CLASSIFICATION OF QUALITY OF DRINKING WATER USING MACHINE LEARNING TECHNIQUE

Submitted in partial fulfilment of the requirements

for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

Bishwadeep ghosh

(38110085)

Bharat Kaushik

(38110078)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH - 2022

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **BISHWADEEP GHOSH. (38110085)**

BHARAT KAUSHIK. (38110078) who carried out the project entitled **"CLASSIFICATION OF QUALITY OF DRINKING WATER USING MACHINE LEARNING TECHNIQUE"** under my supervision from

November 2021 to March 2022.

INTERNAL GUIDE

Dr. J. Refonaa

Head of the Department

Dr.A.Lakshmanan,ME,Ph.D..,

Submitted for Viva voce Examination held on___

Internal Examiner External Examiner

DECLARATION

We BISHWADEEP GHOSH and BHARAT KAUSHIK hereby declare that the Project Report entitled

CLASSIFICATION OF QUALITY OF DRINKING WATER USING MACHINE LEARNING TECHNIQUE done under the guidance of Dr. J. Refonaa(Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer science and engineering.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATES:**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D**, **Dean**, School of Computing

Dr. L. Lakshmanan M.E., Ph.D. , and **Dr.S.Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering for providing necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr./Mr./Ms for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
01	**ABSTRACT**	
02	**EXISTING SYSTEM** 2.1 DRAWBACKS	
03	**INTRODUCTION** 3.1 DATA SCIENCE 3.2 ARTIFICIAL INTELLIGENCE	
04	**MACHINE LEARNING**	
05	**PREPARING DATASET**	
06	**PROPOSED SYSTEM** 6.1 ADVANTAGES	
07	**LITERATURE SURVEY**	
08	**SYSTEM STUDY** 8.1 OBJECTIVES 8.2 PROJECT GOAL 8.3 SCOPE OF THE PROJECT	
09	**FEASIBILITY STUDY**	
10	**LIST OF MODULES**	
11	**PROJECT REQUIREMENTS** 11.1 FUNCTIONAL REQUIREMENTS 11.2 NON-FUNCTIONAL REQUIREMENTS	
12	**ENVIRONMENT REQUIREMENT**	
13	**SOFTWARE DESCRIPTION** 13.1 ANACONDA NAVIGATOR	

	13.2 JUPYTER NOTEBOOK	
14	**PYTHON**	
15	**SYSTEM ARCHITECTURE**	
16	**WORKFLOW DIAGRAM**	
17	**USECASE DIAGRAM**	
18	**CLASS DIAGRAM**	
19	**ACTIVITY DIAGRAM**	
20	**SEQUENCE DIAGRAM**	
21	**ER – DIAGRAM**	
22	**MODULE DESCRIPTION** 22.1 MODULE DIAGRAM 22.2 MODULE GIVEN INPUT EXPECTED OUTPUT	
23	**DEPLOYMENT**	
24	**HTML**	
25	**CSS**	
26	**CODING**	
27	**CONCLUSION**	
28	**FUTURE WORK**	

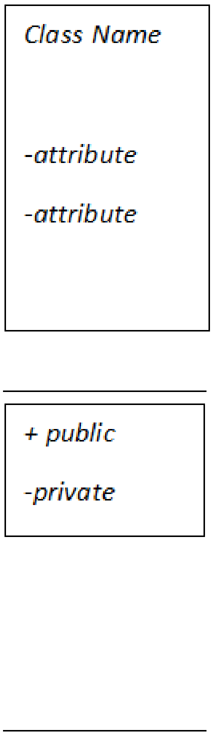
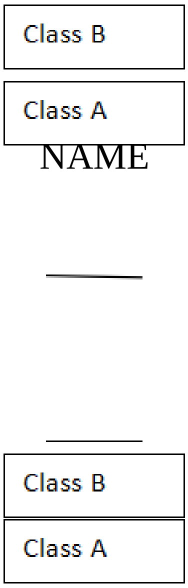
LIST OF FIGURES

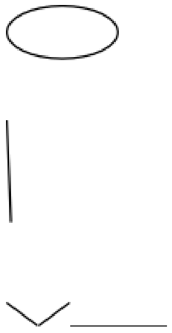
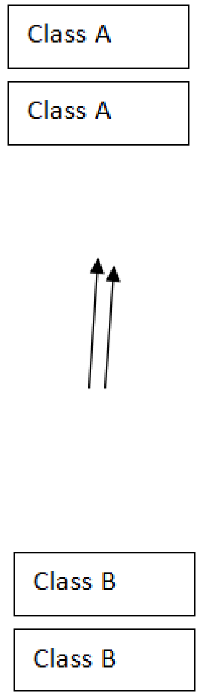
SL.NO	TITLE	PAGE.NO

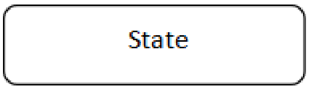
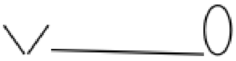
01	SYSTEM ARCHITECTURE	
02	WORKFLOW DIAGRAM	
03	USECASE DIAGRAM	
04	CLASS DIAGRAM	
05	ACTIVITY DIAGRAM	
06	SEQUENCE DIAGRAM	
07	ER – DIAGRAM	
08	MODULE DIAGRAM	

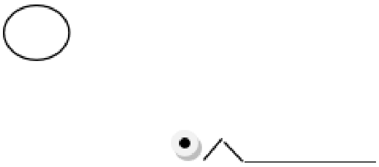
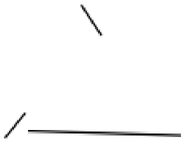
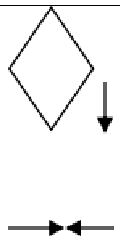
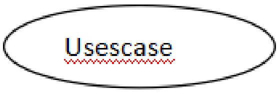
LIST OF SYSMBOLS

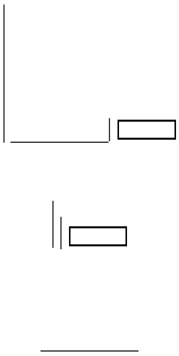
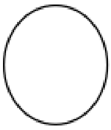
S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of

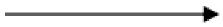
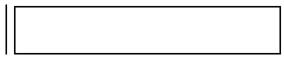
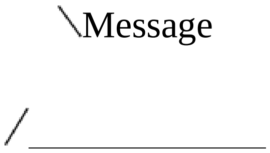
		 Class Name -attribute -attribute + public -private	similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles representsthe way the two classes see each

			other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment

5.	*Relation* (uses)	*uses*	Used for additional process communication.
6.	Relation (extends)	———→EXTENDS	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication	————	Communication between various use cases.
8.	State		State of the processs.
9.	Initial State		Initial state of the object

10.	Final state		F inal state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Usecase		Interact ion between the system and external environment.

14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or acion.
			Represents external

17.	External entity		entities such as keyboard,sensors,etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message		Represents the message exchanged.

1. Abstract:

Generally, Water pollution refers to the release of pollutants into the water that are detrimental to human health and the planet as a whole. It can be described as one of the most dangerous threats that the humanity ever faced. It causes damage to animals, crops, forests etc. To prevent this problem in transport sectors have to predict water quality from pollutants using machine learning techniques. Hence, water quality evaluation and prediction has become an important research area. The aim is to investigate machine learning based techniques for water quality forecasting by prediction results in best accuracy.

The analysis of dataset by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyse the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. Our analysis provides a comprehensive guide to sensitivity analysis of model parameters with regard to performance in prediction of water quality pollution by accuracy calculation. To propose a machine learning-based method to accurately predict the Water Quality Index value by prediction results in the form of best accuracy from comparing supervised classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given transport traffic department dataset with evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

2. EXISTING SYSTEM

The sensor technology for water quality monitoring (WQM) has improved during recent years. The cost-effective sensorised tools that can autonomously measure the essential physical -chemical and biological (PCB) variables are now readily available and are being deployed on buoys, boats and ships. Yet, there is a disconnect between the data quality, data gathering and data analysis due to the lack of standardized approaches for data collection and processing, spatio-temporal variation of key parameters in water bodies and new contaminants. Such gaps can be bridged with a network of multiparametric sensor systems deployed in water bodies using autonomous vehicles such as marine robots and aerial vehicles to broaden the data coverage in space and time. Further, intelligent algorithms could be employed for standardised data analysis and forecasting. The connected sensor technologies for water quality monitoring (WQM) could provide the bridging solution for current disconnect between data quality, data gathering and data analysis and enhance the global data intercomparability. With this in view, this article has reviewed key sensing technologies, sensor deployment strategies and the emerging methods for data analysis.

2.1 Drawbacks

- It is not using machine learning and deep learning algorithms
- It can't thereby better determine the regularity of water pollutant data and achieve more accurate prediction results.

3. INTRODUCTION

Domain overview

3.1 Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist: Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

3.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text

4. MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.

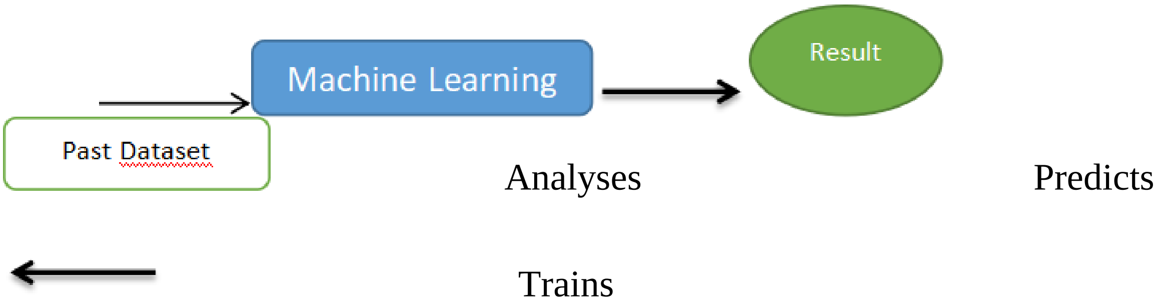


Fig: Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is y = f(X). The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

5. Preparing the Dataset:

- aluminium - dangerous if greater than 2.8

- ammonia - dangerous if greater than 32.5
- arsenic - dangerous if greater than 0.01
- barium - dangerous if greater than 2
- cadmium - dangerous if greater than 0.005
- chloramine - dangerous if greater than 4
- chromium - dangerous if greater than 0.1
- copper - dangerous if greater than 1.3
- fluoride - dangerous if greater than 1.5
- bacteria - dangerous if greater than 0
- viruses - dangerous if greater than 0
- lead - dangerous if greater than 0.015
- nitrates - dangerous if greater than 10
- nitrites - dangerous if greater than 1
- mercury - dangerous if greater than 0.002
- perchlorate - dangerous if greater than 56
- radium - dangerous if greater than 5
- selenium - dangerous if greater than 0.5
- silver - dangerous if greater than 0.1
- uranium - dangerous if greater than 0.3
- is safe - class attribute {0 - not safe, 1 – safe

Access to safe drinking-water is essential to health, a basic human right and a component of effective policy for health protection. This is important as a health and development issue at a national, regional and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.

6. Proposed System:

The proposed method is to build a machine learning model for Water quality. The process carries from data collection where the past data related to Water qualities are collected. Data mining is a commonly used technique for processing enormous data in the domain. The water if found before proper treatment can save lives. Machine learning is now applied and mostly used in health care where it reduces the manual effort and better model makes error less which leads in saving the life. The data analysis is done on the dataset proper variable identification done that is both the dependent variables and independent variables are found. Then proper machine learning algorithm are applied on the dataset where the pattern of data is learnt. After applying different algorithms, a better algorithm is used for the prediction of outcome.

6.1 Advantages:

- These reports are to the investigation of applicability of machine learning techniques for water quality forecasting in operational conditions.
- Finally, it highlights some observations on future research issues, challenges, and needs.

7. LITERATURE SURVEY

General

A literature review is a body of text that aims to review the critical points of current knowledge on and or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them. Loan default trends have been long studied from a socio-economic stand point. Most economics surveys believe in empirical modeling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which it is observing now. Some of the survey's to understand the past and present perspective of loan approval or not.

Review of Literature Survey

Title : Predictive Analysis of Water Quality Parameters using Deep Learning

Author: Archana Solanki, Himanshu Agrawal, Kanchan Khare

Year : September 2015

Lakes and reservoirs are important water resources. Reservoirs are vital water resources to support all living organism. They provide clean water and habitat for a complex variety of aquatic life. Water from such resources can be used for diverse purposes such as, industry usage, agriculture and supplies for drinking water and recreation and aesthetic value. Apart from this, reservoirs also helpful to get hydro-electric power, flood control and scenic beauty. Water collected in such resources can be utilized in drought situation also. Unfortunately, these important resources are being polluted and the quality of water is being influenced by numerous factors. The quality of water is deteriorated by anthropogenic activities, indiscriminate disposal of sewage, human activities and also industry waste. Water quality monitoring of reservoirs is essential in exploitation of aquatic resources conservation. The quality of water helps in regulating the biotic diversity and biomass, energy and rate of succession. In order to reduce effect of contaminated water, it is essential to assess different aspects of water quality. Predicting water quality parameters a few steps ahead can be beneficial to achieve this. The main objective of this study is to provide fairly accurate predictions for variable data. The study shows that deep learning techniques which use unsupervised learning to provide accurate results as compared to the techniques based on supervised learning. The comparison of results show that robustness can be achieve by denoising autoencoder and deep belief network and also successfully handle the variability in the data. Merit of the unsupervised learning algorithms are evaluated on the basis of metrics such as mean absolute error and mean square error to examine the error rate of prediction.

Title: Water Quality Monitoring for Disease Prediction using Machine Learning

Author: Prajakta Patil , Sukanya More , Atharv Deshpande.

Year: 2020

Access to pure drinking water and sanitation has been marked as a fundamental human right as „The Human Right to Water and Sanitation" by the United Nations General Assembly on 28 July 2010. Water related diseases are the primary cause of diseases and deaths around the world with more than 3.4 million deaths per year. Lack of monitoring of water sources and inability to anticipate the proliferation of waterborne diseases are found at the root of these deaths. There has been a compelling need for disease prediction based on water quality. The present study was focused on monitoring of water quality parameters and using these parameters to predict probable waterborne diseases. The main objective of study was to apply machine learning techniques to water quality data in order to make predictions about waterborne diseases. The work involved collecting observations of some of the water quality parameters by leveraging the Internet of Things (IOT). The detailed data, involving observations of all the necessary parameters, was collected from the West Bengal Pollution Control Board's Water Quality Information System. Gradient Boosting Classifier was trained and tested on collected data. The accuracy of result was found to be 0.92 and 0.95 on cross-validation and hold-out data, respectively. Once trained, the model started making predictions based on primary data. The predicted diseases were conveyed in the form of alerts using Push bullet service. The study thus proposed usability of water quality parameters in early prediction of water related diseases.

Title : Implementation of Machine Learning Methods for Monitoring and Predicting Water Quality Parameters

Author: Gasim Hayder , , Isman Kurniawan , Hauwa Mohammed Mustafa.

Year : 11.09.2020

The importance of good water quality for human use and consumption can never be underestimated, and its quality is determined through effective monitoring of the water quality index. Different approaches have been employed in the treatment and monitoring of water quality parameters (WQP). Presently, water quality is carried out through laboratory experiments, which requires costly reagents, skilled labor, and consumes time. Thereby making it necessary to search for an alternative method. Recently, machine learning tools have been successfully implemented in the monitoring, estimation, and predictions of river water quality index to provide an alternative solution to the limitations of laboratory analytical methods. In this study, the potentials of one of the machine learning tools (artificial neural network) were explored in the predictions and estimation of the Kelantan River basin. Water quality data collected from the 14 stations of the River basin was used for modeling and predicting (WQP). As for WQP analysis, the results obtained from this study show that the best prediction was obtained from the prediction of pH. The low kurtosis values of pH indicate that the appearance of outliers give a negative impact on the performance. As for WQP analysis for each station, we found that the WQP prediction in station 1, 2, and 3 give the good results. This is related to the available data of those stations that are more than the available data in other stations, except station 8.

Title : PREDICTION OF WATER QUALITY PARAMETER OF AMBIKA RIVER BY ARTIFICIAL INTELLIGENCE BASED MODELS

Author: Radhika ,K. Hirapra ,Dr. S.S. Singh , Tanmay Rathod

Year : R April 2018

Several techniques such as; Fuzzy Inference System (FIS) and Neural Network (NN) are used for developing of the predictive models to estimate parameters of water quality. The main objective of this study is to compare between the predictive ability of the Artificial Neural Network (ANN) model and Adaptive Neuro-Fuzzy Inference System (ANFIS) model to estimate the Dissolved Oxygen(DO) using data from the sampling sites station at Billimora-amalsad Road on Ambika River in Gujarat, India. The data is obtained from the Gujarat pollution control board(GPCB), during 2010-2017. Total Ten parameters of water quality namely Dissolved Oxygen, Biochemical Oxygen Demand, Chemical Oxygen Demand ,ph, Nitrite, Nitrate, Phosphate, Total Dissolved Solids, Calcium, Magnesium are used to developed the models. The experimental results indicate that the ANFIS model provides a higher correlation coefficient (R2 =0.885) in Training and (R2 =0.818) in Testing with a lower root mean square error (RMSE=0.2307) in training and (RMSE= 0.3574) in Training than the corresponding (RMSE= 0.3863) ANN mode

Title : Water Quality Assessment with Water Quality Indices

Author: Sivaranjani S.1, Amitava Rakshit2 and Samrath Singh

Year : 20,july, 2015

A water quality index provides a single number that expresses overall water quality at a certain location and time based on several water quality parameters. Water quality index (WQI) is valuable and unique rating to depict the overall water quality status in a single term that is helpful for the selection of appropriate treatment technique to meet the concerned issues. These indices utilize various physio-chemical and biological parameters and have been resulted as an outcome of efforts and research and development carried out by different government agencies and experts in this area globally. This review paper includes the water quality assessment with water quality indices being used globally

8. SYSTEM STUDY

8.1 Overview of the system

Water pollution is one of the biggest fears for the green globalization. In order to ensure the safe supply of the drinking water the quality needs to be monitor in real time

In the 21st century, there were lots of inventions, but at the same time were pollutions, global warming and so on are being formed, because of this there is no safe drinking water for the world's pollution. Nowadays, water quality monitoring in real time faces challenges because of global warming limited water resources, growing population, etc. Hence there is need of developing better methodologies to predicting the water quality parameters in real time

- Define a problem
- Preparing data
- Evaluating algorithms
- Improving results
- Predicting results

8.2 Project Goals

- **Exploration data analysis of variable identification**

  - Loading the given dataset
  - Import required libraries packages
  - Analyze the general properties
  - Find duplicate and missing values
  - Checking unique and count values

- **Uni-variate data analysis**

  - Rename, add data and drop the data
  - To specify data type

- **Exploration data analysis of bi-variate and multi-variate**

  - Plot diagram of pairplot, heatmap, bar chart and Histogram

- **Method of Outlier detection with feature engineering**

  - Pre-processing the given dataset
  - Splitting the test and training dataset
  - Comparing the Decision tree and Logistic regression model and random forest etc

- **Comparing algorithm to predict the result**

  - Based on the best accuracy

8.3 Objectives

The goal is to develop a machine learning model for Predicting water Quality using, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

8.4 Problem Description/ Problem Statements

Water pollution is the contamination of water bodies that occur when pollutant are indirectly or directly discharge into water bodies without adequate treatment to remove the harmful sediment It will give an affect to ecosystem and human life and become an issue nowadays.

