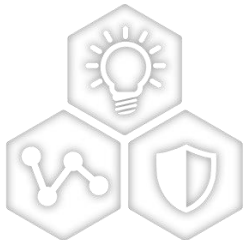


MPLAB Machine Learning Development Suite AIML1001 v1.1



A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

CAE Taiwan
Microchip Technology Inc.
2025 March

Class Objectives

- **Before this class:**
 - Basic C programming language design skills
- **When you finish this course, you will be able to:**
 - The basic information to implement AI/ML at the edge.
 - Be familiar with MPLAB Machine Learning Model Builder.
 - Enable to configure and deploy model on small processors.
- **You won't be able to learn:**
 - Making you an Artificial Intelligence/ML expert.
 - Build a Large language model.

Agenda (1/2)

- Software Prerequisite and Installation
- Hardware Prerequisite
- What is Artificial Intelligence ?
- Why Edge AI ?
- MPLAB® Data Visualizer
 - 🔧 Lab0 Data Visualizer for 6DOF Sensor data
- Set Goal and Problem Identification
 - 🔧 Lab1 Initial Setup
- Data Logging and Upload
 - 🔧 Lab1 Data Collecting
- Labeling and Prepare Data
- MPLAB® Machine Learning Development Suite
 - 🔧 Lab2 Data Labeling and Preparation

Agenda (2/2)

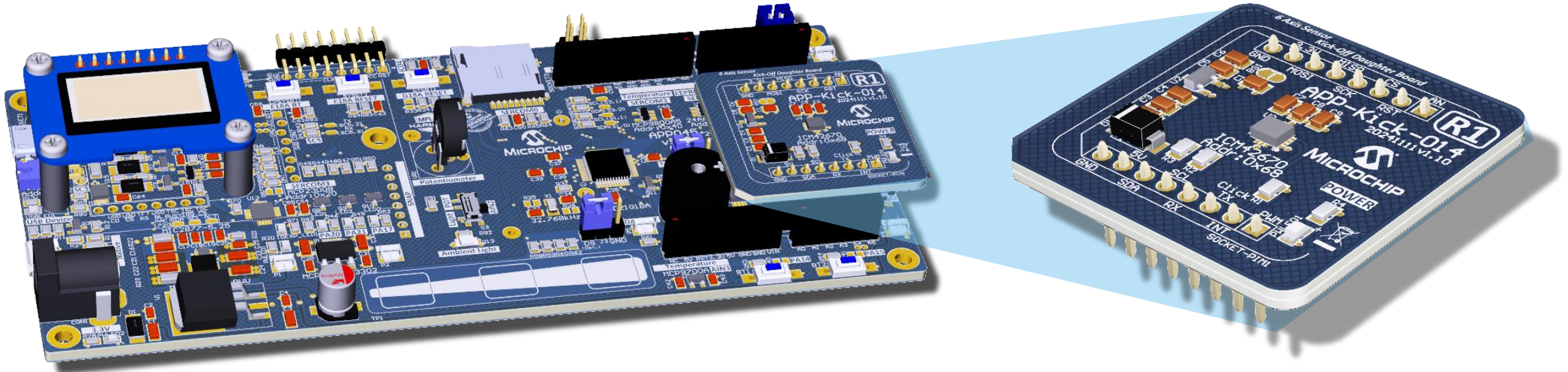
- Model Pipeline and Training
 - 🔧 Lab3 Model Pipeline
 - 🔧 Lab3 Training
- Model Evaluation and Deploy
 - 🔧 Lab4 Model Evaluation
 - 🔧 Lab4 Model Deploy
- Knowledge Pack, Main Flow, and State Machines
- Summary
- Appendix
 - Install the ML Data Collector & ML Model Builder
 - Create and Login MPLAB ML Account
 - Porting Deploy Model to Custom Project
 - JSON Decoder helper Library
 - Large Language Models

Software Prerequisite and Installation

- **MPLAB® X IDE v6.20** <https://www.microchip.com/mplab/mplab-x-ide>
- **MPLAB® XC32 v4.45** <https://www.microchip.com/mplab/compilers>
- **SAMD21 DFP v3.6.144** (Device Family Pack download from X IDE Tools Packs)
- **CMSIS v6.1.0** (Common Microcontroller Software Interface)
- **MPLAB® Data Visualizer v1.3.1677**
(Embedded with MPLAB X IDE v6.20, upgrade is necessary)
- **MPLAB® ML Model Builder v1.1** ([Customer install in Plugin of MPLAB X IDE](#))
- **MPLAB® ML Data Collector v1.2** ([Customer install in Plugin of MPLAB X IDE](#))
- **Apply “myMicrochip” account** ([Register in ML Model Builder after installed](#))

Hardware Prerequisite

- APP045 v5.0 R2 EV Board with SAMD21
- APP-Kick-014 the Accelerometer and Gyroscope (6DOF) daughter board
- Micro USB cable x 1
- On-Board Debugger (APP045 v5.0 R2 available)



What is Artificial Intelligence ?

MUSIC COMPOSED AND CONDUCTED BY ALEXANDRE DESPLAT

THE IMITATION GAME



1950 Turing Test



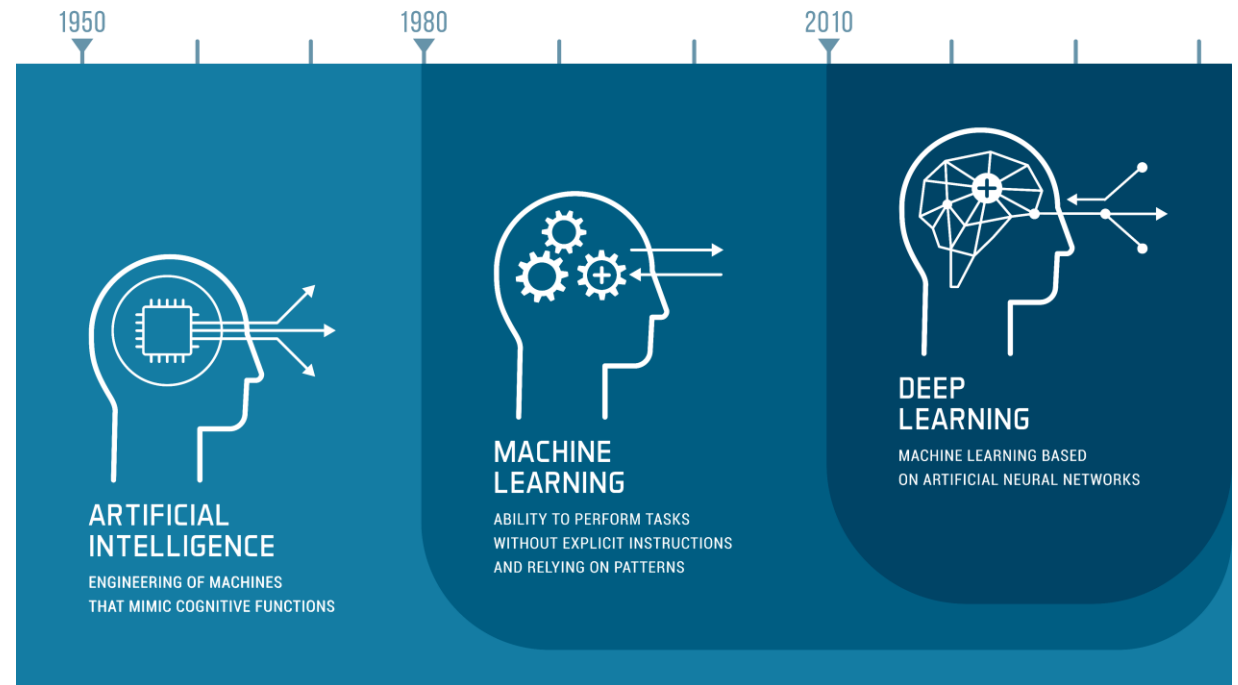
2016 AlphaGo

2022 ChatGPT



AI, ML and DL

- **AI (Artificial Intelligence):** technology that enables computers and machines to simulate **human intelligence** and **problem-solving** capabilities.
- **ML (Machine Learning):** a type of artificial intelligence (AI) focused on building **computer systems that learn from data**.
- **DL (Deep Learning):** a subset of machine learning that uses multi-layered **neural networks**, called deep neural networks, to simulate the complex decision-making power of the human brain.



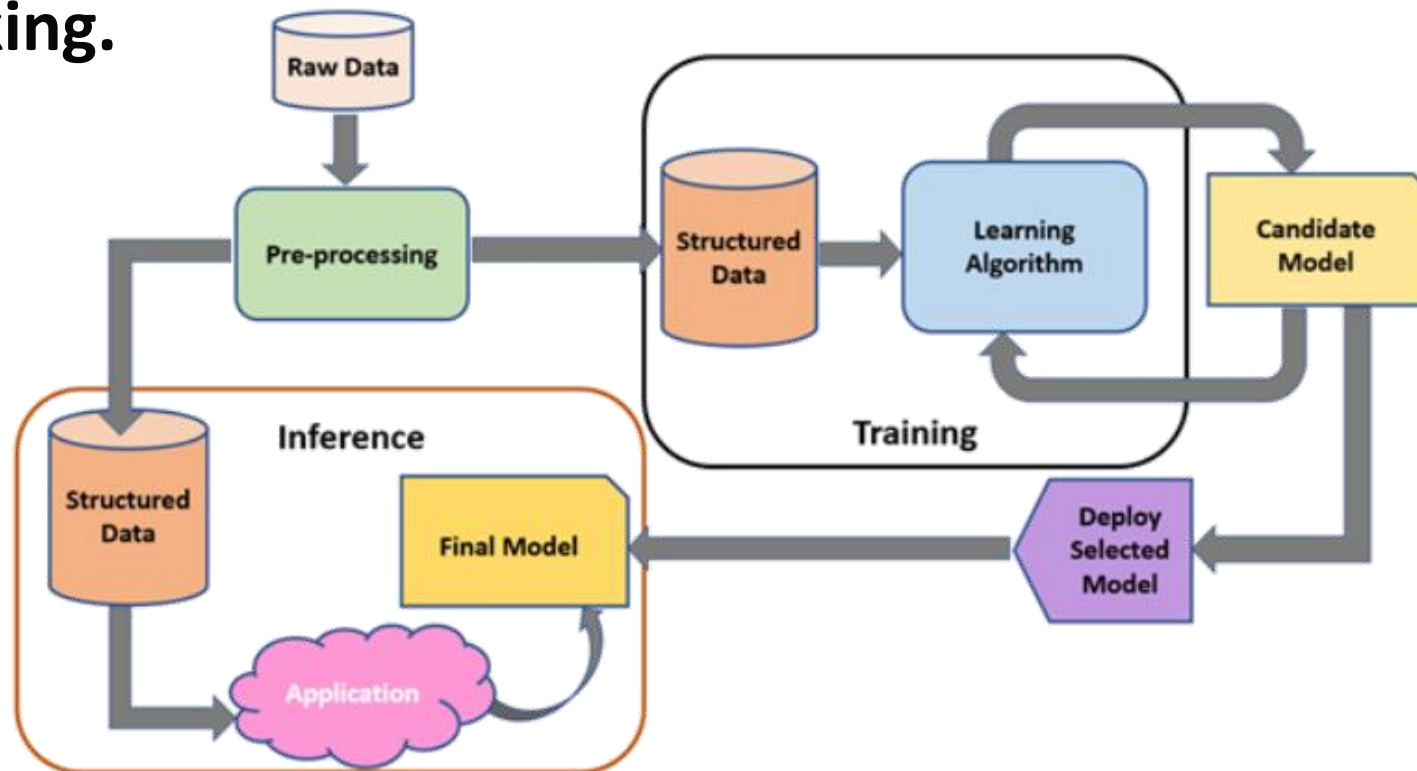
Why is it so popular now?

- **Advancements in computational power**
 - Vastly parallel operations
 - Virtual computing and Data Center
- **Large Datasets**
 - Example data
- **Algorithms**
 - New algorithms
 - Accessibility
- **Automation**
 - 'Intelligence' and automation
 - Sustainability



Machine Learning

- Machine learning (ML) is a technique of using computers to predict things based on past observations.
- A set of algorithmic methods that discovers patterns from seemingly unrelated data, providing you with important information to facilitate decision making.



4 kinds of Machine Learning

Supervised Learning

Give label and data

Give the standard answer while training.

Un-supervised Learning

Unlabeled data

Provide examples. Let computer find the underlying rules in the examples.

Semi-supervised Learning

Small amount of labeled data
+
Large amount of unlabeled data

Reinforcement Learning

Unlabeled data

Tell computer whether the steps are correct, the machine will modify by environment and experience.

Supervised Learning

Let's take memorizing English words as an example....

Supervised Learning

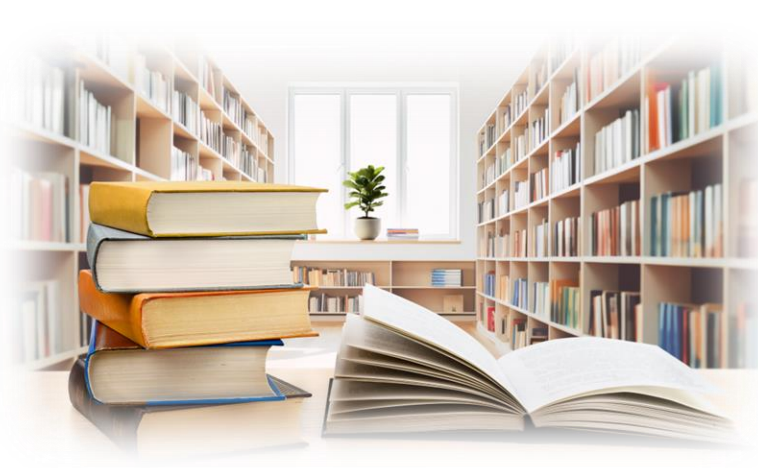
By Flash Cards



Have clear answers and feedback.

Un-supervised Learning

By reading lots of book



There is no clear answer, rely on your own discovery and understanding.

Semi-supervised Learning

By Prefix, Root, and Suffix

-view 觀看



Re-view

Pre-view

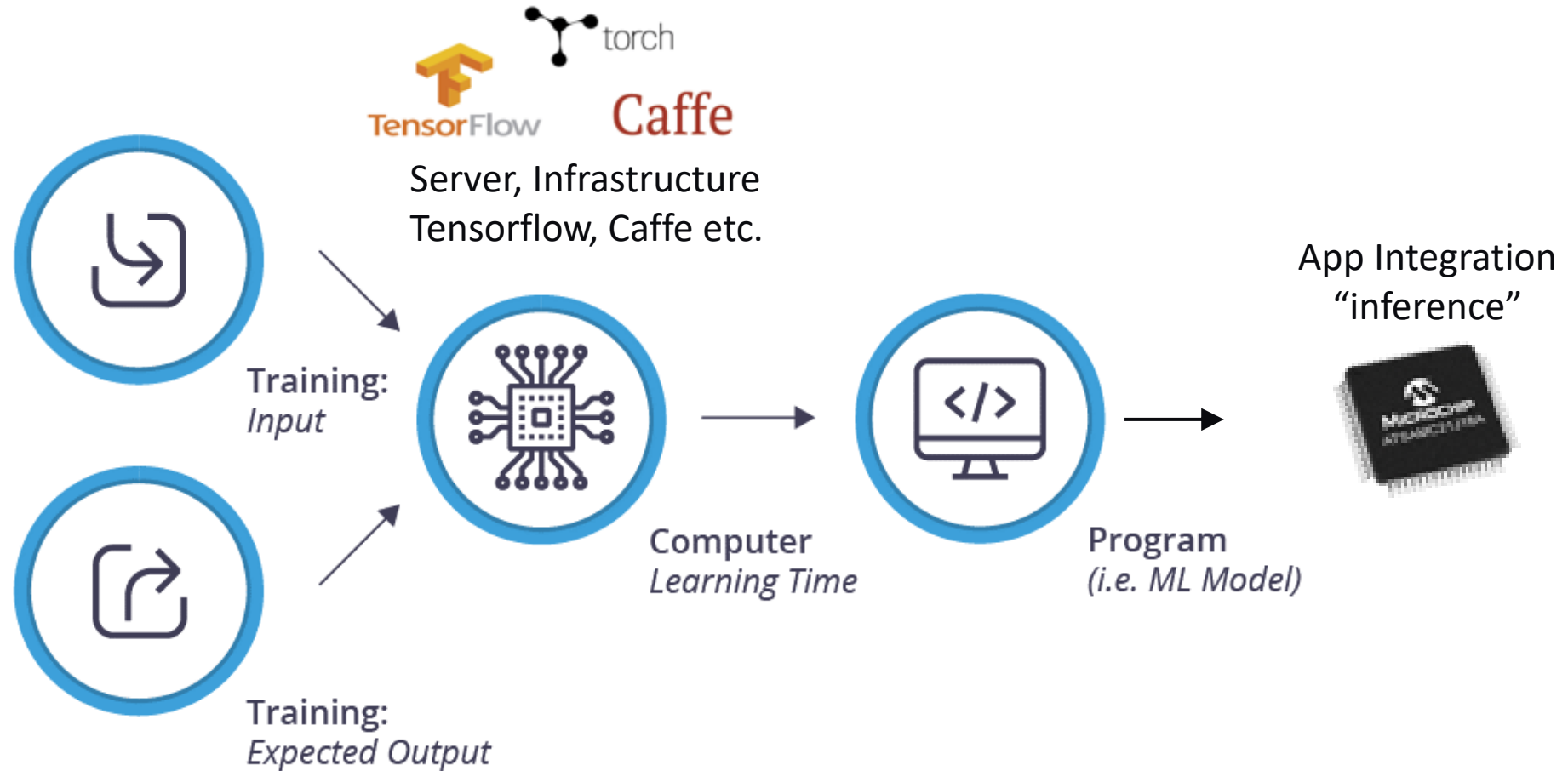
Over-view

Inter-view

複習
預習
概觀
面談

Combines known information with new learning.

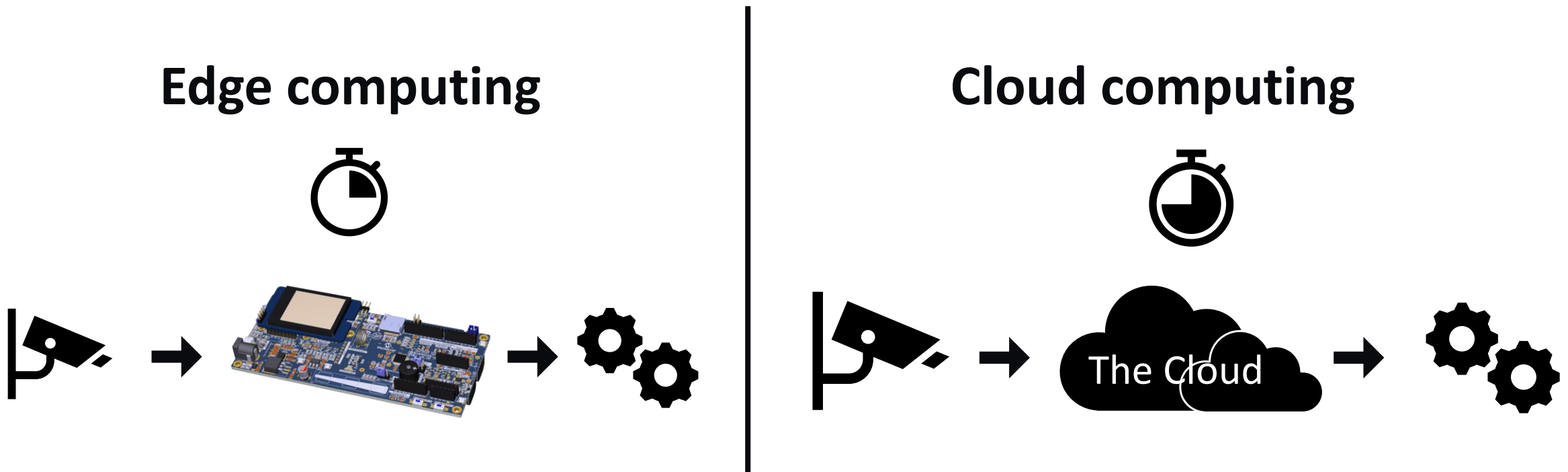
How does Supervised Learning work?



Why Edge AI ?

Differences between Edge computing & Cloud computing

- **Edge Computing:** The data is calculated directly on the **local side**. It can overcome the problem of data transmission delay and facilitate instant judgment and decision-making.
- **Cloud Computing:** Cloud computing is the delivery of computing services over the internet (the cloud).



Why AI at the Edge?

- **Edge AI:** Execute the application of artificial intelligence algorithms on edge computing nodes such as embedded devices and sensors.
- **Edge is the location of Data Processing near the source of Data Generation**



Privacy and Security

Data stays on the device, gets processed locally.



Reduce Power Consumption

Stay operational for longer periods of time.



Real Time Operation

Process data real-time on the edge device, without having to wait for a response back from the cloud.

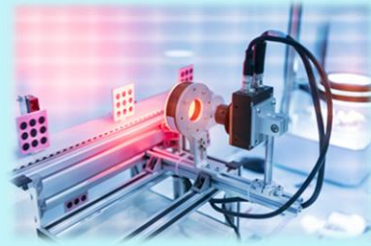


Reduce Cost

Save on compute costs, storage, and less communication.

ML Edge Applications

Smart Embedded Vision



Enhanced quality and productivity at factories

Machine assistance for faster, better



Surveillance / security with granular, real-time monitoring and response

Smart Human Machine Interface (HMI)



Local processing for voice interactive appliances



Real time, interactive gesture recognition

Smart Predictive Maintenance



Detect ageing / environment driven degradation



Predict and prevent system failures

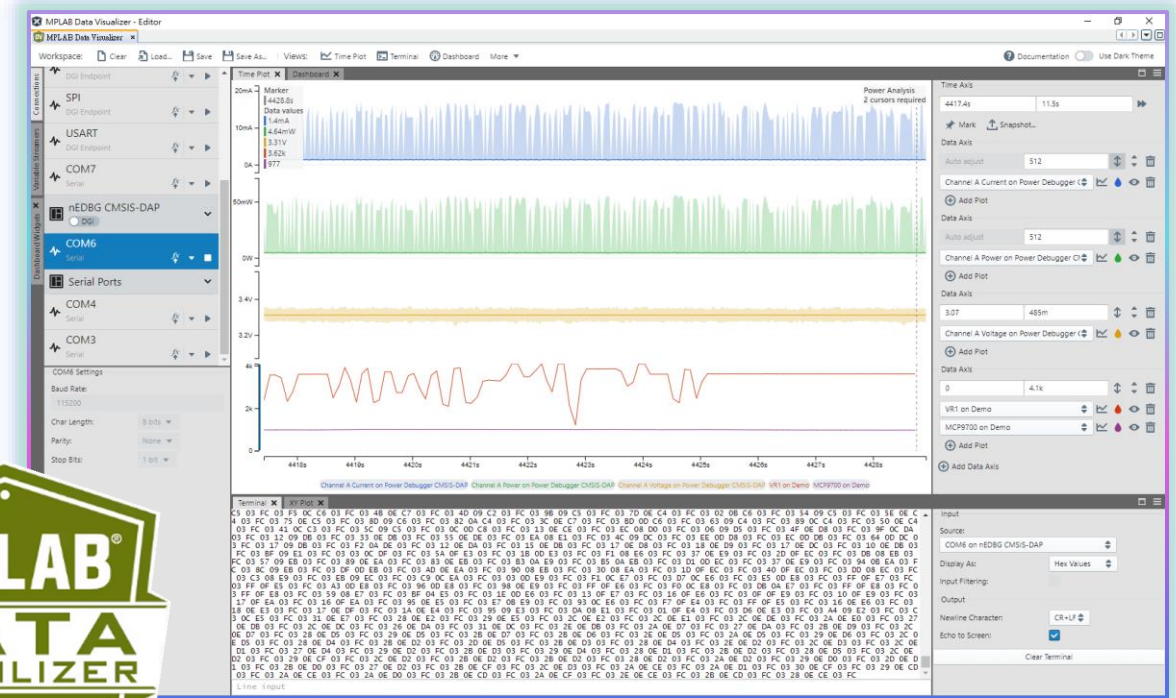


Water / hazardous leak detection and alarm

MPLAB® Data Visualizer

MPLAB® Data Visualizer

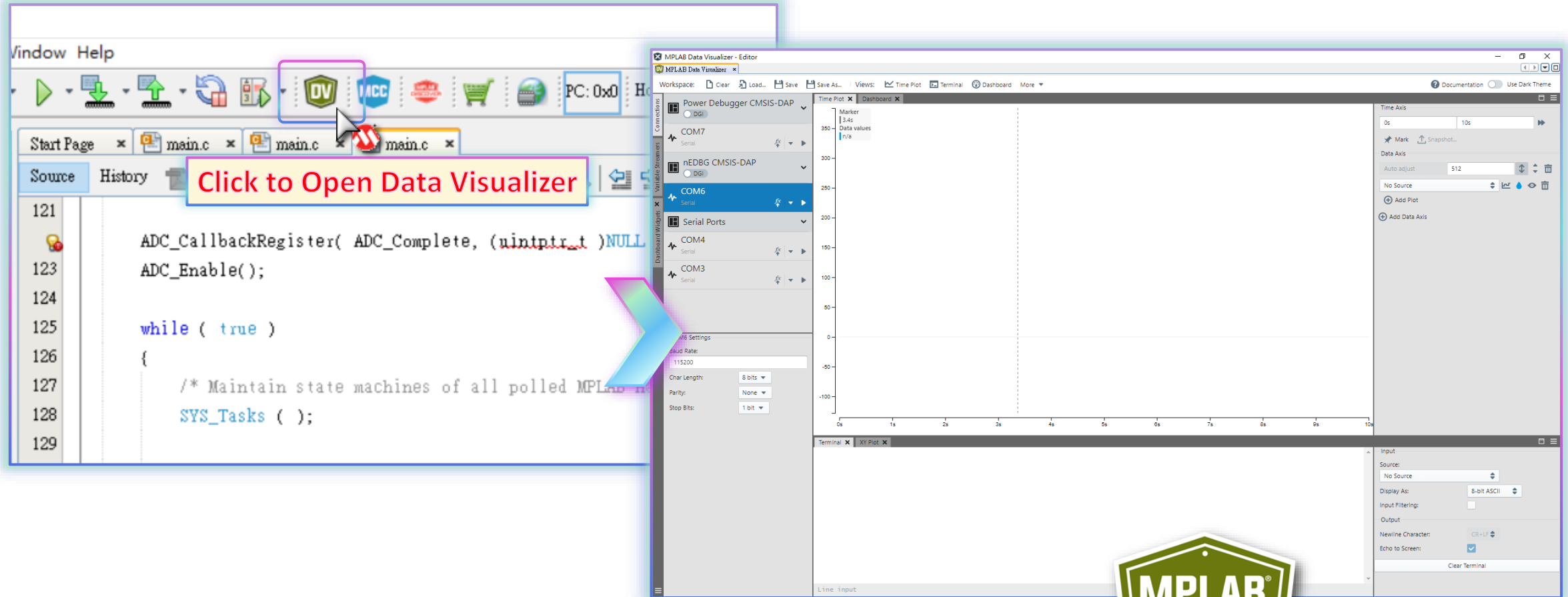
- MPLAB Data Visualizer is a program used for processing and visualizing data.
- It can receive data from various sources such as the DGI (Data gateway Interface) and COM ports.
- And has a graph plotter and an oscilloscope, a terminal and a configurable dashboard with buttons, sliders, and various indicators. It can decode protocols and log data to file.
- It's an MPLAB X IDE plugin and cross platform standalone application. It's installed at X IDE right. don't need do anything before use.



[View Data Stream format for DV](#)

MPLAB® Data Visualizer

- To open MPLAB Data Visualizer. You can click icon  in MPLAB X IDE.

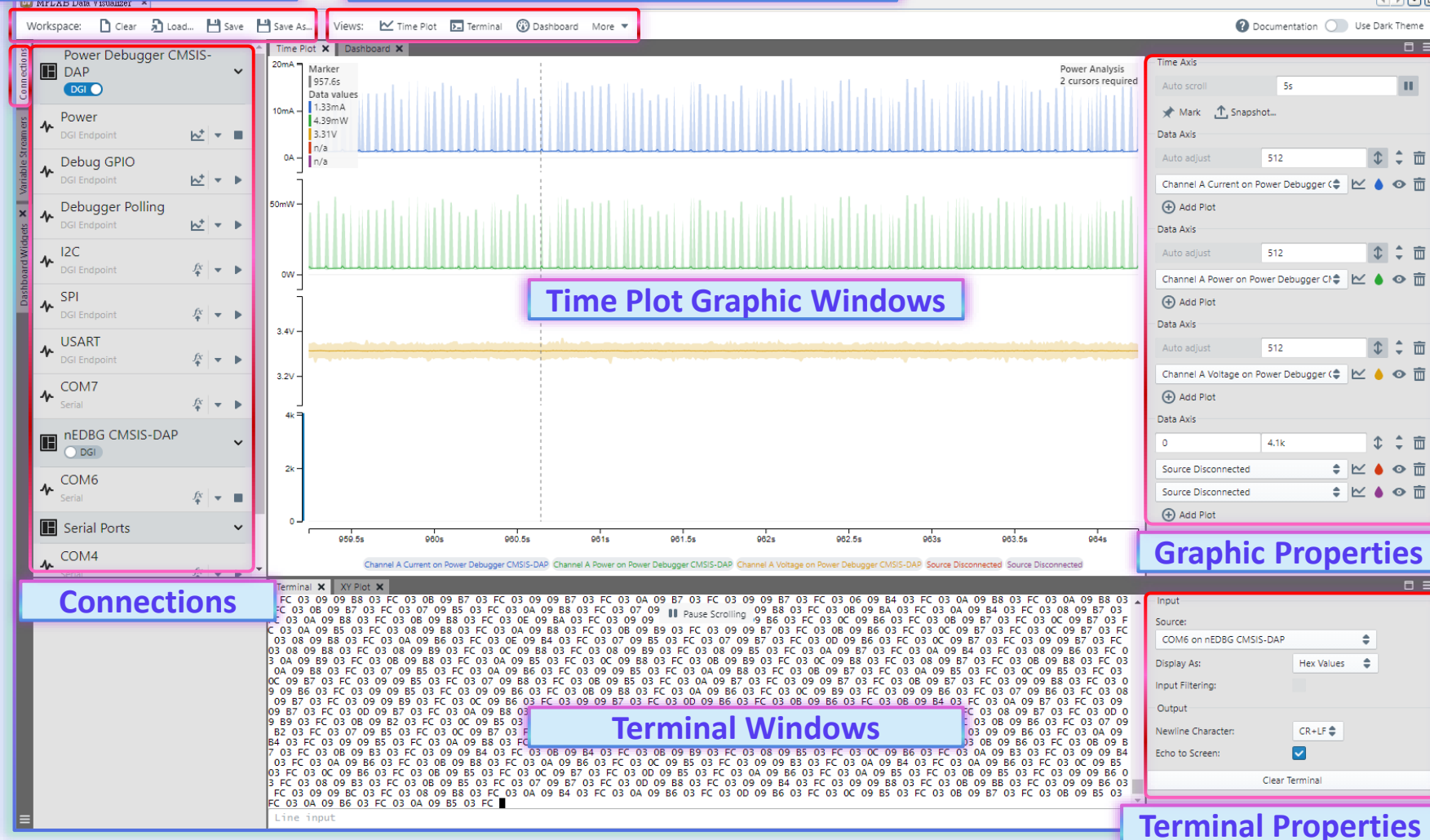


MPLAB® Data Visualizer

Workspace of MPLAB Data Visualizer

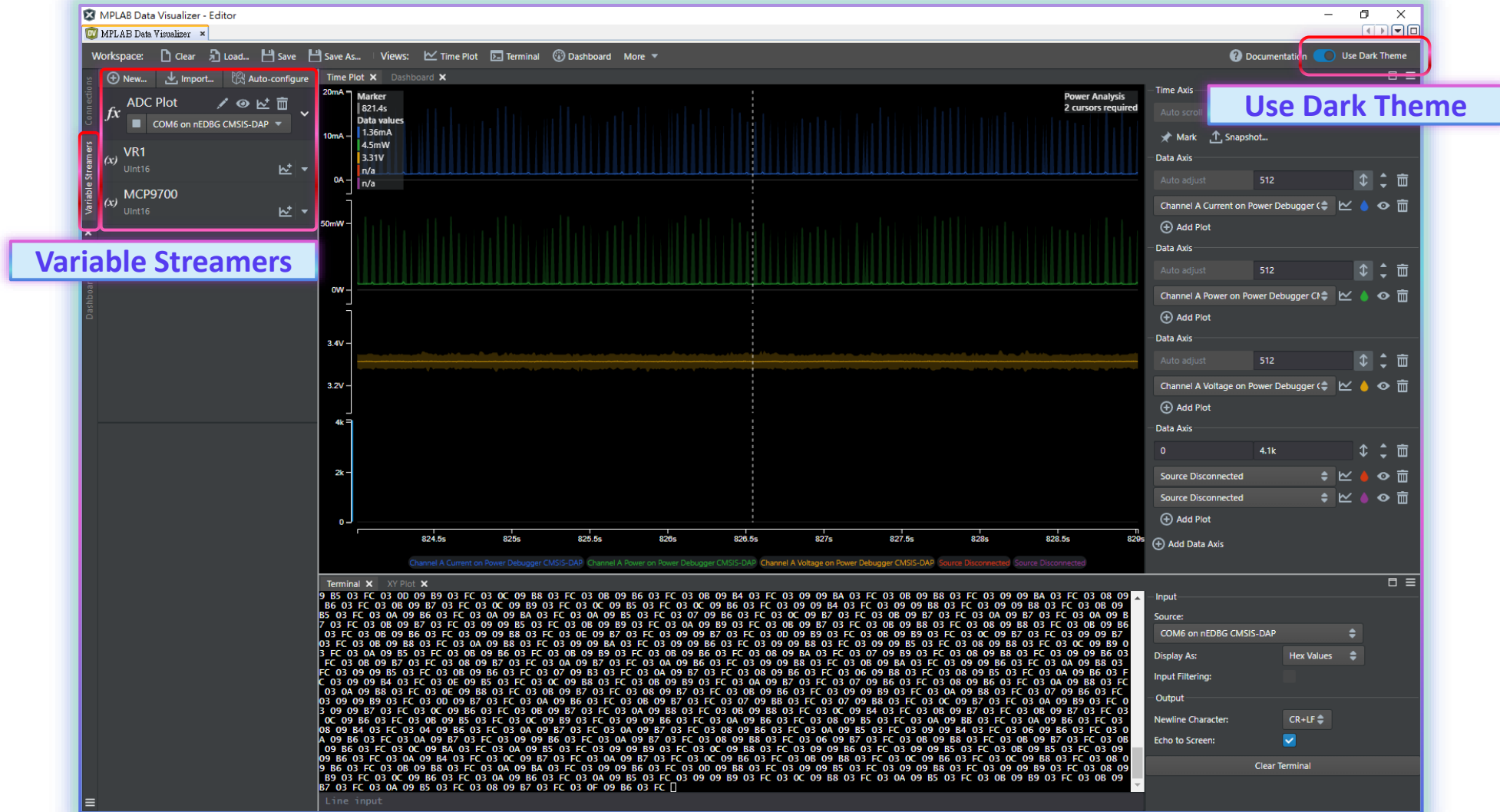
Workspace Clear / Load / Save

Show Plot / Terminal / Dashboard ...



MPLAB® Data Visualizer

Variable Streamers and Dark Theme of MPLAB Data Visualizer

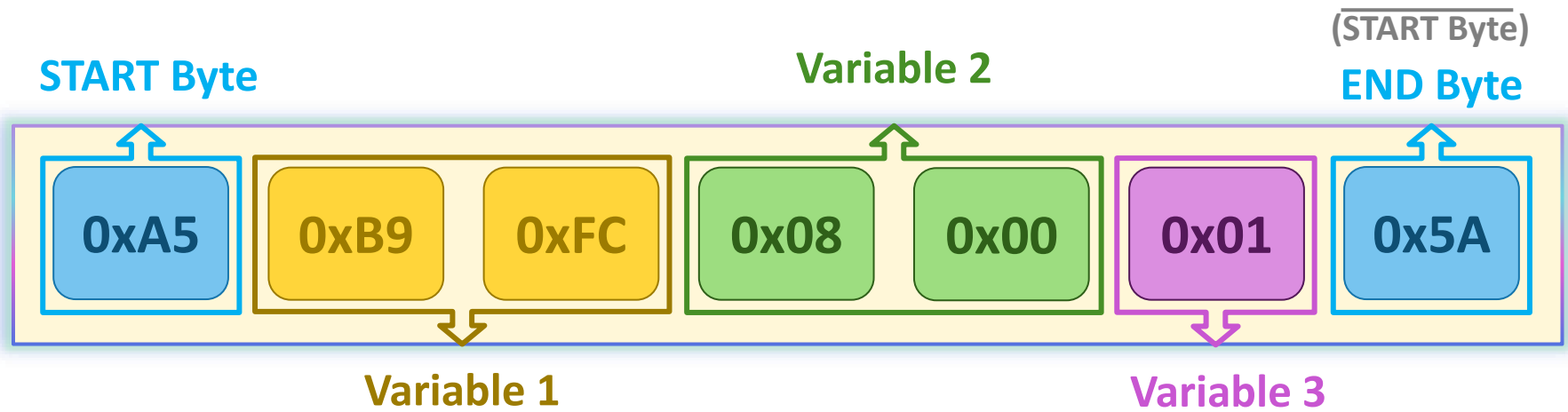


MPLAB[®] Data Visualizer

Data Stream Format for DV

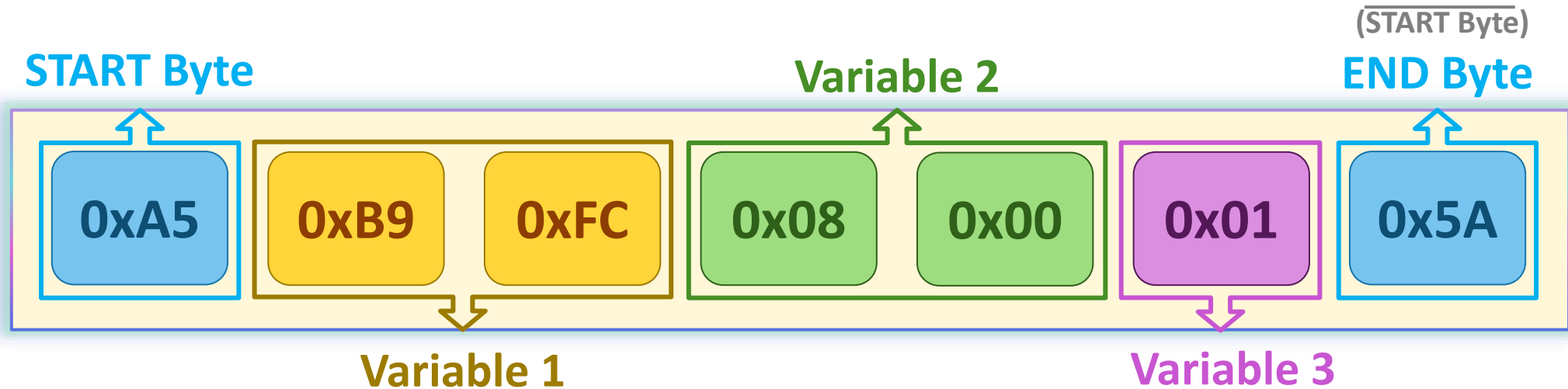
- The Data Stream Format is processed in the same order as the variables specified in the Variable Streamer.
 - Data must be given as little-endian values.
 - The start byte can be arbitrary, but the end byte must be the inverse of the Start byte.

Type	Size (Bytes)
int8	1
int16	2
int32	4
uint8	1
uint16	2
uint32	4
float32	4
float64	8



MPLAB[®] Data Visualizer

Data Stream Format for DV

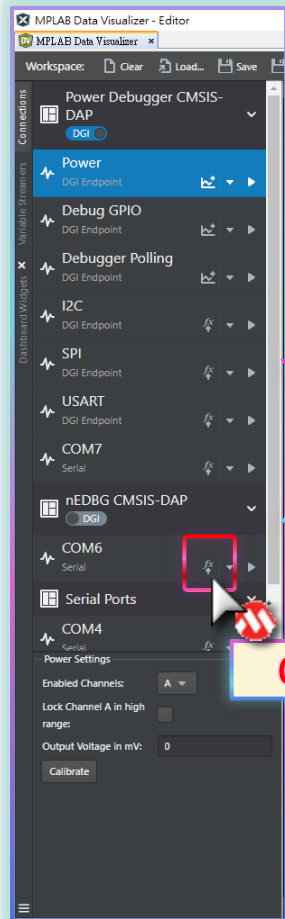


```
printf(“%c%c%c%c%c%c%c”,  
       0xA5,  
       variable[0] & 0xff, (variable[0]>>8) & 0xff,  
       variable[1] & 0xff, (variable[1]>>8) & 0xff,  
       BT1_Get() & 0x01,  
       ~(0xA5));
```

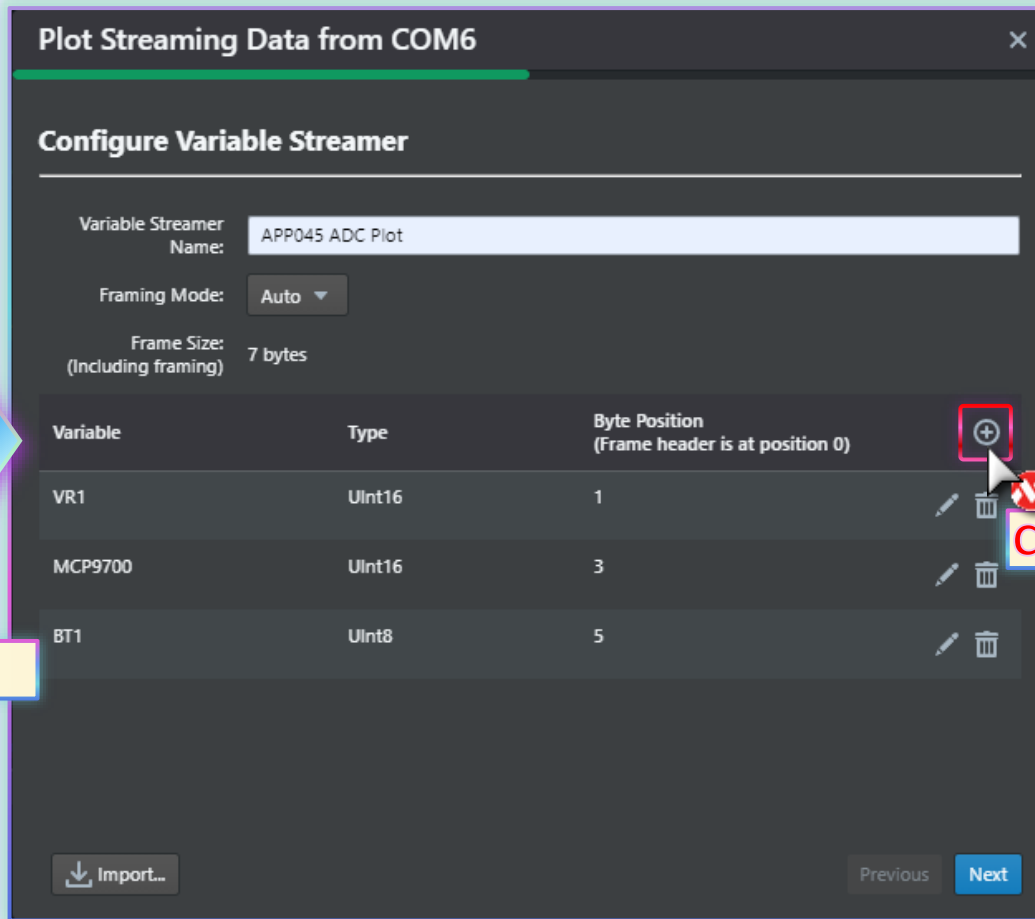
MPLAB® Data Visualizer

Data Streamer Setting

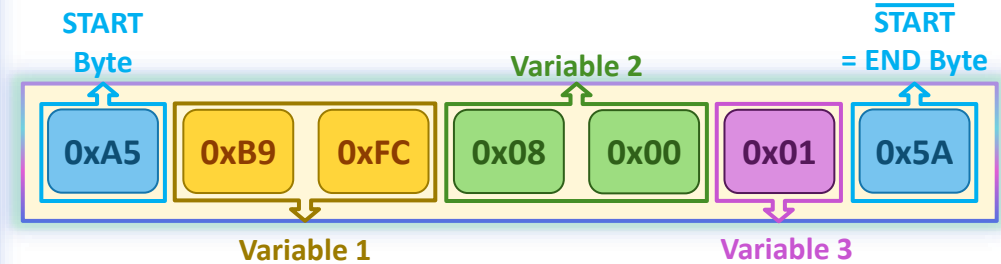
- The correspond format of Data Streamer must be setting at MPLAB Data Visualizer.



Click



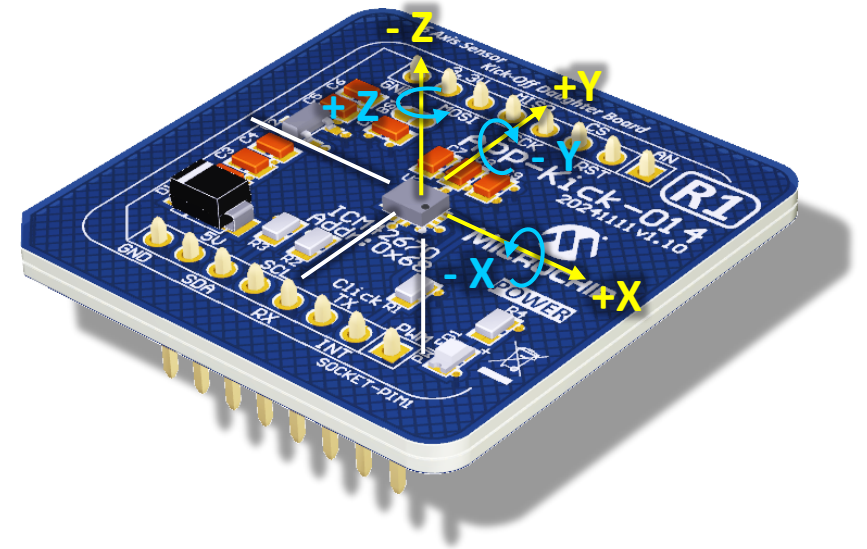
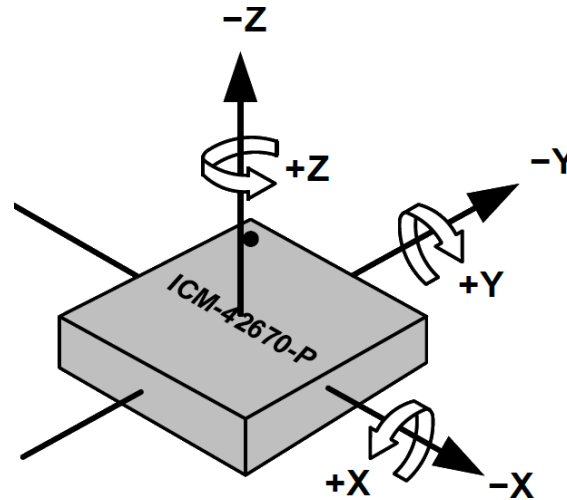
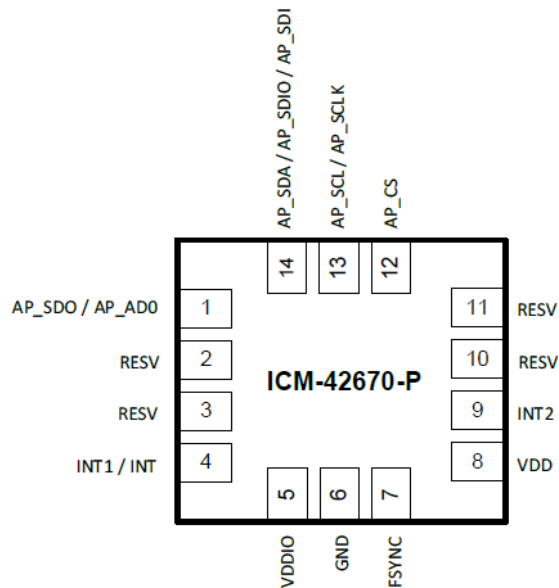
Click to add Variable



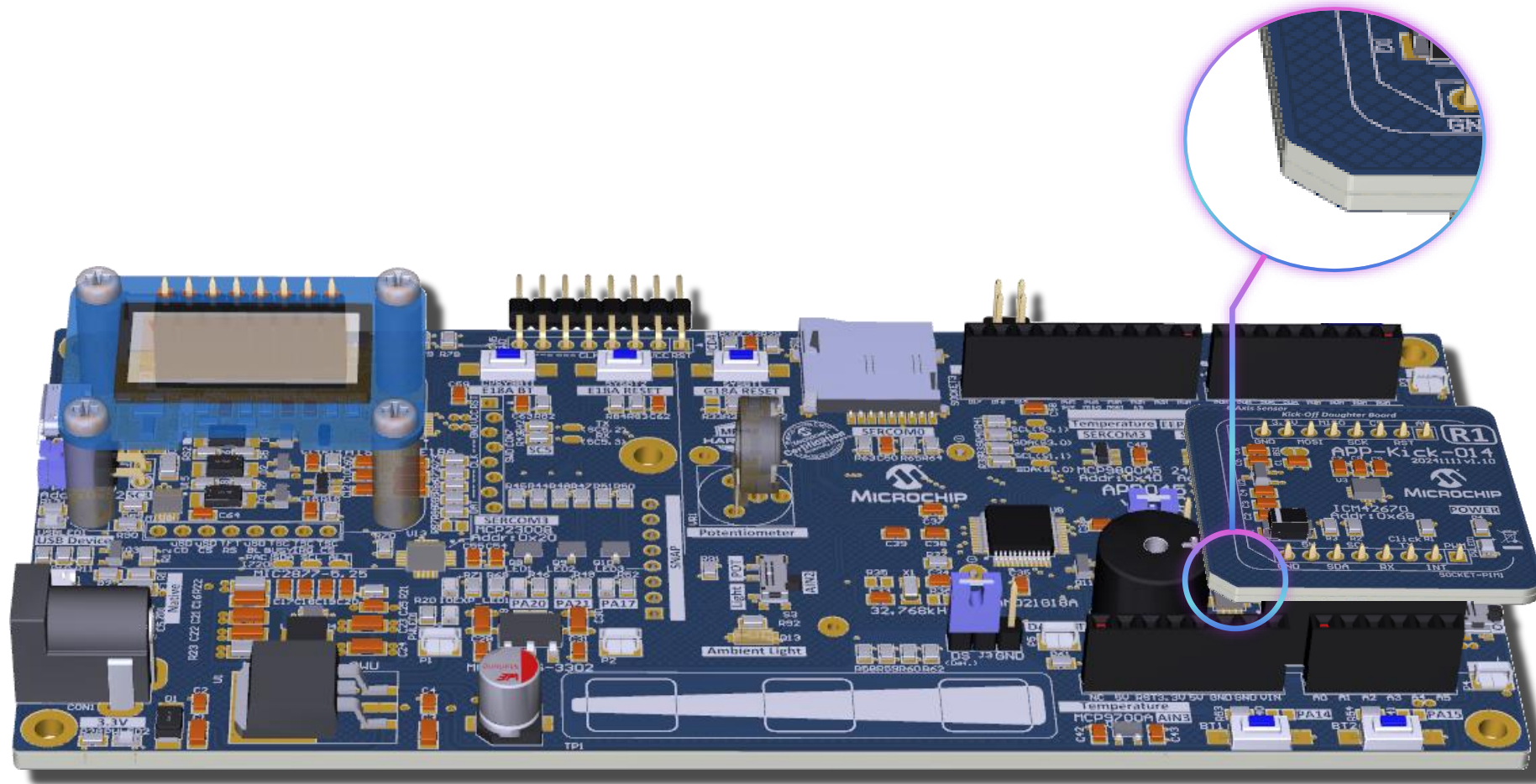
```
printf("%c%c%c%c%c%c",  
      0xA5,  
      variable[0] & 0xff, (variable[0]>>8) & 0xff,  
      variable[1] & 0xff, (variable[1]>>8) & 0xff,  
      BT1_Get() & 0x01,  
      ~(0xA5));
```


APP-Kick-014 6DOF Motion Sensor Board (TDK ICM-42670P)

- The **APP-Kick-014** is design for **6DOF** (Degree Of Freedom) Motion Application.
- The **TDK ICM-42670-P** is a high performance 6-axis MEMS MotionTracking™ device that combines a **3-axis gyroscope** and a **3-axis accelerometer**.
- It has a configurable host interface that supports I3CSM, I2C, and SPI serial communication, features up to 2.25 Kbytes FIFO and 2 programmable interrupts with ultra-low-power wake-on-motion support to minimize system power consumption.



APP045 and APP-Kick-014 Assembly



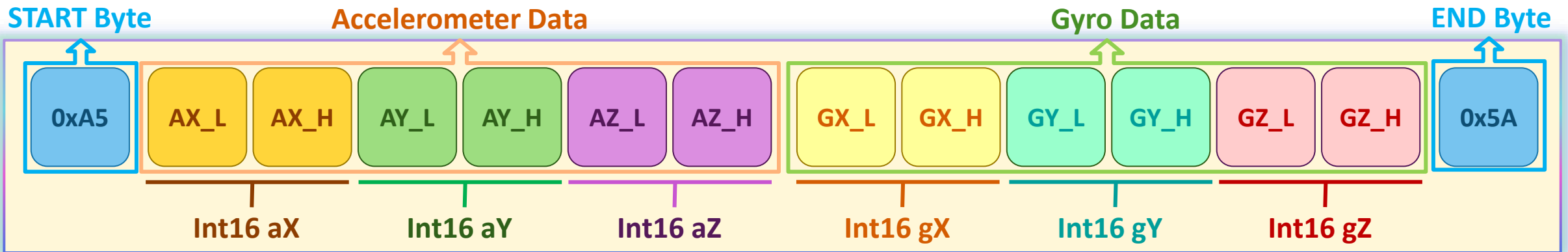
 Lab 0

Data Visualizer for 6DOF Sensor data

Boxing Recognition by Machine Learning

🔧 Lab0 Data Visualizer for 6DOF Sensor data

- Try using MPLAB Data Visualizer to view Gyro and Accelerometer data from the 6-axis motion sensor ICM-42670P.
- You will need to disable the UART messaging you did in the previous Lab and add the new DV protocol as below.
- Load pre-stored DV workspace file to quick config DV's Time Plot.

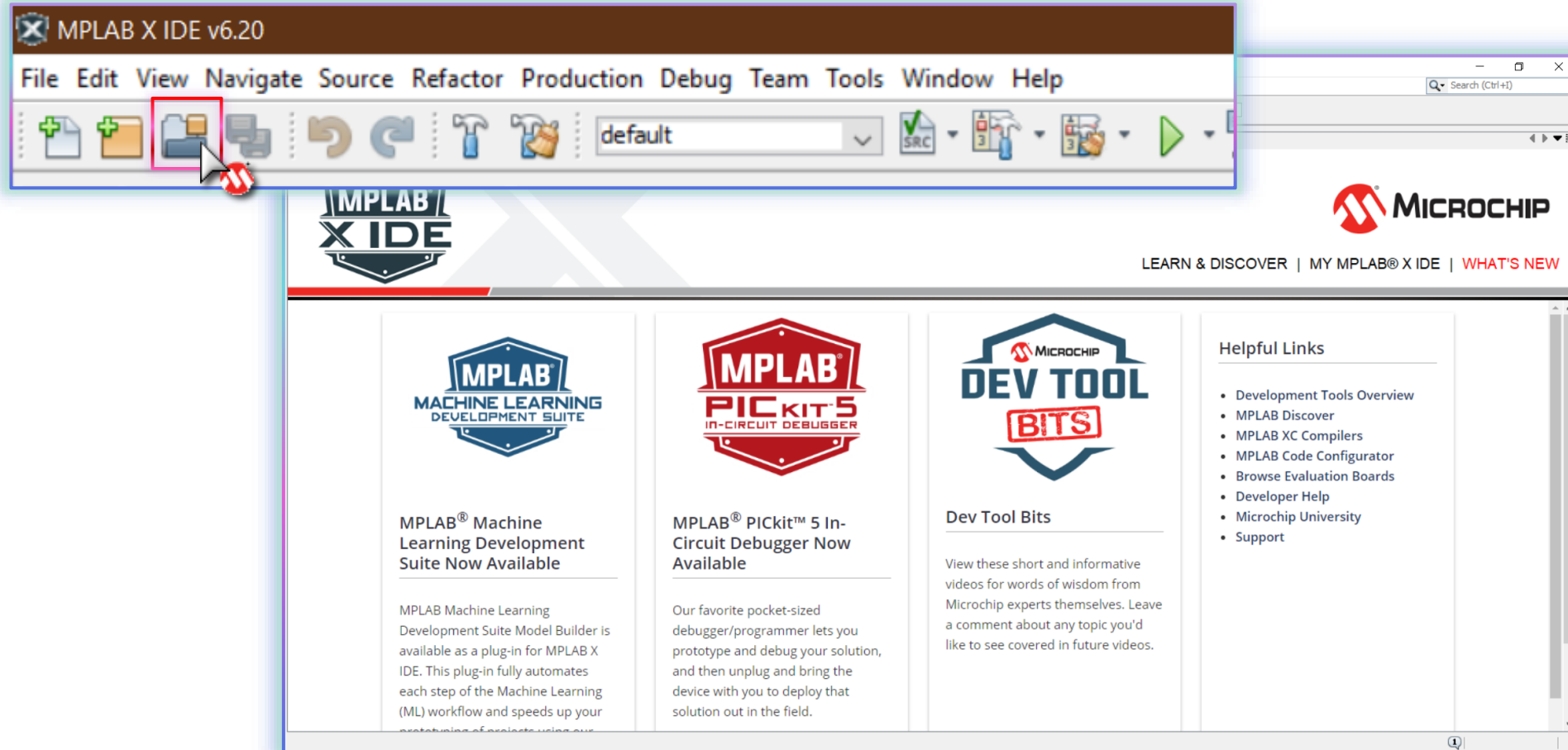


Let's go!

⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 1

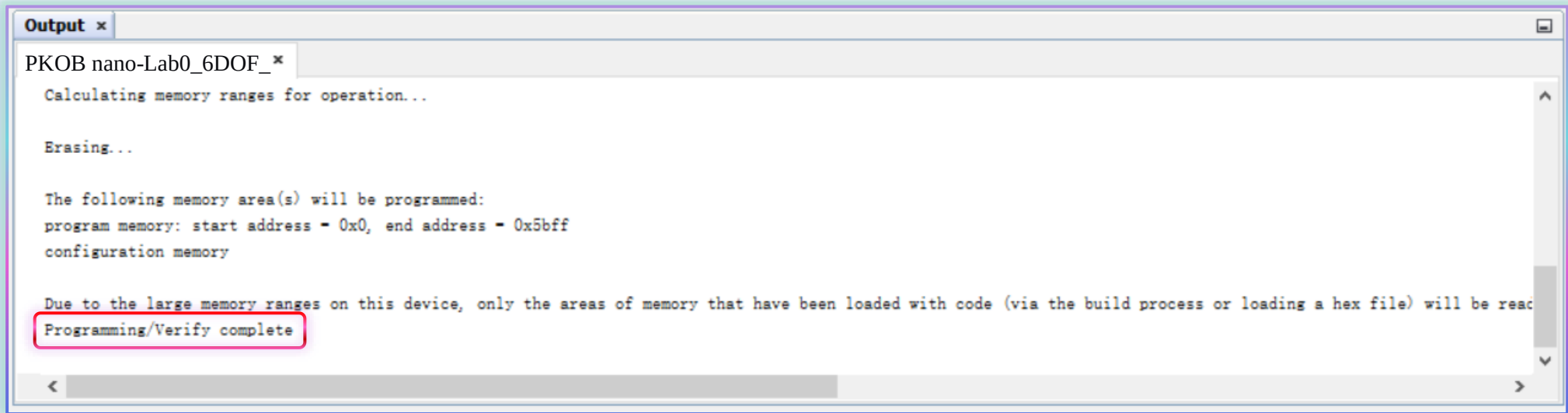
- Open “\AIML1001\Lab0_6DOF_Data_Visualizer\Lab0_6DOF_Data_Visualizer.X” .



⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 2

- Click on icon  and select **Clean and Build Project** to compile your project.
- Click on icon  to **Mark and Program** to EVB and check if any fault while Programming.
- Output Window will show **Programming/Verify Complete**.



The screenshot shows the 'Output' window of an IDE. The title bar says 'Output x'. The content area shows the following text:

```
PKOB nano-Lab0_6DOF_*
```

```
Calculating memory ranges for operation...
```

```
Erasing...
```

```
The following memory area(s) will be programmed:
```

```
program memory: start address = 0x0, end address = 0x5bff
```

```
configuration memory
```

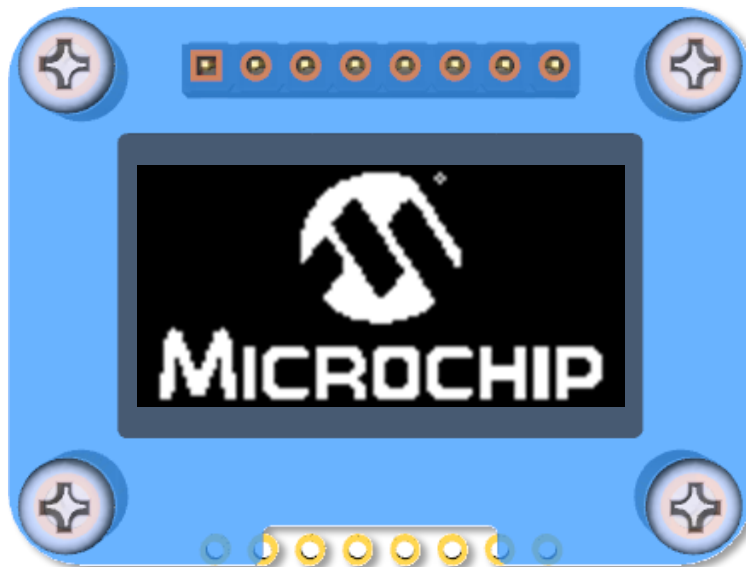
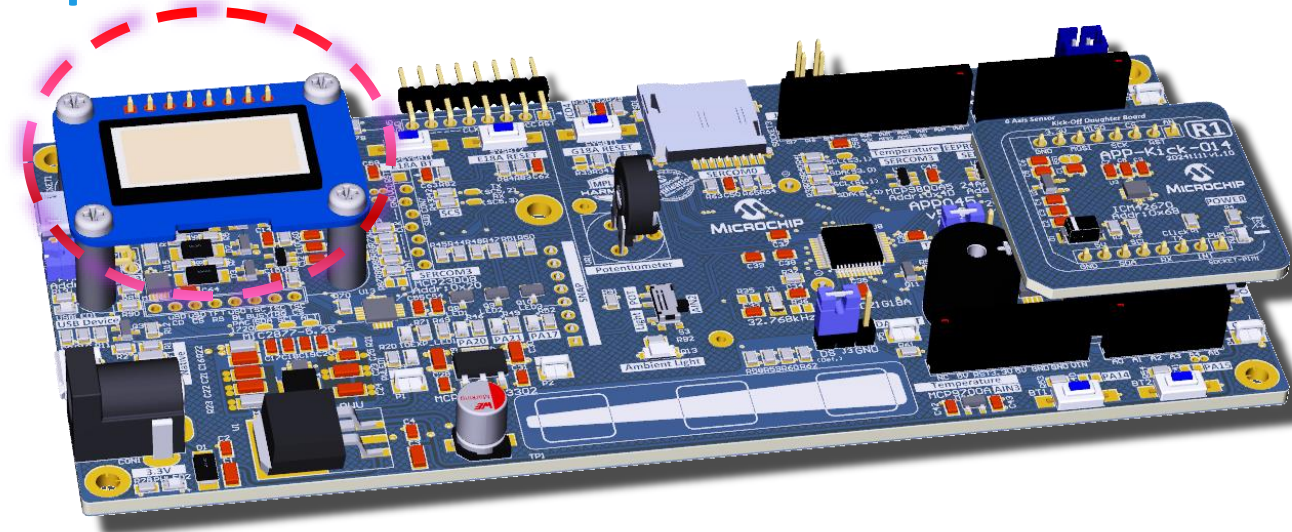
```
Due to the large memory ranges on this device, only the areas of memory that have been loaded with code (via the build process or loading a hex file) will be read
```

```
Programming/Verify complete
```

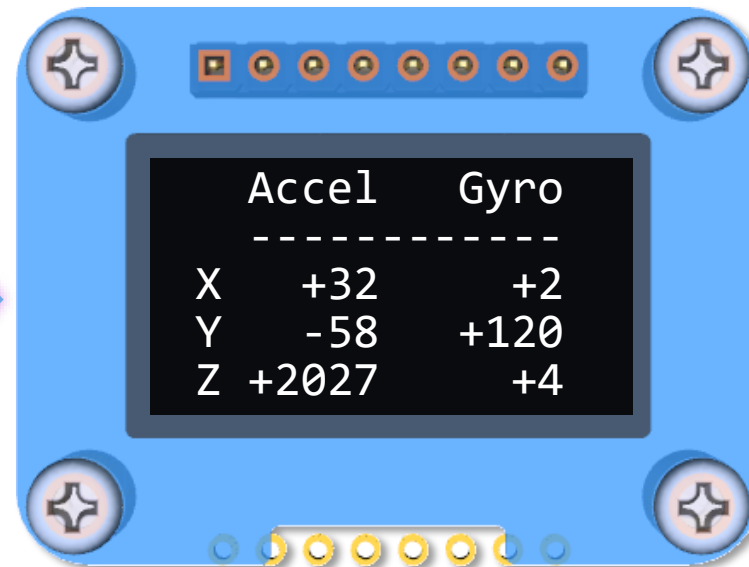
The text 'Programming/Verify complete' is highlighted with a red rectangular box. The window has a scrollbar on the right and a horizontal scrollbar at the bottom.

🔧 Lab0 Data Visualizer for 6DOF Sensor data

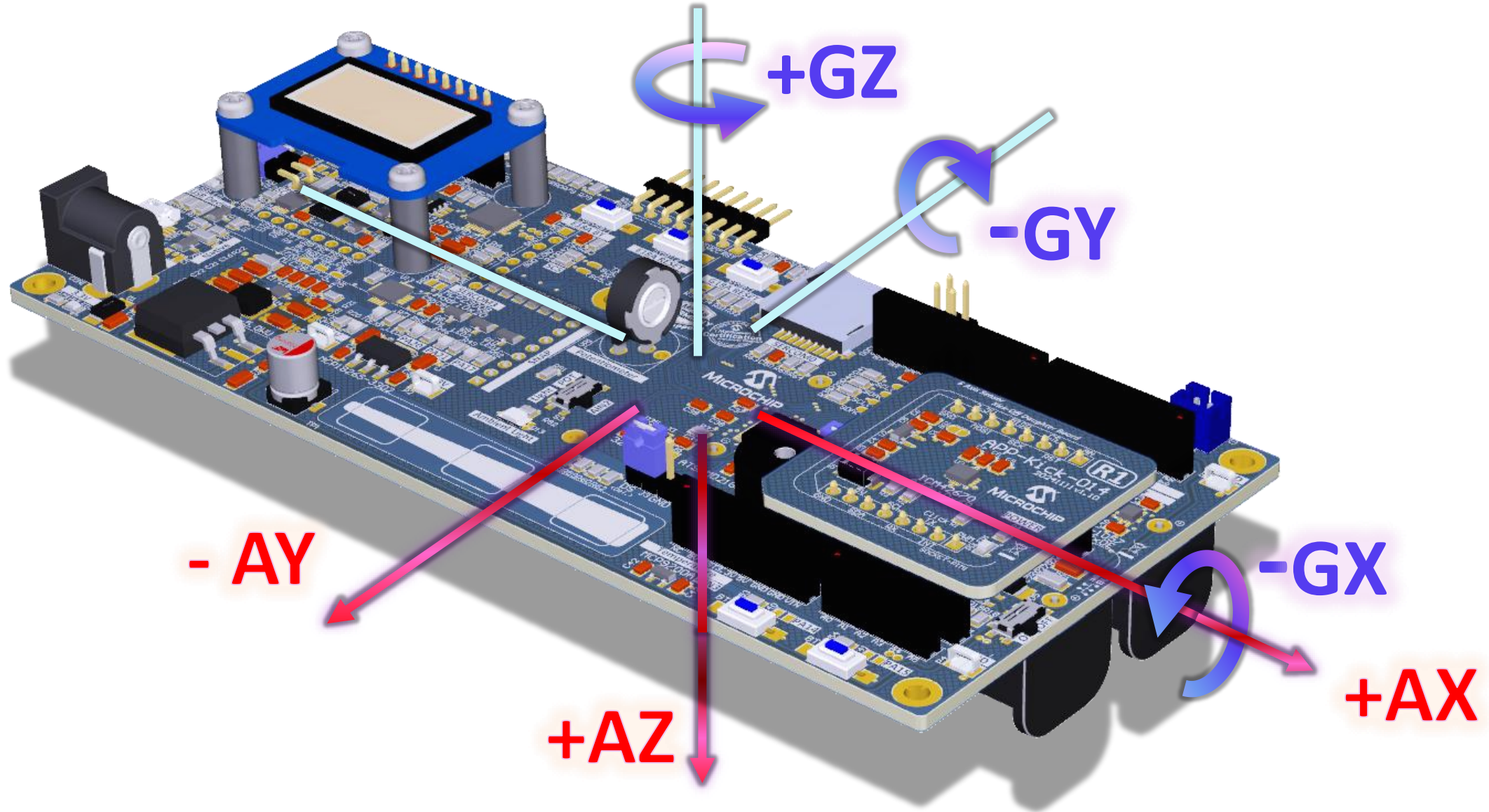
OLED Sensor Data Output



1 Sec.



The Sensor Axis of APP045 and APP-Kick-014



🔧 Lab0 Data Visualizer for 6DOF Sensor data

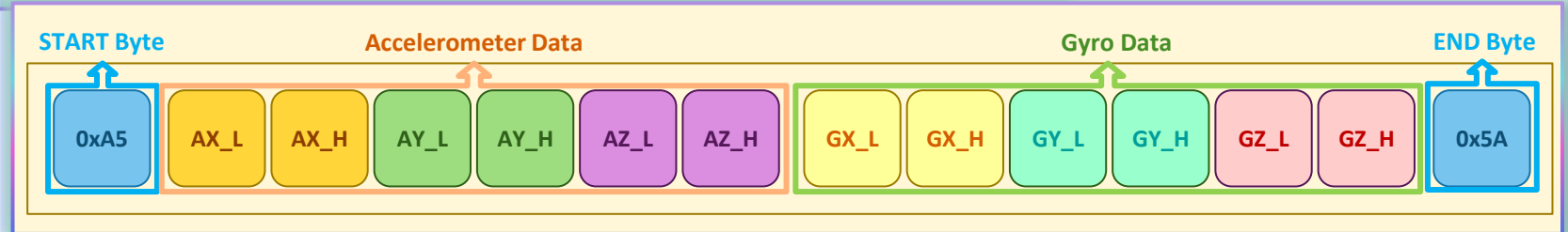
Step 3

- Find **TODO 0.01** and modify the **#if 1** to **#if 0** at Line 212.

```
int main (void)
{
    ...
    while ( true )
    {
        ...
        // TODO 0.01
        #if 0
```

Line 212

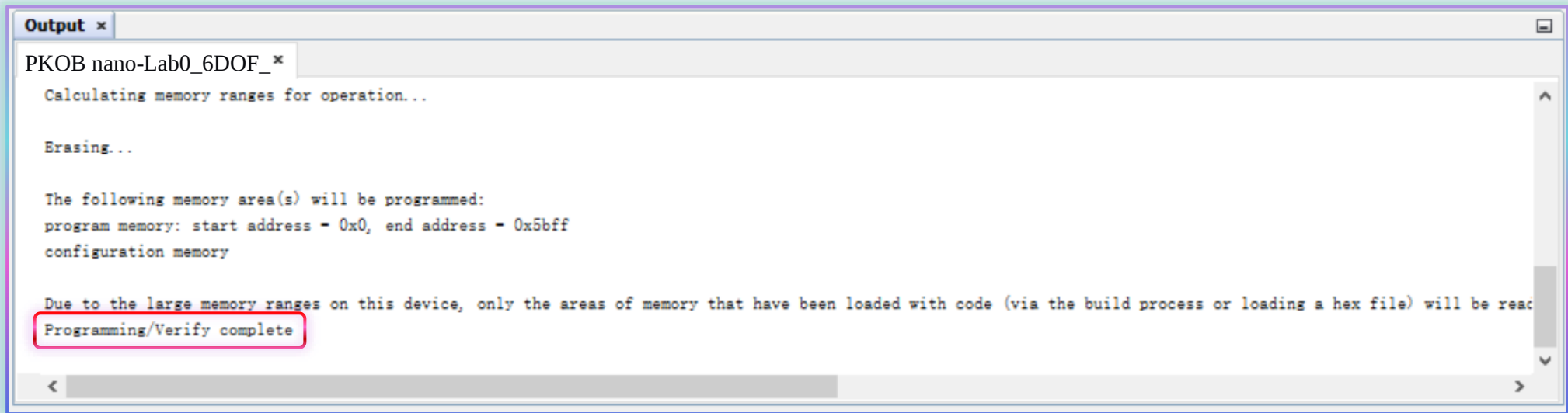
```
        sprintf((char *) USARTRTxBuffer, "    Accel    Gyro" );
        LCM_DrawString(0, 0, (char *) USARTRTxBuffer);
        ...
    #else
        sprintf( USARTRTxBuffer,
            "%c%c%c%c%c%c%c%c%c%c%c%c%c",
            0xA5,
            AxisOutByte_Accel[1], AxisOutByte_Accel[0],
            AxisOutByte_Accel[3], AxisOutByte_Accel[2],
            AxisOutByte_Accel[5], AxisOutByte_Accel[4],
            AxisOutByte_Gyro [1], AxisOutByte_Gyro [0],
            AxisOutByte_Gyro [3], AxisOutByte_Gyro [2],
            AxisOutByte_Gyro [5], AxisOutByte_Gyro [4],
            ~(0xA5));
        SERCOM5_USART_Write( USARTRTxBuffer, 14 );
    #endif
```



⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 4

- Click on icon  and select **Clean and Build Project** to compile your project.
- Click on icon  to **Mark and Program** to EVB and check if any fault while Programming.
- Output Window will show **Programming/Verify Complete**.



The screenshot shows the 'Output' window of an IDE. The title bar says 'Output x'. The main text area contains the following output for a project named 'PKOB nano-Lab0_6DOF_':

```
Calculating memory ranges for operation...

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x5bff
configuration memory

Due to the large memory ranges on this device, only the areas of memory that have been loaded with code (via the build process or loading a hex file) will be read

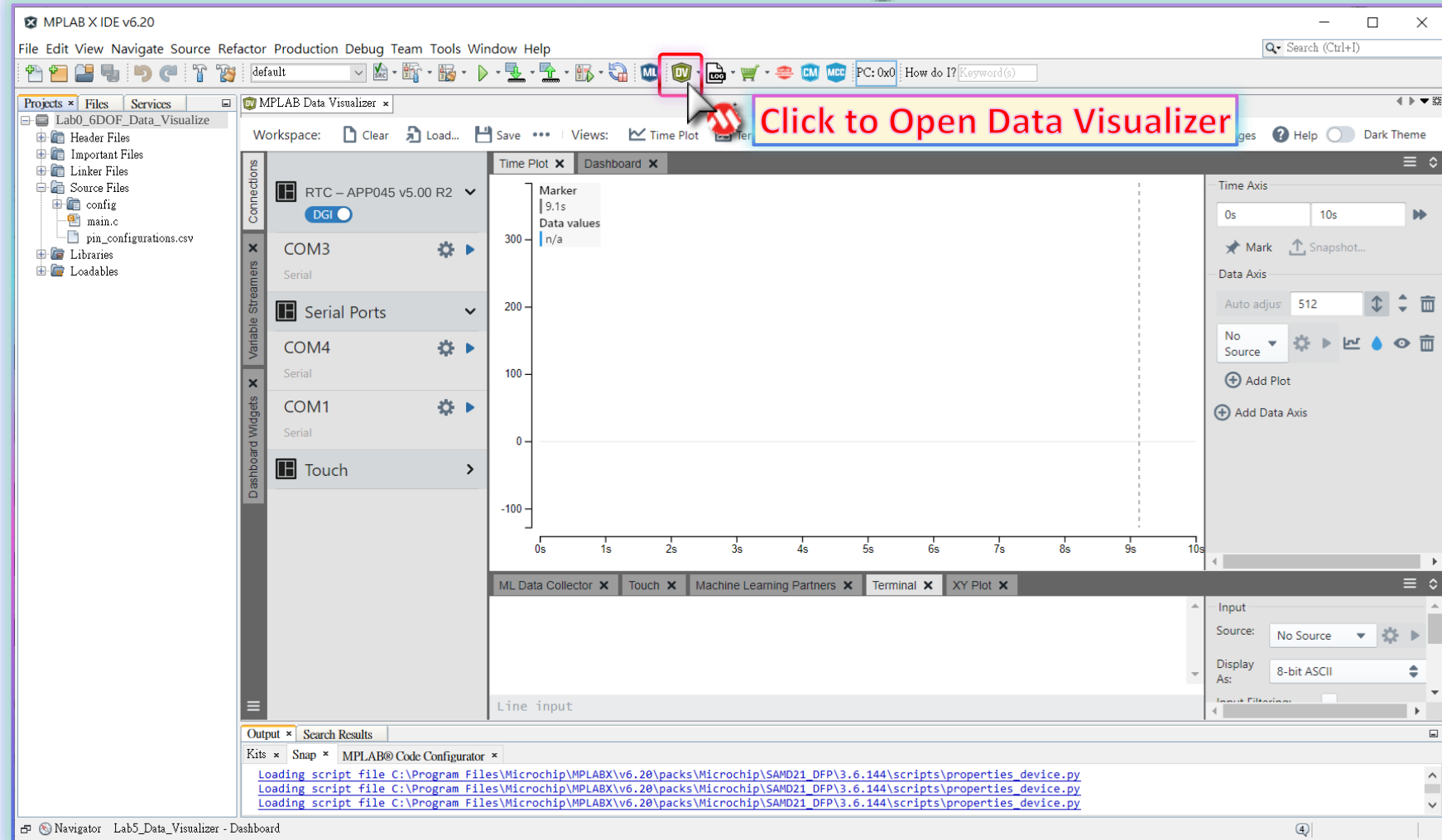
Programming/Verify complete
```

The text 'Programming/Verify complete' is highlighted with a red rectangular box. The window has a scrollbar on the right and a horizontal scrollbar at the bottom.

⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 5

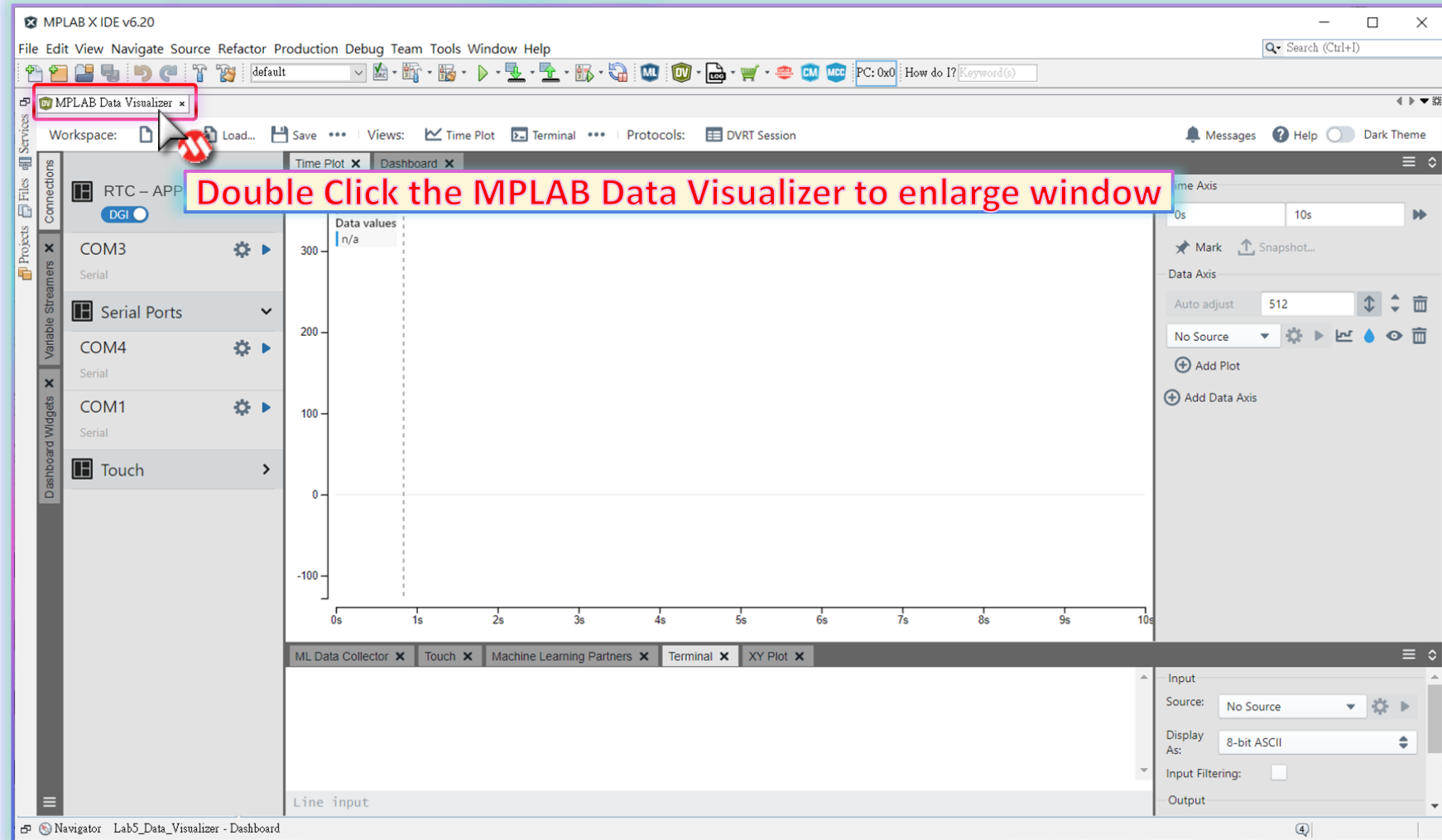
- Open MPLAB Data Visualizer by click the icon



⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 6

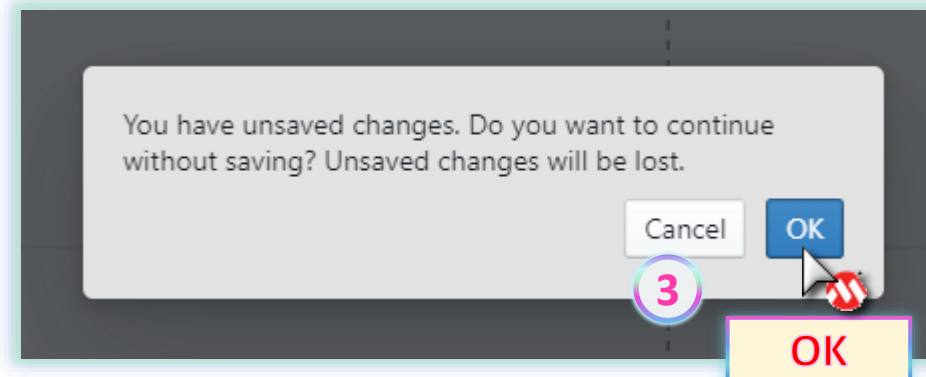
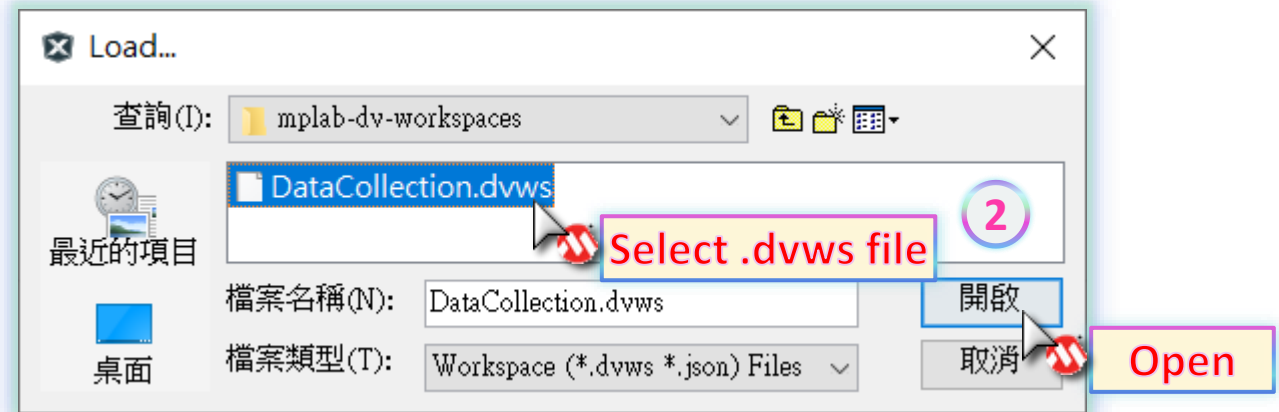
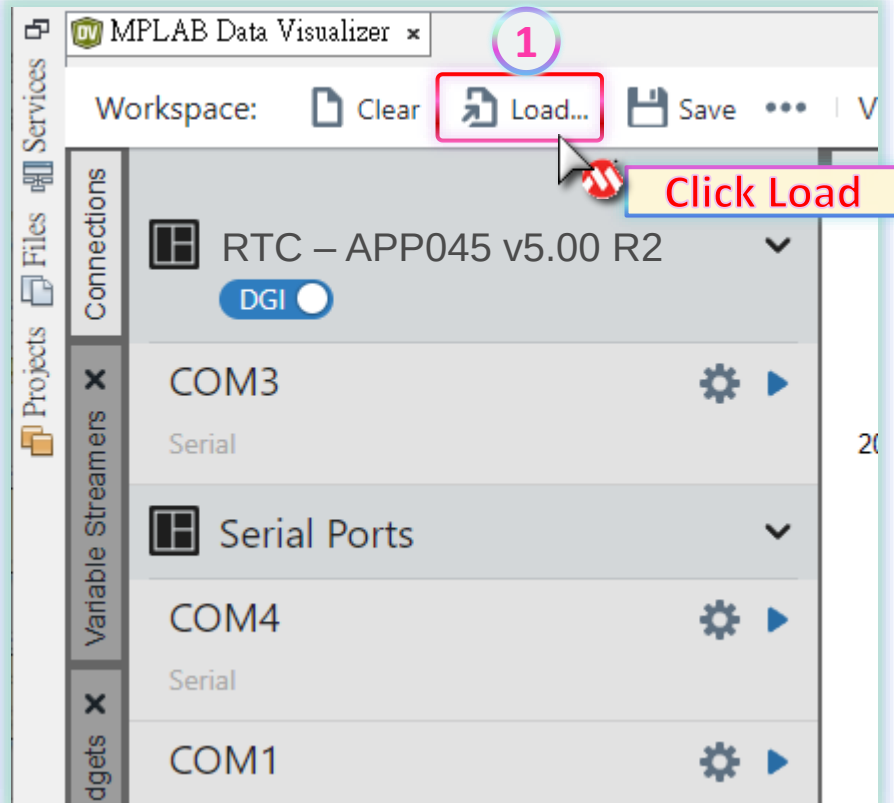
- Double-Click on **MPLAB Data Visualizer** tab to enlarge DV window.



⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 7

- Load Workspace file **DataCollection.dvws** in **\AIML1001\Recourse\mplab-dv-workspace**



⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 8

- Click on ▼ to select correct **COM port** and then click on ⚙️ to config as **115200-n-8-1**.

The image consists of two side-by-side screenshots of the MPLAB Data Visualizer software interface, illustrating the steps to select a COM port and configure its settings.

Left Screenshot:

- 1** A vertical label "Variable Streamers" is on the left side of the interface.
- 2** A red box highlights the dropdown menu in the "Sensor" section, which currently shows "No Source".
- 3** A red box highlights the "COM3 on RTC - APP045 v5.00 R2" option in the dropdown menu.
- 4** A red box highlights the gear icon (settings) next to the selected sensor.
- 5** A red box highlights the "COM3 Settings" dialog box, which is open and shows the "Baud Rate" set to "115200".
- 6** A red box highlights the "COM3" option in the "Variable Streamers" list on the left.
- 7** A red box highlights the "COM3 on RTC - APP045 v5.00 R2" option in the "Variable Streamers" list on the left.
- 8** A red box highlights the "COM3" option in the "Variable Streamers" list on the left.

Right Screenshot:

- 9** A red box highlights the "COM3 Settings" dialog box, which is open and shows the "Baud Rate" set to "115200".
- 10** A red box highlights the "Baud Rate" field in the "COM3 Settings" dialog box, which is set to "115200".
- 11** A red box highlights the "Char Length" field in the "COM3 Settings" dialog box, which is set to "8 bits".
- 12** A red box highlights the "Parity" field in the "COM3 Settings" dialog box, which is set to "None".
- 13** A red box highlights the "Stop Bits" field in the "COM3 Settings" dialog box, which is set to "1 bit".

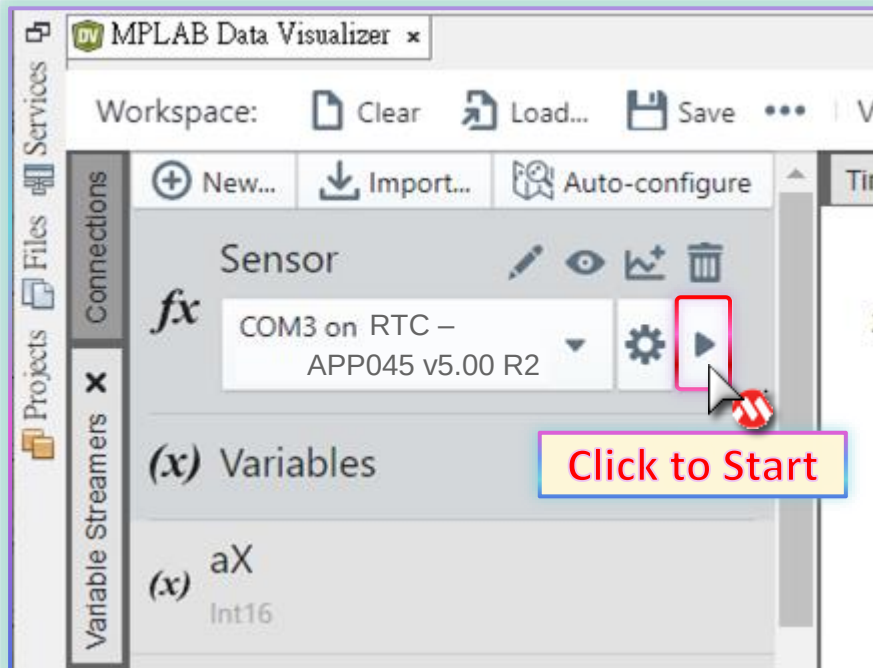
Annotations:

- Click to Select COM Port** (points to the gear icon in the left screenshot)
- Click correct COM Port** (points to the "COM3" option in the left screenshot)

⚙️ Lab0 Data Visualizer for 6DOF Sensor data

Step 9

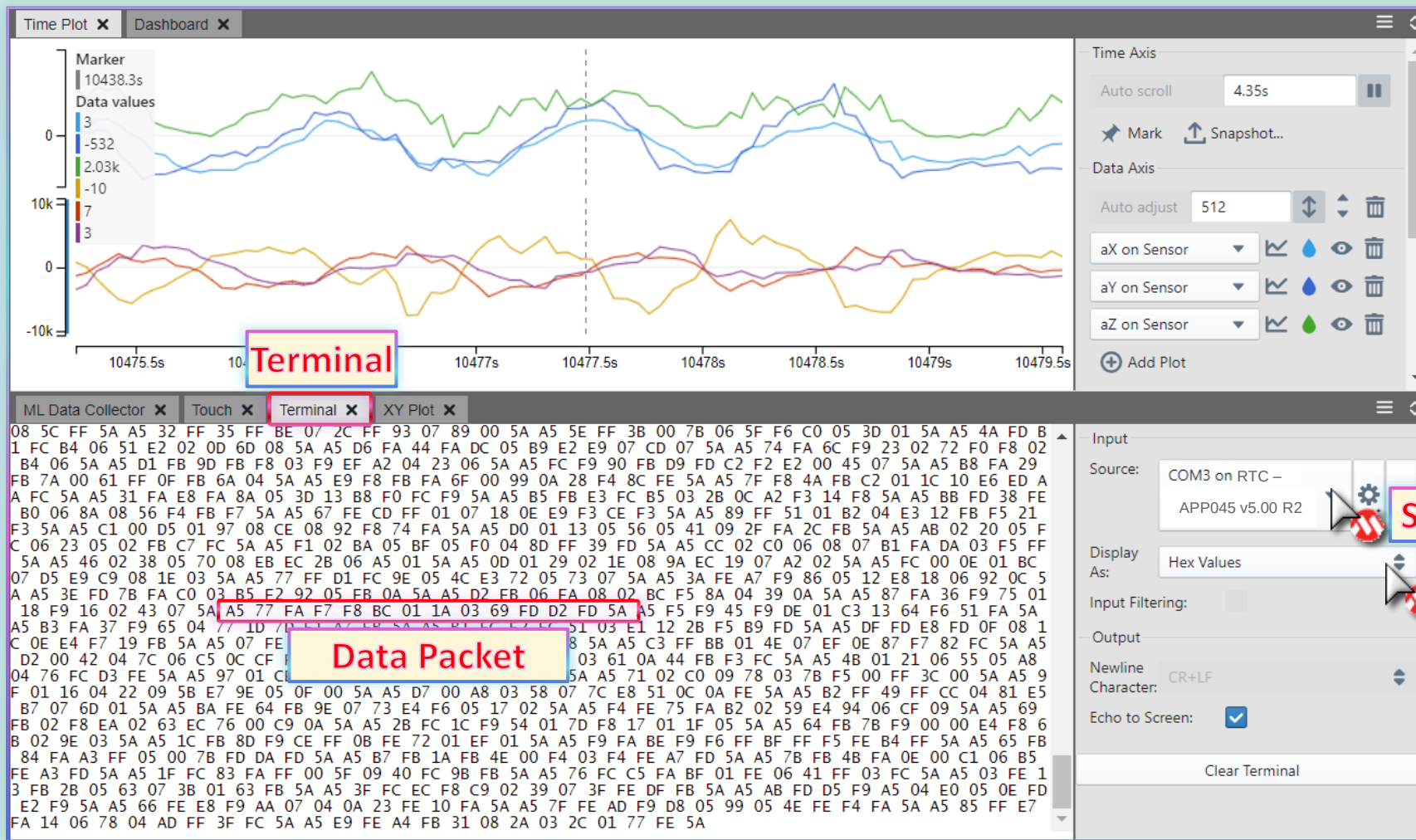
- Click on ▶ to start Data Stream output raw data waveform in **Time Plot** output.



Lab0 Data Visualizer for 6DOF Sensor data

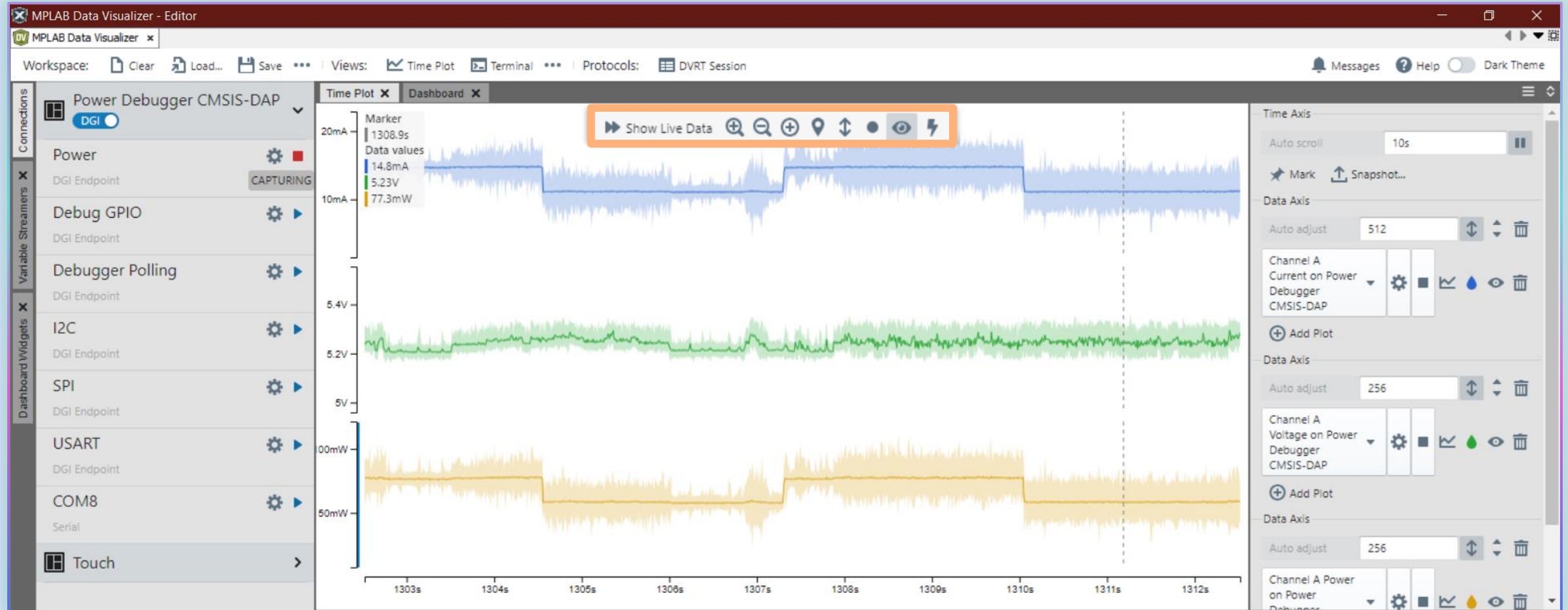
Step 10

- Check DV protocol Raw Data on **Terminal** Window.



MPLAB® Data Visualizer

Data Axis and Plot of Time Plot (Waveform Viewer)



MPLAB® Data Visualizer

Toolbar Control of Time Plot

Show the information of all cursors. Including bandwidth, Time period, and Frequency

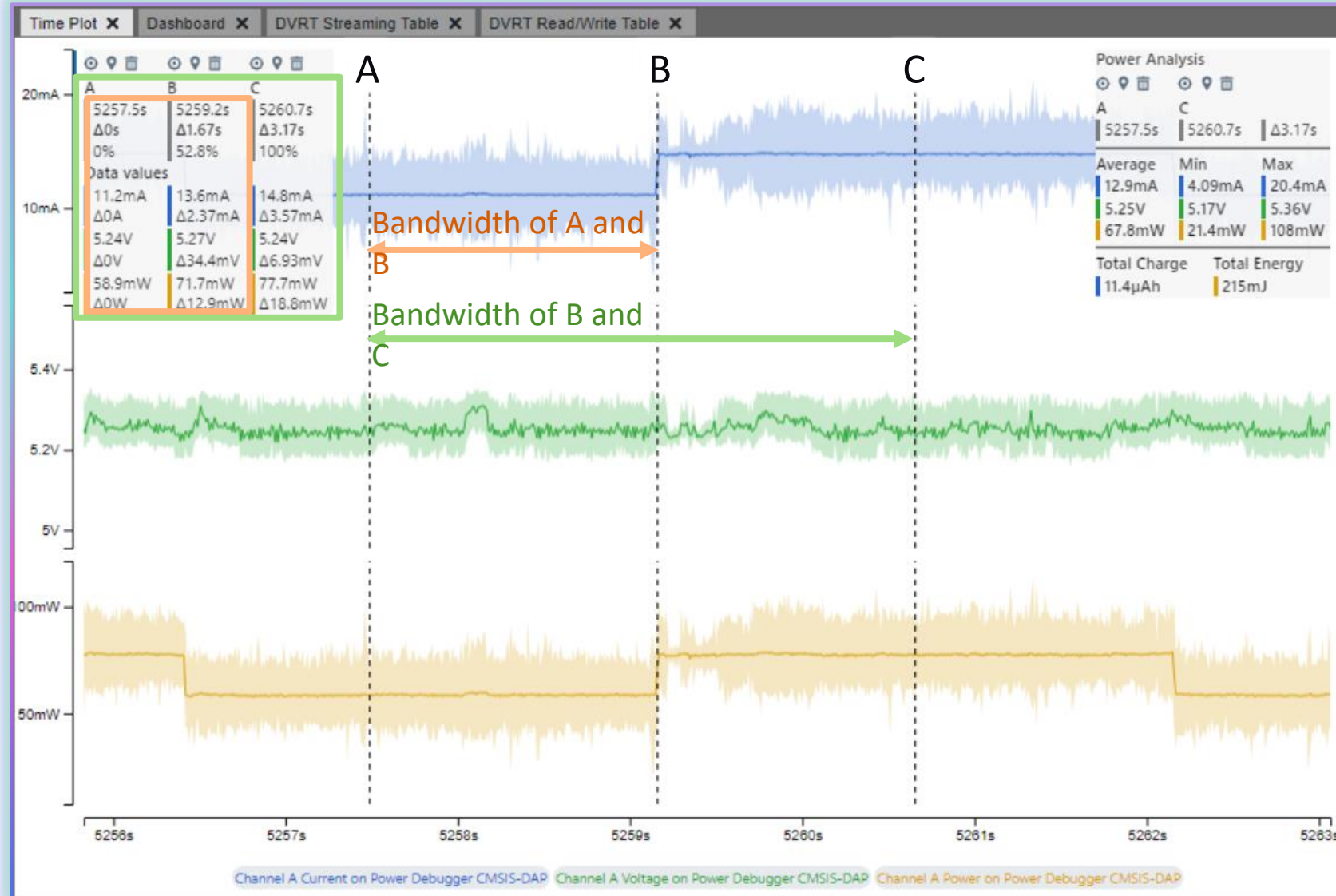


Need cursor to show the window.

Click to show/hide the Axis.

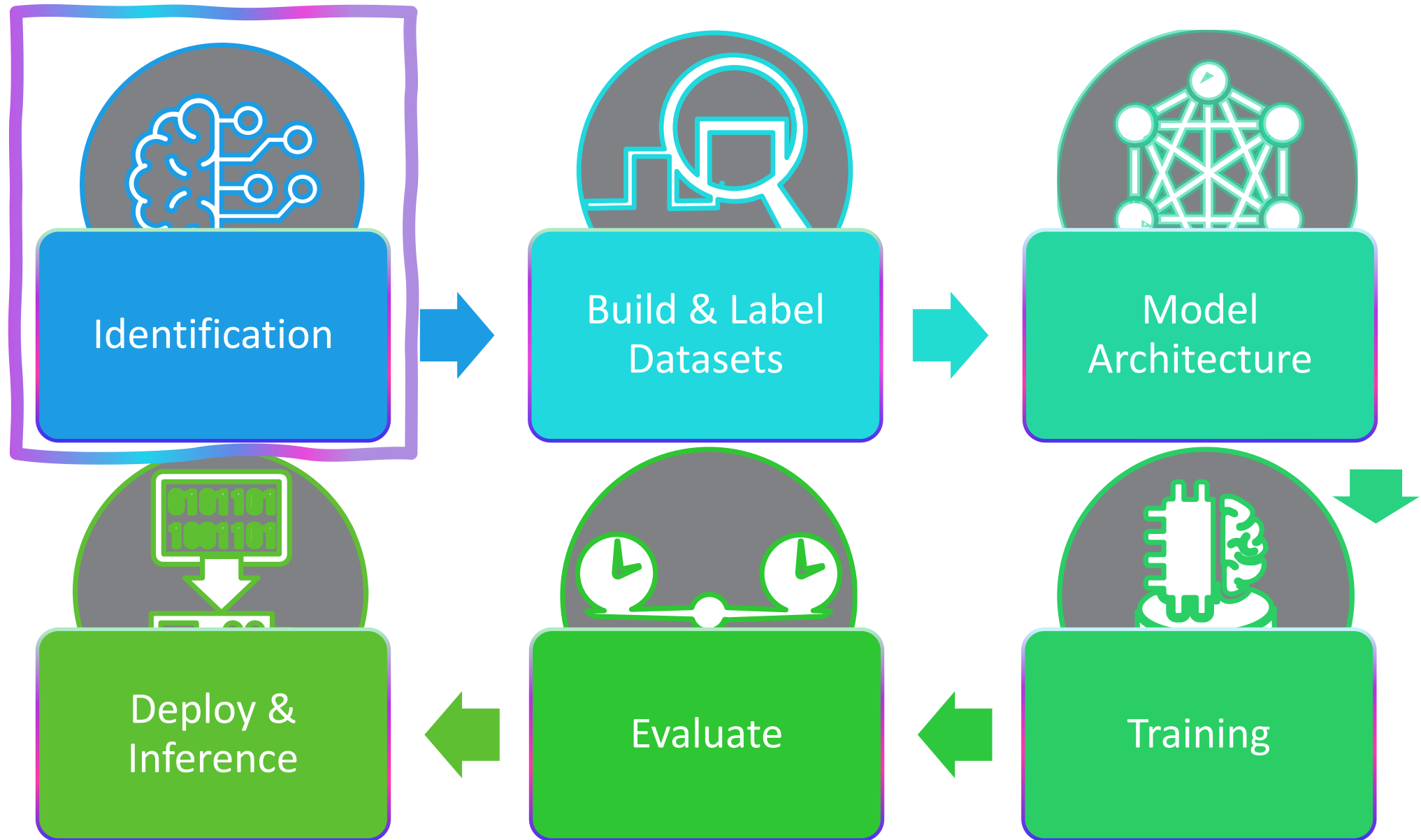
MPLAB® Data Visualizer

Cursor Delta Time Measurement by Time Plot



Set Goal and Problem Identification

Machine Learning Workflow



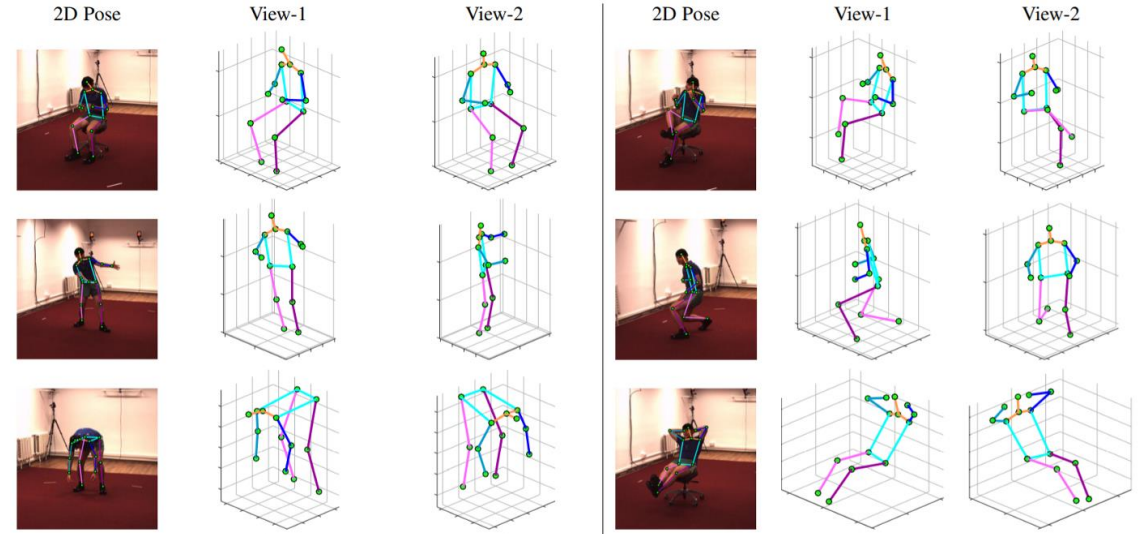
What is the goal of today?

- Build a boxing recognition model and deploy it on the APP045 development board with 6-axis motion sensor ICM42670.
- The model should recognize **4** different movements, which is:
 - Straight
 - Right hook
 - Uppercut
 - Freeze



Problem Identification

- **Detect boxing action posture**
 - Straight, Uppercut, Right Hook, Freeze.
- **Classic approach(es)**
 - Camera, Computer vision
 - Dynamic system, Hand tracking
- **Let's try Machine Learning**
 - Does it offer any benefits?
 - Development speed?
 - Accuracy?
 - Code size?



Lab 1

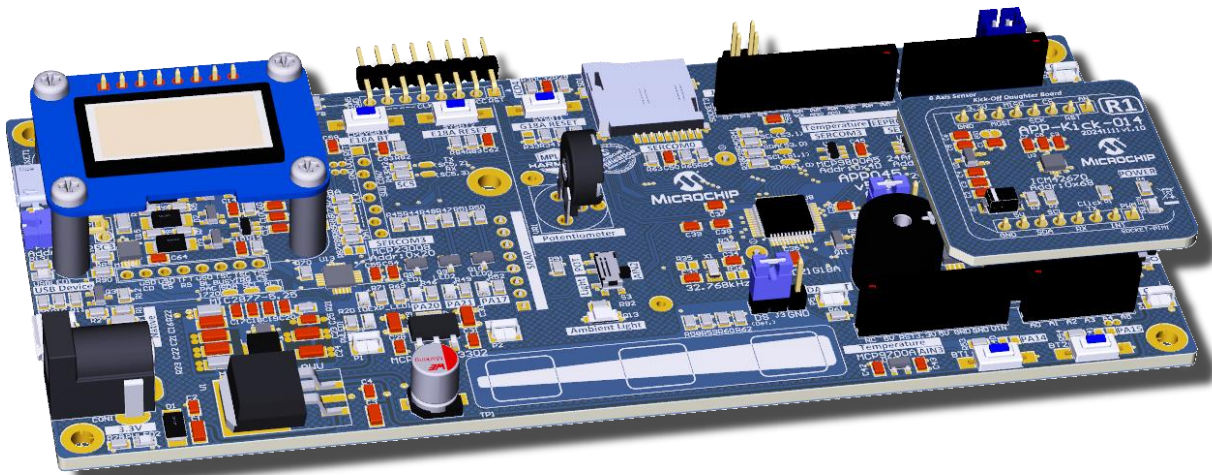
Initial Setup and Data Collecting

Boxing Recognition by Machine Learning

🔧 Lab1 Initial Setup and Data Collecting

Boxing Recognition by Machine Learning

- **Verify the hardware setup and open the project.**
 - APP045 v5.0 R2 EV Board with SAMD21
 - APP-Kick-014 Accel and Gyro (6DOF) daughter board
- **Program the project and make sure it performs data samples in 3 different modes.**
- **Create a ML project by ML Data Collector.**
- **Collect sample data set for Straight, Uppercut, and Right hook three actions, and observing good labelling practice.**

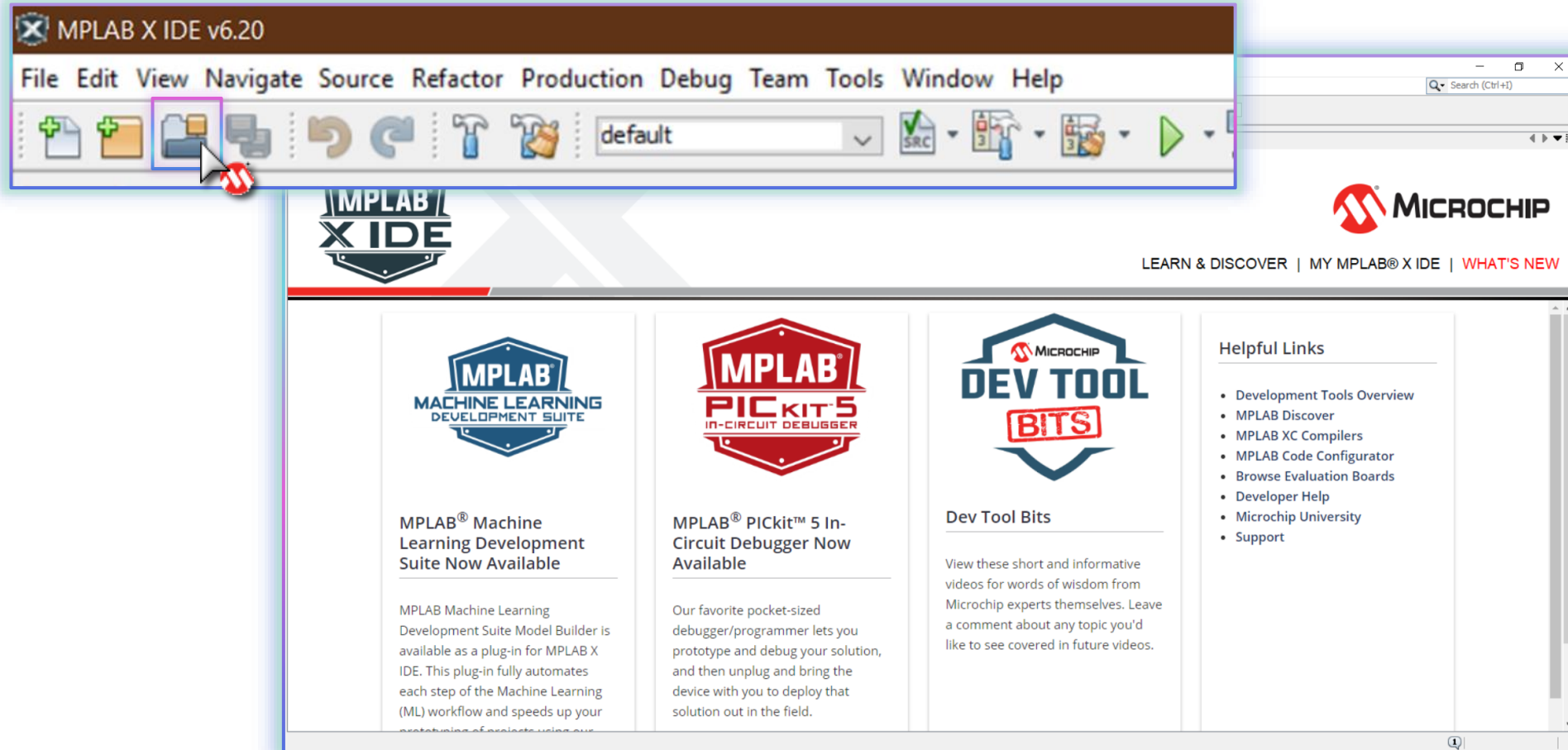


Let's go!

⚙️ Lab1 Initial Setup and Data Collecting

Step 1

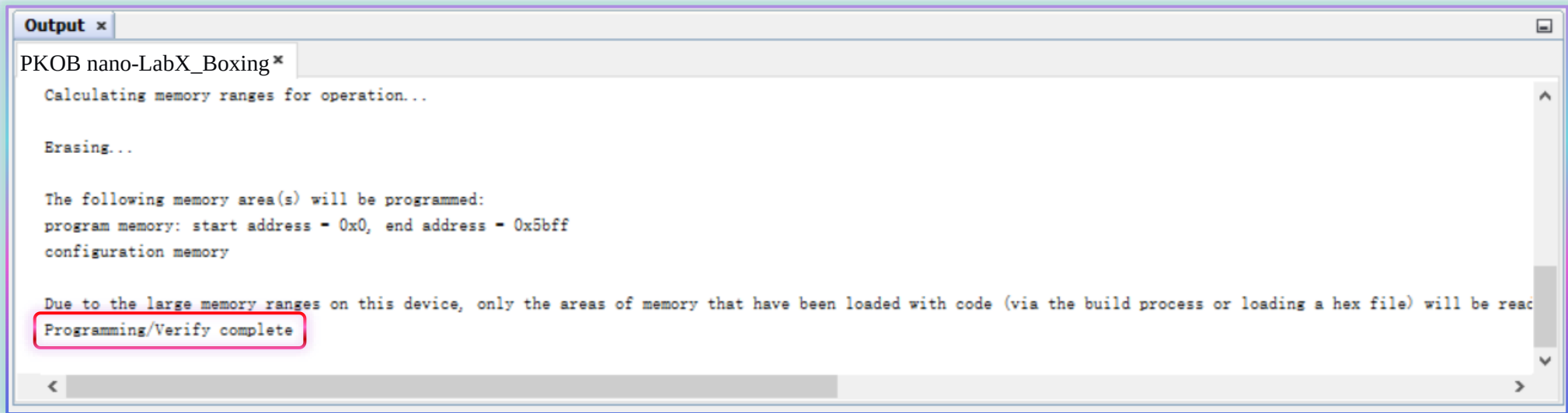
- Open “\AIML1001\LabX_BoxingPoseAIML\LabX_BoxingPoseAIML.X”.



⚙️ Lab1 Initial Setup and Data Collecting

Step 2

- Click on icon  and select **Clean and Build Project** to compile your project.
- Click on icon  to **Mark and Program** to EVB and check if any fault while Programming.
- Output Window will show **Programming/Verify Complete**.



The screenshot shows the 'Output' window of an IDE. The title bar says 'Output x'. The content area shows the following text:

```
PKOB nano-LabX_Boxing*  
Calculating memory ranges for operation...  
  
Erasing...  
  
The following memory area(s) will be programmed:  
program memory: start address = 0x0, end address = 0x5bff  
configuration memory  
  
Due to the large memory ranges on this device, only the areas of memory that have been loaded with code (via the build process or loading a hex file) will be read  
Programming/Verify complete
```

The text 'Programming/Verify complete' is highlighted with a red rectangular box. The window has a scrollbar on the right and a horizontal scrollbar at the bottom.

🔧 Lab1 Initial Setup and Data Collecting

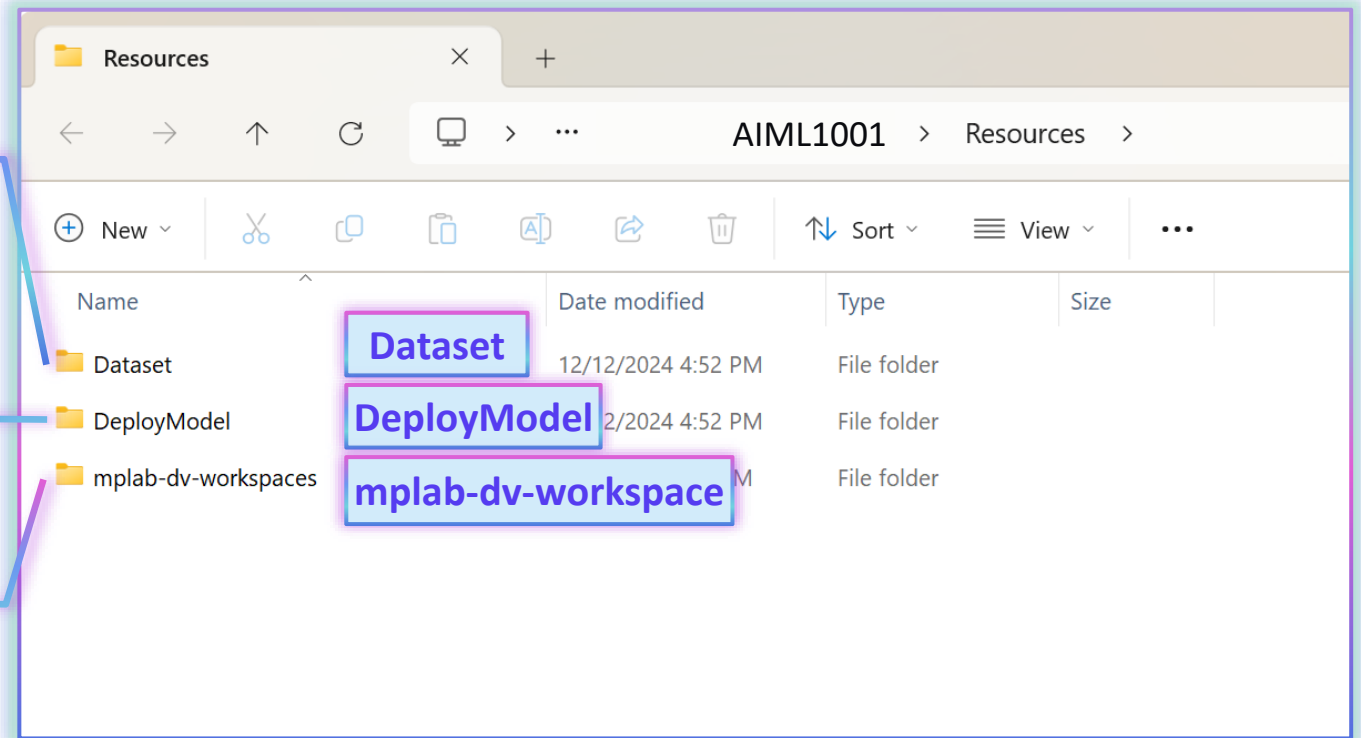
Step 3

- Let's see the folder “\AIML1001\Resources” in the project.
- There are three folders, which is Dataset, DeployModel, and mplab-dv-workspaces.

Dataset: The folder for you to save all the dataset collected in following steps.

DeployModel: Save the final model you download.

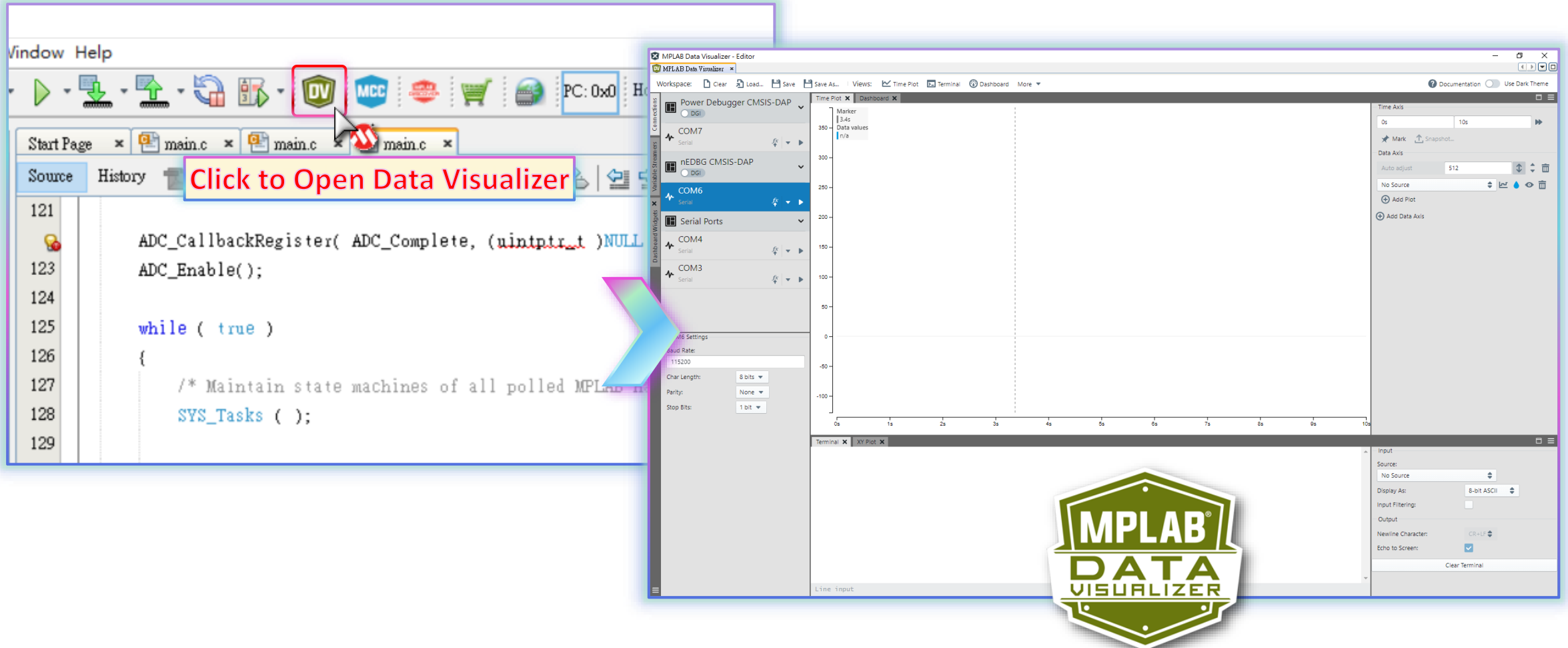
mplab-dv-workspaces: The workspace settings of DV.



⚙️ Lab1 Initial Setup and Data Collecting

Step 4

- Open Data Visualizer. Click icon .

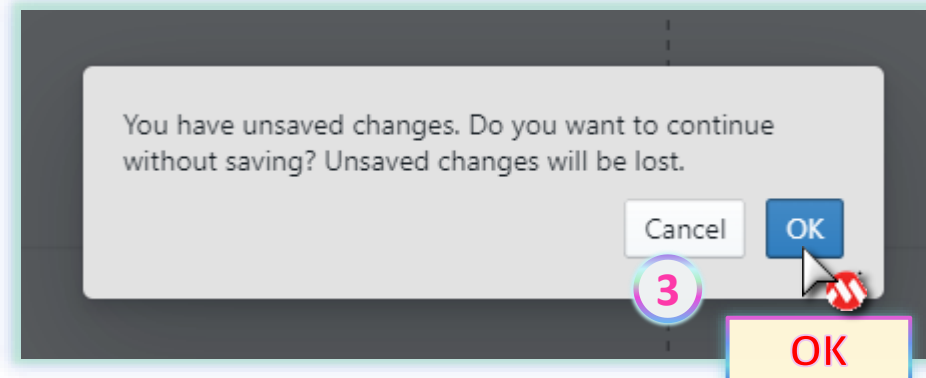
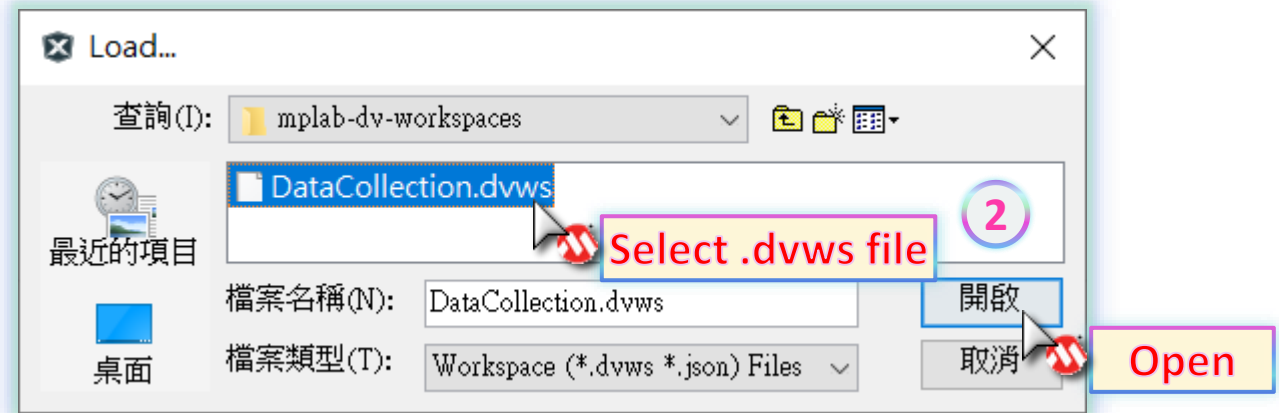
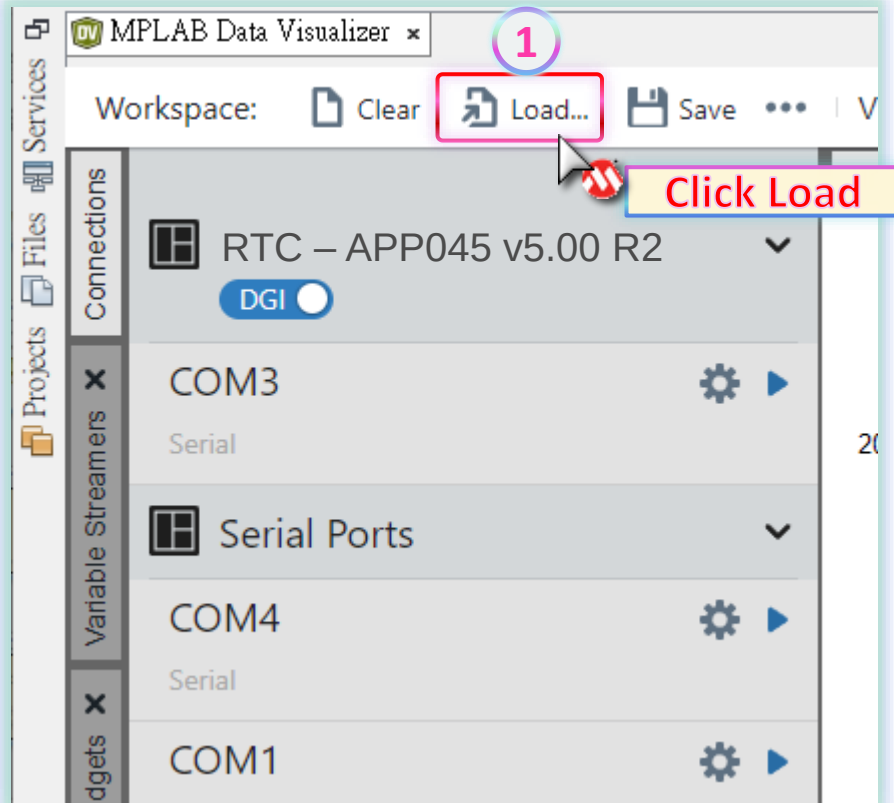


The screenshot illustrates the process of opening the MPLAB Data Visualizer. On the left, the MPLAB IDE interface is shown with the 'Data Visualizer' icon (a green shield with 'DV') highlighted in the toolbar. A red box and a callout arrow point to this icon, with a text box stating 'Click to Open Data Visualizer'. Below the toolbar, a snippet of C code is visible, including `ADC_CallbackRegister( ADC_Complete, (uintptr_t) NULL`, `ADC_Enable();`, and a `while ( true )` loop containing `SYS_Tasks ( );`. On the right, the 'MPLAB Data Visualizer - Editor' window is displayed. It features a 'Workspace' panel on the left with a tree view showing connections to 'Power Debugger CMSIS-DAP', 'COM7', 'nEDBG CMSIS-DAP', 'COM6', 'Serial Ports', 'COM4', 'COM3', and 'Serial'. The main area is a 'Time Plot' showing a graph of 'Data values' over time, with a marker at 3.4s. The right sidebar contains settings for 'Time Axis' (0s to 10s), 'Data Axis' (Auto adjust, 512), and 'Input' (No Source, 8-bit ASCII, Input Filtering, Output, Newline Character, Echo to Screen, and Clear Terminal).

⚙️ Lab1 Initial Setup and Data Collecting

Step 5

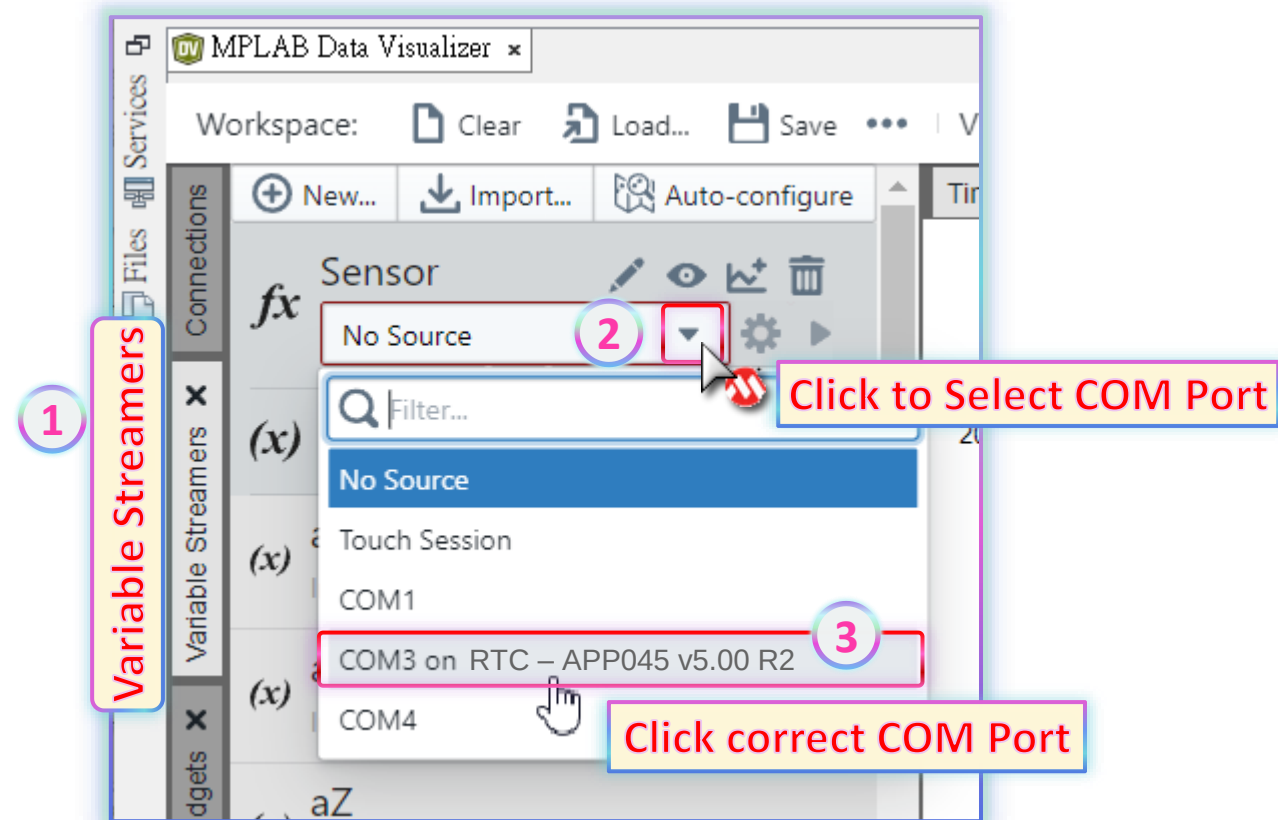
- Load Workspace file **DataCollection.dvws** in **\AIML1001\Recourse\mplab-dv-workspace**



⚙️ Lab1 Initial Setup and Data Collecting

Step 6

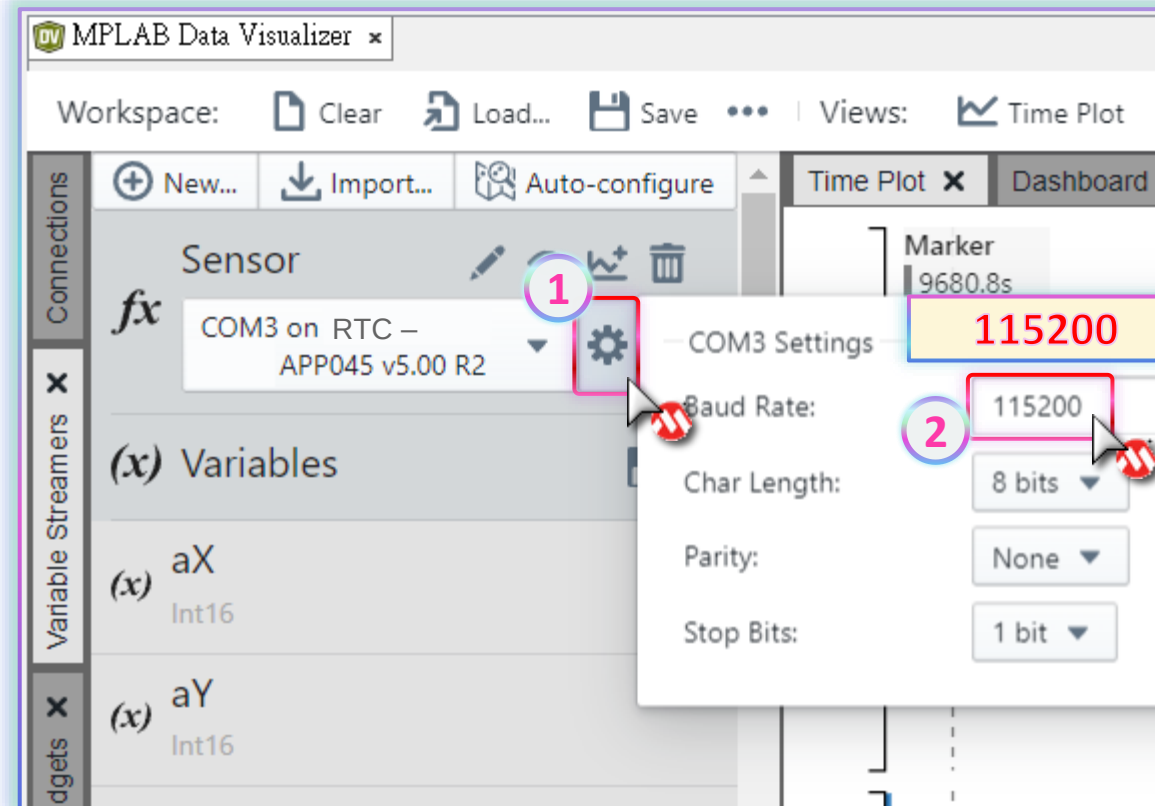
- Go to left tab “**Variable Streamers**” > **Sensor** > **No Source** > **COMxx**



⚙️ Lab1 Initial Setup and Data Collecting

Step 7

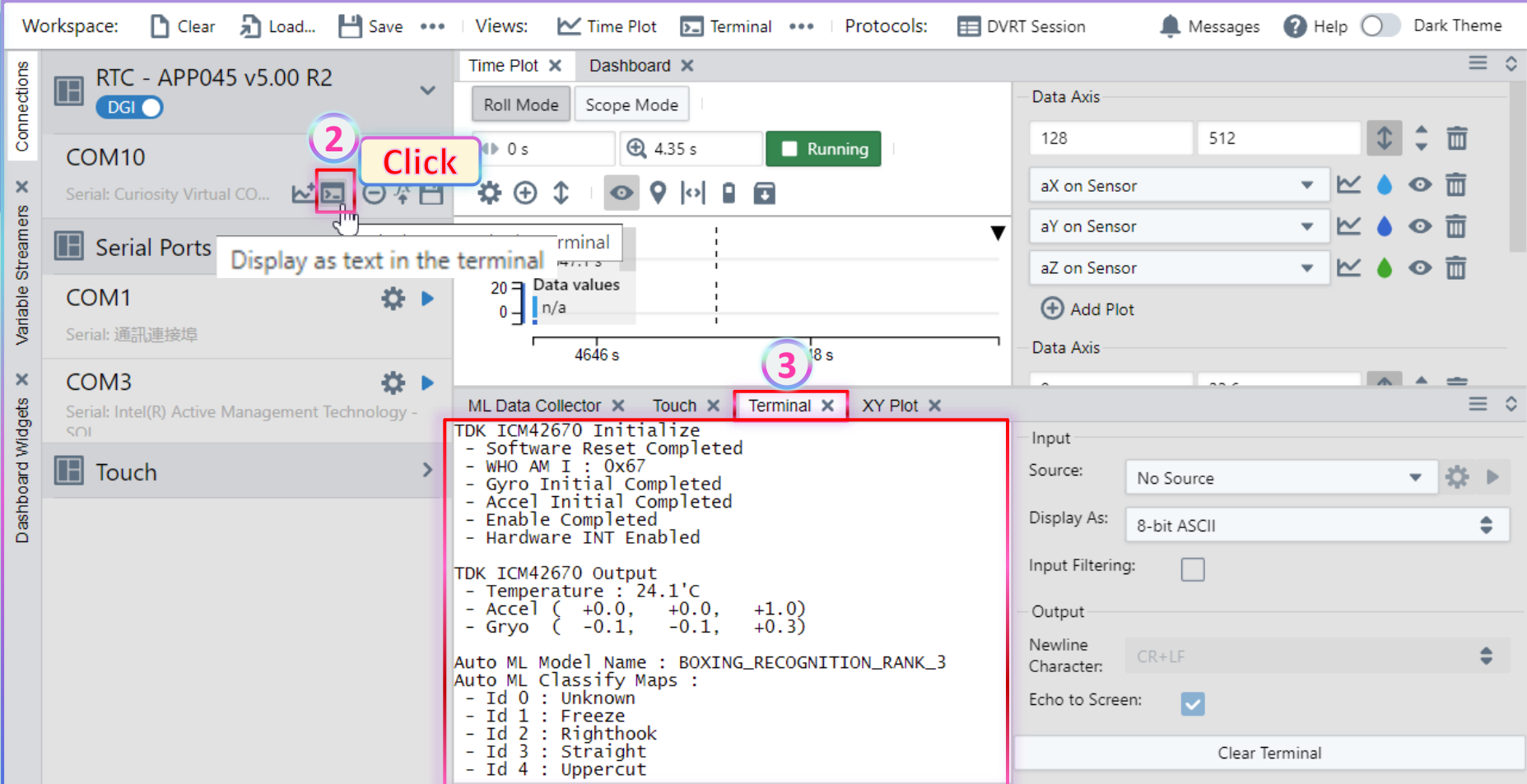
- Sensor > ⚙️ (Source Option) > Baud Rate: 115200



⚙️ Lab1 Initial Setup and Data Collecting

Step 8

- Go to left tab “Connections” Click on  and display the text in **Terminal** output.



The screenshot shows the Microchip Studio IDE interface. On the left, the 'Connections' tab is selected, showing a list of serial ports: COM10, COM1, and COM3. A red box highlights the 'COM10' entry, and a yellow callout bubble with the number '2' and the text 'Click' points to it. A red box also highlights the 'Terminal' icon in the top toolbar, with a yellow callout bubble with the number '3' and the text 'Click' pointing to it. The 'Terminal' window is open at the bottom, displaying the following text:

```
TDK ICM42670 Initialize
- Software Reset Completed
- WHO AM I : 0x67
- Gyro Initial Completed
- Accel Initial Completed
- Enable Completed
- Hardware INT Enabled

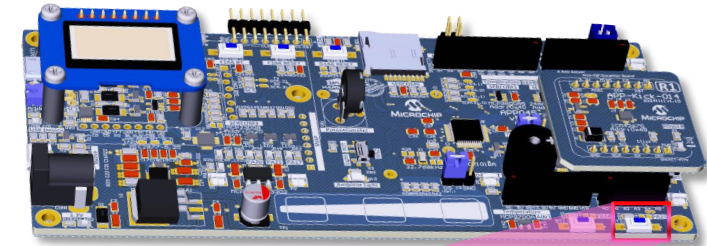
TDK ICM42670 Output
- Temperature : 24.1'C
- Accel ( +0.0, +0.0, +1.0)
- Gryo ( -0.1, -0.1, +0.3)

Auto ML Model Name : BOXING_RECOGNITION_RANK_3
Auto ML Classify Maps :
- Id 0 : Unknown
- Id 1 : Freeze
- Id 2 : Righthook
- Id 3 : Straight
- Id 4 : Uppercut
```

The 'Terminal' window also has a 'Clear Terminal' button at the bottom right.

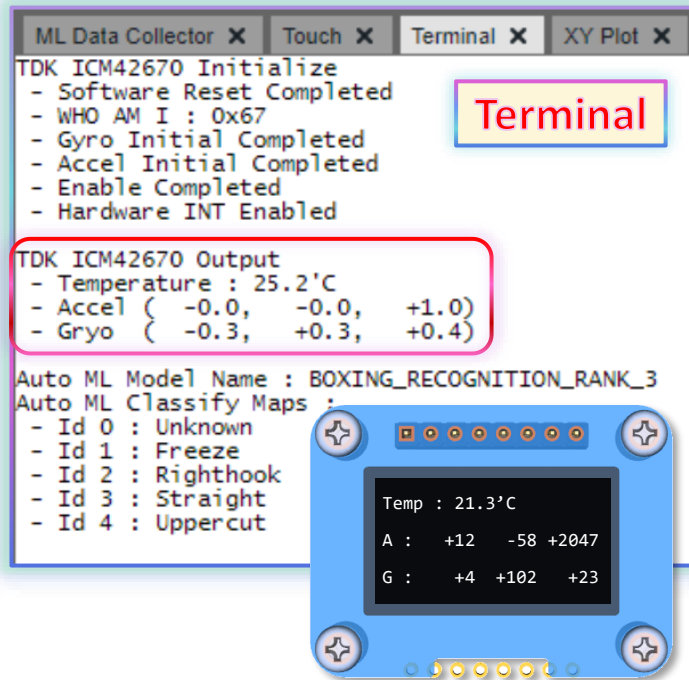
⚙️ Lab1 Initial Setup and Data Collecting

Step 9

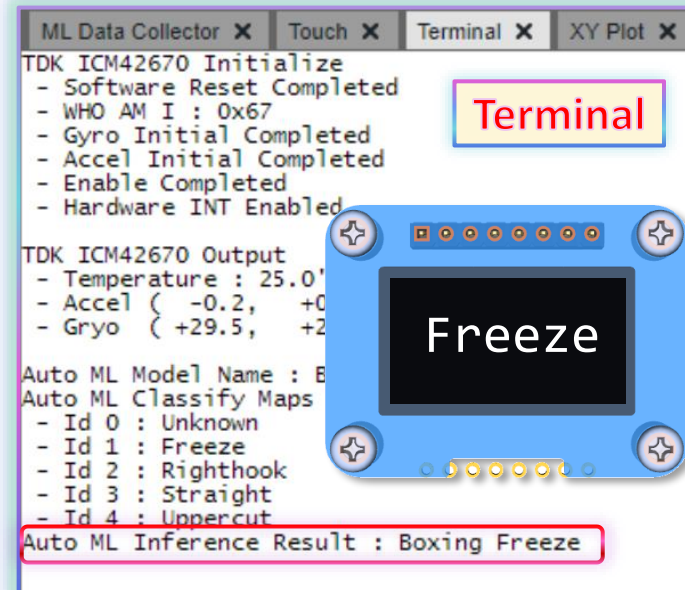


- There are 3 different modes, we can switch the mode by pressing BT2.
 - **Mode 1**: Display 6DOF Sensor Data in DV Terminal and OLED.
 - **Mode 2**: Display **Boxing Stance classification results** of preset ML model in DV Terminal and OLED.
 - **Mode 3**: Display 6DOF Sensor Data waveform in Time Plot of Data Visualizer (**For Data Collection**).

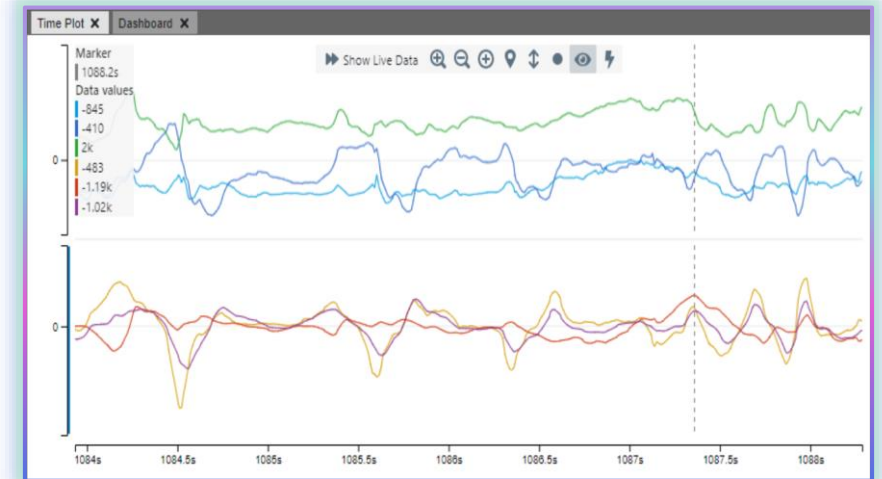
Mode 1



Mode 2



Mode 3

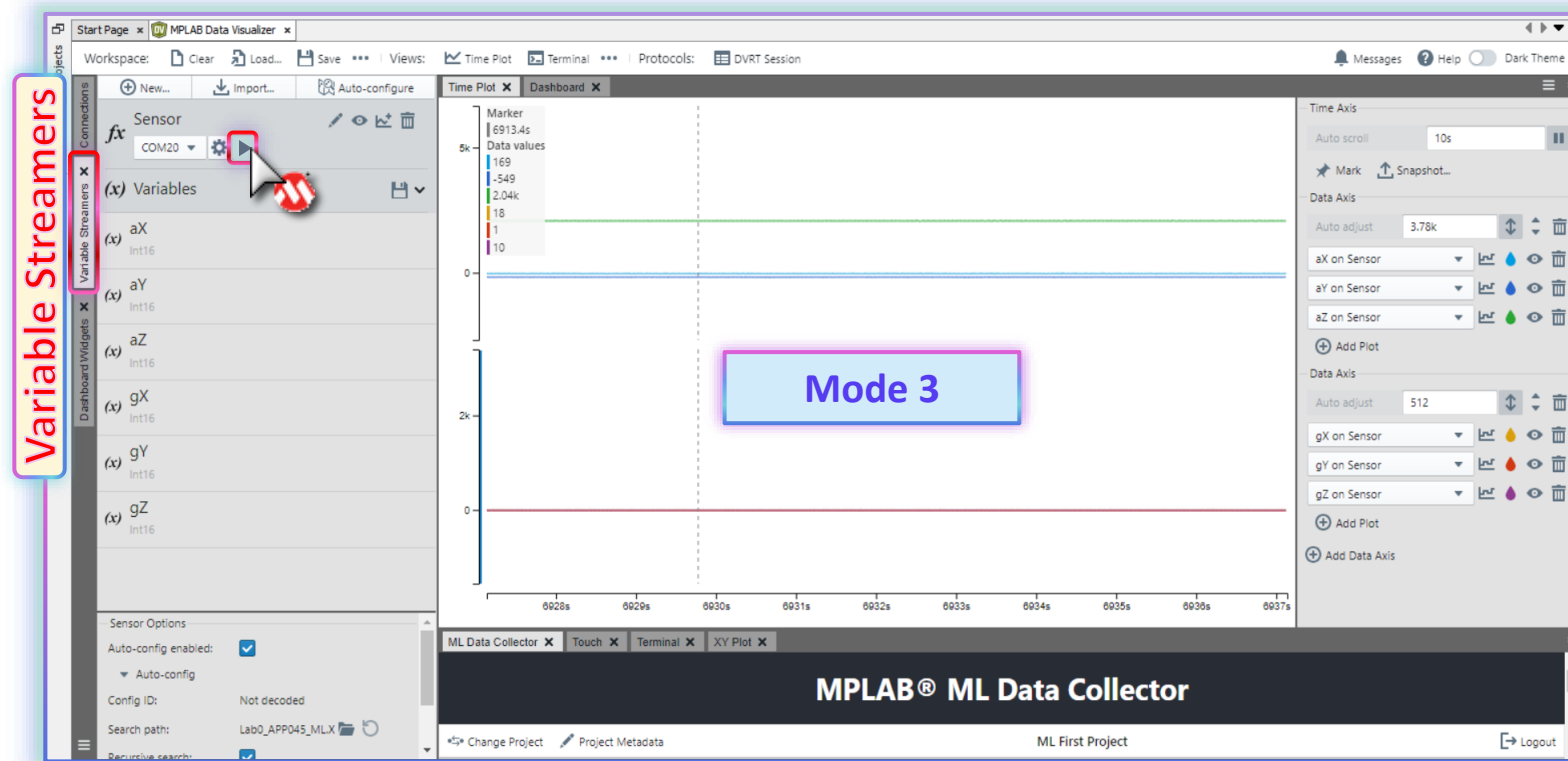


Time Plot

⚙️ Lab1 Initial Setup and Data Collecting

Step 10

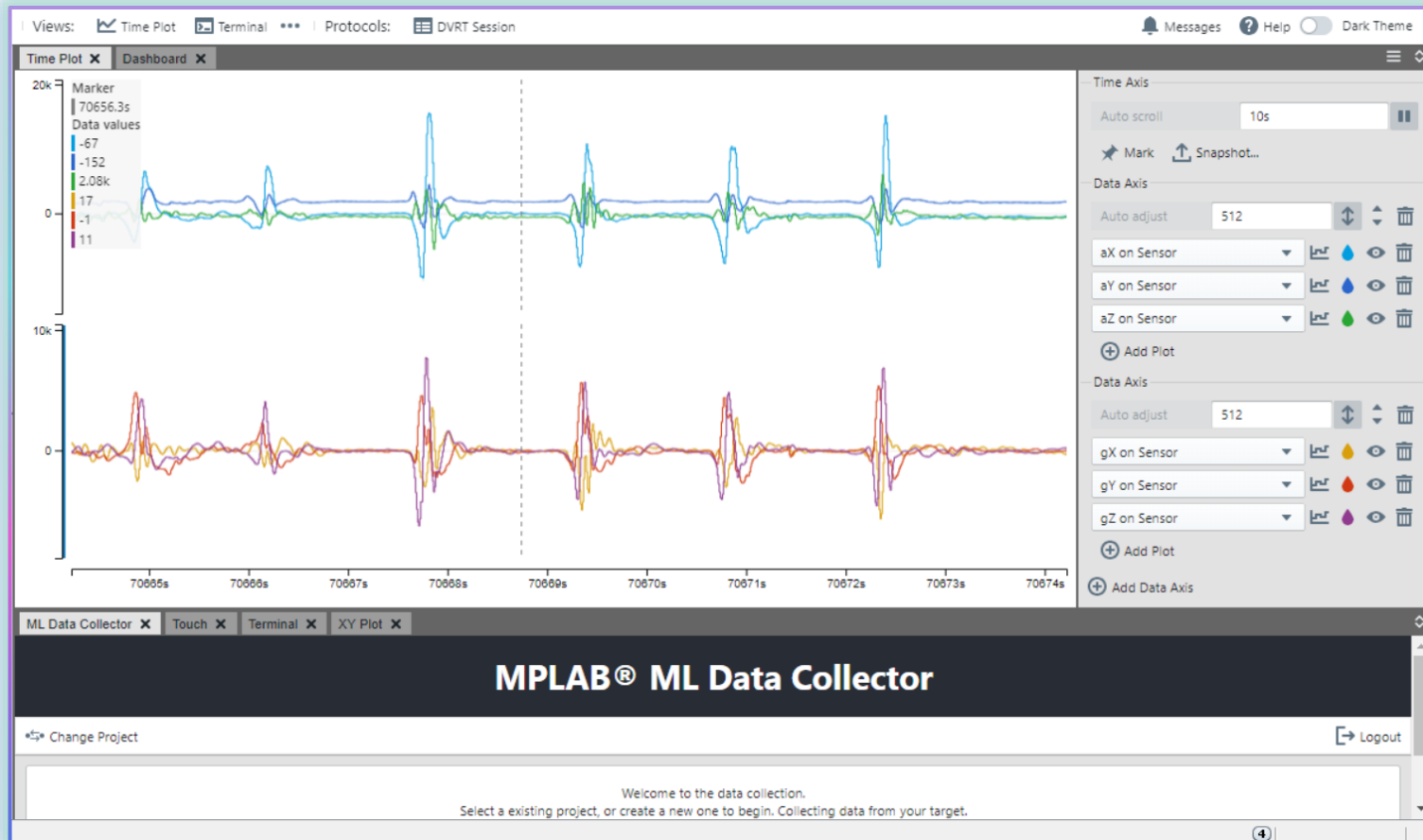
- Finally! We are going start collecting data.
- Press **BT2** two times to switch the mode to **Mode 3**.
- Start **Variable Streamers**, Click ► icon



⚙️ Lab1 Initial Setup and Data Collecting

Step 11

- Time Plot Graphic Window will show the raw data.



Time Axis

70677.1s 10s

Mark Snapshot...

