

COASTAL FLOOD PREDICTION USING MACHINE LEARNING MODEL ANN

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering
degree in Computer Science and Engineering by

Pavuluri Srikanth (Reg No: - 38110555)

Chukka Sudhakar Babu (Reg No: - 38110567)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL OF
COMPUTING

MARCH 2022



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai 600119

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 **Pavuluri Srikanth (38110555), Chukka Sudhakar Babu (38110567)** who carried out the project entitled **"COASTAL FLOOD PREDICTION USING MACHINE LEARNING MODEL ANN"** under my supervision from November 2021 to March 2022.

Internal Guide

Dr. Veena K, Assistant Professor

Head of the Department

Dr. S. Vigneswari, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We **Pavuluri Srikanth, Chukka Sudhakar Babu** hereby declare that the Project Report entitled “**Coastal Flood Prediction Using Machine Learning Model ANN**” is done by me under the guidance of Dr. Veena K, Assistant Professor, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the 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. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me 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. Veena K, Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved the 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.

COASTAL FLOOD PREDICTION USING ML MODEL ANN

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
01	ABSTRACT	11
02	EXISTING SYSTEM 2.1 DRAWBACKS	12
03	INTRODUCTION 3.1 DATA SCIENCE 3.2 ARTIFICIAL INTELLIGENCE	13
04	MACHINE LEARNING	14
05	PREPARING DATASET	17
06	PROPOSED SYSTEM 6.1 ADVANTAGES	19
07	LITERATURE SURVEY	21
08	SYSTEM STUDY 8.1 OBJECTIVES 8.2 PROJECT GOAL 8.3 SCOPE OF THE PROJECT	26
09	FEASIBILITY STUDY	28
10	LIST OF MODULES	29
11	PROJECT REQUIREMENTS 11.1 FUNCTIONAL REQUIREMENTS 11.2 NON-FUNCTIONAL REQUIREMENTS	30
12	ENVIRONMENT REQUIREMENT	31
13	SOFTWARE DESCRIPTION 13.1 ANACONDA NAVIGATOR 13.2 JUPYTER NOTEBOOK	31

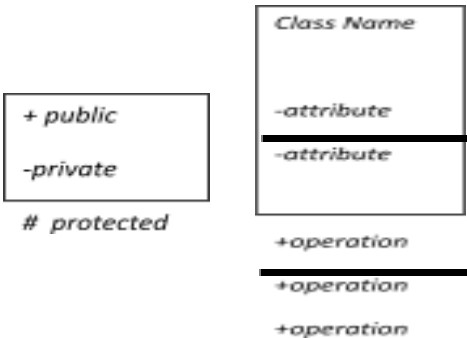
14	PYTHON	39
15	SYSTEM ARCHITECTURE	48
16	WORKFLOW DIAGRAM	49
17	USE CASE DIAGRAM	51
18	CLASS DIAGRAM	52
19	ACTIVITY DIAGRAM	53
20	SEQUENCE DIAGRAM	53
21	ER – DIAGRAM	54
22	COLLABORATION DIAGRAM	54
23	MODULE DESCRIPTION 23.1 MODULE DIAGRAM 23.2 MODULE GIVEN INPUT EXPECTED OUTPUT	55
24	DEPLOYMENT	74
25	HTML	81
26	CSS	85
27	CODING	89
28	CONCLUSION	115
29	FUTURE WORK	116

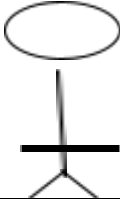
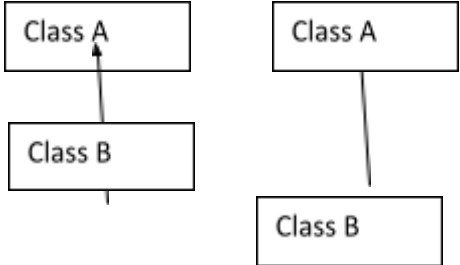
LIST OF FIGURES

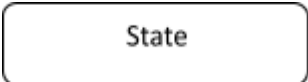
SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	48
02	WORKFLOW DIAGRAM	49
03	USE CASE DIAGRAM	51
04	CLASS DIAGRAM	52

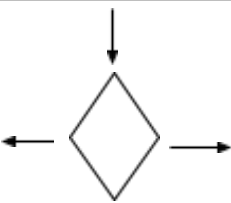
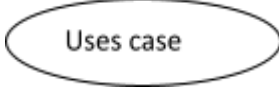
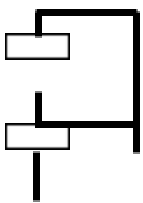
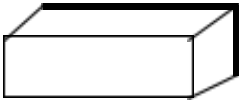
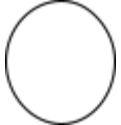
05	ACTIVITY DIAGRAM	53
06	SEQUENCE DIAGRAM	54
07	ER – DIAGRAM	55
08	MODULE DIAGRAM	56

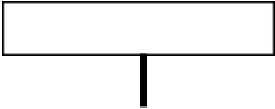
LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	 The diagram shows two examples of a class box. The first box is divided into three horizontal compartments: the top for the class name, the middle for attributes (with '+' for public, '-' for private, and '#' for protected), and the bottom for operations. The second box is similar but only shows attributes and operations. A solid line separates attributes from operations, and a dashed line separates the class name from the attributes.	Represents a collection of similar entities grouped together.
		 The diagram shows two class boxes labeled 'Class A' and 'Class B' connected by a solid line, representing an association between the two classes.	

2.	Association		Associations represent static relationships between classes. Roles represent the way the two classes see each other.
3.	Actor		It aggregates several classes into a single class.
4.	Aggregation		Interaction between the system and external environment

5.	<i>Relation</i> (uses)	<i>uses</i>	Used for additional process communication.
6.	Relation (extends)	EXTENDS →	An Extended 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 process.

9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flows between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Use case		Interaction between the system and external environment.
14.	Component		Represents physical modules which are 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 action.
17.	External entity		Represents external entities such as keyboards,sensors,etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communicates with.
20.	Message	Message 	Represents the message exchanged.

COASTAL FLOOD PREDICTION USING ML MODEL ANN

1. Abstract:

The most dangerous and destructive natural disasters are floods. The floods will cause damage to not only the surroundings, they will affect human daily life too. The flood prediction models will reduce the risk management, loss of human life and these predictive models will help to reduce the financial property damages caused by floods. To reduce the manual mathematical calculations which are followed

throughout the past few years by hydrologists and meteorological departments by introducing and using these neural network methods the decisions and prediction on floods would be precisely known and the cost and time would be saved. By using the neural network techniques which are used to work on the rainfall data set and meteorological data hub this problem can be prevented. For the purpose of data cleaning, identifying the outliers, unwanted data and for clearing the noisy data the data analysis has to be done by a Multilayer perception Classifier (MLP). This has to be done for all the dataset collected. The performance has to know if the flood can happen or not by evaluating and identifying the confusion matrix and the result will be displayed or shown in a flask-based application.

KEY WORDS: Multilayer Perception Classifier (MLP), Artificial Neural Network (ANN), Support Vector Machine (SVM), Terrain Analysis Using Digital Elevation Models (TAUDEMS), Support Vector Classifier (SVC)

2. EXISTING SYSTEM

The early analysis of current evolving and advancing alarm floods examine systems in industry are very important to provide precise information and a safe operation. It will provide instant decision support instead of waiting for the alarm to end. An approach has been proposed to solve the problem with the unlabeled historical data. A vector representation is called an Exponentially Attenuated Component to show or represent the alarm floods, to prioritize the information of prior activated alarms. A semi-supervised approach based on GMM was proposed to communicate the problem of prior classification of currently evolving alarm floods with unlabeled historical data. The exponentially attenuated component (EAC) was used to change the alarm flood sections to feature vectors, to reduce the computational complexity occurring when data is mined.

2.1 Drawbacks

They did not mention any accuracy

It can't thereby better determine the regularity of rainfall data and achieve more accurate prediction results of flash floods.

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.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

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 subfields 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.

3. 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 the use of specialized algorithms. It feeds the training data to an algorithm, and the algorithm uses this training data to give predictions on new test data.

Machine learning can be roughly separated into 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 has no labels. It provided 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 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 observations. 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, biometric identification, document classification etc.



Fig 1.1 Process of Machine learning

Supervised Machine Learning is the majority of practical machine learning that uses supervised learning. Supervised learning is where you 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 or categorical for classification. A classification model attempts to draw some conclusions 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”.

3.Preparing the Dataset:

This dataset contains 641 records. The data has been collected from the sources and the data was split into two sets one is for training and another is testing. The data has been split into a 7:3 ratio. The messy and complex data has been removed or cleansed, converting or mapping the data from one format to another. The data collected may contain the raw data or the data may be in any form or the data may contain the missing values which will lead to inconsistency. To obtain better and best results the preprocessing of data is needed so it will improve the efficiency of the algorithm. The variable conversion and outliers are preprocessed. The process has started from collecting the data and it has to be cleaned to split into a data set where the data has been explored. Using the algorithms, an effective and accurate approach has to be chosen to predict the flash floods. Using the MLP classifier, the accuracy of data is decided on train data and from this the test data is predicted. , which were classified into 2 classes:

- Flood Happen
- Flood Not Happen

4.PROPOSED SYSTEM

Floods are one of the biggest natural disasters causing many lives as well as damages. Different types of floods like river flood, urban flood, coastal flood, and flash flood have

been observed over the years. A flash flood is a direct response to a rainfall having very high intensity in a short time. This kind of flood is seen typically in urban areas where the underlying ground cannot cope, or drain excess water away fast enough via the sewage system and drainage canals in a short amount of time. In recent years, we have seen the impact of floods in cities such as Mumbai, Chennai, Ahmedabad and Kerala. Poor urban planning, inaccurate and delayed forecasting and inadequate flood mitigation systems are the main reasons behind it. Given a dataset from different sources would be to form a generalized dataset, and applied to extract patterns and to obtain results with maximum accuracy. This section of the report will load in the data, check for cleanliness, and then trim and clean the given dataset for analysis. Make sure that the document steps carefully and justify cleaning decisions. 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 the Multi-Layer *Perceptron Classifier* will be applied to the Training set and based on the test result accuracy, Test set prediction is done.

Among the natural disasters, floods are the most destructive, causing massive damage to human life, infrastructure, agriculture, and the socioeconomic system. Governments, therefore, are under pressure to develop reliable and accurate maps of flood risk areas and further plan for sustainable flood risk management focusing on prevention, protection, and preparedness. Flood prediction models are of significant importance for hazard assessment and extreme event management. Robust and accurate prediction contributes highly to water resource management strategies, policy suggestions and analysis, and further evacuation modeling. Thus, the importance of advanced systems for short-term and long-term prediction for flood and other hydrological events is strongly emphasized to alleviate damage. However, the prediction of flood lead time and occurrence location is fundamentally complex due to the dynamic nature of climate conditions. The vast majority of ANN models for flood prediction are often trained with a BPNN. While BPNNs are today widely used in this realm, the MLP, an advanced representation of ANNs, recently gained popularity. The MLP is a class of FFNN which utilizes the supervised learning of BP for training the network of interconnected nodes of multiple layers. Simplicity, nonlinear activation, and a high number of layers are characteristics of the MLP. Due to these characteristics, the model was widely used in flood prediction and other complex hydrogeological models. In an assessment of ANN classes used in flood modeling, MLP models were reported to be more efficient with better generalization ability. Nevertheless, the MLP is generally found to be more difficult to optimize.

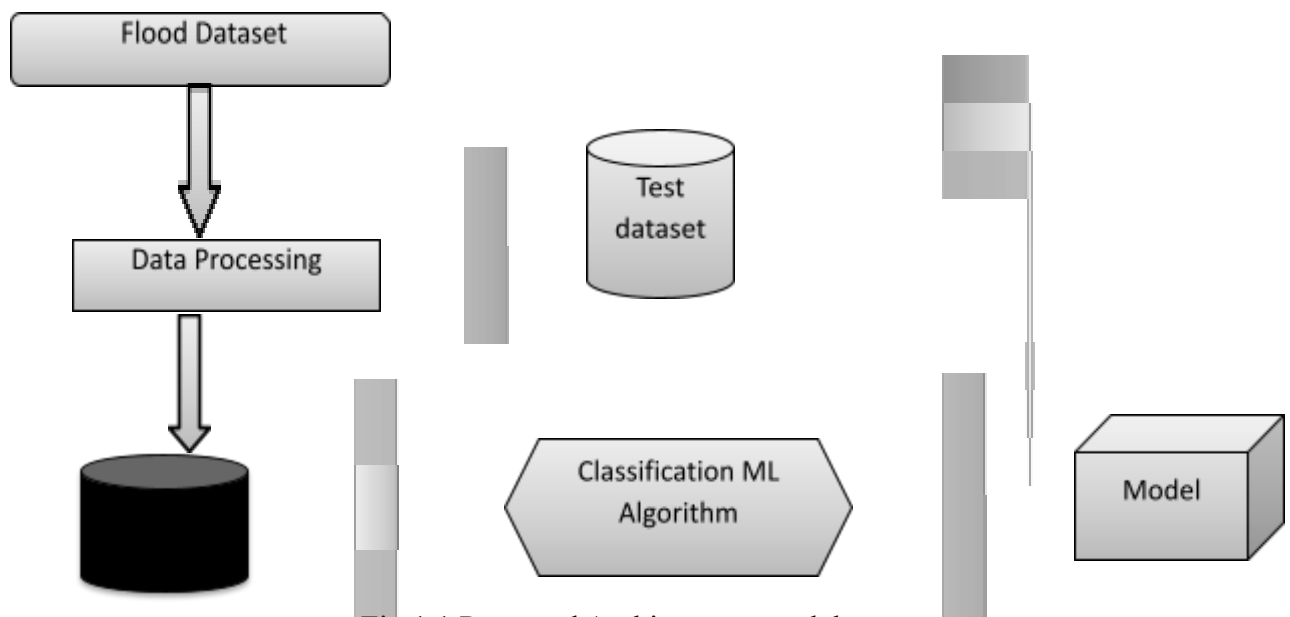


Fig 1.1 Proposed Architecture model

6.1 Advantages:

These reports are to the investigation of applicability of machine learning techniques for Flood prediction.

Finally, it highlights some observations on future research issues, challenges, and needs.

ML is a field of artificial intelligence (AI) used to induce regularities and patterns, providing easier implementation with low computation cost, as well as fast training, validation, testing, and evaluation, with high performance compared to physical models, and relatively less complexity.

