QEEXO AUTOML USER GUIDE

Jan, 14, 2021 -- v.4.4.2

Overview

This document is intended to provide complete instructions on how to use the AWS version of Qeexo AutoML application to automatically create and deploy machine learning models. It will cover:

- System Requirements
- Running the Qeexo AutoML Web App
- Working with Projects
- Data Management
- Building Machine Learning Models
- Training Results
- Notification Center

System Requirements

Requirements to run the AWS version of Qeexo AutoML are listed below.

Hardware Requirements

- For AWS version, we recommend 1 PC running Windows 10 or 1 Mac machine running macOS.
- Please refer to installation guides for a list of Qeexo's supported hardware.

Software Requirements

- Frontend application (installed on Windows 10 or macOS on the host machine)

Running the Qeexo AutoML Web App

Start a browser and navigate to https://automl.qeexo.com

Working With Projects

The basic unit of organization in the Qeexo AutoML system is a "Project". A Project represents a collection of work to solve a specific machine learning problem on a particular Target Hardware. An example of a Project might be something like "TurbinePredictiveMaintenance", where the data, models, and tests are compiled with the end goal of using machine learning for predictive maintenance on Arduino Nano 33 BLE devices attached to turbines.

For an example "Air Gesture" project, refer to *Detecting Air Gestures with QeexoAutoML*.

Creating and Managing Projects

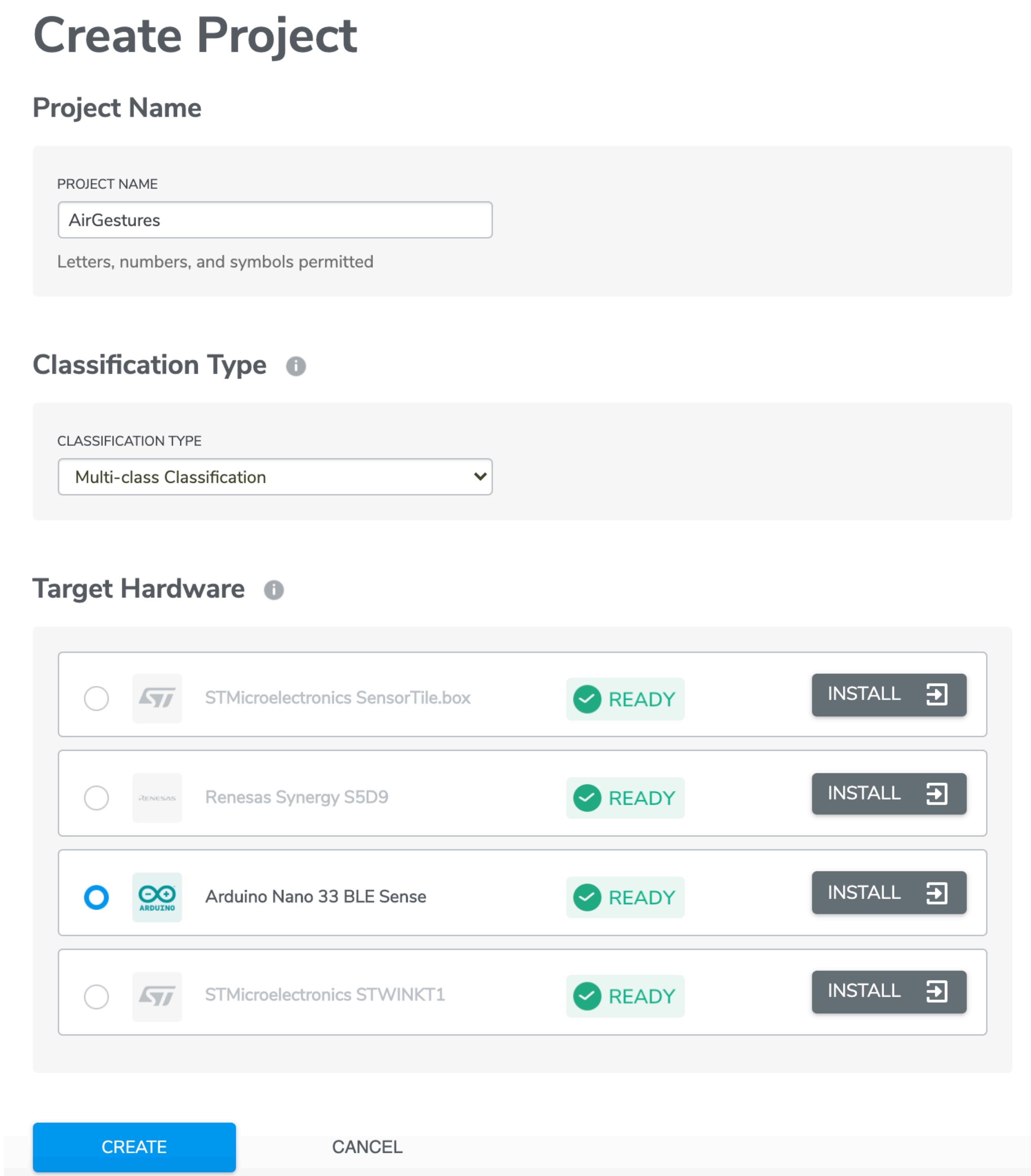
As a new user, you will be taken to the "Create Project" page after logging in, where you can specify a "Project Name", "Classification Type", and the "Target Hardware".

Project Name: Enter a name that is reflective of the purpose of your project.

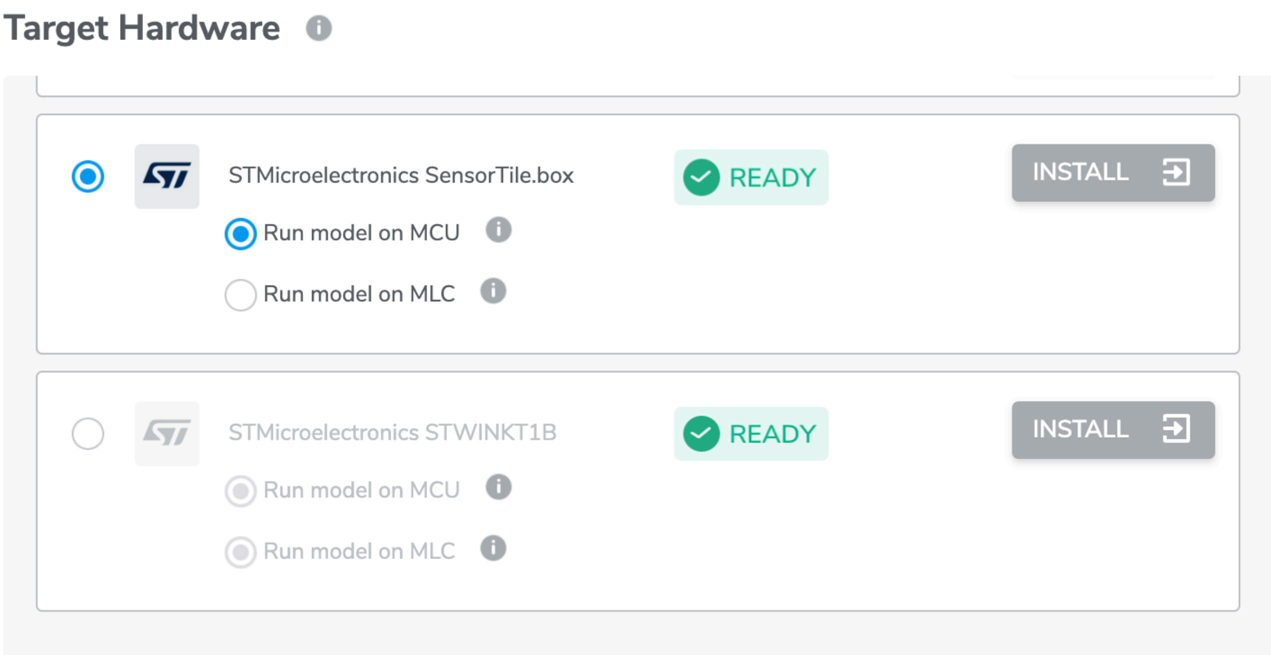
Classification Type: Choose between Single-class, Multi-class or Multi-class Anomaly classification. Single-class is suitable for identifying whether or not a data instance belongs to the given class; multi-class classification can be used for applications distinguishing two or more classes, while Multi-class Anomaly classification is the combination of Single-class and Multi-class classification whereas the model can distinguish the training classes as well as predict 'unknown' if the new datapoint is sufficiently different from any of the training classes.

Note that MLC projects only support multi-class classification.

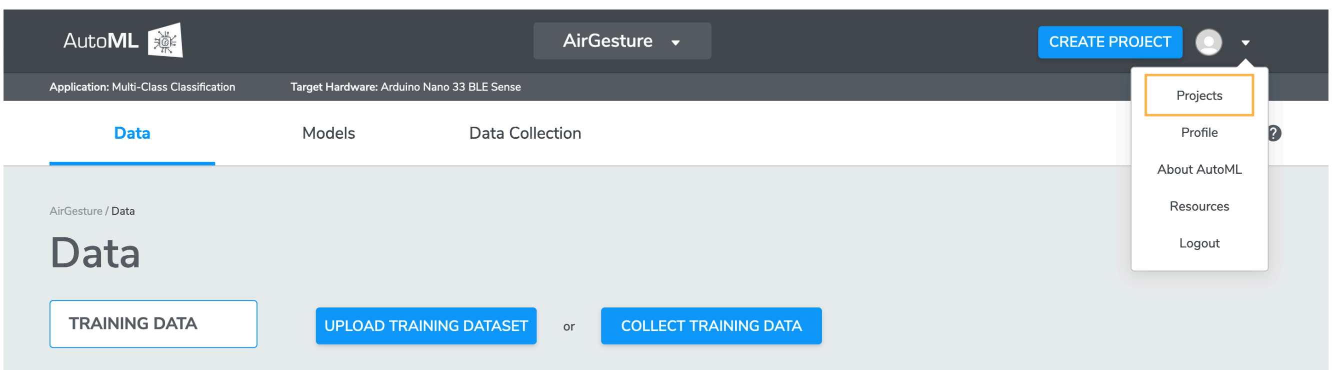
Target Hardware: Select the hardware that will be used in your project. If you do not plan to use any hardware, please select "Arduino Nano 33 BLE Sense".



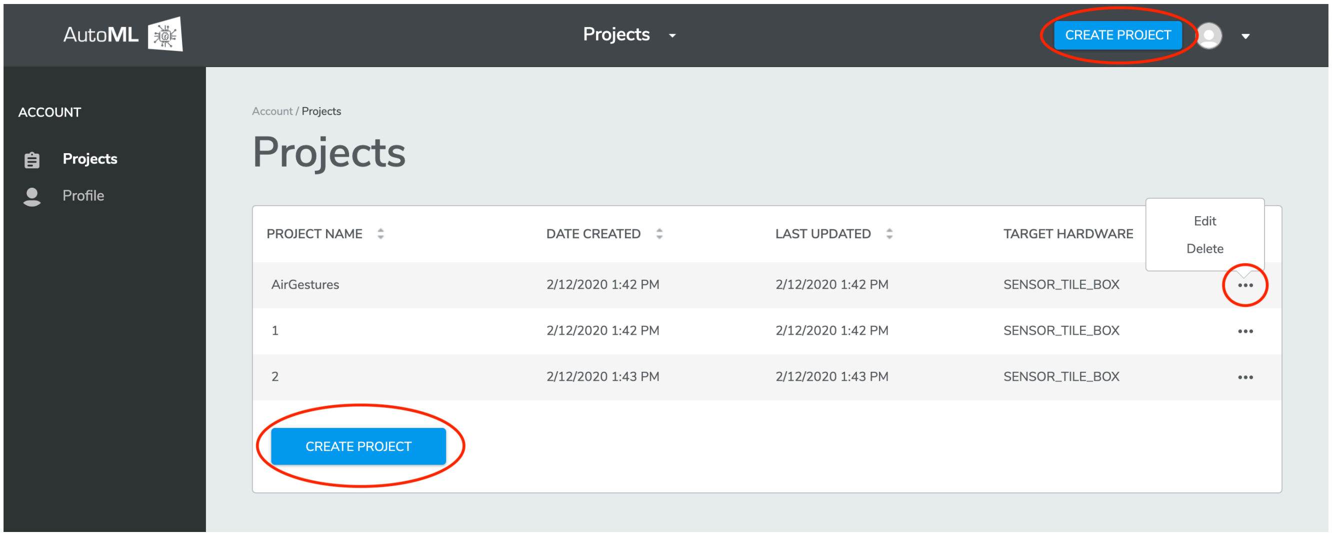
For STWINKT1 and SensorTile.box, there are additional sub-project types: users can select whether the machine learning model runs on the micro-controller (MCU) or the machine learning core (MLC). MLC allows running model on MEM sensor which enable consistent reduction of power consumption.



A list of all Projects created from your account can be found on the Projects page, which can be navigated to from the drop-down menu to the right of the User Profile icon.



Additional Projects can be created from the "Create Project" button either on the Projects page or at the upper right hand corner.



You may delete the Project or edit the Project Name from the ... icon to the right of each Project. You may also switch between projects by selecting from the Projects top center drop-down menu.

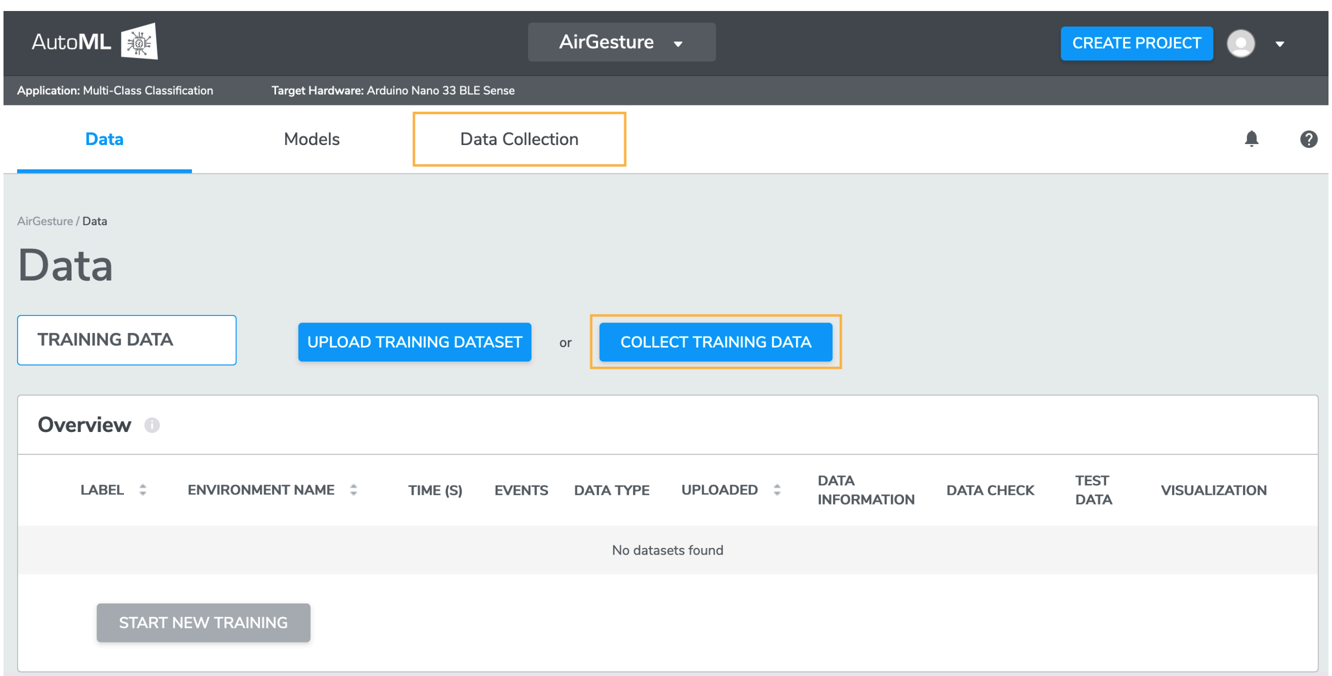
Data Management

After a Project has been created, you will be taken to the Data page, where you can upload or collect data.

We recommend that you first collect data with the Qeexo AutoML web app to ensure that everything is working properly before trying to upload your own data.

Collecting Data

Navigate to the Data Collection page to collect data using the Qeexo AutoML web app. This can be done either by clicking the "Collect Data" button or the "Data Collection" tab.



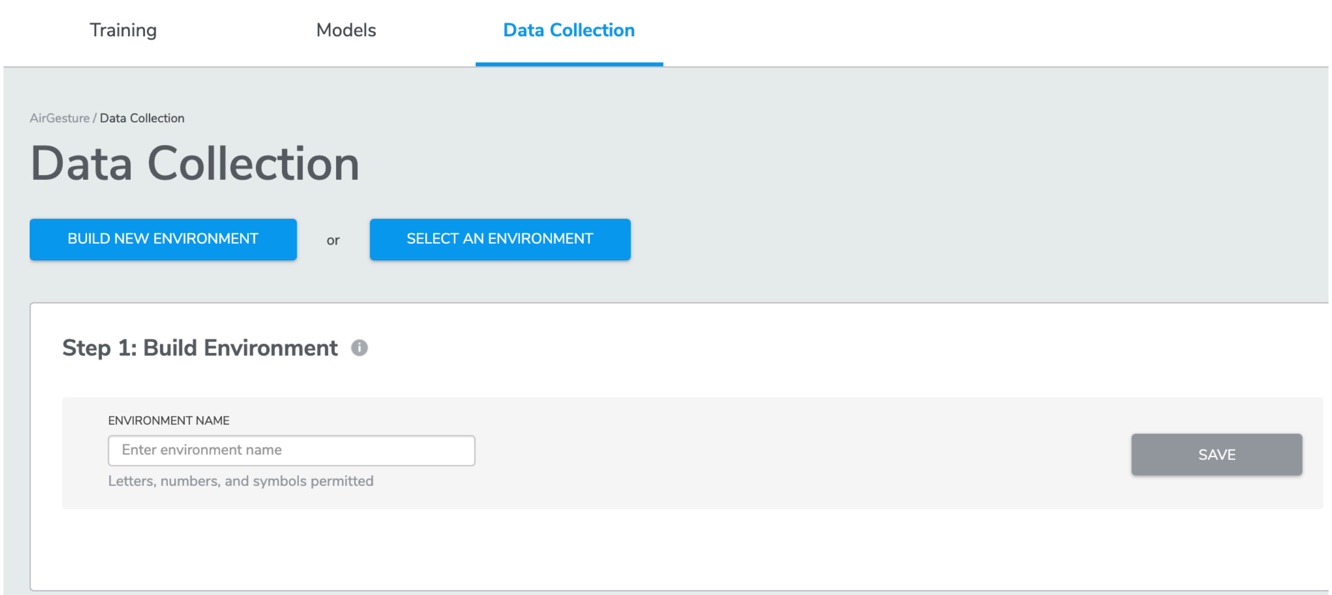
Step 1: Build Environment

An Environment is a physical setting with a given set of properties (e.g. acoustics, temperature, lighting). The range of this set of properties should match the range of the environment where the final machine learning model will eventually run. For example, training the machine learning models with data in your office will likely not work very well once you test the trained models on the factory floor.

Environments also contain information about the given sensor configuration settings. All data collected for a given Environment will have the same sensor configuration.

You can either "Build an Environment" by entering a unique "Environment Name", or "Select an Environment" to add more data to a previously recorded Environment. If selecting an existing Environment, the Sensor Configuration (in Step 2) will automatically populate with the Environment's previous settings.

You should name your Environment something easily recognizable to you, with details about the specific location. For example, "OfficeCoffeeTable" or "VestasTurbineSolano".

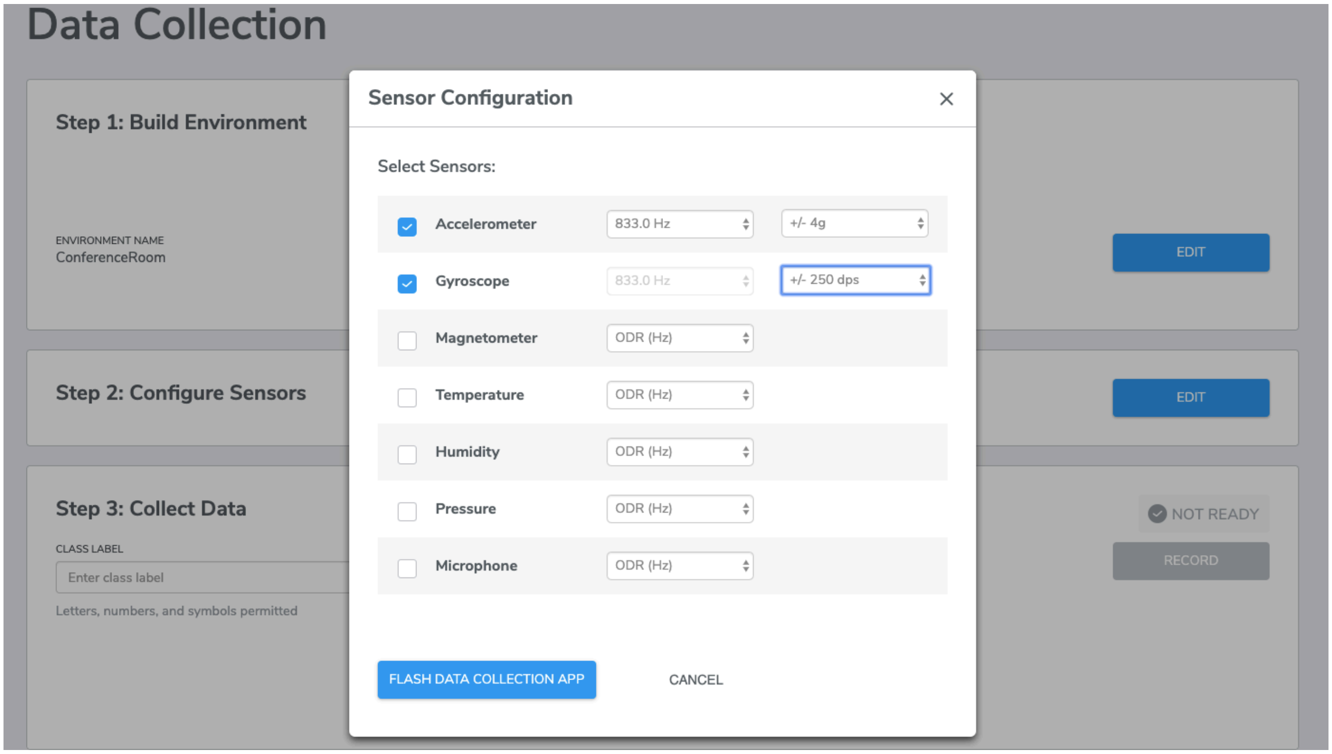


Step 2: Configure Sensors

Click "Edit" in Step 2 to view a list of the supported sensors on the Target Hardware. You may select any combination of the sensors listed on this menu to collect your data. After selecting the sensors, you will need to configure the corresponding sampling rate (ODR, or Output Data Rate) for each sensor, and the full scale range (FSR) when available.

The optimal ODR and FSR configuration depends on the use case. For example, using an ODR of over 1000 Hz will generally give some useful detail when detecting and analyzing turbine vibrations, while ODR of ~100 Hz or less may be more suitable for human activity detection. Having a larger FSR will allow you to see larger variations in sensor values, but the resolution will be compromised, so you will get less detail.

Finding the optimal settings may take some trial-and-error. Qeexo AutoML's data visualization support (see section on Visualizing Data) may help in finding these settings. For example, if the peaks of your signal appear to be cut off, or saturating, you may want to increase the FSR of your sensor.



Currently, a few limitations exist:

- Accelerometer and Gyroscope must share the same ODR.

After selecting the desired sensors and settings, click "Flash Data Collection App" to flash the data collection application to the Target Hardware.

Note: If this is your first data collection for a given sensor configuration, Qeexo AutoML needs to build the data collection application first. This can take a few minutes, but it will only have to be done once for each new configuration.

If you are selecting MLC project type, note that only accelerometer and gyroscope are available.

Step 3: Collect Data

Qeexo AutoML currently supports a variety of supervised classification algorithms for machine learning. For each of these algorithms, all data used for training must have an associated Class Label.

For multi-class and multi-class anomaly case, at least two unique classes must be defined. For most problems, we recommend that at least one of the classes be a "baseline" class that represents the typical environmental noise or behavior.

Whether or not baseline data is necessary depends on the use case and data selected. In general, the classes collected for multi-class classification should represent the full set of possible states for the given Environment. For example, if you want to build a multi-class model which can distinguish between various types of machine vibrations (e.g. slow, medium, fast), you should collect data which represents all possible different types of machine vibrations.

In this case, if the model output will also be used in the "baseline" case where the machine is off, this data should be collected as well.

Baseline Data:

- Baseline data can be collected by setting the data type to Continuous, and leaving data collection application to run while the environment is in a steady state of rest or typical operating behavior.
- Some machine learning problems require collecting baseline data to differentiate events of interest from normal environmental conditions.
- Baseline data is usually associated with each Environment (since different Environments will often have different baseline data characteristics).
- For example, baseline data might be "NoGesture" in gesture recognition, "None" in kitchen appliance detection, or "AtRest" in logistics monitoring.

Data Collection Type

Qeexo AutoML currently supports continuous data collection and data segmentation:

- Continuous

    - Continuous collections will collect data for n consecutive seconds, where n is the Number of Seconds defined.
    - Continuous type data collection should be used to collect data over a fixed time interval where the Class Label does not change.
    - For example, Continuous data can be "Normal" in Predictive Maintenance, "Running" in activity recognition, or "Occupied" in occupancy detection.
    - Note that MLC projects support Continuous data type.

- Data Segmentation

    - Data segmentation is provided as method to quickly crop and label events or regions of interest within a continuous dataset.
    - For example, an Event can be segmented (cropped and labeled) as "Knock" in surface gesture recognition, a "Fall" in human fall detection, or an "Impact" in logistics monitoring.
    - Note that data segmentation is not supported in MLC projects.

Class Label

A Class Label is a machine learning concept, normally a word or phrase to label the *event* or *condition* of interest. For example, "Normal", "WornBearings", and "WindingFailure" can be classes in our Turbine Predictive Maintenance problem.

For non-segmented data, the Class Label applies to all of the data collected. For segmented data, the Class Label applies only to the cropped and labeled region.

You must define one Class Label at a time when collecting data by entering a text string in the given field.

Note that only alphabets, numbers, and underscores are allowed when naming class labels in MLC projects.

Number of Seconds

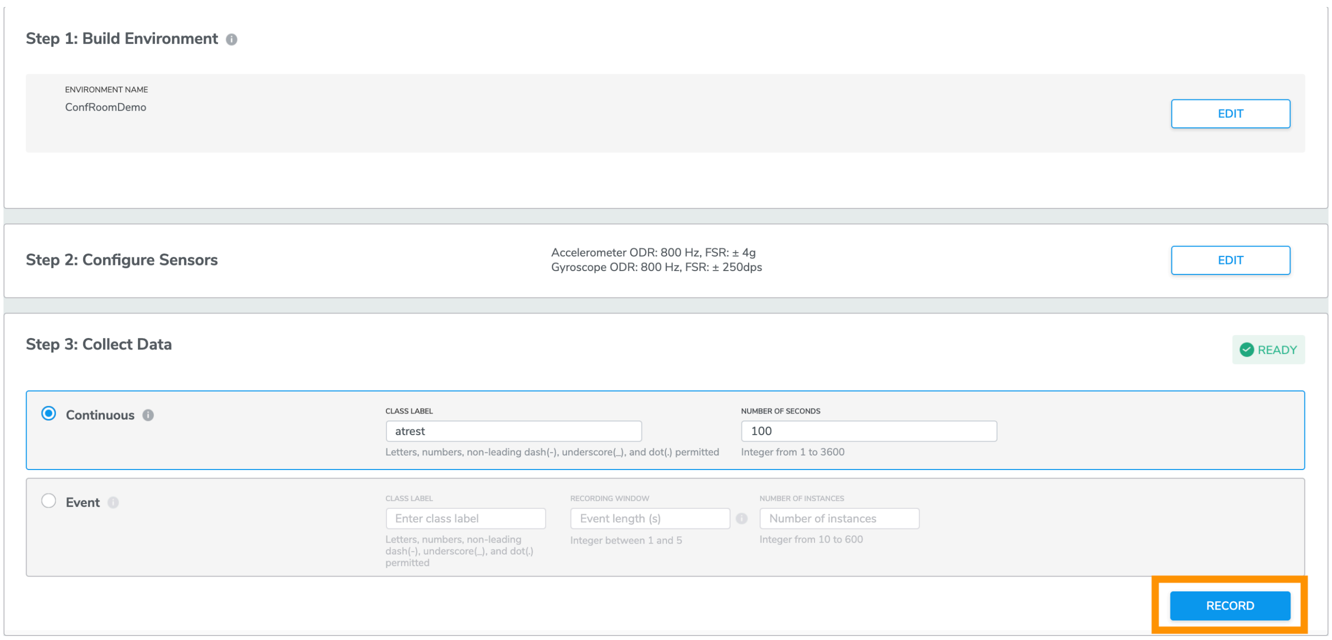
This sets the duration of the data collection.

For data collection, a continuous stream of data will be recorded from the Data Recording page, the recording will stop after n seconds, where n is the number of seconds you entered.

