A project Report on

**Water Demand Forecasting for Households using Machine Learning and Deep Learning Algorithms**

Submitted in the fulfilment of the requirements for the degree

of

[**Master of Science**]

in

Artificial Intelligence with Robotics with Advanced Research

**University of Hertfordshire**

By

**Mr. HARI KISHORE REDDY KONDA**

18056959

Under the esteemed guidance of

**Mrs. MANAL HELAL**

# 

# MSc Final Project Declaration

I, Hari Kishore Reddy Konda at this moment, declare that this dissertation report and project is solo work of mine and submitted for partial fulfilment of requirements for the degree of Master of Science (MSc) in Artificial Intelligence with Robotics with Advanced Research at the University of Hertfordshire (UH).

This project is a solo work of mine, and therefore entirely belongs to me. I did not involve any help from another human in my MSc project.

I, therefore, provide my utmost permission for this report to be made available on the University's website, given that the source is recognized.

# 

# Acknowledgment

First of all, I thank my professor, Mrs. Manal Helal, for the extreme support and help. I want to share my gratitude towards her for being such an excellent character in my life and always motivating me to work harder to finish this dissertation successfully and within the given time limit. I am very thankful for her for believing in me, even in times when I doubted myself, and she helped me push my boundaries as well. She has shown me the path to success with patience and understood my knowledge as well. I am also very thankful to the Computer Science department that provided me with all the support and help needed in completing this research work.

Additionally, I would now like to thank my other professors, who are an essential part of my journey in completing this research work. They provided me with the necessary knowledge at the correct time and guided me at every step. I am also very grateful to my friends and colleagues who had confidence in me and were always ready to help me whenever I needed it. Finally, I would like to thank my family members for their immense support and help, which was very necessary at this time.

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Abstract

Water is one of the primary sources of life forms on earth and is necessary for maintaining the ecosystem. With the increase in the population of humans, water was mainly used up and wasted in the form of dirt and other chemical substances, which makes it unfit for drinking. Water demands are rising every day, and a suitable forecasting algorithm is essential for this purpose. Since most water gets wasted at the household level, we must start working at the ground level. This research mainly focuses on building a water demand forecasting system for households using Machine Learning and deep learning algorithms.

This forecasting algorithm uses four machine learning and two deep learning algorithms to create this system. The algorithms used here are the Support vector regressor, random forest regressor, extra trees regressor, AdaBoost regressor, and the Long short-term memory (LSTM), and Artificial Neural Network (ANN). Based on the dataset, these models are trained and then tested using the two regression evaluation parameter metrics, mean squared error and the root mean squared error. This system has also been created based on the one input symbolism, which helps further water demand forecasting. We measure the accuracy of these algorithms in terms of Root Mean square Error (RMSE), which means that the lower the error, the better the algorithm's performance. We will calculate the algorithm's performance based on the test data value of RMSE, and we find out that the Extra trees regressor algorithm obtains the most negligible RMSE value of 1.08 among all. Thus, we can use the other trees classifier in this system efficiently.

Table of Contents

[**MSc Final Project Declaration**](#_fyg2vtefsmy0)1

[**Acknowledgment**](#_svyhsss1hor5)3

[**Abstract**](#_22it2gp4jlob)3

[**Table of Figures**](#_fgofx8nqbe8k)5

[**Chapter 1: Introduction**](#_6k98k02fwg8y)7

[Background](#_n8n8bvvvu4nk) 9

[Problem Statement](#_4lbs21ecg6yh) 10

**[Aim of the Project](#_k5e2d9lfm7lj)** 10

**[Objectives](#_6nttzqkcs0fm)** 10

[Motivation](#_9wizyyj1qqn0) 11

[Research Questions and Hypothesis](#_kn8gs0ml6x4d) 11

[Research Consideration of Ethical, Legal, Professional and Social Issues](#_nvwfn4bhj1b) 11

**[Description of the Project Area](#_bzrbkickkdxg)** 12

**[Work Plan Chart](#_hz8u88pqr0yb)** 13

**[Chapter 2: Literature Review](#_8pqwt4qzbhux)** 14

**[Chapter 3: Methodology](#_t7zdlkxrga2v)** 21

[Tools and Libraries](#_fwoq9y20imp2) 22

**[Chapter 4: Implementation, results, discussion & analysis](#_hob355dazy2e)** 23

[Dataset Collection](#_6412ye2h037b) 24

[Data Exploration and Analysis](#_xed55gq9lb82) 24

[Resampling](#_1n5tfboxt855) 24

[bfill method for NaN's replacement](#_9op6qf2s2f3s) 25

[MinMaxScaler for Normalization](#_29c3cb7watd5) 25

[Lag function](#_dt1lghtfxhgd) 25

[Applying machine learning and deep learning algorithms](#_hcap9c45a5dl) 29

**[Chapter 5: Conclusion and Future Work](#_xvjqsblqsvvd)** 39

**[References](#_wo0zxafz5mu)** 40

**[Appendix](#_1jat2m3dk10t) 44**

# 

# Table and Figures

Figure 1: Estimation of household water demands 7

Figure2: Work Plan chart 14

Figure3: Literature Review Table 17

Figure4: Workflow Graph 23

Figure 5: [Mean square erro](https://www.google.com/search?client=firefox-b-d&q=mean+squared+error+formula)r 21

Figure 6: [Root Mean Square Error](https://www.google.com/search?q=root+mean+squared+error+formula&client=firefox-b-d&sxsrf=AOaemvLbb96TaqWvbmgjqkgC43VtcHDfhQ%3A1630233717386&ei=dWQrYbeDF4mC8gKVmLKwAg&oq=root+mean+squared+error+formula&gs_lcp=Cgdnd3Mtd2l6EAMyBAgAEEMyBAgAEAoyBAgAEAoyBAgAEAoyBAgAEAoyBAgAEAoyBggAEAoQHjIGCAAQChAeMggIABAIEAoQHjIGCAAQCBAeOgcIABBHELADOgcIABCwAxBDOgcIIxCwAhAnOggIABAHEAoQHjoGCAAQBxAeOgQIABANOgoIABAIEAcQChAeOggIABAIEAcQHkoECEEYAFCfoQ9Yu6UPYJOnD2gBcAJ4AIABcIgBqQOSAQMzLjKYAQCgAQHIAQrAAQE&sclient=gws-wiz&ved=0ahUKEwj3kYDthdbyAhUJgVwKHRWMDCYQ4dUDCA4&uact=5) 21

Figure 7: Workflow Graph 23

Figure 5: The sample of training dataset 26

Figure 6: The sample of testing dataset 27 Figure 7:The graph of training data and its water consumption 27

Figure 8:The graph of training data and its water consumption 28

Figure 9: The LSTM1 Model loss graph 32

Figure 10: The LSTM1 Model loss graph 34

Figure11: The ANN Model loss graph 35Figure12: comparison table of the algorithms for a particular area day water consumption prediction. 36

Figure 13: comparison table of the algorithms for a single house day water consumption prediction. 37

Figure 14: comparison table of the algorithms for a particular location, for every 5 minutes water consumption prediction. 38

# 

# 

# Chapter 1: Introduction

Water is essential for living beings, whether they be micro-organisms or plants or animals, or humans. Water has many uses in the lives of humans, as it is the second most necessary thing to survive after oxygen in the air (Zhang, Wei and Wang, 2008). We cannot imagine life forms on earth without the need for water sources and resources. Water can be found in three states of matter: solid as ice, liquid as drinking water, and gaseous form as water vapours. It is believed that about 70% of the earth's surface is covered with water, and water present is abundant on earth, and it is never going to end till the last living being survives on the ground. However, we humans face water scarcity problems in major metropolitan cities and other arid areas (Zhang et al., 2008). Here, we will look into these problems as well. We also know water is scarce because only 2% or 3% of the water on earth is believed to be fit for drinking and use for everyday purposes out of all these humongous amounts of water reservoirs.

Water on earth can be found in significant sources in oceans. Oceans contain about 75% or more of the water on this planet. Then some lakes are smaller water bodies than oceans; however, they hold enough water and can be considered bigger water bodies, given the size of these lakes (Zhai, Zhang and Zhang, 2009). However, almost 95% of the water on earth is present in oceans and lakes, but the question arises, can we use this water for our everyday lifestyle? These smaller water bodies only consist of 2% to 3% of the water on the earth's surface. The answer here is NO. The oceans and lakes contain very salty water and drinking or using this type of water is very harmful to our bodies. The muscles and nerves of our body might contract due to the amount of salt present in the oceans (Dong and Zhou, 2009).

Moreover, recent technologies like crude oil collection, oil spills, ships wreckage, polluting the oceans with human waste, and dumping of wastes in oceans have led to a significant destruction of the state of these water bodies. The smaller water bodies like rivers, ponds, and groundwater are the primary source of survival of land creatures on earth. However, these sources provide clean water that humans can use, but rivers and ponds are getting polluted, whereas underground water is depleting at a breakneck pace (Wei, 2007).



*Figure 1: Estimation of household water demands.* ('Water Demand \_ Urban Waters, Bengaluru')

The above figure gives a clear picture of the importance of water use in-house domain. From the figure, we can understand the eight main ways of water usage in the household sector.

The water cycle is one of the essential natural cycles, which has been a significant source for replenishing the water on earth. A water cycle is a process in which groundwater from oceans, seas, lakes, rivers, ponds, etc., is evaporated due to heat produced by the sun and these water vapours rise to form clouds which in turn come back in the form of rains on the ground itself, and again fills up the water bodies (Stuhlmacher and Mathieu, 2020). This natural cycle is critical and efficient for replenishing water on earth. Everybody, including the plants and animals, are necessary for maintaining this ecosystem. Since the population of humans has increased considerably, the ecosystem has been destroyed to a more significant factor. However, several steps have been taken to reconstruct this ecosystem so that once again, we can live peacefully on this planet earth (Sossan et al., 2013). The less amount of rainfall on the land surface of the planet is also one of the reasons why we are suffering from this problem first hand.

The factors that lead to less rainfall on the surface of the earth are:

1. Deforestation: removal of extensive forests for more space for humans to survive.

2. Installing more prominent industries: these industries emit smoke and harmful chemicals in the atmosphere, resulting in acid rain, detrimental to living beings.

3. Practicing agriculture: Practicing agriculture on lands without any trees has its disadvantages since the small roots of plants cannot hold the rainwater

Coming to the problem at hand, water is an essential item for household purposes, and we cannot imagine our life without it. Therefore, we need to work and plan efficiently according to the rules for saving water (Rajabi and Estebsari, 2019). We must never wastewater for recreational purposes like in swimming pools and other such activities, as once this freshwater is gone, there is no coming back. Without enough fresh water to support humanity, we are all doomed to non-existence. In this research work, we are building a water demand forecasting system that will predict the demands of water necessary in the household or a particular locality. We will use the algorithms in this research work: support vector regression (SVR), Random Forest Regressor, extra trees regressor, and AdaBoost regressor, and long short-term memory LSTM, and the Artificial Neural Network (ANN).

These algorithms are the regression algorithms and can be evaluated based on that we need to assess these trained models using the appropriate machine learning regression evaluation parameter metrics. The metric for evaluation that we are following in this research work is, mean squared error MSE, and root means square error RMSE. Both are the loss functions, and the way to evaluate them is the lower the loss function, the better the algorithms' performance. We will now check out the objectives of the research work and how this research work is implemented as well.

## Background

With the increasing number of tasks, it is impossible to handle so much work by an average human. Therefore, we chose automation over daily work. Automation can help us to solve a lot of time and resource problems. So, what does automation mean in our daily lives? Any job that can be automatically done with the help of a machine is termed automation.

Moreover, the term intelligent automation refers to the machine deciding for the work given and then automates it as per the conditions provided. This can be done using artificial intelligence. In this project, where we try to forecast the water demands for households, mainly in urban areas, artificial intelligence plays a significant role. This problem is very tough to solve by a statistician within a given period of about one or more two days. However, a machine with artificial intelligence can solve this problem within a few hours or even minutes. Some of the essential terminologies in intelligent automation are:

**Artificial Intelligence:** Artificial intelligence is a form of intelligence provided to machines for a way of thinking. This makes the devices smart and lets them make their own decisions according to the kinds of problems they face in their day-to-day lives. Artificial intelligence mimics the human brain, and since humans create it, they tend to think like them. Artificial intelligence can be implemented using different ways and using different algorithms. However, machine learning algorithms are one of the most popular applications of artificial intelligence to a system (Minsky, 1961).

**Machine Learning:** Machine learning is a set of algorithms beneficial in applying artificial intelligence to machines. This can be done by training the models with the help of a specific type of dataset based on their learning paradigms. Machine learning algorithms are trained using past data, and they tend to learn using that as well. There are three types of machine learning algorithms: Supervised machine learning, unsupervised machine learning, and reinforcement machine learning algorithms (Sodhi et al., 2019).

**Supervised Machine learning:** Supervised machine learning are the kind of machine learning algorithms that are trained using the supervised dataset. A supervised dataset is a form of the dataset with the data and the results within itself. It teaches using the data and their derivatives all along. After training, the testing part is also done in a similar way, where the results are used in a very efficient way to train the models efficiently (IBM, 2020).

**Unsupervised machine learning:** Unsupervised machine learning algorithms are the ones that use the dataset which does not contain the results, and it trains by itself without any external help from the algorithms. It works based on classifying the results by matching a pattern respectively, then classifying all the similar items perfectly, and then finalizing them accordingly (IBM, 2020).

**Deep learning:** Deep learning is a part of machine learning which is the form of the human brain. One of the most popular deep learning algorithms is Neural Networks. It contains three main layers, the input layer, which takes the input data. The hidden layers are a set of multiple layers for working the algorithms and the output layer that finally returns the output of the algorithms. A lot of the decision-making process is completed within the set of hidden layers. These neural networks are of three types: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Artificial Neural Networks (ANN). They are finally classified further into more algorithms as well (IBM, 2020).

## 

## Problem Statement

To maintain the supply of water in the urban areas and manage the supply of water in drought and other situations, it is essential to forecast the problems of water. It is necessary to manage the supply of water throughout the year and check for droughts and other weather conditions. Therefore, to cope with these situations efficiently, it is necessary to forecast the water supply demands in various regions. There may be conditions in which a city or an area does not get enough rainfall to support its people or for farming purposes. In that case, a very efficient plan is necessary, and the people must be ready for such circumstances as the weather conditions are also unnecessarily changing over time.

