Flood Detection and Prediction Using Satellite Imagery and Machine Learning

Saniya Lande, Shilpa Shivarudraiah, Iqra Bismi, Divya Nalam

Department of Applied Data Science, San Jose State University

Data 270: Data Analyst Process

Professor Dr. Eduardo Chan

Date: December 11, 2022

Abstract

Floods are the most commonly occurring natural disaster resulting in human suffering, human loss, and property damage. People suffer due to the lack of a proper flood management system which indicates the need for flood forecasting. Therefore, implementation of machine learning algorithms was done to predict floods accurately. It was observed that supervised machine learning algorithms can face challenges due to hydrological complexity of the data extracted in different formats. Hence, for further research, development of three supervised machine learning models i.e. Support Vector Classifier, Random Forest Classifier, XGBoost Classifier, followed by one unsupervised model i.e. K-means Clustering was conducted to analyze the accurate model for Flood Prediction. With respect to modeling and evaluation done on these models, an accuracy of 97% was observed for the Random Forest Classifier and XGBoost Classifier. On the other hand, K- Means Clustering had an accuracy of 91% whereas Support Vector Classifiers showed the least accuracy of 82%. After looking through the accuracy and other evaluation metrics, it was concluded from the research that Random Forest Classifier and Ensemble models predict floods most accurately based on rainfall, tidal, and satellite images data. Implementing the best models will help to take preventive steps in a timely manner to minimize flood damage. More climatic features like the soil type, and improper drainage system can be included for further improvising the process approach and prediction accuracy.

Keywords: Flood prediction, Support Vector Classifier, Random Forest, XGboost, K-means, satellite images.

1. Introduction

1.1 Project Background and Executive Summary

1.1.1 Project Background, Motivations, Target Problem, Goals, Needs and Importance

Floods are the most frequently occurring natural disasters (46%) which causes human suffering, human loss, property damage, etc. They affect almost three times as many people as were affected by tropical cyclones. Though more people were killed by earthquakes, the number of people affected by floods was higher. To date, many people have suffered due to the lack of a proper flood management system. Hence, flood forecasting was very critical and an important subject to be studied as it can be helpful to reduce vulnerabilities, loss of lives, which will further contribute to national sustainable development. To address this issue, different machine learning algorithms to predict floods would be used.

The main goal of the research was to accurately predict floods early using satellite imagery data along with the current rainfall and tidal data. The focus of analysis was Kerala - a state from the Indian sub-continent as Kerala records high rainfall and has experienced floods before. Implementation of supervised as well as unsupervised machine learning models for predicting floods was done. Understanding which ML algorithm was more robust and gives better flood prediction accuracy was one of the important tasks. By accurately predicting the likelihood of floods, early warnings/alerts can be sent to all the individuals living in the flood prone areas so that necessary actions like allocating resources to help individuals, evacuating them, etc. can be taken to minimize life losses as well as economic losses to some extent.

1.1.2 Project Approaches and Methods

Data was collected from three different sources. First was the INSAT images that were extracted from the Indian Meteorological website. Second, the information about rainfall level was extracted from the open weather API. Lastly, the data about the height and status of tides was collected from Marea API. In addition to this, the INSAT images were updated every hour and these images were collected via python script by specifying latitude and longitude to collect images for specific locations. In this project, images for Kerala were collected, India as the area was highly affected by flood.

For INSAT images, a python script was used to crop images by defining specific coordinates of a 15 km square. The data collected from the weather API and marea API was in json format having mixed data types. So first the json format was converted into structured rows/columns by initializing a data frame in pandas. Tto clean this data, different sklearn modules were used. StandardScaler was used to scale the values in the range (-1,+1). SimpleImputer was used to fill the missing values with the mean/mode of the data.. In this project, implementing different supervised machine learning models was done. These were the SVM, Random Forest and ensemble methods for flood prediction.

Li. et al. (2016) mentioned boosting ensemble models for flood prediction. Boosting model consisted of various sub models which followed a resampling approach and the sample weight was calculated using loss function and learning rate. This resampling technique was applied to hydrological data. SVM was selected as base estimator and the prediction model was obtained by collating all sub-models to improve the overall accuracy. The author calculated the MSE (mean squared error) between both the single and ensemble model. The MSE dropped from

39118 (using a single model) to 36036 (ensemble model), i.e. improved by 7.9%. As similar data was available, ensemble models would be used in order to achieve an increased accuracy.

De Castro. et al. (2021) mentioned that random forest models were efficient in handling large data and prevented overfitting the model by using multiple trees. In this each sample was taken using a bootstrapping technique and further randomisation was generated by selecting a subset of features at each node without replacement. The author had implemented a model on hydrological data to create a flood risk index by integrating the model with GIS techniques. The author also tested various ML classifiers such as KNN, Random Forest, naïve bayes and out of all the models Random Forest had the highest accuracy i.e. accuracy (0.96) with highest correlation coefficient of 0.77. Random forest model in the project would be done as the scope of the project was similar to this paper.

SVM is an excellent classification model that makes no data assumptions (unlike logistic regression where linearity of data was assumed). SVM can also manage outliers (unlike linear regression where any presence of outliers has a huge effect on the output). Razali et al. (2020) in their research compared the performance of four models namely Bayesian Network, Decision Tree, k-nearest Neighbors (k-NN) and Support Vector Machines. Highest accuracy (99.92 %) was observed for Decision Tree followed by SVM which showed an accuracy of 99.76 %. The highest values for Precision, Recall and F-measure were observed for Decision Tree which was 0.999 followed by SVM which was 0.98. Similarly, Sahoo et al. (2021) also observed that SVM when combined with Firefly Algorithm shows the highest Coefficient of Determination (R^2) of 0.98 when compared to other ANN models which was 0.86 and 0.89 respectively for Feed Forward Back Propagation Neural Network (FFBPNN) and Radial Basis Function Neural

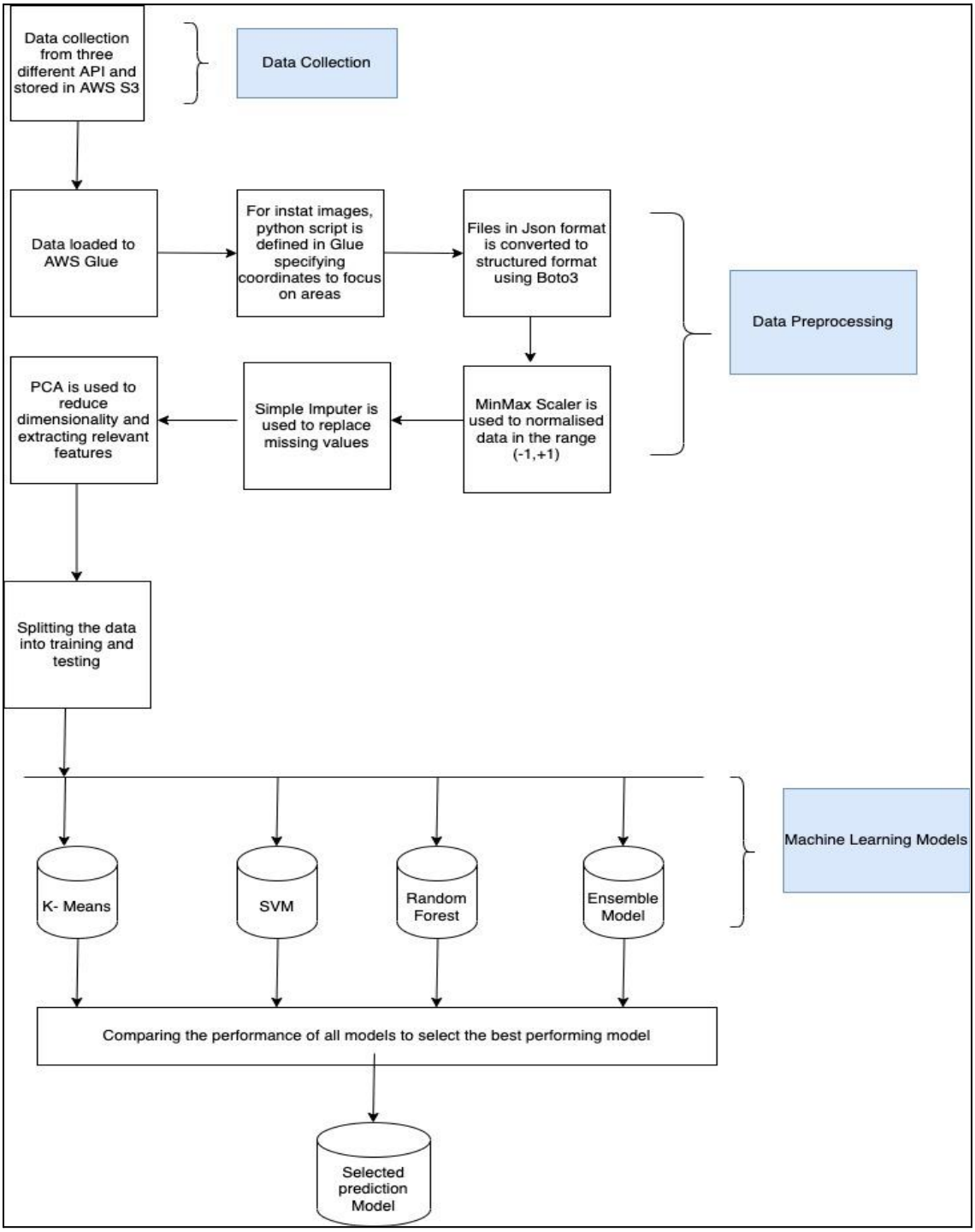
Network (RBFNN) models respectively. From these papers, SVM showed better performance than many other models, hence SVM was included for the analysis.

Unsupervised models operate on unlabeled data, making them ideal for predicting. After clustering the training data utilizing k-means, each row of training data was taken, calculations between the distance between the cluster centroids were made and this point was assigned to the nearest cluster (where each cluster is a particular class or type). According to Chakraborty, S. et al. (2012), clustering is the one of the best tools for detection jobs. Clusters were created based on the parameters or features and each data point was mapped to the closest cluster.

The work-flow diagram for the entire project is as shown in Figure 1.

Figure 1

Workflow diagram



1.1.3 Expected Project Contributions and Applications

This research helped to test various models based upon the prediction accuracy, computational time required, etc. and to determine the most robust machine learning model that can accurately predict the likelihood of floods before the actual floods happen. This project focused on predicting the flood for only a particular state in India. However, after looking at the success rate of these predictions, the same model can be applied globally so that floods across the globe can be predicted accurately thereby minimizing the disaster caused due to it on a larger scale.

1.2 Project Requirements

1.2.1 Functional Requirements

Defining functional requirements in any disaster detection and prediction systems is necessary because there is no room for error, failure or interruption of systems in disaster management. In this case of flood detection and prediction, the system should be able to predict a flood based on the available data (like tide information, rainfall data and others) as soon as possible so that necessary preventive measures and planning can be done at ease. This also means generating alerts and forwarding them to the respective stakeholders immediately. Regarding the data, there was no licensing, authorization or any such legal or regulatory requirements required as the data was collected from open source API's (or data classified as public).

1.2.2 AI Requirements

As discussed earlier, SVM, Random Forest supervised machine learning models and unsupervised machine learning models such as k-means clustering and PCA were used. It was crucial to consider all aspects like data size, performance and accuracy while choosing the

model for the applications. SVM is a good model for classification without any assumptions about data (unlike logistic regression where linearity of data was assumed). SVM can handle outliers as well (unlike linear regression where any presence of outliers has a huge effect on the output). Random Forest is also a good fit for the project as it is perfect for the current case as it uses multiple decision trees therefore eliminating the chance of overfitting. Regression and Classification were few of the popular applications of this model. Regarding unsupervised machine learning models, unsupervised models work on unlabelled data, making it quite suitable for the case of forecasting. In k-means clustering, clustering the training data is done, each row of training data was taken, find the distance of this point from the cluster centroids and assign this point to the closest cluster (where each cluster was a particular class or type). Finding the most important features (features with highest weights) can make simple prediction easier.

1.2.3 Data Requirements

As discussed in the earlier sections, the data was curated from three sources: an Indian Meteorological webpage for INSAT images, OpenWeather API for rainfall data and Marea API for tide data. The INSAT images were extracted with the help of a python script by defining the respective latitude and longitude. The INSAT images consisted of data such as Visual, IR, time and water-vapor which is captured by satellite for the Indian subcontinent. From MareaAPI, the height and state of the tide was noted which was used to determine the tide type.
Talking about the format of data from OpenWeatherAPI and MareaAPI, the format was in json format.

1.3 Project Deliverables

The paper aimed at developing the best machine learning model for the proposed problem of flood prediction. The main goal was to implement supervised Machine learning

models and unsupervised Machine Learning models on satellite imagery to understand which method was more robust and gave better flood prediction accuracy. The paper's end deliverables was to present the best model which leads to early and accurate detection of floods. Also, a report detailing the process of flood prediction model using Machine Learning algorithms was ideally provided. Table 1 provides the summary of the deliverables with a timeline.

Table 1

Project deliverables

Deliverable	Description	Due Date
Project Proposal	Brief overview of the topic selected along with the examples of the data that will be used for the analysis and literature review regarding the topic of research selected.	9/21/22
Chapter 1 Introduction	Overview of project background, requirements, project approach and description of models that will be used for the research (RF,SVM, Ensemble, K-means) along with literature survey, technology survey for relevant research papers	9/28/22

Deliverable	Description	Due date
WBS	Use clickup to set 6 phases of CRISP-DM and the respective components and work packages to list out different tasks in the project	10/3/22
PERT & Gantt Charts	Gantt charts to illustrate the project schedule (different milestone to be achieved)	10/10/22
Chapter 2 Data & Project Management Plan	Data stored on local machine	10/17/22
Data Collection & Data Exploration Plans	INSAT images - collect from IMD website using python script after every 15 minutes Rainfall and tidal data - Collect from Open Weather API and Marea API . Data exploration - Type of files obtained from API is in JSON,convert it to a structured format	10/24/22

Deliverable	Description	Due Date
Data Engineering	Collection of data from different sources such as INSAT images and through API. Cleaning and Validation of data. ETL process	10/31/22
Chapter 3 Data Engineering	data collection, data pre-processing. separating it to training and test data and presenting the results.	11/14/22
Abstract	Summary of the entire research done	11/21/22
Project Presentation	Present the project work with the key learning/Findings.	11/28/22
Individual Report	Individual research paper on respective ML models (SVM, Random Forest, Ensemble and K-Means).	12/5/22
Final Group Report	Final Group report on the project detailing the process of arriving at the best ML model for prediction of floods.	12/5/22

1.4 Technology and Solution Survey

With reference to ensemble model, Felix et al. (2019) mentioned in the paper that floods were one of the most common environmental disasters which lead to damaged buildings, power supply networks and creating hardships in the livelihood of people. Due to this, various machine learning models have been implemented over the years to predict flood detection. In the paper, the authors had implemented a gradient boosting model (a type of ensemble model) to classify the data and predict the flood threat with high accuracy. The gradient boosting model implemented ensemble methods to reduce noise and variance from the data. The model used both bagging and boosting techniques which trained multiple independent models and then implemented them in sequence so that the new model learnt from the error produced by its predecessor. It followed a concept of running and calculating the residual models so that the new model was a strong leaner (i.e. the cost function of the model is at its minimum). Also, the model boosted each weak learner thereby reducing the mean squared error of the final model. In the paper, the author had collected real time data using weather API and created three layers in the architecture. The first layer consisted of data collection in which the data was in its raw form, then the second layer consisted of transforming the data to required format and the last layer consisted of implementing models to generate insights and results from the data. In the paper, the data was collected from the Meteorological Department and Flood Forecasting Commission in which each sample contains required parameters (such as rainfall level, precipitation etc.) to detect the flood threat. The training data was sliced to create classification outcomes and detect the situation where floods occurred or not with respect to rainfall level, precipitation etc. The model was implemented to generate results on an hourly basis and a fuzzy logic algorithm was implemented to tag the outcome as ‘danger’ or ‘safe’ so that concerned authorities can take

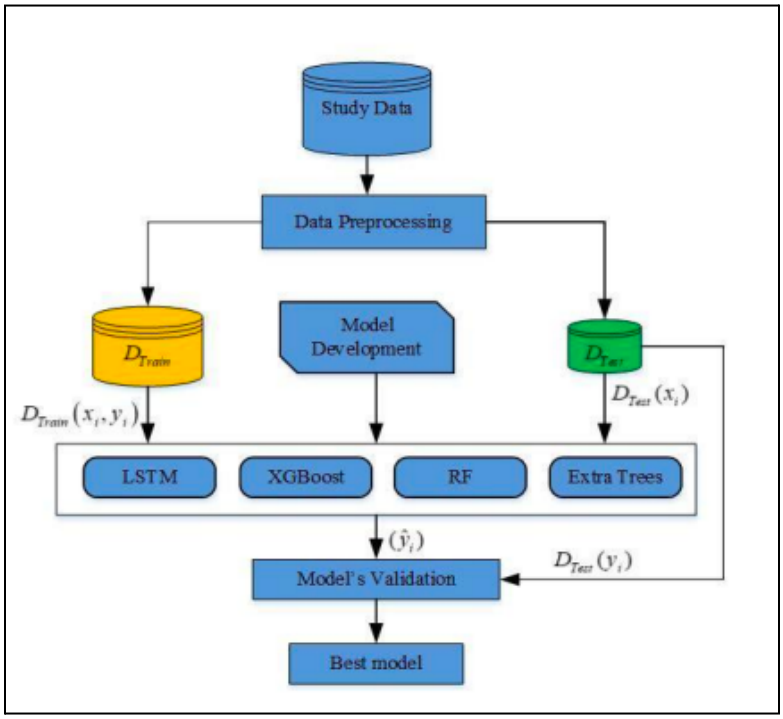
required measures. Lastly, the author concluded the gradient boosting model was robust, highly versatile and efficient in detecting flood threats in real-time scenarios. In future research they aim to expand the number of parameters in real time application as these results will help in getting an early warning.

Nti et al. (2021) mentioned that the environmental threat due to floods is rising significantly. Hence, in the paper the authors aimed to implement XGBoost model (extreme gradient boosting), random forest and extremely randomized trees, LSTM (long short term memory) and the best model was selected as the prediction model. They have measured the performance of models using multiple statistical performance evaluators. The dataset used in the paper contains rainfall details in Ghana and rainfall level was the main feature taken into consideration. Figure 2 shows the four steps in the model framework. Firstly, data cleaning was done by removing outliers, inconsistency and replacing missing values with the average values. Also, the data was normalized in the range (0,1) using MinMaxScaler and then the dataset was split into training (80%) and testing set (20%). The training set was also partitioned into a validation set (15%). LSTM is a special version of RNN (recurrent neural network model). LSTM has the capability to learn from long term dependencies. It consisted of three layers: input layer, hidden layer and output layer. In the paper, the model consisted of two dense layers with activation function set as RELU and learning rate was set at 0.0001. The XGBoost model was used as they were effective and efficient in classification and regression. Difference between random forest model and extreme boosting model was that in RF model trees were built independently whereas in XGboost model tree was added to complement an existing built tree. The XGBoost model was set at learning rate 0.01 with the number of estimators set to 3. On the other hand, RF model was set with 150 estimators and a criterion to measure information gain

was calculated using gini index. Lastly, the ET (ensemble learning technique) model was used to combine the results of multiple correlated decision trees. It was quite similar to RF model but differed in constructing the decision trees.The ET model was set using 150 estimators with criterion set to gini and min_sample_split was set at 2. Ten-fold cross validation was used to train the model and evaluate the performance. Figure 3 shows the comparison of models performance on different parameters.Finally, after comparing the performance of models with respect to RMSE (Root Mean Squared Error), R2 (R square)and MAE (Mean Absolute Error). The best result was shown by the XGboost model with respect to having MAE (around 0.0028), MSE (which was 0.0001), RMSE (approximately 0.0089), R2 (around 0.9945).

Figure 2

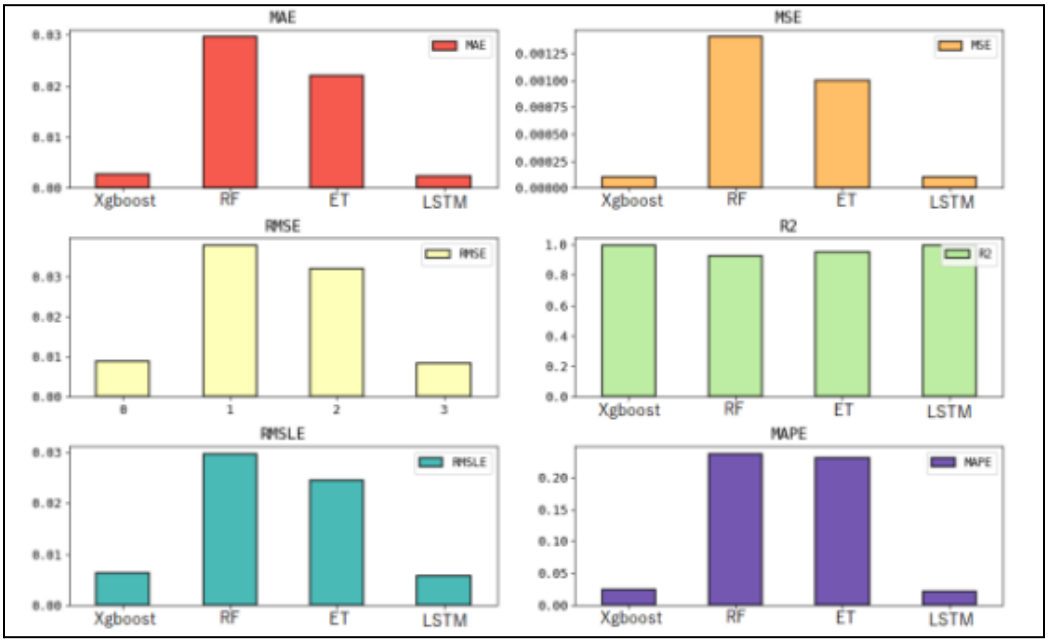
Model Framework



Note. Reprinted from "Enhancing Flood Prediction using Ensemble and Deep Learning Techniques", by Nti, Boateng, Bawah, 2021, *22nd International Arab Conference on Information Technology (ACIT)*, p.5.

Figure 3

Comparison of Model Performance



Note. Reprinted from "Enhancing Flood Prediction using Ensemble and Deep Learning Techniques", by Nti, Boateng, Bawah, 2021, *22nd International Arab Conference on Information Technology (ACIT)*, p.7.

One of the models that was selected for the analysis was Support Vector Machine (SVM). Sahoo et al. (2021) showed a comparison in model performance for various Artificial Neural Network (ANN) models like radial basis function neural network (RBFNN) and feed forward back propagation neural network (FFBPNN), single SVM model and SVM combined with Firefly algorithm. They used the data of river flow for two stations – Dholai and Silchar stations from Assam for flood prediction. SVM finds a hyperplane between N number of features to distinctly classify the data. They used the Firefly algorithm (FA) in conjunction with SVM. The Firefly algorithm was used to select optimal parameters for SVM models. This was done by first initializing the parameters of firefly. The SVM was then trained and the fitness function for firefly was calculated. As per the FA, the attractiveness between fireflies was directly

proportional to the distance between them. Thus, the fireflies were ranked according to their attractiveness. If the peak results were obtained for these parameters, they were considered as optimum and were used with the SVM model. Otherwise, iteration of the cycle would be conducted till the Peak results were obtained. The performance for these models were evaluated from the Coefficient of Determination (R^2), Mean Square Error (MSE) and Root Mean Square Error (RMSE) values which have been summarized in Table 2. It was observed that SVM with a Firefly algorithm (SVM-FF) showed highest accuracy as the values for MSE, RSME was less and the R^2 value was the highest (> 98%). However, it was also observed that the simple SVM model outperformed the other two ANN models as the R^2 value for it was 94.78 % and 94.55% for Silchar and Dholai regions respectively which was higher than FFBNN and RBFNN. Hence, it was concluded that using SVM model gives good results and if combined with Firefly Algorithm outperforms all the other models.

Table 2

Performance comparison for two watersheds

Station	Techniques	MSE		RMSE		R^2	
		Training	Testing	Training	Testing	Training	Testing
Silchar	FFBPNN	0.00371	0.00686	0.05076	0.03332	0.8688	0.8806
	RBFNN	0.00446	0.00698	0.04354	0.02898	0.8941	0.9081
	SVM	0.00453	0.00701	0.04365	0.02887	0.9385	0.9478
	RBF-FA	0.00468	0.00723	0.04354	0.02867	0.9619	0.9731
	SVM-FA	**0.00485**	**0.00754**	**0.04337**	**0.02856**	**0.9807**	**0.9812**
Dholai	FFBPNN	0.00378	0.00698	0.04399	0.03311	0.8693	0.8821
	RBFNN	0.00465	0.00709	0.04201	0.03091	0.8982	0.9095
	SVM	0.00477	0.00754	0.04189	0.03088	0.9397	0.9455
	RBF-FA	0.00491	0.00776	0.04144	0.03078	0.9629	0.9712
	SVM-FA	**0.00528**	**0.00792**	**0.04139**	**0.03064**	**0.9811**	**0.9818**

Note. Reprinted from "Prediction of flood in barak river using hybrid machine learning approaches", by Sahoo, A., Samantaray, S., & Ghose, D., 2021, *Journal Geological Society of India*, *Vol.97*, 186–198.

Shada et al. (2022) used SVM and hybrid wavelet-SVM for hourly flood predictions. As previously discussed, SVM divides the data points with linearly separable patterns into different groups with a hyperplane. However, when the data points were not separable linearly, Kernel mapping can be used which changes this non-linear data representation into linear high-dimensional feature space. The parameter selected greatly affects the performance of the kernel function. They used K fold cross validation as SVM may be affected by overfitting due to the bias in the training data. Here, the training dataset was divided into k subsets which have equal instances. One subset was kept for validation whereas other k-1 subset were used for training. This process was repeated k times after which the average for performance from validation sets is taken. This was the final performance of the k-fold model. For Hybrid wavelet SVM (WSVM) method, a mother wavelet was used which denoises the input rainfall and water level data. This input signal produced after denoising was given as an input to the SVM model for predictions. SVM and WSVM models were created for predicting the water levels for one, three and six hour lead time. The performance of SVM and WSVM was compared with hybrid wavelet-artificial neural network (WANN) which was summarized as in the Table 3. It can be observed that RSME was the least and R^2 was the most high for WSVM models. They concluded that hybrid wavelet SVM models were better for predictions in comparison to single SVM models. Also, the accuracy for WSVM was better than WANN as SVM has better generalization ability in comparison to ANN thus reducing overfitting. For future, other types of wavelet transforms can be explored and combined with SVM to see if there were any changes in the performance metrics.

Table 3

Performance measures for various SVM models and WANN model

Performance measures	Lead time								
	1h			3h			6h		
	SVM	WSVM	WANN	SVM	WSVM	WANN	SVM	WSVM	WANN
E5									
RMSE (m)	0.02	0.02	0.03	0.05	0.05	0.04	0.09	0.09	0.12
R^2	0.99	0.99	0.98	0.91	0.92	0.97	0.75	0.76	0.60
NSC	0.99	0.99	0.97	0.91	0.92	0.96	0.75	0.76	0.54
Dev (%)	0.72	-0.81	1.82	1.48	1.57	1.81	-0.07	0.6	-0.38
Dep (h)	1	1	0	2	2	0	5	5	5
E12									
RMSE (m)	0.02	0.02	0.04	0.04	0.04	0.06	0.08	0.07	0.09
R^2	1.00	1.00	0.97	0.98	0.98	0.96	0.91	0.93	0.86
NSC	1.00	1.00	0.97	0.98	0.98	0.94	0.91	0.92	0.86
Dev (%)	-0.01	-0.22	0	-0.99	-0.81	0.73	0.33	1.55	1.87
Dep (h)	0	0	0	9	2	0	9	8	0
E13									
RMSE (m)	0.25	0.24	0.12	0.45	0.45	0.18	0.59	0.59	0.50
R^2	0.93	0.93	0.98	0.76	0.76	0.90	0.55	0.55	0.60
NSC	0.93	0.93	0.97	0.73	0.73	0.94	0.46	0.47	0.46
Dev (%)	-0.35	0.17	-4.05	4.6	5.35	-6.97	7.40	8.27	-13.80
Dep (h)	0	0	1	-2	-2	1	1	0	0
E15									
RMSE (m)	0.08	0.07	0.15	0.23	0.22	0.15	0.37	0.38	0.37
R^2	0.98	0.99	0.98	0.85	0.87	0.93	0.60	0.59	0.59
NSC	0.98	0.99	0.97	0.85	0.86	0.93	0.58	0.56	0.59
Dev (%)	1.33	0.95	-1.76	2.78	2.69	-4.85	-0.04	0.66	6.46
Dep (h)	0	0	0	1	1	0	5	4	2

Note. Reprinted from "Hourly Flood Forecasting Using Hybrid Wavelet SVM", by Shada, B., Chithra N.R., & Thampi.S.G. 2022, *Journal of Soft Computing in Civil Engineering*, *6*(2), 1–20.

Jabari et al. (2020) stated in their research that, because of global warming and climate change the most destructive natural hazards flood is rapidly growing. The significant challenge in flood mapping was to provide accurate estimation of flood extent and damage amount in affected areas. Machine learning algorithms and statistical methods have been increasingly utilized for the generation of flood susceptibility maps where the algorithms use different factors referred to as conditioning factors to generate flood susceptibility maps. The major issue was the number of conditioning factors and its complexity, where it was possible to provide machine learning algorithms with more conditioning factors and achieve better results. The researchers

have used the Random Forest machine learning algorithm to build a flood model which helps in predicting the probability of a pixel being flooded or not. Here a total of 12 conditioning factors such as altitude, slope, aspect, distance from the river, land-use/cover, TWI, TRI, SPI, curvature, plan curvature, profile curvature, and HAND were selected for flood mapping using Random Forest. The conditioning factors considered for this research were ordinal and nominal where, for a better implication of Random Forest Algorithm, all ordinal factors were normalized from 0 to 1. In generating an accurate flood model, the precision of the data used has a very high impact on it. Sample points were collected from site visits and Sentinel-2 satellite images taken during the time of flood. Generated 740 flooded and non-flooded samples by ground truth data and visual inspection of the NDWI layer. These were randomly divided into training (70%) and testing (30%) points. Random Forest algorithm was implemented and trained for different scenarios. A probability map (0 to 1) was generated and classified into 5 classes for each implementation of Random Forest algorithm to different combinations of factors. Only high and very high classes were considered as flooded areas. Based on Random Forest algorithm analysis the distinguishing between flooded and not flooded areas was done using several different combinations of conditioning factors. Table 4 and Table 5 show two different test scenarios. Considering the various conditioning factors used in this research paper, the major factors contributing towards flood mapping in terms of highest accuracies in both scenarios were altitude or HAND model, slope, aspect, distance from river, and land- use/cover as shown in figure 4. The research involved experiments on comparing several combinations of conditioning factors and analyzed to find the combination that provides the flood model using Random Forest algorithm in accurate prediction of flood. The experiment results have revealed that prediction accuracy will be

degraded by having correlated conditioning factors. Also, the accuracy of predictions does not increase by adding extra conditioning factors.

Table 4

Combinations of different conditioning factors implemented in Scenario 1

Scenario 1	Conditioning Factors
1-a	Altitude-Slope-Aspect- Distance- Land-use/cover
1-b	Altitude-Slope-Aspect- Distance- Land-use/cover -TWI
1-c	Altitude-Slope-Aspect- Distance- Land-use/cover -TWI-TRI
1-d	Altitude-Slope-Aspect- Distance- Land-use/cover -TWI-TRI-SPI
1-e	Altitude-Slope-Aspect- Distance- Land-use/cover -TWI-TRI-SPI-Curvature
1-f	Altitude-Slope-Aspect- Distance- Land-use/cover -TWI-TRI-SPI-Curvature-Plan Curvature
1-g	Altitude-Slope-Aspect-Distance-Land-use/cover-TWI-TRI-SPI-Curvature-Plan Curvature-Profile Curvature
1-h	Altitude-Slope-Aspect-Distance-Land-use/cover-TWI-TRI-SPI-Curvature-Plan Curvature-Profile Curvature-HAND
1-i	Altitude-Slope-Distance- Land-use/cover -TWI-TRI-SPI-Curvature-Plan Curvature-Profile Curvature-HAND
1-j	Altitude-Slope-Distance- Land-use/cover -TWI-SPI-Curvature-Plan Curvature-Profile Curvature-HAND
1-k	Altitude-Slope-Distance- Land-use/cover -TWI-Curvature-Plan Curvature-Profile Curvature-HAND
1-l	Altitude-Slope- Distance- Land-use/cover -Curvature-Plan Curvature-Profile Curvature-HAND
1-m	Altitude-Slope-Distance- Land-use/cover -Curvature-Plan Curvature-HAND
1-n	Altitude-Slope- Distance- Land-use/cover -Plan Curvature-HAND
1-o	Altitude-Slope- Distance- Land-use/cover -HAND
1-p	Altitude-Distance- Land-use/cover -HAND

Note. Reprinted from "Flood Mapping using Random Forest and Identifying the Essential Conditioning Factors; A Case Study in Fredericton, New Brunswick, Canada", by Jabari, S., McGrath, H., & Coleman, D. 2020 *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* V-3, 609-615

Table 5

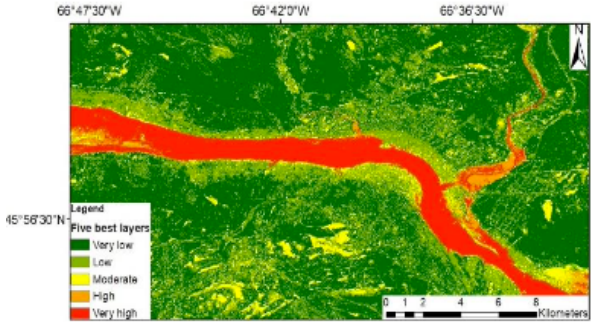
Combinations of different conditioning factors implemented in Scenario 2

Scenario 2	Conditioning factors
2-a	Slope-Aspect-Distance- Land-use/cover -HAND
2-b	Aspect-Distance- Land-use/cover -TWI-HAND
2-c	Aspect-Distance- Land-use/cover -TRI-HAND
2-d	Aspect-Distance- Land-use/cover -SPI-HAND
2-e	Aspect-Distance- Land-use/cover -Curvature-HAND
2-f	Aspect-Distance- Land-use/cover -Plan Curvature-HAND
2-g	Aspect-Distance-Land-use/cover-Profile Curvature-HAND
2-h	Altitude-Aspect-Distance- Land-use/cover -TWI
2-i	Altitude-Aspect-Distance- Land-use/cover -TRI
2-j	Altitude-Aspect-Distance- Land-use/cover -SPI
2-k	Altitude-Aspect-Distance- Land-use/cover - Curvature
2-l	Altitude-Aspect-Distance- Land-use/cover -Plan Curvature
2-m	Altitude-Aspect-Distance- Land-use/cover -Profile Curvature

Note. Reprinted from "Flood Mapping using Random Forest and Identifying the Essential Conditioning Factors; A Case Study in Fredericton, New Brunswick, Canada", by Jabari, S., McGrath, H., & Coleman, D. 2020 *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* V-3, 609-615

Figure 4

Random Forest output using altitude, slope, aspect, distance from the river, and land-use/cover



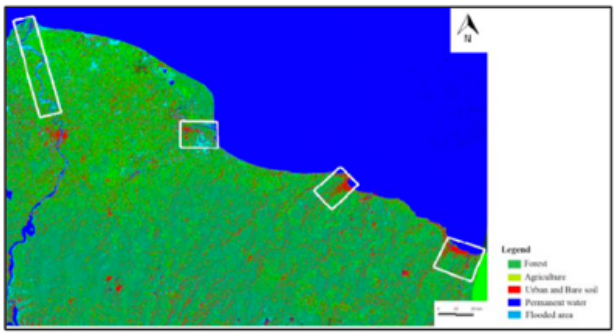
Note. Reprinted from "Flood Mapping using Random Forest and Identifying the Essential Conditioning Factors; A Case Study in Fredericton, New Brunswick, Canada", by Jabari, S., McGrath, H., & Coleman, D. 2020 *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* V-3, 609-615

Kocaman et al. (2020) said that flood events' frequency in recent years have been increasing and it was mostly due to climate change. Thus, flood mapping was essential for flood modeling and can be performed using the data of optical and microwave satellite sensors. Although the flood assessments before, during and after the event have been done based on the optical imagery-based flood analysis methods, they had the limitation of cloud coverage. The increase in availability of temporal and spatial resolution of SAR satellite sensors has made it popular in data provision for flood detection. But it requires a high level of expertise in processing them and visual interpretation of data is difficult. Therefore, the study aimed at application of the fusion approach for Sentinel-1 SAR and Sentinel-2 optical data for flood extent mapping where features obtained from Sentinel-1 and Sentinel-2 processing results were fused in random forest supervised classifiers. The methodology used to obtain and evaluate the classified result to detect flooded areas include pre-processing of Sentinel-1 and Sentinel-2 data. These two data were combined into a fusion dataset and, for predicting the classes, a Random Forest classifier was applied. For pre-processing Sentinel-1 images, from DS1 and DS2 data, appropriate sub-swath and bursts were selected using TOPSAR-split. Split IW images were merged using S-1 slice assembly and then applied for a precise orbit date. Following which radiometric calibration was done for quantitative analysis of SAR data obtained at different times. Later, to Sentinel-1 dual-pol data, a 2x2 covariance matrix was applied where co-pol information was represented by diagonal elements and cross-pol information by off-diagonal elements. Finally, applying Range Doppler Terrain Correction algorithm, the images were terrain-corrected and for geometric correction, bilinear interpolation resampling techniques and 1 arc-second spatial resolution were used. Training areas for classification were selected over true color combinations from DS3 sentinel-2 images. These images involve flood areas. Cloud masks

were applied to these images due to the presence of cloud effect during the time of the flood. DS4 images were used in the production of MNDWI and NDVI bands. Using the formula NDVI = (NIR-Red)/(NIR+Red), NDVI computation was performed. But in residential areas, this might yield overestimation of water classes and hence MNDWI was used for DS4. MNDWI = (Green-SWIR)/(Green+SWIR). Sentinel-1 SLC images used were dual-polarization and for the polarimetric decomposition, alpha (H/A/α) method was used. For the next step, Geometric differences were removed from the images, and they were stacked together. RF classification method was applied to decide the flooded areas. Using the RF method, a land cover map with four classes including flooded surfaces was generated in SNAP software. Classification results of the whole area are shown in figure 5. The results obtained from the Random Forest method were evaluated for the four sub-parts of study area as shown in Figure 6. Based on the initial investigation of the results, urban and bare soil areas could not be separated sufficiently visually. And hence four classes were combined into 2 new classes as shown in Figure 6. Further, to perform the in-depth investigations the Terme region (ROI-2 in Figure 6) was subjected to three different scenarios. The 3 different scenarios are described in Table 6. For all three classifications the same training data, which existed in the selected region was used. The classification results are presented in Figure 7. The most accurate classifications were obtained by fussing the datasets as per the results obtained. The proposed method of feature level fusion methodology in combination with random forest method can be used for flood mapping. The free availability of Sentinel-1 SAR and Sentinel-2 optical datasets along with the SNAP tool of ESA provides the means for this purpose, although it requires some level of expertise for applying the processing algorithm. Ground data sampling and field investigations are the planned future work.

Figure 5

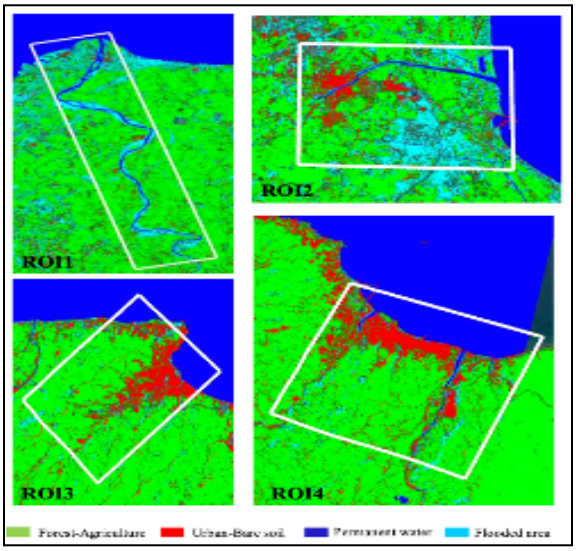
Result of the flood analysis produced for the whole study area



Note. Reprinted from "A Fusion Approach for Flood Mapping using Sentinel-1 and Sentinel-2 Datasets", by Kocaman, S., Nefeslioglu, H. A., & Gokceoglu, C. 2020 *ISPRS The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII–B3*, 641–648.

Figure 6

Zoomed views of sub-areas



Note. Reprinted from "A Fusion Approach for Flood Mapping using Sentinel-1 and Sentinel-2 Datasets", by Kocaman, S., Nefeslioglu, H. A., & Gokceoglu, C. 2020 *ISPRS The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII–B3*, 641–648.

Table 6

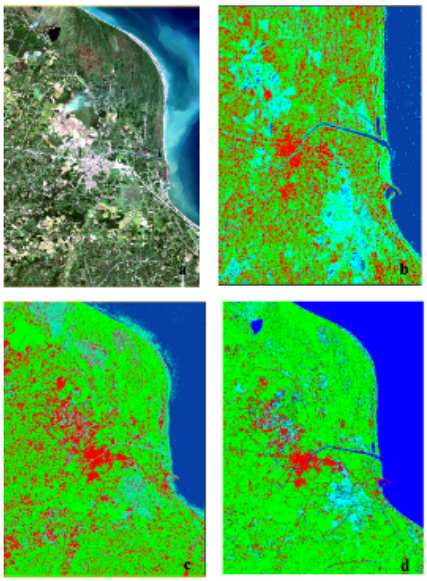
Scenarios applied for classification over ROI-2

Scenario	Band Combination
I	C11 Intensity+C22 Intensity
II	C11 Intensity+ C22 Intensity+ NDVI+ MNDWI
III	C11 Intensity+ C22 Intensity+ NDVI+ MNDWI+Sentinel-2 bands

Note. Reprinted from "A Fusion Approach for Flood Mapping using Sentinel-1 and Sentinel-2 Datasets", by Kocaman, S., Nefeslioglu, H. A., & Gokceoglu, C. 2020 *ISPRS The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII–B3*, 641–648.

Figure 7

Results of flood analysis produced for the ROI-2 (a: Sentinel-2 MSI image in true colour combination, b: Scenario I classification result, c: Scenario II classification result, d Scenario III classification result).



Note. Reprinted from "A Fusion Approach for Flood Mapping using Sentinel-1 and Sentinel-2 Datasets", by Kocaman, S., Nefeslioglu, H. A., & Gokceoglu, C. 2020 *ISPRS The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII–B3*, 641–648.

Clustering is a powerful unsupervised machine learning model. It is used popularly among various prediction and detection tasks. Sanjay et al.(2012) work with incremental k-means clustering model to predict the weather of West Bengal, India. The model took help of an air pollution database for forecasting the weather. The air pollution database consisted of parameters like Carbon Dioxide, Respirable particulate matter, Sulphur Dioxide and Nitrogen Oxides. The database was a dynamic database (i.e data added frequently) hence, incremental k-means clustering. Four different clusters were obtained. The first cluster represented smoggy weather, dust and presence of fly ash due to high amounts of RPM present. The second cluster indicated hot, dry and smoggy days due to Nitrogen Oxides. Hot, smoggy and humid days due to Carbon dioxide were indicated by the third cluster. The fourth cluster represented hot,smoggy days with high chances of acid rain due to the effect of Sulphur Dioxide. The accuracy of the model was measured to be approximately equal to 83.3%. Below are the images of pollution data and the obtained weather forecasting using incremental k-means.

Table 7

Pollution data with respective dates

30/9/2009	188	100	7	67
.....	.....	.....	.....	.....
1/1/2010	200	150	12	107
2/1/2010	220	160	13	110
.....	.....	.....	.....	.....
1/3/2010	260	170	14	105
2/3/2010	270	175	14	112
.....	.....	.....	.....	.....
1/6/2010	190	145	16	120
2/6/2010	200	155	12	118
.....	.....	.....	.....	.....

