Water Scarcity Prediction and Supply Scheduling

Submitted in partial fulfillment of the requirements
Of the degree of

B. E. Computer Engineering

By

Swapnil Verlekar	**Roll No. B-74**	**PID 162127**
Alisha Shah	**Roll No. B-63**	**PID 172251**
Nigel Martis	**Roll No. B-40**	**PID 162069**

Guide

Ms. Snehal Kulkarni
Assistant Professor

Department of Computer Engineering
St. Francis Institute of Technology
(Engineering College)

University of Mumbai

2019-2020

CERTIFICATE

This is to certify that the project entitled **“Water Scarcity Prediction and Supply Scheduling”** is a bonafide work of **“Swapnil Shrikrishna Verlekar (A-74), Alisha Hasmukh Shah (A-63), Nigel Savio Martis (A-40)”** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of B.E. in Computer Engineering.

(Ms. Snehal Kulkarni)
Guide

(Dr. Kavita Sonawane)
Head of Department

(Dr. Sincy George)
Principal

Project Report Approval for B.E.

This project report entitled ***“Water Scarcity Prediction and Supply Scheduling”*** by ***Swapnil Verlekar, Alisha Shah, Nigel Martis*** is approved for the degree of ***B.E. in Computer Engineering.***

Examiners

1.--

2.--

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Swapnil Verlekar, A-74, 162127)

(Alisha Shah, A-63, 172251)

(Nigel Martis, A-40, 162069)

Date:

Abstract

Today, water resource management is one of the major problems we are facing. Presently manual analysis to meet the need of water demand has been practiced. India is very dependent on rainfall through dams as a fresh water resource. This leads to high uncertainty in water supply. In India, one of the cities that is badly affected by water scarcity is Chennai. Nevertheless, in most urban areas, a decent water supply is followed after a scarcity is noticed, but this lacks in rural areas. This project is based on developing a system for an efficient and accurate Water Scarcity Prediction for Chennai. This system simplifies data management, enhances the ability to diagnose scarcity, assists in preventing supply/scheduling errors, and improves operational efficiency. It can also play a pivotal role in knowing about areas with abundance of water resource. This system majorly focuses on automating a Scarcity predictor and providing an accurate visualization graph scheduling a supply from nearby abundant water resources. This system is designed to create a model based on real-time data, which in turn would contribute to most of its accuracy. It eliminates the need to track manual analysis of data to predict the upcoming scarcity if any. Based on the analysis of the data set, it is certain that rainfall in Chennai greatly affects the water level in the resources. The system has been developed using a Random Forest Regressor to create a model at the backend.

Contents

List of Figures

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
i	ANNs	Artificial neural networks
ii	SVM	Support Vector Machines
iii	MLP	Multi-Layer Perceptron
iv	CNN-Bi-LSTM	Convolutional Neural Networks-Bidirectional- Long Short Term Memory
v	ANG	Artificial Neural Genius
vi	SVG	Support Vector Genius
viii	SGD	Stochastic Gradient Descent
ix	BVP	Boundary Value Analysis
x	RBF	Radial Basis Function

Chapter 1

Introduction

Traditional methods for water shortage treatment include manually working after it has occurred. This method of treatment is not very useful and it has disadvantages such as- it is very difficult to treat scarcity after it has occurred. In recent years, Machine Learning has emerged to be useful in prediction. In our project, we will mainly focus on predicting water shortage and scheduling a supply between the reservoirs so that each reservoir has ample quantity of water.

1.1 Description

In today's world everything is becoming computerized and automated. Different organizations have already moved towards computerized systems which made lives easier and faster. One of the most important sectors of any developing nation is the water availability and demand. The organization of people, institutions and resources that deliver services related to water supply to meet the needs of the general public or any individual is referred to as Municipal Corporation (BMC).

Water shortage can mean scarcity in availability due to physical shortage, or scarcity in access due to the failure of institutions to ensure a regular supply or due to a lack of adequate infrastructure. The water shortage already affects every continent. Water use has been growing globally at more than twice the rate of population increases in the last century, and an increasing number of regions are reaching the limit at which water services can be sustainably delivered, especially in arid regions. Water shortage will be exacerbated as rapidly growing urban areas place heavy pressure on neighboring water resources. Climate change and bio-energy demands are also expected to amplify the already complex relationship between world development and water demand. There is not a global water shortage as such, but individual countries and regions need to urgently tackle the critical problems presented by water stress. Water has to be treated as a scarce resource, with a far stronger focus on managing demand. Integrated water resources management provides a broad framework for governments to align water use patterns with the needs and demands of different users, including the environment.

1.2 Problem Formulation

In order to treat water shortage, manual examination of scarcity is often done. This could prove to be an erroneous solution as it is not efficient to treat water shortage after it has occurred. An Automated system that predicts shortage and schedules the supply is beneficial as it works on the problem before it has occurred as well as doesn’t require a lot of manual working analysts. Chennai is a city that is often subjected to water shortage and it is really necessary to analyze and come up with a solution to quench the thirst of people in Chennai. Since all the data is available in the public domain, would it be possible to do some analysis and see whether we can predict scarcity and estimate this water shortage ahead of time so as to plan for it?

1.3 Motivation

The Municipal Corporation has to manually react to water shortage after it has occurred. This testing includes observing the various factors and then deciding on where scarcity has occurred. An automated system which detects water shortage faster and also efficiently when compared to manual scarcity detection is very much needed. In Chennai, the water shortage of 2004 has brought Veeranam Lake as the new means of water supply for the city. Hopefully, this current scarcity (July 2019) will bring more additional sources of water for the ailing city. The city has grown a lot in the last 15 years and so need additional water resources to manage the needs. The city needs to devise better scarcity control methods by estimating the needs ahead of time. And for now,

"Only RAIN can save the city!"

1.4 Proposed Solution

The data plays an important role in the efficiency and accuracy of the model. The proposed solution is, using a supervised learning algorithm which is a Random forest algorithm. The input parameters would be the factors affecting water shortage, i.e. Rainfall, population, water consumption and available water resources. Random forest regression algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. As and when water shortage is predicted, it is displayed. A relation between population and water consumption in a particular reservoir is created from which an ideal amount of water required by that reservoir is derived. All regions in Chennai are mainly dependent on four major reservoirs. Accordingly, a minimum threshold of water required for each individual water resource is deduced. If a particular resource exceeds above the threshold, that additional portion of water can be supplied to a scarcity affected reservoir. Similarly, if one resource has contributed its share of water to one reservoir and if that reservoir is still suffering from scarcity, other resources with abundance of water will contribute for the same. Finally, a diagnosis is conducted and a report is displayed.

1.5 Scope of project

Currently (July 2019), Chennai is facing an acute water shortage due to shortage of rainfall for the past three years (and we had one of the worst floods in history the year before that!). As a result, the water in these resources are depleting along with the groundwater level. Accordingly, this system can be expanded to several other cities and states. Also, analysts can be accurate but also equally expensive; using an automated system eases this process to a very good extent. Availability of analysts is also another criterion, as this system would be available as long as a system specified is available with the software. Remote locations can just have a computer to predict water shortage. This reduces the stress of an Analyst to classify, as our system does it, a final verdict of an analyst is also needed.

Chapter 2

Review of Literature

Two machine learning techniques have been investigated in this paper. These are the artificial neural networks (ANNs) and the support vector machines (SVMs). An approach adopted was to conduct two parallel experiments, one for the ANNs and one for the SVMs. The ANN experiment encapsulated two architectures, the multi-layer perceptron (MLP) and the radial basis function (RBF). The results from the two architectures were compared to come up with the Artificial Neural Genius (ANG). The SVM experiment consisted of many models with different kernel functions and some of these kernel functions had additional arguments such as the degree and the scale. These models were compared against each other in order to determine the Support Vector Genius (SVG). The performance criteria used to determine the geniuses from each experiment were the validation error and the accuracy in their approximation of the target values of the validation data set. The two geniuses were then compared against each other in order to determine the overall genius (OG). The performance parameter used to determine the OG is the generalization ability of each genius. The ANG has proved to outperform the SVG. [1]

Actual in accordance with relevant historical data, neural network model of the process, to train the hidden layer nodes to achieve a dynamic selection of urban water consumption forecast. The experimental results we can see that the process of neural network model can predict the city's water has a certain theoretical and practical value. [2]

The dramatic growth of installed capacity of solar power plants increases their impact on the operation of bulk power systems, and therefore new challenges do arise, in particular the problems of electric energy generation forecasting. Estimation of the possible output of the solar power plant in short-term perspective is mainly based on forecasting of weather conditions for the area under consideration, taking into account the probabilistic nature of the forecasted values. The paper addresses the problem of short-term renewable energy forecasting. The stochastic nature of renewable energy sources affects the power system planning procedures, subsequently reducing the reliability and security of power supply for final customers. The authors propose a machine learning approach for effective day-ahead forecasting using retrospective metering data and open source weather information provided by meteorological services. The paper addresses the problem of feature identification and appropriate error metrics. For the purpose of effective testing the user interface was created and tested on the real solar power plant situated in southern region of Russian Federation. Within the framework of the described model, it is proposed to use a random forest to implement the function with solar power plant output forecasting. Random forest is a machine learning algorithm, using an ensemble of decision trees. The algorithm combines two main ideas: the bagging method and the random sub-space methodology. The algorithm is typically used for classification, regression and clustering. The main idea is to use a large ensemble of decision trees, each of which a very low quality classification result, but due to their large number the result is becoming sufficiently accurate. It can be seen from the correlation matrix, the, obviously, solar irradiation provides the greatest correlation with solar power plant energy output with the correlation factor of 0.9. However, one should take into account that only a limited set of features can be taken into account in short-term solar power plant prediction system. Solar power plant energy output prediction model is implemented for day-ahead forecasting and uses day-ahead weather prediction data, provided by open source meteorological services. Typically, such weather data providers do not predict solar irradiation, mainly due to the complexity of cloudiness identification (height of clouds, type of clouds, etc.). For this reason, this feature cannot be taken as an input one. The other features with the correlation factor exceeding 0.5 are included into the prediction model. The mean forecasting accuracy was obtained to be about 93%. [3]

In this paper, the CNN-Bi-LSTM hybrid model is proposed by CNN, it can effectively extract the inherent characteristics of historical water consumption and meteorological data. And Bi-LSTM can fully reflect the long-term historical process and future trend. CNN-Bi-LSTM model combines the advantages of CNN and Bi-LSTM models as well as the nonlinear fitting ability of the Bi-LSTM to the time series model. The prediction accuracy of urban water forecast was improved compared with these single models. The accuracy of prediction in the CNN-Bi-LSTM model is 1%~2% higher than the LSTM model's accuracy, and the accuracy of prediction was improved by 2%~3% compared with the CNN model. Compared with the Bi-LSTM model, the accuracy of prediction was improved by 0.6%~1.5%. Therefore, this paper analyses the factors that have a greater impact on urban water consumption. It was found that the daily maximum temperature and the historical water consumption information of the past five days have the highest correlation with urban water consumption. Establishing a Temperature Correction Model and Holiday Correction Model, which are based on statistical analysis, can reduce the impact of climate change and holidays on urban water demand forecasting. After the holiday correction, the prediction accuracy can be improved by 2%~5%. After the temperature correction model, the prediction accuracy can be improved by 1%~3%. Training the CNN-Bi-LSTM model takes 129.12 seconds less than the LSTM model and CNN model. For convergence, the CNN-Bi-LSTM model is trained 125 times, which is less than the LSTM and CNN in training times. In conclusion, the validity of the correction model was proved. It was proved that the CNN-Bi-LSTM model has better prediction accuracy and stability than other single models. [4]

Chapter 3

System Analysis

We specify the functional and non-functional (implicit and explicit) requirements of the system in this chapter. We show the interaction of the system and the users is illustrated using use case diagrams.

3.1 Functional Requirements

1. System must prompt user to enter year for which they are expecting to get the prediction result.
2. System must prompt the user to enter rain in mm for 5 months viz June, July, August, September and October.
3. On entering the values for rain and year for prediction, the user must have an option for getting prediction for different reservoir in Chennai.
4. System must show user prediction result based on the reservoir he/she has selected.
5. System must show user the graphs required for making decisions and scheduling details

3.2 Non-Functional Requirements

1. System must be designed to handle multiple requests at single instance
2. System must allow to up-date the trained model without switching off the server, it must also allow to dynamically train the model without shutting down the server.
3. System must be designed in a less coupled manner such that it can be further enhancement can be done quickly.