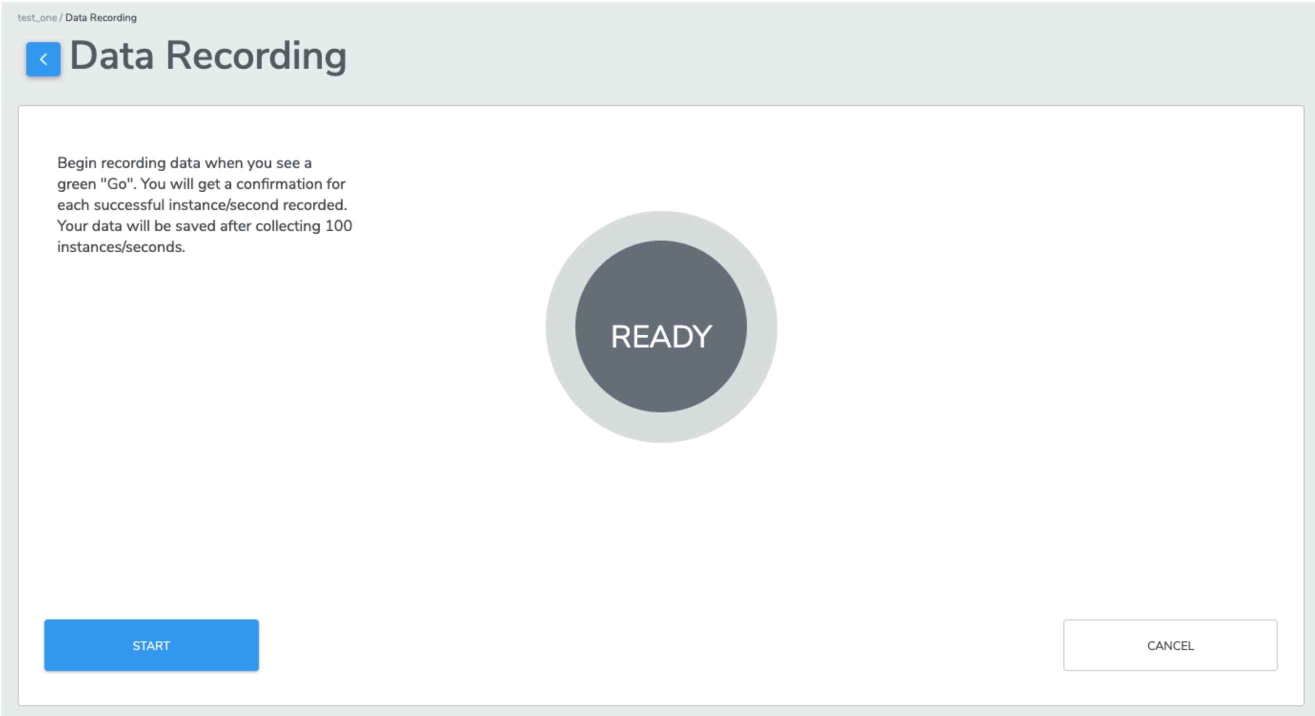
More data generally leads to higher performance. Depending on the complexity of the use case, the number of classes, the quality of the data, and many other factors, the optimal and minimum number of instances or seconds to collect can vary greatly. We recommend starting with at least 30 Seconds for each Class Label, but much more data may be required if the classes are highly variable or if the problem is sufficiently complex.

Recording Data

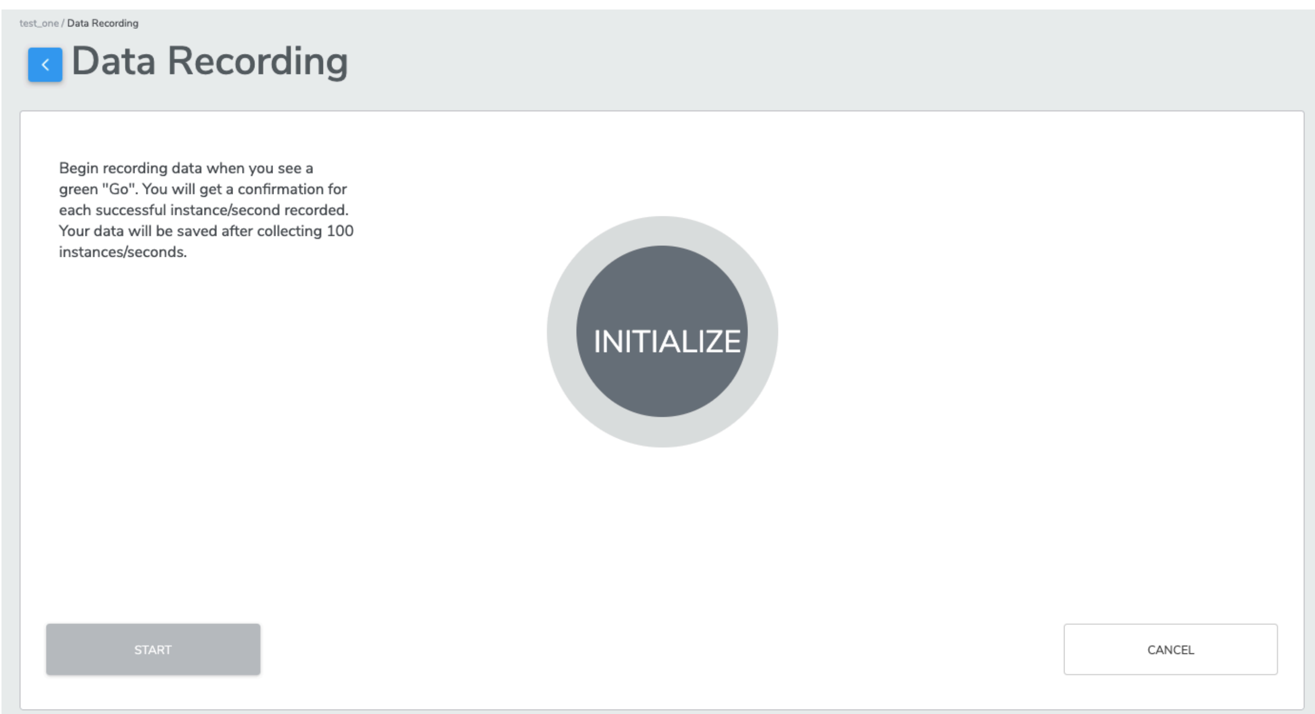
After completing the previous steps, the "Record" button should now become click-able. (If it is not, check previous steps.)



After clicking "Record", you will be directed to the Data Recording page:

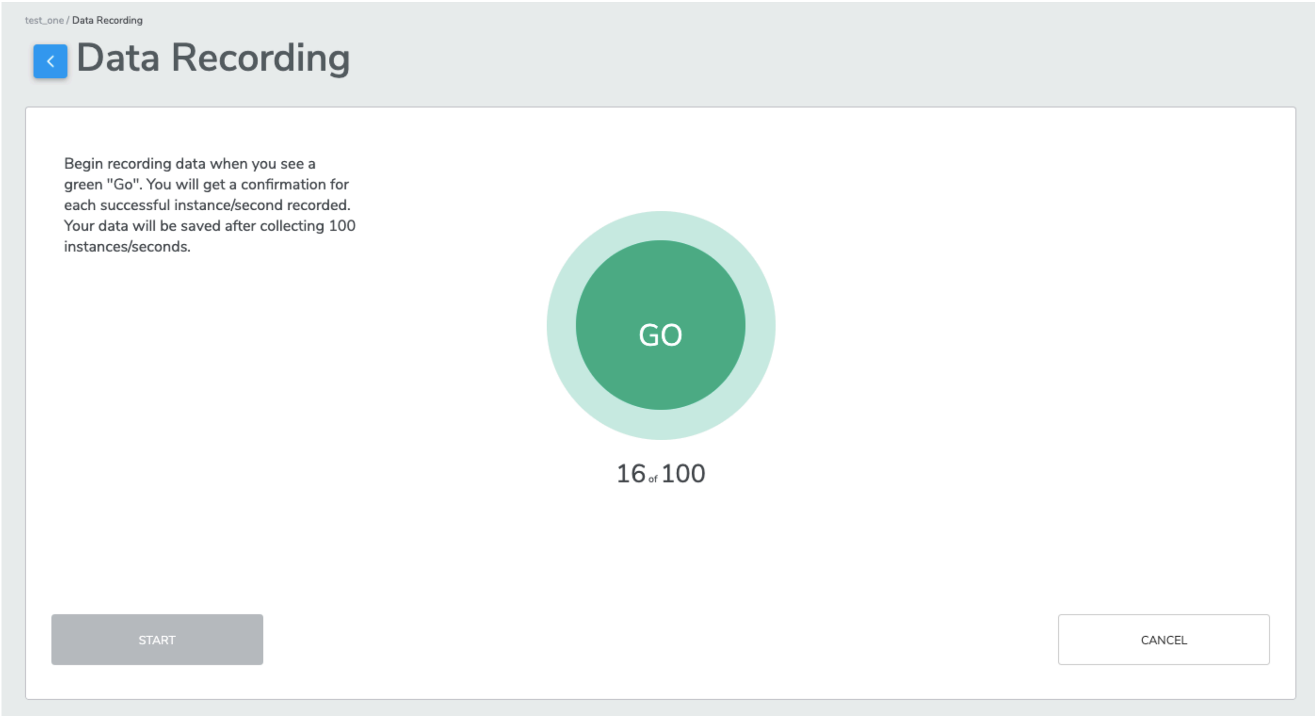


When you are ready to start data collection, click "Start" to begin. The text in the center circle will change from "Ready" to "Initialize" while the data collection software is starting up.

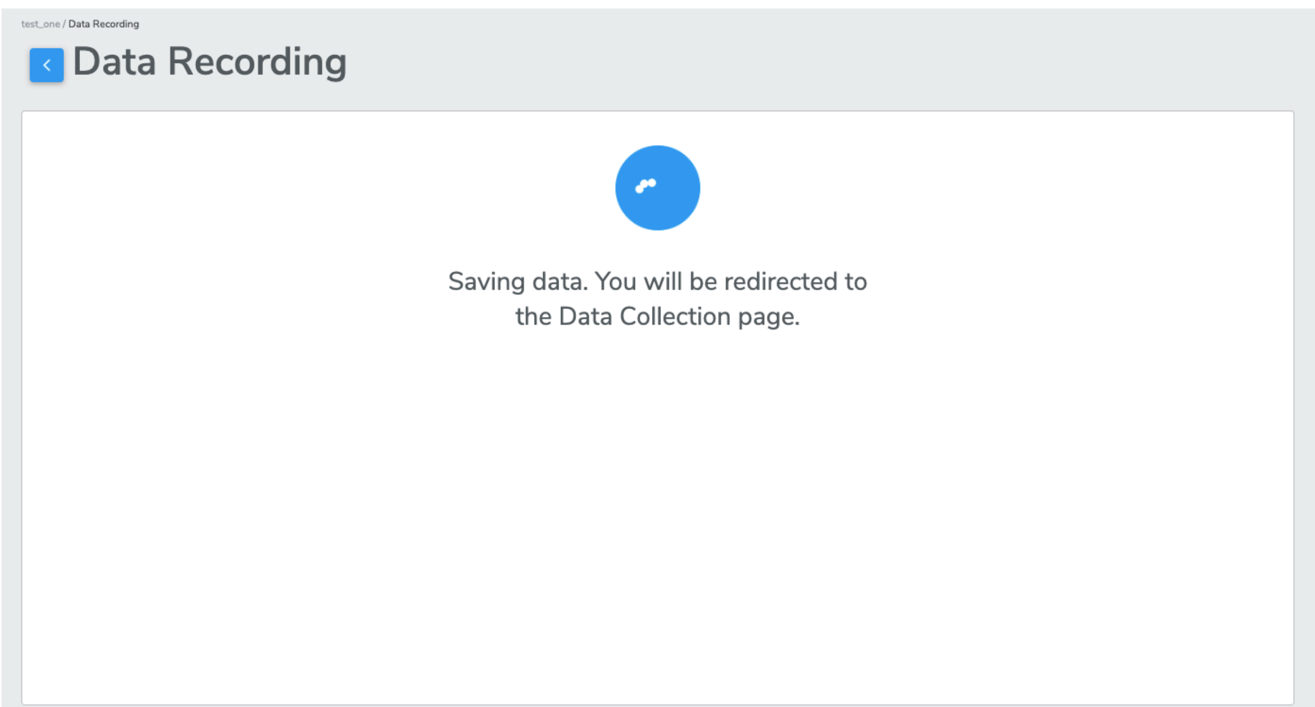


After a few seconds, data collection will start when you see the circle turn green and display "Go". Data is now being collected.

During data collection, the counter will continuously count up by 1 every second until it reaches the desired Instances/Seconds supplied.



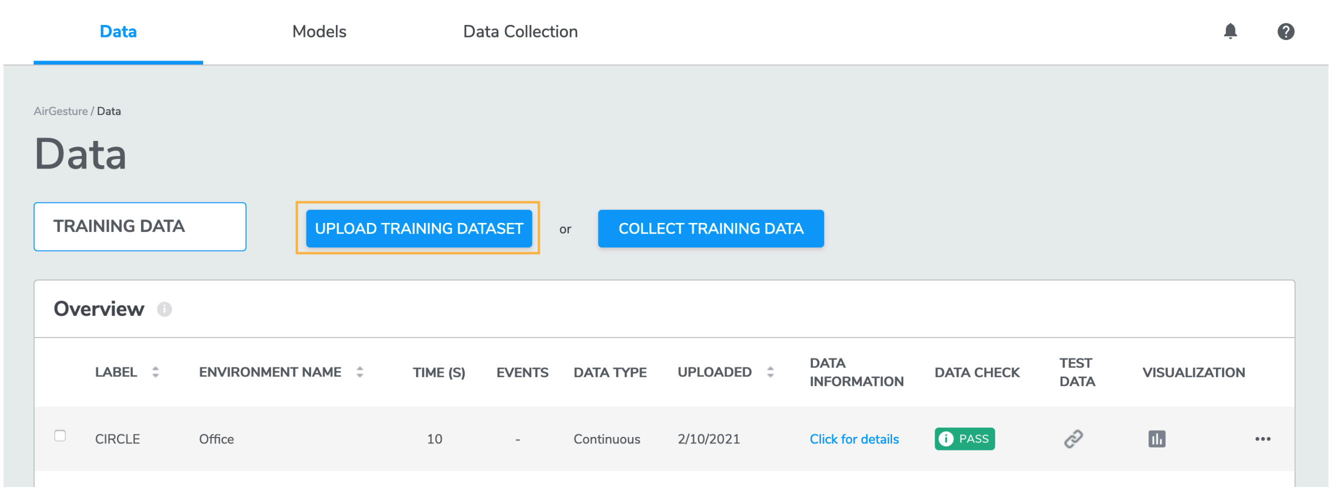
Once the specified number of Seconds have been collected, the labelled data will be uploaded to the database, and user will be redirected to the Data Collection page.



You can collect more data of the same or different Class Label from the Data Collection page. Note that, for a multi-class classification Project, you will need at least 2 distinct classes (2 different Class Labels) to be able to train a machine learning model.

Uploading Dataset

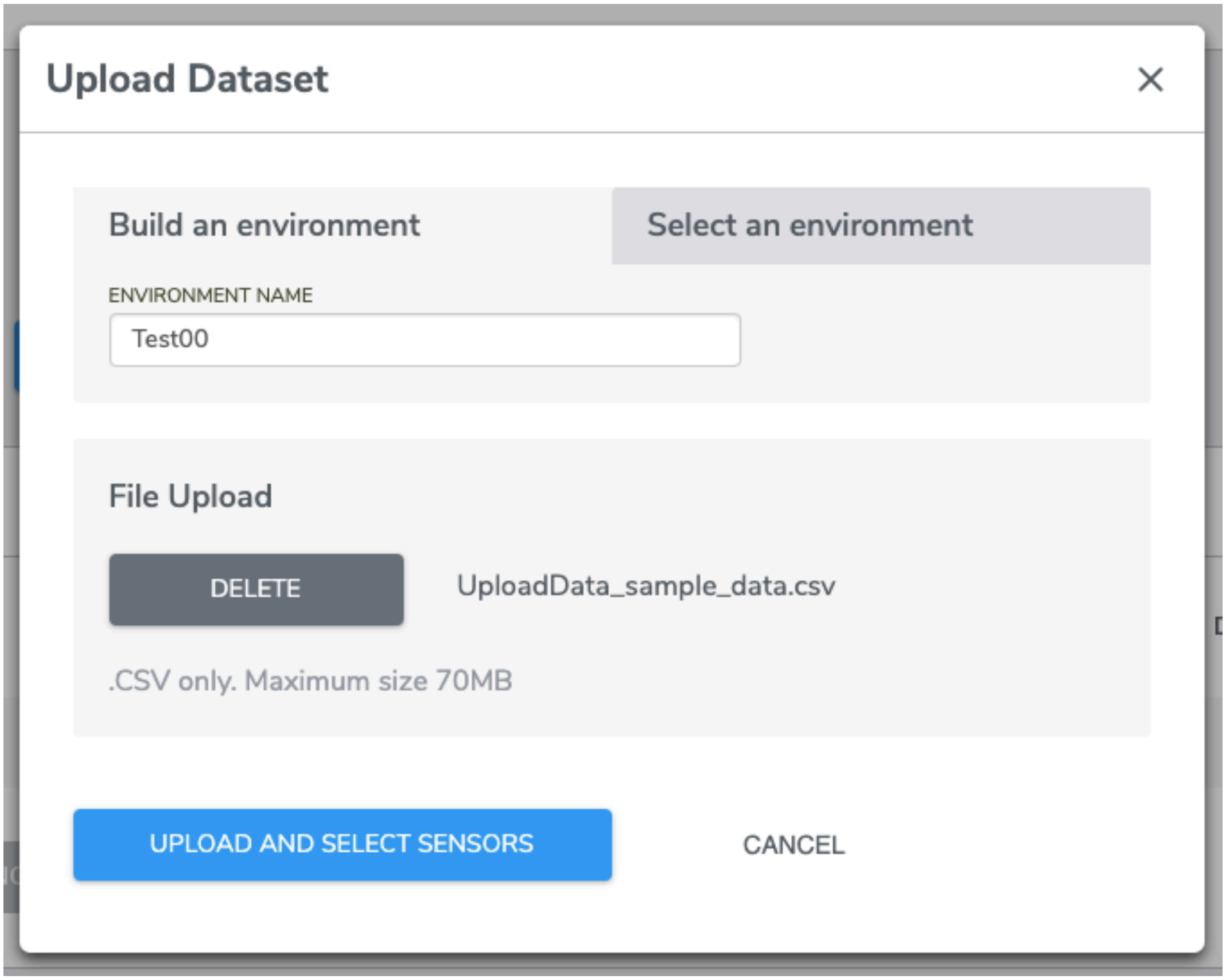
From the Data page, you may upload previously-collected datasets to AutoML directly. These uploaded datasets can be used to train machine learning models, and can be combined with additional data that has been collected through the Qeexo AutoML platform.



Click "Upload Dataset" to upload a single .csv file. Each .csv should contain one or more data collections following the Qeexo-defined data format below. All data contained in the .csv file must come from the same sensor configuration, which you will enter after uploading the .csv file. If you have more than 70 MB of data, you will need to split it into multiple .csv files.

Refer to the file *sample datasets* if you would like some example data for upload.

All rows with the same class label will be treated as a single, continuous collection. AutoML's upload data functionality will produce the best results if each file contains data from only a single collection period on one device. **If there were gaps in the collection period, or if the data was taken from multiple devices, it is recommended to split the separate collections into multiple files for upload.**



Data Format Specification

Qeexo AutoML can accept two different CSV formats:

- V1 is easier to understand and prepare.
- V2 can handle multiple data points in a single line; no upsampling is required thus minimizing the chance of data duplication.

Note that V1 and V2 are for human readability - there is no need to indicate which format because Qeexo AutoML can detect the formatting automatically.

Qeexo AutoML will match the closest ODR of the selected hardware against that of the uploaded data.

V1 CSV format

The data file consists of 3 parts, in the following order: `timestamp`, "sensor data", and `label`.

1. `timestamp` (exactly 1 column):

    - Type: **float** (milliseconds)
    - We up-sample lower-sampling-rate sensor data to match maximum sampling rate. Timestamps should

indicate the maximum sampling rate.
  - Incorrect timestamps may cause data check failure (see Data Check section below).

2. sensor data (1 or more columns from list):
   - Type: **integer**
   - sensor data column names must match the exact strings that Qeexo uses. For example, `accel_x` instead of `accelerometer_x` (see below).
   - Column names not matching pre-defined strings may cause data upload to fail.
   - Refer to the following table for sensors are supported in the hardware you chose:

Sensor	CSV column headers	Arduino Nano 33 BLE Sense	Arduino Nano 33 IoT	RA6M3	SensorTile.box	STWINKT1
Accelerometer	`accel_x` , `accel_y` , `accel_z`	X	X	X	X (MCU, MLC)	X
Gyroscope	`gyro_x` , `gyro_y` , `gyro_z`	X	X	X	X (MCU, MLC)	X
Magnometer	`magno_x` , `magno_y` , `magno_z`				X (MCU)	X
Temperature	`temperature`	X			X (MCU)	X
Humidity	`humidity`	X		X	X (MCU)	X
Pressure	`pressure`	X			X (MCU)	X
Microphone	`microphone`	X		X	X (MCU)	X
Analog microphone	`microphone_analog`					X [1]
Light (single channel)	`light`			X		
Ambient light (CRGB)	`ambient_c` , `ambient_r` , `ambient_g` , `ambient_b` ,	X				
RCDA: Clean Dry Air Resistance	`rcda`			X		
ETOH : Ethanol	`etoh`			X		
TVOC : Total volatile organic	`tvoc`			X		

compounds						
IAQ : Indoor air quality	`iaq`			X		
ECO2 : Estimated Carbon Dioxide	`eco2`			X		
RMOX : Metal Oxide Resistance	`rmox`			X		
Low power accelerometer	`accel_lowpower_x` , `accel_lowpower_y` , `accel_lowpower_z`					X
High sensitivity accelerometer	`accel_highsensitive_x` , `accel_highsensitive_y` , `accel_highsensitive_z`					X

1. `label` (exactly 1 column):
    - Type: **string**
    - Label column contains the verified Class Label (decision) for each row of sensor data.
    - We recommend that each row in `.csv` file is sorted with timestamp and grouped by class label.
2. `data_type` :
    - Type: string ("CONTI" or "EVENT")
    - "CONTI" for continuous data, "EVENT" for event data
    - **Ignore this column while importing data **
3. `recording_id` :
    - Type: int (empty or 1,2,3,...)
    - Empty if data is continuous data or not belong to any recording
    - **Ignore this column while importing data **
4. `event_id` :
    - Type: int (empty or 1,2,3,...)
    - Empty if data is continuous data or not belong to any event
    - **Ignore this column while importing data **

V2 CSV format

The format for uploaded data consists of 3 parts, in the following order: a `timestamp` column, sensor data column(s), and a `label` column.

1. `timestamp` (exactly 1 column):

    - Type: **integer** (milliseconds)
    - Timestamp column contains the time associated with the sample(s) in a given row.
    - Note that each line increases by 50 ms. If the data is sampled at 100Hz, then the number of samples in each line should be 50/(1000/100) or 5 samples.
    - Timestamp column should be in **ascending sorted order**.
    - In the case of a row containing multiple samples, the timestamp should be the time associated with the most recent sample in the row (i.e. the time at the *end* of the given sampling period).
    - Incorrect timestamps may cause a data check warning (see Data Check section below).

2. Sensor data (1 or more columns):

    - Type: **list of integers** (or list of lists of integers)
    - Each sensor data column must represent all channels of a supported sensor type on one of the AutoML-supported hardware platforms. See below table for a list of currently-supported column names.
    - Each cell in the sensor data column contains all of the sensor's samples associated with the given row timestamp. A cell can contain no samples (e.g. ",[],"), a single sample (e.g. ",[100],"), or multiple samples bracketed in a list (e.g. ",[100,99,101],").
    - For sensors with multiple channels, the format is a list of lists of integers, grouped by time. The expected channel ordering for each inner list is x, y, z (for accel, gyro, magno) and c, r, g, b (for ambient light sensor).
    - Incorrect column names will cause data upload to fail.
    - Incorrect sensor configurations will cause a data check warning (see Data Check section below).

Sensor	CSV column header	Arduino Nano 33 BLE Sense	Arduino Nano 33 IoT	RA6M3	SensorTile.box	STWINKT1
Accelerometer (XYZ)	`accel`	X	X	X	X (MCU, MLC)	X
Gyroscope (XYZ)	`gyro`	X	X	X	X (MCU, MLC)	X
Magnometer (XYZ)	`magno`				X (MCU)	X
Temperature	`temperature`	X			X (MCU)	X
Humidity	`humidity`	X		X	X (MCU)	X
Pressure	`pressure`	X			X (MCU)	X
Microphone	`microphone`	X		X	X (MCU)	X
Analog						

microphone	`microphone_analog`					X [^1]
Light (single channel)	`light`			X		
Ambient light (RGBC)	`ambient`	X				
RCDA: Clean Dry Air Resistance	`rcda`			X		
ETOH : Ethanol	`etoh`			X		
TVOC : Total volatile organic compounds	`tvoc`			X		
IAQ : Indoor air quality	`iaq`			X		
ECO2 : Estimated Carbon Dioxide	`eco2`			X		
RMOX : Metal Oxide Resistance	`rmox`			X		
Low power accelerometer	`accel_lowpower`					X
High sensitivity accelerometer	`accel_highsensitive`					X

1. `label` (exactly 1 column):

    - Type: **string**
    - Label column contains the class label for each row of sensor data.

2. `data_type` :

    - Type: string ("CONTI" or "EVENT")
    - "CONTI" for continuous data, "EVENT" for event data
    - **Ignore this column while importing data **

3. `recording_id` :

    - Type: int (empty or 1,2,3,...)
    - Empty if data is continuous data or not belong to any recording
    - **Ignore this column while importing data **

4. `event_id` :

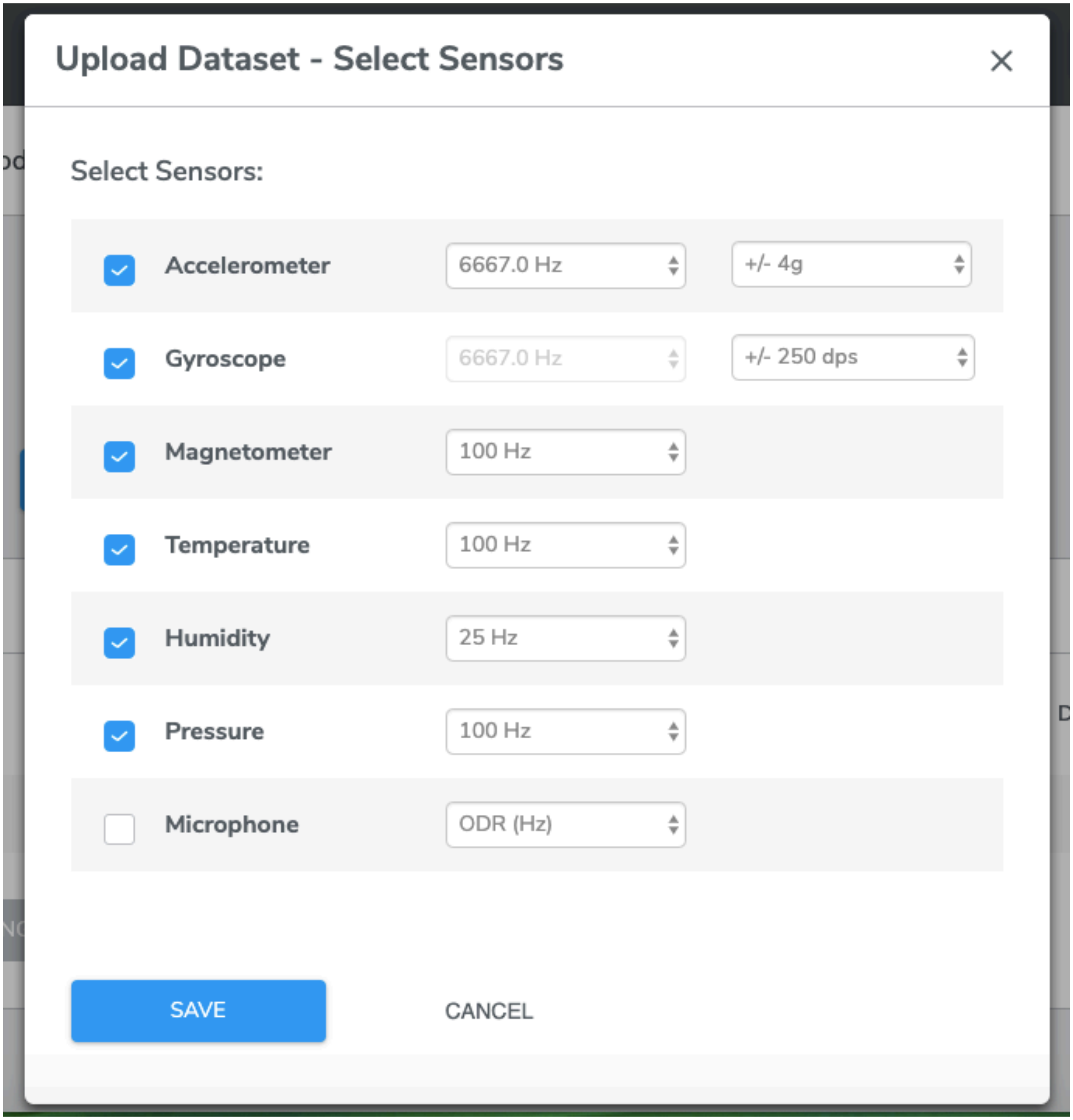
    - Type: int (empty or 1,2,3,...)
    - Empty if data is continuous data or not belong to any event
    - **Ignore this column while importing data **

	A	B	C	D	E	F	G	H	I
1	timestamp	accel	gyro	magno	humidity	pressure	ambient	microphone	label
2	70940	[[-171, 1275, -3833], [-17	[[187, -29, 115], [170, -2	[[2989, -954, -3815], [29	[34]	[989, 989, 989, 989, 989]	[[12, 5, 5, 4], [12	[-41, -40, -39, -39,	NOISE
3	70990	[[-158, 1270, -3837], [-17	[[215, -34, 108], [216, -2	[[2972, -938, -3699], [29	[34]	[989, 989, 989, 989, 989]	[[12, 5, 5, 4], [12	[-17, -20, -23, -21,	NOISE
4	71040	[[-169, 1273, -3823], [-18	[[198, -32, 115], [202, -4	[[2965, -945, -3720], [29	[34, 34]	[989, 989, 989, 989, 989]	[[12, 5, 4, 4], [12	[-27, -26, -27, -24,	NOISE
5	71090	[[-153, 1256, -3806], [-17	[[204, -22, 113], [214, -2	[[2983, -959, -3786], [29	[34]	[989, 989, 989, 989, 989]	[[12, 5, 4, 4], [13	[31, 32, 38, 38, 27	NOISE
6	71140	[[-163, 1264, -3819], [-18	[[207, -32, 114], [193, -2	[[2981, -965, -3814], [29	[34]	[989, 989, 989, 989, 989]	[[13, 6, 5, 4], [13	[18, 21, 20, 18, 21	NOISE
7	71190	[[-160, 1261, -3801], [-16	[[183, -31, 116], [189, -2	[[2980, -962, -3830], [29	[34]	[989, 989, 989, 989, 989]	[[13, 6, 5, 4], [14	[-17, -11, -10, -14,	NOISE
8	71240	[[-173, 1273, -3815], [-17	[[186, -26, 111], [187, -2	[[2984, -948, -3769], [29	[34, 34]	[989, 989, 989, 989, 989]	[[14, 6, 5, 4], [14	[-14, -19, -18, -26,	NOISE
9	71290	[[-186, 1255, -3843], [-16	[[203, -26, 107], [210, -3	[[2988, -945, -3757], [29	[34]	[989, 989, 989, 989, 989]	[[14, 6, 5, 4], [14	[-5, -6, -4, -7, -2, -	NOISE
10	71340	[[-177, 1268, -3814], [-18	[[198, -22, 111], [204, -2	[[2976, -951, -3768], [29	[34]	[989, 989, 989, 989, 989]	[[14, 6, 5, 5], [14	[-16, -16, -17, -16,	NOISE
11	71390	[[-170, 1257, -3818], [-17	[[177, -37, 125], [176, -4	[[2982, -954, -3791], [29	[34]	[989, 989, 989, 989, 989]	[[14, 6, 5, 5], [15	[-14, -14, -16, -14,	NOISE
12	71440	[[-177, 1262, -3827], [-16	[[175, -22, 115], [179, -1	[[2986, -969, -3793], [29	[34, 34]	[989, 989, 989, 989, 989]	[[15, 7, 6, 5], [15	[15, 24, 19, 22, 27	NOISE
13	71490	[[-185, 1276, -3838], [-17	[[211, -20, 113], [223, -2	[[2982, -957, -3774], [29	[34]	[989, 989, 989, 989, 989]	[[15, 7, 6, 5], [15	[-1, -6, -3, -5, 4, -2	NOISE
14	71540	[[-168, 1251, -3809], [-18	[[189, -26, 107], [192, -3	[[2984, -943, -3767], [29	[34]	[989, 989, 989, 989, 989]	[[15, 7, 5, 5], [15	[-15, -15, -14, -14,	NOISE
15	71590	[[-187, 1286, -3828], [-16	[[193, -28, 126], [189, -3	[[2989, -965, -3799], [29	[34]	[989, 989, 989, 989, 989]	[[15, 7, 5, 5], [13	[13, 14, 13, 1, 15,	NOISE
16	71640	[[-175, 1254, -3819], [-18	[[243, -35, 117], [239, -2	[[2986, -960, -3773], [29	[34, 34]	[989, 989, 989, 989, 989]	[[13, 6, 5, 4], [13	[-37, -34, -27, -34,	NOISE
17	71690	[[-167, 1267, -3800], [-16	[[159, -34, 113], [160, -3	[[2991, -974, -3794], [29	[34]	[989, 989, 989, 989, 989]	[[13, 6, 5, 4], [11	[-1, 2, -6, -1, -4, -2	NOISE
18	71740	[[-172, 1267, -3827], [-17	[[220, -39, 119], [213, -3	[[2980, -965, -3767], [29	[34]	[989, 989, 989, 989, 989]	[[11, 5, 4, 4], [11	[-9, -12, -12, -13, -	NOISE
19	71790	[[-170, 1254, -3803], [-17	[[185, -37, 119], [187, -4	[[2995, -954, -3798], [29	[34]	[989, 989, 989, 989, 989]	[[11, 5, 4, 4], [11	[-26, -33, -42, -28,	NOISE

As shown in the image above, all sensor samples are formatted as lists of integers or lists of lists of integers. For the lowest ODR sensor (humidity), there were only 1-2 samples per sampling period (50 ms for this collection), while the highest ODR sensor (microphone) has hundreds of samples per row. For the multi-channel sensors, the x, y, z and c, r, g, b channels are grouped together in-time and recorded inside their own lists.

