this implemetation, monitoring of water quality and the minerals content would be easy and it can lead us to define the mineral content inside water present in different reservoirs, fisheries, hospitals, domestic uses, industrial water. And adopt different practices, in order to refine the water quality as per our needs. Consuming the proper amount of minerals through water intake will lead to a healthy nation ahead.

The industrial release of water are observed to be toxic in nature, which are treated before discharge into rivers, or water bodies. The purity of discharged water is somehow, a challenge as how much percentage of purity is attained with this treatment. Monitoring of water and its content can be made possible with this implementation.

As per growing technology and introduction of technology in each and every field nowadays, we observed as IoT is occupying almost all sectors. With the growth of IoT devices such as sensors and it’s uses, will result into fault into number of devices. With a knowledge of diagnosis and its solution, we can make IoT devices more robust and efficient. As per our research, and the diagnosis performed for various IoT devices, the results for diagnosing a fault for IoT devices may give us some information regarding the faults occurred in our sensors, and prevent us from analysing wrong data for the concerned technology.

Security for IoT devices has always been an issue in case of IoT devices. With the research performed by us in the field of security and attacks done on the IoT devices we can analyse the type of attack and design the security architecture in order to prevent the devices from various threats.

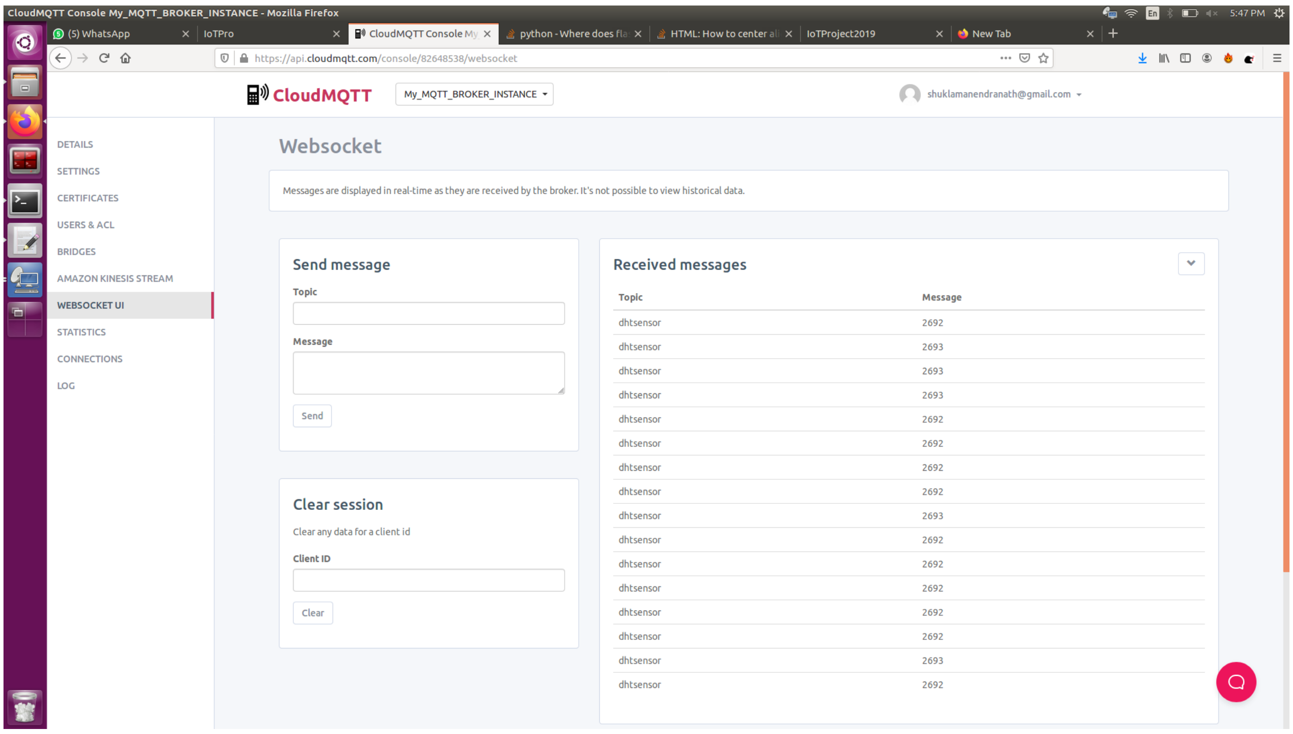
In our nation, we have some places which are debarred from technology and the internet is still an issue for their places. With the machine learning and data analytics technology, our computations done and the designed datasets may result into derivation of results in remote areas, in spite of the fact our online support can’t reach them.

Our system has technology with robust implementation, in cases like failure of any sensor, or misreaded values by the sensors, network problems or attacks performed on it, are tested and passed through almost every possible difficulties to be faced by them and implemeted with solutions which makes it all in one technology and fully safe and best implementation in water monitoring and deriving it’s benefits.

3. RESULTS & DISCUSSIONS

Here, we presents the results for the procedures and algorithms used in our project.

Cloud MQTT Broker



Websocket

Messages are displayed in real-time as they are received by the broker. It's not possible to view historical data.

Send message

Topic

Message

Send

Clear session

Clear any data for a client id

Client ID

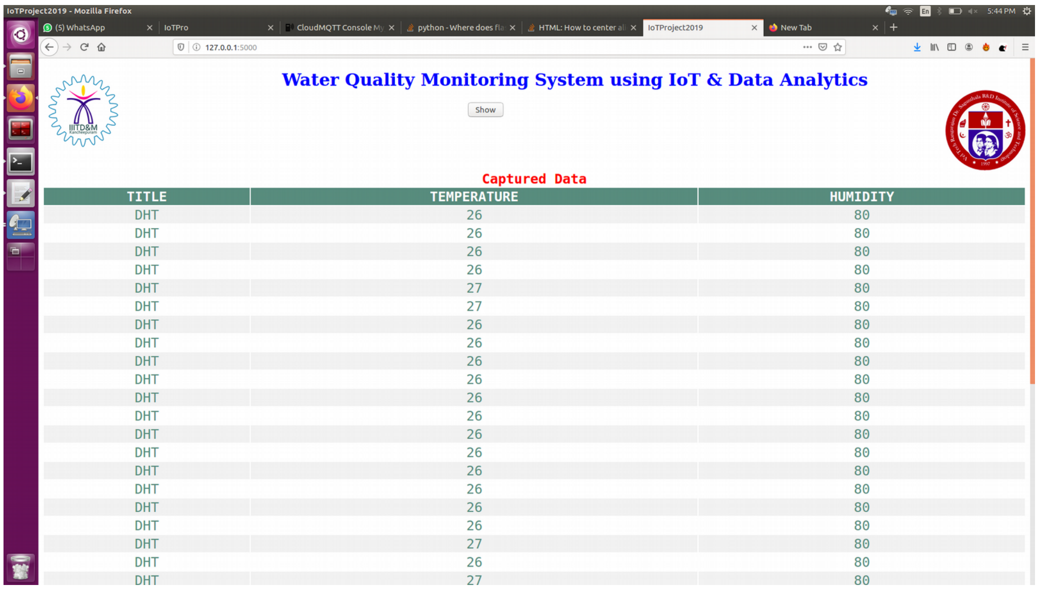
Clear

Received messages

Topic	Message
dhtsensor	2692
dhtsensor	2693
dhtsensor	2693
dhtsensor	2693
dhtsensor	2692
dhtsensor	2692
dhtsensor	2692
dhtsensor	2692
dhtsensor	2693
dhtsensor	2692
dhtsensor	2692
dhtsensor	2692
dhtsensor	2692
dhtsensor	2692
dhtsensor	2693
dhtsensor	2692

Fig. No. 3.3.1 Receiving of messages by Cloud MQTT Broker.