There is a limitation of how much water we use, and water distribution infrastructure is necessary to maintain water control. As one of the vital life needs, water must be saved at all costs for future generations of humans to come and survive. Many studies have been done related to the infrastructure management of water demands. However, we can achieve a lot more using artificial intelligence in this field. For this experimentation, a system can be created using machine learning and deep learning algorithms, mainly the regression algorithms in this case. This system will help us automate demand forecasting to a certain extent as well.

## Aim of the Project

This project aims to forecast water demands for households using Machine Learning and deep learning algorithms, which will help us predict the situations and tackle them with an efficient plan.

## 

## 

## Objectives

Objectives of a project are the essential part of understanding what the research work is trying to explain. Moreover, these research objectives play a crucial role in helping the researchers mention every aspect of the objective and other readers. Therefore, it is essential to maintain consistency with the objectives and ensure that the research clarifies them. The main objectives that are covered in this research work are:

1. To understand the water supply situations within all the areas that are to be considered.

2. To get a basic idea of what methods are being implemented for problems related to natural sources of water scarcity.

3. To provide a forecasting system that would predict water demands in each state or an area using the datasets collected.

4. This system will also help forecast the rise in water demands due to more people moving in, thereby maintaining a constant water supply.

5. Performing the pre-processing data tasks to get precise data will be very efficient and will help train the models.

6. Finally, some regression and deep learning algorithms propose the system to predict water demands.

## Motivation

With the increase in global warming and exponential rise in the population, it seems as if there might be a scarcity of water in the future. Given that there is a minimal amount of freshwater present on the earth's surface, humans must take the necessary measures to save the freshwater that we currently have and then try ways to clean the seawater for use if it can be made possible. Since we can save freshwater, we must apply this method first, and we must unite to save water on a global basis. For this purpose, with the help of this research work, we are building a system that will forecast water supply demands. Finally, we will predict the water demands based on the data obtained from previous years. This research work is, however, limited to the forecasting of water demands for household purposes only. Since there are many families in this world and much water is wasted every day, every moment, it is necessary to stop the wastage of all this water and make efficient use of water to save for future generations.

## Research Questions and Hypothesis

To understand the research work in a more generalized way, the research paper must answer a few questions that might be relevant to other researchers as well.

Q1. Factors that are important for saving water in households, industrial use, and agricultural purposes, and what measures must be taken to control the wastage of water?

Q2. Methods and models that need to be trained and can prove beneficial in water demand forecasting for household purposes.

## Research Consideration of Ethical, Legal, Professional and Social Issues

The considerations being taken in a research work play a critical role in verifying the work and its professionalism. It also signifies that the ethical means are taken care of in this research and that this research work is genuine and unique. Therefore, we have maintained this research work's ethical, legal, professional, and social issues. The considerations taken care of in this research are defined as follows:

**Ethical Considerations:** It is considered very important to maintain ethical practices while performing research work, which helps preserve genuine and uniqueness to our research work. All ethical considerations necessary have been taken care of in this project, and some of the important ones are mentioned here as well:

1. I do the execution and performance of this research without the help of any other human beings.

2. All the works that this research refers to and the citations are provided in this research work.

**Legal Considerations:** To work alongside research in a legal form is very important for maintaining the genuineness of the study and abiding by the laws of research is also very important. All the legal considerations are followed in this research work along with the ones that this research mentions of:

1. The dataset in the project work is open-source and is available free for research and educational purposes.

2. This research work is entirely a work of mine. Therefore, there is no reason to raise any copyright or plagiarism issues as well.

**Professional Considerations:** The research work is carried out in a detailed manner, and the observations and results, along with the conclusion, are mentioned professionally and designed in a way that can be understood very easily. All the professional considerations have been taken care of in this research work along with these as well:

1. All the data obtained for the experiments are noted in a designed and professional manner.

2. The implementations and process of research in the sections of the experiments are explained in a detailed manner along with the concerned images.

**Social Issue considerations**: It is essential to maintain social norms and follow social concerns while doing research work so that no one gets hurt due to the research study provided. All of the social considerations are taken care of, and some of these are mentioned here as well:

1. No individuals or communities are intended to get hurt by any of the statements mentioned in this research work.

2. All the research datasets and the previous works taken into consideration do not raise any issues or hurt someone's position accordingly.

## Description of the Project Area

This research focuses on forecasting water demand and supply management, using Machine Learning and deep learning algorithms. It will also predict the daily water needs of a locality accordingly. This project is soft coded efficiently with the help of the Python3 programming language, which is the latest version of Python. This code has been implemented on the Jupyter notebook framework, considered one of the best choices for working with machine learning algorithms, and fulfils the project's aim and objectives. It is essential to describe the research work.

The steps followed below:

1. Perform significant research to choose a topic that needs the help of automation and finalize it as the research topic.

2. Prepare a proposal for approval to start research on the specific topic and then study the previous works accordingly to understand the research work better.

3. Collect the relevant dataset, which will be used for efficiently training the models.

4. Analyze the dataset and look for its properties, other parameters, and null values.

5. Clean the dataset efficiently using the appropriate pre-processing methods and remove the unnecessary rows and columns. Also, remove any outliers present in the dataset itself.

6. Apply the necessary visualization commands and libraries to visualize the dataset by plotting the graphs and plots efficiently.

7. Apply the feature engineering methods like scaling the dataset and split the dataset into the training and testing modules.

8. Choose the appropriate machine learning and deep learning algorithms to be applied for the modelling of this project and then process the dataset efficiently. Now, train the models efficiently using the training dataset.

9. Use the parameter metrics to evaluate the trained models and test the trained models using the regression evaluation parameter metrics and the testing dataset.

10. Finally, conclude the evaluation after deciding for the best machine learning model using the evaluation factors through regression evaluation, and implement it in the final system.

## 

## 

## 

## Work Plan Chart

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Day 1-10** | **Day 11-25** | **Day**  **26-45** | **Day**  **46-70** | **Day 71-86** | **Day 86-105** | **Day 106 - 130** | **Day 131 - 150** | **Day 150 - 165** |
| Introduction |  |  |  |  |  |  |  |  |  |
| Research work followed |  | |  |  |  |  |  |  |  |
| Proposal preparation |  |  | |  | | | | | |
| Survey of previous researches |  |  |  | |  | | | | |
| Gathering the dataset |  | | |  | |  | | | |
| Data pre-processing |  | | | |  | |  | | |
| Feature engineering |  | | | |  | | |  | |
| Applying Machine Learning and Deep Learning Algorithms |  | | | | |  | | |  |
| Regression evaluation |  | | | | | |  | |  |
| Final report preparation |  | | | | | | |  | |

# Chapter 2: Literature Review

The process of training a machine learning algorithm is called modelling. After we split the dataset into the training and testing algorithms, we need to use the dataset for the training purposes of the algorithms we have chosen to work with. This is one of the most crucial steps of a machine learning project. The machine learning and deep learning algorithms that we are going to use for this research work are:

**Support Vector Regression:** Support vector regression (SVR) is one of the most popular and widely used regression algorithms, which can also be used for classification purposes as a Support Vector Machine (SVM). It is a supervised machine learning algorithm used for predicting discrete values. An SVR works based on hyperplanes and kernels where hyperplane is the best fit line and kernel is a set of mathematical functions that takes the data as input and transforms them into the required form. SVR uses the same principle as that of SVM. It selects the best fit line, which is a hyperplane having the maximum number of points. It doesn't try to minimize the error between the predicted and real values; instead, SVR tries to fit the hyperplane within a provided threshold value, which is the distance between the hyperplane and the boundary line. The Linear SVR algorithm is perfect to use for large datasets and provides a faster implementation than the normal SVR; however, it can only consider the linear kernel during computations (Debasish and Srimanta, 2007).

**Random Forest regressor:** Random Forest regression is a supervised machine learning algorithm formed by combining many decision trees parallelly and then using the resultant variance of all these decision trees as one. For this, first, we need to understand what a decision tree is. A decision tree is a machine learning algorithm that takes the dataset and then chooses a root node for the base of a tree. With the help of that root node, it decides to select another sub-node and then follows the steps of decision making throughout the process, and at last, it reaches the end of the tree, which is the leaf of the tree. Finally, a decision tree is created, and the tree's height mentions the number of decisions made to reach that point. Similarly, the random forest algorithm provides different variance to the decision trees based on the small chunks of the dataset accordingly. This technique is also known as bagging and is very efficient in machine learning, where the result is the majority of predictions of every tree considered (Biau, 2012).

**Extra Trees Regressor:** Extra trees regressor works similarly to that of a random forest, and both are similar ensemble methods. It is composed of many decision trees, where the final result is the majority of predictions of every tree considered. Extra trees consider the whole dataset as a sample, which may increase the variance due to the bootstrapping technique, making it more diversified. This algorithm selects the cut points randomly, and once setting the cut points is done with, it chooses the best one among all the subset of features (Geurts, 2006).

**AdaBoost Regressor:** It is also known as the Adaptive boosting regressor algorithm, which is a supervised machine learning algorithm. An AdaBoost regressor is a form of meta-estimator that fits a regressor on the dataset and then fits additional copies of the regressor on the same dataset, where weights of instances are adjusted accordingly to the error of the current prediction. It supports the impurity-based feature parameters where the higher the feature's importance, the more critical that feature is. The basic idea behind this algorithm is to convert the weak learner into a strong one. It works to fit the sequence of weak learners on different weighted training data and starts by predicting an original dataset that moves on, giving equal weights to each of the observations and then provides more weight to the underperforming learners. The main parameters in this algorithm are n-estimators, learning\_rate, and base\_estimators (Dimitri, 2004).

**Long Short-Term Memory (LSTM):** It is a kind of recurrent neural networks algorithm that is more powerful than the recurrent neural network itself and can work in more adverse conditions. LSTM solves the problem of long-term dependencies faced by RNN, considering which RNN could not perform its operation. LSTM’s structure is a chain-type structure containing four neural networks and cells, the memory blocks. The information is retained by these cells through three gates: the forgotten gate, the input gate, and the output gate. These gates act as a filter for the memory or the cells. I will then produce the required results through a complex procedure. LSTM is popularly used in problems like language modelling, machine translation, chatbots, image captioning, etc (Hochreiter and Schmidhuber, 1997).

**Artificial Neural Networks (ANN):** Artiﬁcial neural networks have extraordinary potential to take in complex undeniable level information from crude data sources because of their nonlinear representation of the hypothesis function. There are three main parts, like the input layer, hidden layer, and output layer. The Input Layer design is totally and particularly resolved once we know the state of our preparation information. There’s just one info layer, and the quantity of neurons containing that layer is equivalent to the number of measurements (sections) in our information. Some NN designs add one additional node for a bias term. The hidden layer is specifically designed to produce an output-specific result. The hidden layer consists of two-third of the neuron of the input layer. This project performs no better, so we introduce dropout and L2 regularization to prevent overfitting. Dropout reduces the effect of the neural network by turning off a few neurons while training in the input layer. Moreover, regularization allows you to introduce a penalty in the neural network. The Output Layer has just one of them. The picked model arrangement controls its size (number of neurons). It is set to 10, equivalent to the number of outputs in the objective vector (no of output class). In this project, we use the softmax layer as the final output layer in the neural network. The reason for choosing the softmax layer is because we are solving multi-class classification problems (Chang and Liu, 2009).

A lot of research work to solve the issues related to water demands and supply has been studied and researched to date. The most common problems among these were increased temperature due to global warming and, therefore, less water supply (Chang *et al.*, 2008). As a result, these research works have been finding an automated way to get a solution to these problems. This section will look at the previous works and study how they can benefit us in continuing our research work. When we face water scarcity at such a high level, it is necessary to encourage the general public to save water smartly. This can be fulfilled by having a personalized water usage recommendation system that would promote water-conscious behaviour (Rahim *et al.*, 2019). This system will help the customers save water by suggesting the most effective ways based on the data collected from their historical uses with the help of intelligent water meters. This will be very advantageous for saving water during droughts or in case of water shortage, and it will also help keep customer's money. Therefore, it presents a recommender system that uses Long short-term memory (LSTM), a neural network that predicts the practical uses of water in 83 houses. The results show that LSTM is useful and achieves an average RMSE value of 0.403.

Water demand forecasting is a need for society as the water level goes down with time. This research paper (Zeng and Song, 2011) presents a real-time application of the system using the Autoregressive Moving average (ARMA) combined with grey forecasting model GM(1,1). The predicted results show that the total demand for water in industrial and domestic use will increase with time. In contrast, the need for an agricultural water supply will decrease with the stipulated time (Xu and Liu, 2009). However, the agrarian demands will still be the maximum. The graph acquired through the total water demand and industrial water demand seems similar to the dataset covering the data between 2010 and 2018 in Qinzhou, China. The accuracy of the p-value obtained in this particular system is about 99.9% which is almost perfect (Alvi *et al.*, 2019).

Agriculture is one of the most significant sources that need water, as the plants need water for proper growth. Water is the primary source of irrigation and forecasting the agricultural water demand is one of the most significant ways to optimize large amounts of water (Mkireb *et al.*, 2018). Most of the farmers follow a type of irrigation system that wastes much water. The most observed system is letting the water run over in the fields. As a result of this type of irrigation, the additional water may get mixed with chemicals and pesticides. It would run into other water body sources like canals, rivers, or ponds. This paper (Li, Ding and Lv, 2010) proposes a hybrid model that combines the rough set theory and the most miniature square support vector machine (LS-SVM) to forecast water demands in irrigation. This model performs better than the other two algorithms, obtaining a 3.87% RMSE. In contrast, the addition RS-WLSVM, another hybrid model, serves the best and achieves an RMSE error function of just 2.19%.