Upload Data and Confirm Sensor Configurations

Qeexo AutoML will detect the most appropriate sensor configurations for your uploaded `.csv` file. You will have the opportunity to accept or select a different values of sensor *type*, *ODR*, and *FSR*. We recommend accepting the auto-detected values. **Incorrect sensor configurations may break the library-building process or generate sub-optimal results.**



If your data is collected at an ODR that is too far off from the supported ODR of the selected hardware, you may need to consider up-sampling or down-sampling the data before uploading them to Qeexo AutoML.

Data Check

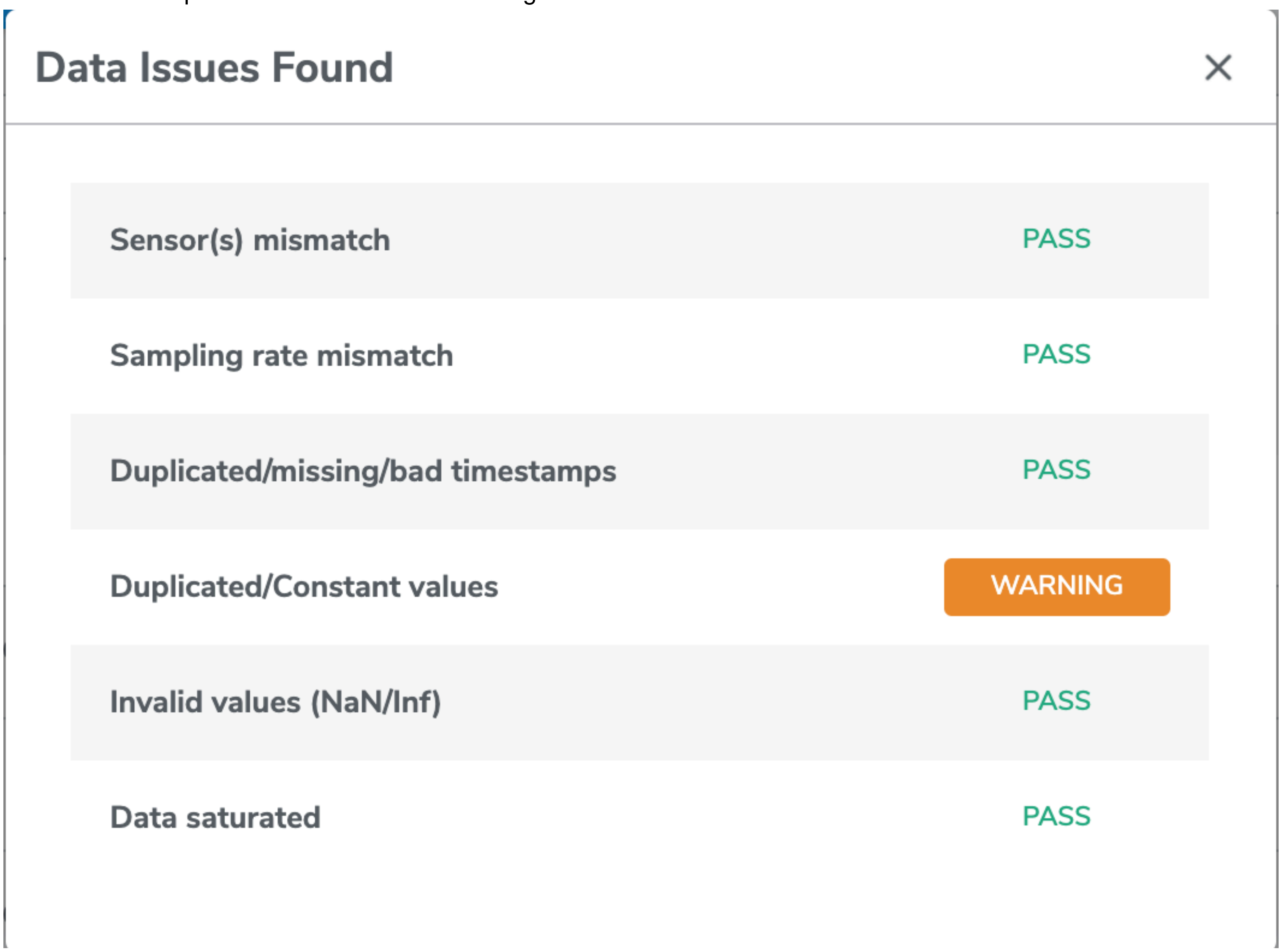
Data check verifies the quality of the data, whether uploaded or collected. A failure in data check will not prevent you from using the data to train machine learning models. However, poor data quality may result in poor model performance.

Qeexo AutoML currently looks for the following data issues:

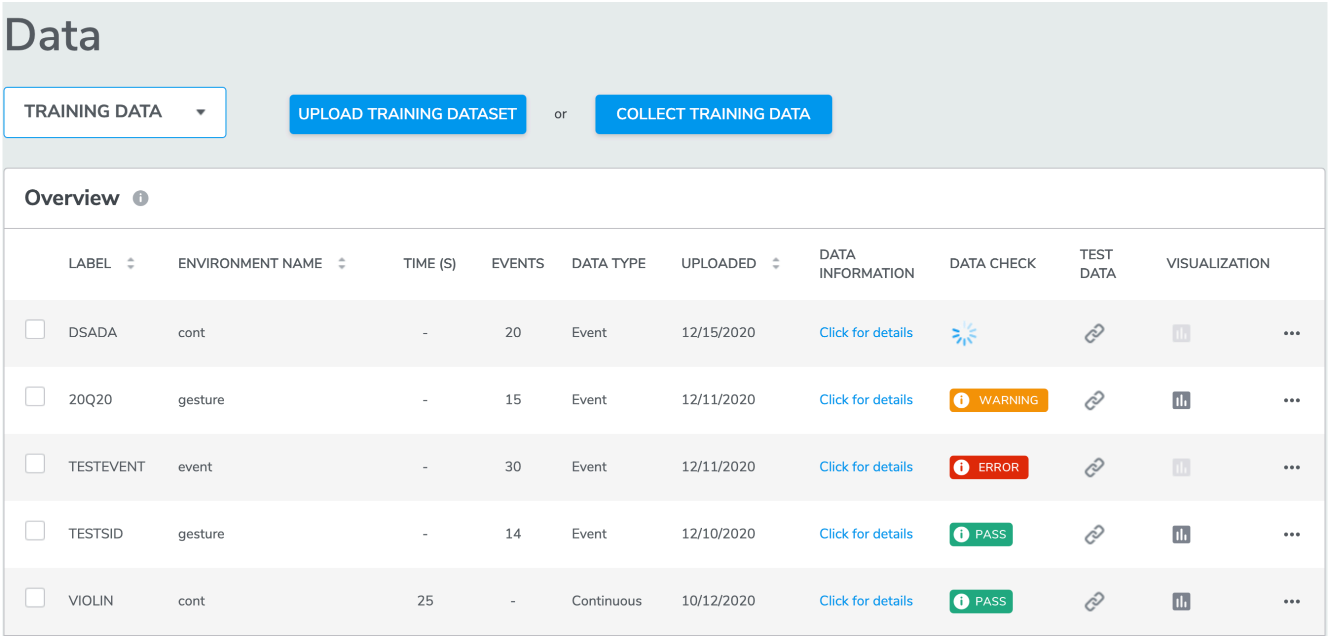
- collected data does not match the selected sensors in the Sensor Configuration step

- collected data does not match the selected sampling rate in the Sensor Configuration step
- collected data contain duplicate or missing timestamps
- collected data has duplicate values or constant values
- collected data contains invalid values including NaN or inf
- collected data is saturated

Here is an example of a data check with warnings:



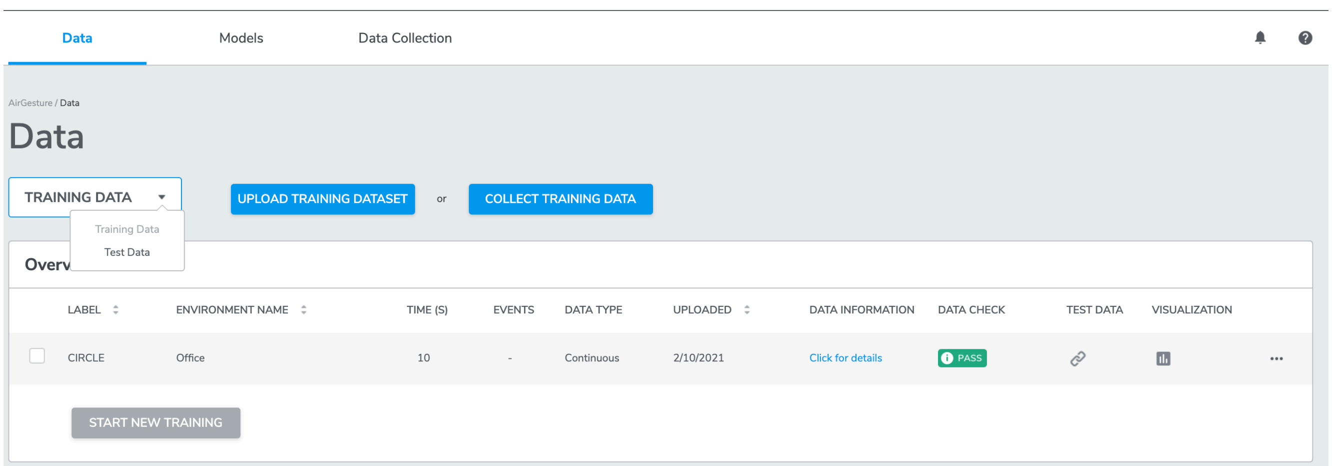
A green PASS icon indicates that data check has passed; a yellow WARNING icon indicates that the data contains one or more issues from the list above; a red ERROR icon indicates that something went wrong during data collection or during data check (connection error or device error), the data may not be usable if it remains ERROR after refresh.



Training data and Test data

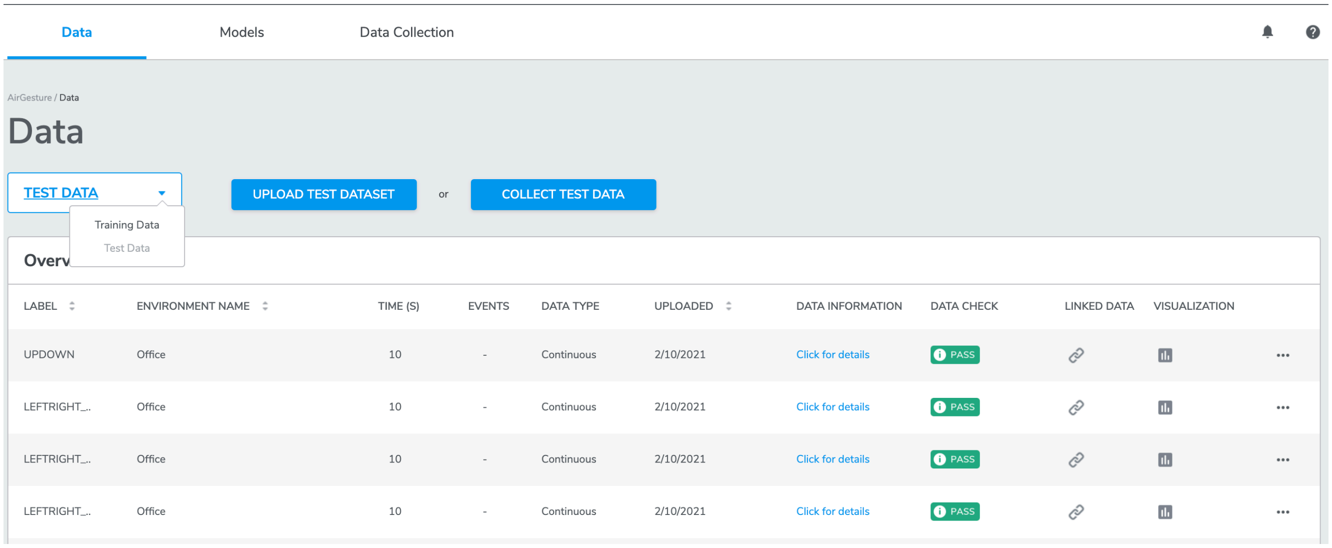
Training data

A set of data instances you plan to train and build models with. To make sure you are collecting or uploading training data, navigate to Data page and make sure Training is selected in the dropdown menu before you collect or update data.



Test data

A set of data instances used to evaluate the performance of a trained model. To make sure you are collecting or uploading test data, navigate to Data page and make sure Test is selected in the dropdown menu before you collect or upload data.



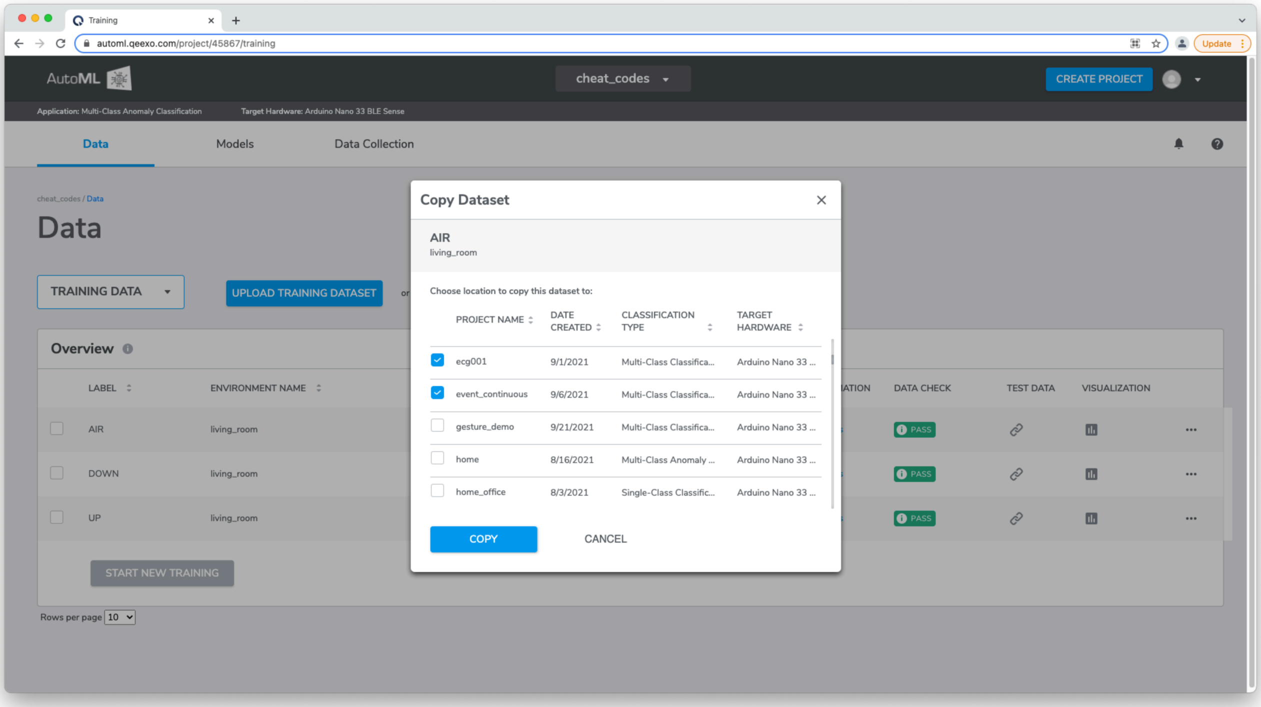
Training and test data operations

AutoML provides operations for users to manage their data collections and imports directly from training or test data. Current options include: `Copy`, `Delete`, `Export`, `Edit Label`, and `Segment`

Copy operation

The `Copy` operation allows users to copy training or test data from a source project to one or more destination projects. The source and destination project(s) must be of the same device kind.

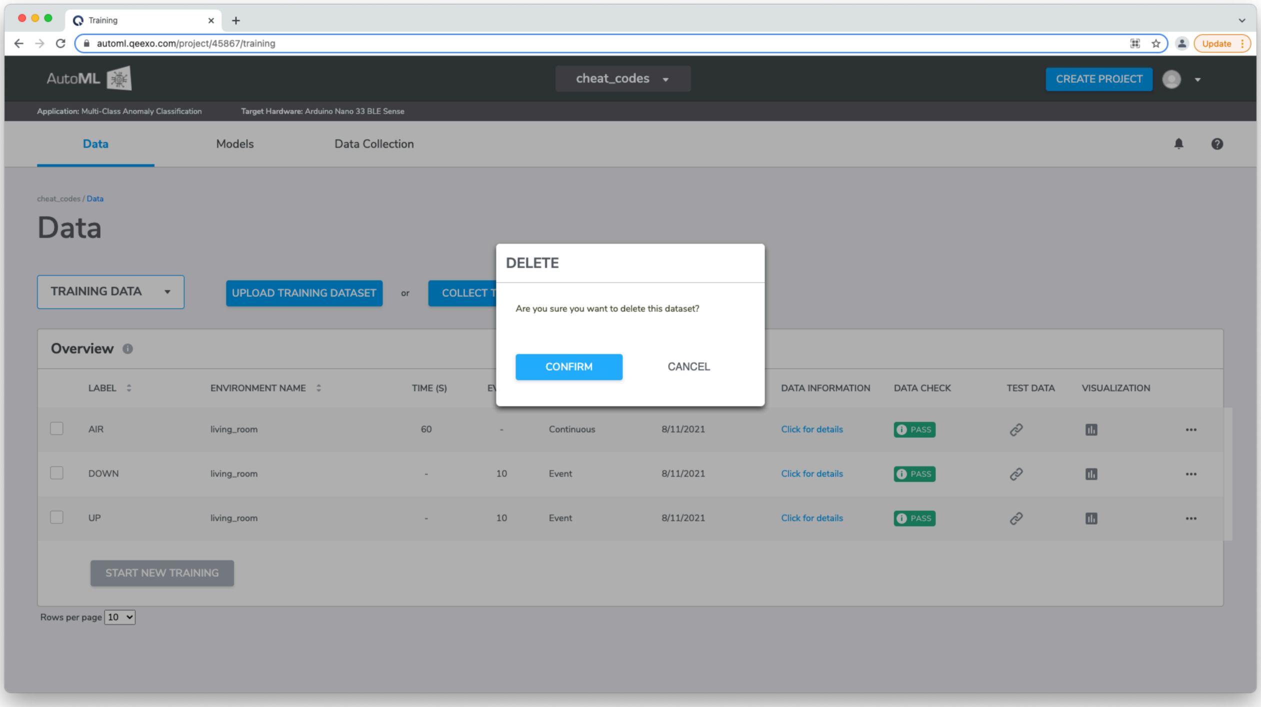
To copy a dataset from a source to one or more destination projects, click the `Copy` option located in the ellipse `...` operations menu for the desired dataset. From the resulting modal, locate and tick one or more destination projects and click `Copy`.



Delete Operation

The `Delete` operation allows users to delete training or test data that has not been associated with a model build.

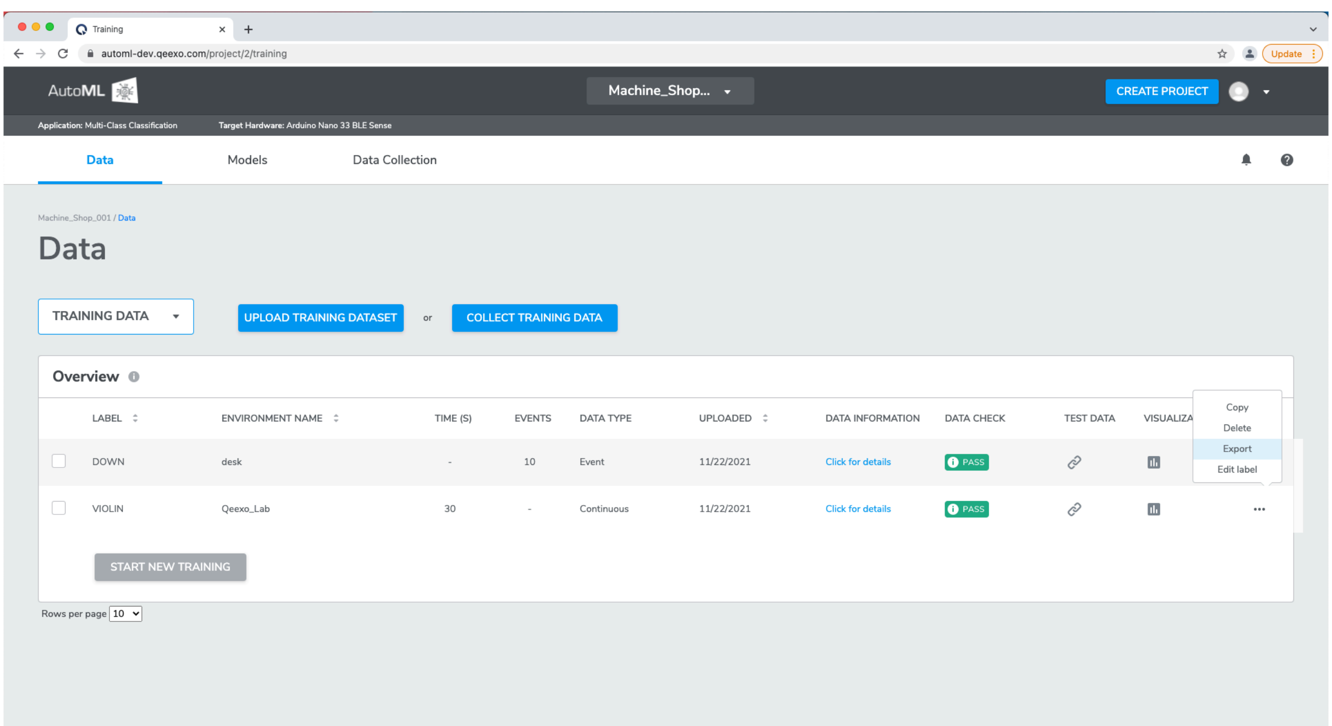
To delete training or test data from a project, click the `Delete` option located in the ellipse `...` operations menu for the desired dataset. From the resulting modal, click ‘Confirm’.



Export operation

The `Export` operation allows users to export training or test data to a .csv file in Qeexo AutoML format.

To export training or test data from a project, click the `Export` option located in the ellipse `...` operations menu for the desired dataset to initiate the download, specify the destination if requested.



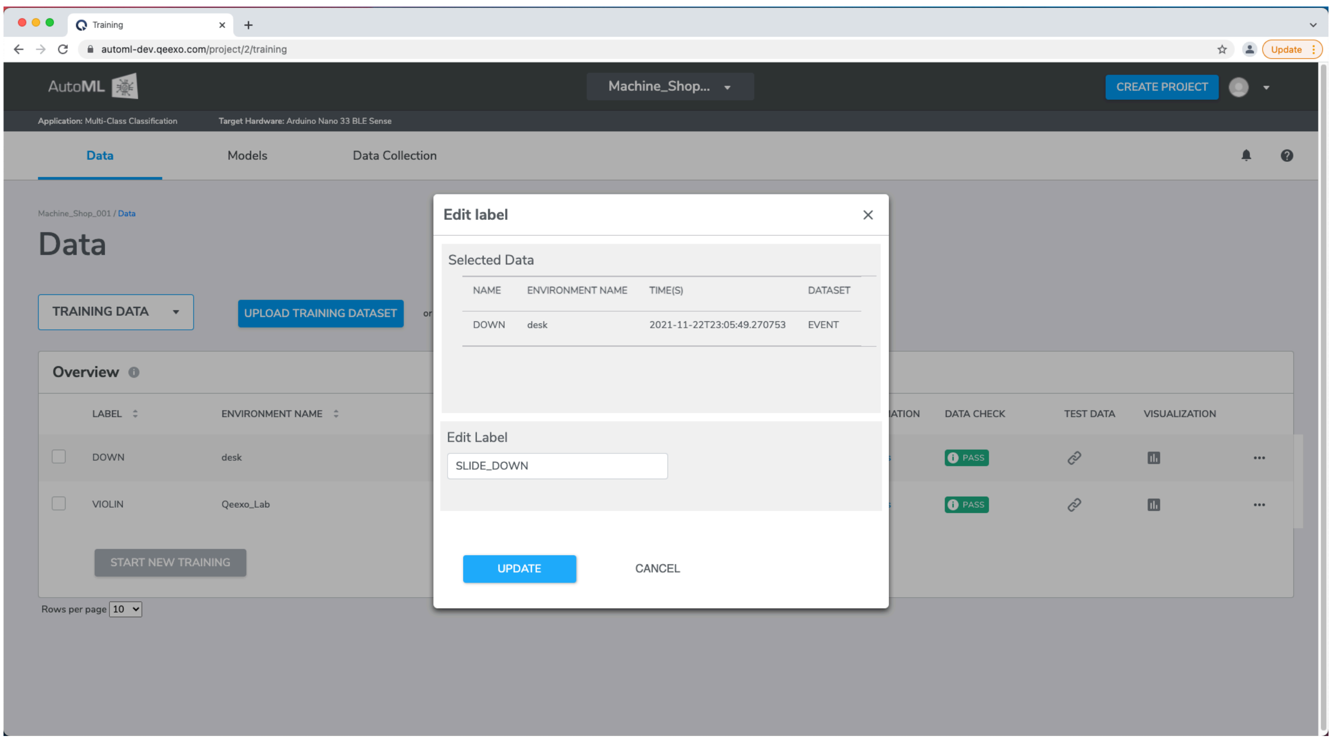
Please note: if your collection contains multiple segmented versions, you will first need to select the desired version number prior to export.

Edit label operation

The `Edit label` operation allows users to modify the given class label of one or more training or test datasets that have not been associated with a model build.

To modify the class label for a single collection, click the `Edit Label` option located in the ellipse `...` operations menu. From the resulting modal, provide a new or existing class label for the collection and click Save.

To modify the class label for one or more collections, start by ticking the desired datasets, next click the `Edit Label` option from the header row, finally, from the resulting modal, provide a new or existing class label for the collections and click Save



Segment Operation

The AutoML data segmentation feature allows you to locate, crop, and label events from your collection displayed in the data visualization editor for use as training or test data. Once segments are created and included in training, a version history allows you to recall and modify past segment sets, and then retrain your models in effort to achieve the highest performing model possible.

Creating Segments

From the training or test data pages, start by clicking on the `...` ellipse menu and select the `Segment` option to load the segmentation data visualization editor. Once open, use the drag bars and your mouse pointer to locate and zoom a region of interest, next click `Create New Segment` to activate the selection tool, following click the start and end points of the desired event and give your segment a label (optionally provide a color tag), finally click `Create` to save your segment. Repeat this process as necessary.

