Flood Detection and Prediction using Satellite Imagery and Machine Learning

Iqra Bismi

Department of Applied Data Science, San Jose State University – San Jose

Data 270: Data Analyst Process

Professor Dr. Eduardo Chan

Date: December 5, 2022

4. Model Development

Introduction

Floods are the most common natural disasters which occur due to heavy rainfall and tides. Due to this, many people, plants and animals are impacted and killed. Hence, flood forecasting is very essential so that certain precautions can be taken. Flood prediction model is complicated due to the complex nature of the hydrology data. With reference to previous research, mostly supervised learning techniques were implemented to predict floods. However, sometimes the data is unlabelled. Hence, the main goal of this project is to compare unsupervised techniques with supervised methods to see which method is more suitable for flood prediction. Data for flood prediction was collected from three different sources. First is the rainfall data collected from OpenWeather API. Rainfall volume was considered for rainfall data. Second is the tidal data collected from Marea API. Tidal height and tide state are the important parameters in tidal data. Last is the INSAT images collected from Indian Meteorological Website. For the flood prediction model, real time data was collected showing weather patterns for Kerala which is located in India.

The fetched data was initially collected in json format and was uploaded and converted into pandas dataframe. Data wrangling steps were implemented i.e. checking for data types, null values and duplicate data and changing the column names for better understanding. In the data transformation phase, pandas merge function was used to combine three data frames using the timestamp values. After this target column was added based on certain threshold values for rainfall, tidal height and humidity. Target column is a binary variable. The categorical variables in the merged data was one hot encoded and then the data was normalized so that the machine learning algorithm is unbiased.

4.1 Model Proposals

The collected data was preprocessed and transformed as per the requirement. In the modeling stage, the preprocessed data was used to implement various machine learning methods. Models were selected by taking references from previous research conducted on floods. Model selected and related work references are discussed below.

4.1.1 XGBoost Model

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.

4.1.2 Related Work References

(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. 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 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 pre processed 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 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.

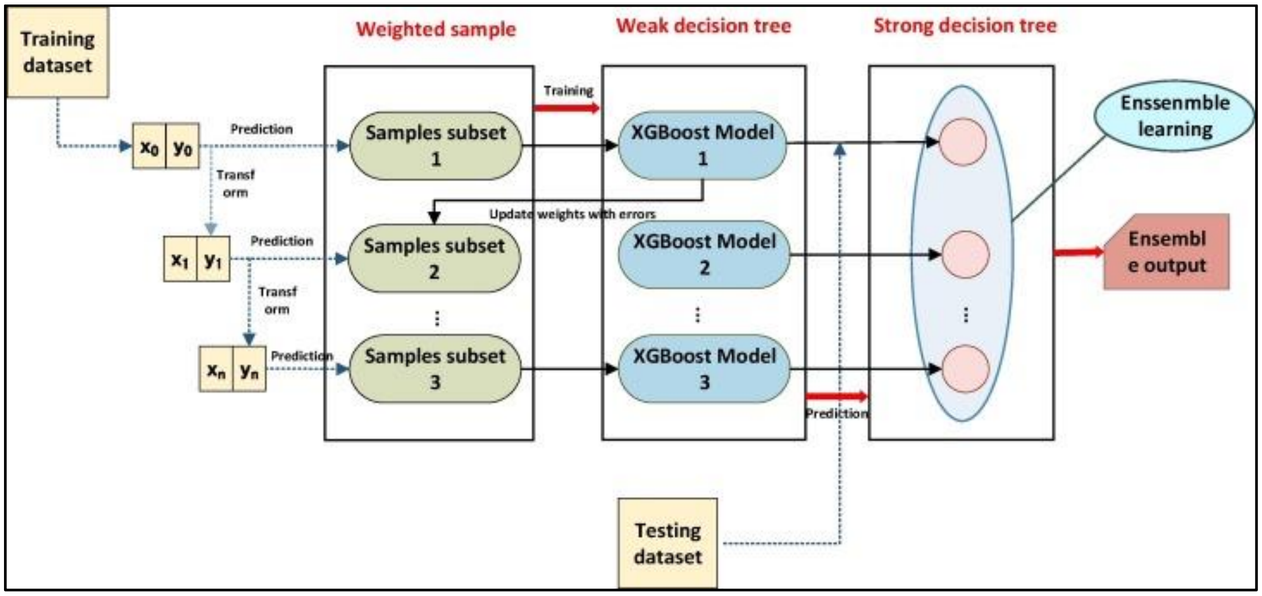
(Ibrahem et al., 2021) aimed to study ground level prediction in Malaysia using machine learning algorithms. They collected data consisting of rainfall level, temperature and evaporation to predict groundwater levels. Three machine learning methods were used; these are XGBoost model, ANN(artificial neural 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.

