Flood Monitoring and Early Warning

1. **Feature Engineering**:
    - Feature Selection: Identify and select relevant features (variables or attributes) from your dataset. This helps reduce noise and improve model performance.
    - Feature Transformation: Transform or preprocess features to make them more suitable for modelling. This can include techniques like scaling, one-hot encoding, or creating new features from existing ones.
    - Handling Missing Data: Address missing values in your dataset, which can involve imputation or removal of instances with missing data.
    - Feature Creation: Generate new features based on domain knowledge or by applying mathematical transformations to existing features.
    - Feature Scaling: Normalize or standardize features to ensure they have similar scales, which can be important for certain algorithms.
2. **Model Training**:
    - Data Splitting: Split your dataset into training, validation, and test sets. The training set is used to train the model, the validation set helps tune hyperparameters, and the test set assesses model performance.
    - Model Selection: Choose an appropriate machine learning algorithm or model architecture based on the nature of your problem (classification, regression, clustering, etc.).
    - Hyperparameters Tuning: Optimize model hyperparameters through techniques like grid search, random search, or Bayesian optimization to improve model performance.
    - Training the Model: Use the training data to train the selected model, adjusting the model's parameters to minimize a chosen loss function.
3. **Model Evaluation**:
    - Performance Metrics: Select appropriate evaluation metrics for your specific problem. Common metrics include accuracy, precision, recall, F1 score, mean squared error, or area under the ROC curve.
    - Cross-Validation: Use techniques like k-fold cross-validation to assess model performance more robustly and reduce the risk of overfitting.
    - Model Interpretability: Understand how the model is making predictions, especially in cases where interpretability is crucial, such as in healthcare or finance.
    - Model Comparison: Compare different models to select the best-performing one, considering trade-offs like complexity, accuracy, and interpretability.
4. **Model Deployment**:

- Once you have a trained and evaluated model, you can deploy it to make predictions on new, unseen data. Deployment may involve creating APIs, integrating the model into a web application, or deploying it to a production environment.
- Ongoing Monitoring: Continue to monitor the model's performance in the production environment and update it as needed to maintain accuracy.

Coding:

```
//Early Flood Detection Using IOT ~ A project by Sabyasachi Ghosh

//<LiquidCrystal.h> is the library for using the LCD 16x2

#include <LiquidCrystal.h>


LiquidCrystal.h lcd(2, 3, 4, 5, 6, 7);  // Create an instance of the LiquidCrystal library

const int in = 8;                          // This is the ECHO pin of The Ultrasonic sensor HC-SR04

const int out = 9;                    // This is the TRIG pin of the ultrasonic Sensor HC-SR04

// Define pin numbers for various components

const int green = 10;

const int orange = 11;

const int red = 12;

const int buzz = 13;


void setup()

{

  // Start serial communication with a baud rate of 9600
```

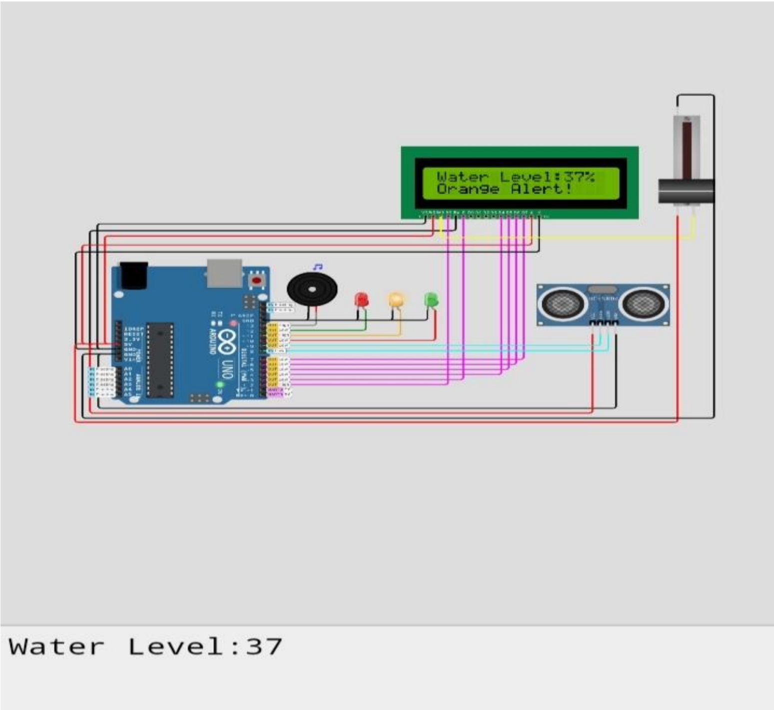
```
  Serial.begin(9600);
  // Initialize the LCD with 16 columns and 2 rows
  lcd.begin(16, 2);
  // Set pin modes for various components
  pinMode(in, INPUT);
  pinMode(out, OUTPUT);
  pinMode(green, OUTPUT);
  pinMode(orange, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(buzz, OUTPUT);
  // Display a startup message on the LCD
  lcd.setCursor(0, 0);
  lcd.print("Flood Monitoring");
  lcd.setCursor(0, 1);
  lcd.print("Alerting System");
  // Wait for 5 seconds and then clear the LCD
  delay(5000);
  lcd.clear();
}

void loop()
{
  // Read distance from the ultrasonic sensor (HC-SR04)
  long dur;
  long dist;
  long per;
  digitalWrite(out, LOW);
```

```
delayMicroseconds(2);
digitalWrite(out, HIGH);
delayMicroseconds(10);
digitalWrite(out, LOW);
dur = pulseIn(in, HIGH);
dist = (dur * 0.034) / 2;
// Map the distance value to a percentage value
per = map(dist, 10.5, 2, 0, 100);
// Ensure that the percentage value is within bounds
if (per < 0)
{
  per = 0;
}
if (per > 100)
{
  per = 100;
}
// Print water level data to serial
Serial.print("Water Level:");
Serial.println(String(per));
lcd.setCursor(0, 0);
lcd.print("Water Level:");
lcd.print(String(per));
lcd.print("%  ");
// Check water level and set alert levels
if (dist <= 3)
{
```

```
  lcd.setCursor(0, 1);
  lcd.print("Red Alert!   ");
  digitalWrite(red, HIGH);
  digitalWrite(green, LOW);
  digitalWrite(orange, LOW);
  digitalWrite(buzz, HIGH);
  delay(2000);
  digitalWrite(buzz, LOW);
  delay(2000);
  digitalWrite(buzz, HIGH);
  delay(2000);
  digitalWrite(buzz, LOW);
  delay(2000);
}
else if (dist <= 10)
{
  lcd.setCursor(0, 1);
  lcd.print("Orange Alert!  ");
  digitalWrite(orange, HIGH);
  digitalWrite(red, LOW);
  digitalWrite(green, LOW);
  digitalWrite(buzz, HIGH);
  delay(3000);
  digitalWrite(buzz, LOW);
  delay(3000);
}
else
```

```
  {
   lcd.setCursor(0, 1);
   lcd.print("Green Alert!  ");
   digitalWrite(green, HIGH);
   digitalWrite(orange, LOW);
   digitalWrite(red, LOW);
   digitalWrite(buzz, LOW);
  }
}
```

<u>Output</u>



Output for web browser