```
#include <PDM.h> //to get microphone input
#include <arduinoFFT.h> //for the Fourier transform
#include "my_model.h"


const int  SAMPLES = 8192; //Must be a power of 2
#define SAMPLING_FREQUENCY 16000

const int downsample_factor = 2;

Eloquent::ML::Port::LogisticRegression model;

short sampleBuffer[SAMPLES];
volatile int samplesRead;
double vReal[SAMPLES/downsample_factor];
double vImag[SAMPLES/downsample_factor];

float features[SAMPLES/downsample_factor/2 + 1];

void onPDMdata(void);

arduinoFFT FFT = arduinoFFT();


void findMinMax(double arr[], int size, float &minVal, float &maxVal) {
  minVal = arr[0];
  maxVal = arr[0];
  for (int i = 1; i < size; i++) {
    if (arr[i] < minVal) {
      minVal = arr[i];
    }
    if (arr[i] > maxVal) {
      maxVal = arr[i];
    }
  }
}

// Function to normalize values using Min-Max Normalization
void minMaxNormalize(double arr[], int size) {
  float minVal, maxVal;
  findMinMax(arr, size, minVal, maxVal);

  for (int i = 0; i < size; i++) {
    arr[i] = (arr[i] - minVal) / (maxVal - minVal);
  }
}


void setup() {
  Serial.begin(115200);
```

```
  while (!Serial) {
    ; // wait for serial port to connect.
  }
  PDM.onReceive(onPDMdata);
  PDM.setBufferSize(SAMPLES);
  if (!PDM.begin(1, 16000)) {
    Serial.println("Failed to start PDM!");
    while (1);
  }
}

void onPDMdata()
{
  int bytesAvailable = PDM.available();
  PDM.read(sampleBuffer, bytesAvailable);
  samplesRead = bytesAvailable / 2;
}

void loop() {
  if (samplesRead) {

    for (int i = 0, j = 0; i < SAMPLES /downsample_factor; i++,j += downsample_factor) {
      vReal[i] = sampleBuffer[j];
      vImag[i] = 0;
    }


    FFT.Windowing(vReal, SAMPLES/downsample_factor, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
    FFT.Compute(vReal, vImag, SAMPLES/downsample_factor , FFT_FORWARD);
    FFT.ComplexToMagnitude(vReal, vImag, SAMPLES/downsample_factor );

    minMaxNormalize(vReal, SAMPLES / downsample_factor + 1);

    for (int i = 0; i < SAMPLES/downsample_factor /2; i++) { /// 2 + 1
      features[i] = vReal[i];
      if (1 == 2){
        Serial.print(vReal[i]);
        Serial.print(',');
      }

    }
    Serial.println();

    Serial.println(model.predict(features));

    samplesRead = 0;
  }
}
```