8.5 Scope of the Project

The scope of this project is to investigate a Water Quality dataset to find the quality using machine learning technique. To find the water quality is based on the actual water conditions its not easy one to predict, but using the AI machine learning technique compare with different algorithms we can build a predictive models to predict the water quality index. Using metrics (accuracy, precision, recall, etc...) we can validate the model accuracy.

9. Feasibility study:

Exploratory Data Analysis of Water Quality Prediction

Multiple datasets from different sources would be combined to form a generalized dataset, and then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms, K-Nearest Neighbor (KNN) and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Building the classification model

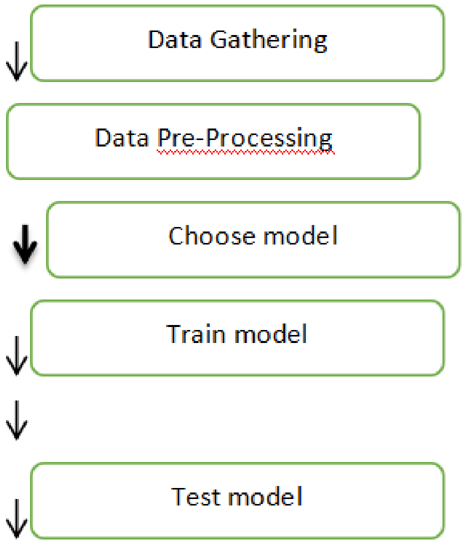
The predicting the water quality problem, decision tree algorithm prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.

- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data’s. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can’t be used directly. It’s used to preprocess then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.



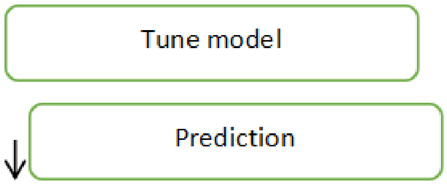


Fig: Process of dataflow diagram

10. List of Modules:

- Data Preprocessing Technique
- Data analysis of visualization
- Comparing Algorithm with prediction in the form of best accuracy result
- Deployment using Flask
-

11.Project Requirements

General:

Requirements are the basic constrains that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements

2. Non-Functional requirements

3. Environment requirements

A. Hardware requirements

B. software requirements

10. Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

11.1 Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

11.2 Environmental Requirements:

1. Software Requirements:

Operating System	: Windows
Tool	: Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor	: Pentium IV/III
Hard disk	: minimum 80 GB
RAM	: minimum 2 GB

13. SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud,[10] PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

13.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

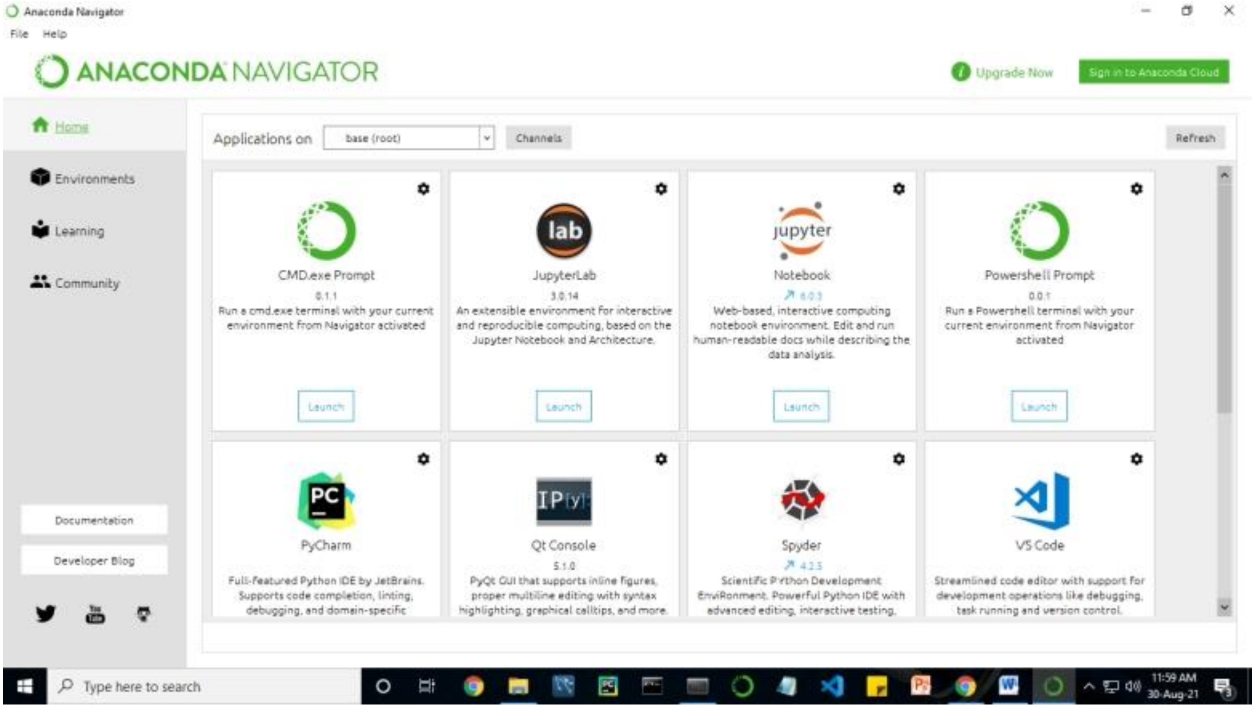
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
↗ 6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

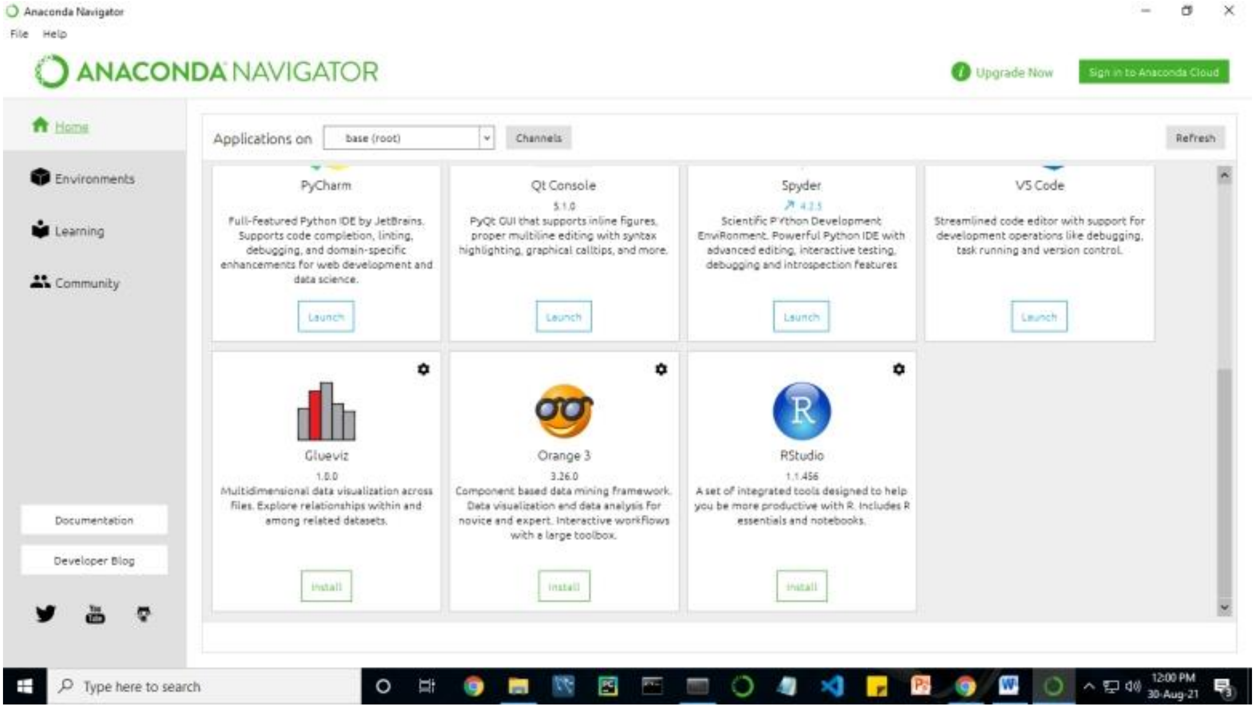
Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

PyCharm
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific

Qt Console
5.1.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
↗ 4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.



PyCharm
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

Qt Console
5.1.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
↗ 4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with conda list on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

Conda :

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. these softwares have great graphical user interface and these will make your work easy to do. you can also use it to run your python script. These are the software carried by anaconda navigator.

13.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda

Running the Jupyter Notebook

Launching *Jupyter Notebook App:* The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy…) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- Launch the jupyter notebook app

- In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.

- Click on the menu *Help* -> *User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- You can run the notebook document step-by-step (one cell a time) by pressing *shift* + *enter*.

- ❖ You can run the whole notebook in a single step by clicking on the menu *Cell* -> *Run All*.
- ❖ To restart the kernel (i.e. the computational engine), click on the menu *Kernel* -> *Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc…).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An IPYNB file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App: The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a "Dashboard" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels). When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results. Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

14. PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one— and preferably only one —obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas

Syntax and Semantics :

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation :

Main article: Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow :

Python's statements include (among others):

- The assignment statement, using a single equals sign =.

- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.

- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.[80][81] However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Expressions :

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point/division. Python also uses the ** operator for exponentiation.

- From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, == compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example A<=B<=C.
- Python uses the words and, or, not for or its boolean operators rather than the symbolic &&, ||, ! used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5)

first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.

- Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.
- Python has a "string format" operator %. This functions analogously ton printf format strings in C, e.g. “spam=%s eggs=%d” % (“blah”,2) evaluates to “spam=blah eggs=2”. In Python 3 and 2.6+, this was supplemented by the format() method of the str class, e.g. “spam={0} eggs={1}”.format(“blah”,2). Python 3.6 added "f-strings": blah = “blah”; eggs = 2; f‘spam={blah} eggs={eggs}’
- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. “spam” + “eggs” returns “spameggs”. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. “2” + “2” returns “2”.
- Python has various kinds of string literals:
  - Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as

  an escape character. String interpolation became available in Python 3.6 as "formatted string literals".

  - Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
  - Raw string varieties, denoted by prefixing the string literal with an r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.
- Python has array index and array slicing expressions on lists, denoted as a[Key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a=1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c==1) {…} is syntactically valid (but probably unintended) C code but if c=1: … causes a syntax error in Python.

Methods :

Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self parameter access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as print, length, comparison, support for arithmetic operations, type conversion, and many more.

Typing :

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type.

Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection.

Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird’s eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

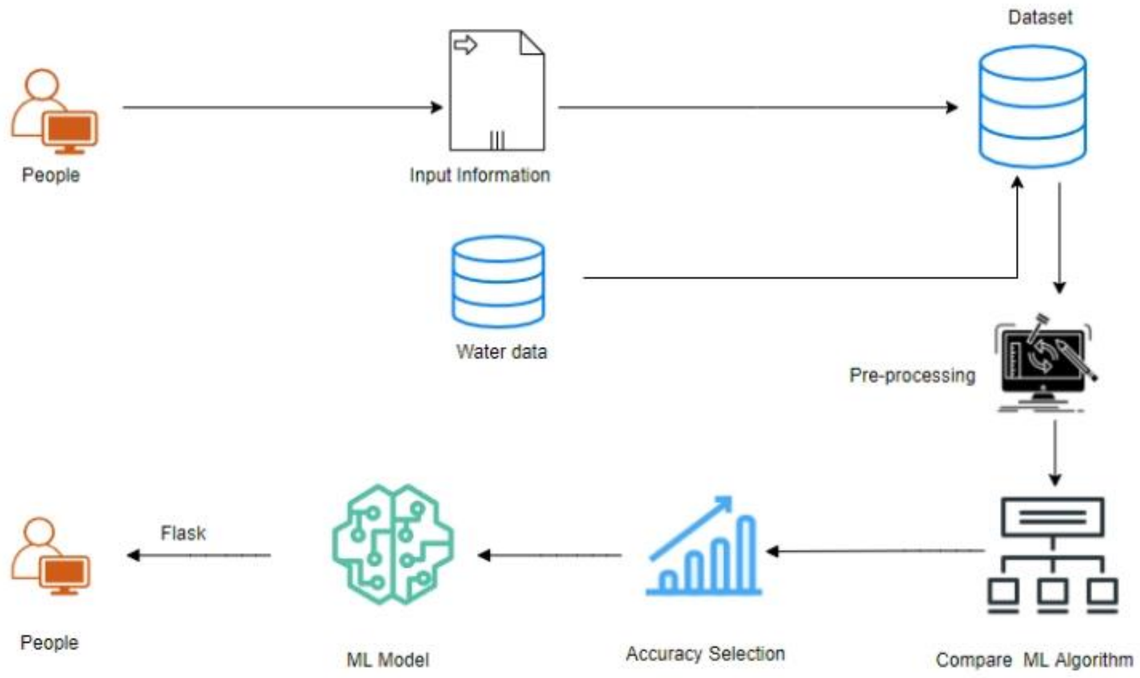
- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

15. System Architecture



16.Work flow diagram

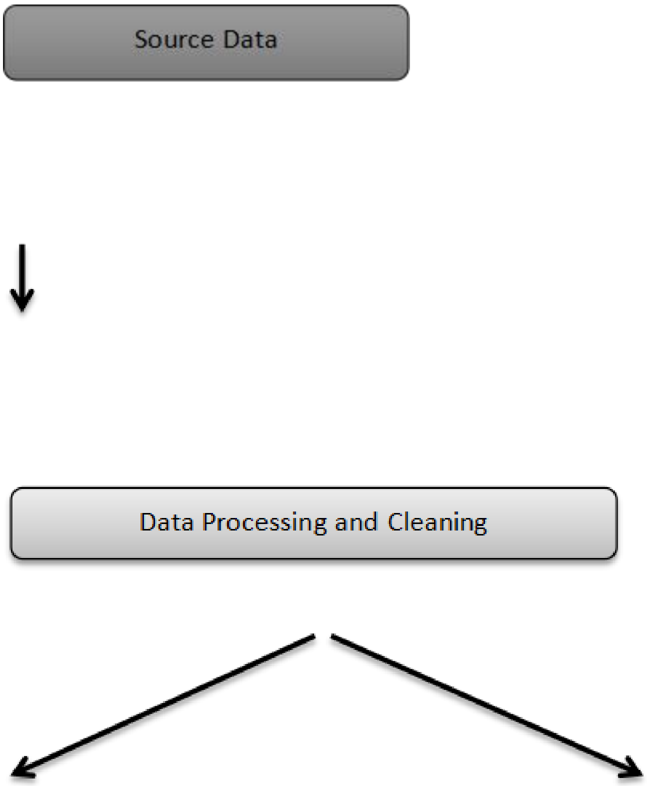


Fig: Workflow Diagram

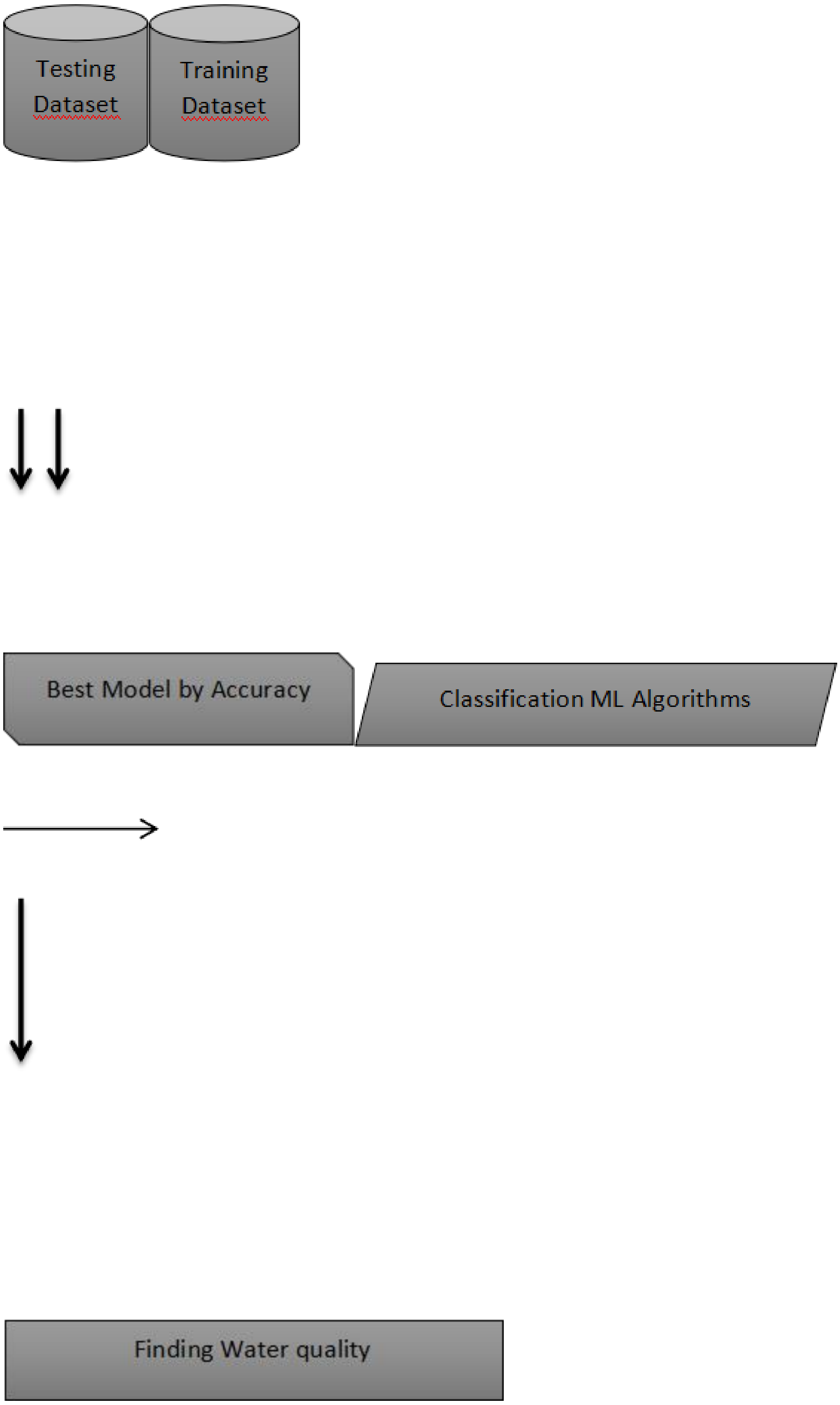
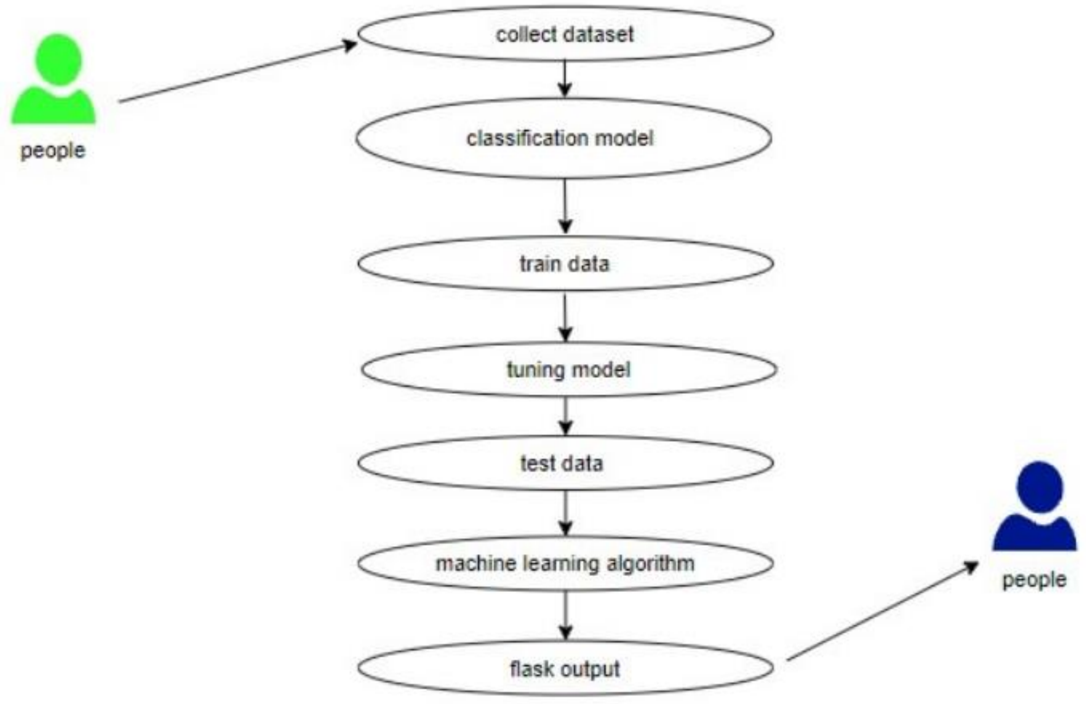