Date	*CO_2*	*RPM*	*SO_2*	*NO_X*
1/9/2009	66	27	5	31
2/9/2009	27	83	5	36
3/9/2009	88	30	5	35
4/9/2009	98	29	5	35
5/9/2009	74	28	5	33
.....	.....	.....	.....	.....
28/9/2009	116	43	6	52
29/9/2009	125	53	6	60

Note: Reprinted from "Weather Forecasting using Incremental K-means Clustering" by Chakraborty, S., Nagwani, N.K., & Dey, L. (2014). ArXiv, abs/1406.4756.

Table 8

Weather prediction obtained using incremental k-means

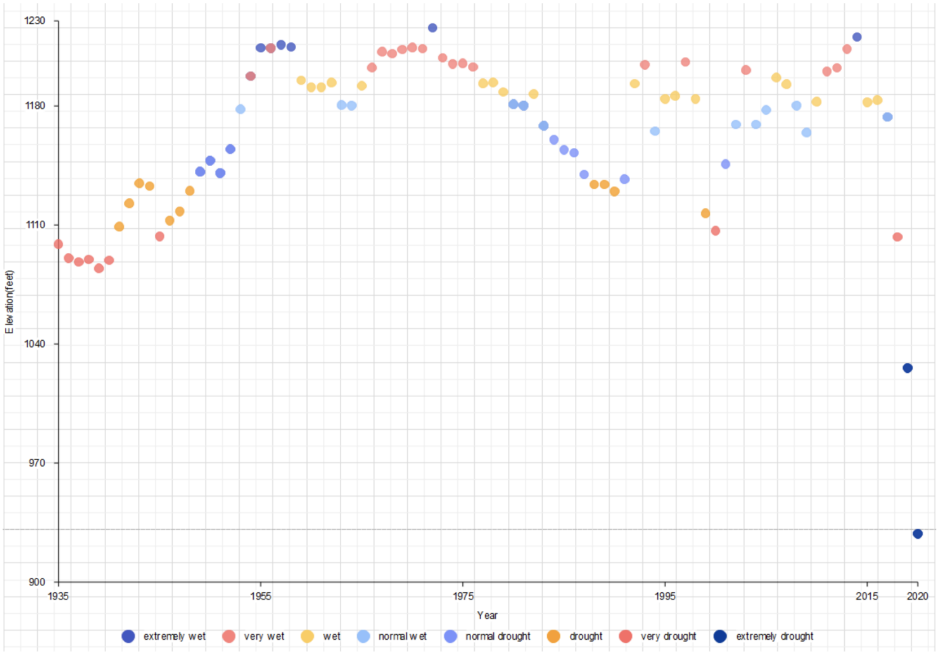
Date	*New data inserted into*	*Weather Category*
1/9/2009	Cluster2	Hot, dry and smogy
2/9/2009	Cluster3	dusty, fly ash, smogy, fog, Mist
3/9/2009	Cluster2	Hot, dry and smogy
4/9/2009	Cluster2	Hot, dry and smogy
……..	……..	……..
28/9/2009	Cluster3	dusty, fly ash, smogy, fog, Mist
29/9/2009	Cluster3	dusty, fly ash, smogy, fog, Mist
30/9/2009	Cluster4	hot, smogy and humid
……	……..	……..

Note: Reprinted from "Weather Forecasting using Incremental K-means Clustering" by Chakraborty, S., Nagwani, N.K., & Dey, L. (2014). ArXiv, abs/1406.4756.

Lake Mead, the largest reservoir of the United States, is located on the Arizona-Nevada border. It is also known for its water shortage issue. Chujie Shen et al. (2022) worked on identification of factors resulting from the low elevation of the lake. The scale of drought was divided into 8 different categories. K Means was used to evaluate and label the respective categories. Later, fitting functions were used to find the relationship of parameters like surface area, depth and volume to the elevation. Expected volume can be determined with the help of volume change, and using the obtained value of expected volume, elevation can be calculated. Also, evaluation of the proposed wastewater recycling model was proposed and necessary parameters for determining the pros and cons of the model were provided. The results obtained were not the most ideal as assumptions of the environment (like wind speed, flatness of the lake surface, humidity) were taken. Below is a scatterplot showing clusters of various drought levels.

Figure 8

Scatterplot illustrating clusters of different drought types



Note: Reprinted from "Prediction Model of Lake Water Volume Based on K-Means" by Chujie Shen, Hongwei Li, Mengqi Chen, Kairui Zhu, and Fan Wang (2022). *In The 2022 5th International Conference on Electronics, Communications and Control Engineering (ICECC 2022). Association for Computing Machinery, New York, NY, USA, 57–62.*

Table 9

Comparison between different models studied for technology survey

Models	Type of Data	Data pre-processing technique	Approach	Performance metrics
SVM	Historical Rainfall data (numeric)	SMOTE	Single SVM model	RSME 0.05 R^2 0.91

Models	Type of Data	Data pre-processing technique	Approach	Performance metrics
SVM-FA	Monthly river flow data (numeric)	Not Specified	SVM combined with Firefly algorithm	RSME 0.04 R^2 0.98
WSVM	Historical monthly, daily Rainfall data (numeric)	SMOTE	Hybrid Wavelet input to SVM model.	RSME 0.05 R^2 0.92
XGBoosting	hydrological data (numeric)	Normalization using Min Max Scaler	Single XGboosting model with learning rate 0.01	RMSE: 0.0089 R2: 0.9945
Random Forest	Sentinel-1 SAR data & Sentinel-2 optical data	Sentinel-1 preprocessing: DS1 & DS2 data, TOPSAR-split, S1 slice assembly,Radiometric Calibration, C2 matrix generation, Polarimetric Speckle Filter, Terrain Correction Sentinel 2 preprocessing: DS3 & DS4 data, cloud masking	Random Forest classifier with fused Sentinel -1 & Sentinel -2 data	Classification results obtained through images and further analyzed visually. Fusing datasets yield most accurate results

Models	Type of Data	Data pre-processing technique	Approach	Performance metrics
Random Forest	Sample points collected through site visits, NDWI and sample points generated from Sentinel-2 satellite image	-	Random Forest algorithm implemented in Rstudio	97.57% overall accuracy and 95.14% Kappa coefficient
Incremental k-means clustering	Air Pollution dataset (dynamic in nature) of West Bengal, India for years 2009 and 2010	Removal of NaN values or other invalid data points	Incremental k-means clustering applied to means	Accuracy of approximately 83.3% achieved through the model
K Means Clustering	Factors influencing low elevation like water inflow, etc. are considered and used.	Removal of NaN values or other invalid data points	K means used to categorize drought levels and fitting functions implemented to determine relationships between parameters and to find elevation though volume	Assumptions regarding the environment like flat lake surface and others are made initially. The model can be used for general estimations but not for technical or serious purposes

1.5 Literature review

Tanim et al. (2022) developed various techniques to detect and predict the nuisance flooding that can occur in urban areas. This was done to minimize the damage caused due to the flooding. Ground data collected from the road closure reports along with Synthetic Aperture Radar (SAR) images extracted from Sentinel 1 satellite imagery were used for training various machine learning algorithms. They used three supervised machine learning classification algorithms namely Support Vector Machine (SVM), Maximum Likelihood Classifier (MLC) and Random Forest (RF) models and one unsupervised algorithm to detect floods in San Diego (California). Three supervised ML algorithms were used to compare and analyze the performance achieved from each one of the models whereas unsupervised algorithms used change detection (CD) approach combining it with Otsu algorithm, iso-clustering methods and fuzzy rules. The challenging task usually was to establish a threshold for water and non-water pixels for analyzing Sentinel 1 image for both supervised as well as unsupervised classification methods. It was observed that unsupervised models showed high precision values, recall values and F1 score as compared to the other three supervised models. Hence, they concluded that an unsupervised flood classification model was a more robust solution for detecting and predicting floods in urban areas as it required less data and computation time as compared to other models. This model could be further used by various urban areas for limiting the risks associated with floods.

Munawar et al. (2021) reviewed various research papers to track the recent developments in image processing and machine learning for managing disasters like flood detection and flood prediction. They analyzed existing technologies used for flood management in order to identify the gaps in these methods which would help to develop new models for better flood analysis.

They observed that the most commonly used techniques in image processing used for flood detection and prediction involved edge detection, segmentation and pixel analysis whereas models like ANN, SVM and WNN were mostly used when applying machine learning approaches. The images required for this analysis were mostly acquired using UAV imaging, remote sensing and SAR. Their study showed a lack of application of models combining both image processing and machine learning, also called hybrid models. Unsupervised machine learning, deep learning and reinforcement learning techniques can be further explored for flood management.

Jain et al. (2020) proposed that the accurate and early flood detection previously relied on hand-crafted functions to automatically identify water from the satellite images which lacked the accuracy and robustness. Hence to overcome the current limitations, experiments were conducted with MedialRval 2019 flood dataset to identify flood using tiered methodology combining water index like features with a deep convolutional neural network-based solution.Different water indexing techniques were explored and water index function was proposed with the use of Green/SWIR and Blue/NIR bands with VGG16, an existing deep neural network method. The experiments to detect flood in images shows that the proposed index was better than all other water index techniques when trained on the VGG16 network. To improve the performance of automatic flood detection, using an appropriate combination of water indices technique with deep CNN model to train provides best results when tested against labeled sentinel-2 high resolution images. In this case, Green/SWIR and Blue/NIR indices were used to reduce the impact of clouds and their shadow for shallow water detection with the VGG16 model performed best compared to other indices (NDWI, MNDW, AWEI). Further these experiments can be extended to get better flood mapping.

Moumtzidou et al. (2020) proposed that identifying and confirming the flood scenes in a timely manner due to increase in rainfall in urban areas was very crucial as it helps resolve the crisis and prepare for future instances. To identify the flood event in time-series, two successive in time Sentinel-2 images were compared by using pre-trained Deep Convolutional Neural Networks (DCNN) with different sets of three water sensitive satellite bands as input. The proposed method here was based on ML techniques, where the algorithm builds mathematical models based on training data to make predictions. Using pre-trained CNN, models were built to differentiate and tag the satellite images as either flooded or non-flooded. For this experiment, VGG-16, VGG-19 and ResNET-50 CNN models were used and then results were compared with image differencing, ratio differencing and change vector analysis which were commonly used baseline change detection techniques. The results showed that the performance of the neural network drastically improved with F-score of 62% with applied augmentation techniques to the proposed method when compared to traditional remote sensing techniques with F-score of 22%. Thus, it was concluded that the proposed DCNN method with applied augmentation to the training dataset was way better than standard remote sensing techniques. Further the DCNN models can be trained from scratch using satellite images without pre-trained weights.

Chamola et al. (2021) discussed screening, predicting, forecasting and tracing of a disaster and necessary recovery procedures. Major concerns like building a robust system, connectivity and provisioning of proper disaster relief services was discussed and looked at in the paper. Machine learning algorithms deal with multidimensional and large data, and these models work great especially for recognition and classification. Identifying trends is one popular term used along with these algorithms. These algorithms can be used to construct reliable models and detect outliers. The performance of ML algorithms tends to improve as the data size

increases. A detailed review of various technologies used in disaster and pandemic management and how ML algorithms (like SVM, Random Forest, K-Means, ANN) can be deployed and used by them for this end was another major contribution of the paper. Talking about the limitations, this model failed to sustain extreme weather conditions, and battery life turned out to be a drawback as well.

Sharma et al. (2021) worked with various machine learning classifiers like Decision Tree (DT), Random Forest (RF), etc. to categorize flood severity into four classes: Information, Advisory, Watch and Warning. The training datasets obtained from the Model of Models (MoM) were utilized for the classification. Model of Models integrated flood forecasting models along with the rainfall forecast generated to predict flood severity. The MoM was also used to obtain flood extent and flood depth with the help of SAR imagery. The primary goal was integration of large data from forecasting models, classifying severity scores and sending alert or warning messages to circulate among stakeholders without human intervention. Talking about the results, using ML classifiers along with MoM worked for automation of the entire process. The future work includes focusing on deploying adaptive ML algorithms given that MoM variables were generated daily.

Li. et al. (2016) mentioned in the paper that due to various factors such as weather, human activities and drainage basin's physical location, flood prediction process was extremely uncertain and complex. Hence, to overcome this issue the author developed a model to precisely forecast water and rainfall level in order to effectively create a dispatching method. For this, the boosting learning based flood forecasting model was developed to improve the accuracy and efficiency of flood forecasting. In this paper, the Wangjiaba station was chosen as the research subject, and flood data was taken for the period 1998-2000. The proposed model consisted of

boosting learning modules and data pre-processing algorithms. The data pre-processing algorithm consisted of extracting relevant features using nonlinear KPCA. The factors were selected on the basis of their correlation coefficient between input and output. Those with the higher coefficient (the threshold correlation coefficient was set at 0.2) were chosen as the input. The boosting model trained each SVM regression model as per different weights and then the prediction model was obtained by collating all sub-models to improve the overall accuracy. The overall accuracy was calculated by the relative error in the prediction results during training of each sub-model and the weights were determined by loss function and overall accuracy of the model. The author also compared this model with a single SVM model and it was observed that using the boosting method the MSE (mean squared error) was much smaller. Therefore, in the paper it was concluded that KPCA was one of the most efficient methods to extract features from non-linear data. By iteratively feeding data to the boosting model, the algorithm can identify different characteristics from the data which improves the overall efficiency of the forecasting model.

de Castro. et al. (2021) mentioned that extreme climate change is increasing the frequency of environmental disasters such as floods. Furthermore, the authors aimed to design an efficient flood prediction system for the urban areas as cities are developing significantly and an efficient flood prediction model was required to identify flood risk factors. Hence, to overcome this problem the authors proposed an efficient flood prediction system by using a combination of Machine Learning Classifiers (Random Forest Model) along with GIS. They developed a prediction tool to identify flood events in Lisbon. By combining both the algorithms, the model can identify sensible factors and risk indices for the occurrence of floods in cities. GIS model was used to identify areas of higher likelihood of being flooded under extreme weather

conditions. Hence, the hub spots were installed across the entire city along with flood history. Then the scores obtained from Random Forest Model and GIS model were combined to create a flood risk index. The two step process was able to effectively identify the risk factors by using geospatial analytics for the entire city. Therefore, in this paper it was concluded that combining ML classifiers with GIS was an efficient method to determine key predictors and factors responsible for flood. The authors implemented different ML classifiers such as SVM, KNN, Random Forest etc to compare the performance. Out of all these Random Forest had the highest accuracy (0.96) along with highest correlation coefficient of 0.77.

Lu & Feng (2010) developed a precise technique for flood forecasting as it is a very common natural disaster which affects the large areas of the community and is an enormous threat to human life and property. To deal with the pattern recognition, mathematical models or graphs were created based on the relevant historical data in the traditional flood forecasting methods. To identify the correct model, there was lots of research being conducted to find the relationship between independent variable x and the dependent variable y in [y = f(x)] but results were less satisfactory. Hence to solve this problem, they used artificial neural network (ANN) technology as an alternate method. According to them, estimation response of input and output along with mapping of the stimuli effects were obtained from the combination of nonlinear functions and they have proved to be more feasible in flood forecasting. They also mentioned that self-learning, self-organization, self-adaption, and fault tolerance were the advantages of this method. In this research, they used the peak stage of the upper and lower reaches station at Dadu River, China as the input and output variable respectively. Multiple variables in both input and output layers were allowed in ANN technology which was very important for flood calculations as the stage, discharge and hydrological variables were functions of many influential variables.

0.02% was the maximum error seen by them in this example which was very good for forecasting floods, and it would have been very difficult to obtain using the traditional approach. They also acknowledged the fact that there were still some problems for further study of forecasting flood and in the future the performance can be increased using neural networks.

Flood is the most destructive natural disaster, and it causes damage to life and property every year. Kia et al. (2011) in their study specifically aimed to develop a flood model using various flood causative factors to model and simulate flood-prone areas by using ANN techniques and geographical information system (GIS). This study was targeted to the southern part of Peninsular Malaysia. They mentioned that previously ANN methods used rainfall and runoff data as input and output of flood modeling with accounting for other factors. For this study they developed ANN model in MATLAB using seven flood causing factors and using GIS, remote sensing data & field survey to generate relevant thematic layers such as rainfall, elevation, flow accumulation, slope, land use and geology. They used ANN to produce water levels and GIS to construct flood maps. Coefficient of determination (R^2), the sum squared error, the mean square error, and the root mean square were used to measure the model's performance in this research. Based on the sensitivity analysis shown in this paper, elevation was the most important factor for flood mapping. Elevation has the highest weight values, $R^2 = 0.931$ followed by slope ($R^2 = 0.963$) and then land use ($R^2= 0.986$). To speed up the computation process, they have converted the thematic data layers into ASCII format. They analyzed the flood susceptibility map qualitatively using equal area classification schemes and developed the MLP prototype to apply different flood factors to model the flood, integrated with GIS, results in spiral form were presented. Further, this model can be integrated with a real-time warning system to reduce the flood damages significantly.

Jabari et al. (2020) mentioned that for two consecutive years in 2018 and 2019, the city of Fredericton, New Brunswick, Canada has been flooded. In their research study they used machine-learning algorithms to model the 2018 flood in Fredericton. But the major challenge using machine-learning algorithms was the number of features and its complexities as the input to generate acceptable results. For 12 different flood conditioning factors including slope, altitude, distance from river, aspect, terrain wetness index (TWI), land-use/cover, stream power index (SPI), terrain roughness index (TRI), curvature, profile curvature, plain curvature, and height above the nearest drainage (HAND), they analyzed the effect of several combination of these factors using Random Forest to model 2018 flood. They used Random Forest over other ML methods for its quick predictions, the capacity of handling unbalanced data, low bias, high dimensional data, and its robustness. It also predicted the importance of each variable. From the results of their research, it was concluded that using these 5 factors namely, Altitude or HAND, aspect, land-use/cover, slope, and distance from the river provides highest prediction accuracy of 97.57% and kappa coefficient of 95.14%. They also observed a few points such as there was no impact on the accuracy of prediction when extra conditioning factors were added, there was a decrease in accuracy of prediction when correlated layers were included and mainly both HAND and altitude were major factors and they both had the same effect on prediction accuracy.

Due to climate change, there was an increase in frequency of flood events in recent years. Kocaman et al. (2020) in their research study used the data of flood events in Ordu Province (Turkey) which occurred on August 8th, 2018, to map the full flood extent. They obtained Sentinel-1 SAR and Sentinel-2 optical datasets from the Copernicus Open Access Hub of the European Space Agency (ESA). Then these datasets were processed using SNAP tools which were freely provided by ESA. Further, they obtained the features from the processing results of

Sentinel-1 and Sentinel-2 which were fused in Random Forest supervised classifiers. Even though using Sentinel-2 optical data alone eases the training sample selection for flooded areas and provides better optical data of settlement area, these optical data prevent from mapping full extent of flood due to clouds and hence Sentinel-1 data was also used. In this study, different feature combinations were evaluated, and the results have been visually assessed. Based on the results, the proposed methods can be used to map the flooded area effectively. Further they suggest that their study can be expanded by including the field investigation data and ground data.

Prasad et al.(2021) mentioned in the paper that floods are one of the major threats in India which is causing damage to large numbers of flora and fauna and causing intense damage to the livelihood, infrastructure and property. To overcome this problem, machine learning is contributing significantly in creating cost effective solutions such as early warnings. In the paper, the author implemented multiple models to the time series data of daily runoff, weekly runoff and flood runoff of the Kaveri river, India. Flood dataset was taken for the years 1998-2018 of river Kaveri, India. Data pre-processing was done using statistical approaches i.e. replacing the missing values with the mean of the data. They have fetched the data from Kaggle which consisted of six features. These were reservoir name, flow data, inflow, present storage, reservoir level and outflow. They have predicted the future values of Kaveri river runoff using a logistic regression model. Major factors affecting the flood were inflow of reservoir and rainfall. Multiple ML models (which are logistic regression models, ANN, KNN) were implemented to predict the future values based on daily runoff, weekly runoff and flood runoff. Out of three models, Table 10 shows that logistic regression model had the highest accuracy i.e. 99.3%. On the other hand, rainfall prediction was done using ensemble techniques which includes Random

Forest Regressor, Decision Tree Regressor and KNN. Out of these, Table 11 shows that Random Forest regressor R-squared test scores were most accurate i.e. 1.00. Finally, the author concluded that using different models to gain insights for different factors such as rainfall level, flood runoff was highly efficient. In the future work, they aim to train this model on more rivers and reservoirs to provide an effective solution. They also aim to include more climatic features to improve accuracy.

Table 10

Performance of flood prediction algorithms

Algorithm	Accuracy
Logistic Regression	99.3%
Linear Discriminant Analysis	97.9%
KNN	97.0%

Note. Reprinted from "Ensemble Model-Based Prediction for River Management: A Case Study on River Kaveri and Coastal Karnataka", by Prasad, B. S., Shreya, S., Sinha, T., & Priya, S., 2021, *International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*

Table 11

Performance of rainfall prediction algorithms

Algorithm	R-Squared Test score
Random Forest Regressor	1.00
Decision Tree Regressor	0.826
KNN	0.761

Note. Reprinted from "Ensemble Model-Based Prediction for River Management: A Case Study on River Kaveri and Coastal Karnataka", by Prasad, B. S., Shreya, S., Sinha, T., & Priya, S., 2021, *International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*

Ghorpade et al. (2021) mentioned in the paper that the areas affected by the flood were at a great risk of losing human lives and infrastructure. They mentioned that rainfall is one of the most common causes of floods. In the paper, they had predicted that by 2050, around 450 million people would be badly impacted by the floods and the global flood risk index will increase by 187% because of climate change. They have implemented various ML models to compare the performance of each model in reducing flood risk. These are Decision Trees, SVM, Linear Regression. In the paper, the authors had used hydrological data and climatic variables for predicting water discharge time series. The multiple linear regression model was used to predict the time series of water discharge for a specific time period. Wind speed, precipitation, tide and temperature were used as features. The value of the coefficient of determination was in the range 91.25 to 93.2% which tells about the performance of the model to predict values. Likewise, SVM was used to forecast floods at Bird Creek catchment in Oklahoma by using rainfall and river flow as features. The coefficient of determination was approximately 99.3% and RMSE (root mean squared error) was around 0.244 meter. Furthermore, decision trees were used as they are robust and use boosting techniques to increase the accuracy of the model. In the paper, they implemented decision trees for classifying the flash floods in watersheds. Features that were taken into account were river density, altitude, ground slope, rainfall and stream power index. Lastly, the authors concluded that decision tree algorithms were most suitable for flood prediction as they have a RMSE of 0.216 with 94.3% accuracy.

Dai et al. (2021) aimed at developing a Machine Learning algorithm for predicting floods in the coastal areas from data collected using various Internet-of-Things devices and sensors. The weather data like direction and speed of wind, humidity, temperature and the hydrological data like depth of flood, height of the wave, height of tide was collected using these IoT devices.

Initially, the data was classified into 5 levels depending upon the level/intensity of flood depth with level 0 having lowest flood depth of less than 0.1 m whereas level 5 having the flood depth higher than 2.5 m. They then applied Linear Regression (LR), Lagrangian Support Vector Machine (LSVM), Quantum-Enhanced SVM (QSVM), RF and back propagation Neural Network (BPNN) models to train on the complete training set. They also designed an ensemble based Bayesian model (BMC-EL) which was integrated with base learners namely Back Propagation Neural Network (BPNN) and RF models which were trained only on a subset of the training dataset. They evaluated the accuracy for each of the models which was summarized in Table 12. It can be seen that all the models showed coefficient of determination (R^2) greater than 0.9 which means that these models captured the non-linear relationship of floods accurately. BMC-EL showed lowest MSE (0.0072), lowest MAE (528.4) and lowest RMSE (0.0847m) values as compared to the other models. Hence, it was considered to be the best prediction model.

Table 12

Performance metrics for ML models

Technique	MSE (m)	MAE (m)	RMSE (m)	R^2
LR	0.1597	2258.21	0.3996	0.9972
LSVM	0.1593	2627.34	0.3991	0.9943
QSVM	0.1178	1941.94	0.3432	0.9972
BPNN	0.1151	2071.19	0.3392	0.9870
RF	0.0247	976.47	0.1571	0.9559
BMC-EL	0.0072	528.40	0.0847	0.9799

Note. Reprinted from "Ensemble Learning Technology for Coastal Flood Forecasting in Internet-of-Things-Enabled Smart City", by Dai, W., Tang, Y., Zhang, Z., & Cai, Z. 2021, *International Journal of Computational Intelligence Systems*, *14*(1)

Another study by Razali et al. (2020) developed Bayesian network (BN), Decision Tree (DT), k-Nearest Neighbors (kNN) and SVM for flood risk predictions. They also used CRISP-DM methodology to divide the tasks into 6 stages for efficiently solving the business problem. They used historical numerical rainfall data (time series data) for their analysis. Firstly, the data was pre-processed using resampling to remove any imbalance from it. They used the Synthetic Minority Over-sampling Technique (SMOTE) technique wherein extra training data called as synthetic data was created to combat the imbalance in the data. They compared the performance of each model from precision, accuracy, f-measure and recall which have been summarized in the table 13. It can be seen that Bayesian Network showed the highest accuracy of 99.94% as compared to others. The precision, Recall and F-measure were also very high (0.99). However, all models showed good performance as their accuracy, Precision, Recall values and F-measure all were above 99%. For future works, an ensemble model can be implemented for further increase in the prediction accuracy.

Table 13

Performance metrics for various techniques

Techniques	Accuracy		Precision		Recall		F-Measure	
	Normal	Smote	Normal	Smote	Normal	Smote	Normal	Smote
Bayesian Network	99.94%	99.68%	0.999	0.997	0.999	0.997	0.999	0.997
Decision Tree	99.89%	99.92%	0.999	0.999	0.999	0.999	0.999	0.999
k-Nearest Neighbours	99.50%	99.73%	0.995	0.997	0.995	0.997	0.995	0.997
Support Vector Machine	99.50%	99.76%	0.995	0.998	0.995	0.998	0.995	0.998

Note. Reprinted from "Machine learning approach for flood risk prediction", by Razali, N., Ismail, S., & Mustapha, A., 2020, *IAES International Journal of Artificial Intelligence (IJ-AI)*, *9*(1), 73.

Razafipahatelo D. et al. (2014) proposed an automatic three step method based on clustering for flood detection. A Digital Elevation Model (DEM) was first utilized as background

data to identify areas with a high likelihood of flooding. Then, a technique known as kernel k-means was used to separate the moist and dry pixels. Finally, a non-linear clustering with a log ratio image was applied in the features space to separate the flooded pixels from the permanent water. The results obtained by this model were compared with that of the manual and color composite methods. The analysis indicated that the model works well for flood detection purposes.

Hongshi Xu et al. (2018) proposed an approach evaluating urban flood risk based on urban flood inundation model, k-means clustering and improved entropy weight method and was applied to flood risk zones in Haidian island using GIS technology. Based on the principles of being systematic, measurable, operational and universal, seven flood risk indicators were chosen to create the index system by merging the natural hazard index system and hydrological models. The results showed variance between entropy weights and AHP weights when subjective and objective weights were calculated using the AHP and entropy weight approach. Therefore, an improved entropy weight technique for combining AHP and entropy weight method was provided in this research. To create the flood risk map in the research region, a method combining the k-means cluster algorithm and improved entropy weight method was applied.

Table 14 summarizes the various models and theri respective performance metrics.

Table 14

Summary of the models and the respective performance metrics

Previous papers Authors	Models	Type of Data	Data pre-processing technique	Performance metrics
Dai et al.	LSVM	Hydrological data captured from IoT devices	Flood intensity classification and K-fold cross validation	RSME = 0.3991 R^2 = 0.9943
	QSVM			RSME = 0.3432 R^2 = 0.9972
	RF			RSME = 0.1571 R^2 = 0.9559
Prasad et al.	Random Forest Regressor	Rainfall level(numeric)	Used statistical methods to clean data	R2 = 1.00
Ghorpade et al.	Decision Tree	hydro data (numeric)	-	RMSE= 0.216; R2= 0.9945
Jain et al.	Deep Convolutional Neural Networks	MediaEval 2019 flood dataset-Sentinel-2 data (image)	-	F1 - 0.96 Kappa - 0.92

Previous papers Authors	Models	Type of Data	Data pre-processing technique	Performance metrics
Moumtzidou et al.	Deep Neural Networks	Sentinel-2 data (image)	-	F-Score 62%
Tanim et al.	RF	Data From road closure reports along with SAR images for flood detection	Image pre-processing done using SNAP tool for noise removal and filtering.	Precision ~ 0.5 Accuracy ~ 0.65
	SVM			Precision ~ 0.85 Accuracy ~0.85
	MLC			Precision ~0.75 Accuracy ~0.82
	Unsupervised (CD, iso-clustering, fuzzy rules)			Precision ~ 0.8 Accuracy ~ 0.84
Razafipahatelo D. et al.	Digital Elevation Model (DEM), kernel k-means, non - linear clustering	SAR Images of flooded regions	-	Manual result = 807.63 ha Log Ratio in fs = 844.74 ha Ratio in fs = 1429.27 ha Ratio = 682 ha

Previous papers Authors	Models	Types of Data	Data pre-processing technique	Performance metrics
Xu H. et al.	Urban flood inundation model, k-means clustering and improved entropy weight method	Information about river, rains and drainage, previous storm records and physical parameters like slope, distance from river etc,	Removal of invalid data	High risk zones overlap when compared with the references therefore making it a feasible approach (but limited by data provided).

2. Data and Project Management Plan

2.1 Data Management Plan

Data was collected from three different sources. The imagery data was collected from IMD website. No licensing nor cost was required for collecting data from this website. The rainfall data was obtained using OpenWeather API by subscribing to the developer plan for OpenWeatherMap API. It usually costs 180 USD/month for subscription but it was free for students. The tidal data was acquired using Marea API and is free for usage. Data was collected for an approximate duration of one month. As the rainfall data on IMD website was updated every hour, the data collected for one month was included for analysis which consisted of almost 1270 images and corresponding to the hours and dates of these images, rainfall and tidal data was collected. Size of the data collected was approximately five GB. Data was collected from open-source API and IMD website. So there were no legal limitations to the collection or usage of the data.

The data was stored on a local machine and was accessed only by team members.The updates on the records were tracked by saving it as different versions for retrieval in any case of accidental deletion. As the data was collected from three different sources, three different folders were created on the local machine named “data270project” and the folder names for INSAT images, OpenWeather API and Marea API were “Insat_Images”, “Tidal_Data”, “Rainfall_Data” respectively. The INSAT images were in the jpeg format whereas the tidal data and rainfall data was in JSON format. Python libraries, pandas and data frames were used to convert unstructured data(JSON format) into structured data. The transformed data was stored again on a local machine by creating two different folders (named as “tidal_clean_data”,“rainfall_clean_data”) and the files were stored in .csv format. Furthermore, ETL was done using python libraries

which included data wrangling such as removing the null values, detecting the outliers and scaling the data. Following which implementation of the Machine Learning Models (ie., K-Means clustering, SVM, XGBoost, Random Forest Model) was done in the Jupyter Notebook.

Data was stored on a local machine which allows access to data only by team members and contributors for pre-processing and modifying. Once the project was completed the detailed report was posted on Github for free access to the public where data manipulation was restricted to admin roles and contributors only.

The files content included the raw data collected from Marea API, OpenWeather API and INSAT images extracted from the Indian meteorological website, processed data (data preprocessing done with the help of python libraries) and the final results obtained.
The data obtained from OpenWeatherAPI and Marea API was in the .json format. All the collected data was stored on a local machine and was accessed only by team members. The whole process of data exploration, data collection and data wrangling was divided and scheduled equally using project management tool ClickUp.

2.2 Project Development Methodology

Software development life cycle is the model which provides various stages to successfully completing the project. It consists of several types such as CRISP-DM, Waterfall and many more. The project used the CRISP-DM model of SLDC. It stands for Cross Industry Process of Data Mining. It is one of the industry proven types of model which guides the data mining efforts within the project. There are basically six phases of CRISP-DM such as business understanding, data understanding, data preparation, modeling, evaluation and deployment. The reason behind using the model is such that it is considered as one of the most appropriate

methods used for data mining in regards with meeting the requirement of the industrial projects undertaken. Each phase has been explained in detail in the following sections.

2.2.1 Business Understanding

The first phase of CRISP-DM is Business Understanding, where a project problem statement was defined, to address with the relevant solution. It helped in understanding the problem at hand with respect to its context, scale, and intensity. Due to global climate changes, natural hazards have been increasing, where floods contribute to the most part of it causing human loss and property damage. People are affected three times more by floods when compared to tropical cyclones and it is due to lack of a proper flood management system. Hence flood forecasting becomes very critical in saving human lives and avoiding property damage. To address this issue and forecast & predict the floods project different machine learning algorithms were implemented in the research project. Based on the research papers during the literature survey it was found that supervised machine learning algorithms can face challenges in predictions due to complexity of data and the amount of time required for computation. So, the main goal of the research project was to implement both supervised and unsupervised machine learning models on satellite imagery, tidal and weather data, and then based on evaluation of different models in comparison with performance metrics choose the best model to predict floods with the better accuracy.

2.2.2 Data Understanding

The second phase of the CRISP-DM is the data understanding. The data was collected from three different sources and were of different types. The INSAT images for India were obtained from the Indian Meteorological website which had the information about the water vapor, fog, rainfall, etc. These images were cropped to focus the analysis only for the state of

Kerala.The information about the amount of rainfall (cm) for Kerala was extracted from an API which was in JSON format. The data about the tidal length and tidal status for Kerala was also obtained from another API which was also in JSON format. The data which were of different formats was then stored in three separate folders on the local machine.

2.2.3 Data Preparation

Next phase is the data preparation, the data collected were in Json format.This was transformed into structured format using pandas “jsonify” function which converted Json format into rows and columns for further analysis. Data wrangling i.e. removing or replacing the null values, detecting the outliers, scaling the data in the range(0, 1) etc. was done using python on Jupyter Notebook. The cleaned, transformed and normalized data was then split into training and testing. The dataset used was time series based and hence the first 80% of the records in continuity were used for training and the last 20% of the data set was used for testing.

2.2.4 Modeling

In the modeling phase of CRISP-DM methodology, machine learning models for Flood Detection and Prediction were built. Worked on both supervised and unsupervised machine learning models. Based on the data used, SVM, Random Forest, XGBoost and K-means Clustering were used for model building. With the help of the cleaned data obtained from the previous stages of CRISP-DM, models were built. In this step,the training data set obtained after the split was used for model building. Say, for instance, in the case of SVM, datasets obtained from splitting for training and testing sets based on time series split were used. In SVM, SVC from sklearn.svm was imported and used for model building using python on Jupyter Notebook following which building a classifier and fitting it accordingly was performed. Once all models

were built and ready to be executed, evaluated them for their performance based on confusion matrix, accuracy, recall and F1 scores along with ROC curves for each of the models built.

2.2.5 Evaluation

With obtained performance metrics and ROC curves from the Modeling phase of CRISP-DM and other factors such as training time, time taken for prediction and others, the evaluation of the models built for the use case i.e. Flood Detection and Prediction was performed. For example, prioritizing fast prediction as time matters in case of disaster management. Looking at models that generalize well. Based on these parameters and other business needs, choosing the model that best fits the use case of flood prediction. Cross validation techniques were also used to obtain accurate performance metrics and to validate the models built. Based on this step, the deployment phase was initiated to meet any left out requirements.

2.2.5 Deployment

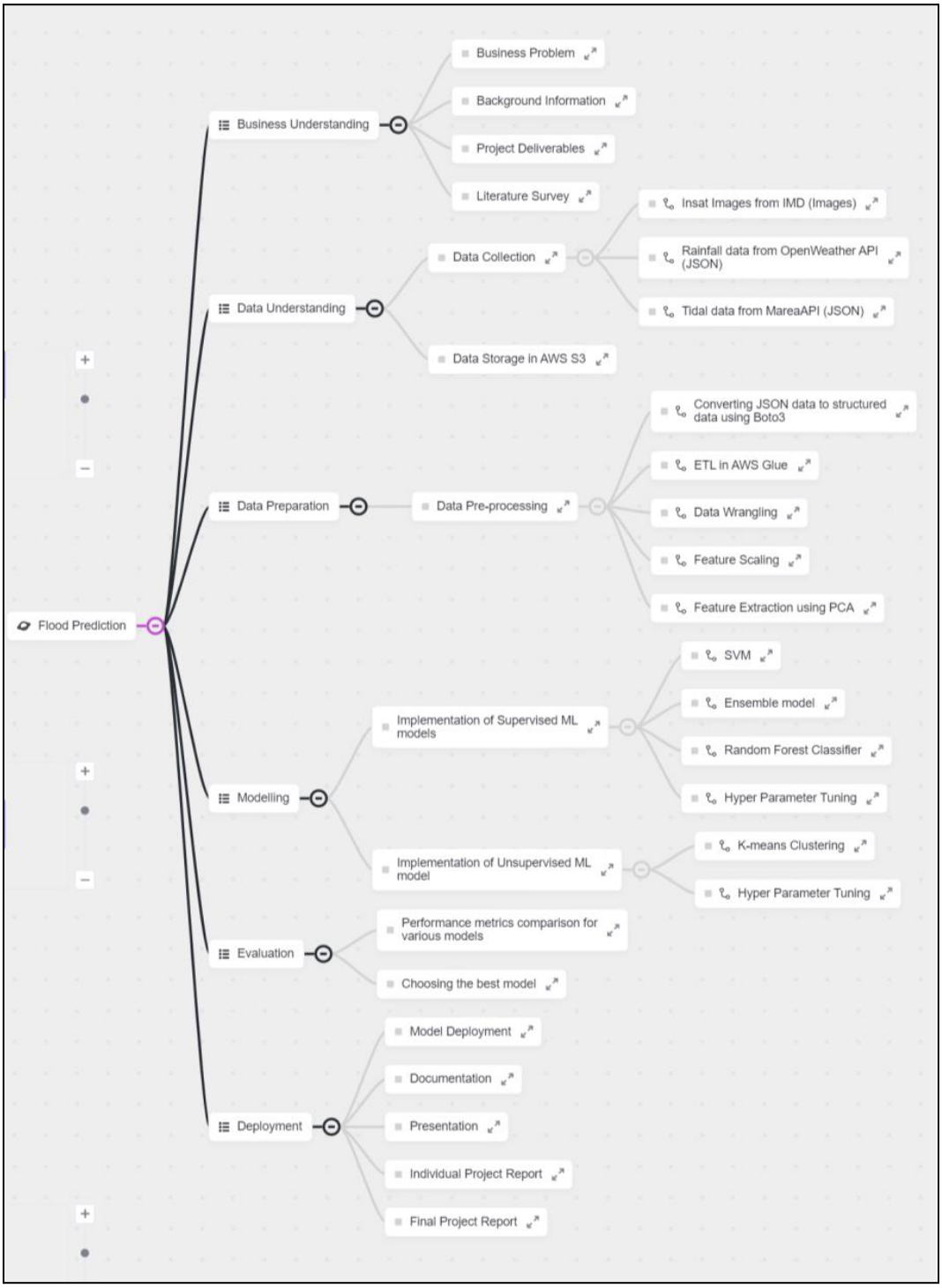
In this phase, the model that performs best for Flood Detection and Prediction was deployed. It included data preparation steps as well so that the model can work efficiently along with monitoring and maintenance details. Finally, the generation of the final report summarizing the research findings was performed.

2.3 Project Organization Plan

CRISP-DM is a commonly used project development technology in the industry; the same was used to establish project work breakdown structure of research project of flood prediction. There are six different phases as shown in Figure 9.

Figure 9

Work breakdown structure



The team needed a plan for this project to understand and dive into possible tasks and subtasks that need to be accomplished to successfully complete the project. The plan includes understanding different phases of the project and in each phase, what are the necessary tasks to be completed by everyone parallelly with possible timelines. A task may be completed by one individual in the team or sometimes it requires collaboration with the other team members. Figure 9 represents the work breakdown structure in alignment with the CRISP-DM methodology which helped the team in better management of the project plan.

The work breakdown structure of the project followed the basic flow of any data engineering project, where it started with data pre-processing initially and then to model development and validation. Business Understanding is the crucial phase of the project where the problem statement was defined, selecting the models for project research, and designing the plan accordingly was carried out. Data Understanding and Data Preparations are the phases where the focus is on the type and forms of data which needs to be collected for research and determining the preprocessing techniques required to prepare the data for the Modeling phase. Although Modeling and Evaluation Phases have a limited number of goals to be accomplished, it was more time consuming than the Data Preparation phase. In Modeling phase, Machine learning models were built based on different algorithms such as SVM, Random Forest Classifier, XGBoost and K-means clustering followed by Evaluation Phase which included checking the veracity of the various models built based on accuracy, recall, F1 score, confusion matrix and area under the curve. The last phase being the deployment included, reviewing of the project and generating the final report eventually. At this point the best model based on evaluation of performance metrics was deployed as the final deliverable.

2.4 Project Resource Requirements and Plan

Regarding project resources as shown in Table 15, data was obtained from various sources and the obtained data was processed with the help of various services and platforms. Data was collected from IMD, OpenWeatherAPI and Marea API. The data extracted from open sources, and the information collected was merged to a dataset. The data exploration and collection was carried out for about a month of the planned schedule. Data storage was done on a local machine. Data preprocessing and ETL was done using python libraries on Jupyter Notebook. Following which the various machine learning models were built in Jupyter Notebook and visualizations were made with the help of matplotlib and seaborn packages of python.

Table 15

Resources used and cost estimations

Utility	Type of Resource	Tools used	Time Taken	Estimating the Cost
Data Source	DataSet	API (OpenWeather API, MareaAPI) IMD Website	1 month	Free
Local machine	Hardware	64 bit	4 months	500 USD
Data Pre-processing	Software	Pandas, Jupyter Notebook	2 months	Free
Data Storage	Hardware	64 bit	4 months	Free
Machine Learning Modeling	Software	Jupyter Notebook	2 months	Free
Machine Learning Framework	Software	Sklearn	2 months	Free
Visualization tool	Software	Jupyter Notebook	2 months	Free

2.5 Project Schedule

2.5.1 Gantt Chart

Gantt chart as shown in Appendix A helps in organizing the project in a timely manner and also keeps a track of time estimates made for the research project. The tasks were equally assigned to all the members of the group. Dependencies between each task to be completed were identified and depicted in the chart. The Gantt chart of the project includes each task, start date, due date, assignees for each task, the time estimate to complete each task as well as the dependencies between them.

Figure 10 shows the Gantt chart for Business understanding phase. The tasks within this phase are listed here. All the tasks in this phase were done by all the team members and this task was completed within the due date specified. The tasks are shown in green which means that these are completed tasks. The arrows shown in the figure show the dependencies between various tasks.

Figure 10

Gantt chart for Business Understanding phase

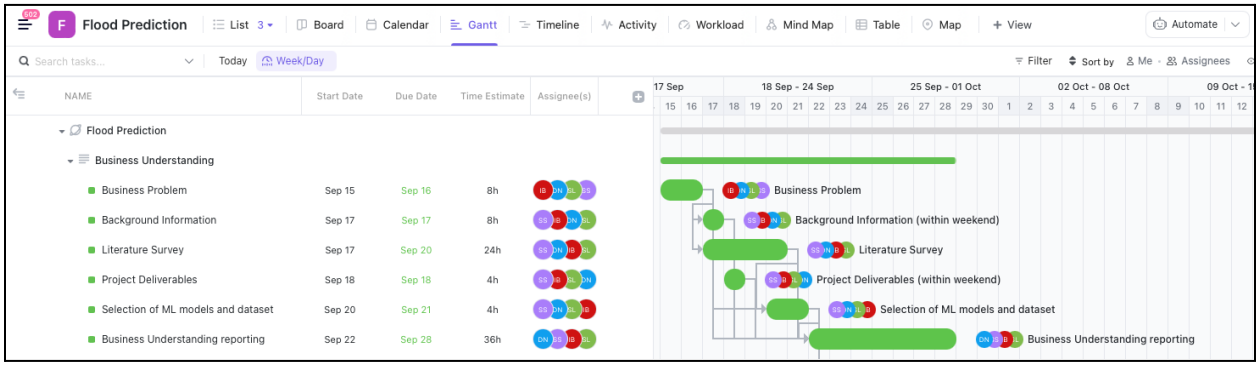


Figure 11 shows the gantt chart for the Data and Project Management tasks. This phase can only be started after the project deliverables have been listed out in the first phase. This

phase was further divided into various tasks like creating the WBS, Pert chart, Gantt chart and data management plan respectively. These tasks were also performed by all the members together and the estimated time for completing this task was 55 hours.