Data Axis for Accelerometer

Auto adjust 512

aX on Sensor

aY on Sensor

aZ on Sensor

+

Data Axis for Gyroscope

Auto adjust 512

gX on Sensor

gY on Sensor

gZ on Sensor

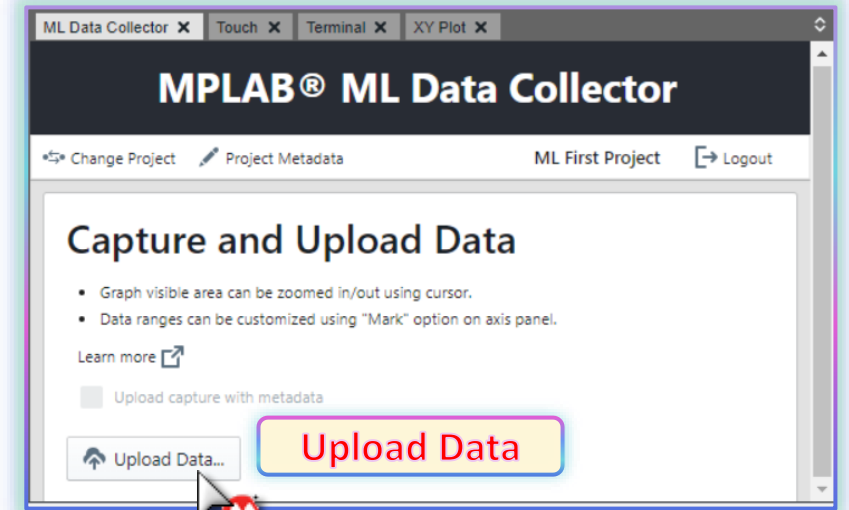
+

+

⚙️ Lab1 Initial Setup and Data Collecting

Two Ways to Capture and Upload Data

1. Upload and save Data by **MPLAB ML Data Collector**.
The data will be saved to **.csv** file in your local PC and upload to the ML Project you created **automatically**.



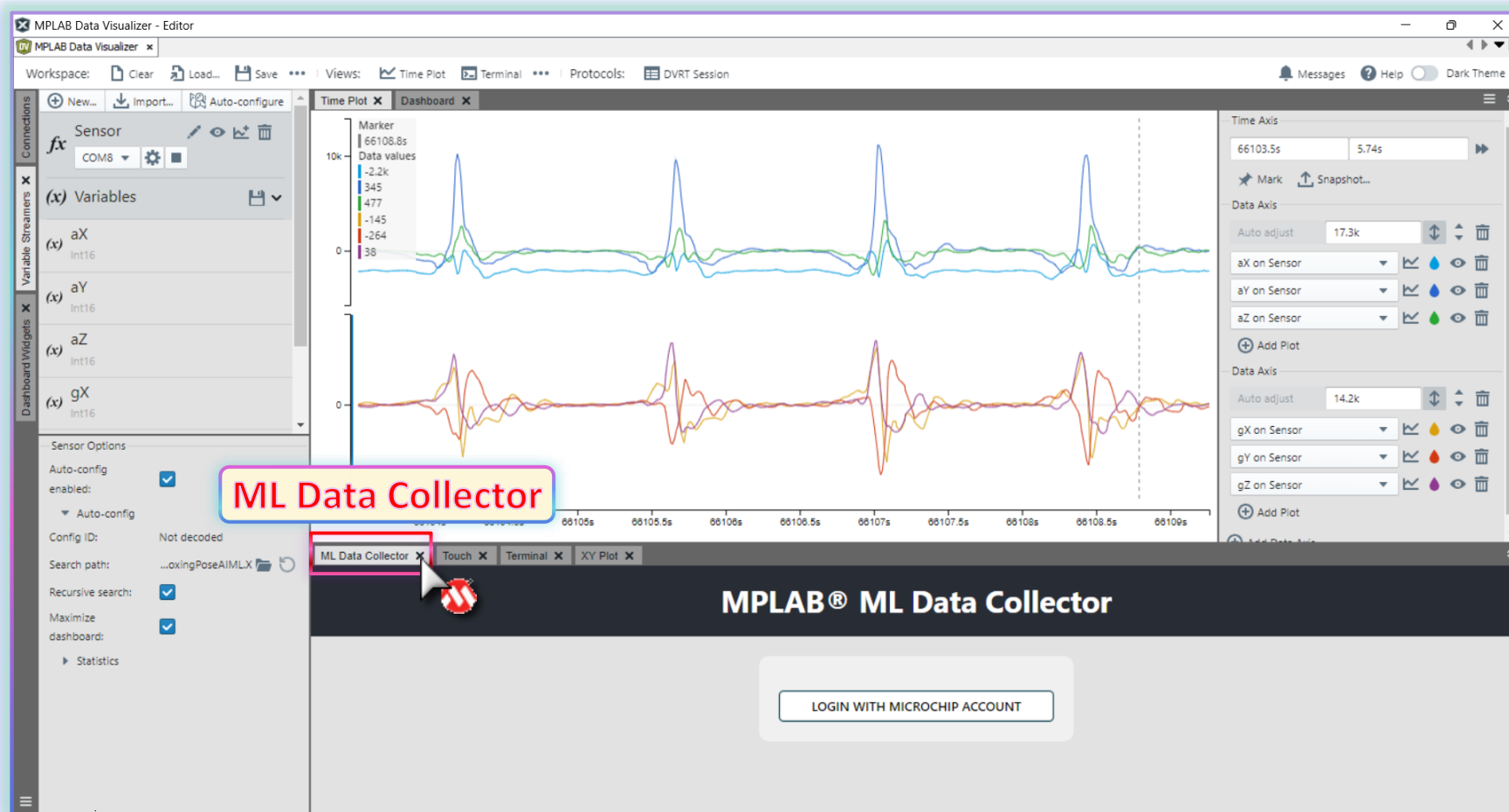
2. **Snapshot** Data by click on  in Time Plot of DV to save the **.csv** file in your local PC. Then upload the **.csv** file to the ML Project you created **manually**.



⚙️ Lab1 Initial Setup and Data Collecting

Step 12

- Choose **ML Data Collector** tab in the below window.

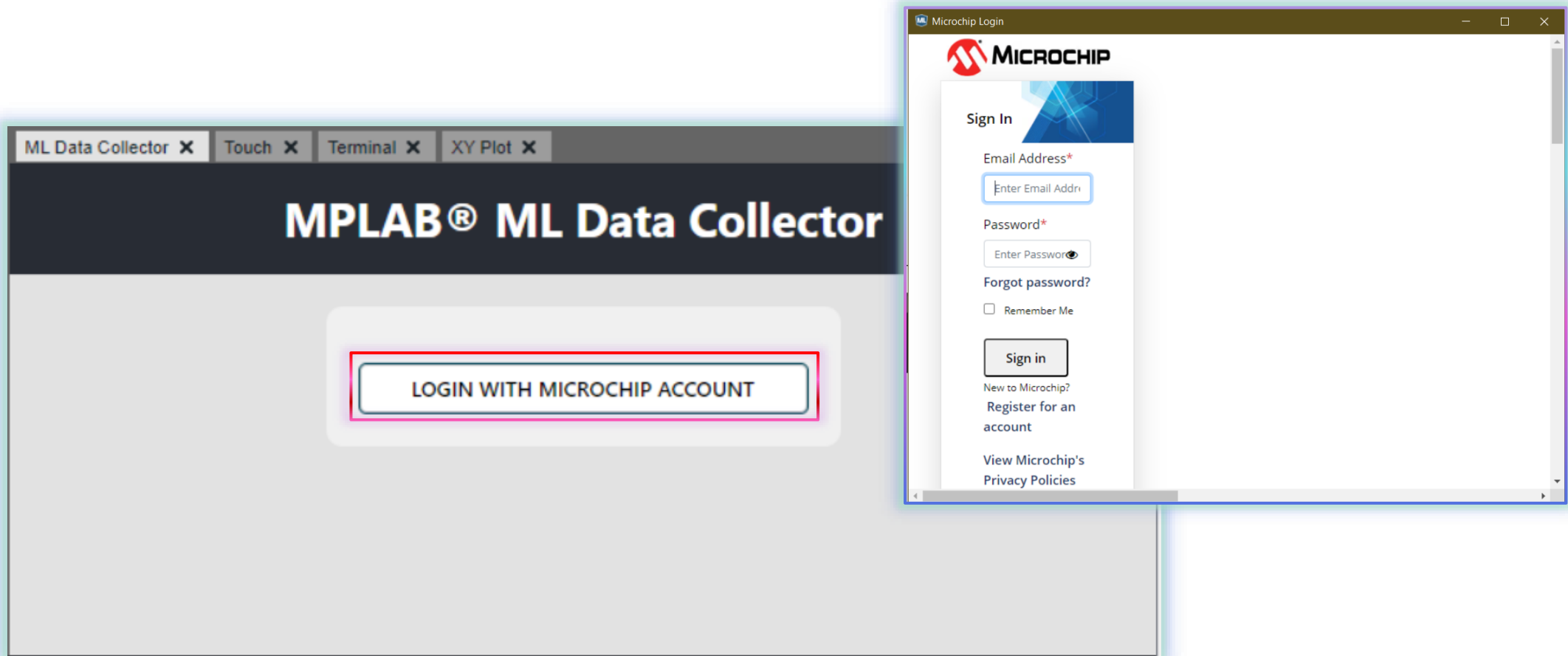


Appendix : Install the ML Data Collector & ML Model Builder

⚙️ Lab1 Initial Setup and Data Collecting

Step 13

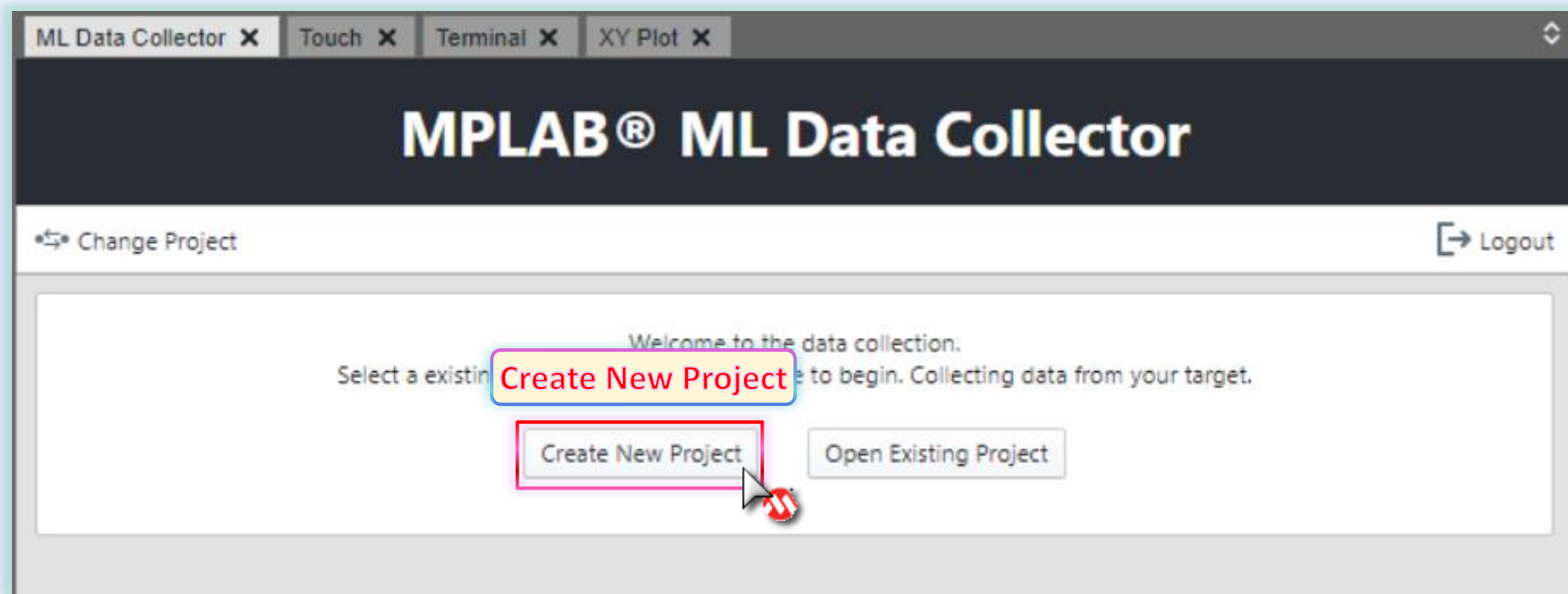
- Log in with your **myMicrochip** credentials in the MPLAB ML Data Collector tab.



⚙️ Lab1 Initial Setup and Data Collecting

Step 14

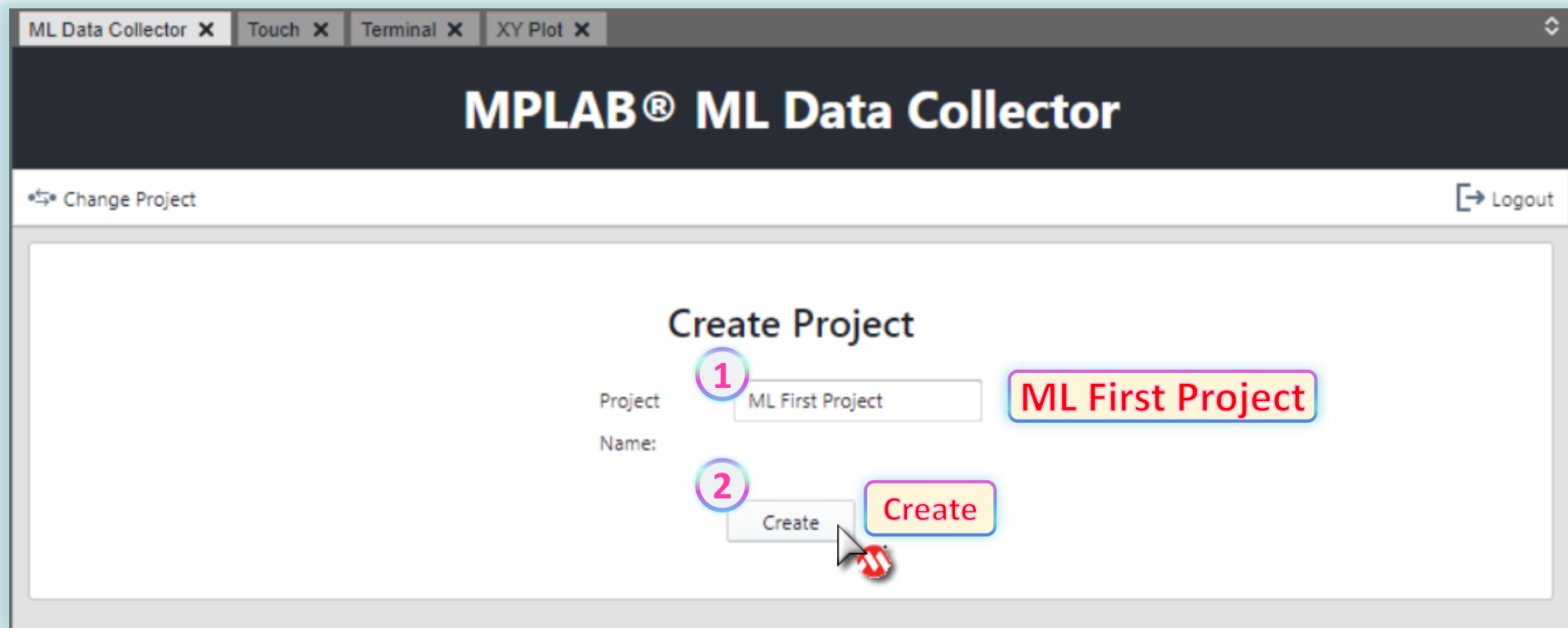
- Click the **Create New Project** button.



⚙️ Lab1 Initial Setup and Data Collecting

Step 15

- Input a name for your project.
Project Name: **ML First Project**

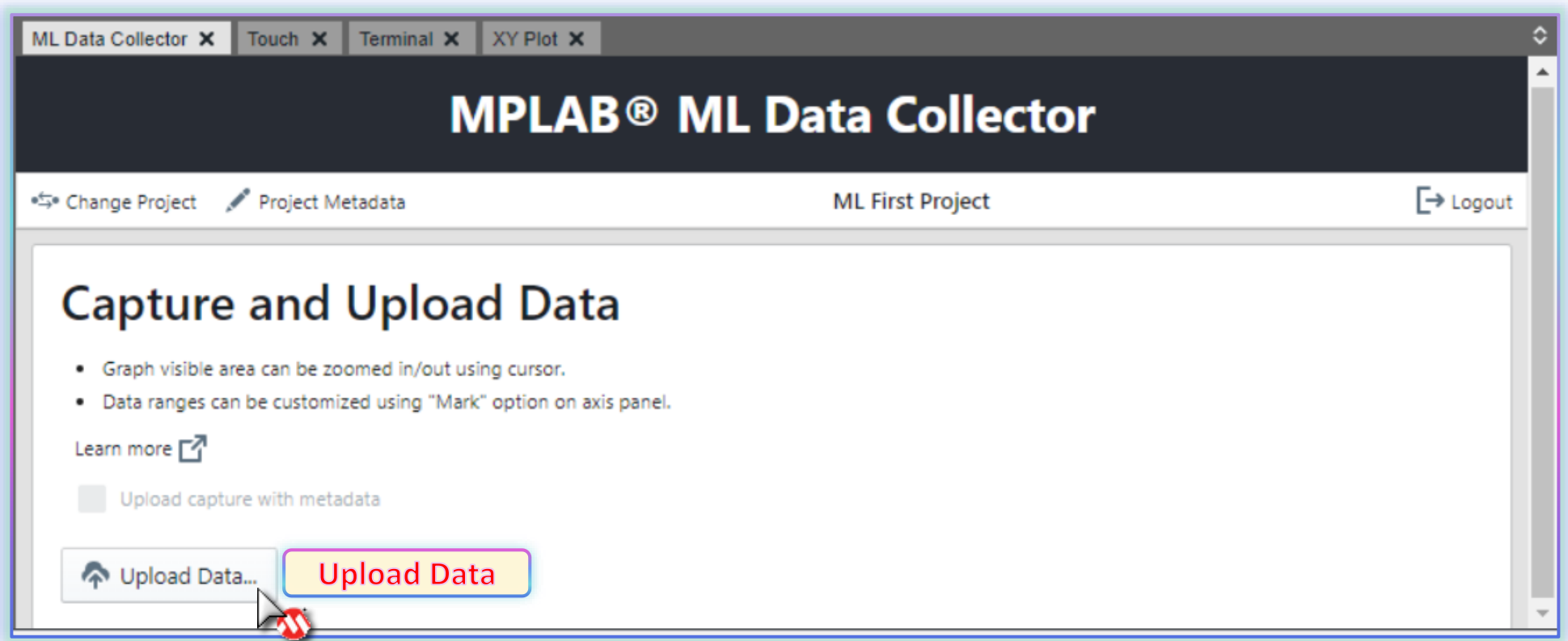


The screenshot shows the MPLAB ML Data Collector web interface. At the top, there are tabs for 'ML Data Collector', 'Touch', 'Terminal', and 'XY Plot'. The main header reads 'MPLAB® ML Data Collector'. Below the header, there is a 'Change Project' link and a 'Logout' button. The central area is titled 'Create Project'. It contains a form with a 'Project Name:' label, a text input field containing 'ML First Project', and a 'Create' button. A red box highlights the text 'ML First Project' in the input field. A red box also highlights the 'Create' button. A mouse cursor is pointing at the 'Create' button. There are two numbered callouts: '1' points to the input field, and '2' points to the 'Create' button.

⚙️ Lab1 Initial Setup and Data Collecting

Step 16

- Upload and save Data by ML Data Collector
- Click **Upload Data**



⚙️ Lab1 Initial Setup and Data Collecting

Step 17

- Profile Name: **Boxing ML**
- Device Name: **ICM42670P**
- Sample Rate (Hz): **100Hz**

ML Data Collector x Touch x Terminal x XY Plot x

MPLAB® ML Data Collector

Change Project Project Metadata ML First Project

Create Sensor Configuration

Profile Name: **Boxing ML**

Device Name: **ICM42670P**

Sample Rate (Hz): **100**

Sensor Names :

- aX
- aY
- aZ
- gX
- gY
- gZ

Back **Create** Create

Data Logging and Upload

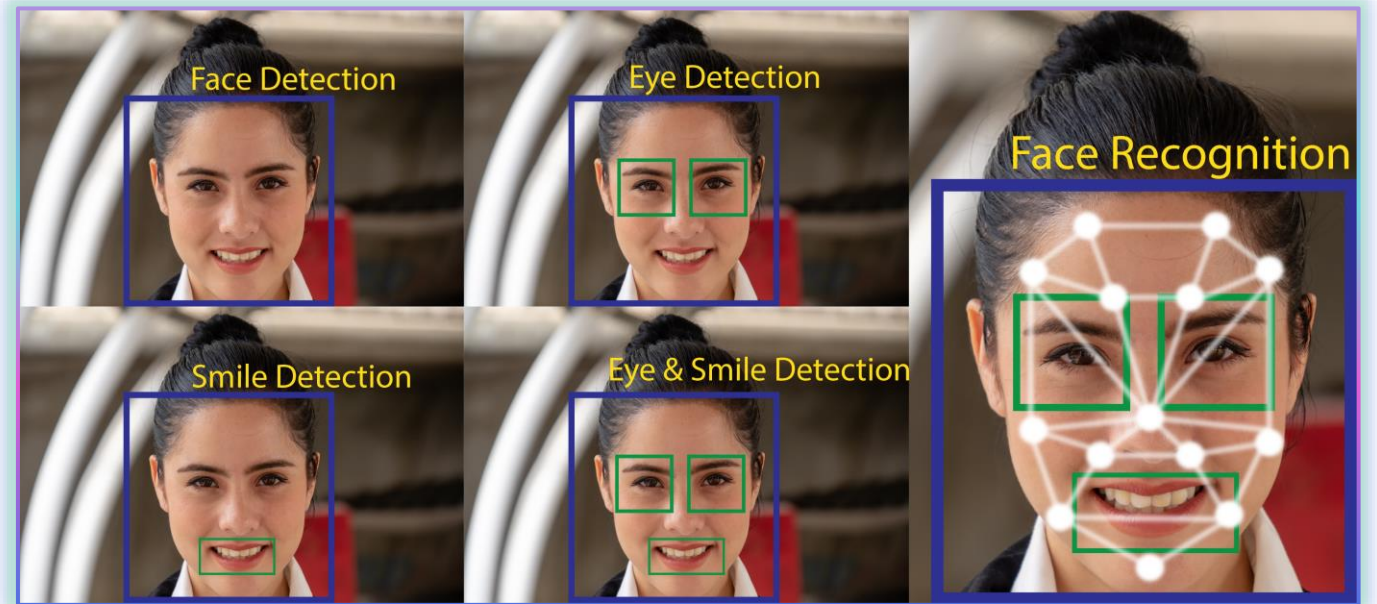
Select and Collect dataset

- **Data is the basis of model training.**

- Quantity
- Quality
- Diversity
- Performance evaluation
- Ethical

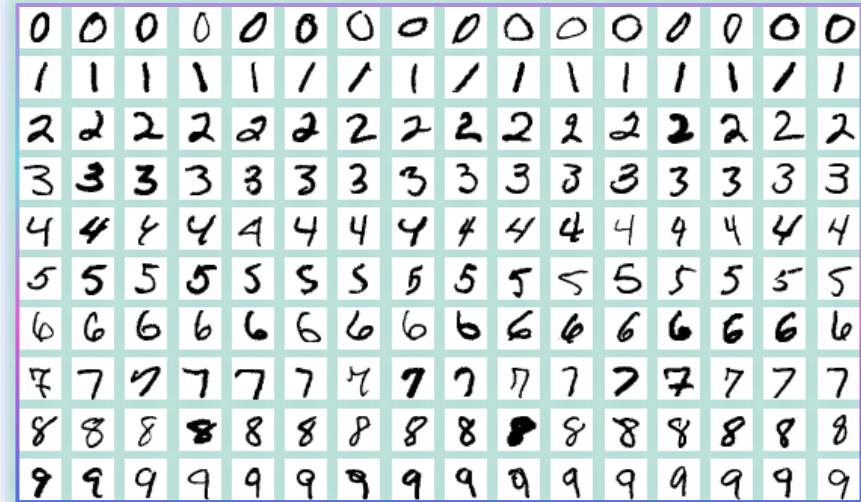
- **Where to get the data?**

- Acquire it
- Buy it
- Build it



Example MNIST

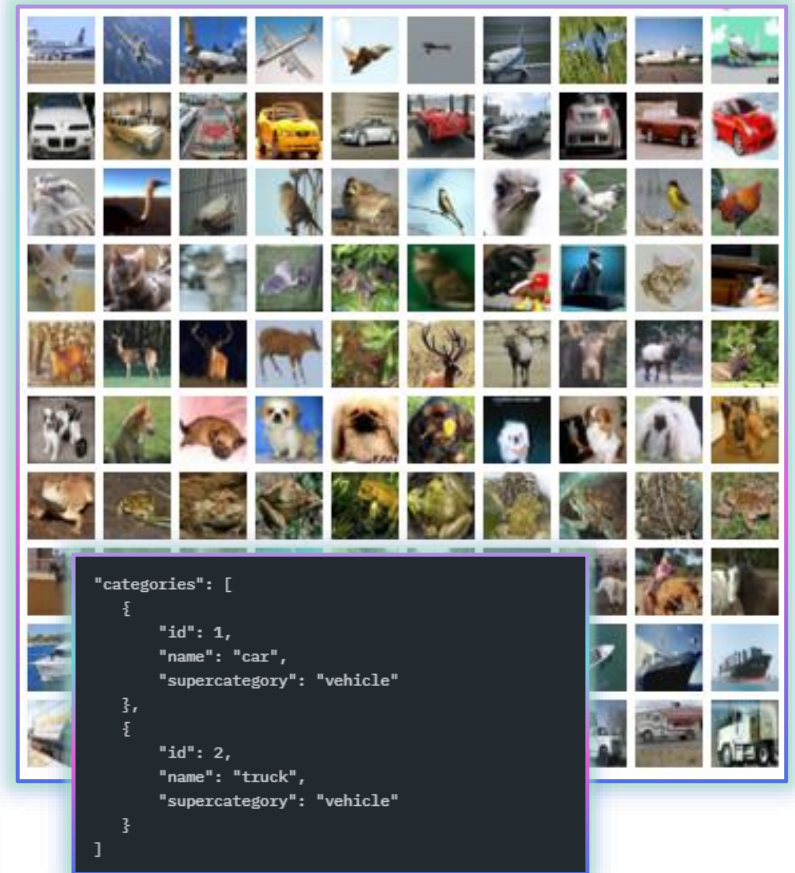
- **Collection of handwritten digits widely used for training and testing in the field of machine learning and computer vision**
- **70,000 images**
 - 60,000 training samples
 - 10,000 testing samples
- **Image Properties**
 - Grayscale images (0-255 levels)
 - Size of 28x28 pixels
- **Labels**
 - Each image is labelled with the digit it represents (0 through 9)
- **Benchmark for evaluating the performance of image processing systems**
- **Availability**
 - The dataset can be freely downloaded for use with various data science and machine learning libraries
- **Useful for creating Convolutional Neural Networks**



Example MS COCO

- The Microsoft Common Objects in Context – Object detection, segmentation, and caption. Pre-divided into training, validation, and test sets
- 80 object categories with a wide range from everyday scenes.
 - Diverse and Complex Scenes
 - Closely resemble real-world scenarios
- Detailed annotations
 - Object bounding boxes
 - Segmentation masks
 - Image captions
- Over 200,000 labelled images
- Image Captions
- Updated yearly
- Free for researchers

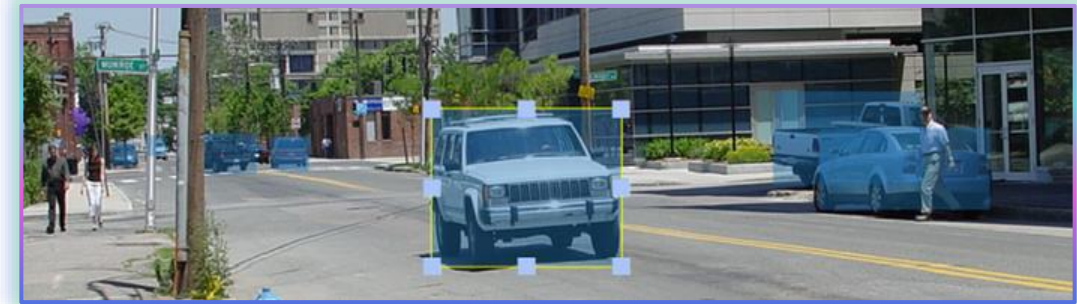
```
"annotations": [  
  {  
    "id": 1,  
    "image_id": 1,  
    "category_id": 1,  
    "segmentation": [  
      [  
        56.97,  
        56.97  
      ],  
      [  
        56.97,  
        56.97  
      ],  
      ...  
    ]  
  },  
  {  
    "area": 2351.45,  
    "bbox": [  
      56.97,  
      56.97,  
      56.97,  
      56.97  
    ],  
    "iscrowd": 0  
  },  
  ...  
]
```



Obtaining Datasets

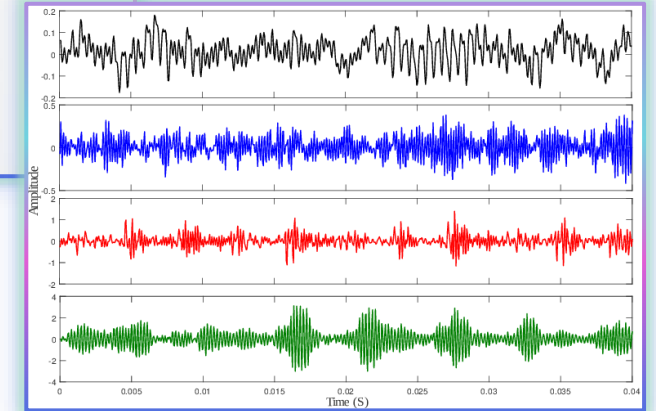
- **Limited number of *free* datasets**
 - Check licenses
- **Dataset Providers**
 - Kaggle (public datasets)
 - Subsidiary of Google
 - Companies that will create a dataset (for a fee)
 - Scale AI
 - MTurk (Amazon Mechanical Turk)
 - Crowdsourced marketplace for data labelling
 - Most focus on image analysis or Generative AI
- **Create your own**
 - Sometimes only viable option
 - New products/fields-of-play
 - Where data confidentiality is important
 - You know your data
 - *But you can be too close to it*

kaggle



Vibration Analysis

- **Bearing Data Center (CWRU)**
 - Vibration sensor samples at 12kHz
 - EDM Artificial Damage
- **Machinery Failure Data**
 - Extensive set of vibration readings
 - Common failure modes for broad range of machines
- **NASA Acoustics and Vibration Database**
 - Great for building rockets



🔧 Lab1 Initial Setup and Data Collecting

Step 18

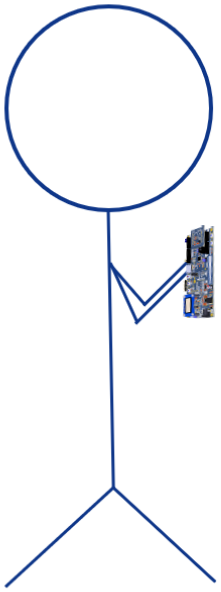
- Before starting capture data, you can first observe the results of different actions motion sensor.
- Ensure **uniformity** and **quality** in executing every action.
- Start moving your body, perform **Straight**, **Uppercuts**, and **right hooks** 15 times each, and save them into three different files.
- It is *recommended* that the length of each data file should not longer then **30** seconds while today's training.
- It is *recommended* perform **one movement per second**, and rest for about one second between each movement, and the rest interval can be regarded as an action- **Freeze**.
- Proper data organization and naming rules will facilitate data retrieval and analysis.

🔧 Lab1 Initial Setup and Data Collecting

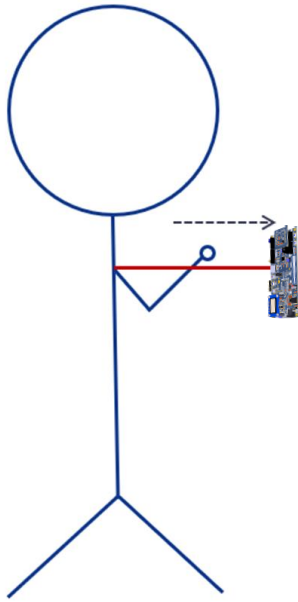
Step 19

- How to do Straight, Right hook, and Uppercut?

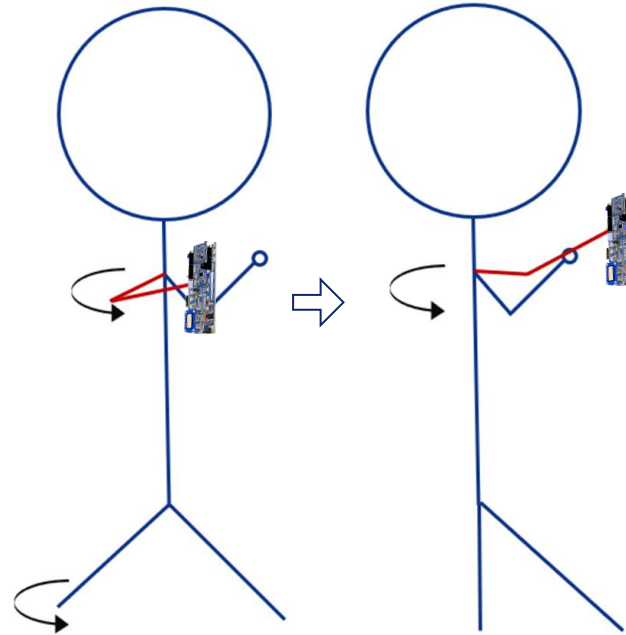
Freeze
(Ready Position)



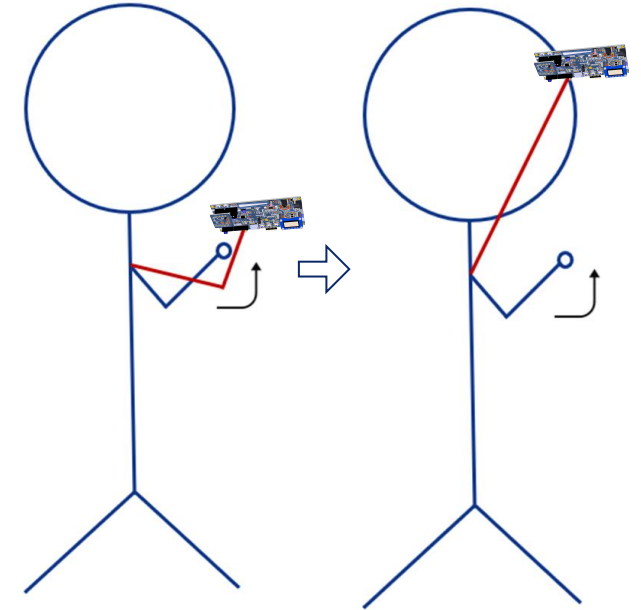
Straight



Right hook



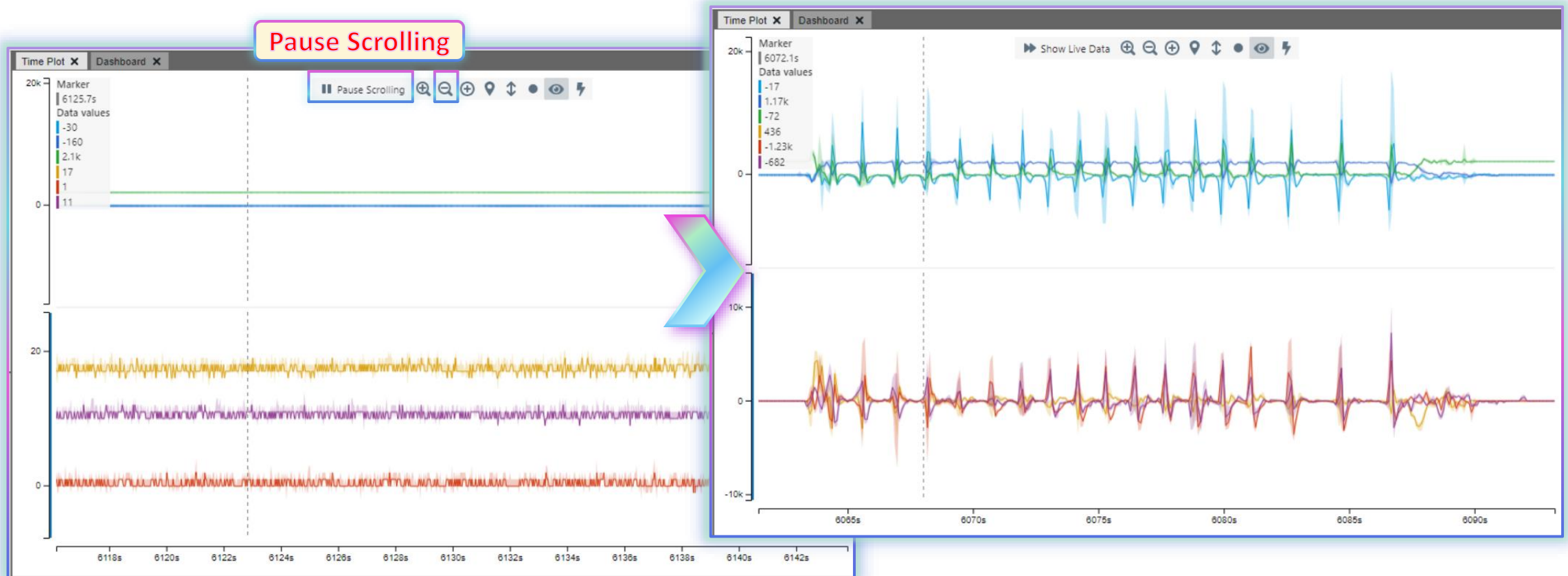
Uppercut



⚙️ Lab1 Initial Setup and Data Collecting

Step 20

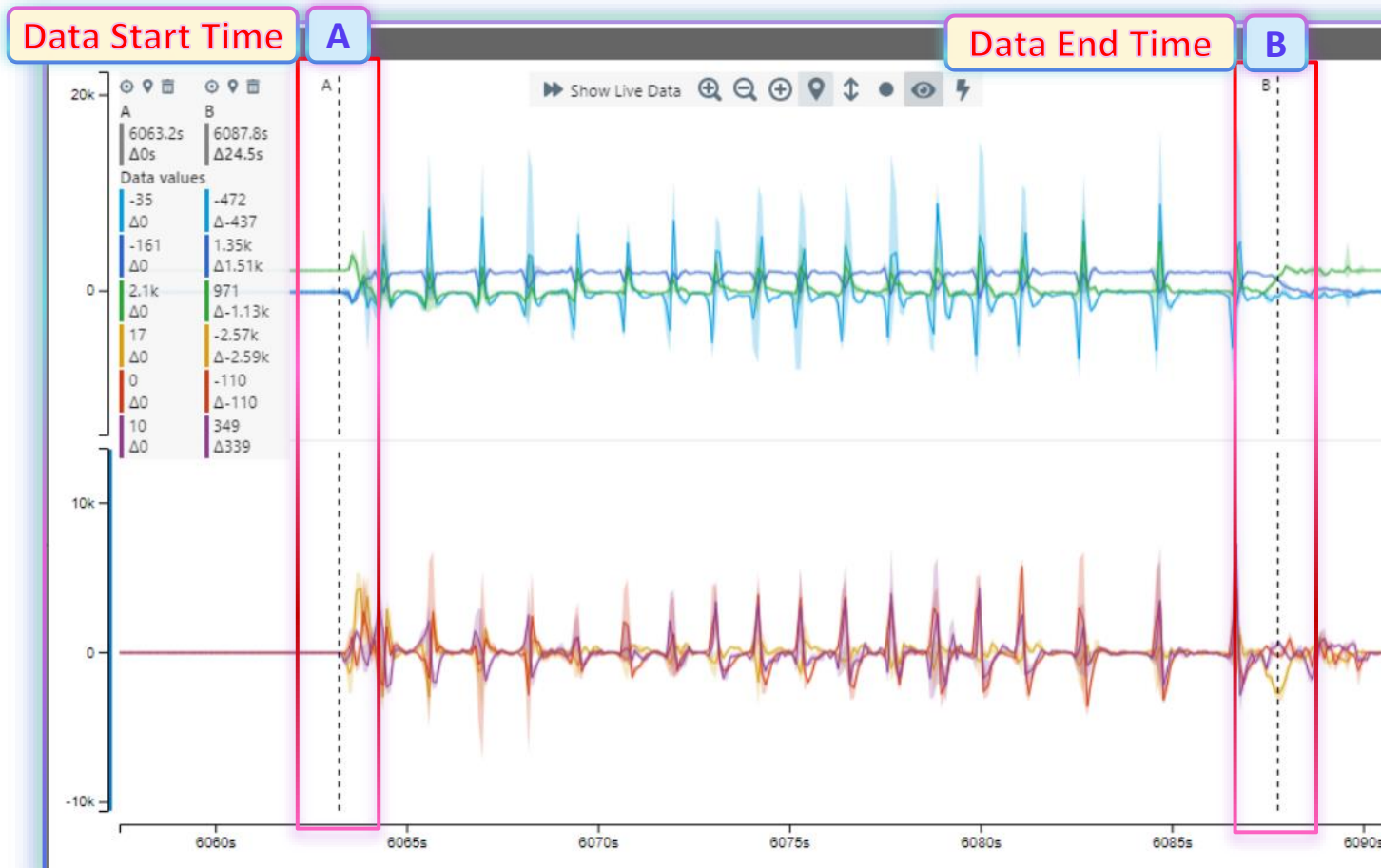
- Click **Pause Scrolling** and 🔍 to zoom out the window to show the all actions your do.



⚙️ Lab1 Initial Setup and Data Collecting

Step 21

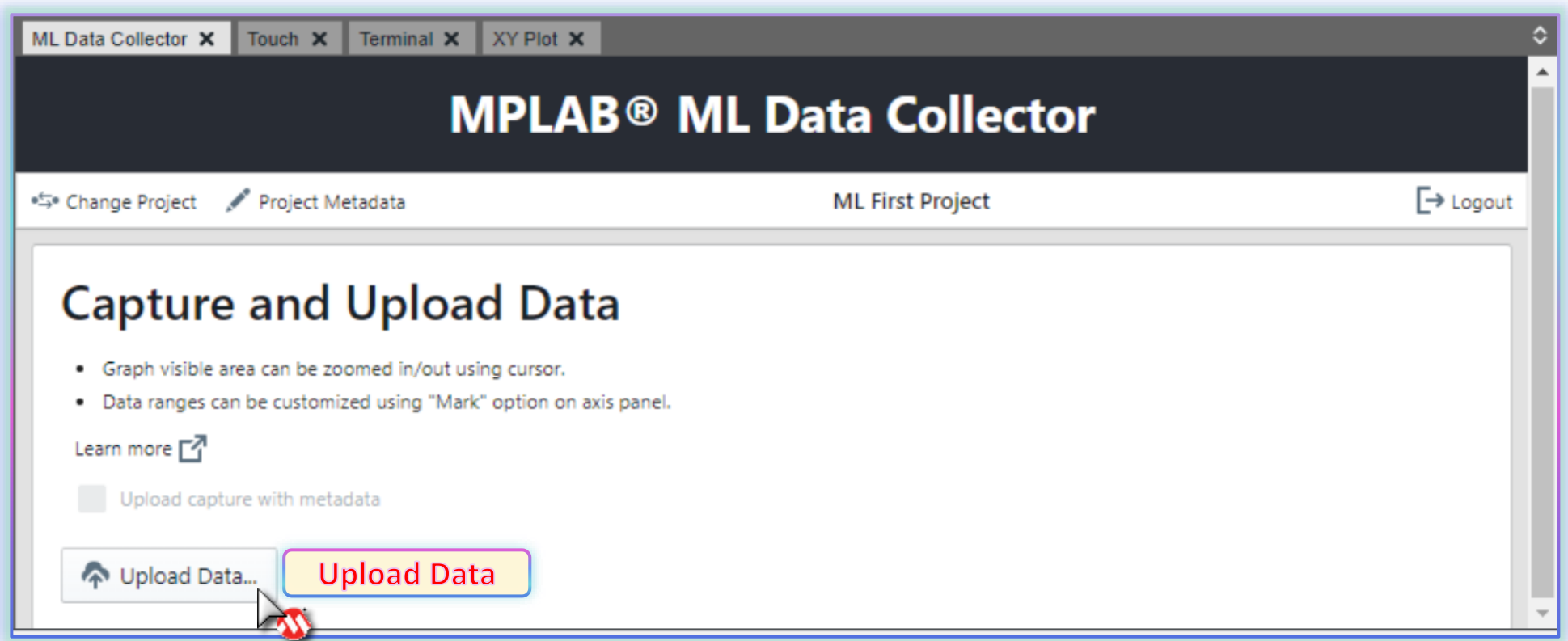
- Add two vertical cursor  and move to the time you start and stop the movements.



⚙️ Lab1 Initial Setup and Data Collecting

Step 22

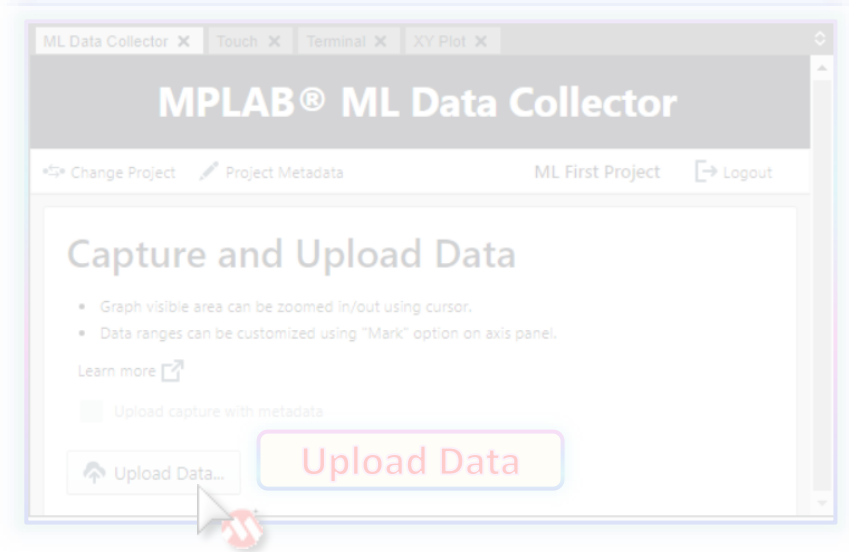
- Capture and Upload Data in ML Data Collector
- Click **Upload Data**



🔧 Lab1 Initial Setup and Data Collecting

Two Ways to Capture and Upload Data

1. Upload and save Data by **MPLAB ML Data Collector**. The data will be saved to **.csv** file in your local PC and upload to the ML Project you created **automatically**.



2. ✓ **Snapshot** Data by click on  in Time Plot of DV to save the **.csv** file in your local PC. Then upload the **.csv** file to the ML Project you created **manually**.



⚙️ Lab1 Initial Setup and Data Collecting

Step 23

- Enter the **time period** of the movement to take snapshot. (Use cursor A and B automatically)
- **Don't check the Include stamp and Allow scientific notation.**

Save Data Snapshot

Data Capture

Snapshot start (s): 7789.496832 **Data Start Time**

Window end (s): 7799.496832 **Data End Time**

Duration (s): 10 **Auto Generated**

Sources to include:

- ☐ All sources
- ☒ aX on Sensor
- ☒ aY on Sensor
- ☒ aZ on Sensor
- ☒ gX on Sensor
- ☒ gY on Sensor
- ☒ gZ on Sensor
- ☐ Touch Session on Touch
- ☐ COM20

2 Next



Save Data Snapshot

File Format

Preset: ML Upload

Format: ☒ CSV **3** **Don't check the Include stamp**

JSON

CSV options: ☐ Include timestamps

☒ Header row

Data value format: ☐ Allow scientific notation **4** **Don't check the Allow scientific notation**

Decimal places: 3

Data layout: ☒ Table ☐ Windowed

Preview

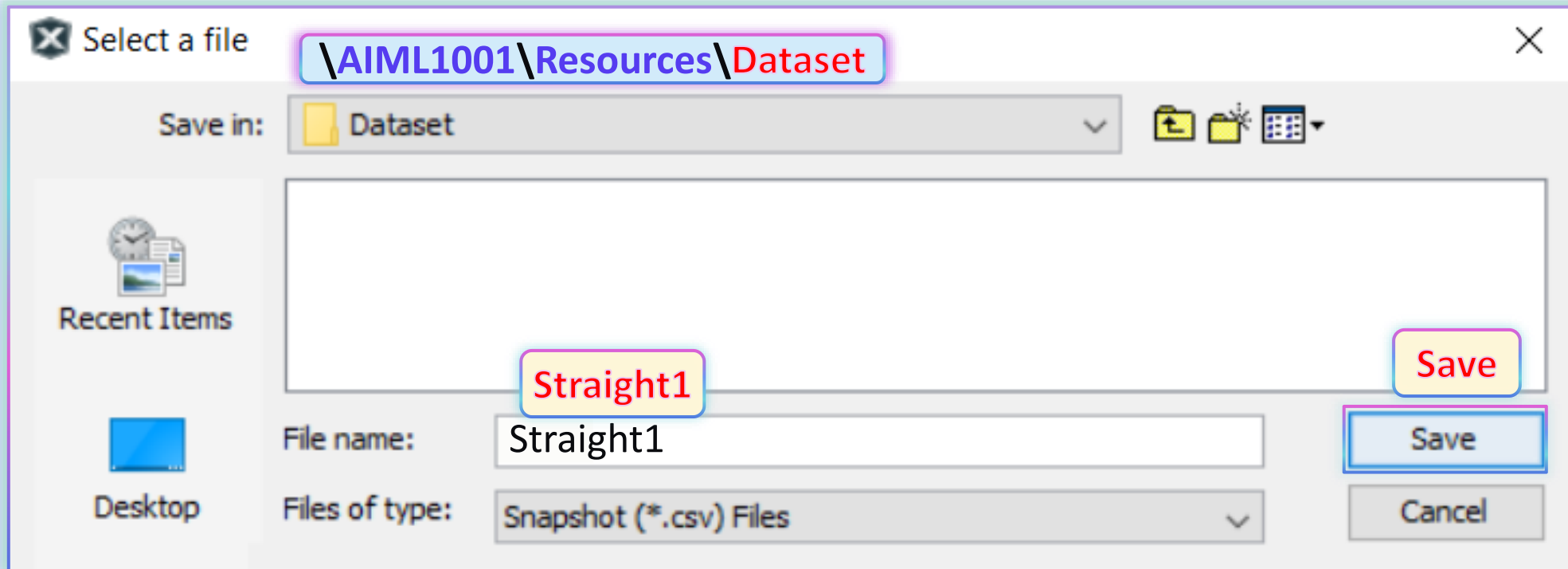
```
aX, aY, aZ, gX, gY, gZ
330, -2162, 151, -306, -62, -79
325, -2169, 168, -350, -90, -85
306, -2171, 185, -380, -106, -79
281, -2165, 197, -395, -93, -62
267, -2153, 209, -395, -61, -40
248, -2153, 214, -374, -17, -16
220, -2146, 215, -339, 35, 17
217, -2142, 217, -297, 76, 50
222, -2132, 206, -252, 100, 64
```

5 Save

⚙️ Lab1 Initial Setup and Data Collecting

Step 24

- Save the .csv files in folder “\AIML1001\Resources\Dataset”.
- Name the file as Straight1.



🔧 Lab1 Initial Setup and Data Collecting

Result

- Show File uploaded successfully after the dataset uploaded successfully to the cloud under your **myMicrochip** account.



- Repeat step 19 to step 24.

Build, Upload and Save all the data file.

- **Straight1**.CSV.
- **Uppercut1**.CSV.
- **Righthook1**.CSV.

Labeling and Prepare Data

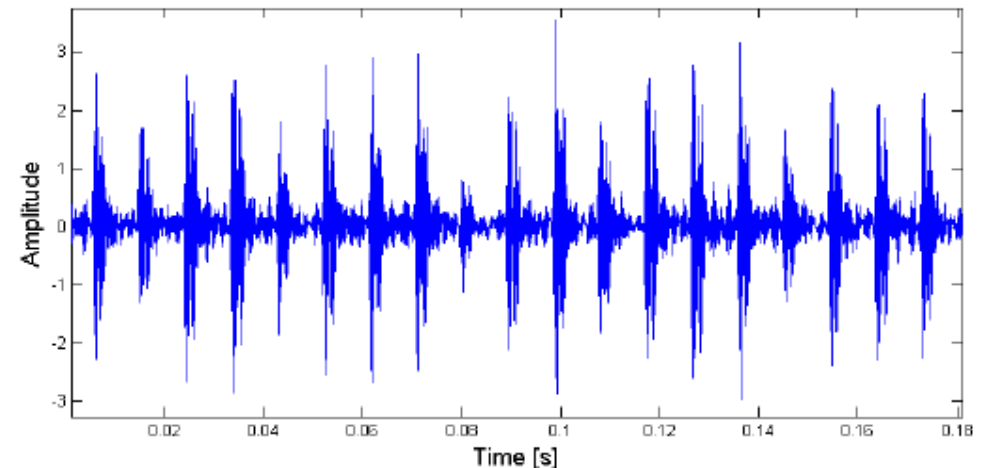
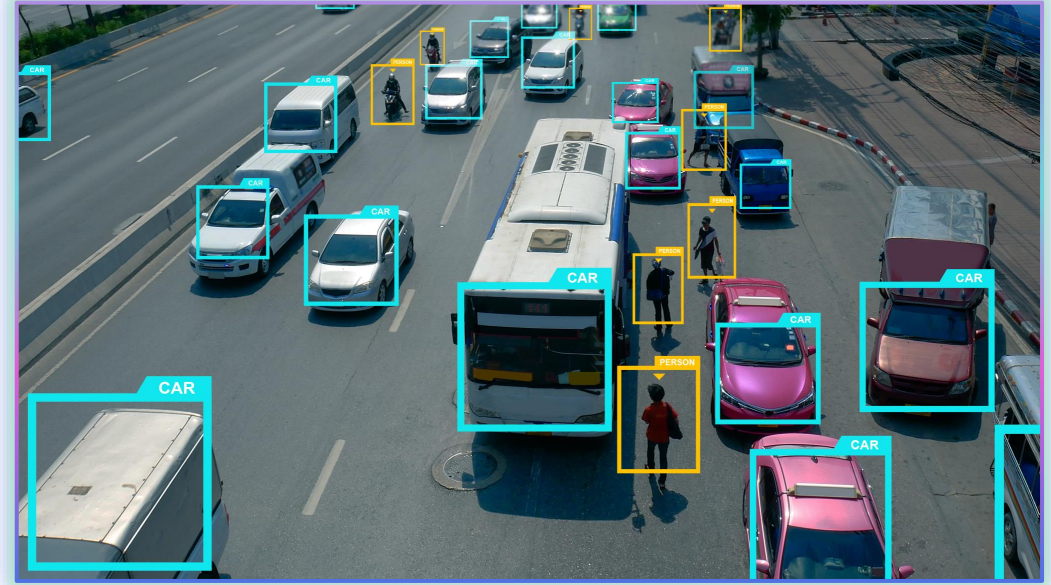
The importance of data

- **Google's Research Director Peter Norvig has been quoted many times on the value of more data in analytics:**
 - “We don't have better algorithms. We have more data.”
 - “Simple models and a lot of data trump more elaborate models based on fewer data.”



Organize and label data

- **How to Organize and label data?**
 - By Application
 - By Purpose
- **Things to note**
 - Consistency and Quality
 - Quality Control of Labeled Data
 - Marking tools and platforms



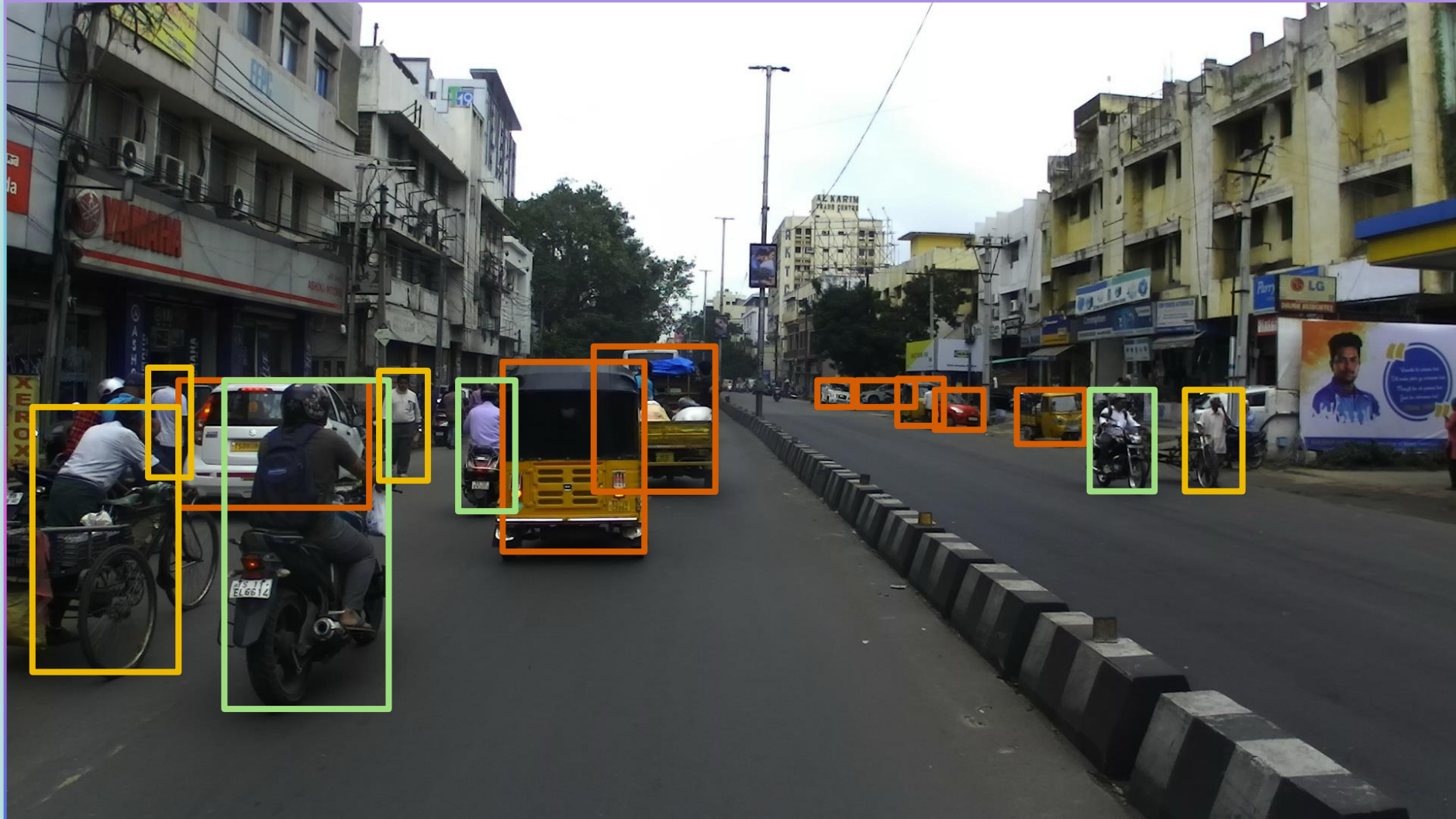
What do you see in this picture?

- What will you label this Picture ?



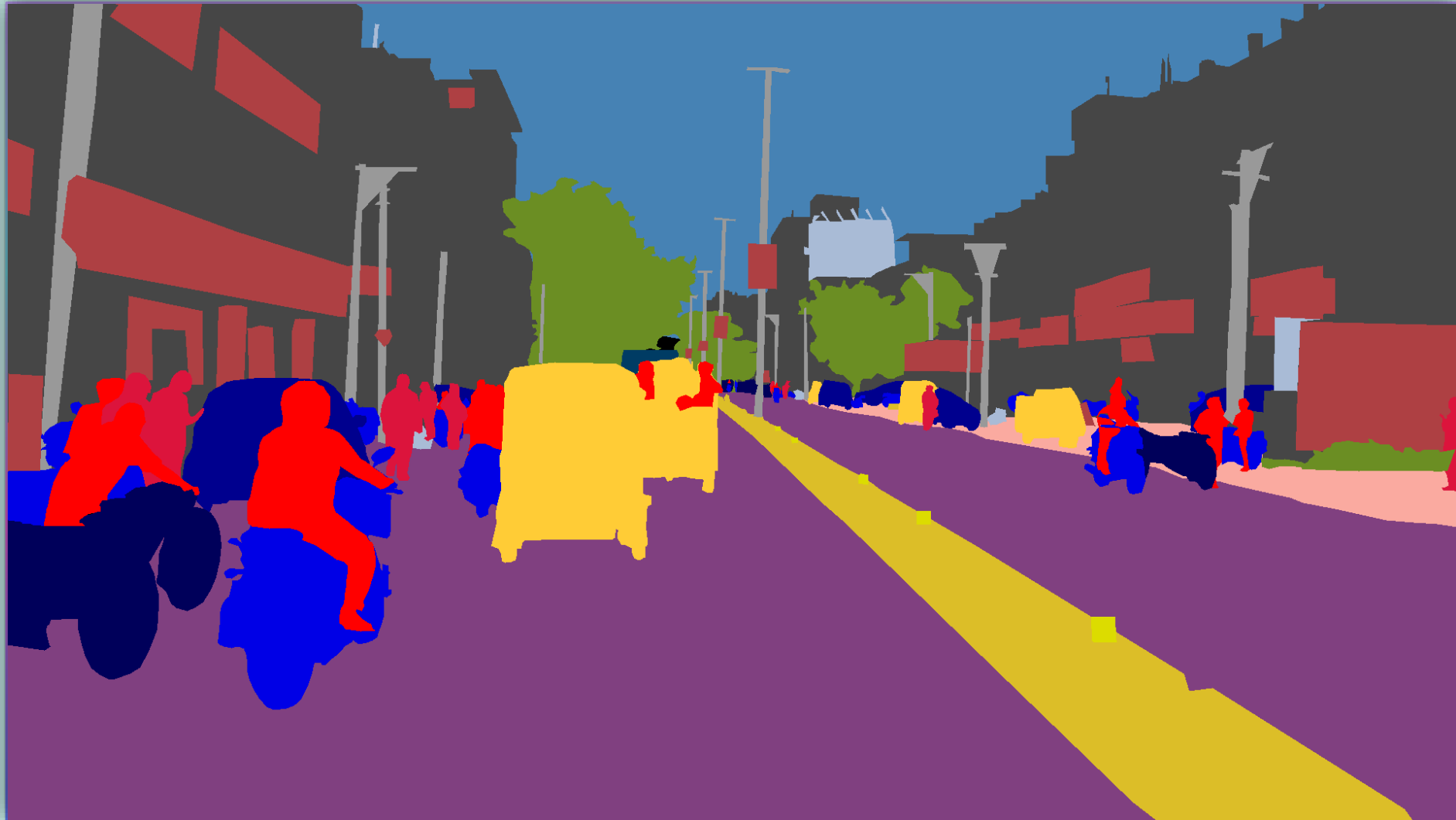
What do you see in this picture?

- By Application - In Object Detection...



What do you see in this picture?

- By Application - In Semantic Segmentation...



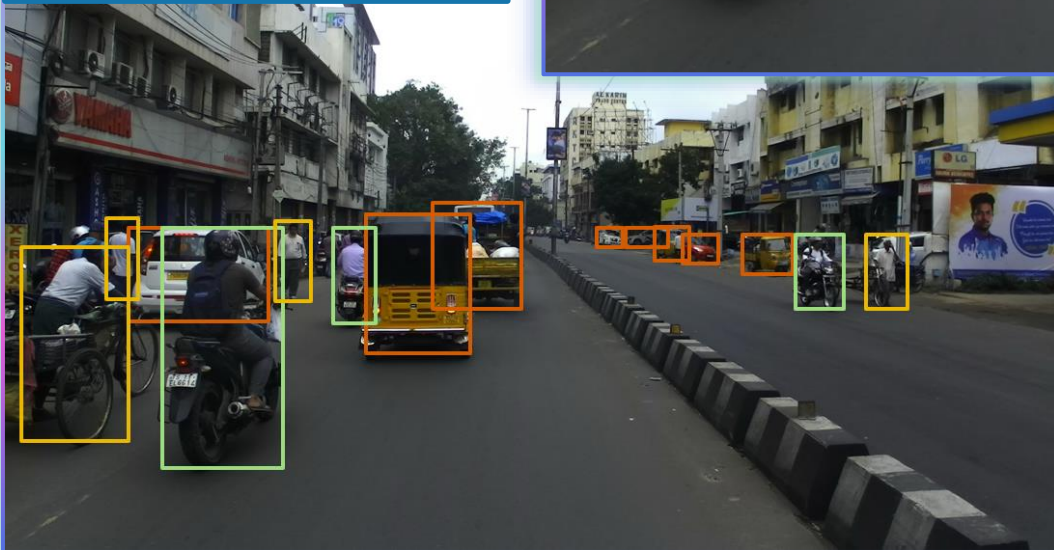
What do you see in this picture?

- By Application

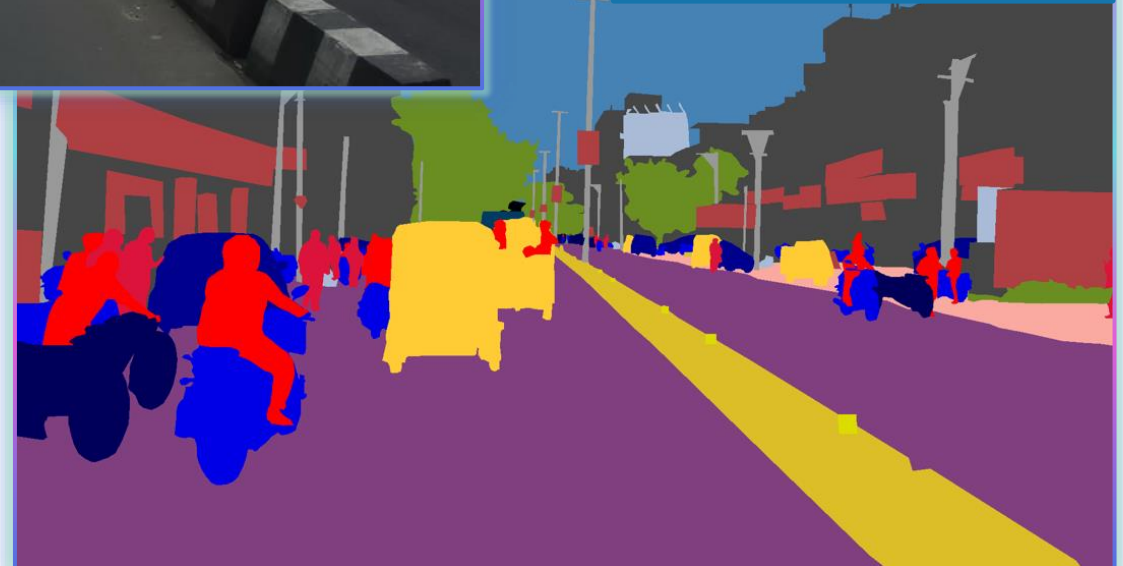
Original Photo



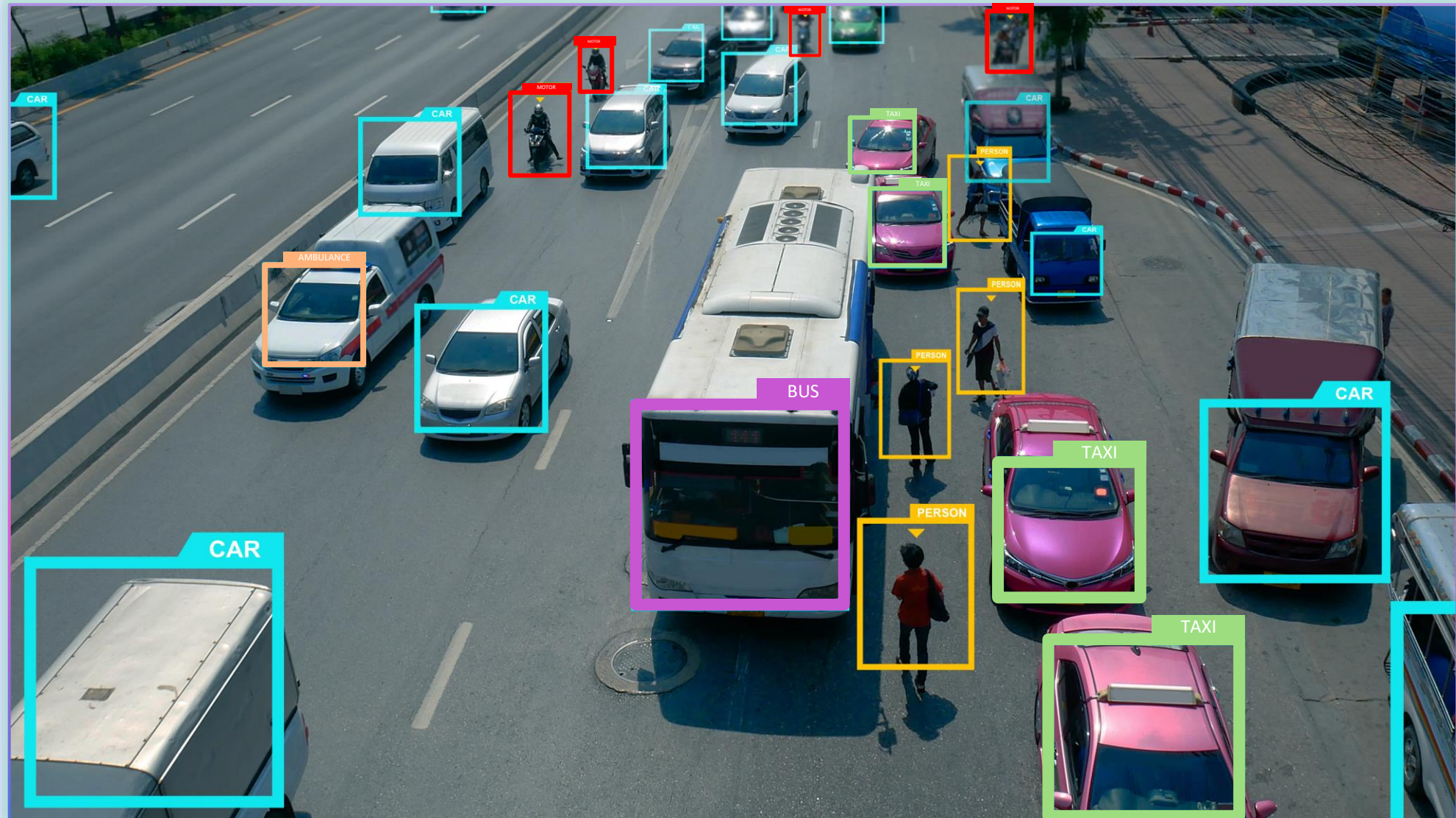
Object Detection



Semantic Segmentation



What will you select in this picture?



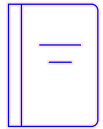
Segment Label and Metadata

- **Segment Label:** Describe the types of our events.
 - The outputs from the classifier.
- **Metadata:** Data about the data. Provide the description of the dataset and define custom information that can be used in training.
 - Data quality management
 - Feature engineering
 - Model training and evaluation
 - Model tracking and version management



Segment Label and Metadata

- Take 2025 Winter Elite training materials as example...



Metadata

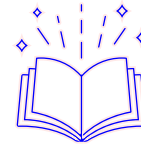
Title: 2025 Winter Elite Training Materials

Author: Microchip Taiwan CAEs

Publisher: Microchip

Publication date: 2025/02/10

Chapter: 11



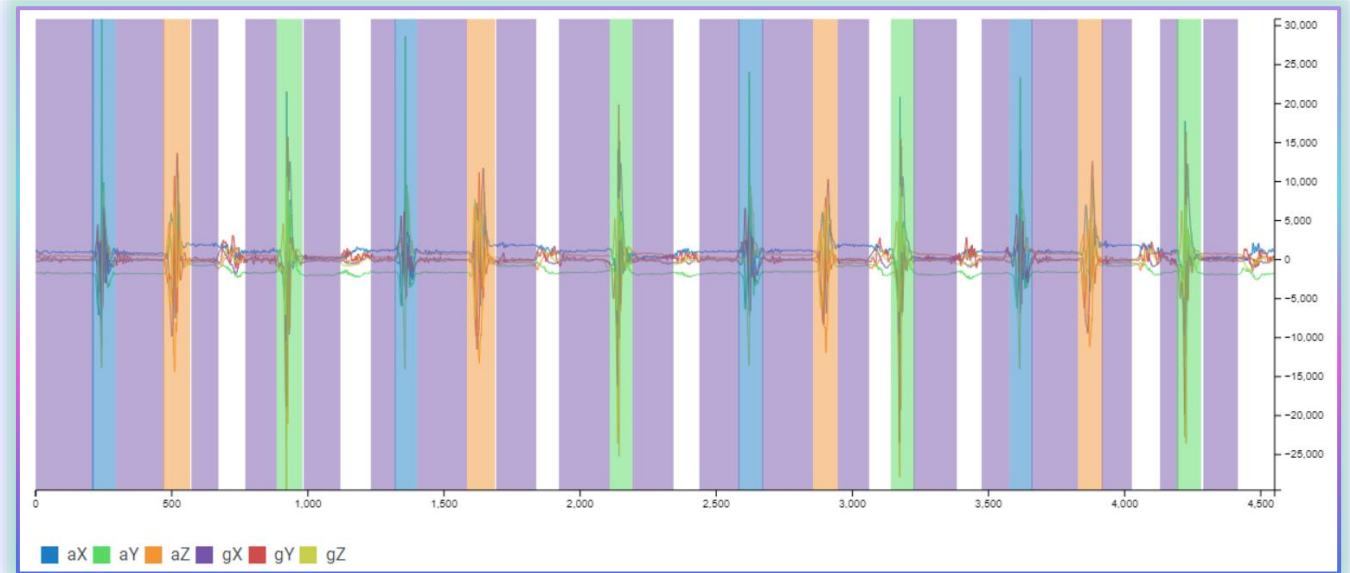
Segment

Catalog:

- Section 1: SAMD21 ARM Cortex-M0+ Microcontroller Introduction
- ...
- Section 10: SERCOM - SPI - OLED SSD1306
- Section 11: Machine Learning Boxing Recognition

Example Data of Boxing Recognition

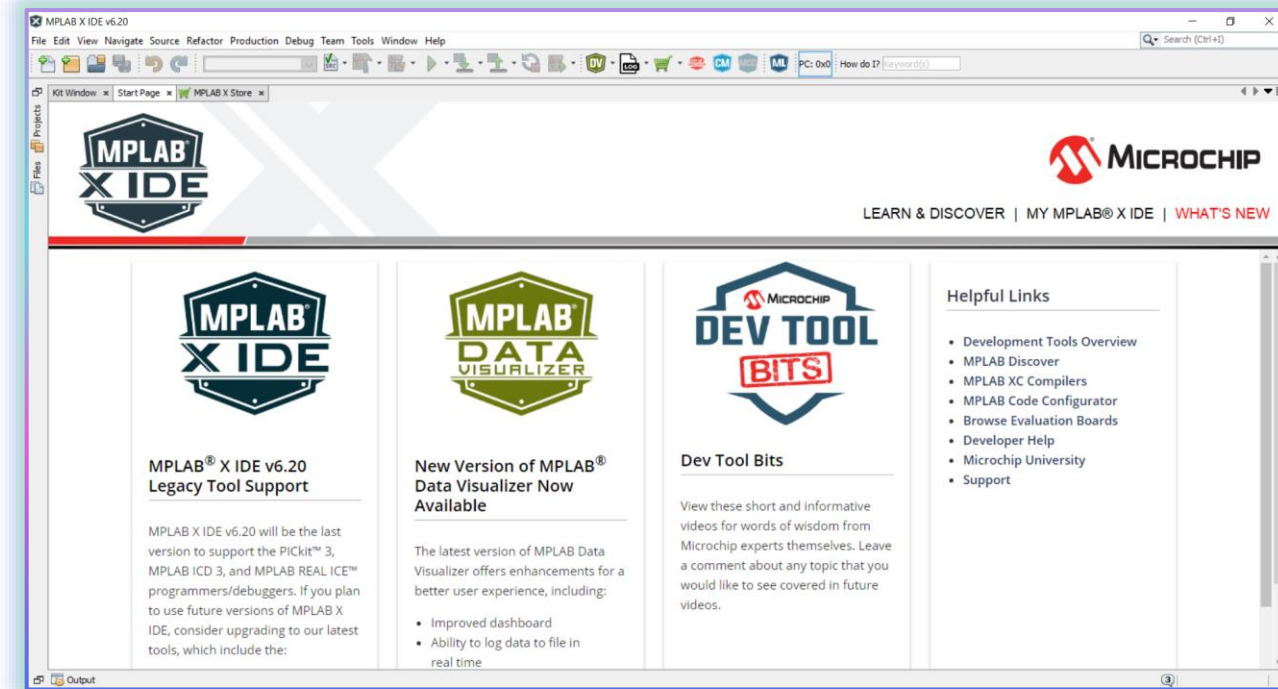
- **Boxing demo**
 - Accelerometer (AX, AY, AZ)
 - Gyroscope (GX, GY, GZ)
- **Multiple Events**
 - Each signal has 'quiet' spells
 - Amplitude, spectrum vary
- **Metadata**
 - Dominant hand
 - Gender



MPLAB[®] Machine Learning Development Suite

MPLAB® Machine Learning Development Suite

- The MPLAB ML Model Builder plug-in is an application that filters and optimizes your labeled sensor data through machine learning algorithms. It generates a model (Knowledge Pack) ready to be flashed into the firmware of your device of choice.



MPLAB® Machine Learning Development Suite

- Create compact supervised and anomaly-detection algorithms that can run on edge based MCUs and MPUs
- Automate each step of the ML flow process.
 - Data preparation
 - Label and filter data
 - Feature extraction
 - Training
 - Validation
 - Deployment
- Generate AutoML AutoML-powered code
 - Create a prototype in days or weeks (or 3 hours)
- Fine-tune aspects of your algorithm
 - Advanced users can develop with Google Colab or Jupiter Notebooks
- Firmware generation for 8-, 16- and 32-bit XC compilers
 - Library and Source Code export options



MPLAB® Machine Learning Development Suite

Licensing Information for MPLAB ML Development Suit

	Evaluation License	Pro License
Price	Free	\$99/year
Supported Silicon	Microchip 8-, 16-, 32-bit MCUs and dsPIC® DSCs	
Data	1 GB	Up to 10 GB
Labels	2500	Unlimited
Output	Library	Source Code
AutoML CPU Time	5H/month	250H
Application	Evaluation Only	Production
Where to Purchase/Install	Free MPLAB X IDE Plug-in	SW67G42A

The Workspace of ML Development suite

- Home Page of ML Development suite

MICROCHIP MPLAB ML Model Builder

Project Search

Open Project [CREATE PROJECT](#)

	NAME	FILES	PIPELINES	SIZE (MB)	QUERIES	MODELS	SEGMENTS	CREATED DATE	DELETE
	ML First Project	9	1	1.55	1	20	285 of 2500	5/30/2024	

Rows per page: 10 1-1 of 1

Last opened project: ML First Project

The Workspace of ML Development suite

- Each step of the ML flow process.

