**Machine Learning on ESP 8266 with Scikit-Learn and micromlgen.**

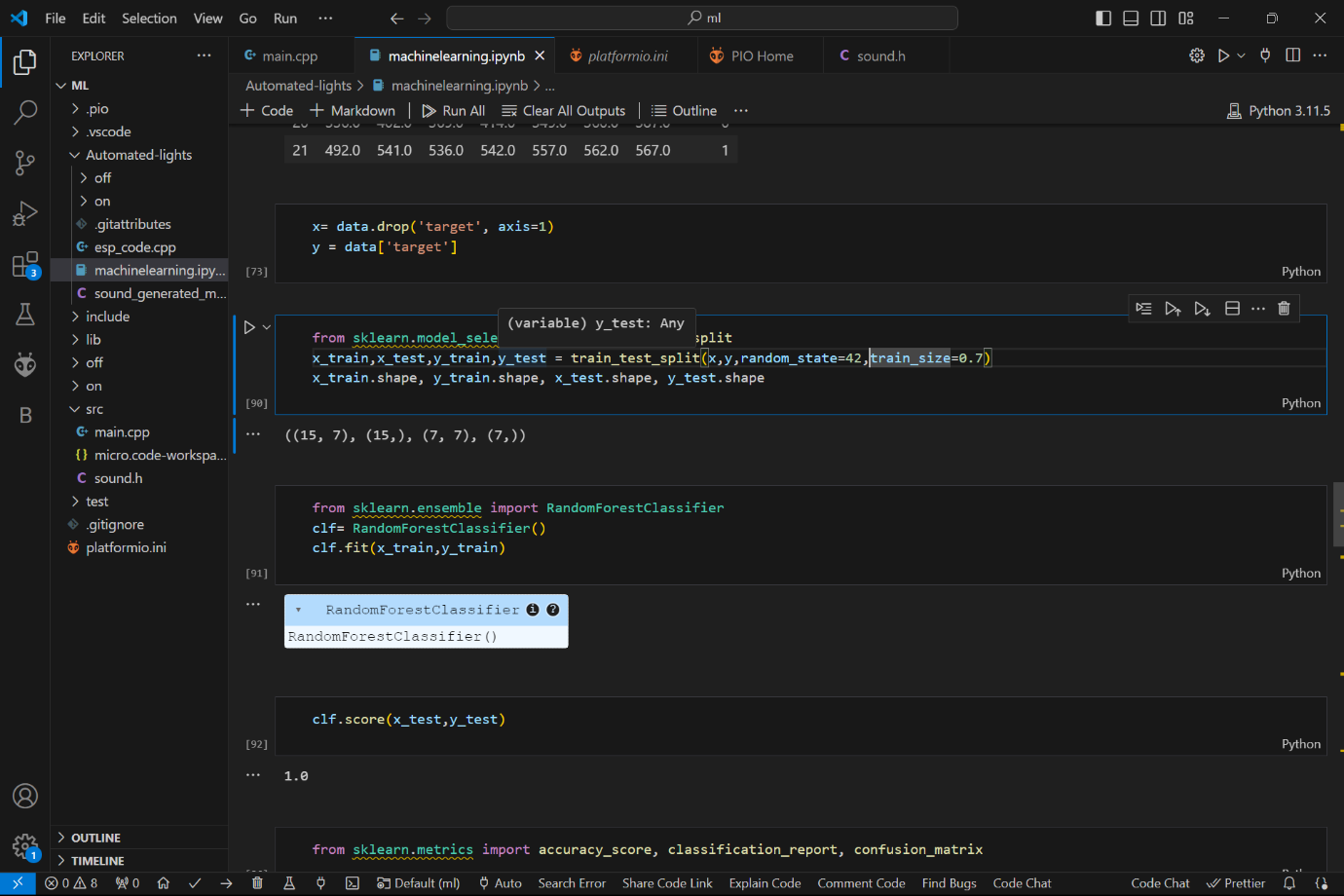
In the ever-evolving field of IoT (Internet of Things), integrating machine learning capabilities into microcontrollers has opened exciting new possibilities. Recently, I had the opportunity to perform machine learning on an ESP 8266 microcontroller, a popular and cost-effective module known for its Wi-Fi capabilities. In this post, I'll walk you through the process of training a model using scikit-learn in Python and then converting it to C code using micromlgen for deployment on the ESP 8266.

**Why Machine Learning on Microcontrollers?**

Microcontrollers like the ESP 8266 are limited in resources but widely used in various IoT applications due to their low power consumption (3.3V), low cost and efficiency. Adding machine learning capabilities to these devices can enable real-time decision-making at the edge, reducing latency and the need for constant connectivity to a central server. This improves data security as no external server is involved.

**Step 1: Training the Model with Scikit-Learn**

First, we need to train our machine learning model. The data will be collected from ESP 8266 using excel data streamer. Initially this data is read from an analog microphone which converts sound to analog signal. In this case it was between **ON** and **OFF** sound signals. Before training the model one needs to perform exploratory data analysis (**EDA)** as this prepares input for the model. There after training the model using Scikit-learn a module called RandonForestClassifier. I chose this model since only classifies categorical data i.e., between on and off.