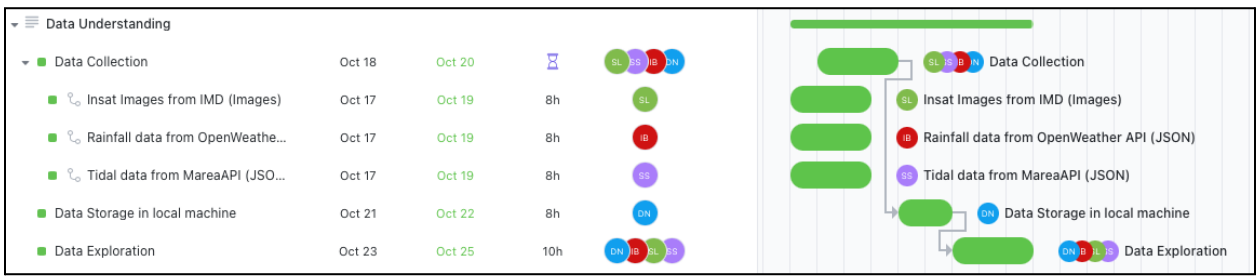
Figure 11

Gantt chart for Data and Project Management Tools task

Data Understanding Phase has been shown in Figure 12 which includes tasks like data collection, data storage and data exploration. Data collection is dependent on the Business Understanding phase and can be started only after the first phase is completed. Data collection was done from three different sources and was assigned to three members. This task was completed simultaneously by assignees and took around eight hours to complete. Data collection was the blocking task for the data storage task.The data storage task was assigned to the fourth member and was accomplished in time. Next task was data exploration which was performed by all the team members once the data was stored on the local machine. The entire phase started on October 10 and was completed by 25th October, 2022.

Figure 12

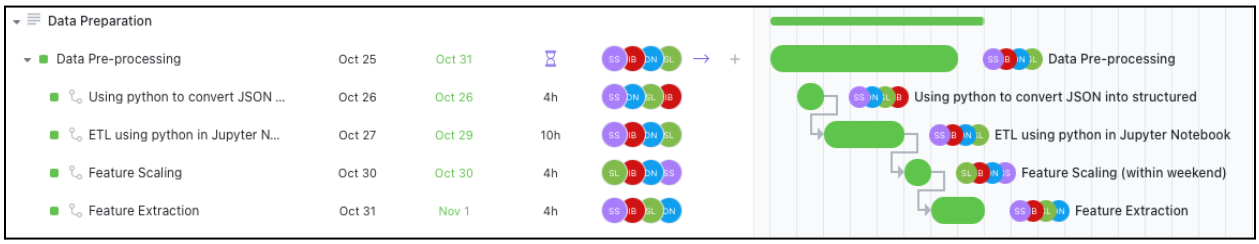
Gantt chart for Data Understanding phase



Next phase is the Data Preparation phase as shown in Figure 13 which was assigned to all the members. Data storage was the blocking task for this phase. This task included the data pre-processing steps and each subtask within this phase was dependent on the previous task to be completed..

Figure 13

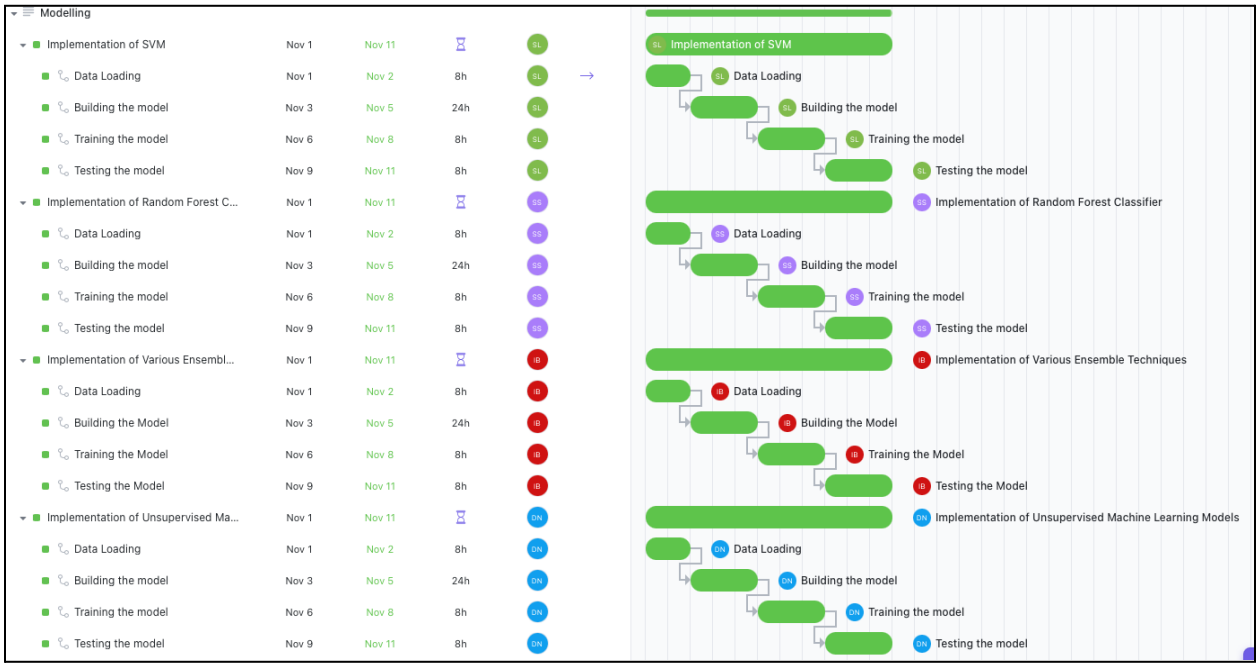
Gantt chart for Data Preparation phase



Modeling phase could not be started until the Data Preparation phase was completed. In this phase as shown in Figure 14, each member worked on one model respectively. The subtasks within the implementation of models like data loading, building a model, training a model, testing the model for each model was done simultaneously by the members and this phase was estimated to be completed by 11th November, 2022.

Figure 14

Gantt chart for Modeling phase



The next phase is model evaluation as shown in Figure 15. The model implementation was the blocking task for this phase and was started once the model was implemented. The precision, recall, F1 score and AUC-ROC curve for each model was calculated and evaluated by each member for their respective models. Finally, the best model was selected which was done by all the members together and this phase was estimated to be completed by 19th November, 2022.

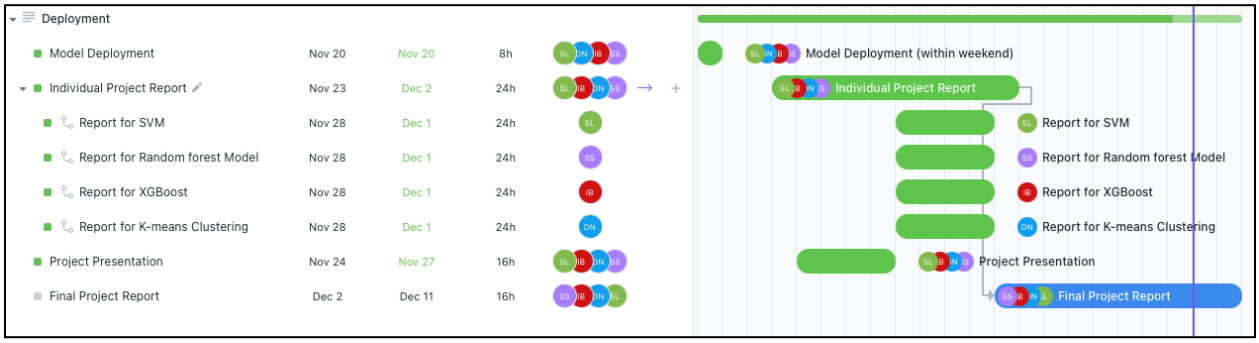
Figure 15

Gantt chart for Evaluation phase

The final stage was deployment where the model was deployed by all the members. This task was performed once the best model was selected. The Individual project report was prepared by each member for their respective models. Whereas the final presentation and the final report was done by all the members together which was dependent on the individual report created by each member. The expected due date to finish the final project report was 11th December, 2022 as shown in Figure 16.

Figure 16

Gantt chart for Deployment phase

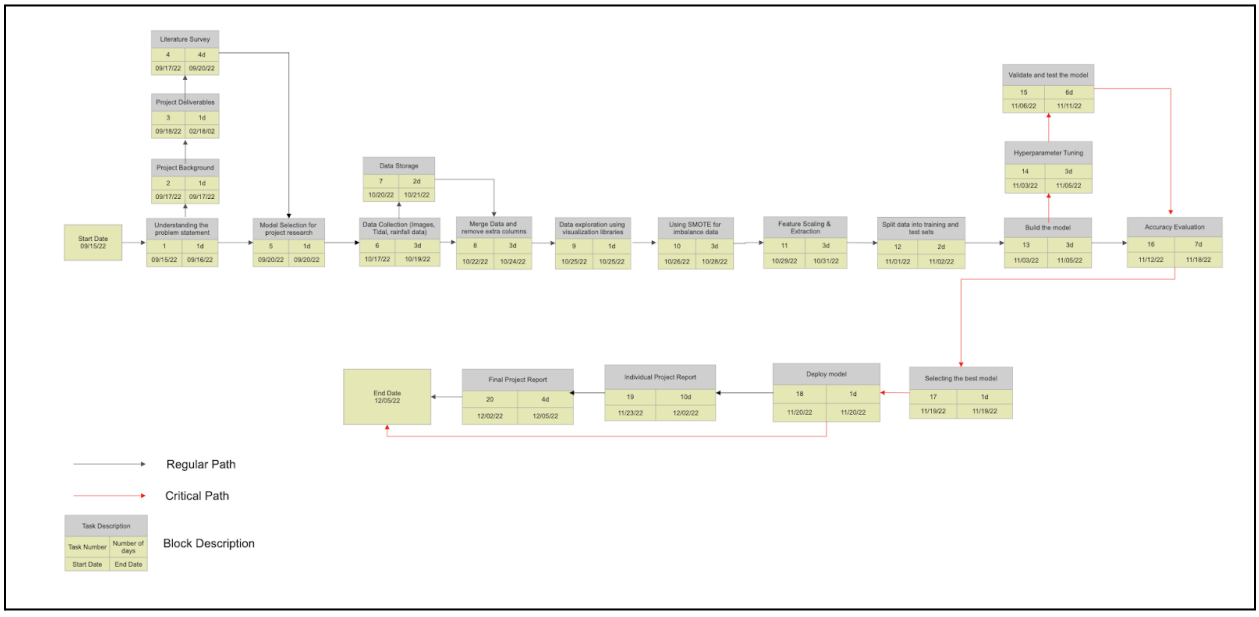


2.5.2 PERT Chart

PERT Chart stands for Program Evaluation Review Technique. The processes explained above in the Gantt chart are depicted in the PERT Chart so that the task and activities can be analyzed at a higher level of granularity as shown in Figure 17.

Figure 17

PERT Chart

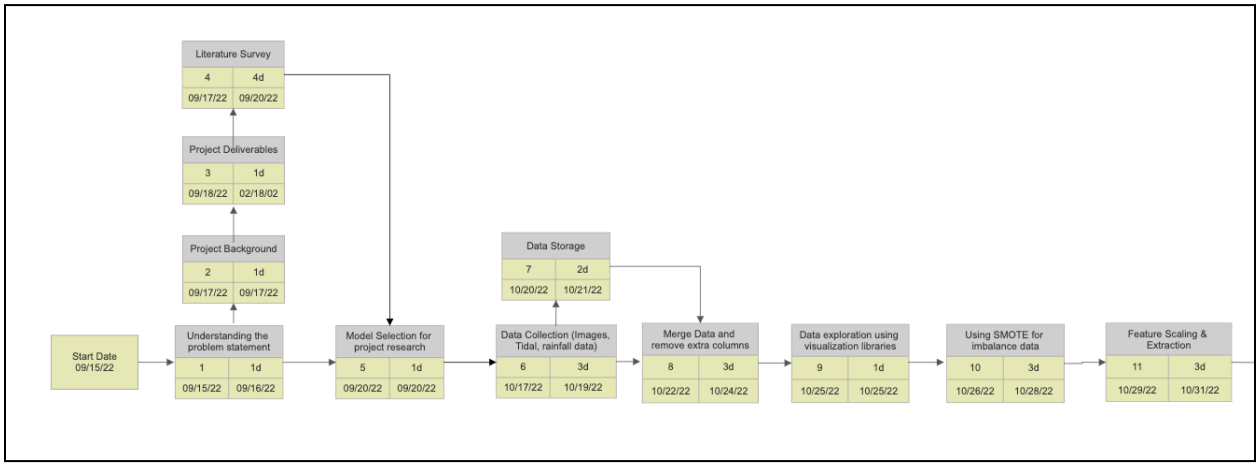


All the activities and tasks in Figure 17 were analyzed and programmed so that these specific processes were done as per the given time limit frame. These tasks were structured according to the stages involved in the project and the time limit was listed below of each task. The arrows in the charts show the transition from one stage to the other. Starting point is the "start date". The red line shows the critical tasks which includes business understanding , model selection, data collection and data exploration , building the model, accuracy evaluation, selecting the model and lastly followed by deploying the model and finishing the project by 12th December, 2022. These critical tasks were evaluated on the basis of longest duration (i.e. activities taking more time as compared to other activities).

The first critical task is "business understanding" which involves understanding the problem statement, project background by referring to relevant research papers. Next phase involved the Model selection i.e. selecting the relevant ML models after taking reference from research papers. After this, the next step was data collection i.e. collecting data from different sources as per the need of the project and data exploration which was done using python. The steps are shown in Figure 18.

Figure 18

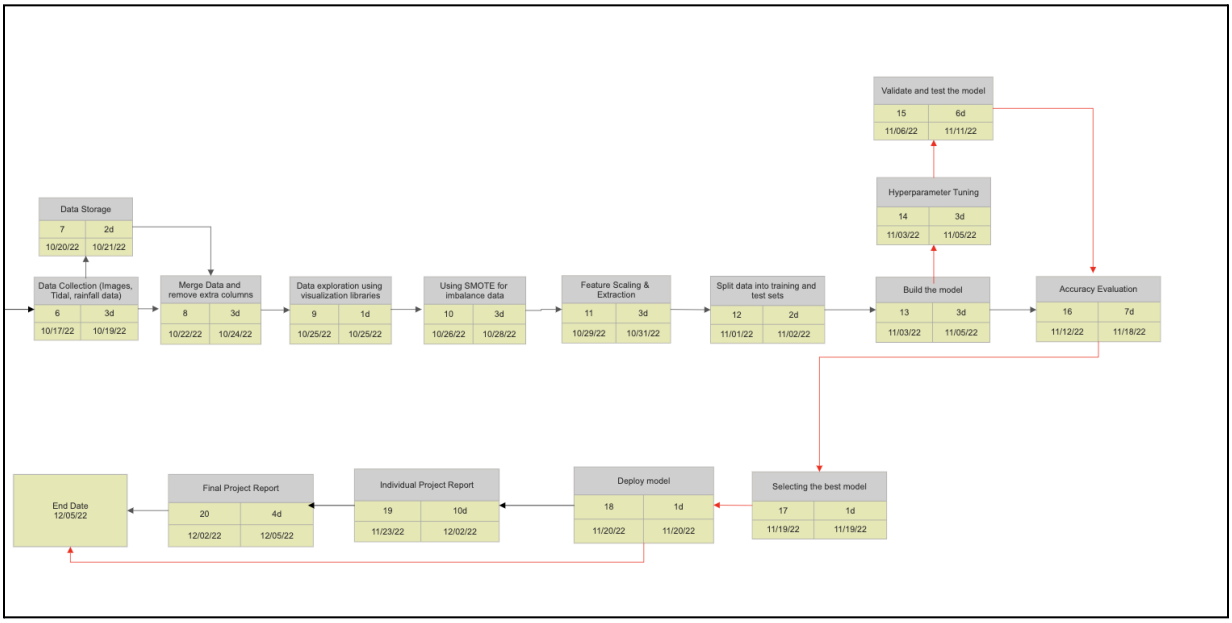
PERT Chart for first three phases



The next phase involves feature selection followed by splitting the data into training and testing and then building the machine learning model. After doing these activities, the performance metrics of each classifier was analyzed by comparing precision, recall, F1 score, accuracy, coefficient of determination etc. Then the second last stage was the model deployment where reports for each model were prepared separately followed by drafting the project report (collating all the results) and project presentation. The end date of this project was set to 12th December, 2022 as shown in Figure 19.

Figure 19

PERT Chart for last three phases



3. Data Engineering

3.1 Data Process

In this section, the entire process from data collection to data preparation has been discussed. For data collection, various parameters such as tidal characteristics, rainfall data, and information from INSAT images were considered. These were collected from multiple sources and processed into suitable formats to make prediction and detection easy.

For rainfall data, the volume of rainfall are calibrated using tipping gauges. The data was collected from Open Weather API. For tidal data, tidal height and tide state are the key parameters in determining tide type. These records were collected from Marea API where calibration was done using acoustic tide gauges and other equipment. For INSAT images, Indian Meteorological Website was used to collect real-time images. All the above data mentioned i.e., tidal data, weather data, and INSAT images were extracted by running a Python script that fetches relevant data and images every 15 minutes. 1270 records were collected accordingly.

For exploratory data analysis purposes, the datatypes of each column in rainfall, and tidal tables were checked for merging them and performing operations on them efficiently. The features that were not relevant to the analysis were removed. From rainfall data, features such as rainfall (in mm), humidity, and cloudiness were considered. For tidal data, tidal height and status were considered. For INSAT images, images were cropped to focus on the desired area of interest i.e., Cochin, Kerala, and image features were enhanced to distinguish cloud patterns from the ground. The number of bone-weight pixels was counted as pixel count and this count was used as a feature.

Once necessary features were selected, data was merged on the timestamp. The missing values were removed using the drop function if there were fewer NaNs or Simple Imputer from

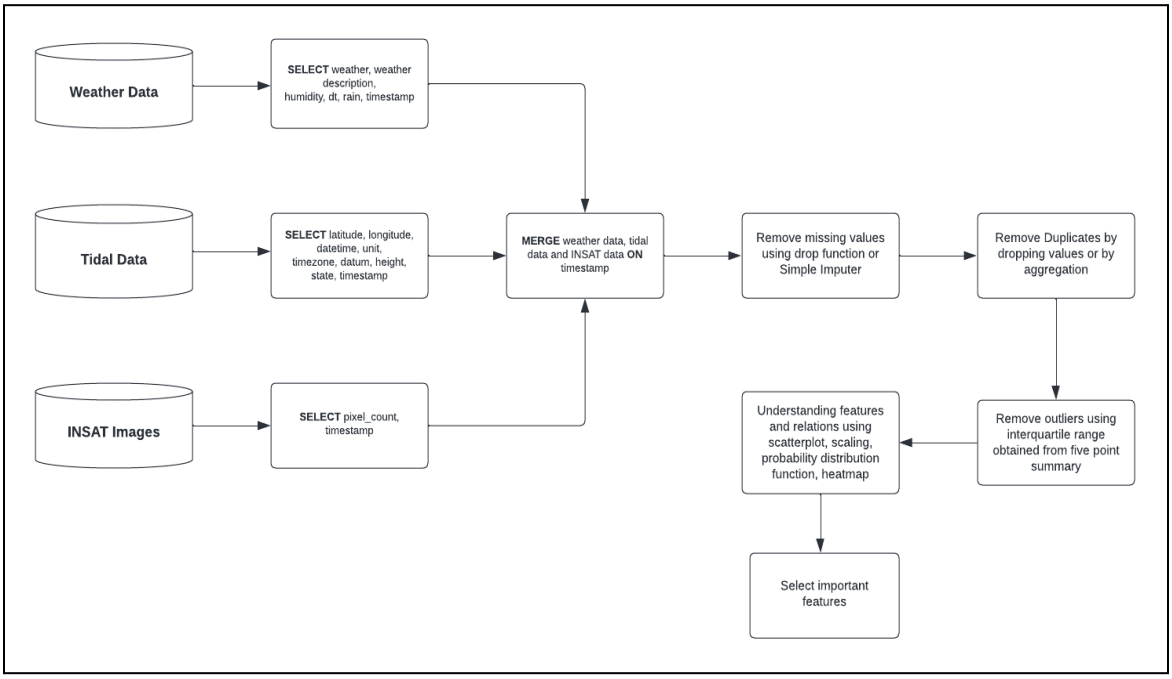
scikit-learn was used to replace NaNs with mean or mode. Duplicates were removed using the drop function or aggregation. The columns were further split, for instance, the weather column was split down to weather and the description to understand the features more.

To understand the features and the relationship between them, scatterplots were plotted, scaling was performed, and probability distribution was observed. A five-point summary was used to remove outliers and the correlation was studied with the help of heatmaps. The important features were then selected i.e., features with the most weights.

The data transformation approach selected was data normalization as each column represented data in different scales. Data augmentation was also performed as the data was highly imbalanced. Synthetic Minority Over-Sampling Techniques (SMOTE) were used to duplicate the minority samples from the training data for reducing bias during training. Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) regularization techniques were not applicable as the data included all the relevant features and reducing dimensions would lead to loss of information and less accurate models. Data was then split into training and testing sets using 80:20 ratio. Figure 20 shows the flowchart that illustrates the entire data process.

Figure 20

Flowchart illustrating the data process



3.2 Data Collection

The most important parameters considered for flood prediction for this analysis were the volume of rainfall recorded, cloudiness percentage humidity, tidal height, tidal status, and INSAT images showing weather patterns for Kerala. The data collection plan from different sources has been described in the below sections.

Imaging through satellites was used to obtain information about various locations on the Earth, and there are many satellites, each with its specific purpose. Multiple types of sensors collect radiation reflected back from the Earth. Satellite images are usually classified into three types: visible, infrared, and water vapor. In the case of visible satellite imaging, these images can be viewed during the day only, as light reflected by the clouds is captured and requires sunlight. This property is called albedo, and this calibration requires an imaging radiometer. In the case of infrared satellite imaging, images can be viewed during the day and night. Heat radiated from the

Earth is used to identify clouds. Lastly, as the name suggests, water vapor satellite images finds the amount of vapor available in the atmosphere.

Visible satellite imagery was used for the analysis. Clouds show up as white in these images and the ground is gray. So, different clouds radiate light differently based on density, the sun's angle and other parameters. This information was used to identify and understand cloud behavior. Examples of different cloud types are mentioned in Figure 21.

Figure 21

Different types of clouds and respective parameters

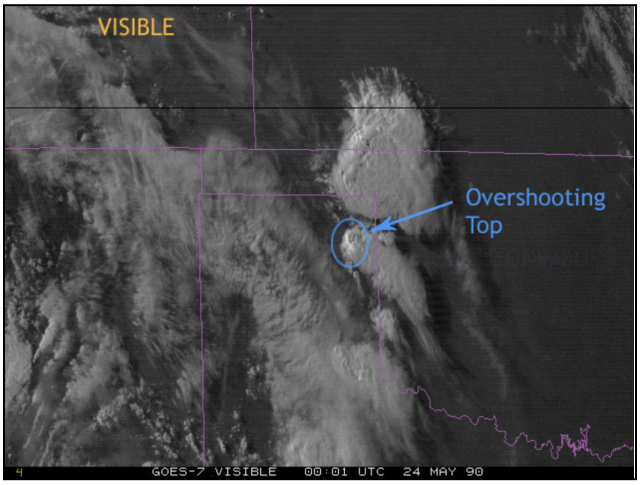
Cloud Level	Prefix/ Suffix	Typical Altitudes	Layer Clouds	Heap Clouds	Hybrid
High	"Cirro-"	20,000 - 40,000 feet (6-12 kilometers)	**Cirrus** **Cirrostratus**	**Cirrocumulus**	
Mid	"Alto-"	6500 - 20,000 feet (2-6 kilometers)	**Altostratus**	**Altocumulus**	
Low		100 - 6500 feet (bases below 2 km)	**Stratus**	**Cumulus**	**Stratocumulus**
Precipitating	"Nimbo-" "-nimbus"	100 - 6,500 feet (bases below 2 km)	**Nimbostratus**		**Cumulusnimbus** (Thunderstorms!)
Significant Vertical Development		100 - 50,500 feet (0.1-15 kilometers)			

Note. From Satellite Meteorology - Cloud Identification. (https://cimss.ssec.wisc.edu/)

For flood detection and prediction, Nimbostratus clouds and Cumulonimbus clouds were observed from these images. These clouds produce precipitation. Nimbostratus clouds indicate moderate to heavy rainfall whereas cumulonimbus clouds indicate heavy rainfall that takes place for a short time period. Cumulonimbus clouds can be identified as shown in Figure 22.

Figure 22

Satellite image of Cumulonimbus Clouds



Note. From Satellite Meteorology - Cloud Identification. (https://cimss.ssec.wisc.edu/)

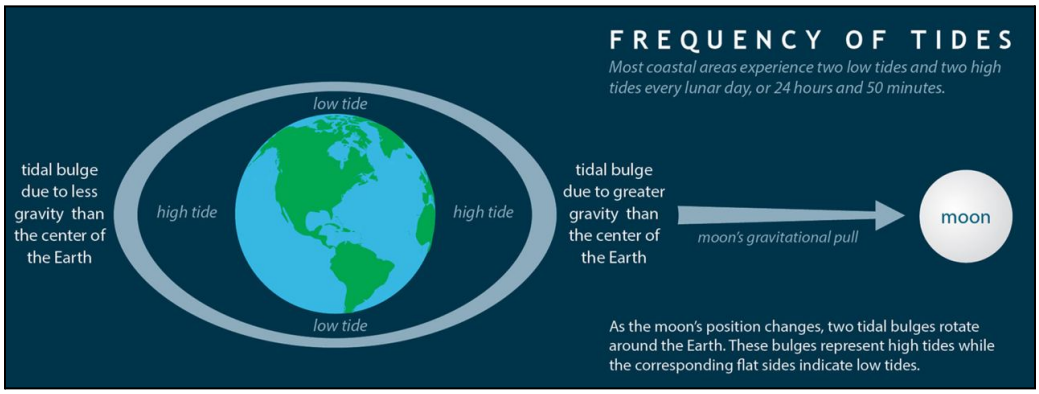
Once the images were captured by satellite, they were sent over to the station via a transmitter. These images were shared via the Internet with people who can help determine cloud behavior or perform preprocessing methods on the image. Preprocessing includes cropping images, converting them to usable formats, adding metadata like acquisition data, etc. Once preprocessing was done, these images were stored in the database for further analysis.

Tides are the rise and fall of sea levels caused by the combined effects of the gravitational forces exerted by the moon and the sun. When combined with the rotation of the earth, these forces are exhibited in daily raising and falling of the water level. The most important factor is the gravitational pull of the moon. High tides are formed on the side of the earth closer to the moon and on the opposite side of the earth due to the moon's gravitational force and centrifugal force of the earth respectively. Low tides are formed in the areas in between the high tide region on earth as shown in Figure 23. Every 24 hours and 50 minutes, two high tides and two low tides are experienced at almost every coastal region. It takes around 6 hours and 12.5 minutes for the

water level to go from high to low and vice versa. Like most of the coastal region, the Indian coastal region also experiences two high and two low tides.

Figure 23

Formation of tides



Note. The two tidal bulges caused by inertia and gravity will rotate around the Earth as the moon's position changes. From How Frequent are Tides?, by NOAA's National Ocean Service Education. (n.d.). (https://oceanservice.noaa.gov/facts/tidefrequency.html).

To measure the height of the Tide, the water level at each rise and fall was physically measured. There are many methods but an Acoustic Tide gauge was used (Figure 24) to measure and record the water level at fixed intervals of time. Along with the Tide gauge, the station also includes Tide staff, a Tidal benchmark, a solar panel, and an antenna as shown in Figure 25.

First, a tidal benchmark was established as it was simply a physical mark such as a brass disc set in concrete from where the vertical datum was defined. This was a very important part of the station as it defines the key reference parameter to measure the height of the water level accurately. A benchmark denotes how high or low the vertical datum was from a particular mark. The benchmark location is generally available with the government hydrographic department for that country and its port. It usually takes years of continuous observation to establish the vertical datum. For this data collection, Mean Sea Level (MSL) was used as the vertical datum.

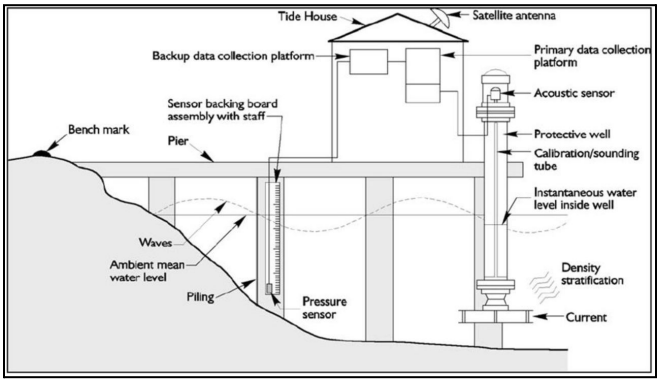
Then a reference level was set from which Tide height was measured. A tide staff measuring in feet or centimeters was mounted next to the gauge of the pier structure. The zero mark on the staff was set to Mean Sea Level as per the benchmark reading and these zero marks was the vertical datum (or vertical reference) to measure the height of the water level (or Tide level).

An acoustic tide gauge with advanced microprocessor-based technologies was used which has improved measurement accuracies and allowed for customized data collection. Here a sound wave was sent down a half-inch wide sounding tube and the time taken for the sound wave to reflect back after hitting the water surface was recorded. A transducer and microcomputer were used to convert this recorded time into an accurate height of the water/Tide level. The data was collected throughout the day. The processed Tide level data was then categorized as ‘High Tide’, ‘Low Tide’, ‘Raising’, and ‘Falling’ for a given time.

Solar panels were used to power the electronics of this setup and an antenna was used to transmit real-time data to the data center. All the sensitive electronics, transmitting equipment, data storage, and power units were enclosed in a protective housing. Care had been taken to ensure that the water level measured using the Tide gauge agreed with the Tide staff reading.

Figure 24

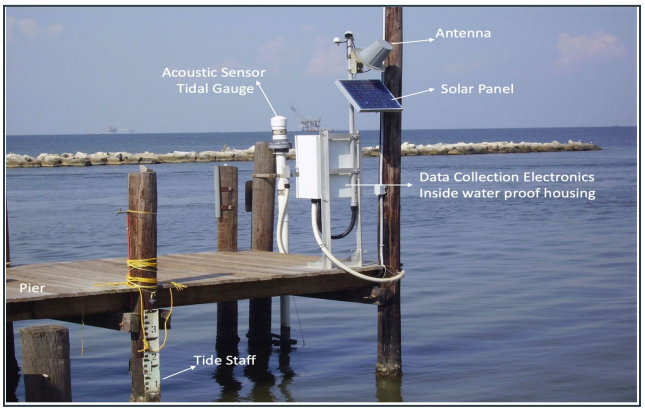
Schematic diagram of Acoustic Tide Gauge System



Note. Schematic diagram of Acoustic Tide Gauge System. From Indian Tsunami Early Warning System. by Indian National Center for Ocean Information Services(INCOIS). (n.d.). (https://tsunami.incois.gov.in/TEWS/Abouttideguage.jsp). Copyright Reserved INCOIS@2018

Figure 25

Acoustic Tide Gauge Setup



Note. A NOAA water level monitoring station with an acoustic sensor on Dauphin Island, Alabama. From Researchers model difference in east coast sea level rise, by Virginia Institute of Marine Science. (https://news.climate.columbia.edu/2017/05/18/researchers-model-differences-in-east-coast-sea-level-rise/)

For the collection of rainfall data, tipping rain gauges were used. It was used to measure the volume of rainfall in mm. Figure 26 shows the tipping bucket rain gauge. During rainfall, the water flowed into the funnel placed on the top of this gauge. This water was then directed into the two tipping buckets. Once the water exceeded the limit on the level of the water that can be collected into this bucket, the stop screw was hit and the other tipping bucket was lifted and started to fill. This process of filling and tipping is continuous whenever it rains. Once the stop screw was hit, it led to the activation of a switch that records the rain information electronically. The number of times the switch has been triggered was counted and the sensor automatically calculated and displayed the rainfall precipitation. The volume of the rainfall data was recorded every hour for a duration of one month.

Figure 26

Tipping bucket rain gauge

Note. From What is a tipping bucket rain gauge, and how does it work?, 2020, Instrument Choice(https://www.instrumentchoice.com.au/news/what-is-a-tipping-bucket-rain-gauge-and-how-does-it-work.)

An optical drum cloud base recorder as shown in Figure 27 was used for calculating cloudiness percentage. The drum consists of a transmitter that directs a light beam for a minute from eight degrees to 85 degrees with respect to the horizontal. The receiver was placed at a known distance which detects the light only vertically. The recorder was placed at the transmitter as well as the receiver which records the cloud base height if the light was scattered by the clouds and was captured by the receiving module. Combining data from all angles, the percentage of cloud coverage was estimated.

Figure 27

Optical drum cloud base recorder

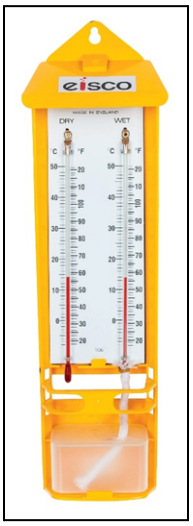
Note. From Automated Cloud Base and Visibility Measurement, 2021, Skybrary (https://www.skybrary.aero/articles/automated-cloud-base-and-visibility-measurement.)

Data for relative humidity was measured using wet and dry bulb hygrometers (Figure 28). The dry bulb thermometer measures the temperature of the air. The wet bulb thermometer was wrapped with a muslin cloth which was dipped into a beaker full of water. It measures the

temperature during evaporative cooling. The difference between the temperature measured from the dry bulb and the wet bulb thermometer was noted. The table was then used as shown in Table 16 to find the relative humidity by looking at the point for the corresponding dry bulb temperature and the difference between the dry and wet bulb temperature. The unit of measurement for relative humidity was percentage.

Figure 28

Dry and Wet bulb Hygrometer



Note. From Mason's Hygrometer - Wet & Dry Bulb Thermometer - Wall-Mounted, Eisco (https://www.eiscolabs.com/products/ph0232b.)

Table 16

Table to measure relative humidity

Dry Bulb (°C)	Number of degrees difference between the wet- and dry-bulb readings (°C)									
	1	2	3	4	5	6	7	8	9	10
10	88%	77	66	56	45	35	26	16	7	--
11	89	78	67	57	47	38	28	19	11	2
12	89	79	68	59	49	40	31	22	14	5
13	89	79	69	60	51	42	33	25	16	9
14	90	80	70	61	52	43	35	27	19	11
15	90	80	71	62	54	45	37	29	22	14
16	90	81	72	63	55	47	39	31	24	17
17	91	82	73	64	56	48	41	33	26	19
18	91	82	73	65	57	50	42	35	28	21
19	91	82	74	66	58	51	44	37	30	24
20	91	83	75	67	59	52	45	38	32	26
21	91	83	75	68	60	53	47	40	34	27
22	92	84	76	69	61	54	48	41	35	29
23	92	84	77	69	62	56	49	43	37	31
24	92	84	77	70	63	57	50	44	38	32
25	92	85	77	71	64	57	51	45	40	34
26	92	85	78	71	65	58	52	46	41	35
27	93	85	78	72	65	59	53	47	42	37
28	93	86	79	72	66	60	54	49	43	38
29	93	86	79	73	67	61	55	50	44	39
30	93	86	80	73	67	61	56	50	45	40
31	93	86	80	74	68	62	57	51	46	41
32	93	87	80	74	68	63	57	52	47	42
33	93	87	81	75	69	63	58	53	48	43
34	93	87	81	75	69	64	59	54	49	44

Note. From Relative Humidity Table | Relative Humidity Chart, *2*019, Test and Measurement World(https://www.test-and-measurement-world.com/Terminology/Relative-Humidity-Table-or-Chart.html.)

For this project, the data for humidity, cloudiness percentage, weather description, the volume of rainfall, etc. was extracted from Open Weather API. The tidal height and the state of the tide were extracted from Marea API and the INSAT images were fetched from the Indian Meteorological Department website. The data was extracted by running a python script for each of the APIs and the website and the data were fetched after every 15 minutes. The sample code for collecting this data is as shown in Figure B1, Figure B2 and Figure B3 respectively from

Appendix B. API key, latitude, and longitude for Kerala (Cochin) were given as input to fetch the data, and the data extracted was stored in a file. BlockingScheduler was used to bring the data every 15 minutes.

The data collection process started on October 26, 2022, and the data was collected till November 11, 2022. Around 1273 instances of the data were collected till November 11, 2022. The tidal and the open weather API data was collected in .txt format which was then stored with a. json extension. We loaded this raw data into pandas to view them.

3.2.1 Raw Dataset samples

The samples of the raw data collected from Open Weather API is as shown in Figure 29. The dataset loaded into pandas showed around 1273 instances and 15 columns. The entire data extracted showed the latitude and longitude for Cochin, weather descriptions like cloudy, rainy, snowy, wind speed, timezone, etc. Only the relevant features were extracted like the timestamp (dt) specified in Unix epoch time (integer datatype), the volume of rain (in mm and float data type), Weather which mentioned the weather description (object datatype), humidity (in percentage and integer datatype), and cloudiness (in percentage and integer datatype) which has been explained in the Data cleaning section. The data collected from Open Weather API till November 11, 2022, constituted 128 KB of memory. The datatype for the relevant features has been shown in Figure 30.

Figure 29

Samples from the Raw dataset extracted from Open Weather API.

```
weather_data= pd.read_json(r"/Users/iqrabismi/Desktop/allWeather.json", lines=True)
```

```
weather_data
```

	coord	weather	base	main	visibility	wind	clouds	dt	sys	timezone	id	name	cod	snow	rain
0	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 271.89, 'feels_like': 266.75, 'temp_m...	7077	{'speed': 4.85, 'deg': 87, 'gust': 4.71}	{'all': 100}	1666771218	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
1	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 271.89, 'feels_like': 266.75, 'temp_m...	7077	{'speed': 4.85, 'deg': 87, 'gust': 4.71}	{'all': 100}	1666772211	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
2	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	10000	{'speed': 5.04, 'deg': 84, 'gust': 4.55}	{'all': 100}	1666773111	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
3	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	10000	{'speed': 5.04, 'deg': 84, 'gust': 4.55}	{'all': 100}	1666774011	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
4	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	10000	{'speed': 5.04, 'deg': 84, 'gust': 4.55}	{'all': 100}	1666774911	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 30

Datatypes of relevant features for Open Weather API data

```
weather_data.dtypes
```

```
dt                      int64
rain                  float64
Weather                object
Weather_description    object
humidity                int64
cloudiness              int64
dtype: object
```

The tidal data had 1273 rows and 15 columns namely the latitude, longitude, origins, datum, timestamp, heights, etc. some of which were irrelevant and were dropped which has been explained in the data cleaning section. The data was fetched in JSON format so most of the data

was stored in the form of keys and value pairs (dictionary). So, the datatype for these raw datasets was mostly of object datatype as shown in Figure 32.

Figure 31

Samples from the Raw dataset extracted from Marea API.

```
tidal_data= pd.read_json(r"/Users/iqrabismi/Desktop/allTides.json", lines=True)
```

```
tidal_data
```

	disclaimer	status	latitude	longitude	origin	datums	timestamp	datetime	unit	timezone	datum	extremes	heights	source
0	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:00:18	2022-10-26 08:00:18+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666771218, 'height': -0.459390...	FES2014
1	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:16:51	2022-10-26 08:16:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666772211, 'height': -0.388316...	FES2014
2	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:31:51	2022-10-26 08:31:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666773111, 'height': -0.318326...	FES2014
3	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:46:51	2022-10-26 08:46:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666774011, 'height': -0.244201...	FES2014
4	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 09:01:51	2022-10-26 09:01:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666774911, 'height': -0.167119...	FES2014

Figure 32

Datatype for raw dataset fetched from Marea API.

```
tidal_data.dtypes
```

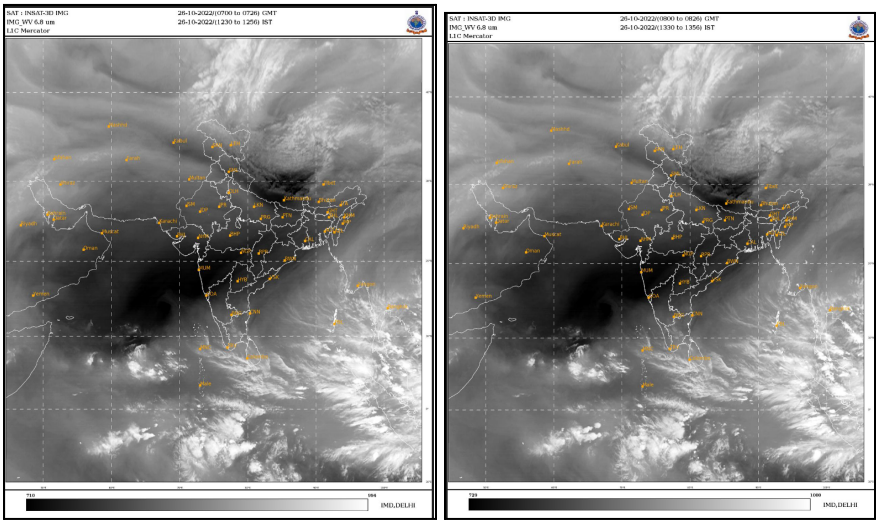
```
disclaimer                 object
status                      int64
latitude                  float64
longitude                 float64
origin                     object
datums                     object
timestamp          datetime64[ns]
datetime      datetime64[ns, UTC]
unit                       object
timezone                   object
datum                      object
extremes                   object
heights                    object
source                     object
copyright                  object
dtype: object
```

The relevant features from this raw dataset that were considered for the analysis were the timestamp (Unix epoch time and timestamp datatype), tidal height (in meters and float data type) which measured the height of the tide at that particular time, and tidal state which mentioned the state of the tide (RISING, FALLING and object data type). The latitude and longitude were specified for the region of Cochin (Kerala). Raw tidal data constituted around 1.07MB of data.

The INSAT images collected required space of about 84.3 MB and around 1270 images were fetched from the IMD website. Figure 33 shows the sample of these INSAT images. These images show the water vapor (clouds) content for India.

Figure 33

Two samples of Insat images



The data collection plan is shown in Table 17.

Table 17

Data Collection Plan

Data Collection Plan

Project number: Group 4 | Project title: Flood Detection and Prediction U | Project leader: | Date: 10/24/2022

Description of the data collection

INSAT Images are collected using visible satellite imagery. The brightness of a group of pixels is used to determine chances of precipitation.
Volume of the rainfall in mm is collected using Tipping Bucket Rain Guage
Height of the tide in cm/ft is collected using Acoustic Tide Gauge
Humidity (%) data is collected using wet and dry bulb hygrometer
Cloudiness (%) is collected using Optical Drum Cloud Base Recorder

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)

		1	2	3	4	5	6
What?	**Variable title**	satellite images	Volume of rain	Cloudiness	Humidity	Height of Tide	
	Input (X) or output (Y) variable?	X	Y	Y	Y	Y	
	Unit of measurement	Lumen	mm	%	%	m	
	Data type	Continuous	Continuous	Continuous	Continuous	Continuous	
	Collection method	Automated	Automated	Automated	Automated	Automated	
	If manual				Measuring	Measuring	
MSA	Gauge/instrument	Imaging Radiometer	Tipping bucket rain gauge	Optical drum cloud base recorder	Dry and Wet bulb hygrometer	Acoustic Tide Gauge	
	Location	Kerala (Cochin)	Kerala (Cochin)	Kerala (Cochin)	Kerala (Cochin)	Kerala (Cochin)	
	Gauge calibrated?	Yes	Yes	Yes	Yes	Yes	
	Measurement system checked?	Yes	Yes	Yes	Yes	Yes	
	Precision (R&R) adequate?	Yes	Yes	Yes	Yes	Yes	
	Accuracy adequate?	Yes	Yes	Yes	Yes	Yes	
Historical data	Historical data exist?	Yes	Yes	Yes	Yes	Yes	
	Source of historical data	Previous images on DB	Open Weather API	Open Weather API	Open Weather API	Marea API	
	Historical data representative/reliable?	Yes	Yes	Yes	Yes	Yes	
	Mean	735	189.23	46.33	82.7	N/A	
	Upper specification limit	947	477.52	71	98.1	2.1	
	Lower specification limit	522	20.32	24	38.8	-0.1	
	Standard deviation	N/A (as it varies)	145.37	2.07	8.39	N/A	
Sampling	Minimum sample size (MSS)	1270	1273	1273	1273	1273	
	Sampling frequency	15 minutes	15 minutes	15 minutes	15 minutes	15 minutes	
	Sub-grouping needed?	No	No	No	No	No	
	Sub-group size	N/A	N/A	N/A	N/A	N/A	
	Stratification needed? (time, shift)	Yes	Yes	Yes	Yes	Yes	
Who?	**Data collector**	Data Analyst	Data Analyst	Analyst	Analyst	Data Analyst	
	Operational definition exist?	Yes	Yes	Yes	Yes	Yes	
	Data collector trained?	Yes	Yes	Yes	Yes	Yes	
	Resources available for data collector?	Yes	Yes	Yes	Yes	Yes	
When?	**Start date**	26-Oct	26-Oct	26-Oct	26-Oct	26-Oct	
	Due date	10-Nov	10-Nov	10-Nov	10-Nov	10-Nov	
	Duration (in days)	28	28	28	28	28	

Additional Comments - e.g. resolution needed, sampling method, R&R results, storing of data, handling outliers, using filters, etc.