3. 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. They are 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 reorganization, 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 standpoint. 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 surveys to understand the past and present perspective of loan approval or not.

A Hybrid Machine Learning Approach for Classifying Aerial Images of Flood-Hit Areas. The classification is done on an image dataset which consists of two subgroups one is aerial images with flood-affected areas and other is aerial images without any flooded affected areas. The downloaded dataset is randomly partitioned then the classification of images is done. This classification involves hybridization of the SVM classifier along with k-means clustering. The SVM classifier is used to train various categories of the dataset. Comparative studies show that SVM classifier is one of the preferable classifiers among all the others that are widely used in many remote sensing applications and provides a much better accuracy when compared to other algorithms like decision trees. It was then used for solving multi-class classification problems. It builds a hyperplane that separates both the categories and is used for categorizing new images. Classifier is evaluated based on various parameters; some of them being accuracy, prediction speed and training time. The confusion matrix is built to represent the performance of the classifier. This matrix represents the number of correctly and incorrectly classified examples from the dataset. The performance of this methodology is evaluated. It is capable of classifying the flooded areas with good accuracy. This model classified 90% of flooded images correctly. A comparison is done with the accuracy of the model by training it with different environments and the results had been

plotted. Kernel functions are also changed for the SVM classifier to plot the difference in the prediction and training time.

Smart flood disaster prediction system using IOT & Neural Networks. The integrated approach combines the scalability of IOT and reliability of artificial neural networks to handle data provided by a sensor network and by effective communication between these two components, an early prediction of flood is done. It has been proven that the LevenbergMarquardt training algorithm with the NARX network gives better results and provides real-time flood prediction with step-ahead alert. The prediction chart indicates that the event of flood is predicted one time-step ahead and warns the communities at risk using Thing Tweet in simple language with the use of the Internet. Thus by harnessing latest technological disruptions such as IOT and machine learning, big data, predictive analytics along with social media and mobility allows effective emergency & disaster management for smart nations. The main aim of this system is to monitor humidity, temperature, pressure, rainfall, river water level and to find their temporal correlative information for flood prediction analysis. The IOT approach is deployed for data collection from the sensors and communication over Wi-Fi and an ANN approach is used for analysis of data in flood prediction.

Flood Prediction Using Multi-Layer Artificial Neural Network in a Monitoring System with Rain Gauge, Water Level, Soil Moisture Sensors. This study was about to implement a real-time monitoring system capable of measuring parameters such as rainfall intensity, soil moisture, water level and rate of increase in water level. The different sensors were integrated into a system, wherein the data was logged and stored. A prediction model based on a multi-layered artificial neural network was developed and was tested using an actual setup. This study validated the response of the multi-layered network model. The flood prediction model had a small deviation with reference to the actual water level, with RMSD of 2.2648. In the Philippines that was a major issue as it led to damage of properties, damage to infrastructures or even loss of lives. Current systems adhere to solve issues to prevent devastating disasters caused by floods. Multi-layered artificial neural networks with the aid of MATLAB were used in the development of the prediction model. In the training, test, validation and overall dataset, the network showed a very good goodness-of-fit specifically 0.99889 for the training dataset, 0.99362 for the test data set, 0.99764 for the validation dataset and 0.99795 considering all the data in the dataset. The network was then programmed and integrated into the system in the actual setup. The model is validated by

running trials with certain inputs and predicted flood levels as the output and is compared to the actual flood level after a certain period of time. The flood prediction system showed an RMSD value of 2.2648 which signifies a small overall difference between the predicted flood level and actual flood level across the whole dataset tested in the actual setup.

Flood Loss Prediction of Coastal Cities Based on GM-ANN. AHP will be used to determine the weight of flood factors. Considering the characteristics of different factors, GM is applied to get predictive values of flood factors. The AHP-GM-ANN model was constructed to predict flood loss in coastal cities. AHP is used to determine the weight; GM (1, 4) and GM (1, 1) are adopted to predict flood factors of different characteristics. The ANN model is combined to predict flood loss in Shenzhen. From these predictive results, compared with the simple GM model, AHP-GM-ANN model would significantly reduce the predictive error while reliable results are obtained. The AHP-GM-ANN model not only solved nonlinear variable relationship problems in the predictive method, but also improved the accuracy of the system of quantitative prediction methods. For each step of establishment in the model, a corresponding error analysis was made which enhanced the correctness and scientific security of the predictive method system.

Floods Prediction Using Radial Basis Function (RBF) Based on the Internet of Things (IoT). Artificial Neural Networks had been trained on the data that are in the form of water level data and rainfall data which is used to predict the water level and daily rainfall for the next month. The parameters used to get the smallest error rate in the prediction process of water level and rainfall with the best radial basis function in a neural network are to use 700 times iterations and use the learning rate equal to 0.00007. Here the flood was predicted using Radial Basis Function. The data was received from Citarum River Hall. The result from the Radial Basis Function Neural Network is sent to an android application that displays the opportunity of flooding. Using epoch as much as 700 giving error value of TMA equal to 0.027 and error value of CH equal to 0.002, a learning rate of 0.00007 giving error value of TMA equal to 0.286 and error value CH equal to 0.002, and a hidden neuron of 2 giving error value of TMA equal to 0.6483 and error value of CH equal to 15.999 can be used to predict the flooding.

An Efficient Automated Hybrid Algorithm to Predict Floods in a Cloud Environment. Data mining classification algorithms have been implemented on flood data sets in Odisha. The results had been evaluated on a standalone machine which shows that GA-SVM yields the

highest accuracy of 86.36% and enhances the performance on various evaluation criteria. These models are also validated on cloud environments which proved to be more precise in comparison to the standalone machine. In a cloud environment, GA-SVM yields an accuracy of 88.57% with an AUC value of 0.816. The execution time in the Cloud platform shows the better result which is 0.11ms as compared to 6.22ms using the standalone machine. Therefore, it can be clearly validated that the hybrid approach outperforms in terms of accuracy as well as execution time on Cloud environments. This approach is expected to reduce the number of deaths and other losses caused by floods by giving prior information about flood occurrences. Different algorithms like Artificial Bee Colony, Particle Swarm Optimization, FP Growth can be combined with this to make better decisions and avoid flood prone situations. Optimization based feature selection Genetic Algorithm (GA) has been combined with classification algorithms to predict the occurrence of floods. The experimental results show that the GA-SVM algorithm outperforms in terms of accuracy and total execution time in comparison to other hybrid algorithms.

Multiple Input Single Output (MISO) ARX and ARMAX Model of Flood Prediction System: Case Study Pahang. It has proposed a design of 7 hours flood prediction models using MISO ARX and ARMAX structure and compared the prediction performances. Results showed that the ARMAX model has a higher Best Fit percentage value and smaller rmse value. This means the ARMAX model has better prediction performance. An accurate flood water level prediction model is required as an alarm system that would warn the affected area and residents to prepare for evacuation due to the upcoming severe flood. This paper compared the prediction performances of a developed flood prediction models that were designed using Multiple-Input Single-Output (MISO) Autoregressive with Exogenous Input (ARX) and MISO Autoregressive Moving Average with Exogenous Input (ARMAX) structure. The models were designed using the Matlab System Identification toolbox of parametric models. The location for the case of study was at Pahang River, Temerloh, Pahang with four upstream stations and one downstream station or observed location. The data used were obtained from the Malaysian Department of Drainage and Irrigation. Simulation results showed that the prediction performance of flood prediction models designed by ARMAX structure showed better Best Fit value and smaller rmse values as compared to the model designed using ARX structure.

Context-based Hydrology Time Series Data for A Flood Prediction Model Using LSTM. Semarang is a city in Indonesia that is often hit by floods. LSTM is general and effective at

handling complex long-term temporal dependencies and one of the deep learning methods which is utilized as an algorithm for predicting floods based on precipitation and water levels time series data in two points, upstream and downstream. By using the LSTM algorithm, the model yielded the MAPE value of 3.6%. This means that the error percentage generated in the model to predict the water level in downstream rivers is 3.6% compared to the actual data used as testing. When compared to the multiple regression linear, LSTM resulted in more accurate water elevation level prediction. Flooding in the downstream area is caused by high rainfall and water flowing from the dam when it is unable to hold water. The height of the water in the dam and the river downstream needs to be monitored. Context awareness provides information to the user about what will happen by processing the data and makes predictions about the events that will occur in real time

4. SYSTEM STUDY

8.1 Objective

Floods may only last a few hours and can cause considerable damage and a possible threat to life. Flood prediction requires quantitative knowledge about infiltration and runoff dynamics, which is generally gained at the local scale. When scaling up local measurements to the catchment scale, account needs to be taken off the catchment's organization (connectivity and patchiness).

For this purpose, we developed a new method to map soil vulnerability to floods based on two steps: (1) identification of the flow processes at the plot scale, and (2) up-scaling this knowledge to the catchment scale. Excess surface runoff was scaled up by means of terrain analysis using digital elevation models (TauDEMs) calibrated with in situ sprinkling experiments of three rainfall-simulation intensities carried out on 57 plots under grassland and forest that dominate in the investigated area.

The marked differences in textural and structural porosities between forest and grassland plots appear to control runoff processes. On the one hand, forest soil has a

higher storage capacity than grassland soil, probably caused by a high unsaturated hydraulic conductivity and root water uptake, and resulting in lower surface runoff.

On the other hand, fine material in the topmost 10 cm of grassland soil helps to build a structure that impedes vertical downward percolation and thus enhances surface runoff. However, within each soil category, slope plays an important role in generating surface runoff.

In addition, raising the rainfall-simulation intensity from 24 to 48 mm h⁻¹ increases the risk of predisposition to surface runoff from middle to high in major parts of the catchment under grassland, whereas forest soils showed vertical percolation in all cases except on slopes steeper than 31.3 degrees.

Scaling up runoff processes using TauDEM based on sprinkling experiments provided new quantitative insights into flow processes and enabled us to trace the hydrological connectivity between zones of various predispositions to excess surface runoff under different land uses.

These promising results indicate that the approach is suited for mapping soil vulnerability to floods under varying land use and climate at any scale. Our study showed that the peak discharge can be significantly reduced if the succession and connectivity of land use are well planned.

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

□ **Exploratory data analysis of bi-variate and multivariate**

- 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 algorithms to predict the result**

- Based on the best accuracy

8.3 Scope of the Project

During a flood event, the responsible authorities need to make decisions on the operation of barriers and on evacuation/rescue strategies. Temporary flood protection is sometimes an option and if so it must be decided on quickly. Also, people are served by knowing evacuation routes and a forecast of blocked roads. A system can assist the user by quickly showing the available information on water levels, objects at risk, evacuation routes, resident vulnerability, etc. This support can be given during the preparation of flood event management or during the actual flood event.

6.Feasibility study:

Data Wrangling

This section of the report will load in the data, check for cleanliness, and then trim and clean the given dataset for analysis. Make sure that the document steps carefully and justify the 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 and Support vector classifier (SVC) are applied to the Training set and based on the test result accuracy, Test set prediction is done.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data needs to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion needs to be done.

Building the classification model

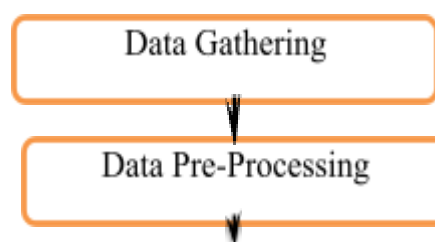
The prediction of Flood, A Random Forest Algorithm prediction model is effective because of the following reasons: It provides better results in classification problems.

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 and has a lot of past data. Data gathering has sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to preprocess what kind of algorithm with the model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved being tuned from time to time with improving accuracy.



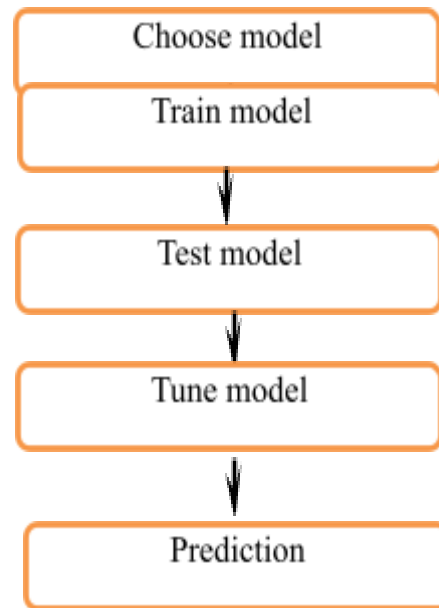


Fig 1.3 Process of data flow diagram

10.List Of Modules:

- Data Pre-processing
- Data Analysis of Visualization
- Comparing algorithms with prediction in the form of best accuracy result
- Deployment Using Flask

11.Project Requirements

General:

Requirements are the basic constraints 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

11.1 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 follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

11.2 Non-Functional Requirements:

Process of functional steps,

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

12.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, 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 the Python package 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. Navigators 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)