The screenshot displays the Microchip ML Development suite workspace. On the left, a sidebar menu is highlighted with a red border, listing the following steps: Change Project, Project Summary, Data Manager, Prepare Data, Build Model, Explore Model, Test Model, and Download Model. Below these are links for Get Started, Documentation, and Support. The main area shows the 'Project: ML First Project' overview. It includes a 'CHANGE PROJECT' button in the top right. A navigation bar at the top lists OVERVIEW, QUERIES, PIPELINES, MODELS, and PROJECT SETTINGS. The overview section features a large Microchip logo and a summary table:

Captures	Queries	Pipelines	Models
9	1	1	20

Segments	Size(MB)
285	1.55

Created Date: 5/30/2024 4:53:08 PM

UUID: 0b53f3c6-b4f9-4cdc-8087-e5197cf38045

Sensors: aX, aY, aZ, gX, gY, gZ

The Workspace of ML Development suite

- The Overview, Queries, Pipelines, Models, and Project Settings of the project.

The screenshot displays the Microchip ML Development suite workspace. The interface includes a left sidebar with navigation options: Change Project, Project Summary, Data Manager, Prepare Data, Build Model, Explore Model, Test Model, Download Model, Get Started, Documentation, and Support. The main area is titled "Project: ML First Project" and features a "CHANGE PROJECT" button. Below the title, there are tabs for Overview, Queries, Pipelines, Models, and Project Settings. The Overview tab is active, showing a summary of project metrics and details. The metrics section includes Captures (9), Queries (1), Pipelines (1), and Models (20). Below this, the Segments section shows 285 segments. The Created Date is 5/30/2024 4:53:08 PM, and the UUID is 0b53f3c6-b4f9-4cdc-8087-e5197cf38045. The Sensors section lists aX, aY, aZ, gX, gY, and gZ.

Captures	Queries	Pipelines	Models
9	1	1	20

Segments	Size(MB)
285	1.55

Created Date
5/30/2024 4:53:08 PM

UUID
0b53f3c6-b4f9-4cdc-8087-e5197cf38045

Sensors
aX, aY, aZ, gX, gY, gZ



Data Labeling and Preparation

Boxing Recognition by Machine Learning

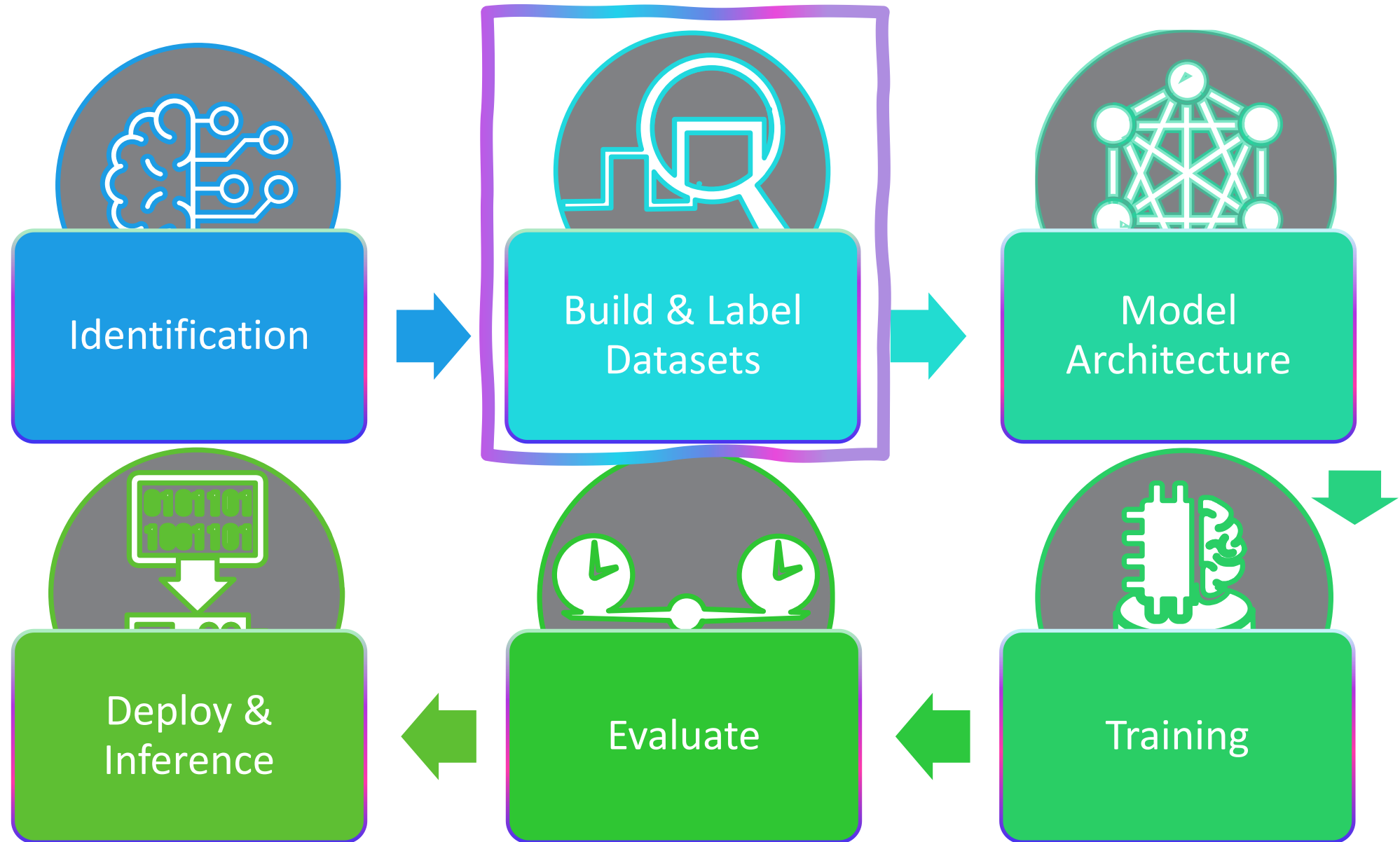
Lab2 Data Labeling and Preparation

Boxing Recognition by Machine Learning

- Try to segment and label the data. This will be a lengthy process but is also the most important part to get the best results.
- Start the ML Builder plugin and log into your account.
- Opened the project you created and add 4 labels of the boxing movements to the project.
 - Straight
 - Uppercut
 - Right hook
 - Freeze
- Label the data for the model generation.

Let's go!

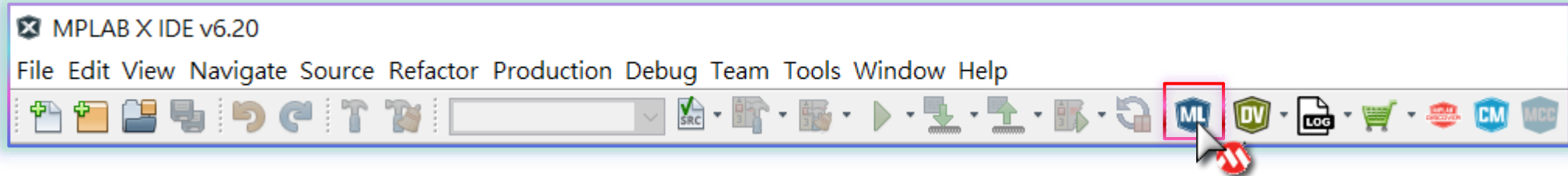
Machine Learning Workflow



⚙️ Lab2 Data Labeling and Preparation

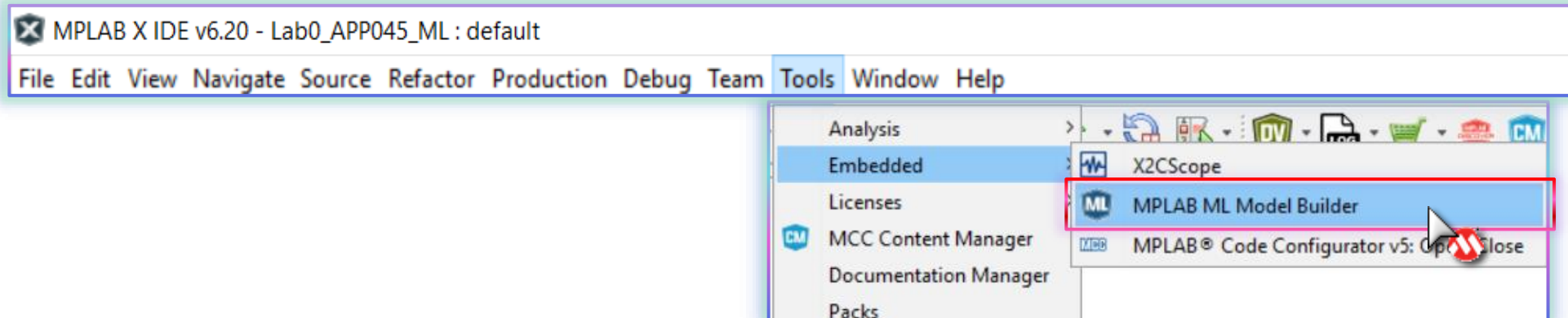
Step 1

- Click  icon



OR

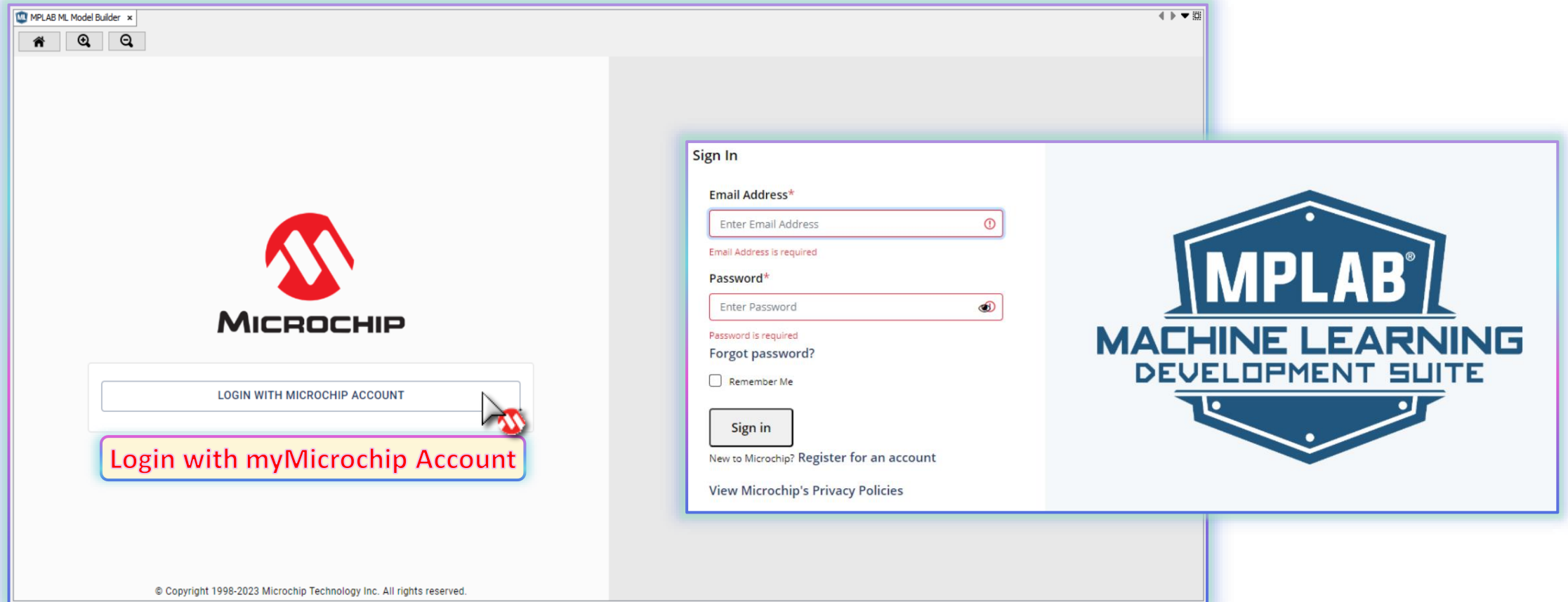
- Tools > Embedded > **MPLAB ML Model Builder**



⚙️ Lab2 Data Labeling and Preparation

Step 2

- Log in with your **myMicrochip** account.

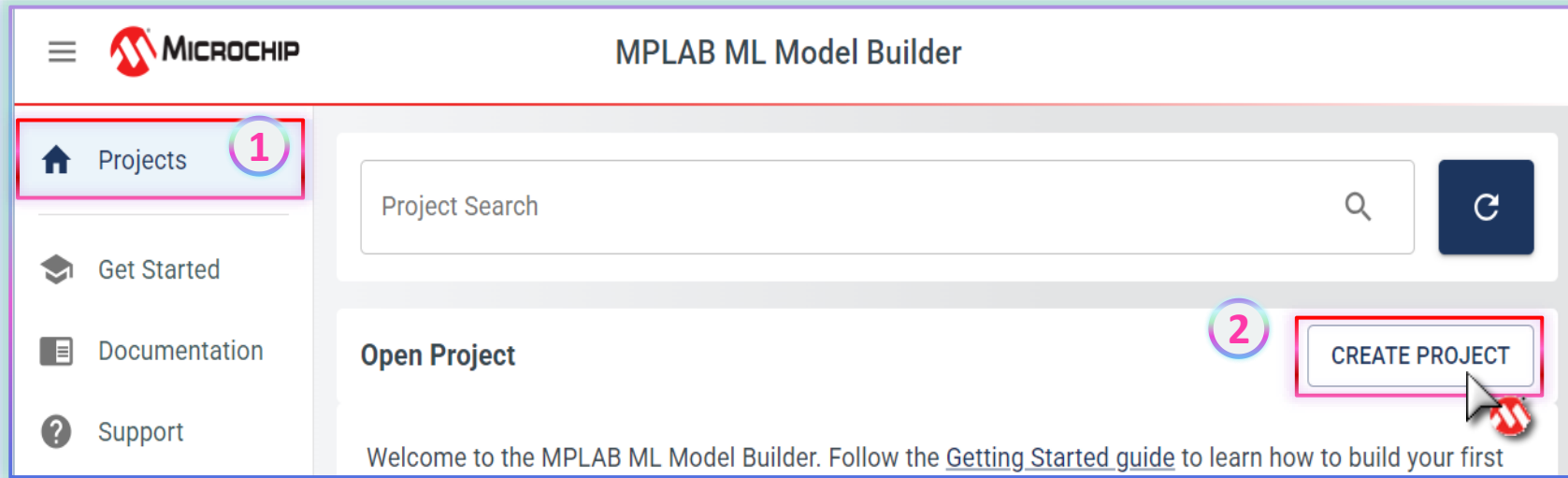


The screenshot displays the MPLAB ML Model Builder interface. On the left, the Microchip logo is visible above a button labeled "LOGIN WITH MICROCHIP ACCOUNT". A red callout box with the text "Login with myMicrochip Account" points to this button. On the right, a "Sign In" modal is open, featuring input fields for "Email Address*" and "Password*", a "Forgot password?" link, a "Remember Me" checkbox, and a "Sign in" button. Below the button, there are links for "New to Microchip? Register for an account" and "View Microchip's Privacy Policies". The background of the modal shows the "MPLAB MACHINE LEARNING DEVELOPMENT SUITE" logo.

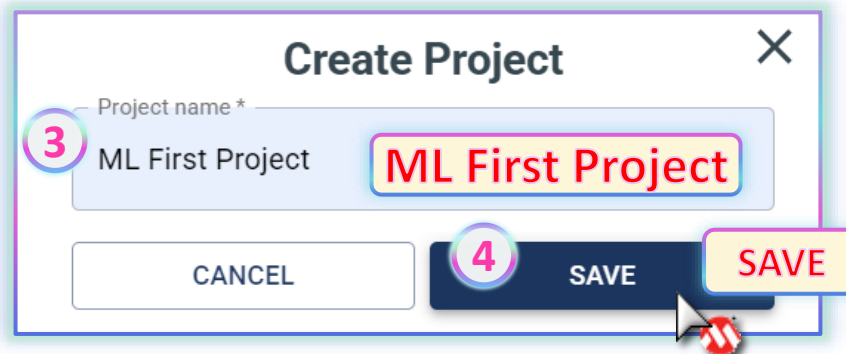
⚙️ Lab2 Data Labeling and Preparation

Create a New Project in ML Model Builder

- Click on **CREATE PROJECT**, if you have not created one using ML Data Logger.



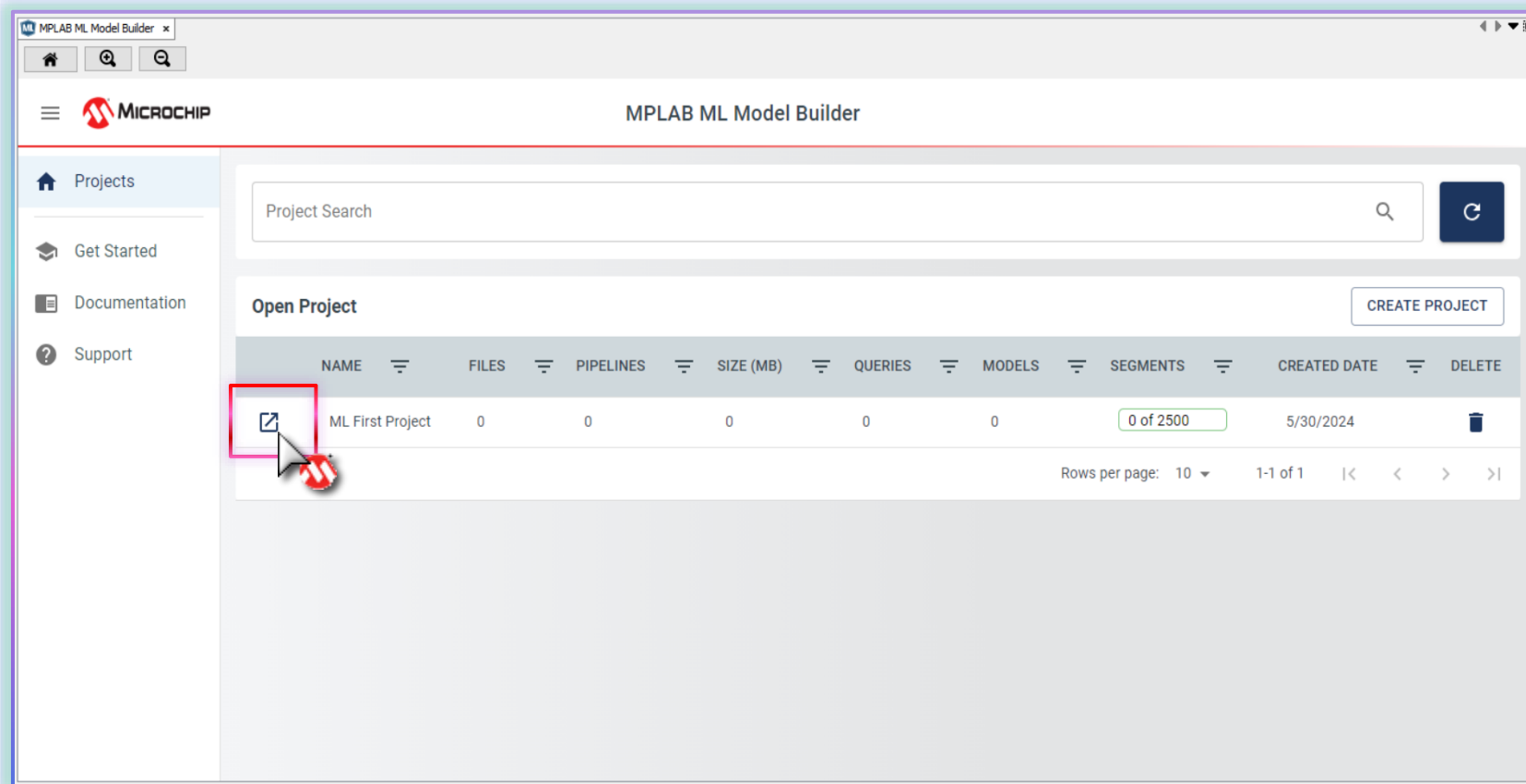
- Input Project Name: **ML First Project** for your project then click **SAVE**



⚙️ Lab2 Data Labeling and Preparation

Step 3

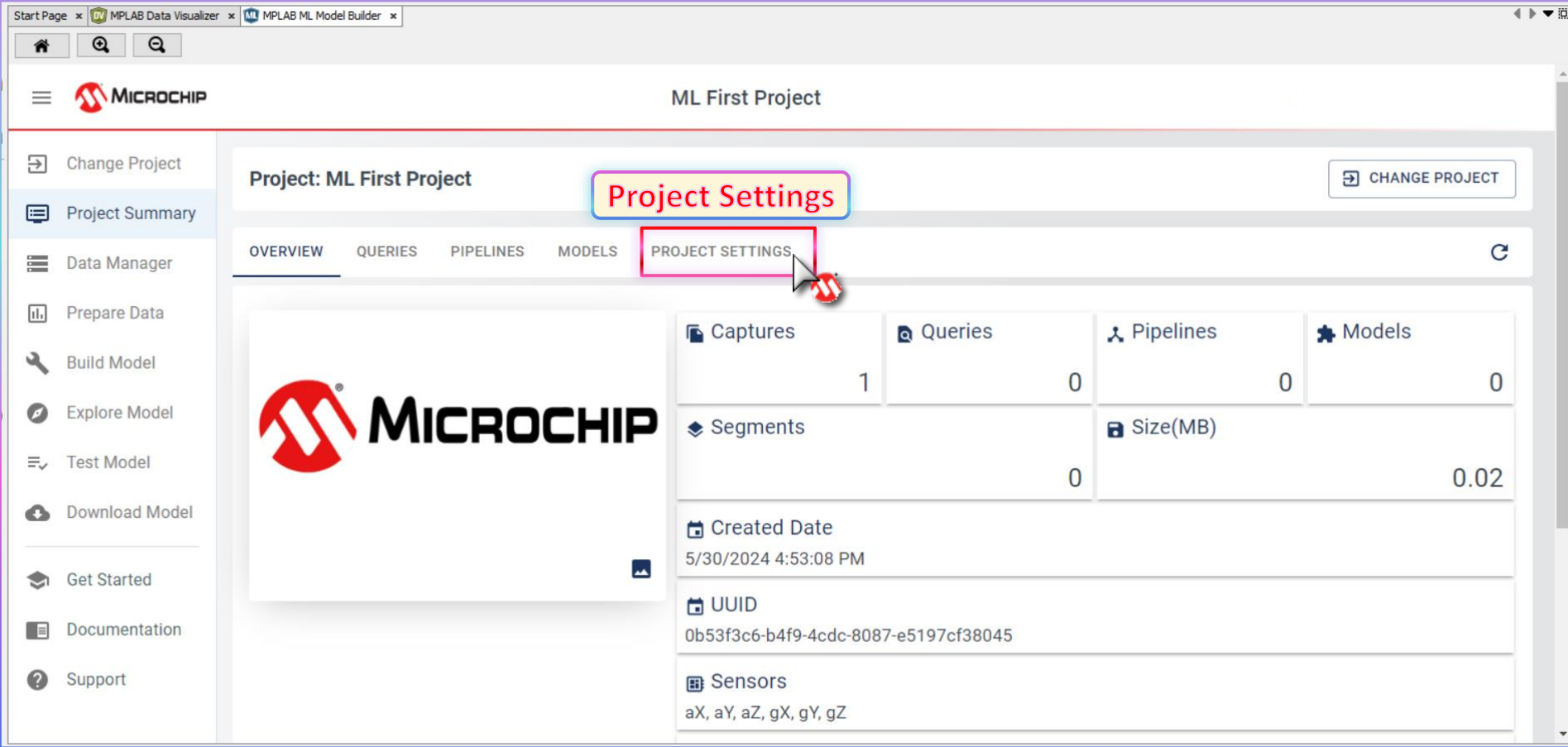
- The project we built will show on the welcome page.
- Click  icon to open **ML First Project**.



⚙️ Lab2 Data Labeling and Preparation

Step 4

- Click **Project Settings**.



The screenshot displays the Microchip ML Model Builder web application. The browser tabs at the top include 'Start Page', 'MPLAB Data Visualizer', and 'MPLAB ML Model Builder'. The application header shows the Microchip logo and the title 'ML First Project'. A left sidebar contains navigation links: 'Change Project', 'Project Summary', 'Data Manager', 'Prepare Data', 'Build Model', 'Explore Model', 'Test Model', 'Download Model', 'Get Started', 'Documentation', and 'Support'. The main content area is titled 'Project: ML First Project' and features a 'CHANGE PROJECT' button. Below this is a tabbed interface with 'OVERVIEW', 'QUERIES', 'PIPELINES', 'MODELS', and 'PROJECT SETTINGS'. The 'PROJECT SETTINGS' tab is selected and highlighted with a red box. A red callout box labeled 'Project Settings' points to this tab. The settings panel on the right contains the following information:

Captures	Queries	Pipelines	Models
1	0	0	0

Segments	Size(MB)
0	0.02

Created Date
5/30/2024 4:53:08 PM

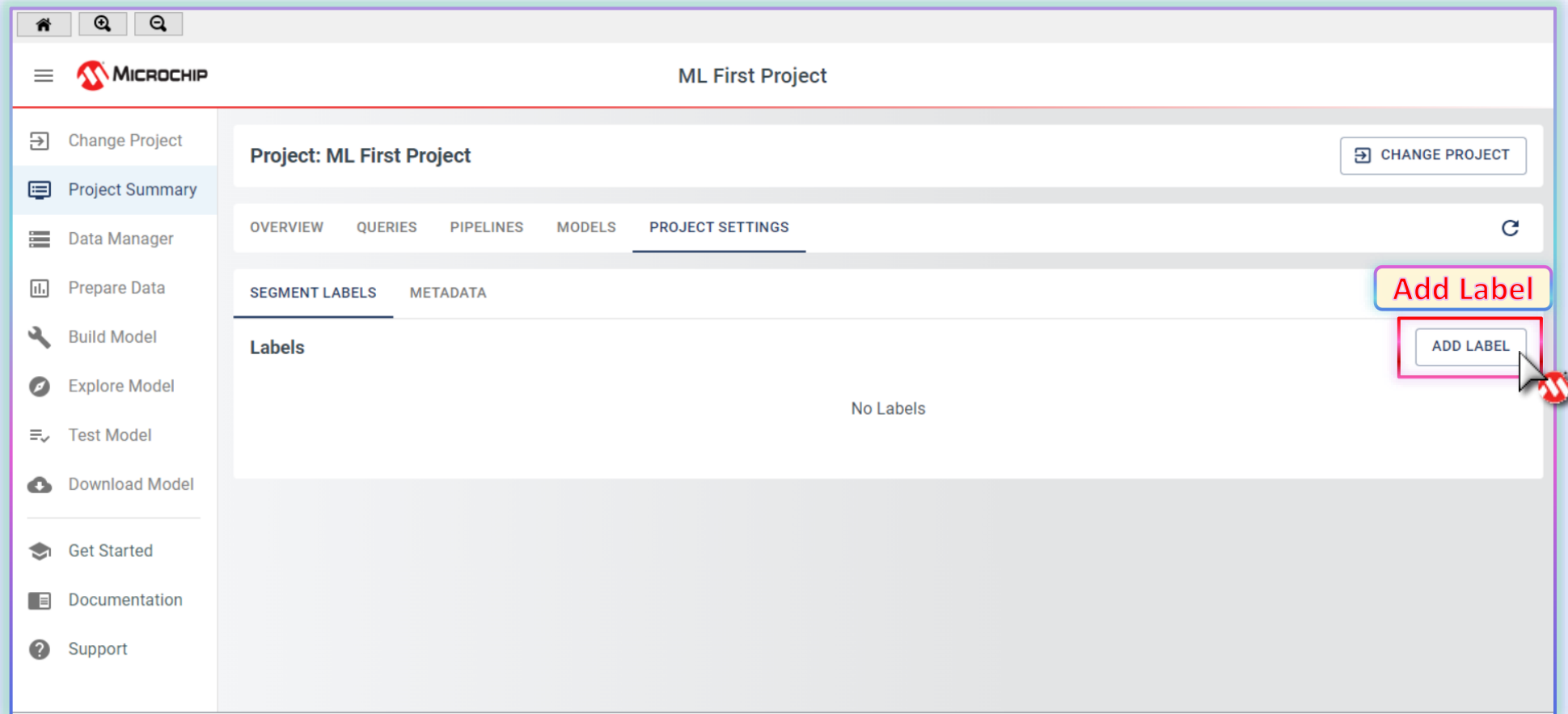
UUID
0b53f3c6-b4f9-4cdc-8087-e5197cf38045

Sensors
aX, aY, aZ, gX, gY, gZ

⚙️ Lab2 Data Labeling and Preparation

Step 5

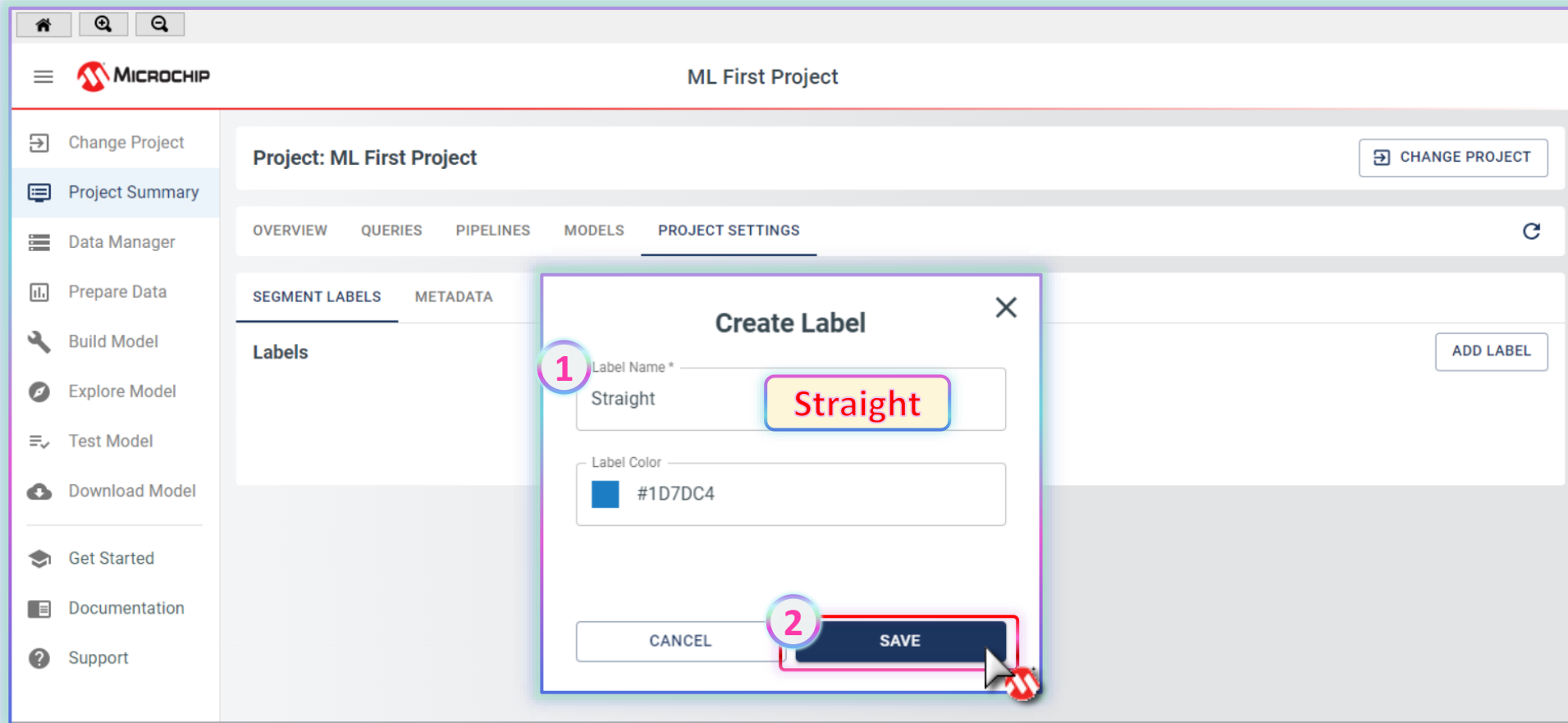
- **Add Label.**



🔧 Lab2 Data Labeling and Preparation

Step 6

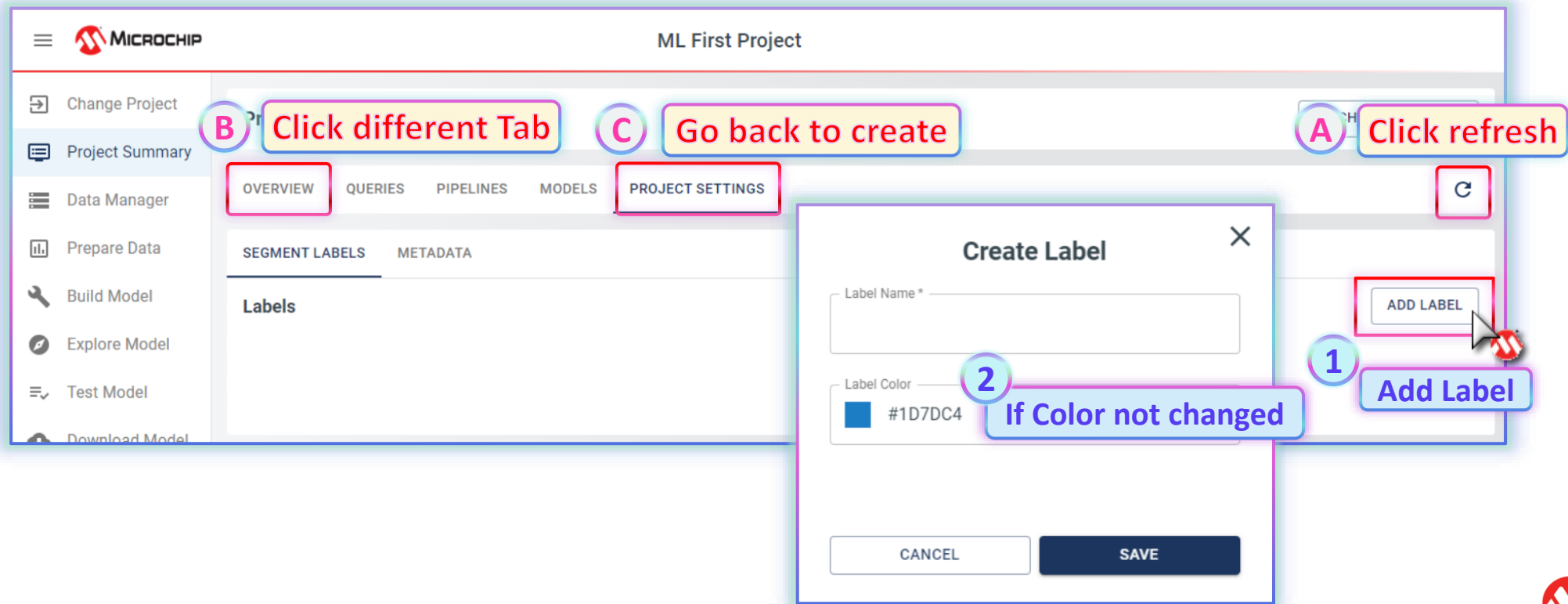
- Create Label.



🔧 Lab2 Data Labeling and Preparation

HINT

- As you add labels, ① ② the color of each label you add will change.
If you find that the **Label color does not change**,
 - ① Try clicking ↻ to refresh the GUI or
 - ② ③ Click on a different label and then go back to create another label name.



⚙️ Lab2 Data Labeling and Preparation

Step 7

- Repeat the **step 5 to step 6** to add segment label. Make sure that the Label color for each label is different.
 - **Straight**
 - **Uppercut**
 - **Righthook**
 - **Freeze**

Project: ML First Project		CHANGE PROJECT	
OVERVIEW		QUERIES	
PIPELINES		MODELS	
PROJECT SETTINGS			
SEGMENT LABELS		METADATA	
Labels		ADD LABEL	
NAME	CREATED DATE	UPDATED DATE	EDIT
DELETE			
 Straight	7/15/2024 5:01:34 PM	7/15/2024 5:01:34 PM	
 Uppercut	7/15/2024 5:01:50 PM	7/15/2024 5:01:50 PM	
 Righthook	7/15/2024 5:02:01 PM	7/15/2024 5:02:01 PM	
 Freeze	7/16/2024 2:09:45 PM	7/16/2024 2:09:45 PM	

Straight

Uppercut

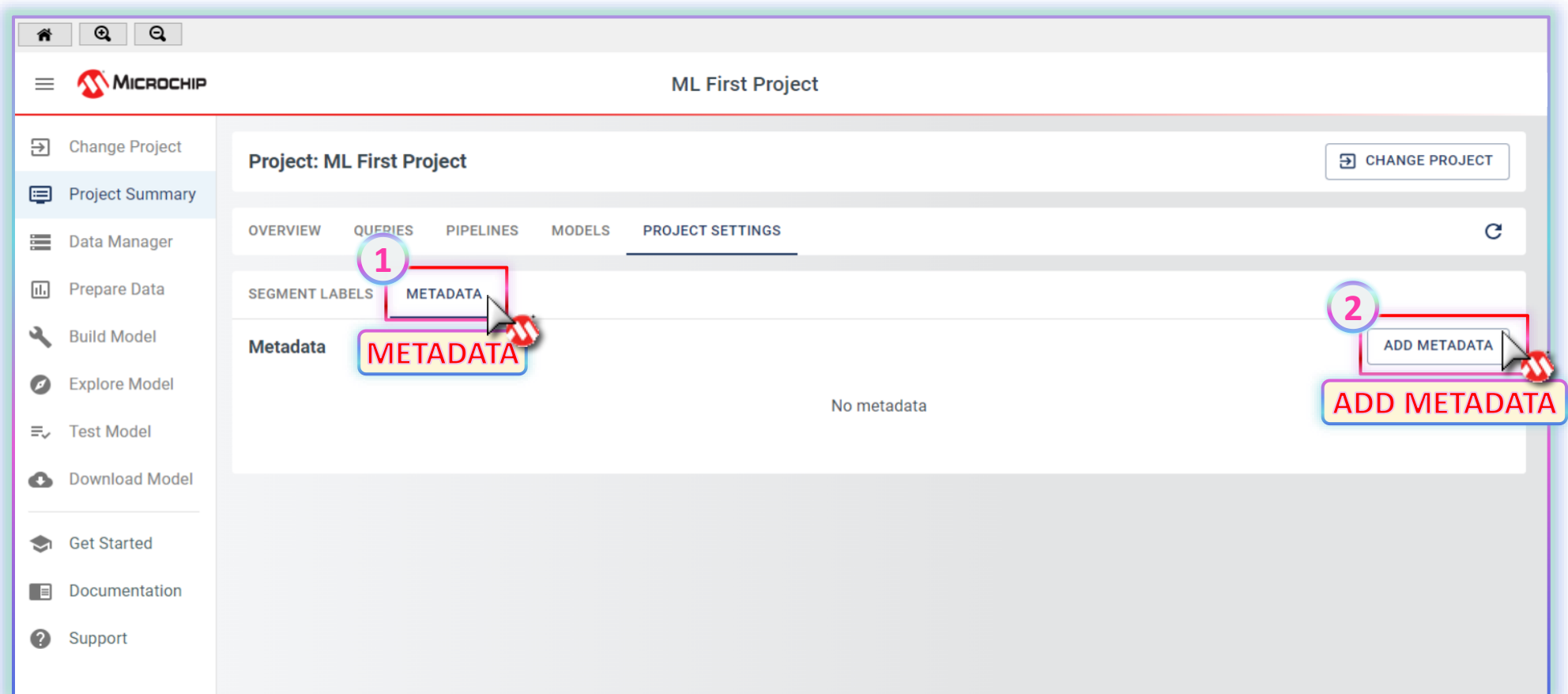
Righthook

Freeze

⚙️ Lab2 Data Labeling and Preparation

Step 7-1 (Optional)

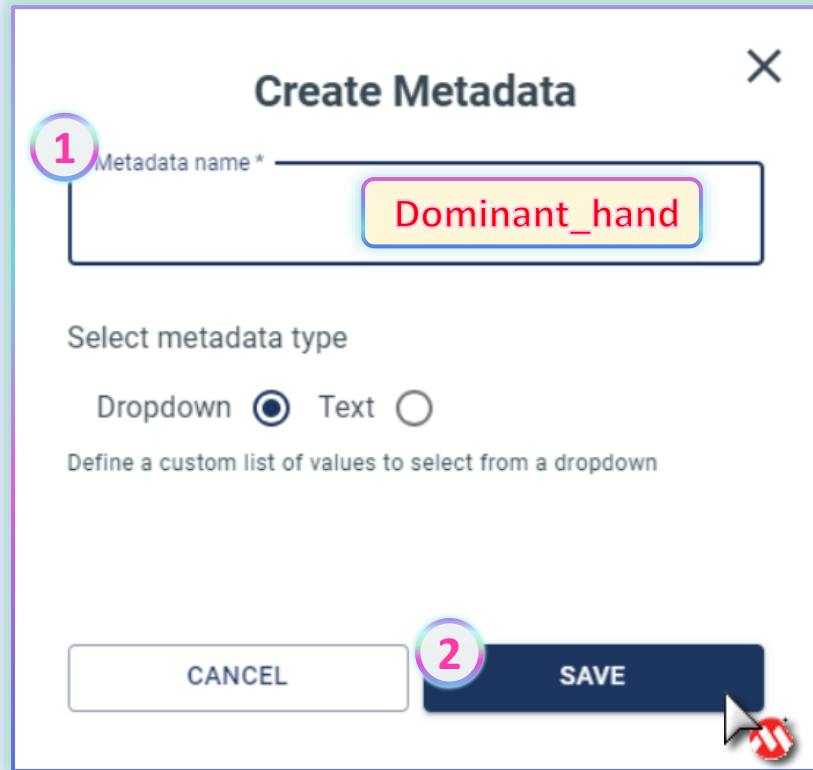
- Add **Metadata**.



⚙️ Lab2 Data Labeling and Preparation

Step 7-2 (Optional)

- Create **Metadata name** and **Metadata values**.



Create Metadata [X]

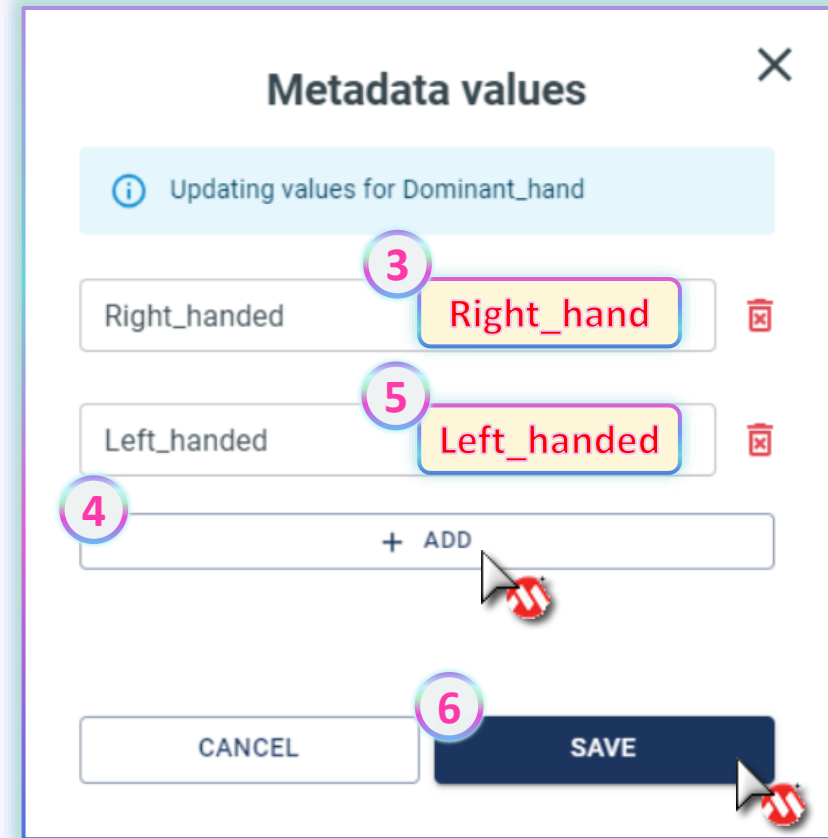
1 Metadata name *

Select metadata type

Dropdown ☒ Text ☐

Define a custom list of values to select from a dropdown

CANCEL 2 SAVE



Metadata values [X]

i Updating values for Dominant_hand

3 Right_handed [X]

5 Left_handed [X]

4 + ADD

CANCEL 6 SAVE

⚡ Lab2 Data Labeling and Preparation

HINT

- Now we are going to start the “Data Labeling” works !!
- The next steps are probably the most important but also the most boring part of the machine learning training process.

But please be patient !

- Observe the aX, aY, aZ, gX, gY, and gZ raw data firstly, then start labeling data, even slightly differences may impact the training result.
- If the event doesn't belong to any label, keep it empty to avoid confusing training.
- Repeat step 8 to step 14 to complete label all sections and files.

⚙️ Lab2 Data Labeling and Preparation

Step 8

- Click **Data Manager**. The collected data files can be found in the Data Manager page.

The screenshot shows the Microchip ML First Project interface. The left sidebar contains a menu with the following items: Change Project, Data Manager (highlighted with a red callout box), Prepare Data, Build Model, Explore Model, Test Model, Download Model, Get Started, Documentation, and Support. The main area is titled 'Data Manager' and 'Session: Training Session'. It features a table of captures with the following columns: CAPTURE NAME, SEGMENTS, SIZE (MB), CREATED DATE, DOMINANT_HAND, OPEN, DOWNLOAD, METADATA, and DELETE. The table contains three rows of data:

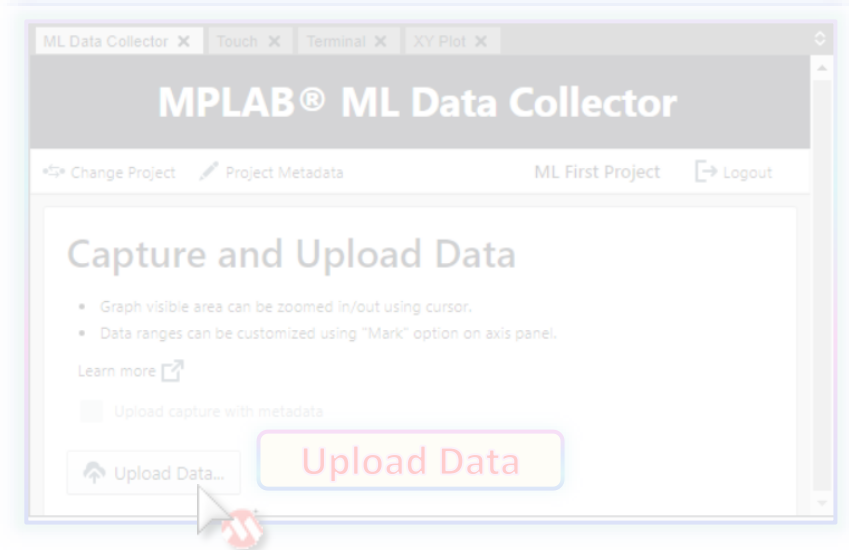
	CAPTURE NAME	SEGMENTS	SIZE (MB)	CREATED DATE	DOMINANT_HAND	OPEN	DOWNLOAD	METADATA	DELETE
☐	Right1.csv	0	0.22	7/16/2024, 2:43 PM					
☐	Straight1.csv	0	0.12	7/16/2024, 2:43 PM					
☐	Upper1.csv	0	0.17	7/16/2024, 2:43 PM					

At the bottom right of the table, it says 'Rows per page: 25' and '1-3 of 3'.

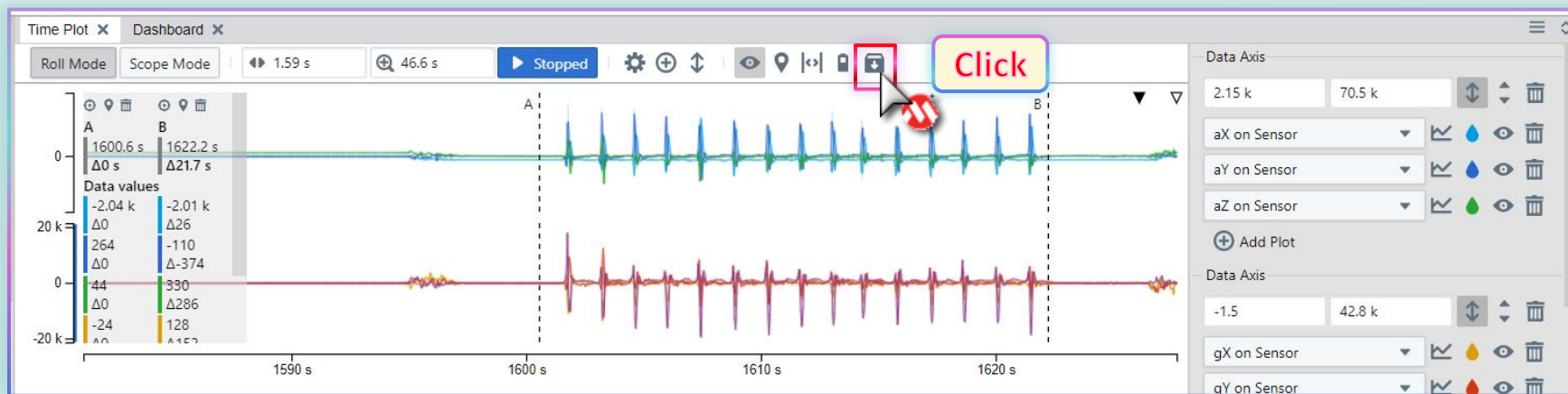
⚙️ Lab2 Data Labeling and Preparation

Two Ways to Capture and Upload Data

1. Upload and save Data by **MPLAB ML Data Collector**. The data will be saved to **.csv** file in your local PC and upload to the ML Project you created **automatically**.



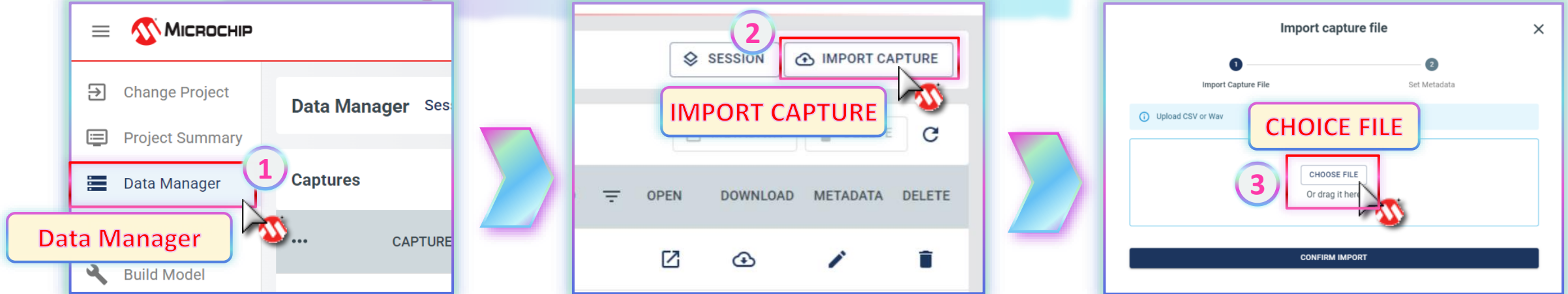
2. ✓ **Snapshot** Data by click on  in Time Plot of DV to save the **.csv** file in your local PC. Then upload the **.csv** file to the ML Project you created **manually**.



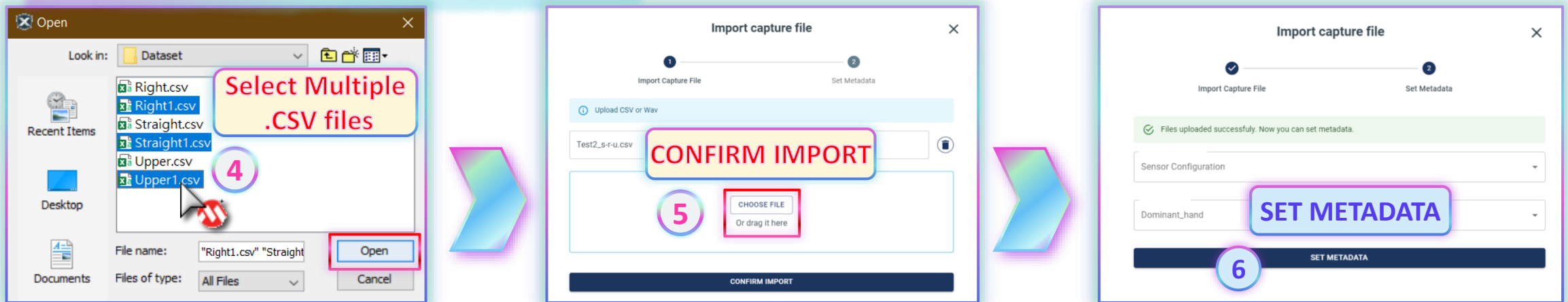
🔧 Lab2 Data Labeling and Preparation

Import the Snapshot .csv files

- Click on **Data Manager > IMPORT CAPTURE > CHOICE FILE.**



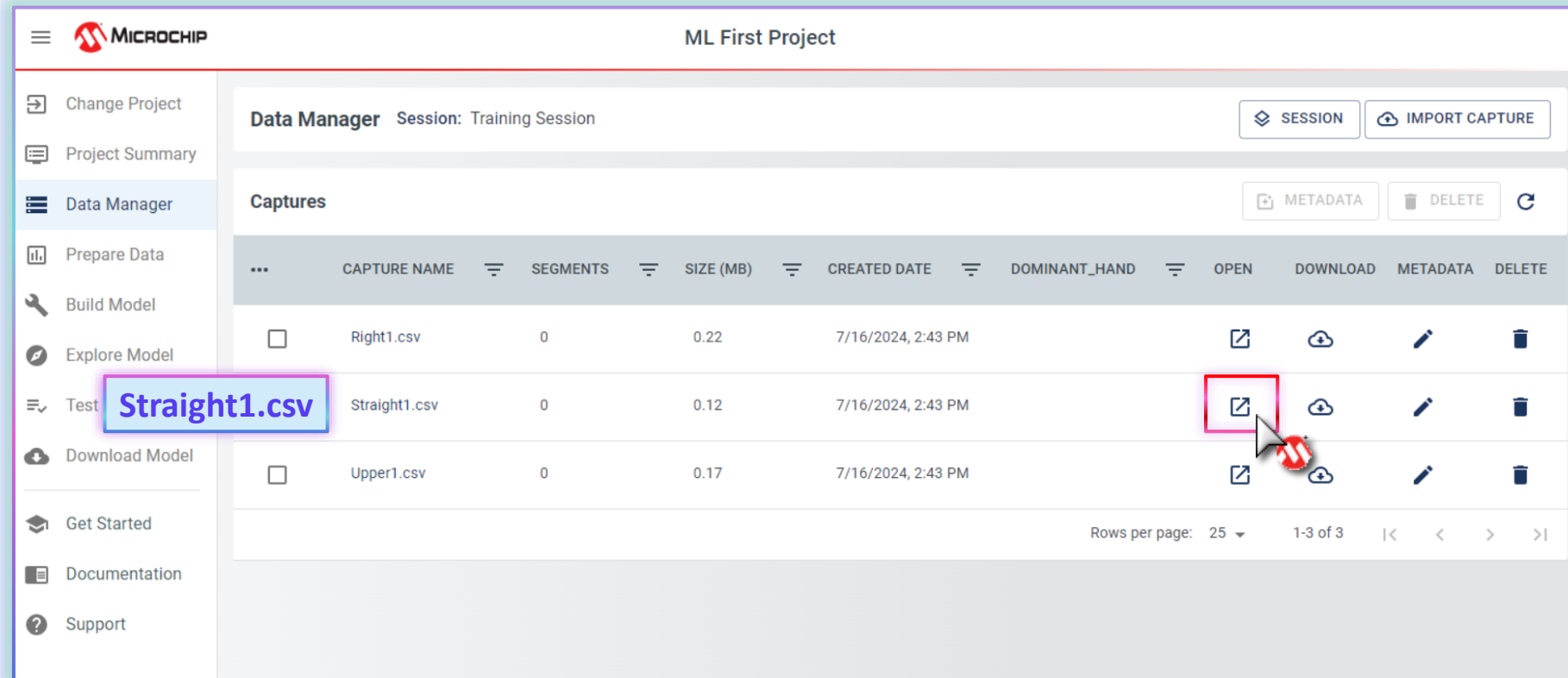
- Select multiple **.CSV** you stored then **CONFIRM IMPORT > SET METADATA(optional).**



⚙️ Lab2 Data Labeling and Preparation

Step 9

- Let's start labeling our data, which also means define segments.
- Open **Straight1.csv** data.



The screenshot shows the Microchip ML First Project Data Manager interface. The left sidebar contains navigation links: Change Project, Project Summary, Data Manager (selected), Prepare Data, Build Model, Explore Model, Test, Download Model, Get Started, Documentation, and Support. The main area displays the 'Data Manager' session for 'Training Session'. It shows a table of captures with columns: CAPTURE NAME, SEGMENTS, SIZE (MB), CREATED DATE, DOMINANT_HAND, OPEN, DOWNLOAD, METADATA, and DELETE. The table lists three captures: Right1.csv, Straight1.csv, and Upper1.csv. The 'Straight1.csv' capture is highlighted with a red box, and a red circle with a white 'X' is placed over the 'OPEN' icon for this capture.

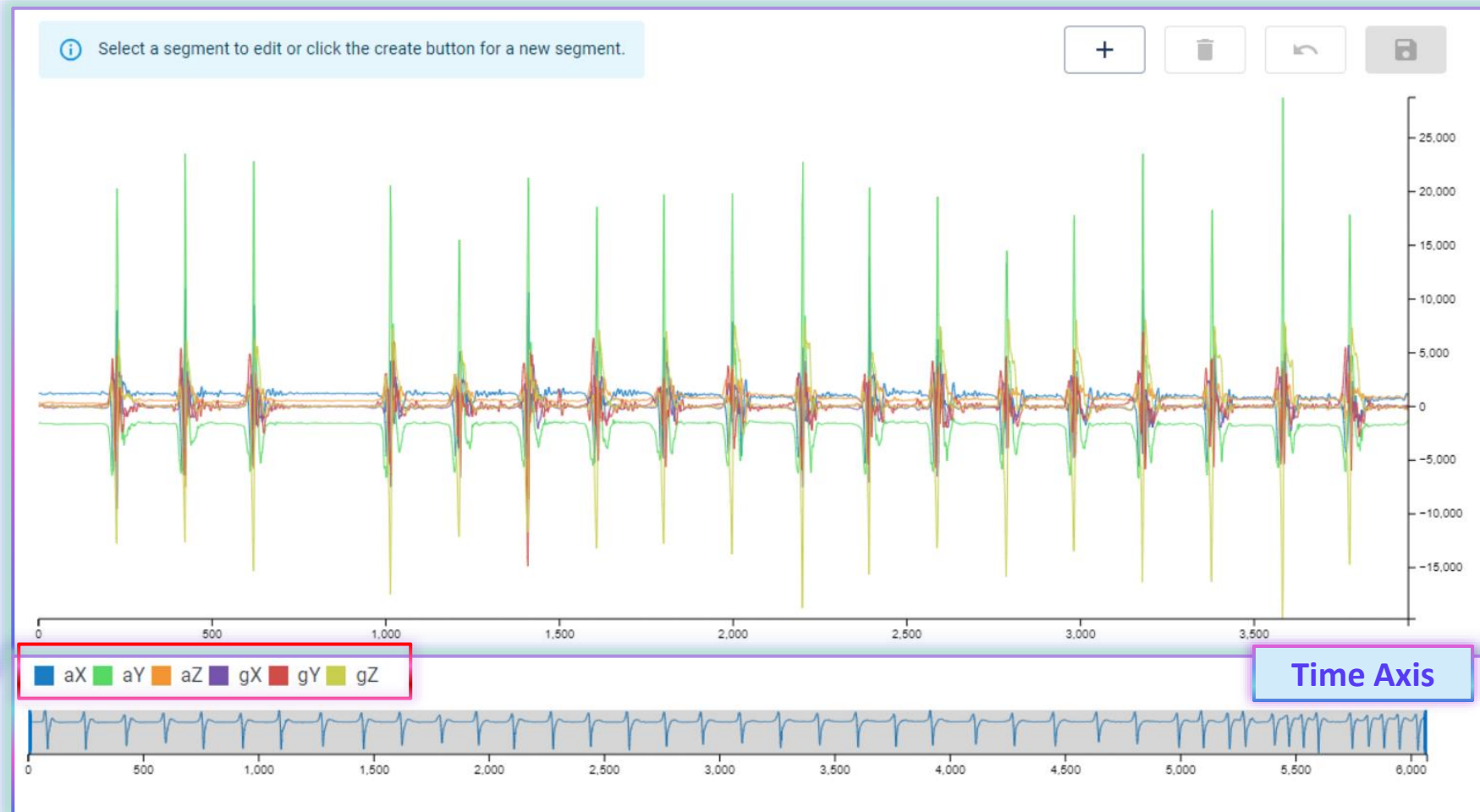
...	CAPTURE NAME	SEGMENTS	SIZE (MB)	CREATED DATE	DOMINANT_HAND	OPEN	DOWNLOAD	METADATA	DELETE
☐	Right1.csv	0	0.22	7/16/2024, 2:43 PM					
☐	Straight1.csv	0	0.12	7/16/2024, 2:43 PM					
☐	Upper1.csv	0	0.17	7/16/2024, 2:43 PM					

Rows per page: 25 1-3 of 3 |< < > >|

⚙️ Lab2 Data Labeling and Preparation

Step 10

- **Segment:** Define where an event is located in your sensor data file.

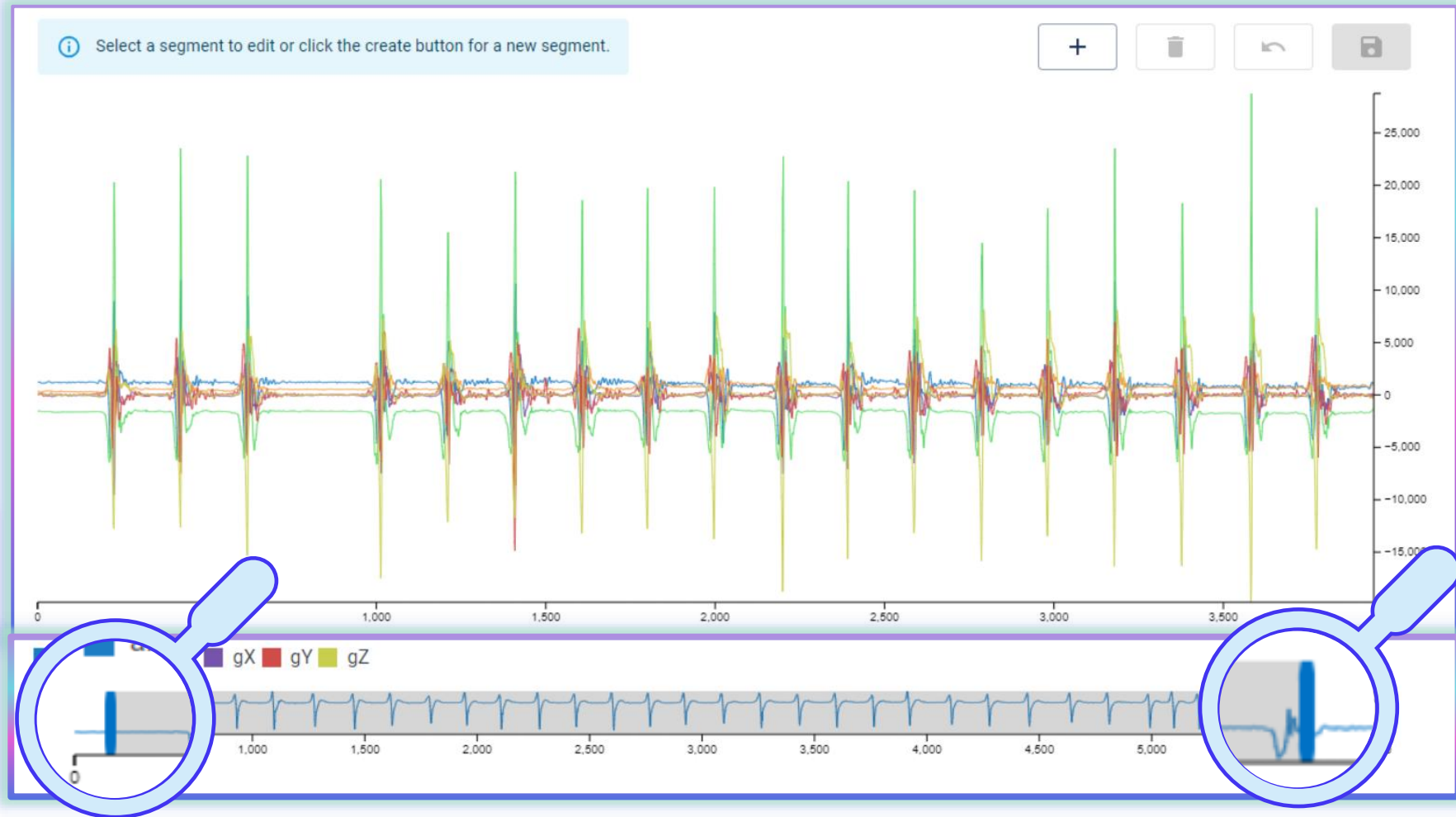


Click to show
or hide the
selected
raw data.

⚙️ Lab2 Data Labeling and Preparation

Step 11

- You may **zoom in** to view the detail of the file.

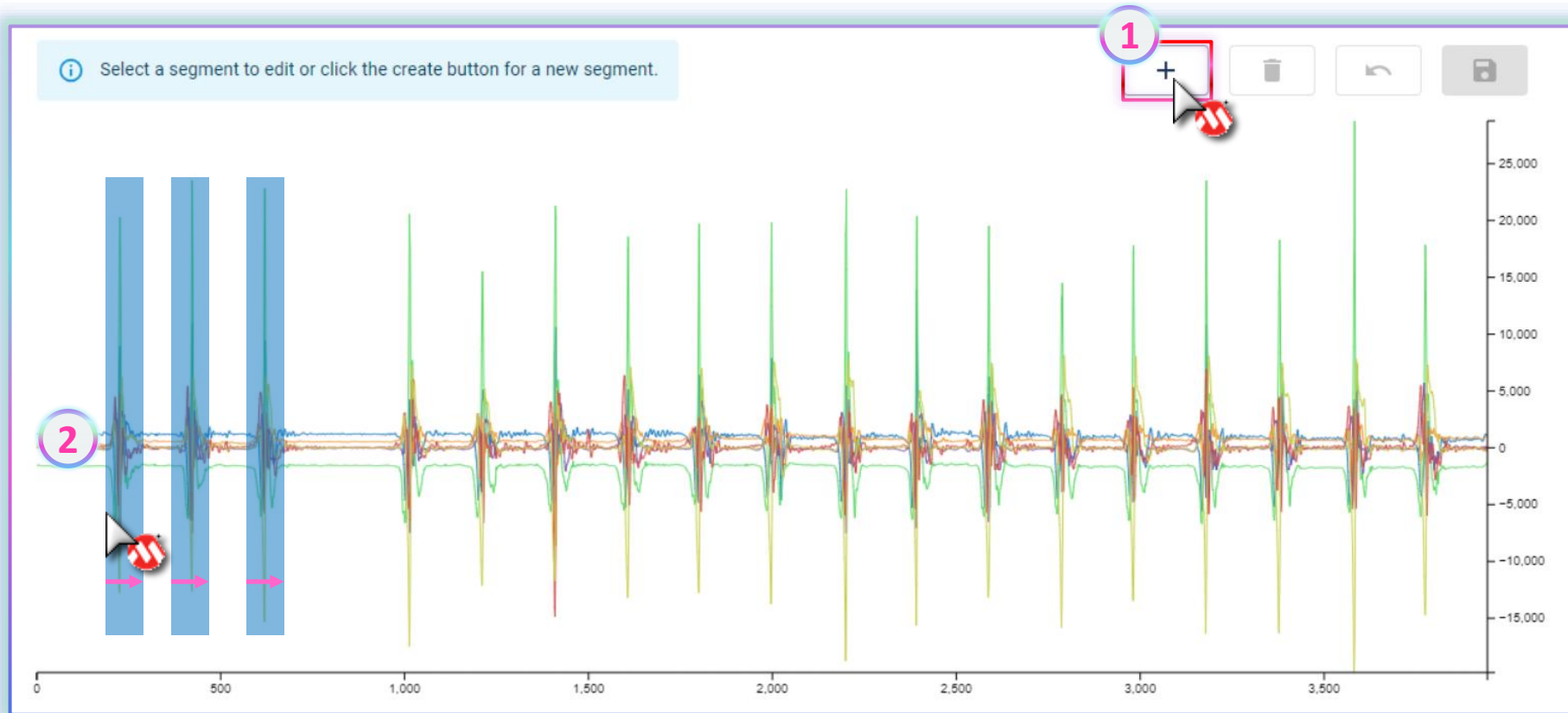


Slide it to
Zoom in the view

⚙️ Lab2 Data Labeling and Preparation

Step 12

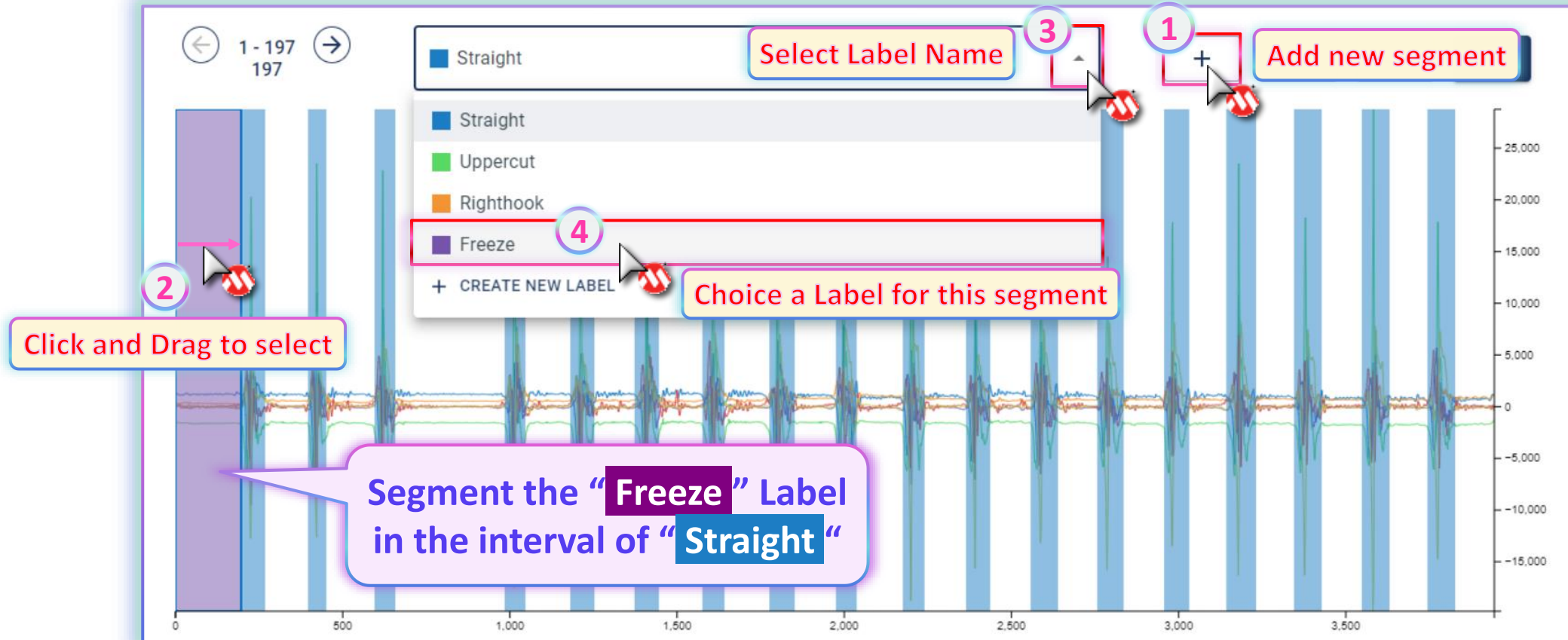
- Click  to add segment.
- Move the cursor over the chart, then **Click and Drag** the area that will be marked as an event. This will place a new segment in the file.



⚙️ Lab2 Data Labeling and Preparation

Step 13

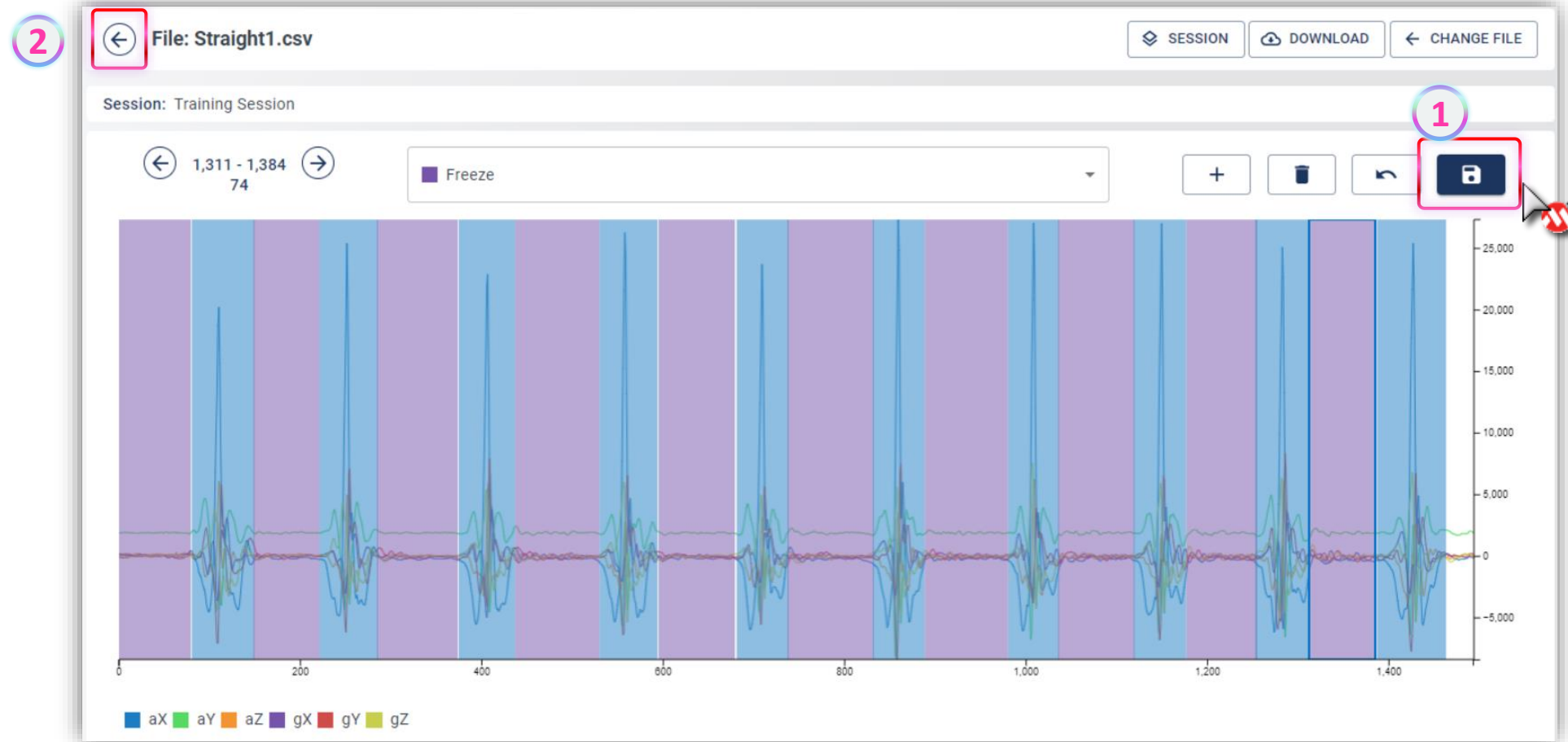
- **Click and Drag** to select the segment first, then choose the **label** for the corresponding selection segment.