3.3 Specification Requirements

1. 8 GB RAM
2. 4 GB GTX 9 Series or higher GPU

3. Quad Core Intel/AMD CPU
4. Browser: Chrome/Opera
5. Storage: Around 50 MB
6. Python : Preferably Python 3.6/3.7 due to availability of libraries
7. Libraries : Keras, Numpy, Pandas, SciKit, Pillow.
8. OS: Windows/ Ubuntu
9. Frontend: HTML, CSS, Javascript
10. Backend IDE: Pycharm
11. Frontend IDE: visual studio code

3.4 Use-Case Diagrams and description

In this section, use-case to depict the users interacts with the system and each other is discussed.

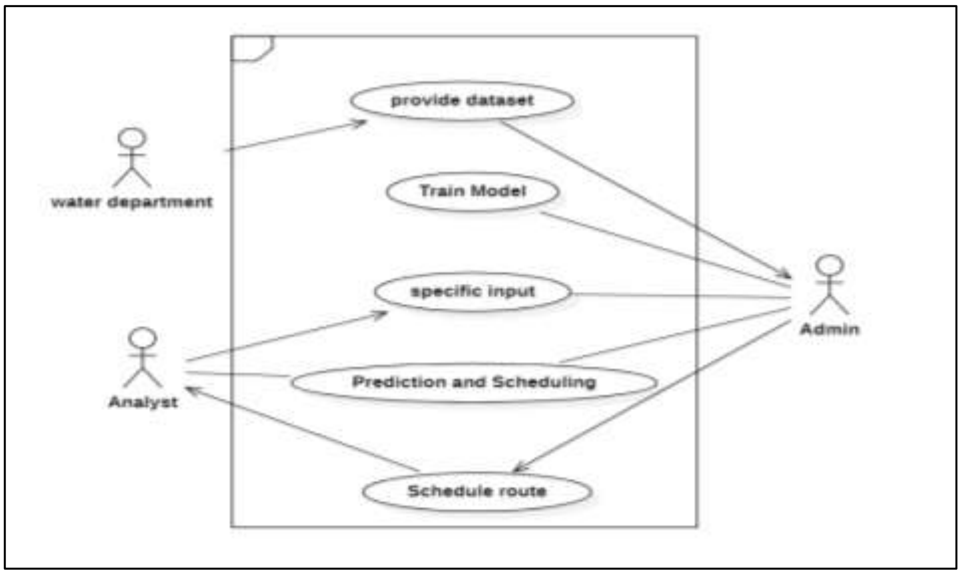


Fig 3.1 Use Case Diagram for system

The actors involved in the system are the Analyst, Admin and the Water Department. The Admin acquires real time data from the Water Department and creates a model. The analyst further uses the model for prediction and generates a schedule if scarcity has occurred

Chapter 4

Analysis Modeling

Analysis modeling deals with modeling and representation of data. In this chapter, we design different models in which information, functions and the behavior of the system is defined and these are translated into the design.

4.1 Class Diagram:

The Class Diagram for the system is discussed in this section.

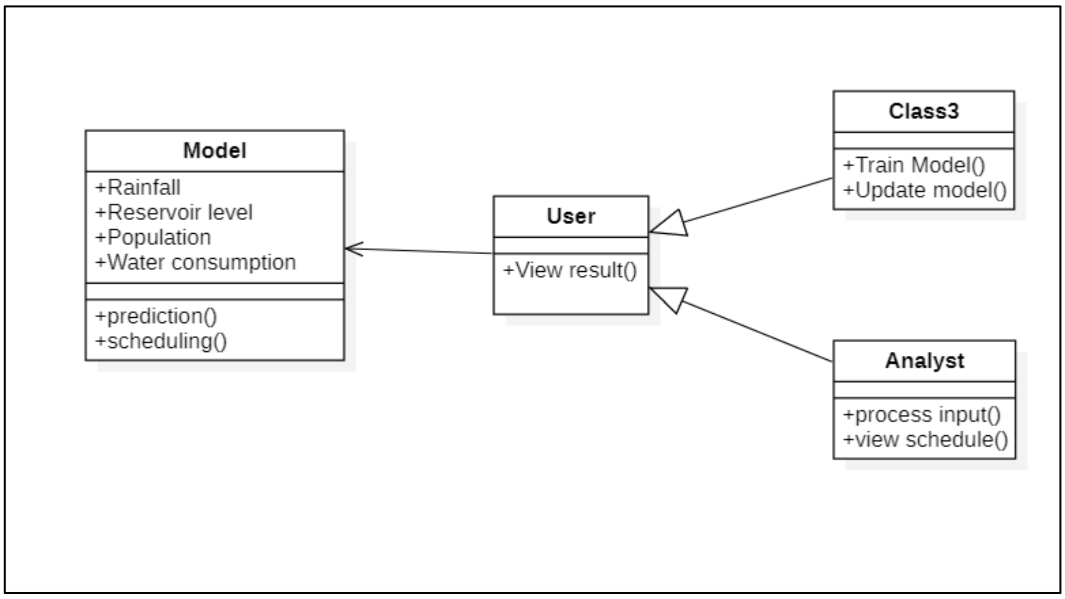


Fig 4.1 Class diagram of system development.

In this class diagram, dataset containing the various features is provided for model building. There are two types of users Admin-controls overall model building and updates model time to time, Analyst-generates results (schedule) for specific cases. Mainly prediction and schedule generation are prime focus, prediction of reservoir level and scheduling for minimum cost and optimal transportation route.

4.2 Activity Diagram:

The Activities performed due to the interaction between the users and the system are discussed in this section.

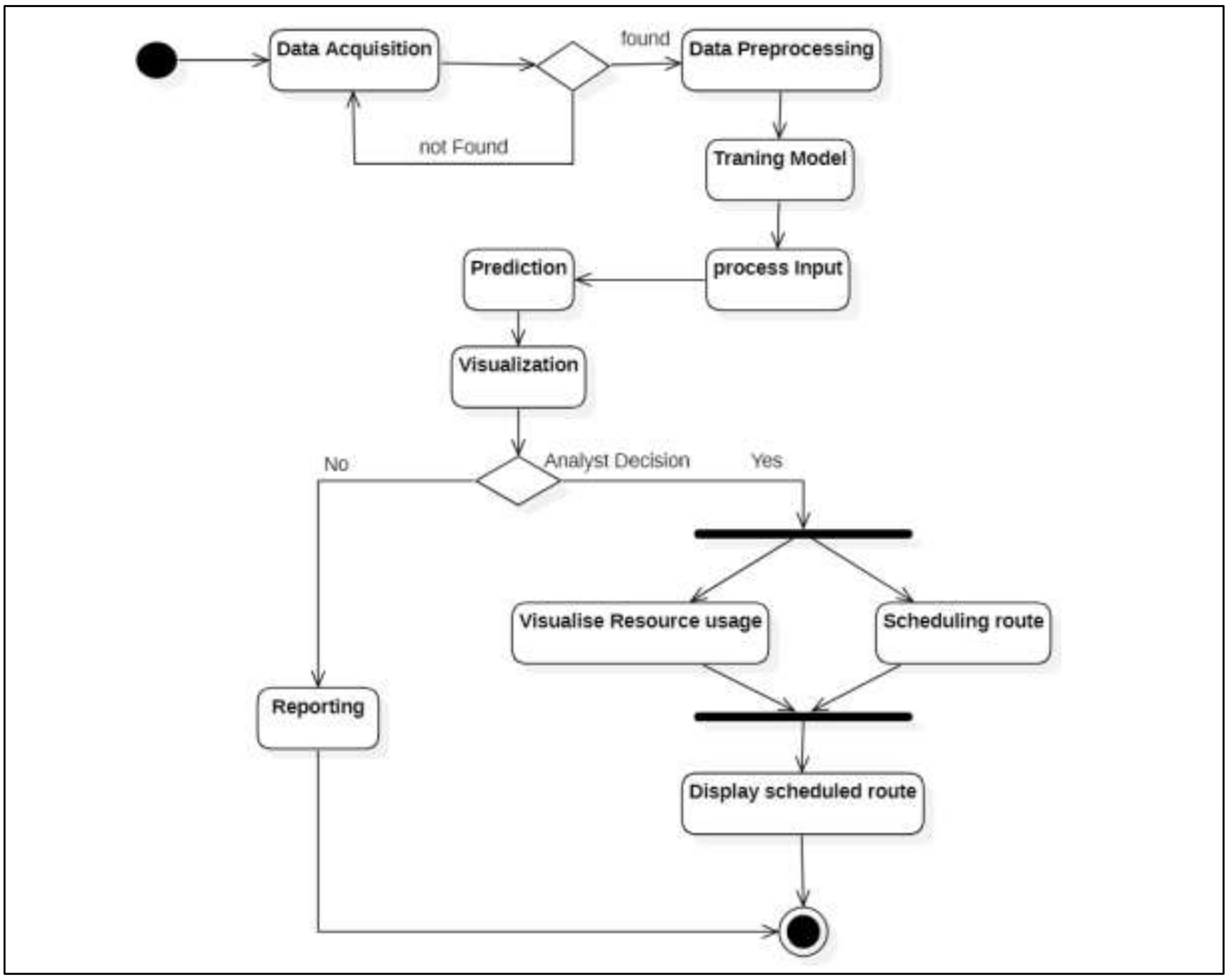


Fig: 4.2 Activity Diagram for prediction and scheduling.

Dataset with the necessary parameters are acquired and merge to get a uniform processed data, enabling us to make productive analysis and get the accuracy variance based on the feature selection. The processed data is than, used to train various models for each reservoir. The system takes values of rainfall in each month of rainy season for a particular reservoir and predicts the reservoir level for the consecutive months. Bar graphs provide better visualization of water level status. Analyst makes decision considering the status and further reporting is done. Visualization of resource usage is given to maintain appropriate level of water for the selected particular reservoir. Scheduling of route is also done for each reservoir from the selected reservoir selecting the best route for transport.

4.3 Functional Modeling:

The flow of data in the system is represented by the DFD diagram given in this section.

4.3.1 DFD (level-0)

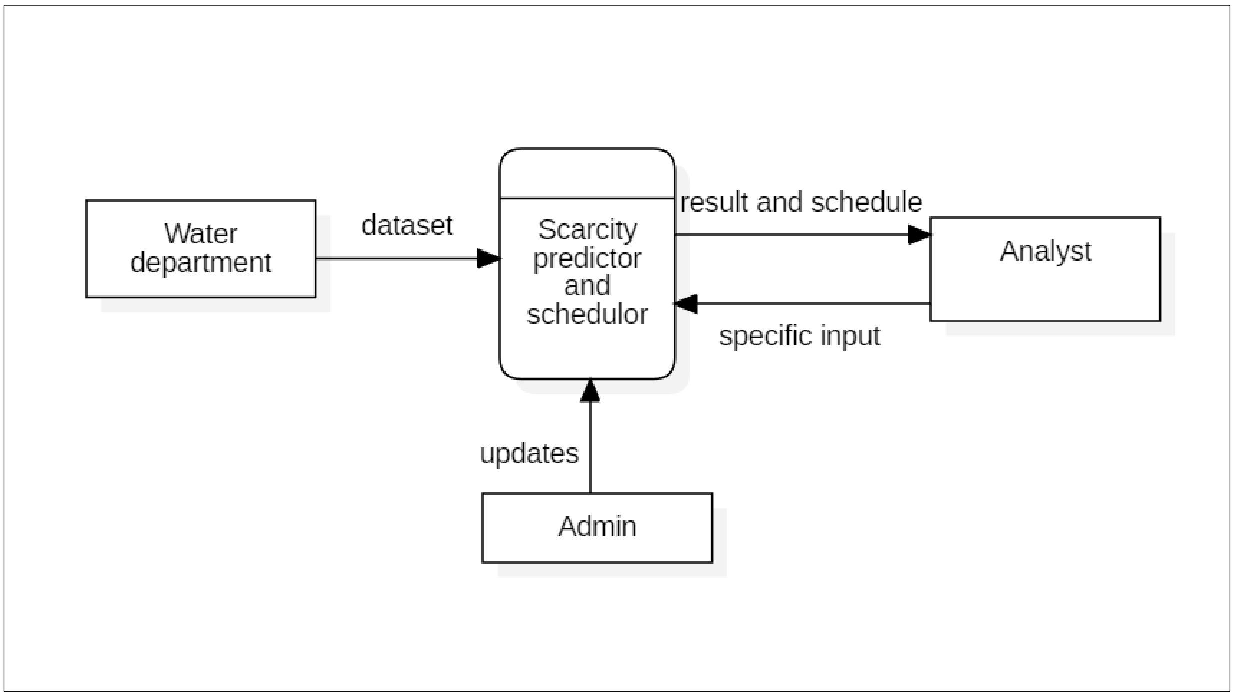


Fig 4.3 DFD for level-0

A data flow diagram (DFD) maps out the flow of information for any process or system. A level 0 data flow diagram (DFD), also known as a context diagram, shows a data system as a whole and emphasizes the way it interacts with external entities. The diagram above is level-0 DFD for our system depicts the major process involved with entities namely Water department, Model, Admin and Analyst.