As we have discussed, the changing environment has a significant role in water demands in this era. Therefore, this paper (Xu, Mei and Yong, 2011) proposes a forecasting model that will check for the water demands in urban areas. In this paper, Dongguan city was taken care of as a study subject. Influence factors are then created based on climate factors, economic factors, and social factors. The model developed for the water demand forecasting will also analyze the response relationship between the water demands and various influence factors. This research algorithm has trained artificial neural networks (ANN) and the Support Vector Machines (SVM) model using the dataset. Additionally, bipartite, radial basis function network and combination forecast has been applied as a part of the RNN. Out of these four combinations, the forecast achieves the most negligible MAPE value of 2.33, typically the most effective algorithm.

|  |  |  |
| --- | --- | --- |
| **Author** | **Algorithms used** | **Methods Applied and workflow** |
| (Zubaidi *et al.*, 2019) | Singular spectrum analysis,  autoregressive models, and multi-level  decomposition | This paper presents a model for the case study of the Baghdad governate for the prediction of municipal water demand in Iraq. This paper uses last ten year's data from (2006-2015) to predict water use, supply, and order in the upcoming year. Here the researchers have taken five years data of the training, for predicting the one-year data, this model aims to get a forecast for the short-term prediction of municipal water on monthly basis. It uses the hybrid univariate singular spectrum analysis and autoregressive model, the SSA\_AR model. The single spectrum analysis plays a significant (SSA) role in data pre-processing and the Autoregressive model AR model used for the forecast the water demand system. The average correlation coefficient of data reaches 0.92 when absolute and predicted values are combined, and the maximum reaches 0.987, the average value of the normalized fitness function reaches 96.3%. |
| (Chang and Liu, 2009) | Artificial neural networks (ANN),  Radial basis function (RBF) | In this paper uses the radical basis function neural network, for the water demand prediction system, which is known as RBF neural network and its quite like the Artificial neural network (ANN) i.e., it has learning and training process. In this article use total 17 water demand predictors are used, including the urban household, rural household, daily water consumption, and Agricultural and Industrial water consumption. And the researchers have taken the actual data of 11 years (1990-2000), with in which eight years data has taken for the training the model and the remaining three years data used for the prediction of the water demand forecasting model. A radial basis function neural network has been set up for the water demand prediction using 17 water demand factors. The dynamic clustering algorithm determines the RBF's width, cluster centre, weight, and the number of nodes present in the hidden layer. The relative error of these algorithms for three years is 2.74%, 3.335, and 1.41%. Average fitting error obtained for RBF is 0.0033%, 0.0017%, 0.0007%, and 0.0005%. The average prediction error for RBF is 4.89%, 4.23%, 10.80%, and 1.44%. |
| (Ji *et al.*,  2014) | Least squares  support vector machine (LS  SVM) and  Ameliorated  teaching-learning based optimization algorithms. | In this paper, the urban demand for water supply is taken care of with the help of (least squares support vector machine) LS-SVM and with tuning-based teaching-learning-based optimization (TLBO). Here the focus is generally on studying the hourly water demand forecasting performances of water supply. The training data has collected from the Yanqiao in Qingcaosha water supply system. The length of the dataset is one year long, the researchers consider the four input features that effect the water demand forecasting system as follows. The first one previous water flow data, the Max and Min temperature of the forecasted time, the perception time, and the final input is holiday information of the forecasted time. And the output label is the quality of the water flow for the LS-SVM model. The tuned model by ATLBO achieves the regression accuracy, which is the best and has the mean square error MSE of 280.75. The MSE of forecast values of the tuned LS SVM model through TLBO is 626.07. |
| (Zhou,  2010) | Particle swarm  optimization,  general regression neural networks | This paper uses a general regression neural network (GRNN) to create a forecasting model based on the particle swarm optimization algorithm for water demand forecasting. GRNN models the non-linear relationship in this system. PSO has been used for the improvement of the performance of GRNN prediction performance. The dataset has been taken for water demand prediction from the Yellow River Basin. Based on the same dataset, the GRNN-PSO performs better than that of the Genetic algorithm based GRNN-GA prediction algorithm and the Backpropagation based genetic algorithm BP-GA prediction algorithm. The research has collected the data for yearly basis from the 1980-2000, the first eighteen years data has taken (1980-1997) for training the model and the final three years data for the testing the model performance (1998-2000), the data set has six input features as follows, the first one is industrial supply, agricultural supply, Irrigation quota, and urban and rural supply, and the final one livestock, and water demand has considered as output label. The research has finally compared the back propagation based on genetic algorithm (BP-GA) model with (GRNN-GA) model. The GRNN-PSO has the better performance. |
| (Liu, Deng and Zhang, 2009) | Adaptive neuro  fuzzy inference  system (ANFIS) | This paper presents a fuzzy theory-based model used to forecast the urban water demand and supply based on the fuzzy theory. This paper introduces the forecasting method based on ANFIS. The parameters being used in this research paper are obtained through original fuzzy rules examples and are optimized based on the adaptive mix learning arithmetic. Finally, the forecasting is obtained in graphical means. For the hourly demand prediction, the researchers have collected the data form the city of north China, with time range of one year (1998-1999), the data consists of hour water demand data which includes 240 samples. The input features are taken into consideration such as day type, population, industrial operation, and water price, and the final one is meteorology. Finally, the researchers have used the fuzzy theory and ANN and they combined both as give an effective model prediction (ANFIS). |
| (Ren *et al.*, 2010) | Feedback Process neural networks, Feed-forward  neural networks. | This paper proposes a model for predicting water consumption in urban areas with the help of neural networks. This paper presents the urban water consumption forecast with the use of specific flowcharts and the researchers have used the feed forward neural network. The results show that the process of neural networks model for the water demand forecasting has certain advantages. |
| (Ren and  Li, 2016) | Empirical mode  decomposition  EMD, dynamic  architecture of  artificial neural networks DANN | Residents' quality of life consideration is an essential factor in this research paper. This water demand forecasting model is mainly based on short term demand forecasting, and this is a multi-scale approach for the distributed water supply networks. The researchers have Taken into consideration for this study consists of day wise demand of the shanghai city of one year (2014), and the input features are daily maximum and minimum temperature of the city factors of rainfall and wind power, and the output label for the model is water demand. The evolution of the models has done by Root Mean Square Error (RMSE), Empirical Mode Decomposition (EMD) and dynamic architecture of Artificial Neural Network (DANN) are used for the optimization of this system. Out of both, DANN has a good performance in this case. This system obtains a MAPE value of 0.8608 as the forecasting data, which is efficient for application. |

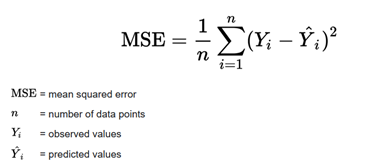
*Table 2: Tabular form of literature review.*

## 

Regression Evaluation Metrics

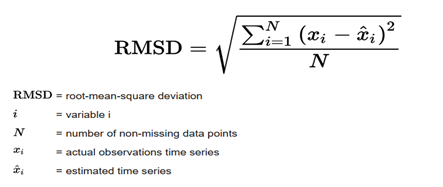
Once we are done with modelling the algorithms using the training dataset, we now need to move forward towards the testing of the dataset. For this purpose, we need to use the evaluation parameters metrics, which is considered very beneficial for evaluating our trained models. Since we are using the regression algorithms in this research work, we need to apply the regression evaluation parameter metrics commonly known as the loss functions since it provides the results in a loss. We have used two regression evaluation metrics in this code script: mean squared error and the root mean squared error. A brief definition is being provided here to explain these algorithms as well (Brownlee 2021).

**Mean Squared Error:** In general, mean squared error MSE is the average or mean of the square of the actual and predicted values difference. In supervised machine learning algorithms, the dataset contains the dependent or target variables and some of the independent variables. If the dependent variable seems numeric, then the regression models will be used to predict it, and MSE is efficiently used for model evaluation. It works by plotting a line among the data points that cover most of the data points, and the normalized distance between the data point and the line is known as an error that will help us get the MSE (Brownlee, 2021)



*Figure 2:* [*Mean square error*](https://www.google.com/search?client=firefox-b-d&q=mean+squared+error+formula)

**Root Mean Squared Error:** Root means square error (RMSE) is calculated as the square root of the mean errors obtained. It is one of the most used and popular evaluation parameters concerning evaluation parameters. RMSE is the standard deviation of the residuals, which is also known as the prediction errors. Residuals measure how far the data points lie from the regression line, and RMSE measures how spread out these residuals are. The RMSE value differs based on the situations and problem statements, so it is not good to fix a matter as the best one (Brownlee, 2021).



*Figure 3:* [*Root Mean Square Error*](https://www.google.com/search?q=root+mean+squared+error+formula&client=firefox-b-d&sxsrf=AOaemvLbb96TaqWvbmgjqkgC43VtcHDfhQ%3A1630233717386&ei=dWQrYbeDF4mC8gKVmLKwAg&oq=root+mean+squared+error+formula&gs_lcp=Cgdnd3Mtd2l6EAMyBAgAEEMyBAgAEAoyBAgAEAoyBAgAEAoyBAgAEAoyBAgAEAoyBggAEAoQHjIGCAAQChAeMggIABAIEAoQHjIGCAAQCBAeOgcIABBHELADOgcIABCwAxBDOgcIIxCwAhAnOggIABAHEAoQHjoGCAAQBxAeOgQIABANOgoIABAIEAcQChAeOggIABAIEAcQHkoECEEYAFCfoQ9Yu6UPYJOnD2gBcAJ4AIABcIgBqQOSAQMzLjKYAQCgAQHIAQrAAQE&sclient=gws-wiz&ved=0ahUKEwj3kYDthdbyAhUJgVwKHRWMDCYQ4dUDCA4&uact=5)

# 

# Chapter 3: Methodology

In this system, we will work with six algorithms, and as we have already discussed. So, first, we are loading the dataset in the code snippet, and then we will analyze the dataset to know more about the features and parameters of the dataset. After we are fully aware of the dataset, we need to perform the pre-processing data tasks and clean the data in the given time. After this is done, we will use the feature engineering methods and process to divide the dataset into training and testing datasets. Then, we will use the min-max scaler to scale the dataset. After the scaling of the dataset is done, we will apply the machine learning models to be trained through the training dataset. After the training, we will apply the testing parameters, which are the regression evaluation parameters, and at last, the best performing algorithm will be finally used in our system for final use.

## Tools and Libraries

For the successful implementation of the project, we are working on Python3 as our programming language, which is a version of Python and is very efficient compared to the other versions of Python. The code is being executed in the Jupyter notebook environment and is very efficient to work on a python background. Some of the essential libraries that have been used in the execution of this project are as discussed here:

**Python:** Python is a prevalent and efficient language among those used for machine learning purposes. It is available as open-source and can be used for multiple purposes as well. Python can be used as a web framework development, software application, and automation systems development language. This project has been implemented with the help of Python, and Python is the choice of most programmers and machine learning engineers when working on a machine learning project. It is because Python provides an extensive amount of powerful tools and libraries. Moreover, Python is a user-friendly language and is very easy to learn and understand (Wikipedia, 2001).

**Anaconda:** Anaconda is a framework that provides multiple working environments for Python and R. It is a very popularly used software, and almost every machine learning engineer uses this application. It contains two of the world's most used python environments, the Jupyter notebook in which we are working, and the Spyder is the second most used environment after the Jupyter notebook (Bhanot).

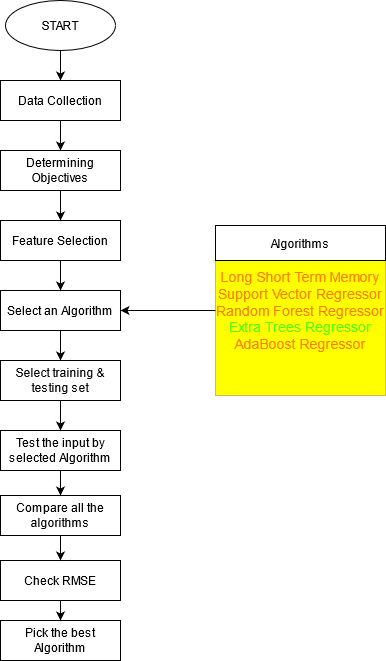
**Jupyter Notebook:** It has been voted as the most used python framework for machine learning projects. It is a compelling and user-friendly framework that serves a lot of purposes. We have chosen to work with this framework because it provides a unique way of giving the output below the code snippet, which is a handy feature and very helpful in working with machine learning (Wikipedia, 2018).

**NumPy:** It is an open-source python library mainly used to deal with arrays and numbers and is very powerful compared to the other algorithms. It is an alternative to Python's list in one of the functionalities but is very much faster. It has a set of other libraries and tools, which helps NumPy work within machine learning (Dataquest, 2020).

**Pandas:** It is one of the most used and most powerful python libraries, which is very useful for loading the dataset and analyzing it. It provides us with a set of other toolkits as well, which makes it very powerful and helpful. Pandas can be used in any machine learning project of Python with ease and are very efficient compared to other ways (Dataquest, 2020).

**Matplotlib:** matplotlib is a compelling and open-source python library that is very useful in visualizing the dataset and helps us in studying the dataset in a very efficient manner. It provides us with many other tools and libraries that come in very handy while dealing with visualization techniques. It also gives us the geographical, geospatial, heatmaps, bar graphs, plots, etc., which are very helpful in visualizing the dataset. The PyPlot library provided by matplotlib provides a MATLAB-like architecture to work with that helps better implement the plots and graphs (Dataquest, 2020).

**Scikit-Learn:** This is a compelling open-source library, which is also known as scikit Library. It is very commonly used for data pre-processing and feature engineering. It also helps in scaling the dataset using the min-max scaler in this case. This library also helps train and test the models in the project research work, which is very useful for this system (Dataquest,2020)



*Figure 4: Workflow Graph*

# Chapter 4: Implementation, results, discussion & analysis

In this chapter, we will go through the implementation of the code and explain the code in detail. We will use machine learning and a deep learning algorithm to forecast the water demands. There are four steps to solve this problem.

1. The first step is to perform data analysis in great depth to understand the data and its patterns.

2. The second step is to perform data pre-processing, in which we prepare the inputs and the target features. In this dataset, we take one input ml models and deep learning models.

3. The third step is to apply several machine learning and deep learning algorithms to make the forecasting. Some of the algorithms that we use are Support vector regression, random forest regression, extra trees regressor, and some boosting algorithms like Adaboost regressor, Long short Term Memory (LSTM), and the final one is Artificial Neural Network (ANN).

4. The fourth and last step is to evaluate the model and make the prediction. The evaluation metrics that we are going to use are Mean Squared Error and Root Mean Squared Error.

## Dataset Collection

The data set has collected data from the IEEE Data-Port, which consists of information about water consumption of different households in different instances. Data is collected from the tenants and neighbours in the Maadi district in Egypt, along with GPS information. The data set used in this project was downloaded from the IEEE Data Port by providing login credentials in the link<https://ieee-dataport.org/open-access/water-resources-management>

This data set is created by Sherif Elsayed Hussein (Hussein, 2020), and that is available to the public for the research purpose. The research work using this data set must cite the data set in their reference. The proposed work cited the data set as a reference.