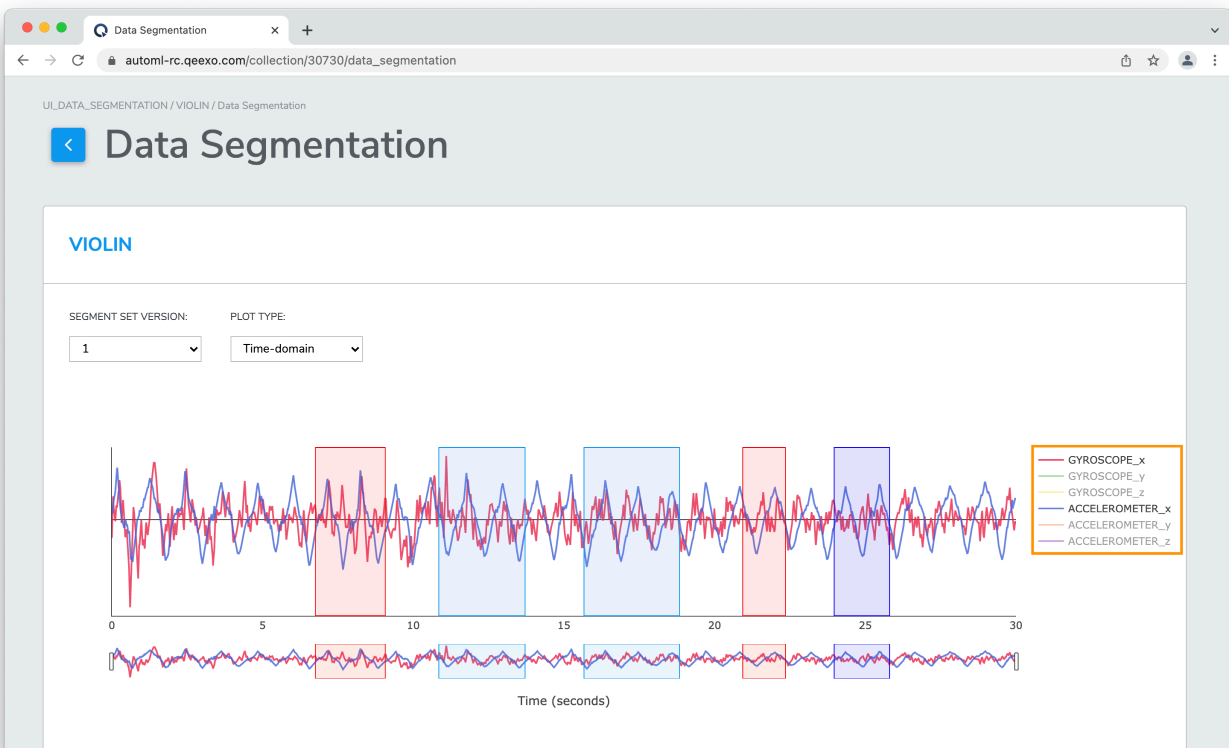
Once complete, use the dedicated back button to return to the `Training` or `Test` data pages and select any combination of compatible segmented and non-segmented data collections to start training.

Show / Hide Sensors & Axes

By default, Qeexo AutoML displays all enabled sensors data included in the collection across a single time or frequency visualization. Data visualizations for each sensor axis may be enabled or disabled by clicking on the individual axes from the `sensor key`. Optionally, double-clicking on an individual axis will enable or disable data visualizations for all other axes.

Please note: disabling the data visualization for any number of individual axes does not exclude those axes from data segmentation, all enabled sensors data is included in each segment regardless of visualization.

!



Modifying / Deleting Segments

At this time the Qeexo AutoML data segmentation feature does not allow for editing of a segment's start / stop selection and label. To modify a segment following creation, simply select `Delete` through the segment's `...` ellipse menu and recreate the segment using the process above.

Segmentation Version History

At the time of training Qeexo AutoML saves a recallable `Version` of your segments, selecting any available version from the `Version` drop-down menu will restore the segments for that collection. Once restored, new segments may be created and / or existing segments may be deleted before using the collection in training.

Test Data Evaluation

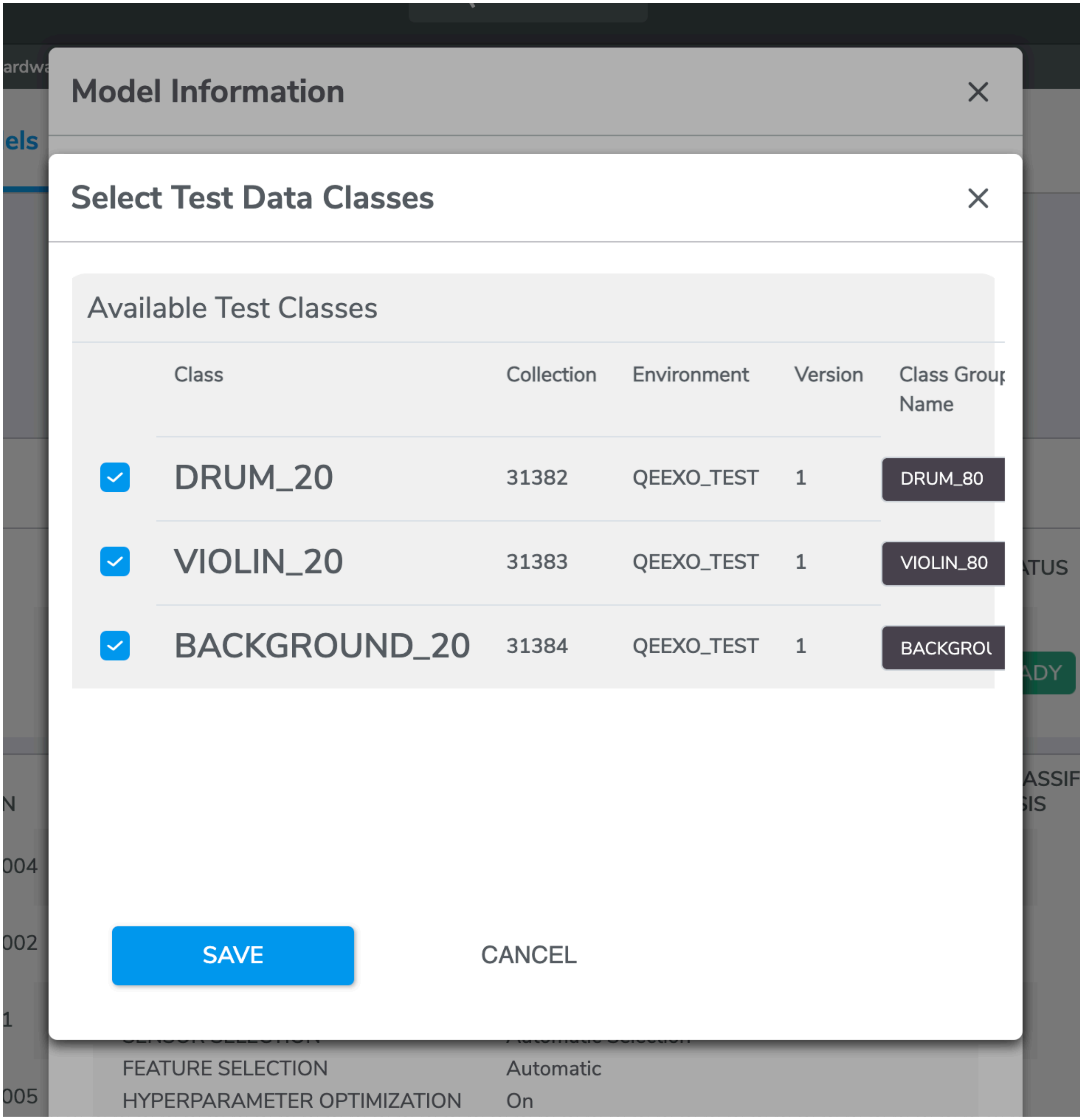
A test dataset can be associated with at most one training dataset, both must share the same sensor configuration.

When a test dataset is linked to a training class label, it will be used to evaluate the performance of the final model created using said training data.

Consider the following example: For a multi-class classification project applied to activity tracking, you want to evaluate the model performance against actual users. You first collect/upload test data for two groups, users age 18 - 64 and user age 65+, you then link the test label to the corresponding training label you want to evaluate against (*ex: running-test-68-john and running-test-38-beth to running-training and walking-test-68-john and walking-test-22-beth to walking-training*). You will see the overall test data accuracy and other matrix after the model is done training. Later, let's say you want to find out how the model performs for users age 65+, you can unlink all test datasets for users under the age of 65 from their corresponding training datasets, and then evaluate the changes, the result will give you an idea of what performance you can expect for users in the age group of 65+ using this particular machine learning model.

Evaluating Test Data

Test data may be collected or uploaded from the `Test Data` page. After test data has been ingested and a model has been trained, test data can be linked to evaluate the model's test performance.

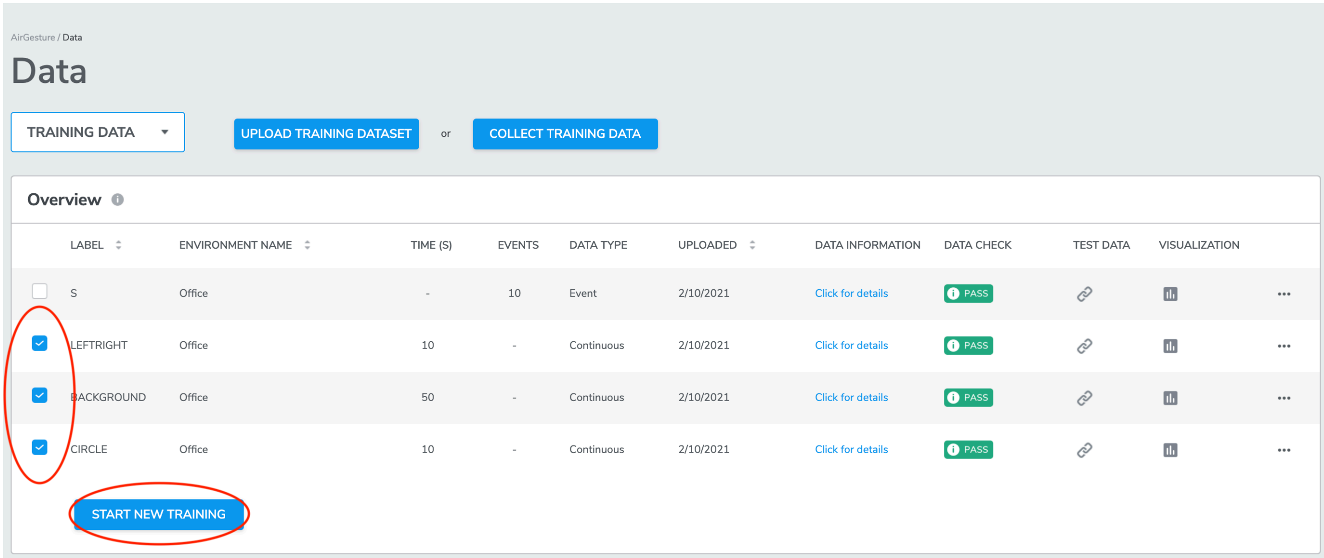


From the models page, clicking on training details for a particular build displays the Model Information. Clicking the 'Edit Test Data' button from this popup displays the Linked Test Data option. Here test data collections on the left may be linked to training data collections on the right by ticking the checkbox for the row and selecting the desired training collection form the drop-down. Clicking Save will dispatch the test run and evaluate the model's performance using the linked test data. Once evaluation is complete Test Perfromance values will be dsipalyed for each of the models.

Viewing and Managing Project Data

All of the Datasets associated with the current Project can be viewed and managed from the Data page. You can review the Dataset Information including its Sensor Configurations and Data Check results, as well as visualize and delete them.

From this page, select Datasets containing more than one Class Label to begin training machine learning models. (Also see section below on Building Machine Learning Models.)

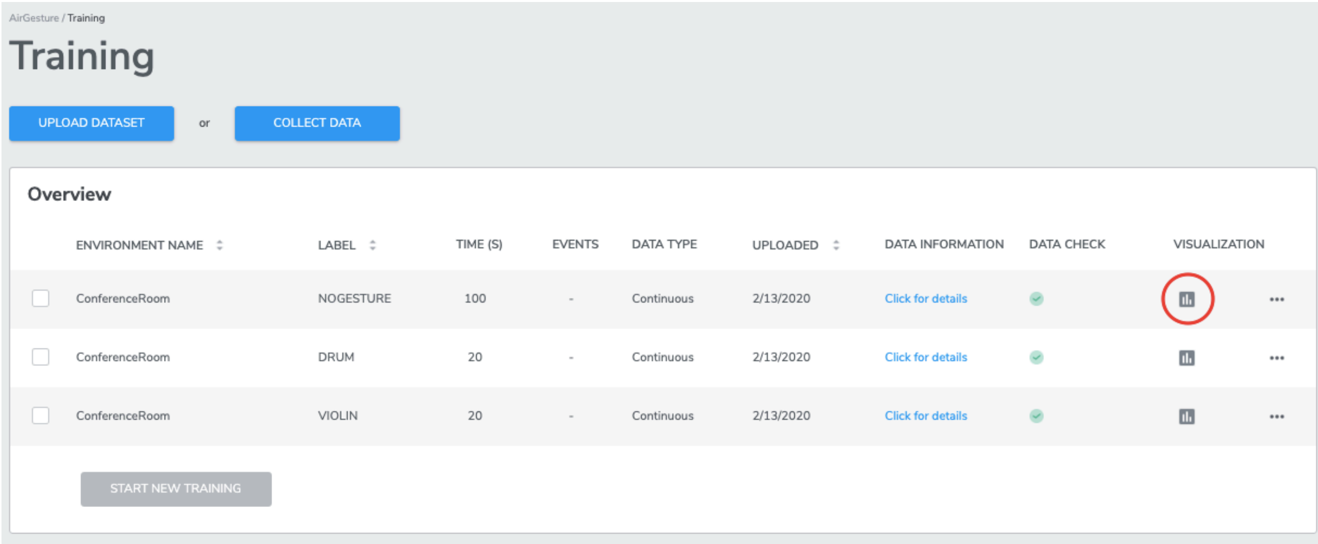


Visualizing Data

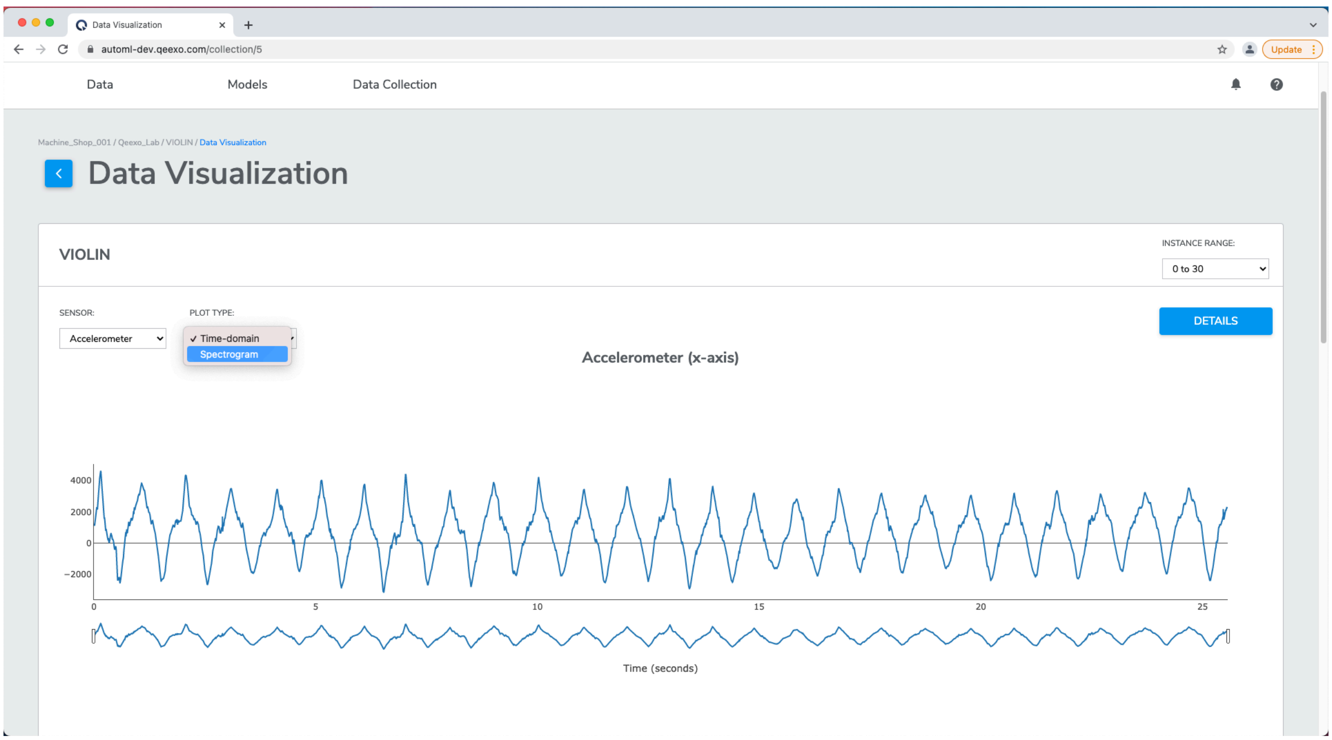
AutoML provides users with the ability to plot and view sensor data directly from the platform using the onboard data visualization tool. Navigate data using scroll, scale, and zoom options and view data in either `Time Domain` or `Frequency Domain`.

`Time Domain` visualization is a visual representation of the signal's amplitude and how it changes over time. With time domain visualization, the x-axis represents time, whereas y-axis represents the signals amplitude.

`Frequency Domain` visualization, also known as spectrogram frequency visualization is a visual representation of the spectrum of frequencies of a signal as it varies with time. With spectrogram frequency visualization, the x-axis represents time, whereas y-axis represents the signal's frequency.



To visualize training or test data, click the data visualization icon.



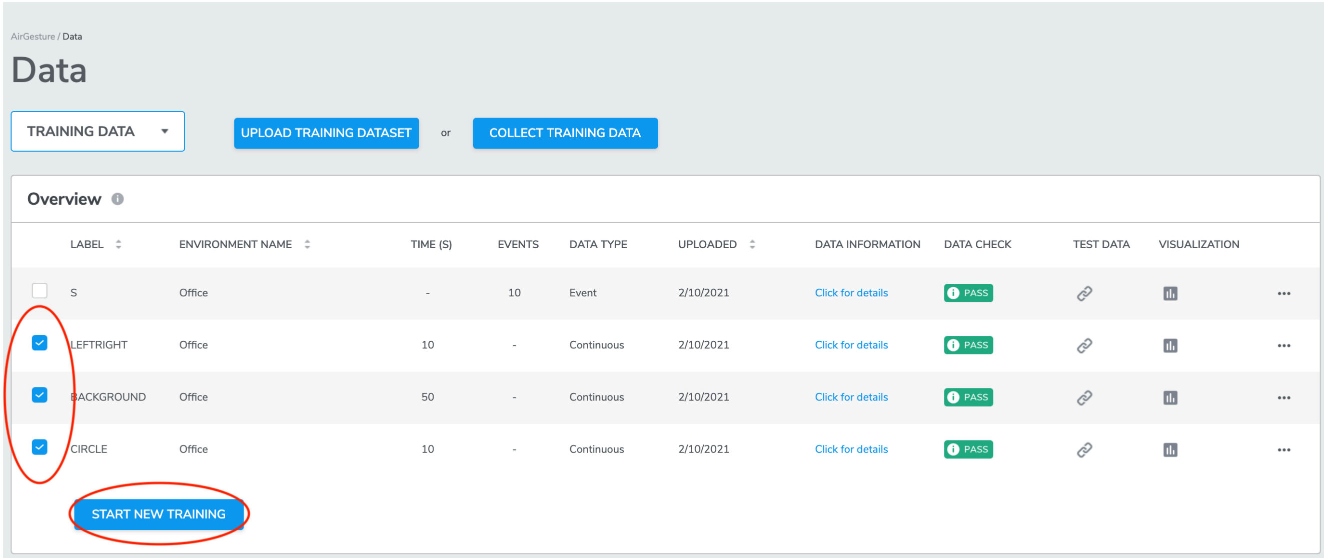
Toggle between different sensors using the `Sensor` drop-down or view frequency and time domain representations using the `Plot Type` drop-down menu.

The first 100 seconds of each Dataset will be shown in visualization. If there are over 100 seconds, the extra will be paginated.

Building Machine Learning Models

Navigate to the Data page to build machine learning models with collected and/or uploaded training data.

Select which training datasets to use to build machine learning models by clicking the checkbox at the left of each Dataset.



Note that the selected Datasets should ideally be from the same Environment, but Qeexo AutoML will allow you to train Datasets from different Environments as long as the selected sensors and Sensor Configuration are identical.

Once the desired Datasets are selected, click "Start New Training" button to configure Training Settings. Note that the "Start New Training" button is only clickable when Datasets containing 2 or more Class Labels are selected in Multi-class classification. However, for One-class classification, the button becomes clickable as soon as the first collection has been selected.

There is a minimum amount of data Qeexo AutoML needs for each Class Label in order to train machine learning models, this minimum is 640 samples from the highest ODR sensor. For example, if your sensor configuration is 104 Hz accelerometer & 25 Hz humidity, you need to collect data for at least 7 seconds (because 104 Hz * 6 seconds < 640 samples).

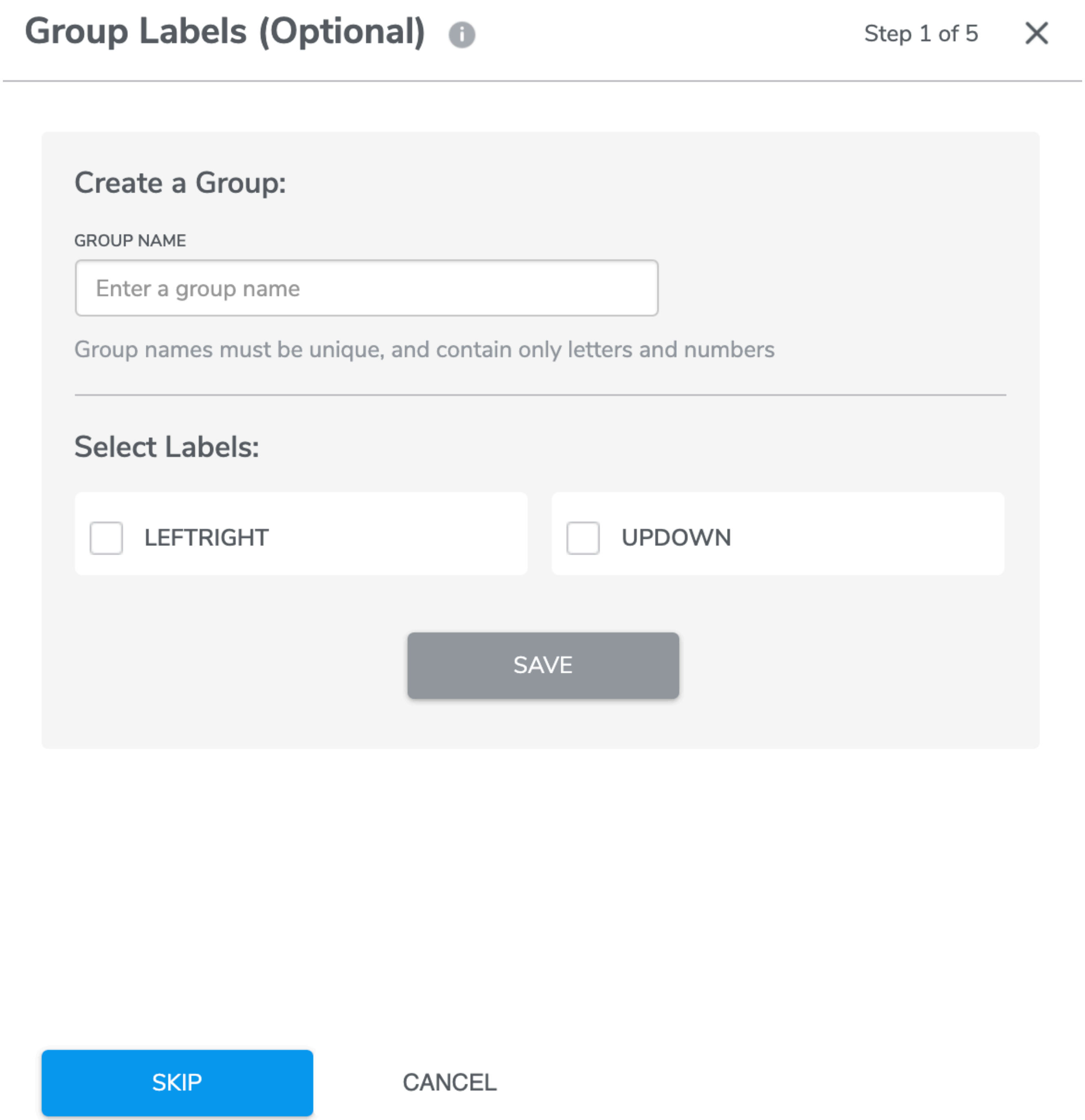
Note that satisfying this minimum requirement does not guarantee performance/accuracy; it is just the minimum amount of data our platform will work with. For good performance, you should likely collect much more than this minimum amount of data.

Training Settings

Step 1. Group Labels

This step is an optional step in case you have many class labels that you want to group together as a single class before model training. Consider the following example:

For a one-class classification project applied to anomaly detection, you may have machinery data that is labelled based on two different types of motion: vertical rotation and horizontal rotation. Since both of these classes are expected behavior, it is convenient to group these labels as a "Normal" group to feed into single-class classification.

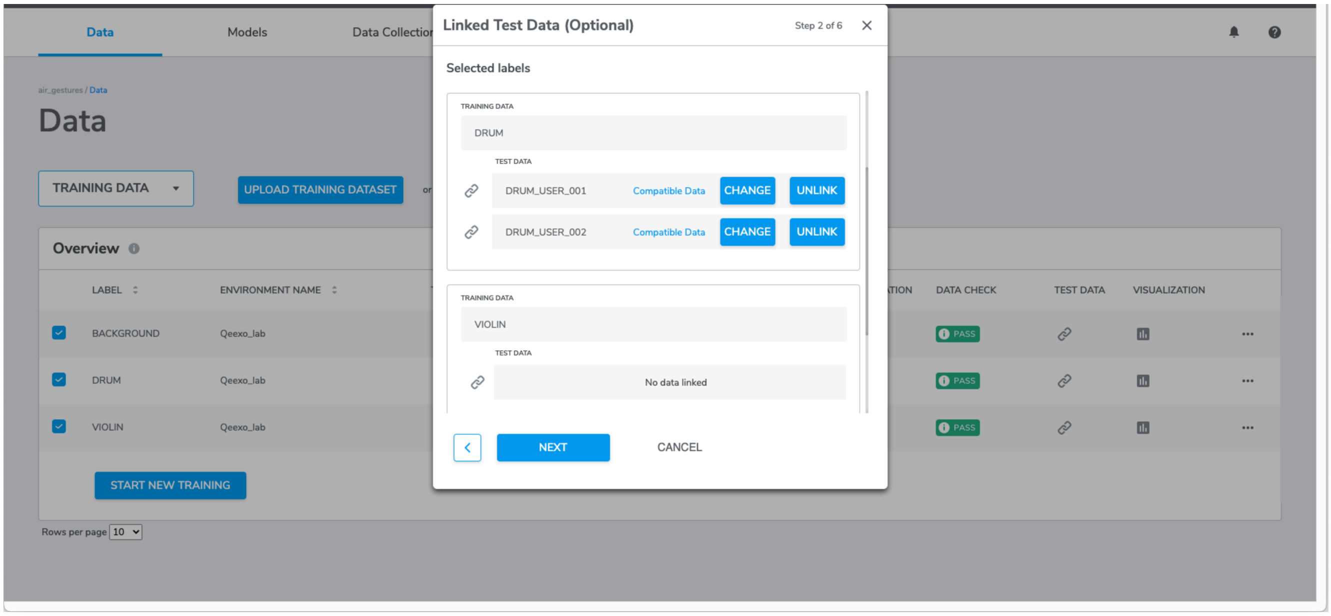


This is an optional step that can be bypassed by pressing the SKIP button.

Step 2. Linked test data

This is an optional step displaying an overview of test data management and whether any test data is linked to the selected training datasets. It gives you a chance to do any last minute adjustments on test data linkage before training your models. From the "Linked Test Data" popup, you can "Unlink" and "Link" datasets. Alternatively, clicking "Change" displays options for "Including" or "Excluding" the test data from evauaiton during training. When

Note: One training data label can link multiple test data labels so long as they are compatible - sharing the same sensor configuration, click the Compatible Data button to validate.



If your project type is NOT MLC-related, proceed to Step 3a and Step 4a

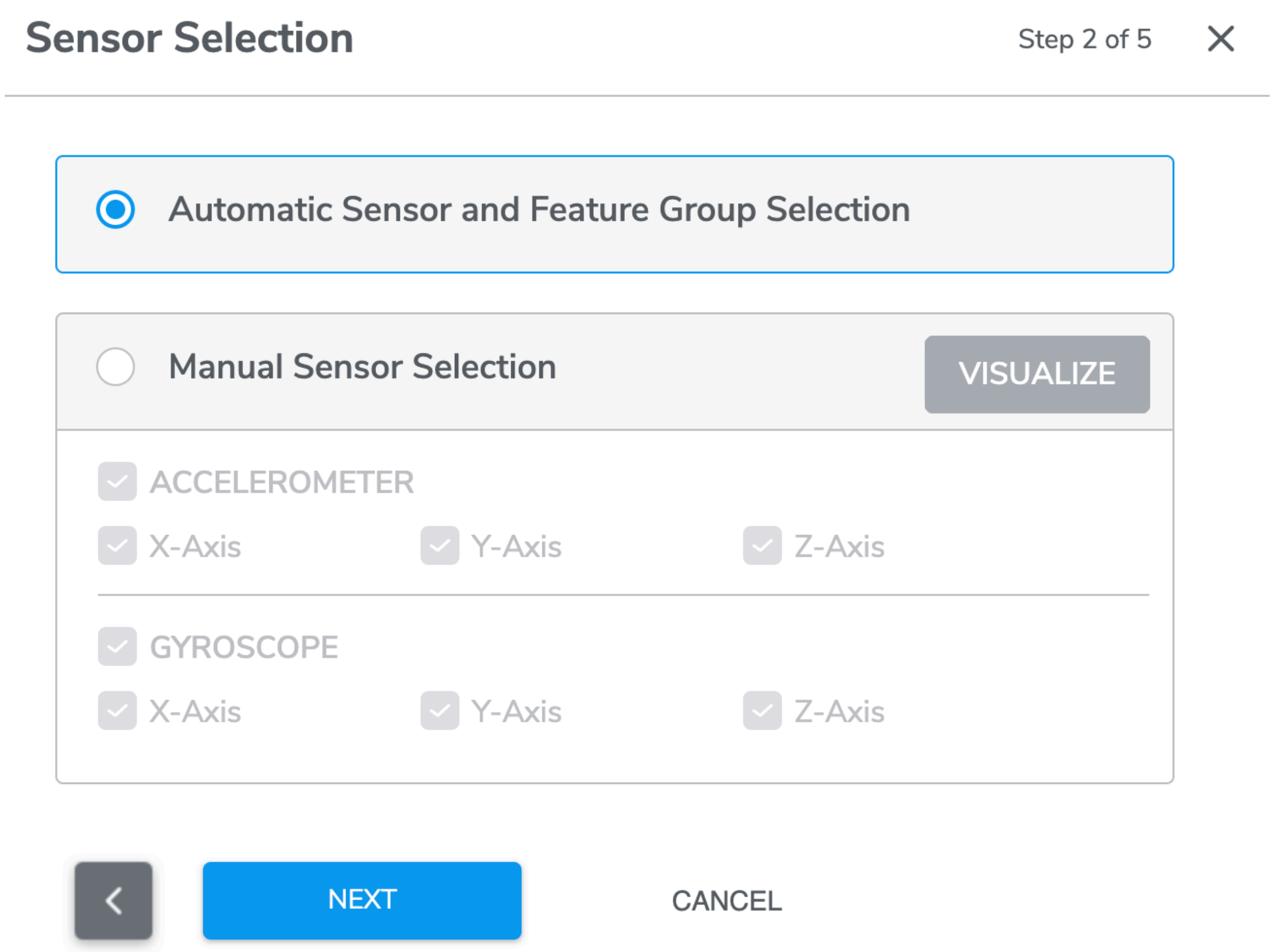
If your project type is MLC, proceed to Step 3b and Step 4b

Step 3a. Sensor Selection (applicable to non-MLC projects)

Next, there is an option to have Qeexo AutoML automatically select sensors and feature groups for optimal model performance. If you have collected data from multiple sensors, but you are not sure which may be helpful for the given problem, it is recommended to enable this feature.