⚙️ Lab2 Data Labeling and Preparation

Step 14

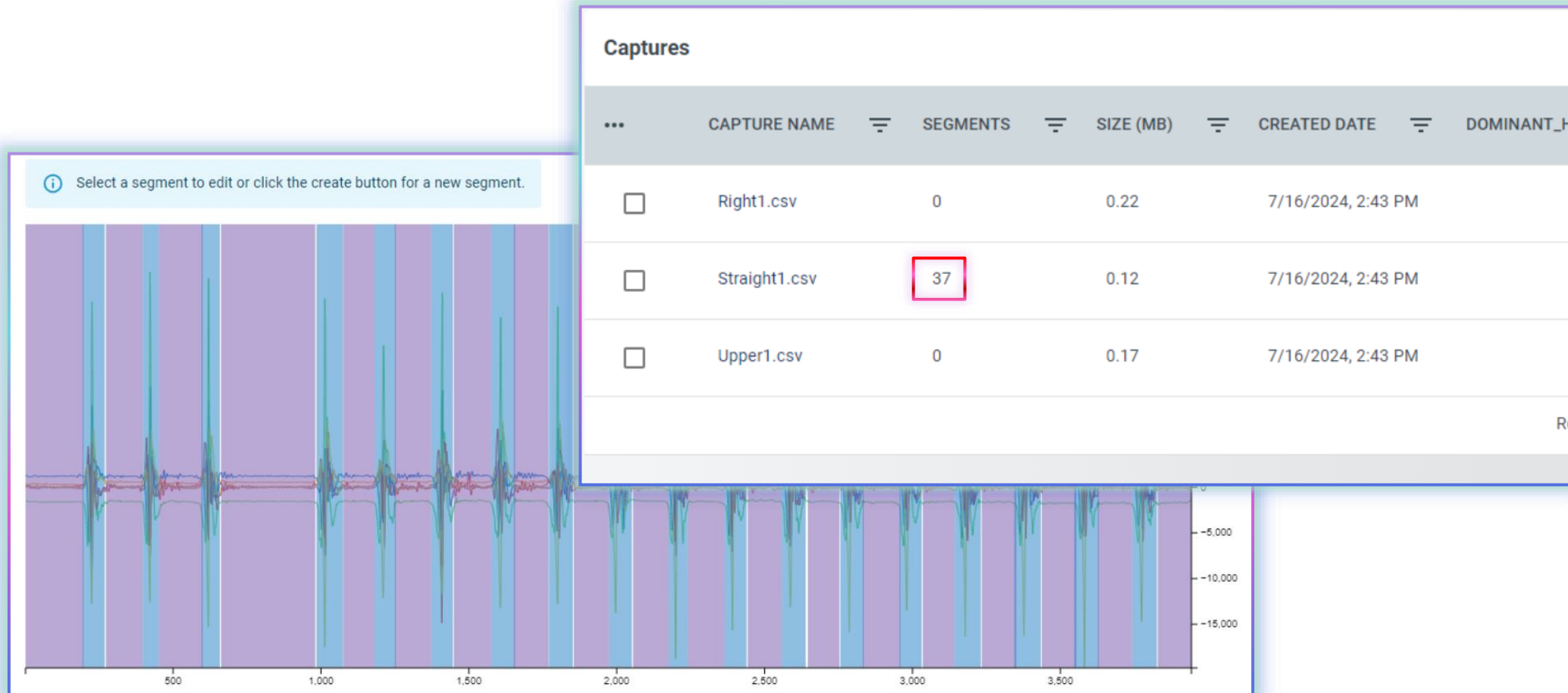
- Finish labeling the segment in each dataset then Click  to save.
- Remember to click “**save**” anytime!



⚙️ Lab2 Data Labeling and Preparation

Step 15

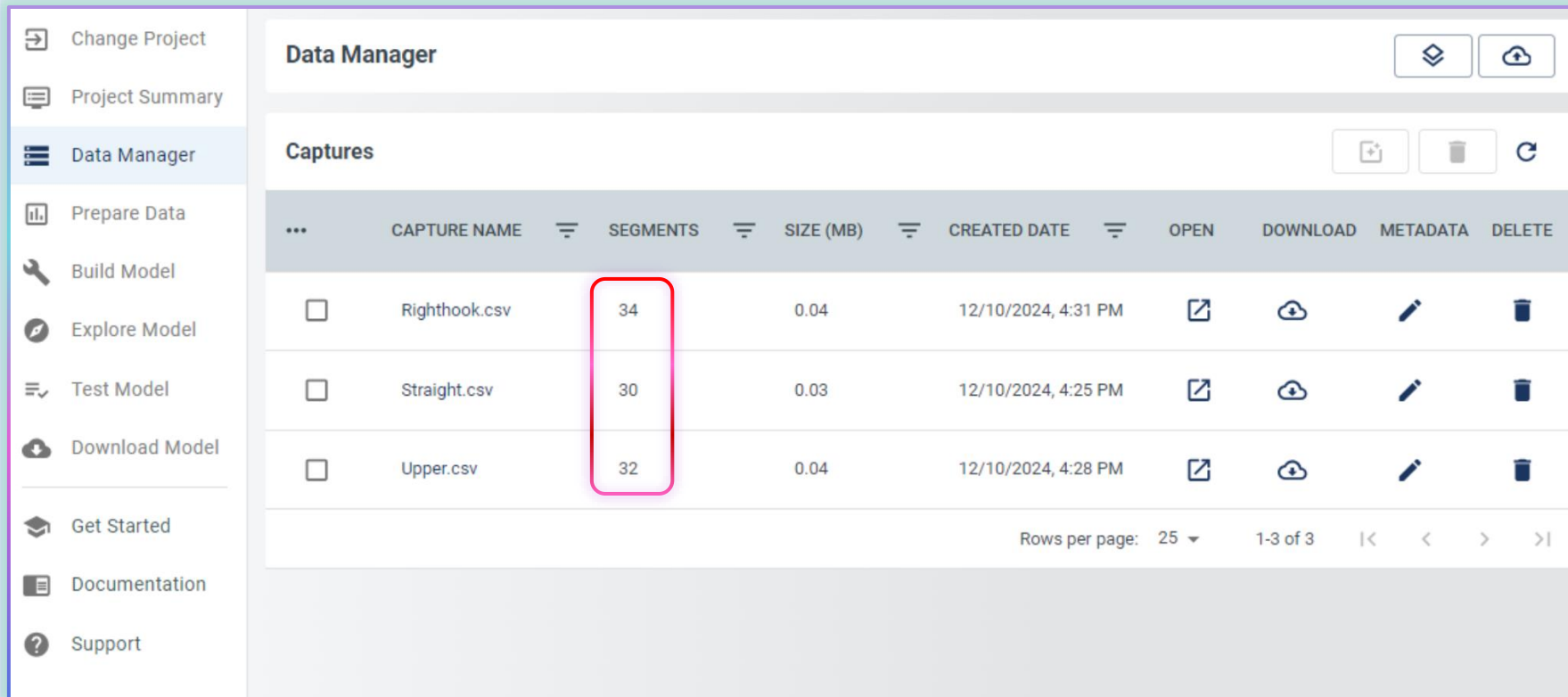
- The data manager will show the number of segments in each data.



⚙️ Lab2 Data Labeling and Preparation

Result

- Repeat **step 9 to step 15** to label all the segment and file.
- Make sure you **label** and **save** all the dataset. There will be number in the SEGMENTS column.



The screenshot shows the 'Data Manager' interface. On the left is a sidebar with navigation options: Change Project, Project Summary, Data Manager (selected), Prepare Data, Build Model, Explore Model, Test Model, Download Model, Get Started, Documentation, and Support. The main area is titled 'Data Manager' and contains a 'Captures' table. The table has columns: CAPTURE NAME, SEGMENTS, SIZE (MB), CREATED DATE, OPEN, DOWNLOAD, METADATA, and DELETE. Three rows are visible: Righthook.csv, Straight.csv, and Upper.csv. The 'SEGMENTS' column values are 34, 30, and 32 respectively. A red rectangular box highlights the 'SEGMENTS' column header and its values. At the bottom right of the table, it says 'Rows per page: 25' and '1-3 of 3'.

...	CAPTURE NAME	SEGMENTS	SIZE (MB)	CREATED DATE	OPEN	DOWNLOAD	METADATA	DELETE
☐	Righthook.csv	34	0.04	12/10/2024, 4:31 PM				
☐	Straight.csv	30	0.03	12/10/2024, 4:25 PM				
☐	Upper.csv	32	0.04	12/10/2024, 4:28 PM				

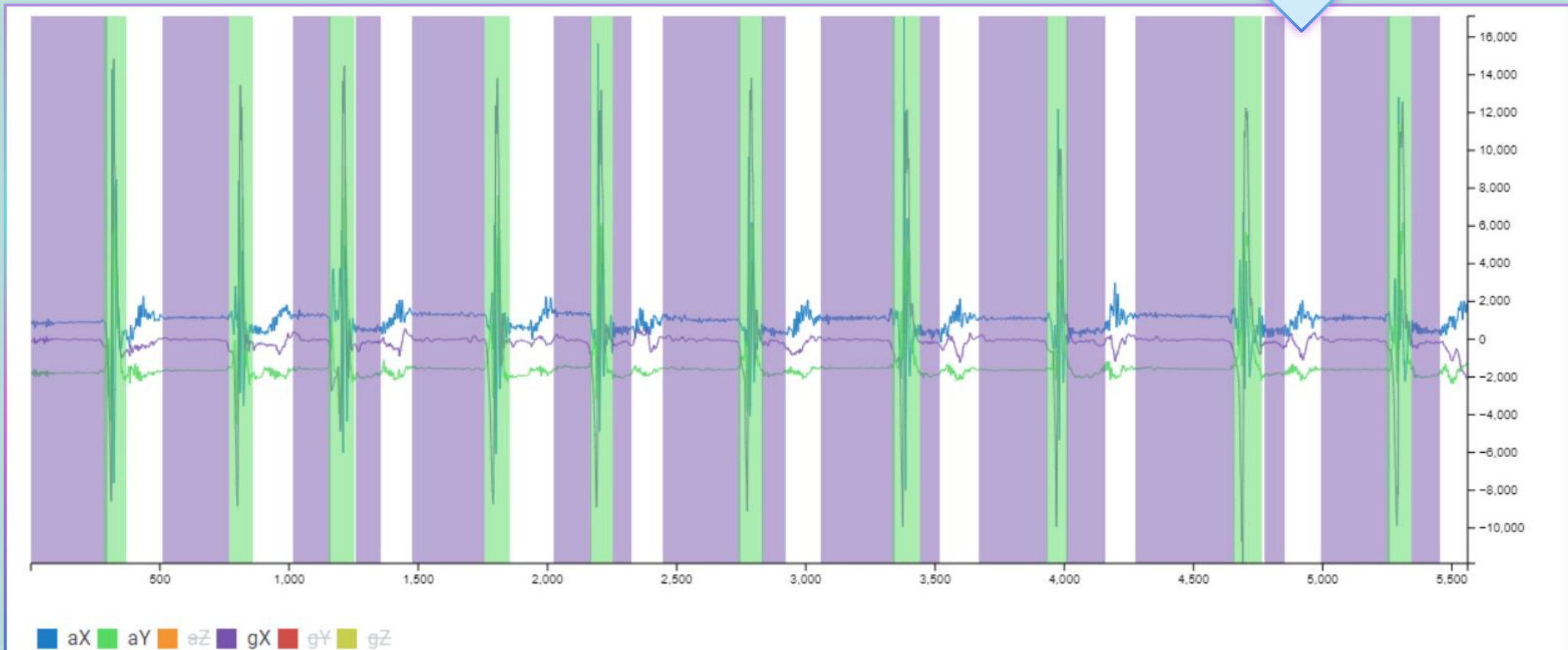
Rows per page: 25 1-3 of 3

⚙️ Lab2 Data Labeling and Preparation

Result & Discussion

- Take **Uppercut1.csv** as example.

Why keep this unlabeled?
Is it also an uppercut?



Model Pipeline and Training

Querying the data

Querying data:

The process of **extracting data from a database**, repository or other source **for** use in **training** a machine learning model.

- Only a subset may be required
- Metadata comes in useful
 - Only select samples from right-handed
 - Or samples taken by male
- Create models better tuned for certain scenarios or markets

Build Model - Pipeline

The model training process includes three main processes :

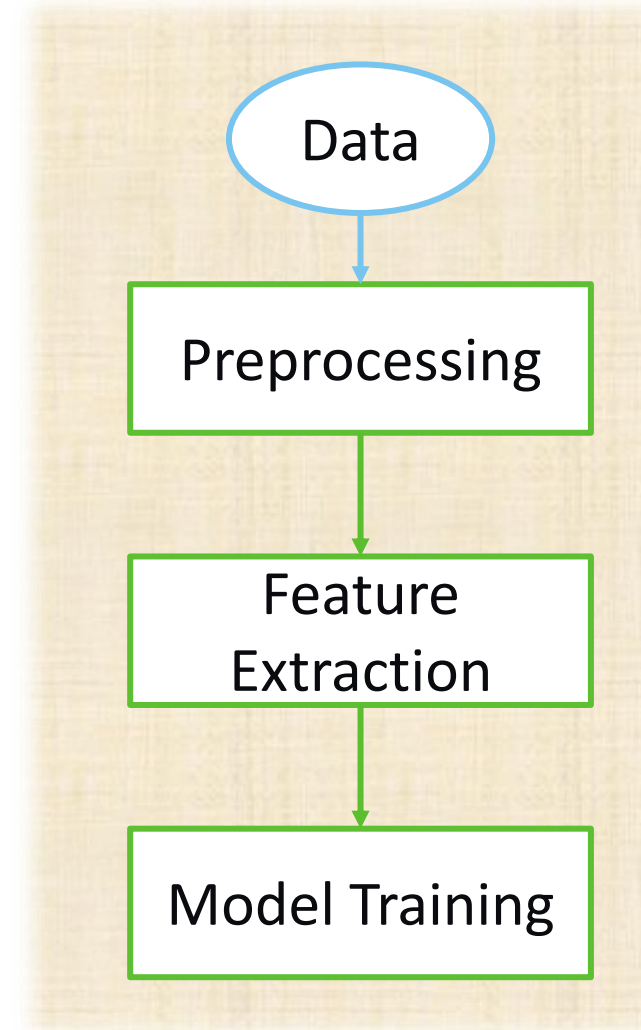
- **Preprocessing**

- Input Query
- Segmenter
- Sensor Transform

- **Feature Extraction**

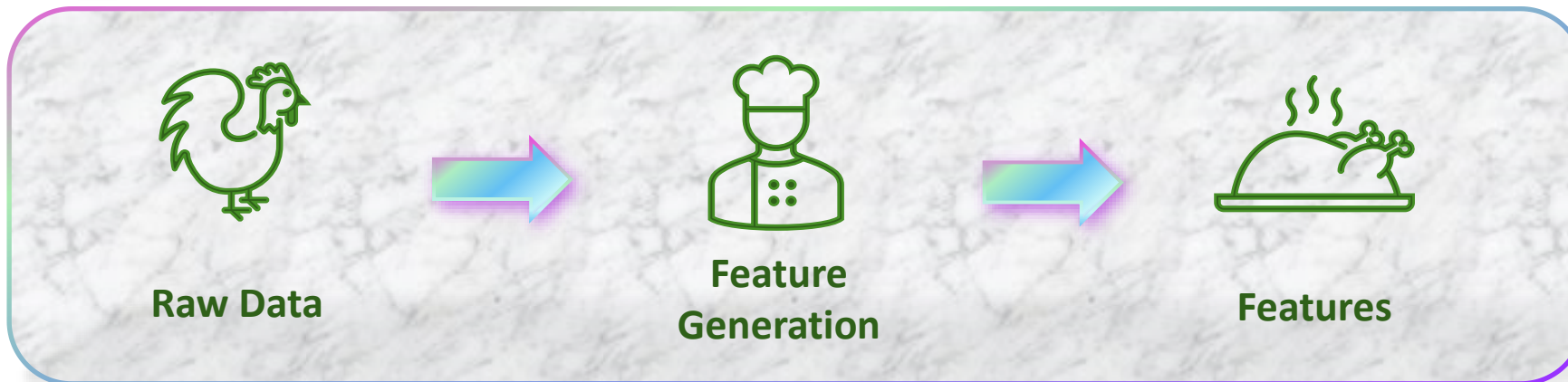
- Feature Generator
- Feature Selector

- **Model Training**



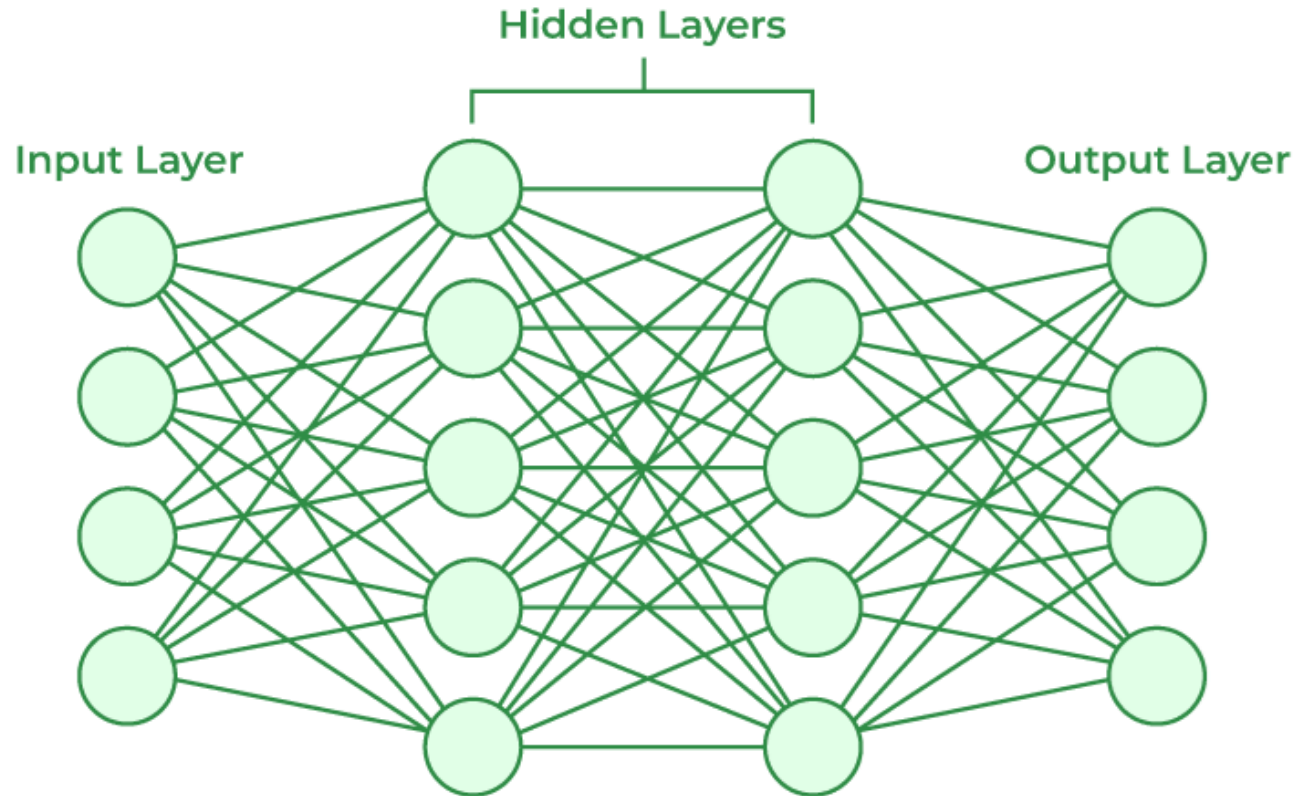
Feature Generation

- “More data beats clever algorithms, but better data beats more data,” said Peter Norvig.
- The process of converting raw data into a format that a machine learning model can understand. May include extracting information from the raw data, creating new features, or transforming the data to improve the performance of the model.
 - Which sensor data to use
 - Generate features
 - Select features
 - Classification



AI/ML Models

- How does an AI/ML model look like in your mind?

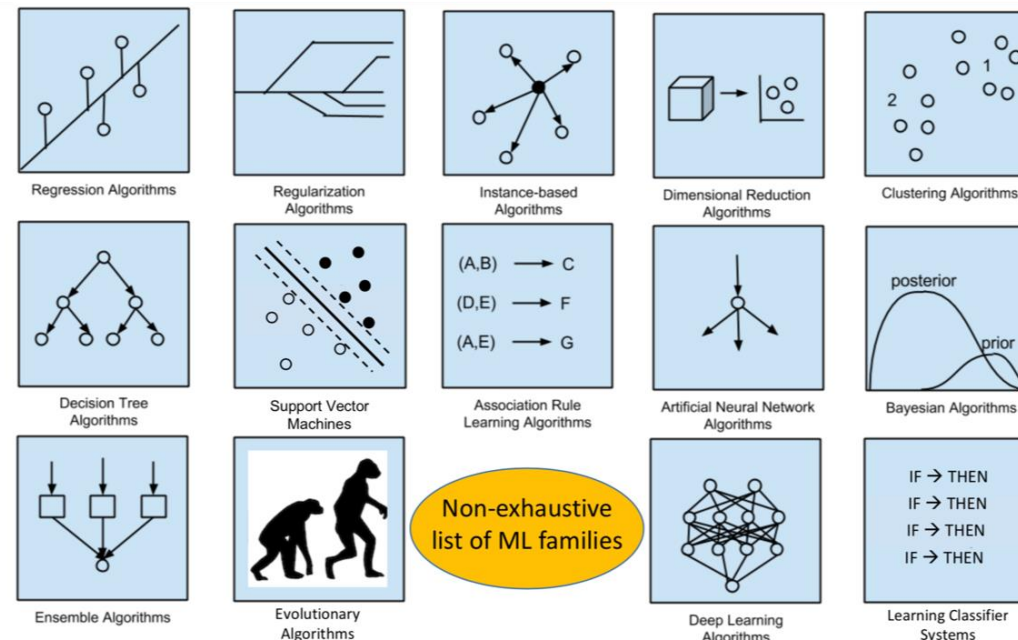


The answer is.....Yes and no.

Not all AIML models are Neural Networks.

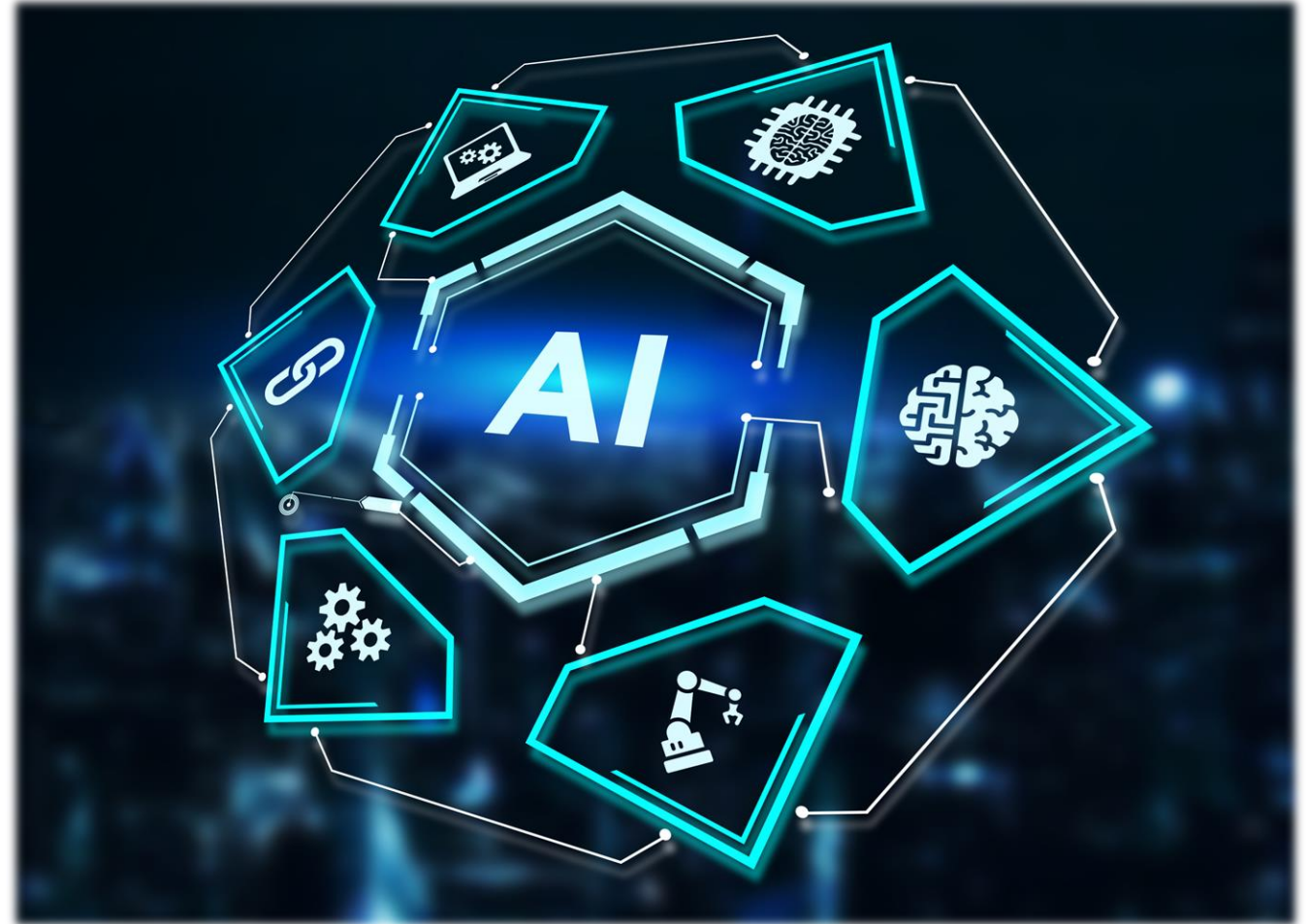
ML Models

- **Not all models are neural networks**
- **To build an application we need to know what type of model to use.**
- **Depending upon the problem space different models may be more appropriate.**
- **Certain architectures suit different problems**
 - YOLO (You only look once) well suited to object detection using deep convolutional neural network.
 - CNN (convolutional neural network) can be used for vibration analysis but others like Bayesian networks might be better.



Common ML model

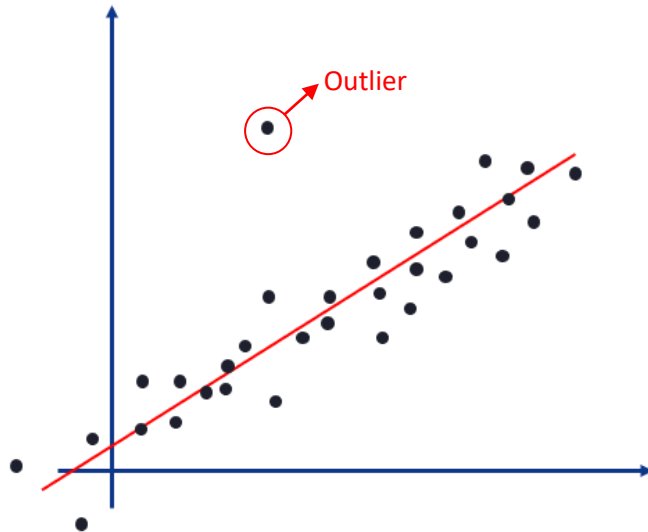
- **Supervised Learning**
 - Linear/Logistic Regression
 - Decision Tree
 - Support Vector Machine
- **Unsupervised Learning**
 - K-means Clustering
 - Hierarchical Clustering
- **Deep Learning**
 - CNN
 - RNN



Supervised Learning - Linear/Logistic Regression

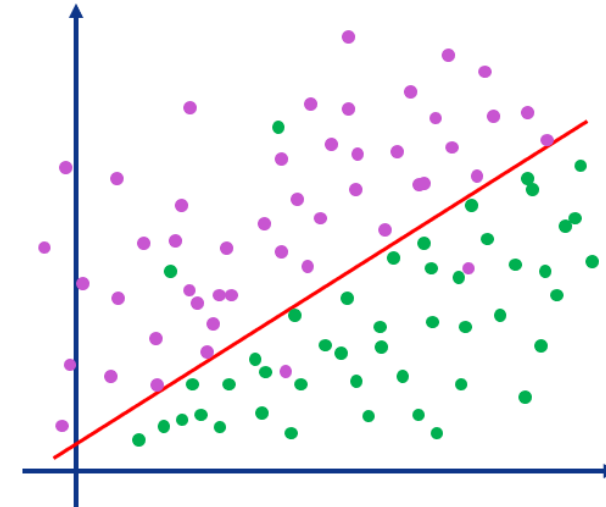
- **Linear Regression**

- Find the relationship between independent variables and dependent variables.
- Example:
 - Optimize pricing strategies
 - Demand analysis



- **Logistic Regression**

- The output of logistic regression is binary, either 1 or 0.
- Example:
 - Whether there is fraud based on transaction behavior patterns
 - Whether a tumor is malignant

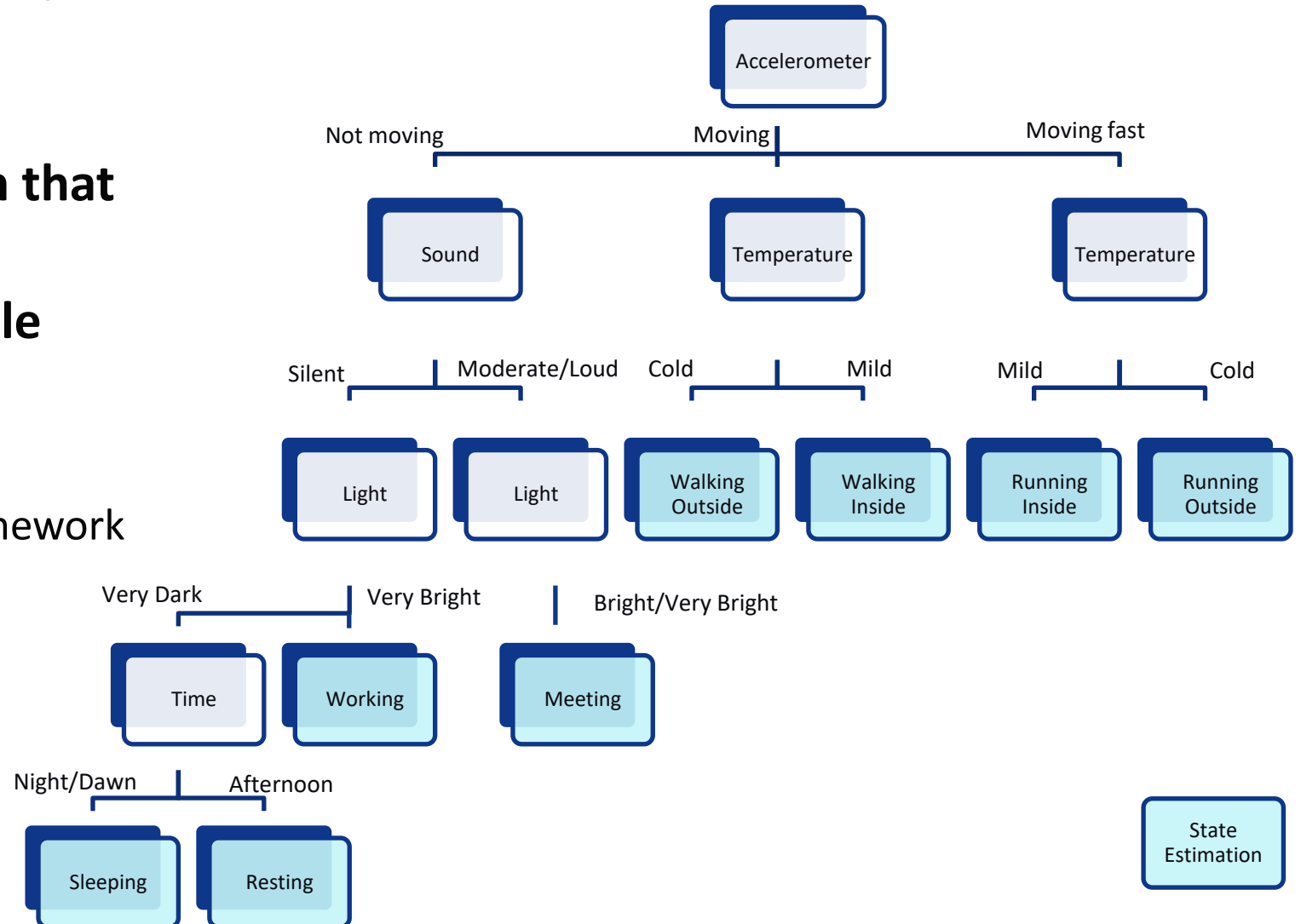


Supervised Learning - Decision Tree

Tree-like graph of decisions and possible consequences

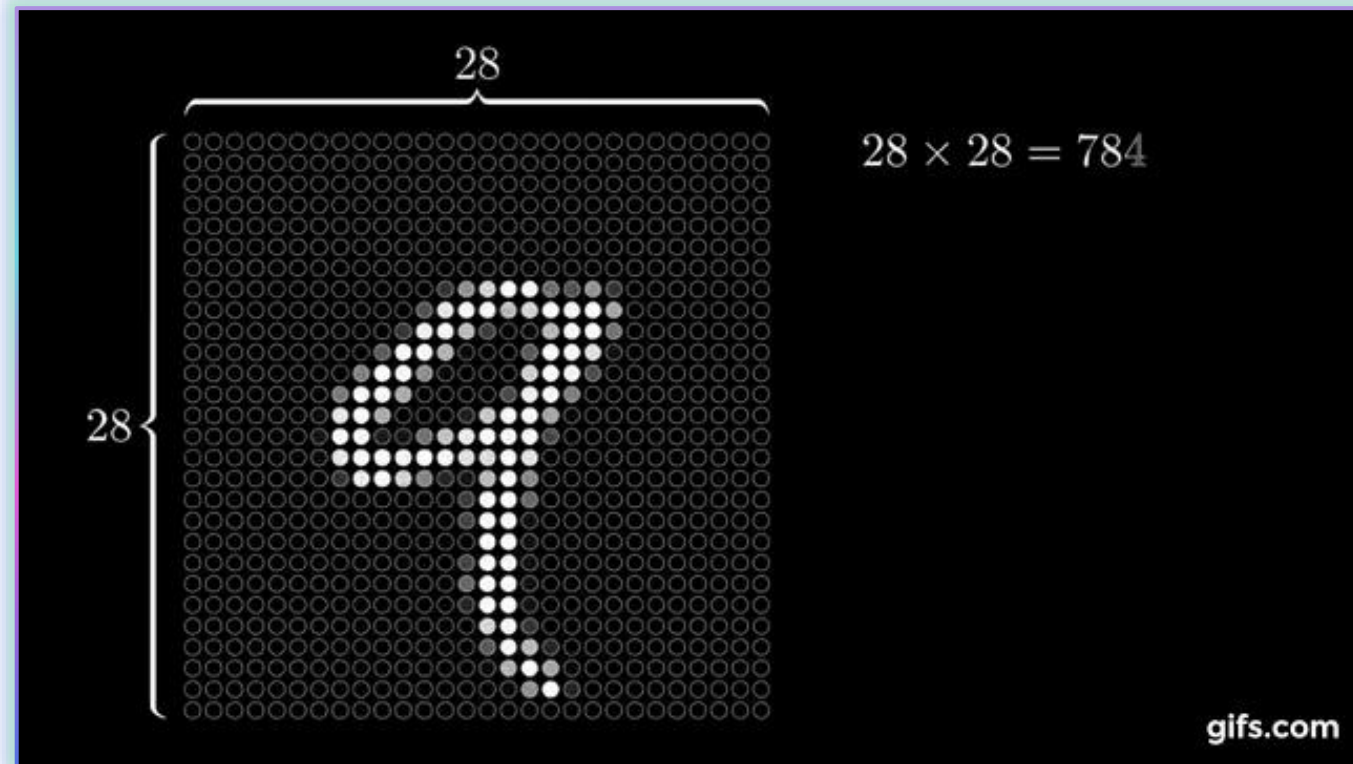
- Each node is a feature
- Branching is decision rule on that feature
- Training requires splitting rule generation and pruning
- **Example:**
 - Provide a decision-making framework

Example Movement Sensor



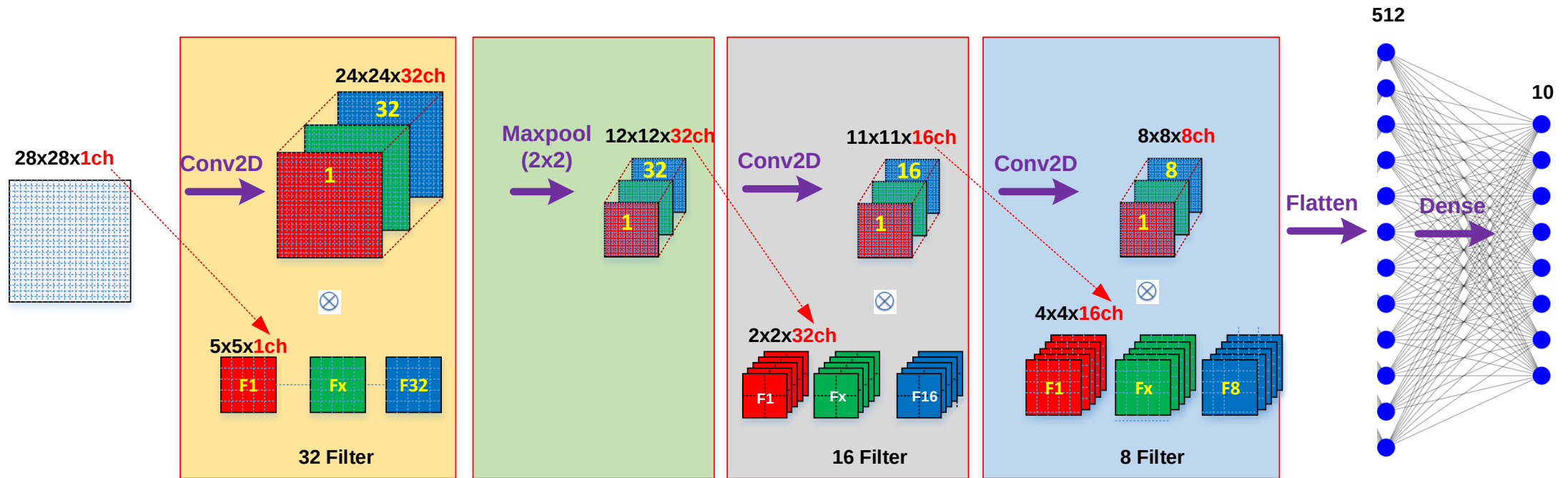
Deep Learning – Artificial Neural Network

- **Fully connected layers of neurons**
 - Best for tabular data, time series forecast, basic classification.
- **Touch example**
 - Touch sensor with resolution of 28 X 28 means 784 pieces of data to feed into the network with 10 desired outputs.



Deep Learning – Convolution Neural Network

- **Convolutions applied to local input regions**
 - The architecture most commonly used in image recognition.
 - Best for spatially related data
- **Example:**
 - Image Classification, Edge Detection



AutoML

Automated machine learning, AutoML is the process of automating the time-consuming, iterative tasks of machine learning model development.

- **Implement ML solutions without extensive programming knowledge**
- **Save time and resources**
- **Apply data science best practices**
- **Provide agile problem-solving**

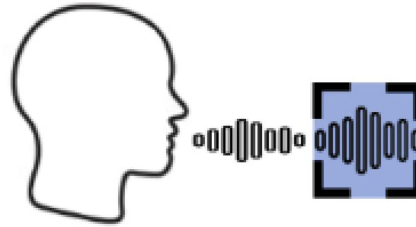


Select the Model

MPLAB® ML Suite Templates

- Keyword Spotting
- Vibration Classification
- Activity Recognition
- Gesture Recognition
- Audio Classification

Keyword Spotting



Vibration Classification



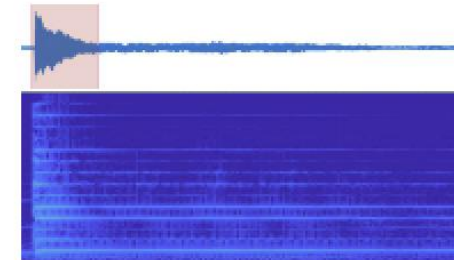
Activity Recognition



Gesture Recognition



Audio Classification





Model Pipeline and Training

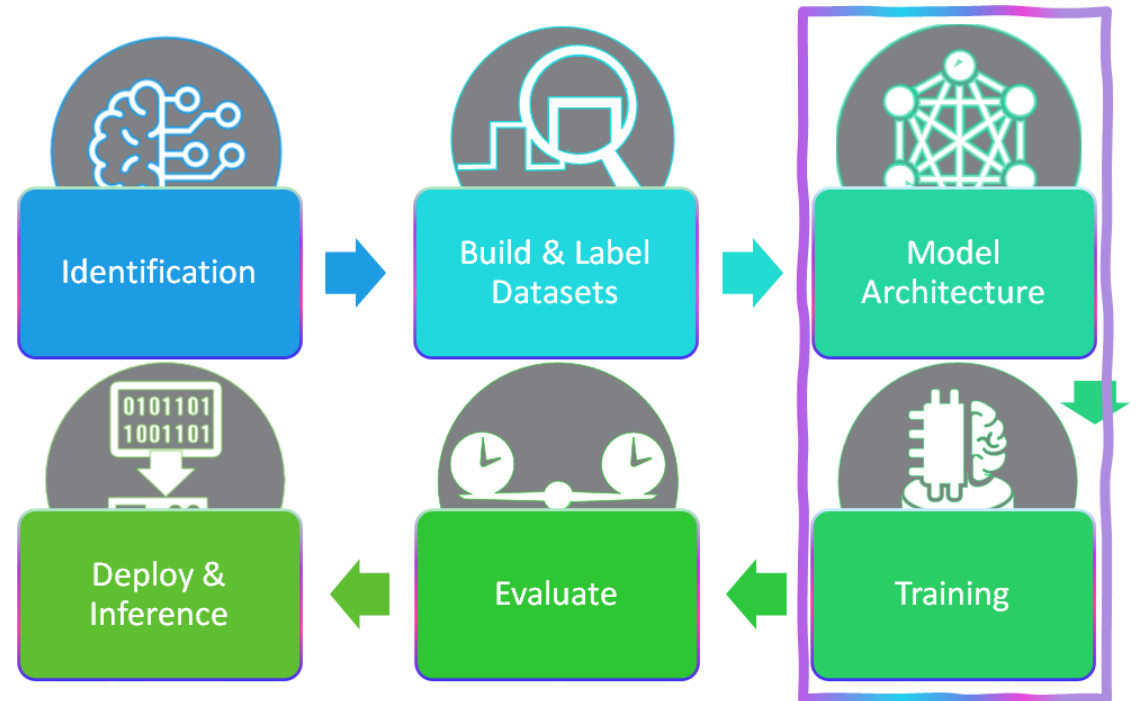
Boxing Recognition by Machine Learning

⚙️ Lab3 Model Pipeline and Training

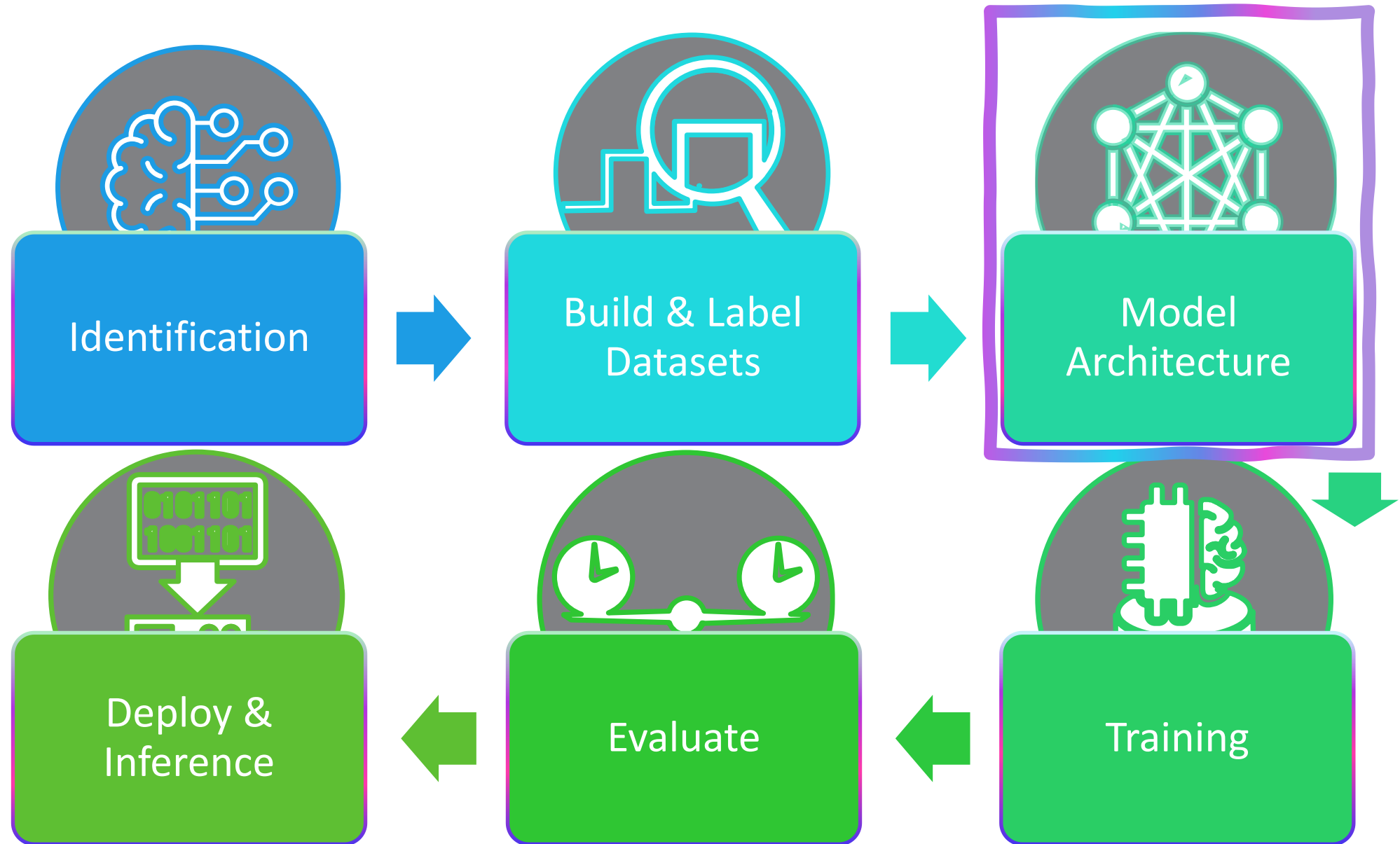
Boxing Recognition by Machine Learning

- Create a **query** for the data and then pass these data into the model builder.
- Select the **proper model** for the project, and the model builder will need to configure with some initial settings to help it understand how to best process the training sets.
- Start Training your unique model.

Let's go!



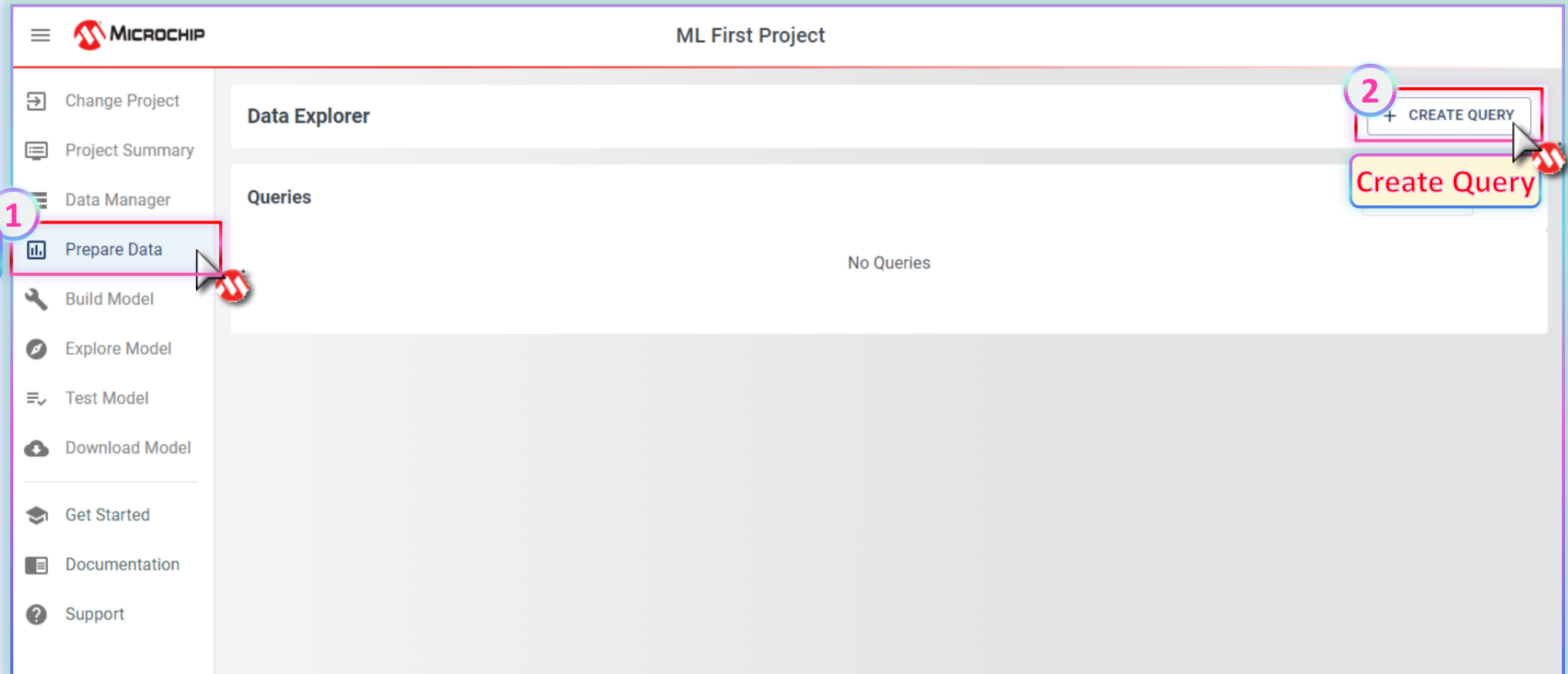
Machine Learning Workflow



⚙️ Lab3 Model Pipeline and Training

Step 1

- Click **Prepare Data** -> **Create Query**



⚙️ Lab3 Model Pipeline and Training

Step 2

- Select and Fill in all the blank.

Must choose again!!!!

The screenshot shows the 'Create Query' form with the following fields and annotations:

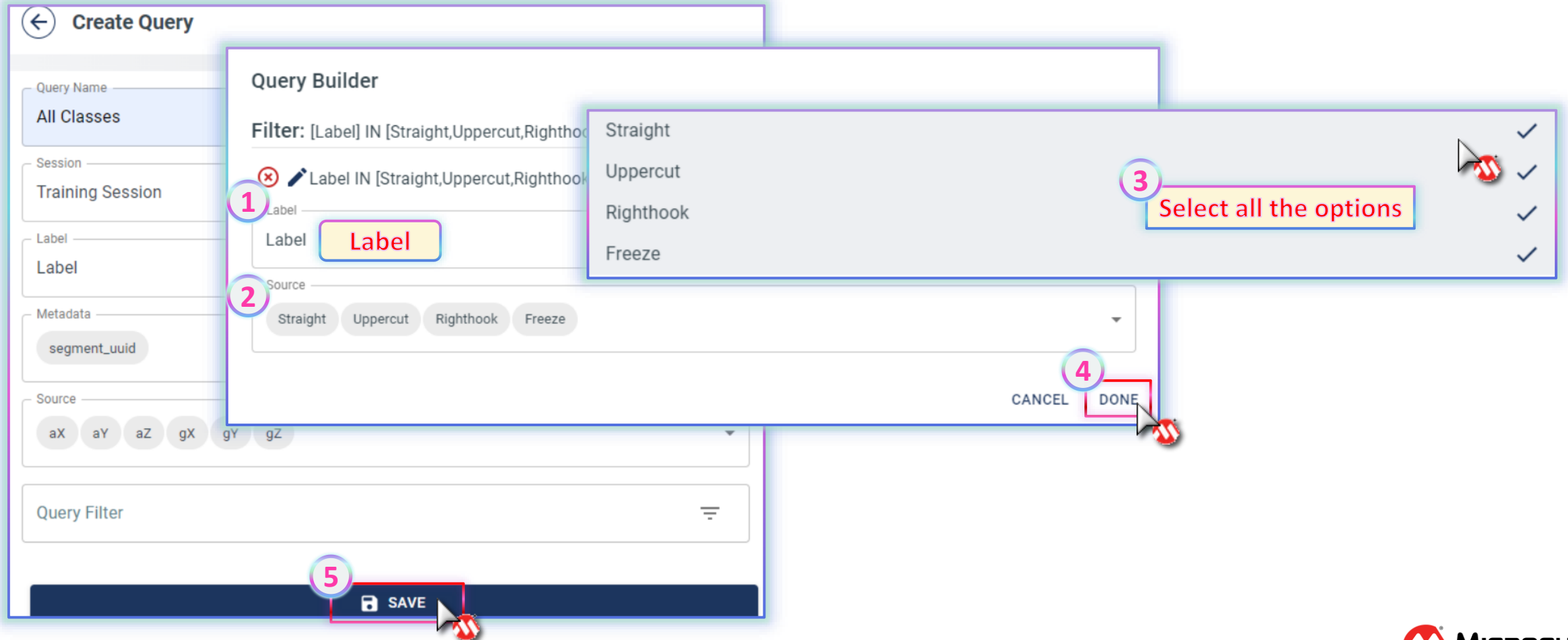
- 1** Query Name: All Classes (Callout: All Classes)
- 2** Session: Training Session (Callout: Training Session)
- 3** Label: Label (Callout: Label)
- 4** Metadata: segment_uuid (Callout: Segment_uuid)
- 5** Source: aX aY aZ gX gY gZ (Callout: aX aY aZ gX gY gZ)
- 5** Query Filter: (Callout: Must choose again!!!!)

SAVE

⚙️ Lab3 Model Pipeline and Training

Step 3

- Click  to open the list



Create Query

Query Name
All Classes

Session
Training Session

Label
Label

Metadata
segment_uuid

Source
aX aY aZ gX gY gZ

Query Filter

Query Builder

Filter: [Label] IN [Straight,Uppercut,Righthook]

1   Label IN [Straight,Uppercut,Righthook]

2 Source
Straight Uppercut Righthook Freeze

3 Select all the options

4 CANCEL DONE

5 SAVE

⚙️ Lab3 Model Pipeline and Training

Step 4

- Save the Query.

 **Create Query**

Query Name

All Classes

Session

Training Session

Label

Label

Metadata

segment_uuid

Source

aX aY aZ gX gY gZ

Query Filter

[Label] IN [Straight,Uppercut,Righthook,Freeze]

 **SAVE**

⚙️ Lab3 Model Pipeline and Training

Step 5

- Click  to build Cache.

Queries

DELETE 

...	NAME	COLUMNS	FILTER	SEGMENTS	LAST MODIFIED	CREATED DATE	UUID	STATUS	REBUILD
☐	All Classes	aX aY aZ gX gY gZ	[Label] IN [Straight,Uppercut,Righthook,Freeze]		7/17/2024 5:23:21 PM	7/17/2024 5:23:21 PM	c66e7cd8- 376c-4a4e- 87f5- 41b3fcc66c5	 NOT BUILT	

Rows per page: 25 1-1 of 1 |< < > >|



Queries

DELETE 

...	NAME	COLUMNS	FILTER	SEGMENTS	LAST MODIFIED	CREATED DATE	UUID	STATUS	REBUILD
☐	All Classes	aX aY aZ gX gY gZ	[Label] IN [Straight,Uppercut,Righthook,Freeze]	113	7/17/2024 5:23:43 PM	7/17/2024 5:23:21 PM	c66e7cd8- 376c-4a4e- 87f5- 41b3fcc66c5	 CHECKED	

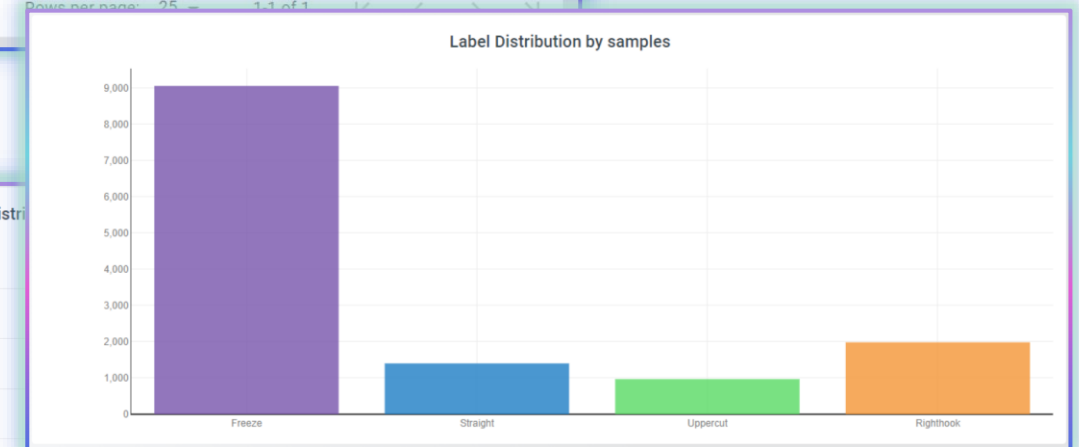
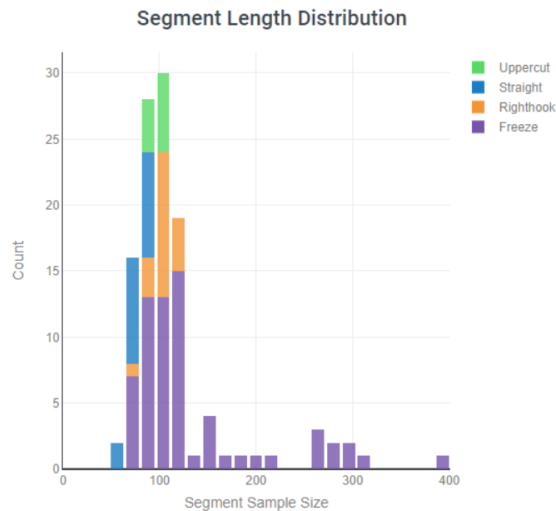
Rows per page: 25 1-1 of 1 |< < > >|

⚙️ Lab3 Model Pipeline and Training

Step 6

- Click the name of queries - **All classes** to see details.

Queries										DELETE	↺
...	NAME	COLUMNS	FILTER	SEGMENTS	LAST MODIFIED	CREATED DATE	UUID	STATUS	REBUILD		
☐	All Classes	aX aY aZ gX gY gZ	[Label] IN [Straight,Uppercut,Righthook,Freeze]	113	7/17/2024 5:23:43 PM	7/17/2024 5:23:21 PM	c66e7cd8-376c-4a4e-87f5-41b3fcc66c5	✓ CACHED	↺		



⚙️ Lab3 Model Pipeline and Training

Step 7

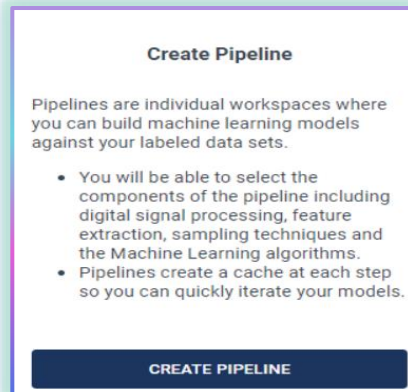
- Click Build Model

The screenshot displays the 'Build Model 1' interface. On the left sidebar, the 'Build Model' option is highlighted with a red box and a red arrow. The main area is titled 'Build Model 1' and contains a 'Create Pipeline' section with a description and a 'CREATE PIPELINE' button. Below this, there are five template cards: 'Keyword Spotting', 'Vibration Classification', 'Activity Recognition', 'Gesture Recognition', and 'Audio Classification'. Each card has a 'SELECT TEMPLATE' button. The 'Keyword Spotting' card features an icon of a head with sound waves. The 'Vibration Classification' card features an icon of a motor. The 'Activity Recognition' card features an icon of a person running. The 'Gesture Recognition' card features an icon of a person performing a gesture. The 'Audio Classification' card features an icon of a sound wave.

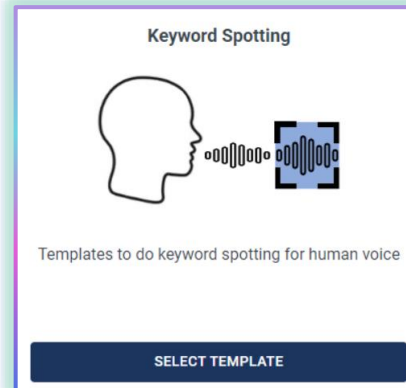
⚙️ Lab3 Model Pipeline and Training

Step 8

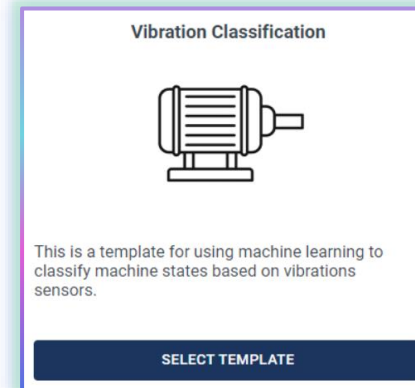
- MPLAB® ML Suite Templates, choose **Gesture Recognition**.



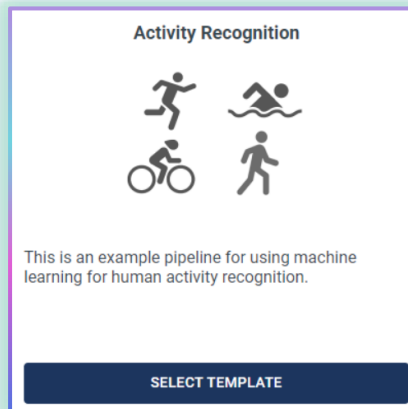
Create Pipeline



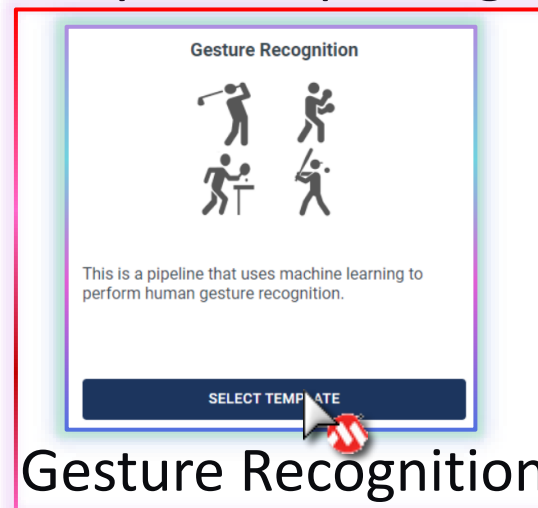
Keyword Spotting



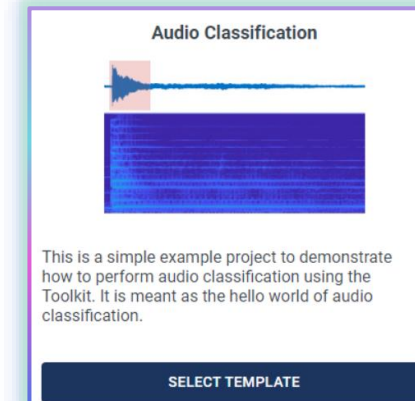
Vibration Classification



Activity Recognition



Gesture Recognition

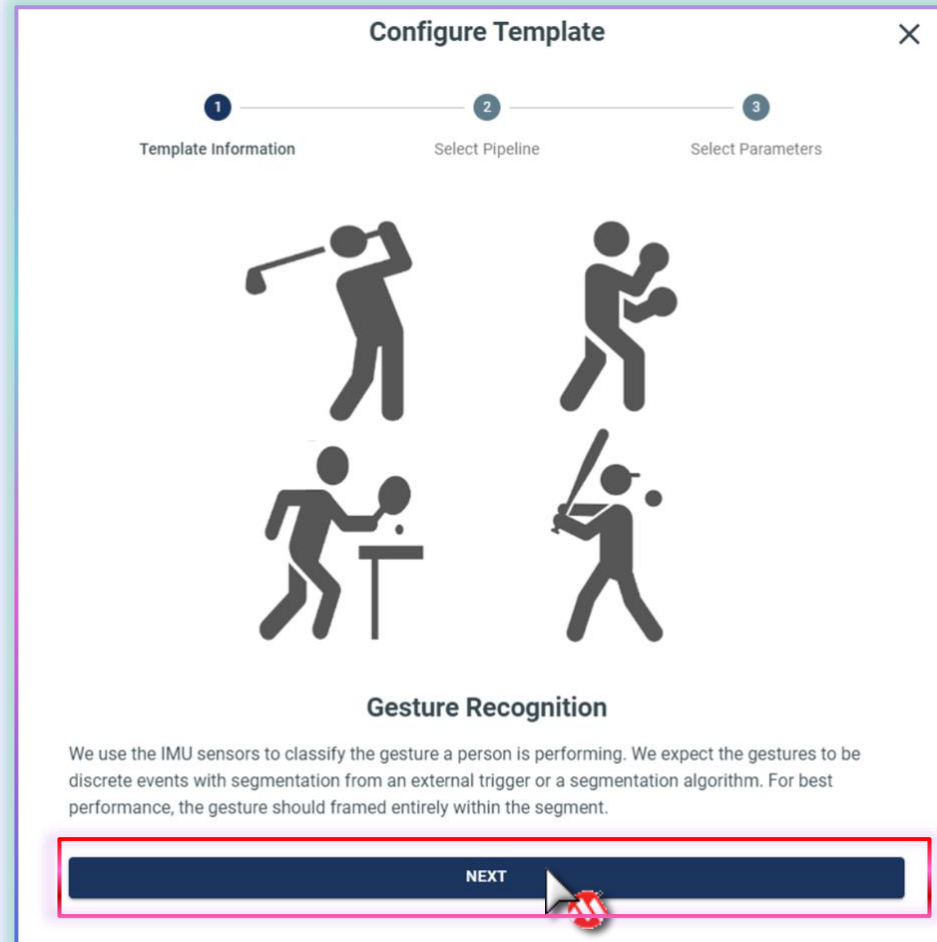


Audio Classification

⚙️ Lab3 Model Pipeline and Training

Step 9

- Slide to bottom and click next.



⚙️ Lab3 Model Pipeline and Training

Step 10

- It will auto select the pipeline that best fits.
- Choose the **1st** one.

Configure Template [X]

1 ✓ 2 3
Template Information Select Pipeline Select Parameters

1 Select pipeline that best fits your project.

This pipeline uses features optimized for separating discrete motions combined with supervised machine learning classification algorithms to identify different gestures.
Expected Sensors: AccelerometerX, AccelerometerY, AccelerometerZ, GyroscopeX, GyroscopeY, GyroscopeZ

This pipeline uses features optimized for separating discrete motions combined with supervised machine learning classification algorithms to identify different gestures.
Expected Sensors: AccelerometerX, AccelerometerY, AccelerometerZ

This pipeline uses features optimized for separating discrete motions combined with supervised machine learning classification algorithms to identify different gestures. This pipeline also includes a trigger used for segmentation.
Expected Sensors: AccelerometerX, AccelerometerY, AccelerometerZ, GyroscopeX, GyroscopeY, GyroscopeZ, Trigger

NEXT

⚙️ Lab3 Model Pipeline and Training

Step 11

- Name the model **Boxing Recognition**.
- Choose the Query.

Boxing Recognition

Select the query you create

CREATE PIPELINE

⚙️ Lab3 Model Pipeline and Training

Step 12

- Fill in all the sensor Columns.

Name	All Classes		
Label Column	Label		
Columns	aX aY aZ gX gY gZ		
Metadata Columns	segment_uuid		
Session	Training Session		

 The imported pipeline and the query All Classes have different sensor columns. To import this pipeline, please, select how to replace them.

Sensor Columns

AccelerometerX	→	aX	aX	×
AccelerometerY	→	aY	aY	×
AccelerometerZ	→	aZ	aZ	×
GyroscopeX	→	gX	gX	×
GyroscopeY	→	gY	gY	×
GyroscopeZ	→	gZ	gZ	×

CREATE PIPELINE

Lab3 Model Pipeline and Training

Step 13

- Click next.

ⓘ Pipeline Imported

When importing the pipeline the following parameters were reset to default values. Click the next button to review/update the changes to the following parameters:

Step: Isolation Forest Filtering

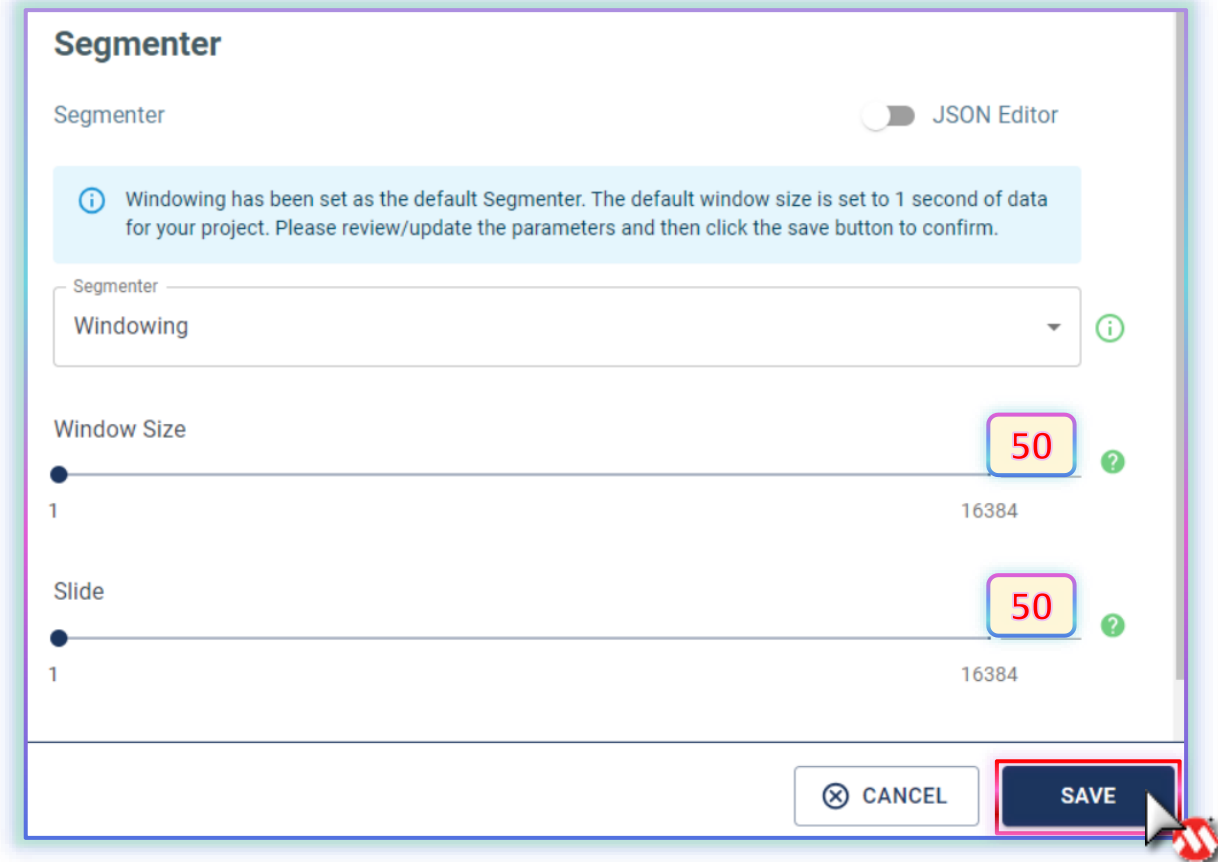
Parameter: Filtering Label

NEXT

⚙️ Lab3 Model Pipeline and Training

Step 14

- Adjust and confirm the segmenter.
- **Window Size: “100” refers to the window size in samples.**
 - By picking “Windowing” and “100” on 100 hz data, there is a 1 second window size. Which means every 1 second you will get a new classification.
- **Slide: Control the amount of overlap between this signal and the last.**



The screenshot shows a configuration window titled "Segementer". At the top right, there is a toggle switch for "JSON Editor" which is currently turned off. Below this, a light blue information box contains the text: "Windowing has been set as the default Segmenter. The default window size is set to 1 second of data for your project. Please review/update the parameters and then click the save button to confirm." The main configuration area includes a dropdown menu labeled "Segmenter" with "Windowing" selected. Below this are two sliders: "Window Size" and "Slide". Both sliders have a range from 1 to 16384. The "Window Size" slider is currently set to 50, and the "Slide" slider is also set to 50. Both values are displayed in a yellow box with a red border. At the bottom right of the window, there are two buttons: "CANCEL" and "SAVE". The "SAVE" button is highlighted with a red border and a mouse cursor is pointing at it.

⚙️ Lab3 Model Pipeline and Training

Step 15

- Adjust and confirm the outlier filter.

Outlier Filter

Sampler

JSON Editor

Review the values for the parameter: Filtering Label

Sampler

Isolation Forest Filtering

Filtering Label

Straight Uppercut Righthook Freeze

Outliers Fraction

0.01

0.05

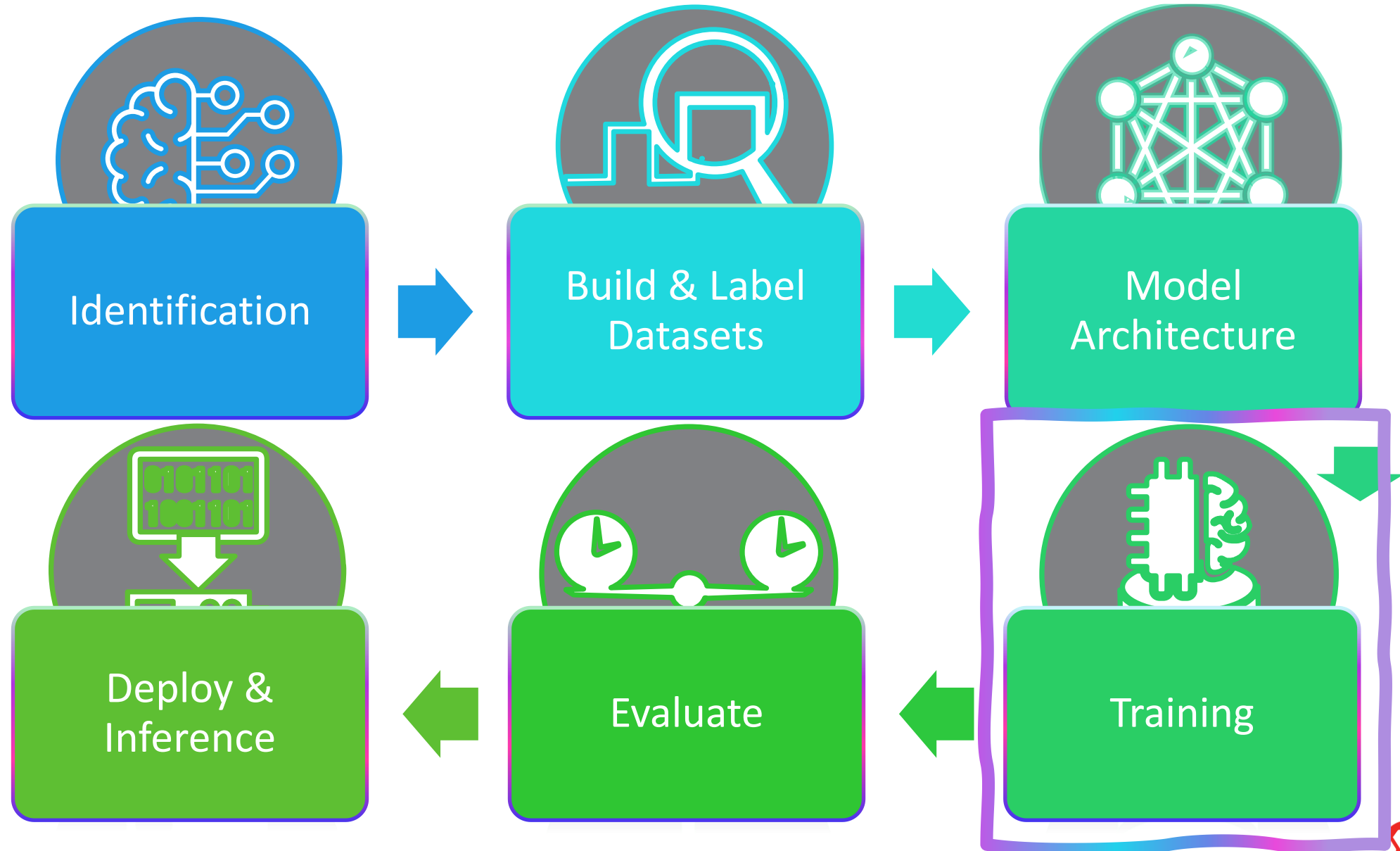
1

Assign Unknown

CANCEL

SAVE

Machine Learning Workflow



⚙️ Lab3 Model Pipeline and Training

Step 16

- Start running pipeline, and wait...

The screenshot displays the MPlab Machine Learning Suite interface for a pipeline named 'Boxing Recognition'. The interface is divided into several sections:

- Top Bar:** Contains a back arrow, the pipeline name 'Pipeline: Boxing Recognition', and three buttons: 'RUN PIPELINE' (highlighted with a red box and a 'Run Pipeline' callout), 'STOP PIPELINE', 'EXPORT', and 'SWITCH PIPELINE'.
- Pipeline Settings:** A section on the left with a pencil icon for editing.
- Preprocessing:** A section containing three steps: 'Input Query' (All Classes), 'Segmenter' (Windowing), and 'Sensor Transform' (Magnitude). Each step has a radio button and an edit icon.
- RESULT:** A section on the right with a 'RESULT' tab and a 'LOGS' tab (selected). The 'LOGS' tab shows a 'RUNNING' status with a progress indicator and a list of steps: 'Optimizing Query...', 'Query is optimized...', and 'AutoML Pipeline submitted to queue'. Below these steps, it indicates '10.0/10.0 CPU hours remaining' and provides a link to the Microchip website for more information.

A large blue arrow points from the 'RUN PIPELINE' button towards the 'LOGS' tab, indicating the flow of the process.

⚙️ Lab3 Model Pipeline and Training

Result

- Training Complete! 5 different models generated.