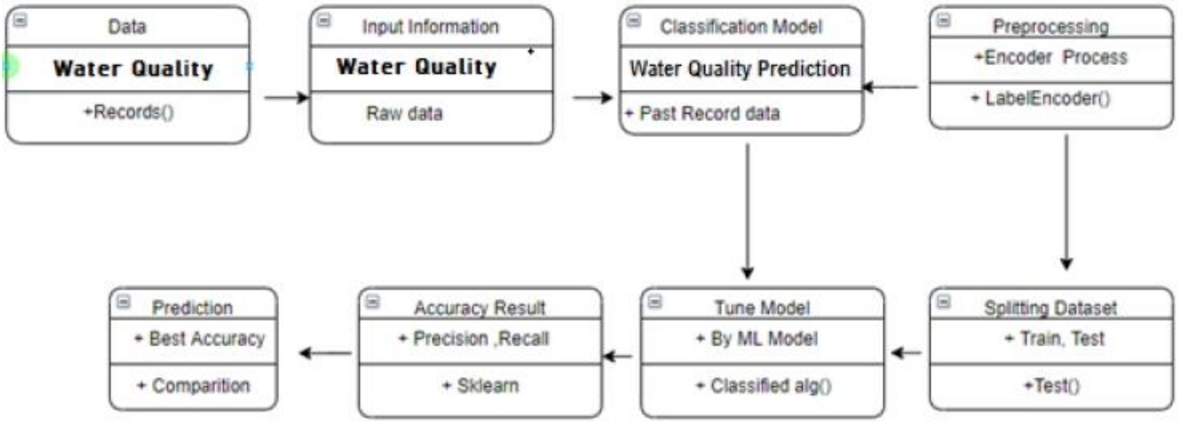
Fig: Workflow Diagram

17.Use Case Diagram



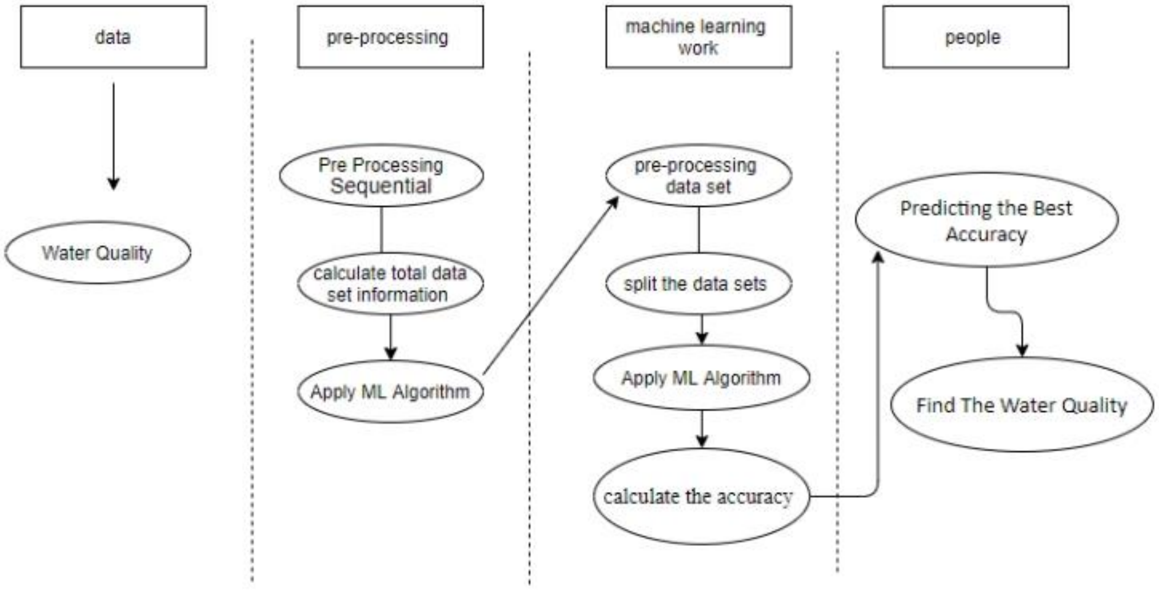
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

18.Class Diagram:



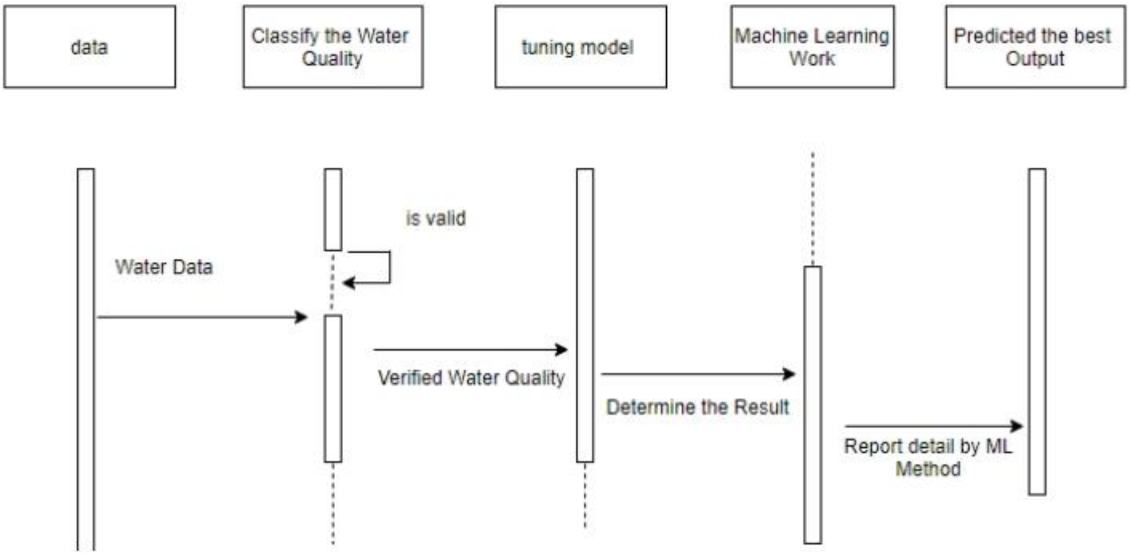
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

19. Activity Diagram:



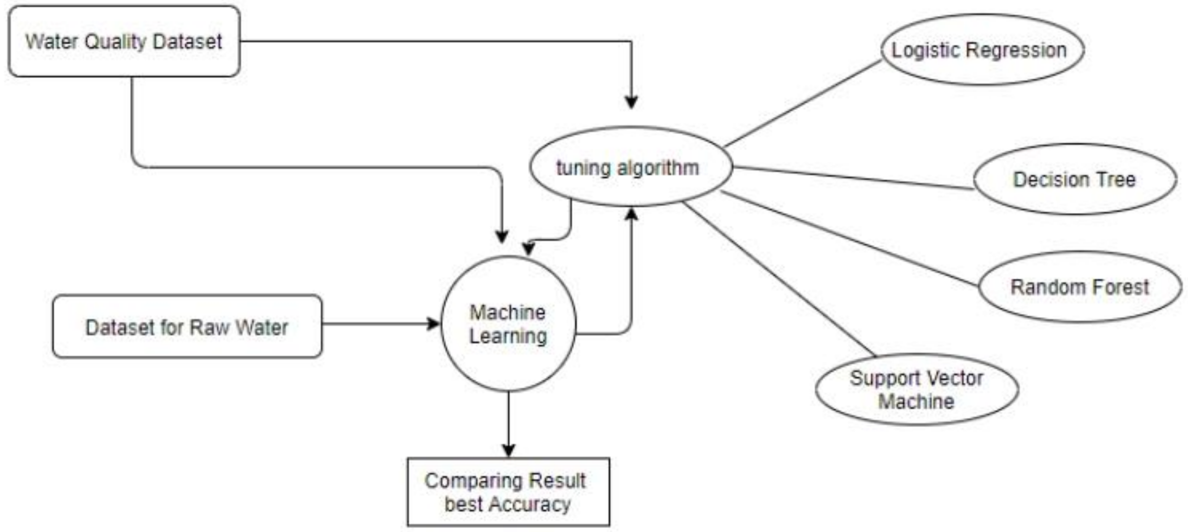
Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

20. Sequence Diagram:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

21. Entity Relationship Diagram (ERD)

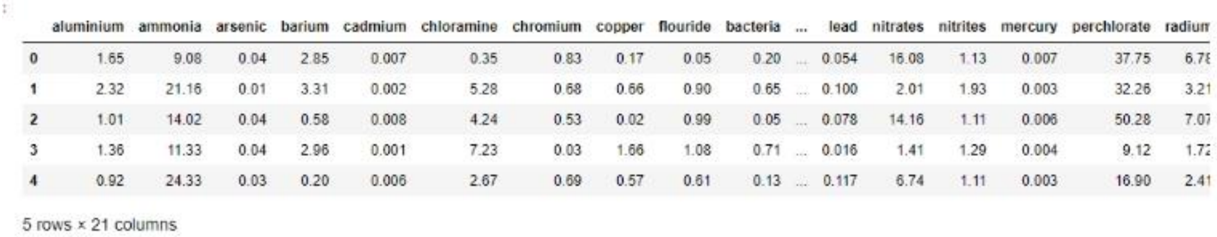


An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

22. Module description:

Data Pre-processing

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

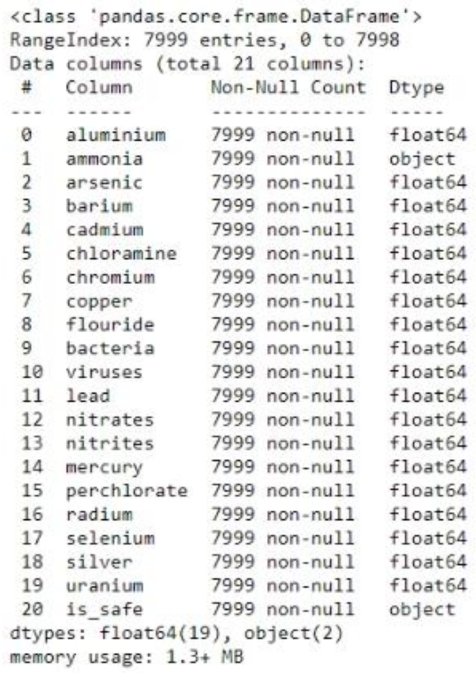


	aluminium	ammonia	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	...	lead	nitrates	nitrites	mercury	perchlorate	radium
0	1.65	9.08	0.04	2.85	0.007	0.35	0.83	0.17	0.05	0.20	...	0.054	16.08	1.13	0.007	37.75	6.78
1	2.32	21.16	0.01	3.31	0.002	5.28	0.68	0.66	0.90	0.65	...	0.100	2.01	1.93	0.003	32.26	3.21
2	1.01	14.02	0.04	0.58	0.008	4.24	0.53	0.02	0.99	0.05	...	0.078	14.16	1.11	0.006	50.28	7.07
3	1.36	11.33	0.04	2.96	0.001	7.23	0.03	1.66	1.08	0.71	...	0.016	1.41	1.29	0.004	9.12	1.72
4	0.92	24.33	0.03	0.20	0.006	2.67	0.69	0.57	0.61	0.13	...	0.117	6.74	1.11	0.003	16.90	2.41