4.1.3 XGBoost Model algorithm

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 1 represents the XGBoost model algorithm.

Figure 1

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 1, 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.

4.1.4 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.

4.2 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 the XGBoost model. 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 includes 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 1 summarizes the different python libraries used.

Table 1

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
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
xgboost	xgboost.XGBClassifier	Model implementation	To implement extreme gradient boosting model
Pickle	pickle.dump, pickle.load	Saving and reloading	Used to saved the trained

		the model	model
imblearn	imblearn.over_sampling, imblearn.pipeline	SMOTE, imblean pipeline	Oversampling technique, Creating pipeline for ML model
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 and Data Flow

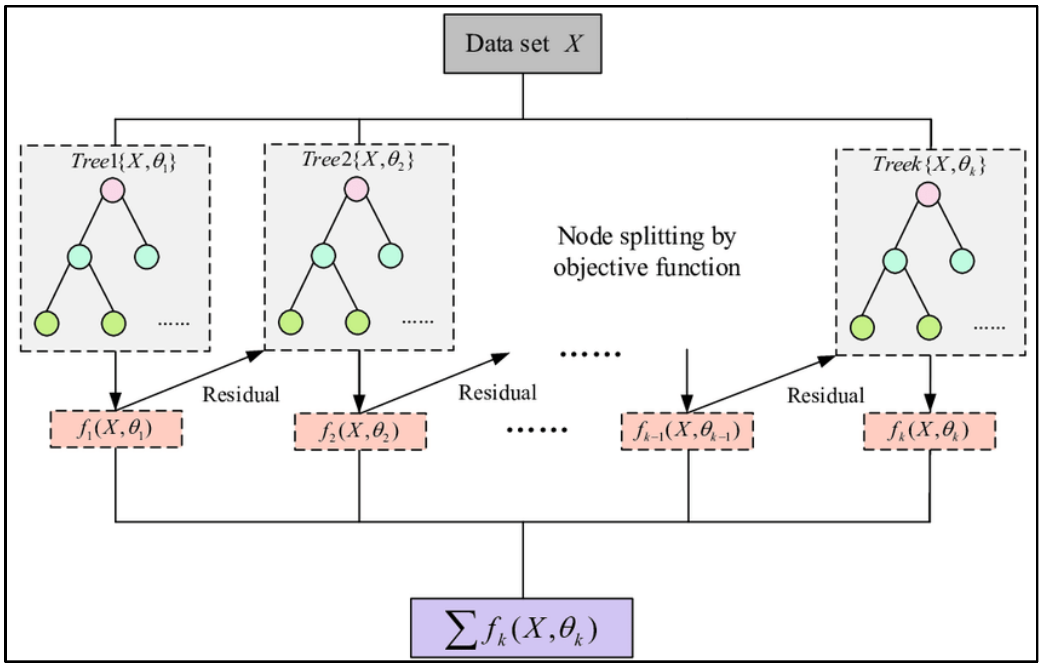
XGBoost model is one of the supervised machine learning algorithms. It is based on boosting ensemble techniques which uses regularization techniques along with gradient boosting. In boosting multiple trees known as learners are trained sequentially. These base estimators tried to correct the wrongly 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 efficiently the model learns from 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 2 shows the architecture for the XGBoost model.