RESULT

LOGS ✓

TRAINING SUMMARY ✓

✓ COMPLETED

✓ Status: Running, Time: 0:05:03, Iteration 2/2, Step: 5/8, Name: Feature Selection, Population: 1

✓ Status: Running, Time: 0:05:08, Iteration 2/2, Step: 5/8, Name: Feature Selection, Population: 1

✓ Status: Running, Time: 0:05:13, Iteration 2/2, Step: 6/8, Name: Isolation Forest Filtering, Population: 1

✓ Status: Running, Time: 0:05:18, Iteration 2/2, Step: 7/8, Name: Min Max Scaling, Population: 1

✓ Status: Running, Time: 0:05:23, Iteration 2/2, Step: 8/8, Name: Model Training, Population: 16

✓ Status: Running, Time: 0:05:28, Iteration 2/2, Step: 8/8, Name: Model Training, Population: 16

✓ Status: Running, Time: 0:05:33, Iteration 2/2, Step: 8/8, Name: Model Training, Population: 16

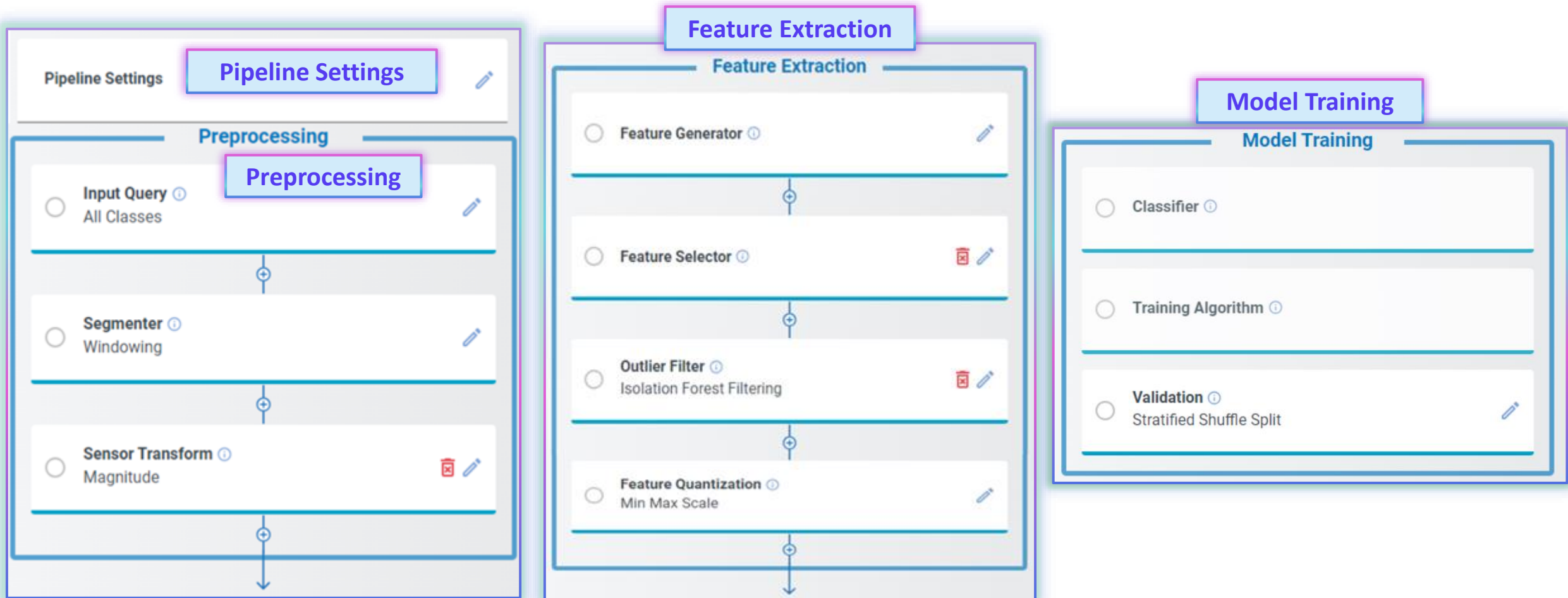
✓ Status: Running, Time: 0:05:38, Iteration 2/2, Step: 8/8, Name: Model Training, Population: 16

RESULT					
LOGS ✓					
TRAINING SUMMARY ✓					
MODEL NAME	ACCURACY	CLASSIFIER SIZE(B)	NUM. FEATURES	SENSITIVITY	F1-SCORE
Boxing_Recognition_rank_0	100	1376	2	100	100
Boxing_Recognition_rank_1	100	1096	3	100	100
Boxing_Recognition_rank_2	100	193	8	100	100
Boxing_Recognition_rank_3	100	239	8	100	100
Boxing_Recognition_rank_4	100	232	12	100	100

⚙️ Lab3 Model Pipeline and Training

Pipeline Settings

- Pipeline overview, Click  to edit.



Model Evaluation and Deploy

How good is the model?

5 models generated

- rank_4 will usually be the best

But may not be the smallest or fastest

- **Model tested against inputs**

Using stratified split

Confusion Matrix

- An important summary
- Compare actual against predicted

Model: light_pipeline_rank_4

← CHAN

CONFUSION MATRIX FEATURE VISUALIZATION FEATURE EMBEDDING FEATURE SUMMARY MODEL SUMMARY PIPELINE SUMMARY KNOWLEDGE PACK SUMMARY

	ambient	torch_blink	torch_high	torch_low	torch_mid	UNK	Support	Sensitivity(%)
ambient	9	0	0	0	0	0	9	100.00
torch_blink	0	8	0	0	1	0	9	88.89
torch_high	0	0	8	0	0	0	8	100.00
torch_low	0	0	0	9	0	0	9	100.00
torch_mid	0	0	0	0	9	0	9	100.00
Predicted	9	8	8	9	10	0	44	
Pos_Predic(%)	100.00	100.00	100.00	100.00	90.00		Acc(%)	97.73

	ambient	torch_blink	torch_high	torch_low	torch_mid	UNK	Support	Sensitivity(%)
ambient	44	0	0	0	0	0	44	100.00
torch_blink	0	19	0	0	25	0	44	43.18
torch_high	0	0	42	0	0	0	42	100.00
torch_low	0	0	0	44	0	0	44	100.00
torch_mid	0	0	0	0	44	0	44	100.00
Predicted	44	19	42	44	69	0	218	
Pos_Predic(%)	100.00	100.00	100.00	100.00	63.77		Acc(%)	88.53

	ambient	torch_blink	torch_high	torch_low	torch_mid	UNK	Support	Sensitivity(%)
ambient	9	0	0	0	0	0	9	100.00
torch_blink	0	8	0	0	1	0	9	88.89
torch_high	0	0	8	0	0	0	8	100.00
torch_low	0	0	0	9	0	0	9	100.00
torch_mid	0	0	0	0	9	0	9	100.00
Predicted	9	8	8	9	10	0	44	

	ambient	torch_blink	torch_high	torch_low	torch_mid	UNK	Support	Sensitivity(%)
ambient	35	0	0	0	0	0	35	100.00
torch_blink	0	31	0	0	4	0	35	88.57
torch_high	0	0	34	0	0	0	34	100.00
torch_low	0	0	0	35	0	0	35	100.00
torch_mid	0	0	0	0	35	0	35	100.00
Predicted	35	31	34	35	39	0	174	

Confusion Matrix

- The Confusion Matrix tab shows the averaged confusion matrix for the validation data.

Pos_Predic(%):

$$\frac{TP}{TP + FP} \times 100\%$$

Sensitivity (Recall)(%):

$$\frac{TP}{TP + FN} \times 100\%$$

Accuracy(%):

$$\frac{TP + TN}{TP + FN + FP + TN} \times 100\%$$

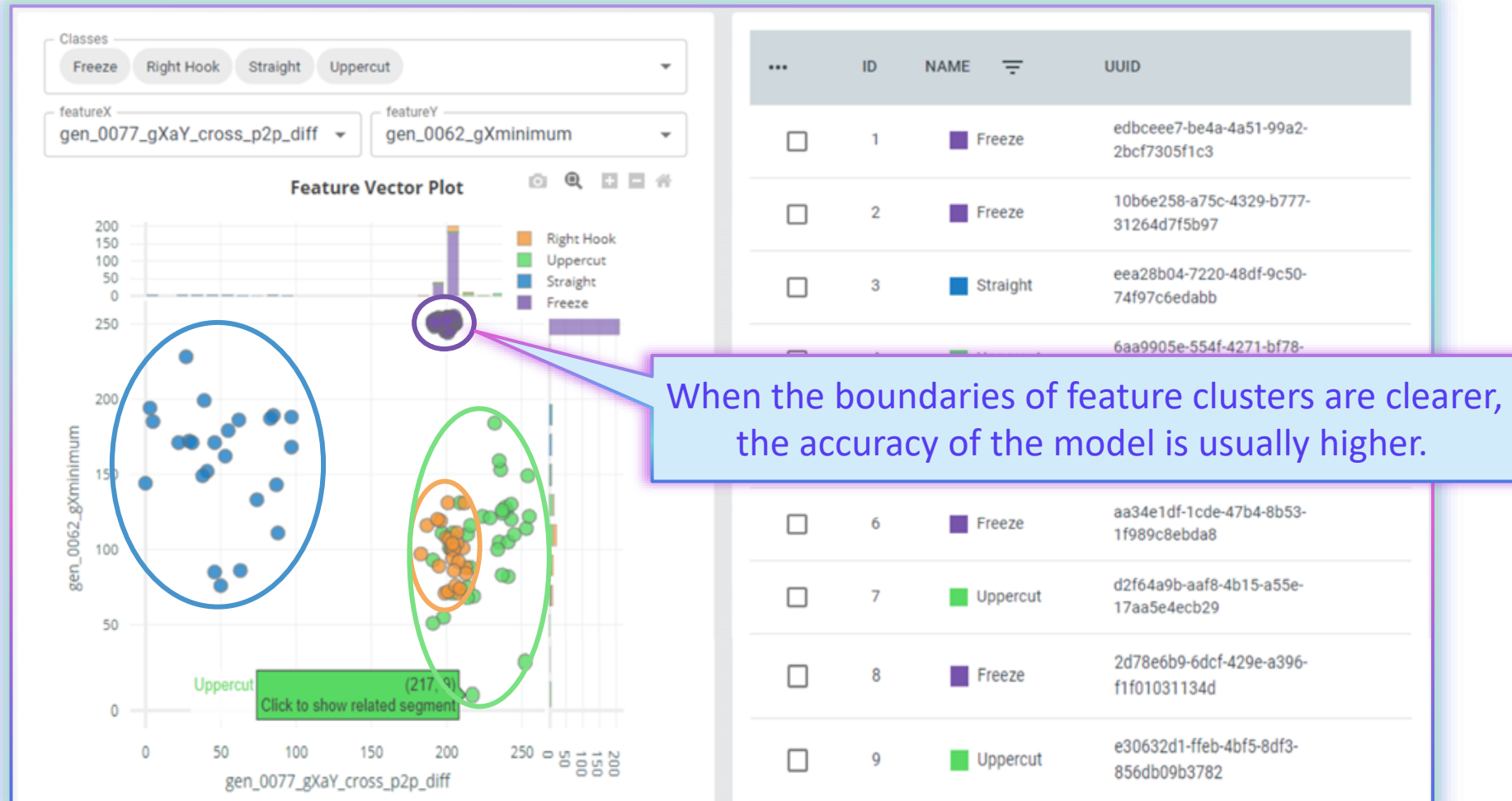
What's the prediction? Positive/Negative

Is the prediction correct? True/False

Confusion Matrix		Predicted	
		Yes	No
Actual	True	TP (True Positive)	FN (False Negative)
	False	FP (False Positive)	TN (True Negative)

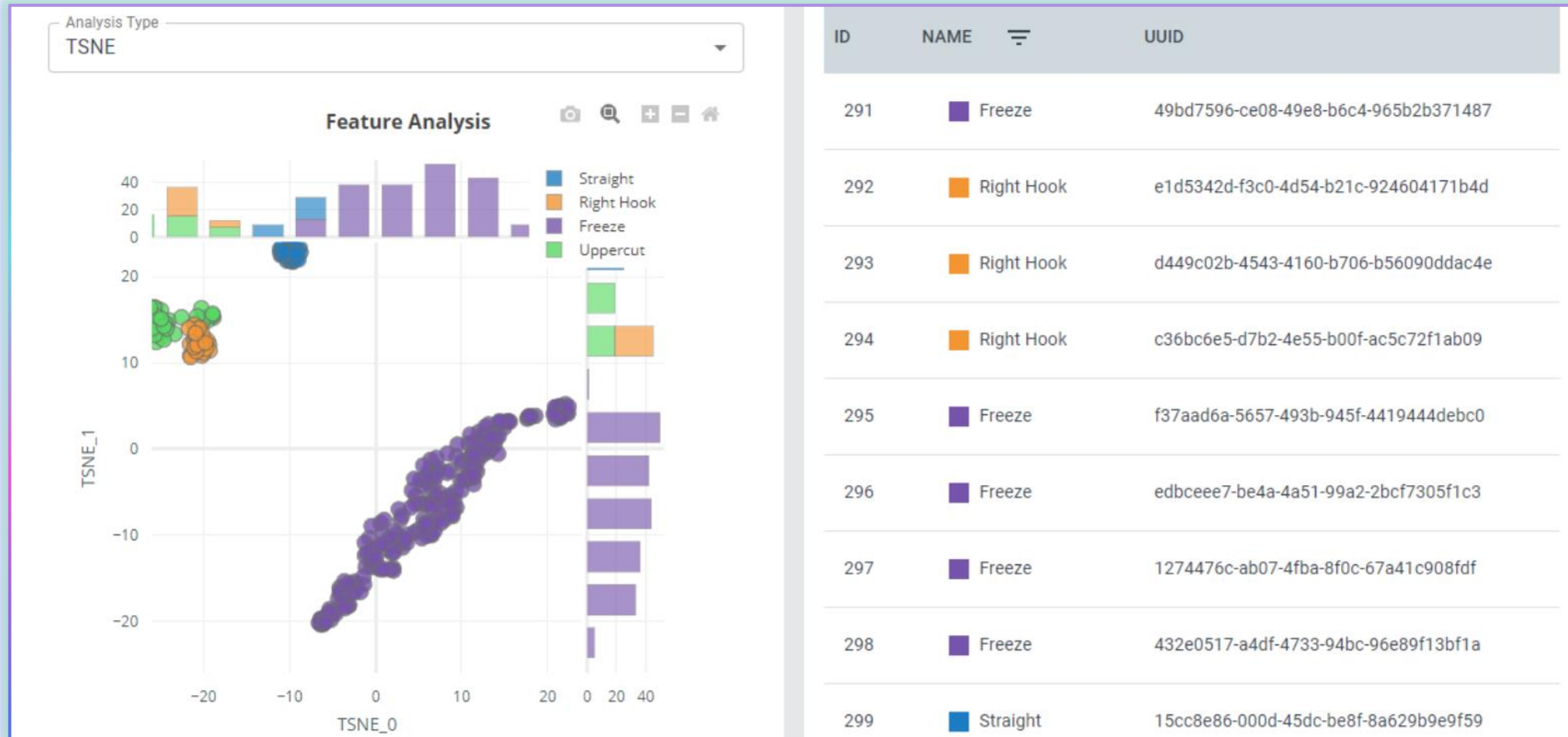
Feature Visualization

- Provide feature visualization as a 2-D comparison plot and feature insights chart.
- How well features are clustered according to classification.



Feature Embedding

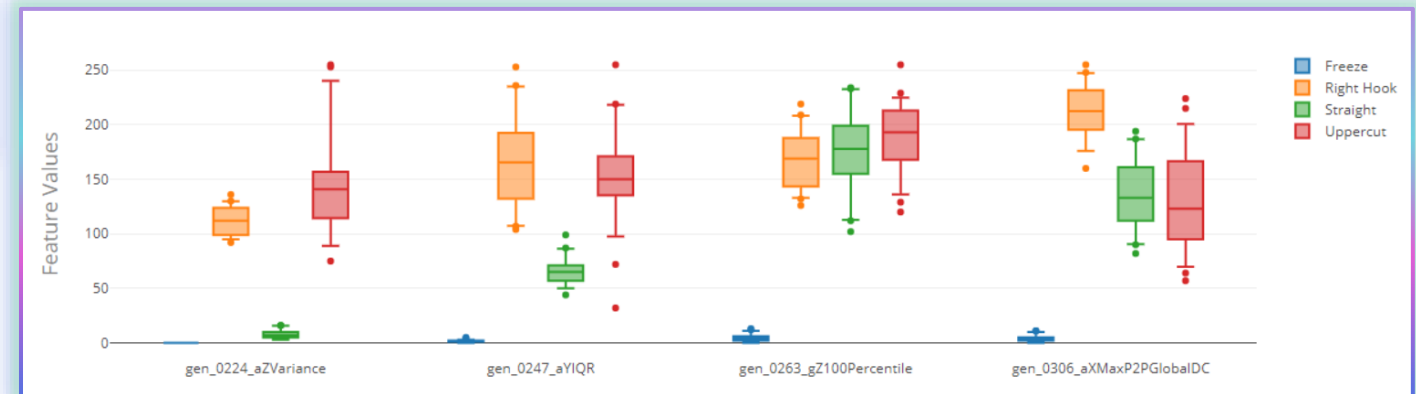
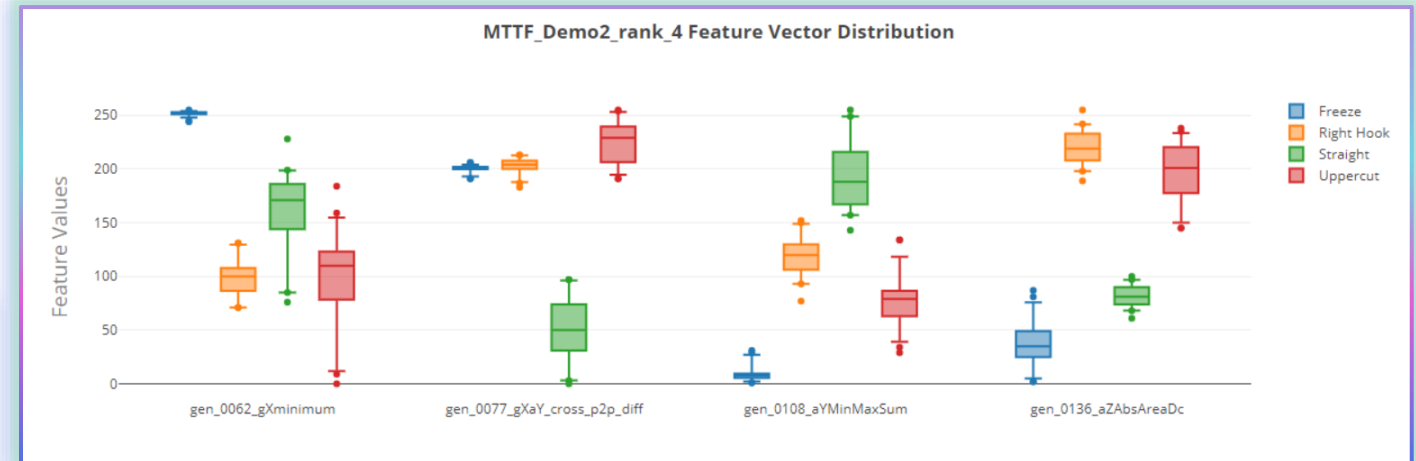
- The Feature Embedding tab provides three different embedding algorithms: UMAP, PCA, and TSNE.



Feature Summary

- Indication of what features are actually used to classify the results. Contain information about the features that are used during the feature extraction step of the Knowledge Pack.

FEATURE	CATEGORY	GENERATOR	SENSORS
gen_0062_gXminimum	Statistical	Minimum	gX
gen_0077_gXaY_cross_p2p_diff	Column Fusion	Two Column Peak To Peak Difference	gXaY
gen_0108_aYMinMaxSum	Amplitude	Global Min Max Sum	aY
gen_0136_aZAbsAreaDc	Area	Absolute Area of Low Frequency	aZ
gen_0224_aZVariance	Statistical	Variance	aZ
gen_0247_aYIQR	Statistical	Interquartile Range	aY
gen_0263_gZ100Percentile	Statistical	100th Percentile	gZ
gen_0306_aXMaxP2PGlobalDC	Amplitude	Global Peak to Peak of Low Frequency	aX



Model Summary

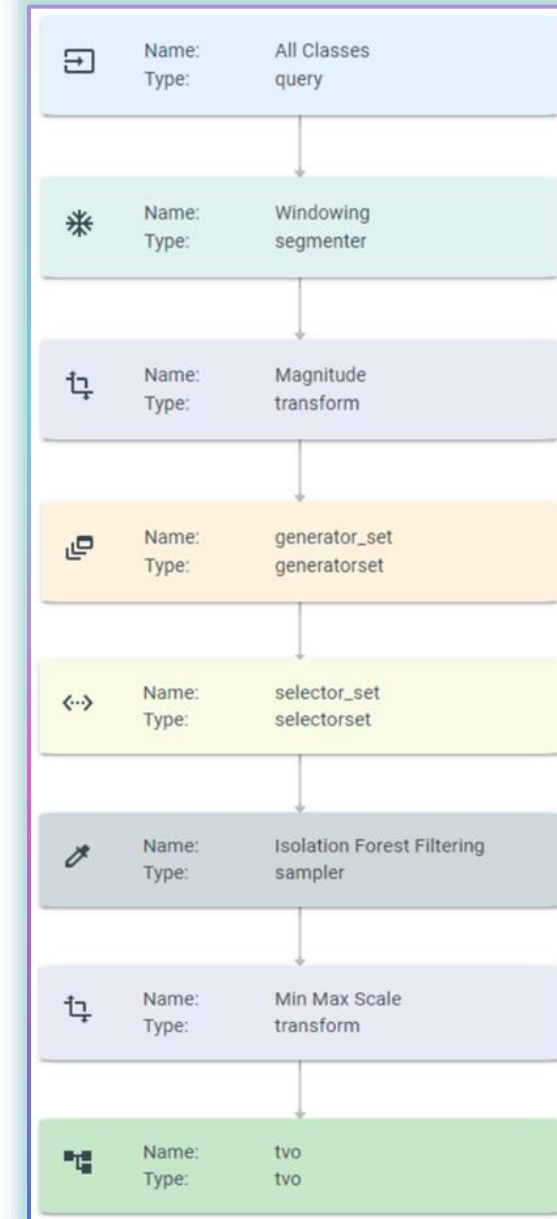
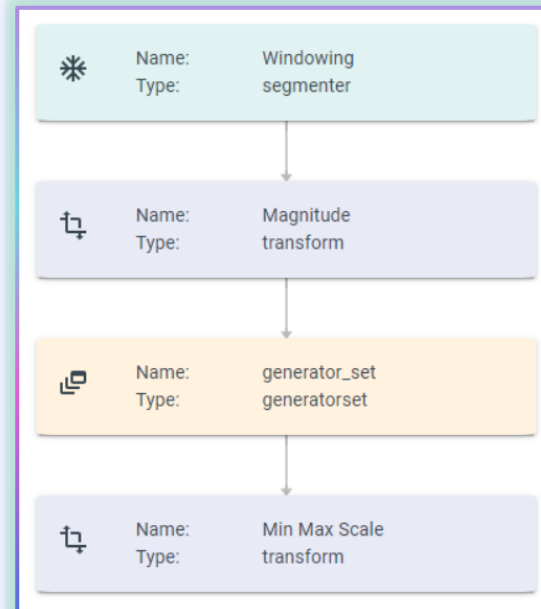
- The Model Summary tab describes the classifier, classifier parameters, and training algorithm that was used to generate the final model.

CLASSIFIER	UUID	OPTIMIZER	NEURONS	CLASSIFICATION_MODE	DISTANCE_MODE
PME	d6027a4e-5937-4547-b9fd-0dcf0381855f	RBF with Neuron Allocation Optimization	14	KNN	LSUP
1-1 of 1 < < > >					

CATEGORY	CATEGORY ID	AIF	PATTERN
Uppercut	4	62	126,237,72,201,149,154,129,104
Right Hook	2	43	119,196,129,207,124,130,194,160
Freeze	1	100	251,200,8,73,0,3,5,9
Straight	3	100	171,22,215,79,5,87,233,90
Uppercut	4	37	68,214,81,161,97,120,211,174
Uppercut	4	59	107,199,134,214,134,203,210,95
Right Hook	2	41	94,204,130,222,122,163,187,244
Right Hook	2	53	84,213,100,198,110,107,188,208
Uppercut	4	58	126,240,80,217,162,144,197,102
Uppercut	4	78	111,197,134,228,149,153,139,125
Right Hook	2	55	92,208,100,200,102,160,174,203
Uppercut	4	76	131,209,67,145,75,108,229,224
Uppercut	4	100	55,198,51,145,89,32,163,159
Right Hook	2	100	71,199,95,224,113,253,149,176
1-14 of 14 < < > >			

Pipeline & Knowledge Pack Summary

- Pipeline summary consists of data input, signal conditioning, signal preprocessing, feature extraction, *sampling*, and *model training*.
- Knowledge Packs summary consists of data input, signal conditioning, signal preprocessing, feature extraction, and *classification*.





Lab4

Model Evaluation and Deploy

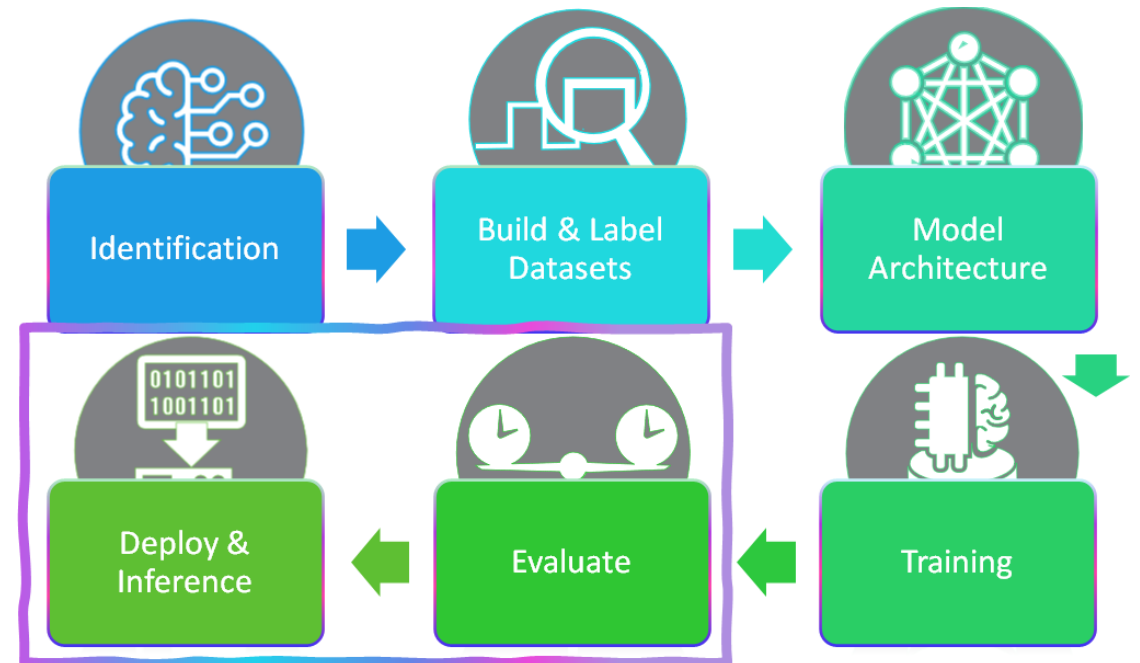
Boxing Recognition by Machine Learning

⚙️ Lab4 Model Evaluation and Deploy

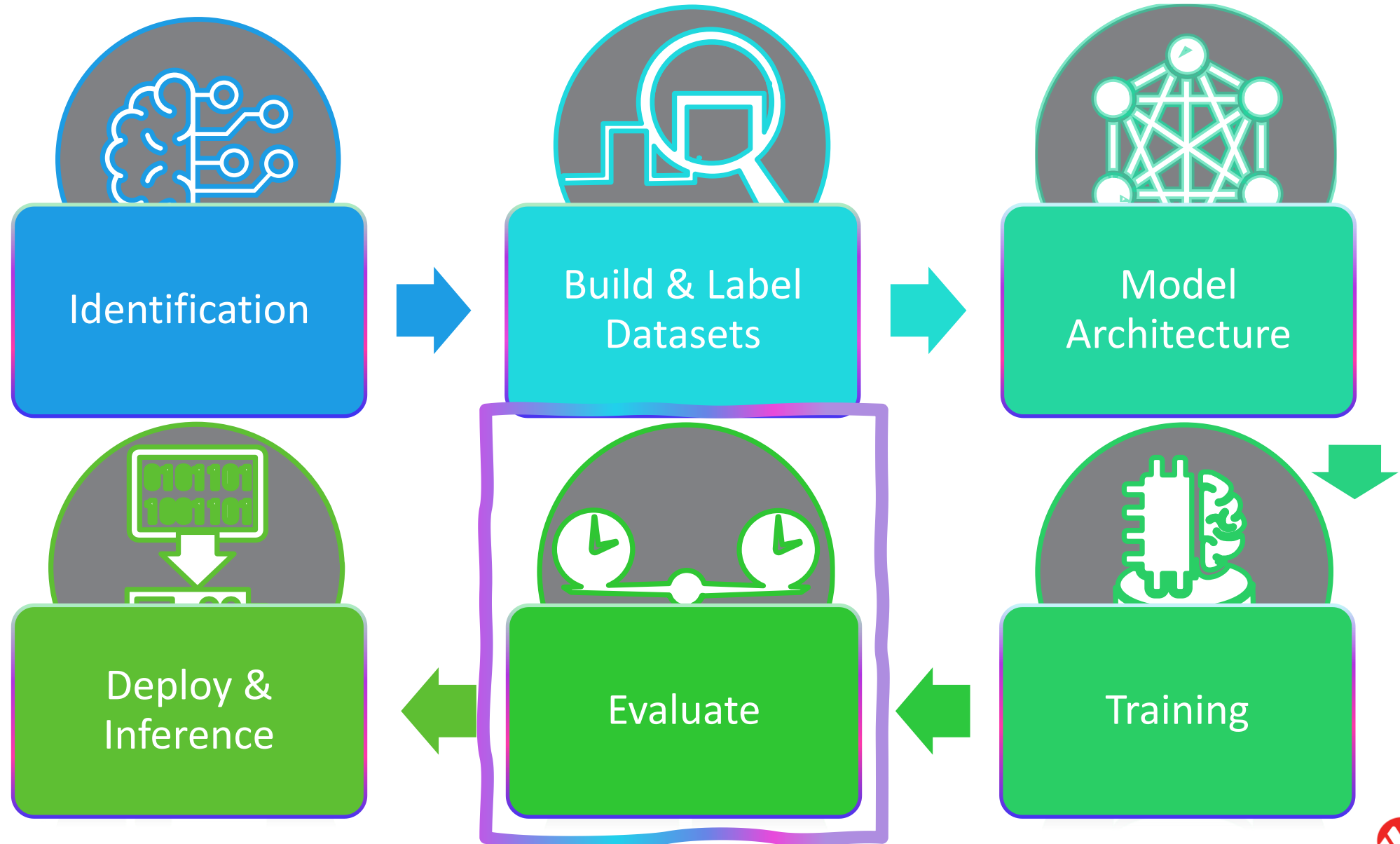
Boxing Recognition by Machine Learning

- Look and observe the results and select one of the model to **test** and inference.
- View the detail of the model by exploring the model.
- Download the trained model as a library file, **deployed** the model to the MCU.
- Start **inferencing** your model.

Let's go!



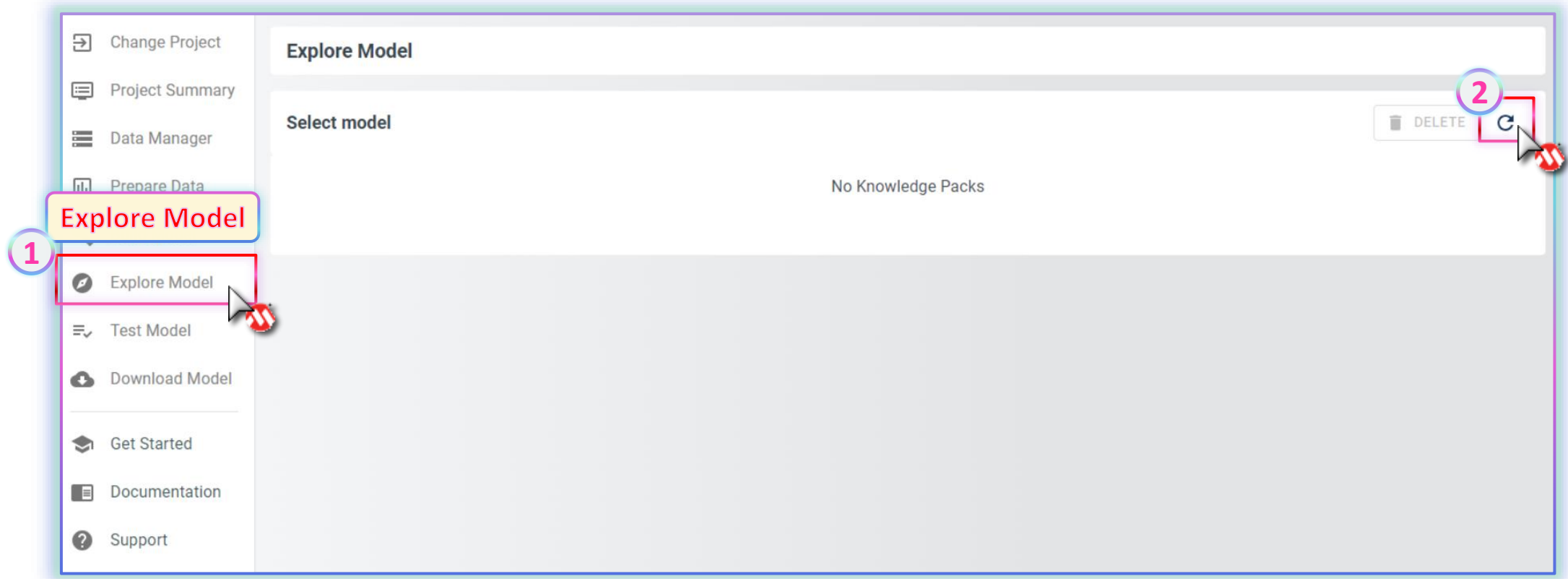
Machine Learning Workflow



⚙️ Lab4 Model Evaluation and Deploy

Step 1

- Click Explore model and ↻ to reload.



⚙️ Lab4 Model Evaluation and Deploy

Step 2

- The details of 5 model will show on the window.
- Click **rank_4** model to explore it. (**Usually rank_4 is the best**)

Select model										DELETE			
...	NAME	CLASSIFIER	ACCURACY	MODEL SIZE (BYTES)	FEATURE COUNT	PIPELINE	CREATED DATE	UUID	EXPLORE	TEST	DOWNLOAD	RENAME	DELETE
☐	Boxing_Recognition_rank_4	PME	100.00 %	259	12	Boxing Recognition	7/18/2024 3:53:03 PM	9faa0712-69c7-4d53-971f-a3914588a878					
☐	Boxing_Recognition_rank_3	PME	100.00 %	193	8	Boxing Recognition	7/18/2024 3:53:03 PM	c0c01323-e35a-41b8-a187-6b9dcf3bcc43					
☐	Boxing_Recognition_rank_2	PME	100.00 %	216	8	Boxing Recognition	7/18/2024 3:53:03 PM	44661fed-3f22-488e-b39e-71ba2666b1aa					
☐	Boxing_Recognition_rank_1	PME	100.00 %	1348	3	Boxing Recognition	7/18/2024 3:53:03 PM	acce015-efbb-429e-b955-f4597c2302b1					
☐	Boxing_Recognition_rank_0	Decision Tree Ensemble	100.00 %	1352	2	Boxing Recognition	7/18/2024 3:53:03 PM	d80ddabc-0e07-43c0-9751-0b500df91c6e					

⚙️ Lab4 Model Evaluation and Deploy

Step 3

- The window will show **confusion matrix, feature visualization, feature embedding...etc** of the model.

CONFUSION MATRIX FEATURE VISUALIZATION FEATURE EMBEDDING FEATURE SUMMARY MODEL SUMMARY PIPELINE SUMMARY KNOWLEDGE PAC >							
Validation							
	Freeze	Right Hook	Straight	Uppercut	UNK	Support	Sensitivity(%)
Freeze	43	0	0	0	0	43	100.00
Right Hook	0	5	0	0	0	5	100.00
Straight	0	0	5	0	0	5	100.00
Uppercut	0	0	0	8	0	8	100.00
Predicted	43	5	5	8	0	61	
Pos_Predic(%)	100.00	100.00	100.00	100.00		Acc(%)	100.00

Training							
	Freeze	Right Hook	Straight	Uppercut	UNK	Support	Sensitivity(%)
Freeze	215	0	0	0	0	215	100.00
Right Hook	0	26	0	0	0	26	100.00
Straight	0	0	25	0	0	25	100.00
Uppercut	0	0	0	39	0	39	100.00
Predicted	215	26	25	39	0	305	
Pos_Predic(%)	100.00	100.00	100.00	100.00		Acc(%)	100.00

Fold 0 - train							
	Freeze	Right Hook	Straight	Uppercut	UNK	Support	Sensitivity(%)
Freeze	172	0	0	0	0	172	100.00
Right Hook	0	21	0	0	0	21	100.00
Straight	0	0	20	0	0	20	100.00
Uppercut	0	0	0	31	0	31	100.00
Predicted	172	21	20	31	0	244	
Pos_Predic(%)	100.00	100.00	100.00	100.00		Acc(%)	100.00

Fold 0 - validation							
	Freeze	Right Hook	Straight	Uppercut	UNK	Support	Sensitivity(%)
Freeze	43	0	0	0	0	43	100.00
Right Hook	0	5	0	0	0	5	100.00
Straight	0	0	5	0	0	5	100.00
Uppercut	0	0	0	8	0	8	100.00
Predicted	43	5	5	8	0	61	
Pos_Predic(%)	100.00	100.00	100.00	100.00		Acc(%)	100.00

⚙️ Lab4 Model Evaluation and Deploy

Step 4

- Click **Test model**.

Change Project

Project Summary

Data Manager

Prepare Data

Build Model

Test Model

Download Model

Get Started

Documentation

Support

Model: Boxing_Recognition_rank_4

← CHANGE MODEL

Evaluate your model by selecting one or more capture files for classification.

Session: Training Session

Captures

▶ RECOGNIZE

■ SUMMARIZE

■ STOP

↻

...	NAME	ACCURACY	SEGMENTS	SIZE(MB)	RESULTS	CREATED	CAPTURE_UUID	CAPTURE_CONFIGURATION_UUID	DC
☐	Straight1.csv		37	0.12		2024-07-16T06:43:20.252Z	564450f1-8e51-4572-b3a3-722e00d6a8e6		
☐	Upper1.csv		27	0.17		2024-07-16T06:43:20.961Z	7ab77451-9556-4ebe-b460-85222fd37493		
☐	Right1.csv		49	0.22		2024-07-16T06:43:21.675Z	a3d470d2-bb9f-4749-a7e7-65c7e6f7ee53		

⚙️ Lab4 Model Evaluation and Deploy

Step 5

- Select ☒ the files which want to test, may be multiple.
- Click **RECOGNIZE** and wait.

The screenshot shows the Lab4 Model Evaluation interface. At the top, there is a light blue informational box that says: "Evaluate your model by selecting one or more capture files for classification." To the right of this box is a "Session" dropdown menu currently set to "Training Session". Below the session menu is a "RECOGNIZE" button highlighted with a red box and a pink circle with the number 2. To the left of the "RECOGNIZE" button is a "Captures" table. The table has columns: NAME, ACCURACY, SEGMENTS, SIZE(MB), RESULTS, CREATED, CAPTURE_UUID, and CAPTURE_CONFIGURATION_UUID. There are three rows of captures. The first row is "Right1.csv" with 49 segments and 0.22 MB. The second row is "Right2.csv" with 0 segments and 0.20 MB. The third row is "Straight1.csv" with 37 segments and 0.12 MB. In the first column of the table, there are checkboxes. The checkboxes for "Right2.csv" and "Straight1.csv" are checked and highlighted with red boxes and pink circles with the number 1. To the right of the table are three buttons: "RECOGNIZE" (with a play icon), "SUMMARIZE" (with a list icon), and "STOP" (with a square icon). There is also a refresh icon (circular arrow) to the right of the "STOP" button.

...	NAME	ACCURACY	SEGMENTS	SIZE(MB)	RESULTS	CREATED	CAPTURE_UUID	CAPTURE_CONFIGURATION_UUID	DC
☐	Right1.csv		49	0.22		2024-07-16T06:43:21.675Z	a3d470d2-bb9f-4749-a7e7-65c7e6f7ee53	22113093-e1ac-42ef-ac40-0e95a9b9acc3	R
☒	Right2.csv		0	0.20		2024-07-22T09:41:07.030Z	792f7714-f3e3-437e-9116-af979dd66a3a	22113093-e1ac-42ef-ac40-0e95a9b9acc3	R
☒	Straight1.csv		37	0.12		2024-07-16T06:43:20.252Z	564450f1-8e51-4572-b3a3-722e00d6a8e6	22113093-e1ac-42ef-ac40-0e95a9b9acc3	R

⚙️ Lab4 Model Evaluation and Deploy

Step 6

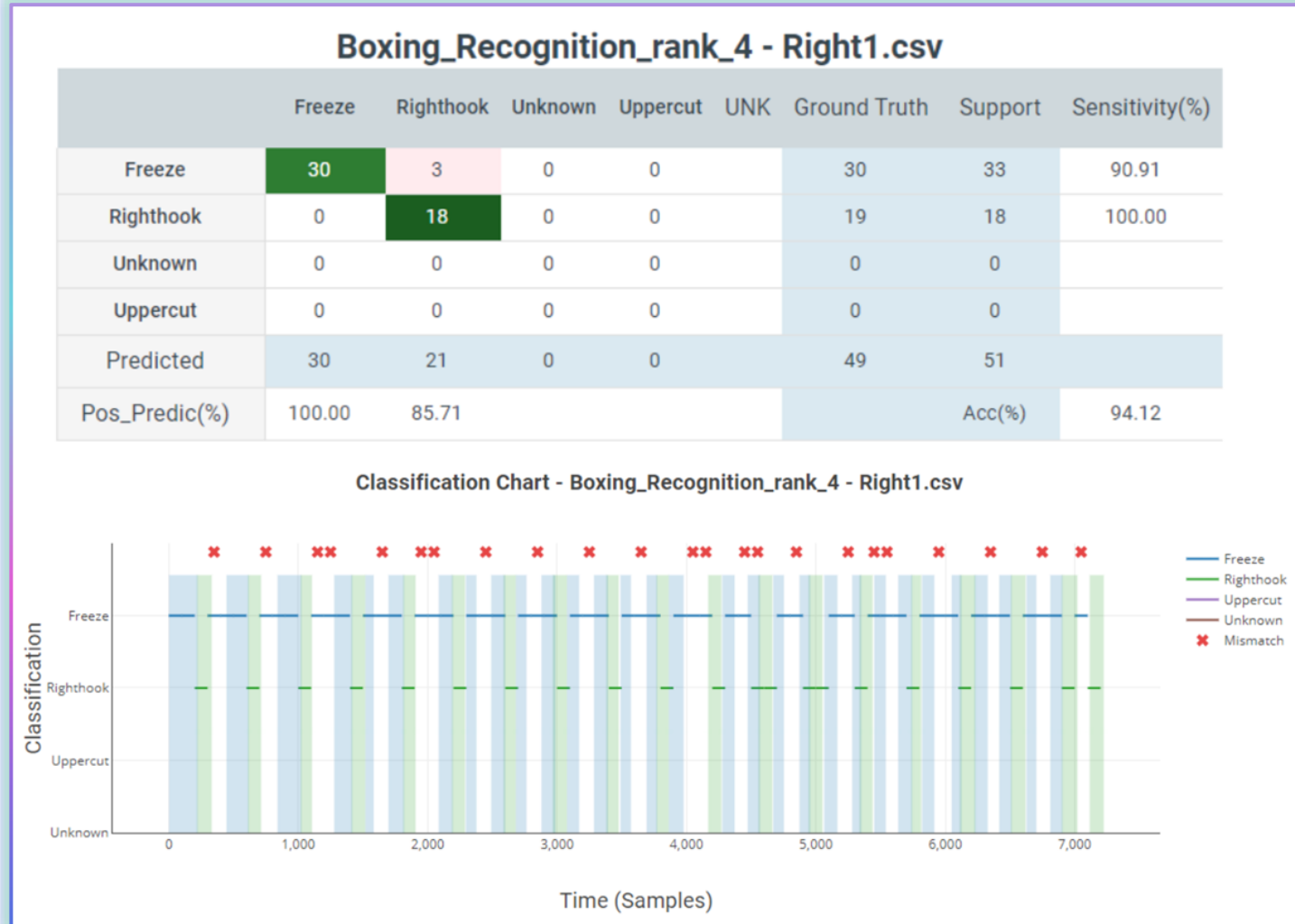
- The result will show in the result field.
- Click  RESULTS to view details.

Captures											
▶ RECOGNIZE ■ SUMMARIZE ■ STOP ↺											
...	NAME	ACCURACY	SEGMENTS	SIZE(MB)	RESULTS	CREATED	CAPTURE_UUID	CAPTURE_CONFIGURATION_UUID			
☒	Right1.csv	94.1	49	0.22	RESULTS	2024-07-16T06:43:21.675Z	a3d470d2-bb9f-4749-a7e7-65c7e6f7ee53	22113093-e1ac-42ef-ac40-0e95a9b9acc3			
☐	Right2.csv		32	0.20	RESULTS	2024-07-22T09:41:07.030Z	792f7714-f3e3-437e-9116-af979dd66a3a	22113093-e1ac-42ef-ac40-0e95a9b9acc3			
☐	Straight1.csv		37	0.12		2024-07-16T06:43:20.252Z	564450f1-8e51-4572-b3a3-722e00d6a8e6	22113093-e1ac-42ef-ac40-0e95a9b9acc3			
☒	Straight2.csv	63.5	43	0.19	RESULTS	2024-07-22T09:41:07.631Z	c16e90e1-f502-4bb3-bbda-46eefb385009	22113093-e1ac-42ef-ac40-0e95a9b9acc3			
☐	Straight3.csv		38	0.15		2024-07-22T09:41:08.231Z	29cd7292-3ce1-4143-8442-df9bbe7defac	22113093-e1ac-42ef-ac40-0e95a9b9acc3			

⚙️ Lab4 Model Evaluation and Deploy

Step 7

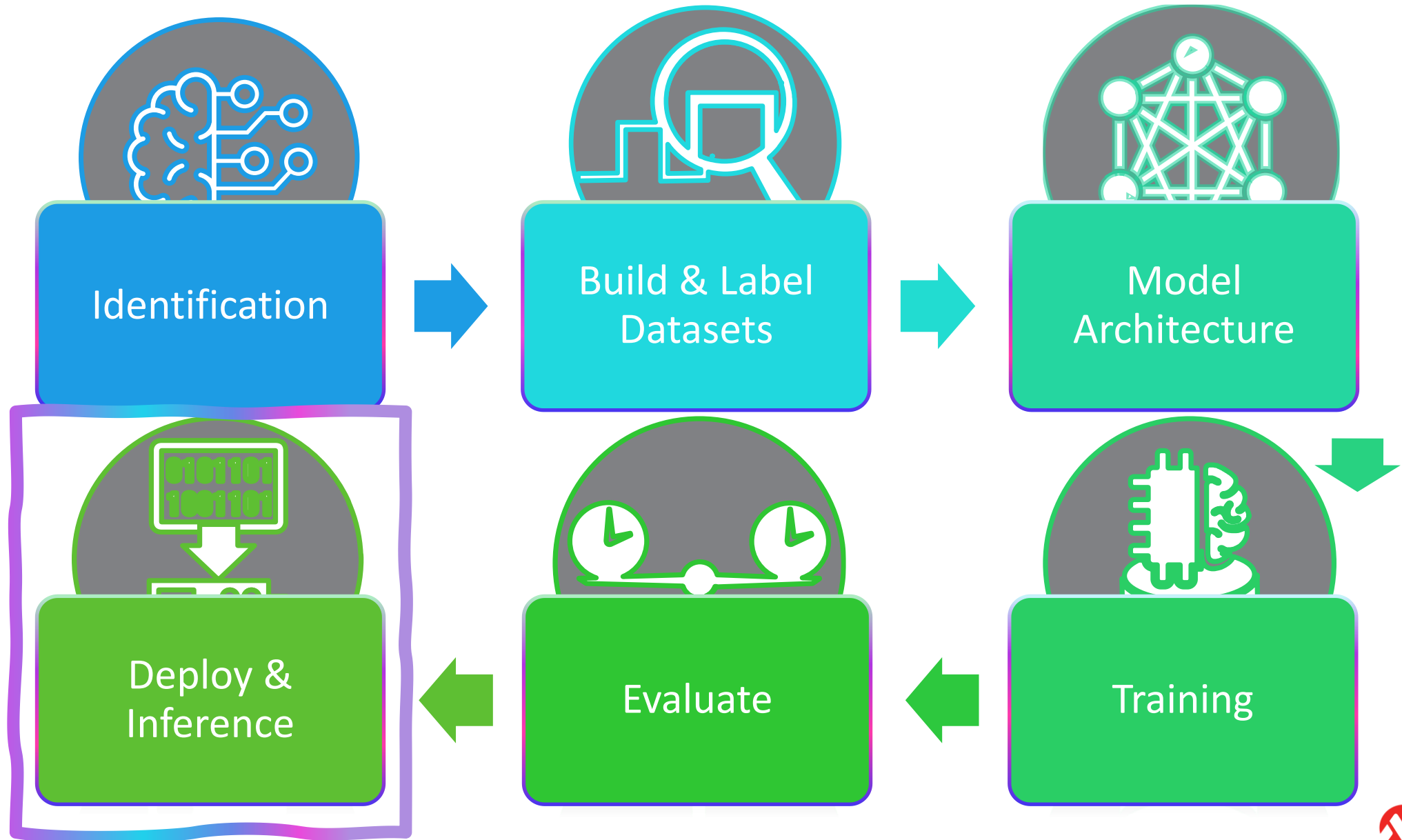
- Slide to the bottom to view detail.



Review from Start to End

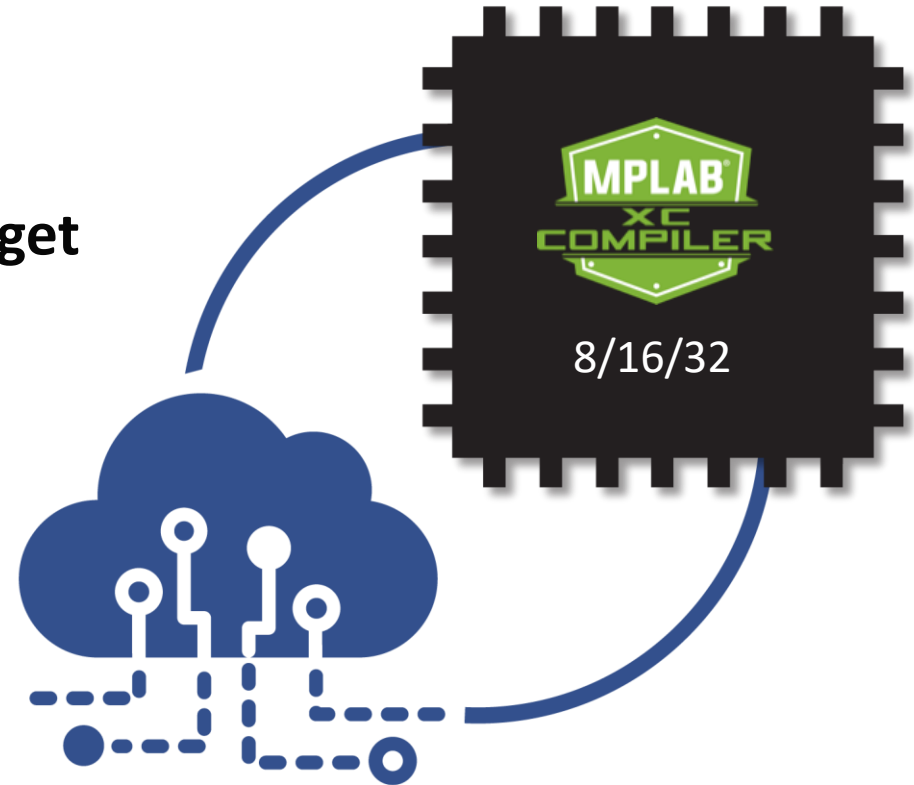
階段		步驟	拳擊姿態識別
Project Summary		Segment Labels (標籤建立)	決定需要識別的拳擊動作類別以及元數據分類
		Metadata (元數據建立)	決定需要元數據劃分類型 (可選)
Data Manager		Data Labeling (數據標籤標註)	標記數據集中每段數據所對應的拳擊動作
		Metadata Given (元數據賦予)	記錄感測器型號、測量條件、動作執行者資訊等 (可選)
Prepare Data		Create Query (查詢建立)	根據標籤與元數據來準備適當的拳擊數據
Build Model	Preprocessing (數據預處理)	Select Template Model (選擇模型範本)	選擇合適的模型範本 (如 Gesture Recognition)
		Input Query (輸入查詢)	使用 Create Query 產生的數據作為模型輸入數據
		Segmenter (資料分割器)	依據固定時間窗口(如 1 秒)或動作觸發事件切割數據片段
		Sensor Transform (感測器轉換)	進行感測數據預處理，如座標旋轉、標準化或濾波
	Feature Extraction (特徵提取)	Feature Generator (特徵生成器)	計算每個片段的統計特徵 (如平均加速度, 最大角速度, 頻率域特徵等)
		Feature Selector (特徵選擇器)	選擇對於區分不同拳擊動作最有效的特徵
		Outlier Filter (異常值過濾器)	去除明顯異常的動作，如誤觸感測器導致的錯誤數據
		Feature Quantization (特徵量化)	透過數值縮放與離散化，使特徵符合模型輸入需求
	Model Training (模型訓練)	Classifier (分類器)	設定模型分類類型，如決策樹、SVM、神經網絡等
		Training Algorithm (訓練演算法)	選擇並執行合適的訓練算法，如隨機森林、KNN、深度學習等
		Validation (驗證)	使用交叉驗證或留存數據集驗證模型的準確性
Explorer Model		Select Model (選擇模型)	選擇適合的 AI 模型來進行識別
		Model Explorer (模型探索)	查看模型的分類效果 (如 Confusion Matrix, feature visualization/embedding)
Test Model		Test Model (測試模型)	使用測試數據評估模型準確度
Download Model		Model Deploy (模型部署)	下載並部署模型到目標設備，如微控制器 (MCU) 或邊緣設備

Machine Learning Workflow



Deploy the model

- **The model exists in the cloud**
 - Mathematical filters
 - Transformations
 - Python code
- **Needs to be converted to run on Embedded Target**
 - Converting precision
 - Doubles->char or int
 - Possibly involving regeneration
 - Convert language
 - Python to 'C' library
 - Account for architecture (8/16/32bit)
- **Implement calls to model**
 - Integrate it into your code



Silicon Platforms for ML Edge

Smart Embedded Vision

High performance video

- Real time / near real time
- High frame rate / large image size
- Multiple input streams
- Simultaneous multiple DNNs

Input sensors / data

Camera: Up to 4 interfaces

MIPI, parallel, HDMI

Mic: Multiple audio streams

Additional sensors

PolarFire
/ PF SoC
FPGA

Smart HMI

- Visual / audio / touch / gesture
- Recognition and interaction
- Single stream

Input sensors / data

Camera: MIPI, parallel, USB

Mic: Multiple audio streams

Touch surface

SAMA7
MPU

Smart Predictive Maintenance

- Time series / linear sensor data
- Simultaneous, multiple sensor
- Variety of sensor type supported

Input sensors / data

IMU (Inertial Measurement Unit)

Temp, Current, Pressure

Mic: Audio Stream

SAMD21
MCU

AVR
MCU

Deploy the model

- **Select the optimal pipeline**
 - That achieves your desired performance
 - Verify the results with test data
 - Take more samples
- **Select your compiler target**
 - Select the processor
- **Create and download the knowledge pack**
 - zip file
 - Library (libmplabml.a)
 - Header files (6)
 - Source output is possible
 - Separate License

Project: lighten [↗](#)

← Model: lighten_torch_pipeline_rank_4

Select a Target

Compiler

MPLAB XC8

MPLAB XC16

MPLAB XC32

Windows x86_64

x86 GCC Generic

Development Platform

[SELECT PLATFORM](#)

Project: lighten [↗](#)

Download Knowledge Pack

Platform - MPLAB XC32 [←](#)

Format [▼](#)

Library [▼](#)

Processor [▼](#)
Microchip ATSAME54P20A

Float Options [▼](#)
Soft

Compiler [▼](#)
MPLAB XC32 4.00

Hardware Accelerator [▼](#)
CMSIS

Data Source [▼](#)
Custom

Application [▼](#)
AI Model Runner

Output [▼](#)
Serial

Debug/Profiling Settings [▼](#)

[DOWNLOAD](#)

Deploy the model

- To interface with a [Knowledge Pack](#), use the files that get downloaded with your Knowledge Pack.
 - [Kb.h](#)
 - [sml_recognition_run.c](#)
- For this demo only 3 function calls needed
 - [kb_model_init\(\)](#) : Before any other functions are called, must initialize the Knowledge Pack.
 - [kb_reset_model\(\)](#) : Called after every run of the model
 - [kb_run_model\(\)](#) : Add new data to the model and classify it

⚙️ Lab4 Model Evaluation and Deploy

Step 8

- Select Download Model

Download Model

Select model

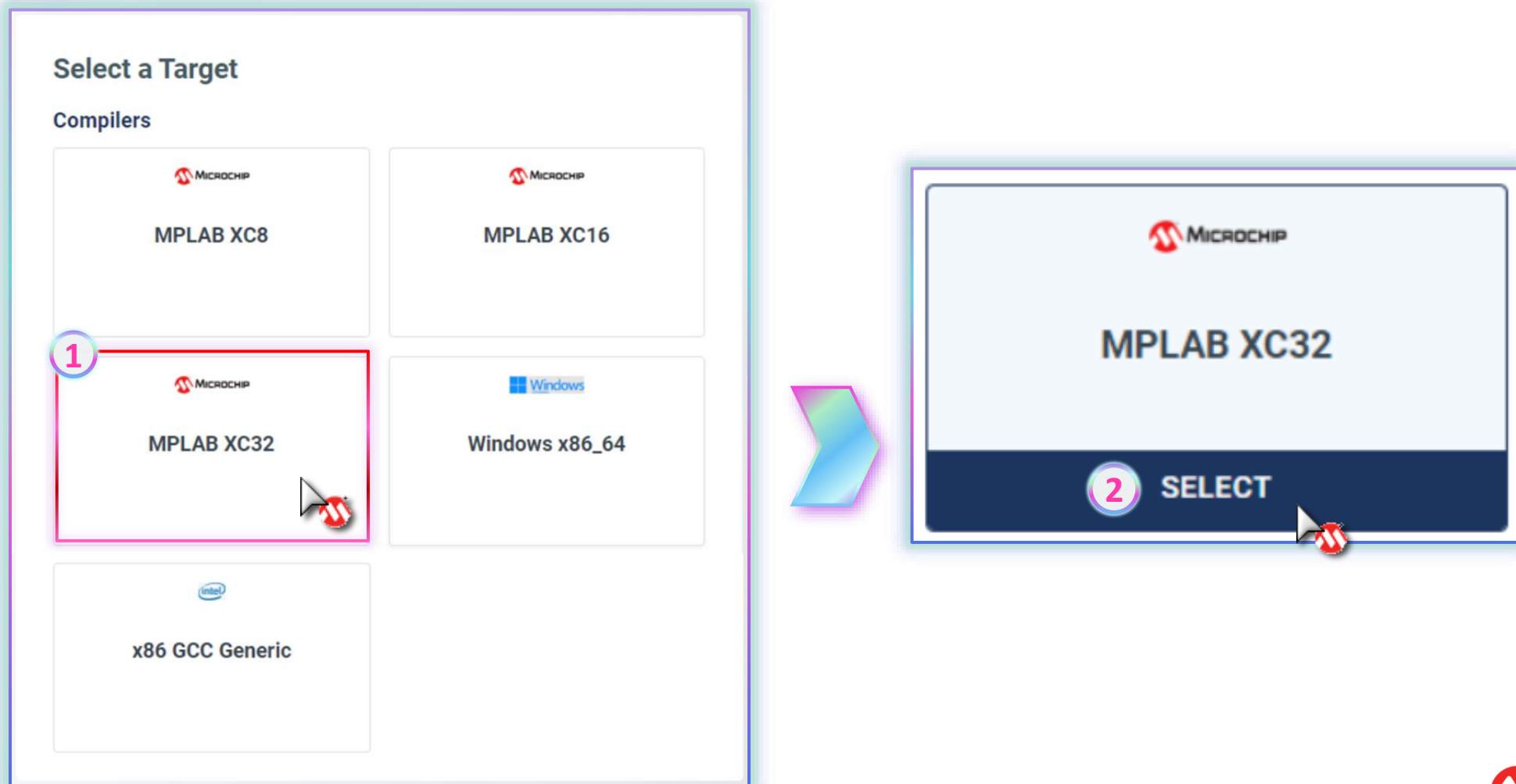
DELETE

...	NAME	CLASSIFIER	ACCURACY	MODEL SIZE (BYTES)	FEATURE COUNT	PIPELINE	CREATED DATE	UUID	EXPLORE	TEST
☐	Boxing_Recognition_rank_4	Decision Tree Ensemble	100.00 %	1370	12	Boxing Recognition	7/22/2024 5:03:02 PM	5dbdcf02-e399-4e3c-9bc4-d773cfa5cb62		
☐	Boxing_Recognition_rank_3	PME	100.00 %	232	12	Boxing Recognition	7/22/2024 5:03:02 PM	7527d561-a065-4f4d-ac8f-3623f64713e8		
☐	Boxing_Recognition_rank_2	Decision Tree Ensemble	100.00 %	2588	4	Boxing Recognition	7/22/2024 5:03:02 PM	d413c60b-4c85-4116-8bf4-2b8488390286		
		Decision				Boxing	7/22/2024	e8e33f67-2e20-455e		

⚙️ Lab4 Model Evaluation and Deploy

Step 10

- Select **MPLAB XC32**.



⚙️ Lab4 Model Evaluation and Deploy

Step 11

- Configure the knowledge pack -> **DOWNLOAD**

The screenshot shows the 'Download Knowledge Pack' interface. On the left, a vertical sidebar contains five buttons: 'Library' (blue), 'ATSAMD21G18A' (yellow), 'Library' (blue), 'Custom' (pink), and 'AI Model Runner' (blue). Below these is a 'Serial' button. A red circle with the number '1' is next to the first 'Library' button. A red circle with the number '2' is next to the 'DOWNLOAD' button at the bottom. A blue arrow points from a pink box labeled 'Important' to the 'Data Source' dropdown, which is set to 'Custom'. The main form has a title 'Download Knowledge Pack' and a back arrow. It contains several dropdown menus: 'Format' (Library), 'Processor' (ATSAMD21G18A), 'Compiler' (MPLAB XC32 4.35), 'Data Source' (Custom), 'Application' (AI Model Runner), and 'Output' (Serial). Below these is a 'Debug/Profiling Settings' section. On the right, the 'Platform' section shows 'Name: MPLAB XC32', 'Manufacturer: Microchip', 'Description: Compile libraries for Microchip 32 bit processors.', and 'Resources: Firmware Documentation'. Below this is the 'Class Map' section with three buttons: '1 - Freeze', '2 - Righthook', and '3 - Uppercut'. The 'Knowledge Pack Resource Estimates' section includes 'Estimated Memory Usage' (SRAM Used: 2108 Bytes, Stack Size: 1114 Bytes, Flash Used: 6480 Bytes) and 'Estimated Latency' (Feature Extraction Latency: 2.154 ms (172300), Total Latency: 2.15375 ms (172300)). Each estimate has a green question mark icon.

Download Knowledge Pack

Platform - MPLAB XC32

Format: Library

Processor: ATSAMD21G18A

Compiler: MPLAB XC32 4.35

Data Source: Custom

Application: AI Model Runner

Output: Serial

Debug/Profiling Settings

Platform

Name: MPLAB XC32

Manufacturer: Microchip

Description: Compile libraries for Microchip 32 bit processors.

Resources: [Firmware Documentation](#)

Class Map:

1 - Freeze 2 - Righthook 3 - Uppercut

Knowledge Pack Resource Estimates

Estimated Memory Usage

SRAM Used: 2108 Bytes

Stack Size: 1114 Bytes

Flash Used: 6480 Bytes

Estimated Latency

Feature Extraction Latency: 2.154 ms (172300)

Total Latency: 2.15375 ms (172300)

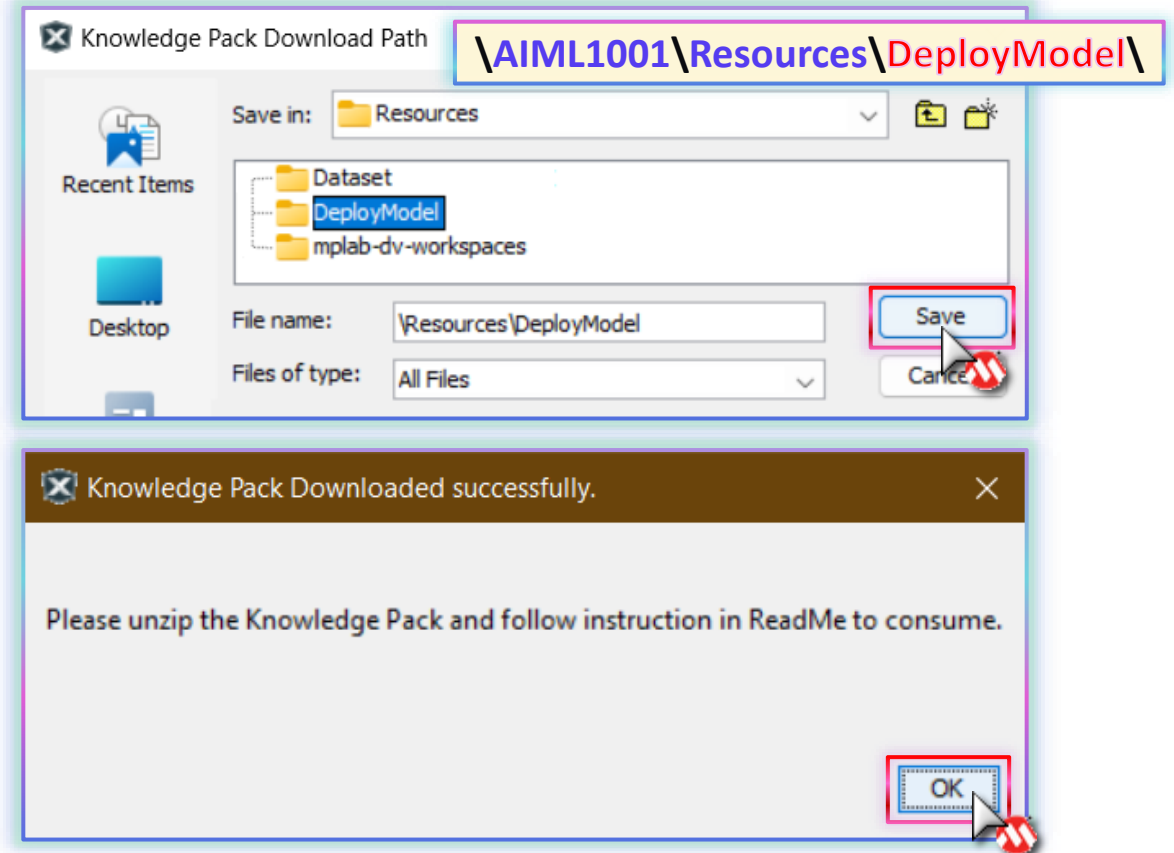
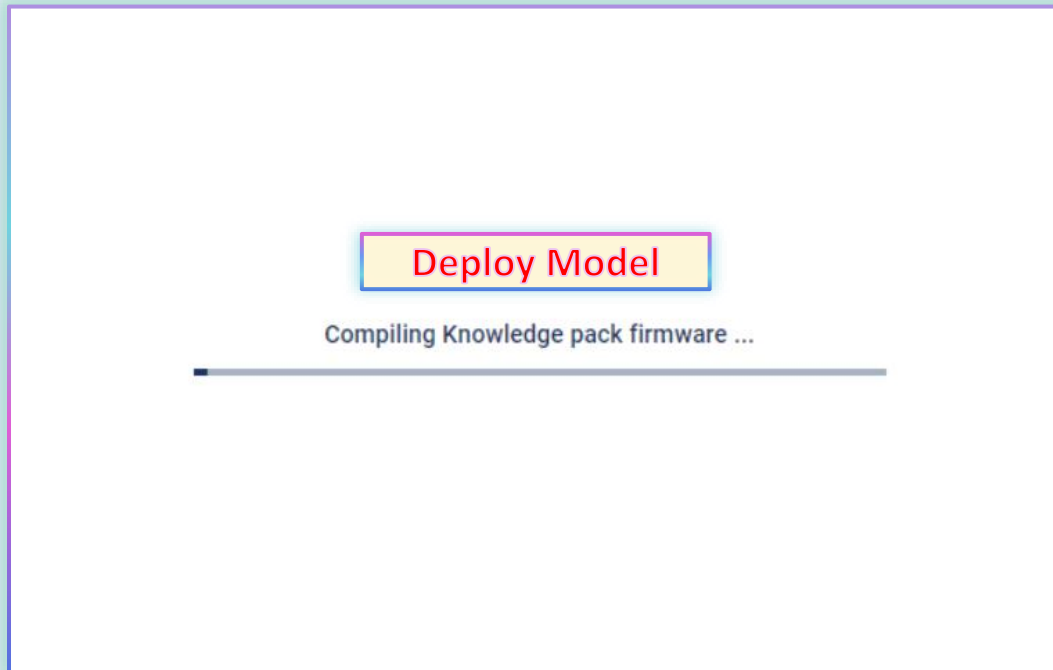
Important

DOWNLOAD

⚙️ Lab4 Model Evaluation and Deploy

Step 12

- Save the zip file under “\AIML1001\Resources\DeployModel\”.

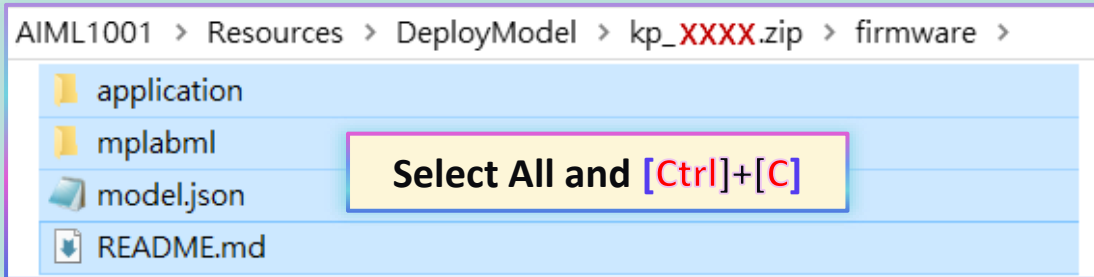


⚙️ Lab4 Model Evaluation and Deploy

Step 13

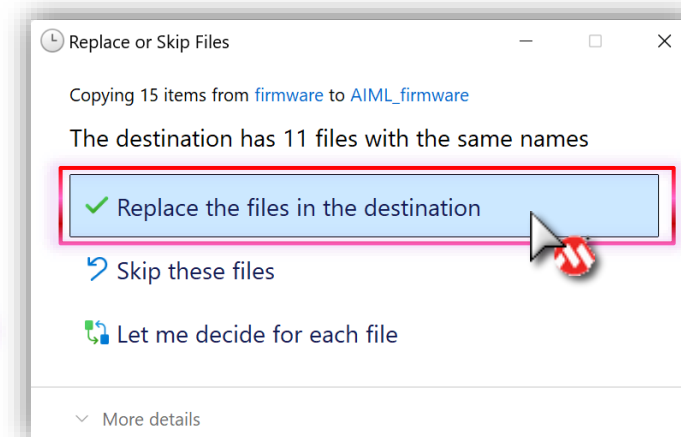
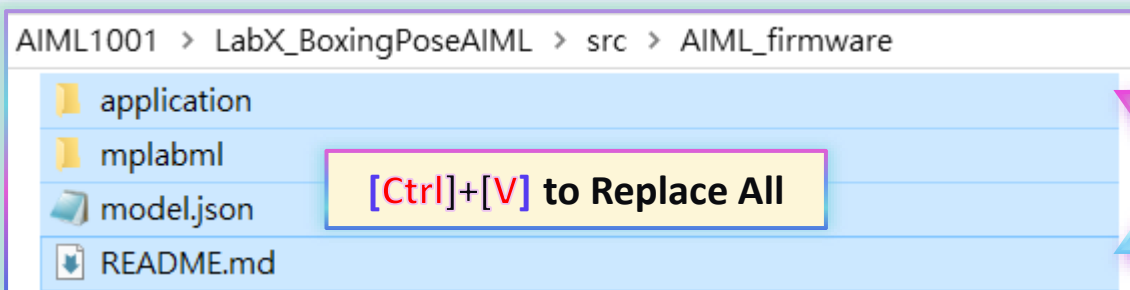
- Open downloaded Deploy Model [kp_XXXX].zip file in “\DeployModel”, Enter its folder “\firmware” and **Select all files**, press [Ctrl]+[C] to copy all files in it.

\AIML1001\Resources\DeployModel\[kp_XXXX].zip\firmware*.*



- Go to “\AIML1001\LabX_BoxingPoseAIML\src\AIML_firmware”, Paste and overwrite all files by [Ctrl]+[V]

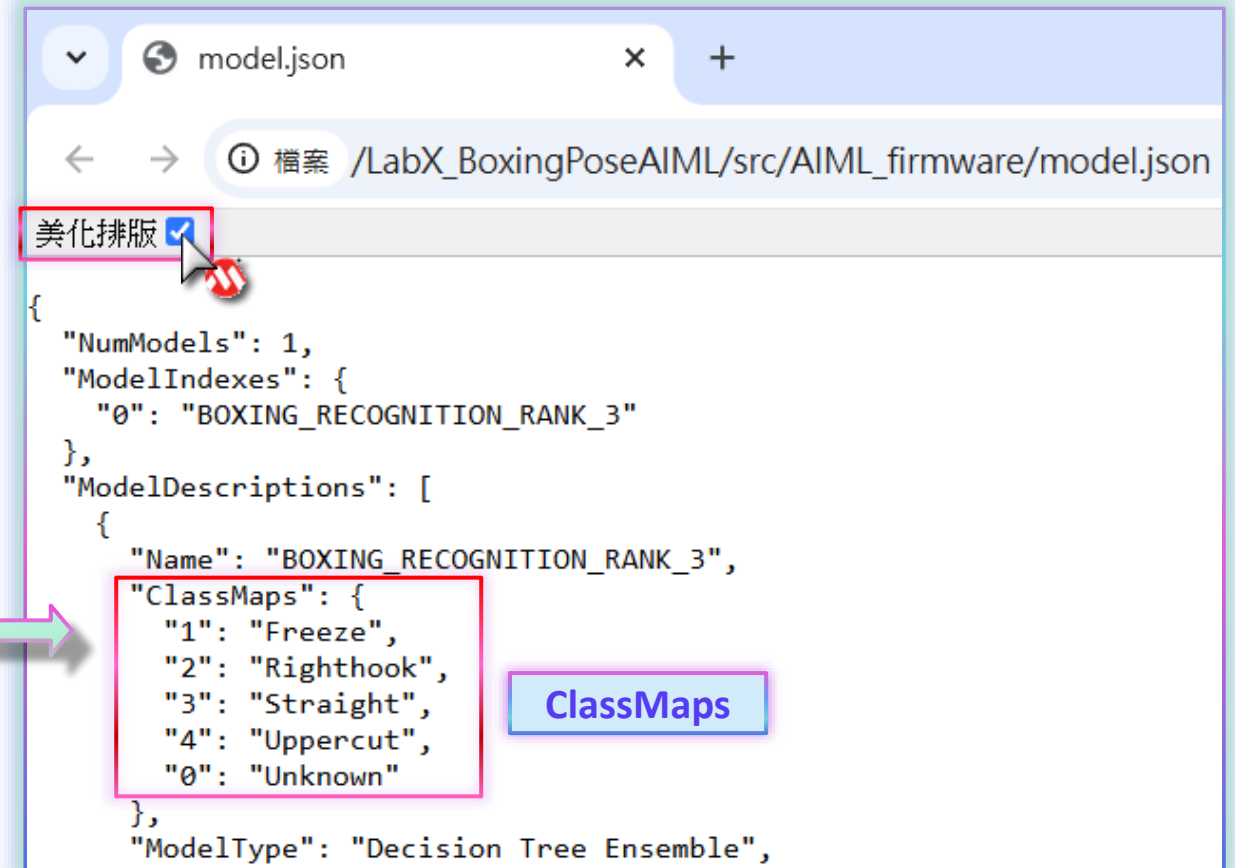
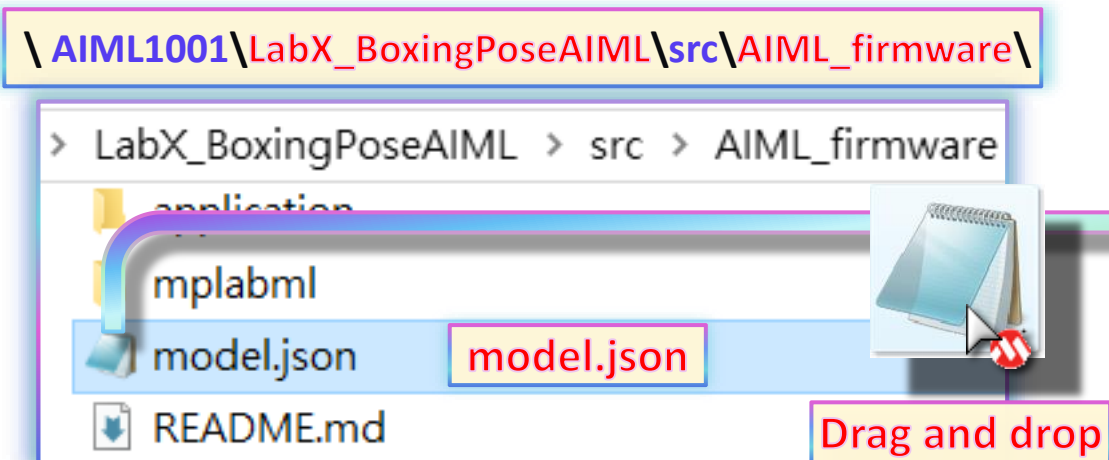
\AIML1001\LabX_BoxingPoseAIML\src\AIML_firmware\



⚙️ Lab4 Model Evaluation and Deploy

Step 14

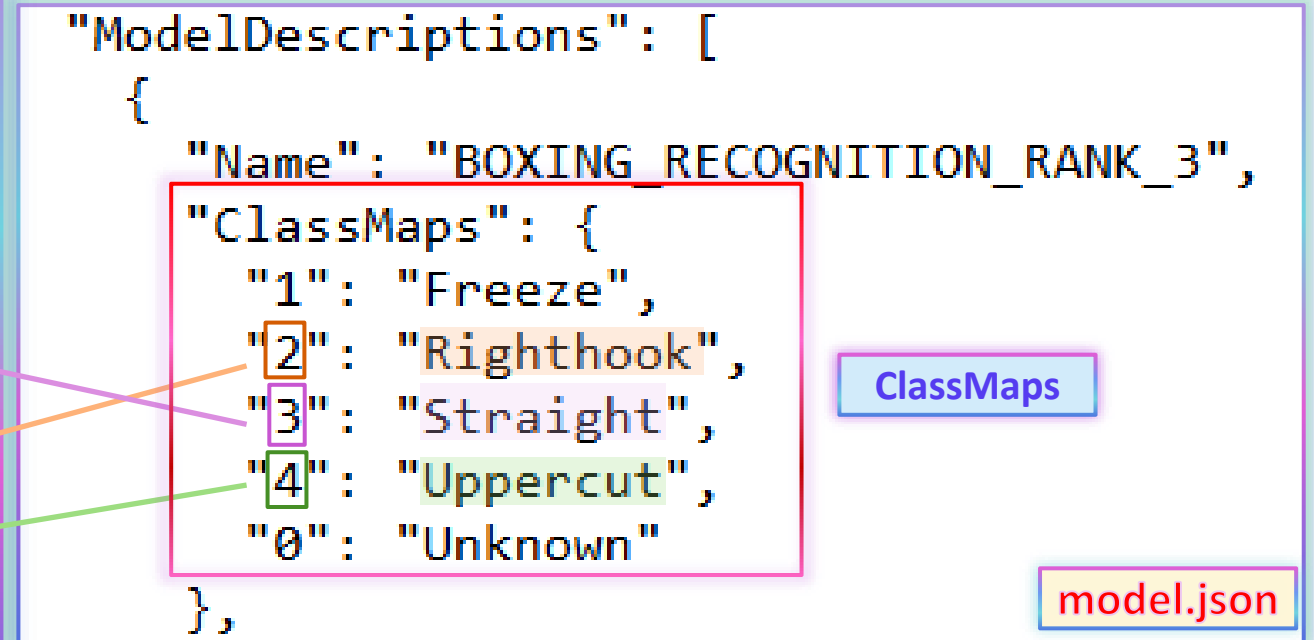
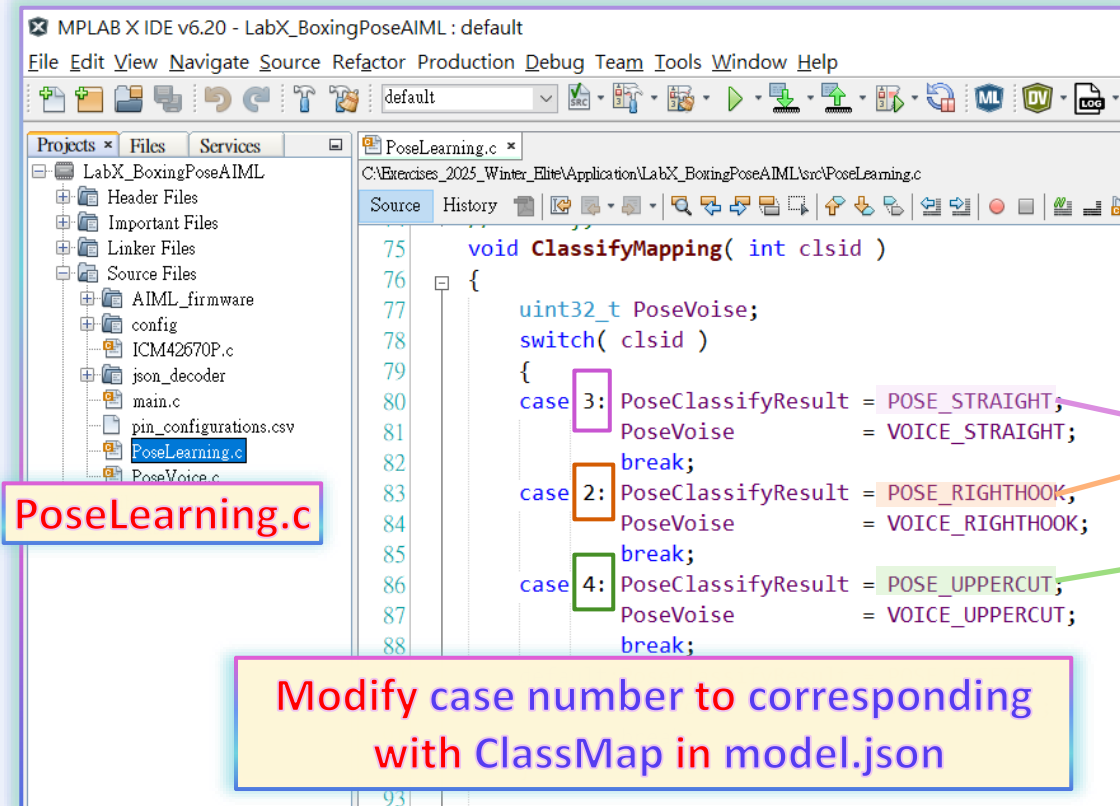
- Open the Chrome web browser
- Find a JSON file “**model.json**” in “**\LabX_BoxingPoseAIML\src\AIML_firmware**”.
- **Drag and Drop** it to Chrome.
- Check the “**Pretty-print**” option.
- Look at the “**ClassMaps**” section in the JSON file.