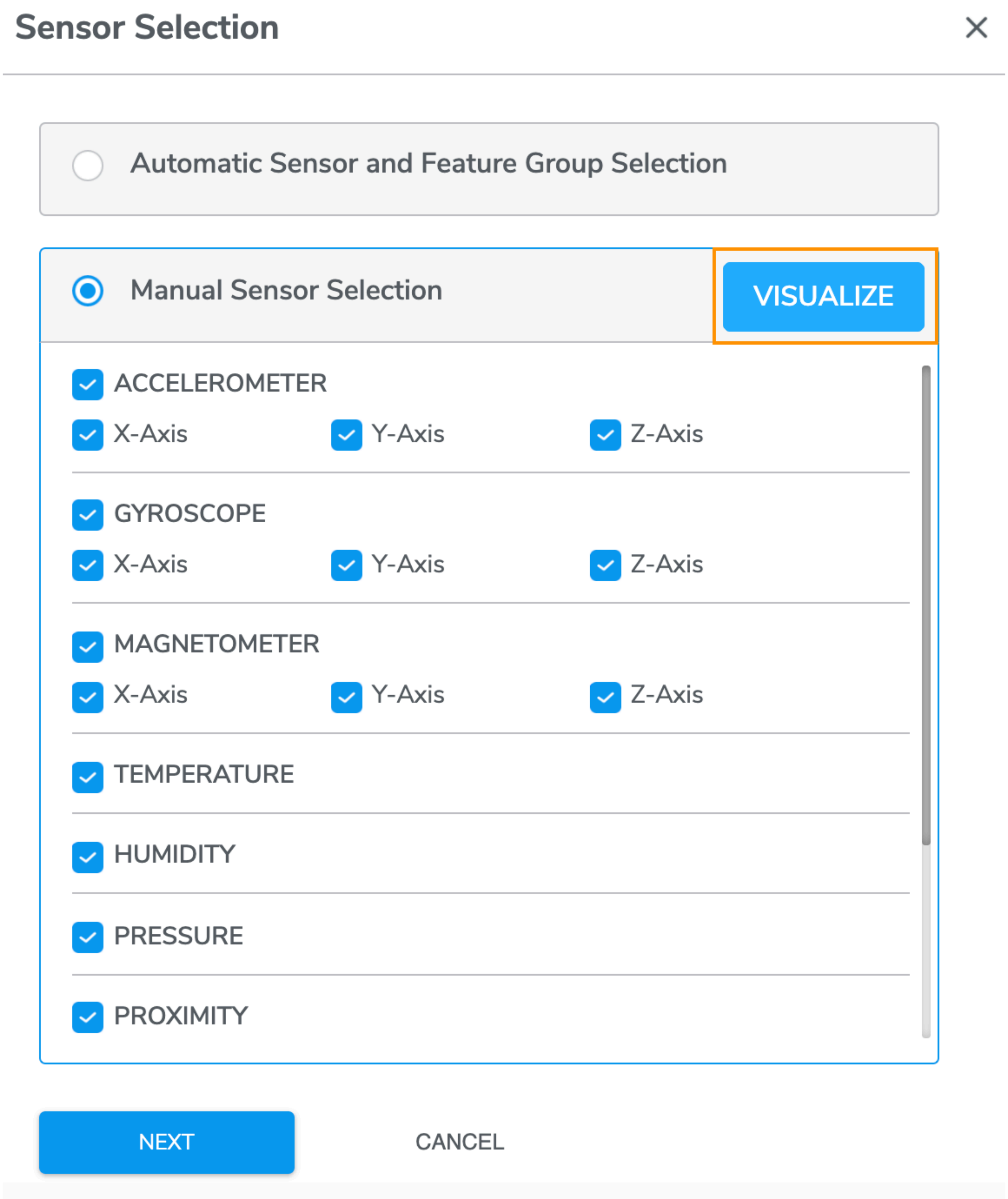
Qeexo AutoML computes hundreds of statistical and signal-processing-based features from raw sensor data. Turning on Automatic Sensor and Feature Group Selection will automatically select a subset of sensor inputs as well as a subset of features relevant to the current use case. Using a smaller number of sensors and/or features will reduce classification latency and model size. This operation may increase the training time due to the additional computation required to select features. This increase in time largely depends on the amount of data and the machine learning algorithm used.

Note that this automatic selection applies to both sensor and feature groups. If you know what sensors should be used for the problem, but you still want to enable feature selection, that is possible on the following page.

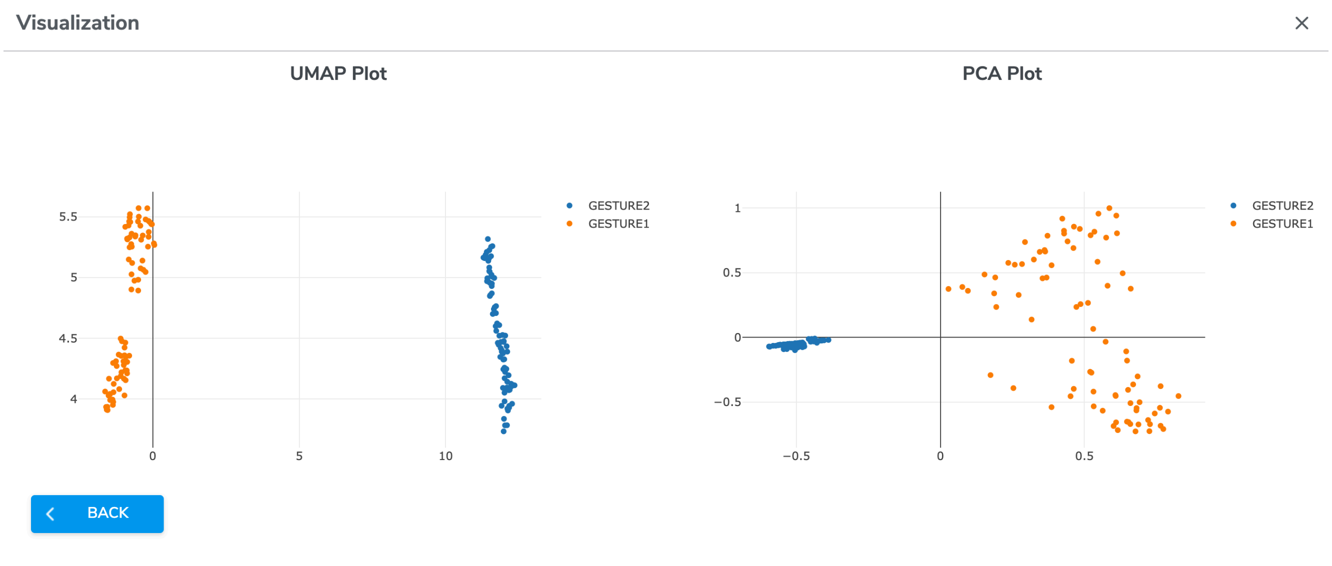


If Manual Sensor Selection is enabled, you will be able to select a subset of sensors to feed into the ML pipeline. Note that:

- All sensors included in [Project Creation] are available for selection in this stage
- Individual axes from multi-channel sensors can be selected independently. These include:
  - X-, Y-, and Z-axis selection for Accelerometer, Gyroscope, and Magnetometer sensors
  - Red, Green, Blue and Clear Light channel selection for Ambient Light sensor



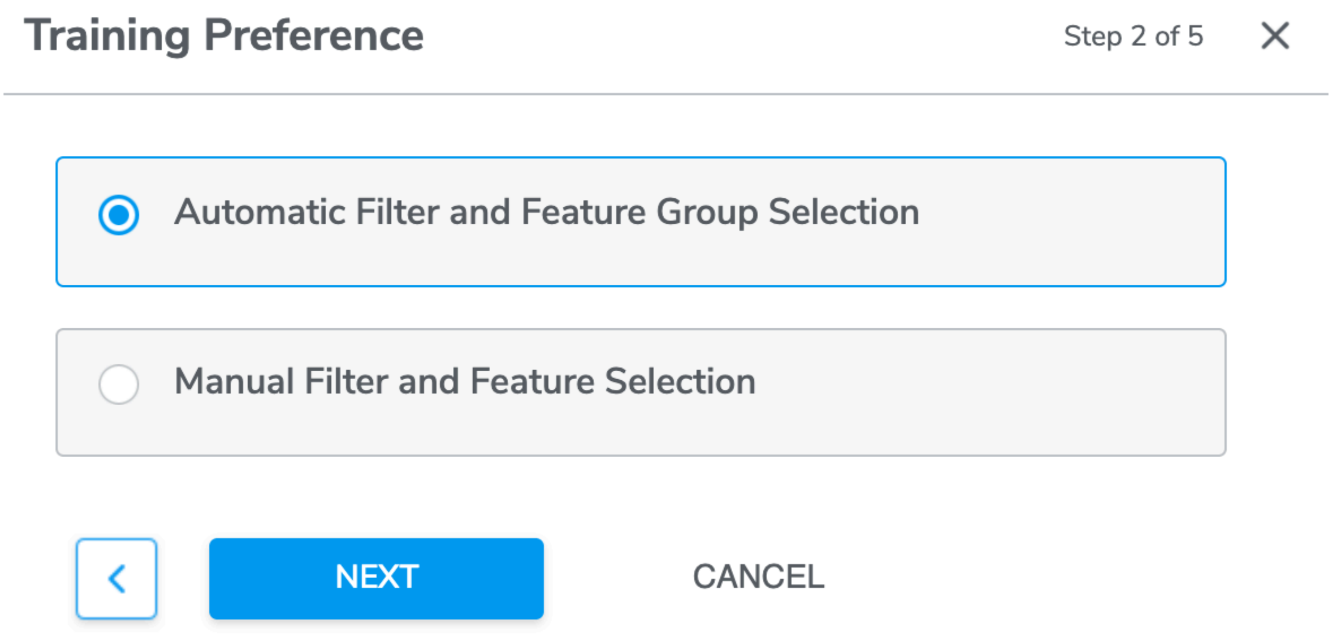
Under manual selection, you can visualize data collected from sensors using a general-purpose dimensionality reduction algorithms called UMAP(Uniform Manifold Approximation and Projection) and PCA(Principal Component Analysis). UMAP is a novel manifold learning technique for dimension reduction. For further details about UMAP algorithm, please refer to "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction" available at https://arxiv.org/abs/1802.03426. PCA linearly transforms input features into a lower-dimensional space where the variance of the features is maximally preserved. PCA is very well established statistical technique for dimensionality reduction while preserving the variance. Similar to UMAP plot, we project features into two-dimensional space for the PCA visualization as well.These visualizations can often determine which sensors might be useful for classification. Examining these UMAP and PCA plots before making a final manual sensor selection is encouraged.



Press NEXT when you are ready to proceed to the next stage. Advance to Step 4a.

Step 3b. Filter configuration (applicable to MLC projects)

If MLC users would like Qeexo AutoML to automatically choose filters and features for their models, select `Automatic Filter and Feature Group Selection` and click NEXT (please advance to Step 5b "Inference Setting"):



For the users who want to manually select filters, they can choose "Manual Filter and Feature Selection" and click NEXT.

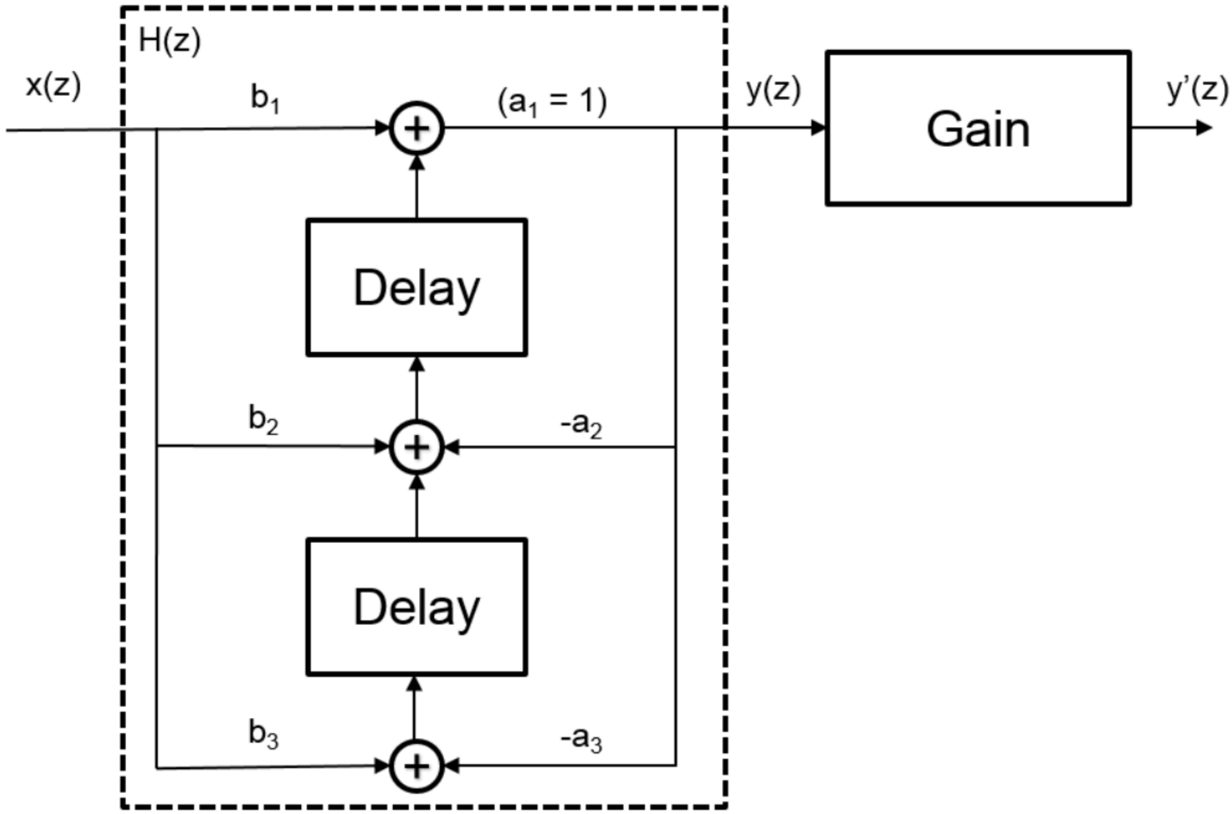
Users will be able to configure up to 7 filters that are available in Machine Learning Core logic. Currently the following filters are available:

- High-pass filter
- Band-pass filter
- IIR1 filter
- IIR2 filter

Note that filter configuration is entirely optional. According to [ST's documentation](), the transfer function of the filter is defined as:

$$H\left(z\right) = \frac{b_1 + b_2 z^{-1} + b_3 z^{-2}}{1 + a_2 z^{-1} + a_3 z^{-2}}$$

where by the relationship between input, transfer function, gain, and output is illustrated like this:

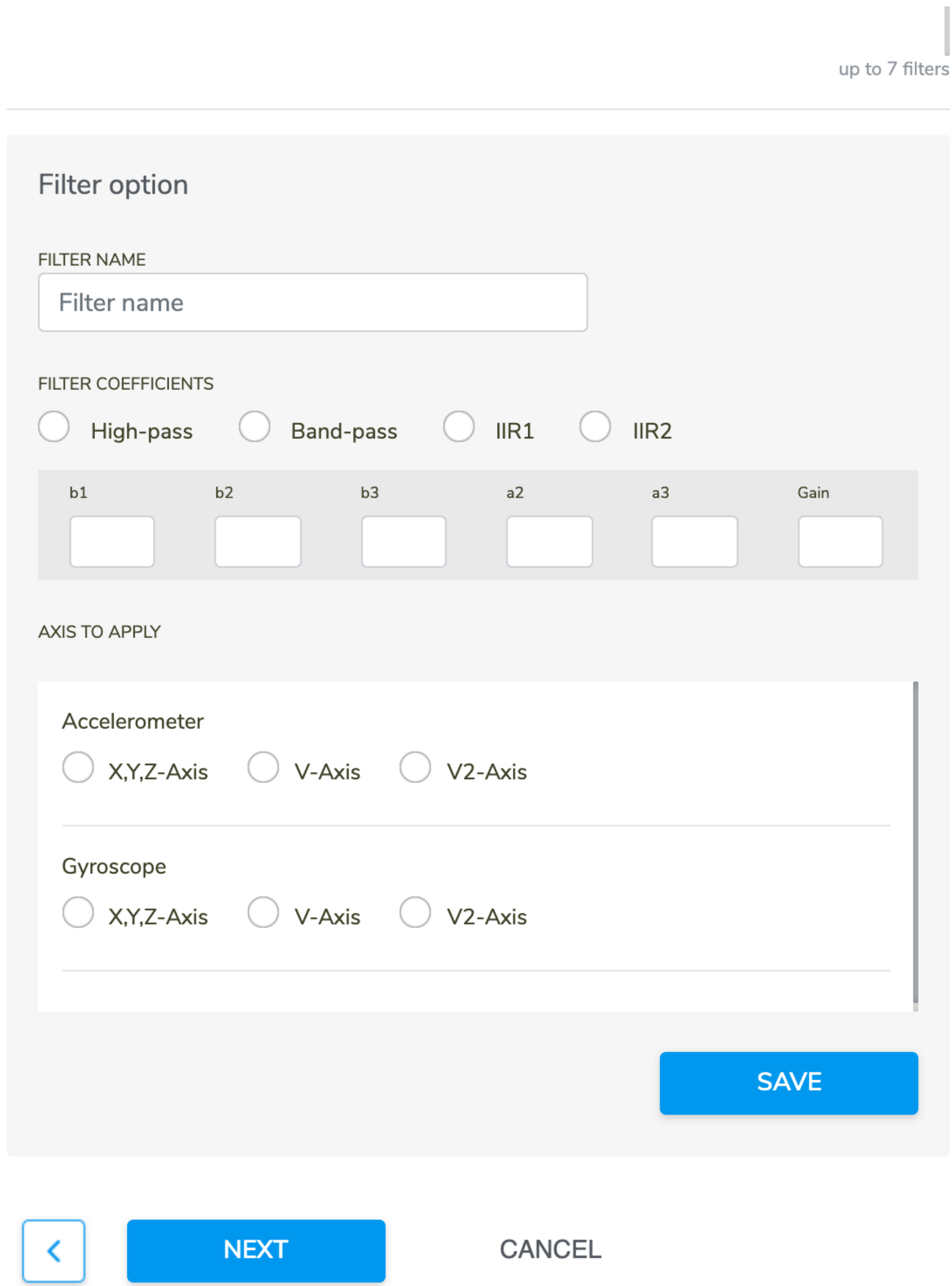


With this understanding, some of the coefficients are configurable depending on the filter type:

Filter type / Coefficients	**b1**	**b2**	**b3**	**a2**	**a3**	**Gain**
High-pass filter	0.5	-0.5	0	0	0	1
Band-pass filter	1	0	-1	Configurable	Configurable	Configurable
IIR1 filter	Configurable	Configurable	0	Configurable	0	1
IIR2 filter	Configurable	Configurable	Configurable	Configurable	Configurable	1

For a detailed explanation and example of filter coefficients, refer to the [ST documentation]().

Qeexo AutoML allows for filter to be added and removed:



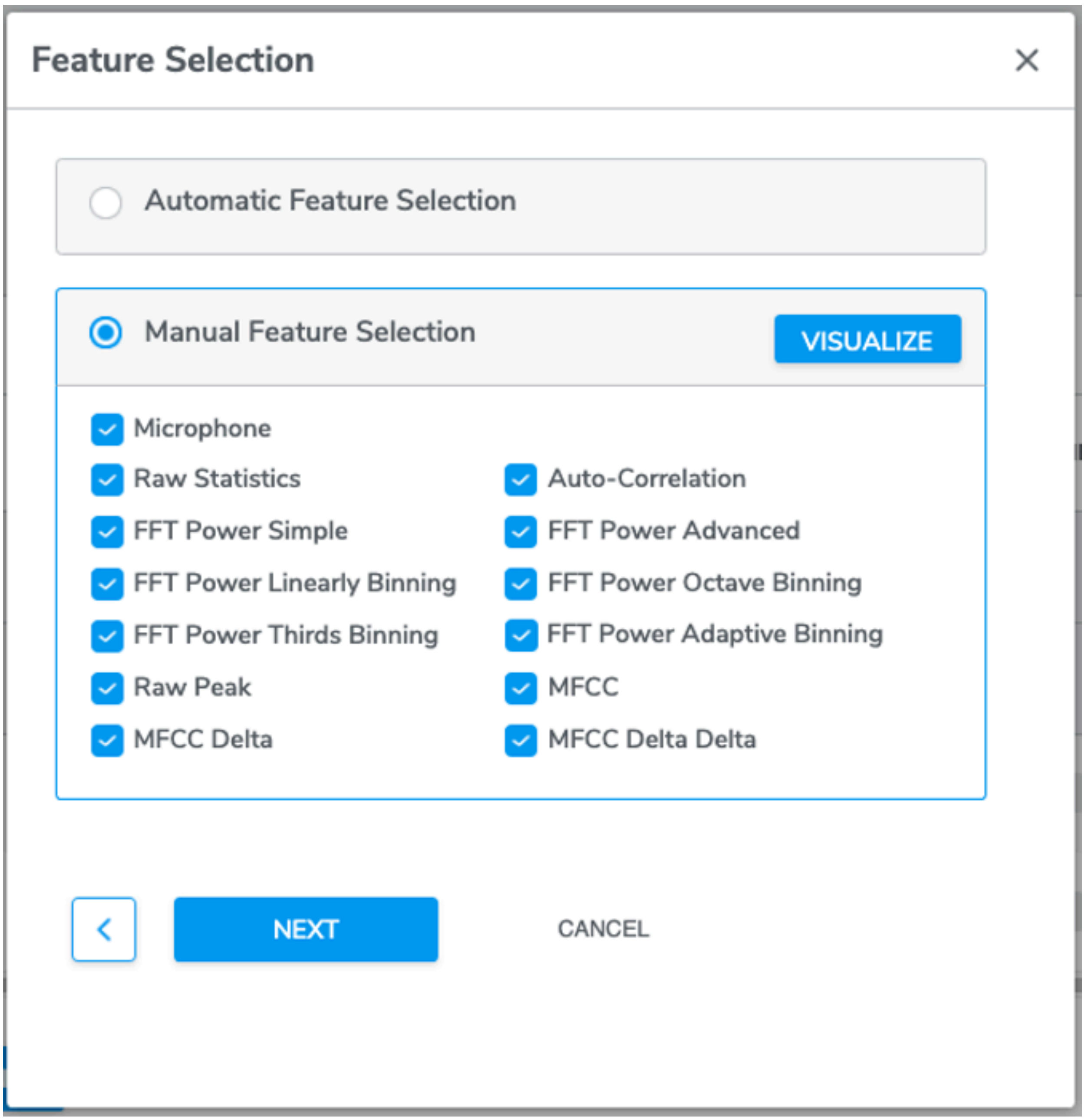
Note that `V2-Axis` means sum of squares of x, y, and z-axis (i.e. x^2+y^2+z^2) while `V-Axis` is the square root of `V2-Axis` (i.e. sqrt(x^2+y^2+z^2)).

Click NEXT to move forward to Step 4b.

Step 4a. Feature group configuration (applicable to non-MLC projects)

Note that this step is automatically skipped when "automatic sensor and feature group selection" is chosen in the previous screen.

When you chose manual selection in the previous sensor screen, you will be presented an option to select between automatic and manual feature selection:



You may want to refer to the following description of features during manual selection:

Feature group	Description
Raw Statistics	Computes the statistics based on the raw sensor data such as range, standard deviation, skewness, etc.
Auto-Correlation	Compute the auto-correlations of raw sensor data, measuring how much the signal correlates with itself through time
Raw Peak	Computes two representations, linear fit and zero crossing rate (ZCR). Linear fit computes the statistics (mean, standard deviation) on coefficients when fitting a linear regression line to the signal peaks; ZCR Computes the zero-crossing rate on signal peaks. Zero crossing rate is the number of times signal crosses zero.
FFT Power Simple	Computes the statistics based on the power of Fast Fourier Transform (FFT) performed on raw sensor data. Includes quantities such as range, mean, standard deviation, skewness, root mean square, etc.
FFT Power Advanced	Computes several other quantities based on the power of FFT. Includes the spectral centroid, bandwidth, rolloff point, decrease, flatness, linear slope, etc.
FFT Power Linearly Binning	Computes the power spectrum of FFT coefficients binned linearly.
FFT Power Octave Binning	Computes the power spectrum of FFT coefficients bucketed into octaves.
FFT Power Thirds Binning	Computes the power spectrum binned into music thirds where numbers are binned in 3 logarithmically spaced steps.
FFT Power Adaptive Binning	Computes the power spectrum binned adaptively using the training data to highlight areas of interest in frequency space. Often useful for problems where a significant portion of the information is concentrated within a specific frequency band.
MFCC	Computes the Mel-frequency cepstral coefficients (MFCC) based on the power of FFT coefficients. MFCCs are the discrete cosine transform of the Mel-filterbank coefficients where Mel-filterbank coefficients are the log of power at 16 distinct Mel-frequencies.
MFCC Delta	Computes the difference between the current coefficients and the previous coefficients in MFCC features.
MFCC	

Delta Delta	Computes the difference between the current and the previous delta value in MFCC delta features.

If the automatic option is chosen, Qeexo AutoML will select the optimal set of features based on final model accuracy as input into model training.

Note that features are grouped under each sensor. For each of the selected sensors, at least one feature group is required under the manual mode.

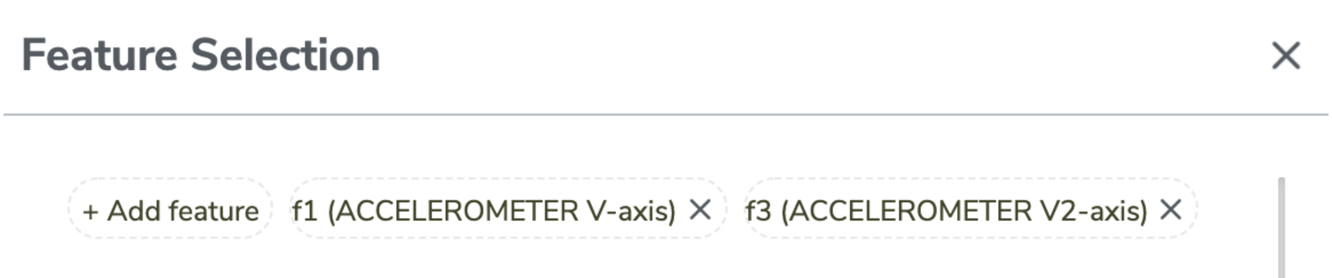
Similar to sensor selection screen, users can visualize data collected from sensors as well as feature groups using the UMAP visualization tool described in the previous section.

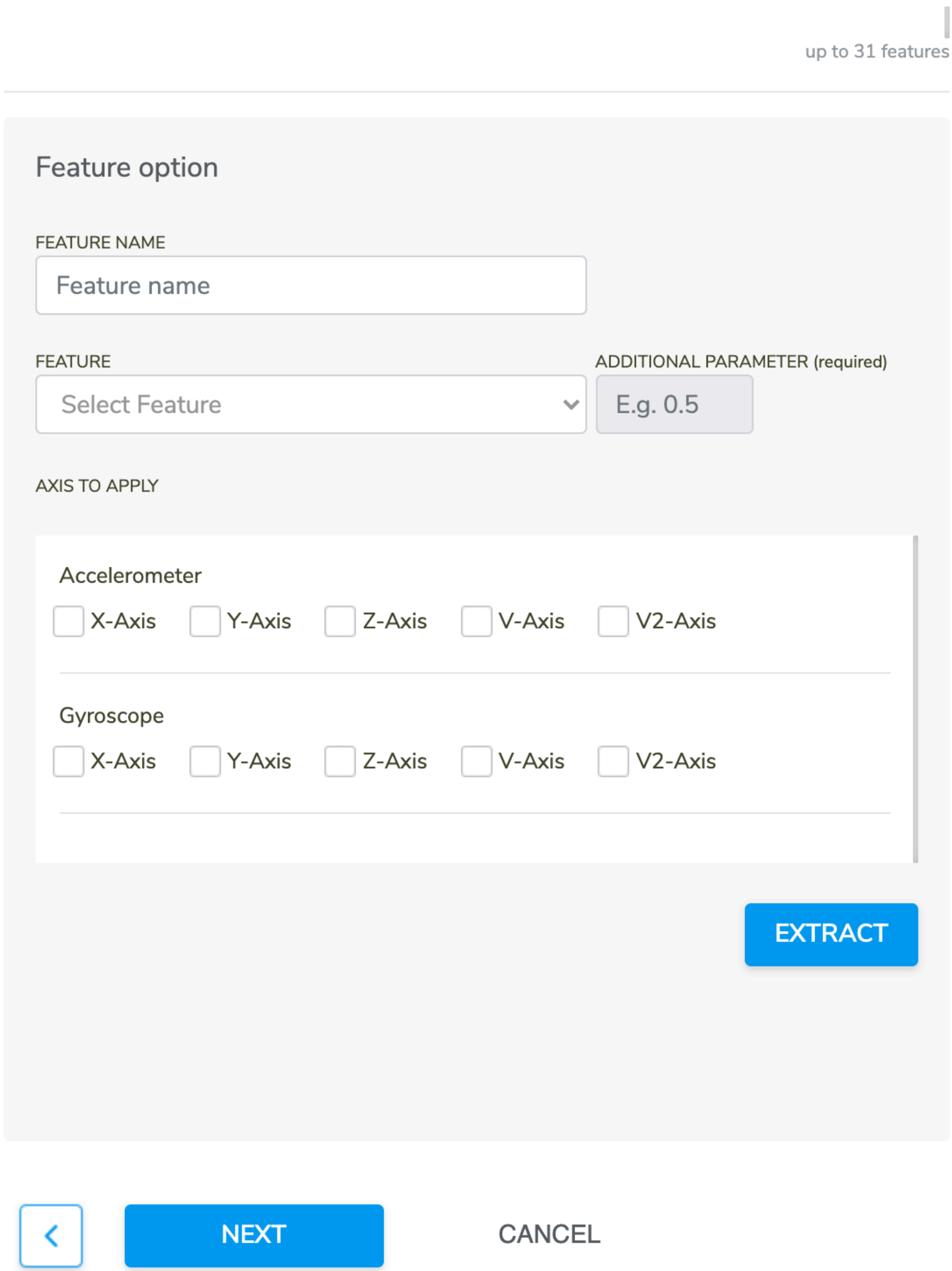
Step 4b. Feature Selection (applicable to MLC projects)

Similar to Filter configuration, users can add and remove feature manually. There are 12 features to choose from:

Feature	Additional parameter
Mean	Not needed
Variance	Not needed
Energy	Not needed
Peak to Peak	Not needed
Zero Crossing	Required
Positive Zero Crossing	Required
Negative Zero Crossing	Required
Peak Detector	Required
Positive Peak Detector	Required
Negative Peak Detector	Required
Minimum	Not needed
Maximum	Not needed

Refer to [ST documentation] for more details about the features available in MLC logic.