4.3.2. DFD (level-1)

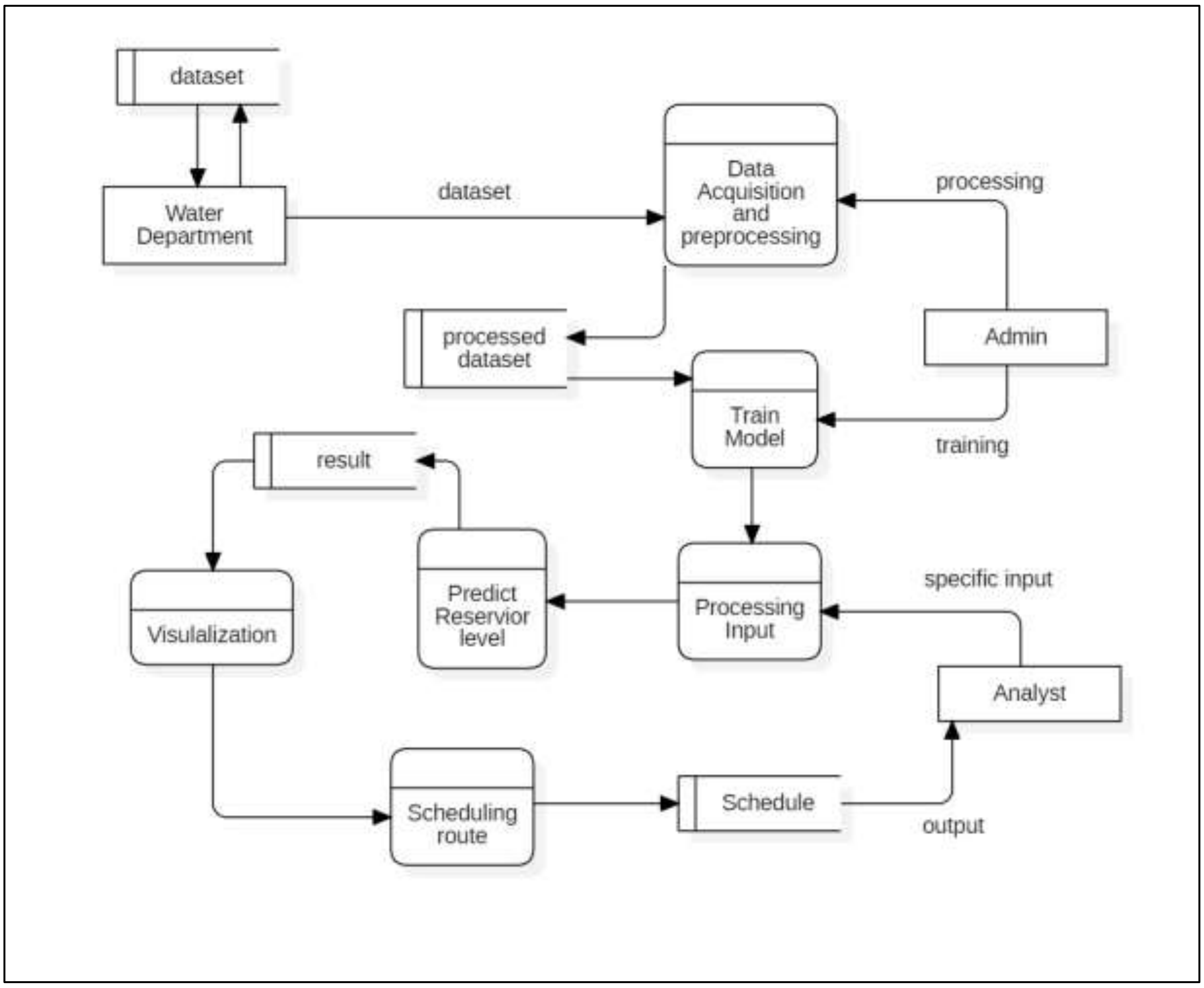


Fig 4.4 DFD level -1

The diagram above is level-1 DFD for our system gives the overall flow of data through the system during model building and running model for a user given specific input. Mentioned above are the repositories at each stage where data is collected and stored. This level gives a detailed flow of data within the system marking all the data stores and processes involved at each stage. The primary data which is in the form of dataset is merged and combined with other dataset to get a reliable source for model building.

4.4 Timeline Chart

The Timeline chart given in Fig 4.5 illustrates the timeline of the tasks done in the first and second half phases of the project.

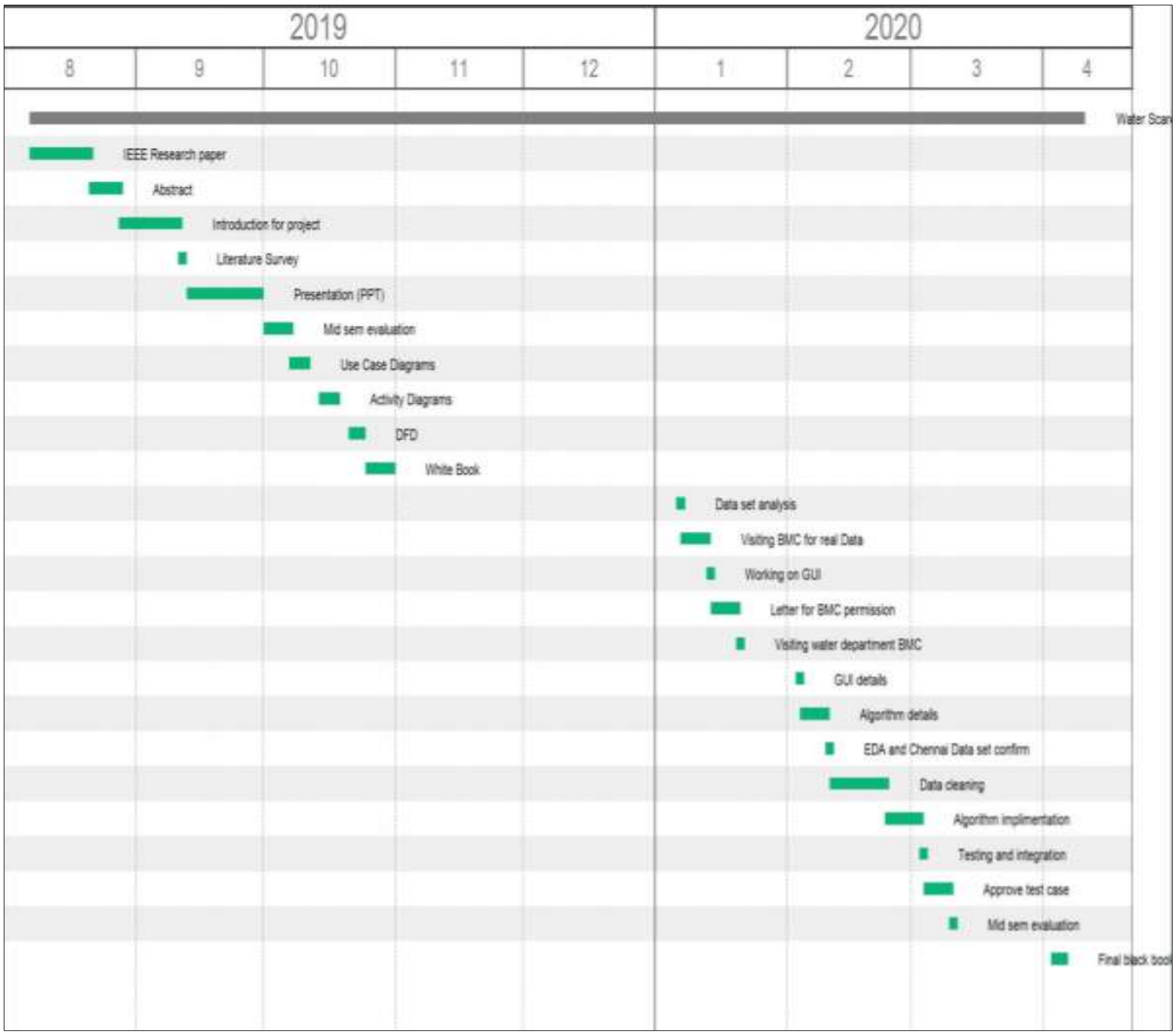
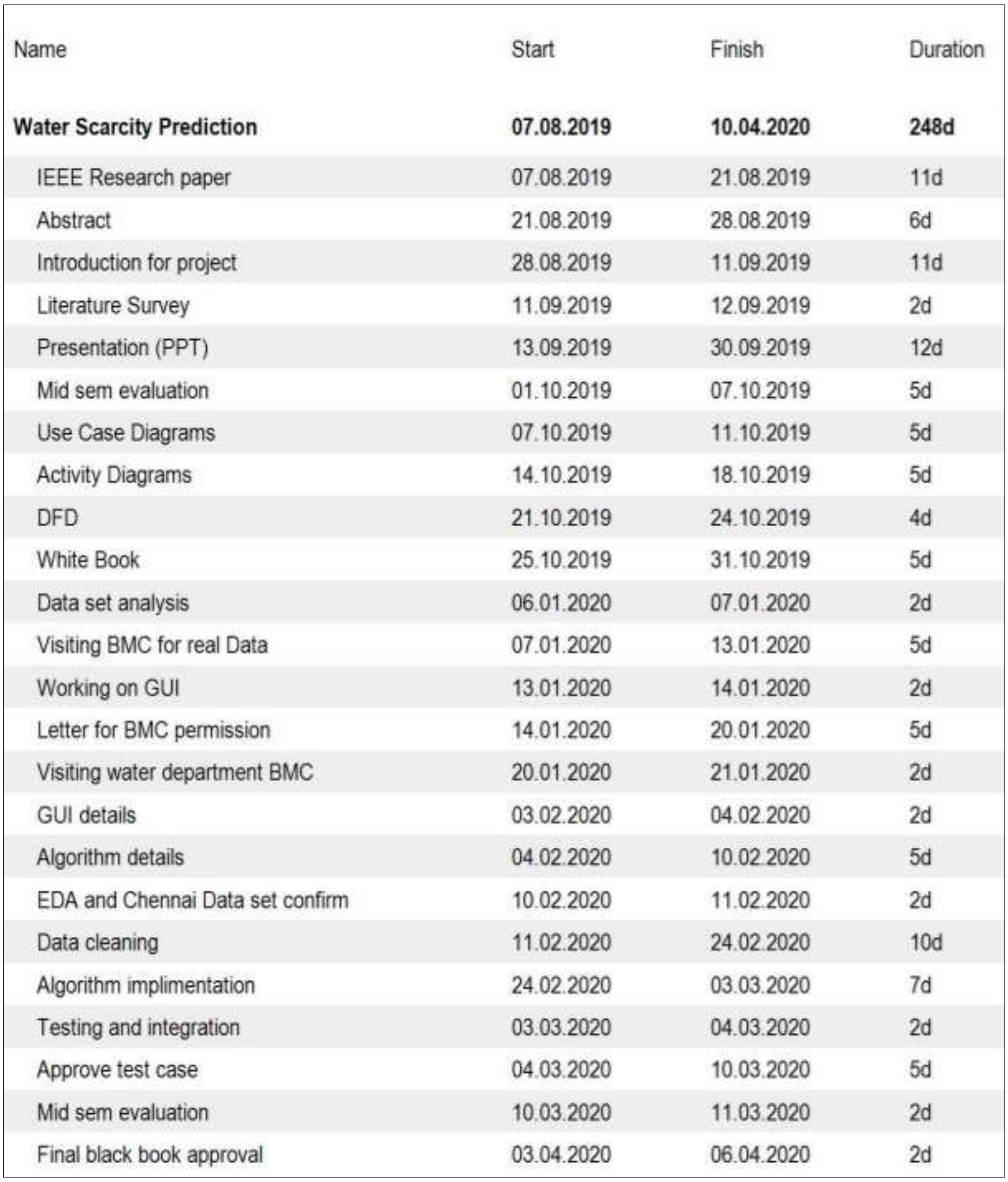


Fig 4.5 Timeline Chart

The time-line for the project is divided into two phases, the seventh semester (first phase) and the eighth semester (second phase). Most of the requirement gathering and analysis is done in the first phase of total time-line and implementation in later phase is done.

The table below gives a detailed task description along with the start date and the end date of the task.