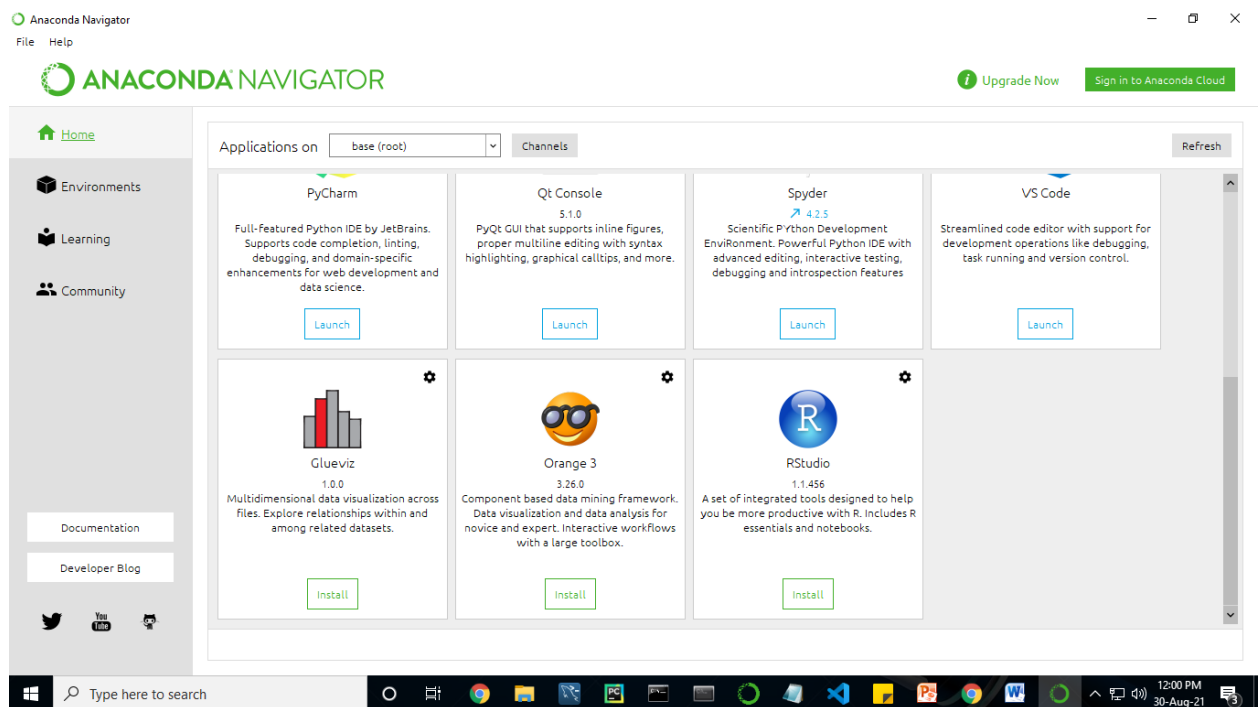
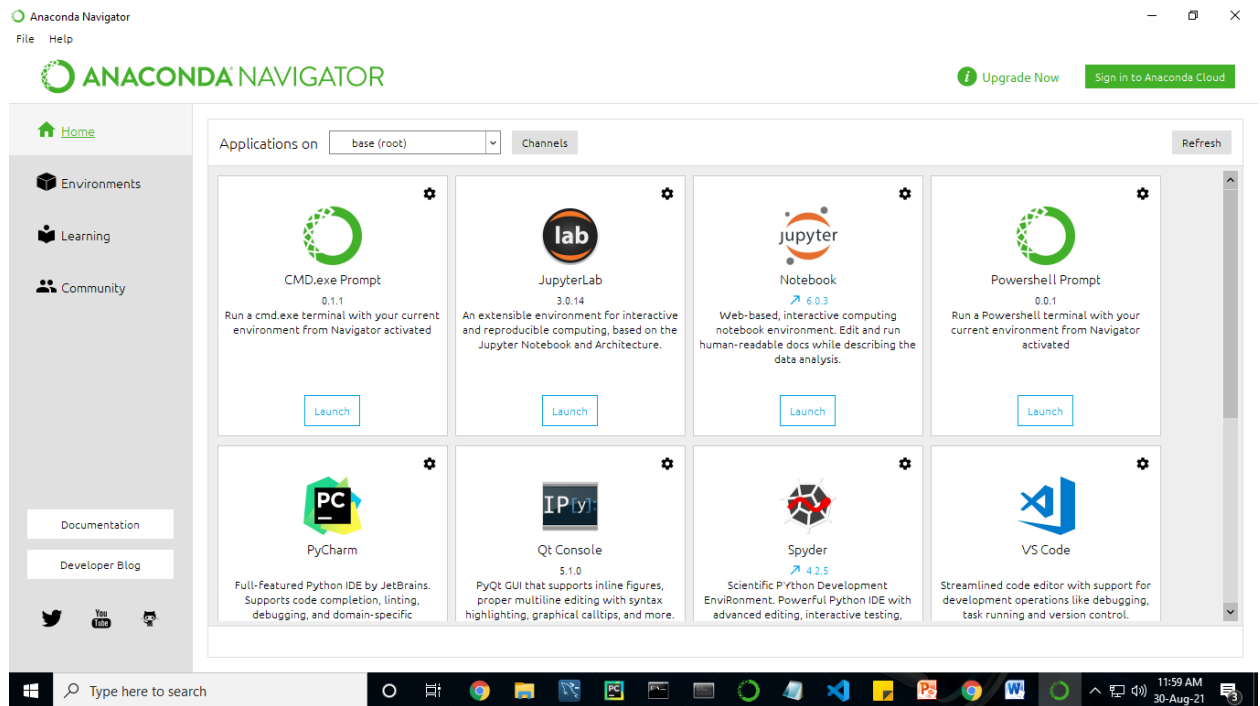


Fig 2.1 Screenshots of Anaconda Navigator

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. The 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 a 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 a 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 an 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 Perez.

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

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.

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:

- 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 exception clauses; it also ensures that clean-up code in a final 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 coroutines.
- 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 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 coroutine-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 numeric by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals ()` method.) Python's 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 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 to `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 `"spam eggs"`. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"22"`.
- Python has various kinds of string literals:
 - Strings are 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.

15. System Architecture

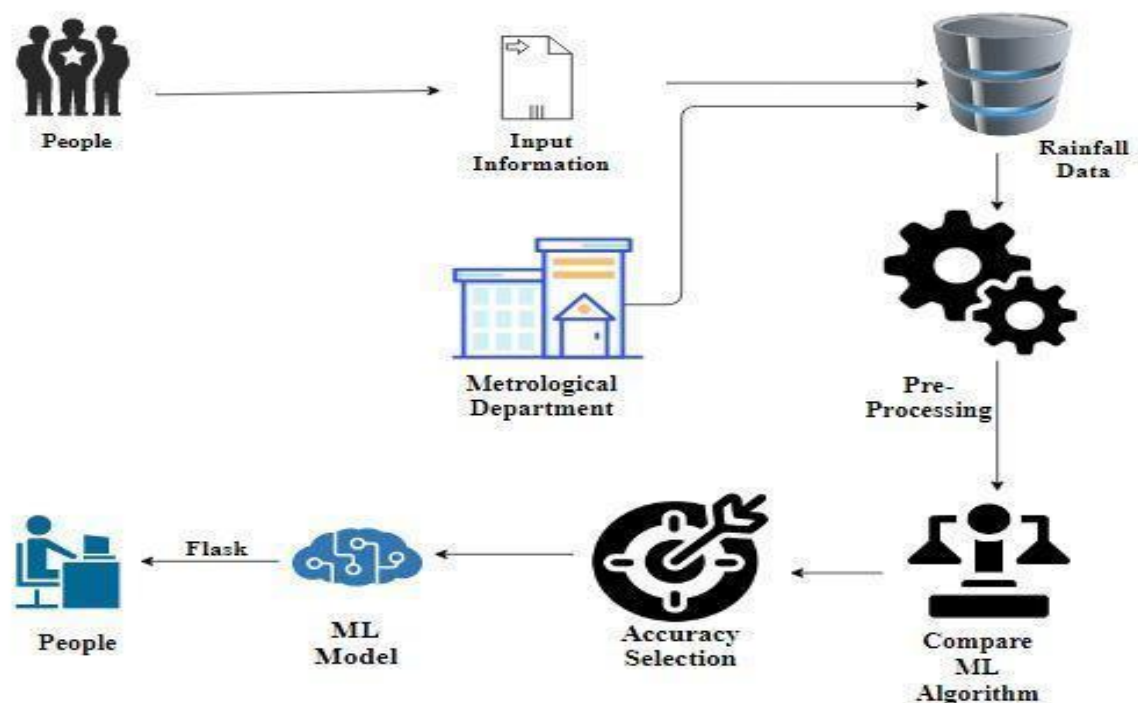


Fig 1.4 System architecture diagram of prediction model

16. Workflow diagram

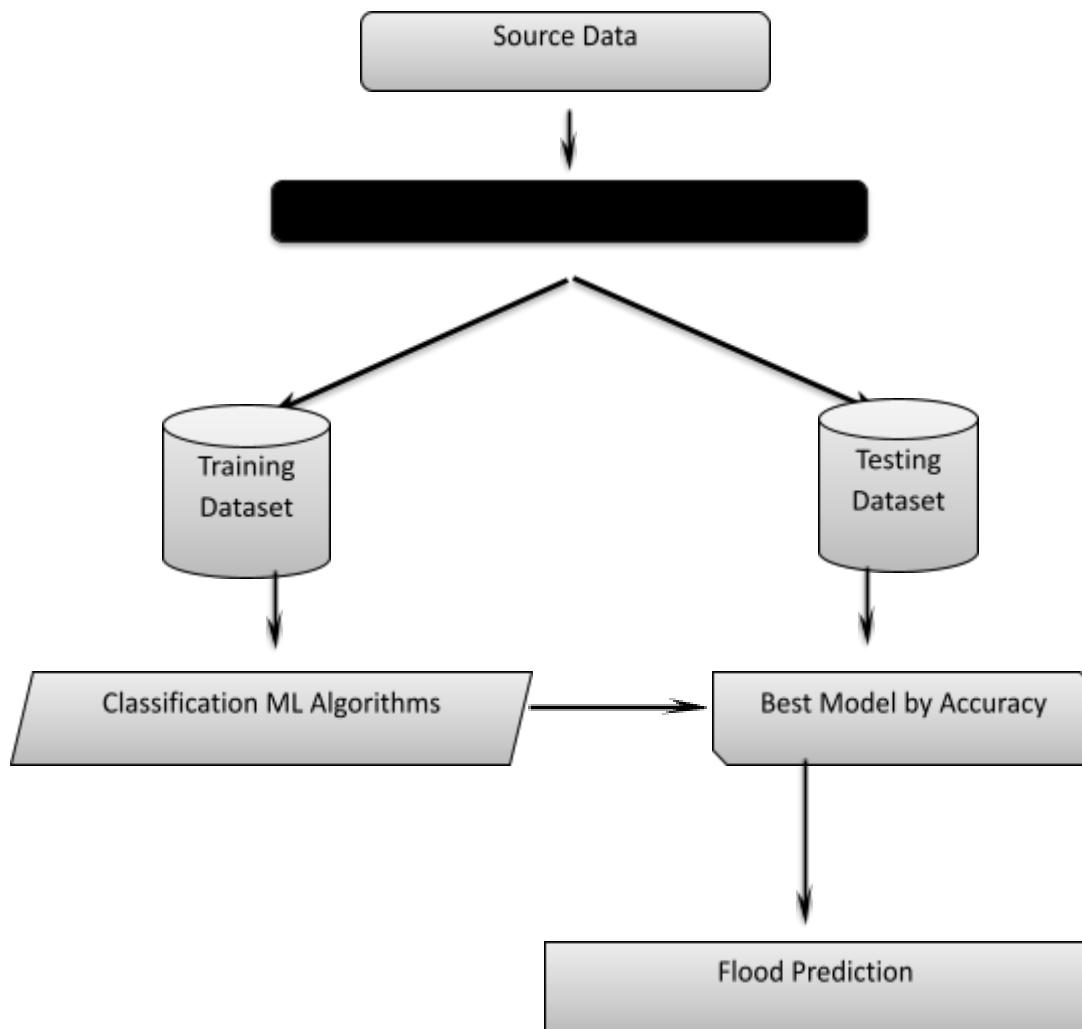


Fig 1.5 Workflow Diagram of prediction model for training and testing dataset

17. Use Case Diagram

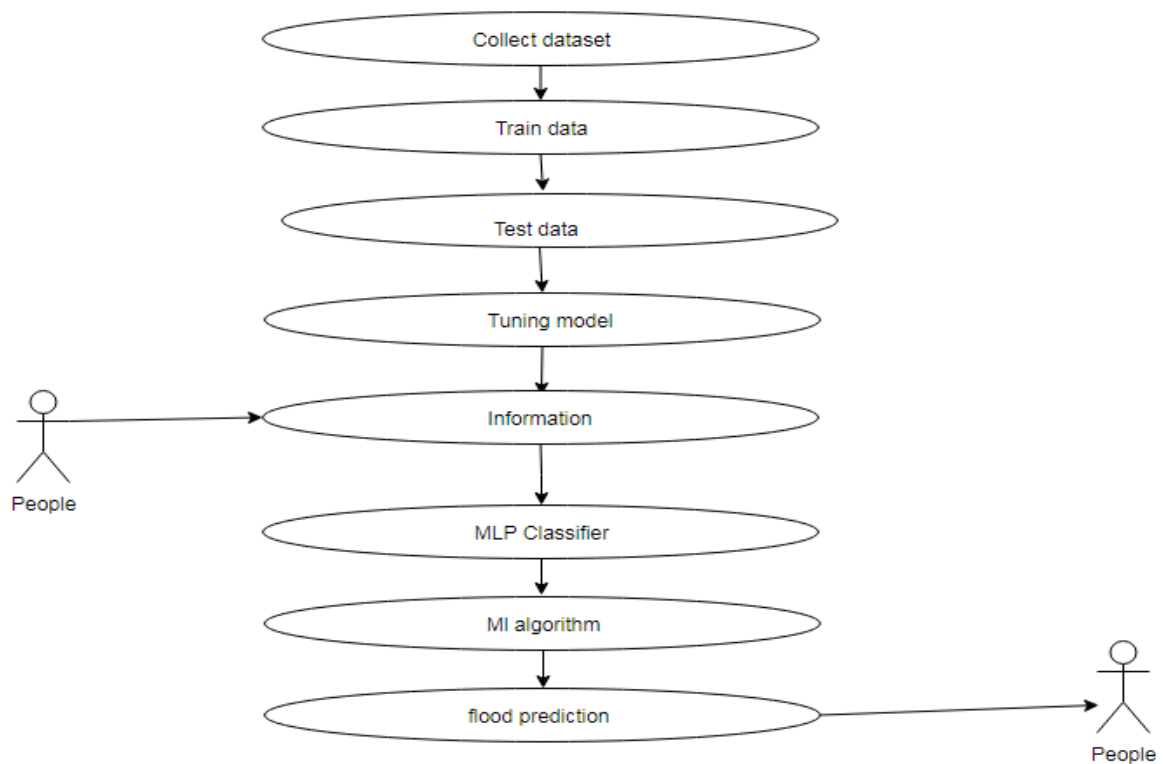


Fig 3.1 Use case diagram of the model

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 be said that use cases are nothing but system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors (people).