## Data Exploration and Analysis

The first step is to look through the data, and we have found out that the data is recorded every single minute of a 24-hour period. So, there are a total of 1440 data points recorded every single day. There are a total of 20 CSV files present in the dataset. There are 4 full columns present in each CSV file. All the columns are of integer type. One of the columns is the date timestamp column. Next, we check the shape of each CSV file, their value counts distribution, maximum, minimum, and the last is their difference between maximum and minimum.

Next, we select only 3 CSV as a sample for future exploration. The files that we will take are 111, 211, 311 are chosen as training files, and 112, 212, 312 are designated as testing files.

Here we perform data pre-processing. So, for that, we must create a function called read\_df, which will do the resamples and save the training dataset and the test dataset. This function first reads the CSV dataset, and then we assign a header to all 4 columns which is ID, time, water consumption, and unknown. Next, we drop the anonymous columns, and then we set the time columns as the index. Now we resample the dataset into a one-day time period (D), and then we check for the missing values in the dataset and fill with the backward fill function, and last convert the ID column into the integer types.

### Resampling

The procedure of resampling entails taking multiple samples from the original data samples. Based on the actual data, resampling generates a unique sampling distribution. To generate the unique sample distribution, the resampling approach uses experimental methods rather than analytical ones. We need a **DateTime** type index or column to do the resampling method ("Resample" 2019).

Syntax:

resample(arguments)

Eg: data.resample('D').sum()

Here, we can observe from the above step. We resample all the data points into the one day time period. Where ‘D’ indicates the day time stamp.

### bfill method for NaN's replacement

The function bfill() is used to backfill the dataset's missing values. It will fill the NaN values in the panda's data frame in reverse order.

With the method='ffill' option, you can use pandas.DataFrame.fillna. The term 'ffill' refers to 'forward fill,' and it propagates the most recent valid observation forward. The alternative is 'bfill,' which functions in the same way as 'fill,' but in the opposite direction (Stackoverflow, n.d.).

### MinMaxScaler for Normalization

One of the most prevalent methods of data normalization is min-max normalization. The minimum value of each feature is converted to a 0, the maximum value is converted to a 1, and all other values are converted to a decimal between 0 and 1.

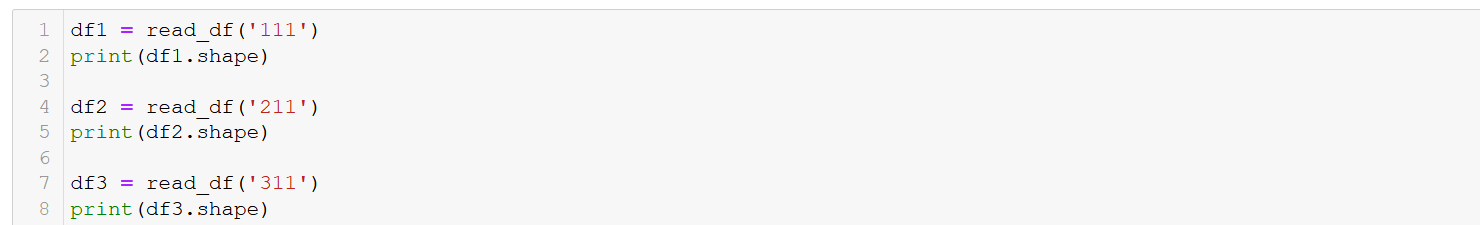
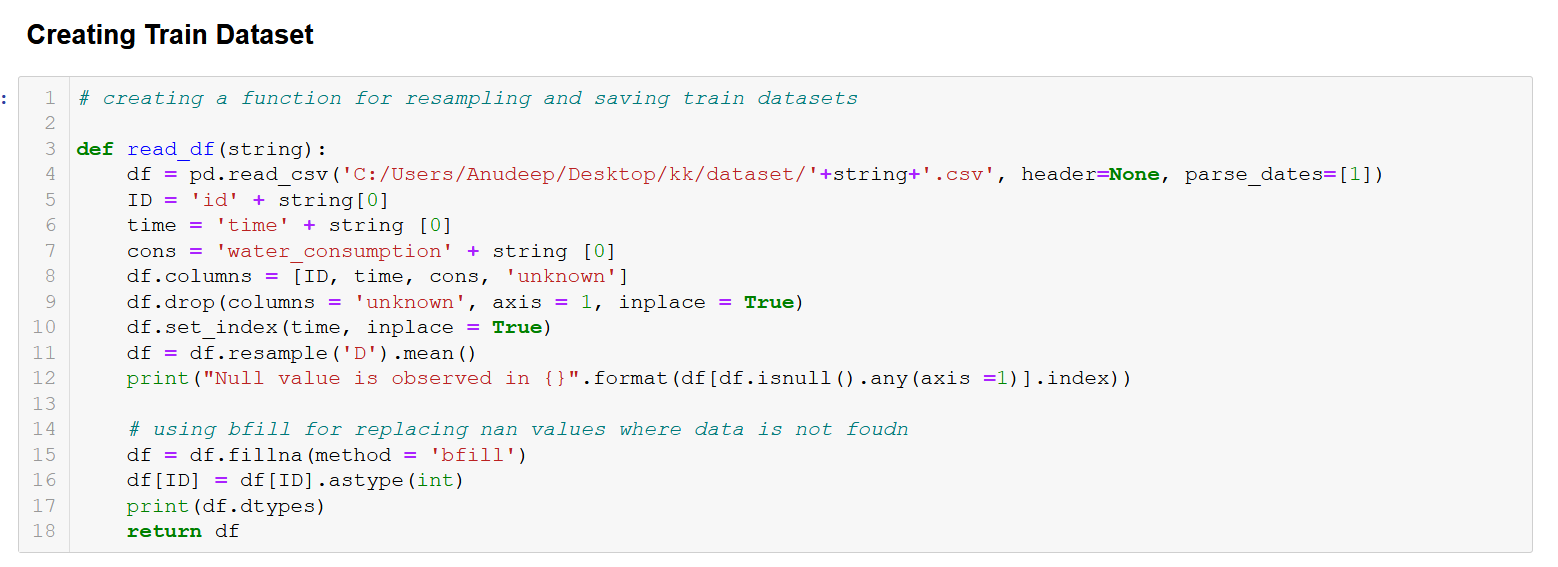
The most common way to "**normalize**" a vector is to divide it by its norm. It also frequently refers to rescaling by the vector's minimum and range to make all elements sit between 0 and 1, shifting all numeric column values in the dataset to a single scale ("Replacing NaN", n.d.).

### Lag function

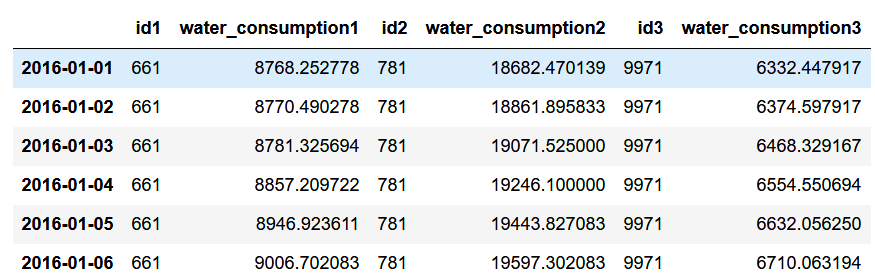
Because modelling time series data can be difficult, it's understandable that most of the data enthusiasts put off learning about it until they must. Before we can apply machine learning models to time series data, we must convert it to an “absorbable” format for our models, which frequently entails generating delayed variables, which can quantify auto-correlation, or how previous values of a variable influence with future values.

lag one:

After we set the date column as an index, we simply use the shift technique available to the data frame and indicate the number of steps to lag (in our Modelling, we use one step shift), and we can even use a negative number as the shift, which would signify those future values are influencing the past (Towards Data Science, 2019).



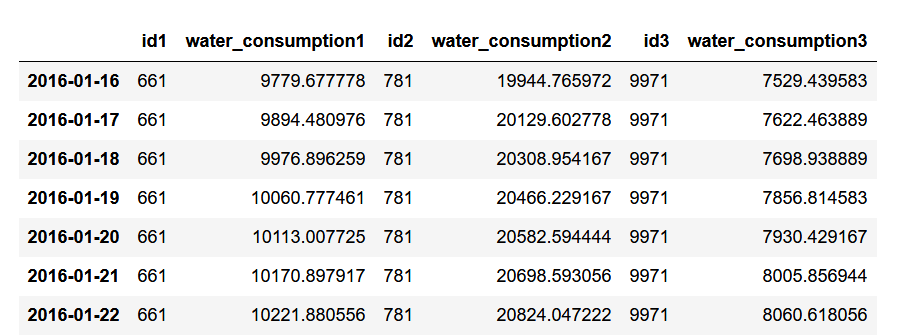
Now we concatenate all three data frames into ones. The processed data looked as follows:



*Figure 5: the sample of training dataset.*

Now we create another data frame known as train\_df\_cum. The resultant output of that column is the sum of water consumption 1, water consumption 2, and water consumption 3. And then, we check the shape of new data frames, which are 15 rows and 1 column. Moreover, at last, we review the info of the training dataset.

Now we repeat the same process for the testing dataset also.



*Figure 6: the sample of testing dataset.*

Now we create copies of both data frames and assign them to train and test data frames. Next, we use the help of the matplotlib library and plot the training data.

Chart, line chart

Description automatically generated

*Figure 7: The graph of training data and its water consumption in m3 (cubic meters)*

The above figure illustrates. That water consumption in the number of days and its consumption volume in cubic meters (m3), one cubic meter equals 1000 liters. Here we consider the three house's water consumption data with the first 15 days of a Month (01-01-2016 to 15-01-2016) into Training the model.

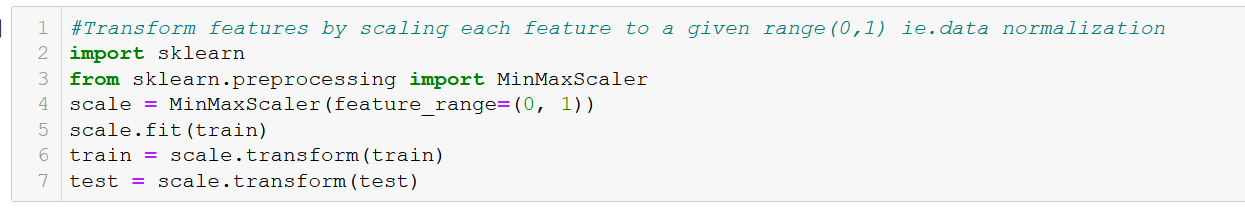
Chart, line chart

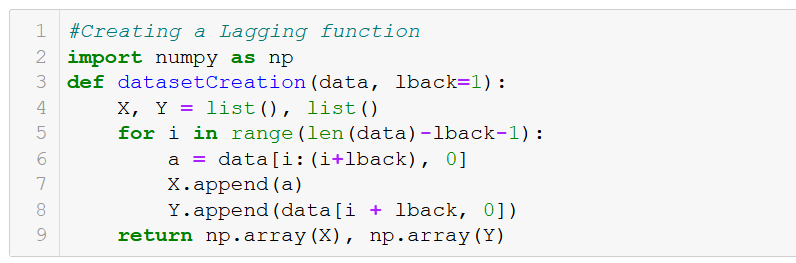
Description automatically generated

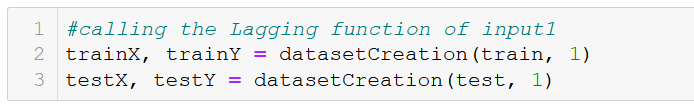
*Figure 8: The graph of testing data and its water consumption in m3 (cubic meters)*

The above figure illustrates. That water consumption in the number of days and its consumption volume in cubic meters (m3), one cubic meter equals 1000 liters. Here we consider the three house's water consumption data with the second 15 days of a Month (16-01-2016 to 31-01-2016) into Testing the model.

Now we perform data normalization on the training and test data with the help of the MinMax scaler function. Now we are at the last step of data pre-processing, creating a dataset creation function.

Using this function, we create various inputs:

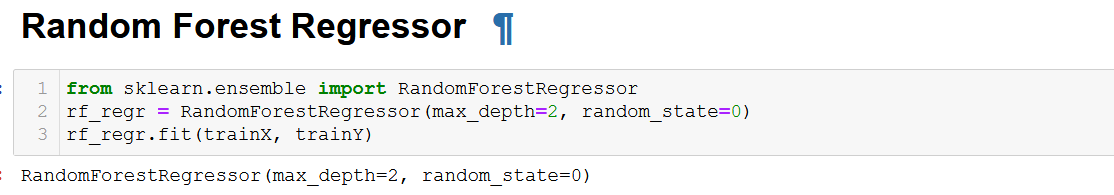


The first one is that we make one input dataset in which we apply machine learning algorithms. Next, we reshape this dataset and use it for a deep learning algorithm, which is LSTM and ANN.

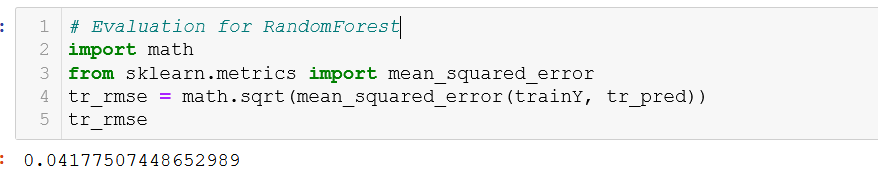
## Applying machine learning and deep learning algorithms

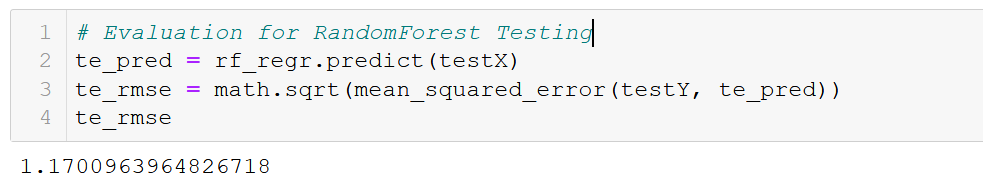
In this step, we build the machine learning and deep learning models, and at last, we compare the results and find out which models perform the best among all.

The first model that we will choose is the **Random Forest Regressor** model with parameters max depth equal to 2, and random state is set equal to zero. Then we fit the model on training data which is trainX and trainY. Next, we predict the result of testing data. At last, we evaluate the model on different types of input using **Root** **Mean Squared Error**.