⚙️ Lab4 Model Evaluation and Deploy

Step 15

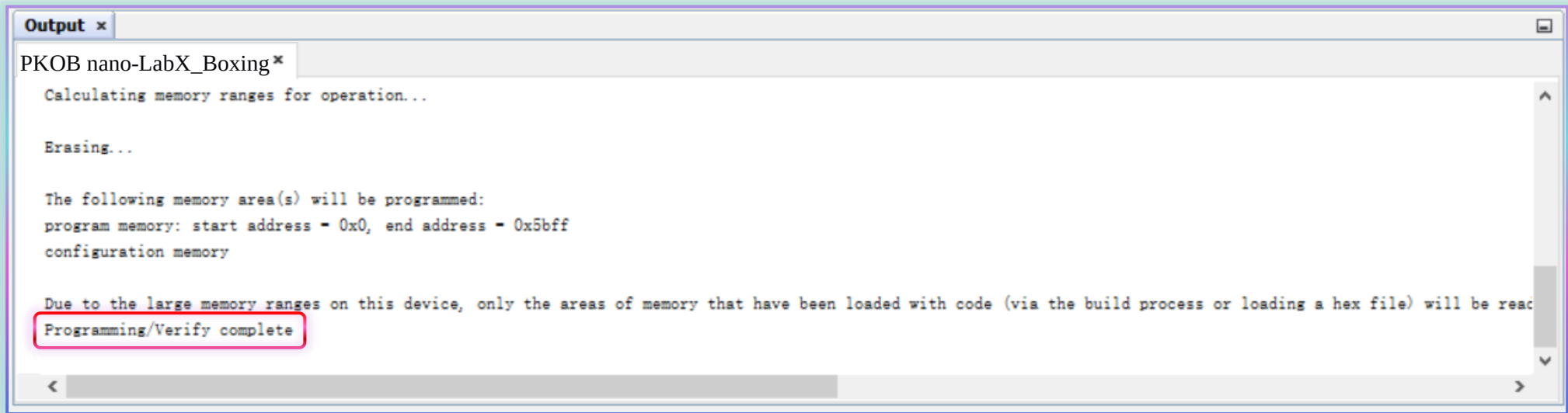
- Open “**PoseLearning.c**” of current project **LabX_BoxingPoseAIML** in MPLAB X IDE.
- Look at **Line 80, 83, 86** and make sure the **case number of pose** is correct mapping to **ClassMaps** in deploy Mode JSON file. If not, replace with the corresponding number.



⚙️ Lab4 Model Evaluation and Deploy

Step 16

- Click on icon  and select **Clean and Build Project** to compile your project.
- Click on icon  to **Mark and Program** to EVB and check if any fault while Programming.
- Output Window will show **Programming/Verify Complete**.



The screenshot shows the 'Output' window of an IDE. The title bar says 'Output x'. The content area shows the following text:

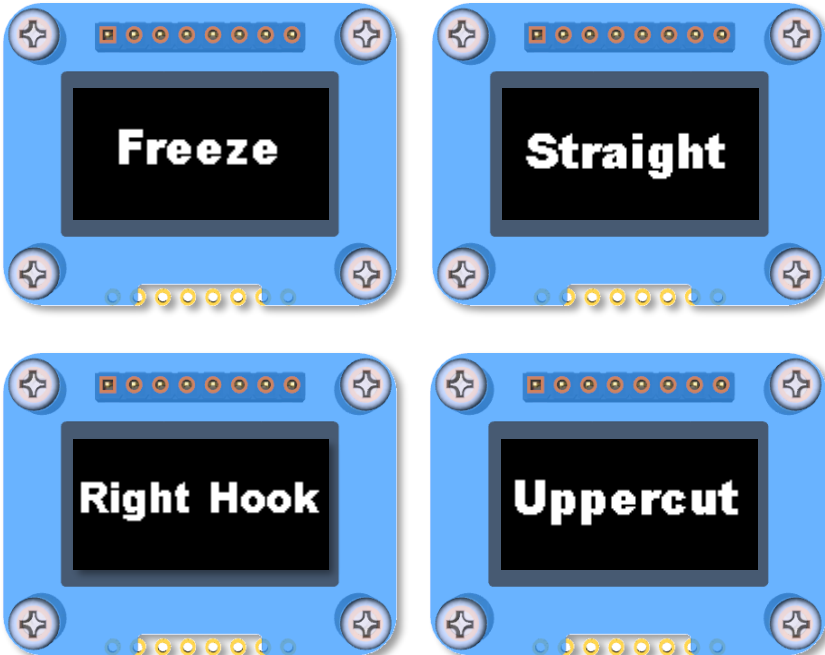
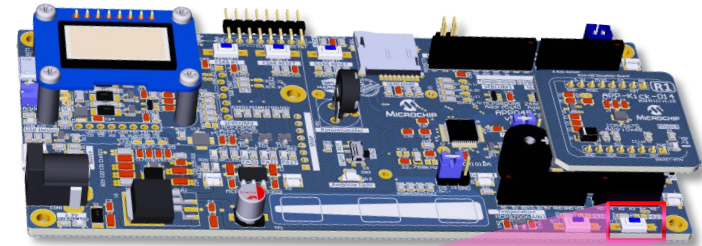
```
PKOB nano-LabX_Boxing*  
Calculating memory ranges for operation...  
  
Erasing...  
  
The following memory area(s) will be programmed:  
program memory: start address = 0x0, end address = 0x5bff  
configuration memory  
  
Due to the large memory ranges on this device, only the areas of memory that have been loaded with code (via the build process or loading a hex file) will be read  
Programming/Verify complete
```

The text 'Programming/Verify complete' is highlighted with a red rectangular box. The window has a scrollbar on the right and a horizontal scrollbar at the bottom.

⚙️ Lab4 Model Evaluation and Deploy

Result

- Switch the operating mode to **Mode 2** by press one time of **BT2**.
 - Mode 1: Display 6DOF Sensor Data in DV Terminal and OLED.
 - **Mode 2**: Display **Boxing Stance classification results** of preset ML model in DV Terminal and OLED.
 - Mode 3: Display 6DOF Sensor Data waveform in Time Plot of Data Visualizer (For Data Collection).
- Start doing some boxing!!!

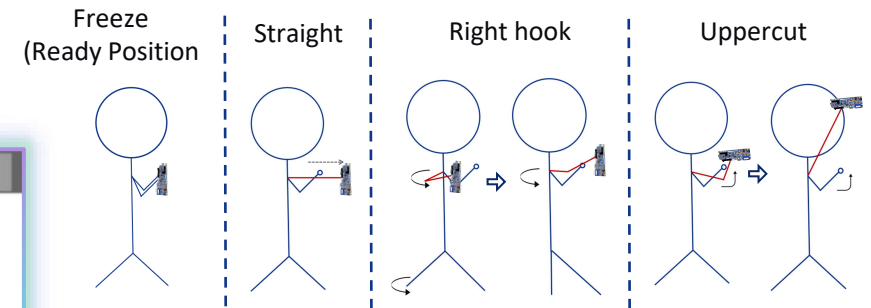


Mode 2

```
ML Data Collector X Touch X Terminal X XY Plot X
TOK ICM42670 Initialize
- Software Reset Completed
- WHO AM I : 0x67
- Gyro Initial Completed
- Accel Initial Completed
- Enable Completed
- Hardware INT Enabled

TOK ICM42670 Output
- Temperature : 25.0'C
- Accel ( -0.2, +0.2, +1.0)
- Gryo ( +29.5, +2.4, -12.6)

Auto ML Model Name : BOXING_RECOGNITION_RANK_3
Auto ML Classify Maps :
- Id 0 : Unknown
- Id 1 : Freeze
- Id 2 : Righthook
- Id 3 : Straight
- Id 4 : Uppercut
Auto ML Inference Result : Boxing Freeze
```



Terminal

Auto ML Inference Result : Boxing Freeze

Auto ML Inference Result : Boxing Straight

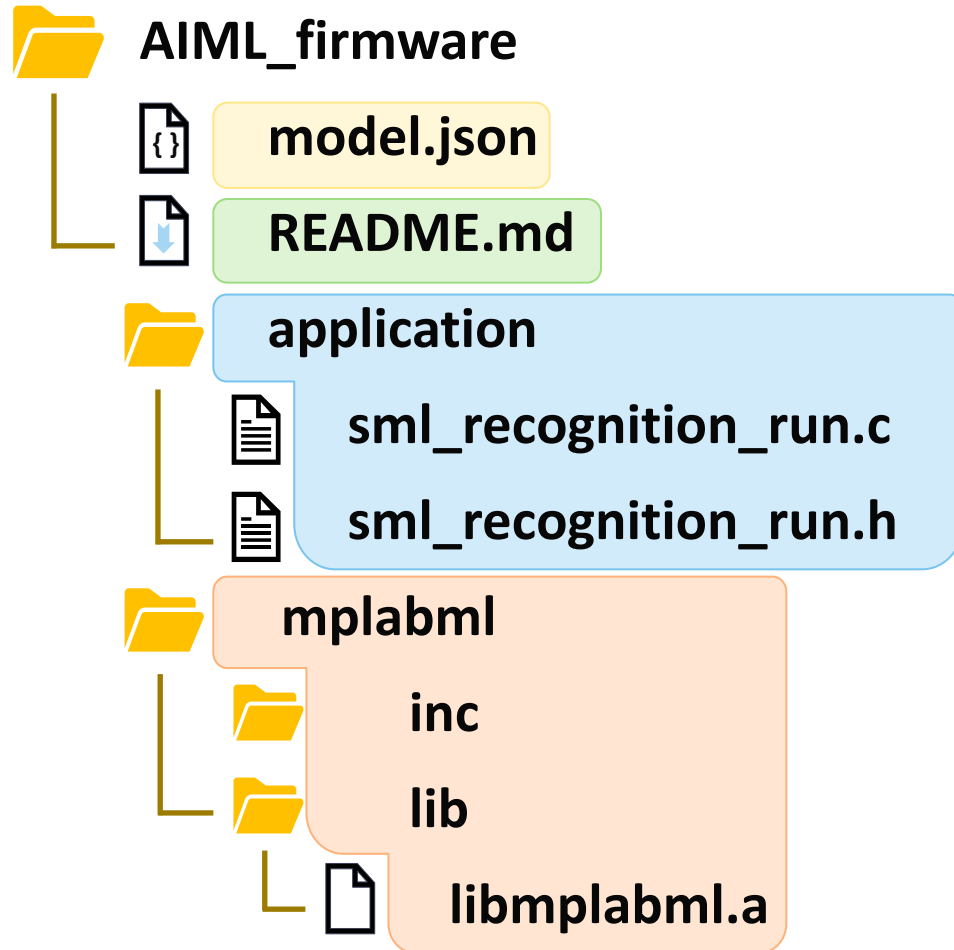
Auto ML Inference Result : Boxing Right hook

Auto ML Inference Result : Boxing Uppercut

Knowledge Pack, Main Flow, and State Machines

Composition of ML Deploy Model Knowledge Pack

Folders and Files



model.json: Contains the structure and parameters of the machine learning model.

README.md: Description file, including the integration of knowledge pack.

application: Contains the main code for running machine learning models. Responsible for loading models, processing input data, performing inference.

mplabml: The inc folder contains header files (.h), which define various functions and structures used in the project. libmplabml.a is the library of the related to ML.

Composition of ML Deploy Model Knowledge Pack

model.json

- Contains the structure and parameters of the machine learning model. If you want to see the graphical interface, you can view it in the [ML Model Builder](#) of [Explore model](#).

```
{
  "NumModels": 1,
  "ModelIndexes": {
    "0": "BOXING_RECOGNITION_RANK_3"
  },
  "ModelDescriptions": [
    {
      "Name": "BOXING_RECOGNITION_RANK_3",
      "ClassMaps": {
        "1": "Freeze",
        "2": "Righthook",
        "3": "Straight",
        "4": "Uppercut",
        "0": "Unknown"
      },
      "ModelType": "Decision Tree Ensemble",
      "FeatureNames": [
        "gX25Percentile",
        "aXAbsAreaDc",
        "aYAbsAreaDc",
        "aXP2P",
        "gZMaxP2PGlobalDC",
        "gXgZ_cross_mean_diff",
        "gYaX_cross_mean_diff",
        "aYaZ_cross_p2p_diff"
      ]
    }
  ],
}
```

Model name.

ClassMaps: {"1": "Freeze"}

"1" stand for the output

"Freeze" stand for the meaning of output

Feature Summary: what features are actually used to classify the results. All the parameter of the feature is also included.

Composition of ML Deploy Model Knowledge Pack

sml_recognition_run.c

- **Includes functions for:**
 - **Running inference:** Processing incoming sensor data and passing it through the trained model.
 - **Handling model output:** Interpreting classification or anomaly detection results.
 - **Integrating with firmware:** Ensuring real-time execution and efficient resource use on microcontrollers.

```
void sml_output_results(uint16_t model, uint16_t classification)
{
    memset(serial_out_buf, 0, SERIAL_OUT_CHARS_MAX);
    kb_sprint_model_result(model, serial_out_buf, false, false, true);
    printf("%s\n", serial_out_buf);
}

int32_t sml_recognition_run(int16_t *data, int32_t num_sensors)
{
    int32_t ret;

    ret = kb_run_model((int16_t *)data, num_sensors, KB_MODEL_BOXING_RECOGNITION_RANK_3_INDEX);
    if (ret >= 0){
        sml_output_results(KB_MODEL_BOXING_RECOGNITION_RANK_3_INDEX, ret);
        kb_reset_model(0);
    };

    return ret;
}
```

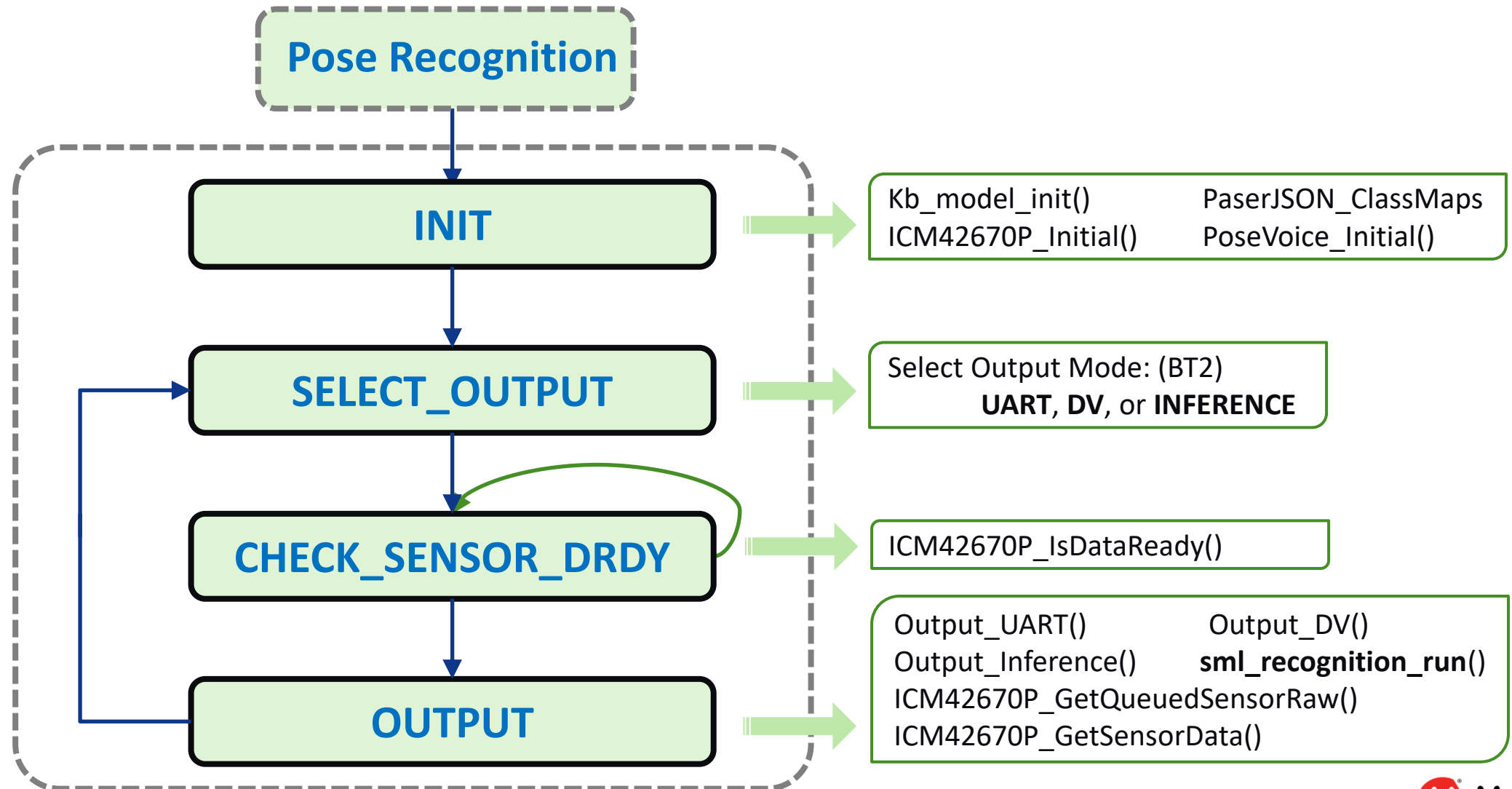
Composition of ML Deploy Model Knowledge Pack

libmplabml.a

- A static library file to provides precompiled machine learning functions optimized for embedded systems for microcontrollers.
- Header Files for Interface Functions :
 - **kb.h**
 - The main header file that includes function prototypes and references to other knowledge base (KB) files.
 - Provides an interface to interact with the ML model and run inferences.
 - **kb_define.h**
 - Contains macro definitions and constants used across the KB system.
 - Defines buffer sizes, thresholds, and model-specific parameters.
 - **kb_output.h**
 - Manages the output format of the machine learning inference.
 - Defines structures or functions to handle classification results, anomaly detection, etc.
 - **kb_typedef.h**
 - Defines custom data types (typedefs, structs, and enums) used in the KB system.
 - Ensures consistency across different hardware and software implementations.
 - **model_json.h**
 - Contains the ML model in a serialized JSON format (converted into C header form).
 - Stores model architecture, weights, and metadata needed for inference.
 - **testdata.h**
 - Likely includes predefined test data for validating the model during development.
 - Used for debugging and verifying inference correctness without live sensor input.

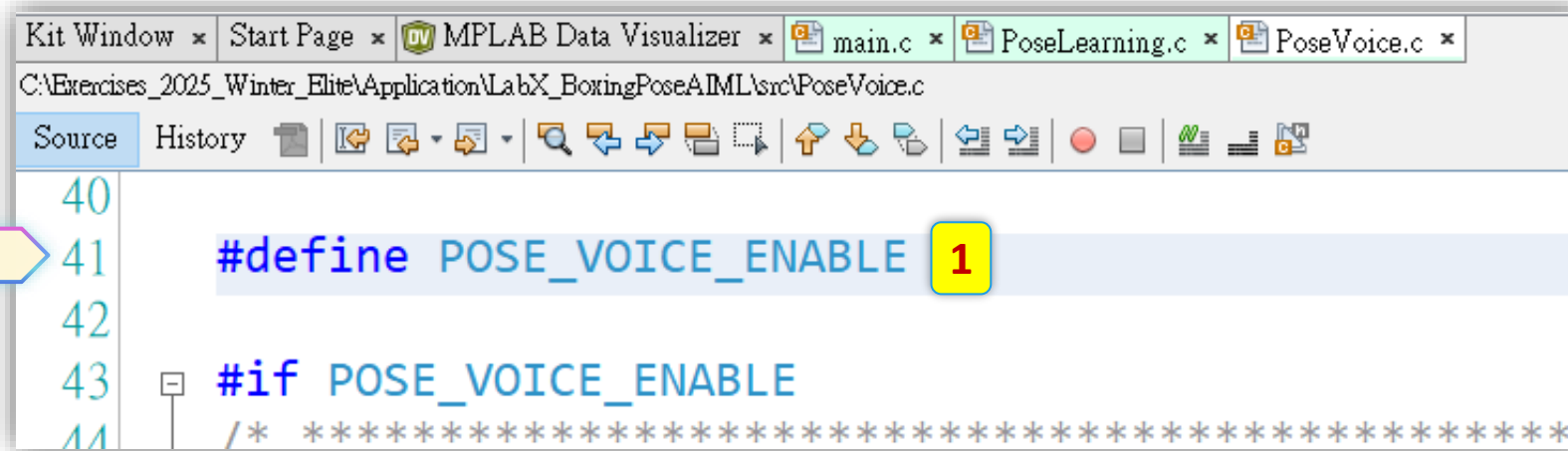
Main Flow and State Machines

- The State Machine implemented in “[PoseLearning.c](#)” as below.

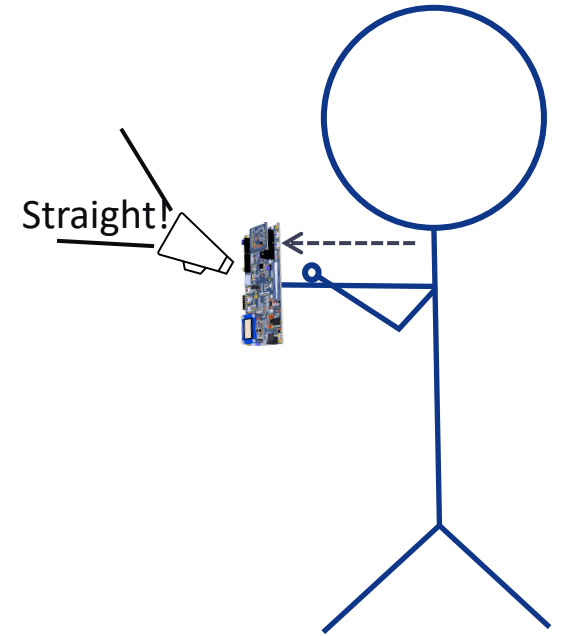


PCM Voice for Boxing Pose

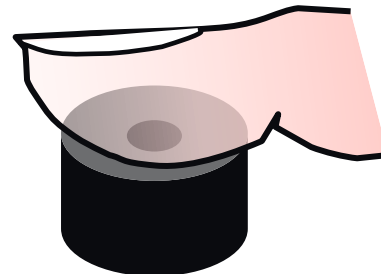
- We have implemented a PCM voice as a hidden bonus in the current application.
- Open “PoseVoice.c” file.
- Modify the **#define POSE_VOICE_ENABLE** to **1** in **Line 41**.
- Program again and enjoy it !



```
Kit Window * Start Page * MPLAB Data Visualizer * main.c * PoseLearning.c * PoseVoice.c *
C:\Exercises_2025_Winter_Elite\Application\LabX_BoxingPoseAIML\src\PoseVoice.c
Source History
40
41 #define POSE_VOICE_ENABLE 1
42
43 #if POSE_VOICE_ENABLE
44 /* *****
```



- **Thumb on the Buzzer** will make the PCM Voice clearer and louder ! (Or use a tap)



Summary

MPLAB® ML Suite

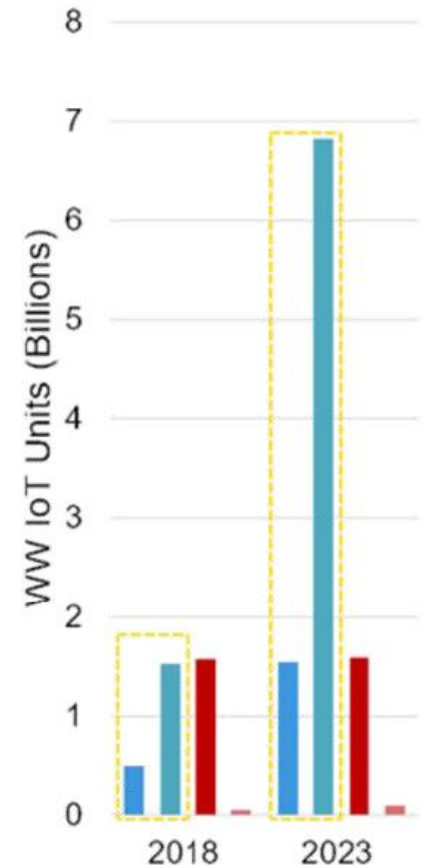
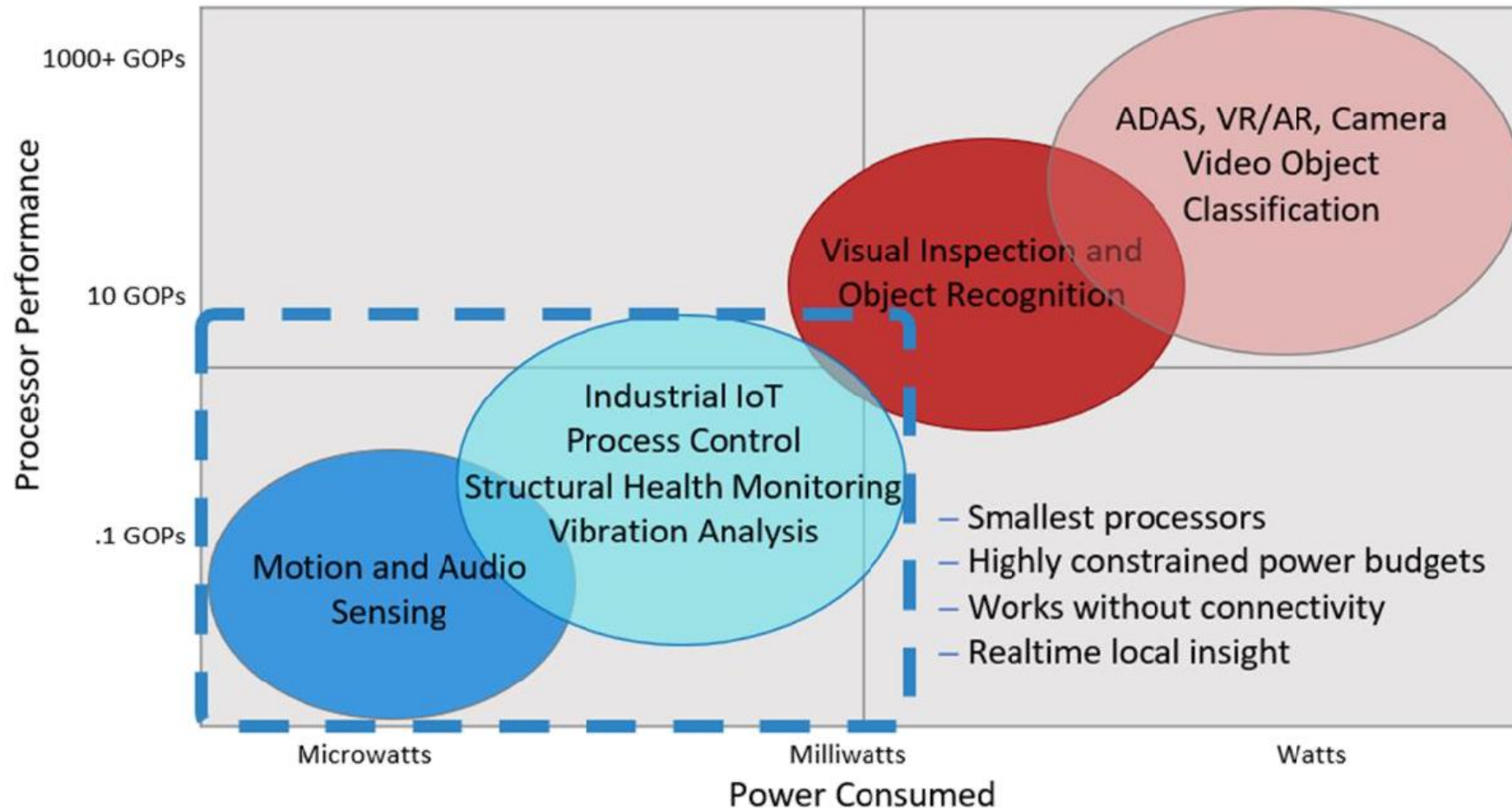
- The ML Suite Plugin in MPLAB X is a plugin
- Behind this is a rich API based on SensiML
 - Easily accessed from Python
 - Customize functions
 - Feature extraction
- **Can tuned by Jupyter Notebooks**
 - Web-based live notebooks
 - Code
 - Notes
 - Visualization



[Related document](#)



MPLAB® ML Suite Addresses Billions of Time-Series IoT Endpoints



Source: IHS 2017

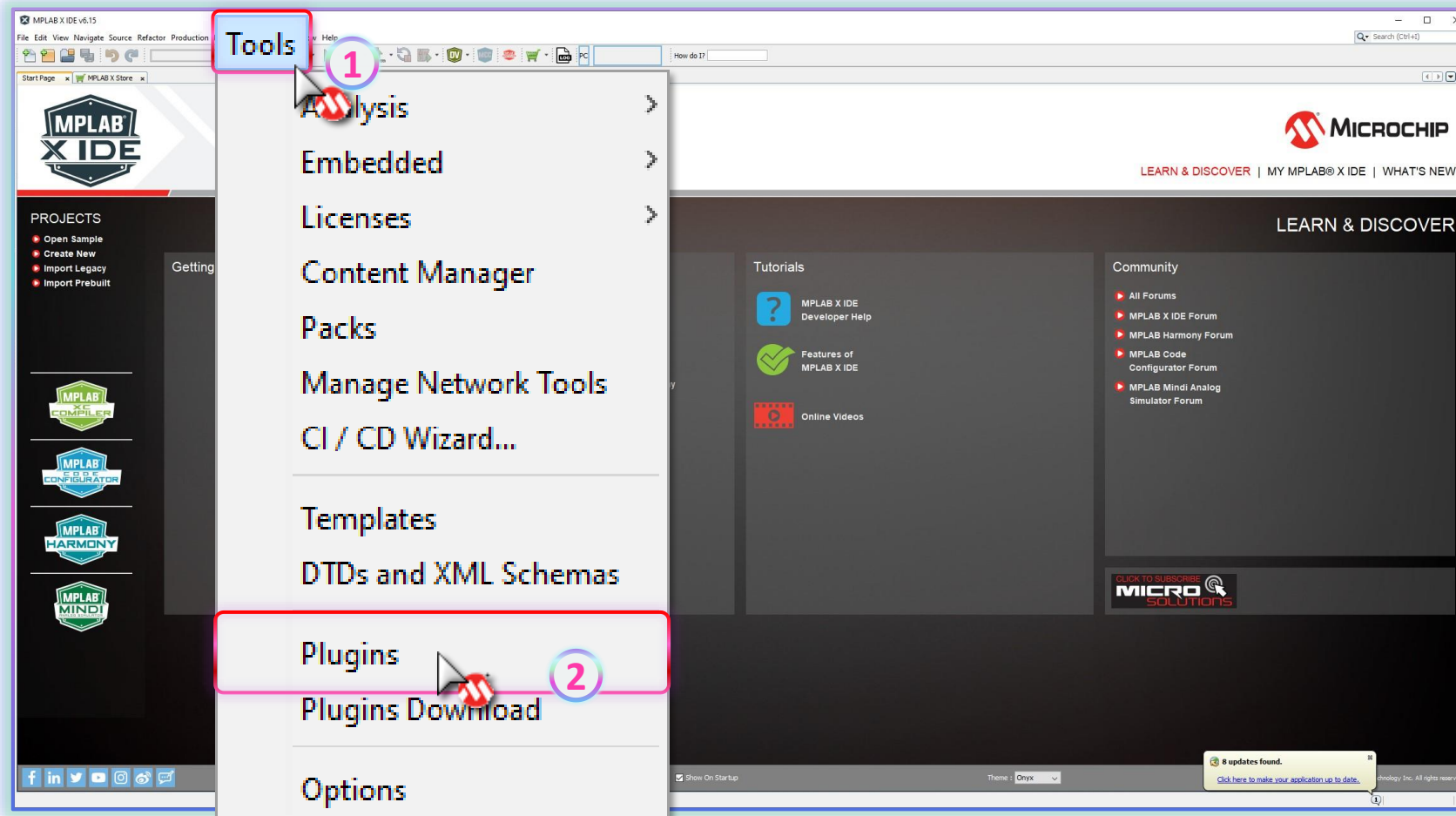
Appendix

Install the ML Data Collector & ML Model Builder

Install the ML Data Collector & ML Model Builder

Step 1

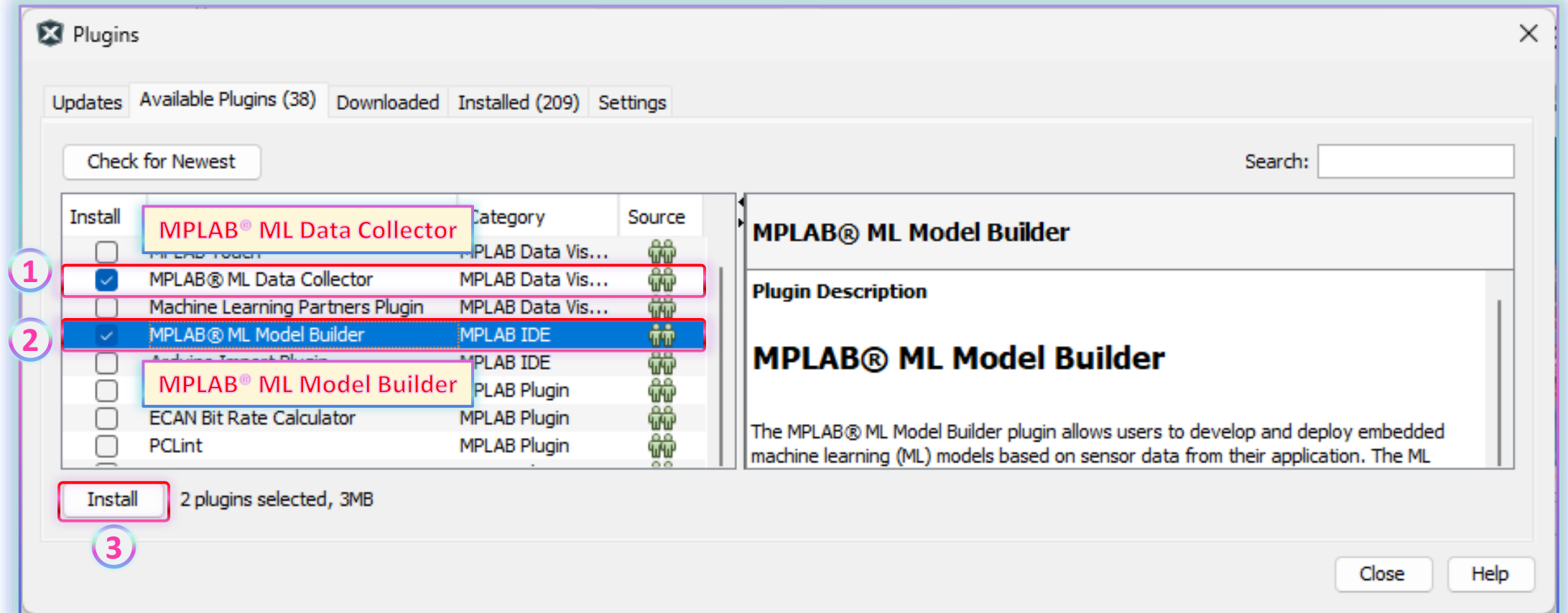
- Open **Plugins** installer



Install the ML Data Collector & ML Model Builder

Step 2

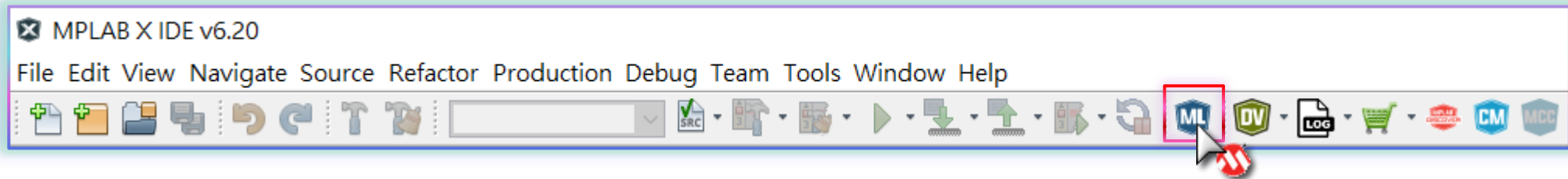
- Install the “MPLAB ML Data Collector” and “MPLAB ML Model Builder” Plugins.



Install the ML Data Collector & ML Model Builder

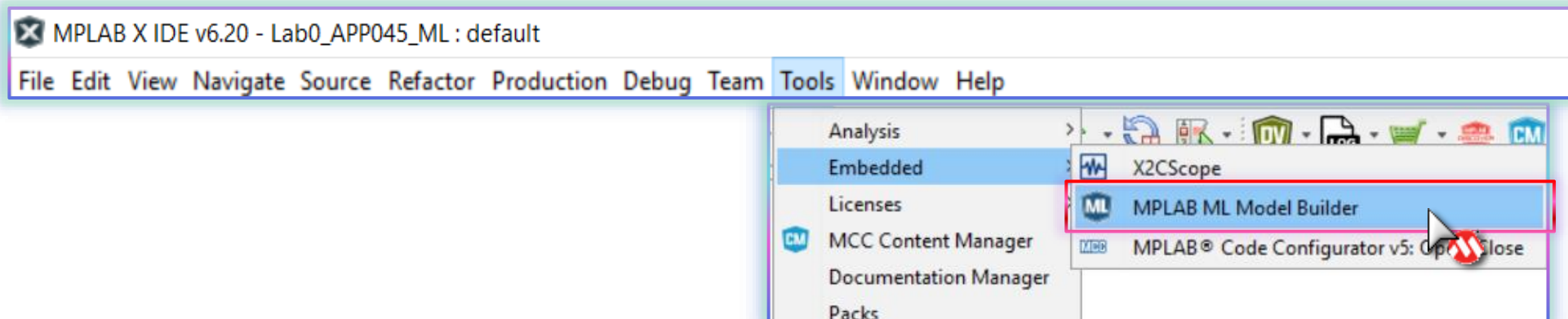
Step 3

- After install completed, please check your **MPLAB X IDE** to find the **ML Model builder**.
- Click  icon



OR

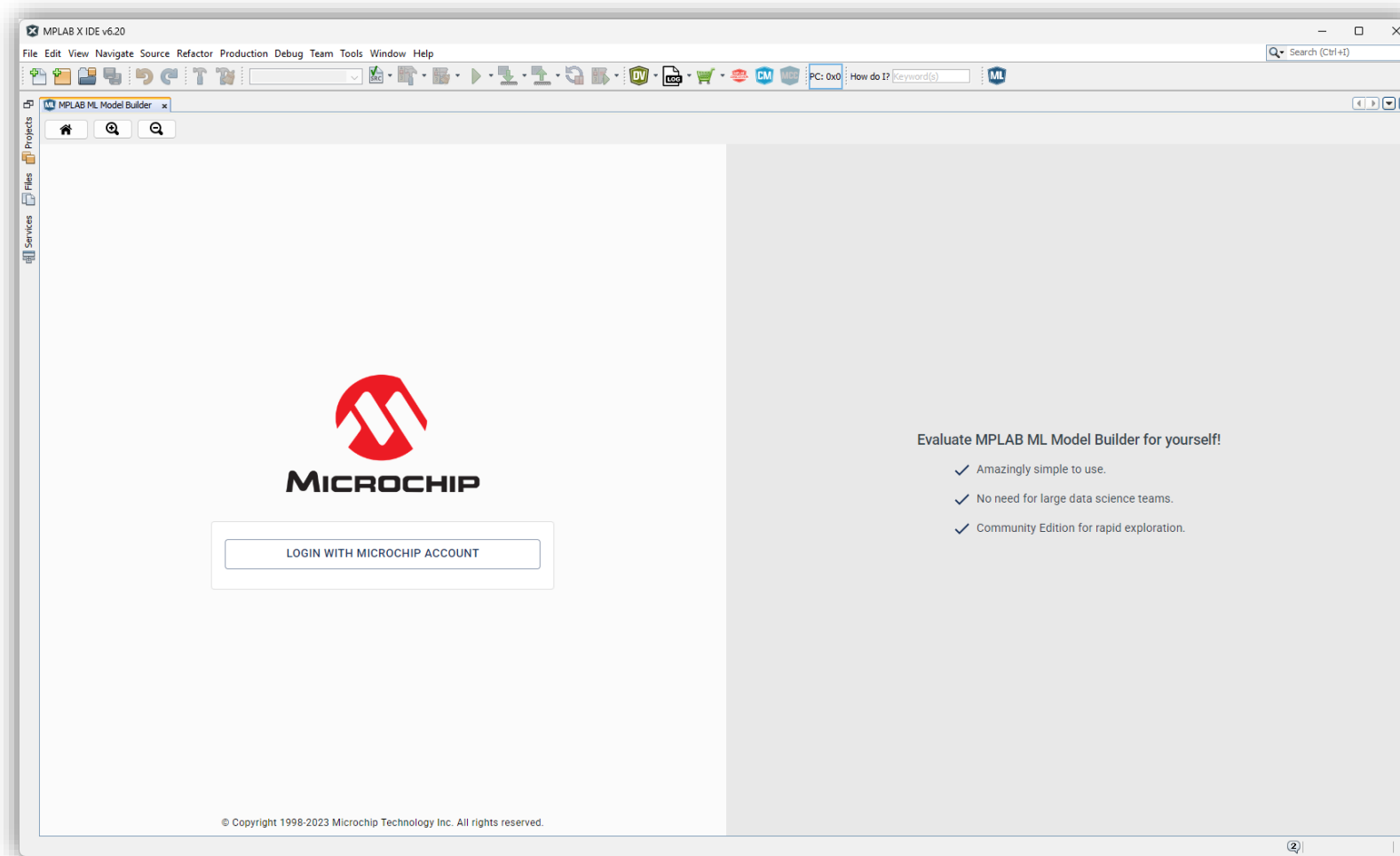
- **Tools > Embedded > MPLAB ML Model Builder**



Create and Login MPLAB ML Account

Create MPLAB ML Account

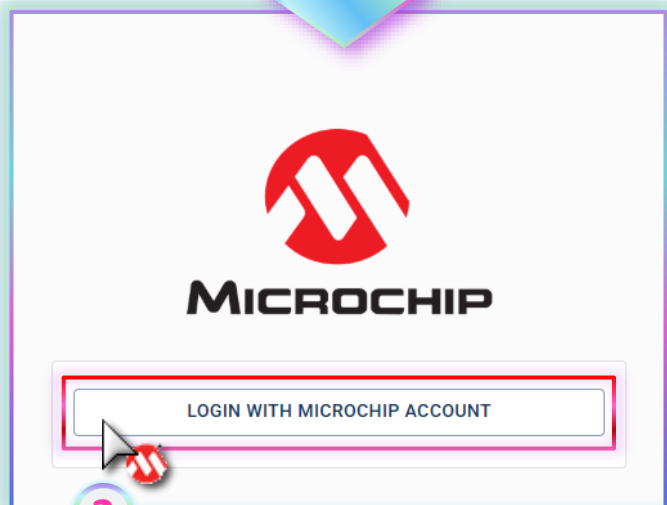
- This course must have a **MPLAB ML Account** to continue, **please create it first !**



Create MPLAB ML Account

- Create a **MPLAB ML Account**.

1 Click on the icon to open MPLAB ML Model Builder



2 Click "Login with Microchip Account"

Sign In

Email Address*
Enter Email Address

Password*
Enter Password

[Forgot password?](#)

☐ Remember Me

Sign in

New to Microchip? [Register for an account](#)

3 Click "Register for an account"

[Website Terms & Conditions](#)

4 Fill in your information
(Please do not fill in wrong information arbitrarily !)

Create Your Account

*Required Fields. Optional fields will help us to complete your profile and improve your experience on our site.

First Name *
Enter First Name

Last Name *
Enter Last Name

Email Address *
Enter Email Address

Password *
Enter Password

Re-type Password *
Re-type Password

☐ Show Password

Company Name *
Enter Company Name

Job Function
-- Select Job Function --

Industry
-- Select Industry --

Country *
-- Select Country --

State or Province *
Enter State or Province

Address Line 1 *
Enter Address Line 1

Address Line 2
Enter Address Line 2

City *
Enter City Name

Postal/Zip Code *
Enter Zip Code

☐ Future offers or information about Microchip's products or services

5 Click "Save my profile and Continue"

Save my profile and Continue

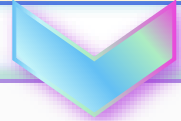
Login MPLAB ML Account

- Login your **MPLAB ML Account**.



1

Click on the icon to open
MPLAB ML Model Builder



2

Click "Login with Microchip Account"



3

Fill in your information and Click "Sign in"

Sign In

Email Address*

Enter Email Address

Password*

Enter Password

Forgot password?

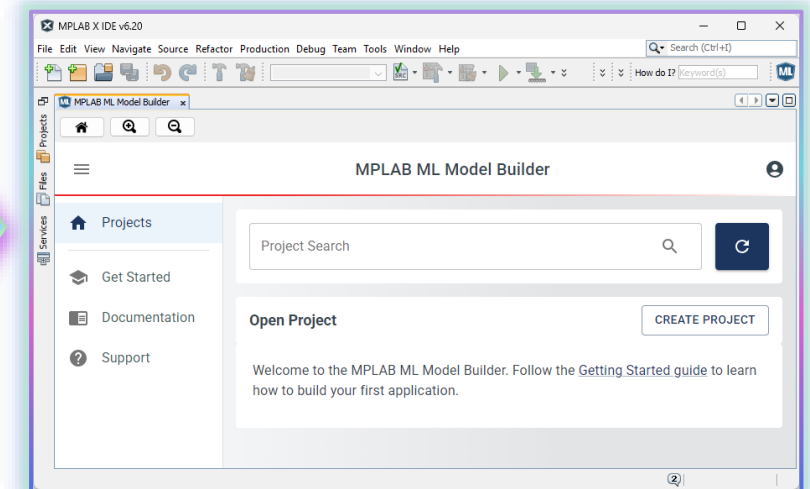
☐ Remember Me

Sign in

New to Microchip? [Register for an account](#)

[View Microchip's Privacy Policies](#)

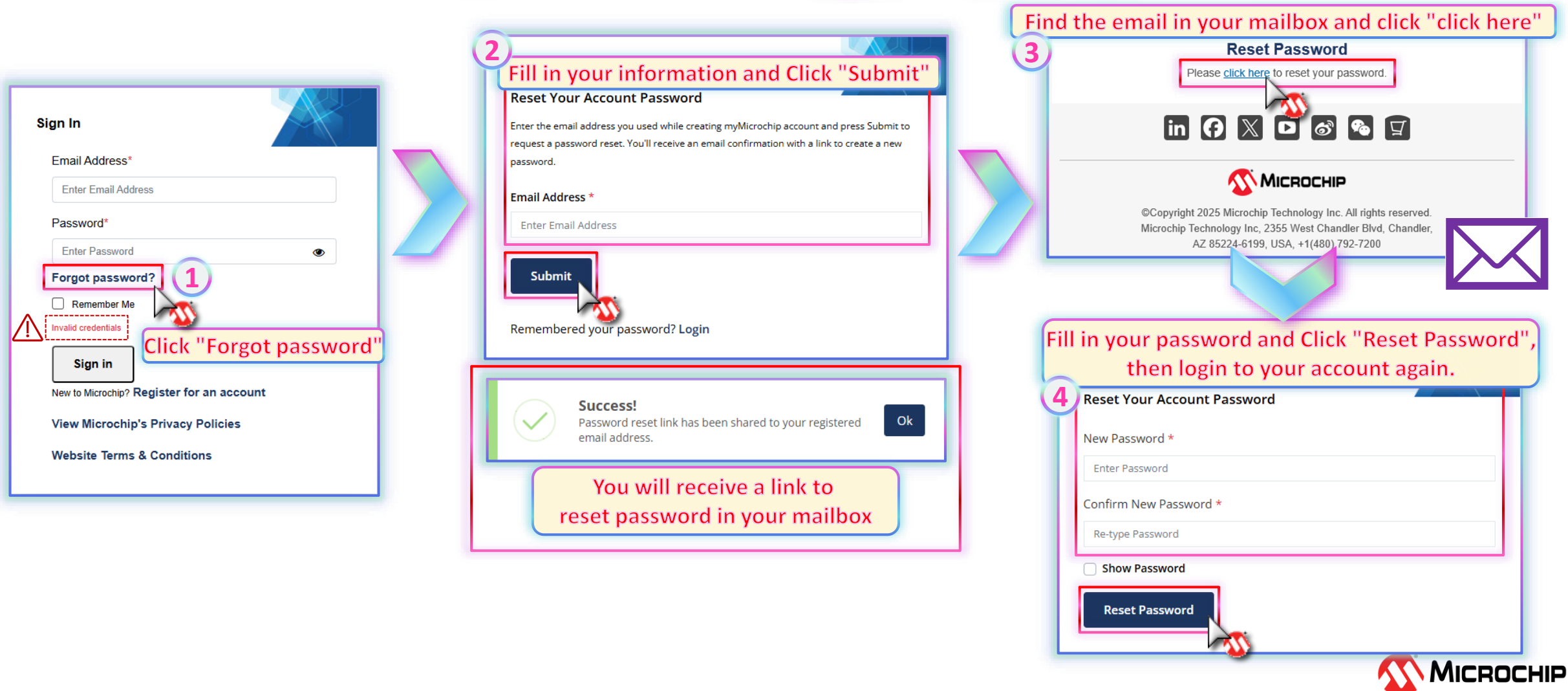
[Website Terms & Conditions](#)



Login MPLAB ML Account

⚠ Invalid credentials

- Please try to **Reset Your Password** when the "Invalid credentials" message appears.

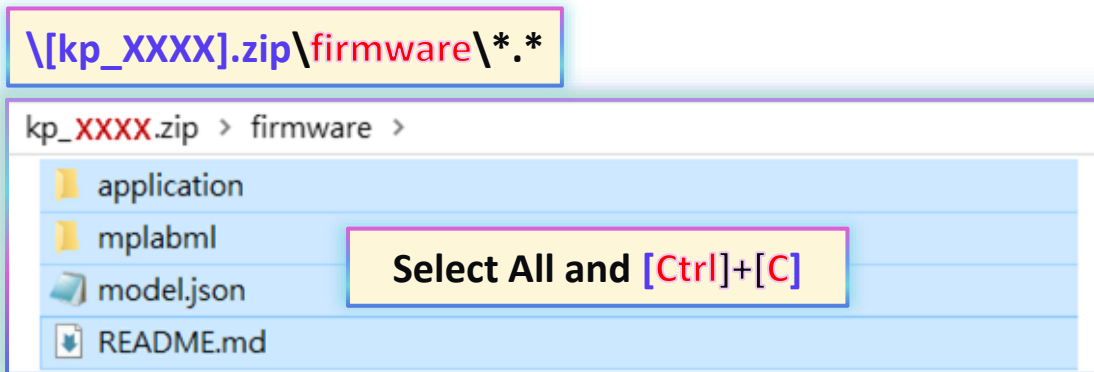


Porting Deploy Model to Custom Project

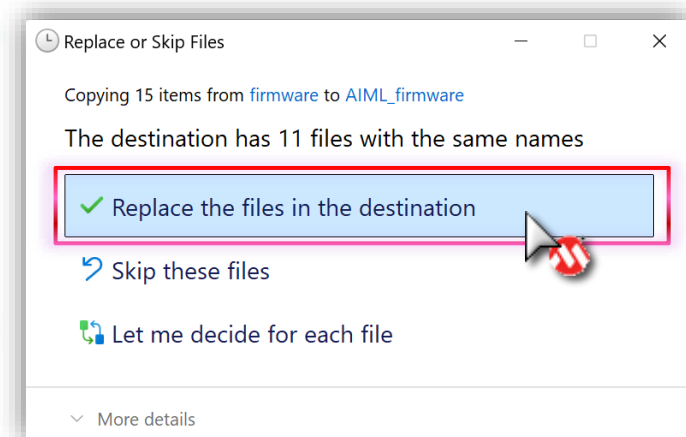
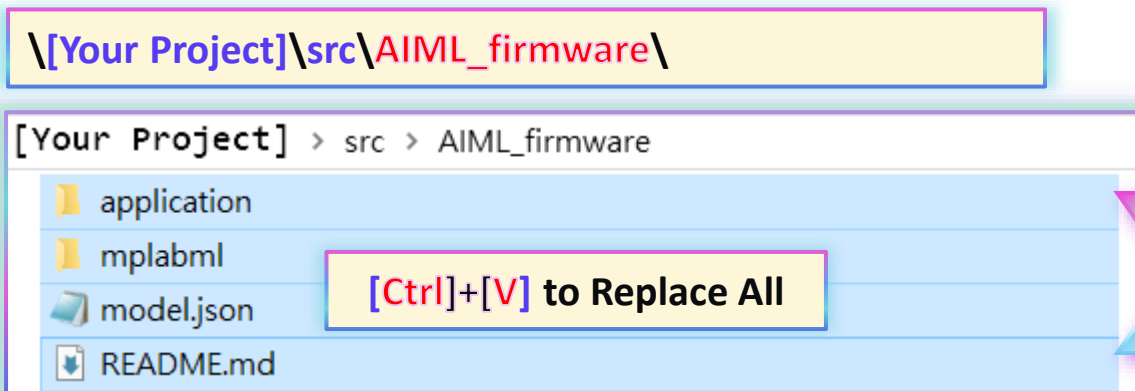
Porting Deploy Model to Custom Project

Step 1

- Open downloaded ML Deploy Model `[kp_XXXX].zip`, Enter its folder “`\firmware`” and **Select all files**, press `[Ctrl]+[C]` to copy all files in it.



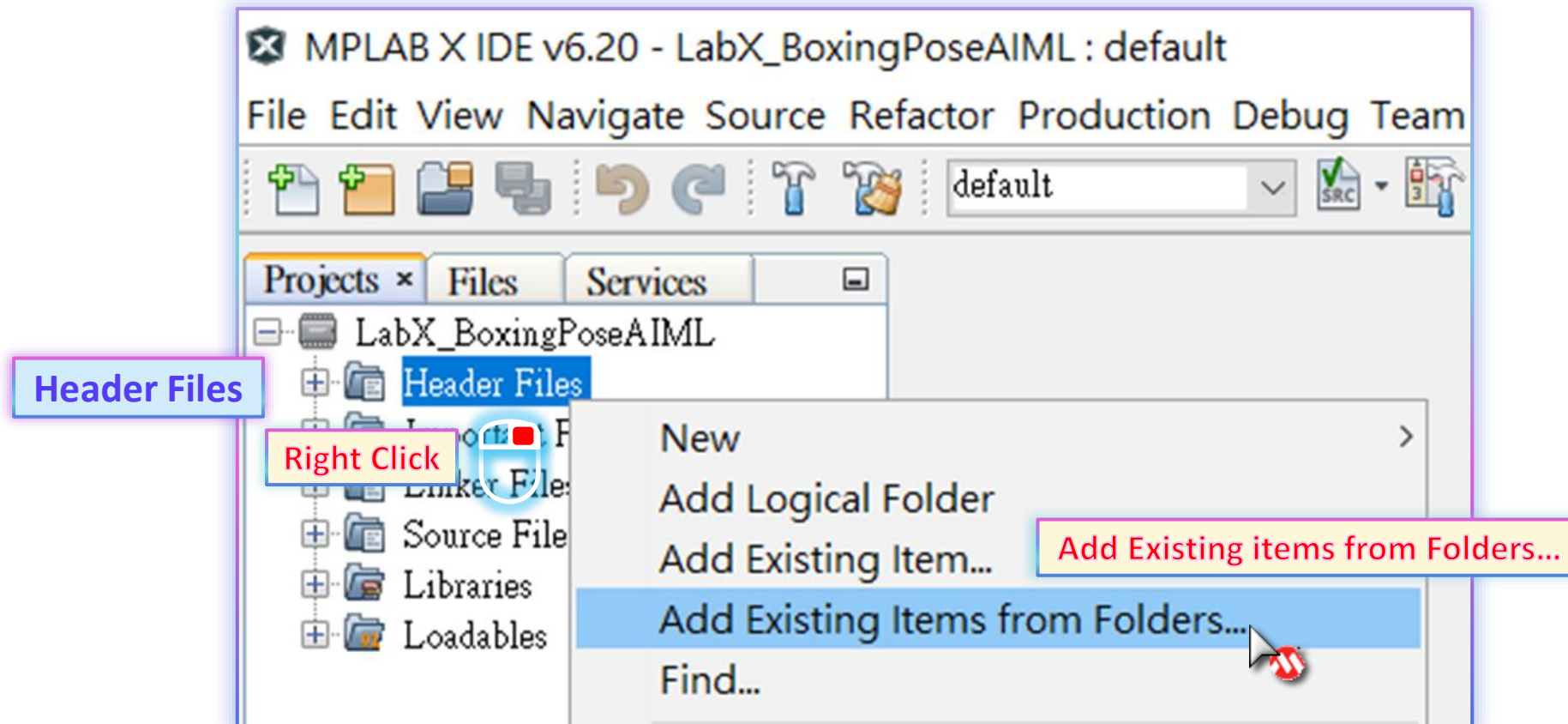
- Create a folder “`AIML_firmware`” under “`\[Your Project]\src\`”, Paste all files copied from above step by `[Ctrl]+[V]`



Porting Deploy Model to Custom Project

Step 2

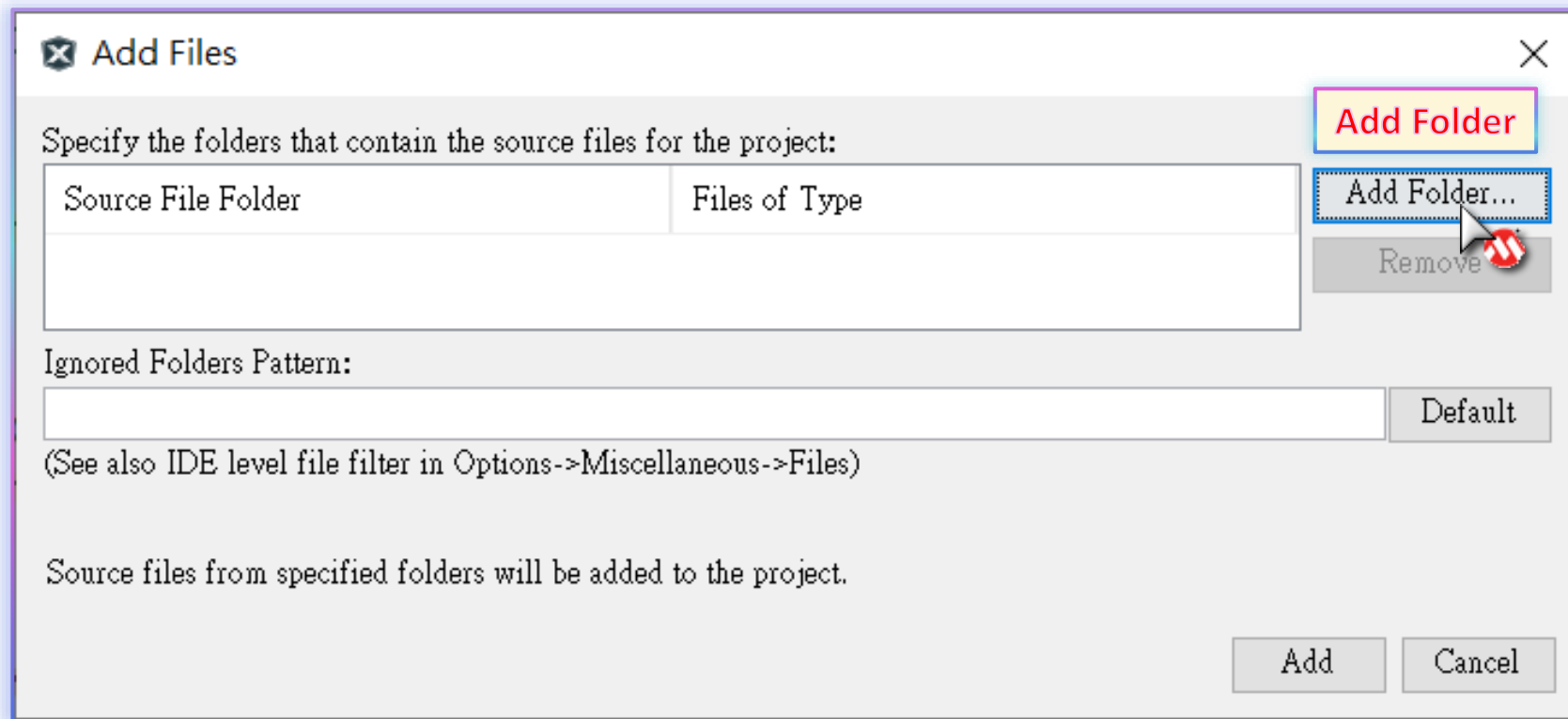
- Open your project in MPLAB X IDE.
- Right click on **Header Files** -> **Add Existing Items from Folders**.



Porting Deploy Model to Custom Project

Step 3

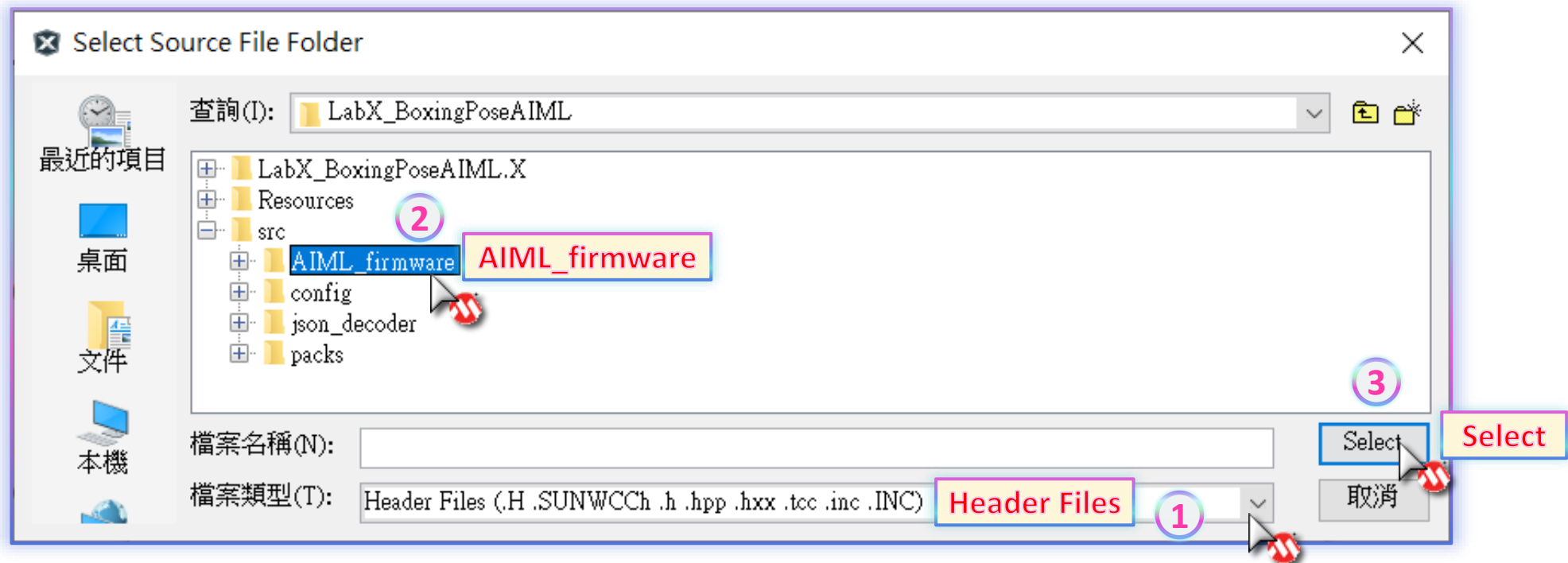
- Click **Add Folder**.



Porting Deploy Model to Custom Project

Step 4

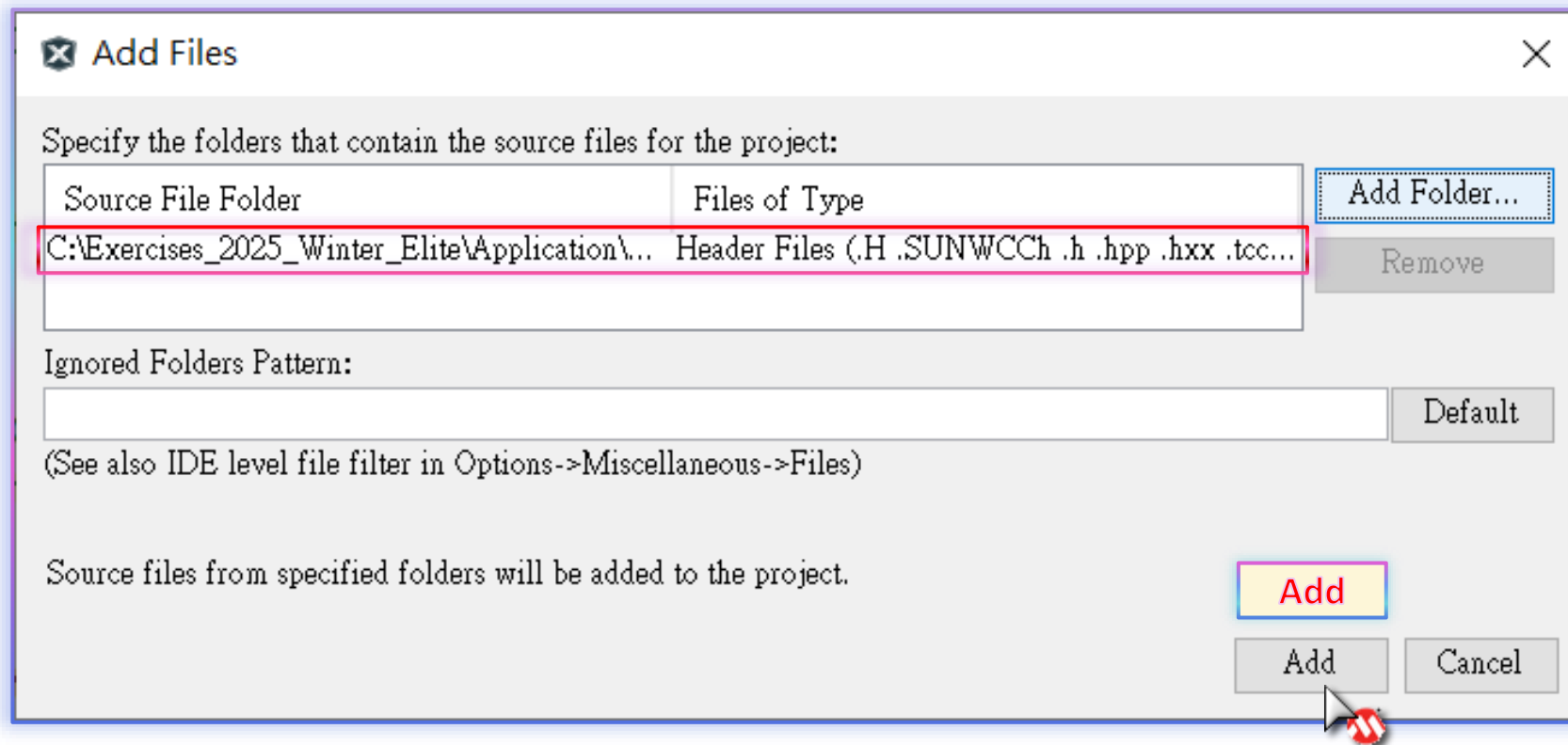
- Choice Files of type : **Header Files (.H .SUNWCCh .h .hpp .hxx)**
- Select folder “\[Your Project]\src\AIML_firmware\”
- Click **Select**.



Porting Deploy Model to Custom Project

Step 5

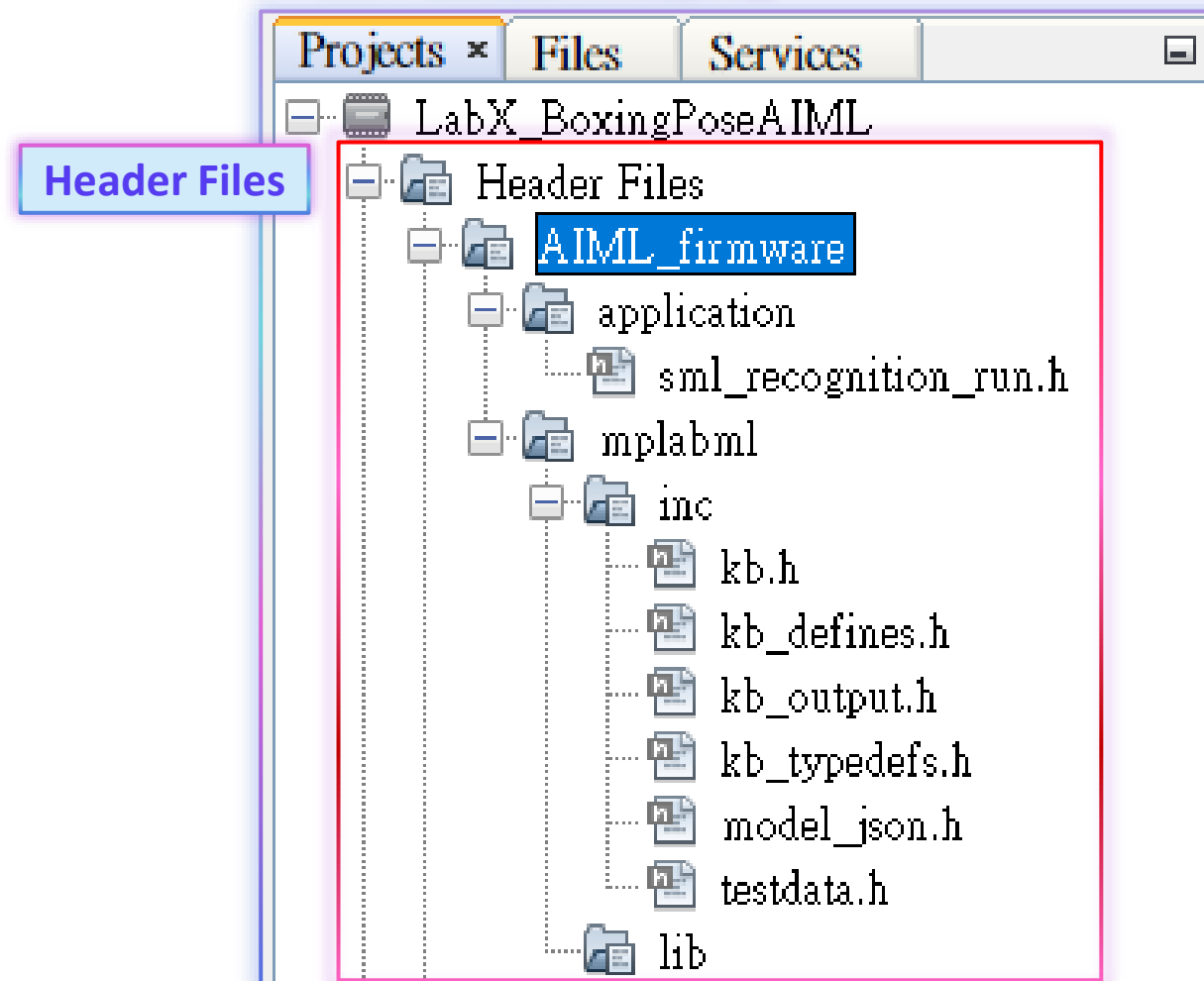
- Click **Add** then all the **.h** files in folder “\AIML_firmware\” will be added to project.



Porting Deploy Model to Custom Project

Step 6

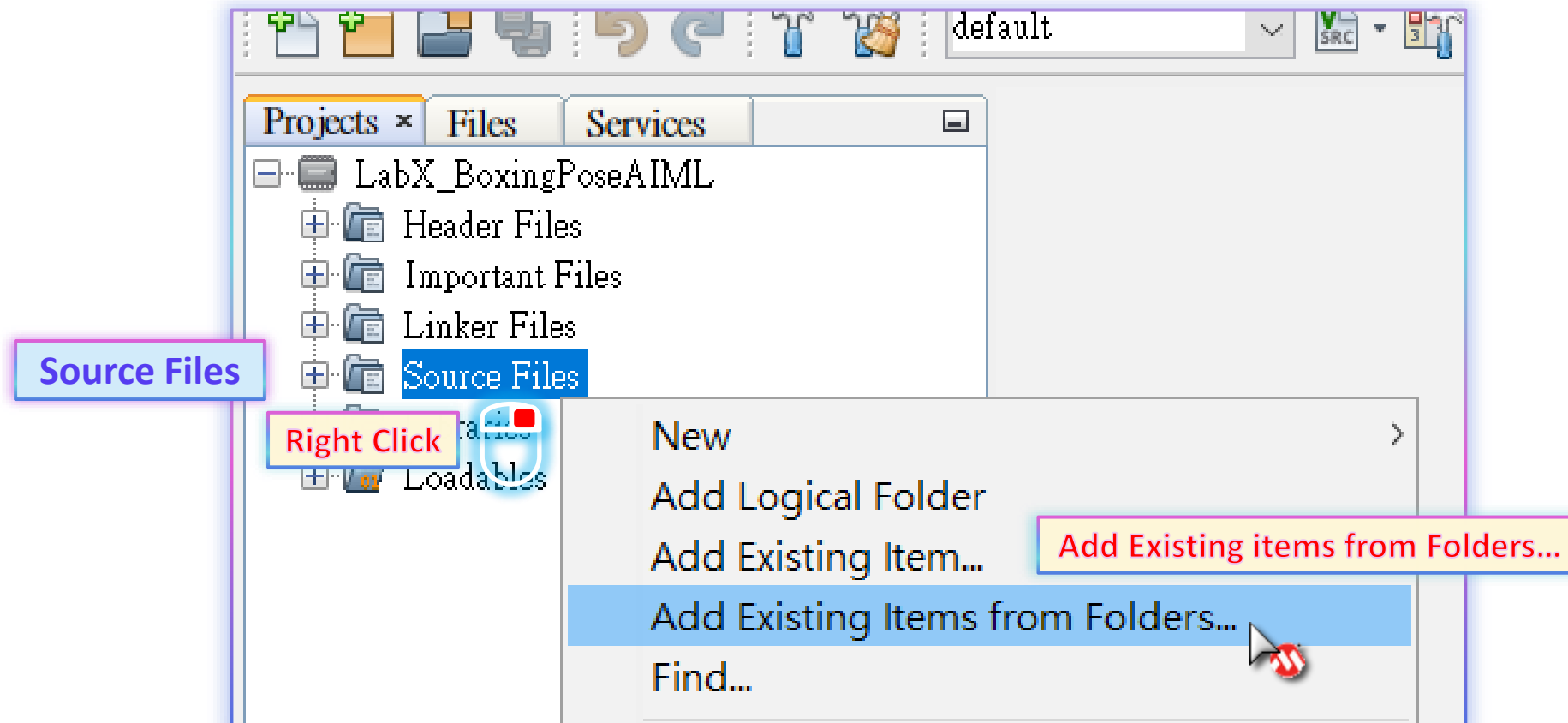
- Check if all the .h files in “\AIML_firmware\” are added into Header File folder.