Name	Start	Finish	Duration
Water Scarcity Prediction	**07.08.2019**	**10.04.2020**	**248d**
IEEE Research paper	07.08.2019	21.08.2019	11d
Abstract	21.08.2019	28.08.2019	6d
Introduction for project	28.08.2019	11.09.2019	11d
Literature Survey	11.09.2019	12.09.2019	2d
Presentation (PPT)	13.09.2019	30.09.2019	12d
Mid sem evaluation	01.10.2019	07.10.2019	5d
Use Case Diagrams	07.10.2019	11.10.2019	5d
Activity Diagrams	14.10.2019	18.10.2019	5d
DFD	21.10.2019	24.10.2019	4d
White Book	25.10.2019	31.10.2019	5d
Data set analysis	06.01.2020	07.01.2020	2d
Visiting BMC for real Data	07.01.2020	13.01.2020	5d
Working on GUI	13.01.2020	14.01.2020	2d
Letter for BMC permission	14.01.2020	20.01.2020	5d
Visiting water department BMC	20.01.2020	21.01.2020	2d
GUI details	03.02.2020	04.02.2020	2d
Algorithm details	04.02.2020	10.02.2020	5d
EDA and Chennai Data set confirm	10.02.2020	11.02.2020	2d
Data cleaning	11.02.2020	24.02.2020	10d
Algorithm implimentation	24.02.2020	03.03.2020	7d
Testing and integration	03.03.2020	04.03.2020	2d
Approve test case	04.03.2020	10.03.2020	5d
Mid sem evaluation	10.03.2020	11.03.2020	2d
Final black book approval	03.04.2020	06.04.2020	2d

Fig 4.6 Duration for each task in Timeline Chart

The start and end date for each task is specified with the duration in accordance with the above timeline chart. Listed above are all tasks executed in phased manner.

Chapter 5

Design

In this chapter, we specify the flow diagram of the system. It specifies the logical flow of the system allows the steps in a process to be ordered in a sequence and displayed graphically. A flow chart helps with monitoring progress and status reporting.

5.1 Architectural Design:

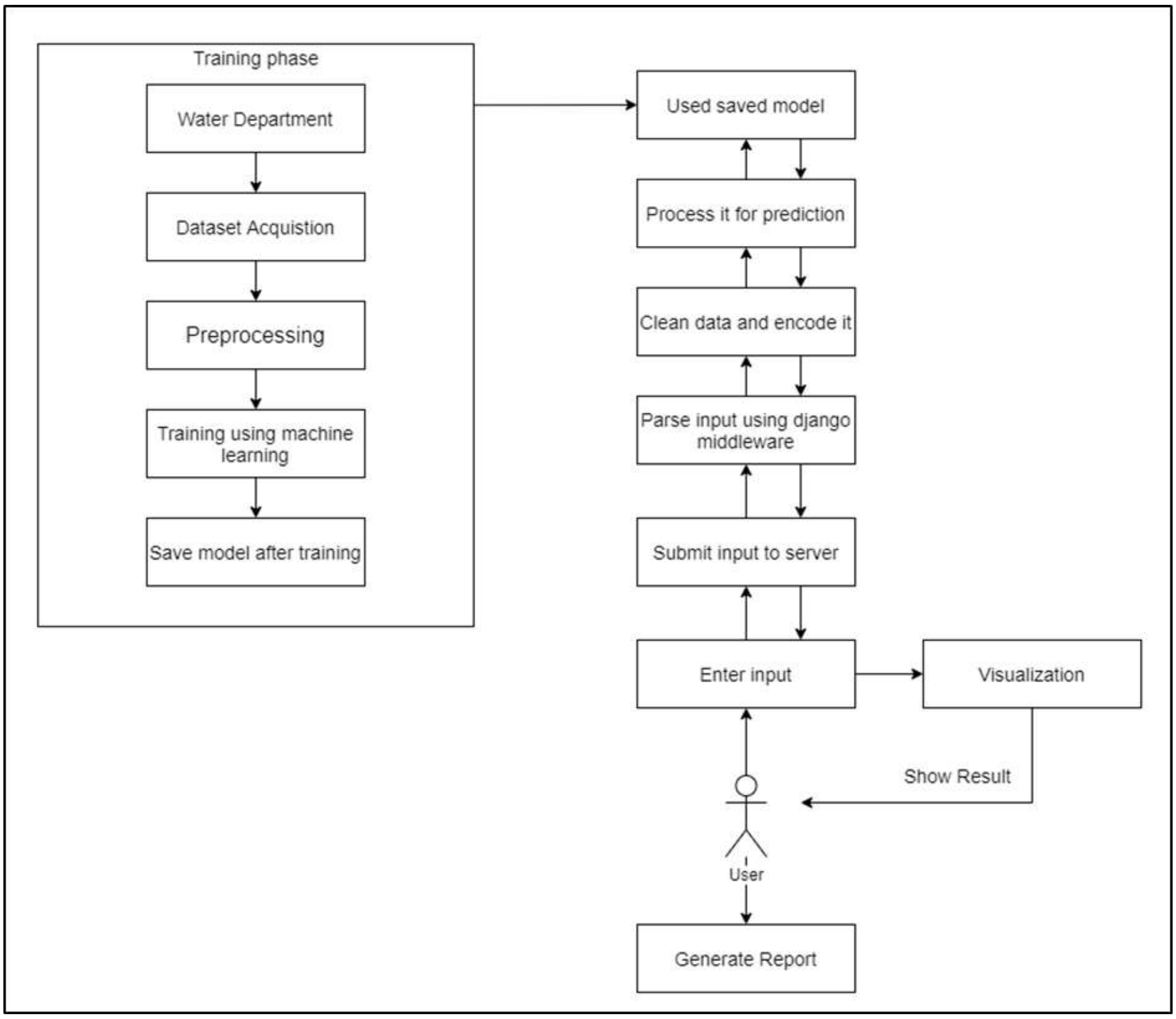


Fig 5.1 Flow of System

The above shows the strategic flow of the system. Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.

Training:

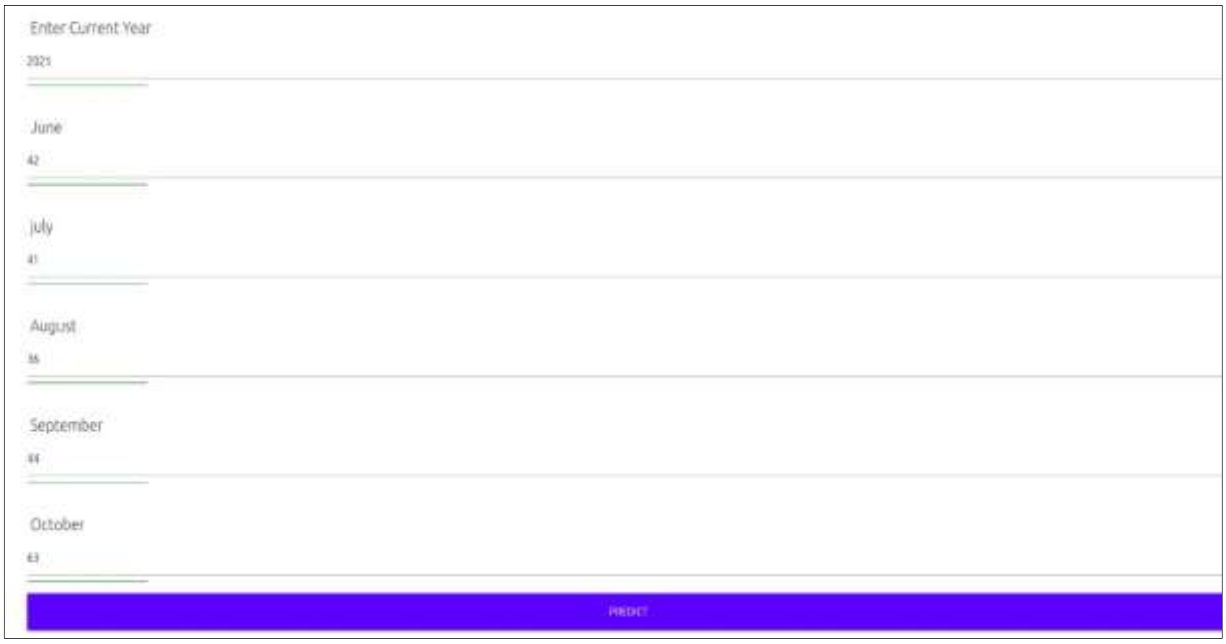
- The data for various features like rainfall, reservoir level, population is acquired through various reliable sources.
- Preprocessing of data is done which includes several steps from merging the data to formulating features like population using growth rate, water consumption to form the dataset.
- Model is trained and fit on this dataset for making predictions and trained model is saved.

Working:

- User provides the input to the system by selecting the year for prediction, and rainfall for each rainy month viz. June, July, August, September and October.
- Data passed is cleaned and encoded to parse the input using django middleware.
- Reservoir is selected amongst the four reservoir viz. Poondi, Cholavaram, Redhills, Chembarambakkam.
- Water level for each consecutive month is displayed using graphical representation.
- Schedule for resource usage for particular reservoir is generated.
- System also provides the optimal route for supply transport from selected reservoir to each of the remaining reservoir.

5.2 User Interface Design:

Fig 5.2 Home Page for Web Application.