from micromlgen import port

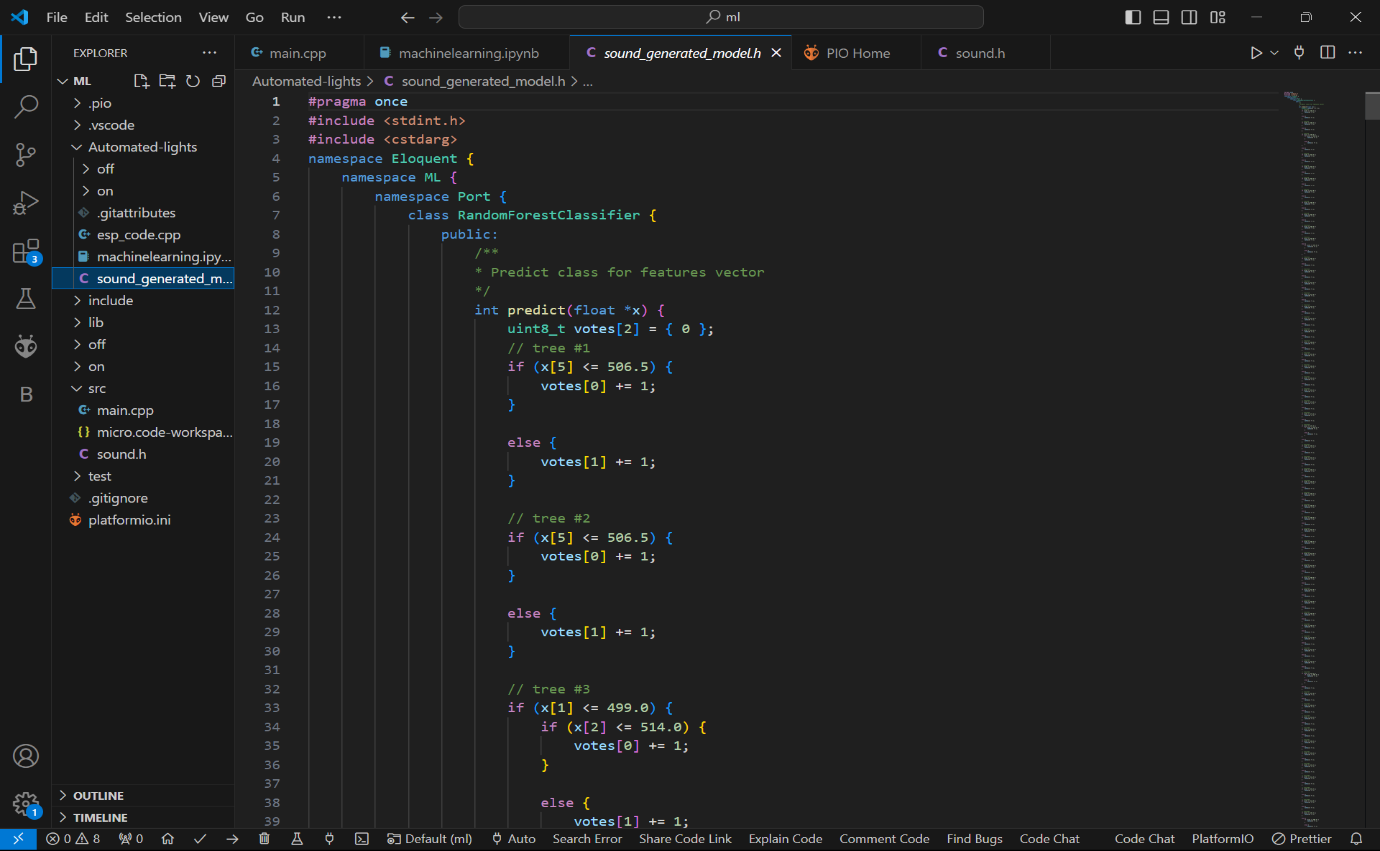
c=port (clf=clf, classname='RandomForestClassifier')

print(c)

### Step 2: Converting the Model to C Code with micromlgen.

Once the model is trained, we can convert it to C code using micromlgen, a library that helps bridge the gap between Python-trained models and embedded C code. Micromlgen is installed by single command: **pip install micromlgen.** Micromlgen simplifies the process of converting the model into a format that can be embedded in microcontroller firmware. This is crucial for deploying machine learning models on devices with limited resources.

Printing **C** will output a similar model but now in C-language with the ESP 8266 can interpret. Therefore, one is required to create a file named it and paste contents from output of print(c) on it. The file is named with .h extension for the compiler to comprehend it.



As you can see sound.h file and the generated model by micromlgen.

### Step 3: Deploying the Model on ESP 8266

### Now that we have our model in C-language, we can comfortably use it on ESP 8266. But before use, one needs to install Eloquent library which helps to integrate the model on to the board. The model is then included in #include “sound.h” and the initialized.

Eloquent::ML::Port::RandomForestClassifier clf;

### clf is used to initialize the model, then use .predict object to predict the values provided. One speaks to the microphone then analog values are collected then stored in an array then fed into the model which will predict whether it is ON or OFF. The predicted outcome will then be applied to turn on lights, devices or doors. Relays, MOSFETs and transistors will be enable control of high voltage from 3.3V ESP 8266.

### 

**Best Practices and Tips**

1. **Optimize Your Model**: Given the resource constraints of microcontrollers, consider using lightweight models or techniques such as pruning and quantization.
2. **Memory Management**: Be mindful of memory usage. The ESP 8266 has limited RAM and flash storage, so ensure your model fits within these limits.
3. **Power Consumption**: If your device operates on battery power, consider the power consumption implications of continuous inference, and optimize accordingly.

**Conclusion**

By following these steps, you can leverage the power of machine learning on an ESP 8266 microcontroller. This process demonstrates how accessible and powerful embedded machine learning has become, enabling smart and autonomous IoT devices. Whether it's for home automation, environmental monitoring, or any other application, the combination of scikit-learn and micromlgen makes it possible to deploy sophisticated models on the simple board.

<https://github.com/Mark-Kitur/Automated-lights>

[kiturmark@gmail.com](mailto:kiturmark@gmail.com)