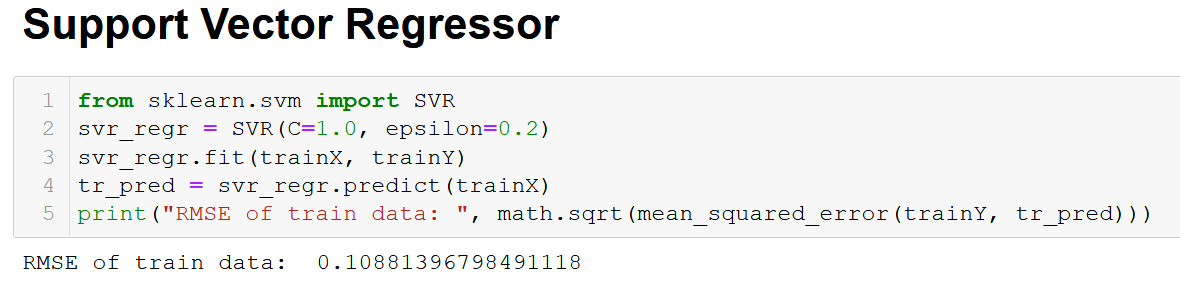


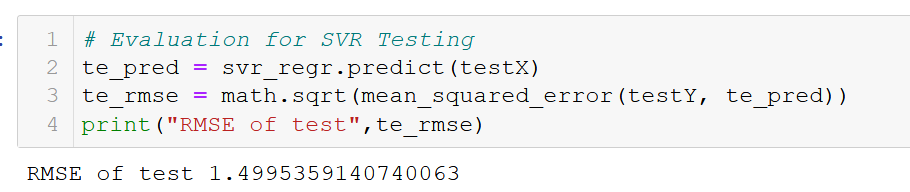
For the training data, the RMSE score that we are getting on one input datais 0.04177. Moreover, for the testing, we are getting 1.17.



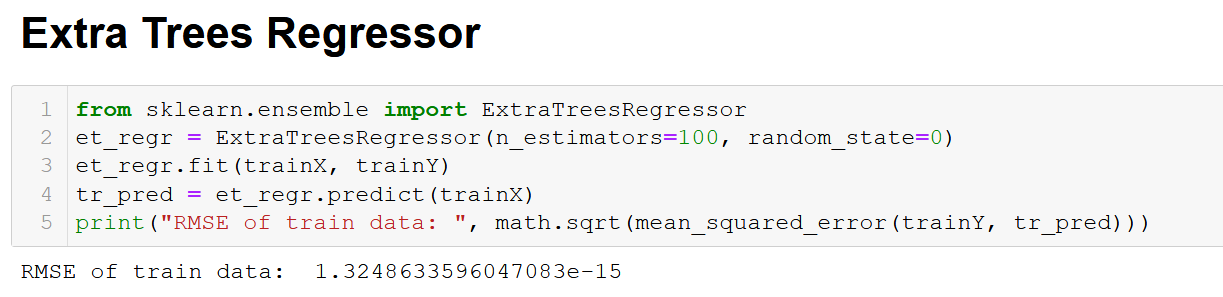


The second model that we implement is the **Support Vector Regressor** model with parameters C value equal to 1.0 and epsilon is set equal to 0.2. Then we fit the model on training data which is trainX and trainY. Next, we predict the result of testing data. At last, we evaluate the model on different types of input using **Root Mean Squared Error**.



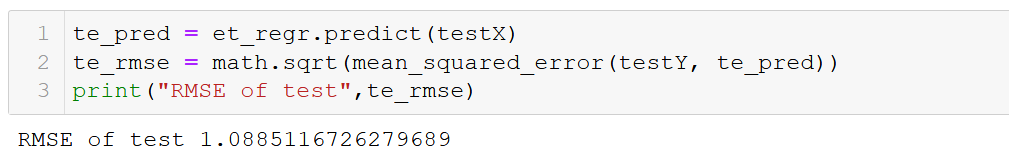
For the training data, the RMSE score that we are getting on one input data is 0.1088. Moreover, for the testing, we got 1.499.

The third model that we will choose is the **Extra Tree Regressor** model with parameters number of the estimator is equal to 100, and random state is set equal to 0. Then we fit the model on training data which is trainX and trainY. Next, we predict the result of testing data. At last, we evaluate the model on different types of input using **Root** **Mean Squared Error**.

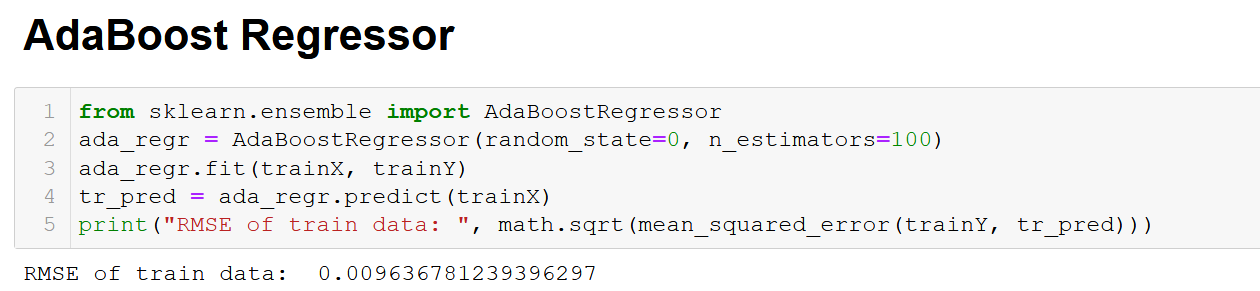


For the training data, the RMSE score that we are getting on one input data is 1.3248633596047083e-15 ~ 0.000000001

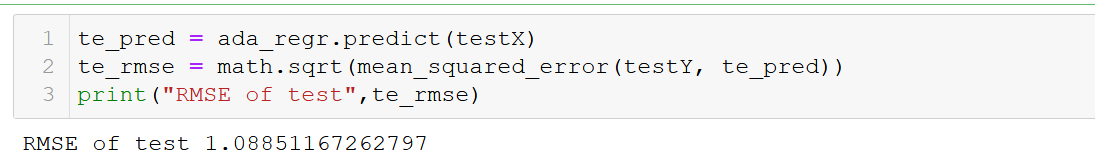
Moreover, for the testing, we got 1.088.



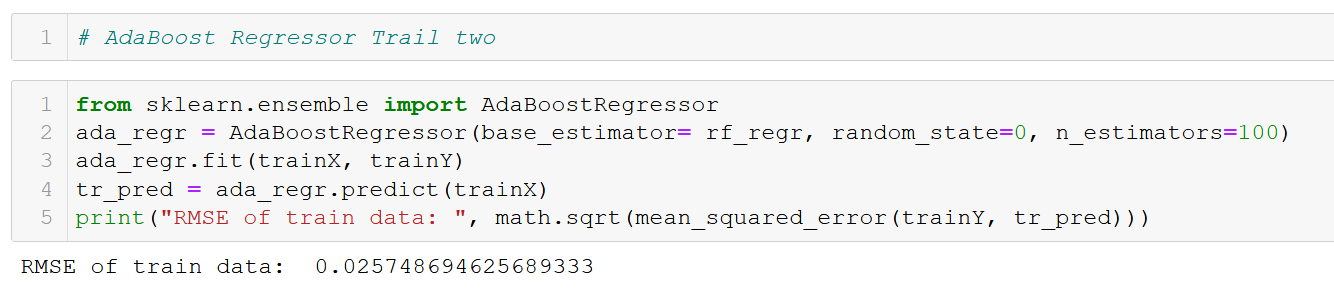
The fourth model that we will choose is the **AdaBoost Regressor** model with parameters no of the estimator equal to 100, and random state is set equal to 0. Then we fit the model on training data which is trainX and trainY. Next, we predict the result of testing data. At last, we evaluate the model on different types of input using **Root** **Mean Squared Error**.



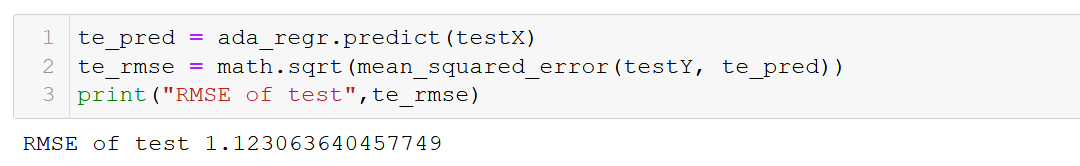
For the training data, the RMSE score that we are getting on one input data is 0.0096. Moreover, for the testing, we got 1.088.



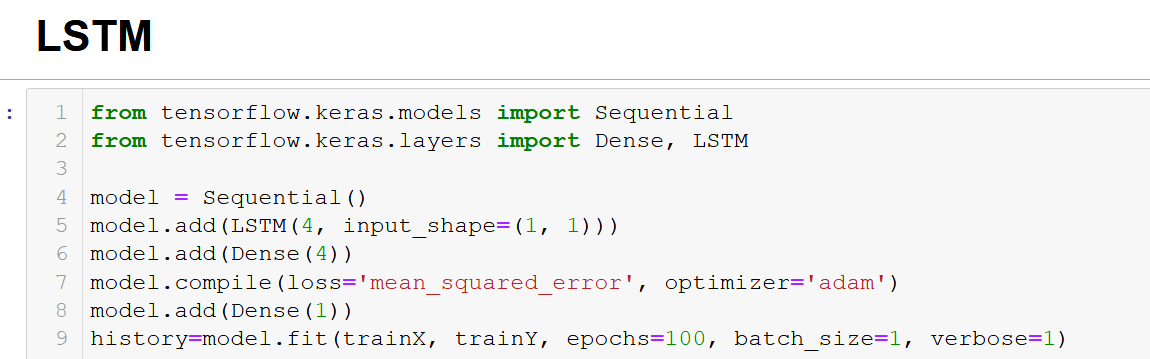
The fifth model that we use is the **AdaBoost Regressor** model with parameters **base estimator as random forest regressor**, no of the estimator is equal to 100 and random state is set equal to 0. Then we fit the model on training data which is train and trainY. Next, we predict the result of testing data. At last, we evaluate the model on different types of input using **Root** **Mean Squared Error**.

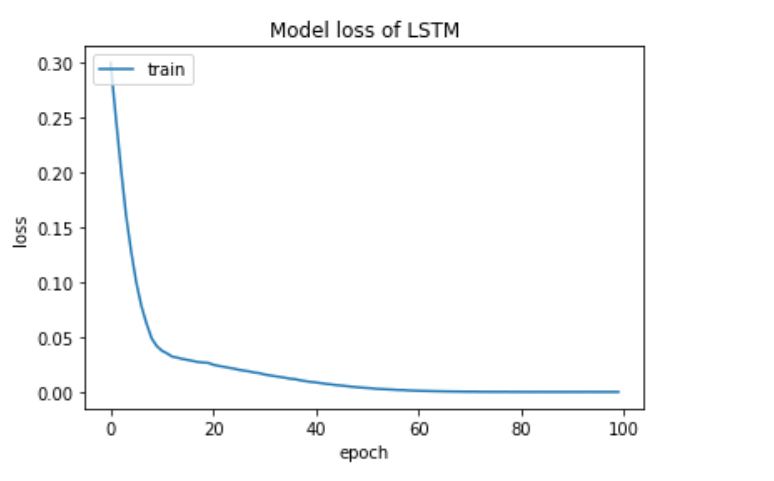


For the training data, the RMSE score that we are getting on one input data is 0.0257. Moreover, for the testing, we got 1.12

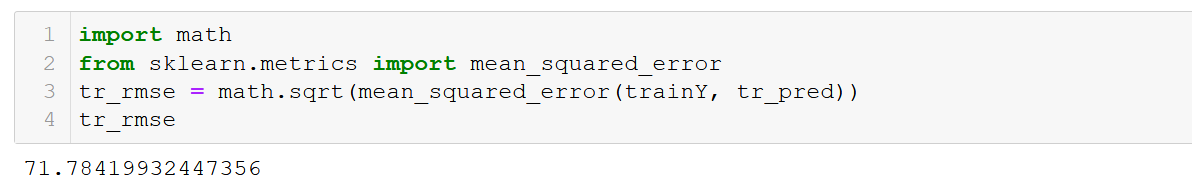


The first Deep learning algorithm that we are going to use is **LSTM**. First, we initialize the layer by calling the Sequential module. Next, we define the filter size as 4, and the input shape is (1,1). After that, we restrict the output layer by calling it the Dense layer. Then we compile the model with loss as mean squared error and optimizer as Adam. Next, we fit the model on a training dataset, epochs are set to 100, and a batch size of 1 and set verbose is equal to 2. We evaluate the model on different types of input using **Root** **Mean Squared Error**.

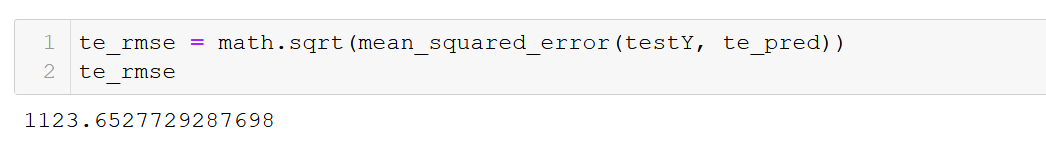


We trained LSTM with 1 hidden unit. A lower number of units is used so that it is less likely that LSTM would perfectly memorize the sequence. We use the Mean Square Error loss function and Adam optimizer. We train the model with 1 sequence per batch for 100 epochs. From the plot below, we can observe that training loss continues equally after the 60 epochs.

