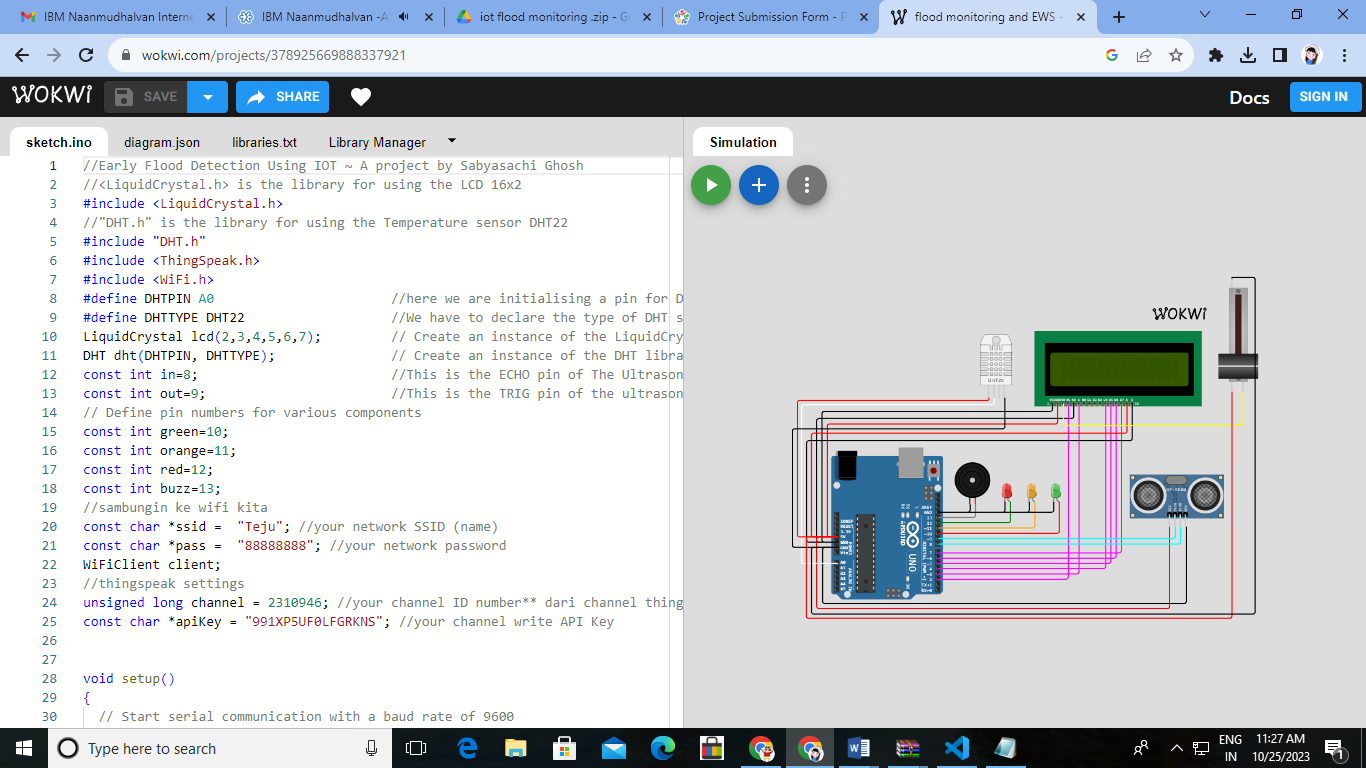
It seems like you're interested in understanding how different activities, such as feature engineering, model training, evaluation, flood monitoring, and early warning, are related. Let me break down these components and explain how they fit together in the context of flood monitoring and early warning systems:

1. **Flood Monitoring**: Flood monitoring involves the continuous collection of data related to water levels, weather conditions, and other relevant environmental factors. Various sensors, such as river gauges, weather stations, and satellite imagery, are used to gather this data. The primary goal of flood monitoring is to track changing conditions and assess the potential for flooding.
2. **Feature Engineering**: Feature engineering is a crucial step in machine learning and data analysis. In the context of flood monitoring, it involves selecting and creating relevant features (input variables) from the collected data. For example, you might engineer features like historical rainfall patterns, river discharge rates, ground elevation, and land use in the flood-prone areas. Feature engineering helps improve the quality of input data for the machine learning models.
3. **Model Training**: Machine learning models are trained using historical data to predict future events or conditions. In the context of flood monitoring, you can train models to predict the likelihood of a flood occurring based on the engineered features. Common machine learning techniques include decision trees, random forests, neural networks, or more specialized models like hydrological models for flood prediction.
4. **Evaluation**: After training a model, it is crucial to evaluate its performance. You need to assess how well the model can predict flooding events or the severity of potential floods. Common evaluation metrics include accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC-ROC). The choice of metrics depends on the specific objectives of the early warning system.
5. **Early Warning**: The ultimate goal of flood monitoring and predictive modeling is to provide early warnings to the relevant authorities and the public. When the model predicts an imminent flood event, alerts and warnings can be issued, allowing people to take necessary precautions and evacuate if required. The effectiveness of the early warning system depends on the accuracy of the models, the reliability of the data, and the timeliness of alerts.

The flow of these activities can be summarized as follows:

1. Data is continuously collected through monitoring stations.
2. Data goes through feature engineering to extract relevant information.
3. Machine learning models are trained using historical data and engineered features.
4. These models are then used to make predictions regarding potential floods.
5. The models are continuously evaluated and updated to improve accuracy.
6. When the model predicts a flood, an early warning is issued to the relevant authorities and the public.

This integrated approach helps in proactive flood management and response, potentially saving lives and reducing property damage in flood-prone areas.



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

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

#include <LiquidCrystal.h>

//"DHT.h" is the library for using the Temperature sensor DHT22

#include "DHT.h"

#include <ThingSpeak.h>

#include <WiFi.h>

#define DHTPIN A0                       //here we are initialising a pin for DHT22

#define DHTTYPE DHT22                   //We have to declare the type of DHT sensor we are using for its correct functionality

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

DHT dht(DHTPIN, DHTTYPE);               // Create an instance of the DHT library for the DHT22 sensor

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;

//sambungin ke wifi kita

const char \*ssid =  "Teju"; //your network SSID (name)

const char \*pass =  "88888888"; //your network password

WiFiClient client;

//thingspeak settings

unsigned long channel = 2310946; //your channel ID number\*\* dari channel thingspeak yg telah kita buat

const char \*apiKey = "991XP5UF0LFGRKNS"; //your channel write API Key

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);

  // Initialize the DHT sensor

  dht.begin();

  // Set initial states for LEDs and buzzer to LOW (off)

  digitalWrite(green,LOW);

  digitalWrite(orange,LOW);

  digitalWrite(red,LOW);

  digitalWrite(buzz,LOW);

  // 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();

  //connect to WiFi

**Serial**.print("Connecting to: "); **Serial**.println(ssid);

    WiFi.begin(ssid, pass);

    while (WiFi.status() != WL\_CONNECTED) {

      delay(500);

**Serial**.print(".");

    }

**Serial**.println("\nWiFi connected\n");

    ThingSpeak.begin(client); //initialize ThingSpeak

}

void loop()

{

  // Read temperature and humidity from the DHT22 sensor

  float T = dht.readTemperature();

  float H = dht.readHumidity();

  // Check if the sensor data is valid

  if (isnan(H) && isnan(T)) {

    lcd.print("ERROR");

    return;

  }

  float f = dht.readTemperature(true);

  // 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 sensor data and percentage value to serial

**Serial**.print(("Humidity: "));

**Serial**.print(H);

**Serial**.print(("%  Temperature: "));

**Serial**.print(T);

**Serial**.print("%   Water Level:");

**Serial**.println(String(per));

  lcd.setCursor(0,0);

  lcd.print("Temperature:");

  lcd.setCursor(0,1);

  lcd.print("Humidity   :");

  lcd.setCursor(12,0);

  lcd.print(T);

  lcd.setCursor(12,1);

  lcd.print(H);

  delay(1000);

  lcd.clear();

  lcd.print("Water Level:");

  lcd.print(String(per));

  lcd.print("%  ");

  // Check water level and set alert levels

  int x = ThingSpeak.writeField(channel, 1, dist, apiKey);

  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);

  }

}