Figure 2

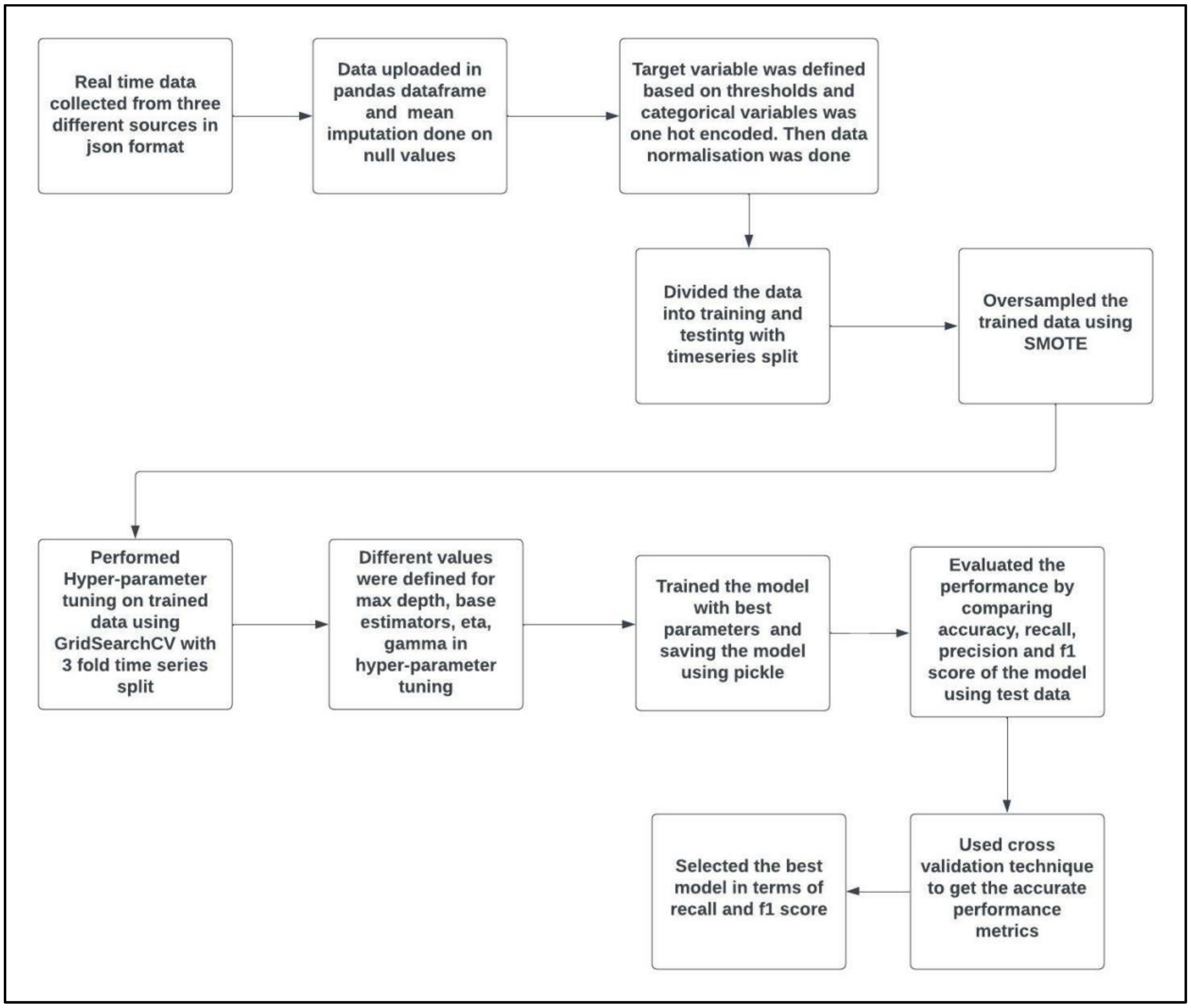
XGBoost Classifier architecture



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)

Figure 3

Flowchart for data flow using XGBoost Model


Real time data collected from three different sources in json format
Data uploaded in pandas dataframe and mean imputation done on null values
Target variable was defined based on thresholds and categorical variables was one hot encoded. Then data normalisation was done
Divided the data into training and testintg with timeseries split
Oversampled the trained data using SMOTE
Performed Hyper-parameter tuning on trained data using GridSearchCV with 3 fold time series split
Different values were defined for max depth, base estimators, eta, gamma in hyper-parameter tuning
Trained the model with best parameters and saving the model using pickle
Evaluated the performance by comparing accuracy, recall, precision and f1 score of the model using test data
Used cross validation technique to get the accurate performance metrics
Selected the best model in terms of recall and f1 score

In the dataflow stage, 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 data frame(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 includes 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 a flood event 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 transformed into a two dimensional array and the target variable was also converted into a separate one dimensional array. Tidal state and weather description were categorical variables which consisted of two and six 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.