Heroku Cloud Deployment



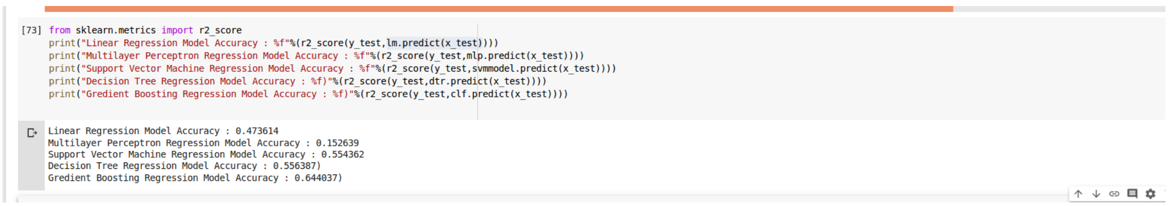
Water Quality Monitoring System using IoT & Data Analytics

Captured Data

TITLE	TEMPERATURE	HUMIDITY
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	27	80
DHT	27	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	26	80
DHT	27	80
DHT	26	80
DHT	27	80

Fig. No.3.3.2 Deployment of data on Cloud Heroku Server

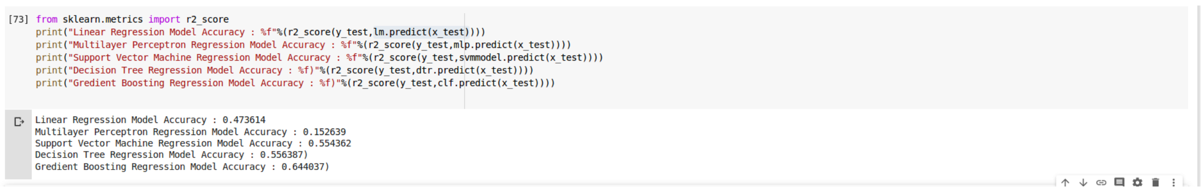
Comparison between different regression models



```
[73] from sklearn.metrics import r2_score
     print("Linear Regression Model Accuracy : %f"%(r2_score(y_test,lm.predict(x_test))))
     print("Multilayer Perceptron Regression Model Accuracy : %f"%(r2_score(y_test,mlp.predict(x_test))))
     print("Support Vector Machine Regression Model Accuracy : %f"%(r2_score(y_test,svmmodel.predict(x_test))))
     print("Decision Tree Regression Model Accuracy : %f)"%(r2_score(y_test,dtr.predict(x_test))))
     print("Gredient Boosting Regression Model Accuracy : %f)"%(r2_score(y_test,clf.predict(x_test))))
```

```
Linear Regression Model Accuracy : 0.473614
Multilayer Perceptron Regression Model Accuracy : 0.152639
Support Vector Machine Regression Model Accuracy : 0.554362
Decision Tree Regression Model Accuracy : 0.556387)
Gredient Boosting Regression Model Accuracy : 0.644037)
```

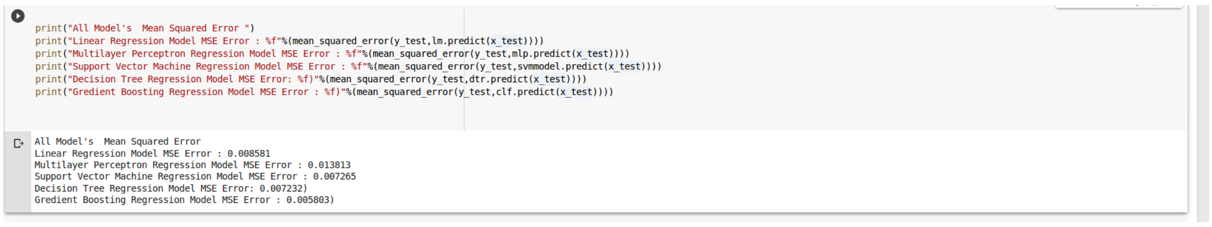
Fig. No. 3.3.3 R2 score of different regression models



```
[73] from sklearn.metrics import r2_score
     print("Linear Regression Model Accuracy : %f"%(r2_score(y_test,lm.predict(x_test))))
     print("Multilayer Perceptron Regression Model Accuracy : %f"%(r2_score(y_test,mlp.predict(x_test))))
     print("Support Vector Machine Regression Model Accuracy : %f"%(r2_score(y_test,svmmodel.predict(x_test))))
     print("Decision Tree Regression Model Accuracy : %f)"%(r2_score(y_test,dtr.predict(x_test))))
     print("Gredient Boosting Regression Model Accuracy : %f)"%(r2_score(y_test,clf.predict(x_test))))
```

```
Linear Regression Model Accuracy : 0.473614
Multilayer Perceptron Regression Model Accuracy : 0.152639
Support Vector Machine Regression Model Accuracy : 0.554362
Decision Tree Regression Model Accuracy : 0.556387)
Gredient Boosting Regression Model Accuracy : 0.644037)
```