5 rows × 21 columns

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

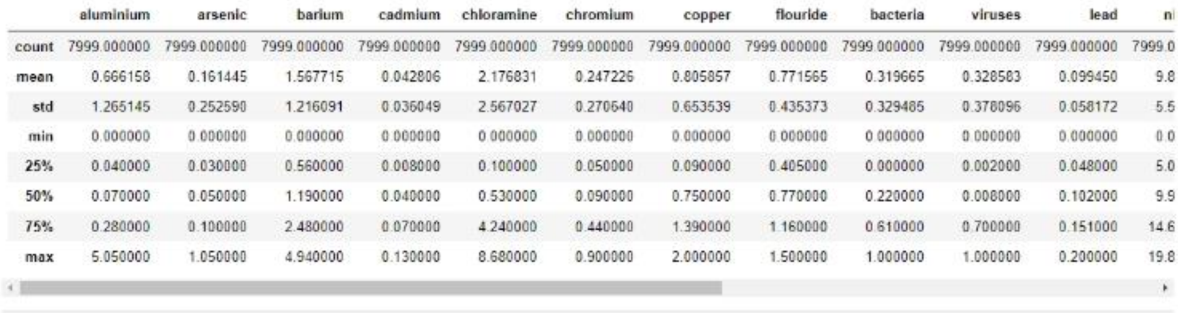
A number of different **data cleaning** tasks using Python’s Pandas library and specifically, it focus on probably the biggest data cleaning task, **missing values** and it able to **more quickly clean data**. It wants to **spend less time cleaning data**, and more time exploring and modeling.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7999 entries, 0 to 7998
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   aluminium    7999 non-null   float64
 1   ammonia      7999 non-null   object
 2   arsenic      7999 non-null   float64
 3   barium       7999 non-null   float64
 4   cadmium      7999 non-null   float64
 5   chloramine   7999 non-null   float64
 6   chromium     7999 non-null   float64
 7   copper       7999 non-null   float64
 8   flouride     7999 non-null   float64
 9   bacteria     7999 non-null   float64
 10  viruses      7999 non-null   float64
 11  lead         7999 non-null   float64
 12  nitrates     7999 non-null   float64
 13  nitrites     7999 non-null   float64
 14  mercury      7999 non-null   float64
 15  perchlorate  7999 non-null   float64
 16  radium       7999 non-null   float64
 17  selenium     7999 non-null   float64
 18  silver       7999 non-null   float64
 19  uranium      7999 non-null   float64
 20  is_safe      7999 non-null   object
dtypes: float64(19), object(2)
memory usage: 1.3+ MB
```

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It’s important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it’s important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.



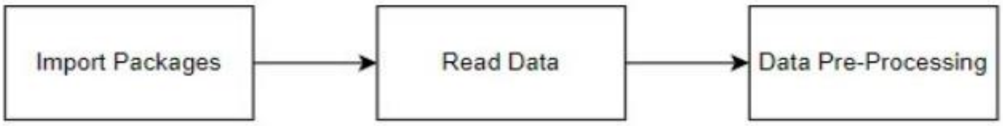
	aluminium	arsenic	barium	cadmium	chloramine	chromium	copper	flouride	bacteria	viruses	lead	ni
count	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.000000	7999.0
mean	0.666158	0.161445	1.567715	0.042806	2.176831	0.247226	0.805857	0.771565	0.319665	0.328583	0.099450	9.8
std	1.265145	0.252590	1.216091	0.036049	2.567027	0.270640	0.653539	0.435373	0.329485	0.378096	0.058172	5.5
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.040000	0.030000	0.560000	0.008000	0.100000	0.050000	0.090000	0.405000	0.000000	0.002000	0.048000	5.0
50%	0.070000	0.050000	1.190000	0.040000	0.530000	0.090000	0.750000	0.770000	0.220000	0.008000	0.102000	9.9
75%	0.280000	0.100000	2.480000	0.070000	4.240000	0.440000	1.390000	1.160000	0.610000	0.700000	0.151000	14.6
max	5.050000	1.050000	4.940000	0.130000	8.680000	0.900000	2.000000	1.500000	1.000000	1.000000	0.200000	19.8

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame

- To specify the type of values
- To create extra columns

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

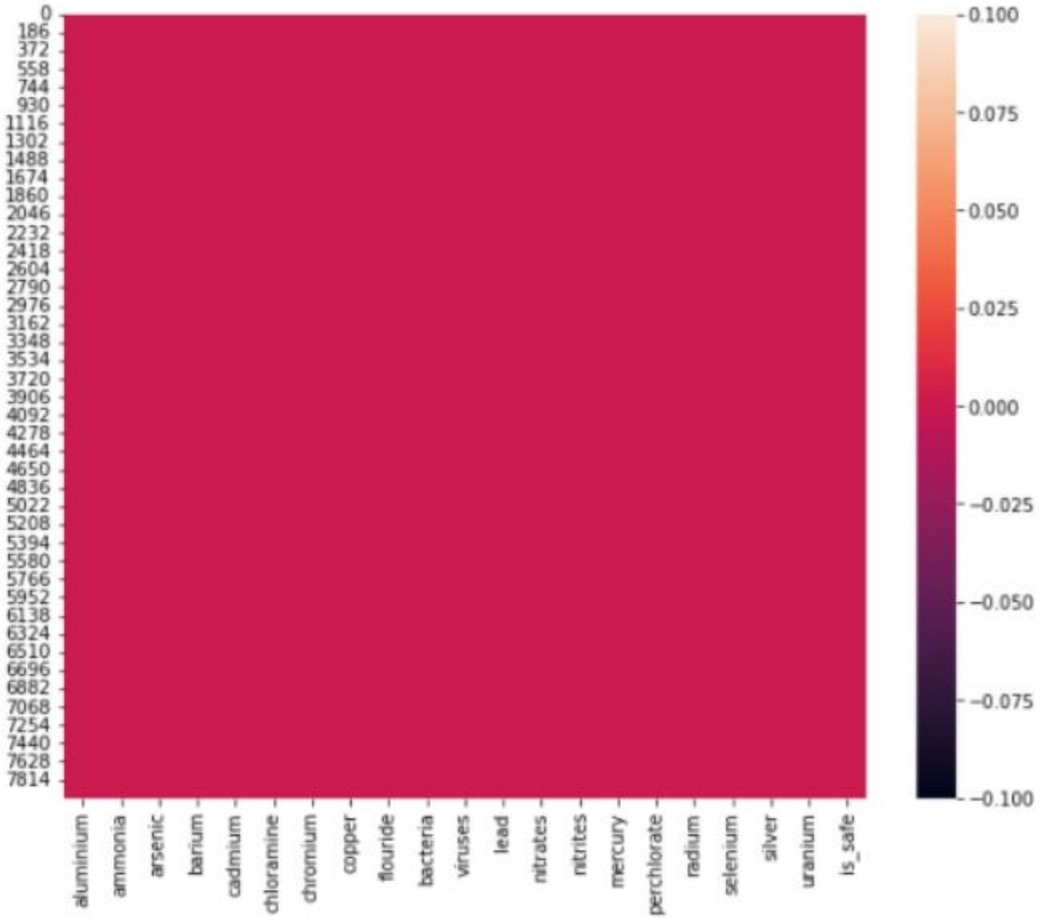
output : removing noisy data

Data Validation/ Cleaning/Preparing Process

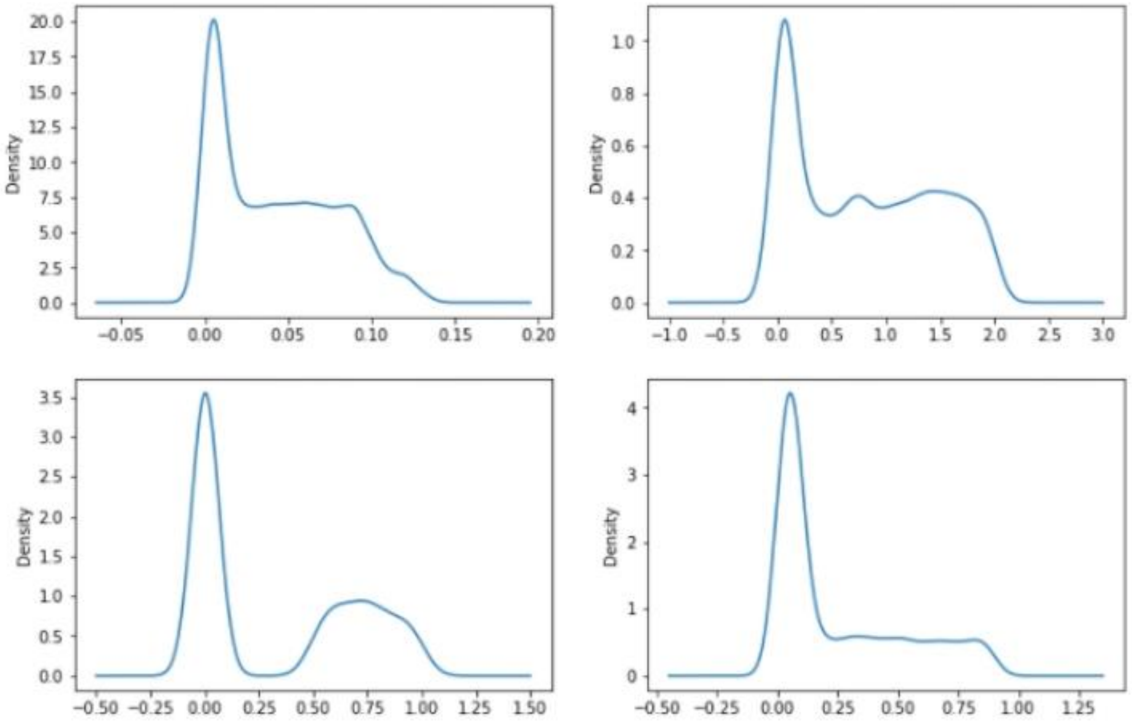
Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process.

The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

Exploration data analysis of visualization



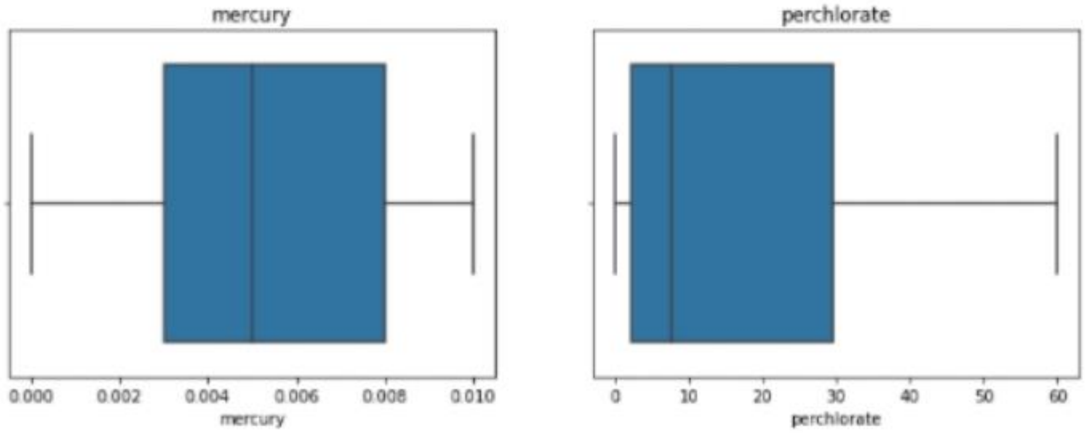
Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.



Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.



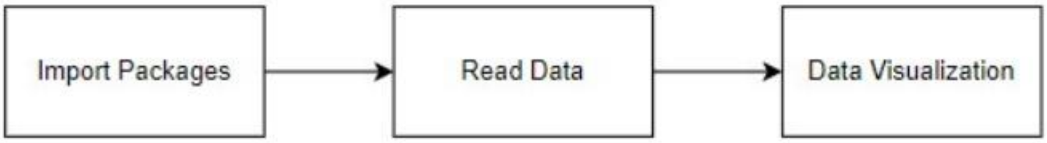
False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

True Positive Rate(TPR) = TP / (TP + FN)

False Positive rate(FPR) = FP / (FP + TN)

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct. (When the model predicts default: how often is correct?)

Precision = TP / (TP + FP)

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

Recall = TP / (TP + FN)

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

General Formula:

F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula:

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

ALGORITHM AND TECHNIQUES

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like train_test_split, DecisionTreeClassifier or Logistic Regression and accuracy_score.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

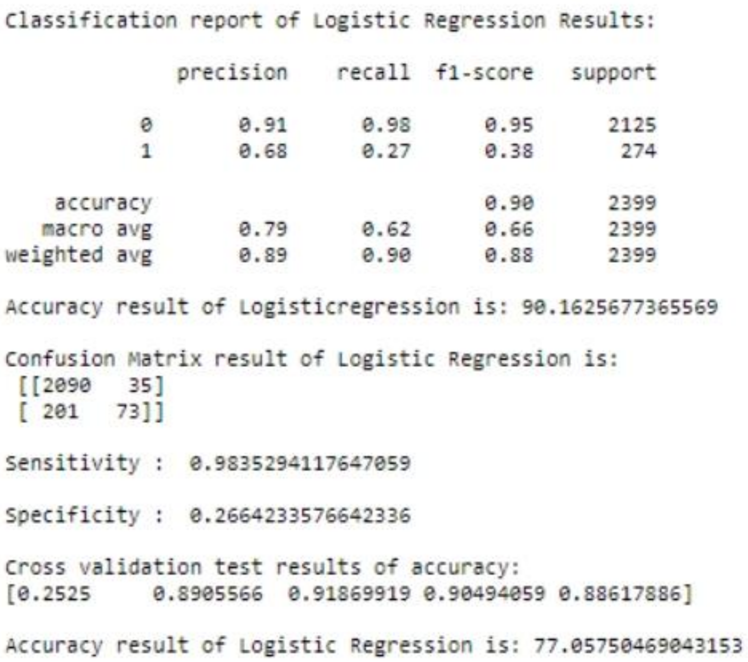
Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.

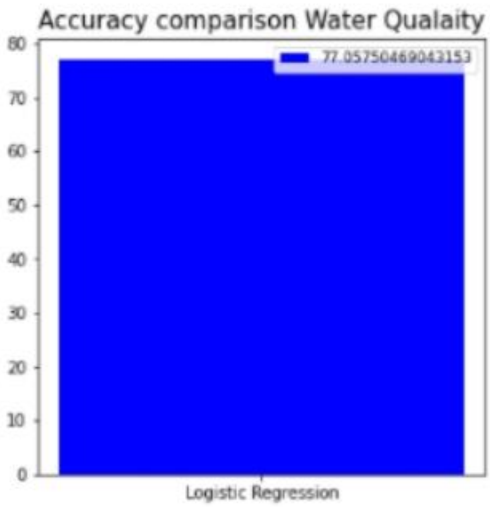
- Data manipulation can be done easily with data frames.

Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

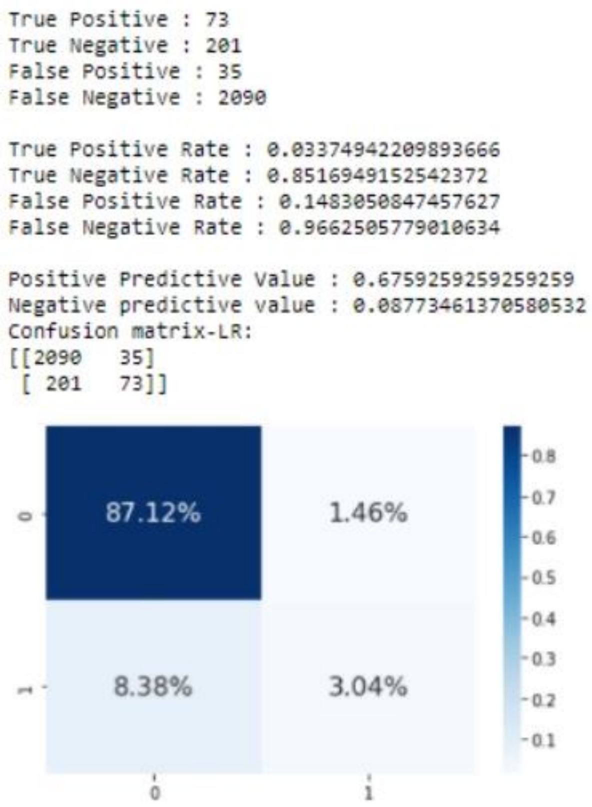


```
Classification report of Logistic Regression Results:

              precision    recall  f1-score   support

           0       0.91      0.98      0.95      2125
           1       0.68      0.27      0.38       274

    accuracy                           0.90      2399
   macro avg       0.79      0.62      0.66      2399
weighted avg       0.89      0.90      0.88      2399

Accuracy result of Logisticregression is: 90.1625677365569

Confusion Matrix result of Logistic Regression is:
 [[2090   35]
 [ 201   73]]

Sensitivity :  0.9835294117647059

Specificity :  0.2664233576642336

Cross validation test results of accuracy:
[0.2525     0.8905566  0.91869919 0.90494059 0.88617886]

Accuracy result of Logistic Regression is: 77.05750469043153
```



In other words, the logistic regression model predicts P(Y=1) as a function of X. Logistic regression Assumptions:

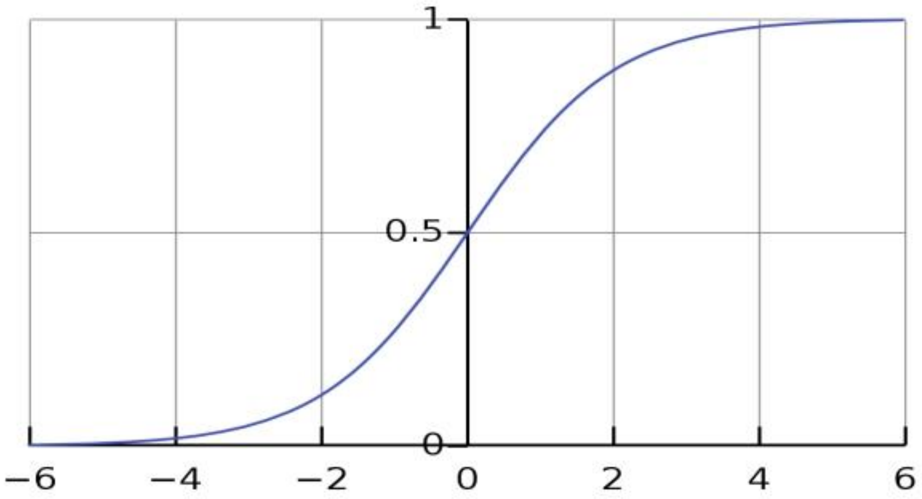
- Logistic regression predicts the output of a categorical dependent variable. qTherefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.



```
True Positive : 73
True Negative : 201
False Positive : 35
False Negative : 2090

True Positive Rate : 0.03374942209893666
True Negative Rate : 0.8516949152542372
False Positive Rate : 0.1483050847457627
False Negative Rate : 0.9662505779010634

Positive Predictive Value : 0.6759259259259259
Negative predictive value : 0.08773461370580532
Confusion matrix-LR:
[[2090   35]
 [ 201   73]]
```

- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- 

1 / (1 + e^-value)

- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}; \textbf{0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binary or Binomial**

  In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

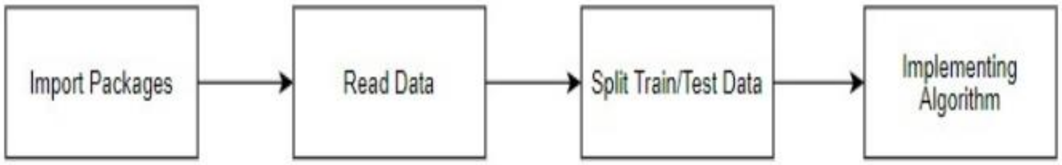
- **Multinomial**

  In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

- **Ordinal**

  In such a kind of classification, dependent variable can have 3 or more possible **ordered** types or the types having a quantitative significance. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

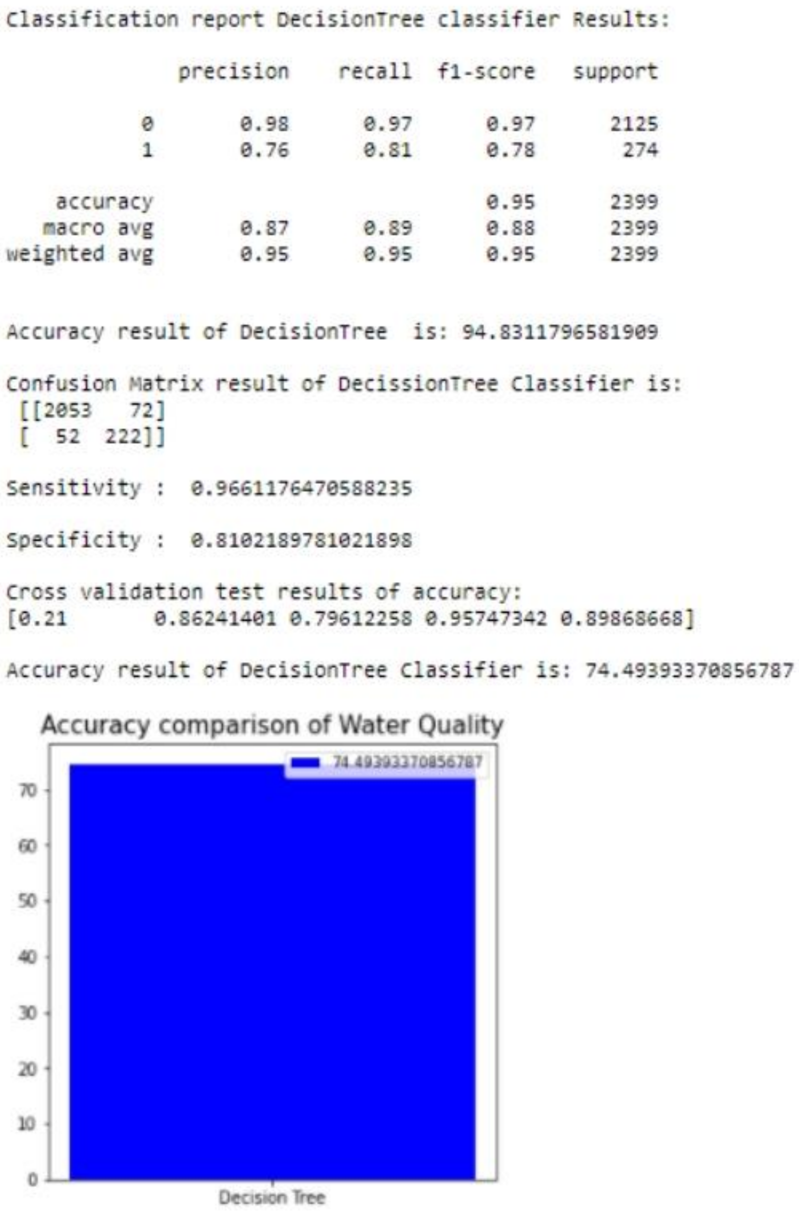
output : getting accuracy

Decision Tree

Introduction to Decision Tree

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below −



```
Classification report DecisionTree classifier Results:

              precision    recall  f1-score   support

           0       0.98      0.97      0.97      2125
           1       0.76      0.81      0.78       274

    accuracy                           0.95      2399
   macro avg       0.87      0.89      0.88      2399
weighted avg       0.95      0.95      0.95      2399


Accuracy result of DecisionTree  is: 94.8311796581909

Confusion Matrix result of DecissionTree Classifier is:
 [[2053   72]
 [  52  222]]

Sensitivity :  0.9661176470588235

Specificity :  0.8102189781021898

Cross validation test results of accuracy:
[0.21       0.86241401 0.79612258 0.95747342 0.89868668]

Accuracy result of DecisionTree Classifier is: 74.49393370856787
```

In the above decision tree, the question are decision nodes and final outcomes are leaves. We have the following two types of decision trees.

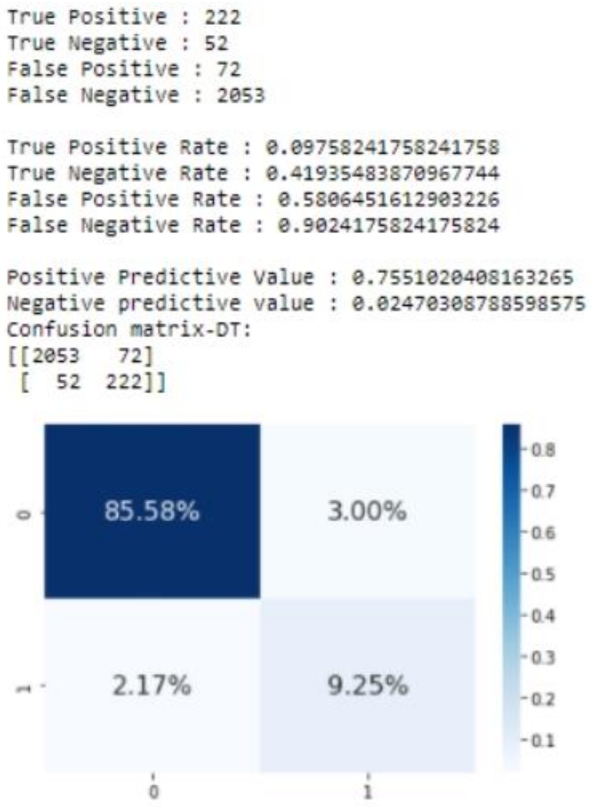
- **Classification decision trees** − In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.

- **Regression decision trees** − In this kind of decision trees, the decision variable is continuous.

Implementing Decision Tree Algorithm

Gini Index

Important Terminology related to Decision Tree



```
True Positive : 222
True Negative : 52
False Positive : 72
False Negative : 2053

True Positive Rate : 0.09758241758241758
True Negative Rate : 0.41935483870967744
False Positive Rate : 0.5806451612903226
False Negative Rate : 0.9024175824175824