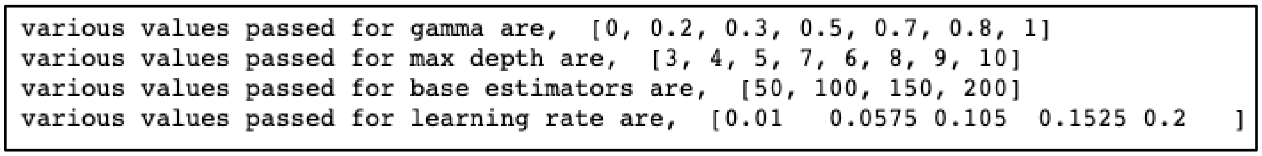
The data is a time series data so time series splitting was done i.e. manually assigning the first 80% of the data to the training dataset (with first 916 records) and the last 20% of the data to the test data set.

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 as shown in figure 4. These are max depth which is to define the depth of the tree (values ranges from three to 10), base estimators which is used to determine the number of estimators in the model (values are 50, 100,150, 200), gamma is the minimum loss reduction to create a new tree-split (values range from zero to one with a difference of 0.2) and learning rate which tell how quickly the model will fit the residual errors(values are in the range 0.01 to 0.2 with 5 steps using np.linspace).Also, time series cross validation technique was used in GridSearchCv with value set to 3. 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. In figure 5, we can see multiple combinations of parameters defined which resulted in 1120 combinations with three fold time series splitting thereby three iterations i.e. total 3360 fits. The parameter n_jobs was set to -1 so that all processors can run parallel.

Figure 4

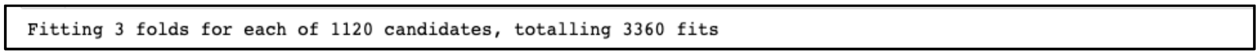
Parameters defined for XGBoost Hyper-parameter tuning



```
various values passed for gamma are,  [0, 0.2, 0.3, 0.5, 0.7, 0.8, 1]
various values passed for max depth are,  [3, 4, 5, 7, 6, 8, 9, 10]
various values passed for base estimators are,  [50, 100, 150, 200]
various values passed for learning rate are,  [0.01   0.0575 0.105  0.1525 0.2   ]
```

Figure 5

Total combinations in XGBoost model hyper parameter tuning



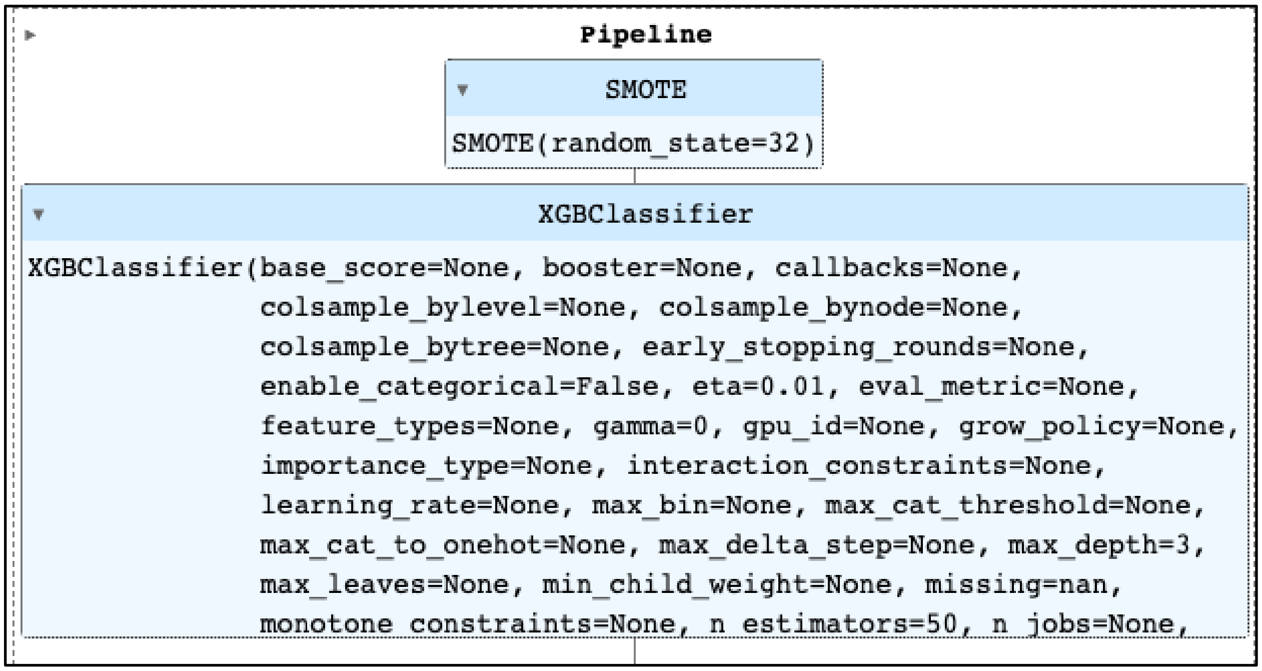
```
Fitting 3 folds for each of 1120 candidates, totalling 3360 fits
```