INSAT Images:	Subgrouping is performed to identify cloud types. Subgrouping is done after PCA is performed as subgrouping size would vary based on the image. Sampling methods used: Spectral an d S

Continuous Improvement Toolkit . www.citoolkit.com

Guide:
1st, describe the reason behind the data collection. Why are we collecting data? how will the data help? and what should we do with the data once it has been collected?
2nd, identify the variables for which data needed to be collected, then fill the asked information for each variable. Try at least to fill the **bolded** fields.
3rd, add any additional comments you believe are important to the data collection process.
For the start and due dates, use the date forma: DD-MMM. E.g. 14-Mar.
You need only to fill the white cells.

3.3 Data Pre-Processing

3.3.1 Exploratory Data Analysis Plan

Data was collected from three sources. Rainfall and tidal data were converted into structured format (csv) whereas INSAT images were in jpeg. For exploratory data analysis on rainfall data and tidal data, the data types were first checked along with the total records in the data using df.info() command. As the rainfall and tidal data contained many features for different hydrological purposes (such as floods, droughts, etc.) the parameters that were relevant to the flood prediction model were considered. For rainfall data, rainfall (mm), humidity, and cloudiness % were considered as descriptive features in the model. On the other hand, for tidal data, tidal height and tidal status were considered.

After this, the missing values were checked in the data by importing the 'missingno' function in python. By using this module, missing values were visualized. If there were missing data, then the proportion of missing data was checked by leveraging pandas functions (i.e df.query('col== np.nan').value_counts(normalize= True)). If the proportion of values was minimal i.e. around 5%, then these null values were dropped, else SimpleImputer from sklearn was used to replace the null values with mean or mode as per the datatype. The next step in the validity check included duplicate values i.e. the data was checked to know if there were duplicate values. If there were any duplicate values then these values were eliminated by dropping the duplicates or by aggregating the values using groupby.

Next, the relationship between different features was checked by creating a scatter plot for all relevant features which was helpful in getting insights if there were positive and negative relationships between different attributes. If there was a difference in magnitude in the values then the min-max scaler was used to scale the values in the range (0,1). In addition to this, if one

of the features was skewed then logarithm was taken for that feature (as log is a monotonic increasing function and log transformation follows a nearly normally distributed data) to make it normally distributed. The distribution of these attributes was also checked to see if the features were normally distributed or skewed. This was done by using histograms or qq plots from statsmodels. In addition to this, a cumulative distribution plot or probability density plot was created in order to gain better insights into the distribution of the data. Then the next step included a five-point summary for all features, which was used to gain insights on minimum, median, quartiles, and maximum values. To visualize this, the seaborn function was used to create box plots or violin plots (i.e. getting five-point summaries along with distribution of the data). After the plots were created, the presence of outliers was checked. If outliers were visible in the plots, then the results were cross-checked by calculating the upper and lower limits using the interquartile range. Then according to this, values were either dropped or clipped. Furthermore, the correlation values between different features were checked by using df.corr() command and this data was visualized using a heatmap. The brighter colors such as reddish shade mapped to higher values whereas darker colors mapped to lower values. With this, a better understanding of the relationship between different features was understood. Also, cluster maps from seaborn was used to see the highly correlated features clustered together (i.e. hierarchical clustering). Lastly, the statistical differences between different target features and descriptive features were checked, for which a Wilcoxon test (non-parametric method) or two samples independent t-test (a parametric method) was conducted. A random forest model was also implemented and the important features were observed using feature_importance command.

For the INSAT images, the Python Image library in python was used as it gives flexibility in working with spatial and vector data. If the composite images were dark then the images were

stretched in the range of 0-255 pixels to increase the visual contrast of the image. Using the clip method, focused only on affected areas. Lastly, plot spectra were plotted to understand the nature of pixels. A dataframe with the timestamps and the pixel count for each INSAT image was then created and was merged with the tidal and rainfall data for further analysis.

Figure 34 shows the raw dataset from Open Weather API which had a lot of irrelevant features. The dataset after dropping irrelevant features is as shown in Figure 34. The weather data had null values for rainfall which were replaced by zero as it meant that there was no rain for that particular timestamp. This can be seen from Figure 36. Similarly, the raw dataset for tidal data is as shown in Figure 37. After dropping irrelevant features from the tidal data, the tidal data were merged and is as shown in Figure 38. For INSAT images, values with the color code other than [255,255, 255] were removed and a dataframe with timestamp and the pixel count was created which can be seen in Figure 39.

Figure 34

Raw dataset for Open Weather API

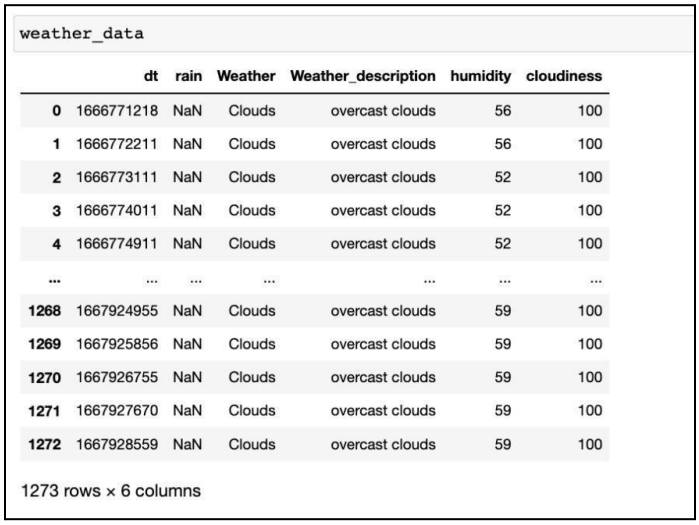
```
weather_data= pd.read_json(r"/Users/iqrabismi/Desktop/allWeather.json", lines=True)
```

```
weather_data
```

	coord	weather	base	main	visibility	wind	clouds	dt	sys	timezone	id	name	cod	snow	rain
0	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 271.89, 'feels_like': 266.75, 'temp_m...	7077	{'speed': 4.85, 'deg': 87, 'gust': 4.71}	{'all': 100}	1666771218	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
1	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 271.89, 'feels_like': 266.75, 'temp_m...	7077	{'speed': 4.85, 'deg': 87, 'gust': 4.71}	{'all': 100}	1666772211	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
2	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	10000	{'speed': 5.04, 'deg': 84, 'gust': 4.55}	{'all': 100}	1666773111	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
3	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	10000	{'speed': 5.04, 'deg': 84, 'gust': 4.55}	{'all': 100}	1666774011	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
4	{'lon': 9.94, 'lat': 76.26}	[{'id': 804, 'main': 'Clouds', 'description': ...	stations	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	10000	{'speed': 5.04, 'deg': 84, 'gust': 4.55}	{'all': 100}	1666774911	{'sunrise': 1666774297, 'sunset': 1666790195}	3600	0		200	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 35

Open Weather API data after dropping irrelevant features



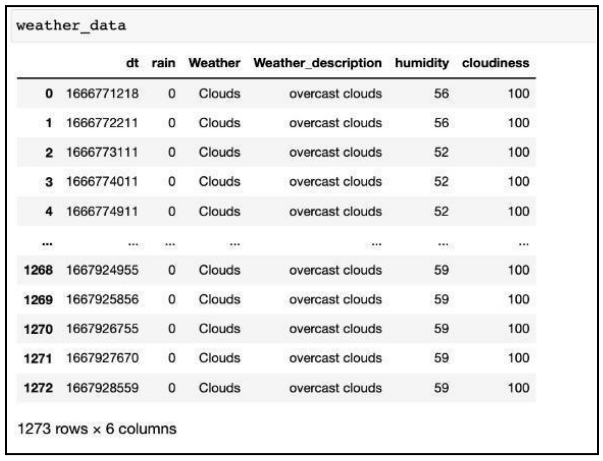
weather_data

	dt	rain	Weather	Weather_description	humidity	cloudiness
0	1666771218	NaN	Clouds	overcast clouds	56	100
1	1666772211	NaN	Clouds	overcast clouds	56	100
2	1666773111	NaN	Clouds	overcast clouds	52	100
3	1666774011	NaN	Clouds	overcast clouds	52	100
4	1666774911	NaN	Clouds	overcast clouds	52	100
...	...	...	...	...	...	...
1268	1667924955	NaN	Clouds	overcast clouds	59	100
1269	1667925856	NaN	Clouds	overcast clouds	59	100
1270	1667926755	NaN	Clouds	overcast clouds	59	100
1271	1667927670	NaN	Clouds	overcast clouds	59	100
1272	1667928559	NaN	Clouds	overcast clouds	59	100

1273 rows × 6 columns

Figure 36

Open Weather API data after replacing NaN values with zero values



weather_data

	dt	rain	Weather	Weather_description	humidity	cloudiness
0	1666771218	0	Clouds	overcast clouds	56	100
1	1666772211	0	Clouds	overcast clouds	56	100
2	1666773111	0	Clouds	overcast clouds	52	100
3	1666774011	0	Clouds	overcast clouds	52	100
4	1666774911	0	Clouds	overcast clouds	52	100
...	...	...	...	...	...	...
1268	1667924955	0	Clouds	overcast clouds	59	100
1269	1667925856	0	Clouds	overcast clouds	59	100
1270	1667926755	0	Clouds	overcast clouds	59	100
1271	1667927670	0	Clouds	overcast clouds	59	100
1272	1667928559	0	Clouds	overcast clouds	59	100

1273 rows × 6 columns

Figure 37

Raw dataset for Tidal data

tidal_data

	disclaimer	status	latitude	longitude	origin	datums	timestamp	datetime	unit	timezone	datum	extremes	heights	source
0	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:00:18	2022-10-26 08:00:18+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666771218, 'height': -0.459390...	FES2014
1	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:16:51	2022-10-26 08:16:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666772211, 'height': -0.388316...	FES2014
2	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:31:51	2022-10-26 08:31:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666773111, 'height': -0.318326...	FES2014
3	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:46:51	2022-10-26 08:46:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666774011, 'height': -0.244201...	FES2014
4	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 09:01:51	2022-10-26 09:01:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666774911, 'height': -0.167119...	FES2014
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1268	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-11-08 16:29:16	2022-11-08 16:29:16+00:00	m	UTC	MSL	[{'timestamp': 1667929980, 'height': -0.557096...	[{'timestamp': 1667924956, 'height': -0.422229...	FES2014

Figure 38

Dataset after merging open weather and tidal data

	rain	Weather	humidity	cloudiness	latitude	longitude	timestamp	timestamps	tidal_height	pixel_count	tidal_state_RISING	Weather_description_broken clouds	Weather_description_light rain	Weather_description_light snow	Weather_description_moderate rain	Weather_description_ove c
0	0.0	Clouds	0.173077	100	76.26	9.94	2022-10-26 08:00:18	1666771218	0.158979	0.071429	1.0	0	0.0	0.0	0.0	
1	0.0	Clouds	0.173077	100	76.26	9.94	2022-10-26 08:16:51	1666772211	0.213084	0.035714	1.0	0	0.0	0.0	0.0	
2	0.0	Clouds	0.096154	100	76.26	9.94	2022-10-26 08:31:51	1666773111	0.266363	0.035714	1.0	0	0.0	0.0	0.0	
3	0.0	Clouds	0.096154	100	76.26	9.94	2022-10-26 08:46:51	1666774011	0.322790	0.035714	1.0	0	0.0	0.0	0.0	
4	0.0	Clouds	0.096154	100	76.26	9.94	2022-10-26 09:01:51	1666774911	0.381467	0.017857	1.0	0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1140	0.0	Clouds	0.230769	100	76.26	9.94	2022-11-08 14:59:15	1667919555	0.468670	0.500000	0.0	0	0.0	0.0	0.0	
1141	0.0	Clouds	0.230769	100	76.26	9.94	2022-11-08 15:44:15	1667922255	0.313749	0.500000	0.0	0	0.0	0.0	0.0	
1142	0.0	Clouds	0.230769	100	76.26	9.94	2022-11-08 15:59:15	1667923155	0.267441	0.500000	0.0	0	0.0	0.0	0.0	
1143	0.0	Clouds	0.230769	100	76.26	9.94	2022-11-08 16:59:15	1667926755	0.127954	0.500000	0.0	0	0.0	0.0	0.0	
1144	0.0	Clouds	0.230769	100	76.26	9.94	2022-11-08 17:29:19	1667928559	0.093132	0.500000	0.0	0	0.0	0.0	0.0	

1145 rows × 17 columns

Figure 39

Raw dataset for insat images

insat_table

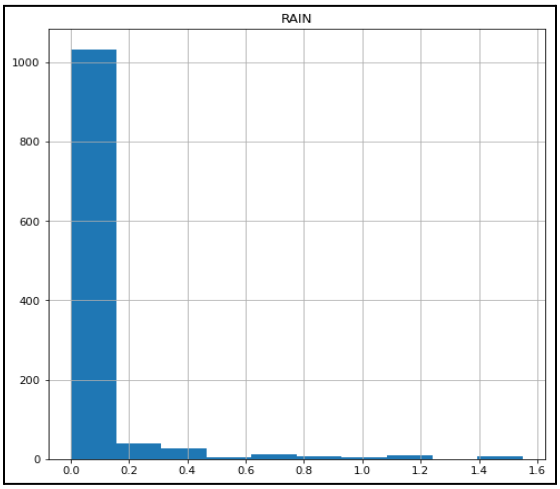
	Unnamed: 0	timestamp	pixel_count
0	0	2022-10-26 01:00:00	7
1	1	2022-10-26 01:16:00	5
2	2	2022-10-26 01:31:00	5
3	3	2022-10-26 01:46:00	5
4	4	2022-10-26 02:01:00	5
...	...	...	...
1265	1265	2022-11-08 08:59:00	26
1266	1266	2022-11-08 09:14:00	32
1267	1267	2022-11-08 09:29:00	32
1268	1268	2022-11-10 10:29:00	24
1269	1269	2022-11-10 10:44:00	24

1270 rows × 3 columns

The distribution of descriptive features was also analyzed. From Figure 40, it can be seen that rain had the most value with level 0 which shows that there was less rainfall. But mostly the rainfall was quite constant with minimalistic fluctuation.

Figure 40

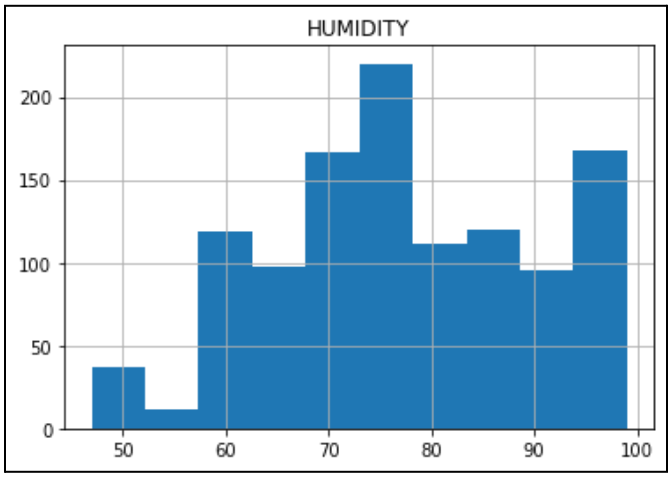
Histogram showing distribution of rainfall level



With reference to Figure 41, the humidity level showed a slightly normal distribution.

Figure 41

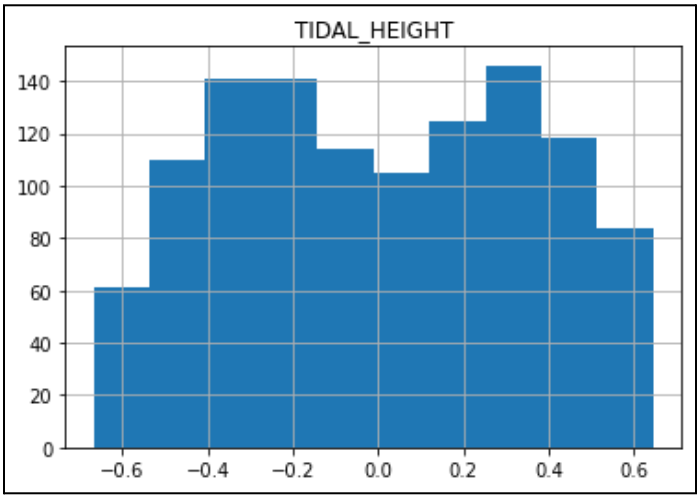
Histogram showing the distribution of humidity



From Figure 42, it was observed that tidal height was bimodal.

Figure 42

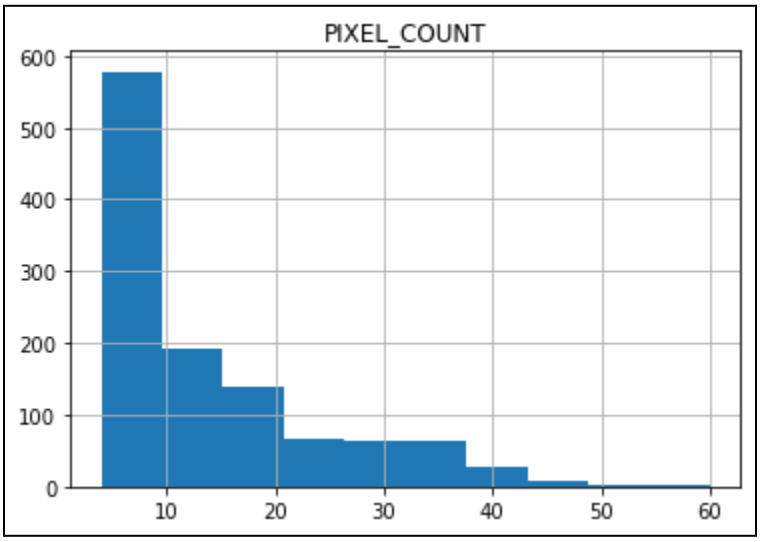
Histogram showing the distribution of tidal height



In Figure 43, it can be seen that pixel count had exponential distribution.

Figure 43

Histogram showing the distribution of pixel count



3.3.2 Data Cleaning

Tidal Data

The tidal data extracted from Marea API contained a lot of features that were irrelevant for the analysis like the disclaimer, status, origin, datums, source, copyright, etc. which can be seen in the Figure 44. These features had no importance while predicting floods. Hence these columns were noise and were dropped which can be seen from Figure 45.

Figure 44

Tidal dataset

	disclaimer	status	latitude	longitude	origin	datums	timestamp	datetime	unit	timezone	datum	extremes	heights	sou
0	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:00:18	2022-10-26 08:00:18+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666771218, 'height': -0.459390...	FES2(
1	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:16:51	2022-10-26 08:16:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666772211, 'height': -0.388316...	FES2(
2	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:31:51	2022-10-26 08:31:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666773111, 'height': -0.318326...	FES2(
3	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 08:46:51	2022-10-26 08:46:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666774011, 'height': -0.244201...	FES2(
4	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-10-26 09:01:51	2022-10-26 09:01:51+00:00	m	UTC	MSL	[{'timestamp': 1666787226, 'height': 0.5704027...	[{'timestamp': 1666774911, 'height': -0.167119...	FES2(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1268	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-11-08 16:29:16	2022-11-08 16:29:16+00:00	m	UTC	MSL	[{'timestamp': 1667929980, 'height': -0.557096...	[{'timestamp': 1667924956, 'height': -0.422229...	FES2(
1269	NOT SUITABLE FOR NAVIGATIONAL PURPOSES. Marea ...	200	76.26	9.94	{'latitude': 76.25, 'longitude': 9.9375, 'dist...	{'LAT': -0.84, 'HAT': 0.752, 'MLLW': -0.475000...	2022-11-08 16:44:15	2022-11-08 16:44:15+00:00	m	UTC	MSL	[{'timestamp': 1667929980, 'height': -0.557096...	[{'timestamp': 1667925855, 'height': -0.464931...	FES2(

Figure 45

Tidal dataset after dropping irrelevant columns

	latitude	longitude	timestamp	datetime	unit	timezone	datum	heights	extremes
0	76.26	9.94	2022-10-26 08:00:18	2022-10-26 08:00:18+00:00	m	UTC	MSL	[{'timestamp': 1666771218, 'height': -0.459390...	[{'timestamp': 1666787226, 'height': 0.5704027...
1	76.26	9.94	2022-10-26 08:16:51	2022-10-26 08:16:51+00:00	m	UTC	MSL	[{'timestamp': 1666772211, 'height': -0.388316...	[{'timestamp': 1666787226, 'height': 0.5704027...
2	76.26	9.94	2022-10-26 08:31:51	2022-10-26 08:31:51+00:00	m	UTC	MSL	[{'timestamp': 1666773111, 'height': -0.318326...	[{'timestamp': 1666787226, 'height': 0.5704027...
3	76.26	9.94	2022-10-26 08:46:51	2022-10-26 08:46:51+00:00	m	UTC	MSL	[{'timestamp': 1666774011, 'height': -0.244201...	[{'timestamp': 1666787226, 'height': 0.5704027...
4	76.26	9.94	2022-10-26 09:01:51	2022-10-26 09:01:51+00:00	m	UTC	MSL	[{'timestamp': 1666774911, 'height': -0.167119...	[{'timestamp': 1666787226, 'height': 0.5704027...
...	...	...	...	...	...	...	...	...	...
1268	76.26	9.94	2022-11-08 16:29:16	2022-11-08 16:29:16+00:00	m	UTC	MSL	[{'timestamp': 1667924956, 'height': -0.422229...	[{'timestamp': 1667929980, 'height': -0.557096...
1269	76.26	9.94	2022-11-08 16:44:15	2022-11-08 16:44:15+00:00	m	UTC	MSL	[{'timestamp': 1667925855, 'height': -0.464931...	[{'timestamp': 1667929980, 'height': -0.557096...
1270	76.26	9.94	2022-11-08 16:59:15	2022-11-08 16:59:15+00:00	m	UTC	MSL	[{'timestamp': 1667926755, 'height': -0.500146...	[{'timestamp': 1667929980, 'height': -0.557096...
1271	76.26	9.94	2022-11-08 17:14:25	2022-11-08 17:14:25+00:00	m	UTC	MSL	[{'timestamp': 1667927665, 'height': -0.527510...	[{'timestamp': 1667929980, 'height': -0.557096...
1272	76.26	9.94	2022-11-08 17:29:19	2022-11-08 17:29:19+00:00	m	UTC	MSL	[{'timestamp': 1667928559, 'height': -0.545890...	[{'timestamp': 1667929980, 'height': -0.557096...

1273 rows × 9 columns

The ‘heights’ column gave the required data for the analysis like timestamp, height of the tide and state of the height i.e., Rising tide or Falling tide. The timestamp was extracted and presented in the data frame with the column name timestamps. Height was extracted and renamed as tidal_height and the state was extracted and renamed as tidal_state. The tidal_height gave the information about height of the tide in meters and tidal_state gave information whether the tide was rising or falling. Other irrelevant columns like latitude, longitude, timezone, etc. were dropped and the data after extracting relevant features is as shown in Figure 46.

Figure 46

Tidal dataset after extracting relevant values from the existing columns

	timestamp	timestamps	tidal_height	tidal_state
0	2022-10-26 08:00:18	1666771218	-0.459391	RISING
1	2022-10-26 08:16:51	1666772211	-0.388316	RISING
2	2022-10-26 08:31:51	1666773111	-0.318326	RISING
3	2022-10-26 08:46:51	1666774011	-0.244202	RISING
4	2022-10-26 09:01:51	1666774911	-0.167120	RISING
...	...	...	...	...
1268	2022-11-08 16:29:16	1667924956	-0.422229	FALLING
1269	2022-11-08 16:44:15	1667925855	-0.464931	FALLING
1270	2022-11-08 16:59:15	1667926755	-0.500146	FALLING
1271	2022-11-08 17:14:25	1667927665	-0.527511	FALLING
1272	2022-11-08 17:29:19	1667928559	-0.545890	FALLING

1273 rows × 4 columns

The tidal data extracted from Marea API had no null or missing values for further processing. The sample code for cleaning tidal data is shown in Appendix B (Figure B4).

Weather Data

The weather data extracted from Open Weather API contained a lot of features that were irrelevant for the analysis like the wind, sunrise and sunset time, timezone, etc. These features had no importance while predicting floods. Hence these columns were considered as noise and were dropped which can be seen from Figure 47.

Figure 47

Rainfall dataset after dropping irrelevant columns

```
weather_data= weather_data[['weather','main','clouds','dt','rain']]
```

```
weather_data
```

	weather	main	clouds	dt	rain
0	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 271.89, 'feels_like': 266.75, 'temp_m...	{'all': 100}	1666771218	NaN
1	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 271.89, 'feels_like': 266.75, 'temp_m...	{'all': 100}	1666772211	NaN
2	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	{'all': 100}	1666773111	NaN
3	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	{'all': 100}	1666774011	NaN
4	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 272.08, 'feels_like': 266.87, 'temp_m...	{'all': 100}	1666774911	NaN
...	...	...	...	...	...
1268	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 273.05, 'feels_like': 271.53, 'temp_m...	{'all': 100}	1667924955	NaN
1269	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 273.1, 'feels_like': 273.1, 'temp_min...	{'all': 100}	1667925856	NaN
1270	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 273.1, 'feels_like': 273.1, 'temp_min...	{'all': 100}	1667926755	NaN
1271	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 273.1, 'feels_like': 273.1, 'temp_min...	{'all': 100}	1667927670	NaN
1272	[{'id': 804, 'main': 'Clouds', 'description': ...	{'temp': 273.1, 'feels_like': 273.1, 'temp_min...	{'all': 100}	1667928559	NaN

1273 rows × 5 columns

The ‘main’ key present in the ‘weather’ column described the type of weather (cloud, rain, snow) and the ‘description’ key within this column described the weather (overcast clouds, light rain, moderate rain, light snow, etc). Only these values were relevant from the weather column. So only the ‘main’ and the ‘description’ values were extracted, were renamed as ‘weather’ and ‘weather_description’ for each timestamp and the other key-value pairs from the ‘weather’ column were dropped. The humidity (percentage) column was also created by fetching the key values for humidity from the ‘main’ columns. The ‘clouds’ column was renamed as cloudiness (percentage). The ‘rain’ column displayed the amount of rain recorded (in mm) for 1 hour. ‘dt’ was the time in seconds from the epoch unix time (January 1, 1990 00:00:00) till the time at which the data was recorded. The dataset for the rainfall data after the changes were implemented and the noisy and irrelevant features were removed is as shown in Figure 48.

Figure 48

Rainfall dataset after extracting relevant values from the existing columns

	dt	rain	Weather_description	humidity	cloudiness
0	1666771218	0	overcast clouds	56	100
1	1666772211	0	overcast clouds	56	100
2	1666773111	0	overcast clouds	52	100
3	1666774011	0	overcast clouds	52	100
4	1666774911	0	overcast clouds	52	100
...	...	...	...	...	...
1268	1667924955	0	overcast clouds	59	100
1269	1667925856	0	overcast clouds	59	100
1270	1667926755	0	overcast clouds	59	100
1271	1667927670	0	overcast clouds	59	100
1272	1667928559	0	overcast clouds	59	100

1273 rows × 5 columns

Once the noisy data was removed, the missing values were checked in the dataset. It can be seen from Figure 49 that the 'rain' feature had around 1112 missing values. Rest all the features had no missing values. Rain feature had valid missing values. The missing values represented that there was no rain for the given period of time. This problem of missing values can be handled by replacing them with zeros as zeros rightly express that there was no rainfall recorded for that particular timestamp which can be seen in Figure 50. Also, the volume of rain for one hour was fetched and stored in place of the key-value pairs in the rain column which can be seen from Figure 51. The sample code for cleaning weather data is shown in Appendix B (Figure B5).

Figure 49

Count of null values for each feature in rainfall data

```
weather_data.isnull().sum()

dt                          0
rain                     1112
Weather                     0
Weather_description         0
humidity                    0
cloudiness                  0
dtype: int64
```

Figure 50

Data after replacing missing values with zero for rain feature

	dt	rain	Weather_description	humidity	cloudiness
114	1666873911	0.41	light rain	88	100
115	1666874811	0.41	light rain	88	100
116	1666875711	0.41	light rain	88	100
117	1666876611	0.41	light rain	88	100
118	1666877511	0.26	light rain	85	100
...	...	...	...	...	...
1031	1667708055	0.12	light rain	82	100
1032	1667708955	0.12	light rain	82	100
1157	1667825055	0.12	light rain	84	100
1158	1667825955	0.12	light rain	84	100
1159	1667826855	0.12	light rain	84	100

161 rows × 5 columns

Figure 51

Data with non-null rain values

	dt	rain	Weather_description	humidity	cloudiness
114	1666873911	0.41	light rain	88	100
115	1666874811	0.41	light rain	88	100
116	1666875711	0.41	light rain	88	100
117	1666876611	0.41	light rain	88	100
118	1666877511	0.26	light rain	85	100
...	...	...	...	...	...
1031	1667708055	0.12	light rain	82	100
1032	1667708955	0.12	light rain	82	100
1157	1667825055	0.12	light rain	84	100
1158	1667825955	0.12	light rain	84	100
1159	1667826855	0.12	light rain	84	100

161 rows × 5 columns

INSAT Images

Incomplete and missing data were already handled i.e., the raw images extracted from the Indian Meteorological website had no incomplete and missing data. A final check was conducted to find any incomplete or missing values by converting the image to a NumPy array. For all the raw images, the count of NANs was zero. Figure 52 is an example of one such raw image where such data was checked.

Figure 52

Handling incomplete or missing data in INSAT Images

```
np.isnan(arr).sum()
```

```
0
```

The primary region of focus for flood prediction and detection was the region of Kerala, India. Therefore, the images were cropped to remove other regions. The images were cropped using the Python Image Library. The noise aspect of the images such as grain was taken care of in the Indian Meteorological Website, where to improve the noise performance, facilities

collected a couple of samples (0.2 - sec or 0.4-sec step & dwell time) of the same region. Also, scientific data analysis software with noise reduction and image restoration functionalities was employed. A final check to crosscheck the grain was done using some online resources such as the Topaz Denoise AI Application. Later, ImageEnhance in the Python Image Library was used to change the contrast that helped to distinguish between the clouds and the ground in the satellite images.

In Figure 53, to predict the severity of the rainfall based on the image, pixel count with the color code [255, 255, 255] was taken into consideration. Once, the pixel counts were appended to a list and timestamps appended to another list, these lists were combined to form a dataframe to export the obtained data to the .csv format. The sample code for cropping, enhancing the images and extracting pixel counts is shown in Figure B6.

The datatype of the timestamp was converted to handle the inconsistency and facilitate the merge of this data with other data (tidal data and weather data).

Figure 53

Final Pixel Dataset

	timestamp	pixel_count
0	2022-10-26 01:00:00	7
1	2022-10-26 01:16:00	5
2	2022-10-26 01:31:00	5
3	2022-10-26 01:46:00	5
4	2022-10-26 02:01:00	5
...	...	...
1265	2022-11-08 08:59:00	26
1266	2022-11-08 09:14:00	32
1267	2022-11-08 09:29:00	32
1268	2022-11-10 10:29:00	24
1269	2022-11-10 10:44:00	24

1270 rows × 2 columns

3.4 Data Transformation

Once all the missing values and noisy data problems were handled in each dataset, the dataset from Open Weather API and Marea API was merged based on unix time. Inner join was performed so that there were no missing values. The merged data can be seen in Figure 54. The data was reduced to 1145 instances. Around 10% of instances were eliminated because of the missing value problem. The INSAT data was then merged to this merged tidal and rainfall data on timestamp. The sample code for merging the dataset is shown in Appendix B (Figure B7).

The irrelevant columns like the timestamps repeated in the merged datasets were dropped and the final data cleaning on the merged dataset was completed which can be seen from Figure 55.

Figure 54

Merged tidal, rainfall and satellite image data

	dt	rain	Weather_description	humidity	cloudiness	timestamp	timestamps	tidal_height	tidal_state	pixel_count
0	1666771218	0.0	overcast clouds	56	100	2022-10-26 08:00:18	1666771218	-0.459391	RISING	8
1	1666772211	0.0	overcast clouds	56	100	2022-10-26 08:16:51	1666772211	-0.388316	RISING	6
2	1666773111	0.0	overcast clouds	52	100	2022-10-26 08:31:51	1666773111	-0.318326	RISING	6
3	1666774011	0.0	overcast clouds	52	100	2022-10-26 08:46:51	1666774011	-0.244202	RISING	6
4	1666774911	0.0	overcast clouds	52	100	2022-10-26 09:01:51	1666774911	-0.167120	RISING	5
...	...	...	...	...	...	...	...	...	...	...
1140	1667919555	0.0	overcast clouds	59	100	2022-11-08 14:59:15	1667919555	-0.052566	FALLING	32
1141	1667922255	0.0	overcast clouds	59	100	2022-11-08 15:44:15	1667922255	-0.256078	FALLING	32
1142	1667923155	0.0	overcast clouds	59	100	2022-11-08 15:59:15	1667923155	-0.316911	FALLING	32
1143	1667926755	0.0	overcast clouds	59	100	2022-11-08 16:59:15	1667926755	-0.500146	FALLING	32
1144	1667928559	0.0	overcast clouds	59	100	2022-11-08 17:29:19	1667928559	-0.545890	FALLING	32

1145 rows × 10 columns

Figure 55

Merged data after data cleaning

	rain	Weather_description	humidity	cloudiness	tidal_height	tidal_state	pixel_count
0	0.0	overcast clouds	56	100	-0.459391	RISING	8
1	0.0	overcast clouds	56	100	-0.388316	RISING	6
2	0.0	overcast clouds	52	100	-0.318326	RISING	6
3	0.0	overcast clouds	52	100	-0.244202	RISING	6
4	0.0	overcast clouds	52	100	-0.167120	RISING	5
...	...	...	...	...	...	...	...
1140	0.0	overcast clouds	59	100	-0.052566	FALLING	32
1141	0.0	overcast clouds	59	100	-0.256078	FALLING	32
1142	0.0	overcast clouds	59	100	-0.316911	FALLING	32
1143	0.0	overcast clouds	59	100	-0.500146	FALLING	32
1144	0.0	overcast clouds	59	100	-0.545890	FALLING	32

1145 rows × 7 columns

A target binary feature was created for predicting floods if the observed humidity was greater than 90 %, tidal height greater than 0.6 and when the weather description showed light and moderate rain. The dataset with the target feature is as shown in Figure 56.

Figure 56

Merged dataset after adding target feature

	rain	Weather_description	humidity	cloudiness	tidal_height	tidal_state	pixel_count	target
0	0.0	overcast clouds	56	100	-0.459391	RISING	8	0
1	0.0	overcast clouds	56	100	-0.388316	RISING	6	0
2	0.0	overcast clouds	52	100	-0.318326	RISING	6	0
3	0.0	overcast clouds	52	100	-0.244202	RISING	6	0
4	0.0	overcast clouds	52	100	-0.167120	RISING	5	0
...	...	...	...	...	...	...	...	...
1140	0.0	overcast clouds	59	100	-0.052566	FALLING	32	0
1141	0.0	overcast clouds	59	100	-0.256078	FALLING	32	0
1142	0.0	overcast clouds	59	100	-0.316911	FALLING	32	0
1143	0.0	overcast clouds	59	100	-0.500146	FALLING	32	0
1144	0.0	overcast clouds	59	100	-0.545890	FALLING	32	0

1145 rows × 8 columns

After merging the data, tidal_state and weather_description were object data types and the machine learning algorithm works for numeric data only. As categorical features were not ordinal, so one hot encoding was used to convert these nominal data into numeric types. For this pd.get_dummies was used to convert these columns into numeric types. By using this function, a new column was created for each level in categorical feature and the value ‘one’ showed the presence of that level for a particular instance. One of the columns for each categorical feature was dropped to avoid a dummy variable trap. Figure 57 shows the one hot encoding for categorical features.

Figure 57

One hot encoded data set

tidal_state_HIGH TIDE	Weather_description_broken clouds	Weather_description_light rain	Weather_description_light snow	Weather_description_moderate rain	Weather_description_overcast clouds
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
1	0	0	0	0	1
...	...	...	...	...	...
1	0	1	0	0	0
1	0	1	0	0	0
1	0	1	0	0	0
1	0	1	0	0	0
1	0	1	0	0	0

Next step included scaling the data, as there was a difference in magnitude between different descriptive features. For this, an array named X was defined which included all relevant descriptive features which were to be included in the predictive model. After this, a min-max scaler was used to scale the data in the range (0,1). Figure 58 shows the scaled data.

Figure 58

Scaled data in the range (0,1)

```
scaler= MinMaxScaler()
X= scaler.fit_transform(X)
```

```
X
```

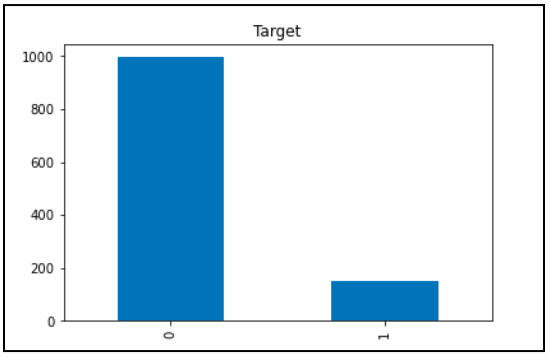
```
array([[0.03846154, 0.94235958, 1.        , ..., 0.        , 1.        ,
        0.2       ],
       [0.03846154, 0.94235958, 1.        , ..., 0.        , 1.        ,
        0.12      ],
       [0.03846154, 0.94235958, 1.        , ..., 0.        , 1.        ,
        0.12      ],
       ...,
       [0.98076923, 0.79132886, 1.        , ..., 1.        , 0.        ,
        0.33333333],
       [0.98076923, 0.79132886, 1.        , ..., 1.        , 0.        ,
        0.32      ],
       [0.98076923, 0.79132886, 1.        , ..., 1.        , 0.        ,
        0.32      ]])
```

3.4.1 Data Regularization

The target class was highly imbalanced as the data was real-time and there were few instances that were contributing towards the likelihood of having floods. For this reason, oversampling on training data was implemented so that the target class was balanced. Figure 59 shows the imbalanced target class.

Figure 59

Target Class

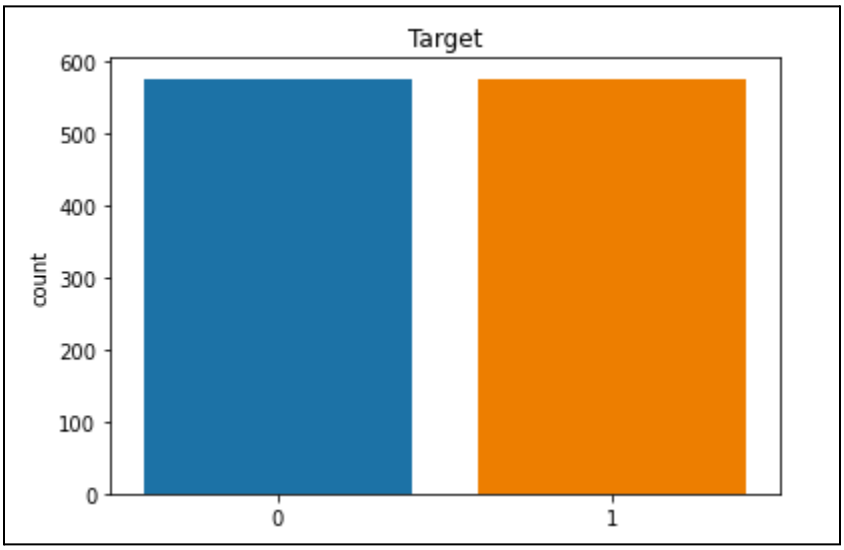


For this reason, oversampling technique (i.e. smote from sklearn.imblearn) was used to make the class balanced. This was important to minimize the bias introduced in the model due to the preference of the majority class. In SMOTE, synthesized data was generated to make the

class balanced. In this, the majority target class count was the target and the function generated synthesized data in minority class to make the count equal. Figure 60 shows the balanced data.

Figure 60

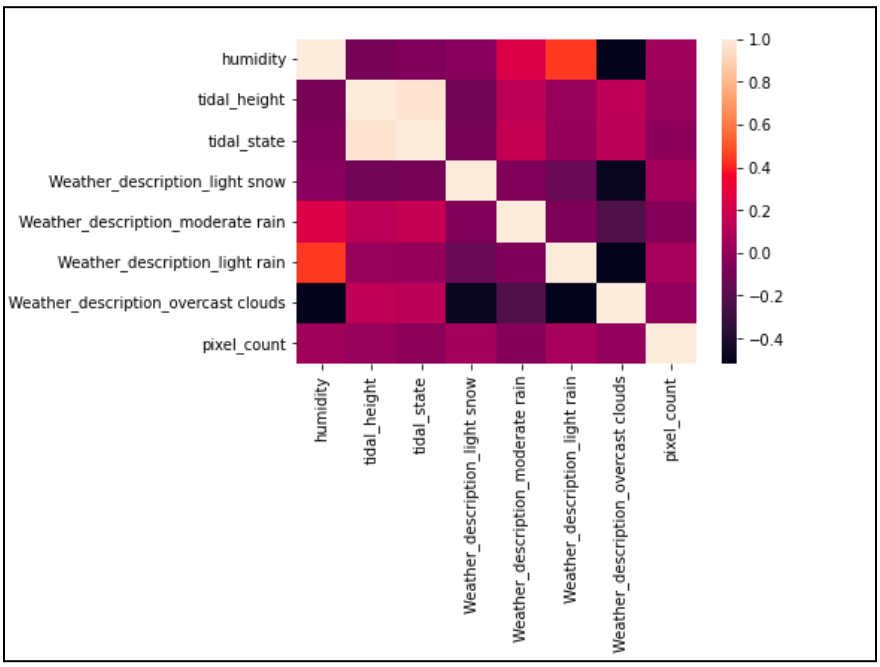
Data Augmentation using Over-Sampling Technique (SMOTE)



With reference to PCA, PCA is usually performed by orthogonally transforming correlated data into the uncorrelated form. PCA is done when either the dimensionality is high or when there is multicollinearity in the dataset. With respect to Figure 61, there was no multicollinearity and also the features were one hot encoded so PCA cannot be done as the encoded data were not in the same coordinate plane. To implement PCA, it is necessary to have data points in the same coordinate plane.

Figure 61

Heat-Map showing correlation coefficients for descriptive features



Singular Value Decomposition also is a method for dimensionality reduction. For this case, a smaller number of features were present and all the features were relevant for the analysis and prediction. A broader view for flood predictions which included rainfall data, tidal data and satellite images to make predictions was considered for getting a more accurate and more reliable model. Any form of dimensionality reduction was not required and hence SVD was not implemented for this analysis.

3.5 Data Preparation

Once cleaning, filtering and data transformation was performed, the next step was to segment the data for the modeling. The data was split into training and testing using 80:20 ratio. The quantity and quality of the dataset used to train, and test helped to build the model with more accuracy.

The scikit-learn library in python has a function ‘test train split’ which splits the dataset into the train, test, and validation datasets. This function randomly splits the data into different sets and each of the sets performs a specific task such as the training set trains the model, the validation set validates and evaluates the performance of the model and test data tests the model built for its accuracy before deployment.

