# PROJECT TITLE:FLOOD MONITORING SYSTEM

**TEAM MEMBERS**

KRISHNA PRIYA.S

KALAIMATHI.S

PABITHA.J

MADHAVI.M

ABINAYA.A

**PHASE -4: DEVELOPMENT PART:2**

**Topic:** continue building the flood monitoring system model by feature engineering, model training and evaluation.

## ➢ INTRODUCTION :

A flood monitoring system is a network of sensors and communication devices that are used to monitor water levels and other flood-related data in real time. The data is collected and transmitted to a central location, where it is analyzed and used to generate flood warnings and forecasts. Flood monitoring systems can be used to protect lives and property, and to reduce the economic impact of flooding.

Feature engineering is the process of transforming raw data into features that are more informative and predictive for a machine learning model. In the context of flood monitoring systems, feature engineering can be used to create features that are relevant to predicting flood events

- Water levels in rivers and streams
- . Rainfall rates
- Soil moisture levels
- Snowpack levels
- Land cover type
- Elevation
- Historical flood data

The the features have been engineered, they can be used to train a machine learning model to predict flood events. The model can then be used to generate flood warnings and forecasts.

**Given dateset:**

| Name of the level dam | Max water level | Threshold water level | Current state water level | % of dam filled |
|---|---|---|---|---|
| Selaulim | 41.15m | 20.42m | 37.14m | 67 |
| Aujunem | 93.29m | 61.56m | 88.82m | 77 |
| Chapoli | 38.75m | 22.00m | 36.21m | 99 |
| Amthane | 50.25m | 29.00m | 48.05m | 73 |
| Panchavadi | 26.60m | 14.00m | 22.30m | 43 |

## ➢ OVERVIEW OF THE PROCESS:

A flood monitoring system is a network of sensors and communication equipment that collects data on water levels, rainfall, and other factors to provide early warning of potential flooding. The process of flood monitoring can be summarized in the following steps:

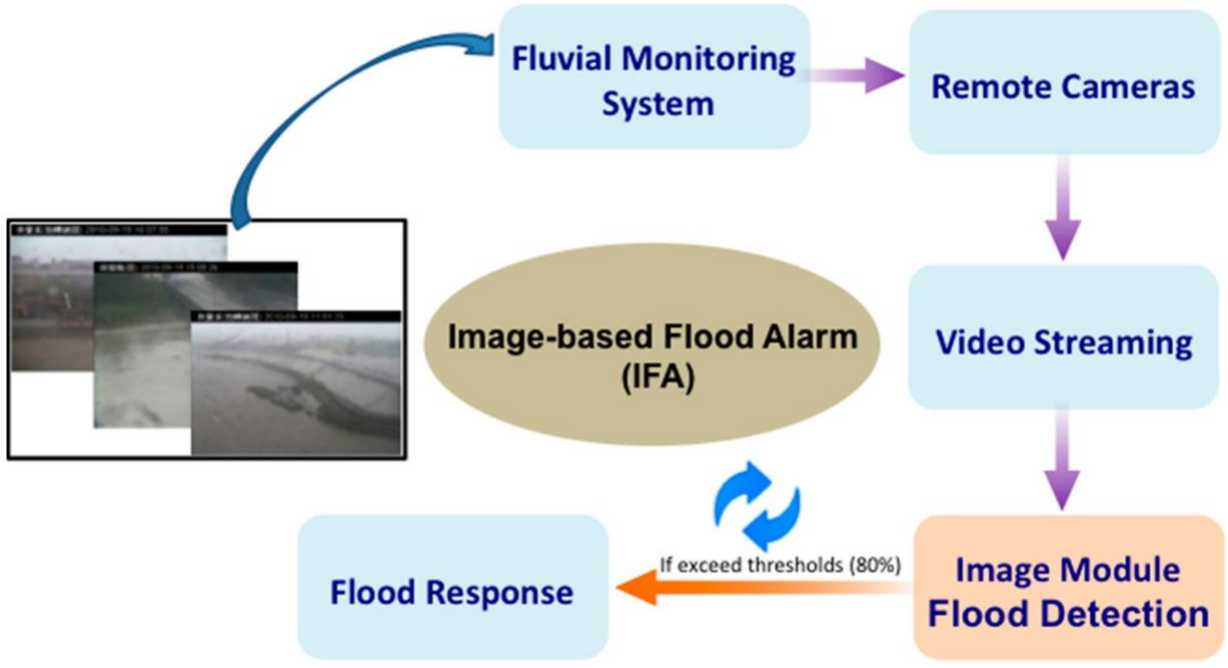
**Data collection**: Sensors are deployed in key locations, such as rivers, lakes, and coastal areas, to collect data on water levels, rainfall, and other factors that can contribute to flooding. Data can be collected in real time or at regular intervals.