Proceed to Step 5b for Inference Setting.

Step 5a. Inference Setting (applicable to non-MLC projects)

Inference Settings

See below sections for more details about the Instance Length and Classification Interval.

Here are the recommended values for a couple of sample use cases:

Use Case	Instance Length (ms)	Classification Interval (ms)	Max Sensor ODR (Hz)
Air Gesture Recognition	2000	250	417
Machine Vibration Classification	300	100	6660
Human Presence Detection	5000	2000	100

Instance Length

This is sets the length of the sensor data which the model will use to make classifications. The maximum allowable value for this property is set based on the highest sensor ODR in the given sensor configuration.

The optimal value depends on the real-world *time duration* of the events corresponding to the Class Labels that you want to capture/detect with the sensors, as well as the selected ODRs used during data collection.

For example, in machine vibration problems, the most important requirement for the signal data is high ODR, because higher ODRs will increase the amount of high frequency information available to the model. For these type of problems, we should often use the highest available sampling rate, and then use the maximum allowable instance length for that ODR (~300 ms @ 6.6kHz).

However, in human activity detection and monitoring, 300 ms would likely not be a sufficient instance length for good performance. We need to make sure the window is long enough to capture some human motion and interaction with the environment, which may require several seconds of data in total. For this case, we should pick an instance length which we want to achieve (e.g. 5000 ms), and then we should use the highest possible ODR which still allows us to achieve this instance length.

Classification Interval

This sets the number of milliseconds between requests for classification.

For example, if Classification Interval is set to 200 ms, Qeexo AutoML will produce a classifier that classifies incoming data at a rate of 5 Hz.

This value should be set relative to the average time between class changes in the real world -- if the classes may change quickly or last for a very short period of time, Classification Interval needs to be set low so that classifications are run near-constantly in order to capture the events of interest.

For problems like human presence detection, where based on the nature of the problem the classes cannot change often, we may select a higher value (e.g. 2000 ms) in order to save on power consumption and network bandwidth.

Step 5b. Inference Setting (applicable to MLC projects)

MLC has two attributes affecting model inference: Window length and MLC output data rate.

Window Length

All the features are computed within a defined time window, which is called Window length. It represents number of samples, ranging from 1 to 255. Depending on the pipeline configuration, the allowed range for the Window Length may be subject to further restrictions to ensure a minimum number of training instances are created per class label.

MLC Output Data Rate

This parameter governs how fast MLC outputs classification result. Four rates are available: 12.5 Hz, 26 Hz, 52 Hz, and 104 Hz. This parameter must be less than or equal to the Inertial Measurement Unit (IMU) output data rate (ODR). When the MLC ODR is less than the IMU ODR, the sensor data will be downsampled inside the MLC to the MLC ODR before being used in the machine learning model.

Step 6. Model Settings

This page determines which model types are trained. For multi-class classification, it also contains switches for enabling a few different features related to model building. Note that only one model (Decision Tree) is available for MLC projects.

Single-class Classification

Algorithm Selection

For single-class classification, Qeexo AutoML supports the following machine learning algorithms:

- Isolation Forest (IF)
- Local Outlier Factor (LOF)
- One Class Random Forest (ORF)
- One Class SVM (OCSVM)

Multi-Class Anomaly Classification

Algorithm Selection

For Multi-Class Anomaly Classification, Qeexo AutoML supports the following machine learning algorithms:

- Decision Tree (DT) Unknown
- Gradient Boosting Machine (GBM) Unknown
- Logistic Regression (LR) Unknown
- Random Forest (RF) Unknown
- XGBoost (XGB) Unknown

Multi-class Classification

Hyperparameter Tuning

Hyperparameters are a set of adjustable parameters of machine learning models. These parameters affect the accuracy, runtime, and size of machine learning models. Different models have different parameters depending on the model architecture. AutoML provides built-in option for tuning these hyperparameters. There is a simply switch users need to flip if hyperparameter optimization is desired. If this option is enabled, AutoML tunes hyperparameters using a collection of optimization techniques tailored to TinyML applications. It maximizes accuracy while it ensures that all resource usages are under constraints (e.g., firmware binary size and memory usage). This option will often improve final model accuracy at the expense of additional runtime for model-building.

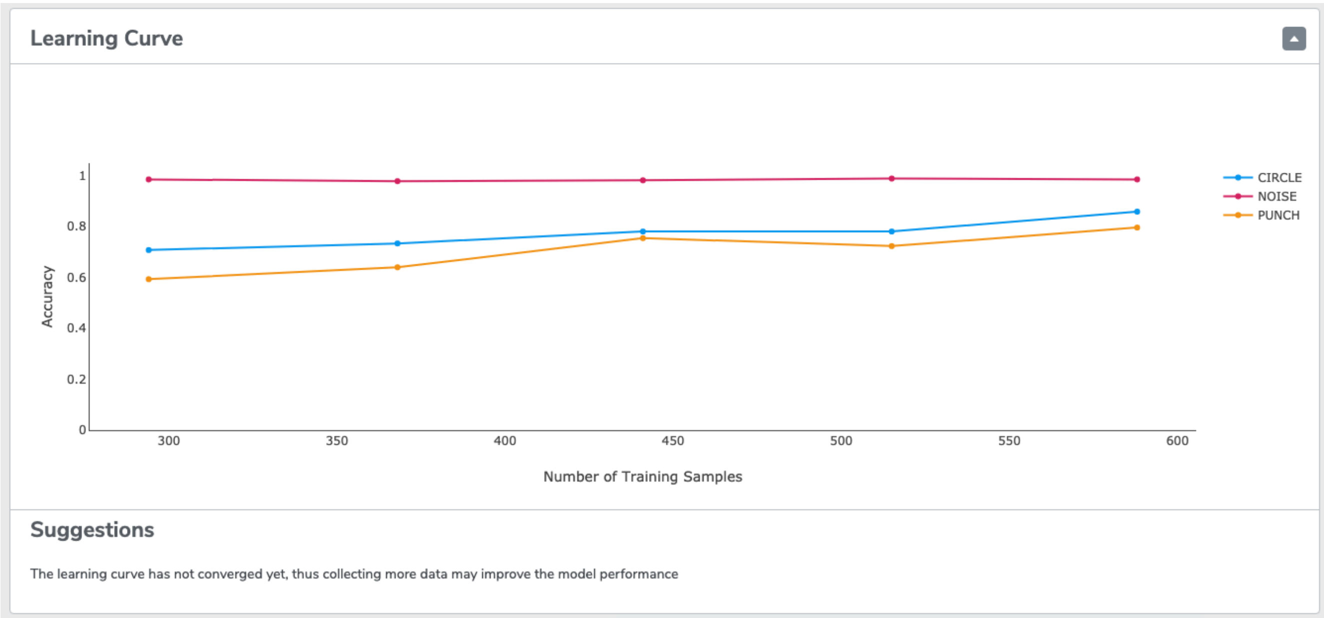
There are three settings that affect the duration of the hyperparameter tuning stage:

- Optimizer Time Limit:
- Optimizer Number of Trials:
- Optimizer Error Threshold:

Generate Learning Curve(s)

If enabled, this option will produce learning curves for the given data set. Learning curves visualize how your model is improving as more data is added. These curves can be extrapolated, which can be useful for determining if the model may benefit from additional data collection.

As shown in the example below, the "Circle" and "Punch" gestures are still improving with additional data. It is likely that they would continue to improve if more data is collected.



Note: If the dataset that is used for training is very small, the learning curves may not be accurate. The model may be very good at classifying the limited data it's seen, but might not generalize to new cases. In that case, even if the learning curve does not show it, it is safe to assume that final model performance will improve with additional data collection.

Algorithm Selection

For multi-class classification, Qeexo AutoML supports the following machine learning algorithms:

Ensemble Methods:

- Gradient Boosting Machine (GBM)
- Random Forest (RF)
- XGBoost (XGB)

Neural Networks:

- Artificial Neural Network (ANN)
- Convolutional Neural Network (CNN)
- Convolutional Recurrent Neural Network (CRNN)
- Recurrent Neural Network (RNN)

Support Vector Machines:

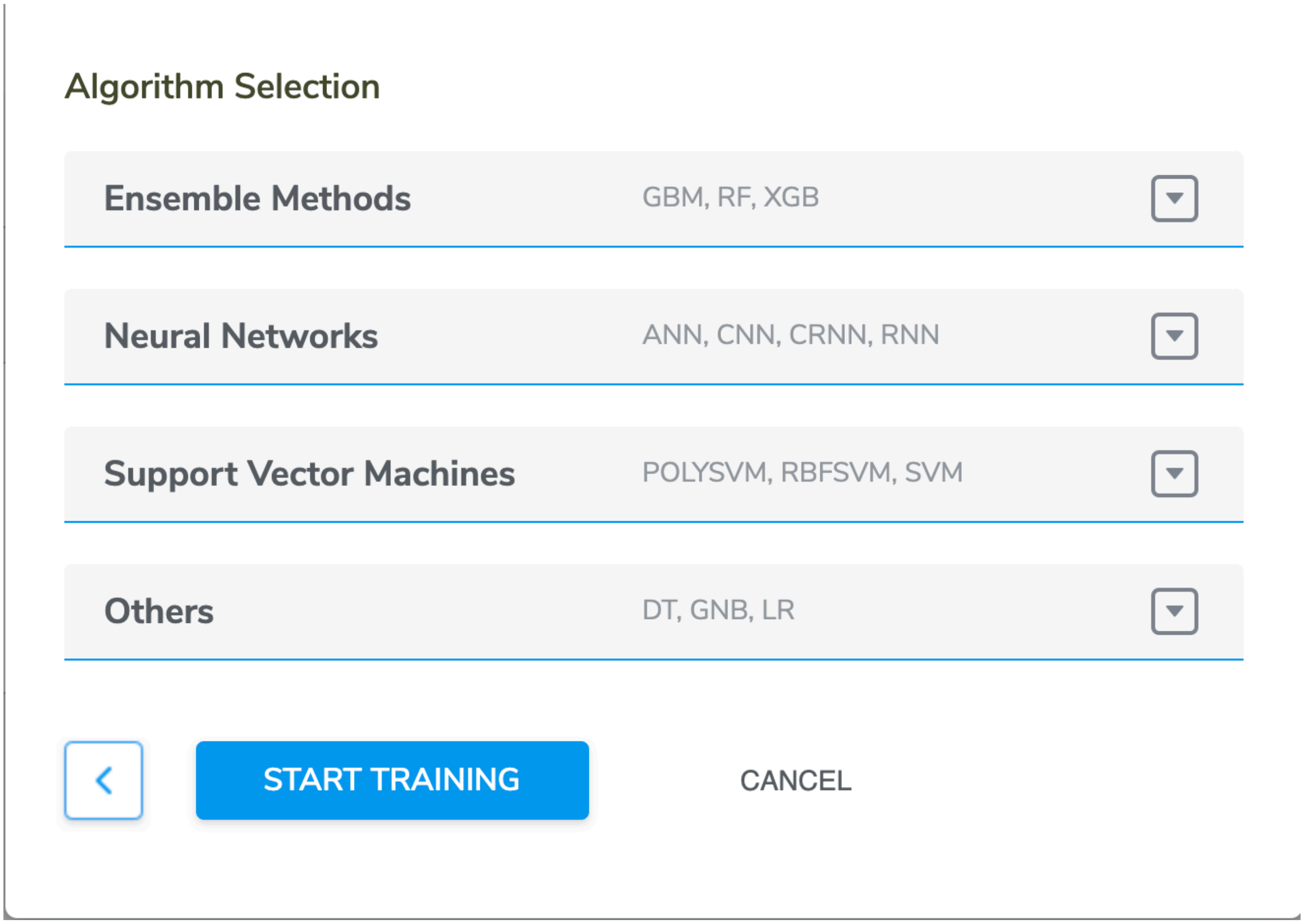
- Polynomial Support Vector Machine (POLYSVM)
- RBF Support Vector Machine (RBFSVM)
- Support Vector Machine (SVM)

Others:

- Decision Tree (DT)
- Gaussian Naive Bayes (GNB)
- Logistic Regression (LR)

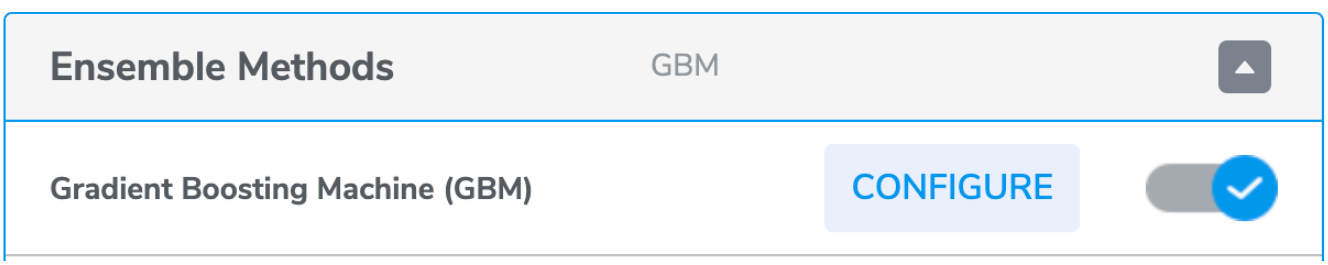
Support for additional algorithms will be added in the future.

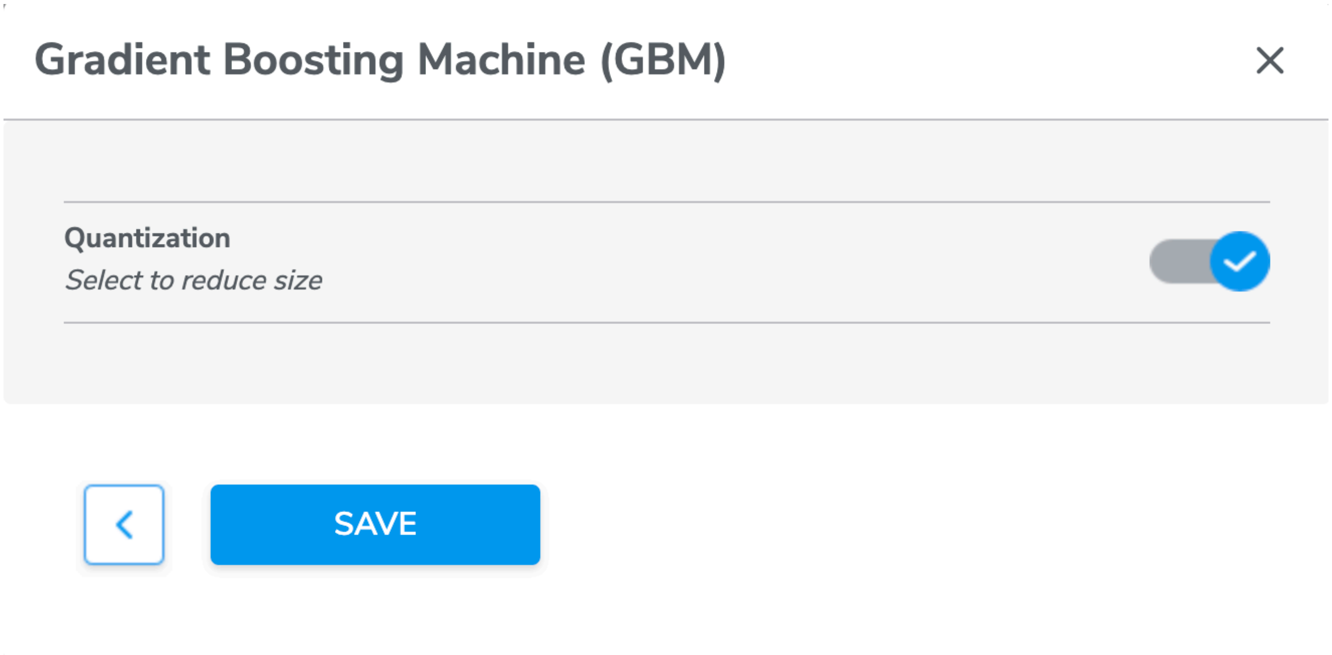
Note that Neural Networks models may take longer to train, due to the significant computation required for the training process.



CONFIGURE

Pressing CONFIGURE (available for some models) will yield the following configuration screen:





Quantization denotes an option to conduct quantization-aware training so as to achieve model size reduction.

There are additional configurable options to fine tune the ANN model:

Artificial Neural Network (ANN)

×

Quantization

Select to reduce size

Configurable Parameters

LEARNING RATE

0.01

Value from 0.00001 to 0.5

LAYER 1 UNITS

5

Value from 2 to 10

LAYER 2 UNITS

10

Value from 2 to 10

LAYER 3 UNITS

10

Value from 2 to 10

DROPOUT RATE

0.1

Value from 0.1 to 0.9

EPOCHS

50

Value from 1 to 1000

BATCH SIZE

20

Value from 10 to 200

ACTIVATION

ReLU

BATCH NORMALIZATION

False

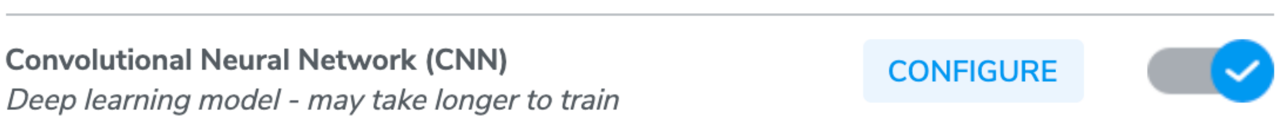
SAVE

CANCEL

Configurable Option	Description
Learning rate	Scaling parameter which sets the step size at each iteration in optimization of the cost function
Layer 1 units	Number of nodes in layer 1
Layer 2 units	Number of nodes in layer 2
Layer 3 units	Number of nodes in layer 3
Dropout rate	Fraction of units to drop during each training round, applied to all network layers
Epochs	Number of passes through the complete training dataset; one epoch means the network will use each training instance exactly once
Batch size	Number of training examples in one training round; higher batch sizes may have faster runtimes, but are more likely to get stuck in local optima
Activation	Function applied to the outputs of the neurons
Batch normalization	If true, apply normalization process to the output of each layer, typically helpful for improving the convergence and stability of the training process

Note: many of these parameters interact with each other in unique and non-intuitive ways. Unless you have significant experience tuning deep learning models, you may want to consider using the automatic hyperparameter optimization tool.

Similarly there are configurable options to fine tune the CNN model:



Configurable Option	Description
Tensor Length limit	Threshold length that determines whether to stop adding convolution layers (reducing the length limit will lead to more convolution layers)
Learning rate	Scaling parameter which sets the step size at each iteration in optimization of the cost function
Batch size	Number of training examples in one training round; higher batch sizes may have faster runtimes, but are more likely to get stuck in local optima
Dense layer units	Number of nodes in the final network layer
Dropout rates	Fraction of units to drop during each training round, applied to all network layers
Epochs	Number of passes through the complete training dataset; one epoch means the network will use each training instance exactly once
Input layer filters	Number of filters in the first convolution layer
Intermediate layers filters	Number of filters in all the intermediate convolution layers
Input layer strides	Number of samples to move at each step along one direction for the first convolution layer
Intermediate layers strides	Number of samples to move at each step along one direction for all intermediate convolution layers
Augment	If true, apply data augmentation technique to prevent overfitting; will lead to higher training time due to larger amount of data
Batch normalization	If true, apply normalization process to the output of each layer, typically helpful for improving the convergence and stability of the training process
Activation	Function applied to the outputs of the neurons
Input layer kernel size	Filter kernel size in the first convolution layer
Intermediate kernel size	Filter kernel size in the all intermediate convolution layers

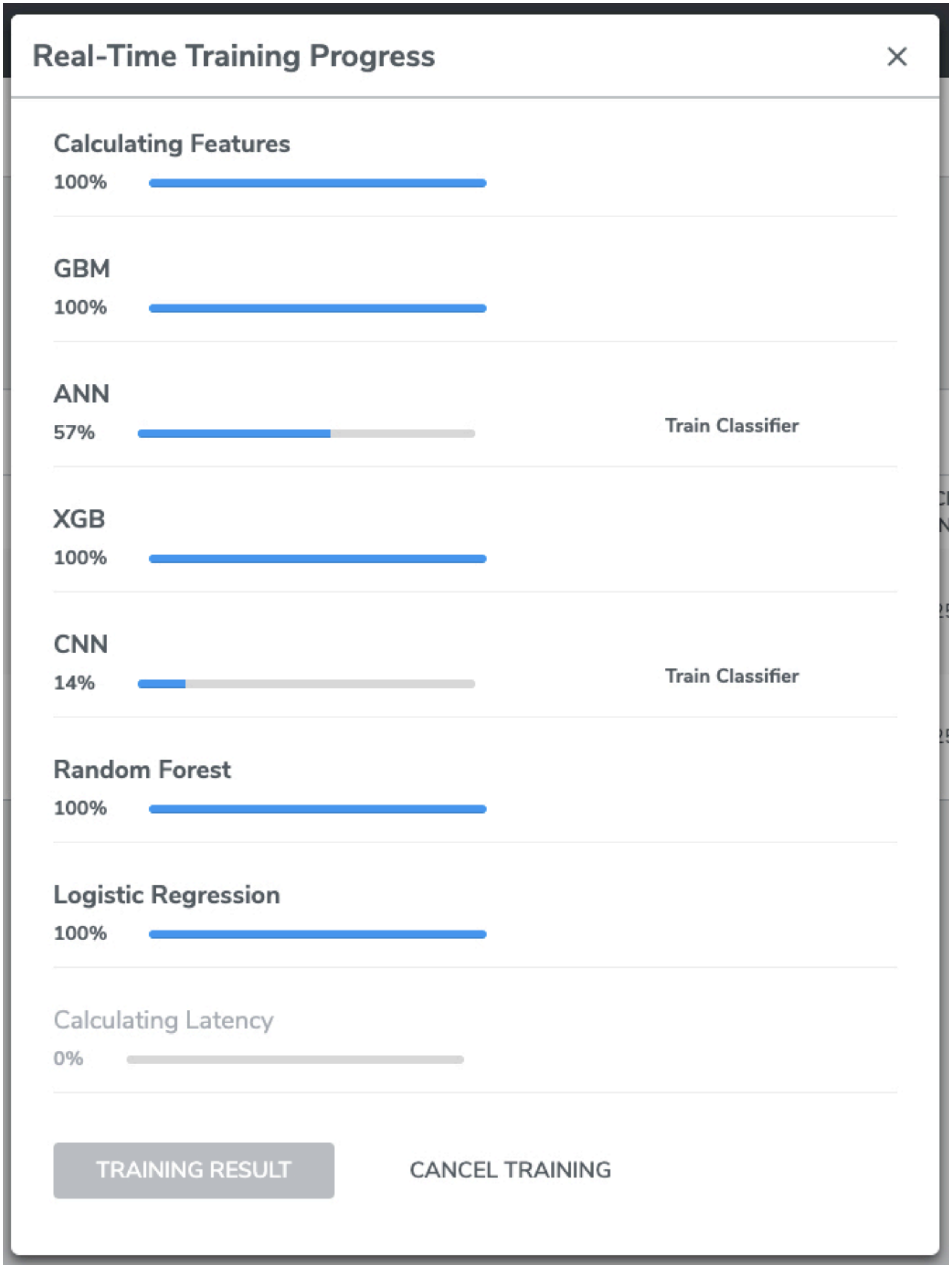
Configuration sub-menu for other algorithms will be added in the future.

Training Process

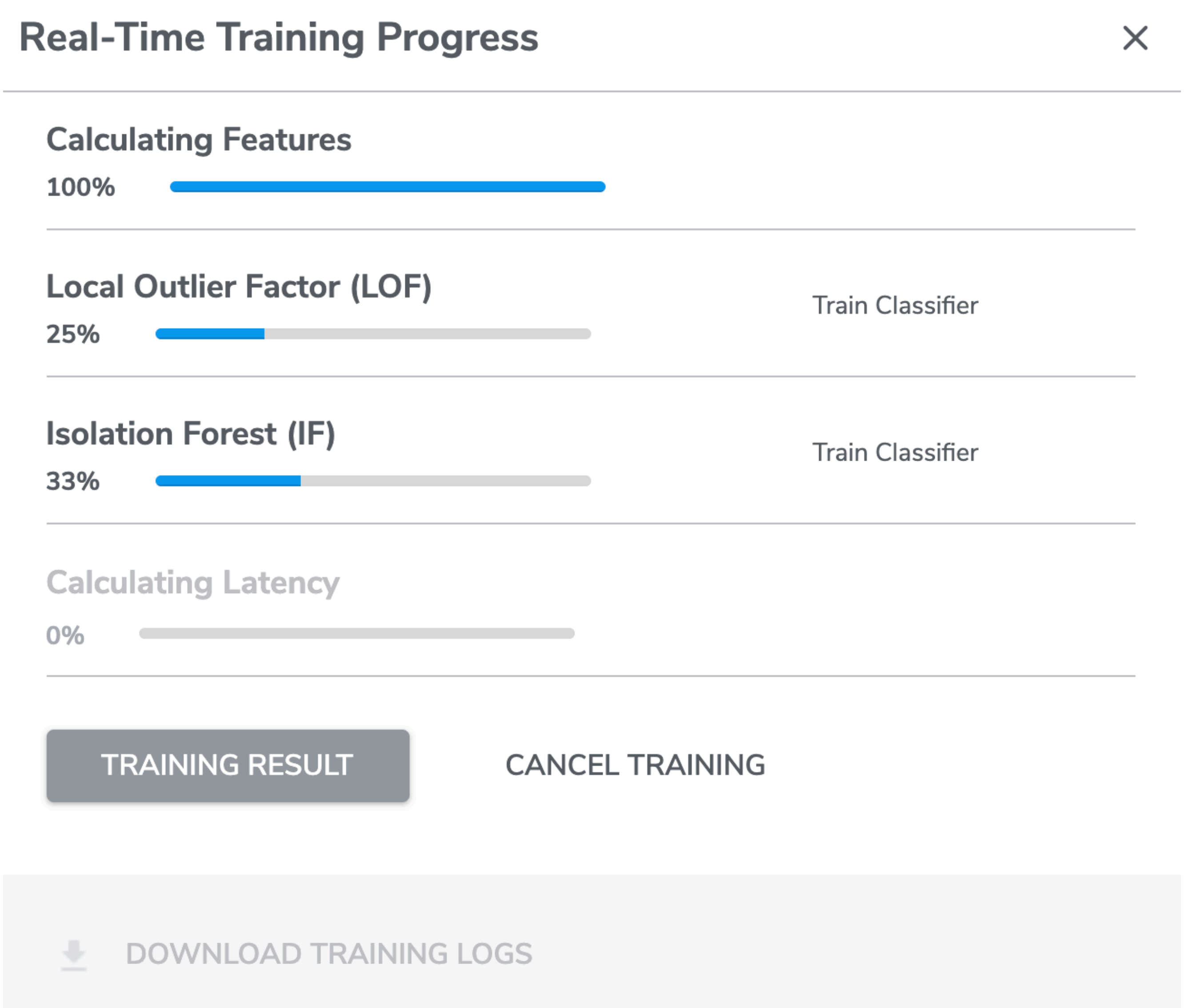
Click "Start Training" with one or more selected machine learning algorithms to begin the training process. Selecting more than one type of algorithm is recommended, so that results could be compared.

Real-Time Training Progress pops up after training begins. The top row shows the progress of common tasks (e.g. featurization, data cropping, etc.) shared between different algorithms, followed by the build progress of each of the selected models.