After getting the best parameters from the hyper-parameter the model was 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 and reloading time during model 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. The XGBoost model was 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 and tested as shown in figure 6.

Figure 6

Pipeline created for XGBoost model for cross validation method



```
Pipeline
SMOTE
SMOTE(random_state=32)
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eta=0.01, eval_metric=None,
              feature_types=None, gamma=0, gpu_id=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=3,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, n_estimators=50, n_jobs=None,
```

Lastly, comparison and analysis was done with other algorithms (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 i.e.. 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. Below table 2 summarizes the comparison between different models implemented in this project on various parameters.

Table 2

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
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 Methods

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. Trade-off is usually done between precision and other metrics as it depends on the use case that which metric is important. In this project, precision is 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 is detecting a false case it won't have an adverse impact although extremely less precision score is 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 is 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

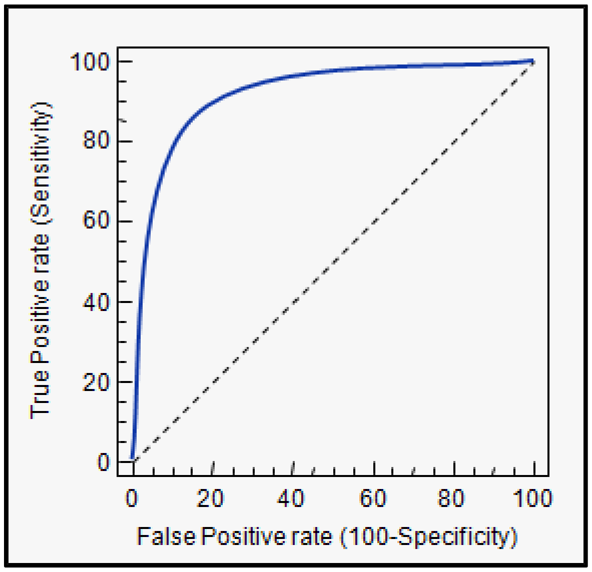
In classification models when the target class is binary type, ROC (receiver operator characteristic curve) is the most efficient to evaluate the models' performance and the error produced by the algorithm. In the roc curve the x axis defines the false positive rate and the y axis defines the true positive 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. Ideal case is when the false positive rate(fpr) 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 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 we can identify whether the algorithm is a good classifier or not. In this project, the model aims to get minimal false positive rate.

Figure 7 shows the ROC curve.