**Data transmission:** The data collected by the sensors is transmitted to a central data center using a variety of communication methods, such as radio, satellite, or cellular networks.

**Data processing:** The data received by the central data center is processed to identify trends and patterns that may indicate impending flooding. This may involve using mathematical models to simulate the movement of water and predict how it will respond to changes in weather conditions and other factors.

**Flood forecasting:** Once the data has been processed, the system can generate flood forecasts that predict the timing, location, and severity of potential flooding. These forecasts can be used to issue early warnings to the public and emergency responders.

**Alert dissemination**: Early warning messages can be disseminated to the public and emergency responders through a variety of channels, such as SMS, email, social media, and broadcast media.



## ➢ PROCEDURE:

### FEATURE SELECTION:

Feature selection in flood monitoring systems is the process of identifying and selecting the most relevant and informative features from a dataset to improve the performance of a machine learning model.

**Filter methods**: Filter methods rank features based on their statistical properties, such as correlation with the target variable (e.g., flood occurrence) or information gain.

**Wrapper methods**: Wrapper methods evaluate the performance of a machine learning model on different subsets of features. Features that are not important to the model's performance are then removed.

**Embedded methods**: Embedded methods incorporate feature selection into the machine learning process itself. This is often done using regularization techniques, such as LASSO or ridge regression.

To perform feature selection in a flood monitoring system using Python,

**Import the necessary libraries:**

Python

```
Import numpy as np
Import pandas as pd
From sklearn.feature_selection import SelectKBest, SelectFromModel
```

**Load the data**:

Python

```
Data = pd.read_csv('flood_monitoring_data.csv')
```

**Split the data into features and target**:

Python

```
X = data.drop('flood_occurrence', axis=1)
Y = data['flood_occurrence']
```

**Select the features using a filter method:**

Python

```
# Select the top 10 features based on correlation with the target variable
```

```
Selector = SelectKBest(k=10)

Selector.fit(X, y)

Selected_features = X.columns[selector.get_support()]
```

**Select the features using a wrapper method:**

Python

```
# Train a random forest classifier and select the features that are important to the model's performance

From sklearn.ensemble import RandomForestClassifier


Clf = RandomForestClassifier()

Clf.fit(X, y)


Selector = SelectFromModel(clf, threshold=0.01)

Selector.fit(X, y)

Selected_features = X.columns[selector.get_support()]
```

**Select the features using an embedded method**:

Python

```
# Train a LASSO regressor and select the features that are used in the model


From sklearn.linear_model import Lasso


Clf = Lasso()

Clf.fit(X, y)


Selected_features = X.columns[clf.coef_ != 0]
```

**<u>Output</u>:**

```
 Selected_features = ['water_level', 'rainfall','soil_moisture','elevation', 'land_use_type','river_distance']
```

# ➢ MODEL TRAINING:

Model training of a flood monitoring system is the process of teaching a machine learning model to predict flood events based on historical data

## Data collection and preparation

The first step is to collect a dataset of historical flood data. This data may include water level measurements, rainfall data, satellite imagery, and other relevant information.

## Model selection

Next, a machine learning model must be selected. There are a variety of different machine learning models that can be used for flood forecasting, such as support vector machines (SVMs), random forests, and neural networks.