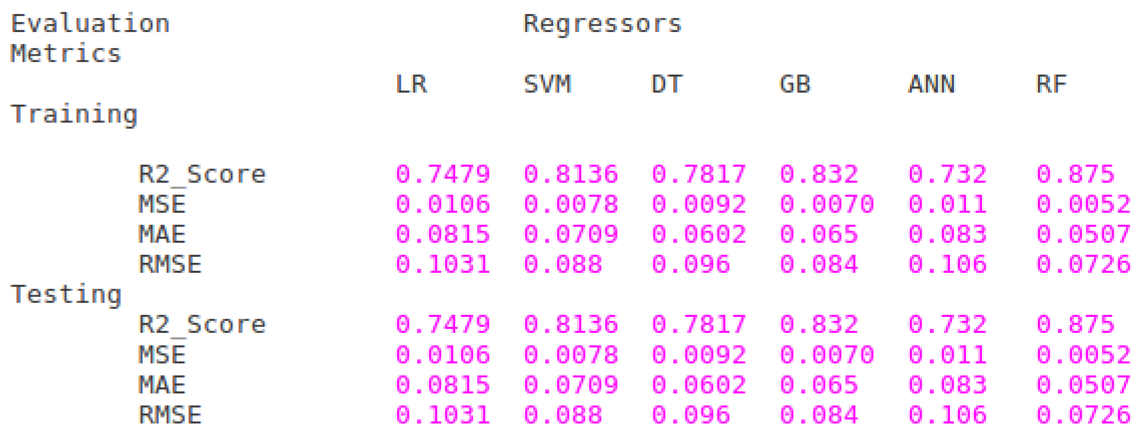
Fig. No. 3.3.4 Accuracy of different regression models



```
print("All Model's  Mean Squared Error ")
print("Linear Regression Model MSE Error : %f"%(mean_squared_error(y_test,lm.predict(x_test))))
print("Multilayer Perceptron Regression Model MSE Error : %f"%(mean_squared_error(y_test,mlp.predict(x_test))))
print("Support Vector Machine Regression Model MSE Error : %f"%(mean_squared_error(y_test,svmmodel.predict(x_test))))
print("Decision Tree Regression Model MSE Error: %f)"%(mean_squared_error(y_test,dtr.predict(x_test))))
print("Gredient Boosting Regression Model MSE Error : %f)"%(mean_squared_error(y_test,clf.predict(x_test))))
```

```
All Model's  Mean Squared Error
Linear Regression Model MSE Error : 0.008581
Multilayer Perceptron Regression Model MSE Error : 0.013813
Support Vector Machine Regression Model MSE Error : 0.007265
Decision Tree Regression Model MSE Error: 0.007232)
Gredient Boosting Regression Model MSE Error : 0.005803)
```

Fig. No. 3.3.5 RMS value of different regression models



Evaluation Metrics		Regressors					
		LR	SVM	DT	GB	ANN	RF
Training							
	R2_Score	0.7479	0.8136	0.7817	0.832	0.732	0.875
	MSE	0.0106	0.0078	0.0092	0.0070	0.011	0.0052
	MAE	0.0815	0.0709	0.0602	0.065	0.083	0.0507
	RMSE	0.1031	0.088	0.096	0.084	0.106	0.0726
Testing							
	R2_Score	0.7479	0.8136	0.7817	0.832	0.732	0.875
	MSE	0.0106	0.0078	0.0092	0.0070	0.011	0.0052
	MAE	0.0815	0.0709	0.0602	0.065	0.083	0.0507
	RMSE	0.1031	0.088	0.096	0.084	0.106	0.0726

Fig. No. 3.3.6 Tabulated value of all training and tested values.

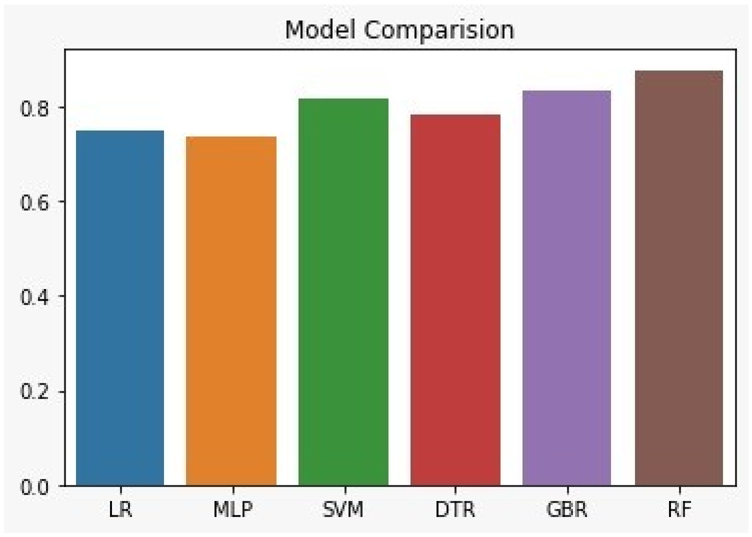


Fig. No. 3.3.7 Comparison between different models based in their accuracy.

Fault Diagnosis

As a result we tested dht11 sensor which is having the tolerence value of +/- 2 and withstand with in range(0,50).
So, as a result we concluded that there is no fault value detected, it shows that sensor is not damaged.
We represented the graph below which shows that the particular IoT device is not damaged.

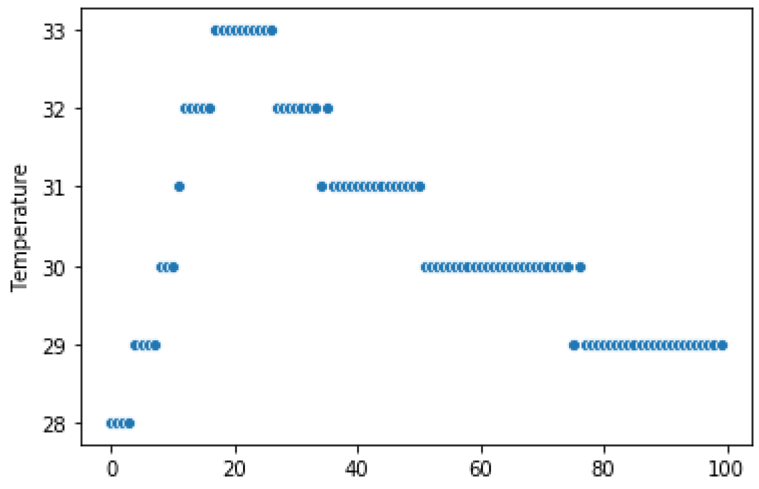


Fig. No. 3.3.8 Fault Diagnosis

Output:

```
initialising temp value
total number(values) reading :14
no fault 1.5
sensor dected correct value
no fault 0
sensor dected correct value
no fault 0.5
sensor dected correct value
no fault 0.5
sensor dected correct value
no fault 0
sensor dected correct value
no fault 1
sensor dected correct value
no fault 1
sensor dected correct value
no fault 1
sensor dected correct value
no fault 0
sensor dected correct value
no fault 1
sensor dected correct value
no fault 2
sensor dected correct value
no fault 1
sensor dected correct value
no fault 0
sensor dected correct value
```