In this project research, time series data was used where each row of data was dependent on the previous row. As a result, the scikit-learn library wasn’t used to split data, as the function randomly splits data into different sets. Dealing with time-series data and maintaining the continuity of the data, the first 80% of the data was used as training data (Figure 62) and the rest 20% was used as a testing dataset (Figure 63).

Further to which SMOTE technique was implemented to do oversampling on the 80% split of training data which helped to create a larger data sample, which was a balanced dataset considering approximately equal instances of target level values as shown in Figure 64, avoiding any overfitting of the data by the model built. Once the respective model was fed with the data that had been split, the training set helped to train the model while the test data helped to test the accuracy of the model built. The k -fold cross validation technique was also used as the dataset size was small and to evaluate the performance of the data on the cross-validated data. The number of folds was set to three. The target variable in the project research was Flood, it had two levels 0 and 1. 1 indicated the presence of the flood and zero represented the absence of flood.

Figure 62

The Training Dataset

x_train

	rain	humidity	cloudiness	tidal_height	pixel_count	tidal_state_RISING	Weather_description_broken clouds	Weatl
0	0.0	0.173077	1.000	0.158979	0.071429	1.0	0	
1	0.0	0.173077	1.000	0.213084	0.035714	1.0	0	
2	0.0	0.096154	1.000	0.266363	0.035714	1.0	0	
3	0.0	0.096154	1.000	0.322790	0.035714	1.0	0	
4	0.0	0.096154	1.000	0.381467	0.017857	1.0	0	
...	...	...	...	...	...	...	...	
911	0.0	0.538462	0.950	0.427510	0.321429	0.0	0	
912	0.0	0.538462	0.950	0.387400	0.321429	0.0	0	
913	0.0	0.903846	0.975	0.348877	0.339286	0.0	0	
914	0.0	0.903846	0.975	0.312611	0.339286	0.0	0	
915	0.0	0.903846	0.975	0.279249	0.303571	0.0	0	

916 rows × 10 columns

y_train

	target
0	0
1	0
2	0
3	0
4	0
...	...
911	0
912	0
913	0
914	0
915	0

916 rows × 1 columns

Note. 80% of the data is split for training the model

Figure 63

The Testing Dataset

x_test

	rain	humidity	cloudiness	tidal_height	pixel_count	tidal_state_RISING	Weather_description_broken clouds	Weather_descr
916	0.0	0.903846	0.975	0.249399	0.303571	0.0	0	
917	0.0	0.961538	0.975	0.223616	0.321429	0.0	0	
918	0.0	0.961538	0.975	0.202388	0.321429	0.0	0	
919	0.0	0.961538	0.975	0.175149	0.178571	0.0	0	
920	0.0	0.865385	1.000	0.169679	0.107143	0.0	0	
...	...	...	...	...	...	...	...	
1140	0.0	0.230769	1.000	0.468670	0.500000	0.0	0	
1141	0.0	0.230769	1.000	0.313749	0.500000	0.0	0	
1142	0.0	0.230769	1.000	0.267441	0.500000	0.0	0	
1143	0.0	0.230769	1.000	0.127954	0.500000	0.0	0	
1144	0.0	0.230769	1.000	0.093132	0.500000	0.0	0	

229 rows × 10 columns

y_test

	target
916	0
917	0
918	0
919	0
920	0
...	...
1140	0
1141	0
1142	0
1143	0
1144	0

229 rows × 1 columns

Note. 20% of the data is split for testing the model

Figure 64

Training Dataset after SMOTE

	rain	humidity	cloudiness	tidal_height	pixel_count	tidal_state_RISING	broken_
0	0.000000	0.173077	1.0	0.158979	0.071429	1.0	
1	0.000000	0.173077	1.0	0.213084	0.035714	1.0	
2	0.000000	0.096154	1.0	0.266363	0.035714	1.0	
3	0.000000	0.096154	1.0	0.322790	0.035714	1.0	
4	0.000000	0.096154	1.0	0.381467	0.017857	1.0	
...	...	...	...	...	...	...	
1555	0.180000	0.903846	1.0	0.454785	0.232267	0.0	
1556	0.621554	0.980769	1.0	0.294240	0.054496	0.0	
1557	1.000000	0.961538	1.0	0.637106	0.275355	0.0	
1558	0.132733	0.871772	1.0	0.309741	0.153794	1.0	
1559	0.370000	0.980769	1.0	0.712106	0.035714	0.0	

1560 rows × 10 columns

Note. Oversampling of training data using SMOTE technique to have a balanced training data.

3.6 Data Statistics

The Tidal data fetched from Marea API and the rainfall data fetched from Open Weather API had around 1273 instances and 15 columns each whereas 1270 images were fetched from the IMD website which constituted the data from October 26, 2022, to November 11, 2022. In the pre-processing steps, only relevant features were kept and the rest features were dropped from each data source. After this step, the number of columns for tidal data reduced from fifteen to four and for rainfall data reduced from fifteen to five. The number of instances remained the same as there were no missing values for our data. In the pre-processing steps for the satellite images, the images were cropped and enhanced and pixel count for each image was extracted to create a dataframe that included three columns which were the ID, timestamp, and the pixel count for the mentioned timestamp. A dataframe with 1270 instances and three columns was

then created from these images. In the data transformation step, all the data frames were merged into one based on the timestamp. Inner join was used to take care of the missing values. The number of instances from the merged data was reduced to 1145 and columns to 10 altogether. A final cleaning was done on the merged dataset to extract only relevant features and a target feature was also added. After a final cleaning, 1145 instances, and eight features were present. One hot encoding was done to consider the categorical features into the analysis. Hence the number of columns increased to 11. In the data transformation phase, data augmentation was performed only on the training dataset. Hence, the data was first divided into training and testing using 80:20 ratio. The training dataset had the size of 916 X 11. After performing SMOTE technique, the number of instances of training data increased to 1560. The testing dataset size was 229 X 11. Table 18 shows a summary for the data statistics for each pre-processing step.

Table 18

Summary of Data preprocessing steps and the size of data in each step

Stage	Dataset	Statistics (Rows x Columns)
Raw	Tidal data	1273 X 15
	Rainfall data	1273 X 15
	Insat Images	1270 images
Preprocessing – Data Cleaning	Tidal data	1273 X 4
	Rainfall data	1273 X 5
	Insat Images	1270 X 3
Data Transformation	Merging data from three sources into one (merged data)	1145 X 10
	Cleaning the merged data	1145 X 8
	One code encoding	1145 X 11
Preparation	Training	916 X 11
Data Preparation and Data Transformation	Data Augmentation (SMOTE) on Training set	1560 X 11
Data Preparation	Testing	229 X 11

Figures 65, 66, 67 and 68 show the data statistics visually for understanding the number of instances/features for each step from raw dataset to data preparation step.

Figure 65

Number of instances in raw dataset and after data pre processing

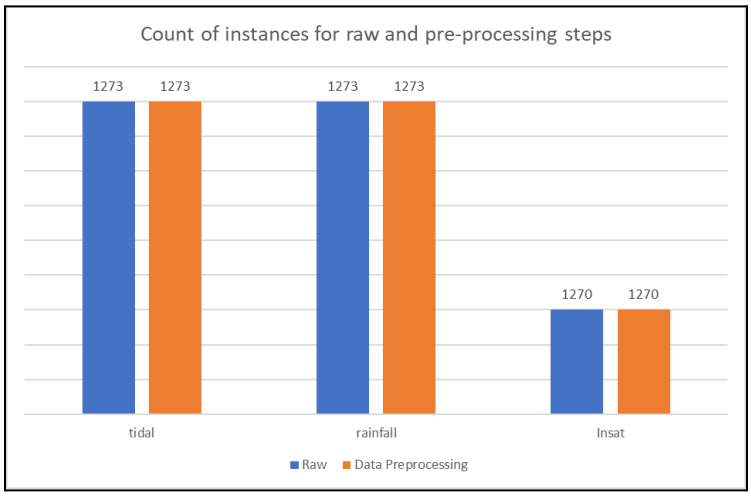


Figure 66

Number of features in raw dataset and after data pre processing

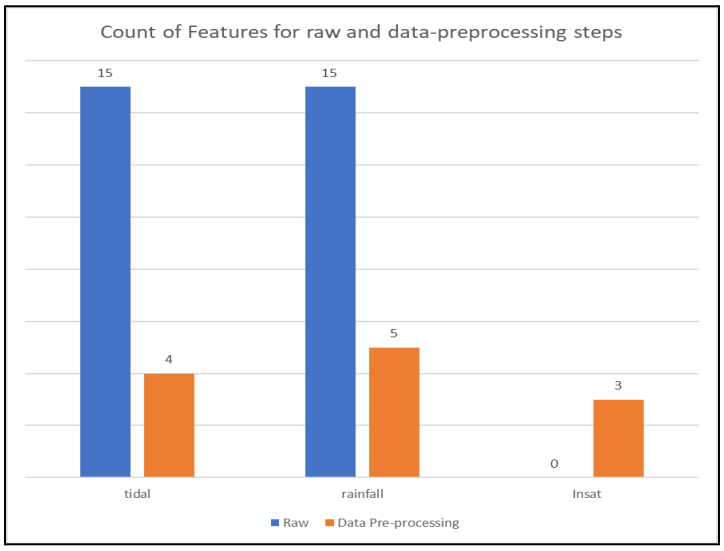


Figure 67

Number of features for data transformation phase

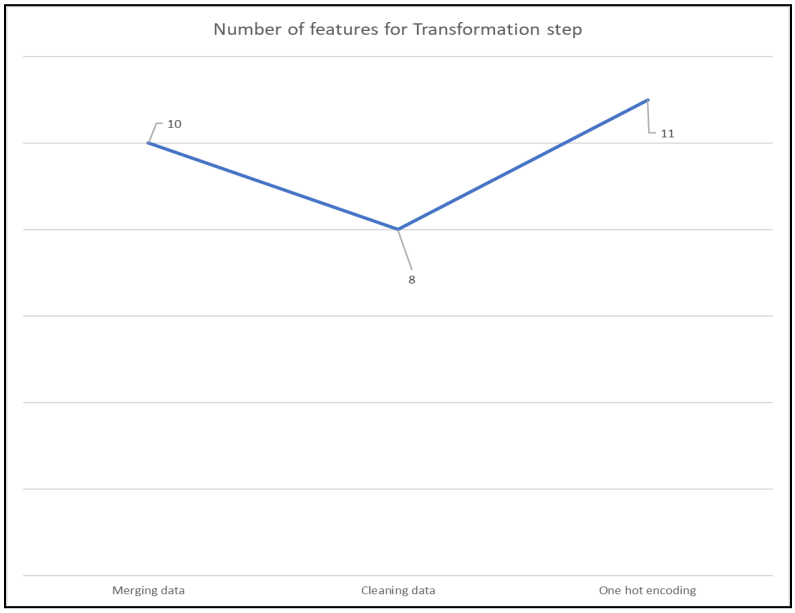
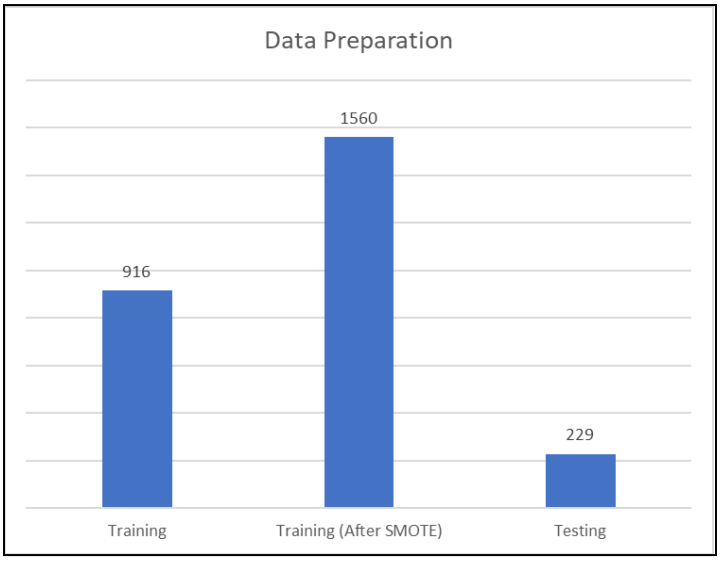


Figure 68

Number of instances for training and testing dataset



3.7 Data Analytics

To draw scatter plots among various descriptive features, a new dataframe was created that contained only numeric features and then scatterplots were created using pairplots. As shown in Figure 69, it can be seen that there was no such correlation between descriptive features. This also validated that there was no multicollinearity in the data. To cross verify, a heat map was also plotted showing coefficients between various descriptive features as shown in Figure 70. From the heatmap it was observed that correlation coefficient was quite low among all descriptive features.

Figure 69

Pair plot between various descriptive features

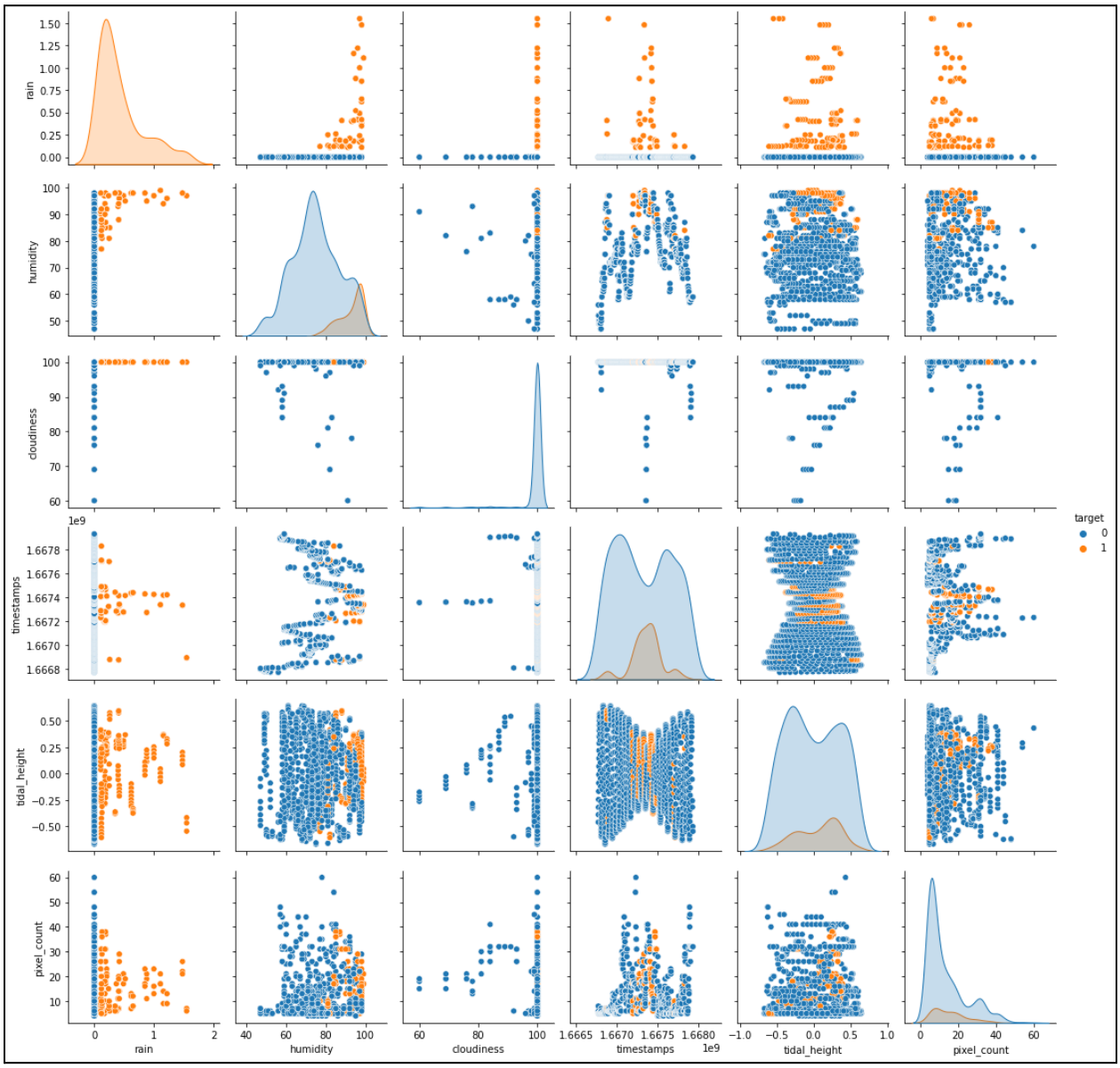
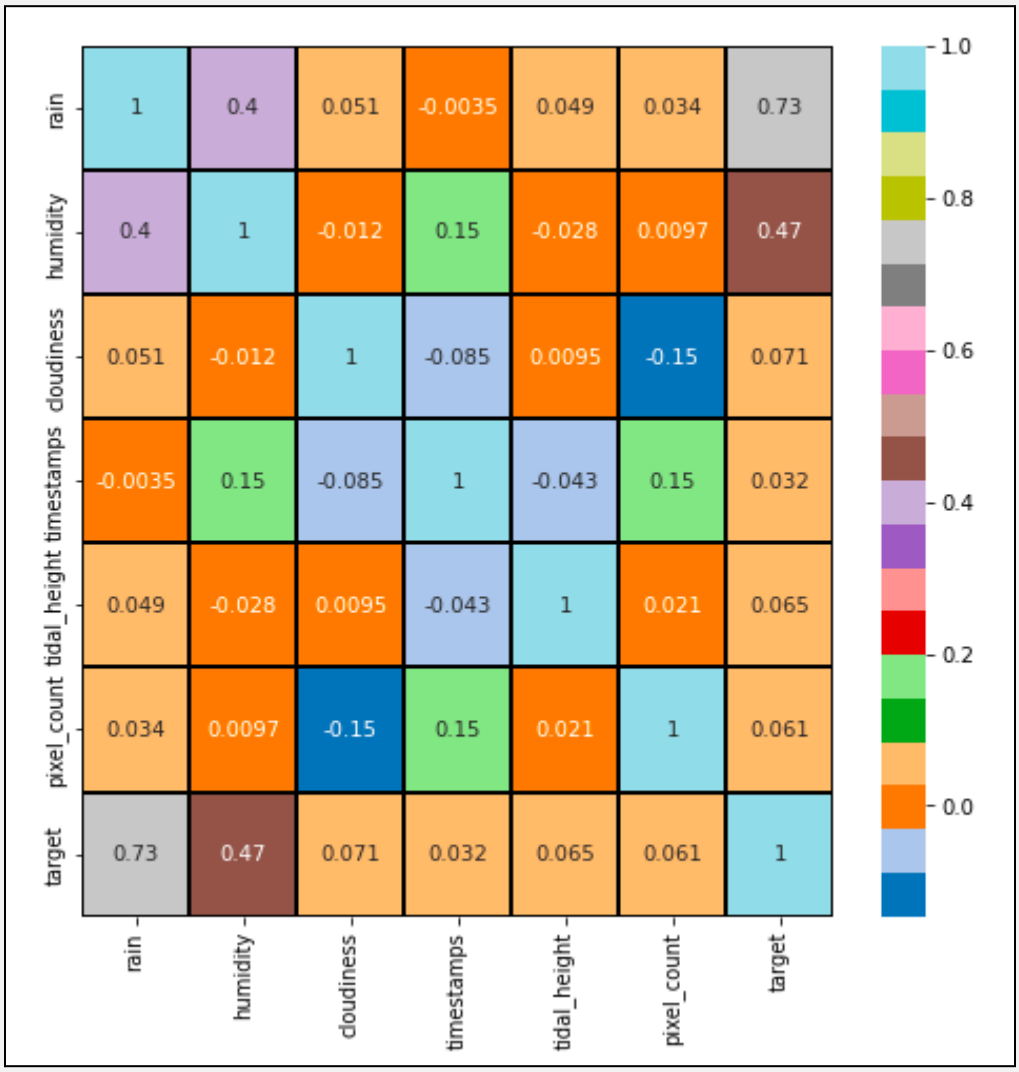


Figure 70

Heat Map showing correlation between descriptive features



A time series line graph was plotted that showed the trend for tidal height, humidity and pixel count as shown in Figure 71. It was observed that tidal height showed a decreasing and increasing trend with lots of fluctuations for both the levels in the target feature. However, when there was a flood, tidal height rose dramatically. With reference to pixel count in Figure 71, it was observed that pixel count varied rapidly for each time interval. Also, with reference to

humidity, from Figure 72, it was observed that when there was a flood, the humidity level was almost constant i.e. between timestamp 16668 and 16672. Lastly, it was clearly evident from Figures 72 and 73 that when there was a flood, rainfall level was quite high as compared to non-flood situations.

Figure 71

Line graph showing trend for tidal height and pixel count

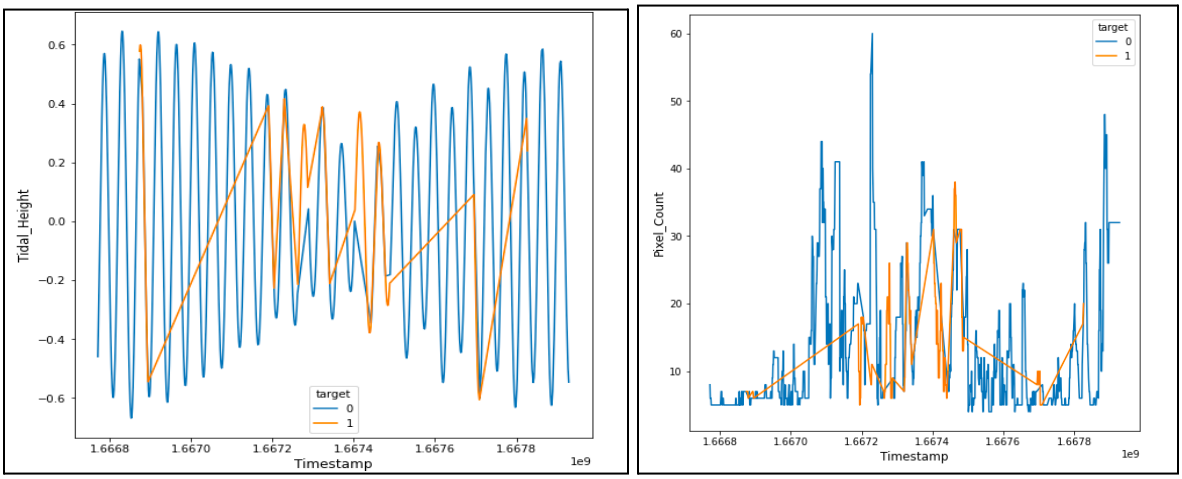
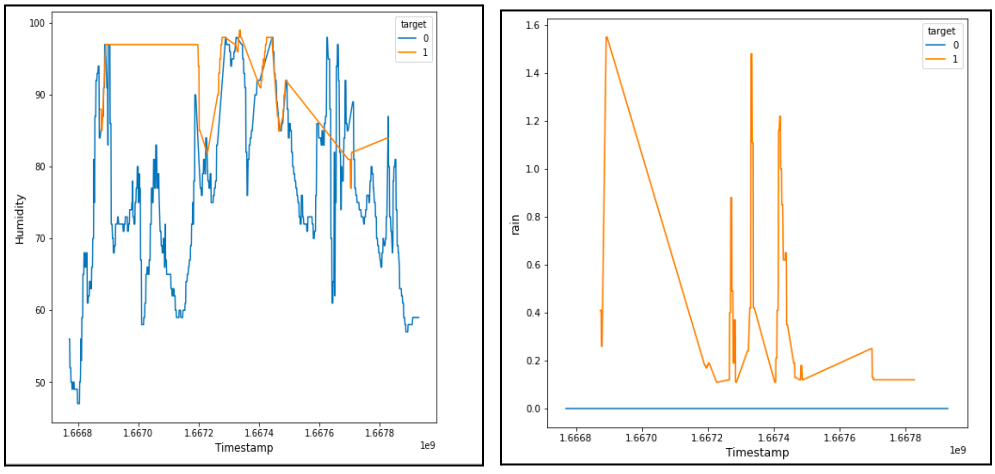


Figure 72

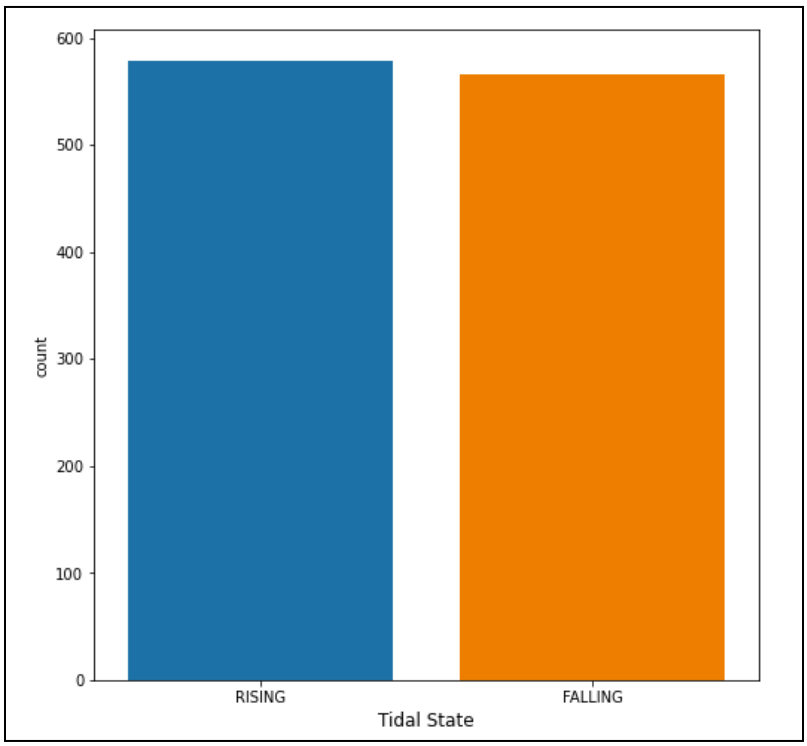
Line graph showing trend for humidity and rain



A univariate plot (count plot) was plotted that showed frequency for tidal state i.e., rising and falling as shown in Figure 73. From this figure, it was observed that the frequency of both the states was quite similar.

Figure 73

Count plot showing frequency for Tidal State



The five-point summary plot for pixel count, humidity, and tidal height was also plotted. From the Figure 74, it was clearly seen that pixel count and target were correlated. Also, there were a few outliers observed when target state= 0. From Figure 75, it was observed that the q2 (median value) was quite high for humidity when the flood happened. However, with reference to tidal height (Figure 76), the max value of both states was quite close but there was a significant difference between the median value in both states.

Figure 74

Box plot showing frequency for pixel count wrt target

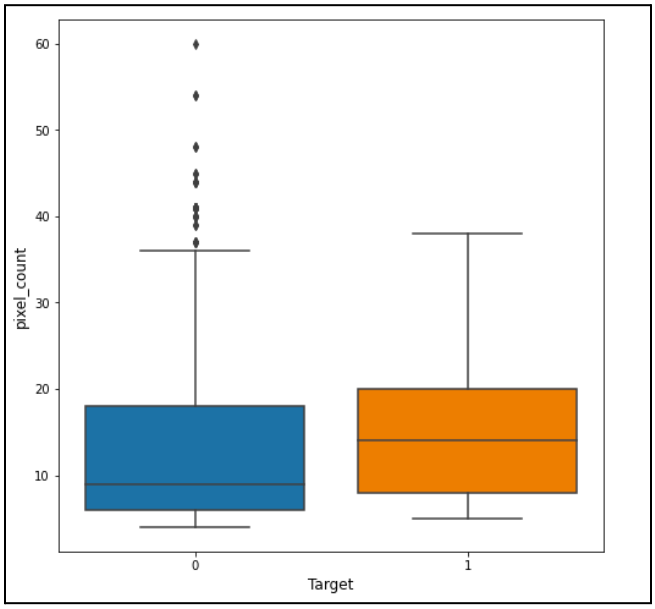


Figure 75

Box plot showing frequency for humidity with respect to target

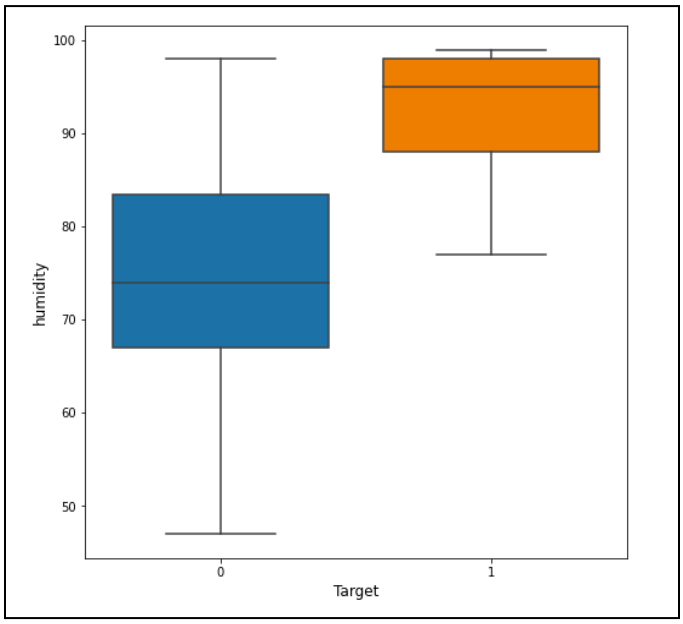
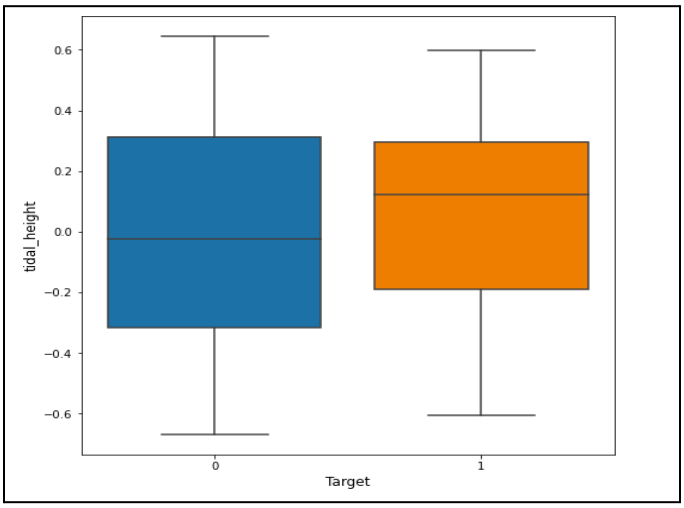


Figure 76

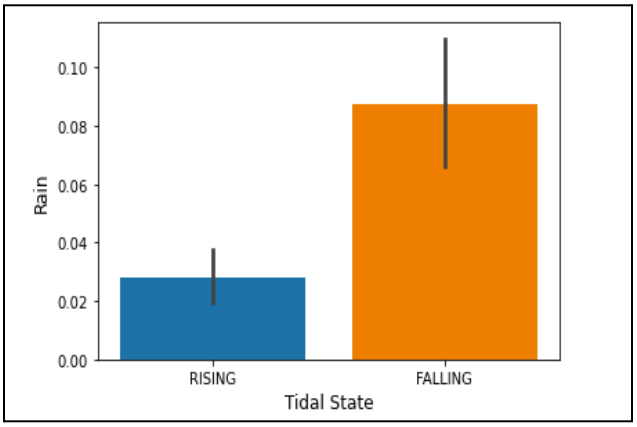
Box plot showing frequency for tidal height with respect to target



Bar plot was drawn to see the comparison between rain level and tidal state and it was evident from Figure 77 that rainfall was high when the tidal state was ‘falling’.

Figure 77

Bar plot showing frequency for tidal state wrt rain



4. Model Development

4.1 Model Proposals

To compare the performance between supervised and unsupervised learning for flood predictions, multiple models were implemented. These are Random Forest model, Support Vector model, XGBoost model and KMeans Clustering. The performance and working methodology of all these models is discussed below.

4.1.1 Random Forest Classifier

The model chosen to predict floods was Random Forest Classifier which is used for both classification and regression. It is a bagging ensemble technique based on decision trees. Several decision trees are combined into one in this supervised machine learning model for an ensemble method which was done because a prediction based on a single decision tree may be inaccurate and a combined average from multiple decision trees will enhance the model's performance.

Literature Review.

A thorough examination and research on the Random Forest model is conducted to understand how the model helps in predicting floods, along with its strengths and limitations. Few of the research work conducted by different authors are discussed in this section.

(Ghorpade et al., 2021) mentioned in their paper, floods are commonly occurring natural disasters due to which many humans have lost their life, livelihood has been destroyed and in turn has a huge impact on national economics. The objective of this paper was to demonstrate and review a few of the major machine learning algorithms that most professionals can use for forecasting floods. They have noticed that using historic climate data and other region-specific parameters to train machine learning based models are very useful in forecasting floods. They also state the fact that machine learning models outperforms the previously used statistical

models to forecast floods based on recent studies. The reason that these machine learning models have gained popularity is because they can handle complex data and develop relationships between several parameters. In this paper, authors discuss and provides insights of accuracy and performance of Linear regression, Decision Tree (DT), Ensemble Learning-Random Forest, Artificial Neural Network (ANN), Support Vector Machines (SVM), and K-Nearest Neighbors (KNN) models used for flood predictions. They have used Root Mean Square Error and Mean Square Error to evaluate these models. In conclusion, they have found that Long Short-Term Memory (LSTM) which is an evolution of Recurrent Neural Network (RNN) is better for predicting floods and regional data plays a major role in enhancing the training process of these models which in turn gives a more robust solution.

(Hashi et al., 2021) defines flood as water overflowing onto normally dry ground or an increase in water that significantly endangers human life. It is known to cause significant financial loss to goods and properties in addition to endangering human lives. The main goal of their study was to suggest a novel and reliable method for a real-time flood detection system based on Deep Learning, Naive Bayes J48, Random Forest, and Convolutional Neural Networks to measure water level and flood measurements with potential humanitarian consequences before they happen. In-depth research is done on how to simply simulate a revolutionary approach of detecting water levels utilizing an Arduino with GSM modems based hybrid model. In this study, there were two significant pieces of work: First, observe the river in real-time while gathering data for a dataset using the Arduino and GSM devices. The second stage is to analyze the gathered data and apply it to the three chosen algorithms to determine the water level accuracy that is enhancing the efficacy of the flood detection techniques. Based on their analysis, Table 19

shows classifier output for their classification and the accuracy by class of the various classification techniques is shown in Table 20. As result, they discovered that the Naive Bayes algorithm, which is crucial, performs at 88.4% accuracy, whereas the Random Forest algorithm achieves 98.7% accuracy when compared to other classification methods. Additionally, J48 achieved 84.2% accuracy, which is comparable to the Naive Bayes algorithm but slightly less accurate. In terms of the CNN method, it achieved 87% accuracy, which appears to be less accurate at first glance, but performed better overall in terms of other metrics like precision or recall. Additionally, it was discovered that applying typical NLP techniques had a negative impact on the model's performance. In other words, it's critical for studies like this one to maintain the data's simplicity. Finally, authors suggest that their architecture could be improved later by including video surveillance and GPS tracking of the deployed equipment and clustering algorithms could be used to enhance the output of the model.

Table 19

Performance metrics for various models

Parameters	Methods		
	Random Forest	**Naive Bayes**	**J48**
Correctly classified instance	98.7%	88.4%	84.2%
Incorrect classified instances	22.8%	2.8%	2.9%
Root mean squared error	0.0904	0.1387	0.1970
Total number of instances	1000	1000	1000

Note. Table of Detected water level in term of three algorithm reprinted from “Real-time Flood Dectection System Based on Machine Learning Algorithms with Empasis on Deep Learning”, by Hashi, A. O., Abdirahman, A. A., Elmi, M. A., Hashi, S. Z. M., & Rodriguez, O. E. R. (2021),

International Journal of Engineering Trends and Technology, *69*(5), p. 253 (https://doi.org/10.14445/22315381/ijett-v69i5p232). Copyright 2021 Seventh Sense Research Group.

Table 20

Performance metrics for Machine Learning models

Method	Accuracy by Class		
	True Positive	True Negative	Recall
Random Forest	0.989	0.004	0.976
Naive Bayes	0.886	0.014	0.888
J48	0.842	0.021	0.842

Note. Table of Detected water level in term of three algorithm reprinted from "Real-time Flood Dectection System Based on Machine Learning Algorithms with Empasis on Deep Learning", by Hashi, A. O., Abdirahman, A. A., Elmi, M. A., Hashi, S. Z. M., & Rodriguez, O. E. R. (2021), *International Journal of Engineering Trends and Technology*, *69*(5), p. 253 (https://doi.org/10.14445/22315381/ijett-v69i5p232). Copyright 2021 Seventh Sense Research Group.

(Garcia et al., 2015) have mounted a real-time urban flood monitoring system in Earnshaw and San Diego Streets. They have used a ground-based pressure sensor and a rain gauge as part of the system, and these are both connected to a nearby created data recorder with wireless capabilities over a GPRS network. Data is received by TCP servers from the stations and processes it. Then the visual data and real-time flood updates are announced through mobile and web services. To give vehicle drivers and other people an early warning advice, a system for

anticipating floods was put into place utilizing the Random Forest algorithm. They have chosen Random Forest as the solution model because it is simpler to develop and has lower processing requirements than ANN (Artificial Neural Network) while still offering consistent prediction performance. They have taken only days with rain and flood level data into account for the training set since not all day's experience rain and flooding. The Random Forest classifier receives the training set as input and produces a prediction model with an OOB score of 0.9375, indicating good predictive results. They then tested the model against 2 previous datasets and the projected values closely matched the pattern of the measured flood levels. The prediction model performed well with the computed coefficient of determination R2 greater than 0.90 for both the datasets. They mentioned that the model understates the rise in the flood level, but the coefficient of determination of the predicted values showed outstanding prediction accuracy. Finally, they suggest that, to improve the system for anticipating floods, extra algorithm optimization work can be done as well as collecting flood level data in the specific area.

(Motta et al., 2021) emphasizes that the environmental disasters are predicted to become more severe and frequent because of extreme weather which is one of several repercussions of climate change. In addition, it is anticipated that metropolitan areas will expand dramatically over the coming years, necessitating the implementation of suitable defenses against such dangers, particularly flooding. This paper proposes an enhanced data-driven methodology that combines a Machine Learning classifier with GIS statistics. By offering geospatial analytics for the entire city, this two-step strategy will be able to improve the spatial depiction in the outcome produced by the predictive model. The modules NumPy, SciPy, pandas, scikit-learn, and imbalanced-learn were used by them to create the Machine Learning models with Python.

Regarding the GIS layer, ArcGIS Pro's Spatial Analysis toolbox was utilized. They have implemented six classification models: Logistics Regression (LR), Support Vector Machine (SVM), Gaussian Naïve-Byes (NB), Random Forest (RF), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP) and their performance was assessed using the respective scikit-learn implementations. The results from testing and training each predictive model revealed that non-linear models performed better overall, particularly when it came to reducing the sensitivity for the minority class (i.e., Recall). Additionally, the techniques employed in this paper produced superior results for the Random Forest model with a Matthews Correlation Coefficient of 0.77 than other non-linear models like MLP and KNN. The accuracy of linear models like LR and SVM looked to be maximized at the expense of recall. In contrast, although having the highest Recall, the NB model produced the worst outcomes because of its poor Accuracy. Based on the hotspot areas with a higher Flood Risk Index noted in the historical data, the GIS model was able to modify the sensitivity of the predictions made by the ML model. When they used equivalent weights for both Random Forest and GIS models, the model sensitivity reached 0.84. Finally, as part of future work, they suggest more research on the weights used to calculate the Flood Risk Index for improving the model's predictive ability.

(Tavus et al., 2020) states that flood mapping is crucial for flood modeling as well as risk and hazard investigation, and it can be done utilizing information from optical and microwave satellite sensors. Even if cloud coverage is a constraint of optical image-based flood analysis approaches, they are often employed for flood evaluations. Synthetic Aperture Radar (SAR) satellite sensors have gained popularity as a source of data for flood detection due to their increased temporal availability and spatial resolution. On the other hand, its processing could

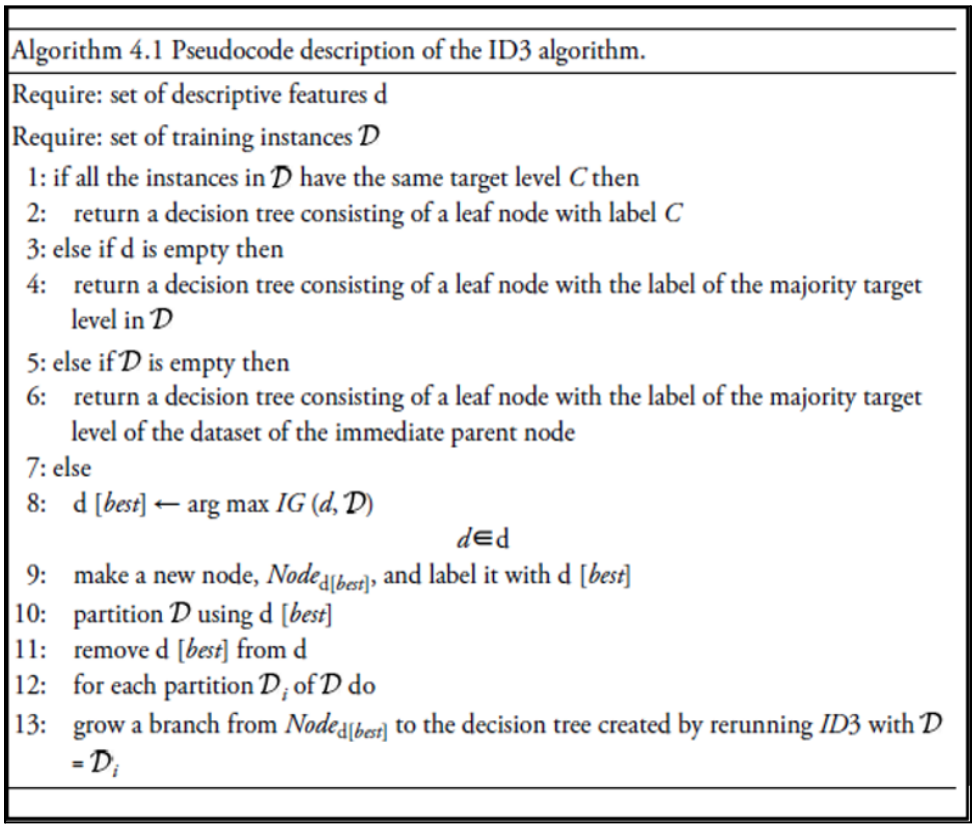
demand a high level of knowledge, and it might be challenging to interpret the data visually. They have used a fusion approach to combine features obtained from Sentinel-1 SAR and Sentinel-2 optical processed data and Random Forest supervised classifier was applied for predicting classes. For the four sub-part of study area, they then analyzed the results from the Random Forest model. The urban and bare soil areas, as well as the forest and agricultural areas, could not be sufficiently distinguished after the initial visual examinations of the data. These four classes were consequently combined to form two new classes. Three alternative scenarios were then used to conduct in-depth analyses. All three categories made use of the identical training data and their findings demonstrate that combining the datasets produces the most accurate predictions with Random Forest model applied. Finally, they have proposed that their method is an efficient way to map the flooded areas and they plan to include field investigation and ground data sampling for their future work.

Random Forest Classifier Working Methodology.

Literature survey and background research gave insights and an understanding about how the Random Forest model is better suited for classification problems. Random Forest model is explained in detail in this section where it is an ensemble technique which builds multiple decision trees and combines them in the training phase. The decision tree splits the data according to the best feature using the ID3 algorithm as shown in Figure 78; this process is continued to the point where there are no features to split. The tree obtained using the ID3 algorithm will have a root node, leaf node and sub-trees.

Figure 78

ID3 Algorithm



Algorithm 4.1 Pseudocode description of the ID3 algorithm.