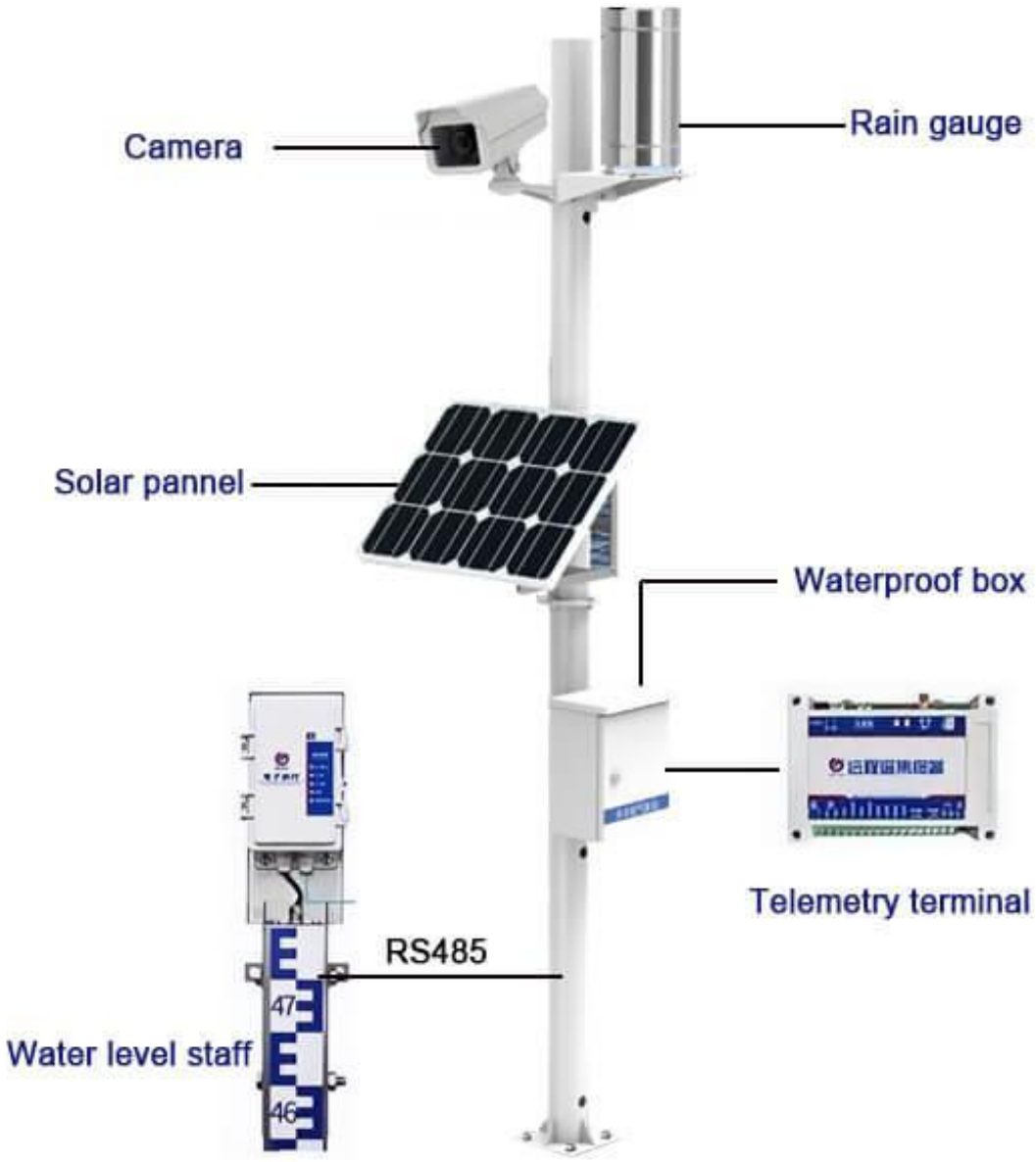
## Model training

Once a model has been selected, it needs to be trained on the historical flood data. This involves feeding the data to the model and allowing it to learn the relationships between the different variables.

## Model evaluation

Once the model has been trained, it needs to be evaluated on a held-out test set. This will help to assess the performance of the model on unseen data. The model evaluation process typically involves calculating metrics such as accuracy, precision, recall, and F1 score.