Enter Current Year

2021

June

42

july

41

August

36

September

44

October

63

Fig 5.3 User Interface to provide Input

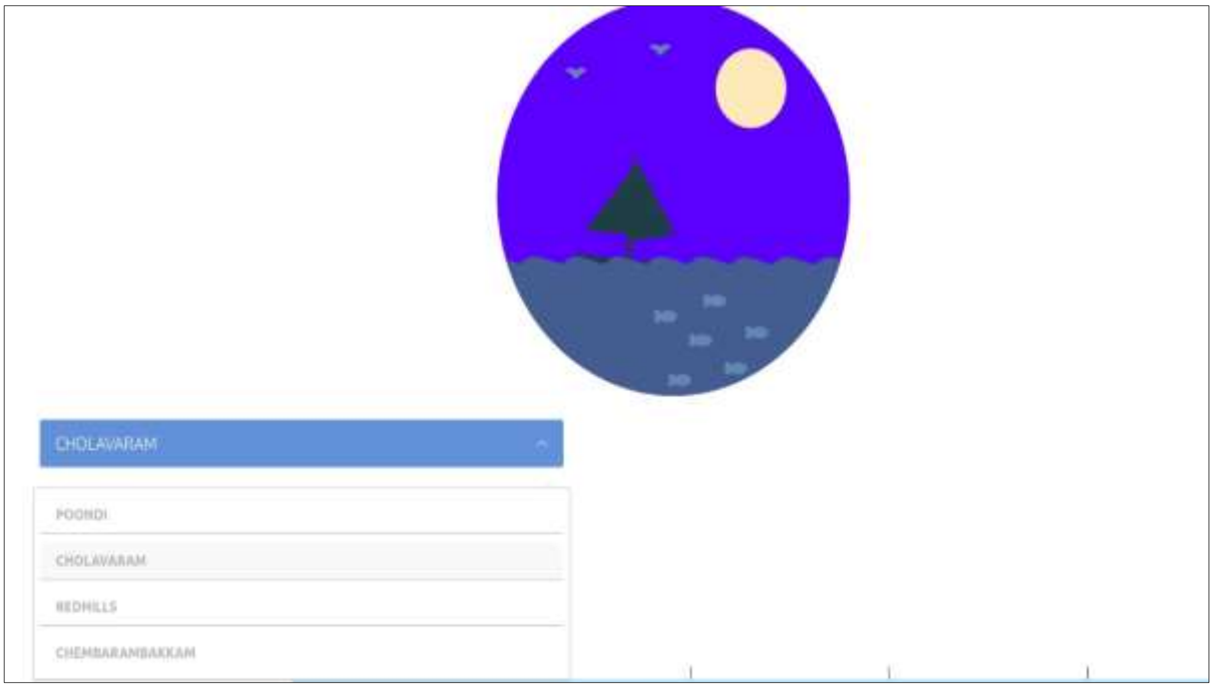


Fig 5.4 Select the reservoir

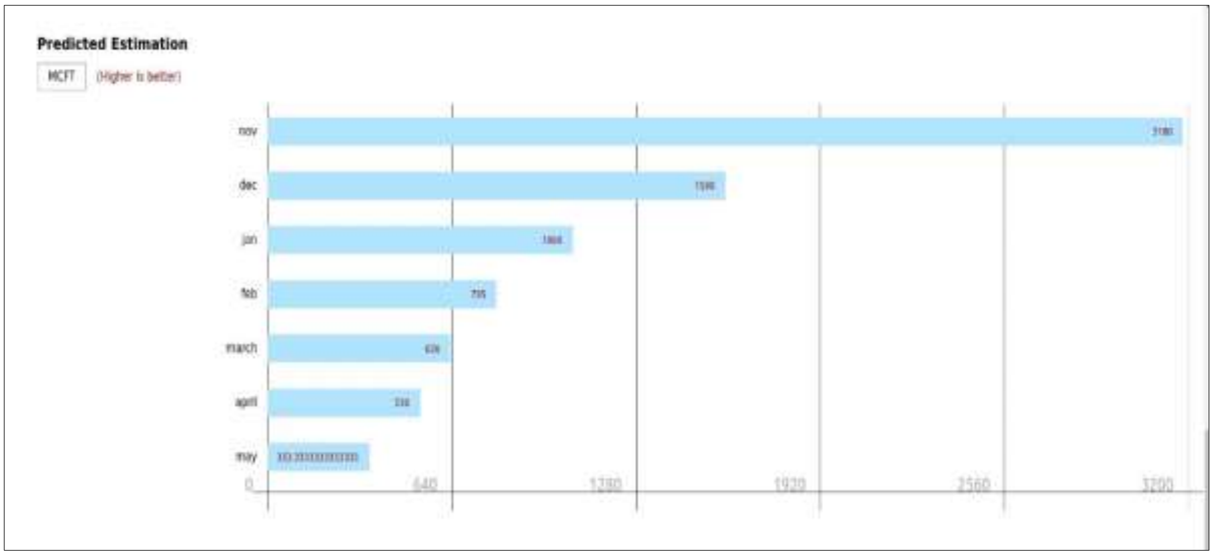


Fig 5.5 Prediction of reservoir level

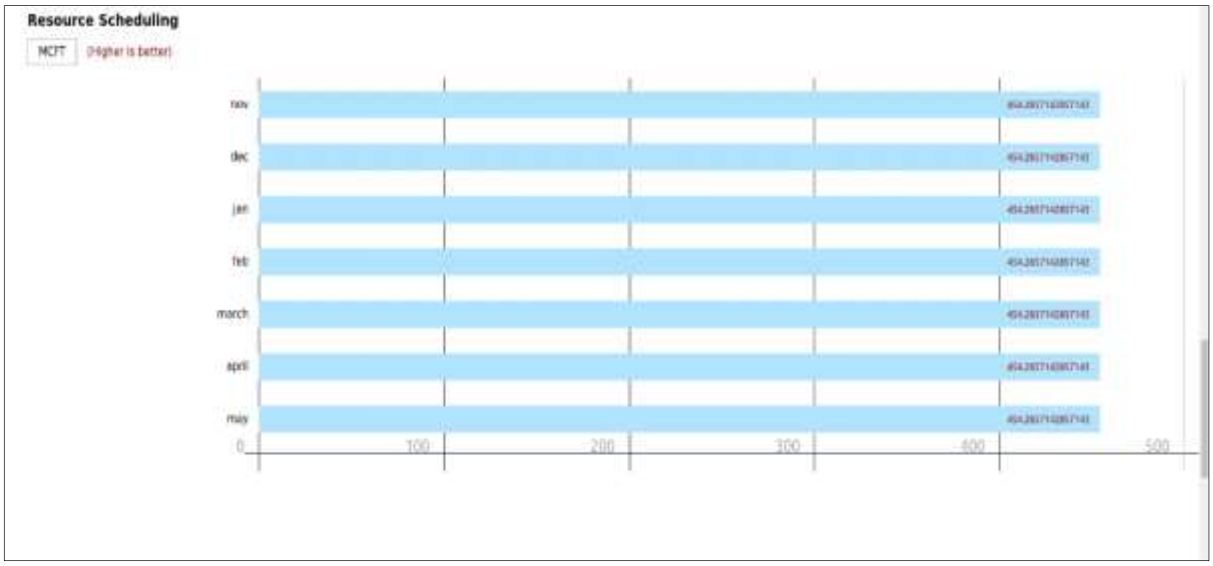


Fig 5.6 Resource usage schedule

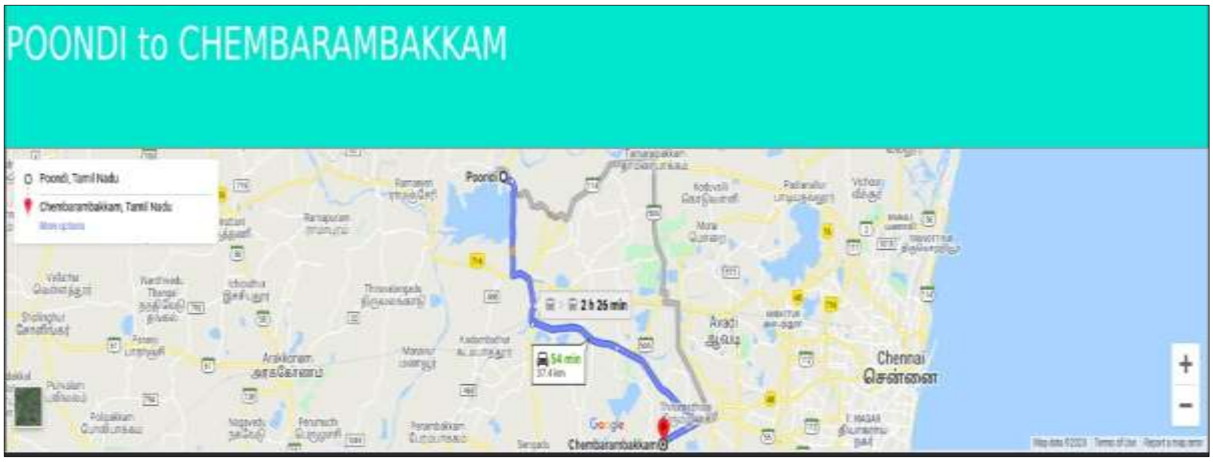


Fig 5.7 Optimized route to other resource

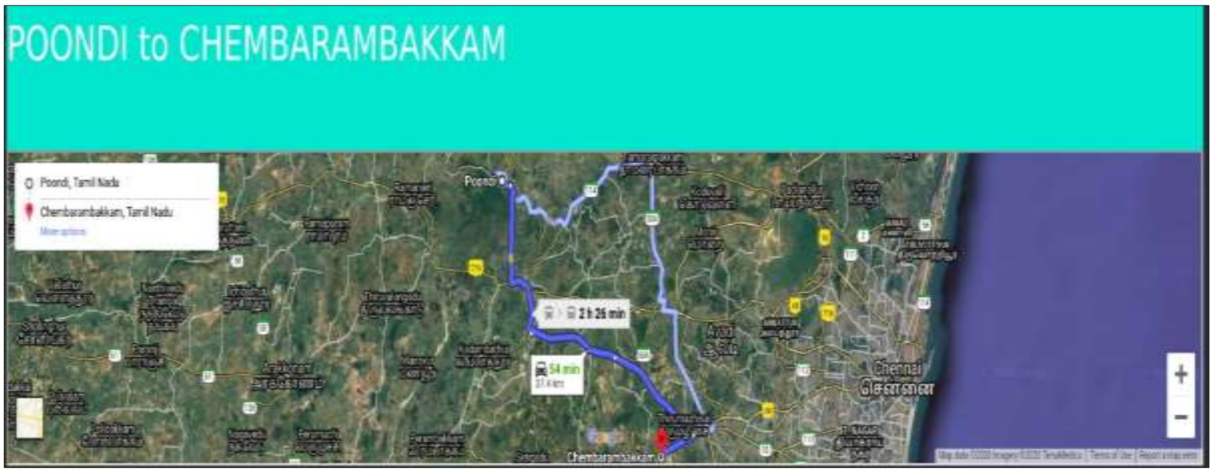


Fig 5.8 Satellite view of the route

Chapter 6

Implementation

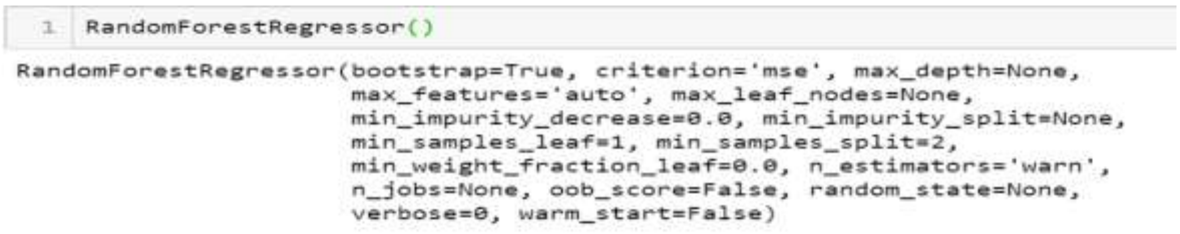
Water Scarcity predictor focuses on Chennai as a region, to train and build model for predicting water level for different reservoirs. We have studied and implemented various regression models to test for their performance and accuracy.

6.1 Algorithms:

6.1.1 Random Forest Regression:

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result. It is a type of additive model that makes predictions by combining decisions from a sequence of base models.

More formally we can write this class of models as: g(x) = f0(x) + f1(x) + f2(x) +.. where the final model g is the sum of simple base models fit. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.



```
RandomForestRegressor()
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators='warn',
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

We can understand the working of Random Forest algorithm with the help of following steps:

Step 1 − First, start with the selection of random samples from a given dataset.

Step 2 − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.

Step 3 − In this step, mean is taken of every predicted result and selected as final result.

The following diagram will illustrate its working −

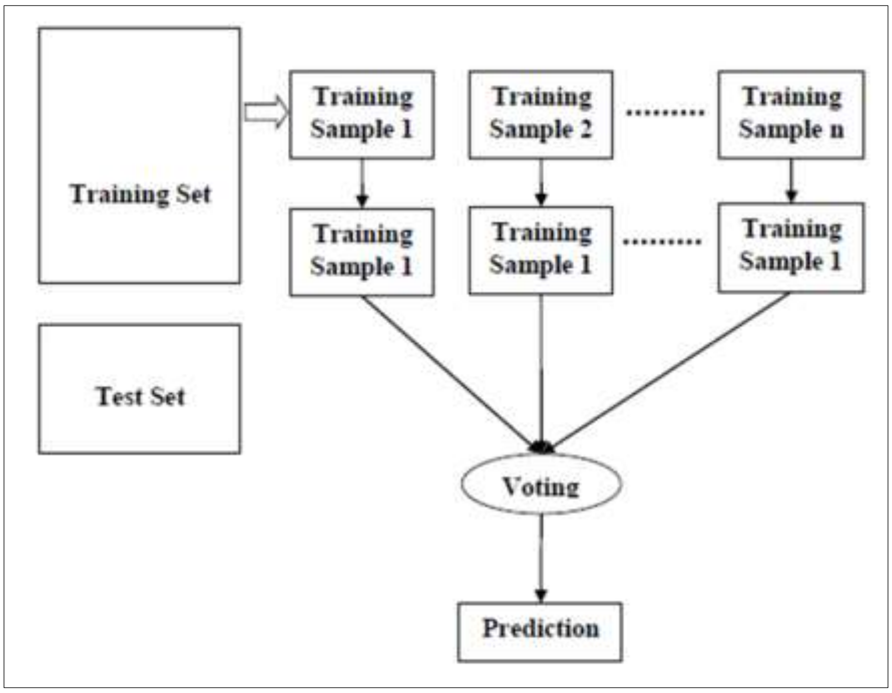


Fig 6.1 Working of Random Forest Regression.

Advantages with Random forest Regression:

1. Fits a number of decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
2. The random forest model is very good at handling tabular data with numerical features.
3. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target.

Parameter:

- n_estimators: integer, optional (default=10) The number of trees in the forest.

- Criterion: string, optional (default=”mse”) The function to measure the quality of a split. Supported criteria are “mse” for the mean squared error, which is equal to variance reduction as feature selection criterion, and “mae” for the mean absolute error.
- max_depth: integer or None, optional (default=None) The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
- min_samples_split: int, float, optional (default=2) The minimum number of samples required to split an internal node:

6.1.2 Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

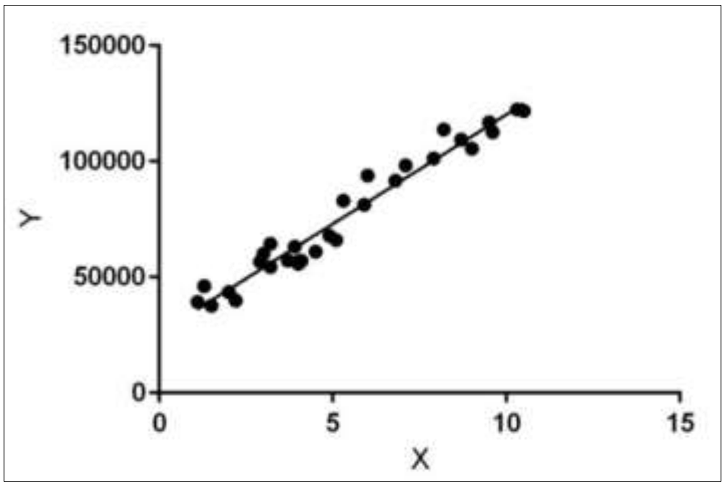


Fig 6.2 Plot of Linear regression model.

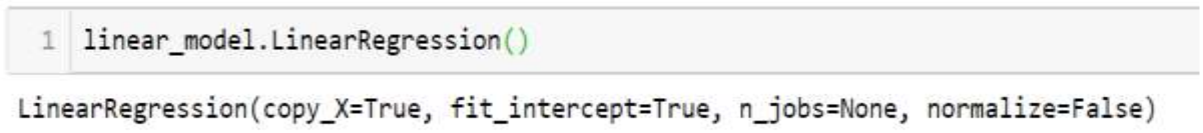
Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y (output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

$$y = \theta_1 + \theta_2 . x$$

While training the model we are given:
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)
When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x

Class sklearn linear_model



```
linear_model.LinearRegression()
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Parameter:

- copy_X: bool, optional, default True. If True, X will be copied; else, it may be overwritten.
- fit_interceptbool: optional, default True. Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).
- n_jobs: int or None, optional (default=None). The number of jobs to use for the computation.