18. Class Diagram:

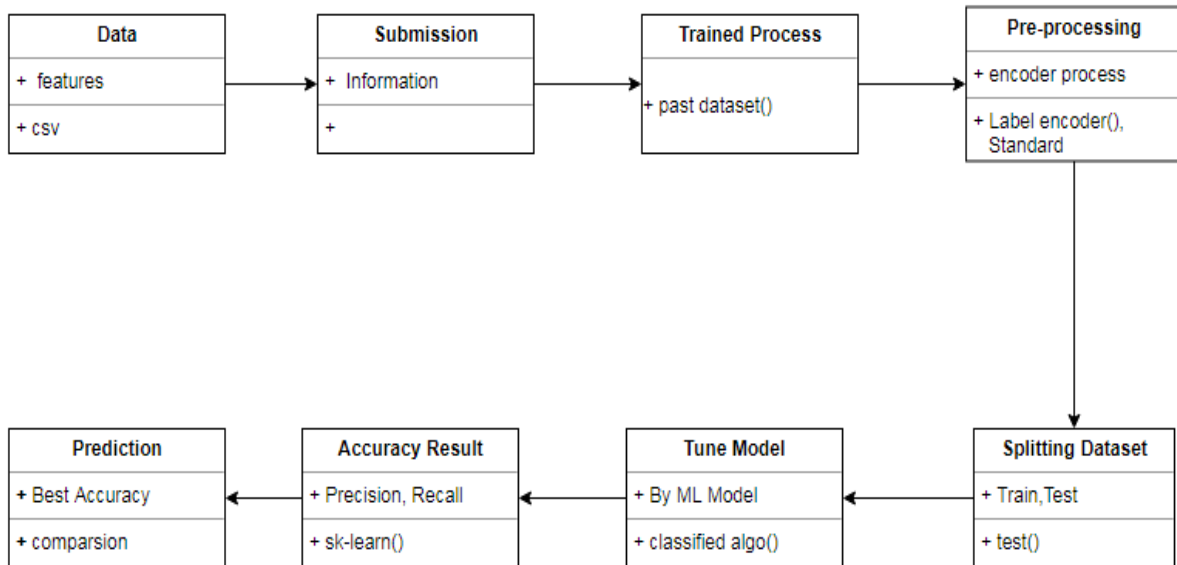


Fig 3.2 Class diagram of model

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:

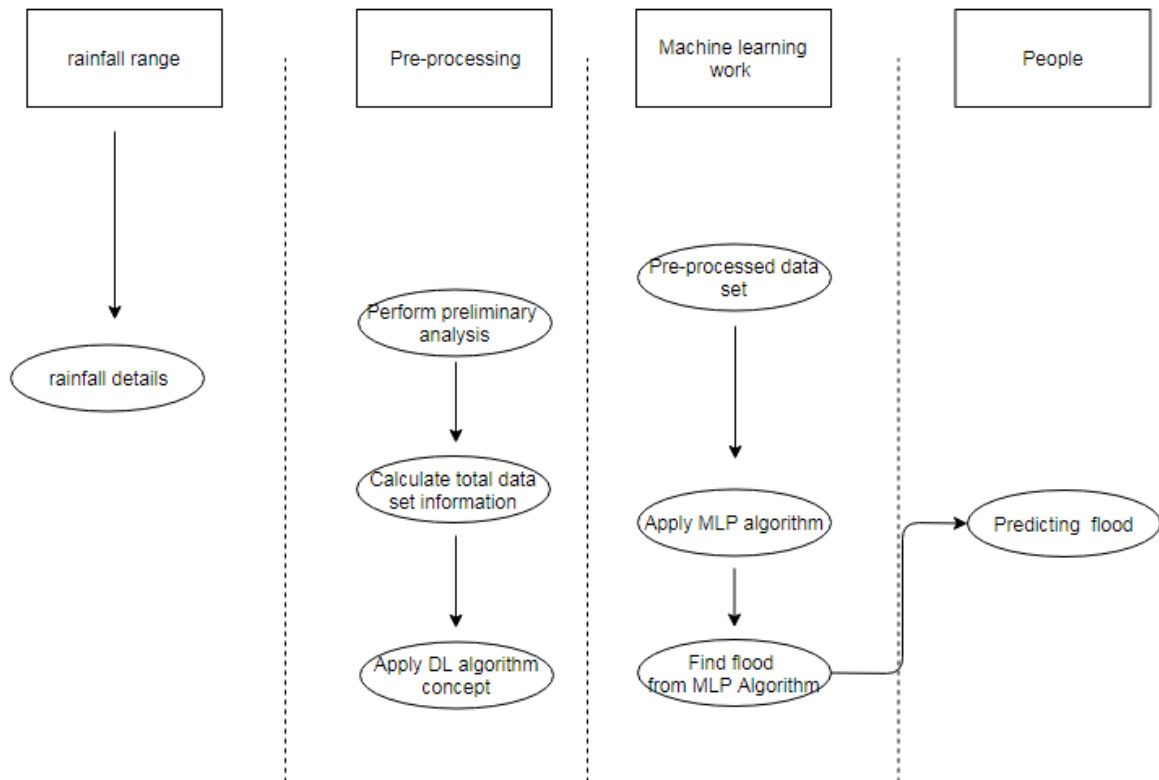


Fig 3.3 Activity diagram of model for predicting flood

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the 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 the activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered a flow chart. Although the diagrams look like a flow chart, they are not. It shows different flows like parallel, branched, concurrent and single.

20. Sequence Diagram:

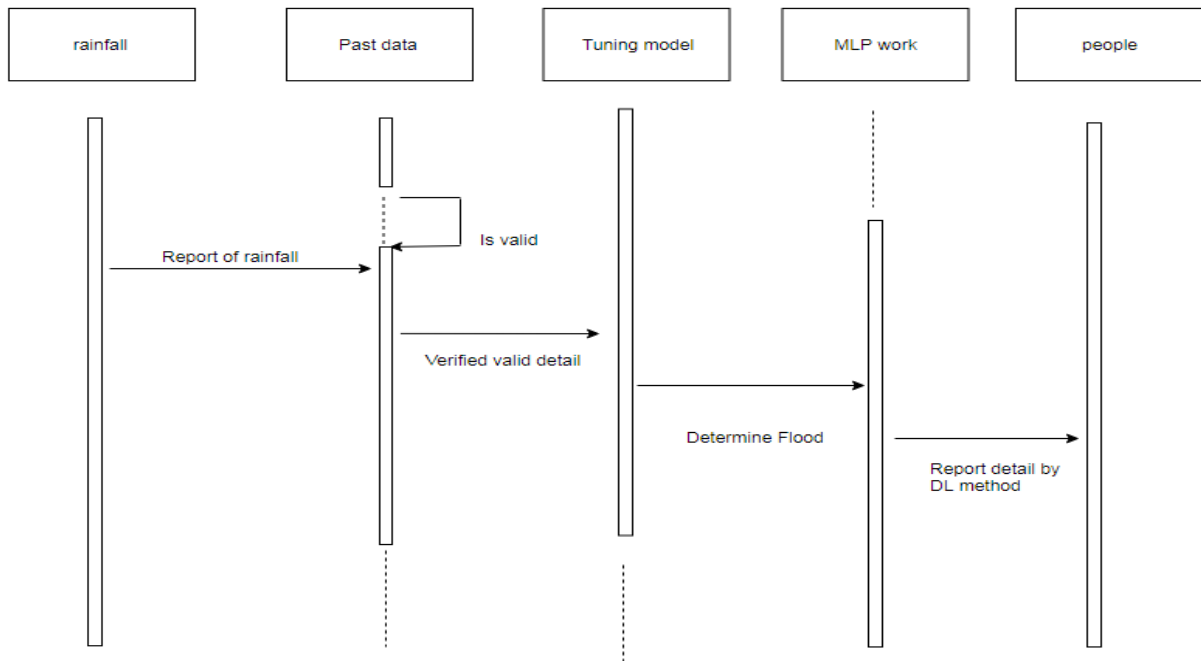


Fig 3.4 Sequence diagram for understanding logical view

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):

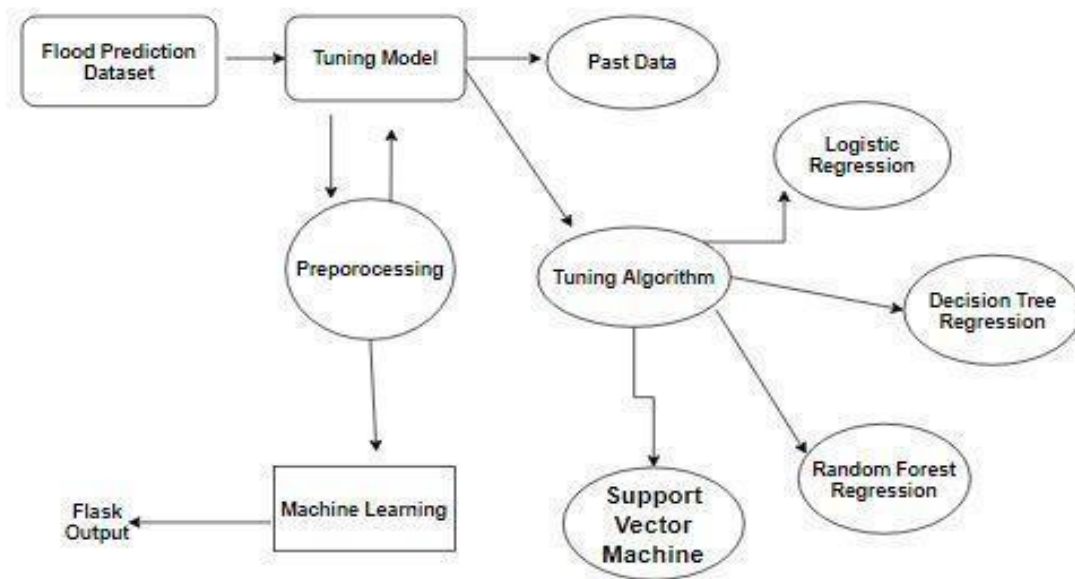


Fig 3.5 Entity relationship diagram for graphical view of predicting model

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. Collaboration Diagram:

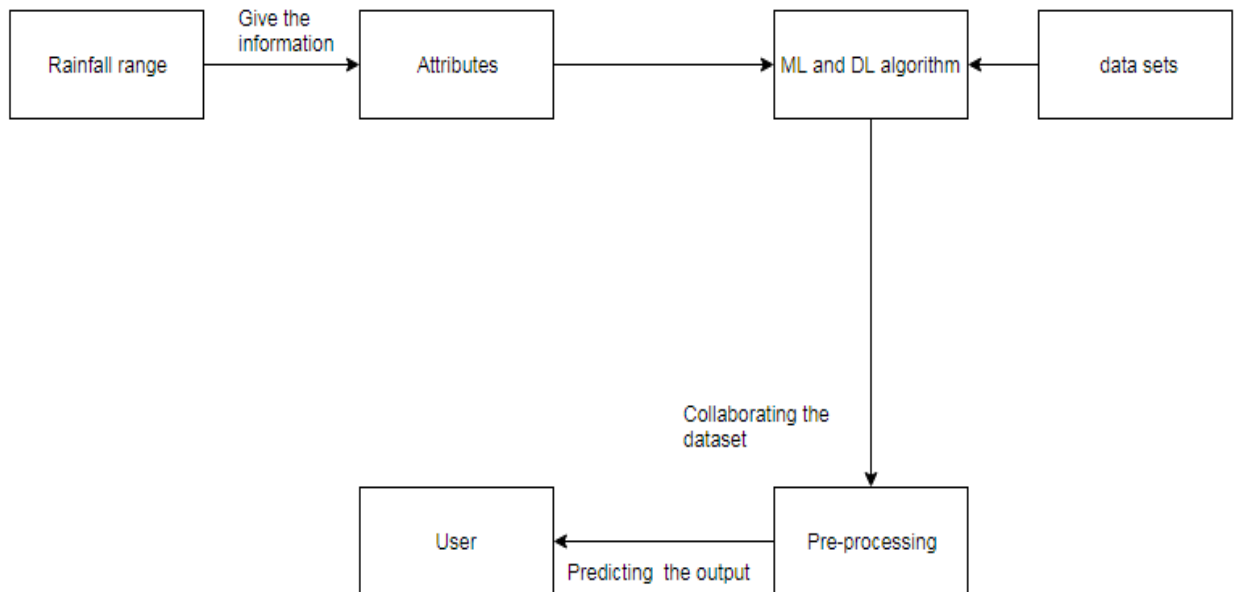


Fig 3.6 Collaboration diagram of Prediction model

A collaboration diagram is a type of visual presentation that shows how various software objects interact with each other within an overall IT architecture and how users (like doctor or patient) can benefit from this collaboration. A collaboration diagram often comes in the form of a visual chart that resembles a flow chart. It can show, at a glance, how a single piece of software complements other parts of a greater system.

23. 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 a given dataset. To find 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.

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.

Some of the 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 joining 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.

Variable identification with Uni-variate, Bi-variate and Multivariate 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

Data Validation/ Cleaning/Preparing Process

Importing the library packages by loading the given dataset. 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 renaming the given dataset and dropping the column etc. to analyze the uni-variate, bi-variate and multivariate processes. 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.