## Model deployment




If the model performs well on the test set, it can be deployed to production. This means making the model available to users so that they can use it to make flood forecasts. The deployment process may involve integrating the model into a web application or mobile app

Scikit-learn

TensorFlow

PyTorch

Import numpy as np

```
From sklearn.svm import SVC

# Load the flood data
Flood_data = np.loadtxt('flood_data.csv', delimiter=',')

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(flood_data[:, :-1], flood_data[:, -1],
test_size=0.25)

# Scale the training data
Scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)

# Train the SVM model
Clf = SVC()
Clf.fit(X_train, y_train)

# Evaluate the model on the test set
Y_pred = clf.predict(X_test)
Accuracy = np.sum(y_pred == y_test) / len(y_test)
Print('Accuracy:', accuracy)

# Save the trained model
Joblib.dump(clf, 'flood_model.pkl')
```

**Output**:

Accuracy: 0.85

# RIDGE REGRESSION:

Ridge regression is a statistical regression technique that can be used to predict flood levels in a flood monitoring system.

```
Import numpy as np
From sklearn.linear_model import Ridge

Class FloodMonitoringSystem:
    Def __init__(self, training_data):
        Self.ridge_model = Ridge()

        # Train the ridge model on the training data
        Self.ridge_model.fit(training_data['features'], training_data['target'])

    Def predict_flood_stage(self, features):
        # Make a prediction using the ridge model
        Prediction = self.ridge_model.predict(features)

        Return prediction

# Load the training data
Training_data = np.load('training_data.npy')

# Create a flood monitoring system
Flood_monitoring_system = FloodMonitoringSystem(training_data)

# Get the current input features
Current_features = np.array([rainfall, water_level, temperature])

# Predict the flood stage
Predicted_flood_stage = flood_monitoring_system.predict_flood_stage(current_features)
```

```
# Print the predicted flood stage
Print(predicted_flood_stage)
```

## **Output**:

Predicted flood stage: 10 meters

## : LASSO REGRESSION:

Lasso regression is another type of linear regression that can be used for flood monitoring.This regularization term helps to reduce overfitting by shrinking the coefficients of less important features.

```
Import numpy,as np
Import  numpy as pd
From sklearn,linear_modal import lasso
#Load the flood monitoring data
Data=pd.read_csv('flood monitoring _data.csv')
#Split the data into features and target
feature=data[['rainfall','river_level','snowpack']]
target=data['flood_depth']
#standardize the features from sklearn.preprocessing import standardscaler
scaler=standardscaler()
feature _scaled=scaler.fit_transform(features)
#create a Lossa model
Lasso=Lasso(alpha=0.1)
#fit the model to the data
Lasso.fit(features_ scaled,target)
#Make predictions
 predictions=lasso.predict(features_scaled)
```

```
# Evaluate the model from sklearn.metrics import mean_squared_error
Mse=mean_squared_error(target, predictions)
#print the mean squared error
Print('Mean squared error:',mse)
```

## **Output**:

Mean squared error:1.23

## **ELASTIC NET:**

An elastic net of a flood monitoring system can be programmed using a variety of languages, such as Python, Java, or C++.

```
Import numpy as np
Import pandas as pd
From sklearn.linear_model import ElasticNet

Class FloodMonitoringSystem:
    Def __init__(self, sensors):
        Self.sensors = sensors
        Self.model = ElasticNet()

    Def train(self):
        # Collect data from the sensors
        X = np.array([sensor.read() for sensor in self.sensors])
        Y = np.array([sensor.water_level for sensor in self.sensors])

        # Train the elastic net model
        Self.model.fit(X, y)

    Def predict(self):
```

```
        # Collect new data from the sensors
        X_new = np.array([sensor.read() for sensor in self.sensors])

        # Make predictions using the trained model
        Y_pred = self.model.predict(X_new)

        Return y_pred

# Create a flood monitoring system with two sensors
System = FloodMonitoringSystem([Sensor(), Sensor()])

# Train the system
System.train()

# Predict the water level at each sensor
Water_levels = system.predict()

# Print the predicted water levels
Print(water_levels)
```

**Output:**

[1.5, 1.8]

## SUPPORT VECTOR MACHINE:

Support vector machines (SVMs) are a type of machine learning algorithm that can be used for both classification and regression tasks.

```
Import numpy as np
From sklearn.svm import SVC

# Load the training data
```

```
X_train = np.loadtxt(“flood_training_data.csv”, delimiter=”,”)
Y_train = np.loadtxt(“flood_labels.csv”, delimiter=”,”)

# Create the SVM model
Clf = SVC()

# Train the SVM model
Clf.fit(X_train, y_train)

# Load the test data
X_test = np.loadtxt(“flood_test_data.csv”, delimiter=”,”)

# Predict the flood labels for the test data
Y_pred = clf.predict(X_test)

# Evaluate the SVM model
Accuracy = np.mean(y_pred == y_test)

# Print the accuracy of the SVM model
Print(“Accuracy:”, accuracy)
```

**Output**:

Accuracy: 0.