Figure 7

ROC curve showing true positive and false positive rate



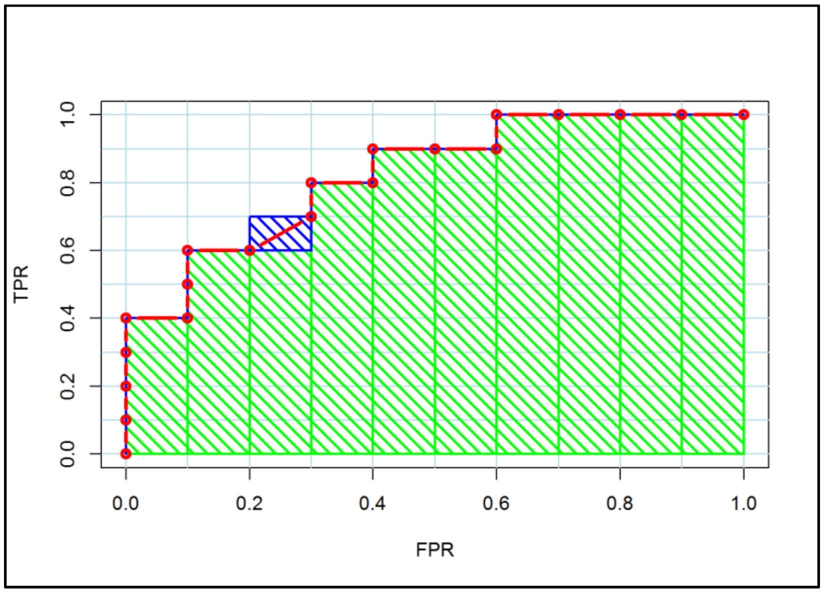
Note. ROC Curve. From "*ROC curve analysis*", by Schoonjans, F. (2022, April 10), . MedCalc, (https://www.medcalc.org/manual/roc-curves.php)

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 worst case occurs 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 a cumulative sum of these values is taken 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 8 shows how AUC score is calculated for a ROC curve.

Figure 8

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)

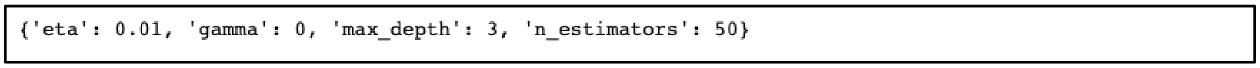
4.5 Model Validation and Evaluation

4.5.1 XGBoost Model

The dataset was split into training and testing data and best parameters were found using hyper-parameter tuning. Hyperparameter tuning was performed using sklearn GridSearchCv which checks the accuracy using multiple combinations of parameters and finds the best parameter which has the highest accuracy. The best parameters for the model are shown in figure 9. It took around 166 seconds to find the best parameters.

Figure 9

Training the XGBoost model using best parameters

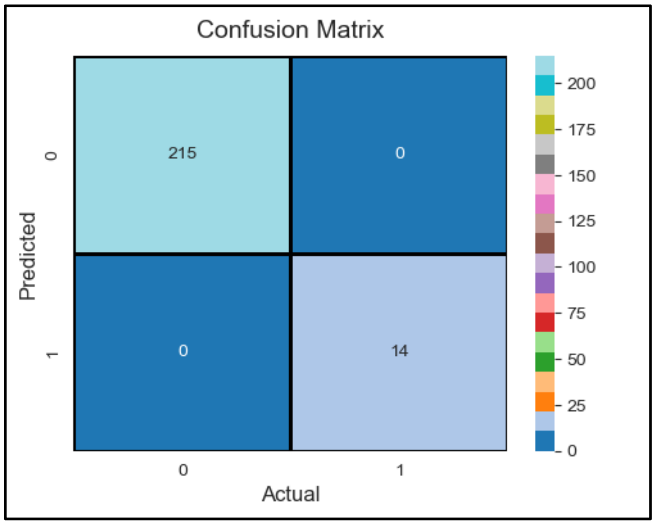


```
{'eta': 0.01, 'gamma': 0, 'max_depth': 3, 'n_estimators': 50}
```

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 10.

Figure 10

Confusion matrix for XGBoost model

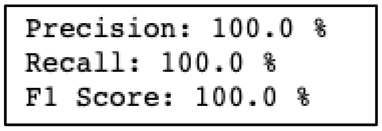


From figure 10, it is evident that the model might be overfitting due to the small dataset and the results provided by the model are quite unrealistic. In the next phase, recall precision and f1 score was evaluated to see the sensitivity of the model. The recall, precision and f1 score for the XGBoost model are also 100%. This means that there are no false positives and false negatives values which is also evident from the confusion matrix as shown in figure 8. Below figure 11 shows the f1 score, precision and recall for the XGBoost model. The roc curve and the

AUC score as shown in figure 12 shows that the area under the curve is one. This means that the tpr (true positive rate) and fpr(false positive rate) is 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 13. From figure 13, it can be seen that when the threshold value was high the fpr was zero and tpr was one. This is one of the ideal cases which rarely happens. When threshold value increases then the fpr increases to one. This happens due to target class imbalance in which the true negative is zero and the false positive value is more.