Worm Attack

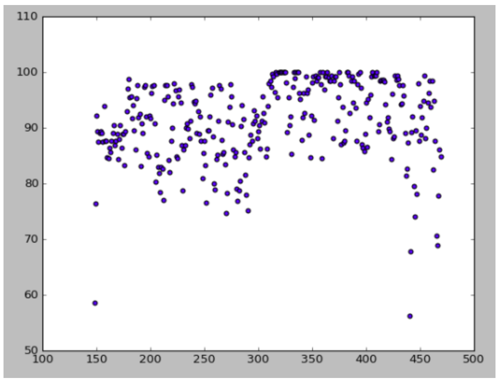


Fig No. 3.3.9 CPU usage

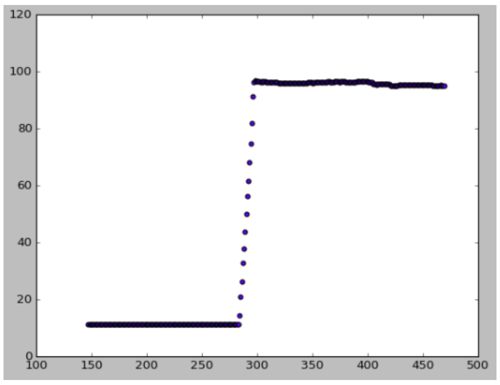


Fig. No. 3.3.10 Memory usage

4. CONCLUSION

Concluding, we have performed our research work in the direction of IoT and Water quality monitoring using data analytics, and we have came up with the solution which will help people monitor the quality of water in different areas such as domestic, hospitals, agriculture, aquaculture, industries and factories. It will provide a solution to a very common problem of regulating the mineral intake of water in aquaculture and providing an appropriate solution to a healthy growth to fisheries. It also comes with a robust solution which can also be efficient in cases like absence of internet and online solutions in remote areas. The prepared dataset with a huge amount of dataset will provide an upto accurate solution which can help people to perform agriculture as well as aquaculture in those areas with less interaction with internet. Our project has been tested with possible hardware as well as software faults. So, in those adverse situations too, the user can be alarmed with the diagnosis performed by the algorithm embedded into it. We have also performed some kinds of IoT attacks on our devices and tested the device robustness. Even, in cases such as sensor failure we have pre defined datasets with machine learning algorithms, which will not affect the device be affected with its values and predictions. However, more technology and modernisation will come up with more features and will make the system be enhanced with more kinds of features and increase the robustness of the project.

4. APPENDIX

Program code for running CloudMQTT (pub-sub)

```
import time
import paho.mqtt.client as mqtt
import sqlite3
mss=0

def on_connect(client,userdata,flag,rc):
    #print("connected with code"+str(rc))
    client.subscribe("dhtsensor")
    #print("Client subscribed")
def on_message(client,userdata,msg):
    global mss
    mss=msg.payload

client = mqtt.Client()


with sqlite3.connect("IIITDM.db") as db:
        cursor=db.cursor()
#cursor.execute("delete from mydata1;")
cursor.execute('''
    CREATE TABLE IF NOT EXISTS mydata1(
    temperature INTEGER(20) NOT NULL,
    humidity INTEGER(20) NOT NULL);
    ''')

client.on_connect = on_connect
client.on_message = on_message
client.username_pw_set("jczytaef","-xKcLL-mab9Q")
client.connect("tailor.cloudmqtt.com", 14837,60)

client.loop_start()
```

```
s=0
while s!=10:
    temperature=int(mss)//100
    humidity=int(mss)%100
    if(temperature!=0 and humidity!=0):
        cursor.execute("insert into mydata1 (temperature,humidity) values
(?,?)",(temperature,humidity))
    db.commit()
    #print(mss)
    time.sleep(2)
    s+=1

client.loop_stop()


cursor.execute("DELETE FROM mydata1 WHERE temperature IS
NULL;")

mydata=0
cursor.execute("SELECT * FROM mydata1 where temperature is not
null;")
mydata=cursor.fetchall()
print(mydata)
 #   print(i[0])
```

Program Code for Backend

```
from flask import Flask, render_template, request,redirect
import paho.mqtt.client as mqtt
import time
import sys
import os
import sqlite3
import subprocess
import re
#from connect import mydata
#The sensor is connected to GPIO17 on Pi
app = Flask(__name__)
```

```
#mess=""



@app.route("/", methods=['GET', 'POST'])
def d():

    #print(request.method)

    '''broker ="tailor.cloudmqtt.com"
    port = 14837
    username = "jczytaef"
    password = "-xKcLL-mab9Q"
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message'''
    result=temp=humidity=[]

    if request.method == 'POST':
        if request.form.get('Show') == 'Show':
            gg=getdata()
            result=re.findall(r'\d+',gg)
            temp=[result[i] for i in range(len(result)) if(i%2==0)]
            humidity=[result[i] for i in range(len(result)) if(i%2!=0)]

        #elif  request.form.get('ON') == 'ON':
            #print("ON")
         #       result+=on(broker,port,username,password,client)
            #return redirect(url_for('index'))
        else:
                # pass # unknown
            return render_template("index.html")
    elif request.method == 'GET':
            # return render_template("index.html")
        print("No Post Back Call")
                                                                return
render_template("index.html",temp=temp,humidity=humidity,l=len(temp))
```

```
def getdata():
    mydata=os.popen("python connect.py").read()
    #print(type(mydata))
    return mydata


def temp(broker,port,username,password,client,cursor,db):

    client.username_pw_set(username,password)
    client.connect(broker,port,60)



          #humidity,  temperature  =  Adafruit_DHT.read_retry(sensor_name,
sensor_pin)
    #print(mess)
    temperature,humidity=mess,5
      cursor.execute("insert  into  user  (temperature,humidity)  values  (?,  ?)",
(temperature,humidity))
    db.commit()

    #s=("Temperature: %f & Humidity: %f"%(temperature,humidity))
    #ret = client.publish("Status",ss)
    cursor.execute("SELECT * FROM user")
    #ss=cursor.fetchall()
    return cursor.fetchall()

def on(broker,port,username,password,client):
    client.username_pw_set(username,password)
    client.connect(broker,port)
    ret = client.publish("Status","Light glowing")
    return "Led Turned On"
```

```
if __name__ == '__main__':
    app.run()
```

Program Code for Machine Learning & Data Analaytics

Iot Project Machine Learning Implementation

In [0]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn import svm
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
import matplotlib.pyplot as plt
```

In [0]:

```
data = pd.read_csv("NEW.csv")
mydata=pd.DataFrame(data)
```

In [7]:

```
mydata.head()
```

Out[7]:

	Temperature	Salinity	DO %	PH	Turbidity
0	13.16	33.06	82.90	7.88	3.3
1	13.18	33.13	83.50	7.89	4.4
2	13.13	33.25	84.20	7.90	4.6
3	13.10	33.34	84.81	7.91	5.8
4	13.05	33.45	85.60	7.92	3.5

In [0]:

```
mydata = mydata.apply (pd.to_numeric, errors='coerce')
mydata = mydata.dropna()
```

In [9]:

```
mydata.shape
```

Out[9]:

```
(39988, 5)
```

In [10]:

```
np.any(np.isnan(mydata))
```

Out[10]:

```
False
```

In [11]:

```
np.all(np.isfinite(mydata))
```

Out[11]:

```
True
```

In [0]:

```
Y=mydata['PH']
x=mydata.drop('PH',axis=1)
```

In [0]:

```
x=x.drop('Salinity',axis=1)
```

In [14]:

```
x
```

Out[14]:

	Temperature	**DO %**	**Turbidity**
0	13.16	82.90	3.3
1	13.18	83.50	4.4

	Temperature	DO%	Turbidity
2	13.13	84.20	4.6
3	13.10	84.81	5.8
4	13.05	85.60	3.5
...	...	...	...
39993	16.30	92.20	15.3
39994	16.28	92.20	15.6
39995	16.30	92.20	17.6
39996	16.34	92.40	19.2
39997	16.39	92.90	17.3

39988 rows × 3 columns

In [15]:

```
mydata.describe().transpose()
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
Temperature	39988.0	23.628710	3.799761	5.17	21.84	24.42	26.32	30.98
Salinity	39988.0	23.665266	3.549520	12.37	21.22	22.64	27.03	34.03
DO%	39988.0	102.596056	26.736588	17.62	87.90	98.20	112.40	308.20
PH	39988.0	7.932014	0.203872	7.31	7.80	7.93	8.06	8.83
Turbidity	39988.0	4.371219	5.388780	0.10	2.40	3.17	4.25	149.20

In [16]:

```
Y.head()
```

Out[16]:

```
0    7.88
1    7.89
2    7.90
3    7.91
4    7.92
Name: PH, dtype: float64
```

In [0]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [0]:

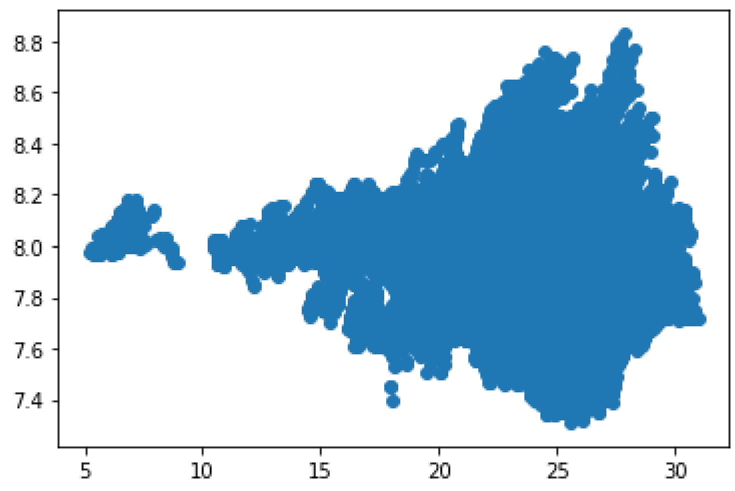
```
x_train,x_test,y_train,y_test=train_test_split(x,Y,test_size=0.25,random_state=5
)
```

In [19]:

```
phdata=mydata['PH']
temp=mydata['Temperature']
#plt.plot(temp, phdata, 'b-', label='ph Value')
#plt.plot(years, temp, 'g-', label='Temperature Value')
plt.scatter(temp, phdata)
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x7f337340e320>
```

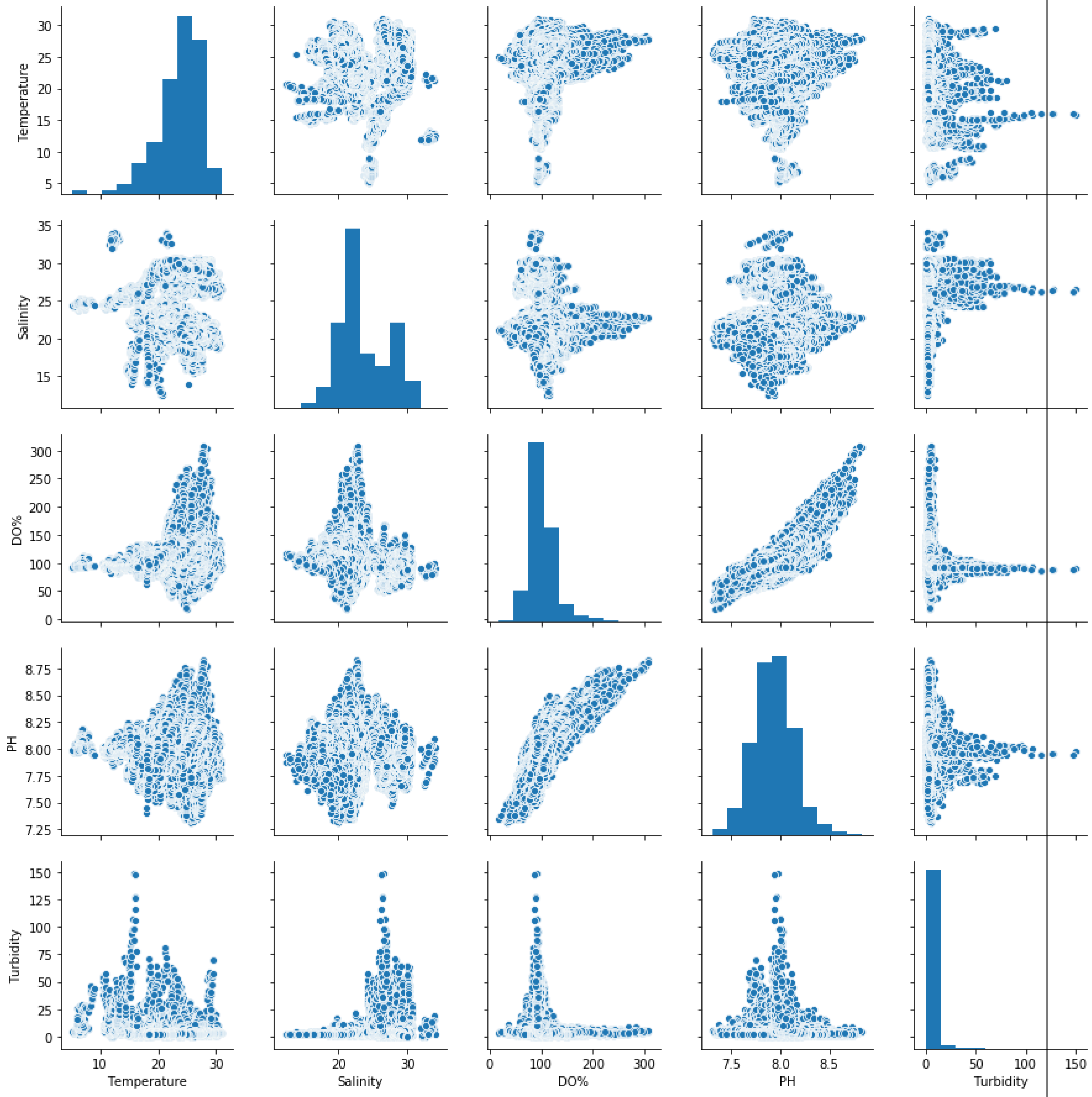


In [20]:

```
import seaborn as sns
sns.pairplot(mydata)
```

Out[20]:

```
<seaborn.axisgrid.PairGrid at 0x7f337349d160>
```



In [21]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(29991, 3)
(9997, 3)
(29991,)
(9997,)
```

In [0]:

```
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2,
          'learning_rate': 0.01, 'loss': 'ls'}
dtr=DecisionTreeRegressor(random_state=1)
svmmodel=svm.SVR()
clf = GradientBoostingRegressor(**params)

lm=LinearRegression()
rdf=RandomForestRegressor()
mlp = MLPRegressor(hidden_layer_sizes=(4,4,4), activation='relu', solver='adam', max_iter=500)
```

In [43]:

```
lm.fit(x_train,y_train)
mlp.fit(x_train,y_train)
clf.fit(x_train,y_train)
dtr.fit(x_train,y_train)
rdf.fit(x_train,y_train)
svmmodel.fit(x_train,y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:193: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[43]:

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
    gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
    tol=0.001, verbose=False)
```

In [24]:

```
from sklearn.metrics import r2_score,mean_absolute_error
```

```
print("Linear Regression Model Accuracy : %f"%
(r2_score(y_test,lm.predict(x_test))))
print("Multilayer Perceptron Regression Model Accuracy : %f"%
(r2_score(y_test,mlp.predict(x_test))))
print("Support Vector Machine Regression Model Accuracy : %f"%
(r2_score(y_test,svmmodel.predict(x_test))))
print("Decision Tree Regression Model Accuracy : %f)"%
(r2_score(y_test,dtr.predict(x_test))))
print("Gredient Boosting Regression Model Accuracy : %f)"%
(r2_score(y_test,clf.predict(x_test))))
print("Random Forest Regression Model Accuracy : %f)"%
(r2_score(y_test,rdf.predict(x_test))))
```

```
Linear Regression Model Accuracy : 0.747980
Multilayer Perceptron Regression Model Accuracy : 0.737079
Support Vector Machine Regression Model Accuracy : 0.813688
Decision Tree Regression Model Accuracy : 0.781713)
Gredient Boosting Regression Model Accuracy : 0.832190)
Random Forest Regression Model Accuracy : 0.874542)
```

In [25]:

```
print("All Model's  Mean Squared Error ")
print("Linear Regression Model MSE Error : %f"%
(mean_squared_error(y_test,lm.predict(x_test))))
print("Multilayer Perceptron Regression Model MSE Error : %f"%
(mean_squared_error(y_test,mlp.predict(x_test))))
print("Support Vector Machine Regression Model MSE Error : %f"%
(mean_squared_error(y_test,svmmodel.predict(x_test))))
print("Decision Tree Regression Model MSE Error: %f)"%
(mean_squared_error(y_test,dtr.predict(x_test))))
print("Gredient Boosting Regression Model MSE Error : %f)"%
(mean_squared_error(y_test,clf.predict(x_test))))

print("Random forest Regression Model MSE Error : %f)"%
(mean_squared_error(y_test,rdf.predict(x_test))))
```

```
All Model's  Mean Squared Error
Linear Regression Model MSE Error : 0.010648
Multilayer Perceptron Regression Model MSE Error : 0.011109
Support Vector Machine Regression Model MSE Error : 0.007872
Decision Tree Regression Model MSE Error: 0.009223)
Gredient Boosting Regression Model MSE Error : 0.007090)
```

```
Random forest Regression Model MSE Error : 0.005301)
```

In [26]:

```
print("RMSE")
print("Linear Regression: %f"%(np.sqrt(mean_squared_error(y_test,lm.predict(x_test)))))
print("Neural Network: %f"%(np.sqrt(mean_squared_error(y_test,mlp.predict(x_test)))))
print("Support Vector Machine: %f"%(np.sqrt(mean_squared_error(y_test,svmmodel.predict(x_test)))))
print("Decision Tree: %f"%(np.sqrt(mean_squared_error(y_test,dtr.predict(x_test)))))
print("Gradient Boosting: %f"%(np.sqrt(mean_squared_error(y_test,clf.predict(x_test)))))
print("Random Forest: %f"%(np.sqrt(mean_squared_error(y_test,rdf.predict(x_test)))))
```

```
RMSE
Linear Regression: 0.103191
Neural Network: 0.105399
Support Vector Machine: 0.088725
Decision Tree: 0.096037
Gradient Boosting: 0.084204
Random Forest: 0.072807
```

In [27]:

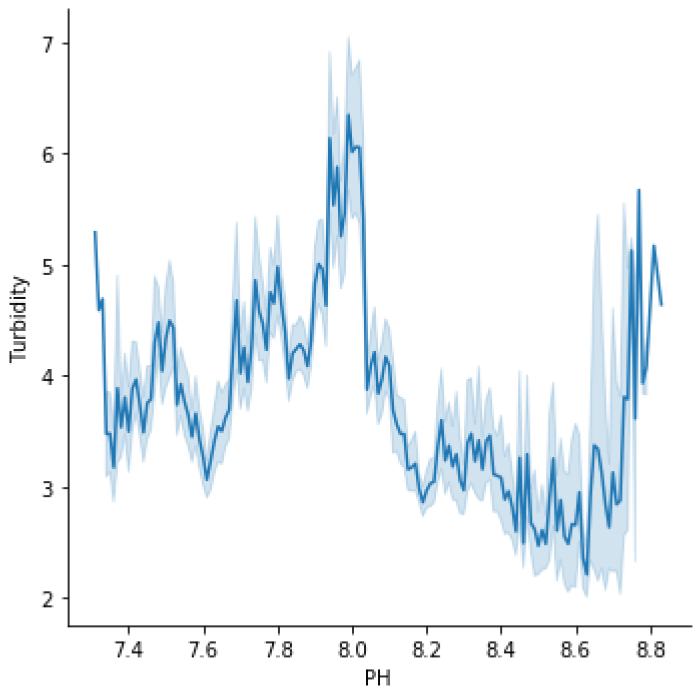
```
print("MAE")
print("Random forest Regression Model MSE Error : %f)"%(mean_absolute_error(y_test,rdf.predict(x_test))))
print("LR : %f)"%(mean_absolute_error(y_test,lm.predict(x_test))))
print("Gradient Boosting: %f)"%(mean_absolute_error(y_test,clf.predict(x_test))))
print("Decision Tree : %f)"%(mean_absolute_error(y_test,dtr.predict(x_test))))
print("Support Vector : %f)"%(mean_absolute_error(y_test,svmmodel.predict(x_test))))
print("Neural Network : %f)"%(mean_absolute_error(y_test,mlp.predict(x_test))))
```