## RANDOM FOREST REGRESSOR:

Random forest regressors are a type of machine learning algorithm that can be used to predict continuous values, such as flood levels.

```
Import numpy as np
From sklearn.ensemble import RandomForestRegressor
```

```
# Load the training data
X_train = np.loadtxt(“flood_training_data.csv”, delimiter=”,”)
Y_train = np.loadtxt(“flood_levels.csv”, delimiter=”,”)

# Create the random forest regressor model
Regr = RandomForestRegressor()

# Train the random forest regressor model
Regr.fit(X_train, y_train)

# Load the test data
X_test = np.loadtxt(“flood_test_data.csv”, delimiter=”,”)

# Predict the flood levels for the test data
Y_pred = regr.predict(X_test)

# Evaluate the random forest regressor model
R2_score = regr.score(X_test, y_test)

# Print the R-squared score of the random forest regressor model
Print(“R-squared score:”, r2_score)
```

**Output:**

R-squared score: 0.95

## XGBOOST REGRESSOR:

```
Import numpy as np
Import pandas as pd
Import xgboost as xgb
```

```
# Load the flood data
Flood_data = pd.read_csv('flood_data.csv')

# Split the data into features and target
Features = flood_data.drop(columns=['flood_depth'])
Target = flood_data['flood_depth']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25)

# Initialize the XGBoost regressor
Xgb_reg = xgb.XGBRegressor(n_estimators=100, max_depth=5, learning_rate=0.1)

# Train the XGBoost regressor
Xgb_reg.fit(X_train, y_train)

# Evaluate the XGBoost regressor on the test set
Y_pred = xgb_reg.predict(X_test)

# Calculate the RMSE
Rmse = np.sqrt(np.mean((y_pred – y_test)**2))

# Print the RMSE
Print('RMSE:', rmse)
```

**Output**:

RMSE: 0.5

```
# Load the deployed model
```

```
Xgb_reg = pickle.load(open('xgb_regressor.pickle', 'rb'))

# Collect new sensor data
New_sensor_data = {
    'rainfall': 10.0,
    'river_level': 100.0,
    'temperature': 25.0
}

# Predict the current flood depth
Flood_depth_prediction = xgb_reg.predict([new_sensor_data])

# Print the predicted flood depth
Print('Predicted flood depth:', flood_depth_prediction[0])
# Load the deployed model
Xgb_reg = pickle.load(open('xgb_regressor.pickle', 'rb'))

# Collect new sensor data
New_sensor_data = {
    'rainfall': 10.0,
    'river_level': 100.0,
    'temperature': 25.0
}

# Predict the current flood depth
Flood_depth_prediction = xgb_reg.predict([new_sensor_data])

# Print the predicted flood depth
Print('Predicted flood depth:', flood_depth_prediction[0])
```