6.1.3 Multi-Layer Perceptron Regressor:

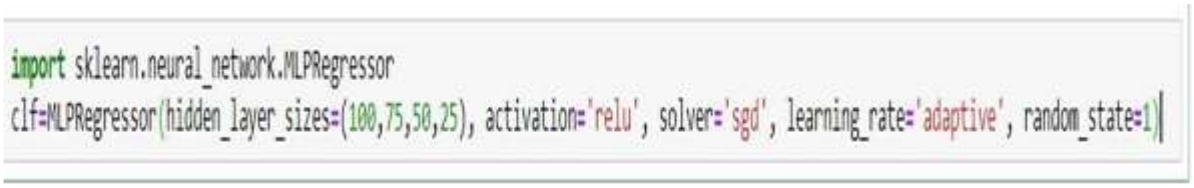
Class MLP Regressor implements a multi-layer perceptron (MLP) that trains using backpropagation with no activation function in the output layer, which can also be seen as using the identity function as activation function. Therefore, it uses the square error as the loss function, and the output is a set of continuous values.
MLP Regressor also supports multi-output regression, in which a sample can have more than one target.

MLP trains using Stochastic Gradient Descent, Adam, or L-BFGS. Stochastic Gradient Descent (SGD) updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation.

Model for predicting reservoir level:

1. The input layer will be given 4 parameters which are population, water consumption, water level and rainfall.
2. The activation function for the hidden layers is ReLu. F(x)=max(0,x)
3. The number of hidden layers are experimented between 1 and 4 i.e. between number of input nodes and output nodes.
4. Finally, 4 hidden layers were chosen and each layer had 100, 75, 50, 25 neurons respectively.
5. Weights are initialized to small random values.
6. The solver for weight optimization is stochastic gradient descent.

Class sklearn.neural_network.MLPRegressor



```
import sklearn.neural_network.MLPRegressor
clf=MLPRegressor(hidden_layer_sizes=(100,75,50,25), activation='relu', solver='sgd', learning_rate='adaptive', random_state=1)
```

Parameters:

- hidden_layer_sizes: The ith element represents the number of neurons in the ith hidden layer.
- Activation: ‘relu’, the rectified linear unit function, returns f(x) = max(0, x)
- Solver: ‘sgd’ refers to stochastic gradient descent.
- Learning_rate: ‘adaptive’ keeps the learning rate constant to ‘learning_rate_init’ as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss, or fail to increase validation score, the current learning rate is divided by 5.
- Random_state: If int, random_state is the seed used by the random number generator.

6.2 Working of the project:

6.2.1 Training

1) **Random Forest Regression:**

```
from sklearn.externals import joblib
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor(max_depth=2, random_state=0)

currentLabel = 3
labels = ["POONDI","CHOLAVARAM","REDHILLS","CHEMBARAMBAKKAM"]
modelsMonth = ["nov","dec","jan","feb","march","april","may"]
for indx,m in enumerate(modelsMonth):
        df = pd.read_csv("../../dataset/"+labels[currentLabel]+"_final.csv")
        dataset = df
        df2 = df
        for i in range(10):
                df2 = df2.append(df2)
        df = df2.reset_index(drop=True)
        dataset = df
        X_update = dataset.iloc[:,1:6]
        X_update["population"] = getDFYearlyPopulation(df)
        X_update["consumption"] = getDFYearlyConsumption(df)
        X = X_update.values
        print(dataset.iloc[:, 1:6])
        Y = dataset.iloc[:,6+indx:6+1+indx].values
        print(dataset.iloc[:,6+indx:6+1+indx])

        from sklearn.model_selection  import train_test_split
        print("Spliting Dataset 60-40%")
        x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size = 0.40, random_state
=0 )
        print("=====> Starting to Train")
        print("x train==========")
        clf.fit(x_train, y_train.ravel())
        y_pred = clf.predict(x_test)
        print("Training complete")

        from sklearn.metrics import accuracy_score
        accuracy = clf.score(x_test,y_test)
        print("accuracy")
        print(accuracy)
        print(m)
```

2) Linear Regression:

```
from sklearn.externals import joblib
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
from sklearn.model_selection import train_test_split

from sklearn import linear_model
clf = linear_model.LinearRegression()

currentLabel = 3
labels = ["POONDI","CHOLAVARAM","REDHILLS","CHEMBARAMBAKKAM"]
modelsMonth = ["nov","dec","jan","feb","march","april","may"]

for indx,m in enumerate(modelsMonth):
        df = pd.read_csv("../../dataset/"+labels[currentLabel]+"_final.csv")
```

```
dataset = df
df2 = df
for i in range(10):
        df2 = df2.append(df2)
df = df2.reset_index(drop=True)
dataset = df
X_update = dataset.iloc[:,1:6]
X_update["population"] = getDFYearlyPopulation(df)
X_update["consumption"] = getDFYearlyConsumption(df)
X = X_update.values
print(dataset.iloc[:, 1:6])
Y = dataset.iloc[:,6+indx:6+1+indx].values
print(dataset.iloc[:,6+indx:6+1+indx])

from sklearn.model_selection  import train_test_split

print("Spliting Dataset 60-40%")
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size = 0.40, random_state
=0 )

print("=====> Starting to Train")
print("x train==========")

clf.fit(x_train, y_train.ravel())

y_pred = clf.predict(x_test)
print("Training complete")
from sklearn.metrics import accuracy_score
accuracy = clf.score(x_test,y_test)
print("accuracy")
print(accuracy)
print(m)
```

3) Multi-Layer Perceptron Regressor:

```
from sklearn.externals import joblib
```

```
import pandas as pd
import numpy as np
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
from sklearn.model_selection import train_test_split

class MaxStdScaler(BaseEstimator, TransformerMixin):
    def __init__(self, copy=True, factor=1.0):
        self.factor = float(factor)
        self.copy = copy

    def fit(self, X):
        X = check_array(X, copy=self.copy, estimator=self, dtype=np.float64)
        self.scaler = np.max(np.std(X, axis=0, ddof=1)) / self.factor
        return self

    def transform(self, X):
        X = check_array(X, copy=self.copy, estimator=self, dtype=np.float64)
        X /= self.scaler
        return X

    def inverse_transform(self, X, copy=None):
        if copy is None:
            copy = self.copy
        X = check_array(X, copy=copy, estimator=self, dtype=np.float64)
        X *= self.scaler
        return X

currentLabel = 3
labels = ["POONDI","CHOLAVARAM","REDHILLS","CHEMBARAMBAKKAM"]
modelsMonth = ["nov","dec","jan","feb","march","april","may"]

for indx,m in enumerate(modelsMonth):
        df = pd.read_csv("../../dataset/"+labels[currentLabel]+"_final.csv")
        dataset = df
        df2 = df
```

```
for i in range(10):
        df2 = df2.append(df2)
df = df2.reset_index(drop=True)
dataset = df
X_update = dataset.iloc[:,1:6]
X_update["population"] = getDFYearlyPopulation(df)
X_update["consumption"] = getDFYearlyConsumption(df)
X = X_update.values
print(dataset.iloc[:, 1:6])
Y = dataset.iloc[:,6+indx:6+1+indx].values
print(dataset.iloc[:,6+indx:6+1+indx])

from sklearn.model_selection  import train_test_split

print("Spliting Dataset 60-40%")
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size = 0.40, random_state
=0 )

clf2 = MLPRegressor(hidden_layer_sizes = (100, 75, 50, 25), activation = 'relu',
solver = 'sgd', learning_rate = 'adaptive', alpha=1e-5, random_state = 1)

print("=====> Starting to Train2")
scaler_x2 = MaxStdScaler()
x_train2 = scaler_x2.fit_transform(x_train)
scaler_y2 = MaxStdScaler(factor=15.0)
y_train2 = scaler_y2.fit_transform(y_train)

print("x train==========")

clf.fit(x_train, y_train.ravel())

y_pred = clf.predict(x_test)
print("Training complete")
from sklearn.metrics import accuracy_score
accuracy = clf.score(x_test,y_test)
print("accuracy")
print(accuracy)
```

```
        print(m)



def getPopultionData(year):
        population_stats = {
                "1950": "1491293",
                "1955": "1705272",
                "1960": "1914797",
                "1965": "2394412",
                "1970": "3044204",
                "1975": "3593728",
                "1980": "4187412",
                "1985": "4735899",
                "1990": "5332057",
                "1995": "5934398",
                "2000": "6593279",
                "2005": "7476986",
                "2010": "8506193",
                "2015": "9677072",
                "2019": "10711243",
                "2020": "10971108",
                "2025": "12336047",
                "2030": "13814367",
                "2035": "15375797"
        }
        all_stats = {}
        for i,x in population_stats.items():
                all_stats[int(i)] = x
                all_stats[int(i)+1] = x
                all_stats[int(i)+2] = x
                all_stats[int(i)+3] = x
                all_stats[int(i)+4] = x
        if year in all_stats:
                return int(all_stats[year])
        return ((year-2039)*(0.021648)) * int(all_stats[2035])
```

```
def getTotalWaterConsumptionInCubicMeter(population):
        return 0.1*(population)

def getDFYearlyPopulation(df):
        poplation_data = []
        for i,x in enumerate(df["Year"]):
                poplation_data.append(getPopultionData(x))
        return poplation_data

def getDFYearlyConsumption(df):
        consumption_data = []
        for i, x in enumerate(df["Year"]):
        consumption_data.append(getTotalWaterConsumptionInCubicMeter(getPopultionDat
a(x)))
        return consumption_data
```

6.2.2 Model

```
from django.shortcuts import render
from django.http import JsonResponse
import pandas as pd
import json

from training.predict import *;
from project.settings import *

def getPopultionData(year):
        population_stats = {

                "1950": "1491293",
                "1955": "1705272",
                "1960": "1914797",
                "1965": "2394412",
                "1970": "3044204",
                "1975": "3593728",
                "1980": "4187412",
                "1985":"4735899",
                "1990": "5332057",
                "1995": "5934398",
                "2000": "6593279",
                "2005": "7476986",
                "2010": "8506193",
                "2015": "9677072",
                "2019": "10711243",
                "2020": "10971108",
                "2025": "12336047",
                "2030": "13814367",
                "2035": "15375797"

        } if getDjangoViews() else None

        all_stats = {} if getDjangoViews() else None

        for i,x in population_stats.items():
                all_stats[int(i)] = x if getDjangoViews() else None
                all_stats[int(i)+1] = x if getDjangoViews() else None
                all_stats[int(i)+2] = x if getDjangoViews() else None
                all_stats[int(i)+3] = x if getDjangoViews() else None
                all_stats[int(i)+4] = x if getDjangoViews() else None



        if year in all_stats:
                return int(all_stats[year]) if getDjangoViews() else None

        return ((year-2039)*(0.021648)) * int(all_stats[2035]) if getDjangoViews() else None



def getTotalWaterConsumptionInCubicMeter(population):
        return 0.1*(population) if getDjangoViews() else None


def getDFYearlyPopulation(df):
        poplation_data = [] if getDjangoViews() else None
        for i,x in enumerate(df["Year"]):
                #print("======")
                # print(x)
                poplation_data.append(getPopultionData(x)) if getDjangoViews() else None

        return poplation_data if getDjangoViews() else None

def getDFYearlyConsumption(df):
        consumption_data = []
        for i, x in enumerate(df["Year"]):
        consumption_data.append(getTotalWaterConsumptionInCubicMeter(getPopultionDat
a(x)))
        return consumption_data if getDjangoViews() else None

# Post  Reservoir , Get Prediction
def postReservoir(request):
   folder_location_label = ["nov","dec","jan","feb", "march","april", "may" ] if getDjangoViews()
else None
   data = dict(request.POST) if getDjangoViews() else None
   print(data)
   input_data = [int(x) for x in data['data[]']] if getDjangoViews() else None

   print("=================>")
   total_population = getPopultionData(input_data[0]) if getDjangoViews() else None
   input_data = input_data[1:] + [total_population] +
[getTotalWaterConsumptionInCubicMeter(total_population)] if getDjangoViews() else None

   final_result = [] if getDjangoViews() else None
   eval_value =0
   for folder in folder_location_label:
      path = "./models/"+data["res"][0]+"/"+folder+"/"+folder+"_model.pkl"
      value = getPrediction(input_data,path).tolist()[0] if getDjangoViews() else None
      value =round(value, 2) if getDjangoViews() else None
   eval_value = round(value, 2) if (value>eval_value) else eval_value if getDjangoViews()
else None
      final_result.append({"name":folder,"value": round(value, 2) if (value>0) else 0}) if
getDjangoViews() else None

return JsonResponse({"data": final_result})
```