```
MAE
Random forest Regression Model MSE Error : 0.050989)
LR : 0.081530)
```

```
Gradient Boosting: 0.065031)
Decision Tree : 0.060240)
Support Vector  : 0.070914)
Neural Network  : 0.082698)
```

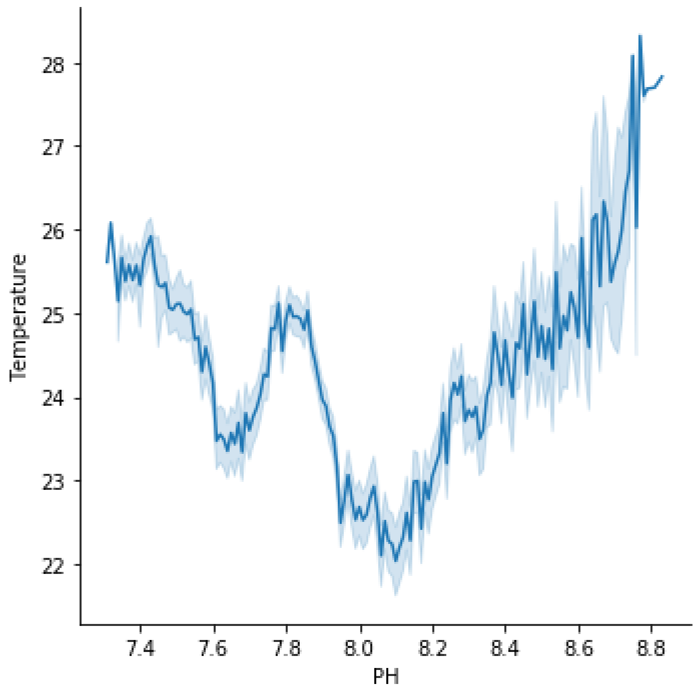
In [28]:

```
sns.relplot(x="PH", y="Turbidity", kind="line", data=mydata);
```



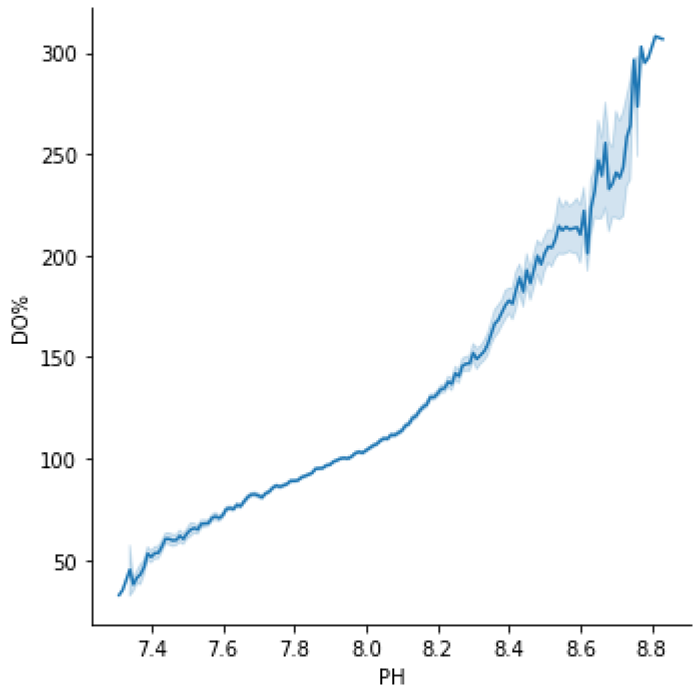
In [29]:

```
sns.relplot(x="PH",        y="Temperature",        kind="line",        data=mydata);
```



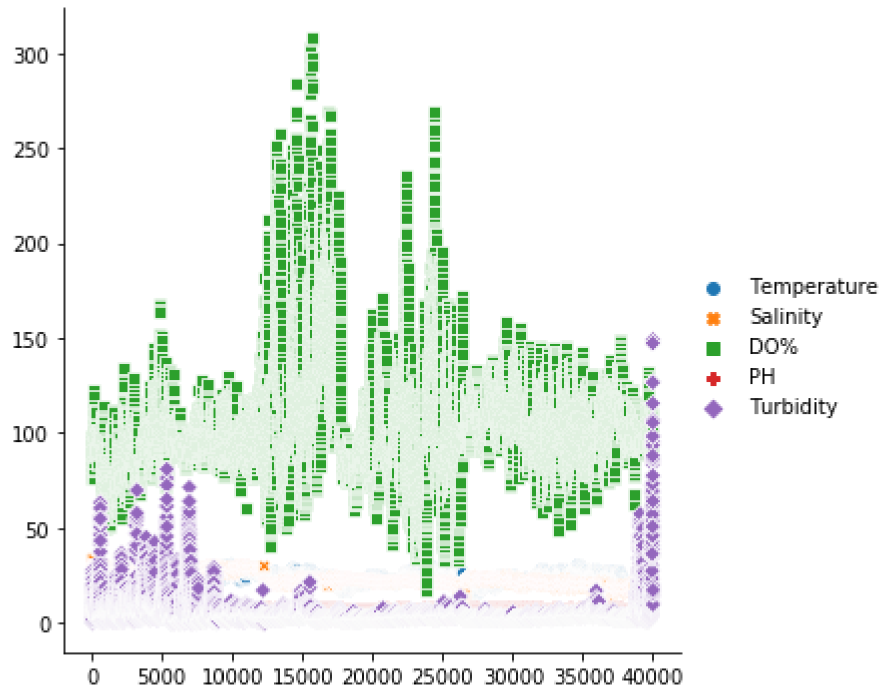
In [30]:

```
sns.relplot(x="PH", y="DO%", kind="line", data=mydata);
```



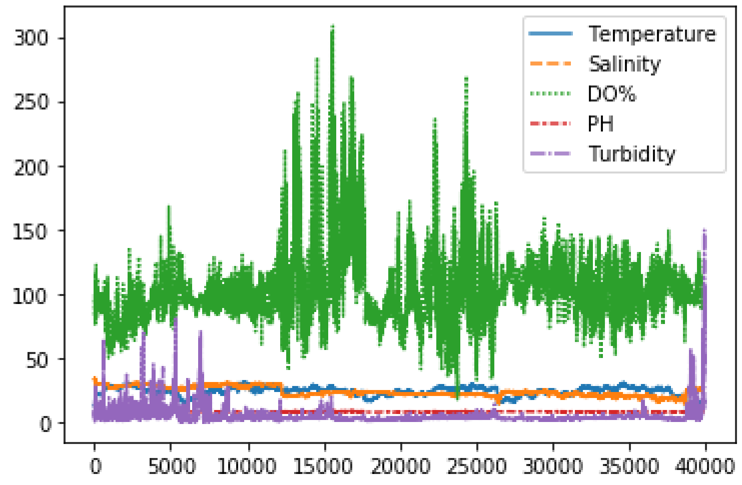
In [31]:

```
sns.relplot(data=mydata);
```



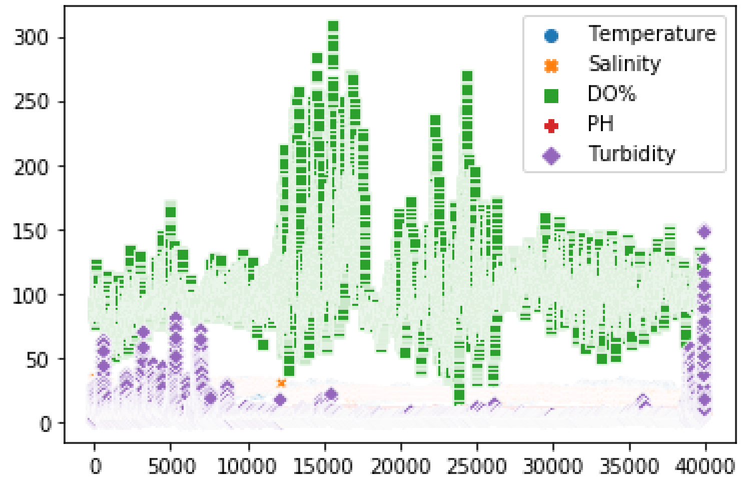
In [32]:

```
sns.lineplot(data=mydata);
```



In [33]:

```
sns.scatterplot(data=mydata);
```



In [0]:

```
from sklearn import model_selection
import seaborn as sns
```

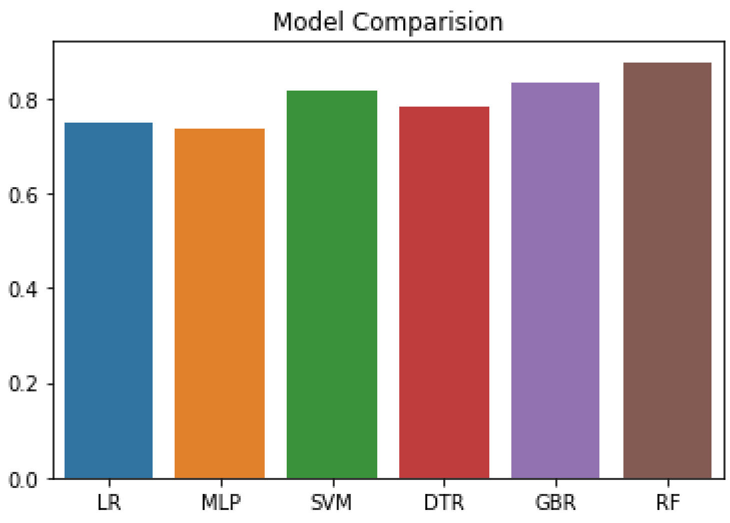
In [61]:

```
models = []
models.append(('LR', lm))
models.append(('MLP',mlp))
models.append(('SVM', svmmodel))
models.append(('DTR', dtr))
models.append(('GBR', clf))
models.append(('RF', rdf))
# evaluate each model in turn
results = []
names = []
```

```
for name, model in models:
        #kfold = model_selection.KFold(n_splits=10, random_state=seed)
        cv_results =r2_score(y_test,model.predict(x_test))
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)
# boxplot algorithm comparison
print(results)

ax=sns.barplot(names,results).set_title("Model Comparision")
```

```
LR: 0.747980 (0.000000)
MLP: 0.736195 (0.000000)
SVM: 0.813688 (0.000000)
DTR: 0.781713 (0.000000)
GBR: 0.832188 (0.000000)
RF: 0.876246 (0.000000)
[0.7479804441879938, 0.7361951783245889, 0.8136879304973647, 0.7817127082684153, 0.8321877678436286, 0.8762462931688467]
```



Program Code for Fault detection and Monitoring

```
#assumed temperature value based on sensor condition

print("initialising temp value")
temp_value=[26.5,13,25,25.5,26,26,25,26]
ln=len(temp_value)
s=0
print("total number(values) reading :"+str(ln))
for i in range(0,ln-1):
    if(temp_value[i]>50) or (temp_value[i]<0):
        print("fault not satisfied with sensor model condition")
            if  (abs(temp_value[i]-temp_value[i+1])<=1)  or  (abs(temp_value[i]-
temp_value[i+1])==0) or (abs(temp_value[i]-temp_value[i+1])<=2):
        s=abs(temp_value[i]-temp_value[i+1])

        print("no fault"+" "+ str(s))
    else:
         print("ALERT! SENSOR GOT DAMAGED, IT MAY RETURN FAULT
VALUE")
        break
```

Program Code for Worm Attack monitoring CPU usage.

```
import matplotlib.pyplot as plt
import numpy as np
import psutil
import time
plt.ion() ## Note this correction
fig=plt.figure()
plt.axis([0,1000,0,1])

i=0
x=list()
y=list()

while i <1000:
    temp_y=psutil.cpu_percent()
    mm=psutil.virtual_memory()[2]
    x.append(i);
    y.append(temp_y);
    plt.scatter(i, temp_y)


    i+=1;
    plt.show()
    plt.pause(0.0001) #Note this correction
```

Program Code for Worm Attack monitoring RAM usage.

```
import matplotlib.pyplot as plt
import numpy as np
import psutil
import time
plt.ion() ## Note this correction
fig=plt.figure()
plt.axis([0,1000,0,1])

i=0
x=list()
y=list()
z=list()

while i <1000:
    temp_y=psutil.cpu_percent()
    mm=psutil.virtual_memory()[2]
    x.append(i);
    y.append(mm);
    plt.scatter(i, mm)


    i+=1;
    plt.show()
    plt.pause(0.0001) #Note this correction
```

Program Code for Worm Attack malicious script

```
l=[]
c=1
for i in range(1000000000):
    c=c**i
    l.append(c*5)
```

5. REFERENCES

[1] M N Barabde, S R Danve, “Continuous Water Quality Monitoring System for Water Resources at Remote Places”, International Journal of Engineering Research and General Science, Vol.2, no.3, Part 2 , May-June 2015, ISSN 2091-2730.

[2]X. Shen, M. Chen, J. Yu, "Water environment monitoring system based on neural networks for shrimp cultivation", *Proc. Int. Conf. Artif. Intell. Comput. Intell.*, pp. 427-431, 2009.

[3]Te Unmanned Autonomous Cruise Ship for water quality monitoring and Sampling
Jia Shuo Zhang YonghuiTong Kebi 2017 transaction on Computer Systems, Electronics and Control vol.2, no.5, Part 1 , March-July 2017, ISSN 2055- 2744

[4]Design of Monitoring Systems for Rural Drinking Water Source Based on WSN
exin LinWeixiZng ; Huili Yin ; Sheng Jiang ; Guohui Jiao ; Jieping Yu
2017 vol.4,no.3,part 2 on Computer Network, Electronic and Automation 2017