Require: set of descriptive features d

Require: set of training instances $\mathcal{D}$

1: if all the instances in $\mathcal{D}$ have the same target level C then
2: return a decision tree consisting of a leaf node with label C
3: else if d is empty then
4: return a decision tree consisting of a leaf node with the label of the majority target level in $\mathcal{D}$
5: else if $\mathcal{D}$ is empty then
6: return a decision tree consisting of a leaf node with the label of the majority target level of the dataset of the immediate parent node
7: else
8: d [*best*] $\leftarrow \arg\max_{d \in \mathrm{d}} IG(d, \mathcal{D})$
9: make a new node, $Node_{\mathrm{d}[best]}$, and label it with d [*best*]
10: partition $\mathcal{D}$ using d [*best*]
11: remove d [*best*] from d
12: for each partition $\mathcal{D}_i$ of $\mathcal{D}$ do
13: grow a branch from $Node_{\mathrm{d}[best]}$ to the decision tree created by rerunning *ID3* with $\mathcal{D} = \mathcal{D}_i$

Note. Pseudocode description of ID3 algorithm reprinted from "Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies", 2015. P. 134-135, The MIT Press. Copyright 2015 by MIT

The ID3 algorithm works in a way where it requires a set of descriptive features present in the dataset under consideration. It constructs the tree recursively, beginning at the root node and approaches down to the leaf nodes. The data will then be split into training and testing. There are three basic checks involved in the algorithm before moving on to the next iteration. The first one is to check if all the instances in the given dataset have the identical target level, and in that case a leaf node with target class is built as a decision tree. If the first check fails, then

it moves on to the next one where the dataset feature is checked if it's empty. If it's empty then the output decision tree is returned with a leaf node containing the majority class. The third check is performed when the previous two checks have been failed. The check would be to look for if the training set is empty or not. If it's empty, a leaf node with target class that is majority at the parent node is obtained. As mentioned above, when all 3 mentioned checks fail the next reiteration will begin. Information Gain is the metric used to calculate the importance of the descriptive features in the dataset. Based on the Information gain the descriptive feature with higher value is chosen to be the parent node of the decision tree and then the data is further split on it. Once the feature is used to split the data it would be removed from the dataset and is not considered on the same path again for further data split. All these steps are repeated until it passes the required checks and the final decision tree is built.

On decision trees, several techniques, like bagging and boosting, can be used to enhance the model's performance. The bagging ensemble technique prevents overfitting and minimizes bias in the multiple decision trees by sampling the training data for the same model numerous times. Using the sklearn packages, one may establish the Random Forest model as the basis classifier and then a hyperparameter to control bagging. The hyperparameters that the base classifier can have has been discussed in more detail in the next section.

Model Optimization.

When compared to a decision tree, Random Forest is more robust and optimized. This model needs to be tuned to handle the data, to prevent problems with underfitting or overfitting the model. The sklearn random classifier model's parameter can be changed to achieve this. The outcome of this optimization will improve the Random Forest model's prediction accuracy. The

model tends to over fit, and it is because of the working mechanism of the algorithm. The "n estimator" parameter, that lets the user choose the count of decision trees a model can have, was created to address the problem of overfitting and, to ensure that the leaf node does not grow any further, the maximum depth can be set. An estimator may not be useful in some circumstances, in which case the overfitting can be reduced by experimenting with a range of "max depth" parameter values. Given that Random Forest is made up of several decision trees, it suggests that numerous tasks can be carried out concurrently. The "n jobs" parameter can be optimized to increase the number of jobs that can execute concurrently. This frequently aids in improving the algorithm's performance. Several hyperparameters, which are preset before the learning process, can be used to adjust the Random Forest model. Random Forests can be hyperparameter tuned to improve performance, reduce loss, and generate better output. Hyper parameter tuning along with cross validation techniques applied on the data helps provide better results by the model built comparatively.

4.1.2 Support Vector Machine

Literature Review.

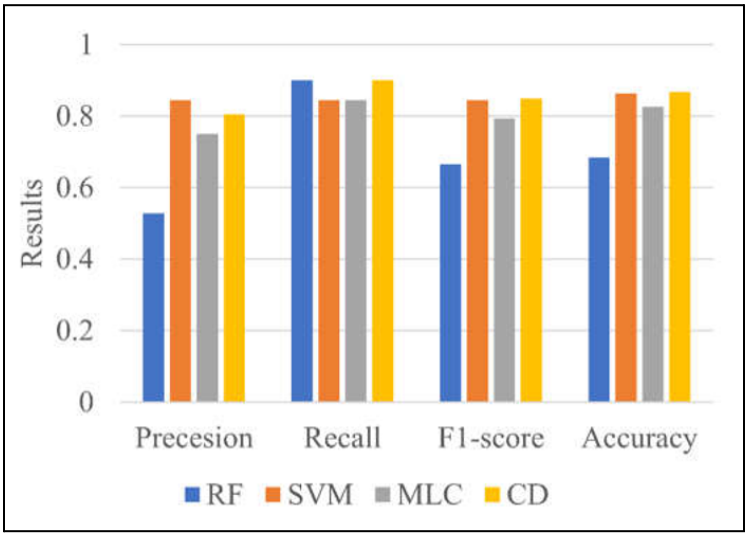
Tanim et al. (2022) studied a number of techniques used to detect as well as predict the floods occurring in urban areas. Their main objective was to reduce the damage caused by floods. They used the Synthetic Aperture Radar (SAR) images that were fetched from the Sentinel 1 Satellite as well as the road closure reports to get the ground data to train various machine learning algorithms. They implemented three supervised machine learning algorithms namely Maximum Likelihood Classifier (MLC), Support Vector Machine (SVM) and Random Forest (RF) models and one unsupervised algorithm for detecting floods in San Diego,

California. They compared and analyzed the performance for the supervised machine learning algorithms whereas they used change detection (CD) approach along with Otsu algorithm, fuzzy rules and iso-clustering combined with unsupervised model. They faced challenges in establishing a threshold level from the Satellite images for distinguishing between water and non-water pixels supervised as well as unsupervised algorithms. Figure 79 shows a performance comparison between all the mentioned machine learning models for classifying them into non-water and water pixels. The precision (0.85) and accuracy (0.85) values for SVM were the highest. The F1-score for SVM and CD was the same and higher than the RF and MLC models. It was observed that SVM had the highest accuracy when compared to other models for classification problems. One of the reasons for such high accuracy in SVM was because they tuned the parameter λ used in radial basis function (RBF) kernel that controls the kernel spread using GridSearchCV in order to increase the accuracy for water pixels and non-water pixels respectively (Tanim et al, 2022, p. 9).

$$z(x, y) = exp(-\lambda|(|x - y|)|^2) \quad (1)$$

Figure 79

Performance metrics for RF, SVM, MLC and CD algorithms



Note. Reprinted from "Flood Detection in Urban Areas Using Satellite Imagery and Machine Learning", *Water. 14,* 1140 (https://doi.org/10.3390/w14071140).

Razali et al. (2020) studied and implemented Support vector Machine (SVM), Bayesian network (BN), k-Nearest Neighbors (kNN) and Decision Tree (DT) for predicting floods using historical rainfall data. The data collected for their analysis was highly imbalanced. Hence, they performed a data augmentation technique called Synthetic Minority Over Sampling (SMOTE) where synthetic data was created to balance the data and avoid any bias during training. Table 21 shows the comparison between these models using Accuracy, Precision, Recall and F-measure as the performance metrics. All the models showed comparable performances. It was observed that after performing SMOTE, accuracy for SVM increased more as compared to any other models. Hence, it can be stated that SVM models work better for classification problems especially when the data is balanced.

Table 21

Performance indicator for models

Techniques	Accuracy		Precision		Recall		F-Measure	
	Normal	Smote	Normal	Smote	Normal	Smote	Normal	Smote
Bayesian Network	99.94%	99.68%	0.999	0.997	0.999	0.997	0.999	0.997
Decision Tree	99.89%	99.92%	0.999	0.999	0.999	0.999	0.999	0.999
k-Nearest Neighbours	99.50%	99.73%	0.995	0.997	0.995	0.997	0.995	0.997
Support Vector Machine	99.50%	99.76%	0.995	0.998	0.995	0.998	0.995	0.998

Note. Reprinted from "Machine learning approach for flood risk prediction", by Razali, N., Ismail, S., & Mustapha, A., 2020, *IAES International Journal of Artificial Intelligence (IJ-AI)*, *9*(1), 73 (https://doi.org/10.11591/ijai.v9.i1.pp73-80)

Another study by Sahoo et al. (2021) compared performance between various Artificial Neural Network models like Feed Forward Back Propagation Neural Network (FFBPNN), Radial Basis Function Neural Network (RBFNN) along with the standard SVM model and firefly algorithm in combination with SVM model. They collected data about the river flow for Dholai and Silchar stations for predicting floods. The Firefly algorithm was used to identify the best parameters of SVM when combined with these models. The parameters were termed as fireflies and the attractiveness between them was proportional to the difference between their distances. These parameters (fireflies) were then ranked in the increasing order of their attractiveness. Thus, the fireflies were ranked according to their attractiveness. Those parameters at which best results were observed were considered as the optimal parameters and were further used along with the SVM model. The cycle was repeated until the best parameters were obtained. Table 22 shows the comparison between the various models based on the RMSE, MSE and R^2 values. The SVM model combined with Firefly algorithm showed the highest accuracy with $R^2 = 0.9812$ for testing the dataset extracted from Silchar station. Hence, they concluded

that the single SVM model outperformed most ANN models and when further optimized using the firefly algorithm, gave most accurate results for flood predictions.

Table 22

Comparison between performance for two watersheds based on MSE, RMSE and R^2

Station	Techniques	MSE		RMSE		R^2	
		Training	Testing	Training	Testing	Training	Testing
Silchar	FFBPNN	0.00371	0.00686	0.05076	0.03332	0.8688	0.8806
	RBFNN	0.00446	0.00698	0.04354	0.02898	0.8941	0.9081
	SVM	0.00453	0.00701	0.04365	0.02887	0.9385	0.9478
	RBF-FA	0.00468	0.00723	0.04354	0.02867	0.9619	0.9731
	SVM-FA	**0.00485**	**0.00754**	**0.04337**	**0.02856**	**0.9807**	**0.9812**
Dholai	FFBPNN	0.00378	0.00698	0.04399	0.03311	0.8693	0.8821
	RBFNN	0.00465	0.00709	0.04201	0.03091	0.8982	0.9095
	SVM	0.00477	0.00754	0.04189	0.03088	0.9397	0.9455
	RBF-FA	0.00491	0.00776	0.04144	0.03078	0.9629	0.9712
	SVM-FA	**0.00528**	**0.00792**	**0.04139**	**0.03064**	**0.9811**	**0.9818**

Note. From "Prediction of flood in barak river using hybrid machine learning approaches", by Sahoo, A., Samantaray, S., & Ghose, D., 2021, *Journal Geological Society of India*, *Vol.97*, 186–198.

Y Shi et al. (2016) proposed as well as explored the potential of implementing Support Vector Machine models for accurate prediction of future floods. They used the rainfall and the river flow data for River Don for predicting river flow discharge as this area was affected by severe flooding in the past decades. Significant correlations were observed between the river flow at time t and the previous rainfall data at time t -1. However, for higher lead times, correlation between rainfall and the river flow started decreasing. They trained the SVM models and measured the RMSE values for predictions made by the model for river flow discharge for one week (Q_t), two weeks (Q_{t+1}) and seven weeks (Q_{t+6}) ahead respectively. RMSE values for the training and validation dataset for various time intervals have been summarized in Table 23.

They observed that the RMSE was less for the calibration and validation dataset when the predictions were made for the next week. However, the RMSE values increased significantly when the model predicted the river flow discharge two weeks and seven weeks later. Hence, they concluded that SVM accurately predicted floods for shorter lead times and were used for providing short-notice flood warnings.

Table 23

RMSE values for training and validation set at different time intervals

Prediction Target	Calibration (Training)	Validation (Testing)
Q_t	0.326	0.450
Q_{t+1}	0.634	0.779
Q_{t+6}	0.817	0.928

Note. From "Flood Prediction Using Support Vector Machines (SVM)", by Y Shi, Khaled Taalab, & Tao Cheng, 2016, *Proceedings of the 24th GIS Research UK (GISRUK) Conference. GIS Research UK (GISRUK): London, UK. (2016)*.

Dai et al. (2021) developed a machine-learning algorithm to predict floods by collecting data using sensors and Internet-of-Things devices. These devices were used to collect the weather data like temperature, humidity, wind speed, etc. as well as hydrological data like height of tide and flood depth. They used Quantum - Enhanced Support Vector Machines (QSVM), Lagrangian Support Vector Machines (LSVM), Linear Regression (LR), Random Forest (RF), Back Propagation Neural Network (BPNN) on the training data. They trained a subset of the training data with a Bayesian model integrated with BPNN model termed as Ensemble-based Bayesian Model (BMC-EL). LSVM was computationally fast and was able to handle very large datasets (in millions) which simple SVM couldn't handle. QSVM on the other hand, uses

quantum technology for computation thereby increasing the performance of standard SVM models running on CPUs and GPUs. Table 24 summarizes various metrics used to analyze the model performance. All the models showed values greater than 0.95. However, it was observed that both the optimized SVM models (LSVM and QSVM) showed lowest RMSE and MSE values and comparatively higher values indicating that SVM models when optimized were used for accurate flood predictions.

Table 24

Comparison between various models based on MSE, MAE, RMSE and R^2 values

Technique	MSE (m)	MAE (m)	RMSE (m)	R^2
LR	0.1597	2258.21	0.3996	0.9972
LSVM	0.1593	2627.34	0.3991	0.9943
QSVM	0.1178	1941.94	0.3432	0.9972
BPNN	0.1151	2071.19	0.3392	0.9870
RF	0.0247	976.47	0.1571	0.9559
BMC-EL	0.0072	528.40	0.0847	0.9799

Note. From "Ensemble Learning Technology for Coastal Flood Forecasting in Internet-of-Things-Enabled Smart City", by Dai, W., Tang, Y., Zhang, Z., & Cai, Z. 2021, *International Journal of Computational Intelligence Systems*, *14*(1) (https://doi.org/10.1007/s44196-021-00023-y).

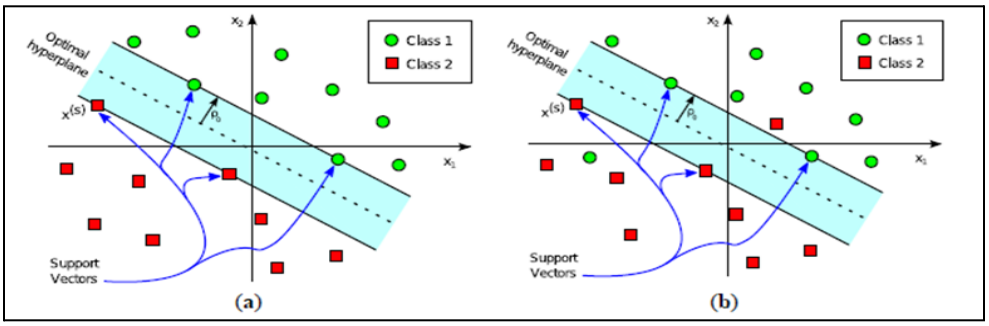
Support Vector Classifier Working Methodology.

Support Vector Machine (SVM) is supervised machine learning widely for classification and regression problems. When dealing with a binary classification problem, SVM creates a

hyperplane to separate the two classes. This is done by performing calculations for the best hyperplane boundary. Hard Margin optimality is used when the training set can be separated perfectly (Ruiz-Gonzalez et al., 2014). Hard margin selects a decision boundary such that the distance between the hyperplane and the closest training data point is maximized. However, Soft margin optimality can be used when a perfect classification is not possible. The soft margin tries to select a hyperplane such that the misclassification rate is minimized as well as the distance between the hyperplane and the first correctly classified training point is maximized. This hyperplane is determined by a subset of input training vectors, also called as support vectors. The SVM classifier can be then used along with the different subset of training data for classification. The class is usually labeled as +1 or -1 depending upon which side the input vector falls with respect to the decision boundary. Figure 80 shows the application of linear- based SVM to classify linearly and nonlinearly separable classes.

Figure 80

(a) SVM classifier representing linearly separable pattern and (b) non-linearly separable pattern



Note. Reprinted from "An SVM Based Classifier for Estimating the State of Various Rotating Components in Agro-Industrial Machinery with a Vibration Signal Acquired from a Single Point on the Machine Chassis", by Ruiz-Gonzalez, R., Gomez-Gil, J., Gomez-Gil, F., &

Martínez-Martínez, V., 2014, *Sensors*, *14*(11), 20713–20735 (https://doi.org/10.3390/s141120713).

Linear SVM binary classification problem can be mathematically represented as follows: For a given training dataset $\{x_i, d_i\}^N_{i=1}$, the objective is to find the optimum bias, slack variables and weight vector to satisfy constraints given in equation 1 (Ruiz-Gonzalez et al., 2014, p. 8).

$$d_i (w^T x_i + b) \geq 1 - \zeta_i, \forall_i = 1, 2, ..., N \quad (2)$$

$$\zeta_i \geq 0, \forall_i = 1, 2, ..., N$$

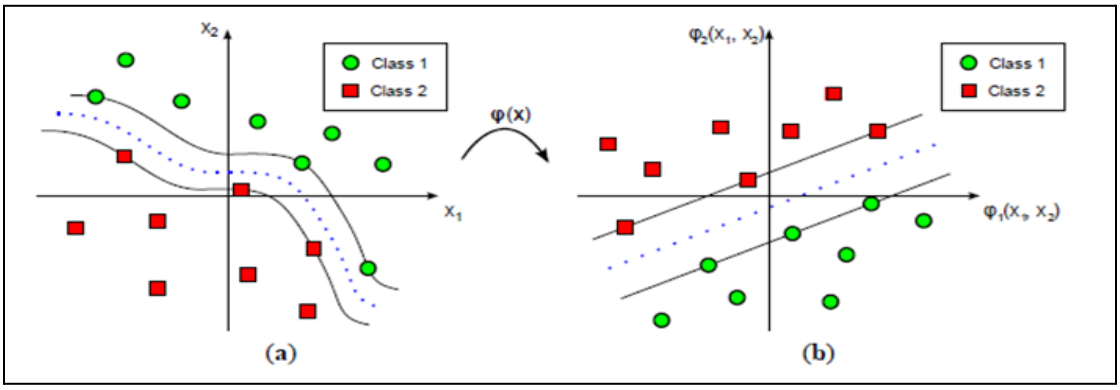
and to minimize the below cost function:

$$\phi(w, \zeta) = 1/2\, w^T w + C \sum^N_{i=1} \zeta_i \quad (3)$$

where $x_i \in R^{m0}$ represents the input vector, $\{\zeta\}^N_{i=1}$ corresponds to slack variables, $d_i \in \{-1, 1\}$ represents the class that corresponds to the input vector, and is a parameter specified by the user for finding a balance between the maximum inter class margin and the misclassification. For problems that cannot be solved using linear SVM, the non-linear transformation φ(.) can be applied to feature space with dimension $m_f > m_o$. Figure 81 shows a representation for an SVM classifier using non-linear kernels.

Figure 81

SVM Classifier using non-linear kernel mapping input vectors from input space (a) to feature space (b)



Note. Reprinted from "An SVM Based Classifier for Estimating the State of Various Rotating Components in Agro-Industrial Machinery with a Vibration Signal Acquired from a Single Point on the Machine Chassis", by Ruiz-Gonzalez, R., Gomez-Gil, J., Gomez-Gil, F., & Martínez-Martínez, V., 2014, *Sensors*, *14*(11), 20713–20735 (https://doi.org/10.3390/s141120713).

The hyperplane in the feature space *φ(x)* satisfies equation 4 *(*Ruiz-Gonzalez et al., 2014, p. 9).

$$w^T \phi(x) + b = 0 \qquad (4)$$

given $x_i \in R^{m0}$ and $\phi(x)$. Lagrange multiplier is used to find the optimal weight vector which is represented by equation 5 where α_i is the Lagrange multiplier coefficient (Ruiz-Gonzalez et al., 2014, p. 9).

$$w = \sum_{i=1}^{N} a_i \, d_i \, \phi(x_i) \tag{5}$$

The optimal hyperplane can then be obtained from equation 6. (Ruiz-Gonzalez et al., 2014, p. 9)

$$\sum_{i=1}^{N} a_i \, d_i \, \phi(x_i)^T \, \phi(x) + b = 0 \tag{6}$$

Replacing $a_i d_i$ with u_i and $\phi(x)^T \phi(x_i)$ as $K(x, x_i)$, the decision function (y) is then written as follows (Ruiz-Gonzalez et al., 2014, p. 9).

$$y = \sum_{i=1}^{N} u_i \, K(x, x_i) + b \tag{7}$$

For linear classifiers, $K(x, x_i)$ is the Euclidean inner product between input vector (x) and the support vector (x_i) whereas $K(x, x_i)$ is the Euclidean inner product between $\phi(x)$ and $\phi(x_i)$ for non-linear classifiers (Ruiz-Gonzalez et al., 2014, p. 10). SVM uses the radial basis function (RBF) kernel as a default. It is represented by equation 8.

$$K(x, y) = exp(-\gamma \, ||x - y||^2) \tag{8}$$

where γ is defined by the user. Other non-linear kernels available for SVM are polynomials and sigmoids. Sigmoid kernel is represented by equation 9 (Ruiz-Gonzalez et al., 2014, p. 10).

$$K(x, y) = tanh(\gamma x^T y + c_0) \tag{9}$$

where γ and c_0 are parameters that are defined by the user. The polynomial kernel is as shown in equation 10 where γ and c_0 defined by user and d is the degree of the polynomial Ruiz-Gonzalez et al., 2014, p. 10).

$$K(x, y) = (\gamma x^T y + c_0)^d \quad (10)$$

Model Optimization.

SVM models can be optimized by selecting appropriate parameters like the relaxation factor ζ , penalty constant C, and certain parameters from the kernel function like the spread of the kernel γ and width parameter σ in case of RBF. Optimal value for these parameters can be found using the gradient descent method. An algorithm named 'Leave one support vector out cross validation' can be used to find optimal parameters for RBF kernel (Yuan & Chu, 2007). Firefly algorithm can also be used to find the optimal parameters for SVM classifier. Selecting the optimal parameters and then training the SVM classifier with these parameters can significantly improve the model prediction performance.

4.1.3 XGBoost Classifier

In selecting the models for supervised learning technique, XGBoost model was preferred because of fast computational and efficient utilization of resources. It is one of ensemble boosting methods which combines gradient boosting technique with regularization to avoid overfitting the model by pruning the trees in the early stage. This model works well with imbalanced data which was suitable for our project as there were a few flood events i.e. the target class was imbalanced.

Literature Review.

(Nguyen. et al., 2021) implemented various techniques to detect the hourly water level in Jungrang urban basin in order to detect a flood event in advance so that precaution can be taken . In their research, an XGBoost hybrid model(a combination of genetic algorithm(GA) and differential evolution(DE) algorithm) was implemented along with CART and random forest model. The XGBoost model leverages strength from decision trees and gradient boosting models and the XGBoost hybrid model was achieved by integrating it with two algorithms i.e. with genetic algorithms and different evolutionary algorithms. These algorithms are used to find the best parameters during hyper parameter tuning. The genetic algorithm is based on the fundamental concept of evolution and natural selection whereas differential evolution was based on global optimization of non-differentiable function. Rainfall data was collected and preprocessed from 2003 to 2020 for the month of June to September and then data was preprocessed. All the machine learning techniques were implemented to evaluate the performance and XGBoost model outperformed random forest model and the CART in terms of relative error which ranged from (2%-9%) in XGBoost model to (3%-10%) in random forest model. Also, time complexity was less in the XGBoost hybrid model as compared to the random forest model. Also the performance of the DE- XGBoost model was better than GE- XGBoost model. Hence, XGBoost model was selected in hourly water level prediction.

(Nti. et al., 2021) implemented multiple machine learning algorithms to detect the flood events in Ghana where flood has affected millions of people, damaged property and flora and fauna. In their research they implemented random forest model (RF), extreme gradient boosting model(XGB) , extremely randomized trees and long term short memory(LSTM). XGBoost model was used as it works well for complex hydrology data and it is easy to scale due to its

learners. The data contained rainfall level details for Ghana and the data was preprocessed using min max scaler. Then these four machine learning algorithms were implemented and their performance was evaluated by finding mean squared error (MSE), root mean squared (RMSE), mean absolute percentage error(MAPE) and mean absolute error(MAE). Out of all the models, XGBoost model had the best results i.e. least error rate. The values are MAE (0.0028), MSE (0.0001), RMSE (0.0089) and MAPE (0.024). Hence, in the end they implemented a hybrid model by integrating XGBoost model with another high performing model to decrease the error rate.

(Kraisangka. et al., 2022) aimed to mitigate flood and drought events in Thailand by implementing a machine learning method to find the reliable hydro-parameter information for better decision making. Hence, in their research they implemented decision trees, random forest, support vector regressor and XGBoost model to analyze which model is more suitable. Time series data was collected from 2000 to 2021 and the data consisted of nine features which contained details above the average inflow in the past n days. Then a model was implemented and a 10 fold cross validation technique was used for all models to compare the performance of models. The XGBoost model was optimized using the ridge regularization parameter. The model with lowest error(mean squared error and mean absolute error) and highest accuracy(r squared) was selected. Out of all the models, random forest regressor and XGBoost regressor had the least error i.e. mean absolute error was 3.3 and 3.4 for random forest and XGBoost model. Also the accuracy of both the models was around 91%.

(Ma. et al., 2021) stated that flash floods are the most dangerous natural disaster. Flash flood means stream floods triggered by heavy rainfall on hilly areas. Therefore, in their research they aimed to minimize the flash flood risk by leveraging machine learning techniques. Data was

collected for the low plateau region in South Western China. They implemented the XGBoost model and support vector machine to find the flood risk index. XGBoost model was used as it has remarkable results in many fields and the algorithm has less time complexity as compared to other ensemble methods. Also, they automatically utilize the CPU'S multi threading to perform parallel computation. Hence, improving the prediction accuracy and training speed. The results were calculated which showed that the XGBoost model outperformed the support vector machine. The XGBoost model had accuracy of 0.84 in the testing period and the five indices (precision, recall. F1 score, kappa and accuracy) were higher than support vector machines. XGBoost method provided more reliable results for flash flood risk index which were validated by another flash flood inventory. Hence, XGBoost model was selected as the best predictive model.

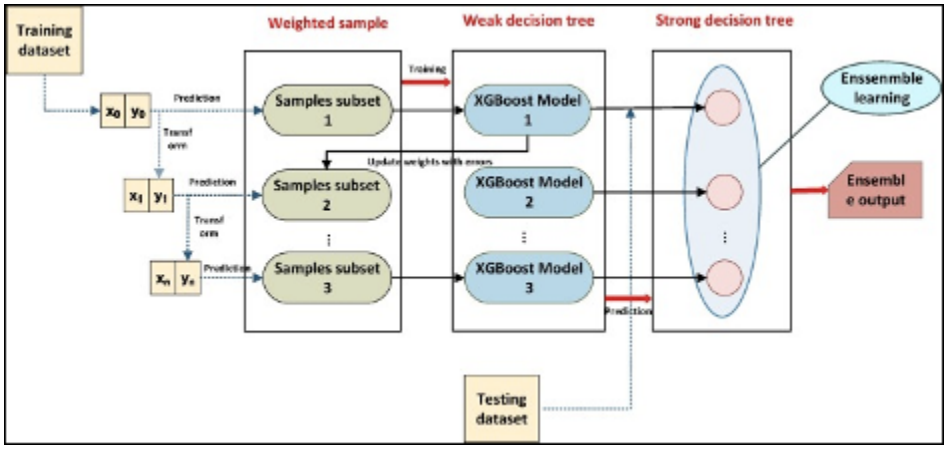
(Ibrahem et al., 2021) aimed to study ground level prediction in Malaysia using machine learning algorithms. They collected data consisting of rainfall level, temperature and evaporation to predict groundwater levels. Three machine learning methods were used; these are the XGBoost model, ANN(artificial neural network) and support vector regression. The XGBoost model was preferred by scientists due to its high computational speed. The performance of the model was observed for four days and a cross correlation method was used to select the best model. The results showed that the XGBoost model performed best as compared to support vector regression and artificial neural network for different input values. R squared value for the XGBoost model was 0.92 which was high as compared to other models. Also, the XGBoost model had the least mean absolute error (MAE) which is 11%, less than that of ANN and support vector machines.

XGBoost Classifier Working Methodology.

The XGBoost model used in previous research papers gave insights about its algorithm and working methodology. The model is able to generalize well by combining results from various learners. The algorithm uses CART(classification and regression tree) as base estimators. Input to the tree depends on the prediction made by the previous tree. The XGBoost model is a highly versatile method which is famous for its efficient resource management. Figure 82 represents the XGBoost model algorithm.

Figure 82

XGBoost Model algorithm



Note. XGBoost algorithm adapted from "XGBoost-based method for flash flood risk assessment", by Ma, M., Zhao, G., He, B., Li, Q., Dong, H., Wang, S., & Wang, Z. (2021), *Journal of Hydrology*, *598*, 126382, p.7 (https://doi.org/10.1016/j.jhydrol.2021.126382)

From figure 82, it can be seen that the model is composed of CART in which the residual errors are fed as a label to the next estimator. The weights for wrongly predicted values are updated so that model gives more preference to those values. At each stage, residuals are calculated which depend on the learning rate which is known as 'eta'. The value for learning rate

has to be chosen wisely. If the learning rate is too high then the model might miss a minimal loss function values and if the values are too low then it takes more time in fitting the model. Thus, increasing the time complexity. Hence, after performing classification on each learner, output values are combined to predict the final class of the target values. The model can be hyper-tuned on various parameters such as lambda(regularization parameter), gamma(loss function), eta(learning rate) , number of estimators and max depth. Hyper-parameter tuning can be implemented by using sklearn GridSearchCv and the XGBoost model is imported using XGBoost library which is a separate library in python packages. Also, XGBoost model is based on ensemble boosting method which is suggested when the model underfits i.e. when there is more bias.

Model Optimization.

For the XGBoost model, optimisation can be done by varying the max depth level, gamma which is used to define the minimum loss function to make a new split in the tree, learning rate which tells how quickly the model will fit the errors and lambda which is a regularization parameter. Also, L1(Lasso) and L2(Ridge) methods can be selected as per the nature of the data i.e. if the data consist of high dimensional spaces then L1 method can be used to reduce the loss function. Thus, preventing overfitting. All possible combinations of these values can be passed during the hyper-parameter tuning to select the best parameter. For classifiers, the XGBoost model predicts probabilities for each instance and then a similarity score is calculated which can be reduced by increasing the value for lambda. In each iteration when the residuals are fed to the new learner the error rate decreases. Also, they are famous for parallel computation thereby utilizing all cores, processors and increasing the computational speed. This can be achieved by setting the n_jobs parameter to -1 so that all processing can be done

simultaneously by all cores in the CPU. Due to its parallel computation nature and sequential learning by implementing gradient boosting technique, XGBoost model is widely used for large complex datasets.

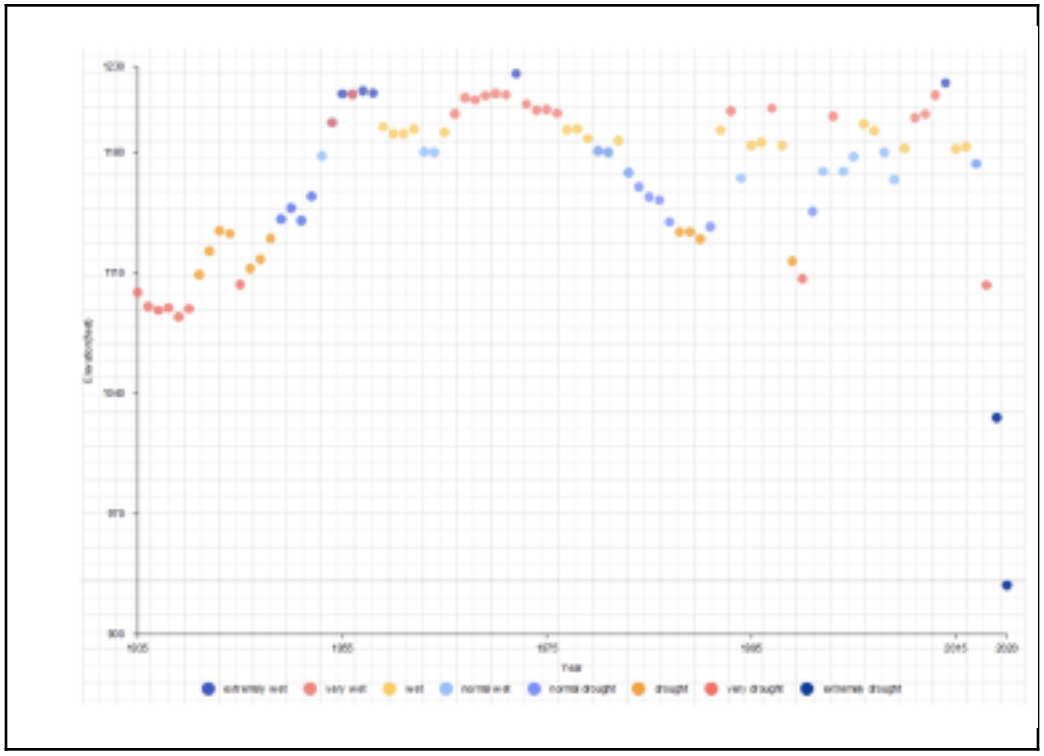
4.1.4 KMeans Clustering

Literature Review.

K-Means Clustering is a popular clustering algorithm used to find groups or classes of an unlabelled dataset. Chujie Shen et al. (2022) used k-means clustering to classify the output label i.e. scale of drought into 8 classes. This research was based on Lake Mead. Factors leading to the low elevation of the lake were identified, and fitting functions were used to understand the data. The elevation change can be determined with volume change. Although this model was simple to implement, it was not the ideal one to choose as other parameters like wind speed and others were assumed. Below is a scatter plot illustrating all data points classified into eight different categories.

Figure 83

Scatterplot representing clusters of various drought types



Note. Reprinted from "Prediction Model of Lake Water Volume Based on K-Means" by Chujie Shen, Hongwei Li, Mengqi Chen, Kairui Zhu, and Fan Wang (2022). *In The 2022 5th International Conference on Electronics, Communications and Control Engineering (ICECC 2022). Association for Computing Machinery, New York, NY, USA, 57–62.* (https://arxiv.org/pdf/1406.4756.pdf)

Another case where k-means clustering can be used for classification is for weather prediction. Sanjay et al. (2012) worked with an incremental k-means clustering model to forecast the weather in West Bengal, India. Information regarding air pollution was used to predict and classify accordingly. The dataset consists of features like Carbon Dioxide,

inhalable particulate matter and other components. The data is added frequently and hence, it is incremental k-means clustering. Four groups are obtained after the process of clustering. The first group represents days with hazy weather, dust, and fly ash as a result of high RPM levels. The second cluster indicates days that are hot, dry, and cloudy due to Nitrogen Oxides. The third cluster represents hot, smoggy, and humid days caused by carbon dioxide. The fourth cluster represents hot, hazy days with chances of acid rain due to Sulphur Dioxide. The obtained table with the output label i.e., weather category is shown as Table 25.

Coming to implementing k-means clustering for detecting flood, Hongshi Xu et al. (2018) proposed a method for calculating the flood risk in cities based on an inundation model, improvised entropy weight method (where entropy is a measure of randomness) and k-means clustering. The approach was applied to risk zones on the Haidian island using GIS technology. By combining the natural hazard index system and hydrological models, seven risk indicators were picked to create the index system. When weights were calculated using the AHP and entropy weight methods, the results displayed a deviation between entropy weights and AHP weights. As a result, this study provided an improvised entropy weight technique for merging AHP and the entropy weight methods. A method combining the k-means clustering approach
and the improvised entropy weight method was used to create the risk map in the region of research.

Table 25

Weather prediction obtained using incremental k-means

Date	*New data inserted into*	*Weather Category*
1/9/2009	Cluster2	Hot, dry and smogy
2/9/2009	Cluster3	dusty, fly ash, smogy, fog, Mist
3/9/2009	Cluster2	Hot, dry and smogy
4/9/2009	Cluster2	Hot, dry and smogy
.....	.....	.....
28/9/2009	Cluster3	dusty, fly ash, smogy, fog, Mist
29/9/2009	Cluster3	dusty, fly ash, smogy, fog, Mist
30/9/2009	Cluster4	hot, smogy and humid
.....	.....	.....

Note: Reprinted from "Weather Forecasting using Incremental K-means Clustering" by Chakraborty, S., Nagwani, N.K., & Dey, L. (2014). *2014 IEEE Canada International Humanitarian Technology Conference - (IHTC)* (10.1109/IHTC.2014.7147515)

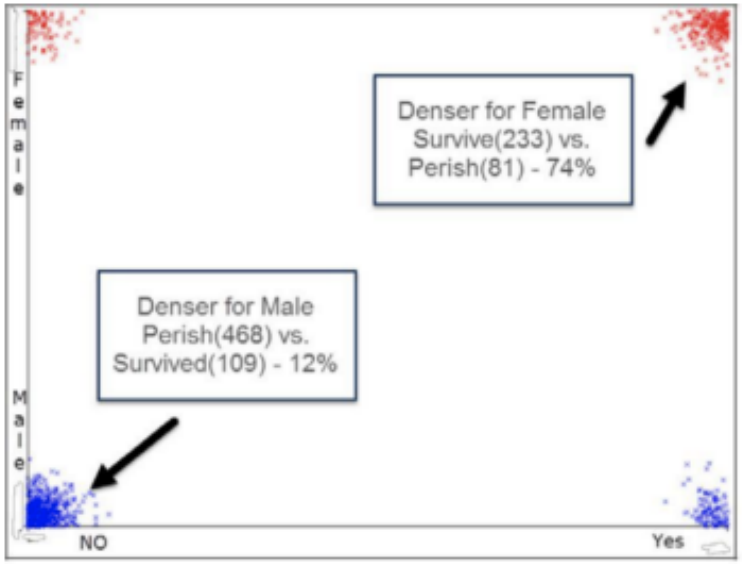
Razafipahatelo D. et al. (2014) proposed an automatic three-step method for flood detection based on clustering. First, a Digital Elevation Model (DEM) is utilized as the basic data to recognize areas prone to flooding. The moist and dry pixels are then separated using a technique known as kernel k-means. Finally, a non-linear clustering approach with a log ratio image is used to separate the flooded pixels from the permanent water. The model's output was compared to that of the manual and colour composite methods. The analysis revealed that the model is effective for flood detection.

To understand models better, working with the most-known or common datasets like the Iris dataset, MNIST dataset and others help. Sherlock J. et al. (2018) worked with the famous Titanic dataset to classify the data based on the passengers and predict their

survival accordingly. Though k- means clustering was implemented due to its simplicity, it helped in understanding certain associations only as shown in Figures 84 and 85. The actual cause-and-effect relationships cannot be strongly concluded.

Figure 84

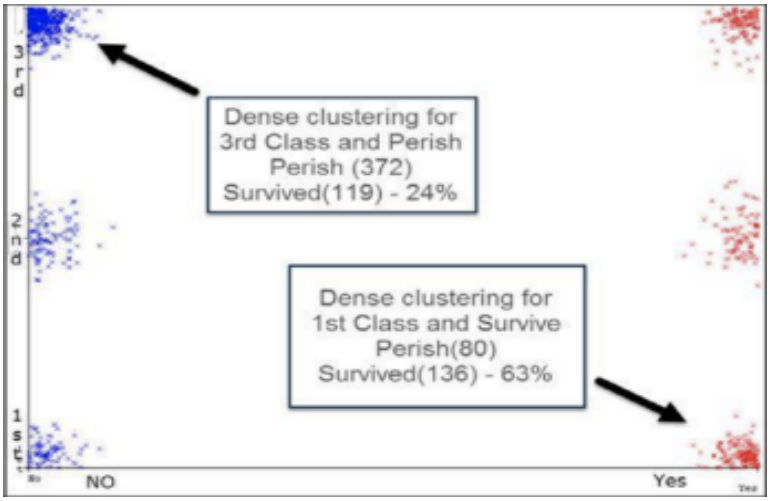
Simple K-Means for Survived vs. Gender



Note. Reprinted from "Classification of Titanic Passenger Data and Chances of Surviving the Disaster" by Sherlock, J., Muniswamaiah, M., Clarke, L., & Cicoria, S. (2018). (https://doi.org/10.48550/arXiv.1810.0985)

Figure 85

Simple K-Means for Survived vs. Passenger Class



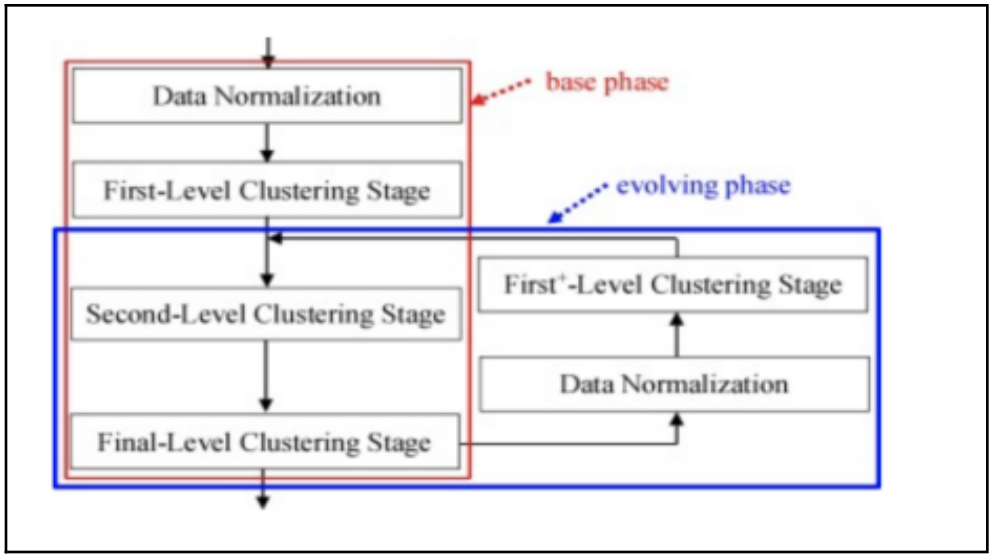
Note. Reprinted from "Classification of Titanic Passenger Data and Chances of Surviving the Disaster" by Sherlock, J., Muniswamaiah, M., Clarke, L., & Cicoria, S. (2018). (https://doi.org/10.48550/arXiv.1810.09851)

KMeans Clustering Working Methodology.

K-Means Clustering is a popular unsupervised machine learning algorithm that works on classifying N records in a dataset into k clusters in which each record belongs to the cluster with the nearest mean. For the number of clusters i.e k, the rule of thumb can be used i.e. k is equal to sqrt(n/2) where n is the number of records. In k-means clustering, centroids are initialized, then the records are classified, and the centroids are computed again until convergence. Figure 86 is similar to the description mentioned before.

Figure 86

Framework illustrating a k-means algorithm



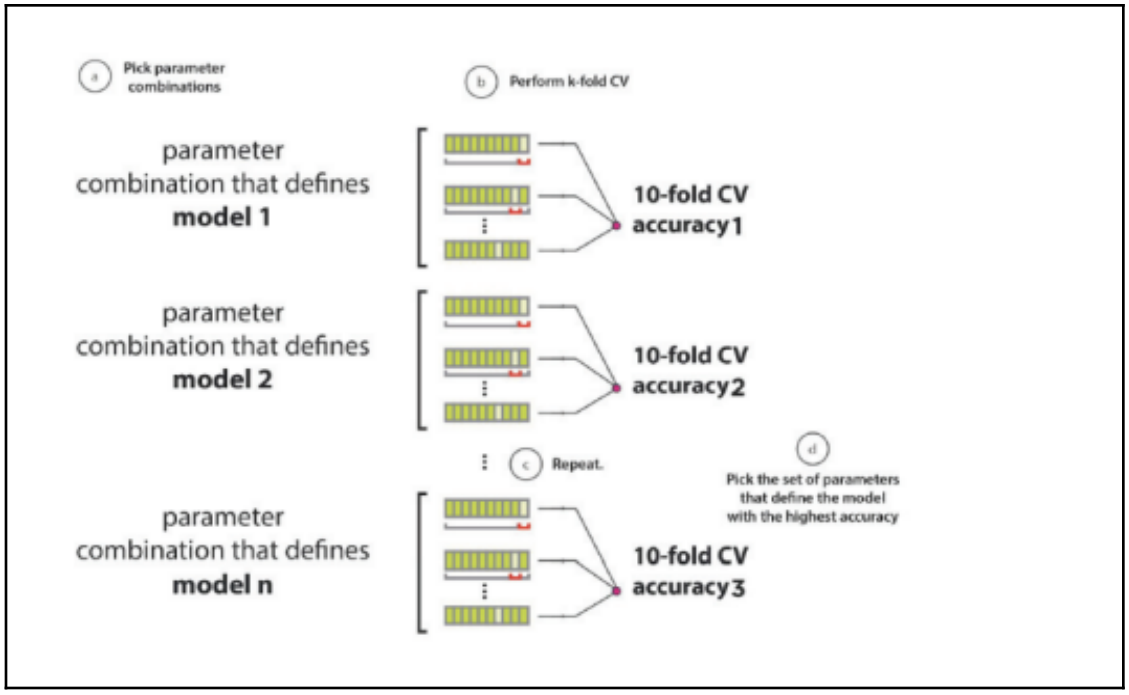
Note. Reprinted from "Two improved k-means algorithms" by Chang T., Chu S., Yu S. and Chan Y. (2018). (https://doi.org/10.1016/j.asoc.2017.08.032)

Model Optimization.