Chapter 7

Testing

Testing process evaluates our system with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

7.1 Test Cases:

7.1.1 White Box Testing:

Case 1: Is correct population is mapped based on year

Expected result: Population must be mapped correctly

My output: Population is mapped correctly

Test output: test passed

Case 2: Is correct water consumption level calculated based on population

Expected result: Water consumption level must be calculated based on population

My output: Water consumption level is calculated based on population

Test output: test passed

Case 3: Are all 7 input considered for training the model

Expected result: 7 inputs must be considered for training the model

My output: 7 inputs is considered for training the model

Test output: test passed

Case 4: Are all input one hot encoded

Expected result: All input must be one hot encoded

My output: All input is one hot encoded

Test output: test passed

Case 5: After training all result saved in as model for reuse

Expected result: Result must be saved in as model for reuse

My output: Result are be saved in as model for reuse

Test output: test passed

Case 6: Are saved model used for prediction

Expected result: Saved model must be used for prediction

My output: Saved model is used for prediction

Test output: test passed

Case 7: Is user able to enter data from frontend

Expected result: User is able to enter data from frontend

My output: User is able to enter data from frontend

Test output: test passed

Case 8: Entered input parsed in backend using Django.

Expected result: Entered input must be correctly parsed in backend.

My output: Entered inputs are correctly parsed in backend.

Test output: test passed

Case 9: Entered input used in prediction correctly.

Expected result: Entered input must be used in prediction correctly

My output: Entered input are used in prediction correctly

Test output: test passed

Case 10: Prediction result shown to user

Expected result: Prediction result must be shown to user

My output: Prediction result are shown to user

Test output: test passed

7.1.2 Black Box Testing:

Case 1: System must prompt user to enter year for which they are expecting to get the prediction result.
My output: System is prompting user to enter year for which they are expecting to get the prediction result.
Test output: test passed

Case 2: System must prompt the user to enter rain in mm for 5 months viz June, July, august, September and October.
My output: System is prompting the user to enter rain in mm for 5 months viz June, July, august, September and October.
Test output: test passed

Case 3: On entering the values for rain and year for prediction, the user must have an option for getting prediction for different reservoir in Chennai.
My output: On entering the values for rain and year for prediction, the user are having an option for getting prediction for different reservoir in Chennai.
Test output: test passed

Case 4: System must show user prediction result based on the region he/she has selected.
My output: System is showing user prediction result based on the region he/she has selected.
Test output: test passed

Case 5: System must show user the graphs required for making decisions and scheduling details
My output: System is showing user the graphs required for making decisions and scheduling details
Test output: test passed

7.2 Type of Testing Used:

We are using Black Box Testing and White Box Testing for our testing.

What is White Box Testing?

WHITE BOX TESTING is testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.

It is one of two parts of the **Box Testing** approach to software testing. Its counterpart, **Black box testing**, involves testing from an external or end-user type perspective. On the other hand, White box testing is based on the inner workings of an application and revolves around internal testing.

The term "White Box" was used because of the see-through box concept. The clear box or White Box name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

What are Black Box testing techniques?

There are various test case design techniques applied for black-box testing:

Boundary Value Analysis

It is the widely used black-box testing, which is also the basis for equivalence testing. Boundary value analysis tests the software with test cases with extreme values of test data. BVA is used to identify the flaws or errors that arise due to the limits of input data.

For example: Taking inputs for a test case data for an age section should accept a valid data of anything between 1-100. According to BVP analysis, the software will be tested against four test data as -1, 1, 100, and 101 to check the system’s response using the boundary values.

Equivalence partitioning

This test case designing techniques checks the input and output by dividing the input into equivalent classes. The data must be tested at least once to ensure maximum test coverage of data. It is the exhaustive form of testing, which also reduces the redundancy of inputs.

For example: Taking inputs for a test case data for the example mentioned above will have three classes from which one data will be tested.

Valid class: 1 to 100 (any number), Invalid class: -1 (checking the lowest of lowest), Invalid class: 101(highest of highest).

Chapter 8

Result and Discussions

We have explored the data for different resources and made analysis to support the system. The exploration of data made, enables us to draw inferences and analyses the problem in hand with a greater dept.

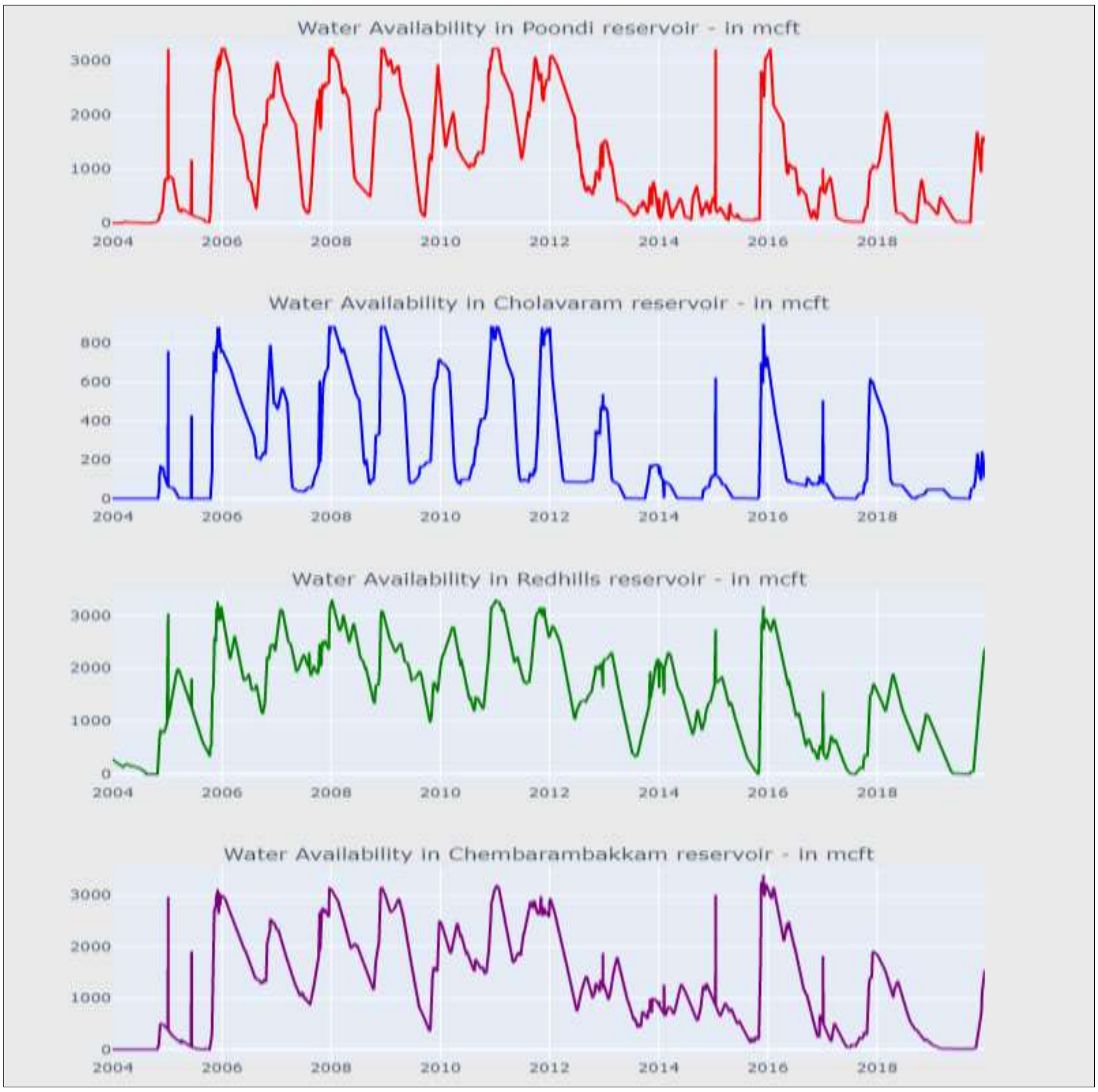


Fig 8.1: Plot of Water Levels in Four Major Reservoirs.

Inference:

- We could clearly see that every year there is a detrimental phase and a replenishment phase (mainly during october to december)
- There was a very bad water scarcity phase seen during 2004.
- We can also see a bad phase during 2014-15 but there was to water availability in two reservoirs (Redhills and Chembarambakkam) and so it was a savior.
- Now coming to recent times, the data shows that there is no water availability in any of the four major reservoirs.

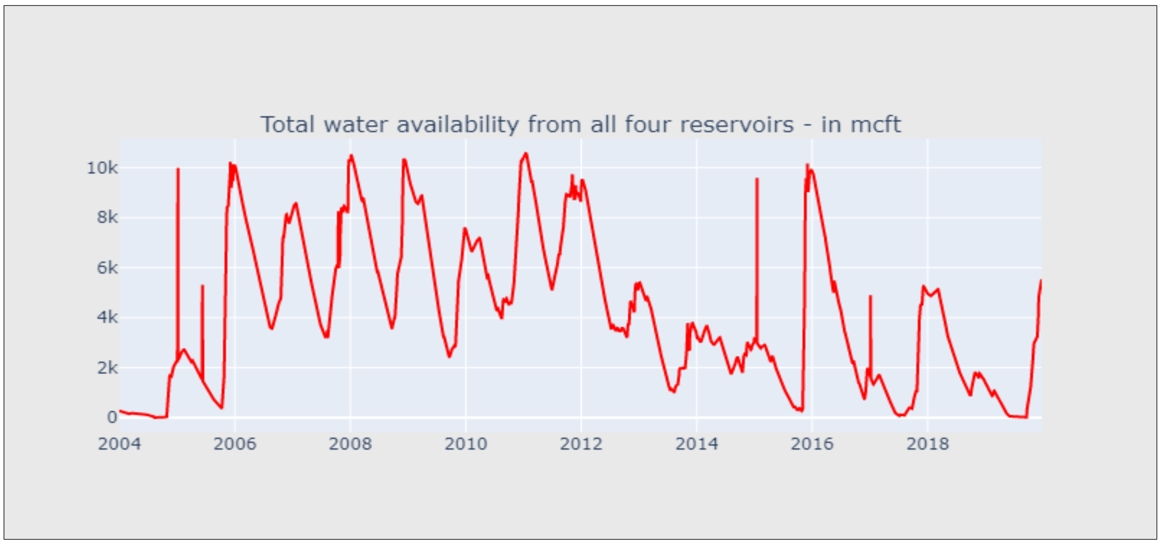


Fig 8.2: Plot of Water Levels in Four Major Reservoirs

Inference:

- Summing up all the water availability from four reservoirs, we can see that the water levels reached almost zero thrice (2004, 2017 and 2019)
- Generally after the rainfall, the reservoirs used to get replenished to about 10K mcft. until 2012 which is not the case afterwards due to the lack of rainfall.
- Only during the (in) famous Chennai floods of 2015, it has reached the 10K level after 2012.
- 2017 is similar to 2019 in terms of depletion of water availability but the levels reached close to 0 during end of august unlike now where the levels reached in the beginning of June itself

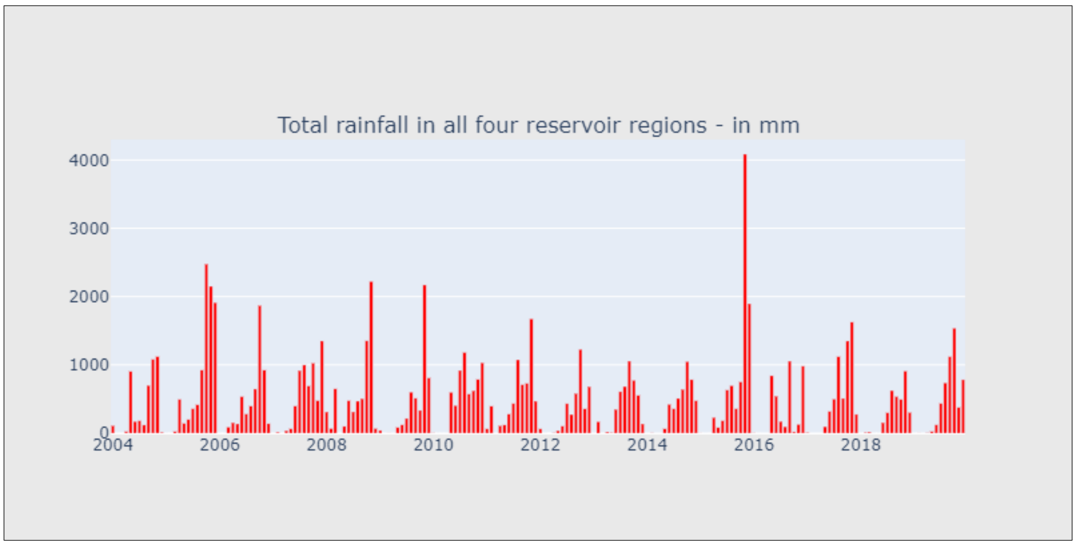


Fig 8.3: Plot of Water Levels in Four Major Reservoirs

Inferences:

- Looks like the city gets some rains in the month of June, July, August and September due to south west monsoon.
- Major rainfall happens during October of every year which is due to North-east monsoon.
- During the initial years rain from north-east monsoon is much higher than south-west monsoon. But seems like last few years, they both are similar (reduction in rains from north-east monsoon).
- We have got some good rains in August and September 2019, but the water reservoir levels are yet to go up.