Positive Predictive Value : 0.7551020408163265
Negative predictive value : 0.02470308788598575
Confusion matrix-DT:
[[2053   72]
 [  52  222]]
```

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.

3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

Assumptions while creating Decision Tree

Below are some of the assumptions we make while using Decision tree:

- In the beginning, the whole training set is considered as the **root.**
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- Records are **distributed recursively** on the basis of attribute values.
- Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

Entropy

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random.

From the above graph, it is quite evident that the entropy H(X) is zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because it projects perfect randomness in the data and there is no chance if perfectly determining the outcome.

Mathematically Entropy for 1 attribute is represented as:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Where **S → Current state, and Pi → Probability of an event *i* of state S or Percentage of class *i* in a node of state S.**

Information Gain

Information gain or **IG** is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Information gain is a decrease in entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. ID3 (Iterative Dichotomiser) decision tree algorithm uses information gain.

Mathematically, IG is represented as:

$$\text{Information Gain(T,X) = Entropy(T) - Entropy(T, X)}$$

In a much simpler way, we can conclude that:

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

Information Gain

Where “before” is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

Split Creation

A split is basically including an attribute in the dataset and a value. We can create a split in dataset with the help of following three parts −

- **Part 1: Calculating Gini Score** − We have just discussed this part in the previous section.
- **Part 2: Splitting a dataset** − It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute.

After getting the two groups - right and left, from the dataset, we can calculate the value of split by using Gini score calculated in first part. Split value will decide in which group the attribute will reside.

- **Part 3: Evaluating all splits** − Next part after finding Gini score and splitting dataset is the evaluation of all splits. For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree.

Building a Tree

As we know that a tree has root node and terminal nodes. After creating the root node, we can build the tree by following two parts −

Part 1: Terminal node creation

While creating terminal nodes of decision tree, one important point is to decide when to stop growing tree or creating further terminal nodes. It can be done by using two criteria namely maximum tree depth and minimum node records as follows −

- **Maximum Tree Depth** − As name suggests, this is the maximum number of the nodes in a tree after root node. We must stop adding terminal nodes once a tree reached at maximum depth i.e. once a tree got maximum number of terminal nodes.
- **Minimum Node Records** − It may be defined as the minimum number of training patterns that a given node is responsible for. We must stop adding terminal nodes once tree reached at these minimum node records or below this minimum.

Terminal node is used to make a final prediction.

Part 2: Recursive Splitting

As we understood about when to create terminal nodes, now we can start building our tree. Recursive splitting is a method to build the tree. In this method, once a node is created, we can create the child nodes (nodes added to an existing node) recursively on each group of data, generated by splitting the dataset, by calling the same function again and again.

Prediction

After building a decision tree, we need to make a prediction about it. Basically, prediction involves navigating the decision tree with the specifically provided row of data.

We can make a prediction with the help of recursive function, as did above. The same prediction routine is called again with the left or the child right nodes.

Assumptions

The following are some of the assumptions we make while creating decision tree −

- While preparing decision trees, the training set is as root node.
- Decision tree classifier prefers the features values to be categorical. In case if you want to use continuous values then they must be done discretized prior to model building.
- Based on the attribute’s values, the records are recursively distributed.
- Statistical approach will be used to place attributes at any node position i.e.as root node or internal node.

Advantages:

- The main advantage of decision trees is how **easy** they are **to interpret**. While other machine Learning models are close to black boxes, decision trees provide a graphical and intuitive way to understand what our algorithm does.

- Compared to other Machine Learning algorithms Decision Trees require **less data** to train.
- They can be used for **Classification** and **Regression**.
- They are **simple**.
- They are tolerant to **missing value**s.

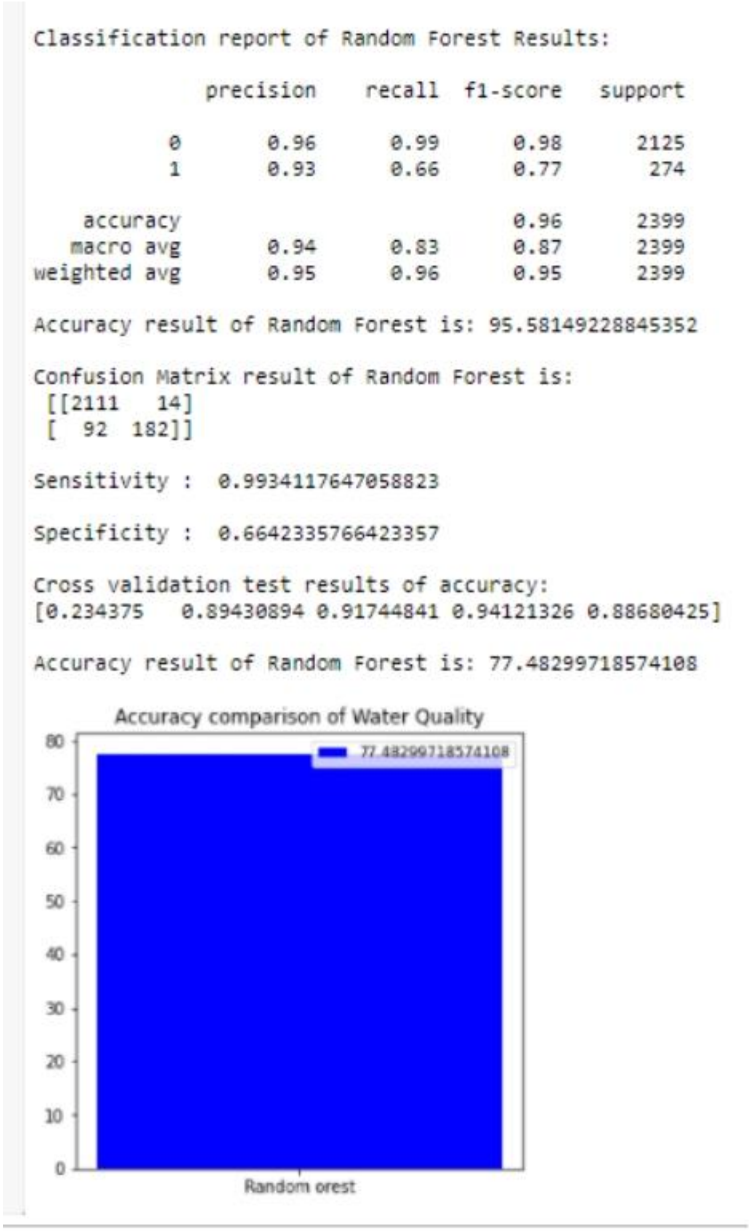
Disadvantages

- They are quite prone to **over fitting** to the training data and can be sensible to outliers.

They are weak learners: a single decision tree normally does not make great predictions, so multiple trees are often combined to make 'forests' to give birth to stronger ensemble models. This will be discussed in a further post.

Random forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.



```
Classification report of Random Forest Results:

              precision    recall  f1-score   support

           0       0.96      0.99      0.98      2125
           1       0.93      0.66      0.77       274

    accuracy                           0.96      2399
   macro avg       0.94      0.83      0.87      2399
weighted avg       0.95      0.96      0.95      2399

Accuracy result of Random Forest is: 95.58149228845352

Confusion Matrix result of Random Forest is:
 [[2111   14]
 [  92  182]]

Sensitivity :  0.9934117647058823

Specificity :  0.6642335766423357

Cross validation test results of accuracy:
[0.234375   0.89430894 0.91744841 0.94121326 0.88680425]

Accuracy result of Random Forest is: 77.48299718574108
```

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:

> Note: To better understand the Random Forest Algorithm, you should have knowledge of the Decision Tree Algorithm.

Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

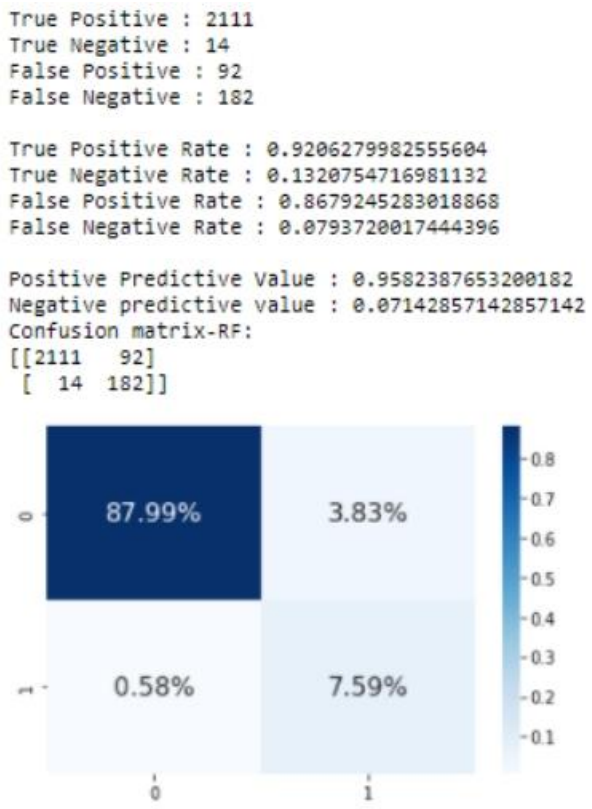
Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

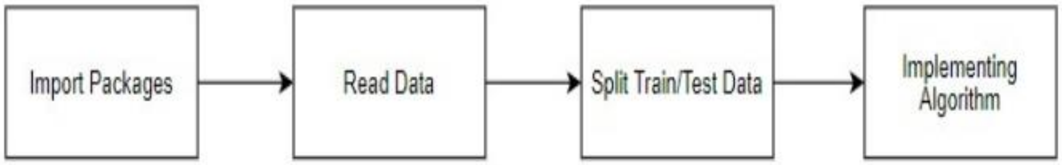


```
True Positive : 2111
True Negative : 14
False Positive : 92
False Negative : 182

True Positive Rate : 0.9206279982555604
True Negative Rate : 0.1320754716981132
False Positive Rate : 0.8679245283018868
False Negative Rate : 0.0793720017444396

Positive Predictive Value : 0.9582387653200182
Negative predictive value : 0.07142857142857142
Confusion matrix-RF:
[[2111   92]
 [  14  182]]
```

The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Implementation in Scikit-learn

For each decision tree, Scikit-learn calculates a nodes importance using Gini Importance, assuming only two child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

- ni sub(j)= the importance of node j
- w sub(j) = weighted number of samples reaching node j
- C sub(j)= the impurity value of node j
- left(j) = child node from left split on node j
- right(j) = child node from right split on node j

sub() is being used as subscript isn’t available in Medium

See method compute_feature_importances in _tree.pyx

The importance for each feature on a decision tree is then calculated as:

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

- fi sub(i)= the importance of feature i
- ni sub(j)= the importance of node j

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$normfi_i = \frac{fi_i}{\sum_{j \in all\, features} fi_j}$$

The final feature importance, at the Random Forest level, is it’s average over all the trees. The sum of the feature’s importance value on each trees is calculated and divided by the total number of trees:

$$RFfi_i = \frac{\sum_{j \in all\, trees} normfi_{ij}}{T}$$

- RFfi sub(i)= the importance of feature i calculated from all trees in the Random Forest model
- normfi sub(ij)= the normalized feature importance for i in tree j
- T = total number of trees

See method feature_importances_ in forest.py

Notation was inspired by this StackExchange thread which I found incredible useful for this post.

Implementation in Spark

For each decision tree, Spark calculates a feature’s importance by summing the gain, scaled by the number of samples passing through the node:

$$fi_i = \sum\nolimits_{j:nodes\ j\ splits\ on\ feature\ i} s_j C_j$$

- fi sub(i) = the importance of feature i
- s sub(j) = number of samples reaching node j
- C sub(j) = the impurity value of node j

See method computeFeatureImportance in treeModels.scala

To calculate the final feature importance at the Random Forest level, first the feature importance for each tree is normalized in relation to the tree:

$$normfi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j}$$

- normfi sub(i) = the normalized importance of feature i
- fi sub(i) = the importance of feature i

Then feature importance values from each tree are summed normalized:

$$RFfi_i = \frac{\sum_j normfi_{ij}}{\sum_{j\in all\ features, k\in all\ trees} normfi_{jk}}$$

- RFfi sub(i)= the importance of feature i calculated from all trees in the Random Forest model
- normfi sub(ij)= the normalized feature importance for i in tree j

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

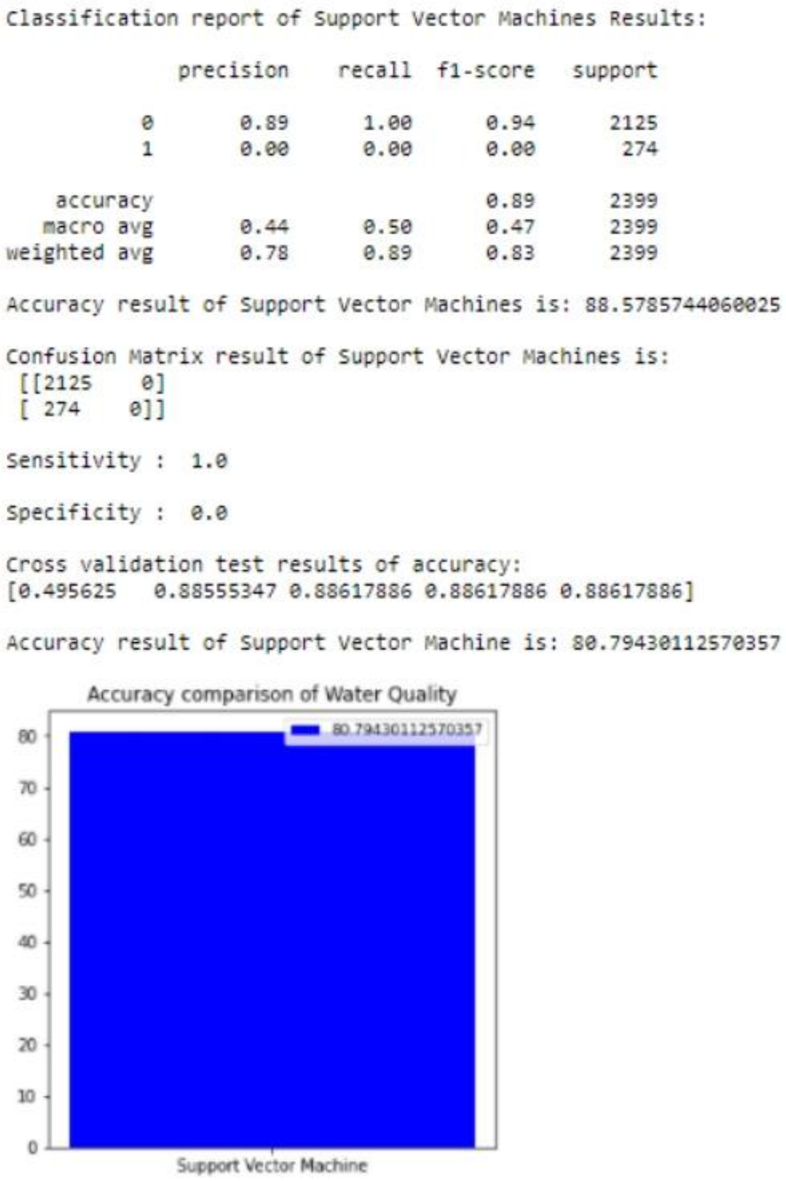
- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Support Vector Machines.

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

The objective of applying SVMs is to find the best line in two dimensions or the best hyperplane in more than two dimensions in order to help us separate our space into classes. The hyperplane (line) is found through the **maximum margin,** i.e., the maximum distance between data points of both classes.

Don't you think the definition and idea of SVM look a bit abstract? No worries, let me explain in details.



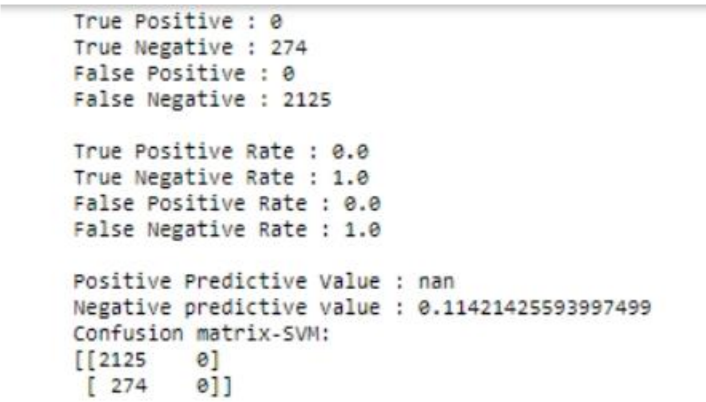
```
Classification report of Support Vector Machines Results:

              precision    recall  f1-score   support

           0       0.89      1.00      0.94      2125
           1       0.00      0.00      0.00       274

    accuracy                           0.89      2399
   macro avg       0.44      0.50      0.47      2399
weighted avg       0.78      0.89      0.83      2399

Accuracy result of Support Vector Machines is: 88.5785744060025

Confusion Matrix result of Support Vector Machines is:
 [[2125    0]
 [ 274    0]]

Sensitivity :  1.0

Specificity :  0.0

Cross validation test results of accuracy:
[0.495625   0.88555347 0.88617886 0.88617886 0.88617886]

Accuracy result of Support Vector Machine is: 80.79430112570357
```

Support Vector, Hyperplane, and Margin

The vector points closest to the hyperplane are known as the **support vector points** because only these two points are contributing to the result of the algorithm, and other points are not. If a data point is not a support vector, removing it has no effect on the model. On the other hand, deleting the support vectors will then change the position of the hyperplane.

The dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

The distance of the vectors from the hyperplane is called the **margin,** which is a separation of a line to the closest class points. We would like to choose a hyperplane that maximises the margin between classes. The graph below shows what good margin and bad margin are.