Once the final dataset is obtained, data is split into training and testing datasets, and then perform hyperparameter tuning using GridSearchCV to determine the optimal values. The performance of the model is dependent on the hyperparameters. GridSearchCV goes through all combinations passed and the evaluation is performed using the cross-validation method like k-fold, stratified, etc. GridSearchCV function is available in the scikit-learn library. The below figure would help in understanding the GridSearchCV process.

Figure 87

Working of GridSearch Cross Validation



Note. Reprinted from "K-Nearest Neighbors and Grid Search CV Based Real Time Fault Monitoring System for Industries" by Ranjan G., Verma A. and Radhika S. (2019), *IEEE 5th International Conference for Convergence in Technology (I2CT), 2019.*

(https://ieeexplore.ieee.org/document/9033691)

4.2 Model Supports

4.2.1 Environment Platform and Tools

As the data is a time series data so data was split into training and testing by manually taking the first 80% percent of the data as trained data and the rest of the data as test data.

Following are the requirements needed to efficiently implement supervised and unsupervised models. Hardware configuration for the system on which the model was run is Intel 12th generation core i7 processor model with 12700k model number, system RAM consisted of 16 gigabytes with 5 gigahertz processor speed.

Software configuration used in this project is an operating system having Windows 11 professional and Microsoft office was used for reports and creating presentations. Lucid charts were used to create flow charts and diagrams. Microsoft Excel was used to create tables for data collection methods. Also, google drive was used for documentation and run models using google colab.

For tools various packages and libraries were imported using python 3.9 version. Data was loaded, explored, cleaned and machine learning algorithms were implemented on a jupyter notebook which is an environment provided by anaconda software. Pillow library was used to load, crop and extract pixels from INSAT images. Below table 26 summarizes the different python libraries used.

Table 26

Tools and python libraries used

Library		Method	Usage
Pandas	Dataframe	Drop,shape,head, apply, read_json,rename, Fillna, describe, merge	To upload data into dataframes, perform data wrangling, data transformation and integration. Also used to checking the statistical values of the data
Pandas	pandas.read_json	To load the json file into data frame	Used to convert json format into rows and columns

Library		Method	Usage
numpy	np.to_array, np.where	Seed, array	Used to put threshold condition in defining the target variable
Seaborn matplotlib	pyplot	Scatter plot, distplot , countplot, barplot, boxplot	Used to analyze distribution of the data, relation between target and independent variable
Sklearn	sklearn.model_selection	Cross_validate, cross_val_score, time_series_split, GridSearchCV	Used in cross validation method, for splitting the time series data and for hyper-parameter tuning
Sklearn	sklearn.metrics	Confusion_marix, preciion_score, recall_score, f1_score, roc_auc_score, make_scorer, roc_curve	To evaluate the performance metrics
Sklearn	sklearn.preprocessing	minmaxscaler	To normalize the data in the range zero to one
Sklearn	sklearn.preprocessing	onehotencoder	To one hot encode the categorical feature
Sklearn	sklearn.ensemble	RandomForestClassifier	To implement Random Forest Classifier
Sklearn	sklearn.svm	SVC	To implement Support Vector Classifier
Sklearn	sklearn.cluster	KMeans	To implement KMeans clustering (Unsupervised)
xgboost	xgboost.XGBClassifier	Model implementation	To implement extreme gradient boosting model
Pickle	pickle.dump, pickle.load	Saving and reloading the model	Used to saved the trained model
imblearn	imblearn.over_sampling, imblearn.pipeline	SMOTE, imblearn pipeline	Oversampling technique, Creating pipeline for ML

Library		Method	Usage
Pillow, python image library (PIL)	pil.image	To extract the pixels of images	Used to record and crop the pixel in the INSAT images

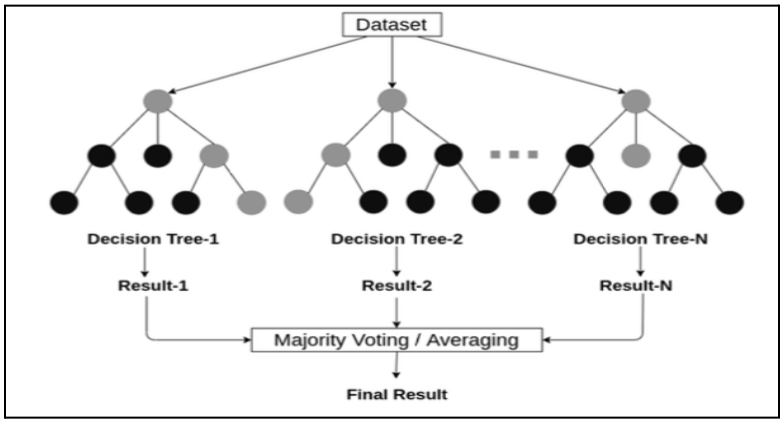
4.2.2 Model Architecture

4.2.2.1 Random Forest Classifier.

The ensemble model that uses the bagging technique, or bootstrap sampling, is the random forest model. The fundamental concept is to merge many decision trees built using the ID3 algorithm to identify the target class rather than relying just on one tree as shown in figure 88. Decision trees serve as the base estimators in the random forest model. The dataset is repeatedly sampled at random, and the sample's size is set at the same level as the initial dataset size. By using feature sampling, sometimes referred to as subspace sampling, it further raises the randomness. Based on the majority votes cast for the class, the target variable class is anticipated.

Figure 88

Random Forest Structure



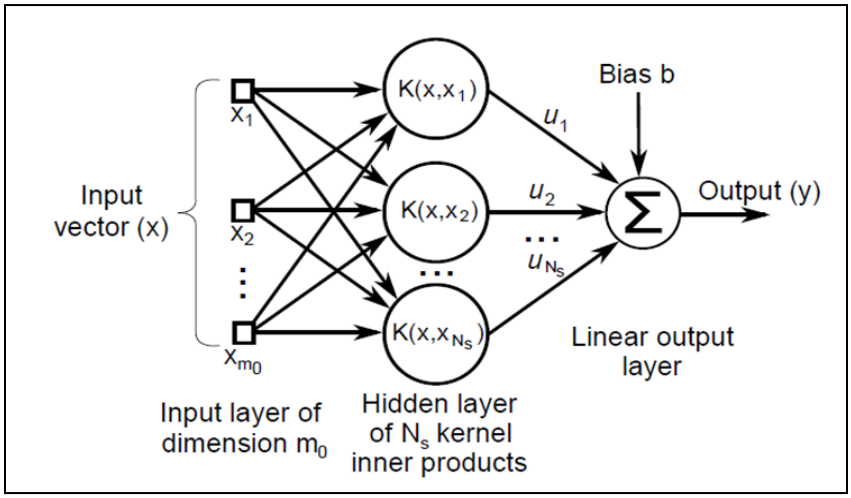
Note. An example of a random forest structure considering multiple decision trees reposted from Machine Learning: Algorithms, Real-World Applications and Research Directions, Sarker, I. H. 2021, *SN Computer Science,2(3), 2:160,* p. 7 (https://doi.org/10.1007/s42979-021-00592-x). Copyright The Author(s), under exclusive licence to Springer Name Singapore Pte Ltd 2021

4.2.2.2 Support Vector Classifier.

Figure 89 shows the architecture for SVM Classifier created from the decision function when the proper support vectors and the weights are computed as shown in equation (7). Initially, the input vector is passed into the SVM classifier where the inner product between each input vector with the support vector (x) is computed $(K(x_i, x))$ (Kelleher et al., 2020). The proper weights are then computed for each combination of the inner product and are summed up after which a bias is introduced. The classification is performed and the output y is obtained which carries a positive or a negative sign indicating whether the input is classified as +1 or -1.

Figure 89

Architecture for SVM Classifier



Note. Reprinted from "An SVM Based Classifier for Estimating the State of Various Rotating Components in Agro-Industrial Machinery with a Vibration Signal Acquired from a Single Point on the Machine Chassis", by Ruiz-Gonzalez, R., Gomez-Gil, J., Gomez-Gil, F., & Martínez-Martínez, V., 2014, *Sensors*, *14*(11), 20713–20735 (https://doi.org/10.3390/s141120713).

4.2.2.3 XGBoost Classifier.

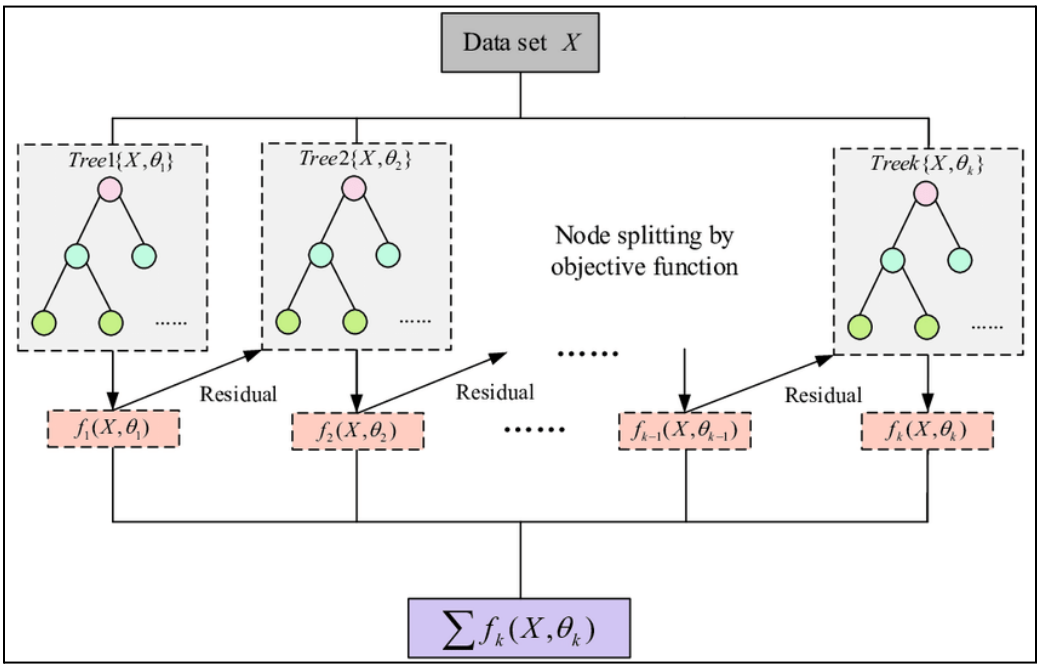
XGBoost model is one of the supervised machine learning algorithms. It is based on boosting ensemble techniques which uses regularization techniques along with gradient boosting. In boosting multiple trees known as learners are trained sequentially. These base estimators tried to correct the wrongly predicted value produced by its predecessors. However, the XGBoost model uses a sequential learning method from boosting techniques but implements parallel computation for system optimization. Due to this, the XGBoost model works efficiently for large and complicated datasets.

In the XGBoost architecture, the algorithm finds the gain based on the similarity score. The residuals are calculated by determining the probabilities which involve initially calculating the log of odds for the whole dataset and then finding the probabilities. Next step includes calculating the residuals i.e. difference between mean and predicted values. Then these residuals are fed as labels to the learners which then calculates the similarity score for the root, left and right branch. Similarity score is defined as ratio of sum of squared residual divided by sum of number of residuals and regularization parameter(which is commonly known as lambda). Hence, if the value of the lambda is more then the gain will reduce. It also uses gamma which is a loss reduction function to create a new split. If the value of gain minus gamma is negative then the branch is pruned, thus avoiding overfitting. Learning rate which is known as 'eta' in xgboost

model is defined in the range zero to one which tells how quickly the model fits the residual by using additional learners. The XGBoost model is good when there is class imbalance in the dataset because when wrong values are predicted more preference is given to the errors. Below figure 90 shows the architecture for the XGBoost model.

Figure 90

XGBoost Classifier architecture



Note. XGboost architecture adapted from "Degradation State Recognition of Piston Pump Based on ICEEMDAN and XGBoost", by Guo, R., Zhao, Z., Wang, T., Liu, G., Zhao, J., & Gao, D. (2020), *Applied Sciences*, *10*(18), 6593, p.6 (https://doi.org/10.3390/app10186593)

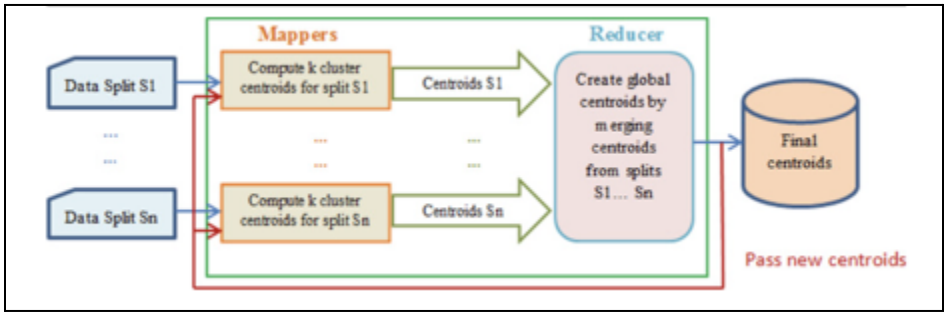
4.2.2.4 K-Means Clustering.

Figure 91 illustrates two k-mean clustering happening simultaneously. If the top part or the bottom part of the diagram is observed, the actual architecture of k-means clustering can be understood. In this clustering, the first step is initializing the centroids. Then, once the clusters

and the respective data points are grouped, new means are computed. The centroids are changed and loop through the entire process until convergence.

Figure 91

Parallel or multiple k means clustering



Note: Reprinted from "An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm" by Sardar T.H., Ansari Z.Future Computing and Informatics Journal (2018) Volume 3., Pages 200-209, ISSN 2314-7288. https://doi.org/10.1016/j.fcij.2018.03.003.

4.2.3 Data Flow

In the dataflow stage as shown in Figure 92, the data was collected by implementing a python script using API library to fetch the latest data and images for 15 minutes. The data consisted of around 1270 records. The fetched data (which was collected from three sources was stored on a local machine) was initially collected in json format and this data was uploaded and converted into pandas dataframe (structured rows and columns) using pandas library. Few columns in tidal and weather data consisted of key-value pairs because of json format and relevant columns were fetched using pandas apply function. After this data wrangling steps were implemented which are checking for data types, null values and duplicate data and changing the column names for better understanding. There were a few missing values in the data and these values were replaced by using sklearn SimpleImputer i.e. replacing the null values by mean

values. As the data collected was real-time data so there were no duplicate values found. For INSAT images, python image tool was used to record the pixels for images and these pixels were recorded in the data frame along with their timestamps. In the data transformation phase, pandas merge function was used to combine three data frames using the timestamp values. After this target column was defined based on certain threshold values for rainfall, tidal height and humidity. The target column is binary variable i.e. one stands for flood cases whereas zero means that there was no flood event.

In the next phase data transformation was performed. Initially all relevant features excluding the target column were converted into an array and the target variable was also converted into a separate array. Tidal state and weather description were categorical variables which consisted of two and six category levels respectively. These descriptive features were one hot encoded using sklearn onehotencoder and drop parameter in the onehotencoder was set to 'first' to avoid a dummy variable trap (to avoid multicollinearity in the data). Then data normalization was done on descriptive features using sklearn MinMaxScaler so that all values are in the range zero to one. This was done so that the machine learning algorithm is not biased towards any variable with higher magnitude.

The data is a time series data so time series splitting was done i.e. manually assigning the first 80% of the data to the training dataset (with first 916 records) and the last 20% of the data to the test data set.

As the target class is imbalanced, an oversampling technique known as SMOTE (synthetic minority oversampling technique) was used to make the data balanced. By using SMOTE, data in the minority class is randomly generated by linear interpolation method in order to make it equal to the majority class. Hence, training data was oversampled using SMOTE.

In the modeling phase, various parameters were defined for hyperparameter tuning and time series cross validation technique was used in GridSearchCv with value set to 3. Appendix C lists the parameters for all models. For three time series cross validation split three iterations was performed. In the first iteration, the first three parts will be the trained data and the fourth part will be the test data. Likewise, in the second iteration the first four parts will be trained data and fifth part will be test data and so on. The parameter n_jobs was set to -1 so that all processors can run parallel. Optimal parameters for each model are shown in figure 93,94,95 and 96 respectively.

Figure 92

Flowchart for data flow using supervised and unsupervised models

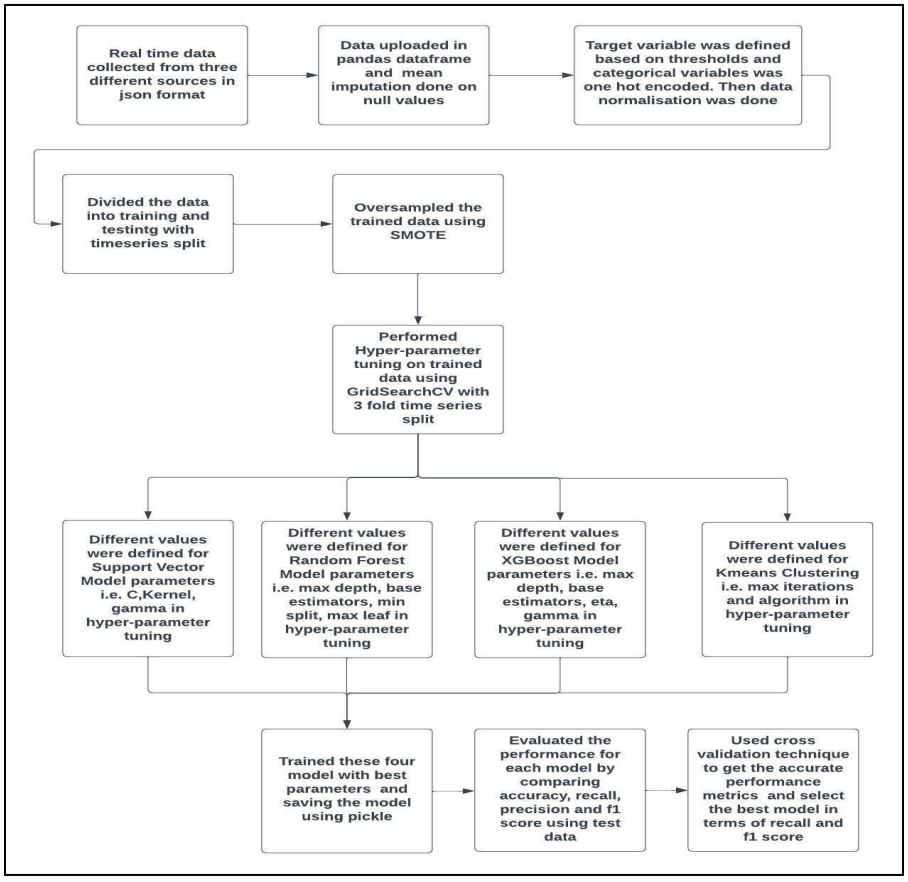


Figure 93

Best Parameters for XGBoost Model

```
{'eta': 0.01, 'gamma': 0, 'max_depth': 3, 'n_estimators': 50}
```

Figure 94

Best Parameters for Random Forest Model

```
{'criterion': 'gini', 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 50}
```

Figure 95

Best Parameters for Support Vector Classifier

```
{'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}
```

Figure 96

Best Parameters for KMeans Clustering

```
{'algorithm': 'lloyd', 'max_iter': 300}
```

After getting the best parameters from the hyper-parameter, all models were trained with the best parameters and then the performance of the model was checked using test data. This trained model was saved using python pickle library to minimize the re-training time and allows to and reload the model during evaluation phase.

The performance of the model was tested using test data. To get the accurate performance metrics cross validation technique was used and the three fold time series split was used in cross validation. This was done using sklearn metric module which includes make_scorer, precision_score, recall score,f1_score. A function was defined to pass these metrics as key- value pairs. Models were defined in cross validation by implementing a pipeline i.e. using an imblearn pipeline in which the data will be first oversampled using SMOTE and then trained with the respective model and tested.

Lastly, comparison was done between all models(support vector classifier, random forest classifiers, k-means clustering) to find which model had the best performance in terms of recall, precision, and f1 score. In a flood prediction model, recall is important because the model should be able to detect a flood case accurately. Hence, more preference was given to higher recall score. In other words, tradeoff was done between recall and other performance metrics to select the best model. Also, comparison was done between supervised and unsupervised learning to see which method is more efficient for flood prediction models.

4.3 Model Comparison and Justification

In the flood prediction model, four models were implemented. These are Support Vector Classifier, XGBoost Model, Random Forest Classifier and K-Means clustering. First three models are based on supervised learning and the last one is based on unsupervised learning. With respect to the architecture, random forest classifier and xgboost model are ensemble techniques but the difference between them is that the former trained the estimators in parallel(ensemble bagging method) while the latter trained the model sequentially (ensemble boosting method). Support vector classifiers are based on defining the hyperplane to classify the data linearly whereas k-mean clustering is fed with unlabeled data which then cluster the data based on similarity such as minimum euclidean distance. Both random forest classifier and xgboost model work efficiently on tabular data, image and audio classification, time series data. Similarly, the support vector classifier works well for image classification by using radial basis kernel trick which is based on euclidean distance and k-means clustering works well for image classification by clustering the data based on pixel values. However, k-means clustering doesn’t work efficiently on time series data and this result was observed in this project where k-means clustering had the least accuracy.

With reference to the data, xgboost model works well with large, complex and structured data sets but it doesn't work well when the data is sparse and unstructured. Also, they are affected by outliers because each learner is forced to correct the error produced by their previous learner. However, random forests are robust to outliers as they give the output based on majority votes or on aggregation. Also, random forest models are not affected by the curse of dimensionality because at each sub model a subset of features is used. Similarly, support vector classifiers work in dimensional spaces but they are affected by when there is class overlap i.e. when there is noise in the data. On the other hand, k-means clustering is affected by the curse of dimensionality as well as by outliers. With respect to overfitting, the XGBoost model uses a regularization parameter which prunes and avoids overfitting whereas a random forest might overfit if the majority of the trees are provided with the same samples but this happens rarely. Similarly, the support vector algorithm avoids overfitting because it uses kernel trick for non linear data and regularization parameter for linear data. Also, xgboost model is good for imbalance data as the learners give more weightage and preference to the errors.

Lastly, with reference to complexity of the model. XGboost model takes less time in training as compared to random forest model. Overall, the random forest model had the highest time complexity. Also, xgboost model has better system efficiency as it implements both sequential learning as well as parallel computation in its architecture. Due to it parallel computation, the model has good computational capacity as it efficiently allocates computational resources ie. CPU, GPU, memory and disk space. This has proved to be best as compared to other models, especially random forest model which is also an ensemble technique. The main advantage of the xgboost model is that it works well for large datasets and works efficiently when the target class is imbalanced. However, its limitations are that it doesn't work well for

sparse data sets and might get impacted by outliers. Table 27 summarizes the comparison between different models implemented in this project on various parameters.

Table 27

Characteristics comparison between different models

Characteristics	XGBoost Model	Random Forest Model	Support Vector Classifier	K-Means Clustering
Type	Supervised	Supervised	Supervised	Unsupervised
Architecture	Ensemble (Boosting)	Ensemble (Bagging)	Linear	Linear decision boundaries
Data type	Efficient for tabular data, image , audio and time series data	Works well on tabular data	Works well for tabular, image and audio data	Efficient for image and audio classification
Data size	Performs well on large, complex and structured datasets.	Deals efficient with outliers, missing values and higher dimensional data.	Works well in higher dimensional spaces but is affected by noise in the data	Works well for large samples but it is affected by curse of dimensionality and noise in the data
overfitting/ underfitting	Avoids overfitting by using regularization parameter	Less prone to overfit	Less likely to overfit as it uses kernel trick	Can overfit is the values of k(i.e. clusters defined) is low
Model Complexity	Takes less time and efficient utilizes all resources(CPU, GPU , memory)	Has the highest time and space complexity with poor utilization of resources	Computational complexity is high and takes time in training	Is complex and time complexity depends on number of clusters defined and data size
Strength	Can handle large and structured datasets. It prevents overfitting easily. It has good computational capacity	Data normalization is not required. It doesn't suffer from the curse of dimensionality. Also, it reduces overfitting	Uses kernel trick to classify non linear data and works well in higher dimensional spaces. It is likely to overfit	It is easy to interpret and can scale to large datasets. Also it is easy to implement and adapts to new examples

Characteristics	XGBoost Model	Random Forest Model	Support Vector Classifier	K-Means Clustering
Limitations	Doesn't work well work with sparse and unstructured data	It is less interpretable with high computational power. Also, it takes in training	Doesn't work well with large sample size or when the target class overlaps	It suffers from dimensionality issues and is affected by outliers

4.4 Model Evaluation

In the flood prediction model, the algorithm's performance was evaluated and compared using accuracy, recall, precision, f1 score,auc (area under the curve) and roc (receiver operating characteristics) curve.

4.4.1 Accuracy

Performance of four models was initially evaluated using accuracy score which is defined as the ratio of number of correct predicted values to the total number of predictions. A good and realistic accuracy score lies in the range of 75-95%. This score is usually calculated by using the test data. Also, if the accuracy score is more on training data then the model is said to be overfitted. In the cross validation method, accuracy score is calculated by taking the average of accuracy scores in each iteration. The accuracy score for the model can be calculated by using sklearn.metrics library. Accuracy is not a good metric to measure a model performance.

Accuracy = Correct predicted values/ total number of predicted values

4.4.2 Confusion Matrix

Confusion matrix is a square matrix where the size of the matrix depends on the number of labels in the target class. As the target class is binary in nature the matrix is 2X2 type. The block in each matrix tells the four possibilities. First, when the actual value is true and the

predicted value is also true, known as true positive. Second, when the actual value is true and the predicted value is false, known as false negative. Third, when the actual value is false and the predicted value is also false, known as true negative. Fourth, when the actual value is false and the predicted value is true, known as false positive. Taking these four cases, a confusion matrix is created which tells the true positive count and true negative count. This matrix is widely used as it gives an overall information about the performance of the model and the error produced by the model. It is generated using sklearn.metrics library. In the flood prediction model, the algorithm aims to reduce the false negative count.

4.4.3 Precision

The confusion matrix tells the precision value which is the ratio of the number of true positives to the sum of true positives and false positives. A good precision value tells that there are less false positives i.e. the model will not detect any effect until there is one. There is always a tradeoff between precision and other metrics and it depends on the use case that which metric is important. In this project, precision was less important because the aim of the model is to correctly identify a flood event so that preventive actions could be taken. Hence, if the model detected a false case it won't have an adverse impact although extremely less precision score was not recommended. Precision score can be generated by using sklearn.metrics library.

Precision = True positive/ (True positive + False positive)

4.4.4 Recall

In the confusion matrix, recall is calculated as the ratio of the number of true positives to the sum of true positives and false negatives. This is also known as the sensitivity of the model which means that model will accurately detect an effect. In this project, recall was the most important metric as the aim of this project is to design a model which can detect a flood situation

in advance in order to implement a few preventive measures. In machine learning models, this sklearn.metrics function is used to calculate the recall for the model

Recall = True Positive/ (True positive + False negative)

4.4.5 F1 Score

F1 is one of the most important metrics in the classification model which is defined as the harmonic mean of precision and recall. The formula for f1 score is defined as the ratio of two by the sum of reciprocal of precision and recall value. This score is used especially when the target class is imbalanced. This is because if any value i.e. precision or recall is less due to machine learning algorithm's preference towards majority class, the fl score will decrease. Hence, this score is used to check the model's performance in case of imbalance target class. In this project, a trade off was done between f1 score and recall score to select the most efficient model with a good recall score and ideal f1 score.

F1 Score= 2/ (1/Precision + 1/Recall)

4.4.6 Roc Curve

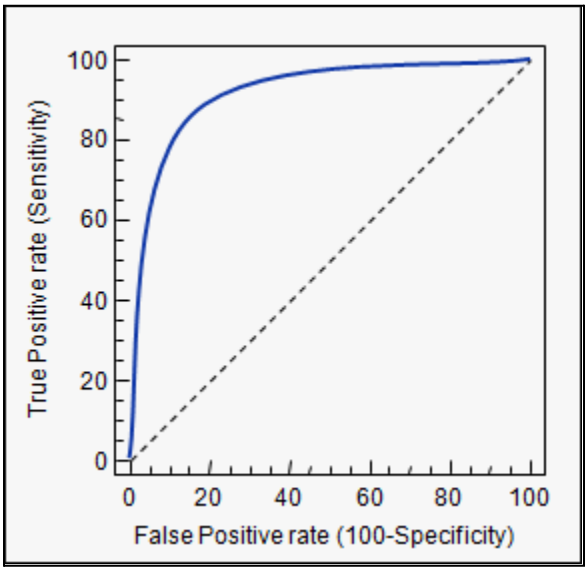
In classification models when the target class is binary type, ROC (receiver operator characteristic curve) is the most efficient to evaluate the models' performance and the error produced by the algorithm. In the roc curve the x axis defines the false positive rate and the y axis defines the true positive rate. True positive rate is also known as sensitivity of the model which is basically the recall and the false positive rate is known as one minus specificity of the model which is the ratio of false positive to the sum of false positive and true negative. The roc curve is created by plotting the false positive rate value and true positive rate value at different thresholds. The ideal case is when the false positive rate(FQR) and true positive rate(TPR) are equal to one but that is not realistic. There is a relationship between threshold, FPR and TPR. If

the threshold is decreased then the sensitivity of the model increases but the specificity decreases.If the threshold value is increased then the sensitivity of the model decreases but specificity increases.

In case of class imbalance, FPR will be high and from this identifying whether the algorithm is a good classifier or not is done. In this project, the model aims to get a less false positive rate. Figure 97 shows the ROC curve.

Figure 97

ROC curve showing true positive and false positive rate



Note. ROC Curve. From "*ROC curve analysis*", by Schoonjans, F. (2022, April 10), . MedCalc, (https://www.medcalc.org/manual/roc-curves.php)

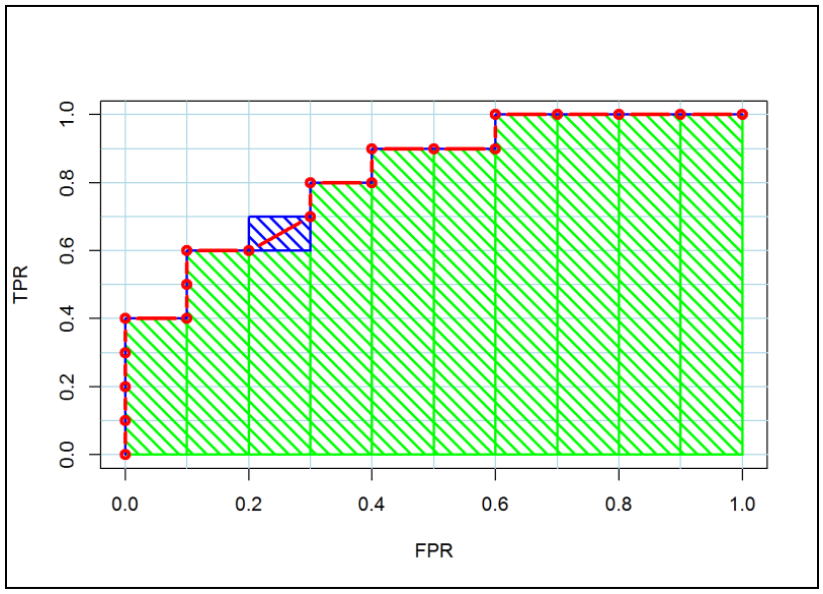
4.4.7 Auc Score

The roc curve also gives the area under the curve which is known as auc score. Ideal case is when the curve has tpr value one and fpr value zero as one of the values and the auc score is one and the worst case is when both tpr and fpr are 0.5. The auc score is calculated by finding the area for the block (i.e. height and width) at each threshold value and these values are then

added to produce the final auc score. In this flood prediction model, as there is class imbalance in the data more preference has to be given to reduce the false positive rate. Figure 98 shows how auc score is calculated for a ROC curve.

Figure 98

Calculating auc (area under the curve) score for roc curve



Note. AUC(area under the curve) for ROC Curve. From "*Calculating AUC: the area under a ROC Curve*", by Blogger, G. (n.d.). (https://blog.revolutionanalytics.com/2016/11/calculating-auc.html)

4.5 Model Validation and Evaluation

4.5.1 Performance Evaluation of Models

4.5.1.1 XGBoost Model.

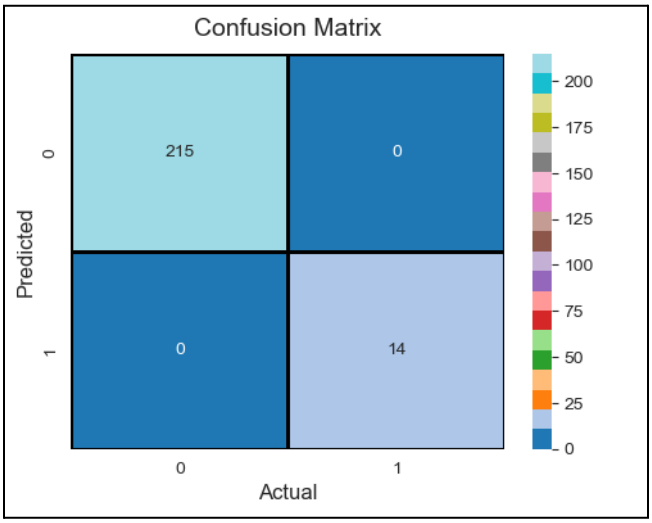
The dataset was split into training and testing data and the best parameters were found using hyper-parameter tuning. Hyperparameter tuning was performed using sklearn GridSearchCv which checks for all possible combinations of parameters and finds the best

parameter which has the highest accuracy. It took around 166 seconds to find the best parameters.

The performance of the model was tested using test data. The accuracy score on test data was 100% as the dataset was small i.e there were 916 instances in training data and 229 instances in test data. Also, the confusion matrix was plotted to see the true positives and true negatives count as shown below in figure 99.

Figure 99

Confusion matrix for XGBoost model



From figure 100, it is evident that the model might be overfitting due to the small dataset and the results provided by the model were quite unrealistic. In the next phase, recall precision and f1 score was evaluated to see the sensitivity of the model. The recall, precision and f1 score for the xgboost model were also 100%. This means that there were no false positives and false negatives values which was also evident from the confusion matrix as shown in figure 8. Below figure 100 shows the precision, recall and f1 score for the xgboost model. The roc curve and the auc score as shown in figure 101 shows that the area under the curve was one. This means that

the tpr (true positive rate) and fpr(false positive rate) was high because of imbalanced data. Further analysis was done on the roc curve by finding the tpr and fpr values at different thresholds as shown in figure 102. From figure 102, it can be seen that when the threshold value was high the fpr was zero and tpr was one. This was one of the ideal cases which rarely happens. When threshold value increases then the fpr increases to one. This happened due to target class imbalance in which the true negative was zero and the false positive value was more.

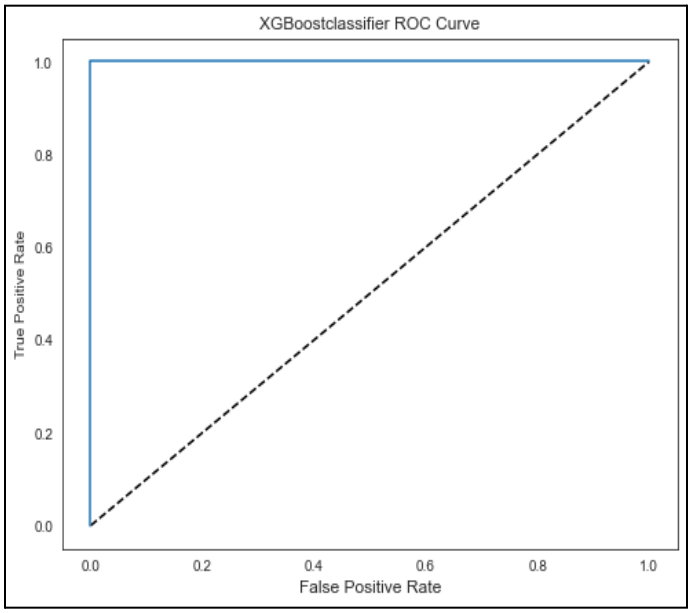
Figure 100

Sensitivity and precision value for XGBoost model

```
Precision: 100.0 %
Recall: 100.0 %
F1 Score: 100.0 %
```

Figure 101

ROC Curve and AUC score for XGBoost model




```
the AUC score for XGBoost model 1.0
```

Figure 102

Tpr and fpr for different threshold values

```
The value for tpr is 0.0 and fpr is 0.0 at threshhold value 1.6964144706726074
The value for tpr is 1.0 and fpr is 0.0 at threshhold value 0.6964144110679626
The value for tpr is 1.0 and fpr is 1.0 at threshhold value 0.30358558893203735
```

Hence, cross validation technique was implemented by implementing a three fold time series split to get the accurate results. For this, a pipeline was created to first oversampled the data and then trained the data with XGBoost best parameters. The figure 104 below shows the cross validated accuracy, recall, precision and f1 score. From this figure 104, it was evident that precision values is 100% which tells that the model doesn't detect any false positives i.e. model did not detect any false flood events whereas recall values was 91% which tells the model detected a few false negatives i.e. model failed to detect a few flood events accurately. In other words, the sensitivity of the model using cross validation was 91%, in statistical terms this is also known as power of the model. Table 28 summarizes the performance metrics on test data and by cross validation method for better understanding.

Figure 104

Three fold cross validated performance metrics results

```
Recall score for xgboost is 0.9183673469387754
Precision score for xgboost is 1.0
F1 score for xgboost is 0.9534883720930232
Accuracy score for xgboost is 0.9860139860139859
```

Table 28

Performance metrics for XGBoost model using test data and cross validation method

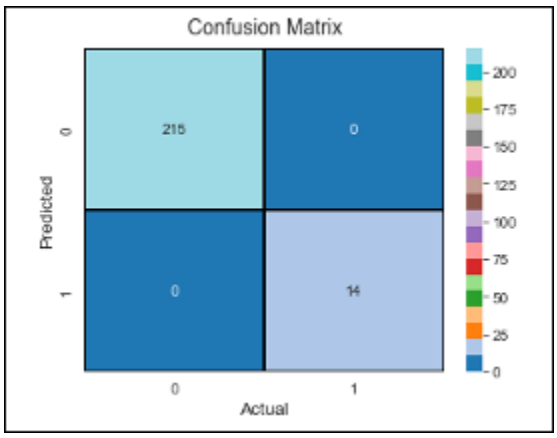
Method	Accuracy	Precision	Recall	F1 Score
Checking the accuracy on test data	100%	100%	100%	100%
Checking the accuracy by cross validation method	98.6%	100%	91.83%	95.34%

4.5.1.2 Random Forest Model.

The augmented data was split into training and testing. Hyper-parameter tuning was used to identify the optimal parameters for training the model. GridSearchCV searches through all potential parameter combinations to identify the one that has the maximum accuracy. The optimal settings were discovered in about 289 seconds. Using test data, the model's performance was evaluated. Due to the limited dataset, the accuracy score for the test data was 100%. Additionally, plotted confusion matrix to determine the number of true positives and true negatives, as seen in Figure 105 below.

Figure 105

Confusion matrix for Random Forest model



Confusion matrix for the model makes it clear that the model was overfitting because of the small dataset and the unrealistic outcomes it produces. The f1 score and recall precision were assessed in the following stage to determine the model's sensitivity. The Random Forest model had 100% recall, precision, and f1 score. The confusion matrix, which is presented in Figure 11, also demonstrates that there were zero false positives or false negatives values. Figure 106 shows precision, Recall, and F1 score for the Random Forest model below. In figure 107, the ROC curve and the AUC score are displayed.

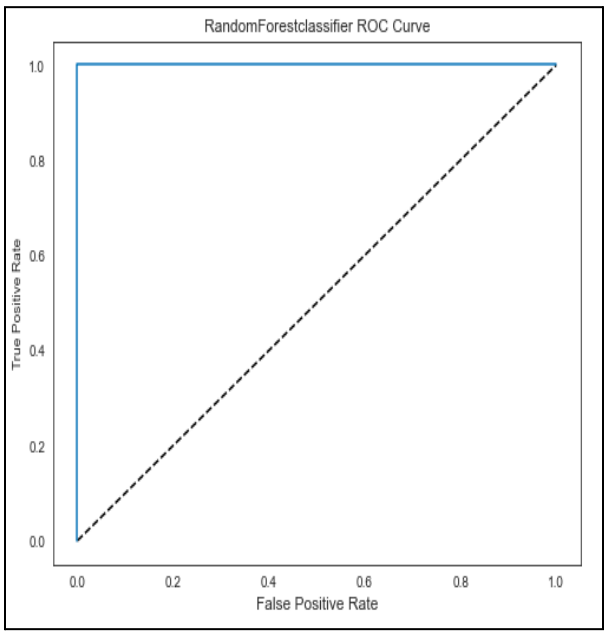
Figure 106

Precision and sensitivity value for Random Forest model

```
Precision: 100.0 %
Recall: 100.0 %
F1 Score: 100.0 %
```

Figure 107

ROC Curve and AUC Score for Random Forest model




```
the auc score for Random Forest Classifier is 1.0
```

Therefore, to obtain accurate results, a three-fold time series split cross-validation technique was used. To do this, a pipeline was developed that oversampled the data before training it using the optimal Random Forest classifier parameters. The accuracy, recall, precision, and f1 score calculated through cross-validation are displayed in figure 108 below. It makes it clear that recall values were 91%, indicating that the model detected some false negatives, or failed to accurately detect some flood events, while precision values were 100%, indicating that the model does not detect any false positives, i.e., false flood events were not detected by the model. Especially, using cross validation, the model's sensitivity was 91%, also known as power of the model in statistical terms. Table 29 gives the summary of Random Forest model performance metrics both on test data and cross validation technique applied.

Figure 108

Three-fold cross validation performance results

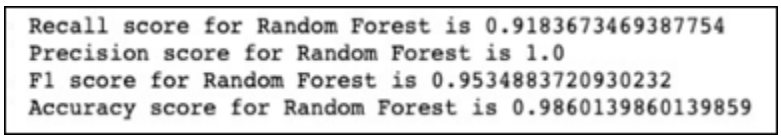


Table 29

Performance metrics of Random Forest classifier

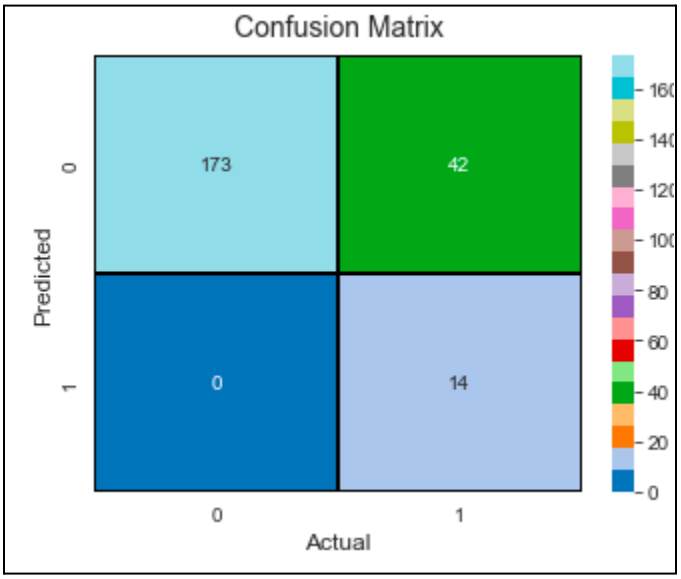
Method	Accuracy	Precision	Recall	F1 Score
Performing accuracy check on test data	100%	100%	100%	100%
Performing accuracy check by Cross Validation Technique	98.60%	100%	91.83%	95.34%

4.5.1.3 Support Vector Classifier.

Initially, all the parameters for SVM were listed out and the best parameters were found using GridSearchCV. The best parameters where the test score was high were observed for C=0.1, gamma = 0.1 and kernel = ‘linear’. The SVC Classifier was then trained using these optimal parameters. Model performance was tested on the testing data. The accuracy obtained for the SVM classifier was 100%. The precision, recall and F1 score were also 100% and the confusion matrix obtained is as shown in Figure 109.

