Code

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "sorting_hat_model.h"

// OLED Configuration
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Button Pins
#define BUTTON_A  15
#define BUTTON_B  2
#define BUTTON_C  18
#define BUTTON_D  19

// Sorting Hat Questions
const char* questions[] = {
   "1. What do you value?",
   "2. What to do if someone cheats?",
   "3. Favorite subject?",
   "4. How do you face challenges?",
   "5. How do friends describe you?",
   "6. What to do with a mystery book?",
   "7. Preferred pet?",
   "8. How do you solve problems?",
   "9. What kind of friends do you like?",
   "10. Dream career?"
};

// Answer Options
const char* options[][4] = {
   {"A) Bravery", "B) Loyalty", "C) Intelligence", "D) Ambition"},
   {"A) Call them out", "B) Let them be", "C) Inform teacher", "D) Gain from it"},
   {"A) Defense Arts", "B) Herbology", "C) Charms", "D) Potions"},
   {"A) Face head-on", "B) Team up", "C) Plan first", "D) Outsmart it"},
   {"A) Bold", "B) Kind", "C) Smart", "D) Resourceful"},
   {"A) Read it now", "B) Check safety", "C) Study it", "D) Use for gain"},
```

```
    {"A) Owl", "B) Toad", "C) Cat", "D) Phoenix"},
    {"A) Act fast", "B) Find a compromise", "C) Analyze first", "D) Outsmart"},
    {"A) Adventurous", "B) Loyal", "C) Thoughtful", "D) Powerful"},
    {"A) Auror", "B) Healer", "C) Scholar", "D) Minister"}
};

int responses[10] = {0};
int questionIndex = 0;  // Current question index

// ML Model
Eloquent::ML::Port::DecisionTree clf;

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

    // Initialize OLED
    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println("SSD1306 allocation failed");
        for (;;);
    }
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(10, 10);
    display.println("Sorting Hat Ready!");
    display.display();
    delay(2000);

    // Setup button pins
    pinMode(BUTTON_A, INPUT_PULLUP);
    pinMode(BUTTON_B, INPUT_PULLUP);
    pinMode(BUTTON_C, INPUT_PULLUP);
    pinMode(BUTTON_D, INPUT_PULLUP);

    showQuestion();
}

void loop() {
    checkButtons();
}

void showQuestion() {
```

```
    display.clearDisplay();
    display.setCursor(0, 0);
    display.println(questions[questionIndex]);

    for (int i = 0; i < 4; i++) {
        display.setCursor(10, 20 + (i * 10));
        display.println(options[questionIndex][i]);
    }

    display.display();
}

void checkButtons() {
    if (digitalRead(BUTTON_A) == LOW) {
        responses[questionIndex] = 1;
        waitForRelease(BUTTON_A);
        nextQuestion();
    } else if (digitalRead(BUTTON_B) == LOW) {
        responses[questionIndex] = 2;
        waitForRelease(BUTTON_B);
        nextQuestion();
    } else if (digitalRead(BUTTON_C) == LOW) {
        responses[questionIndex] = 3;
        waitForRelease(BUTTON_C);
        nextQuestion();
    } else if (digitalRead(BUTTON_D) == LOW) {
        responses[questionIndex] = 4;
        waitForRelease(BUTTON_D);
        nextQuestion();
    }
}

void waitForRelease(int buttonPin) {
    while (digitalRead(buttonPin) == LOW) {
        delay(10);
    }
}

void nextQuestion() {
    questionIndex++;
    if (questionIndex < 10) {
        showQuestion();
```

```
    } else {
        classifyHouse();
    }
}

void classifyHouse() {
    display.clearDisplay();
    display.setCursor(10, 10);
    display.println("Sorting...");

    float features[] = {
        (float)responses[0], (float)responses[1], (float)responses[2],
(float)responses[3],
        (float)responses[4], (float)responses[5], (float)responses[6],
(float)responses[7],
        (float)responses[8], (float)responses[9]
    };

    int house = clf.predict(features);

    display.setCursor(10, 30);
    display.print("House: ");
    switch (house) {
        case 0: display.println("Gryffindor"); break;
        case 1: display.println("Hufflepuff"); break;
        case 2: display.println("Ravenclaw"); break;
        case 3: display.println("Slytherin"); break;
        default: display.println("Unknown"); break;
    }
    display.display();

    Serial.println("Sorting complete!");
    Serial.print("Predicted House: ");
    switch (house) {
        case 0: Serial.println("Gryffindor"); break;
        case 1: Serial.println("Hufflepuff"); break;
        case 2: Serial.println("Ravenclaw"); break;
        case 3: Serial.println("Slytherin"); break;
        default: Serial.println("Unknown"); break;
    }
}
```

Discussion

After playing around with the sorting hat, I realized that not all 10 questions are equally useful. Some of them definitely help determine the house better, but others feel a bit redundant or too random.

If I had to remove a couple to make the experience smoother, I'd probably take out:

Q6: "What to do with a mystery book?" — it's interesting, but kind of vague, and I noticed most people just guessed on it.

Q7: "Preferred pet?" — it's fun, but doesn't really say much about someone's personality or values.

By removing these, the quiz would be faster and probably more engaging, especially for casual users, without losing too much accuracy.

How could I improve the model?

To improve accuracy, I'd like to collect more data from different users — right now we only had around 30 samples, which isn't a lot. I could also play with the max_depth of the decision tree to avoid overfitting. If I had more time, maybe even try other lightweight models like random forests or neural nets with TinyML.

What extra hardware would make this cooler?

I think it would be really fun to add:

- A microphone, so people can speak their answers.
- A touchscreen or capacitive touch buttons instead of push buttons — more modern and easier to use.
- Maybe a small LED ring or vibration motor to give feedback when an answer is selected or the house is revealed.

That way, it wouldn't just feel like answering a quiz — it'd feel like an actual magical interaction.

Is a decision tree still the best choice?

For now, yes — decision trees are great because they're fast and small, which works well with the ESP32. But if I added more complex sensors like audio input or movement, then I'd probably switch to a neural network or random forest since they can handle noisy data better.