Figure 11

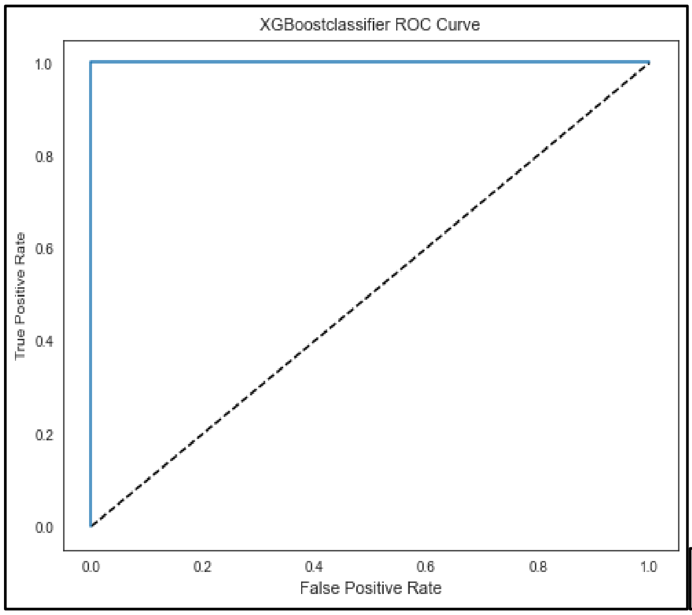
Sensitivity and precision value for XGBoost model

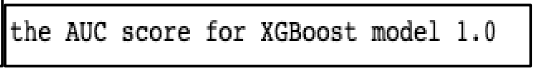


```
Precision: 100.0 %
Recall: 100.0 %
F1 Score: 100.0 %
```

Figure 12

ROC Curve and AUC score for XGBoost model

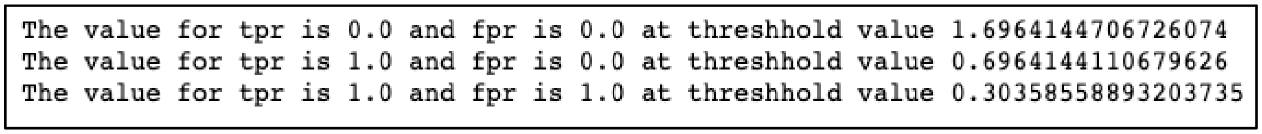




```
the AUC score for XGBoost model 1.0
```

Figure 13

Tpr and fpr for different threshold values

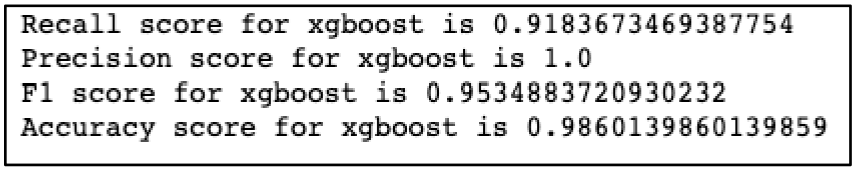


```
The value for tpr is 0.0 and fpr is 0.0 at threshhold value 1.6964144706726074
The value for tpr is 1.0 and fpr is 0.0 at threshhold value 0.6964144110679626
The value for tpr is 1.0 and fpr is 1.0 at threshhold value 0.30358558893203735
```

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 14 below shows the cross validated accuracy, recall, precision and f1 score. From this figure 14, it is 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 is 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 3 summarizes the performance metrics on test data and by cross validation method for better understanding.

Figure 14

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 3

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%

4.5.2 Comparing performance metrics with other models

Table 4 below summarizes the cross validated results for XGBoost model and other models implemented in this project. All these models were trained by using optimal parameters achieved through hyper-parameter tuning.

Table 4

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 4, we can see that 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 is small in size where 90% of the instances didn’t record any flood events in Kerala in 2022. For this reason, the precision score is 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 is 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, f1 score for support vector classifier was less i.e. 83%. For k-means clustering, score was lowest for all metrics; this shows that clustering method was not efficient when the target class is imbalanced. To elaborate on this, the target class is 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.

4.5.3 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 is 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 are 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.

4.5.4 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.

4.5.5 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

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