Porting Deploy Model to Custom Project

Step 7

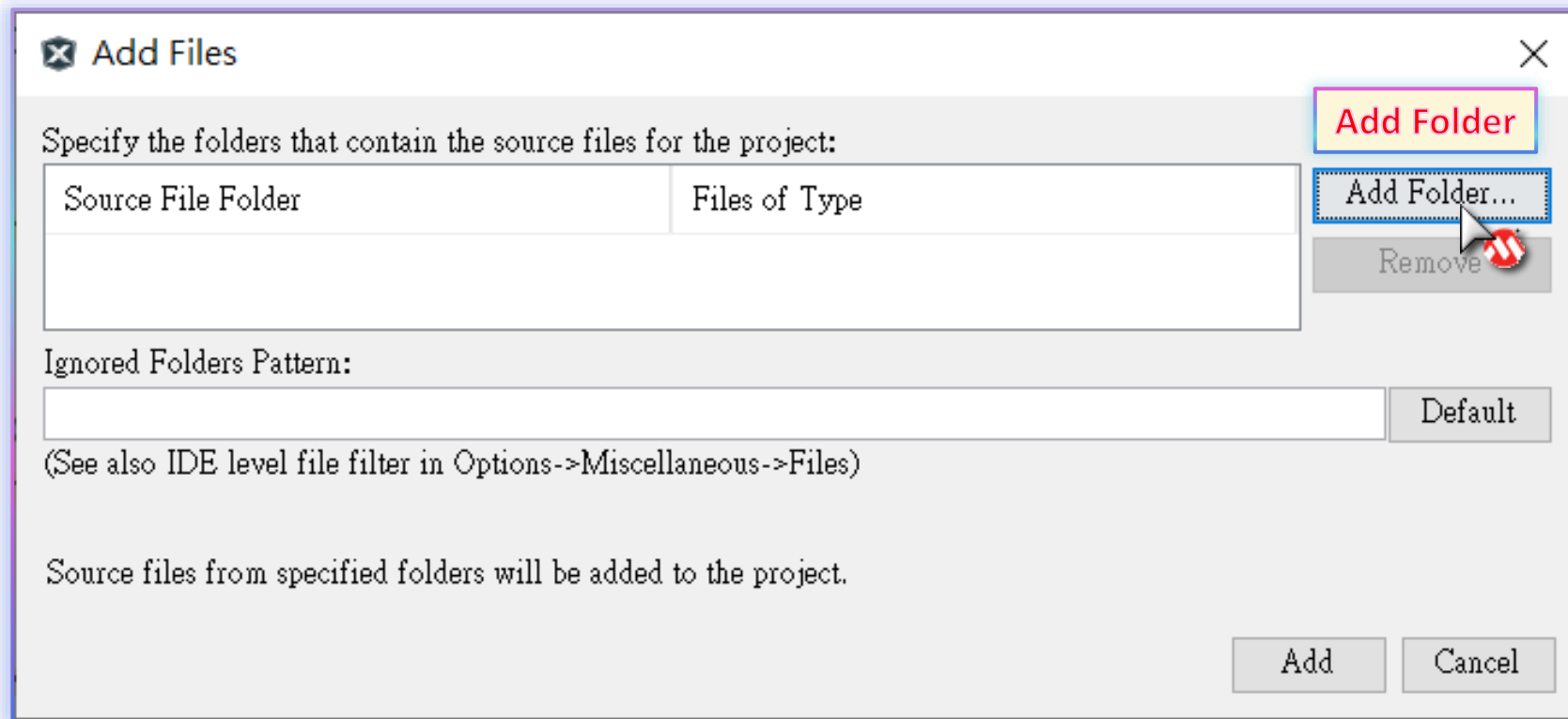
- Return your project in MPLAB X IDE.
- Right click on **Source Files** -> **Add Existing Items from Folders.**



Porting Deploy Model to Custom Project

Step 8

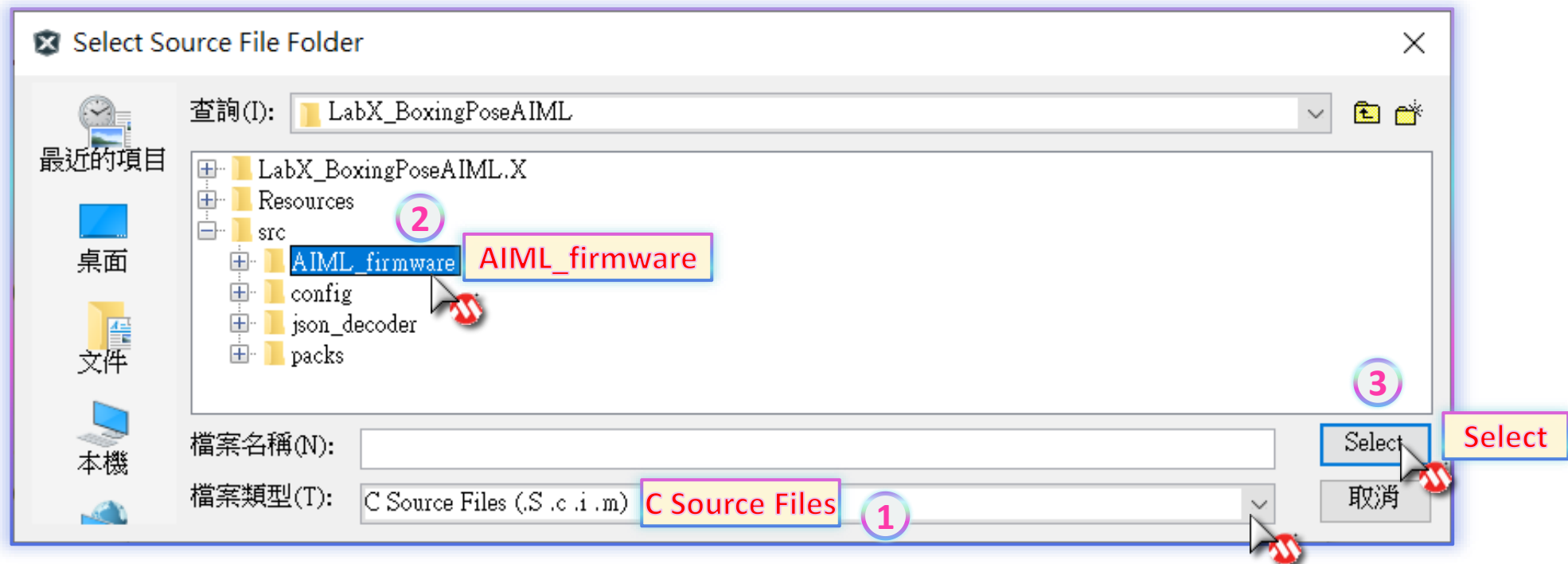
- Click **Add Folder**.



Porting Deploy Model to Custom Project

Step 9

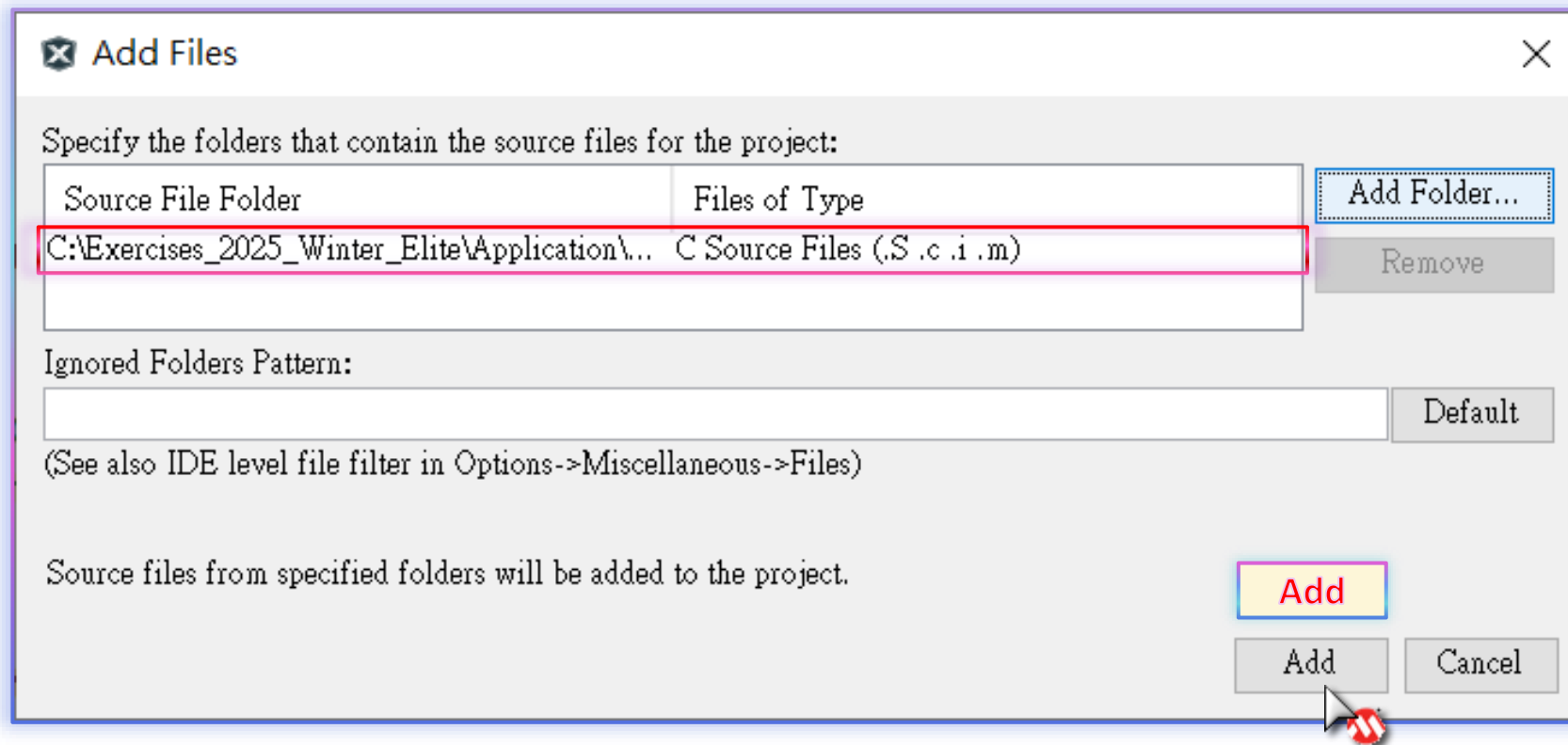
- Choice Files of type : **C Source Files (.S .c .i .m)**
- Select folder “[Your Project]\src\AIML_firmware\”
- Click **Select**.



Porting Deploy Model to Custom Project

Step 10

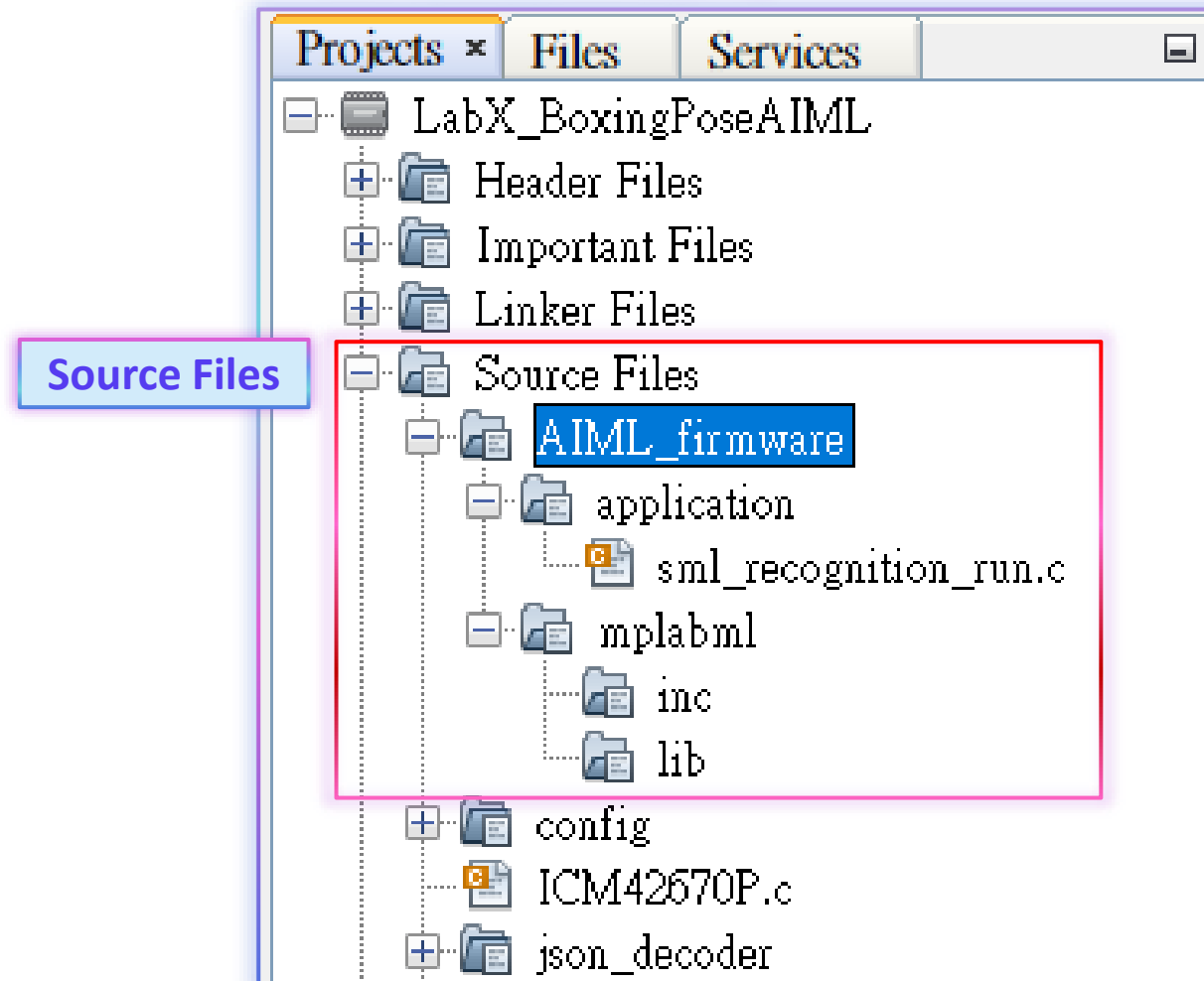
- Click **Add** then all the **.c** files in "**\AIML_firmware**" will be added to project.



Porting Deploy Model to Custom Project

Step 11

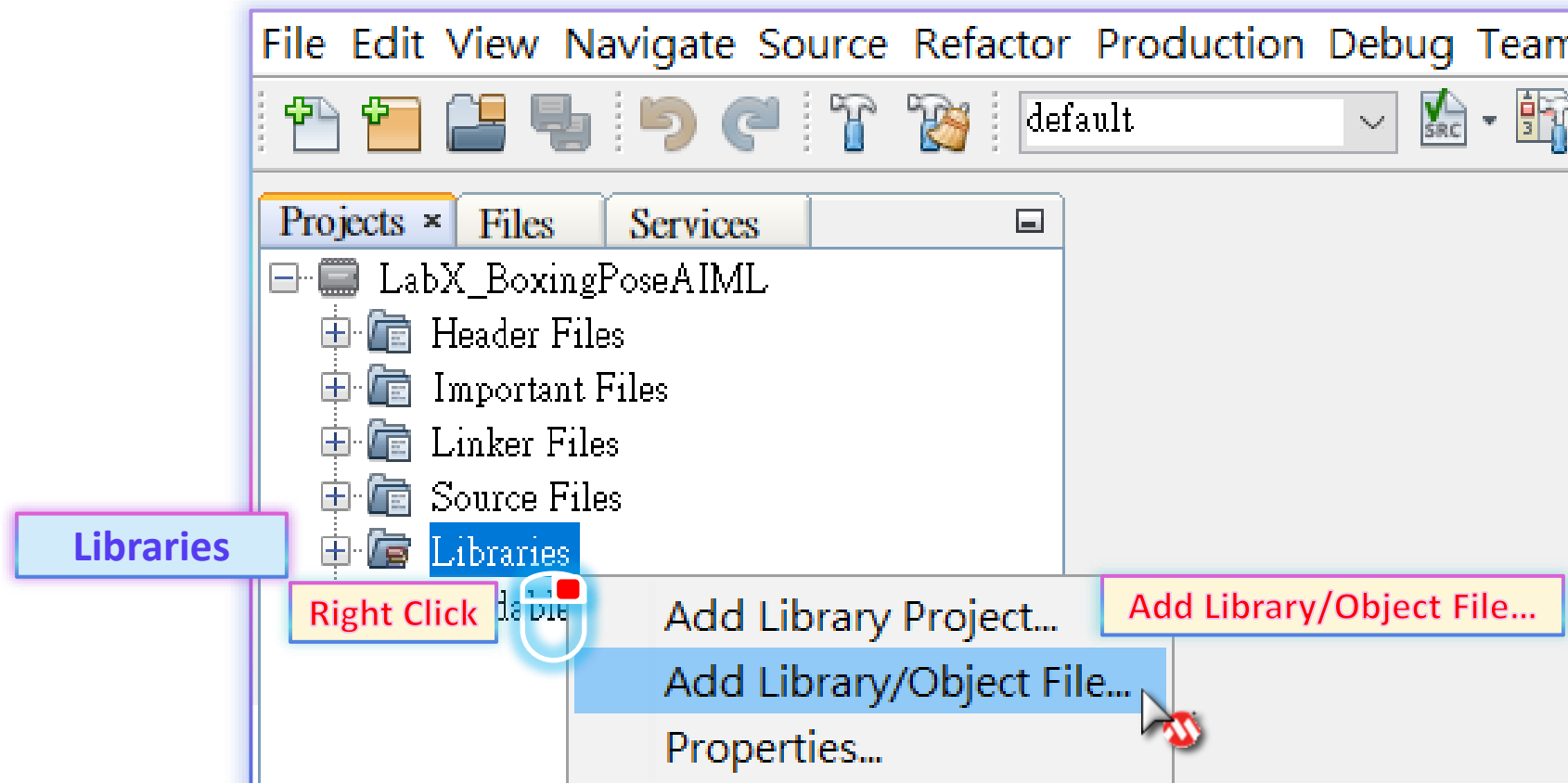
- Check if all the .c files in “\AIML_firmware\” are added into Source File folder.



Porting Deploy Model to Custom Project

Step 12

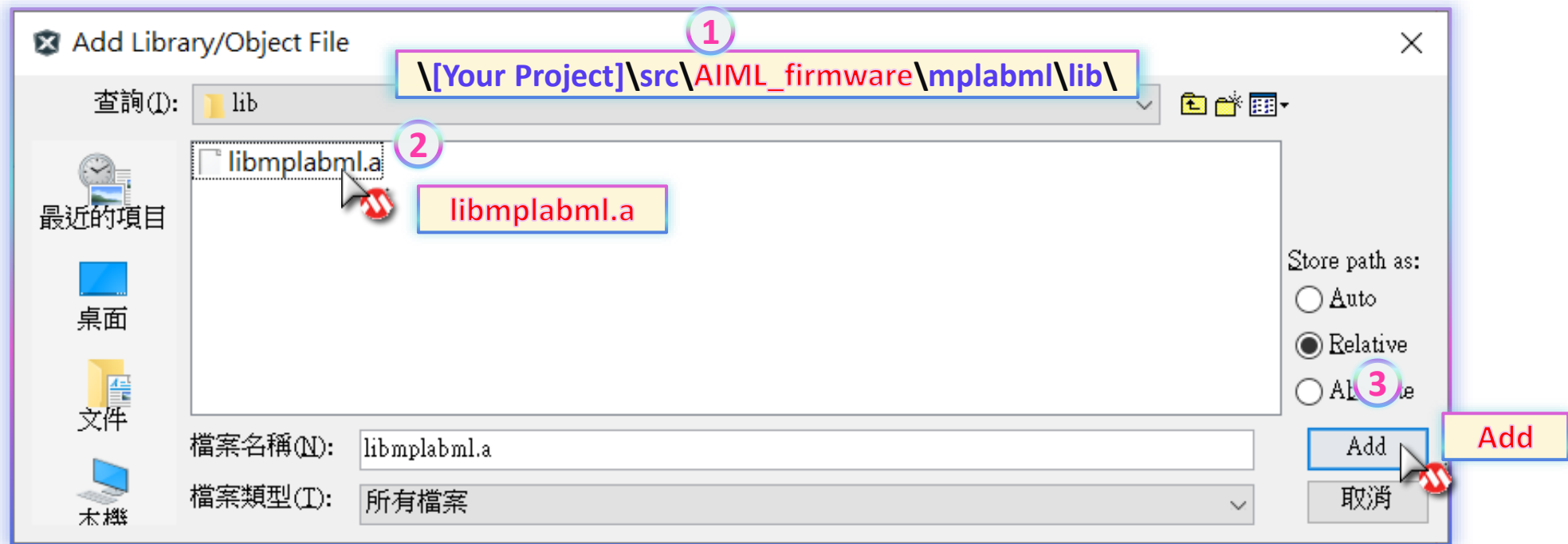
- Return your project in MPLAB X IDE.
- Right click **Libraries** -> **Add Library/Object File...**



Porting Deploy Model to Custom Project

Step 13

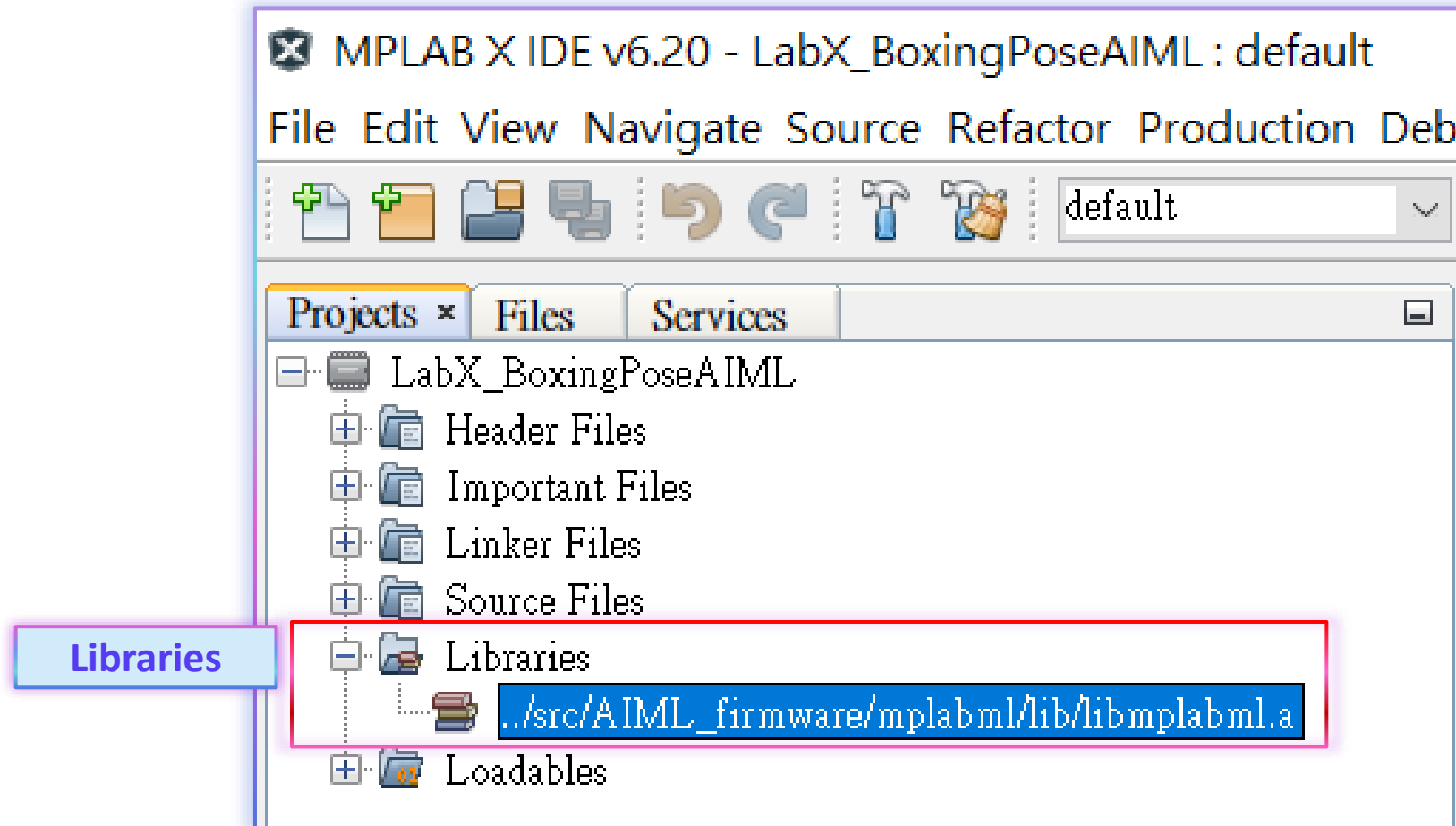
- Find the folder “[Your Project]\src\AIML_firmware\mplabml\lib\”
- Select Library file libmplabml.a
- Click Select.



Porting Deploy Model to Custom Project

Step 14

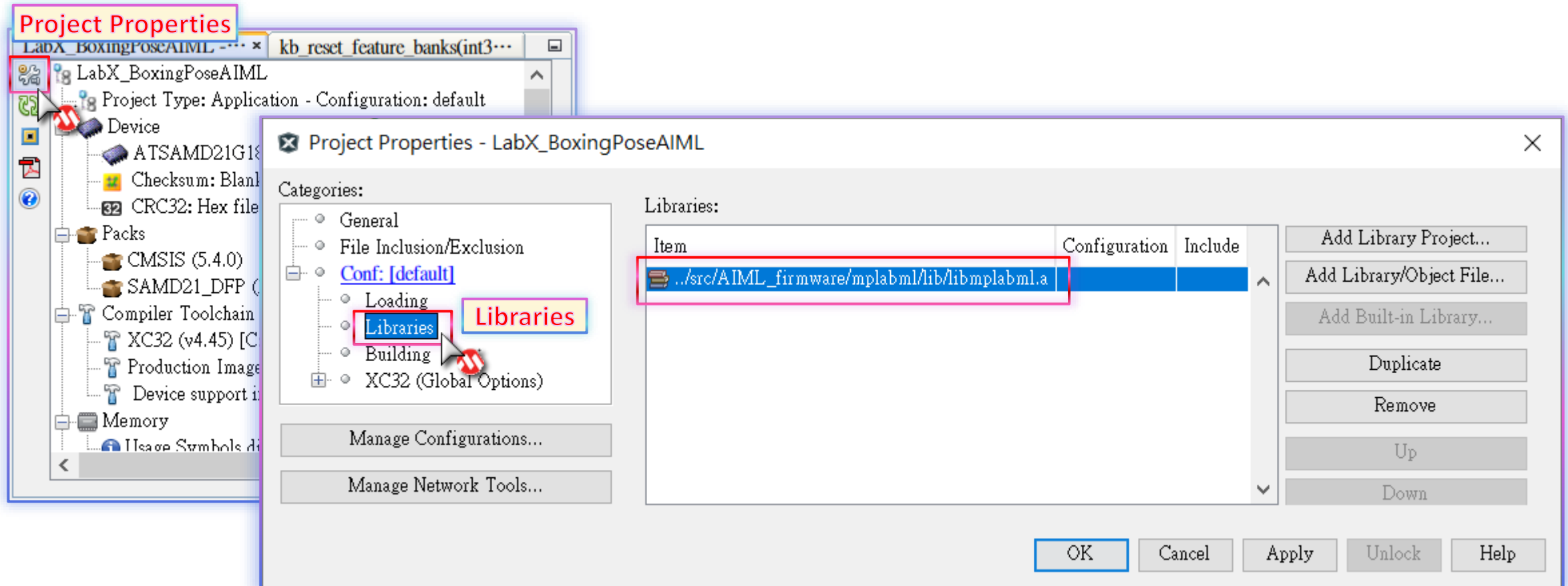
- Check if the **libmplabml.a** files are added into **Libraries** folder of your project.



Porting Deploy Model to Custom Project

Step 15

- You could also check to the **libraries** file in **Project Properties** by click on  .



Porting Deploy Model to Custom Project

Step 16

- Add “\mplabml\inc” & “\application\” in \AIML_firmware\” to Common include dirs.

The screenshot shows the MPLAB X IDE interface with the 'Project Properties' dialog open for the project 'LabX_BoxingPoseAIML'. The 'XC32 (Global Options)' category is selected in the left pane. The 'Common include dirs' field in the 'Options for xc32-gcc (v4.45)' section is highlighted, and a 'Browse...' button is visible. A secondary dialog, 'Common include dirs', is open, showing a list of directories: '..\src\AIML_firmware\mplabml\inc' and '..\src\AIML_firmware\application'. The 'Browse...' button is also highlighted in this dialog. The 'OK' button is visible at the bottom of the 'Common include dirs' dialog.

Project Properties - LabX_BoxingPoseAIML

Categories:

- General
- File Inclusion/Exclusion
- Conf: [default]
- Loading
- Libraries
- Building
- XC32 (Global Options)**
- xc32-gcc
- xc32-g++
- xc32-ld
- xc32-ar
- Analysis

Options for xc32-gcc (v4.45)

Option categories: (All Options) Reset

Stack Smashing Protector: No Stack Protector

Default device support: Do not override

Don't delete intermediate files: ☐

Link-Time Optimizations: ☐

Use GP relative addressing threshold: ☐

Common include dirs: ..\src\AIML_firmware\mplabml\inc

Common include dirs

Destroy Down Up Browse...

..\src\AIML_firmware\mplabml\inc

..\src\AIML_firmware\application

(Enter or 'Browse' string here)

Relative paths are from MPLAB X project directory.

OK Cancel

\AIML_firmware\mplabml\inc

\AIML_firmware\application

Porting Deploy Model to Custom Project

Step 17

- The example Auto ML model inference result.

```
#include "sml_recognition_run.h"

#define SNSR_NUM_AXES 6

void ICM42670P_GetQueuedSensorRaw( uint16 *SensorRaw ) {...}

void Inference( void )
{
    int32_t ClassMapsId;
    uint16_t SensorRawData[SNSR_NUM_AXES];

    while( ICM42670P_GetQueuedSensorRaw( SensorRawData ) )
    {
        ClassMapsId = sml_recognition_run( (int16_t *)SensorRawData, SNSR_NUM_AXES );
        printf( "Auto ML Inference Result : %d", ClassMapsId );
    }
}
```

```
"ModelDescriptions": [
{
    "Name": "BOXING_RECOGNITION_RANK_3",
    "ClassMaps": {
        "1": "Freeze",
        "2": "Righthook",
        "3": "Straight",
        "4": "Uppercut",
        "0": "Unknown"
    },

```

ML Data Collector X Touch X Terminal X XY Plot X

Auto ML Inference Result : 1

JSON Decoder helper Library

JSON Decoder helper Library

Step 1

- There has a **JSON decoder library** been implemented in **LabX_BoxingPoseAIML** project. It's helpful to auto parser the **model_json.h** (**model.json converted into C header form**) to get correct mapping of Classified ID of you deployed Model.

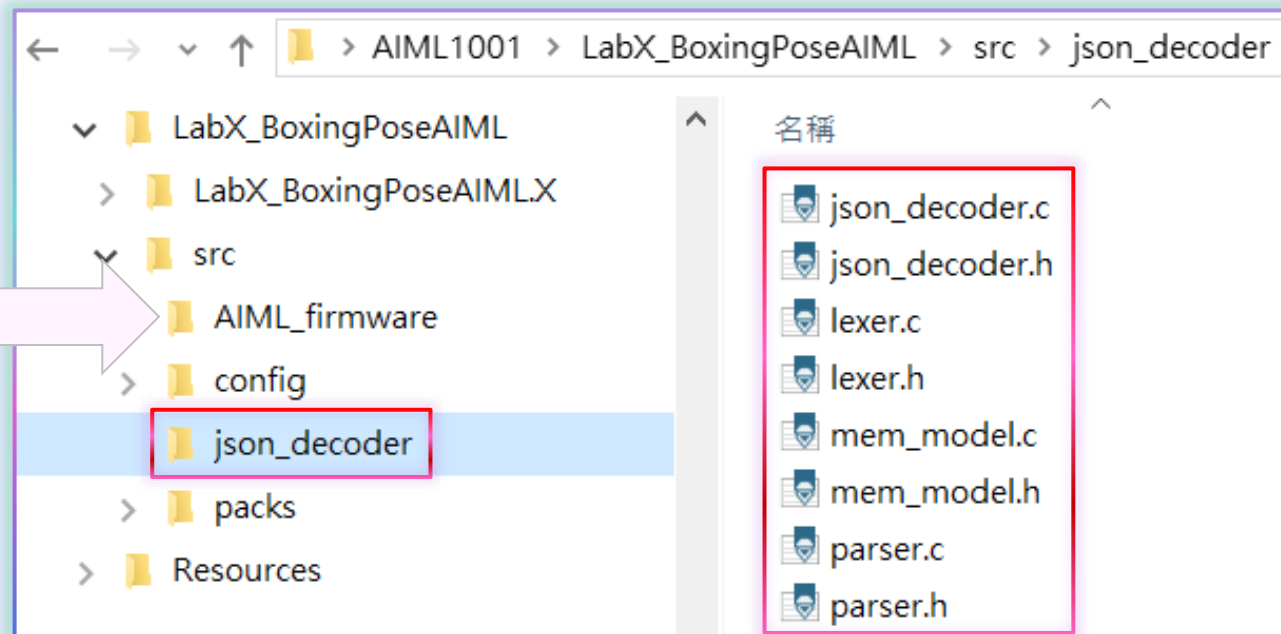
```
#ifndef __MODEL_JSON_H__
#define __MODEL_JSON_H__

const char recognition_model_string_json[] =
{"{\"NumModels\":1,\"ModelIndexes\":{\"0\":\"BOXING_RECOGNITION_RANK_3\"},\"ModelDescriptions\":{\"Name\":\"BOXING_RECOGNITION_RANK_3\",\"ClassMaps\":{\"1\":\"Freeze\",\"2\":\"Righthook\",\"3\":\"Straight\",\"4\":\"Uppercut\",\"0\":\"Unknown\"},\"ModelType\":\"DecisionTreeEnsemble\",\"FeatureFunctions\":[\"25thPercentile\",\"AbsoluteAreaofLowFrequency\",\"AbsoluteAreaofLowFrequency\",\"GlobalPeaktoPeak\",\"GlobalPeaktoPeakofLowFrequency\",\"TwoColumnMeanDifference\",\"TwoColumnMeanDifference\",\"TwoColumnPeakToPeakDifference\"]}}"};

int32_t recognition_model_string_json_len =
sizeof(recognition_model_string_json);

#endif /* __MODEL_JSON_H__ */
```

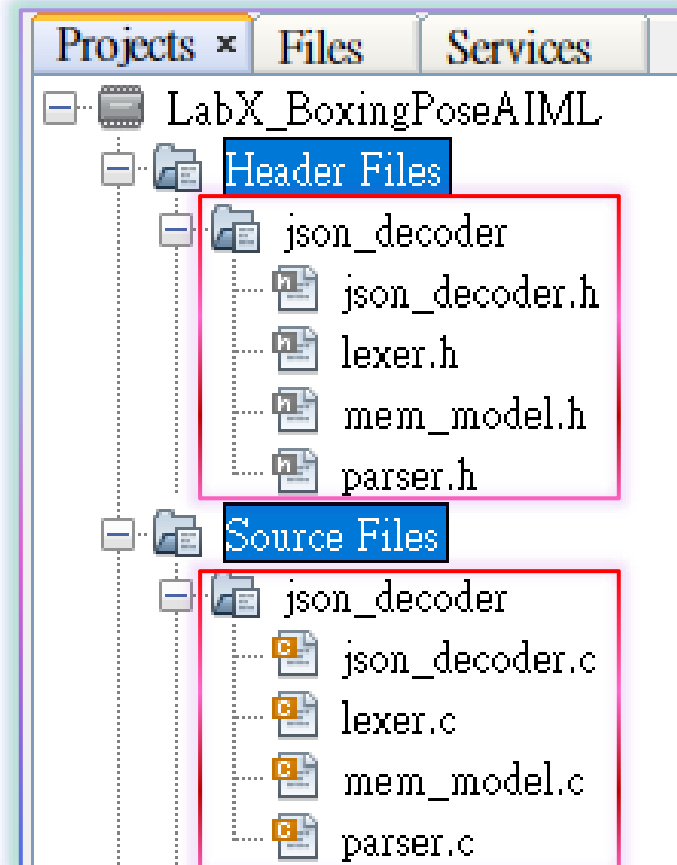
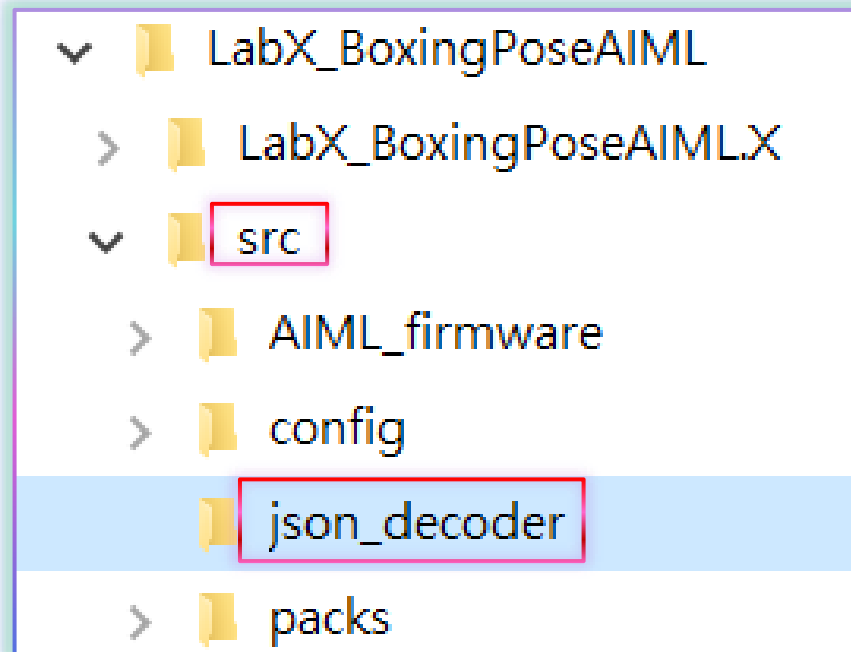
model_json.h



JSON Decoder helper Library

Step 2

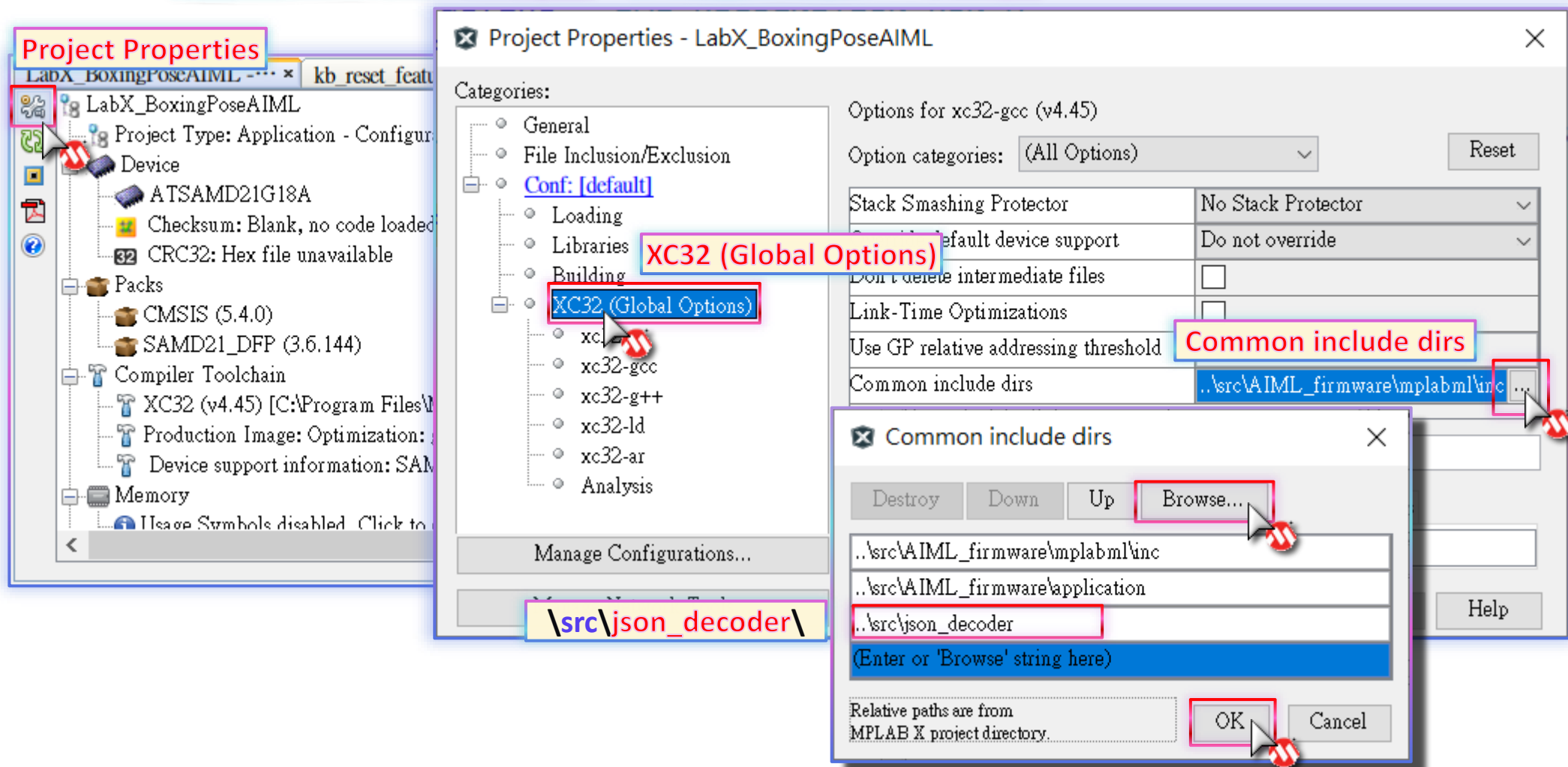
- Copy whole “\json_decoder\” folder to under “\src” folder of your project.
- Add *.h and *.c into “Header Files” and “Source Files” of your Project in MPLAB X IDE.



JSON Decoder helper Library

Step 3

- Add "`..\src\json_decoder\`" to Common include dirs.



JSON Decoder helper Library

Step 4

- The example to add JSON Decoder helper Library into your project.

```
#include "json_decoder.h"
#include "model_json.h"

#define MODEL_NAME_LENGTH    50
#define CLASSIFY_ID_MAXIMUM  5
#define CLASSIFY_NAME_LENGTH 20

typedef struct
{
    char modelName[MODEL_NAME_LENGTH];
    char ClassMaps[CLASSIFY_ID_MAXIMUM][CLASSIFY_NAME_LENGTH];
} MODEL_DESCRIPTIONS;

MODEL_DESCRIPTIONS DeployModel;

void ParserJSON_ClassMaps( char *string_json, int32_t string_json_len )
{
    char jsonbuffer[500];
    jsonNode_t *root, *objModelDescriptions;
    jsonNode_t *objClassMaps;
```

JSON Decoder helper Library

Step 5

- The example to add JSON Decoder helper Library into your project.

```
memcpy( jsonbuffer, string_json, string_json_len );

JSON_DECODER_fromString(jsonbuffer);
JSON_DECODER_getRoot(&root);
JSON_DECODER_getObject(root, "ModelDescriptions", &objModelDescriptions);
JSON_DECODER_getObject(objModelDescriptions, "ClassMaps", &objClassMaps);
JSON_DECODER_getString(objModelDescriptions, "Name", MODEL_NAME_LENGTH, DeployModel.ModelName);

printf("Auto ML Model Name : %s\r\n", DeployModel.ModelName );

printf("Auto ML Classify Maps :\r\n");
for( int Clsid=0 ; Clsid<CLASSIFY_ID_MAXIMUM ; Clsid++ )
{
    char ClsidStr[2];
    memset( DeployModel.ClassMaps[Clsid], 0, CLASSIFY_NAME_LENGTH );
    sprintf( ClsidStr, "%d", Clsid );
    JSON_DECODER_getString(objClassMaps, ClsidStr, CLASSIFY_NAME_LENGTH, DeployModel.ClassMaps[Clsid]);
    printf(" - Id %d : %s\r\n", Clsid+16, Clsid, DeployModel.ClassMaps[Clsid] );
}
}
```

JSON Decoder helper Library

Step 6

- The example to add JSON Decoder helper Library into your project.

```
void ParserJSON_ClassMaps( char *string_json, int32_t string_json_len )
{...}

void main( void )
{
    kb_model_init();
    ParserJSON_ClassMaps( (char *)recognition_model_string_json, recognition_model_string_json_len );
}
```

Text array and length in **model_json.h**

```
#ifndef __MODEL_JSON_H__
#define __MODEL_JSON_H__

const char recognition_model_string_json[] =
{"{\"NumModels\":1,\"ModelIndexes\":{\"0\":\"BOXING_RECOGNITION_RANK_3\"},\"ModelDescriptions\":{\"Name\":\"BOXING_RECOGNITION_RANK_3\",\"ClassMaps\":{\"1\":\"Freeze\",\"2\":\"Righthook\",\"3\":\"Straight\",\"4\":\"Uppercut\",\"0\":\"Unknown\"},\"ModelType\":\"DecisionTreeEnsemble\",\"FeatureFunctions\":{\"25thPercentile\",\"AbsoluteAreaofLowFrequency\",\"AbsoluteAreaofLowFrequency\",\"GlobalPeaktoPeak\",\"GlobalPeaktoPeakofLowFrequency\",\"TwoColumnMeanDifference\",\"TwoColumnMeanDifference\",\"TwoColumnPeakToPeakDifference\"}}}}";

int32_t recognition_model_string_json_len = sizeof(recognition_model_string_json);

#endif /* __MODEL_JSON_H__ */
```

model_json.h

JSON Decoder helper Library

Result

- The JSON parser example is using the [LabX_BoxingPoseAIML.X](#) project.

```
ML Data Collector X Touch X Terminal X XY Plot X
Auto ML Model Name : BOXING_RECOGNITION_RANK_3
Auto ML Classify Maps :
- Id 0 : Unknown
- Id 1 : Freeze
- Id 2 : Righthook
- Id 3 : Straight
- Id 4 : Uppercut
```

Parser Result

```
#ifndef __MODEL_JSON_H__
#define __MODEL_JSON_H__
```

model_json.h

```
const char recognition_model_string_json[] =
{"{\\"NumModels\\":1,\\"ModelIndexes\\":{\\"0\\":\\"BOXING_RECOGNIT
ION_RANK_3\\"},\\"ModelDescriptions\\":[{\\"Name\\":\\"BOXING_RECO
GNITION_RANK_3\\",\\"ClassMaps\\":{\\"1\\":\\"Freeze\\",\\"2\\":\\"Rig
hthook\\",\\"3\\":\\"Straight\\",\\"4\\":\\"Uppercut\\",\\"0\\":\\"Unkno
wn\\"},\\"ModelType\\":\\"DecisionTreeEnsemble\\",\\"FeatureFunci
ons\\":[\\"25thPercentile\\",\\"AbsoluteAreaofLowFrequency\\",\\"A
bsoluteAreaofLowFrequency\\",\\"GlobalPeaktoPeak\\",\\"GlobalPea
ktoPeakofLowFrequency\\",\\"TwoColumnMeanDifference\\",\\"TwoCol
umnMeanDifference\\",\\"TwoColumnPeakToPeakDifference\\"]}}"};
```

```
int32_t recognition_model_string_json_len =
sizeof(recognition_model_string_json);
```

```
#endif /* __MODEL_JSON_H__ */
```

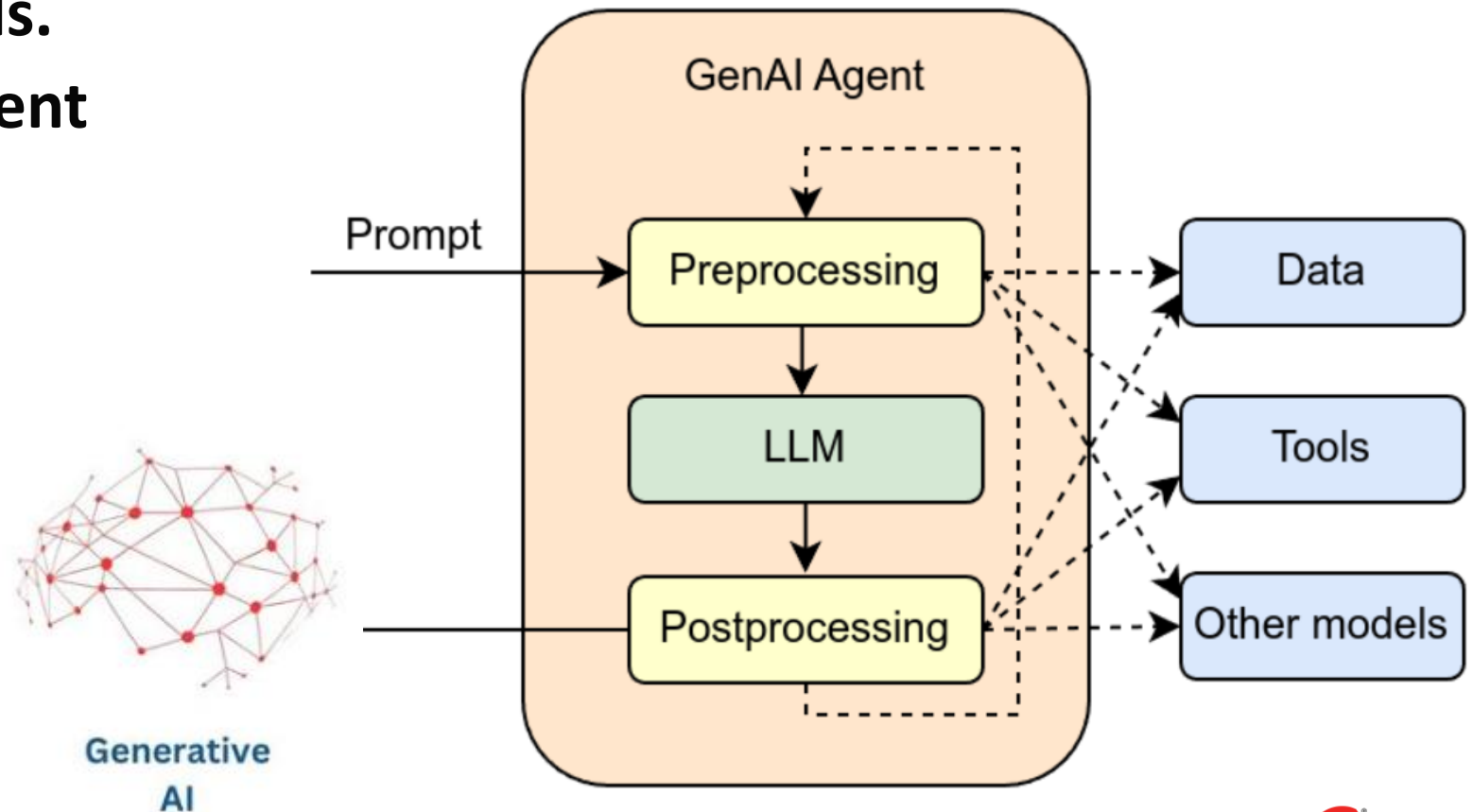
```
{
  "NumModels": 1,
  "ModelIndexes": {
    "0": "BOXING_RECOGNITION_RANK_3"
  },
  "ModelDescriptions": [
    {
      "Name": "BOXING_RECOGNITION_RANK_3",
      "ClassMaps": {
        "1": "Freeze",
        "2": "Righthook",
        "3": "Straight",
        "4": "Uppercut",
        "0": "Unknown"
      },
      "ModelType": "Decision Tree Ensemble",
```

model.json

Large Language Models

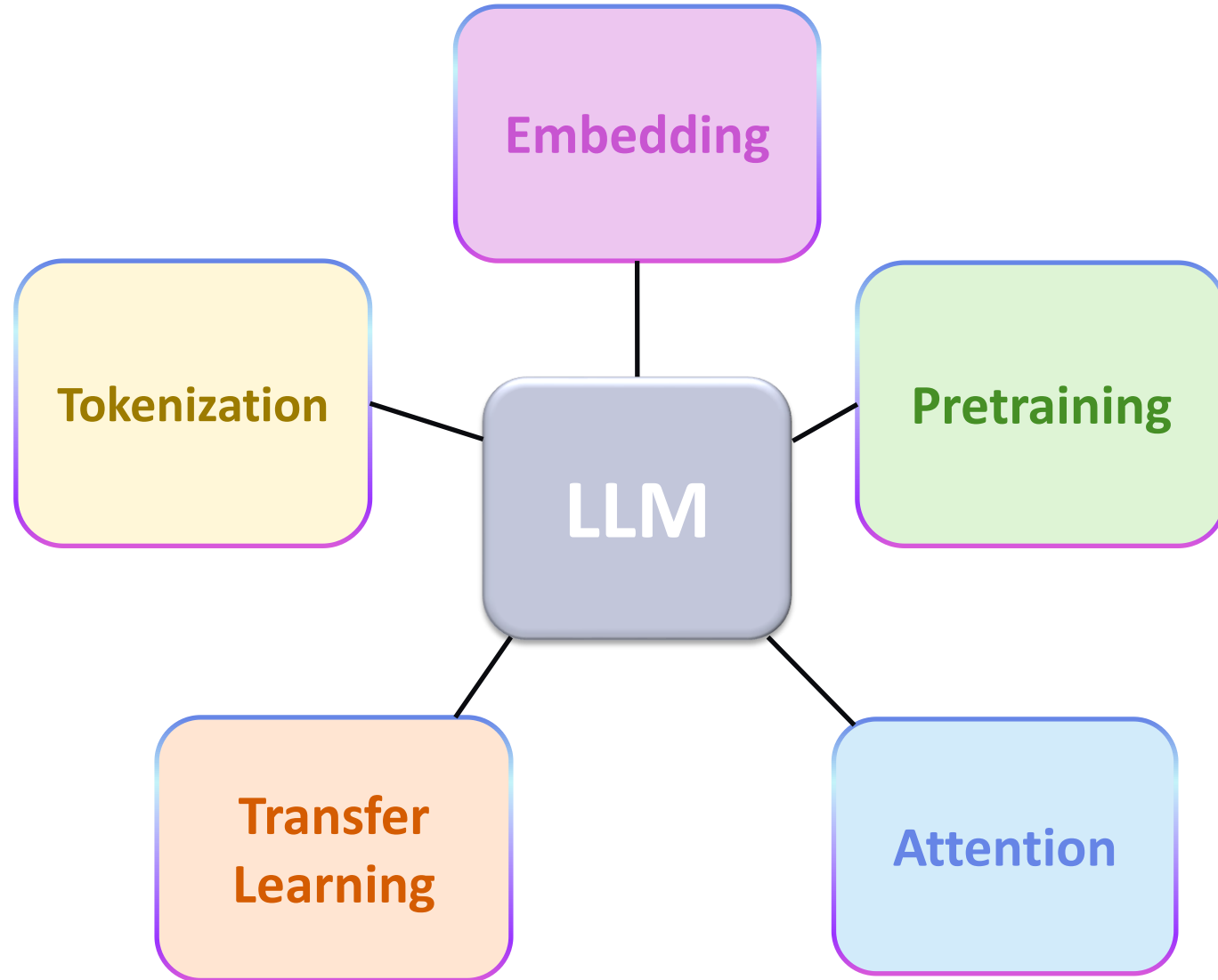
Generative AI

- G-AI models learn patterns and structures of input data and produce new data with similar characteristics.
- Input data such as instructions, images, etc. for G-AI is often prompted by Large Language Models.
- G-AI creates new content such as text, videos., pictures, audio, etc.



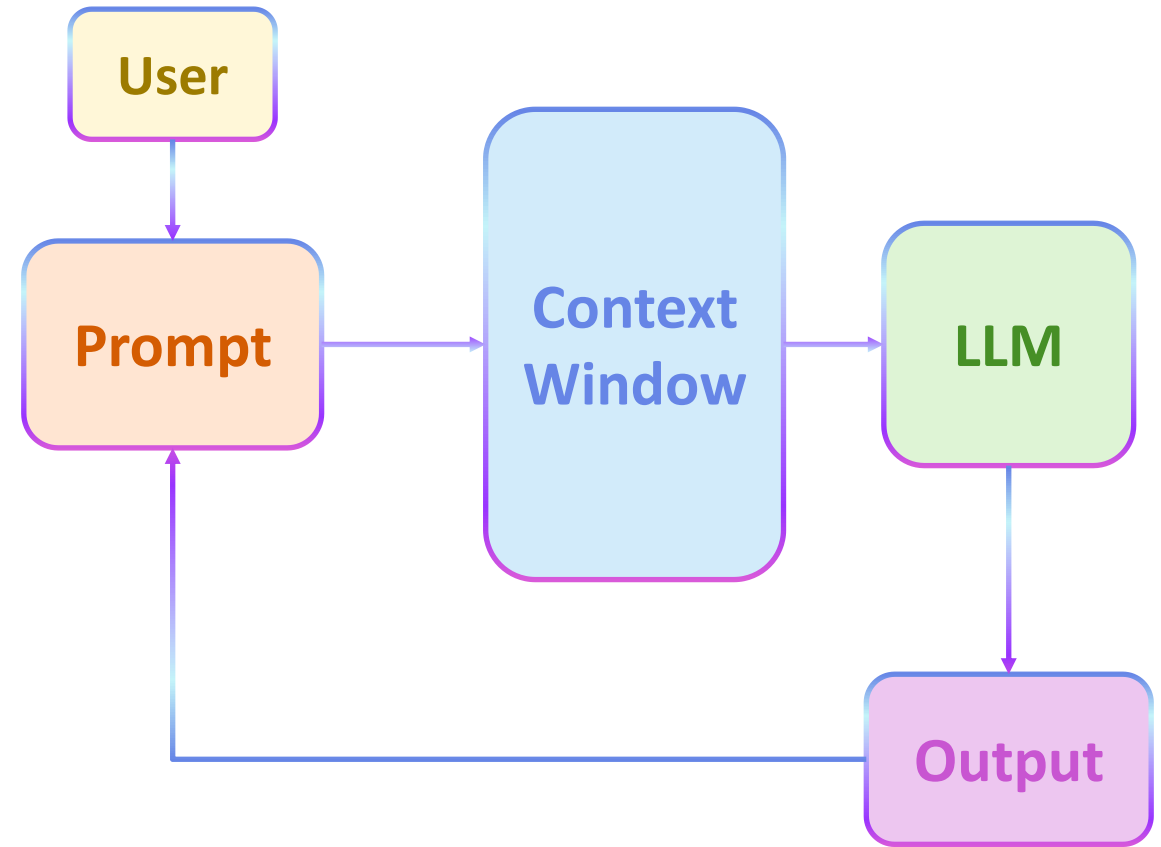
Large Language Models

- LLM's are transformer-based Deep Neural Networks(DNN)
- Transformer is a DL architecture developed by Google and based on multi-head attention mechanism.
- Transfer Learning
- Popular LLM:
 - OpenAI's ChatGPT
 - Microsoft's Copilot
 - Meta's LLaMA
 - Anthropic's Claude model
 - Mistral AI's models



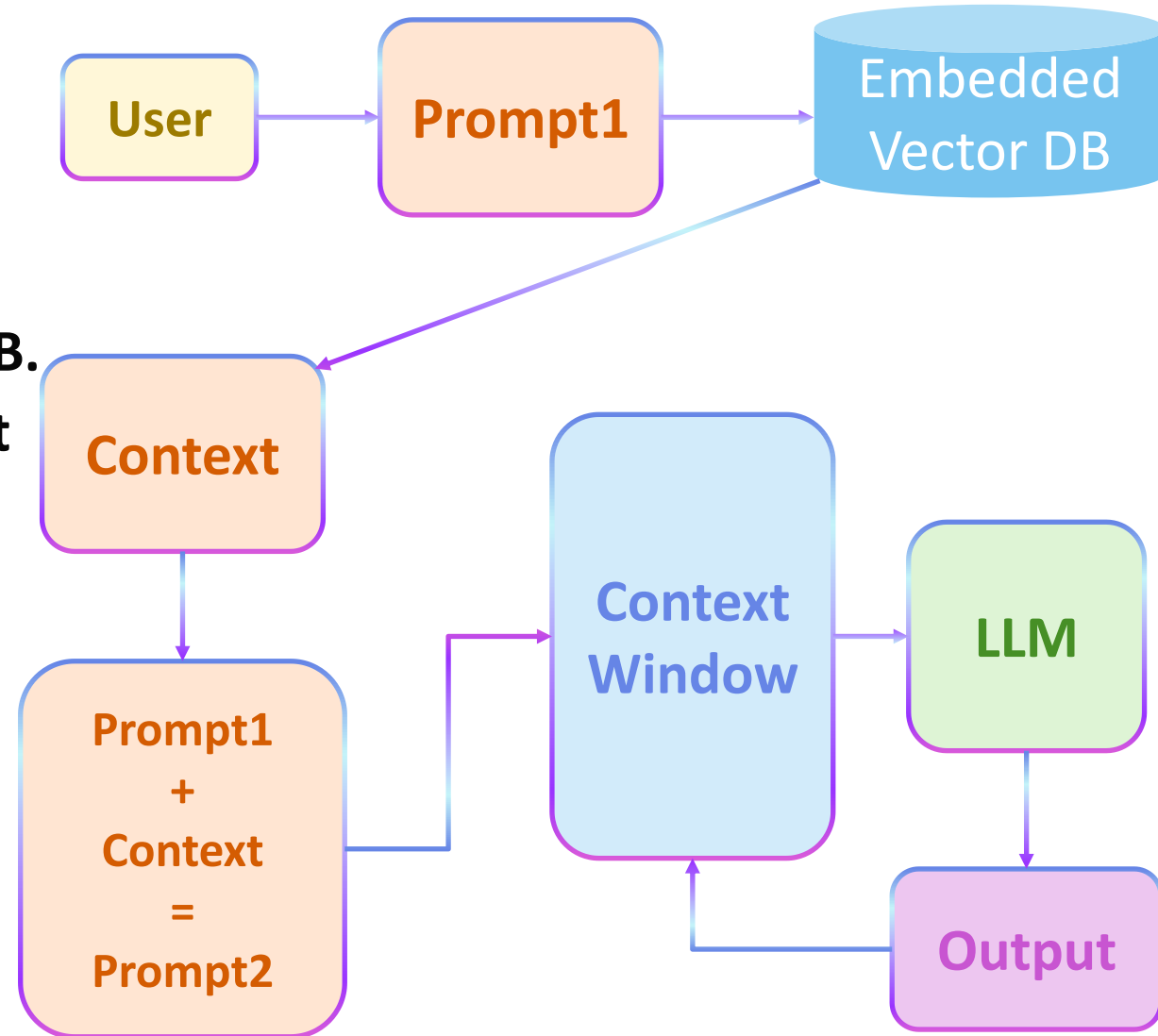
Interfacing with basic LLM prompt

- User initiate session with an initial prompt.
- Prompt goes to LLM context window.
- LLM uses prompt to generate output.
- Output is added to the context window along with additional prompts.



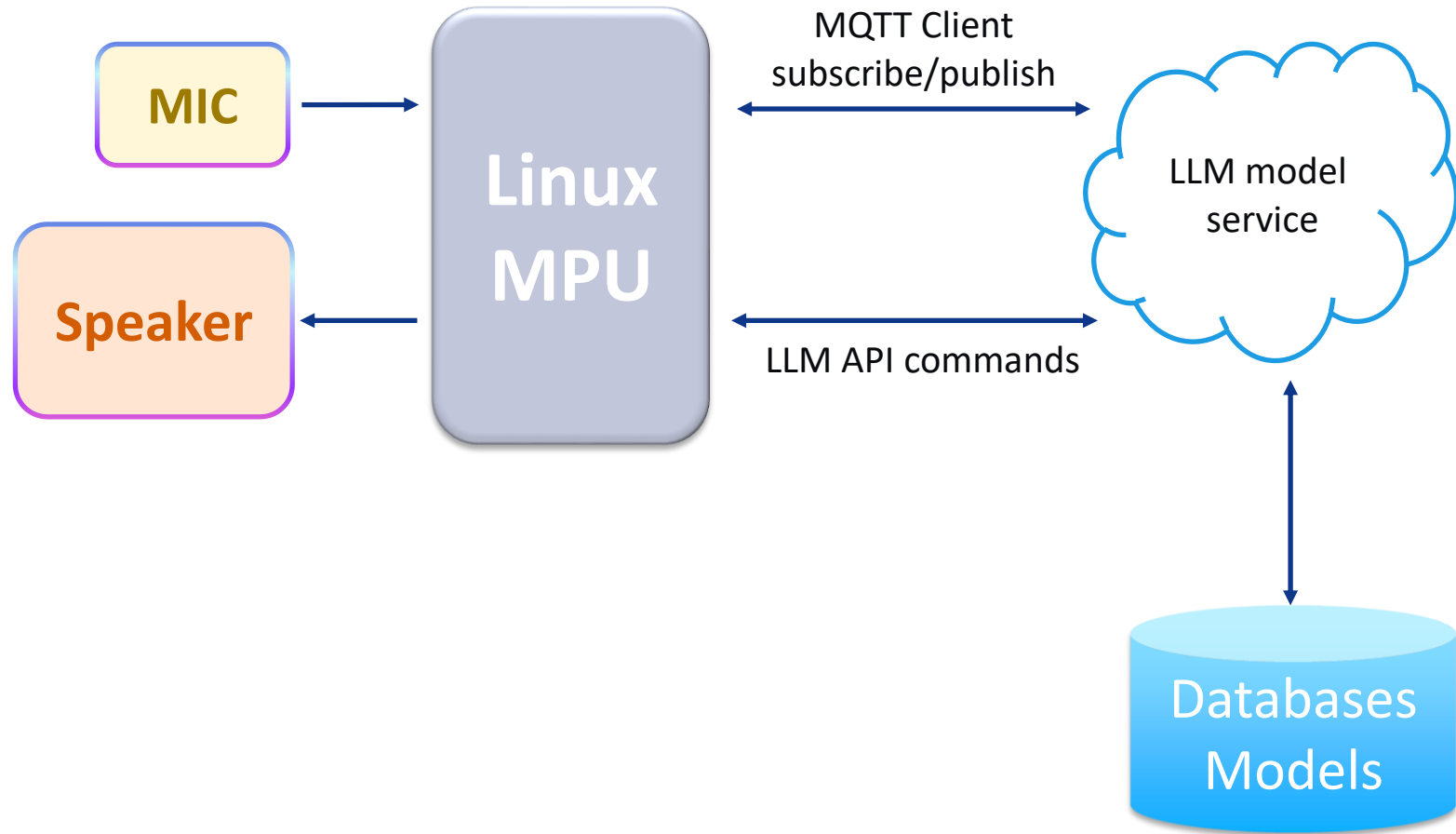
Retrieval Augmented Generation LLM prompt

- RAG provides a performance improvement as compared with LLM alone by reducing bad responses and focusing LLM on relevant data.
- Initiates prompt will search in the vector DB.
- The original prompt is added to the context and the expanded prompt.
- LLM generates output.
- Add New prompts until the output is satisfactory.



Connecting HW to LLM

- Use LLM API's to get answers and return natural language on hardware.
- HW uses inputs from the microphone to take in speech and write text then sends a text prompt to the LLM.
- The LLM responds to the text prompt, then the MPU uses text to speech the read the response out loud on the connected speaker.



Microchip Trademarks

<https://www.microchip.com/en-us/about/legal-information/microchip-trademarks>

Trademark	Description/Appropriate Name	Notes
ClockWorks™	configurator, device, family, product, series	US only
CodeGuard™	security	
CryptoAuthentication™	development library, device, family, IC, library	
CryptoAutomotive™	development kit, security IC, technology	
CryptoCompanion™	chip	
CryptoController™	solution, TPM, trusted platform module	
CryptoMemory®	cryptographic security IC, IC	
Cryptotr®	device, transponder	
dsPIC®	DSC, digital signal controller	
	dsPIC® logo	
dsPICDEM™	demonstration board, development board	
dsPICDEM.net™	board	
Dynamic Averaged Matching™ (DAM™)	architecture	
ECAN™	solution, technology	
Espresso T11™	device	
EtherGREEN™	family, platform, technology	
EtherSynch®	family, platform, technology	US only
EtherOpen™	cable compensation	
Flasher®	controller, family, product	US only
FlexPWR®	technology	
Frequency on Demand®		Registered in other countries (not US)
GenSic®	controller, sensing, technology	
GriffTime™	device, GNSS time server, time server	
HILDO®	family, regulator	
Hyper Speed Control®	architecture, family	US only
HyperLight Load®	mode	US only
In-Circuit Serial Programming™ (ICSP™)	programming capability	
Ideabridge™	dual MOSFET-bridge rectifier, rectifier	
IGAT™	board, demonstration board	
IGLOO®	device, family, FPGA, series	
INOC™	sniffer, technology	
Note: *Always check with the Legal Department for the most current trademarks.		

Trademark	Description/Appropriate Name	Notes
	Microchip name and logo	This is the vertical representation; the horizontal representation is also available.
	Microsemi logo	Do not use
Microsemi®		Do not use
Mind™	analog simulator, analog simulator tool	
MW™	application, coordinator, mesh, network, protocol, software, stack, wireless networking protocol	
MOST®	board, cooperation, design, device, evaluation kit, NetServices, network platform, product, specification, system, technology	
	MOST® logo	
InstaBench®	development suite	US only
MPASM™	assembler	
MPY™	device, product	
MPLAB®	Notes: C compiler, compiler, development ecosystem, ICD, ICE, IDE, IPE, in-circuit debugger, in-circuit emulator, integrated development environment, integrated programming environment, library, programmer, server, starter kit, universal device programmer, IC compiler Tool Names: AI Edge, Analog Designer, Analysis Tool Suite, Code Configurator, Connect Configurator, Data Visualizer, Discover, Extensions, Harmony, Mind™, Network Creator, PICKit™, Programmer-to-Go, Programmer-to-Go Mobile App, PTG, SC Power Simulator, Snap, Xpress	
	MPLAB® Certified logo	
MPLIB™	libraries, object librarian	
Note: *Always check with the Legal Department for the most current trademarks.		

Trademark	Description/Appropriate Name	Notes
Reggie Blocker™	attenuator, family of linear regulators, product line, technology	
REAL ICE™	in-circuit emulator	
RTAX™	FPGA	
RTG4™	FPGA	
SAM-BA®	applet, application, host, in-system programmer, ISP, monitor, software	
SAM-ICE™	emulator	
SenGenuity®	product, sensor	
Serial Quad I/O™ (SQI™)	family, flash device, product	
Silicon Storage Technology®		Japan only
SimpleMAP™	protocol	
SimpleM™	device, family, product	
SmartBuffer™	device, IC	
SmartFusion®	evaluation kit, FPGA, SoC, system-on-chip	US only
SmartHLS™	compiler, compiler software, IDE, software	
SMART-LS™	technology	
SpyASIC®	device	
SST®	Acronym for Silicon Storage Technology	
	SST logo	
A Microchip Technology Company		
STK®	starter kit	
storClad™	encryption technology, technology	
SuperFlash™	chip, macros, memory, technology	
SuperSwitcher™	back regulator, family, regulator	
SuperSwitcher II™	family	
Switcher™	family, product line, switch, technology	
Symcon™		China only
Symconcom®		
Synchradyne™	device, family, product	
SynServer™	clock, device, instrument, oscillator, server	
Note: *Always check with the Legal Department for the most current trademarks.		

Trademark	Description/Appropriate Name	Notes
	Adaptive, "a" logo	Registered in other countries (not US)
Adaptive®	adapter, cable, controller, driver, family, product, series, software, solution	
Adjacent Key Suppression™	technology	
AgileSwitch®	device, digital programmable gate driver, driver, family, gate driver	US only
AKC™	technology	
Analog for the Digital Age™	AFD family tagline	
Any Capacitor™	stable	
AnyIn™	capability, feature, structure	
AnyOut™	capability, feature	
Augmented Switching™	technology	
AVR®	architecture, CPU, device, family, MCU, microcontroller	
	AVR® logo	
AVR Freaks®	forum	
BestTime®	technology	
BitCloud®	SDK, software development kit	
BlueSky®	firewall, subscription service, technology	
BodyCom™	development kit, system, technology	
chipKIT®	I/O shield, board, hardware, network, platform, software	
	chipKIT logo	
ClockStudio™	application, program, software, utility	
Note: *Always check with the Legal Department for the most current trademarks.		

Trademark	Description/Appropriate Name	Notes
IsiDrive™	driver board, integrated solution, power module	
Intelligent Parafire™	technology	
InstaIMOS™	family, power stage family	US only
Inter-Chip Connectivity™	technology	
JitterBlocker™	attenuator, family	
JukeBlox®	platform, technology	
	jukeBlox® registered word mark with a design	Image is not registered; this is a graphic treatment with the registered word mark
KeeLog®	protocol, security IC, technology	
Kleer®	technology	
	Kleer® registered word mark with a design	Image is not registered; this is a graphic treatment with the word mark
Klob-on-Display™ (KLOD™)	device, family, HMI, knob, designer, position wizard, solution, technology, touch controller, user interface, widget, configurator	
LANCheck®	online design review, online review	
Libers®	design file, designer, IDE, SoC design suite, software, tool	US only
LHAMD™	cable diagnostics, diagnostics engine, family, technology	
MarginLink™	signal integrity testing	
maxCrypto™	controller-based encryption, encryption solution	
matStylus®	family, studio, technology, touchscreen controller	
matTouch®	family, studio, technology, touchscreen controller	
maxStore™	driver, storage manager, tool	
Medat B®	bus, controller, device	
megalyS®	device, MCU, microcontroller	
memBrain™	neuromorphic memory product, product, solution, technology	
	Microchip logo	Please do not use this logo; always use the Microchip name and logo.
Note: *Always check with the Legal Department for the most current trademarks.		

Trademark	Description/Appropriate Name	Notes
MPLINK™	linker, object linker	
nGIC™	bare die, diode, gate driver, module, MOSFET, product, solution, technology	
mTouch®	development kit, evaluation kit, library, sensing solution, solution, technology	US only
MultitRAA™	technology	
NetDetach™	technology	
Omniscient Code Generation™ (OCG™)	compilation technology	
OptiLyzr®	network analysis tool, studio, tool	
PIC®	assembler, device, MCU, microcontroller	
	PIC32 logo	
	PIC64 logo	
PICDEM™	demonstration board	
PICDEM.net™	Internet/Ethernet demonstration board	
PICKIT™	debugger, development tool, in-circuit debugger, on-board, programmer, starter kit, tool	
picPower™	technology	
PICSTART®	development programmer	
PICtail™	board, daughter board	
PolarFire®	device, family, FPGA, kit, SoC, System-on-Chip	
PowerSmart™	designer, development suite, digital control library	
Precision Edge®	family, product family	US only
ProASIC®	architecture, board, family, FPGA, kit	US only
ProASIC Plus®	architecture, device, family, FPGA, switch	US only
Prochip Designer®	software suite	
PureSilicon™	oscillator, technology	
QMatrix™	device, sensor	
QTouch®	board, composer, configurator, device, development platform, library, project builder, sensor, solution, technology, tool	
Quiet Wire®	enhanced EM technology, family, technology	
Note: *Always check with the Legal Department for the most current trademarks.		

Trademark	Description/Appropriate Name	Notes
SpecWorld®	ecosystem program, timing	US only
Tachyon®	controller, family, platform, product, technology	
The Embedded Control Solutions Company®		US only; should be in italics
TimeCeum®	primary frequency reference, product	US only
TimeHub™	platform, system	US only
TimePicra®	platform, software suite, synchronization management system	US only
TimeProvider®	clock, enhanced PRTC, iPRTC, extension, system, grandmaster, kit, module, product, series, server, shelf	US only
TimeSource®	PIC, PIC, primary reference clock, primary reference source, shelf, system	
trngAVR®	MCU, microcontroller	
Total Endurance™	software model	
TSHARC™	controller, touchscreen controller	
Turing™	hardware security module, programmer	
Ulio®	bus, communication protocol, firmware, hardware port, memory chip	
USBCheck™	design review	
VarSense™	technology	
VectorBlue™	accelerator, intellectual property, IP, platform, SDK, software development kit, solution, tool, tool chain, tool flow	
Vector®	oscillators, products	
VelocityDriver™	CT, configuration tool, product, solution, software platform, SP	
VerPHY™	cable diagnostics algorithm, cable diagnostics software suite	
ViewSpan™	technology	
WiperLock™	technology	
XMEGA®	Custom Logic (CLC), device, family, MCU, microcontroller, series	
XpressConnect™	retimer	
ZENA™	analyzer, software, wireless adapter	
ZL®		US only
Note: *Always check with the Legal Department for the most current trademarks.		