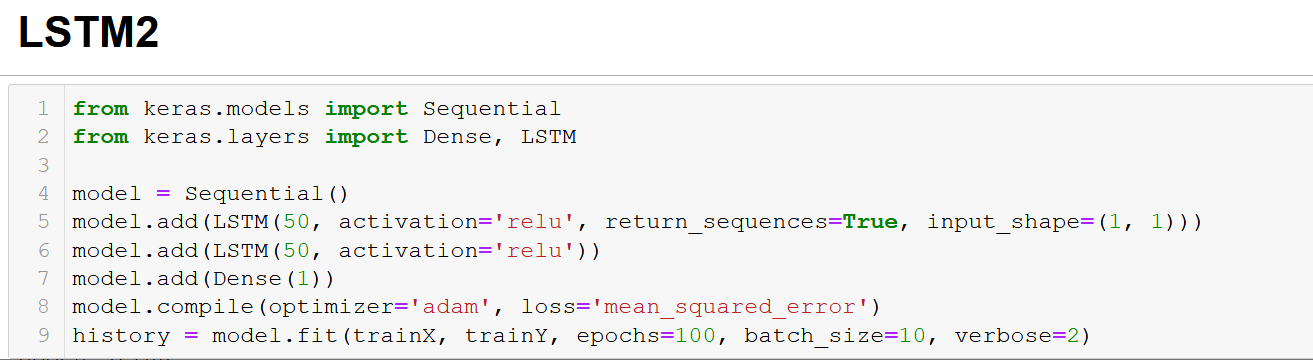
*Figure 9: The LSTM1 Model loss graph*

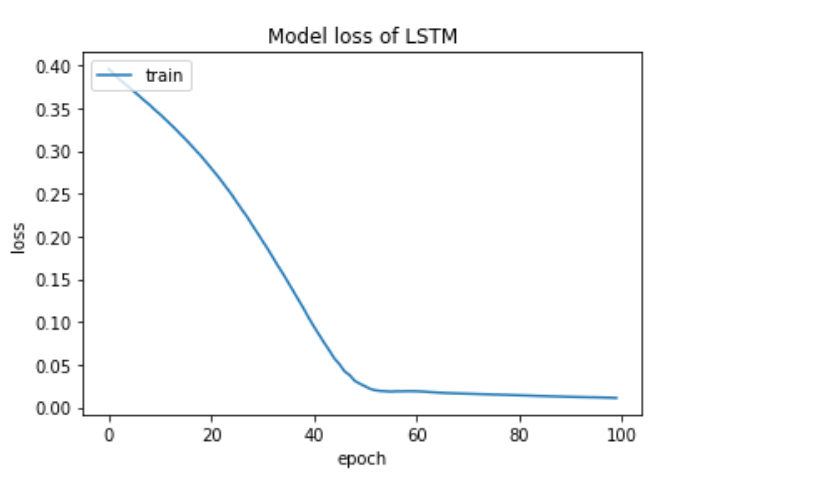
For the training data, the RMSE score that we got on one input data is 71.784.

Moreover, for the testing, we are getting 1123.6527.



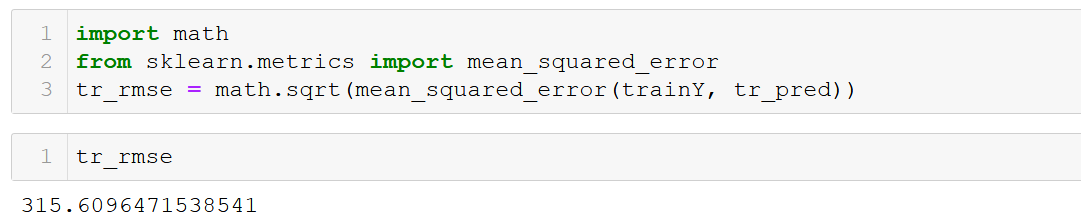
Now for **LSTM**, few changes were made. First, we initialize the layer by calling the Sequential module. Next, we define the filter size as 50, and the input shape is (1,1) with return\_sequences=**True**. Addition to that one more layer with filter size as 50. After that, we restrict the output layer by calling it the Dense layer. Then we compile the model with loss as mean squared error and optimizer as Adam. Next, we fit the model on a training dataset, epochs are set to 100, and a batch size of 10 and set verbose is equal to 2. We evaluate the model on different types of input using **Root** **Mean Squared Error**.

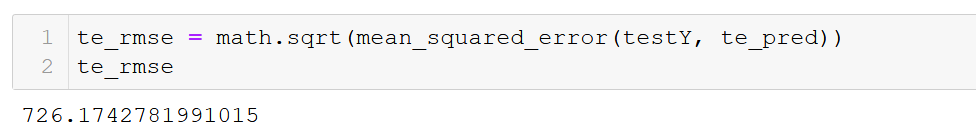


We trained LSTM with 2 hidden units. A lower number of units is used so that it is less likely that LSTM would perfectly memorize the sequence. We use the Mean Square Error loss function and Adam optimizer 'relu' Activation function. We train the model with 10 sequences per batch for 100 epochs. From the plot below, we can observe that Model loss continues equally from 80 epochs.

*Figure 10: The LSTM2 Model loss graph*

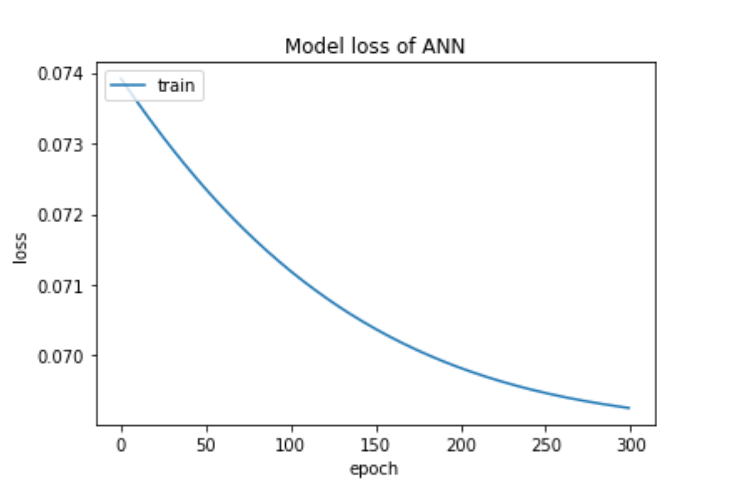
For the training data, the RMSE score that we got on one input data is 315.609



Moreover, for the testing, we got 726.174

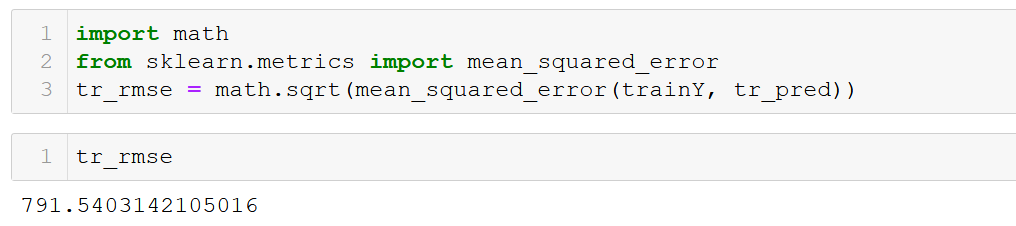
The second Deep learning algorithm that we are going to use is **ANN**. First, we initialize the layer by calling the Sequential module. Next, we define the Dense layer of size 12 with activation function **‘relu**’, Next three layers are Dense layers of size 8, 4, 1 respectively and the last layer of Dense with activation function ‘**sigmoid’.** Then we compile the model with loss as mean squared error and optimizer as Adam. Next, we fit the model on a training dataset, epochs are set to 300. We evaluate the model on different types of input using **Root** **Mean Squared Error**.

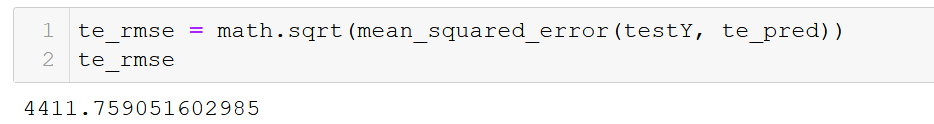


We trained ANN with 4 hidden units. A lower number of units is used so that it is less likely that LSTM would perfectly memorize the sequence. We use the Mean Square Error loss function and Adam optimizer 'relu', and 'sigmoid' Activation functions. We train the model with 300 epochs. From the plot below, we can observe that Model loss continues equally from 300 epochs.

*Figure 11: The ANN Model loss graph*

For the training data, the RMSE score that we got on one input data is 791.54



Moreover, for the testing, we got 4411.75

Day prediction for the combined houses for a particular area. As we can observe from the below table, Extra Trees regressor give better RMSE compared to the other algorithms.

Now we compare all the various algorithms by making a comparison table:

|  |  |
| --- | --- |
| Algorithms used | RMSE for the testing value obtained |
| Random forest regressor | 1.17 |
| Support vector regressor | 1.499 |
| Extra trees regressor | 1.08 |
| Adaboost regressor | 1.088 |
| Adaboost regressor 2 | 1.123 |
| Artificial Neural Network | 4411.75 |
| Long Short term memory1 | 1123.65 |
| Long Short term memory 2 | 726.174 |

*Table 12: comparison table of the algorithms for a particular area day water consumption prediction.*

Day prediction for the single houses (661) for a particular area. As we can observe from the below table, Random Forest regressor give better RMSE compared to the other algorithms.

Now we compare all the various algorithms by making a comparison table:

|  |  |
| --- | --- |
| Algorithms used | RMSE for the testing value obtained |
| Random forest regressor | 0.032 |
| Support vector regressor | 1.506 |
| Extra trees regressor | 1.086 |
| Adaboost regressor | 1.086 |
| Adaboost regressor 2 | 1.128 |
| Artificial Neural Network | 835.469 |
| Long Short term memory | 288.845 |
| Long Short term memory 2 | 244.382 |

*Table 13: comparison table of the algorithms for a single house day water consumption prediction.*

5 Minutes prediction for the combined houses for a particular location. As we can observe from the below table, Extra trees regressor give better RMSE compared to the other algorithms.

Now we compare all the various algorithms by making a comparison table:

|  |  |
| --- | --- |
| Algorithms used | RMSE for the testing value obtained |
| Random forest regressor | 1.125 |
| Support vector regressor | 1.433 |
| Extra trees regressor | 0.981 |
| Adaboost regressor | 1.05 |
| Adaboost regressor 2 | 1.124 |
| Long Short term memory | 736.477 |
| Long Short term memory2 | 795.382 |

*Table 14: comparison table of the algorithms for a particular location, for every 5 minutes water consumption prediction.*

To clarify, we must use the best algorithm like the one with the least RMSE score.

# 

# 

# Chapter 5: Conclusion and Future Work

This is critical research work, and this topic needs attention as water demand forecasting is a solution for the significant problems related to water discrimination on earth. This problem can be solved using a perfect plan execution and with the help of artificial intelligence. In this project, we have tried to forecast the water demand of the household. In this research work, we have used six algorithms, and we evaluated them based on their efficiency using the RMSE value. The algorithms that we have used are support vector regression, random forest regression, extra trees regression, and the AdaBoost regressor, long short-term memory, and Artificial neural network (ANN). Out of these different trees, the Extra tress Regressor algorithm performs the best and gives the 1.089 loss function result through the RMSE evaluation parameter. Hence, we can use these Extra trees Regressor algorithms in the final system.

This project covers almost the forecasting of the water demands in households and covers how we can save water in household uses. However, there are two more ways to wastewater: agricultural and industrial means. These three are the most popular means of wastage of water. As a part of future work, we plan to cover the water demands forecasting through the agricultural and industrial fields. Once all the major three areas are covered, we tend to use hybrid algorithms as well, and we will see to it if they improve the efficiency and accuracy of the algorithm.

# 

# References

Alvi, M. S. Q. *et al.* (2019) 'Dynamic behavioural modeling, simulation and analysis of household water consumption in an urban area: A hybrid approach', *Proceedings - Winter Simulation Conference*, 2018-December(Moon 2016), pp. 2411–2422. doi: 10.1109/WSC.2018.8632309.

Chang, K. *et al.* (2008) 'Urban daily water demand short-term forecasts based on the chaos

theory', Proceedings - 4th International Conference on Natural Computation, ICNC 2008, 4, pp. 696–699. doi: 10.1109/ICNC.2008.623.

https://towardsdatascience.com/timeseries-data-munging-lagging-variables-that-are-distributed-across-multiple-groups-86e0a038460c

Chang, M. and Liu, J. (2009) 'Water demand prediction model based on radial basis function neural network', 2009 1st International Conference on Information Science and Engineering, ICISE 2009, 2(1), pp. 5295–5298. doi: 10.1109/ICISE.2009.1343.

Dong, Y. and Zhou, W. (2009) 'Urban residential water demand forecasting in Xi'an based on RBF model', *2009 International Conference on Energy and Environment Technology, ICEET 2009*, 2, pp. 901–904. doi: 10.1109/ICEET.2009.456.

Ji, G. *et al.* (2014) 'Urban water demand forecasting by LS-SVM with tuning based on elitist teaching-learning-based optimization', *26th Chinese Control and Decision Conference, CCDC 2014*, pp. 3997–4002. doi: 10.1109/CCDC.2014.6852880.

Kofinas, D. *et al.* (2016) 'Daily multivariate forecasting of water demand in a touristic island with the use of artificial neural network and adaptive neuro-fuzzy inference system', *2016 International Workshop on Cyber-physical Systems for Smart Water Networks, CySWater 2016*, pp. 37–42. doi: 10.1109/CySWater.2016.7469061.

Li, X., Ding, L. and Lv, J. (2010) 'Agriculture irrigation water demand forecasting based on rough set theory and weighted LS-SVM', *2010 2nd International Conference on Communication Systems, Networks and Applications, ICCSNA 2010*, 2, pp. 371–374. doi: 10.1109/ICCSNA.2010.5588826.

Liu, H., Deng, T. and Zhang, H. (2009) 'Research on forecasting method of urban water demand based on fuzzy theory', *6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2009*, 6, pp. 389–395. doi: 10.1109/FSKD.2009.639.

Mkireb, C. *et al.* (2018) 'A linear programming approach to optimize demand response for water systems under water demand uncertainties', *2018 International Conference on Smart Grid and Clean Energy Technologies, ICSGCE 2018*, pp. 206–211. doi: 10.1109/ICSGCE.2018.8556696.

Rahim, M. S. *et al.* (2019) 'Predicting Household Water Consumption Events: Towards a Personalised Recommender System to Encourage Water-conscious Behaviour', *Proceedings of the International Joint Conference on Neural Networks*, 2019-July(July), pp. 1–8. doi: 10.1109/IJCNN.2019.8851868.

Rajabi, R. and Estebsari, A. (2019) 'Deep learning based forecasting of individual residential loads using recurrence plots', *2019 IEEE Milan PowerTech, PowerTech 2019*, pp. 1–5. doi: 10.1109/PTC.2019.8810899.

Ren, Y. *et al.* (2010) 'Based on process neural network learning algorithm for prediction of urban water consumption', *2010 International Conference on Computing, Control and Industrial Engineering, CCIE 2010*, 2(1), pp. 34–37. doi: 10.1109/CCIE.2010.127.