```
True Positive : 0
True Negative : 274
False Positive : 0
False Negative : 2125

True Positive Rate : 0.0
True Negative Rate : 1.0
False Positive Rate : 0.0
False Negative Rate : 1.0

Positive Predictive Value : nan
Negative predictive value : 0.11421425593997499
Confusion matrix-SVM:
[[2125    0]
 [ 274    0]]
```

Hard Margin

If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

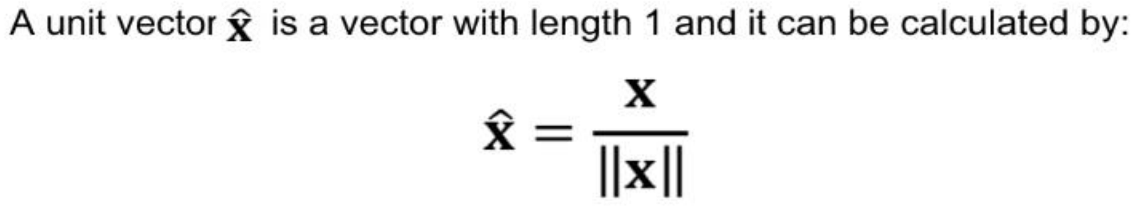
Soft Margin

As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur, which is called soft margin classification. It is better to have a large margin, even though some constraints are violated. Margin violation means choosing a hyperplane, which can allow some data points to stay in either the incorrect side of the hyperplane and between the margin and the correct side of the hyperplane.

In order to find the **maximal margin**, we need to maximize the margin between the data points and the hyperplane. In the following session, I will share the mathematical concepts behind this algorithm.

Linear Algebra Revisited

Before we move on, let’s review some concepts in Linear Algebra.

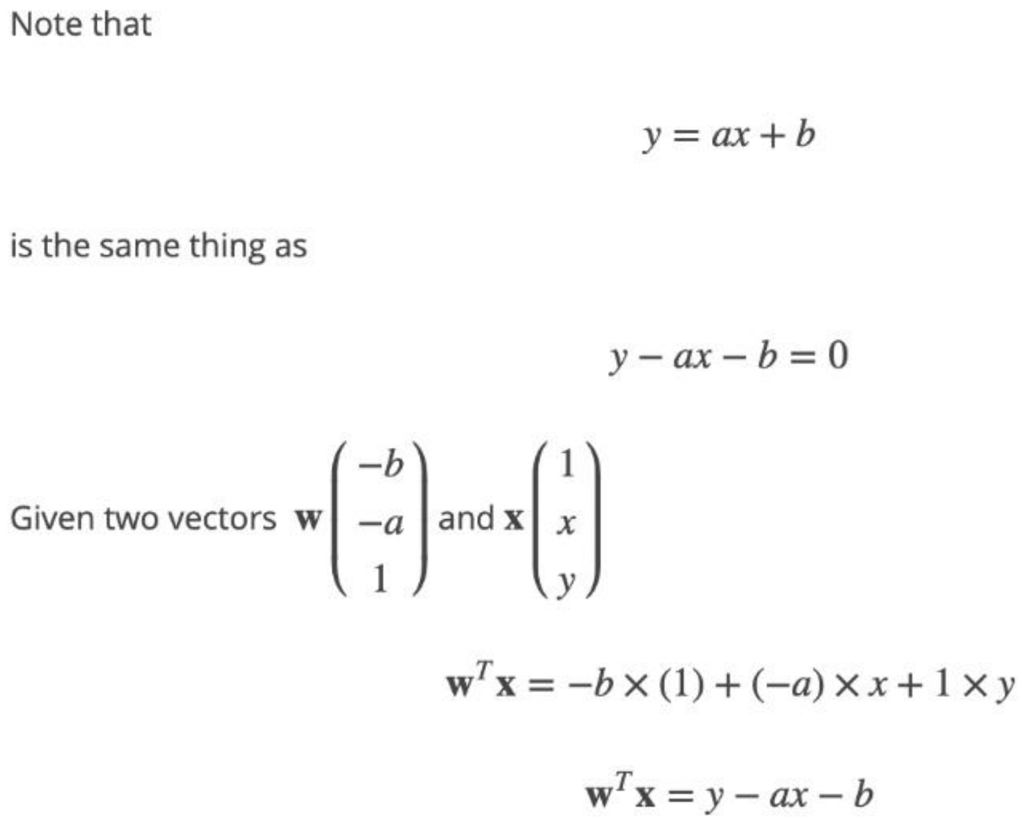


A unit vector $\hat{\mathbf{x}}$ is a vector with length 1 and it can be calculated by:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{||\mathbf{x}||}$$

Maximising the Margin

You probably learned that an equation of a line is **y=ax+b**. However, you will often find that the equation of a hyperplane is defined by:



Note that

$$y = ax + b$$

is the same thing as

$$y - ax - b = 0$$

Given two vectors $\mathbf{w}\begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ and $\mathbf{x}\begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$\mathbf{w}^T\mathbf{x} = -b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w}^T\mathbf{x} = y - ax - b$$

The two equations are just two different ways of expressing the same thing.

For **Support Vector Classifier** (SVC), we use ���T���+��� where ��� is the weight vector, and ��� is the bias.

$$\mathbf{w}^T\mathbf{x} + b = 0$$

You can see that the name of the variables in the hyperplane equation are **w** and **x,** which means they are vectors! A vector has magnitude (size) and direction, which works perfectly well in 3 or more dimensions. Therefore, the application of “**vector”** is used in the SVMs algorithm.

$$(\mathbf{x}^+ - \mathbf{x}^-)\cdot\hat{\mathbf{w}} = (\mathbf{x}^+ - \mathbf{x}^-)\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|} = \mathbf{x}^+\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{x}^-\cdot\frac{\mathbf{w}}{\|\mathbf{w}\|}$$

The equation of calculating the Margin.

Cost Function and Gradient Updates

Maximizing-Margin is equivalent to Minimizing Loss.

In the SVM algorithm, we are looking to maximize the **margin** between the data points and the hyperplane. The loss function that helps maximize the margin is **hinge loss**.

$$L(w) = \sum_{i=1} \underbrace{max(0, 1 - y_i[w^T x_i + b])}_{\text{Loss function}} + \underbrace{\lambda||w||_2^2}_{\text{regularization}}$$

λ=1/C (C is always used for regularization coefficient).

The function of the first term, **hinge loss,** is to penalize misclassifications. It measures the error due to misclassification (or data points being closer to the classification boundary than the margin). The second term is the regularization term, which is a technique to avoid overfitting by penalizing large coefficients in the solution vector. The λ(lambda) is the regularization coefficient, and its major role is to determine the trade-off between increasing the margin size and ensuring that the xi lies on the correct side of the margin.

“Hinge” describes the fact that the error is 0 if the data point is classified correctly (and is not too close to the decision boundary).

When the true class is -1 (as in your example), the hinge loss looks like this in the graph.

We need to minimise the above loss function to find the max-margin classifier.

We can derive the formula for the margin from the **hinge-loss**. If a data point is on the margin of the classifier, the hinge-loss is exactly zero. Hence, on the margin, we have:

$$\Rightarrow y_i[w^T x_i + b] = 1$$

$$\max(0, 1 - y_i[w^T x_i + b]) = 0$$

Note that ���i is either +1 or -1.

Therefore, we have:

$$\mathbf{x}_+ \cdot \frac{\mathbf{w}}{||\mathbf{w}||} - \mathbf{x}_- \cdot \frac{\mathbf{w}}{||\mathbf{w}||} = \frac{1-b}{||\mathbf{w}||} - \frac{-1-b}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Our objective function is then:

$$\max \frac{2}{||\mathbf{w}||} \rightarrow \max \frac{1}{||\mathbf{w}||} \rightarrow \min||\mathbf{w}|| \rightarrow \ \min \frac{1}{2}||\mathbf{w}||^2$$

To minimize such an objection function, we should then use Lagrange Multiplier.

Classifying non-linear data

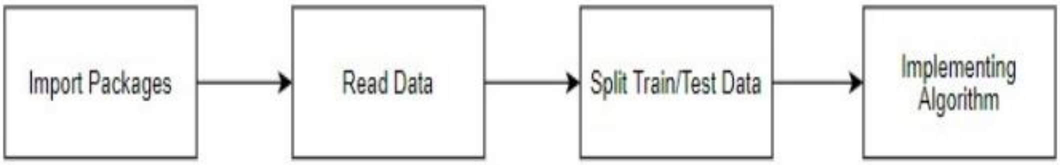
What about data points are not linearly separable?

Non-linear separate.

SVM has a technique called the kernel trick. These are functions that take low dimensional input space and transform it into a higher-dimensional space, i.e., it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. This is shown as follows:

Mapping to a Higher Dimension

MODULE DIAGRAM

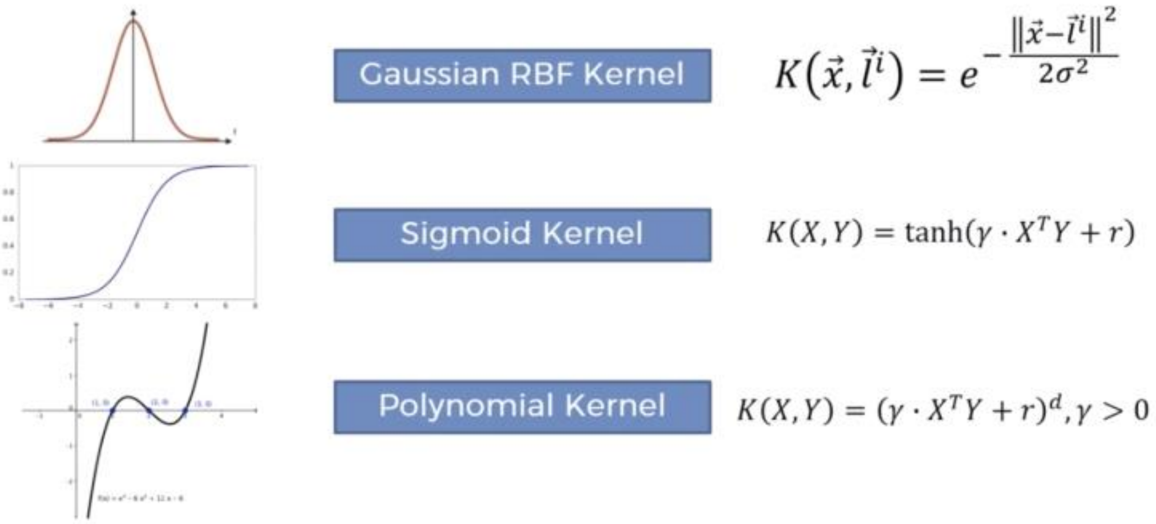


GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Some Frequently Used Kernels



Gaussian RBF Kernel $K(\vec{x},\vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$

Sigmoid Kernel $K(X,Y) = \tanh(\gamma \cdot X^T Y + r)$

Polynomial Kernel $K(X,Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$

Deploy

Flask (web framework)

Flask is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

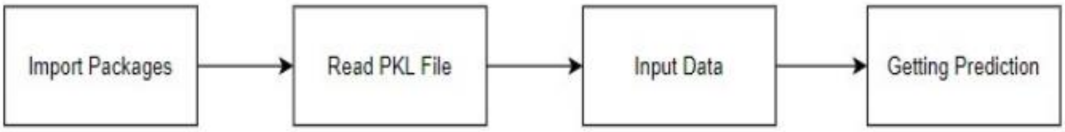
In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Flask is part of the Pallets Projects, and based on several others of them.

Flask is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let’s take a closer look into Flask, so-called “micro” framework for Python.

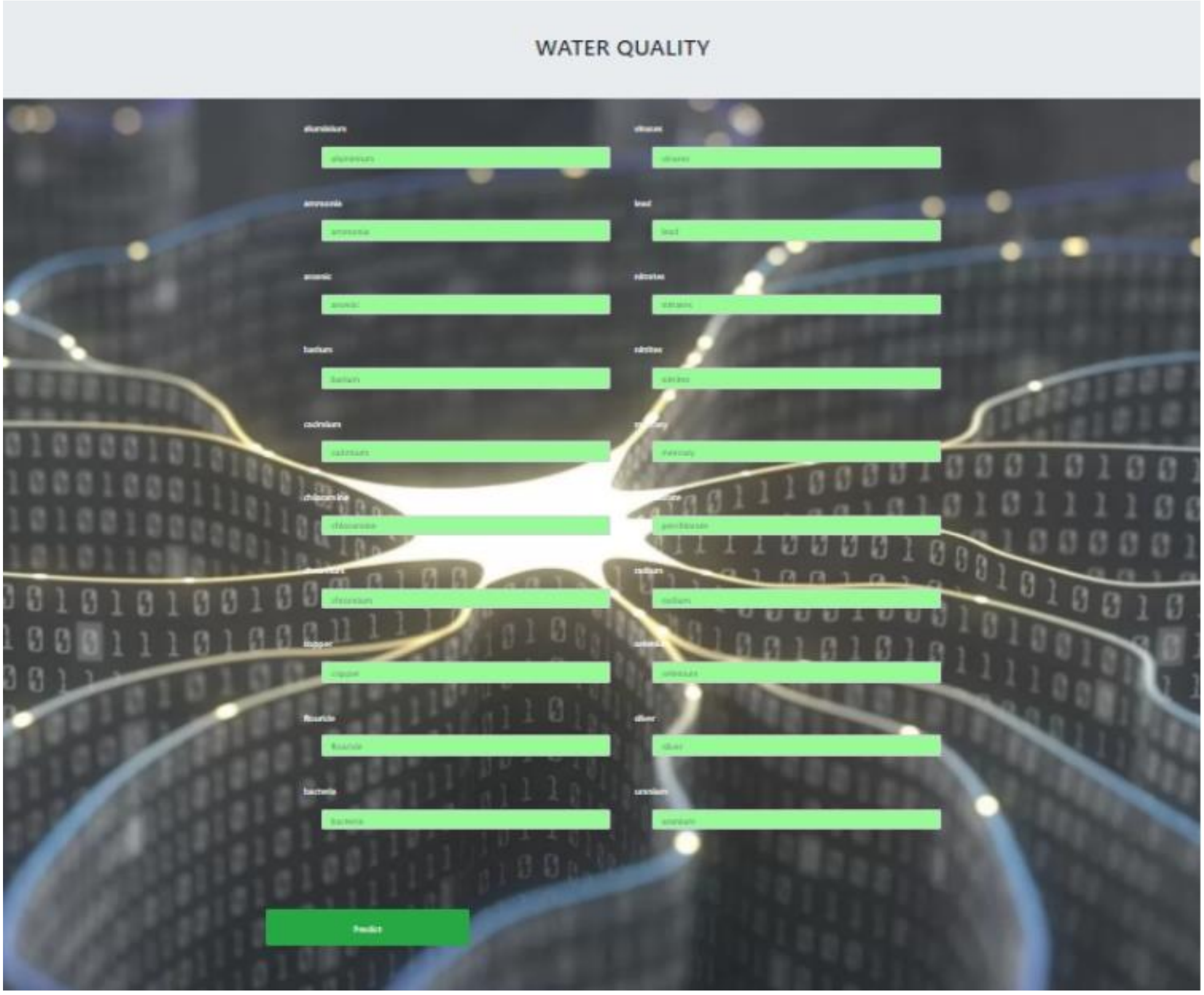
MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data values

output : predicting output



FEATURES:

Flask was designed to be **easy to use and extend**. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to **plug in any extensions** you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Flask as your web application framework if we have immensely powerful Django, Pyramid, and don’t forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Flask is pretty impressive too with its:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies (Client Side Sessions)
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus Flask gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.
- It is super easy to deploy Flask in production (Flask is 100%_WSGI 1.0 compliant”)

- HTTP request handling functionality
- High Flexibility
  The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Flask is one of the most polished and feature-rich micro frameworks available. Still young, Flask has a thriving community, first-class extensions, and an **elegant API**. Flask comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Flask

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user’s interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Flask is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Flask is a good start if you are getting into web development. Flask is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Flask:

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from Python language. Flask provides a library and a collection of codes that can be used to build websites, without the need to do everything from scratch. But Framework flask still doesn't use the Model View Controller (MVC) method.

Flask-RESTful is an extension for Flask that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Flask-Restful is a lightweight abstraction that works with the existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup.

Flask Restful is an extension for Flask that adds support for building REST APIs in Python using Flask as the back-end. It encourages best practices and is very easy to set up. Flask restful is very easy to pick up if you're already familiar with flask.

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

Start Using an API

1. Most APIs require an API key. ...

2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an __init__.py file inside) or a standard module (just a .py file).

For more information about resource loading, see open resource().

Usually you create a Flask instance in your main module or in the __init__.py file of your package.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to view_func.__name__.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.

- **provide_automatic_options** (*Optional[bool]*) – Add the OPTIONS method and respond to OPTIONS requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.

Return type -- None

After_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after_request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use teardown_request() for that.

Parameters:

f (*Callable[[Response*], *Response]*)

Return type

Callable[[Response], Response]

after_request_funcs: t.Dict[AppOrBlueprintKey,

t.List[AfterRequestCallable]]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the after_request() decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

app_context()

Create an AppContext. Use as a with block to push the context, which will make current_app point at this application.

An application context is automatically pushed by RequestContext.push() when handling a request, and when running a CLI command. Use this to manually create a context outside of these situations.

With app.app_context():

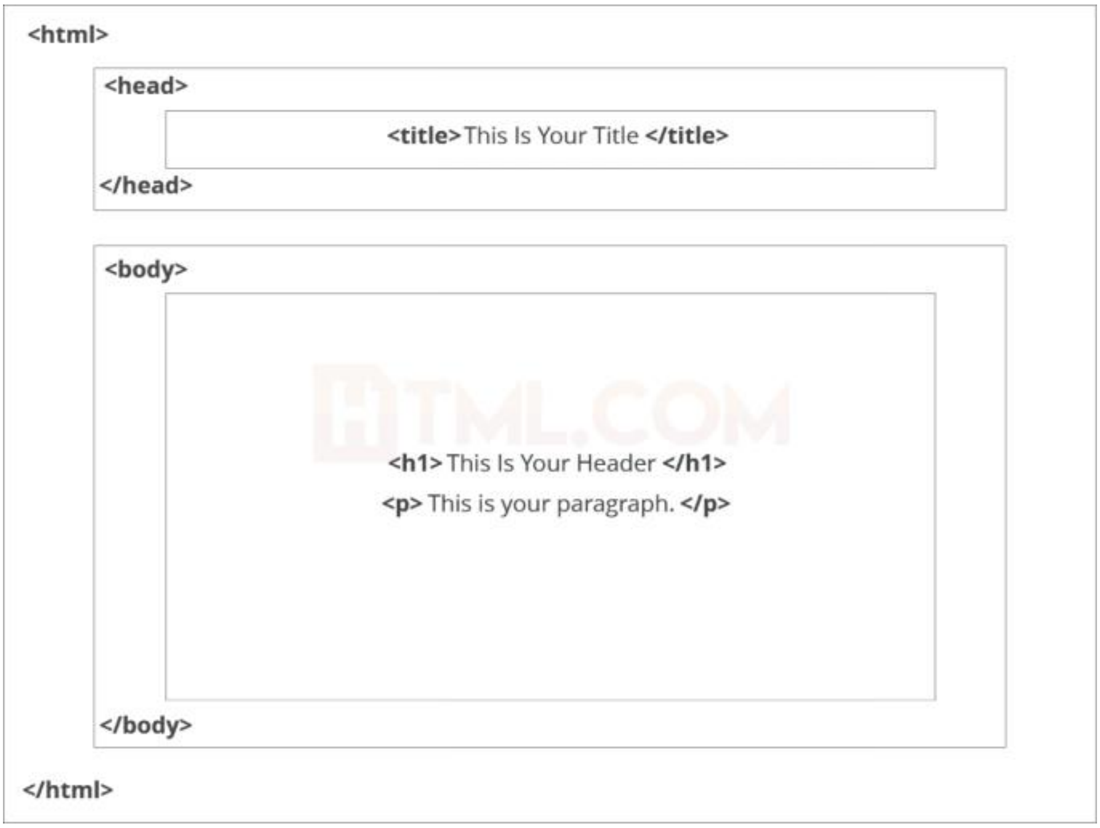
Init_db()

24.HTML Introduction

HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



```
<html>
    <head>
        <title>This Is Your Title </title>
    </head>

    <body>
        <h1> This Is Your Header </h1>
        <p> This is your paragraph. </p>
    </body>
</html>
```

<!DOCTYPE html> — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

<html> — This tag signals that from here on we are going to write in HTML code.

<head> — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

<body> — This is where the **content of the page** goes.

Further Tags

Inside the `<head>` tag, there is one tag that is always included: `<title>`, but there are others that are just as important:

<title>

> This is where we **insert the page name** as it will appear at the top of the browser window or tab.

<meta>

> This is where information *about* the document is stored: character encoding, name (page context), description.

Head Tag

<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.

<meta name="author" content="Conor Sheils">

</header>

Adding Content

Next, we will make `<body>` tag.

The HTML `<body>` is where we add the content which is designed for viewing by human eyes.

This includes **text, images, tables, forms** and everything else that we see on the internet each day.

Add HTML Headings To Web Page

In HTML, headings are written in the following elements:

- `<h1>`
  - `<h2>`
    - `<h3>`
    - `<h4>`
    - `<h5>`
    - `<h6>`

As you might have guessed `<h1>` and `<h2>` should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Search engine bots use this order when deciphering which information is most important on a page.

Creating Your Heading

Let’s try it out. On a new line in the HTML editor, type:

```
<h1>Welcome to My Page</h1>
```

And hit save. We will save this file as “index.html” in a new folder called “my webpage.”

The Moment of Truth: Click the newly saved file and your first ever web page should open in your default browser. It may not be pretty it’s yours… all yours. *Evil laugh*

Element	Meaning	Purpose
<b>	Bold	Highlight important information
<strong>	Strong	Similarly to bold, to highlight key text
<i>	Italic	To denote text
<em>	Emphasised Text	Usually used as image captions
<mark>	Marked Text	Highlight the background of the text
<small>	Small Text	To shrink the text
<strike>	Striked Out Text	To place a horizontal line across the text
<u>	Underlined Text	Used for links or text highlights
<ins>	Inserted Text	Displayed with an underline to show an inserted text
<sup>	Superscript Text	Another typographical presentation style

Add Text In HTML

Adding text to our HTML page is simple using an element opened with the tag `<p>` which **creates a new paragraph**. We place all of our regular text inside the element `<p>`.

When we write text in HTML, we also have a number of other elements we can use to **control the text or make it appear in a certain way**.

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the **<a>** tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href="http://www.google.com">Google</a>
```

Image Tag

In today's modern digital world, images are everything. The **<img>** tag has everything you need to display images on your site. Much like the <a> anchor element, <img> also contains an attribute.