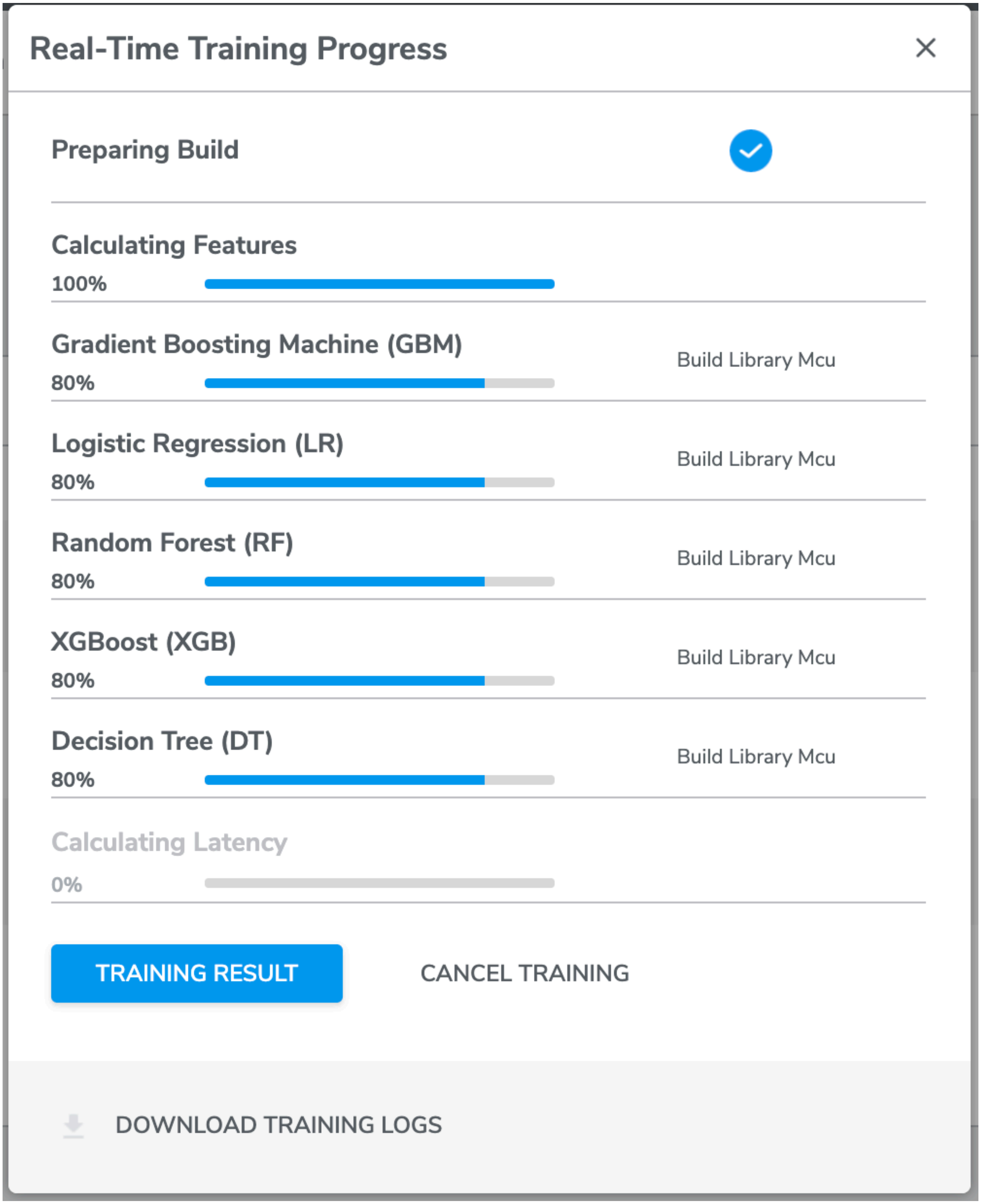
Multi-class classification:



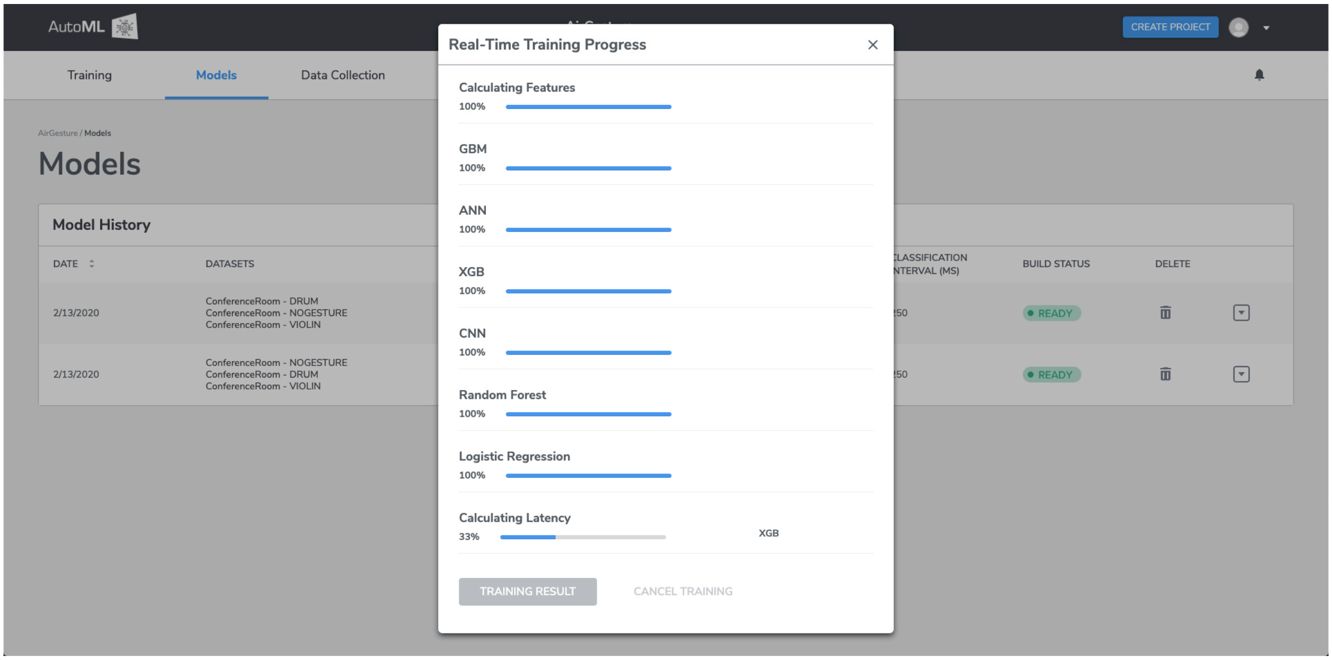
Single-class classification:



Multi-Class Anomaly Classification:

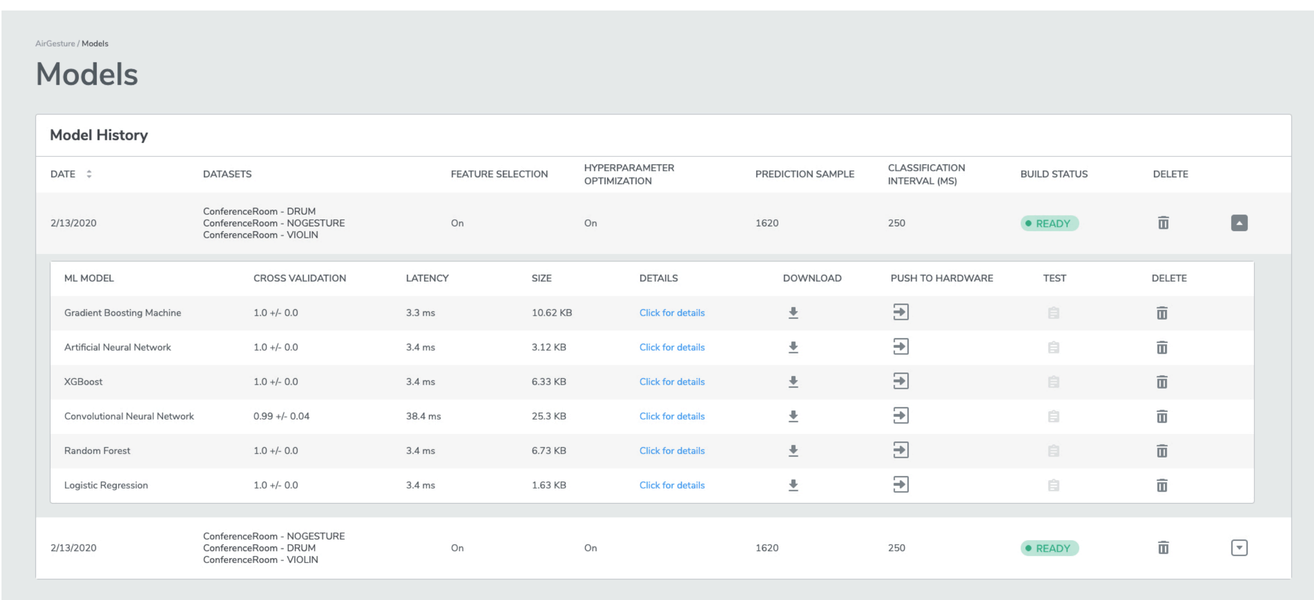


At the end of the training process, Qeexo AutoML will flash, in sequence, each of the built models to the hardware device to test and measure the average latency for performing classifications.



Training Result

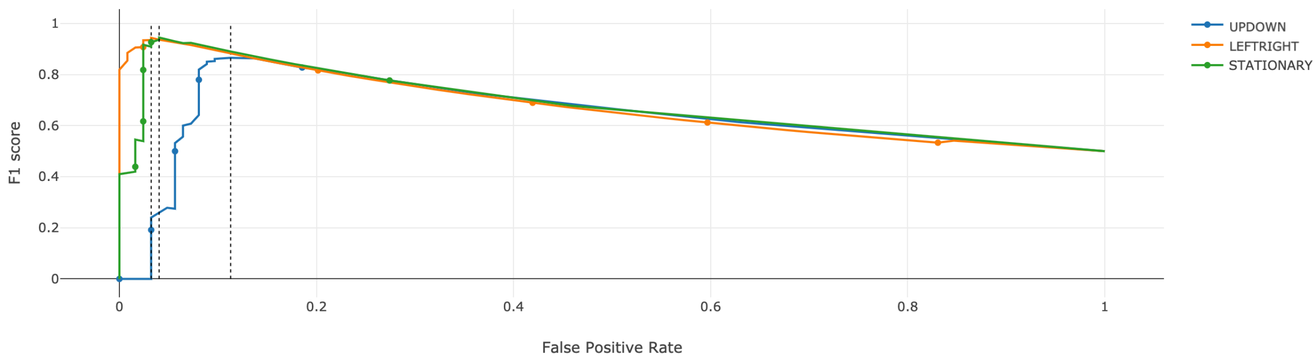
Click "Training Result" to navigate to the Models page (also reachable from the top navigation bar), where all of the previous trainings will be listed, with the most recent one on top. The current training will be expanded to show relevant information, including the type of machine learning model, cross validation accuracy, latency, size, and additional details. It also allows you to save each model to your computer or push a selected model to Target Hardware for Live Testing.



ML Model Each entry is differentiated by the algorithm with which each model had been built. We also call these machine learning "packages" because they include supporting code such as sensor drivers in addition to the machine learning models built by Qeexo AutoML.

Cross Validation This is the average classification accuracy for 8 different models, each trained and tested on different, mutually-exclusive subsets of the given data. This is always a value between 0 and 1, with 0 being the worst accuracy and 1 being perfect accuracy.

F1-Score F1-score also lies within the unit interval; the best score is 1, and it approaches 0 as performance gets worse. F1-score factors false positives, false negatives, and true positives. F1-score thus is an important model performance metric. Accuracy only can obscure some important aspects of model performance if a large proportion of a dataset belongs to one class. In contrast, F1 score is more tolerant to this type of class imbalance problems. Operating the model at the peak of the F1-score means the rate of True Positives and False positives are optimized. To the either side of this point, either True positives or False positives dominates.



Latency Latency is the average time (in milliseconds) required for the machine learning model to compute the prediction of a single instance. It includes time spent on featurization of sensor data and running inference with the model. We calculate this average empirically by first flashing each model to the Target Hardware, running 10 inferences, then taking the average.

Note that the concept of latency is not applicable to MLC projects.

Size This is the memory size of the model parameters and the model interpreter. The model interpreter executes the model parameters in combination with the sensor readings to provide the model results. This measure gives an idea of the impact of this model to on-device memory usage in comparison to other models trained on the same data. Consider the following notes to understand the ML Model Size measurement: - None of the model size measurements include the sensor data processing and featurization code. That can add 10KB – 20KB to the size of the final library for all models except raw-data-based models (CNN, CRNN, RNN) - The static library output from AutoML may be larger than the size reported due to the featurization code as well as other necessary interface utilities - The binary used for flashing to the device for Live Testing from AutoML will be significantly larger as it also must include other system libraries for the target platform - The concept of memory size with MLC projects does not apply as the decision tree is implemented in hardware

Details Press "Click for details" to bring up a pop-up window with additional information about each of the machine learning models. Note that the amount of model details depends on whether the project is for single-class or multi-class classification.

Multi-class classification

UMAP and PCA Plots

In model details we are showing the dimensionality reduction UMAP and PCA plots as visual indications of how the training datasets are "clustered" in the given model.

UMAP Plot

PCA Plot

Confusion Matrix Matrix representation of True Labels and Predicted Labels. Diagonal (upper left to lower right) elements indicates instances correctly classified. Off-diagonal elements indicate instances mis-classified. Summing instances over each row should sum to total instances for the respective class. For Multi-Class Anomaly Classification, there will be an extra unknown class label.

Cross Validation: By-fold Accuracies vs Classes Visual representation of the spread of classification accuracies across the CV folds. This representation is done by-class. If the by-fold points are all shown close to the mean line, this shows that the average by-class accuracy is a precise measurement of how well the model should perform for the given class. More variance in the by-fold points suggests that the model may perform much better or much worse than expected.

Learning Curves Learning Curves illustrate the performance for each class at different number of instances of data collected/uploaded. Each point on the Learning Curve is the cross-validation accuracy at the respective data size. This gives an understanding of whether adding more data will help to improve the classification performance for each class and whether similar performance can be achieved with fewer instances of data.

RoC Curves RoC Curves plot the False Positive Rate (FPR, x-axis) vs. True Positive Rate (TPR, y-axis) for each class in the classification problem. The dotted line indicates flip-of-the coin performance where the model has no discriminative capacity to distinguish between 2 classes. The greater the area under the curve (AUC), the better the model.

Matthew's Correlation Coefficient The Matthew's Correlation Coefficient (MCC) is a measure of discriminative power for binary classifiers. In the multi-class classification case, it can help show you which combinations of classes are the least well understood by your model. The values can range between -1 and 1, although most often in AutoML the values will be between 0 and 1. A value of 0 means that your model is not able to distinguish between the given pair of classes at all, and a value of 1 means that your model can perfectly make this distinction. For Multi-Class Anomaly Classification, there will be an extra unknown class label.

There will be one MCC value for every pair of class labels in the datasets (order does not matter). For example, there will be 3 coefficients for each combination of the 3 class labels, and 6 coefficients for 4 class labels.

Single-class classification

Confusion Matrix For the single-class classification case, we only have data from the one given class. A perfect confusion matrix for single-class models has all of the cases concentrated in the top-left corner, meaning that none of the given class data was classified as not coming from that class.

Cross Validation: By-fold Accuracies vs Classes Similar to the confusion matrix case, the most important information in the single-class classification by-fold results are the left-most case. This will show us how varied our single-class accuracies were for each fold of our cross validation.

Matthew's Correlation Coefficient For single-class classification, there is only one Matthew's Correlation Coefficient, which measures the quality of the classification between the given class and things that do not belong to the given class. The values can range between -1 and 1, although most often in AutoML the values will be between 0 and 1. A value of 0 means that your model is not able to recognize the given class at all, and a value of 1 means that your model can perfectly make this distinction.

Sensitivity Analysis Trade off model accuracy between classes to find a balanced performance that is right for your use case.

Save For non-MLC projects, there are 2 options

- "Save .bin" - download the model as a binary image to your machine.
- "Save .zip" - download a compressed archive of header file and static library whereby users can build custom application on top of the machine learning model.

For MLC projects, users can ONLY save the MLC configuration of the model as `json` file.

Flash to Hardware Flashes the model to the Target Hardware. Target Hardware must be connected.

Live Test Once the model has been flashed to Target Hardware, "Test" becomes clickable, and will take you to Live Classification screen.

Delete When a model is no longer required, you can delete it. A confirmation dialog box will be presented.

Live Classification

Here you can perform live-testing. The screen will display the current class that is predicted by the model that was flashed to the Target Hardware, based on the signals from the enabled sensors for this Project.

Classification Methods

Continuous Classification

Continuous classification is selected on the live test page as default. In continuous classification, the live test screen will display the classification result one after another in a rate you set previously in classification interval. Continuous classification is recommended when the training sets are all continuous data.

MLC Meta Classifier

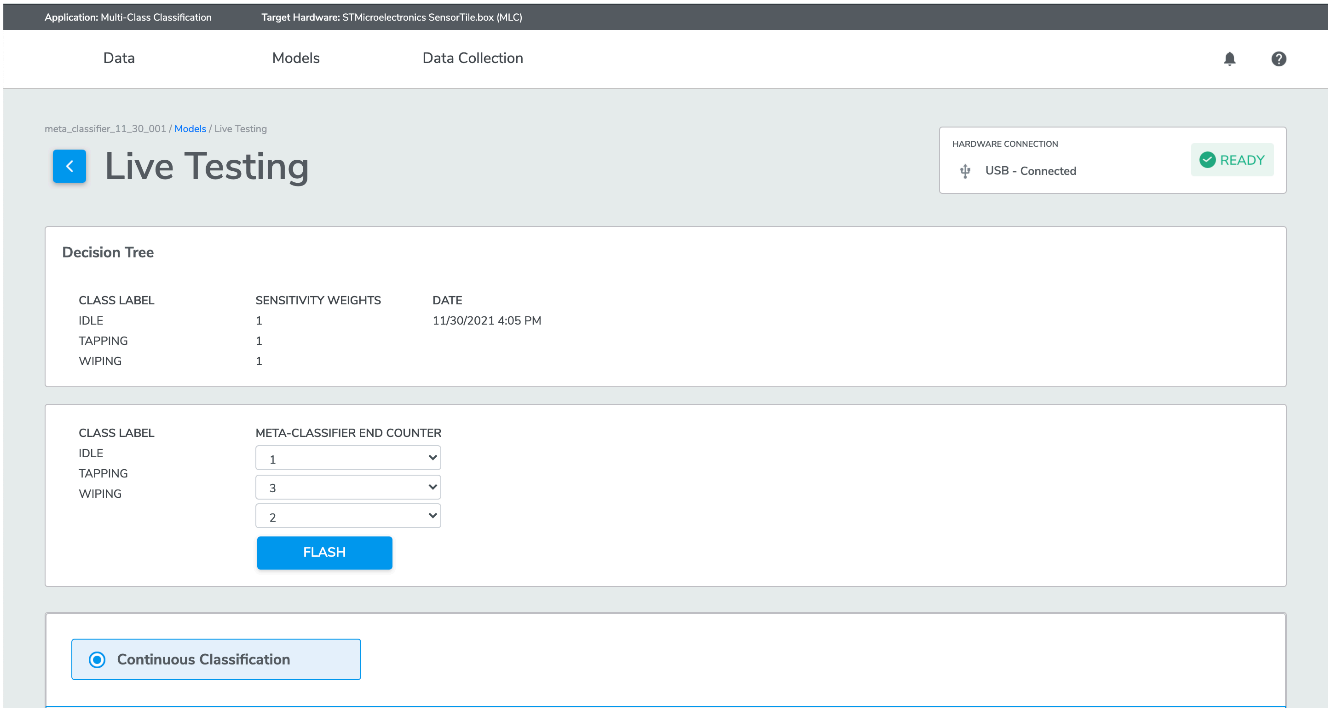
The following applies to MLC projects only.

A meta-classifier is a filter on the outputs of the decision tree. The meta-classifier uses some internal counters to filter the decision tree outputs. The purpose of the meta-classifier is to reduce false positives, avoid generating an output which is not stable, and reduce the transitions on the decision tree result.

Table 8. **Meta-classifier example**

Decision tree result	A	A	A	B	A	B	B	B	A	B	B	B	A	A	A
Counter A (End counter = 3)	1	2	**3**	2	3	2	1	0	1	0	0	0	1	2	**3**
Counter B (End counter = 4)	0	0	0	1	0	1	2	3	2	3	**4**	5	4	3	2
Machine Learning Core result (including meta-classifier)	x	x	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**B**	**B**	**B**	**B**	**A**

The previous table shows the effect of filtering the decision tree outputs through a meta-classifier. The first line of the table contains the outputs of the decision tree before the meta-classifier. Counter A and Counter B are the internal counters for the two decision tree results (“A” and “B”). In the activity recognition example, the result “A” might be walking and the result “B” jogging. When the internal counter “A” reaches the value 3 (which is the End Counter for counter “A”), there is a transition to result “A”. When the internal counter “B” reaches value 4, there is a transition to result “B”.



The user can read/edit the meta-classifier end counters in the live testing page of MLC projects. The default values of end counters after flashing MLC model are 0.

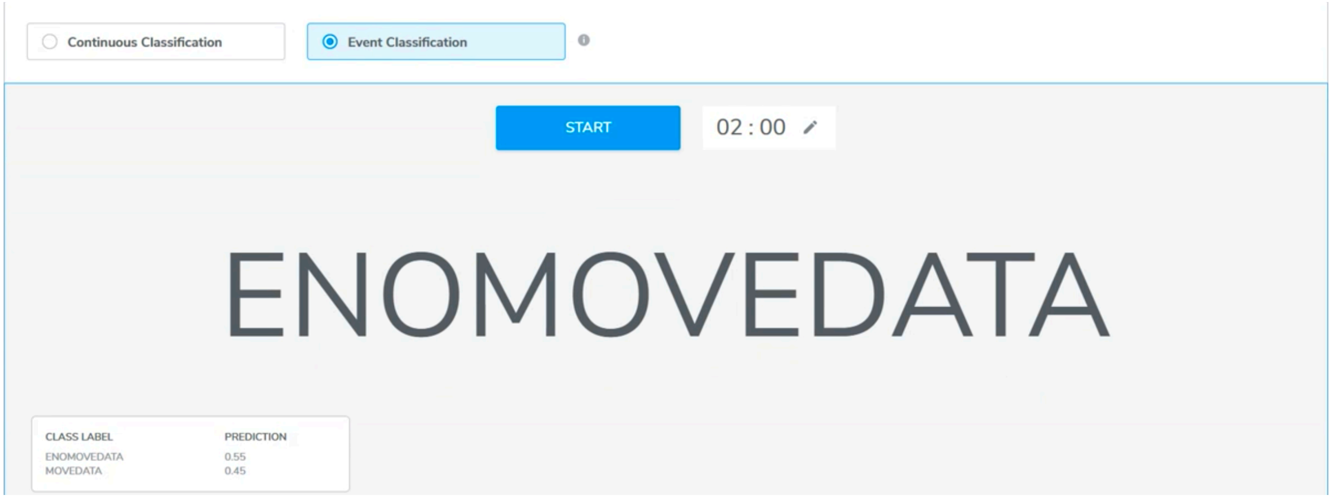
The user can update the meta-classifier end counters in the live testing page within acceptable range (0-14). if the users have increased the end counters of a class label, it will take “longer” for MLC to output the classification result for that label, compared to before the change. Say the user increased the end counter for class “drum” from 1 to 5. It will now take more occurrences of “drum” output to reach to 5 than 1, resulting in potentially “longer time” between classification output changes, thus serving the false positive reduction objective. Setting what end counter values are subject to human judgement, depending on ML use cases.

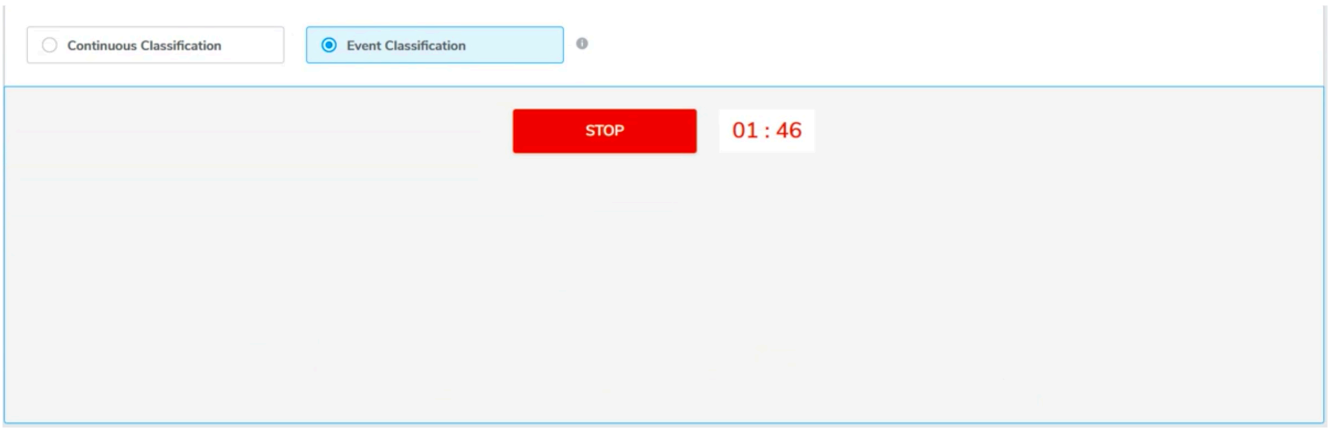
Event Classification (Start/Stop)

In event classification, the model will only read the sensor input within the event window, and give one prediction after each event.

Event classification is recommended when the training set contains segments.

Using GUI button: Press START and begin performa the event. The prediction result will display on the screen after the event ends by either pressing STOP or the n second of event length is up.





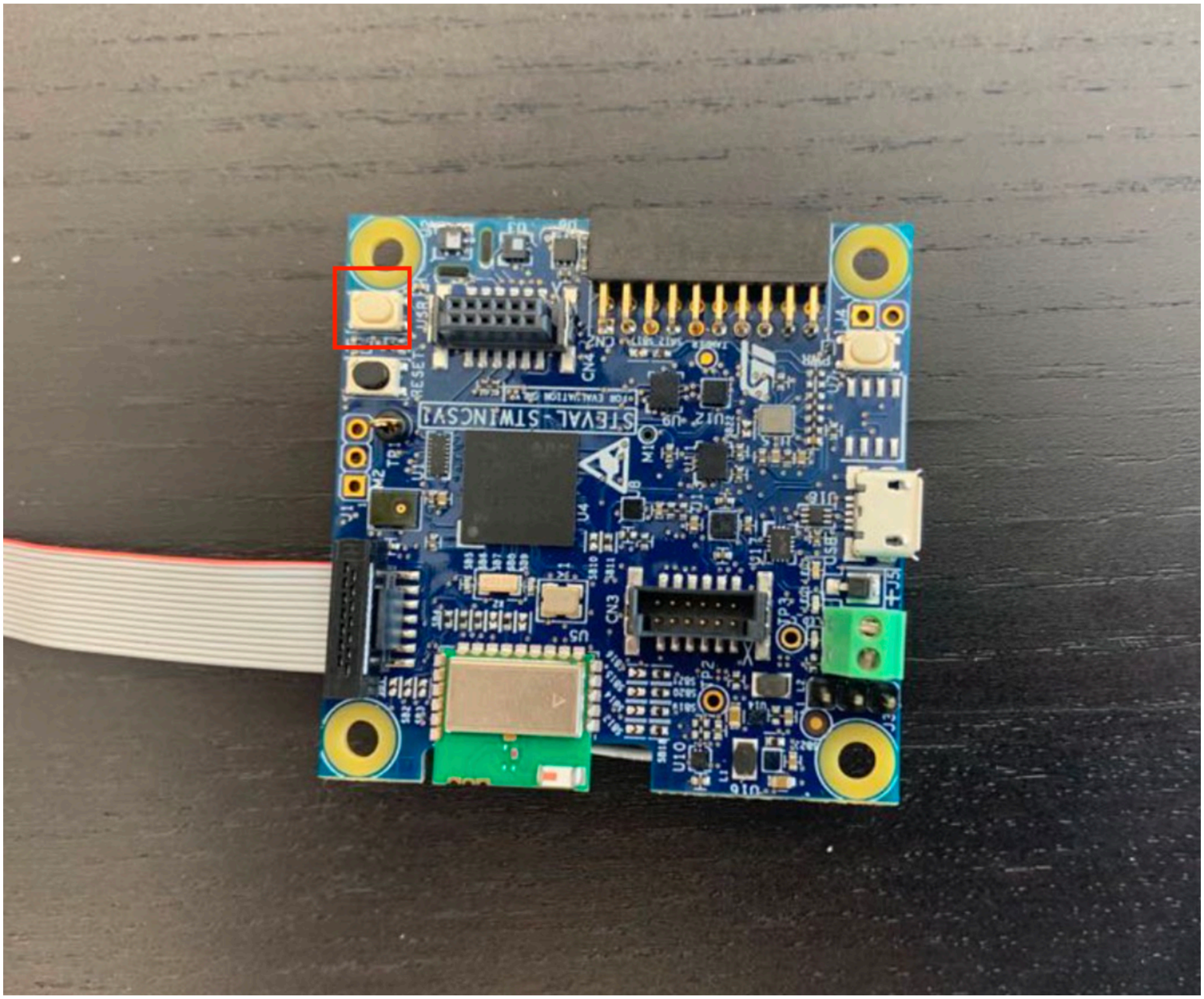
Using hardware button:

AutoML also supports event classification using hardware button for some target hardware. Press the button* on the device to begin the event. The prediction result will display on the screen after the event ends either by releasing the button or the n second of event length is up.