Ren, Z. and Li, S. (2016) 'Short-term demand forecasting for distributed water supply networks: A multi-scale approach', *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, 2016-September, pp. 1860–1865. doi:

10.1109/WCICA.2016.7578578.

Roux, M. and Booysen, M. J. (2017) 'Use of smart grid technology to compare regions and days of the week in household water heating', *Proceedings of the 25th Conference on the Domestic Use of Energy, DUE 2017*, (April), pp. 276–283. doi:

10.23919/DUE.2017.7931855.

Sale, H. *et al.* (2018) 'Prototype for estimation and forecasting of the future demand and generation from households in selected European countries', *Proceedings - 2018 53rd International Universities Power Engineering Conference, UPEC 2018*, pp. 1–6. doi: 10.1109/UPEC.2018.8542004.

Shabri, A. and Samsudin, R. (2014) 'A new approach for water demand forecasting based on empirical mode decomposition', *2014 8th Malaysian Software Engineering Conference, MySEC 2014*, pp. 284–288. doi: 10.1109/MySec.2014.6986030.

Sossan, F. *et al.* (2013) 'Scheduling of domestic water heater power demand for maximizing PV self-consumption using model predictive control', *2013 4th IEEE/PES Innovative Smart Grid Technologies Europe, ISGT Europe 2013*, pp. 4–8. doi:

10.1109/ISGTEurope.2013.6695317.

IBM. (2020, 5 1). Deep Learning. IBM Cloud Education. Retrieved 07 09, 2021, from https://www.ibm.com/cloud/learn/deep-learning

IBM. (2020, 08 19). Supervised Learning. ibm.com. Retrieved 07 08, 2021, from https://www.ibm.com/cloud/learn/supervised-learning

IBM. (2020, 09 21). Unsupervised Learning. IBM Cloud Education. Retrieved 07 09, 2021, from https://www.ibm.com/cloud/learn/unsupervised-learning

Minsky, M. (1961). Steps toward Artificial Intelligence. in Proceedings of the IRE, 49(1), 8-30. Retrieved 06 03, 2021, from https://ieeexplore.ieee.org/abstract/document/4066245

Stuhlmacher, A. and Mathieu, J. L. (2020) 'Chance-Constrained Water Pumping to Manage Water and Power Demand Uncertainty in Distribution Networks', *Proceedings of the IEEE*, 108(9), pp. 1640–1655. doi: 10.1109/JPROC.2020.2997520.

'Water \_ Free Full-Text \_ Machine Learning and Data Analytic Techniques in Digital Water Metering\_ A Review \_ HTML' (no date).

'Water Demand \_ Urban Waters, Bengaluru' (no date).

Wei, X. (2007) 'Forecasting of urban water demand based on chaos theory', *Proceedings of the 26th Chinese Control Conference, CCC 2007*, pp. 441–444. doi:

10.1109/CHICC.2006.4347563.

Xu, X. and Liu, J. (2009) 'The research on forecasting water demand methods for Changzhi city in Shanxi Province', *2009 1st International Conference on Information Science and Engineering, ICISE 2009*, (4), pp. 5261–5264. doi: 10.1109/ICISE.2009.1278.

Xu, Y. S., Mei, Y. D. and Yong, T. (2011) 'Combined forecasting model of urban water demand under changing environment', *2011 International Conference on Electric Technology and Civil Engineering, ICETCE 2011 - Proceedings*, pp. 1103–1107. doi: 10.1109/ICETCE.2011.5775448.

Zeng, M. and Song, Y. (2011) 'Water demand forecasting in Qinzhou, China', *ISWREP 2011 - Proceedings of 2011 International Symposium on Water Resource and Environmental Protection*, 1, pp. 779–782. doi: 10.1109/ISWREP.2011.5893123.

Zhai, C., Zhang, H. and Zhang, X. (2009) 'Application of system dynamics in the forecasting water resources demand in Tianjin Polytechnic University', *2009 International Conference on Artificial Intelligence and Computational Intelligence, AICI 2009*, 1, pp. 273–276. doi: 10.1109/AICI.2009.272.

Zhang, L., Wei, Y. and Wang, Z. (2008) 'Prediction on ecological water demand based on support vector machine', *Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008*, 5(x), pp. 1032–1035. doi: 10.1109/CSSE.2008.442.

Zhang, X. H. *et al.* (2008) 'A Nonlinear dynamics approach for forecasting urban water resources demand', *2008 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2008*, pp. 1–4. doi: 10.1109/WiCom.2008.2733.

Zhou, J. (2010) 'General regression neural network forecasting model based on PSO

algorithm in water demand', *2010 3rd International Symposium on Knowledge Acquisition and Modeling, KAM 2010*, (1), pp. 51–54. doi: 10.1109/KAM.2010.5646238.

Zubaidi, S. L. *et al.* (2019) 'The prediction of municipal water demand in iraq: A case study of baghdad governorate', *Proceedings - International Conference on Developments in eSystems Engineering, DeSE*, October-2019, pp. 274–277. doi: 10.1109/DeSE.2019.00058.

'Water \_ Free Full-Text \_ Machine Learning and Data Analytic Techniques in Digital Water Metering\_ A Review \_ HTML' (no date).

'Water Demand \_ Urban Waters, Bengaluru' (no date).

Bhanot, Karan. Mar 31, 2020. "Anaconda: Start here for data science in Python!" Towards Data Science.<https://towardsdatascience.com/anaconda-start-here-for-data-science-in-python-475045a9627>.

Biau, Gerard. 2012. "Analysis of a Random Forests Model." Journal of Machine Learning Research 13 6:33.<https://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf>.

Brownlee, Jason. 2021. "Regression Metrics for Machine Learning." Machine Learning Mastery.<https://machinelearningmastery.com/regression-metrics-for-machine-learning/>.

Dataquest. 2020. "15 Python Libraries for Data Science You Should Know." Dataquest.<https://www.dataquest.io/blog/15-python-libraries-for-data-science/>.

Debasish, Basak, and Pal Srimanta. 2007. "Support Vector Regression." Neural Information Processing – Letters and Reviews 11 (11).<https://www.researchgate.net/publication/228537532_Support_Vector_Regression>.

Dimitri, Solomatine, and Shrestha Durga. 25/08/2004. AdaBoost.RT: A boosting algorithm for regression problems. Vol. 2. N.p.: IEEE International Conference on Neural Networks - Conference Proceedings.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel. 2006. "Extremely randomized trees." Machine Learning 63 (1).<https://link.springer.com/article/10.1007/s10994-006-6226-1#citeas>.

Hochreiter, Sepp, and Jurgen Schmidhuber. 1997. “LONG SHORT-TERM MEMORY.” Neural Computation 9 (8): 1735-1780.<https://www.bioinf.jku.at/publications/older/2604.pdf>.

Hussein, Sherif Elsayed. 2020. "WATER RESOURCES MANAGEMENT." IEEEDataport.<https://ieee-dataport.org/open-access/water-resources-management>.

Wikipedia. 2018. "Project Jupyter."<https://en.wikipedia.org/wiki/Project_Jupyter>.

Wikipedia. 29 October 2001‎. "programming language." Wikipedia.<https://en.wikipedia.org/wiki/Python_(programming_language)>.

"Normalization." n.d. codecademy. Accessed 07 07, 2021.

<https://www.codecademy.com/articles/normalization>.

Stackoverflow. n.d. "How to replace NaNs by preceding or next values in pandas DataFrame?" Accessed 06 03, 2021.<https://stackoverflow.com/questions/27905295/how-to-replace-nans-by-preceding-or-next-values-in-pandas-dataframe>.

"Time Series Data Munging — Lagging Variables that are Distributed Across Multiple Groups." 2013. Towards Data Science.<https://towardsdatascience.com/timeseries-data-munging-lagging-variables-that-are-distributed-across-multiple-groups-86e0a038460c>.

"Using the Pandas "Resample" Function." 2019. Towards Data Science.<https://towardsdatascience.com/using-the-pandas-resample-function-a231144194c4>.

# Appendix

**\*\*oneinput-ml-day-main.ipynb\*\***

import pandas as pd

import numpy as np

import os

\*\*Please place all the data files in a folder named 'data' at a location where this notebook file is place\*\*

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/111.csv', header=None, parse\_dates = [1])

df.head()

path = 'C:/Users/Anudeep/Desktop/kk/dataset/'

files = os.listdir(path)

for f in files:

df = pd.read\_csv(path+f, header=None, parse\_dates=[1])

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(f)

print(df.shape)

print(df[0].value\_counts())

print(df[1].max())

print(df[1].min())

print(df[1].max() - df[1].min())

# selecting train files from 2016-01-01 to 2016-01-15

train\_files = ['111', '211','311']

### Creating Train Dataset

# creating a function for resampling and saving train datasets

def read\_df(string):

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/'+string+'.csv', header=None, parse\_dates=[1])

ID = 'id' + string[0]

time = 'time' + string [0]

cons = 'water\_consumption' + string [0]

df.columns = [ID, time, cons, 'unknown']

df.drop(columns = 'unknown', axis = 1, inplace = True)

df.set\_index(time, inplace = True)

df = df.resample('D').mean()

print("Null value is observed in {}".format(df[df.isnull().any(axis =1)].index))

# using bfill for replacing nan values where data is not foudn

df = df.fillna(method = 'bfill')

df[ID] = df[ID].astype(int)

print(df.dtypes)

return df

df1 = read\_df('111')

print(df1.shape)

df2 = read\_df('211')

print(df2.shape)

df3 = read\_df('311')

print(df3.shape)

#Concatenating the all the Training data files.

train\_df = pd.concat([df1, df2, df3], axis = 1)

train\_df

train\_df['cum\_cons'] = train\_df['water\_consumption1'] +train\_df['water\_consumption2']+train\_df['water\_consumption3']

train\_df\_cum = train\_df.loc[:, ['cum\_cons']]

print(train\_df\_cum.shape)

train\_df\_cum.head()

train\_df\_cum.info()

train = train\_df[['cum\_cons']].copy()

type(train)

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(train)

plt.show()

train.info()

# Creating Test Dataset

df4 = read\_df('112')

print(df4.shape)

df5 = read\_df('212')

print(df5.shape)

df6 = read\_df('312')

print(df6.shape)

#Concatenating the all the test data files.

test\_df = pd.concat([df4, df5, df6], axis = 1)

test\_df

# Adding up all the test water consumption data together

test\_df['cum\_cons'] = test\_df['water\_consumption1'] +test\_df['water\_consumption2']+test\_df['water\_consumption3']

test\_df\_cum = test\_df.loc[:, ['cum\_cons']]

print(test\_df\_cum.shape)

test\_df\_cum

test = test\_df[['cum\_cons']].copy()

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(test)

plt.show()

import sklearn

from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler(feature\_range=(0, 1))

scale.fit(train)

train = scale.transform(train)

test = scale.transform(test)

import numpy as np

def datasetCreation(data, lback=1):

X, Y = list(), list()

for i in range(len(data)-lback-1):

a = data[i:(i+lback), 0]

X.append(a)

Y.append(data[i + lback, 0])

return np.array(X), np.array(Y)

trainX, trainY = datasetCreation(train, 1)

testX, testY = datasetCreation(test, 1)

# trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

# testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

# X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

import numpy as np

np.random.seed(10)

# Random Forest Regressor

from sklearn.ensemble import RandomForestRegressor

rf\_regr = RandomForestRegressor(max\_depth=2, random\_state=0)

rf\_regr.fit(trainX, trainY)

tr\_pred = rf\_regr.predict(trainX)

# Evaluation for RandomForest

import math

from sklearn.metrics import mean\_squared\_error

tr\_rmse = math.sqrt(mean\_squared\_error(trainY, tr\_pred))

tr\_rmse

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.title('Random Forest Regressor Training')

plt.legend()

plt.show()

te\_pred

# Evaluation for RandomForest Testing

te\_pred = rf\_regr.predict(testX)

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

te\_rmse

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.title('Random Forest Regressor Testing')

plt.legend()

plt.show()

# Support Vector Regressor

from sklearn.svm import SVR

svr\_regr = SVR(C=1.0, epsilon=0.2)

svr\_regr.fit(trainX, trainY)

tr\_pred = svr\_regr.predict(trainX)

print("RMSE of train data: ", math.sqrt(mean\_squared\_error(trainY, tr\_pred)))

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.title('Support Vector Regressor Training')

plt.legend()

plt.show()

# Evaluation for SVR Testing

te\_pred = svr\_regr.predict(testX)

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

print("RMSE of test",te\_rmse)

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.title('Support Vector Regressor Testing')

plt.legend()

plt.show()

# Extra Trees Regressor

from sklearn.ensemble import ExtraTreesRegressor

et\_regr = ExtraTreesRegressor(n\_estimators=100, random\_state=0)

et\_regr.fit(trainX, trainY)

tr\_pred = et\_regr.predict(trainX)

print("RMSE of train data: ", math.sqrt(mean\_squared\_error(trainY, tr\_pred)))

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.title('Extra Trees Regressor Training')

plt.legend()

plt.show()

te\_pred = et\_regr.predict(testX)

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

print("RMSE of test",te\_rmse)

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.title('Extra Trees Regressor Testing')

plt.legend()

plt.show()

# AdaBoost Regressor

from sklearn.ensemble import AdaBoostRegressor

ada\_regr = AdaBoostRegressor(random\_state=0, n\_estimators=100)

ada\_regr.fit(trainX, trainY)

tr\_pred = ada\_regr.predict(trainX)

print("RMSE of train data: ", math.sqrt(mean\_squared\_error(trainY, tr\_pred)))

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.title('AdaBoost Regressor Training')

plt.legend()

plt.show()

te\_pred = ada\_regr.predict(testX)

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

print("RMSE of test",te\_rmse)

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.title('AdaBoost Regressor Testing')

plt.legend()

plt.show()

# AdaBoost Regressor Trail two

from sklearn.ensemble import AdaBoostRegressor

ada\_regr = AdaBoostRegressor(base\_estimator= rf\_regr, random\_state=0, n\_estimators=100)

ada\_regr.fit(trainX, trainY)

tr\_pred = ada\_regr.predict(trainX)

print("RMSE of train data: ", math.sqrt(mean\_squared\_error(trainY, tr\_pred)))

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.title('AdaBoost Regressor Training')

plt.legend()

plt.show()

te\_pred = ada\_regr.predict(testX)

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

print("RMSE of test",te\_rmse)