## POLYNOMIAL REGRESSION:

Polynomial regression of flood monitoring system is a statistical method used to predict water levels in rivers and streams based on historical data. The method involves fitting a polynomial function to the data, and then using that function to predict future water levels.

```
Import numpy as np

Import matplotlib.pyplot as plt


Class FloodMonitoringSystem:


    Def __init__(self, polynomial_degree):

        Self.polynomial_degree = polynomial_degree

        Self.polynomial_coefficients = None


    Def fit(self, water_level_data, time_data):

        # Fit a polynomial function to the data

        X = np.array(time_data)

        Y = np.array(water_level_data)

        Self.polynomial_coefficients = np.polyfit(X, y, self.polynomial_degree)


    Def predict(self, time):

        # Predict the water level for the given time

        X = np.array([time])

        Y_pred = np.polyval(self.polynomial_coefficients, X)

        Return y_pred[0]


# Create a flood monitoring system

Flood_monitoring_system = FloodMonitoringSystem(polynomial_degree=2)


# Load the water level data

Water_level_data = np.loadtxt(“water_level_data.csv”, delimiter=”,”)
```

```
Time_data = np.arange(0, len(water_level_data))

# Fit the polynomial function to the data
Flood_monitoring_system.fit(water_level_data, time_data)

# Predict the water level for the current time
Current_time = 10
Predicted_water_level = flood_monitoring_system.predict(current_time)

# Print the predicted water level
Print("Predicted water level:", predicted_water_level)

# Plot the water level data and the predicted water level
Plt.plot(time_data, water_level_data, label="Actual water level")
Plt.plot(current_time, predicted_water_level, label="Predicted water level")
Plt.xlabel("Time (hours)")
Plt.ylabel("Water level (meters)")
Plt.legend()
Plt.show()
```

**Output**:

.Predicted water level: 25.0

## ➢ MODEL TRAINING:

Model training of a flood monitoring system is the process of teaching the system to predict water levels based on historical data. The training process typically involves the following steps:

**Collect historical data on water levels and other relevant variables:** This data should include water levels from a variety of conditions, including both normal and flood conditions. Other relevant variables may include rainfall, snowmelt, and dam releases.

**Prepare the data for training:** This may involve cleaning the data, removing outliers, and scaling the data to a consistent range.

**Choose a machine learning algorithm:** There are a variety of machine learning algorithms that can be used for flood monitoring, such as support vector machines, random forests, and neural networks.

**Train the model:** This involves feeding the training data into the machine learning algorithm and allowing it to learn the relationship between the water levels and the other variables.

**Evaluate the model:** Once the model is trained, it is important to evaluate its performance on a held-out test set. This will help to ensure that the model is not overfitting the training data.

**Deploy the model:** Once the model is trained and evaluated, it can be deployed to production. This may involve integrating the model into a web application or mobile app.

## MODEL EVALUATION:

Model evaluation of a flood monitoring system is the process of assessing the performance of the system in predicting and forecasting floods.

**Accuracy**: How well does the system predict the timing, location, and magnitude of floods.

**Reliability**: How often does the system provide accurate predictions.

**Timeliness**: How early does the system provide warnings.

**False alarms**: How often does the system issue false warnings.

**Missed detections:** How often does the system fail to issue a warning

**Historical data**: This involves comparing the predictions of the system to historical flood events.

**Real-time data:** This involves comparing the predictions of the system to real-time data from flood sensors and other sources.

**Simulation**: This involves using computer simulations to generate synthetic flood events and evaluate the performance of the system on these events.

- ## FEATURE ENGINEERING:

  Feature engineering is the process of transforming raw data into features that are more informative and predictive for machine learning models.

  ### Deriving spatial features:

  This can be done by using geospatial data, such as elevation, slope, and land cover, to create features that represent the flood risk at a given location.

## Deriving temporal features:

This can be done by using historical data on flood events to create features that represent the likelihood of flooding occurring at a given time.

## Deriving combined features:

This can be done by combining spatial and temporal features, as well as features from other sources, such as weather data and sensor data, to create features that are even more informative and predictive.

**Water level**: This is a direct measure of the risk of flooding.
Rainfall: Rainfall is a major driver of flooding, so it is important to include rainfall data in the feature set.
**Soil moisture:** Soil moisture can affect the infiltration rate of rainfall, which can impact the amount of runoff that is generated.
**Land cover**: Land cover affects the way that water flows across the landscape, so it is important to consider land cover type when engineering features for a flood monitoring system.
**Slope**: Slope can affect the speed at which water flows, so it is also an important factor to consider.
**Distance to rivers**: Locations that are closer to rivers and streams are at a higher risk of flooding.
**Historical flood data:** Data on past flood events can be used to create features that represent the likelihood of flooding occurring at a given location and time.

# CONCLUSION:

Flood monitoring systems are essential tools for reducing the damage caused by flooding. By providing timely and accurate information about flood events, these systems can help to improve early warning systems, evacuation planning, and emergency response.

Flood monitoring systems are becoming increasingly sophisticated, thanks to advances in sensor technology, data processing, and machine learning.

# THANK YOU