Performance of Algorithms:

The Random Forest regression model showed the best results compared to other implement model. The performance was evaluated using the R squared, since the model is regression problem. Random forest regression gave accuracy of 96% which is comparatively large as of linear regression model which is 90%. The MLP regressor gave the least accuracy due to the under fitting of curve causes by lack of data to train the neural network. Implementing the Random forest regressor gave ~6% hike in accuracy over linear regression. Calculating the population from year using population growth rate and thereby water consumption per person for the region enhanced the model’s prediction and improved the models accuracy.

Display water level bar graph:

The below graph displays water level that is predicted by our system for next 7 months. Those bar show the water level each month which can be used to decide if there is scarcity

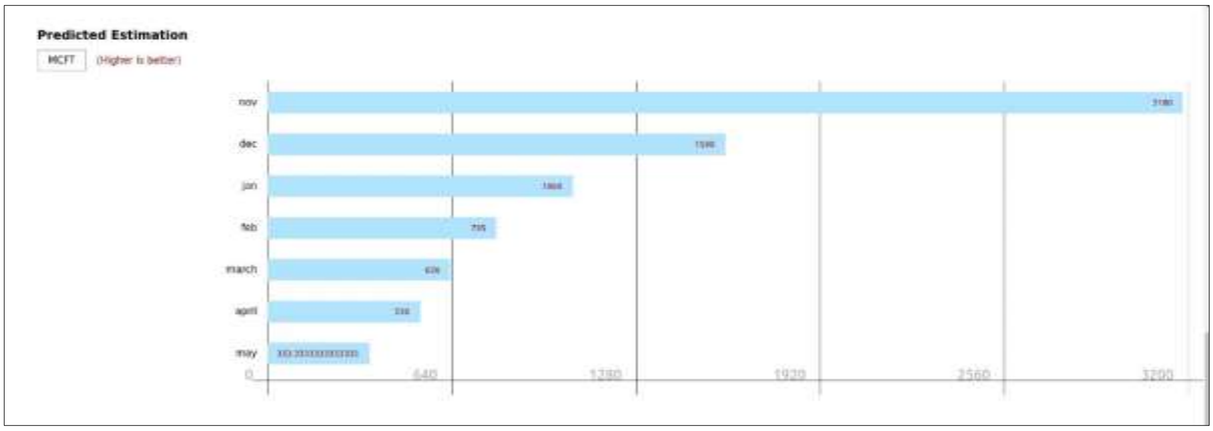


Fig 8.4: Prediction of reservoir level.

Display water level suggestion bar graph:

The below graph displays water level that is suggested by our system for next 7 months. Those bar show the water level each month which can be used to provide water all 7 months without scarcity.

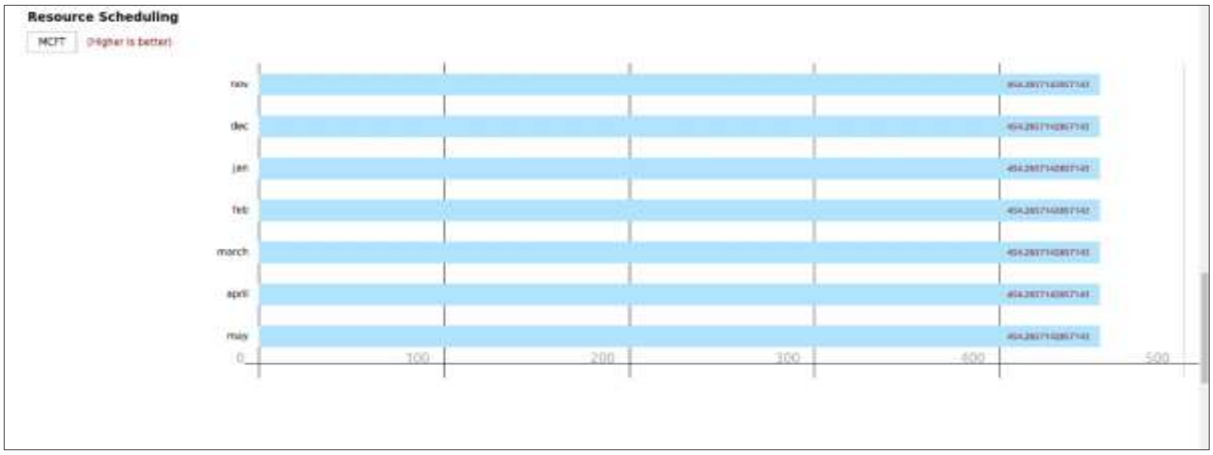


Fig 8.5: Resource usage schedule.

Distance from Current Reservoir to other:

This map show distance from selected reservoir to other reservoir. This will be used to calculate distance and time; it will also help to decide to water scheduling.

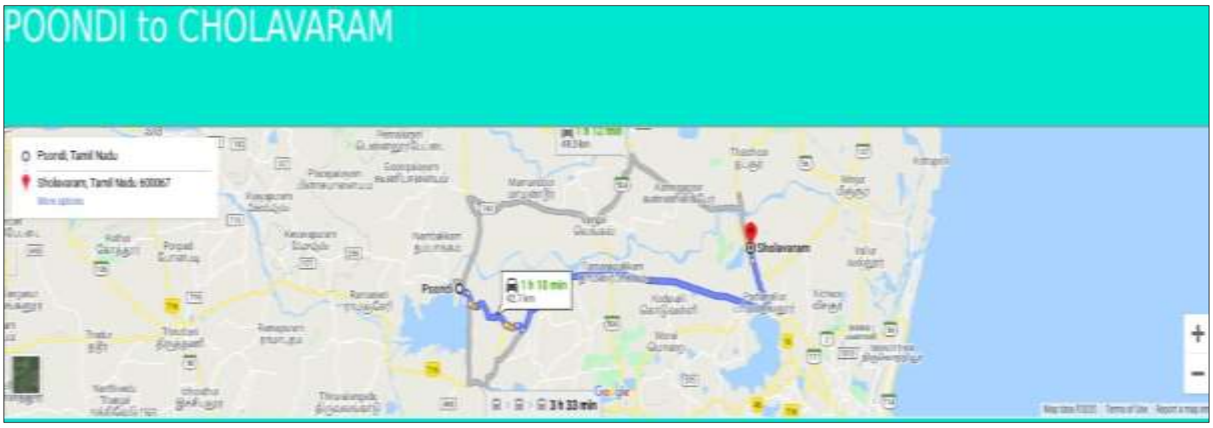


Fig 8.6: Optimized route to other resource.

Satellite View:

Satellite view is shown for map for better resource scheduling.

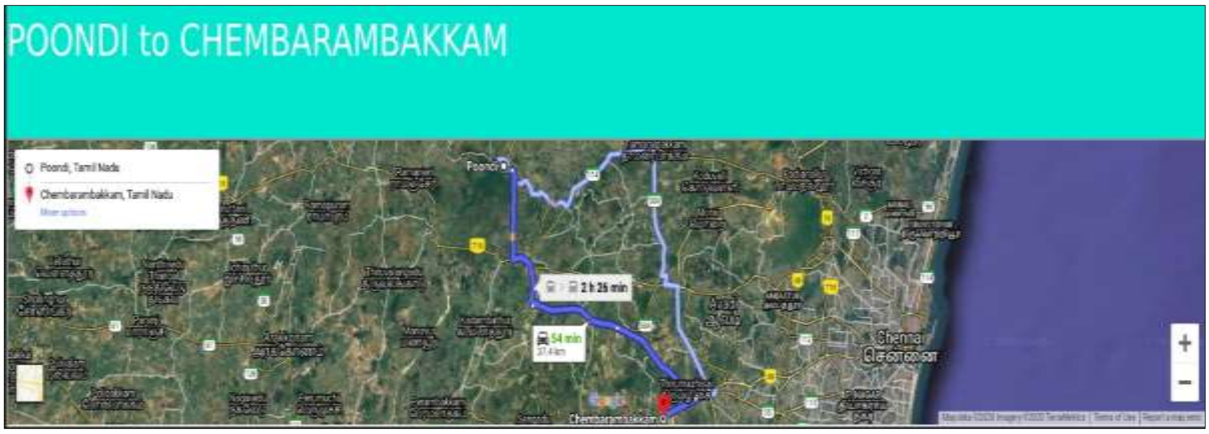


Fig 8.7: Satellite view of the route.

Chapter 9

Conclusions & Future Scope

This Machine Learning based project is aimed at increasing the accuracy of water shortage prediction to improve the water scarcity conditions in Chennai during the lack of rainfall. The system is meant to be used by the Municipal Corporation department in charge of the water sector. It employs the use of a state-of-the-art technology like Random Forest Regressor for creating a prediction model. This water shortage prediction is done based on the rainfall information for 6 months during the rainy season in the 4 major water resources in Chennai. After the prediction, the system creates a schedule to supply water from the resource with abundant water to the resource that is scarce. Overall, this project focuses on solving one of the biggest problems faced by countries all around the world. Water scarcity will be exacerbated as rapidly growing urban areas place heavy pressure on neighboring water resources. Climate change and bio-energy demands are also expected to amplify the already complex relationship between world development and water demand. And because of this very reason, we believe that this project will have a great impact on the world especially the rural areas.

Chapter 10

References

[1] Msiza, Ishmael S., Fulufhelo Vincent Nelwamondo, and Tshilidzi Marwala. "Water demand prediction using artificial neural networks and support vector regression." (2008).

[2] Ren, Yongchang, Tao Xing, Xiaoji Chen, E. Xu, and Ying Zhao. "Based on process neural network learning algorithm for prediction of urban water consumption." In 2010 International Conference on Computing, Control and Industrial Engineering, vol. 2, pp. 34-37. IEEE, 2010.

[3] Khalyasmaa, Alexandra, Stanislav A. Eroshenko, Teja Piepur Chakravarthy, Venu Gopal Gasi, Sandeep Kumar Yadav Bollu, Raphael Caire, Sai Kumar Reddy Atluri, and Suresh Karrolla. "Prediction of Solar Power Generation Based on Random Forest Regressor Model." In 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), pp. 0780-0785. IEEE, 2019.

[4] Hu, Piao, Jun Tong, Jingcheng Wang, Yue Yang, and Luca de Oliveira Turci. "A hybrid model based on CNN and Bi-LSTM for urban water demand prediction." In 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1088-1094. IEEE, 2019.

Acknowledgements

We take this opportunity to thank all those people who have helped and guided us through this project. We wish to sincerely thank our reverend **Bro. Jose Thuruthiyil** and principal, **Dr. Sincy George** for the precious platform to evolve and grow as a student. We would also like to thank HOD of the Computer Engineering Department, **Dr. Kavita Sonawane** and all teaching and non-teaching staff for their immense support and cooperation.

We would like to express our special thanks of gratitude to our teacher, guide **Mrs. Snehal Kulkarni** who guided us and gave us the golden opportunity to do this wonderful project in this domain, which helped us in doing a lot of Research and we came to know about many new concepts, we are really thankful to her.

We also want to thank our family members and friends for their patience and support throughout the project. We are also thankful to all the contributors of online forums, bloggers and YouTubers whose work and insights have helped us understand the perspectives of people around the globe.