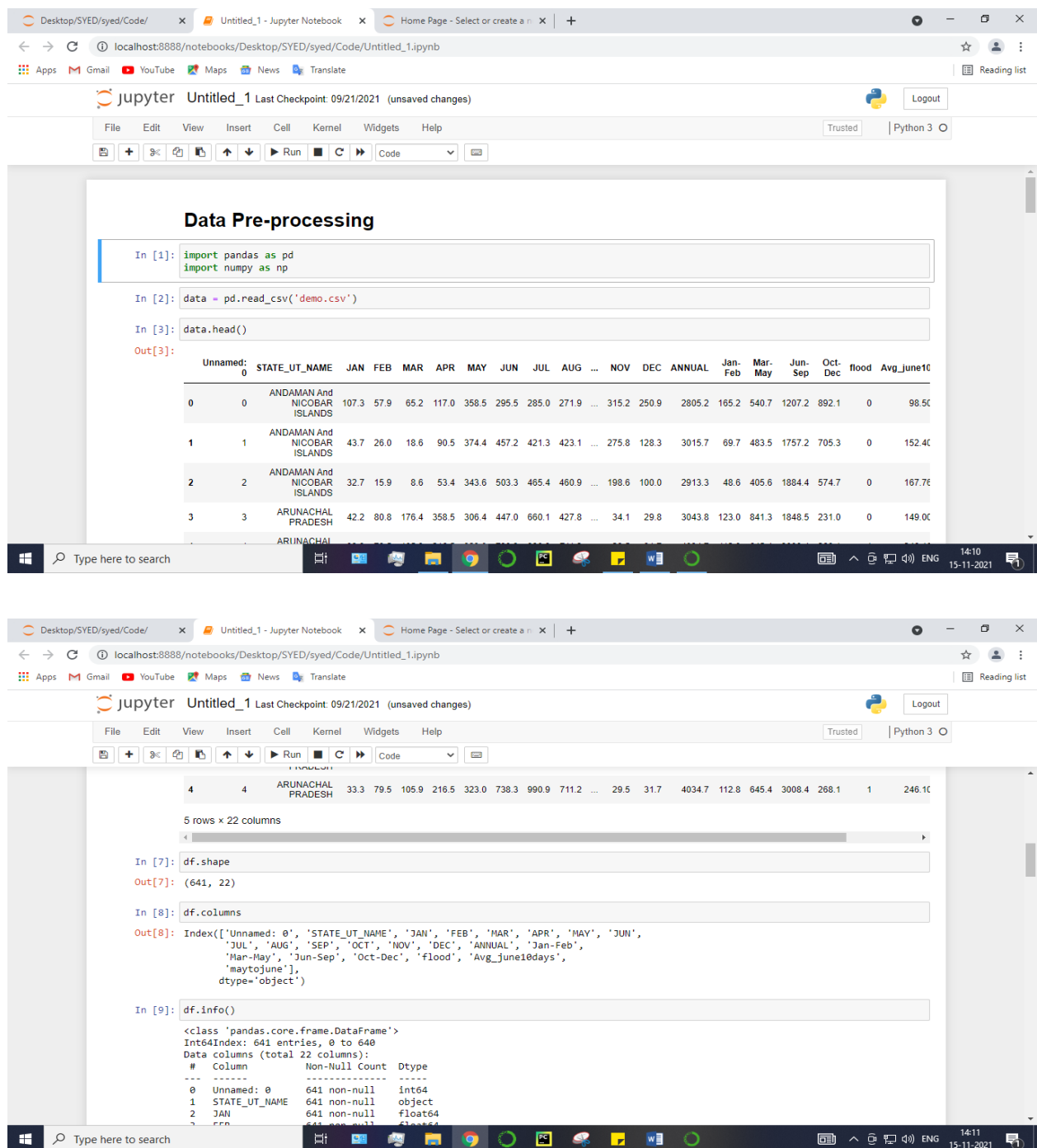
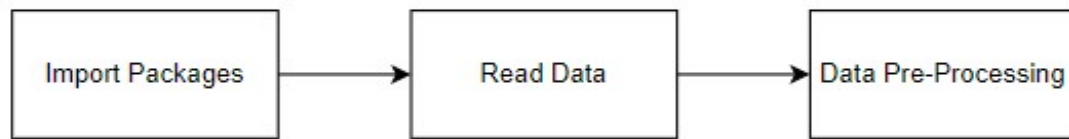


Fig 2.2 Output Screenshots of Data Preprocessing(cleaning) of collected data from sources

23.1 Module Diagram:



23.2 GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

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 of the books mentioned at the end.

Sometimes data does not make sense until it can be looked at in a visual form, such as with charts and plots. Being able to quickly visualize 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.

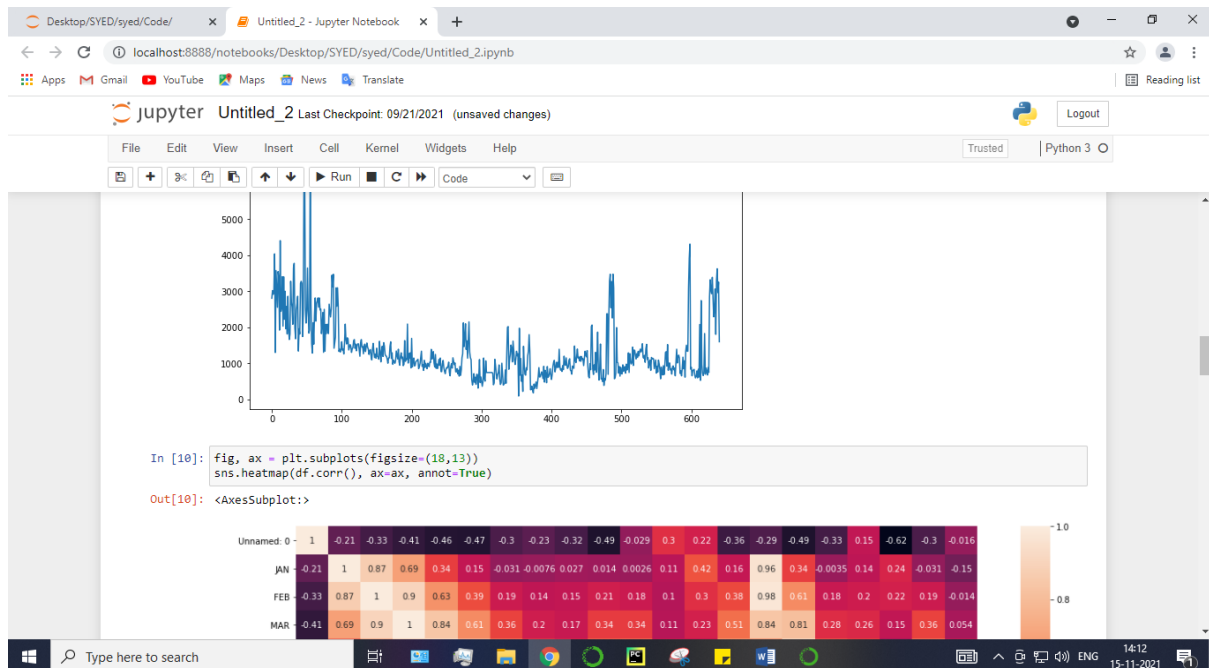
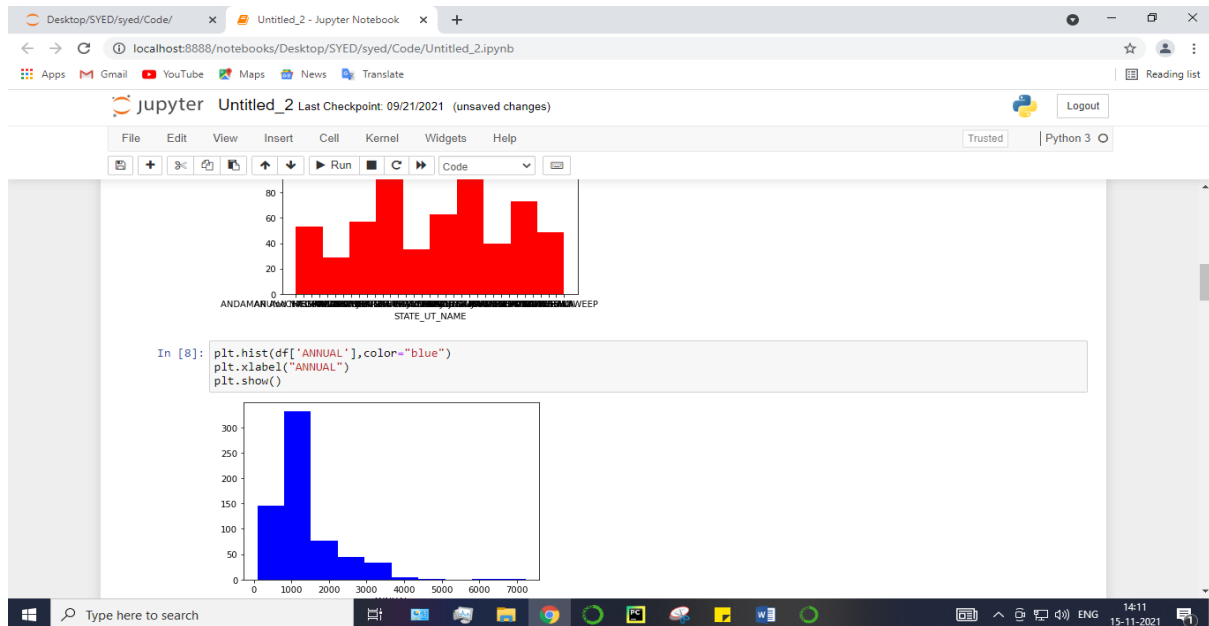
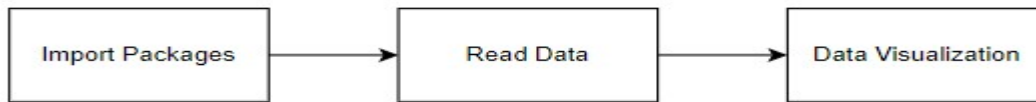


Fig 2.3 Output screenshots of Data Visualization of the collected data from sources

Module Diagram:



GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

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 achieve better results from the applied model in Machine Learning, the method of the data has to be in a proper manner. Some specified Machine Learning models need information in a specified format, for example, the Random Forest algorithm does not support null values. Therefore, to execute a random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data sets should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in a given dataset.

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

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

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

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

Comparing algorithms with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and 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 you 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 Classifier

Random Forest Classifier

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 comparing algorithms, I built a Machine Learning Model using the installed Scikit-Learn libraries. In this library package preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier have to be done. Additionally, splitting the train set and test set. To predict 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 needs the output of the algorithm to be classified as variable data. Higher accuracy predicting results is a logistic regression model by comparing the best accuracy.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

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:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

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 the same.

Precision: The proportion of positive predictions that are actually correct.

$$\text{Precision} = TP / (TP + FP)$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answers is of all passengers that are 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)

$$\text{Recall} = TP / (TP + FN)$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to all observations in the 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:

$$F\text{- Measure} = 2TP / (2 TP + FP + FN)$$

F1-Score Formula:

$$F1 \text{ Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

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 observations. 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, biometric 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 the pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which includes 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 math functions for calculations.
- It is used to read data in numpy arrays and for manipulation purposes.

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 identify the patterns from a given dataset.
- Data manipulation can be done easily with data frames.

MODULE – 3**Logistic Regression:**

It is a statistical method for analyzing 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.).

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

Binary logistic regression requires the dependent variable to be binary.

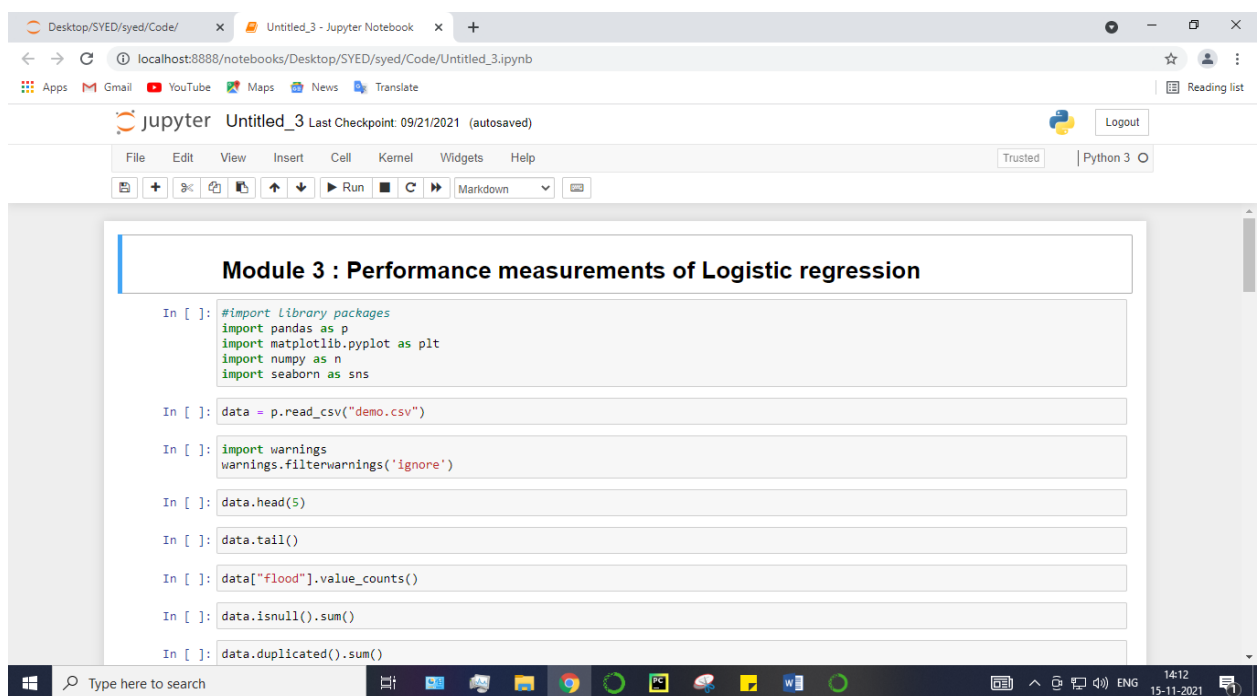
For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.

Only the meaningful variables should be included.

The independent variables should be independent of each other. That is, the model should have little.

The independent variables are linearly related to the log odds.

Logistic regression requires quite large sample sizes.