Figure 109

Confusion matrix for SVM Classifier

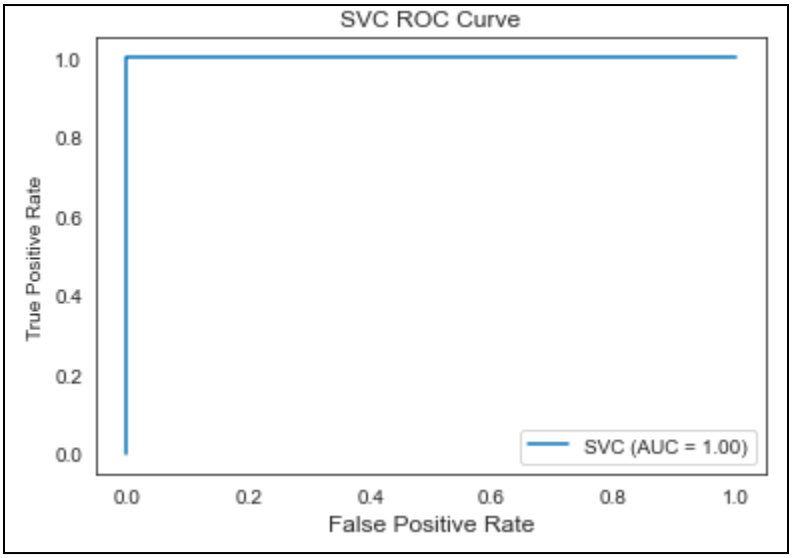


Such high values for all performance metrics indicates that no wrong predictions were made by the model. This can also be viewed from the confusion matrix as the non-diagonal terms representing the FP and FN values are zero. Imbalance in data was observed by looking at the diagonal values where diagonal terms have the values of 215 and 14 respectively. Positive value which corresponds to the column and the row header of zero in the confusion matrix indicates that there won't be any floods whereas one indicates that floods will take place.

The ROC-AUC curve was also plotted to understand the area under the curve. The chart is shown in Figure 110. The Area under the curve (AUC) value was one as the curve lies in the top left corner which again supports the above justification that the model predicted all the instances accurately.

Figure 110

ROC-AUC curve for SVM Classifier



The main reason for such high values for performance metrics was because the dataset considered was very small with around 916 instances for training data and 29 instances for testing data. In order to measure the prediction results of SVM accurately, time series cross validation was performed on the dataset where a pipeline is created to augment the entire data by passing the optimal parameters for SVM Classifier. The time required to fit and score each fold was recorded along with the accuracy, precision, recall and F1 score. Table 30 summarizes the performance metrics observed for testing data and cross validation data.

Table 30

Performance metrics for testing and validation dataset

	Recall	Precision	F1 score	Accuracy
Testing data	100	100	100	100
Cross Validation data	77.77	100	83.33	92.82

Recall value for the cross-validated data was 77.77% which means that around 77.77% of values which had no floods were correctly predicted by the model. Precision remained the same after cross validation which was 100%. This means that all the predictions that the model made stating there won't be any floods are correct out of the total predicted values. F1 score obtained was around 83.33 % which takes into account both the precision as well as the recall in its calculation whereas the accuracy score after cross validation was 92.82%.

Recall value was more important than precision for this analysis as precision talks about correctly predicted positive observations whereas recall talks about correctly predicted positive observations. It was important for this model to see out of all the flood like conditions, how many were accurately depicted as flood like conditions.

4.5.1.4 KMeans Clustering.

Once the model wass trained with the help of the optimal parameters, the evaluation metrics were computed for the test dataset to understand the model performance. The result obtained for k-means clustering is provided below.

Table 31

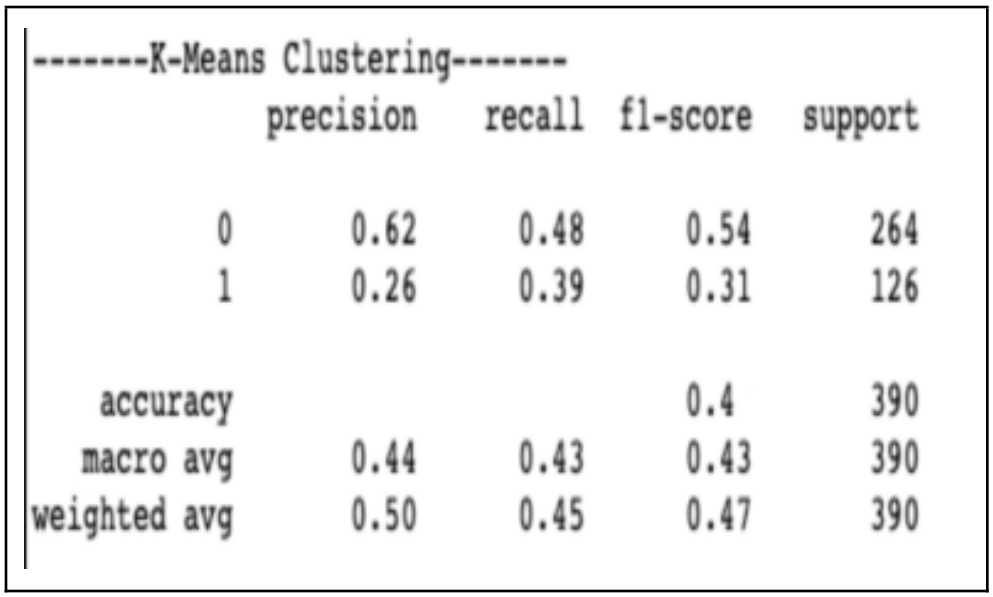
Results obtained by implementing k-means clustering

Model	Recall	Precision	F1 Score	Accuracy
K -Means Clustering	33.33 %	25.66 %	29.00 %	40.09 %

Figure 111 shows classification report for k-means clustering

Figure 111

Classification Report - K Means Clustering



-------K-Means Clustering-------

	precision	recall	f1-score	support
0	0.62	0.48	0.54	264
1	0.26	0.39	0.31	126
accuracy			0.4	390
macro avg	0.44	0.43	0.43	390
weighted avg	0.50	0.45	0.47	390

4.5.2 Models Comparison based on performance metrics

Table 32 below summarizes the cross validated results for both supervised and unsupervised models implemented in this project. All these models were trained by using optimal parameters achieved through hyper-parameter tuning.

Table 32

Cross validated performance metrics for different models

Model	Recall	Precision	F1 Score	Accuracy
XGBoost Model	91.83%	100%	95.34%	98.60%
Support Vector Classifier	77.77%	100%	83.33%	92.82%
Random Forest Classifier	91.8%	100%	95.3%	98.6%
K- Means Clustering	33.33%	25.66%	29.00%	40.09%

From table 32, the metric score for random forest model and xgboost model were almost similar. This may be because both the model uses ensemble techniques and also the dataset was small in size where 90% of the instances didn't record any flood events in Kerala in 2022. For this reason, the precision score was 100% for all supervised models because the model didn't classify any false positive case i.e. detecting a flood event when flood didn't actually happen. However, recall was not 100% as the model failed to identify a flood event accurately. In supervised learning, this was low for support vector classifiers i.e. around 78%. With reference to f1 score, the score was good for ensemble models i.e. xgboost model and random forest classifier (the score was around 95% for both the models). However, the f1 score for the support vector classifier was less i.e. 83%. For k-means clustering, the score was lowest for all metrics;

this shows that the clustering method was not efficient when the target class was imbalanced. To elaborate on this, the target class was binary in nature because of which two clusters were defined. However, the target class contains a majority of non flood events as compared to flood events. Due to this, k-means algorithm favored the majority class while calculating the similarity score for all instances in the data i.e. mapping the flood events to non-flood events class thereby increasing the false negative count. Hence, this shows that supervised learning techniques are efficient in flood prediction as compared to unsupervised learning.

4.4.3 Conclusion

In this project, a mix of supervised and unsupervised algorithms were implemented to analyze which method is more efficient. Unsupervised learning was used as sometimes the hydrology data is not labeled and by implementing unsupervised methods the model can identify a flood and non-flood event. To do this, performance comparison was done between k-means (unsupervised method) and other supervised learning methods to see which method was efficient in detecting a flood event so that preventive measures can be taken in advance.However, after evaluating the results it was concluded that supervised methods were more efficient in classifying hydrology data i.e. mapping the data to flood and non-flood events. In supervised learning methods, xgboost model and random forest model(both are ensemble methods) had good results. However, xgboost model scored higher points in terms of time and space complexity as the model utilizes all resources efficiently and the time taken in training the model is less as compared to random forest model. Therefore, in this project xgboost model which is a supervised method was preferred for flood prediction model.

4.4.4 Limitations

As real time data was collected so the number of flood events were less and only limited records were fetched using api. This happened because less rain happened in Kerala in 2022. Due to this, the model was biased towards the majority class. Although oversampling was performed on trained data, in the test data there were a few instances with flood events because of this the precision was high for all models. In other words, due to the small dataset the model failed to generalize well.

4.4.5 Future scope

In the future work, more data samples will help in training the model effectively. Also, the large dataset will help in identifying the different patterns from the data. Also, deep learning methods such as CNN, RNN can be used to identify the flood event using INSAT images. Also, the data can be recorded for heavy rainfall regions so that model captures all the essential parameters in the data. The project aims to conduct this research further by implementing neural network methods on heavy rainfall regions for better prediction.

References

Amanpreet Singh, Narina Thakur, & Aakanksha Sharma. (2016). A review of supervised machine learning algorithms. *International Conference on Computing for Sustainable Global Development*, 1310–1315.

Biau, G., & Scornet, E. (2016). A random forest guided tour. *TEST*, *25*(2), 197–227. https://doi.org/10.1007/s11749-016-0481-7

Chakraborty, S., Nagwani, N.K., & Dey, L. (2014). Weather Forecasting using Incremental K-means Clustering. *ArXiv, abs/1406.4756*. https://arxiv.org/pdf/1406.4756.pdf

Dai, W., Tang, Y., Zhang, Z., & Cai, Z. (2021, September 24). Ensemble Learning Technology for Coastal Flood Forecasting in Internet-of-Things-Enabled Smart City. *International Journal of Computational Intelligence Systems*, *14*(1). https://doi.org/10.1007/s44196-021-00023-y

de Castro Neto, M., & Sarmento, P. (2021, April). *A mixed approach for urban flood prediction using Machine Learning and GIS.* International Journal of Disaster Risk Reduction, Volume 56, 102154. https://doi.org/10.1016/j.ijdrr.2021.102154

Eisco. Mason's Hygrometer - Wet & Dry Bulb Thermometer - Wall-Mounted.https://www.eiscolabs.com/products/ph0232b

Felix, A. Y., & Sasipraba, T. (2019, December). Flood Detection Using Gradient Boost Machine Learning Approach. *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. https://doi.org/10.1109/iccike47802.2019.9004419

Garcia, F. C. C., Retamar, A. E., & Javier, J. C. (2015). A real time urban flood monitoring system for metro Manila. *TENCON 2015 - 2015 IEEE Region 10 Conference*. https://doi.org/10.1109/tencon.2015.7372990

Ghorpade, P., Gadge, A., Lende, A., Chordiya, H., Gosavi, G., Mishra, A., Hooli, B., Ingle, Y. S., & Shaikh, N. (2021, July 1). Flood Forecasting Using Machine Learning: A Review. *2021 8th International Conference on Smart Computing and Communications (ICSCC)*. https://doi.org/10.1109/icscc51209.2021.9528099

Handelman, G. S., Kok, H. K., Chandra, R. V., Razavi, A. H., Huang, S., Brooks, M., Lee, M. J., & Asadi, H. (2019). Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods. *American Journal of Roentgenology*, *212*(1), 38–43. https://doi.org/10.2214/ajr.18.20224

How are Tides Measured? - The New System - Tides and Water Levels. (n.d.). NOAA's National Ocean Service. Retrieved October 23, 2022, from https://oceanservice.noaa.gov/education/tutorial_tides/tides11_newmeasure.html

Hashi, A. O., Abdirahman, A. A., Elmi, M. A., Hashi, S. Z. M., & Rodriguez, O. E. R. (2021). A Real-Time Flood Detection System Based on Machine Learning Algorithms with Emphasis on Deep Learning. *International Journal of Engineering Trends and Technology*, *69*(5), 249–256. https://doi.org/10.14445/22315381/ijett-v69i5p232

India Meteorological Department. https://mausam.imd.gov.in/imd_latest/contents/satellite.php

Indian National Center for Ocean Information Services(INCOIS). (n.d.). Indian Tsunami Early Warning System. (n.d.). ESSO | Govt. of India. Retrieved October 23, 2022, from https://tsunami.incois.gov.in/TEWS/Abouttideguage.jsp

Jabari, S., McGrath, H., & Coleman, D. (2020, August 3). FLOOD MAPPING USING RANDOM FOREST AND IDENTIFYING THE ESSENTIAL CONDITIONING FACTORS; A CASE STUDY IN FREDERICTON, NEW BRUNSWICK, CANADA.

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, *V–3*, 609–615. https://doi.org/10.5194/isprs-annals-v-3-2020-609-2020

Jain, Pallavi & Schoen-Phelan, Bianca & Ross, Robert. (2020). *Automatic Flood Detection in Sentinel-2 Images Using Deep Convolutional Neural Networks*. https://doi.org/10.1145/3341105.3374023

Kocaman, S., Nefeslioglu, H. A., & Gokceoglu, C. (2020, August 21). A FUSION APPROACH FOR FLOOD MAPPING USING SENTINEL-1 AND SENTINEL-2 DATASETS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII–B3*, 641–648. https://doi.org/10.5194/isprs-archives-xliii-b3-2020-641-2020

Liu, Y., Wang, Y., & Zhang, J. (2012). New Machine Learning Algorithm: Random Forest. *Information Computing and Applications*, 246–252. https://doi.org/10.1007/978-3-642-34062-8_32

Lu, J. (2010, April). The practical research on flood forecasting based on artificial neural networks. *Expert Systems With Applications*, *37*(4), 2974–2977. https://doi.org/10.1016/j.eswa.2009.09.037

Marea API. (n.d.). Marea Tides API. https://api.marea.ooo/

Motta, M., de Castro Neto, M., & Sarmento, P. (2021). A mixed approach for urban flood prediction using Machine Learning and GIS. *International Journal of Disaster Risk Reduction*, *56*, 102154. https://doi.org/10.1016/j.ijdrr.2021.102154

Moumtzidou Anastasia, Bakratsas Marios, Andreadis Stelios, Karakostas Anastasios, Gialampoukidis Ilias, Vrochidis Stefanos, & Kompatsiaris Ioannis. (2020, May 24). *Flood detection with Sentinel-2 satellite images in crisis management systems.* 17th

International Conference on Information Systems for Crisis Response and Management (ISCRAM 2020). https://doi.org/10.5281/zenodo.3702303

Munawar, Hafiz Suliman & Hammad, Ahmed & Waller, Steven. (2021). *A review on flood management technologies related to image processing and machine learning*. Automation in Construction. Volume 132. 103916. https://doi.org/10.1016/j.autcon.2021.103916

Nti, I. K., Nyarko-Boateng, O., Boateng, S., Bawah, F. U., Agbedanu, P. R., Awarayi, N. S., Nimbe, P., Adekoya, A. F., Weyori, B. A., & Akoto-Adjepong, V. (2021, December 21). Enhancing Flood Prediction using Ensemble and Deep Learning Techniques. *2021 22nd International Arab Conference on Information Technology (ACIT)*. https://doi.org/10.1109/acit53391.2021.9677084

OpenWeatherMap.org. (n.d.). *Weather API - OpenWeatherMap*. https://openweathermap.org/api

Pirasteh, S., Pradhan, B., Mahmud, A. R., Sulaiman, W. N. A., & Moradi, A. (2011, December 31). An artificial neural network model for flood simulation using GIS: Johor River Basin, Malaysia. *Environmental Earth Sciences*, *67*(1), 251–264. https://doi.org/10.1007/s12665-011-1504-z

Prasad, B. S., Shreya, S., Sinha, T., & Priya, S. (2021, October 29). Ensemble Model-Based Prediction for River Management: A Case Study on River Kaveri and Coastal Karnataka. *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*. https://doi.org/10.1109/smartgencon51891.2021.9645804

Ray, S. (2019). A Quick Review of Machine Learning Algorithms. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. https://doi.org/10.1109/comitcon.2019.8862451

Razafipahatelo, D., Rakotoniaina, S., & Rakotondraompiana, S. (2014). Automatic floods detection with a kernel k-means approach. *2014 IEEE Canada International Humanitarian Technology Conference - (IHTC)*, 1-4.doi: 10.1109/IHTC.2014.7147515.

Razali, N., Ismail, S., & Mustapha, A. (2020, March 1). Machine learning approach for flood risks prediction. *IAES International Journal of Artificial Intelligence (IJ-AI)*, *9*(1), 73. https://doi.org/10.11591/ijai.v9.i1.pp73-80

Ruiz-Gonzalez, R., Gomez-Gil, J., Gomez-Gil, F., & Martínez-Martínez, V. (2014). An SVM-Based Classifier for Estimating the State of Various Rotating Components in Agro-Industrial Machinery with a Vibration Signal Acquired from a Single Point on the Machine Chassis. Sensors, 14(11), 20713–20735. https://doi.org/10.3390/s141120713

Sahoo, A., Samantaray, S., & Ghose, D.K. (2021, February). Prediction of flood in barak river using hybrid machine learning approaches: A case study. *Journal Geological Society of India*, *Vol.97*, 186–198.

Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Computer Science*, *171*, 1251–1260. https://doi.org/10.1016/j.procs.2020.04.133

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3). https://doi.org/10.1007/s42979-021-00592-x

Satellite Meteorology. *Cloud Identification*. https://cimss.ssec.wisc.edu/

Sathyadevan, S., & Nair, R. R. (2014). Comparative Analysis of Decision Tree Algorithms: ID3, C4.5 and Random Forest. *Computational Intelligence in Data Mining - Volume 1*, 549–562. https://doi.org/10.1007/978-81-322-2205-7_51

Shada, B., Chithra N.R., & Thampi.S.G. (2022). Hourly Flood Forecasting Using Hybrid Wavelet SVM. *Journal of Soft Computing in Civil Engineering*, *6*(2), 1–20. https://doi.org/10.22115/scce.2022.317761.1383

Sharma, Prativa & Kar, Bandana & Wang, Jun & Bausch, Douglas. (2021). *A machine learning approach to flood severity classification and alerting*. 42-47. https://doi.org/10.1145/3486626.3493432

Shen C., Li H., Chen M., Zhu K., and Wang F. (2022). *Prediction Model of Lake Water Volume Based on K-Means.* In The 2022 5th International Conference on Electronics, Communications and Control Engineering (ICECC 2022). Association for Computing Machinery, New York, NY, USA, 57–62. https://doi.org/10.1145/3531028.3531037

Skybrary.(2021, November 29). Automated Cloud Base and Visibility Measurement. https://www.skybrary.aero/articles/automated-cloud-base-and-visibility-measurement

S. Li, K. Ma, Z. Jin and Y. Zhu.(2016) *A new flood forecasting model based on SVM and boosting learning algorithms*. 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 1343-1348. https://doi.org/10.1109/CEC.2016.7743944

Tanim, Ahad & McRae, Callum & Tavakol-Davani, Hassan & Goharian, Erfan. (2022). *Flood Detection in Urban Areas Using Satellite Imagery and Machine Learning*. Water. 14. 1140. https://doi.org/10.3390/w14071140

Tavus, B., Kocaman, S., Nefeslioglu, H. A., & Gokceoglu, C. (2020). A FUSION APPROACH FOR FLOOD MAPPING USING SENTINEL-1 AND SENTINEL-2 DATASETS. *The*

International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, *XLIII–B3*, 641–648. https://doi.org/10.5194/isprs-archives-xliii-b3-2020-641-2020

Test and Measurement World (2019, March 23) *Relative Humidity Table | Relative Humidity*. https://www.test-and-measurement-world.com/Terminology/Relative-Humidity-T%20ble-or-Chart.html

V. Chamola, V. Hassija, S. Gupta, A. Goyal, M. Guizani and B. Sikdar.(2021). *Disaster and Pandemic Management Using Machine Learning: A Survey*. IEEE Internet of Things Journal, Vol. 8. No. 21. pp. 16047-16071. https://doi.org/10.1109/JIOT.2020.3044966

What is a tipping bucket rain gauge, and how does it work? (2020, October 6). Instrument Choice. https://www.instrumentchoice.com.au/news/what-is-a-tipping-bucket-rain-gauge-and-how-does-it-work

Water Level Measurement. (n.d.). Virginia Institute of Marine Science. Retrieved October 23, 2022, from https://www.vims.edu/research/units/labgroups/tc_tutorial/tidemeasure.php

Xu H., Ma C., Lian J.,Xu K., Chaima E.. (2018). *Urban flooding risk assessment based on an integrated k-means cluster algorithm and improved entropy weight method in the region of Haikou, China*. Journal of Hydrology. Volume 563.Pages 975-986, ISSN 0022-1694, https://doi.org/10.1016/j.jhydrol.2018.06.06

Yuan, S., & Chu, F. (2007). Fault diagnosis based on support vector machines with parameter optimisation by artificial immunisation algorithm. Mechanical Systems and Signal Processing, 21(3), 1318–1330. https://doi.org/10.1016/j.ymssp.2006.06.006

Y Shi, Khaled Taalab, & Tao Cheng. (2016). Flood Prediction Using Support Vector Machines (SVM). In: Proceedings of the 24th GIS Research UK (GISRUK) Conference. GIS Research UK (GISRUK): London, UK. (2016).

Zhu, T. (2020). Analysis on the Applicability of the Random Forest. *Journal of Physics: Conference Series*, *1607*(1), 012123. https://doi.org/10.1088/1742-6596/1607/1/012123

Ibrahem Ahmed Osman, A., Najah Ahmed, A., Chow, M. F., Feng Huang, Y., & El-Shafie, A. (2021b). Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia. Ain Shams Engineering Journal, 12(2), 1545–1556. https://doi.org/10.1016/j.asej.2020.11.011

Kraisangka, J., Rittima, A., Sawangphol, W., Phankamolsil, Y., Tabucanon, A. S., Talaluxmana, Y., & Vudhivanich, V. (2022b). Application of Machine Learning in Daily Reservoir Inflow Prediction of the Bhumibol Dam, Thailand. 2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). https://doi.org/10.1109/ecti-con54298.2022.9795552

Ma, M., Zhao, G., He, B., Li, Q., Dong, H., Wang, S., & Wang, Z. (2021b). XGBoost-based method for flash flood risk assessment. Journal of Hydrology, 598, 126382. https://doi.org/10.1016/j.jhydrol.2021.126382

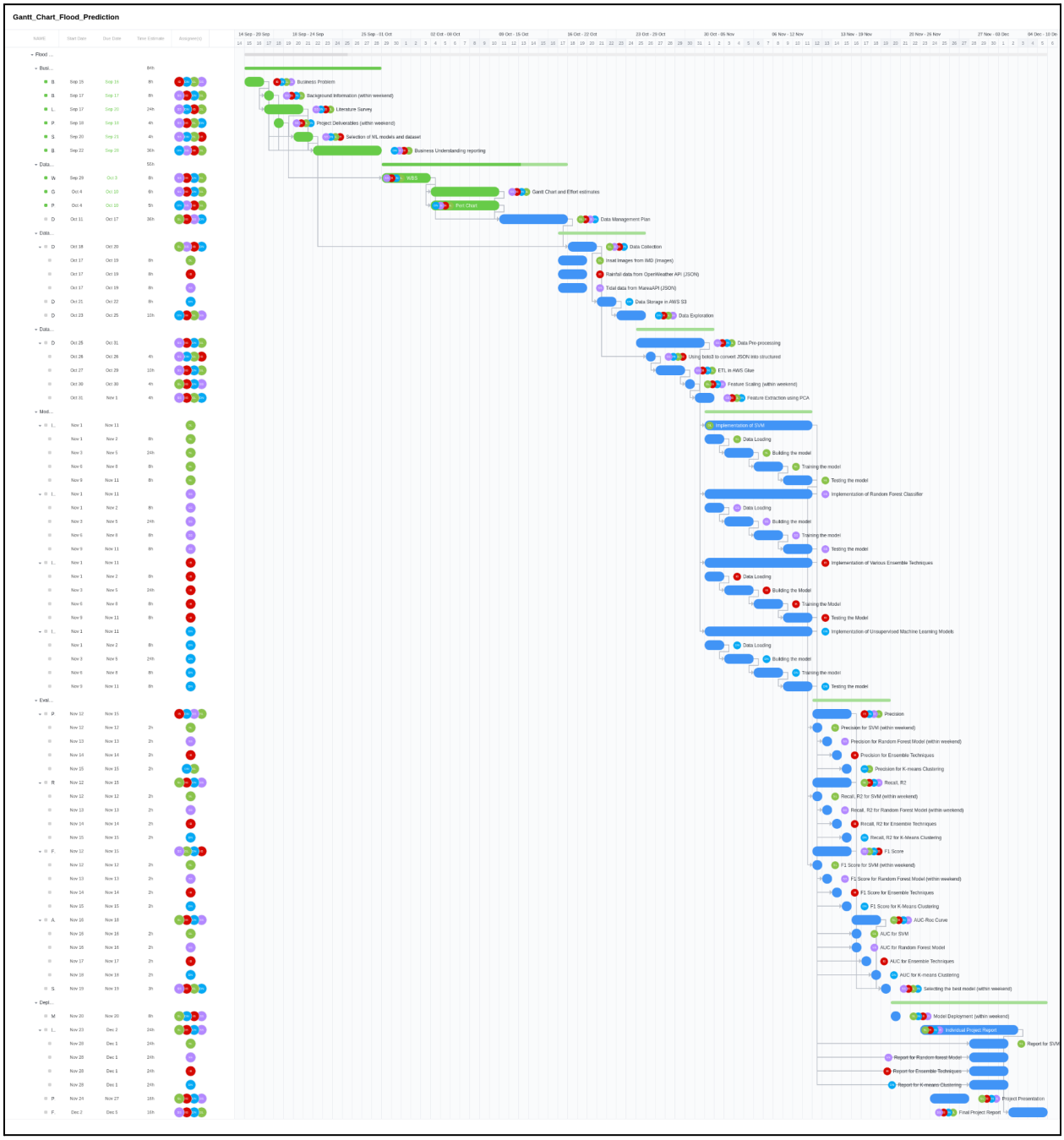
Nguyen, D. H., Hien Le, X., Heo, J. Y., & Bae, D. H. (2021b). Development of an Extreme Gradient Boosting Model Integrated With Evolutionary Algorithms for Hourly Water Level Prediction. IEEE Access, 9, 125853–125867. https://doi.org/10.1109/access.2021.3111287

Nti, I. K., Nyarko-Boateng, O., Boateng, S., Bawah, F. U., Agbedanu, P. R., Awarayi, N. S., Nimbe, P., Adekoya, A. F., Weyori, B. A., & Akoto-Adjepong, V. (2021b). Enhancing

Flood Prediction using Ensemble and Deep Learning Techniques. 2021 22nd International Arab Conference on Information Technology (ACIT). https://doi.org/10.1109/acit53391.2021.9677084

Appendix A

Gantt Chart

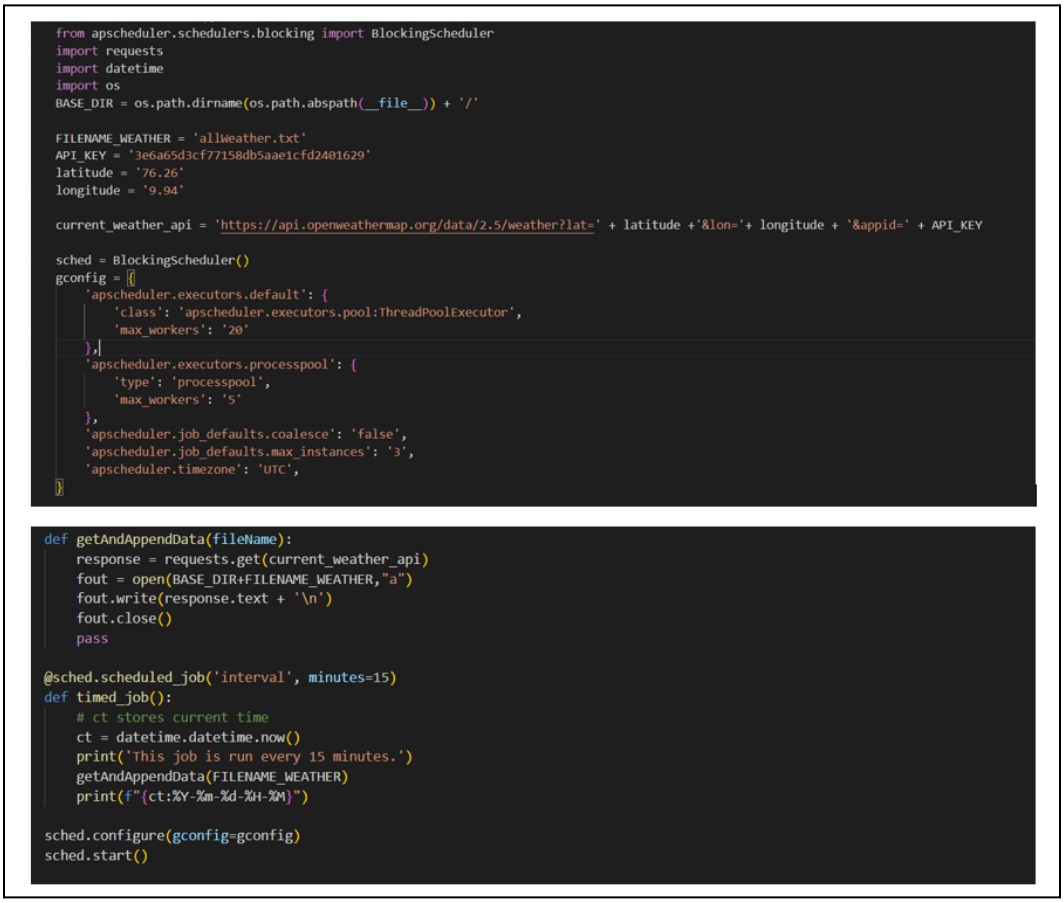


Appendix B

Python codes for data collection and data pre-processing

Figure B1

Python script for fetching data from Open Weather API



```
from apscheduler.schedulers.blocking import BlockingScheduler
import requests
import datetime
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__)) + '/'

FILENAME_WEATHER = 'allWeather.txt'
API_KEY = '3e6a65d3cf77158db5aae1cfd2401629'
latitude = '76.26'
longitude = '9.94'

current_weather_api = 'https://api.openweathermap.org/data/2.5/weather?lat=' + latitude +'&lon='+ longitude + '&appid=' + API_KEY

sched = BlockingScheduler()
gconfig = {
    'apscheduler.executors.default': {
        'class': 'apscheduler.executors.pool:ThreadPoolExecutor',
        'max_workers': '20'
    },
    'apscheduler.executors.processpool': {
        'type': 'processpool',
        'max_workers': '5'
    },
    'apscheduler.job_defaults.coalesce': 'false',
    'apscheduler.job_defaults.max_instances': '3',
    'apscheduler.timezone': 'UTC',
}
```

```
def getAndAppendData(fileName):
    response = requests.get(current_weather_api)
    fout = open(BASE_DIR+FILENAME_WEATHER,"a")
    fout.write(response.text + '\n')
    fout.close()
    pass

@sched.scheduled_job('interval', minutes=15)
def timed_job():
    # ct stores current time
    ct = datetime.datetime.now()
    print('This job is run every 15 minutes.')
    getAndAppendData(FILENAME_WEATHER)
    print(f"{ct:%Y-%m-%d-%H-%M}")

sched.configure(gconfig=gconfig)
sched.start()
```

Figure B2

Python script for fetching tidal data from Marea API

```
from apscheduler.schedulers.blocking import BlockingScheduler
import datetime;
import shutil
import requests
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__)) + '/'

url_ir = 'https://mausam.imd.gov.in/Satellite/3Dasiasec_ir1.jpg'
url_vis = 'https://mausam.imd.gov.in/Satellite/3Dasiasec_vis.jpg'
url_vap = 'https://mausam.imd.gov.in/Satellite/3Dasiasec_wv.jpg'
url_temp = 'https://mausam.imd.gov.in/Satellite/3Dasiasec_ctbt.jpg'

'''
download image
input : filename - image file to be saved as
input : url - image url to download
return : None
'''
def download_image(filename, url):
    response = requests.get(url, stream=True)
    with open(BASE_DIR+ filename, 'wb') as out_file:
        shutil.copyfileobj(response.raw, out_file)
    del response
```

```
sched = BlockingScheduler()
gconfig = {
    'apscheduler.executors.default': {
        'class': 'apscheduler.executors.pool:ThreadPoolExecutor',
        'max_workers': '20'
    },
    'apscheduler.executors.processpool': {
        'type': 'processpool',
        'max_workers': '5'
    },
    'apscheduler.job_defaults.coalesce': 'false',
    'apscheduler.job_defaults.max_instances': '3',
    'apscheduler.timezone': 'UTC',
}

@sched.scheduled_job('interval', minutes=15)
def timed_job():
    # ct stores current time
    ct = datetime.datetime.now()
    print('This job is run every 15 minutes.')
    download_image('ir/'+f"{ct:%Y-%m-%d-%H-%M}" + '.jpg',url_ir)
    download_image('wv/'+f"{ct:%Y-%m-%d-%H-%M}" + '.jpg',url_vap)
    download_image('vis/'+f"{ct:%Y-%m-%d-%H-%M}" + '.jpg',url_vis)
    download_image('temp/'+f"{ct:%Y-%m-%d-%H-%M}" + '.jpg',url_temp)
    print(f"{ct:%Y-%m-%d-%H-%M}")

sched.configure(gconfig=gconfig)
sched.start()
```

Figure B3

Python script for fetching insat images

```
from apscheduler.schedulers.blocking import BlockingScheduler
import requests
import datetime
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__)) + '/'

FILENAME_TIDE = 'allTides.txt'
API_KEY = '3a1639a1-313b-46c4-8da5-b7f577a82be3'
latitude = '76.26'
longitude = '9.94'

current_tide_api = 'https://api.marea.ooo/v2/tides?duration=1440&interval=60&latitude='+ latitude +'&longitude='+ longitude +'&model=FES2

sched = BlockingScheduler()
gconfig = {
    'apscheduler.executors.default': {
        'class': 'apscheduler.executors.pool:ThreadPoolExecutor',
        'max_workers': '20'
    },
    'apscheduler.executors.processpool': {
        'type': 'processpool',
        'max_workers': '5'
    },
    'apscheduler.job_defaults.coalesce': 'false',
    'apscheduler.job_defaults.max_instances': '3',
    'apscheduler.timezone': 'UTC',
}
```

```
def getAndAppendData(fileName):
    response = requests.get(current_tide_api, headers={"x-marea-api-token": API_KEY})
    fout = open(BASE_DIR + FILENAME_TIDE,"a")
    fout.write(response.text + '\n')
    fout.close()

@sched.scheduled_job('interval', minutes=15)
def timed_job():
    # ct stores current time
    ct = datetime.datetime.now()
    print('This job is run every 15 minutes.')
    getAndAppendData(FILENAME_TIDE)
    print(f"{ct:%Y-%m-%d-%H-%M}")

sched.configure(gconfig=gconfig)
sched.start()
```

Figure B4

Data cleaning for Tidal data

```
timeseconds=[]
for i in tidal_data.heights:
    timeseconds.append(i[0]['timestamp'])
```

```
tidal_data['timestamps'] = timeseconds
```

```
height= []
for i in tidal_data.heights:
    height.append(i[0]['height'])
```

```
tidal_data['height']=height
```

```
state=[]
for i in tidal_data.heights:
    state.append(i[0]['state'])
```

```
tidal_data['state']= state
```

```
tidal_data
```

```
tidal_data= tidal_data.rename(columns= {'height':'tidal_height','state':'tidal_state'})
```

```
tidal_data.drop(['extremes','heights','latitude','longitude','datetime','datum','unit','timezone'], inplace=True, axis=1)
```

Figure B5

Data cleaning for Weather data

```
rainfall=[]
for i in weather_data.weather:
    rainfall.append(i[0]['main'])
```

```
weather_data['Weather'] = rainfall
```

```
weather_data
```

```
typeweather=[]
for i in weather_data.weather:
    typeweather.append(i[0]['description'])
```

```
weather_data['Weather_description'] = typeweather
```

```
humidity=[]

for i in weather_data.main:
    humidity.append(i['humidity'])
```

```
weather_data['humidity'] = humidity
```

```
clouds= []

for i in weather_data.clouds:
    clouds.append(i['all'])
```

```
weather_data['cloudiness'] = clouds
```

```
weather_data.drop(['weather','main','clouds'], axis=1, inplace=True)
```

```
weather_data.rain.fillna(0,inplace=True)
```

Figure B6

Cropping, ImageEnhance and Pixel count extraction for insat images

```
# loop for all INSAT images
timestamp_list=[]
pixel_list=[]
from os import listdir
folder_dir = <path_to_images>
count=0
for images in os.listdir(folder_dir):

    # check if the image ends with png
    if (images.endswith(".jpg")):
        img_in = Image.open(<path_to_images>.format(images))
        string = images[:-4]
        timestamp_list.append(string)
        img_out = img_in.crop(box)
        enhancer = ImageEnhance.Contrast(img_out)
        img_out = enhancer.enhance(1.2)
        arr = np.array(img_out)
        count=0
        k=0
        for i in arr:
            for j in i:
                k=np.count_nonzero(j == 255)
            if k==3:
                count+=1
        pixel_list.append(count)

updated_timestamp_list=[]
for j in timestamp_list:
    j = j.replace('-', '')
    updated_timestamp_list.append(j)
df= pd.DataFrame()
df['timestamp'] = updated_timestamp_list
df['pixel_count'] = pixel_list
df['timestamp']=df['timestamp']+'00'
df['timestamp']=pd.to_datetime(df['timestamp'], format='%Y%m%d%H%M%S')
df=df.sort_values(by='timestamp')
df = df.reset_index()
del df['index']
df.to_csv('insat1.csv')
```

Figure B7

Data integration

```
merged_data= pd.merge(weather_data, tidal_data, left_on= 'dt', right_on= 'timestamps')
merged_data= pd.merge_asof(merged_data, insat_table, left_on= 'timestamp',right_on= 'timestamp', allow_exact_matches= False)
```

Appendix C

Python codes for hyper-parameter tuning for all models

Figure C1

Hyper-parameter tuning for SVC

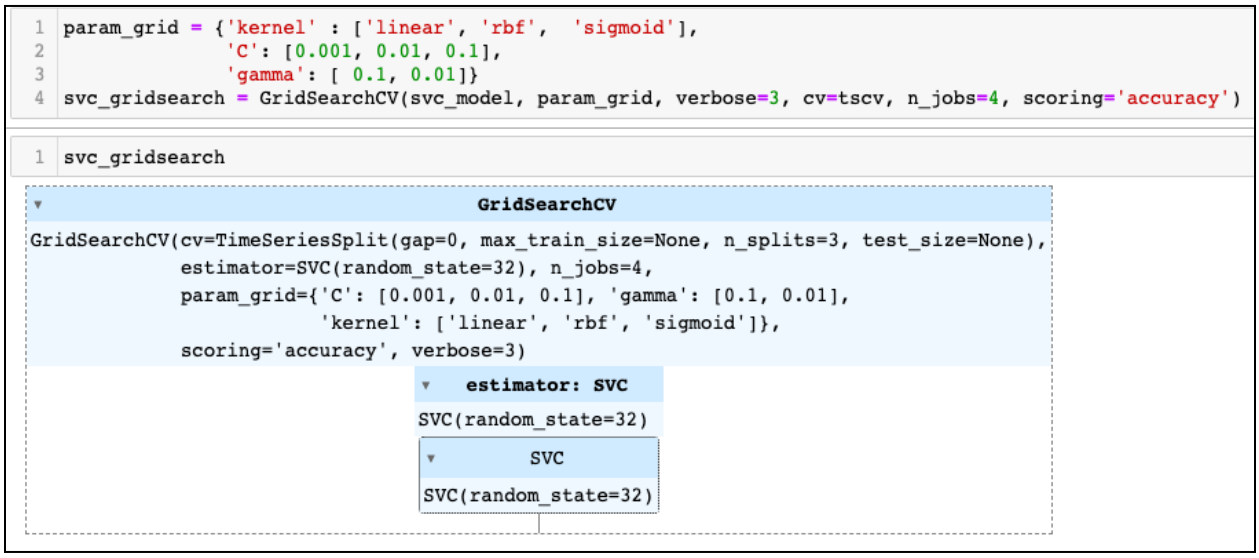


Figure C2

Hyper-parameter tuning for Random Forest Classifier

```
rf_model = RandomForestClassifier()
```

```
rf_param_grid = { 'n_estimators' : [50, 100, 150,200],
                  'max_features' : ['sqrt','log2' ,0.33],
                  'min_samples_leaf' : [1, 5 ,10, 15,20],
                  'criterion' : ['gini', 'entropy'],
                  'min_samples_split' : [3, 4,5]
}
rf_gridsearch = GridSearchCV(rf_model, rf_param_grid, verbose=3, cv= tscv, scoring='accuracy')
```

```
rf_gridsearch
```

```
GridSearchCV
GridSearchCV(cv=TimeSeriesSplit(gap=0, max_train_size=None, n_splits=3, test_size=None),
             estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['sqrt', 'log2', 0.33],
                         'min_samples_leaf': [1, 5, 10, 15, 20],
                         'min_samples_split': [3, 4, 5],
                         'n_estimators': [50, 100, 150, 200]},
             scoring='accuracy', verbose=3)
estimator: RandomForestClassifier
RandomForestClassifier
```

Figure C3

Hyper-parameter tuning for XGBoost Classifier

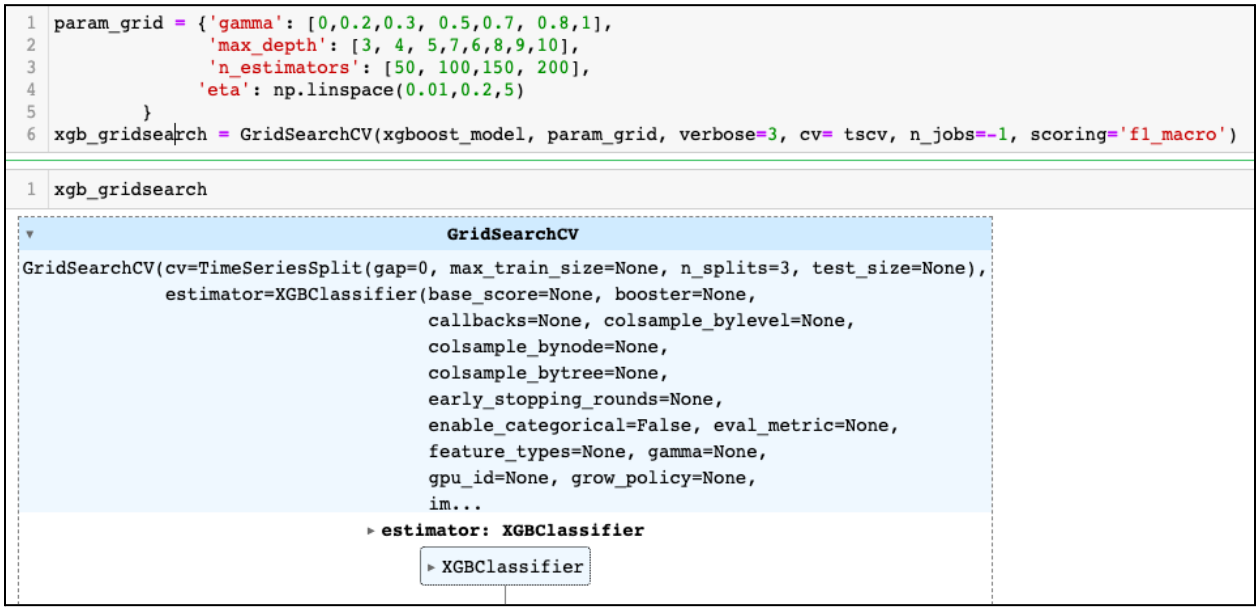


Figure C4

Hyper-parameter tuning for KMeans Clustering