The attribute features information for your computer regarding the **source**, **height**, **width** and **alt text** of the image

```
<img src="yourimage.jpg" alt="Describe the image" height="X" width="X">
```

CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users.CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let’s try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS Syntax

```
Selector {

                    Property 1 : value;

                    Property 2 : value;

                    Property 3 : value;

           }
```

For example:

```
        h1

          {

             Color: red;

              Text-align: center;

          }

         #unique

         {

              color: green;
```

```
}
```

- Selector: selects the element you want to target
- Always remains the same whether we apply internal or external styling
- There are few basic selectors like tags, id’s, and classes
- All forms this key-value pair
- Keys: properties(attributes) like color, font-size, background, width, height,etc
- Value: values associated with these properties

CSS Comment

- Comments don’t render on the browser
- Helps to understand our code better and makes it readable.
- Helps to debug our code
- Two ways to comment:
    - Single line

CSS How-To

- There are 3 ways to write CSS in our HTML file.
    - Inline CSS
    - Internal CSS
    - External CSS

- Priority order
    - Inline > Internal > External

Inline CSS

- Before CSS this was the only way to apply styles

- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost
- Example:

```
<h3 style=” color:red”> Have a great day </h3>

<p  style =” color: green”> I did this , I did that </p>
```

Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document
- Example:

```
 < style>

         h1{

              color:red;

             }

  </style>

 <h3> Have a great day </h3>
```

External CSS

- With the help of <link> tag in the head tag, we can apply styles

- Reference is added
- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document
- Example:

```
<head>

 <link rel="stylesheet" type="text/css" href="name of the Css file">

</head>

         h1{

               color:red;        //.css file

            }
```

CSS Selectors

- The selector is used to target elements and apply CSS
- Three simple selectors
    - Element Selector
    - Id Selector
    - Class Selector

- Priority of Selectors

CSS Colors

- There are different colouring schemes in CSS
- **RGB-**This starts with RGB and takes 3 parameter

- **HEX-**Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA-**This starts with RGB and takes 4 parameter

CSS Background

- There are different ways by which CSS can have an effect on HTML elements
- Few of them are as follows:
  - Color – used to set the color of the background
  - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
  - Image – used to set an image as the background
  - Position – used to determine the position of the image
  - Attachment – It basically helps in controlling the mechanism of scrolling

CSS BoxModel

- Every element in CSS can be represented using the BOX model
- It allows us to add a border and define space between the content
- It helps the developer to develop and manipulate the elements
- It consists of 4 edges
  - Content edge – It comprises of the actual content
  - Padding edge – It lies in between content and border edge
  - Border edge – Padding is followed by the border edge
  - Margin edge – It is an outside border and controls the margin of the element

26. Coding

MODEL 1 .Data validation and pre-processing

```
#import package

import pandas as pd

import numpy as ny

#read dataset
```

```
data = pd.read_csv('water.csv')

# view head

data.head()

#view tail

data.tail()

# shape of the project

data.shape

# size ofthe data

data.size

#columns

data.columns

data.isnull()

data.isnull().sum()

data.dropna()

# information about dataset

data.info()

data.duplicated()

data.duplicated().sum()

# describe

data.describe()

data['arsenic'].nunique()

data['barium'].nunique()

data["uranium"].nunique()
```

```
# check the unique values

data["aluminium"].nunique()

data["viruses"].nunique()

data["mercury"].nunique()

data["silver"].unique()

data["mercury"].unique()

data["arsenic"].unique()

data["uranium"].unique()

data["chloramine"].unique()

data["aluminium"].unique()

data.head()

#min max values

print("minimum values of aluminium :", data["aluminium"].min())

print("mean values of alumimum :",data["aluminium"].mean())

print("maximum values of aluminium :", data["aluminium"].max())

#min max values

print("minimum values of aluminium :", data["chloramine"].min())

print("mean values of alumimum :",data["chloramine"].mean())

print("maximum values of aluminium :", data["chloramine"].max())

#min max values

print("minimum values of aluminium :", data["copper"].min())

print("mean values of alumimum :",data["copper"].mean())

print("maximum values of aluminium :", data["copper"].max())
```

In [ ]:

```
#min max values
print("minimum values of aluminium :", data["flouride"].min())
print("mean values of alumimum :",data["flouride"].mean())
print("maximum values of aluminium :", data["flouride"].max())
#min max values
print("minimum values of aluminium :", data["lead"].min())
print("mean values of alumimum :",data["lead"].mean())
print("maximum values of aluminium :", data["lead"].max())
#min max values
print("minimum values of aluminium :", data["nitrates"].min())
print("mean values of alumimum :",data["nitrates"].mean())
print("maximum values of aluminium :", data["nitrates"].max())
#min max values
print("minimum values of aluminium :", data["radium"].min())
print("mean values of alumimum :",data["radium"].mean())
print("maximum values of aluminium :", data["radium"].max())
data.corr()
data.corr().describe()
data["mercury"].value_counts()
pd.Categorical(data["uranium"]).describe()
data.info()
data.columns
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var_mod = (['aluminium', 'ammonia', 'arsenic', 'barium', 'cadmium',
'chloramine',

       'chromium', 'copper', 'flouride', 'bacteria', 'viruses', 'lead',

       'nitrates', 'nitrites', 'mercury', 'perchlorate', 'radium',
'selenium',

       'silver', 'uranium', 'is_safe'])

for i in var_mod:

           data[i] = le.fit_transform(data[i]). astype(int)

data.head()
```

Model-2:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

data = pd.read_csv("water.csv")

data.head()

data["is_safe"].unique()

data["is_safe"].unique()
```

```
data[data["is_safe"]=='#NUM!']

data = data.drop(data[data["is_safe"]=='#NUM!'].index)

data["is_safe"].unique()

data["is_safe"] = data["is_safe"].astype(float)

data.info()

data.columns

data.isnull().sum()

data =data.dropna()

plt.figure(figsize= (10,8))

sns.heatmap(data.isnull())

plt.show()

plt.figure(figsize=(9,6))

sns.scatterplot(x=data['aluminium'],y=data["barium"])

plt.show()

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

sns.stripplot(x = data["cadmium"], y = data["chromium"])

data.columns

plt.figure(figsize= (12,8))

plt.subplot(2,2,1)

data["cadmium"].plot(kind='density')

plt.subplot(2,2,2)

data["copper"].plot(kind='density')

plt.subplot(2,2,3)
```

```
data["viruses"].plot(kind='density')
plt.subplot(2,2,4)
data["chromium"].plot(kind='density')
plt.show()
data.hist(figsize=(20,35),layout =(19,3) )
plt.show()
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.title("mercury")
sns.boxplot(data["mercury"])
plt.subplot(1,2,2)
plt.title("perchlorate")
sns.boxplot(data["perchlorate"])
plt.show()
#Propagation by variable
def PropByVar(data, variable):
    dataframe_pie = data[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%',
fontsize = 12)
    ax.set_title(variable + ' \n', fontsize = 15)
    return np.round(dataframe_pie/data.shape[0]*100,2)


PropByVar(data, 'is_safe')
```

```
plt.figure(figsize=(15,12))

sns.heatmap(data.corr(), annot =True)

plt.show()

data.columns

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['aluminium', 'ammonia', 'arsenic', 'barium', 'cadmium',
'chloramine',

       'chromium', 'copper', 'flouride', 'bacteria', 'viruses', 'lead',

       'nitrates', 'nitrites', 'mercury', 'perchlorate', 'radium',
'selenium',

       'silver', 'uranium', 'is_safe']

for i in var:

    data[i] = le.fit_transform(data[i])

data.head()
```

Module 3 : Performance measurements of Logistic regression

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n
```

```
#Load given dataset

data = p.read_csv("water.csv")

import warnings

warnings.filterwarnings('ignore')

data.head(5)

data.tail(5)

data.isnull().sum()

data = data.dropna()

data.shape

data.duplicated().sum()

data.info()

data["is_safe"].unique()

data[data["is_safe"]=='#NUM!']

data = data.drop(data[data["is_safe"]=='#NUM!'].index)

data["is_safe"].unique()

data.describe()

data.corr()

df = data

df.columns

#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test
set.
```

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

X = data.drop(labels='is_safe', axis=1)

#Response variable

y = data.loc[:,'is_safe']

#We'll use a test size of 30%. We also stratify the split on the response
variable, which is very important to do because there are so few fraudulent
transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0, stratify=y)
```

Logistic Regression :

```
from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import cross_val_score



logR= LogisticRegression()



logR.fit(X_train,y_train)



predictLR = logR.predict(X_test)



print("")

print('Classification report of Logistic Regression Results:')
print("")

print(classification_report(y_test,predictLR))

x = (accuracy_score(y_test,predictLR)*100)


print('Accuracy result of Logisticregression is:', x)


print("")


cm2=confusion_matrix(y_test,predictLR)

print('Confusion Matrix result of Logistic Regression is:\n',cm2)

print("")

sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])

print('Sensitivity : ', sensitivity2 )

print("")

specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])

print('Specificity : ', specificity2)

print("")


accuracy = cross_val_score(logR, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")
print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100





def graph():

    import matplotlib.pyplot as plt

    data=[LR]

    alg="Logistic Regression"

    plt.figure(figsize=(5,5))

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison Water Qualaity",fontsize=15)

    plt.legend(b,data,fontsize=9)





graph()

TN = cm2[1][0]

FN = cm2[0][0]

TP = cm2[1][1]

FP = cm2[0][1]
```

```
print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)
print("")
TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

cm2=confusion_matrix(y_test, predictLR)
print('Confusion matrix-LR:')
print(cm2)
```

```
s.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16}, fmt='.2%',)

plt.show()
```

Module 4 : Performance measurements of DecisionTree:

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n

#Load given dataset

data = p.read_csv("water.csv")

import warnings

warnings.filterwarnings('ignore')

data.head(5)

data.tail(5)

data.isnull().sum()

data = data.dropna()

data.shape

data.duplicated().sum()

data["is_safe"].unique()

data[data["is_safe"]=='#NUM!']

data = data.drop(data[data["is_safe"]=='#NUM!'].index)
```

```
data["is_safe"].unique()

data.info()

data.describe()

data.corr()

df = data

df.columns

#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test
set.

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

X = data.drop(labels='is_safe', axis=1)

#Response variable

y = data.loc[:,'is_safe']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0, stratify=y)
```

DecisionTree:

```
from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import cross_val_score



DT=DecisionTreeClassifier()
```

```
DT.fit(X_train,y_train)


predictDT = DT.predict(X_test)


print("")
print('Classification report DecisionTree classifier Results:')
print("")
print(classification_report(y_test,predictDT))


print("")
x = (accuracy_score(y_test,predictDT)*100)


print('Accuracy result of DecisionTree  is:', x)
print("")


cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of DecissionTree Classifier is:\n',cm2)
print("")
sensitivity2 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity2 )
print("")
specificity2 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity2)
print("")


accuracy = cross_val_score(DT, X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)
#get the mean of each fold
print("")
print("Accuracy result of DecisionTree Classifier is:",accuracy.mean() *
100)
dt=accuracy.mean() * 100




def graph():
    import matplotlib.pyplot as plt
    data=[dt]
    alg="Decision Tree"
    plt.figure(figsize=(5,5))
    b=plt.bar(alg,data,color=("b"))
    plt.title("Accuracy comparison of Water Quality",fontsize=15)
    plt.legend(b,data,fontsize=9)
```

```
graph()

TN = cm2[1][0]

FN = cm2[0][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)
```

```
NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)



cm2=confusion_matrix(y_test, predictDT)

print('Confusion matrix-DT:')

print(cm2)

s.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16}, fmt='.2%',)

plt.show()
```

Module 5 : Performance measurements of Random Forest algorithms

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as s

import numpy as n

#Load given dataset

data = p.read_csv("water.csv")

import warnings
```

```
warnings.filterwarnings('ignore')

data.head(5)

data.tail(5)

data.isnull().sum()

data = data.dropna()

data.shape

data.duplicated().sum()

data["is_safe"].unique()

data[data["is_safe"]=='#NUM!']

data = data.drop(data[data["is_safe"]=='#NUM!'].index)

data["is_safe"].unique()

data.info()

data.describe()

data.corr()

df = data

df.columns

#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test
set.

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

X = data.drop(labels='is_safe', axis=1)

#Response variable
```

```
y = data.loc[:,'is_safe']

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, stratify=y)
```

Random Forest:

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import cross_val_score



rfc = RandomForestClassifier()



rfc.fit(X_train,y_train)



predictR = rfc.predict(X_test)



print("")

print('Classification report of Random Forest Results:')

print("")



print(classification_report(y_test,predictR))

x = (accuracy_score(y_test,predictR)*100)
```

```
print('Accuracy result of Random Forest is:', x)

print("")

cm1=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of Random Forest is:\n',cm1)

print("")

sensitivity1 = cm1[0,0]/(cm1[0,0]+cm1[0,1])

print('Sensitivity : ', sensitivity1 )

print("")

specificity1 = cm1[1,1]/(cm1[1,0]+cm1[1,1])

print('Specificity : ', specificity1)

print("")



accuracy = cross_val_score(rfc, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Random Forest is:",accuracy.mean() * 100)

RFC=accuracy.mean() * 100




def graph():
    import matplotlib.pyplot as plt

    data=[RFC]

    alg="Random orest"

    plt.figure(figsize=(5,5))

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison of Water Quality")

    plt.legend(b,data,fontsize=9)
```

```
graph()

TN = cm1[0][1]

FN = cm1[1][1]

TP = cm1[0][0]

FP = cm1[1][0]

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)
```

```
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)
print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)
print("")
PPV = TP/(TP+FP)
NPV = TN/(TN+FN)
print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)



cm2=confusion_matrix( predictR,y_test)
print('Confusion matrix-RF:')
print(cm2)


s.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size":
16},fmt='.2%')
plt.show()
```

Module 6 : Performance measurements of SVM

```
#import library packages

import pandas as p

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as n

#Load given dataset

data = p.read_csv("water.csv")

import warnings

warnings.filterwarnings('ignore')

data.head(5)

data.isnull().sum()

data = data.dropna()

data.duplicated().sum()

data["is_safe"].unique()

data[data["is_safe"]=='#NUM!']

data = data.drop(data[data["is_safe"]=='#NUM!'].index)

data["is_safe"].unique()

data.info()

df = data

df.columns
```

```
#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test
set.

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

X = data.drop(labels='is_safe', axis=1)

#Response variable

y = data.loc[:,'is_safe']

#We'll use a test size of 30%. We also stratify the split on the response
variable, which is very important to do because there are so few fraudulent
transactions.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0, stratify=y)
```

SVC

```
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix

from sklearn.model_selection import cross_val_score




s = SVC()



s.fit(X_train,y_train)
```

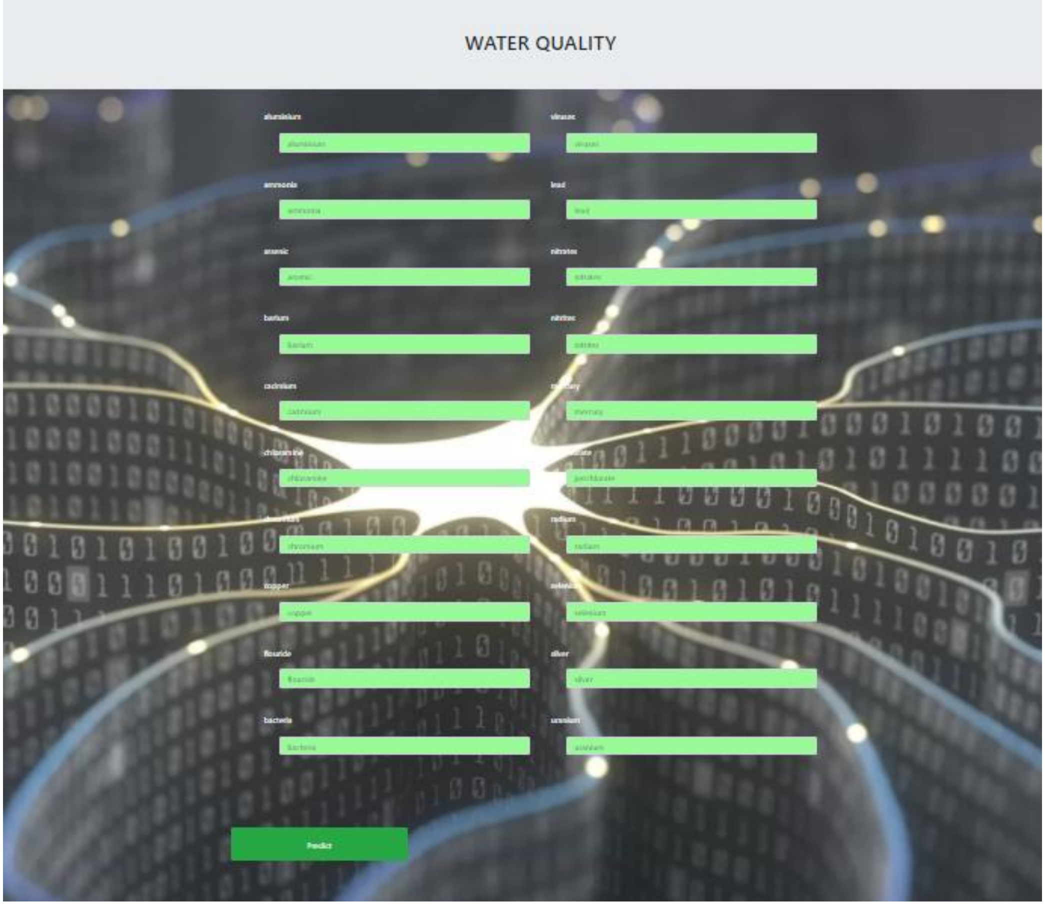
```
predicts = s.predict(X_test)

print("")
print('Classification report of Support Vector Machines Results:')
print("")

print(classification_report(y_test,predicts))
x = (accuracy_score(y_test,predicts)*100)

print('Accuracy result of Support Vector Machines is:', x)
print("")
cm2=confusion_matrix(y_test,predicts)
print('Confusion Matrix result of Support Vector Machines is:\n',cm2)
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
print("")

accuracy = cross_val_score(s,X, y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Support Vector Machine is:",accuracy.mean() * 100)

S=accuracy.mean() * 100



def graph():

    import matplotlib.pyplot as plt

    data=[S]

    alg="Support Vector Machine"

    plt.figure(figsize=(5,5))

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison of Water Quality")

    plt.legend(b,data,fontsize=9)




graph()

TN = cm2[1][0]

FN = cm2[0][0]

TP = cm2[1][1]

FP = cm2[0][1]

print("True Positive :",TP)
print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)




cm2=confusion_matrix(y_test, predicts)

print('Confusion matrix-SVM:')

print(cm2)
```

```
sns.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size": 16}, fmt='.2%',)

plt.show()
```