The screenshot displays a Jupyter Notebook titled 'Untitled_3' running on a local host. The notebook contains a series of Python code cells for data manipulation and analysis. The code includes importing necessary libraries (pandas, matplotlib, numpy, seaborn), reading a CSV file named 'demo.csv', and performing various data checks such as viewing the first five rows, the last five rows, value counts for a specific column, and identifying null or duplicated values.

```
In [ ]: #import Library packages
import pandas as p
import matplotlib.pyplot as plt
import numpy as n
import seaborn as sns

In [ ]: data = p.read_csv("demo.csv")

In [ ]: import warnings
warnings.filterwarnings('ignore')

In [ ]: data.head(5)

In [ ]: data.tail()

In [ ]: data["flood"].value_counts()

In [ ]: data.isnull().sum()

In [ ]: data.duplicated().sum()
```

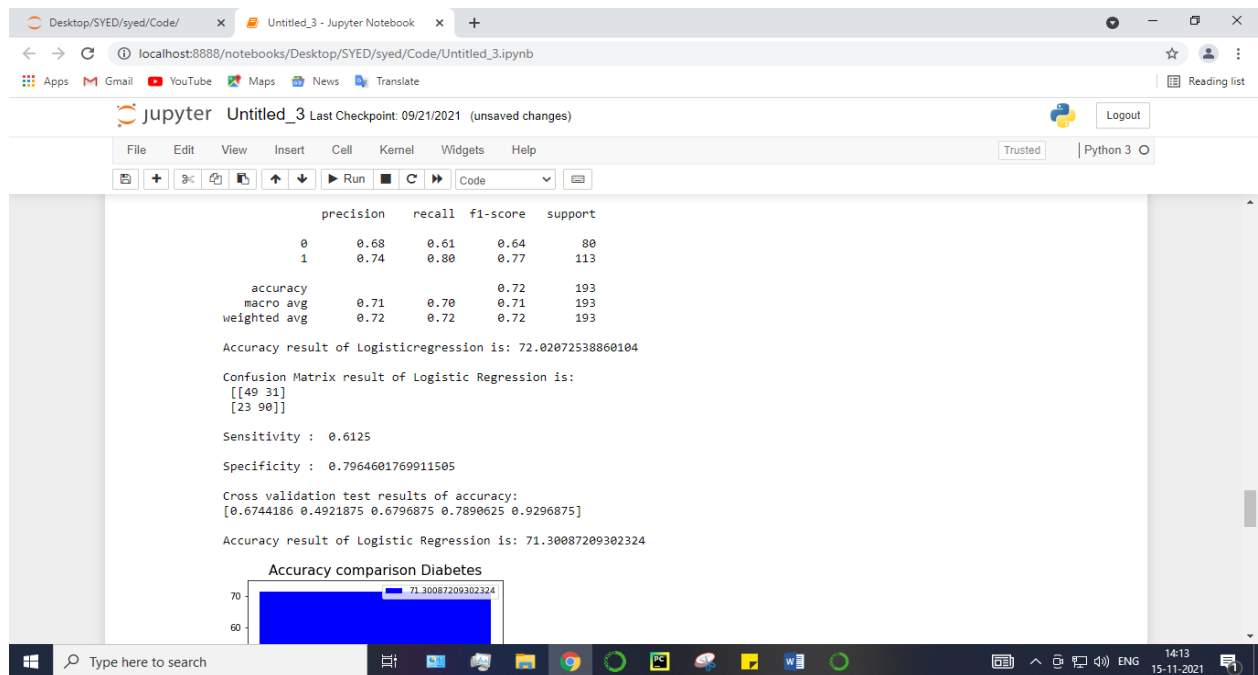
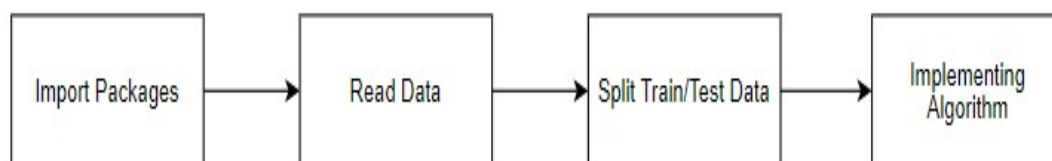


Fig 2.4 Output Screenshots of performance of Logistic Regression Algorithm

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

MODULE – 4:

Decision Tree:

It is one of the most powerful and popular algorithms. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Assumptions of Decision tree:

At the beginning, we consider the whole training set as the root.

Attributes are assumed to be categorical for information gain; attributes are assumed to be continuous.

On the basis of attribute values, records are distributed recursively.

We use statistical methods for ordering attributes as root or internal nodes.

Decision trees build classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor is called the root node. Decision trees can handle both categorical and numerical data. Decision trees build classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes at the top of the tree have more impact on the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted, generating too many branches and may reflect anomalies due to noise or outliers.

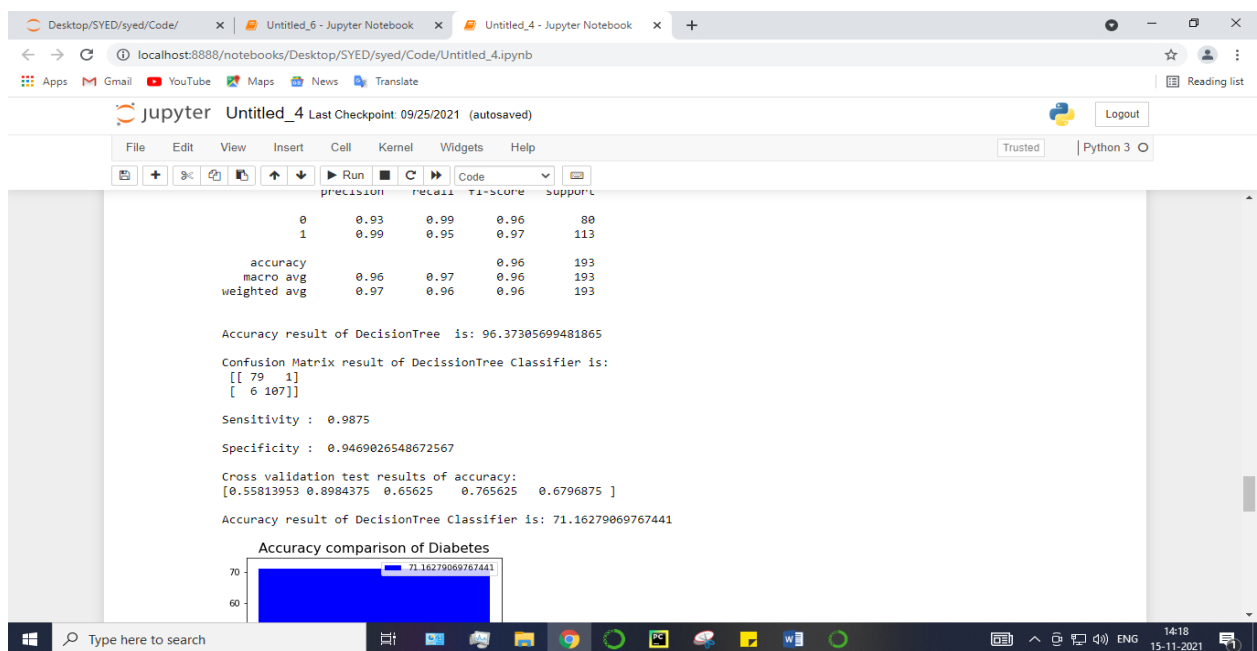
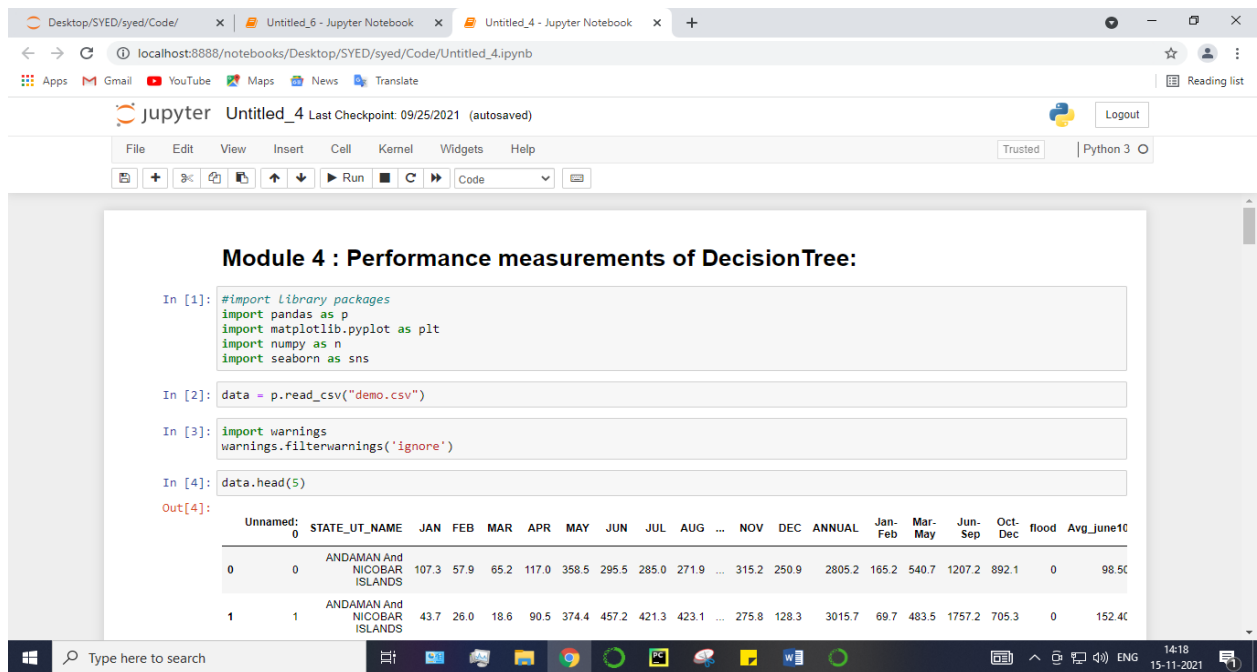
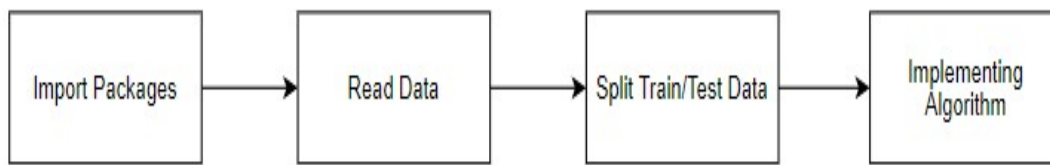


Fig 2.5 Output Screenshots of performance of Decision Tree Algorithm

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

MODULE – 5:

Random Forest Classifier:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over-fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or the same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

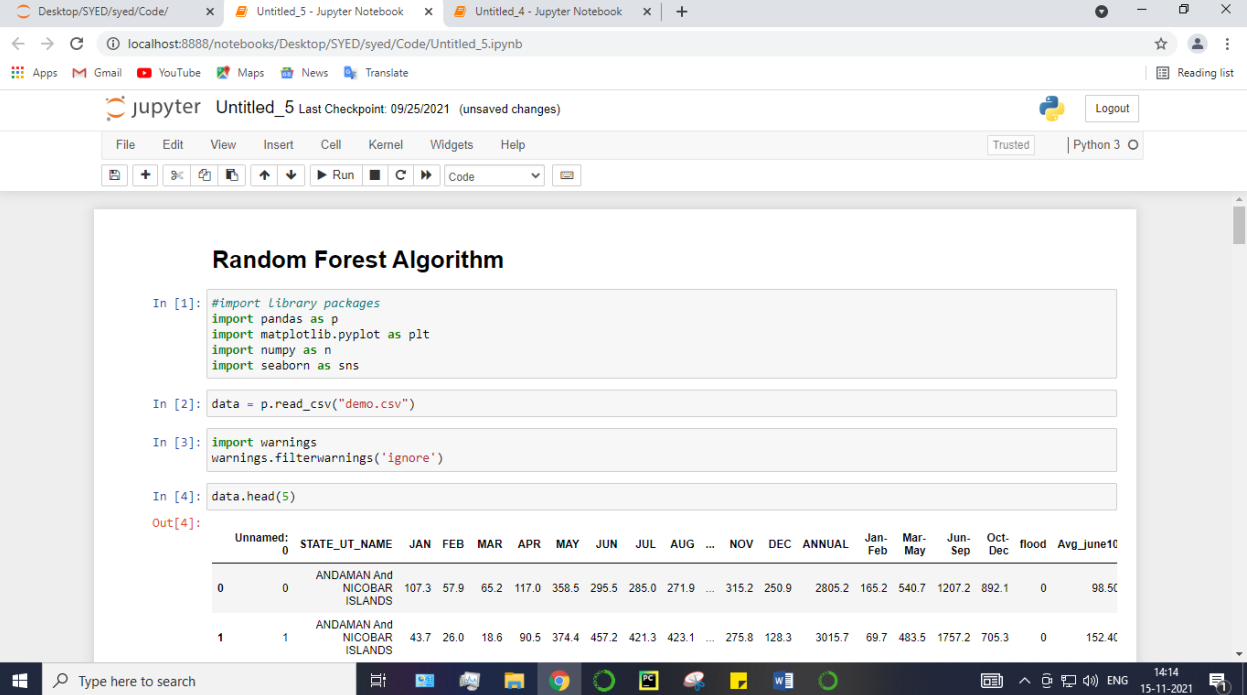
The following are the basic steps involved in performing the random forest algorithm:

Pick N random records from the dataset.

Build a decision tree based on these N records.

Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in the forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.



Random Forest Algorithm

```
In [1]: #import library packages
import pandas as p
import matplotlib.pyplot as plt
import numpy as n
import seaborn as sns
```

```
In [2]: data = p.read_csv("demo.csv")
```

```
In [3]: import warnings
warnings.filterwarnings('ignore')
```

```
In [4]: data.head(5)
```

Out[4]:

Unnamed: 0	STATE_UT_NAME	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	...	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec	flood	Avg_June10
0	ANDAMAN And NICOBAR ISLANDS	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	...	315.2	250.9	2805.2	165.2	540.7	1207.2	892.1	0	98.50
1	ANDAMAN And NICOBAR ISLANDS	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	...	275.8	128.3	3015.7	69.7	483.5	1757.2	705.3	0	152.40

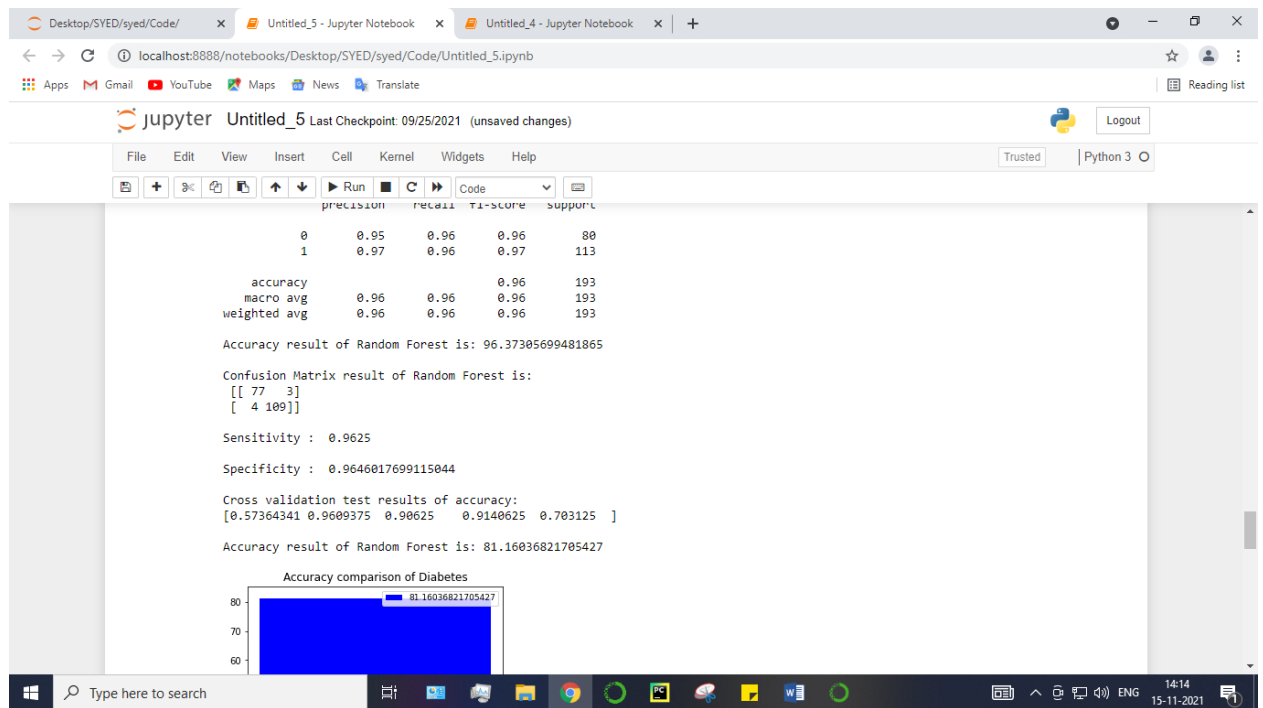


Fig 2.6 Output Screenshots of performance of Random Forest Algorithm

MODULE – 6:

Support Vector Machines

A classifier that categorizes the data set by setting an optimal hyperplane between data. I chose this classifier as it is incredibly versatile in the number of different kernelling functions that can be applied and this model can yield a high predictability rate. Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. They were extremely popular around the time they were developed in the 1990s and continue to be the go-to method for a high-performing algorithm with little tuning.