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.title('AdaBoost Regressor Testing')

plt.legend()

plt.show()

**\*\* oneinput-day-final-main.ipynb \*\***

import pandas as pd

import numpy as np

import os

\*\*Please place all the data files in a folder named 'data' at a location where this notebook file is place\*\*

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/111.csv', header=None, parse\_dates = [1])

df.head()

path = 'C:/Users/Anudeep/Desktop/kk/dataset/'

files = os.listdir(path)

for f in files:

df = pd.read\_csv(path+f, header=None, parse\_dates=[1])

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(f)

print(df.shape)

print(df[0].value\_counts())

print(df[1].max())

print(df[1].min())

print(df[1].max() - df[1].min())

# selecting train files from 2016-01-01 to 2016-01-15

train\_files = ['111', '211','311']

### Creating Train Dataset

# creating a function for resampling and saving train datasets

def read\_df(string):

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/'+string+'.csv', header=None, parse\_dates=[1])

ID = 'id' + string[0]

time = 'time' + string [0]

cons = 'water\_consumption' + string [0]

df.columns = [ID, time, cons, 'unknown']

df.drop(columns = 'unknown', axis = 1, inplace = True)

df.set\_index(time, inplace = True)

df = df.resample('D').mean()

print("Null value is observed in {}".format(df[df.isnull().any(axis =1)].index))

# using bfill for replacing nan values where data is not found

df = df.fillna(method = 'bfill')

df[ID] = df[ID].astype(int)

print(df.dtypes)

return df

df1 = read\_df('111')

print(df1.shape)

df2 = read\_df('211')

print(df2.shape)

df3 = read\_df('311')

print(df3.shape)

# df1\_final = pd.concat([df1, df4])

# df2\_final = pd.concat([df2, df5])

# df3\_final = pd.concat([df3, df6])

#Concatenating all the Training data files.

train\_df = pd.concat([df1, df2, df3], axis = 1)

train\_df

train\_df['cum\_cons'] = train\_df['water\_consumption1'] +train\_df['water\_consumption2']+train\_df['water\_consumption3']

train\_df\_cum = train\_df.loc[:, ['cum\_cons']]

print(train\_df\_cum.shape)

train\_df\_cum.head()

train\_df\_cum.info()

train = train\_df[['cum\_cons']].copy()

type(train)

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(train)

plt.show()

train.info()

# Creating Test Dataset

df4 = read\_df('112')

print(df4.shape)

df5 = read\_df('212')

print(df5.shape)

df6 = read\_df('312')

print(df6.shape)

#Concatenating all the test data files.

test\_df = pd.concat([df4, df5, df6], axis = 1)

test\_df

# Adding up all the test water consumption data together

test\_df['cum\_cons'] = test\_df['water\_consumption1'] +test\_df['water\_consumption2']+test\_df['water\_consumption3']

test\_df\_cum = test\_df.loc[:, ['cum\_cons']]

print(test\_df\_cum.shape)

test\_df\_cum

test = test\_df[['cum\_cons']].copy()

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(test)

plt.show()

import sklearn

from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler(feature\_range=(0, 1))

scale.fit(train)

train = scale.transform(train)

test = scale.transform(test)

import numpy as np

def datasetCreation(data, lback=1):

X, Y = list(), list()

for i in range(len(data)-lback-1):

a = data[i:(i+lback), 0]

X.append(a)

Y.append(data[i + lback, 0])

return np.array(X), np.array(Y)

trainX, trainY = datasetCreation(train, 1)

testX, testY = datasetCreation(test, 1)

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

# X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

import numpy as np

np.random.seed(10)

# LSTM

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM

model = Sequential()

model.add(LSTM(4, input\_shape=(1, 1)))

model.add(Dense(4))

model.compile(loss='mean\_squared\_error', optimizer='adam')

model.add(Dense(1))

history=model.fit(trainX, trainY, epochs=100, batch\_size=1, verbose=1)

model.summary()

loss\_per\_epoch = history.history['loss']

import matplotlib.pyplot as plt

plt.plot(range(len(loss\_per\_epoch)),loss\_per\_epoch)

plt.title('Model loss of LSTM')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train'], loc='upper left')

tr\_pred = model.predict(trainX)

tr\_pred = scale.inverse\_transform(tr\_pred)

trainY = scale.inverse\_transform([trainY])

trainY = trainY.T

import math

from sklearn.metrics import mean\_squared\_error

tr\_rmse = math.sqrt(mean\_squared\_error(trainY, tr\_pred))

tr\_rmse

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.legend()

plt.show()

te\_pred = model.predict(testX)

te\_pred = scale.inverse\_transform(te\_pred)

testY = scale.inverse\_transform([testY])

testY = testY.T

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

te\_rmse

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.legend()

plt.show()

**\*\* oneinput-lstm2-day-final-main.ipynb \*\***

import pandas as pd

import numpy as np

import os

\*\*Please place all the data files in a folder named 'data' at a location where this notebook file is place\*\*

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/111.csv', header=None, parse\_dates = [1])

df.head()

path = 'C:/Users/Anudeep/Desktop/kk/dataset/'

files = os.listdir(path)

for f in files:

df = pd.read\_csv(path+f, header=None, parse\_dates=[1])

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(f)

print(df.shape)

print(df[0].value\_counts())

print(df[1].max())

print(df[1].min())

print(df[1].max() - df[1].min())

# selecting train files from 2016-01-01 to 2016-01-15

train\_files = ['111', '211','311']

### Creating Train Dataset

# creating a function for resampling and saving train datasets

def read\_df(string):

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/'+string+'.csv', header=None, parse\_dates=[1])

ID = 'id' + string[0]

time = 'time' + string [0]

cons = 'water\_consumption' + string [0]

df.columns = [ID, time, cons, 'unknown']

df.drop(columns = 'unknown', axis = 1, inplace = True)

df.set\_index(time, inplace = True)

df = df.resample('D').mean()

print("Null value is observed in {}".format(df[df.isnull().any(axis =1)].index))

# using bfill for replacing nan values where data is not found

df = df.fillna(method = 'bfill')

df[ID] = df[ID].astype(int)

print(df.dtypes)

return df

df1 = read\_df('111')

print(df1.shape)

df2 = read\_df('211')

print(df2.shape)

df3 = read\_df('311')

print(df3.shape)

# df1\_final = pd.concat([df1, df4])

# df2\_final = pd.concat([df2, df5])

# df3\_final = pd.concat([df3, df6])

#Concatenating all the Training data files.

train\_df = pd.concat([df1, df2, df3], axis = 1)

train\_df

train\_df['cum\_cons'] = train\_df['water\_consumption1'] +train\_df['water\_consumption2']+train\_df['water\_consumption3']

train\_df\_cum = train\_df.loc[:, ['cum\_cons']]

print(train\_df\_cum.shape)

train\_df\_cum.head()

train\_df\_cum.info()

train = train\_df[['cum\_cons']].copy()

type(train)

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(train)

plt.show()

train.info()

# Creating Test Dataset

df4 = read\_df('112')

print(df4.shape)

df5 = read\_df('212')

print(df5.shape)

df6 = read\_df('312')

print(df6.shape)

#Concatenating all the test data files.

test\_df = pd.concat([df4, df5, df6], axis = 1)

test\_df

# Adding up all the test water consumption data together

test\_df['cum\_cons'] = test\_df['water\_consumption1'] +test\_df['water\_consumption2']+test\_df['water\_consumption3']

test\_df\_cum = test\_df.loc[:, ['cum\_cons']]

print(test\_df\_cum.shape)

test\_df\_cum

test = test\_df[['cum\_cons']].copy()

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(test)

plt.show()

import sklearn

from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler(feature\_range=(0, 1))

scale.fit(train)

train = scale.transform(train)

test = scale.transform(test)

import numpy as np

def datasetCreation(data, lback=1):

X, Y = list(), list()

for i in range(len(data)-lback-1):

a = data[i:(i+lback), 0]

X.append(a)

Y.append(data[i + lback, 0])

return np.array(X), np.array(Y)

trainX, trainY = datasetCreation(train, 1)

testX, testY = datasetCreation(test, 1)

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

# X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

import numpy as np

np.random.seed(10)

# LSTM2

from keras.models import Sequential

from keras.layers import Dense, LSTM

model = Sequential()

model.add(LSTM(50, activation='relu', return\_sequences=True, input\_shape=(1, 1)))

model.add(LSTM(50, activation='relu'))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mean\_squared\_error')

history = model.fit(trainX, trainY, epochs=100, batch\_size=10, verbose=2)

model.summary()

loss\_per\_epoch = history.history['loss']

import matplotlib.pyplot as plt

plt.plot(range(len(loss\_per\_epoch)),loss\_per\_epoch)

plt.title('Model loss of LSTM')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train'], loc='upper left')

tr\_pred = model.predict(trainX)

tr\_pred = scale.inverse\_transform(tr\_pred)

trainY = scale.inverse\_transform([trainY])

trainY = trainY.T

import math

from sklearn.metrics import mean\_squared\_error

tr\_rmse = math.sqrt(mean\_squared\_error(trainY, tr\_pred))

tr\_rmse

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.legend()

plt.show()

te\_pred = model.predict(testX)

te\_pred = scale.inverse\_transform(te\_pred)

testY = scale.inverse\_transform([testY])

testY = testY.T

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

te\_rmse

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.legend()

plt.show()

**\*\* model-ann-singleinput-day-main.ipynb \*\***

import pandas as pd

import numpy as np

import os

\*\*Please place all the data files in a folder named 'data' at a location where this notebook file is place\*\*

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/111.csv', header=None, parse\_dates = [1])

df.head()

path = 'C:/Users/Anudeep/Desktop/kk/dataset/'

files = os.listdir(path)

for f in files:

df = pd.read\_csv(path+f, header=None, parse\_dates=[1])

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print(f)

print(df.shape)

print(df[0].value\_counts())

print(df[1].max())

print(df[1].min())

print(df[1].max() - df[1].min())

# selecting train files from 2016-01-01 to 2016-01-15

train\_files = ['111', '211','311']

### Creating Train Dataset

# creating a function for resampling and saving train datasets

def read\_df(string):

df = pd.read\_csv('C:/Users/Anudeep/Desktop/kk/dataset/'+string+'.csv', header=None, parse\_dates=[1])

ID = 'id' + string[0]

time = 'time' + string [0]

cons = 'water\_consumption' + string [0]

df.columns = [ID, time, cons, 'unknown']

df.drop(columns = 'unknown', axis = 1, inplace = True)

df.set\_index(time, inplace = True)

df = df.resample('D').mean()

print("Null value is observed in {}".format(df[df.isnull().any(axis =1)].index))

# using bfill for replacing nan values where data is not found

df = df.fillna(method = 'bfill')

df[ID] = df[ID].astype(int)

print(df.dtypes)

return df

df1 = read\_df('111')

print(df1.shape)

df2 = read\_df('211')

print(df2.shape)

df3 = read\_df('311')

print(df3.shape)

# df1\_final = pd.concat([df1, df4])

# df2\_final = pd.concat([df2, df5])

# df3\_final = pd.concat([df3, df6])

#Concatenating all the Training data files.

train\_df = pd.concat([df1, df2, df3], axis = 1)

train\_df

train\_df['cum\_cons'] = train\_df['water\_consumption1'] +train\_df['water\_consumption2']+train\_df['water\_consumption3']

train\_df\_cum = train\_df.loc[:, ['cum\_cons']]

print(train\_df\_cum.shape)

train\_df\_cum.head()

train\_df\_cum.info()

train = train\_df[['cum\_cons']].copy()

type(train)

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(train)

plt.show()

train.info()

# Creating Test Dataset

df4 = read\_df('112')

print(df4.shape)

df5 = read\_df('212')

print(df5.shape)

df6 = read\_df('312')

print(df6.shape)

#Concatenating the all the test data files.

test\_df = pd.concat([df4, df5, df6], axis = 1)

test\_df

# Adding up all the test water consumption data together

test\_df['cum\_cons'] = test\_df['water\_consumption1'] +test\_df['water\_consumption2']+test\_df['water\_consumption3']

test\_df\_cum = test\_df.loc[:, ['cum\_cons']]

print(test\_df\_cum.shape)

test\_df\_cum

test = test\_df[['cum\_cons']].copy()

import matplotlib.pyplot as plt

plt.figure(figsize=(14,8))

plt.plot(test)

plt.show()

import sklearn

from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler(feature\_range=(0, 1))

scale.fit(train)

train = scale.transform(train)

test = scale.transform(test)

import numpy as np

def datasetCreation(data, lback=1):

X, Y = list(), list()

for i in range(len(data)-lback-1):

a = data[i:(i+lback), 0]

X.append(a)

Y.append(data[i + lback, 0])

return np.array(X), np.array(Y)

trainX, trainY = datasetCreation(train, 1)

testX, testY = datasetCreation(test, 1)

trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))

testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# X\_train = np.reshape(X\_train, (X\_train.shape[0], 1, X\_train.shape[1]))

# X\_test = np.reshape(X\_test, (X\_test.shape[0], 1, X\_test.shape[1]))

import numpy as np

np.random.seed(10)

# ANN

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM

model = Sequential()

model.add(Dense(12, input\_dim=1, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(4, activation='relu'))

model.add(Dense(2, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean\_squared\_error', optimizer='adam')

history = model.fit(trainX, trainY, epochs=300)

model.summary()

loss\_per\_epoch = history.history['loss']

import matplotlib.pyplot as plt

plt.plot(range(len(loss\_per\_epoch)),loss\_per\_epoch)

plt.title('Model loss of ANN')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train'], loc='upper left')

tr\_pred = model.predict(trainX)

tr\_pred = scale.inverse\_transform(tr\_pred)

trainY = scale.inverse\_transform([trainY])

trainY = trainY.T

import math

from sklearn.metrics import mean\_squared\_error

tr\_rmse = math.sqrt(mean\_squared\_error(trainY, tr\_pred))

tr\_rmse

plt.plot(trainY, label='Expected')

plt.plot(tr\_pred, label='Predicted')

plt.legend()

plt.show()

te\_pred = model.predict(testX)

te\_pred = scale.inverse\_transform(te\_pred)

testY = scale.inverse\_transform([testY])

testY = testY.T

te\_rmse = math.sqrt(mean\_squared\_error(testY, te\_pred))

te\_rmse

plt.plot(testY, label='Expected')

plt.plot(te\_pred, label='Predicted')

plt.legend()

plt.show()