Flask Deploy:

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import joblib

app = Flask(__name__)

model = joblib.load('dt.pkl')

@app.route('/')

def home():

    return render_template('index.html')



@app.route('/predict',methods=['POST'])

def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    print(final_features)
    prediction = model.predict(final_features)

    output = prediction[0]
    print(output)
    if output == '0':
        output ='water is not safe'
    elif output =='1':
        output = 'Water is safe'
    return render_template('index.html', prediction_text='WATER QUALITY {}'.format(output))
if __name__ == "__main__":
    app.run(host="localhost", port=8012)
```

OUTPUT SCRRENSHOT



27. Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score is will be found out. This application can help to find the Water Quality status.

28. Future Work

- Water Quality prediction with live sensor (IOT) AI model.
- To automate this process by show the prediction result in web application or desktop application.
- To optimize the work to implement in Artificial Intelligence environment.

RE-2022-6632 (3)-plag-report

by Research Experts - Turnitin Report

Submission date: 01-Mar-2022 01:31PM (UTC+0500)
Submission ID: 1773730401
File name: RE-2022-6632_3.docx (679.65K)
Word count: 2117
Character count: 11537

CLASSIFICATION OF QUALITY OF DRINKING WATER USING MACHINE LEARNING TECHNIQUE

5SHWADEEP GHOSH
Computer Science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India
Bishwadeep05ghosh@gmail.com

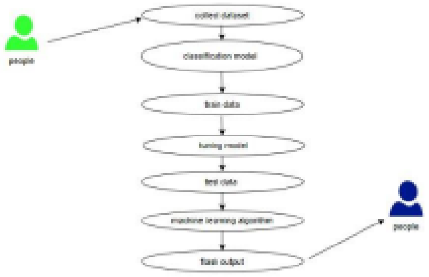
5HARAT KAUSHIK
Computer Science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India
kaushikbharat61@gmail.com

Mrs REFONAA
Department of Computer Science and Engineering,
Chennai, India

Abstract- Water pollution refers to the adding of pollutants into the water bodies that are hazardous to homo sapiens and all the other living organisms. It can be said that it is the most dangerous threat that humankind ever faced. Due to various pollutants the crops, animals, forests are harmed. To prevent this problem in the transportation sector they have to anticipate the water quality from pollutants from machine learning. So, research on the quality of water has become a major field. The motive is to investigate and verify the machine learning methods. Here we will use the supervised machine learning technique to capture several data, for example, univariate analysis, bivariate analysis, the multivariate analysis we analyse the data of the data set, clean the data and prepare the data for better results. Additionally, the data set provided by the transport traffic department will be compared to check the performance of various machine learning algorithms with evaluation classification reports, identity confusion matrix categorizes data from priority, and then compare with the proposed machine learning algorithm for best accuracy and precision. The result is displayed using flask framework to the user where he will get to know the quality of drinking water is good or not and then the water can be processed further.

I INTRODUCTION

Our project aims to build a machine learning model for Water quality. The first process is data collection where previous data of Water Quality measures is collected. Then we will do data mining which is a commonly used technique for the processing of very big datasets in the field of machine learning and deep learning. We think that if bad water quality is found before its consumption can save many lives. Machine learning nowadays is being used in many health care centres which reduces manual effort and can produce a better model which makes it less prone to errors which helps in saving many lives. We have done the proper data analysis on the water quality dataset and found proper variable identification of both dependent and independent variables. The proper machine learning algorithm has also been applied which gave the best results with the best accuracy after trying different algorithms.



Exploratory Data Analysis of Water Quality Prediction:
We have searched multiple datasets from different sources to form a generalized and useful dataset. We have got the sufficient datasets to find out the trends. Then we have applied machine learning algorithms to extract patterns and trends to obtain results with the best accuracy.

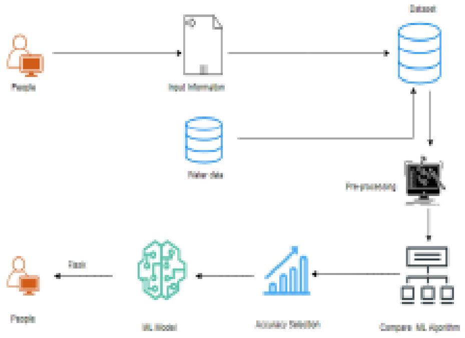
Data Wrangling:
In this part, we will load the data first and then we will clean the data by removing null and unwanted values to make the dataset useful for analysis. In the project, we have documented the cleanliness steps to justify our steps.

Data Collection:
The d5set that we have collected has been split into two parts: A training set and a Test set. We have split the data on a 7:3 ratio. 9e have applied Random Forest, logistics regression, Decision tree, K-Nearest Neighbour (KNN), and Support Vector Machines classifier on our training and testing set.
After applying all algorithms, on the basis of the result of the accuracy of test data, test set prediction has been done.

Building Classification Model:
For predicting our water quality measures, the decision tree algorithm gave the best result it performs better in classification problems. A decision tree is very strong in the pre-processing of the outliers as well. It did not take irrelevant variables into consideration as well. It performed

a well-mixed dataset that is continuous, categorical, and discrete variables. Its errors were comparatively lesser than other algorithms.

Advantages:
This report proves the applicability of machine learning techniques for checking the quality of water. It also tells us some information about future research issues and challenges.



II LITERATURE REVIEW

Title: Predictive Analysis of Water Quality Parameters using Deep Learning
Author: Archana Solanki, Himanshu Agrawal, Kanchan Khare
Year: September 2015

Reservoirs, lakes, and ponds are important water resources. Reservoirs are important for the support of all living organisms. They provide a home to aquatic animals and clean water. Water from such places is used in various places such as industries, farming and water supplies, and recreation and aesthetic value. Water from these resources can be used in drought times and for the production of hydroelectric power. But due to adding of pollutants and the quality of water is compromised. The water quality is deteriorated due to indiscriminate disposal of sewage, anthropogenic activities, industrial waste, and other human activities. Monitoring the quality of water reservoirs is essentials for the conservation of water quality. The quality of water helps in regulating the biodiversity and biomass, energy, and rate of succession. To reduce the effects of pollutants in the water it is essential to assess all of them. So, this study is made to make fairly accurate predictions for variable data. These studies show the comparison between the predictions of supervised learning and unsupervised learning, the result shows the robust behaviour that can be achieved by denoising autoencoder and deep belief network.

Title: Water Quality Monitoring for Disease Prediction using Machine Learning
Author: Prajakta Patil, Sukanya More, Atharv Deshpande.
Year: 2020

Getting pure drinking water is a human right and on 28 July 2010, it was declared as the human right to water and sanitization by UN General Assembly. Water-related disease are the reason for many fatalities across the world more than 3.4 million people die every year due to water bone diseases. Lack of monitoring of the water quality index and the water sources and the inability to anticipate the growth of waterborne diseases are found out to be the reason for deaths. There has been an urgent need for disease prediction in the water bodies the main motive of the study is to imply machine learning techniques to water quality data and use the methods to predict probable water quality diseases. The study involved collecting data samples of some of the water quality parameters by using IoT. The data involving all the required observations were collected from West Bengal Pollution Control Board's Water Quality Information System. On the collected data Gradient Boosting Classifier was trained and tested. There was accuracy of 0.92 and 0.95 on cross-validation and hold-out data respectively. After training with data the predicted diseases were given as alerts using push bullet service.

Title : Water Quality Assessment with Water Quality Indices
Author: Sivaranjani S., Amitava Rakshit and Samrath Singh
Year : 20 july, 2015

The drinking water quality index gives a single number that expresses its quality on the parameters. The water quality index(WQI) is a valuable and unique rating to show the overall condition of the water in a single term that will be very helpful for the processing and treatment of the water for the concerned issues. These variables utilize various biological and physio-chemical parameters and have been resulted in very accurate outcomes in various government agencies.

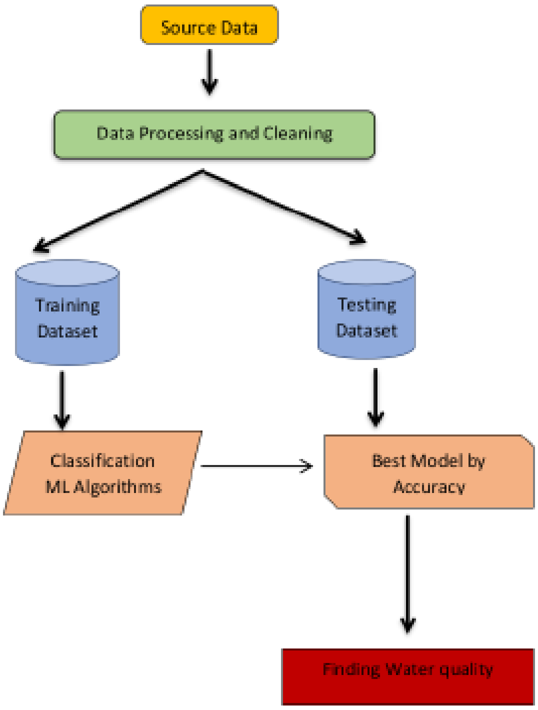
Title : Implementation of Machine Learning Methods for Monitoring and Predicting Water Quality Parameters
Author: Gasim Hayder , Isman Kurniawan , Hauwa Mohammed Mustafa.
Year : 11.09.2020

Need of good and quality water for human consumption can never be underestimated, and the quality of the water is only known after monitoring of the water quality index. Many methods have been applied to the treatment of water and for knowing the water quality index but none of them are so efficient in monitoring water quality parameters. Industries and government agencies use laboratory methods for getting the water quality index, which take costly reagents, and are very time taking and skilled labours to perform the test. so, this makes it mandatory to find a substitute method.

III PROPOSED WORK

The proposed method is to build a machine learning model for checking the drinking water quality index. The process begins from the collection of data set where the past related data of water quality is taken and then data mining is performed for processing the enormous amount of data. If the condition of the water is known before then it can save lives and a lot of resources. Machine learning is now applied in health care sectors to reduce manual effort. The data analysis is done on the whole data set and then proper variable identification is done on both dependent and independent variables. Then the best-suited algorithm is used on the dataset where the pattern of the data is learned.

For our model, we have checked many algorithms like support vector machine, logistic regression, random forest, and decision trees. For our model, the decision tree gave the best accuracy, so we have taken that for our analysis. We have data set containing copper, selenium, fluoride, silver, bacteria, uranium, etc. variables for the detection of water quality. First, we will clean the data using data cleaning and extraction techniques in python. Then we have used data exploration techniques for finding the relationship between the variables and result. Then we have fitted the model using a decision tree. After getting the result and accuracy scores from the decision tree classifier. Now we will show our result using a flask. Flask is a web app that uses python and HTML for showing the user interface of the result. In flask we have to provide the values of variables, then click on predict, then the flask will show the result with the given user interface.



IV RESULT

The proposed method is to but a prediction model for drinking water quality to overcome the issue of using polluted water from resources. The data set is first pre-processed and the columns are analysed too see dependent and independent variables. And then we used four different machine learning algorithms to predict the result and we found decision tree algorithm as the best accuracy providing model. Multiple dataset from various sources are combined to train the ML model. Here you will put all the chemicals amount in water and then it will check the amount in the water and predict the result accordingly.

V CONCLUSION AND FUTURE WORK

- Live sensors (IoT) will be used for water quality prediction using an AI model
- To completely automate the process using IOT and show the predictions and problems on web application
- or any desktop application and integrate it with smartphones too.
- We will work on optimization and better efficiency by making an Artificial Intelligent environment.

The analytical process started from cleaning and processing the data,
finding the best-suited algorithm which gives the best accuracy on evaluation
and gives a better accuracy score. This model will predict the water quality of the drinking water.

VI REFRENCES

[1] D. Kriplani. etal., "Keeping the basin full Smart water for the 21st century," TCS.

[2] G. M. Carr. etal.,, "Water Quality for Ecosystem and Human Health,"UNESCO, Ontario, Canada, 2008 Fröhlich, B. and Plate, J. 2000.

[3] M. Ali. etal.,, "Data Analysis, Quality Indexing and Prediction of Water Quality for the Management of Rawal Watershed in Pakistan," in Islamabad, Pakistan, 2008.

[4] Poor water quality, a serious threat, [Online]. Available at: http://www.deccanherald.com/content/63740/poorwater-quality-serious-threat.html.

[5] Predictive analysis World, [Online]. Available at: http://predictiveanalyticsworld.com/predictiveanalytics.p hp.

[6] Y. Papadimitris. etal.,, "Integrated approach of lake quality monitoring", 2005.

[7] "Planning of water quality monitoring systems Technical document UNEP/GEMS Water", 2012.

[8] "Status of water quality in India -2010, Central pollution control board, Ministry of Environment and forests", 2010.

[9] P. Tirkey. etal.,, "Water quality indices- important tools for water quality assessment,"IJAC, Vol. 1, pp. 15-28, 2013.

[10] P. Chaudhry. etal.,, "Water Quality Assessment of Sukhna Lake of Chandigarh City of India; Hydro Nepal," Vol. 12, pp. 26-31, 2013.

[11] S. Malek. etal.,, "Dissolved Oxygen Prediction Using Support Vector Machine," Vol. 8, pp.153-160, 2014.

[12] http://www.wbpcb.gov.in/

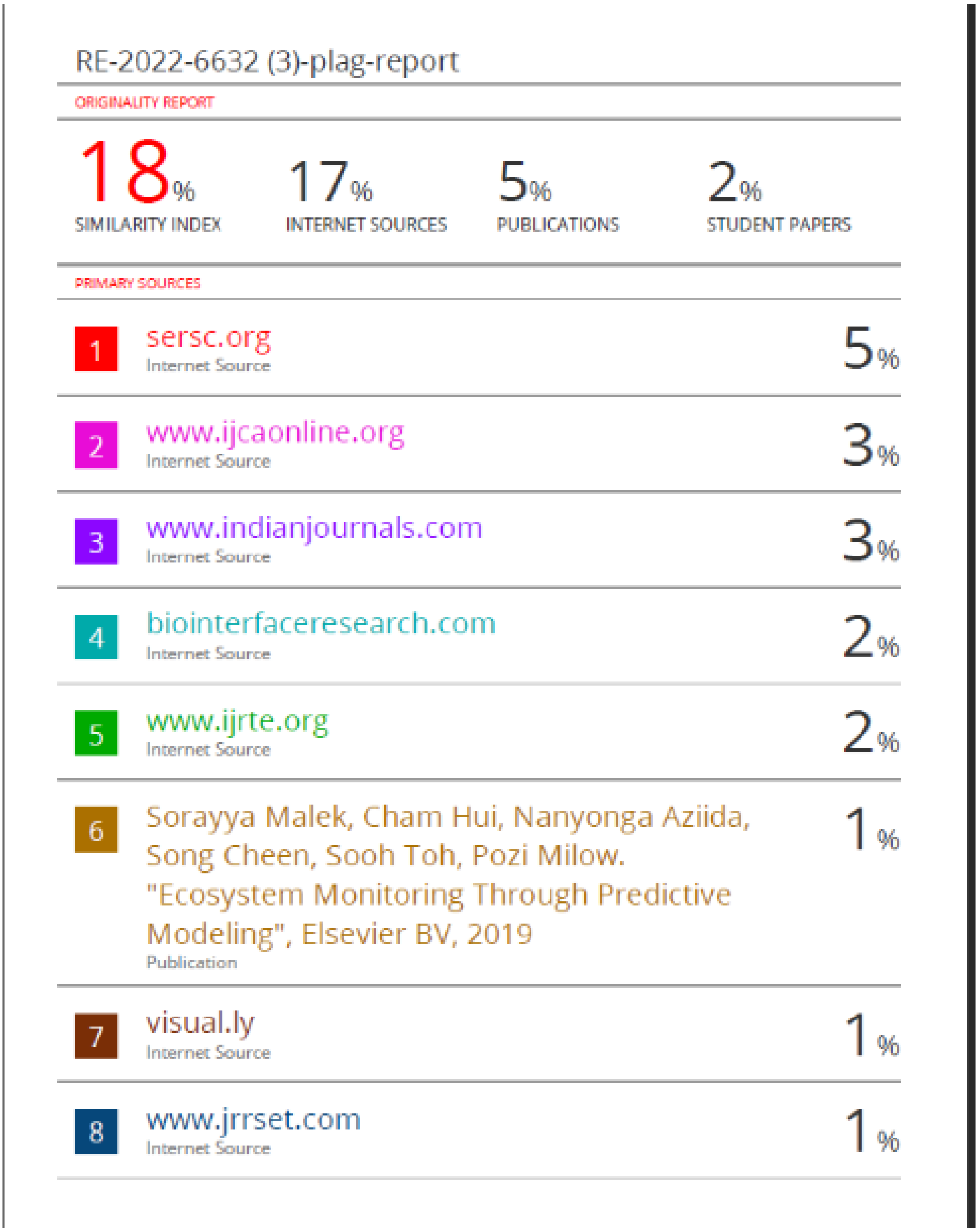
13. Nashwan, M.S.; Ismail, T.; Ahmed, K. Flood susceptibility assessment in Kelantan river basin using copula. *Int. J. Eng. Technol.* **2018**, *7*, 584–590, https://doi.org/10.14419/ijet.v7i2.8876.

14. Hee, Y.Y.; Suratman, S.; Aziz, A.A. Water Quality and Heavy Metals Distribution in Surface Water of the Kelantan River Basin (Malaysia). *Orient. J. Chem.* **2019**, *35*, 1254–1264, https://doi.org/10.13005/ojc/350402.

15. Yen, T..; Rohasliney, H. Status of Water Quality Subject to Sand Mining in the Kelantan River, Kelantatan. *Trop. Life Sci. Res.* **2013**, *24*, 19–34.

16. Tan, M.L.; Ibrahim, L.; Cracknell, P.; Yusop, Z. Changes in precipitation extremes over the Kelantan River Basin , Malaysia. **2016**, https://doi.org/10.1002/joc.4952.

17. Haykin, S. Neural networks: a comprehensive foundation by Simon Haykin. *Knowl. Eng. Rev.* **1999**, *13*, 409–412.



RE-2022-6632 (3)-plag-report

ORIGINALITY REPORT

18%	17%	5%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	sersc.org Internet Source	5%
2	www.ijcaonline.org Internet Source	3%
3	www.indianjournals.com Internet Source	3%
4	biointerfaceresearch.com Internet Source	2%
5	www.ijrte.org Internet Source	2%
6	Sorayya Malek, Cham Hui, Nanyonga Aziida, Song Cheen, Sooh Toh, Pozi Milow. "Ecosystem Monitoring Through Predictive Modeling", Elsevier BV, 2019 Publication	1%
7	visual.ly Internet Source	1%
8	www.jrrset.com Internet Source	1%

9	jultika.oulu.fi Internet Source	<1 %
10	doctorpenguin.com Internet Source	<1 %
11	D. Venkata Vara Prasad, P. Senthil Kumar, Lokeswari Y. Venkataramana, G. Prasannamedha et al. "Automating water quality analysis using ML and auto ML techniques", Environmental Research, 2021 Publication	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

THE END .