- How to disentangle the many names used to refer to support vector machines.
- The representation used by SVM when the model is actually stored on disk.
- How a learned SVM model representation can be used to make predictions for new data.
- How to learn an SVM model from training data.
- How to best prepare your data for the SVM algorithm.
- Where you might look to get more information on SVM.

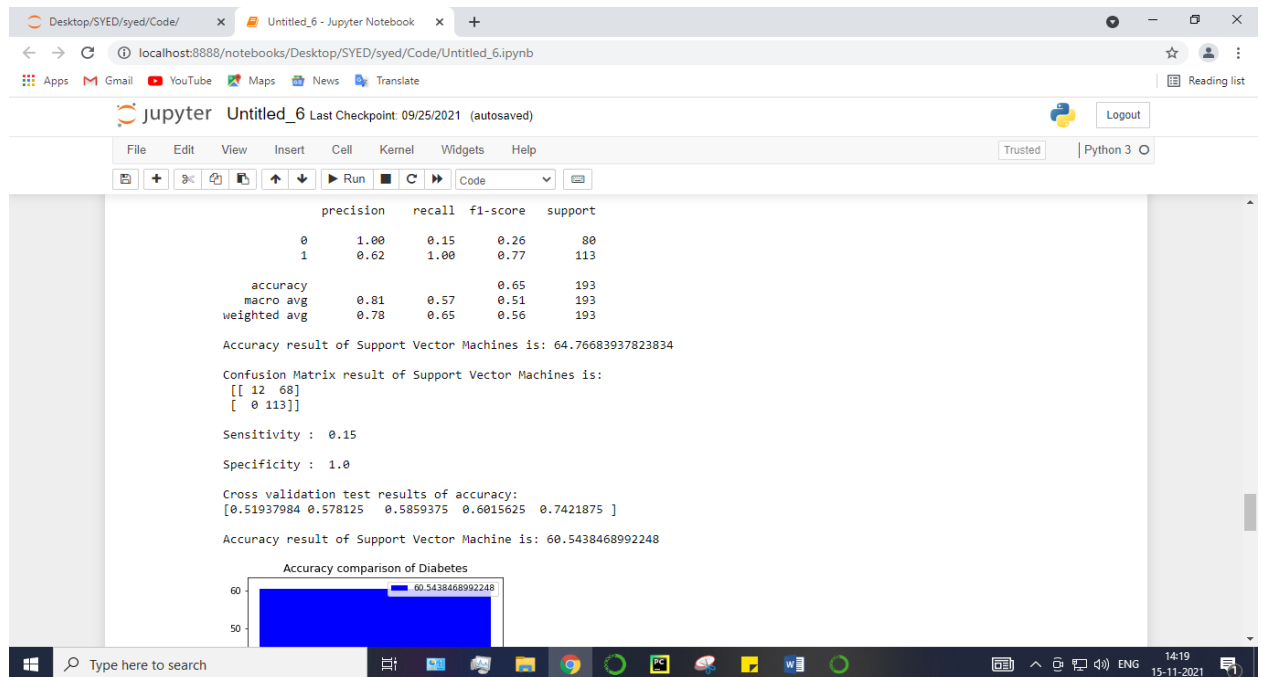
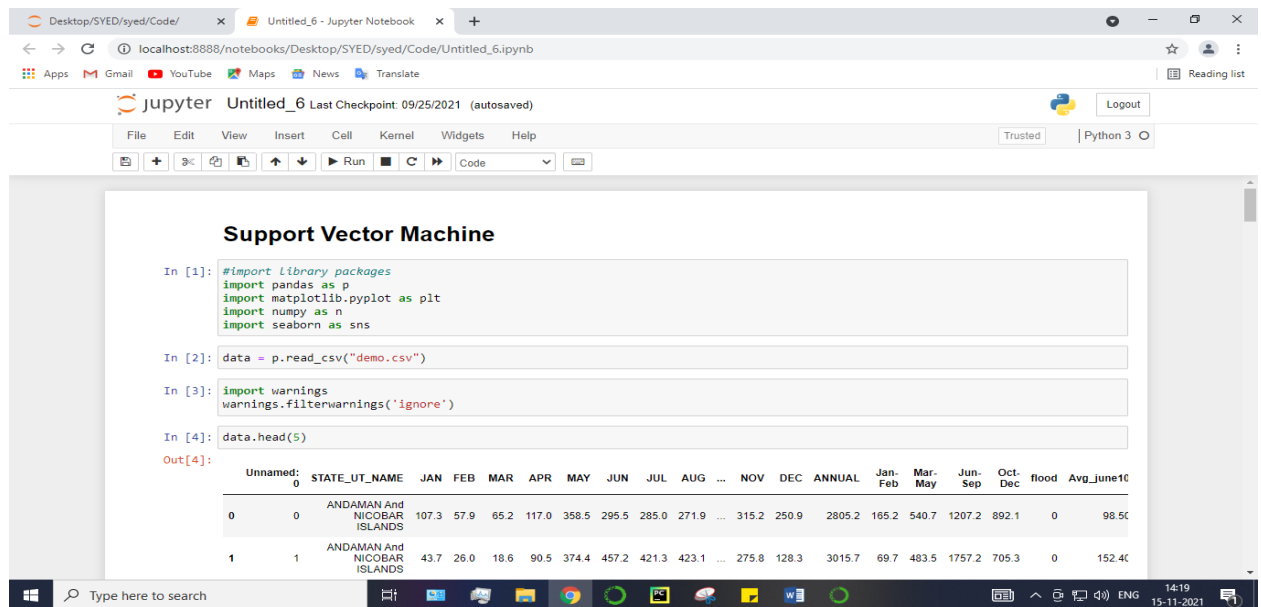


Fig 2.7 Output Screenshots of performance of Support Vector Machine Algorithm

24. Deployment

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 license. 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, the so-called "micro" framework for Python.

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
- 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 is 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 favorite 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 applications 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 solutions 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 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 URLs is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from the 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. The name of the module or package of the application is passed. 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 be executed, 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()`

25. HTML

HTML stands for Hyper Text 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 a 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.



`<!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">  
</head>
```

Adding Content

Next, we will make a `<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."

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 tag has everything you need to display images on your site. Much like the <a> anchor element, also contains an attribute.

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

```

```

26. CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colors, 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 coloring schemes in CSS

- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises 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 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

27. Coding:

Module – 1

```
import pandas as pd
```

```
import numpy as np
```

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

```
data.head()
```

```
data.shape
```

```
df = data.dropna()
```

```
df.head()
```

```
df.shape
```

```
df.columns
```

```
df.info()
```

```
df.describe()
```

```
df.isnull()
```

```
df.isnull().sum()
```

```
df.STATE_UT_NAME.unique()
```

```
df.columns
```

```
df.flood.unique()
```

```
df.JAN.unique().mean()
```

```
df.FEB.unique().mean()
```

```
df.ANNUAL.unique().min()
```

```
df.ANNUAL.unique().max()
```

```
df.ANNUAL.unique().mean()
```

```
df.Avg_june10days.unique()
```

```
df.corr()
```

Module – 2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('demo.csv')
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
df.head()
```

```
df.shape
```

```
df.columns
```

```
plt.hist(df['STATE_UT_NAME'],color="red")
```

```
plt.xlabel("STATE_UT_NAME")
```

```
plt.show()
```

```
plt.hist(df['ANNUAL'],color="blue")
```

```
plt.xlabel("ANNUAL")
```

```
plt.show()
```

```
plt.figure(figsize=(10,6))
```

```
plt.plot(df['ANNUAL'])
```

```
plt.show()
```

```
fig, ax = plt.subplots(figsize=(18,13))
```

```
sns.heatmap(df.corr(), ax=ax, annot=True)
```

```
boxplot = df.boxplot(column=['ANNUAL'], by="flood",figsize=(8,6), fontsize=15)
```

```
plt.show()
```

```
def PropByVar(df, variable):
```

```
    dataframe_pie = df[variable].value_counts()
```

```
    ax = dataframe_pie.plot.pie(figsize=(8,12), autopct='%1.2f%%', fontsize = 12)
```

```

ax.set_title(variable + '\n', fontsize = 15)
plt.show()
return np.round(dataframe_pie/df.shape[0]*100,2)
PropByVar(df,"flood")
plt.show()

plt.figure(figsize=(12,8))
plt.scatter(df["ANNUAL"],df["flood"],color="black")

data1=df.iloc[:,9]
data2=df.iloc[:,9:-1]

a=pd.plotting.scatter_matrix(data1, alpha=0.05, figsize=(18,10), diagonal='hist')
axis = 'off'

plt.show()

```

Module - 3 : Performance measurements of Logistic regression

#import library packages

```

import pandas as p
import matplotlib.pyplot as plt
import numpy as n
import seaborn as sns

```

```

data = p.read_csv("demo.csv")
import warnings
warnings.filterwarnings('ignore')

```

```

data.head(5)
data.tail()
data["flood"].value_counts()
data.isnull().sum()
data.duplicated().sum()
data.columns

```

```
data.shape
```

```
del data['JAN']
del data['Unnamed: 0']
del data['FEB']
del data['MAR']
del data['APR']
del data['MAY']
del data['JUN']
del data['JUL']
del data['AUG']
del data['SEP']
del data['OCT']
del data['NOV']
del data['DEC']
del data['Avg_june10days']
del data['maytojune']
data.head()
```

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod=["STATE_UT_NAME"]
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i]).astype(int)
data.head()
```

#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='flood', axis=1)
#Response variable
y = data.loc[:, 'flood']
```

#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=1,
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 Diabetes",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(cm2)
sns.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 numpy as n
import seaborn as sns

data = p.read_csv("demo.csv")
import warnings
warnings.filterwarnings('ignore')
data.head(5)
data.tail()
data["flood"].value_counts()

```

```
data.isnull().sum()
data.duplicated().sum()
data.columns
data.shape
```

```
del data['JAN']
del data['Unnamed: 0']
del data['FEB']
del data['MAR']
del data['APR']
del data['MAY']
del data['JUN']
del data['JUL']
del data['AUG']
del data['SEP']
del data['OCT']
del data['NOV']
del data['DEC']
del data['Avg_june10days']
del data['maytojune']
data.head()
```

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod=["STATE_UT_NAME"]
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i]).astype(int)
data.head()
```

#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='flood', axis=1)
```

```
#Response variable
```

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

```
#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=1, 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 Diabetes",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(cm2)
sns.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size": 16},
fmt='.2%'.)
plt.show()

```

Module – 5

#import library packages

```

import pandas as p
import matplotlib.pyplot as plt
import numpy as n
import seaborn as sns

```

```

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

import warnings
warnings.filterwarnings('ignore')

```

```
data.head(5)

data.tail()

data["flood"].value_counts()

data.isnull().sum()

data.duplicated().sum()

data.columns

data.shape
```

```
del data['JAN']
del data['Unnamed: 0']
del data['FEB']
del data['MAR']
del data['APR']
del data['MAY']
del data['JUN']
del data['JUL']
del data['AUG']
del data['SEP']
del data['OCT']
del data['NOV']
del data['DEC']
del data['Avg_june10days']
del data['maytojune']
```

```
data.head()

from sklearn.preprocessing import LabelEncoder
```

```
var_mod=["STATE_UT_NAME"]
le = LabelEncoder()
for i in var_mod:
    data[i] = le.fit_transform(data[i]).astype(int)
```

```
data.head()
```

#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='flood', axis=1)
```

#Response variable

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

#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=1,  
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 Diabetes")
    plt.legend(b,data,fontsize=9)

graph()

TN = cm1[1][0]
FN = cm1[0][0]
TP = cm1[1][1]

```

```

FP = cm1[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, predictR)
print(cm2)
sns.heatmap(cm2/n.sum(cm2), annot=True, cmap = 'Blues', annot_kws={"size": 16},
fmt='.2%'.)
plt.show()

```

```

import joblib
joblib.dump(rfc, 'flood.pkl')

```

Module – 6

#import library packages

```
import pandas as p
import matplotlib.pyplot as plt
import numpy as n
import seaborn as sns
```

```
data = p.read_csv("demo.csv")

import warnings
warnings.filterwarnings('ignore')

data.head(5)

data.tail()

data["flood"].value_counts()

data.isnull().sum()

data.duplicated().sum()

data.columns

data.shape
```

```
del data['JAN']
del data['Unnamed: 0']
del data['FEB']
del data['MAR']
del data['APR']
del data['MAY']
del data['JUN']
del data['JUL']
del data['AUG']
del data['SEP']
del data['OCT']
del data['NOV']
del data['DEC']
del data['Avg_june10days']
del data['maytojune']
```

```
data.head()
```

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod=["STATE_UT_NAME"]
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

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

```
data.head()
```

#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='flood', axis=1)
```

#Response variable

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

#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=1,  
stratify=y)
```

Support Vector Machines:

```
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 Diabetes")
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(cm2)

```

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

HTML:

```
<!DOCTYPE html>

<html >

<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>

  <meta charset="UTF-8">

  <title>Flood</title>

  <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">

    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
type='text/css'>

  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>

  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
type='text/css'>

  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>

  <style>

  .back{

background-image: url("{{ url_for('static', filename='image/img.jpg') }}");

background-size:cover;

}

.white{

color:white;

}
```

```

.space{
margin:10px 30px;
padding:15px 10px;
background: palegreen;
width:auto;
}

.nspace{
margin:10px 30px;
background: palegreen;
width:auto;
}

.gap{
padding:10px 20px;
}

</style>

</head>

<body class="back">

<div>

    <div class="jumbotron">

        <h1 style="text-align:center"> FLOOD PREDICTION </h1>

    </div>

    <!-- Main Input For Receiving Query to our ML -->

    <form class="form-group" action="{{ url_for('predict')}}" method="post">

    <div class="container">

        <div class="row">

            <div class="gap col-md-6">

                <label class="white" for="">STATE_UT_NAME</label>

```

```

        <select    class="nspace    form-control"    name="STATE_UT_NAME"
id="STATE_UT_NAME">

        <option    value=0>ANDAMAN    And    NICOBAR
ISLANDS</option>

        <option value=2>ARUNACHAL PRADESH</option>
        <option value=3>ASSAM</option>
        <option value=22>MEGHALAYA</option>
        <option value=21>MANIPUR</option>
        <option value=23>MIZORAM</option>
        <option value=24>NAGALAND</option>
        <option value=31>TRIPURA</option>
        <option value=34>WEST BENGAL</option>
        <option value=29>SIKKIM</option>
        <option value=25>ORISSA</option>
        <option value=15>JHARKHAND</option>
        <option value=4>BIHAR</option>
        <option value=32>UTTAR PRADESH</option>
        <option value=33>UTTARANCHAL</option>
        <option value=12>HARYANA</option>
        <option value=5>CHANDIGARH</option>
        <option value=9>DELHI</option>
        <option value=27>PUNJAB</option>
        <option value=13>HIMACHAL</option>
        <option value=14>JAMMU AND KASHMIR</option>
        <option value=28>RAJASTHAN</option>
        <option value=19>MADHYA PRADESH</option>
        <option value=7>DADAR NAGAR HAVELI</option>
        <option value=8>DAMAN AND DIU</option>

```

```

        <option value=20>MAHARASHTRA</option>
        <option value=10>GOA</option>
        <option value=6>CHATISGARH</option>
        <option value=1>ANDHRA PRADESH</option>
        <option value=30>TAMIL NADU</option>
        <option value=26>PONDICHERRY</option>
        <option value=16>KARNATAKA</option>
        <option value=17>KERALA</option>
        <option value=18>LAKSHADWEEP</option>
    </select>

    <label class="white" for=""> Jan-Feb</label>

    <input type="number" class="space form-control" step="0.01" name="
Jan-Feb" placeholder="Jan-Feb" required="required" /><br>

    <label class="white" for="">Mar-May</label>

    <input type="number" class="space form-control" step="0.01"
name="Mar-May" placeholder="Mar-May" required="required" /><br>

    <label class="white" for="">Jun-Sep</label>

    <input type="number" class="space form-control" step="0.01"
name="Jun-Sep" placeholder="Jun-Sep" required="required" /><br>

    <label class="white" for="">Oct-Dec</label>

    <input type="number" class="space form-control" step="0.01"
name="Oct-Dec" placeholder="Oct-Dec" required="required" /><br>

    <label class="white" for=""> ANNUAL</label>

    <input type="number" class="space form-control" step="0.01"
name="ANNUAL" placeholder="ANNUAL" required="required" /><br>
</div>
</div>
</div>

```

```

<div style="padding:2% 35%">
    <button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>
</div>

</form>

<br>

<br>

<div style="background:skyblue;padding:2% 40%">
    {{ prediction_text }}
</div>

</div>

</body>

</html>

```

Deploy:

```

import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import joblib

app = Flask(__name__)

model = joblib.load('flood.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 = 'Not Happen'
elif output == 1:
    output = 'Happen'

return render_template('index.html', prediction_text='Flood: {}'.format(output))

if __name__ == "__main__":
    app.run(host="localhost", port=8000)

```

OUTPUT SCREENSHOT:

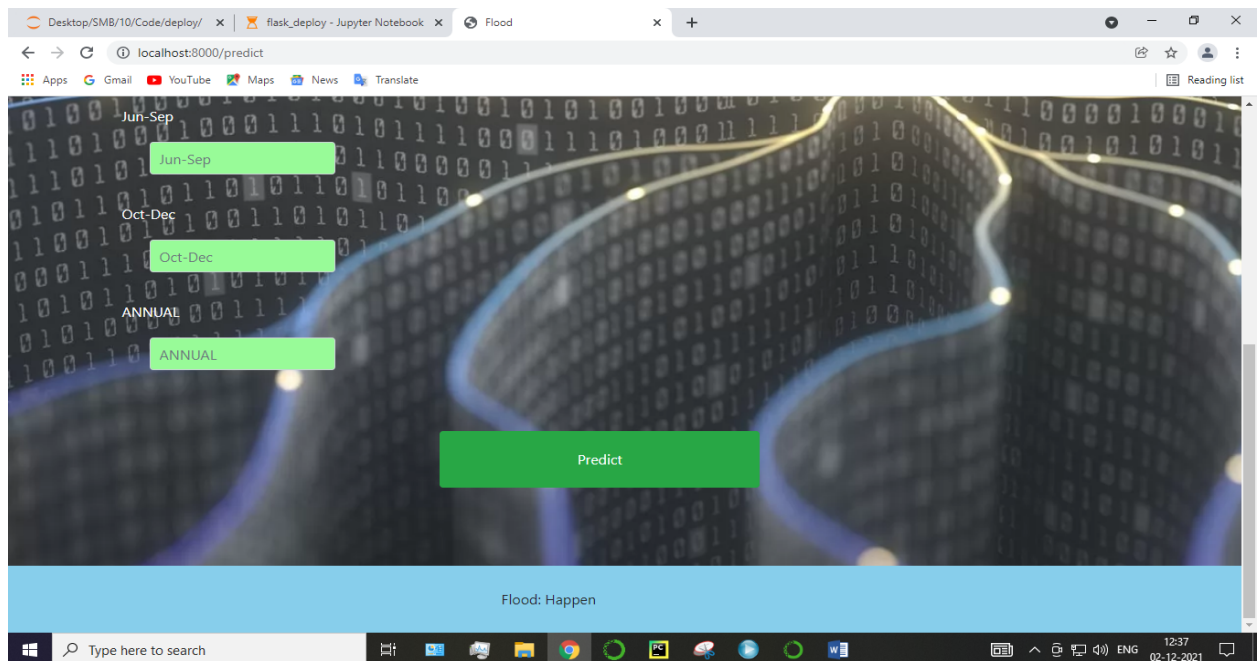
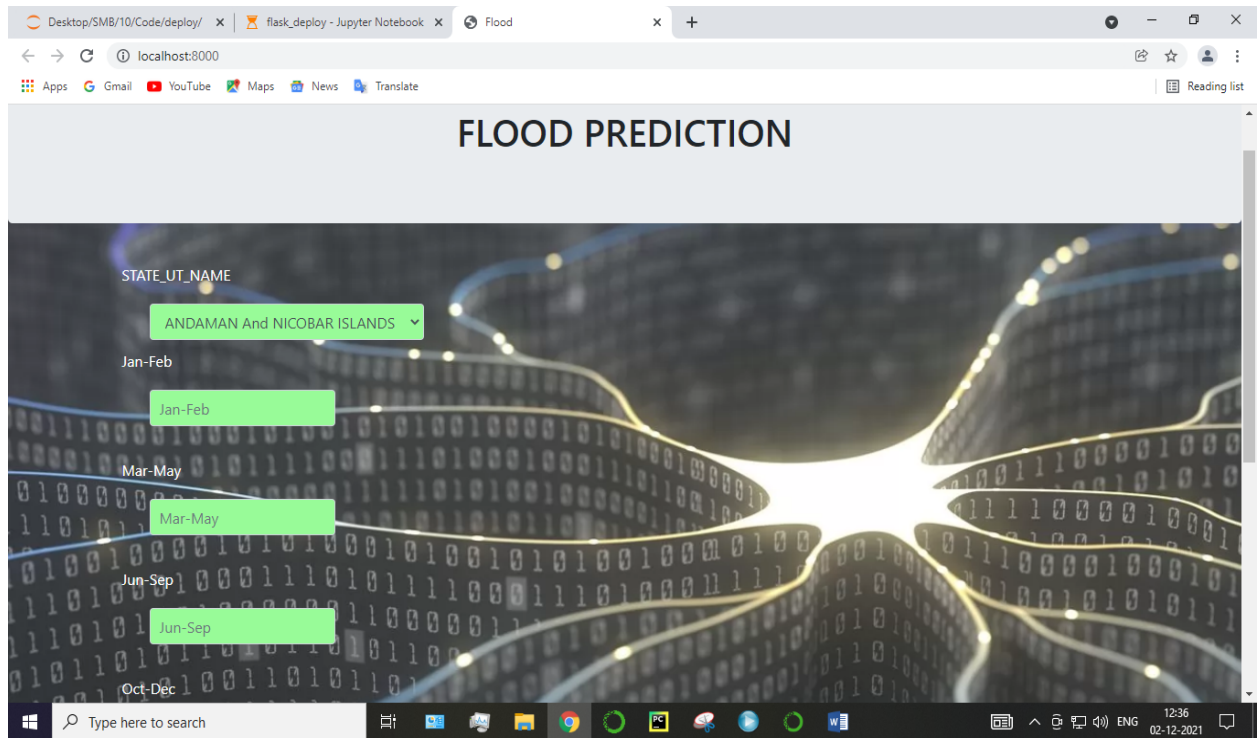


Fig 4. Final Output Screenshots of Flood Prediction Model

28. Conclusion:

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. Finally we predict the flash

flood using a machine learning algorithm with different results. This brings some of the following insights about flood prediction.

29. Future Work:

Disaster management wants to automate the detecting whether the flood happened or not from the eligibility process (real time).

To automate this process by showing the prediction result in web application or desktop application.

- To optimize the work to implement in Artificial Intelligence environment

30. References:

- [1]An IEEE published paper by [1]Bonafilia, D., Tellman, B., Anderson, T., & Issenberg, E. (2020). Sen1Floods11 A data set trained and test deep learning flood algorithms with georeferenced <https://doi.org/10.1109/CVPRW50498.2020.00113>
- [2]https://books.google.co.in/books/about/Flood_Forecasting_Using_Machine_Learning.html?id=mFeNDwAAQBAJ&printsec=frontcover&source=kp_read_button&hl=en&newbks=1&newbks_redir=1&redir_esc=y
- [3] S. Puttinaovarat and P. Horkaew, "Flood Forecasting System Based on Integrated Big and Crowdsourced Data by Using Machine Learning Techniques," in IEEE Access, vol. 8, pp. 5885-5905, 2020, doi: 10.1109/ACCESS.2019.2963819.
- [4] ZENING WU, YIHONG ZHOU, AND HUILIANG WANG College of Water Conservancy Engineering, Zhengzhou University, Zhengzhou 450001, China Corresponding author: Huiliang Wang (wanghuiliang@126.com) Real-Time Prediction of the Water Accumulation Process of Urban Stormy Accumulation Points Based on Deep Learning
- [5] MOHAMMED KHALAF 1, HAYA ALASKAR 2 , ABIR JAAFAR HUSSAIN 3 , THAR BAKER 3 , ZAKARIA MAAMAR 4 , RAJKUMAR BUYYA 5 , PANOS LIATSIS 6 , (Senior WASIQ KHAN 3 , HISSAM TAWFIK 7 ,AND DHIYA AL-JUMEILY

- [6] Ni Komang Ega Kartika, Muhammad Ary Murti, Casi Setianingsih. School of Electrical Engineering, Telkom University, Indonesia, flood prediction using RBF based on IOT
- [7] Akshya .J, P.L.K.Priyadarsini School of Computing SASTRA Deemed to be University Thanjavur, India akshya@sastra.ac.in, priya.ayyagari@it.sastra.ed flood prediction using Aeiral images
- [8] Swapnil Bande Dept. of Electronics & Telecommunication MIT College of Engineering Pune, India swapnilbande11@gmail.com Prof. Dr. Virendra V. Shete Dept. of Electronics & Telecommunication MIT College of Engineering Pune, India virendra.shete@mitcoe.edu.in
- [9] O. M. Rezende, A. B. R. da Cruz de Franco, A. K. B. de Oliveira, A. C. P. Jacob, and M. G. Miguez, “A framework to introduce urban flood resilience into the design of flood control alternatives,” *J. Hydrol.*, vol. 576, pp. 478–493, Sep. 2019.
- [10] H. Wang, C. Mei, J. Liu, and W. Shao, “A new strategy for integrated urban water management in China: Sponge city,” *Sci. China Technol. Sci.*, vol. 61, no. 3, pp. 317–329, Jan. 2018.
- [11] U. C. Nkwunonwo, M. Whitworth, and B. Baily, Flood risk management in urban areas ,” *Sci. Afr.*, vol. 7, Mar. 2020, Art. no. e00269.
- [12] N. Singh and S. R. Mohanty, “Short term price forecasting using adaptive generalized neuron model,” *Int. J. Ambient Comput. Intell.*, vol. 9, no. 3, pp. 419–428, Jul. 2018.
- [13] S. Chatterjee, S. Sarkar, S. Hore, N. Dey, A. S. Ashour, and V. E. Balas, “Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings,” *Neural Comput. Appl.*, vol. 28, no. 8, pp. 2005–2016, Aug. 2017.
- [14] V. A. Dev and M. R. Eden, “Formation lithology classification using scalable gradient boosted decision trees,” *Comput. Chem. Eng.*, vol. 128, pp. 392–404, Sep. 2019.
- [15] L. A. Treinish, A. P. Praino, J. P. Cipriani, U. T. Mello, K. Mantripragada, L. C. V. Real, P. A. Sesini, V. Saxena, T. George, and R. Mittal, “Enabling high-resolution forecasting of severe weather and flooding events in Rio de Janeiro,” *IBM J. Res. Develop.*, vol. 57, no. 5, pp. 7:1–7:11, Sep. 2013.

- [16] J. Noymanee and T. Theeramunkong, “Flood forecasting with machine learning technique on hydrological modeling,” *Procedia Comput. Sci.*, vol. 156, pp. 377–386, Jun. 2019