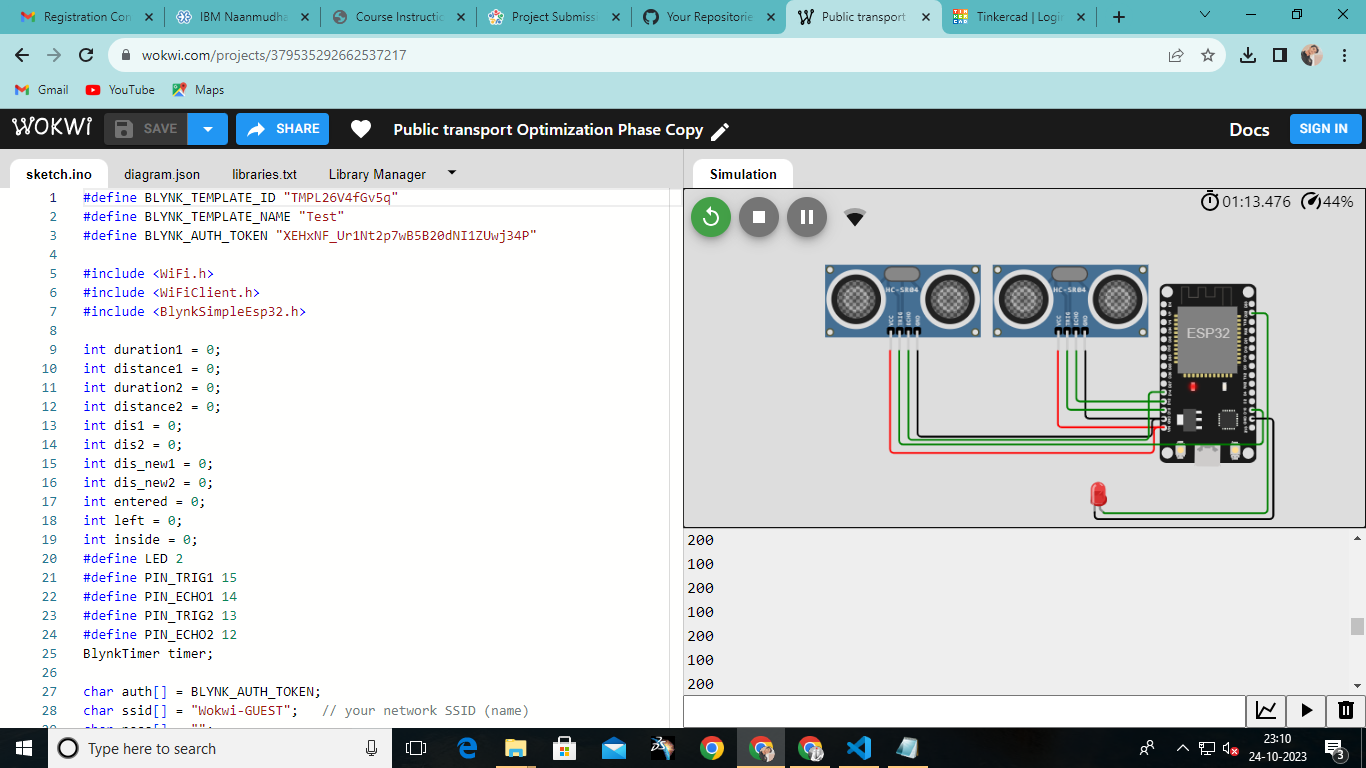
**PUBLIC TRANSPORT OPTIMIZATION**

Certainly, I can provide a general outline of the typical activities involved in a data science or machine learning project, including feature engineering, model training, and evaluation. Keep in mind that the specific steps and techniques can vary based on your project's goals and the nature of your data. Here's a high-level overview:

1. **Problem Definition and Data Collection:**
   * Clearly define the problem you want to solve.
   * Gather relevant data for your project. Ensure data quality and integrity.
2. **Exploratory Data Analysis (EDA):**
   * Understand the structure of your data.
   * Visualize and analyze data to gain insights.
   * Handle missing data and outliers as needed.
3. **Feature Engineering:**
   * Identify and create relevant features from the data.
   * Transform, scale, or encode features as required.
   * Consider domain-specific knowledge for feature selection.
4. **Data Preprocessing:**
   * Split the data into training, validation, and test sets.
   * Normalize or standardize features if needed.
   * Encode categorical variables.
5. **Model Selection:**
   * Choose an appropriate machine learning algorithm for your problem.
   * Consider different models and techniques (e.g., regression, decision trees, neural networks) based on your data.
6. **Model Training:**
   * Train your chosen model using the training data.
   * Tune hyperparameters to optimize model performance.
7. **Model Evaluation:**
   * Evaluate the model using the validation dataset.
   * Choose evaluation metrics relevant to your problem (e.g., accuracy, precision, recall, F1-score, RMSE).
   * Consider cross-validation to assess model generalization.
8. **Model Fine-Tuning:**
   * Adjust the model or hyperparameters based on evaluation results.
   * Guard against overfitting or underfitting.
9. **Final Model Selection:**
   * Choose the best-performing model based on validation results.
10. **Model Testing:**
    * Assess the selected model's performance on the test dataset.
    * This helps ensure your model's generalization to unseen data.
11. **Deployment (if applicable):**
    * If your project involves deploying a model, integrate it into the production environment.
12. **Documentation:**
    * Document your process, including data sources, preprocessing steps, and model details.
13. **Reporting and Visualization:**
    * Communicate your findings and results through reports or presentations.
14. **Project Submission:**
    * If this is part of an assignment or competition, follow the specific submission instructions, which may involve uploading your code and documentation to a platform like GitHub.

Please provide more specific details about your project, and I can offer more targeted guidance on any of these steps.



#define BLYNK\_TEMPLATE\_ID "TMPL26V4fGv5q"

#define BLYNK\_TEMPLATE\_NAME "Test"

#define BLYNK\_AUTH\_TOKEN "XEHxNF\_Ur1Nt2p7wB5B20dNI1ZUwj34P"

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>

int duration1 = 0;

int distance1 = 0;

int duration2 = 0;

int distance2 = 0;

int dis1 = 0;

int dis2 = 0;

int dis\_new1 = 0;

int dis\_new2 = 0;

int entered = 0;

int left = 0;

int inside = 0;

#define LED 2

#define PIN\_TRIG1 15

#define PIN\_ECHO1 14

#define PIN\_TRIG2 13

#define PIN\_ECHO2 12

BlynkTimer timer;

char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "Wokwi-GUEST";   // your network SSID (name)

char pass[] = "";

#define BLYNK\_PRINT **Serial**

long get\_distance1() {

  // Start a new measurement:

  digitalWrite(PIN\_TRIG1, HIGH);

  delayMicroseconds(10);

  digitalWrite(PIN\_TRIG1, LOW);

  // Read the result:

  duration1 = pulseIn(PIN\_ECHO1, HIGH);

  distance1 = duration1 / 58;

  return distance1;

}

long get\_distance2() {

  // Start a new measurement:

  digitalWrite(PIN\_TRIG2, HIGH);

  delayMicroseconds(10);

  digitalWrite(PIN\_TRIG2, LOW);

  // Read the result:

  duration2 = pulseIn(PIN\_ECHO2, HIGH);

  distance2 = duration2 / 58;

  return distance2;

}

void myTimer() {

**Serial**.println("100");

  dis\_new1 = get\_distance1();

  dis\_new2 = get\_distance2();

  if (dis1 != dis\_new1 || dis2 != dis\_new2){

**Serial**.println("200");

    if (dis1 < dis2){

**Serial**.println("Enter loop");

      entered = entered + 1;

      inside = inside + 1;

      digitalWrite(LED, HIGH);

      Blynk.virtualWrite(V0, entered);

      Blynk.virtualWrite(V2, inside);

      dis1 = dis\_new1;

      delay(1000);

      digitalWrite(LED, LOW);

    }

    if (dis1 > dis2){

**Serial**.println("Leave loop");

      left = left + 1;

      inside = inside - 1;

      Blynk.virtualWrite(V1, left);

      Blynk.virtualWrite(V2, inside);

      dis2 = dis\_new2;

      delay(1000);

    }

  }

}

 void setup() {

**Serial**.begin(115200);

  pinMode(LED, OUTPUT);

  pinMode(PIN\_TRIG1, OUTPUT);

  pinMode(PIN\_ECHO1, INPUT);

  pinMode(PIN\_TRIG2, OUTPUT);

  pinMode(PIN\_ECHO2, INPUT);

  Blynk.begin(auth, ssid, pass, "blynk.cloud", 8080);

  timer.setInterval(1000L, myTimer);

}

void loop() {

  Blynk.run();

  timer.run();

}