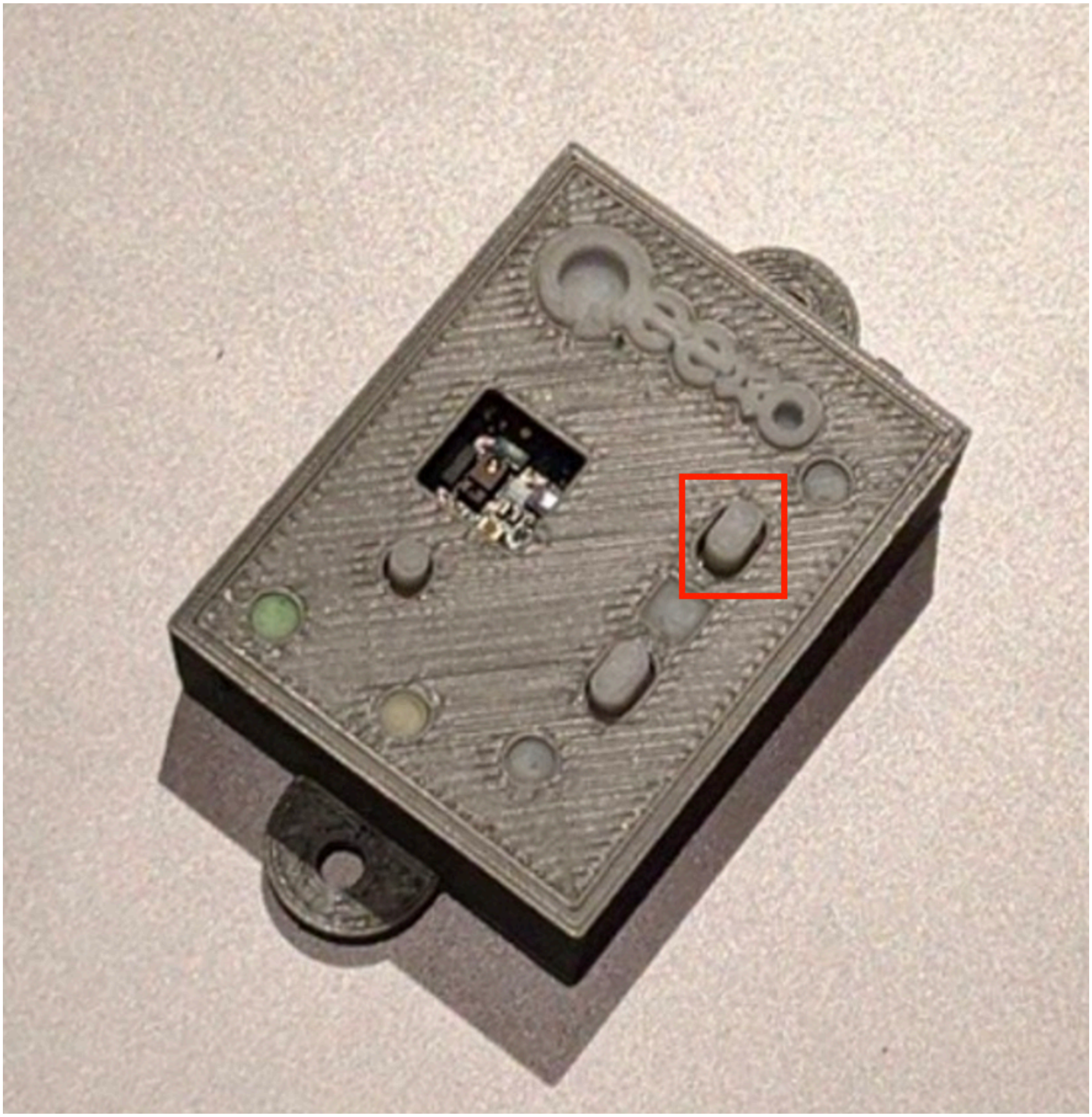
*Current target hardware that supports hardware event classification:

ST SensorTile.Box:

STWINKT1:



Arduino BLE sense and Arduino IoT sense using Qeexo custom case:

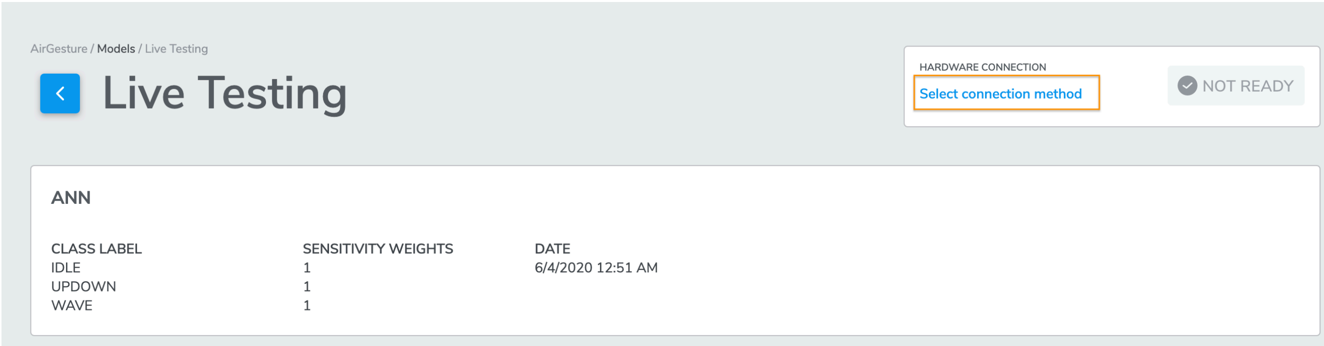


[Contact us](#) and get your own Arduino battery case.

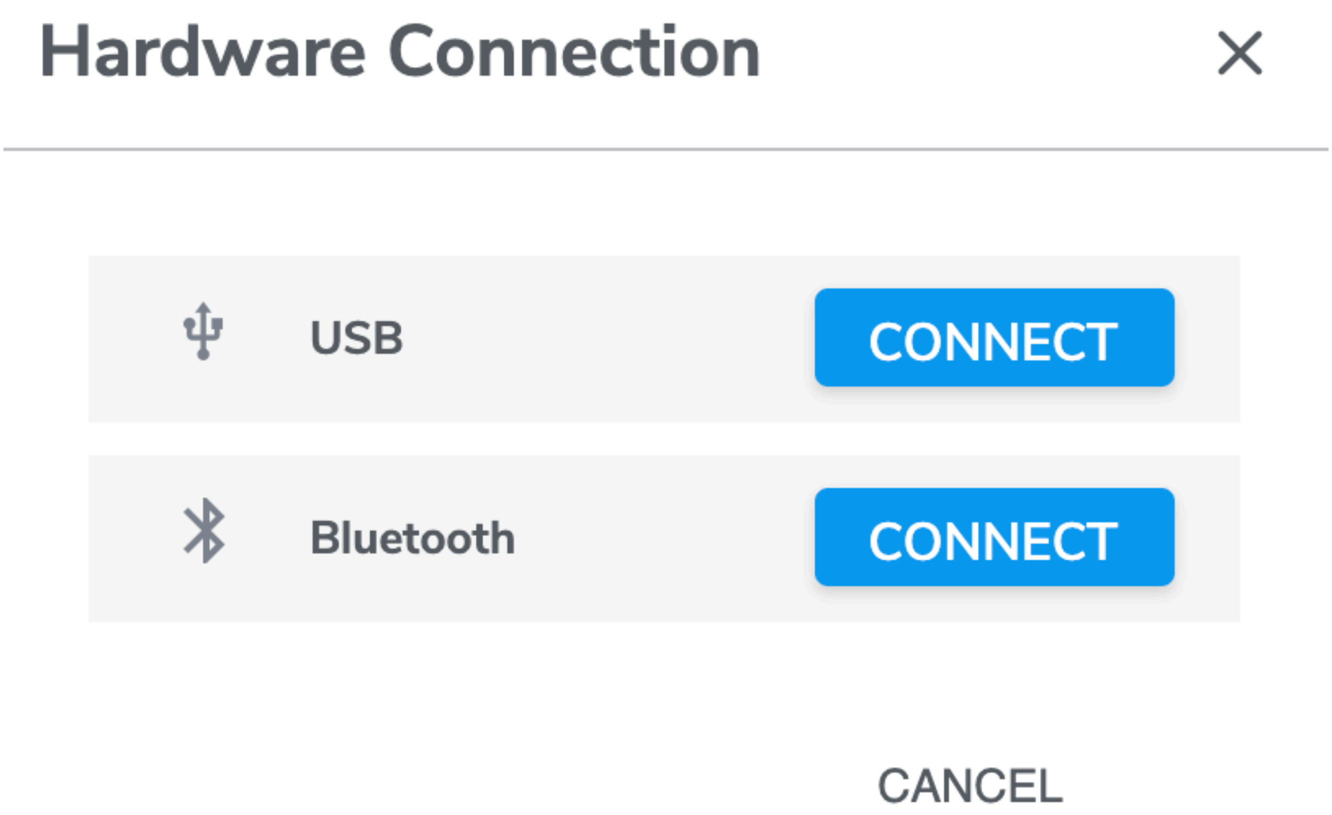
Connection Methods

Choose between USB or Bluetooth

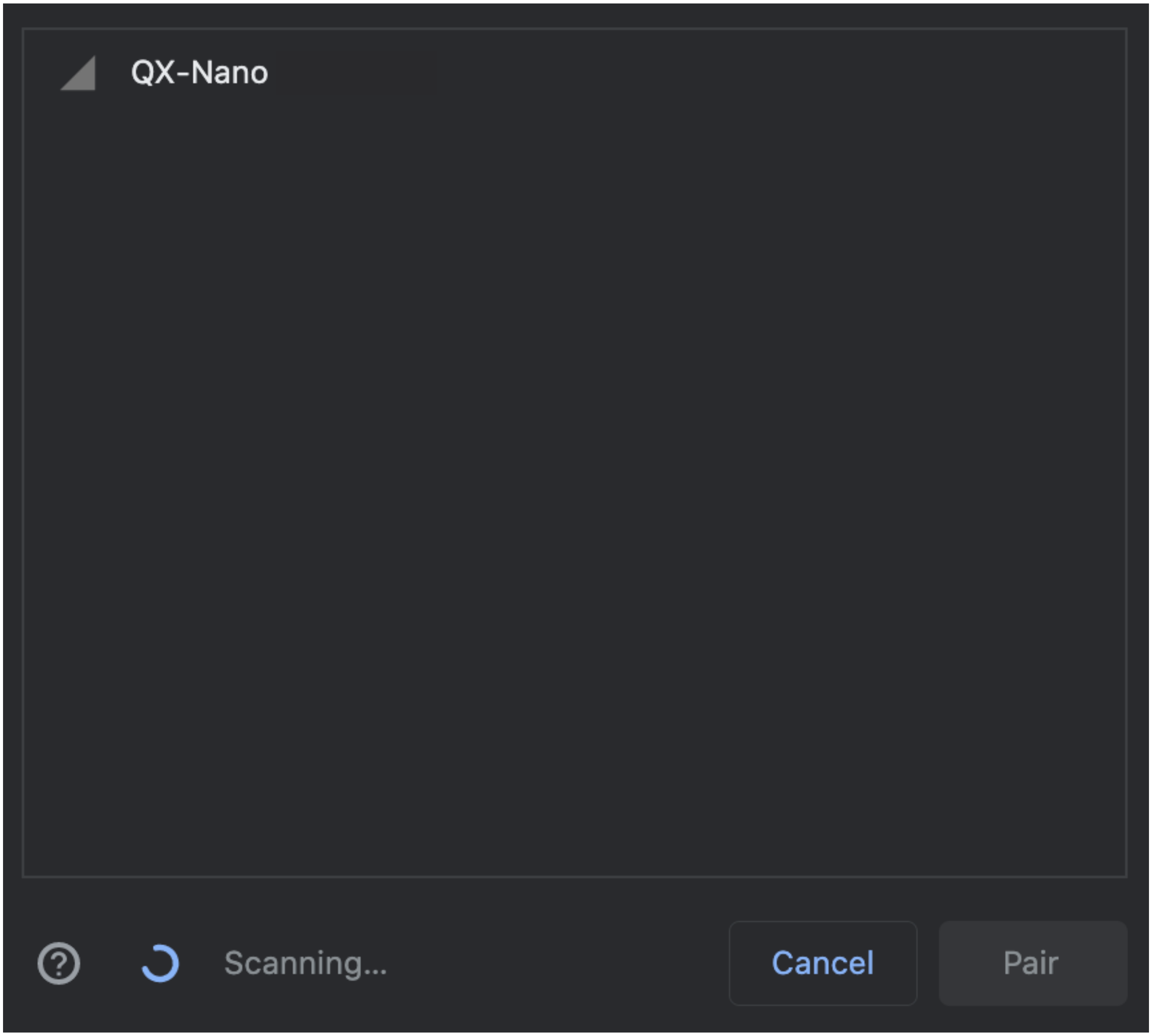
If your hardware supports more than one connection method, select the option which works for your use case:



AutoML supports both USB and Bluetooth for most Target Hardware.



When you press CONNECT for Bluetooth, a screen similar to the following will be presented:



There may be more than one device ready to connect. If so, please choose the one starting with "QX - Nano" to pair with your device.

When you press CONNECT for USB, Live Classification will be conducted via the connected USB cable and no further connection action is required.

For Single-class classification, USB is the only connection method. This is due to limitations for the Manual Calibration option (described below).

Reading Inference from Hardware

Multi-class Classification Project

The machine learning model during live classification outputs two pieces of information: the current predicted class label and the current model output values (called "Probability" in the web app) for each class.

The current predicted class label is based on the class with the highest model output value. Some weighting may also be applied for certain problems, so there may appear to be a minor delay between the maximum class "Probability" and the predicted class label.

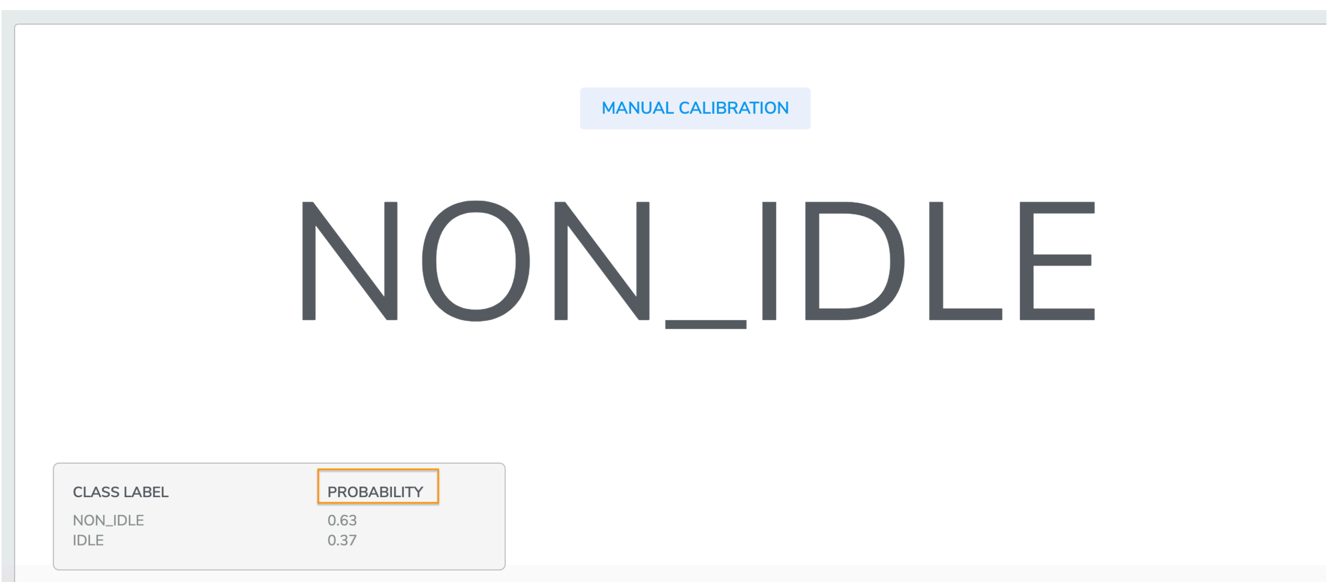
UPDOWN

CLASS LABEL	PROBABILITY
UPDOWN	0.46
WAVE	0.32
IDLE	0.22

The probability table of all class labels are presented as reference.

Single-class Classification Project

For single-class classification, the machine learning model outputs the same two pieces of information. In this case, there will always only be two classes displayed: the given class and the anomaly (i.e. "NON_*") class.



After your single-class model is trained, it may be beneficial to calibrate the **threshold** of your model to tailor-fit your use case and application scenario. Begin the calibration process by clicking "MANUAL CALIBRATION".

Scores for single-class classifiers are in the range (0, 1], i.e., 0 < Score <= 1. Whether the given instance belongs to the given class or the anomaly class is determined by thresholding the Score, as shown below:

```
if Score < threshold:
    Signal is Normal
else:
    Signal is Abnormal (i.e. Anomaly)
```

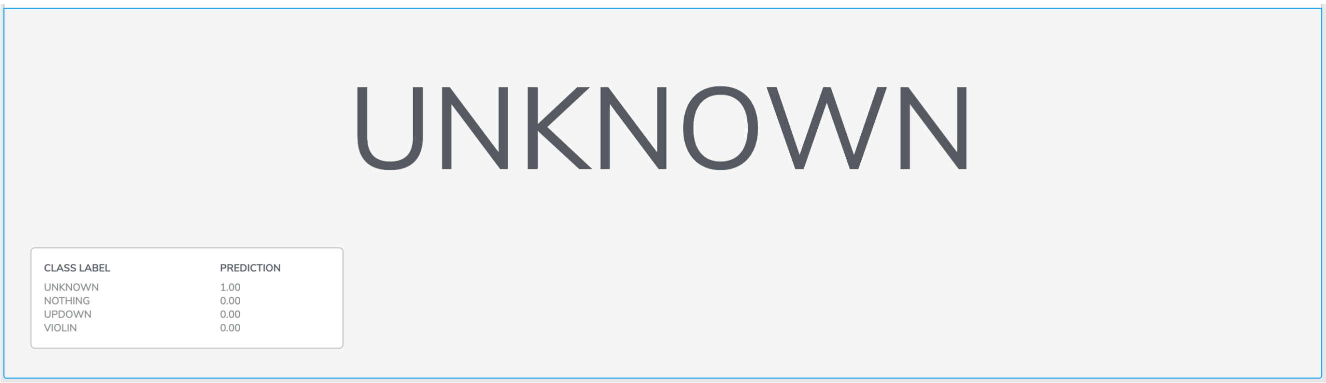
In the Manual Calibration menu, you can see scores (the blue line) generated by the single-class model plotted over time. Qeexo AutoML recommends an initial threshold based on an analysis of the training data, which is shown by the dotted red line. Users can change the threshold and flash the new, manually-selected threshold to the embedded target. The most recent user-selected threshold is shown in the plot by the dotted green line.

General rules of thumb for threshold tuning is the following:

- If you want to be very sensitive to potential anomalies, you should set the threshold to be lower. That means that even small variations in the sensor data away from the collected operating condition would trigger an anomaly. However, please beware that setting the threshold too close to zero may result in too many false positives.
- Likewise, if you only want to detect obviously anomalous data, you should keep the threshold to be higher. That means that minor variations in the sensor data away from the collected operating condition may not trigger anomalies.
- Please find right balance by doing live classification experiments through live classification after flashing a range of thresholds to the embedded device.

Multi-Class Anomaly Classification

The machine learning model during live classification outputs two pieces of information: the current predicted class label and the current model output values (called "Prediction" in the web app) for each class or the unknown class (If the new datapoint is sufficiently different from any of the training classes). The current predicted class label is based on the class with the highest model output value. In other word, the output can be any of the classes you previously collected just like the Multi-Class classification AND the anomaly (Unknown) class.



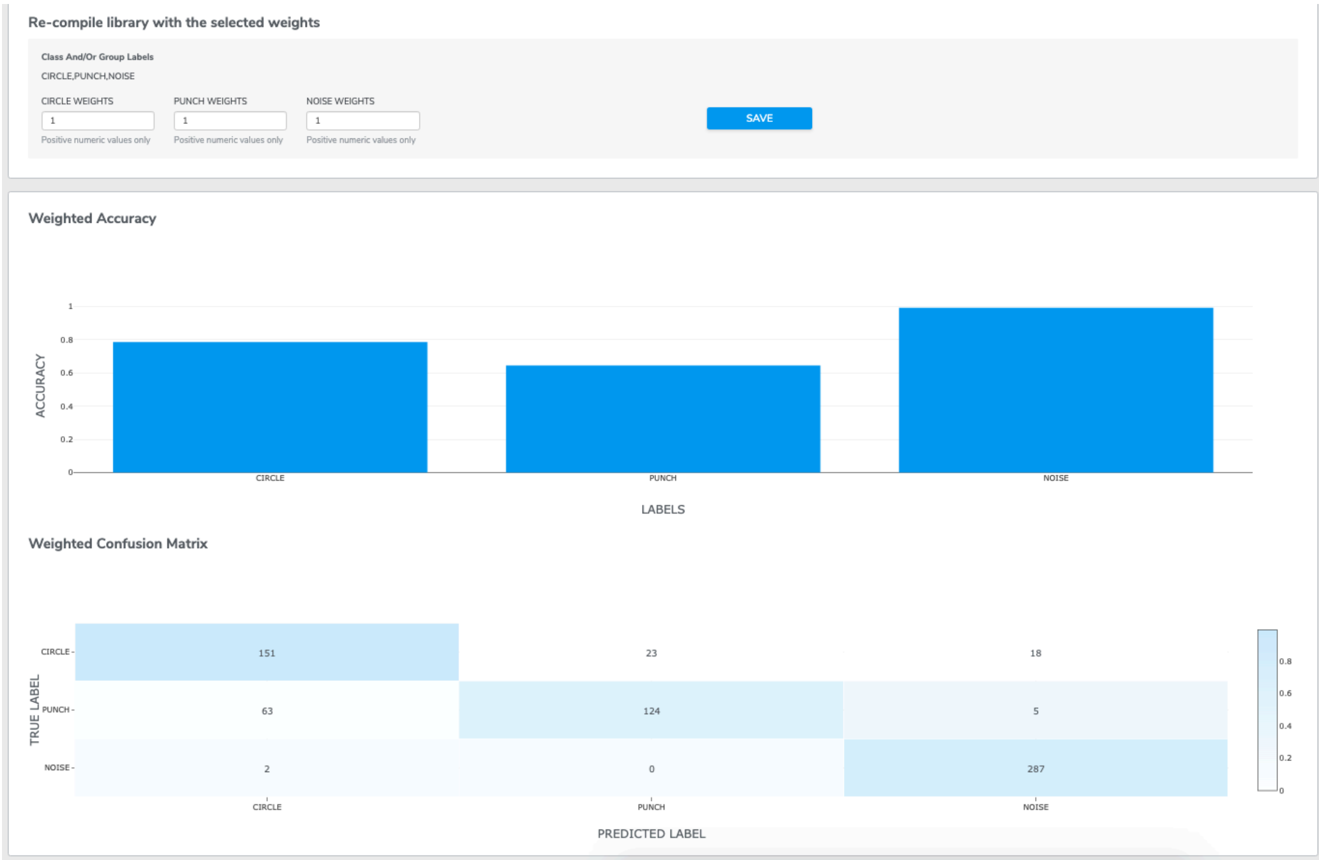
The probability table of all class labels are presented as reference

Live Classification Analysis

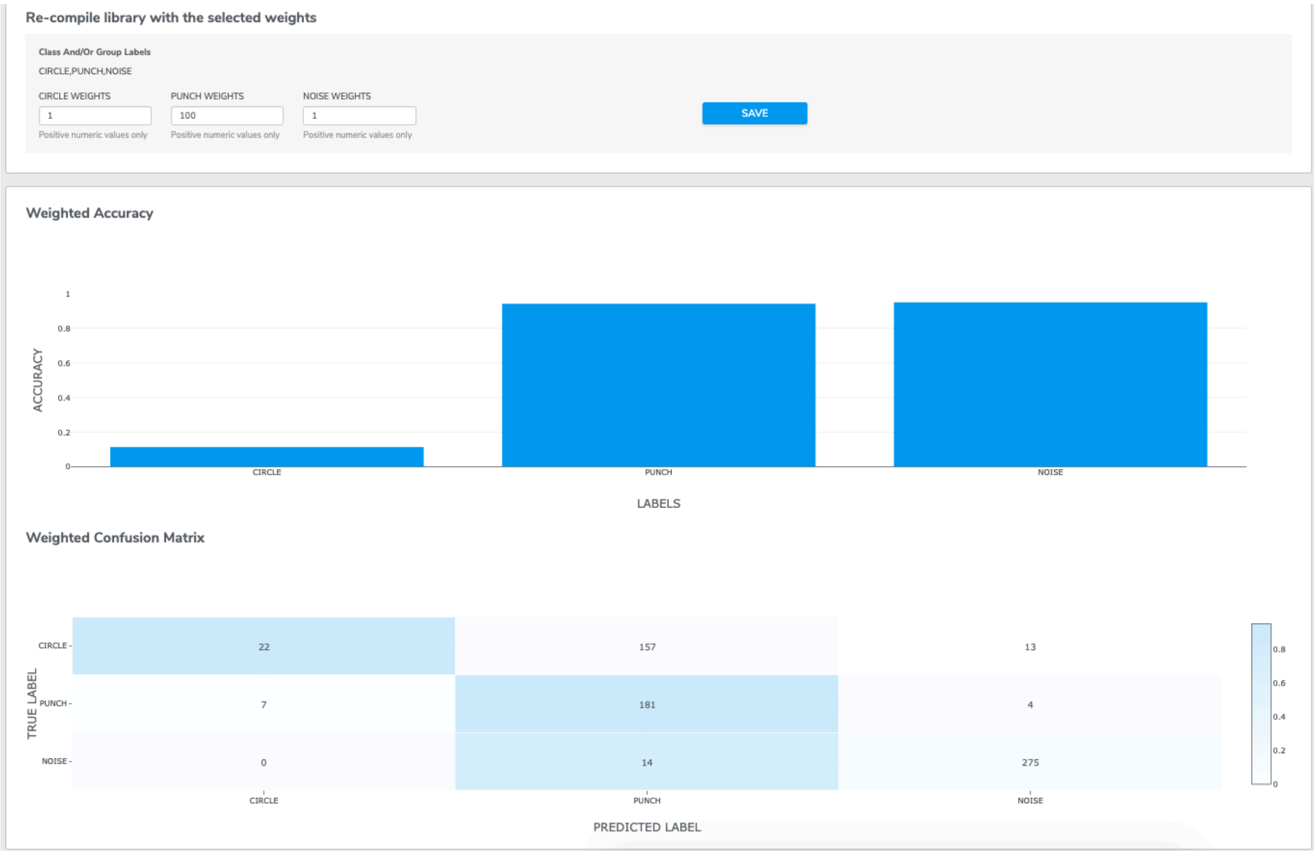
Sensitivity Analysis

For multi-class classification, the Sensitivity Analysis tool allows you to trade off accuracy between classes, depending on your specific use-case. You can re-weight the classes in your model and see how the cross-validation accuracies and confusion matrix is affected.

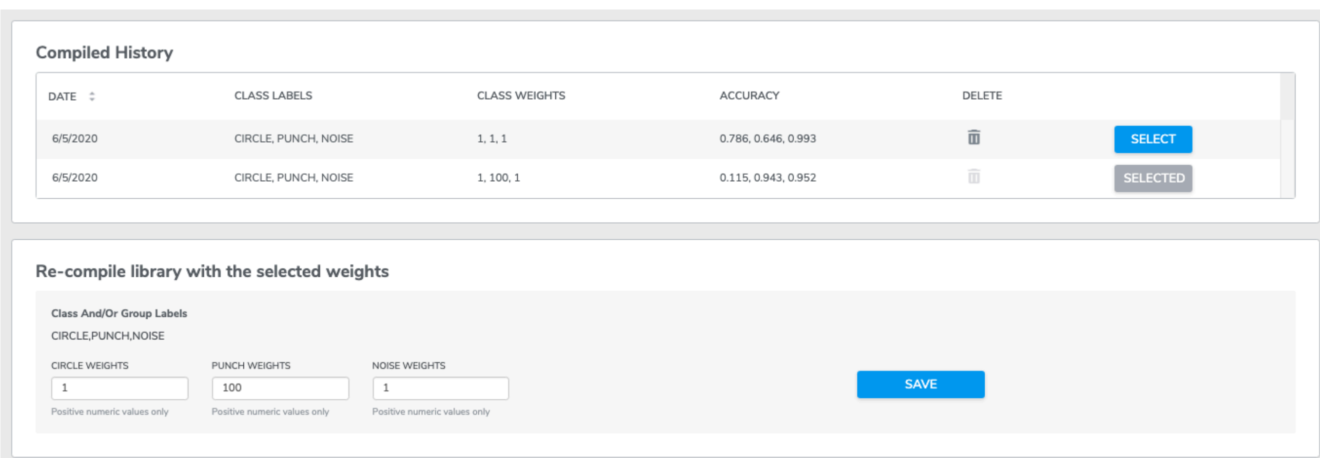
The selected sensitivities are normalized and are used to scale the model output probabilities. Higher values for a given class will make the model more likely to ultimately make a classification of that type.



The easiest way to understand how the Sensitivity Analysis page works is to train a multi-class model and then try a few different values. The accuracy plots and confusion matrix will update in real-time along with your changes to the sensitivities. Notice how the plots change when the sensitivity for the "Punch" class is increased from 1 to 100:



Once you find sensitivity values that seem best for your use-case, press "Save" on the new sensitivity values. This will generate a new binary with your selected values. Click "Select" on the newly-compiled binary, and this updated binary will be the one that is flashed to your device when you go to test live classification.

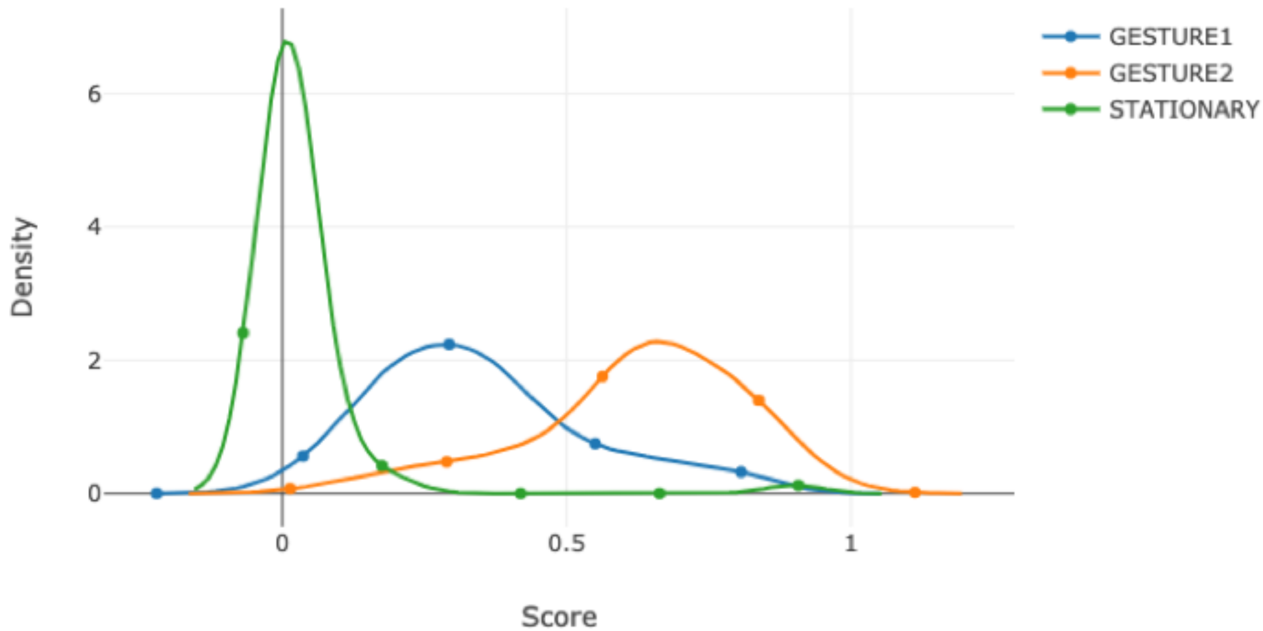


Note that there is no sensitivity analysis for single-class projects.

Live-Data Collection and Analysis

Live-data collection allows users to collect the live data for specific duration, class-by-class and then the subsequent analysis section shows a confusion matrix, ROC curves, Matthews correlation coefficient, and F1 score. Moreover, AutoML estimates the distribution of prediction scores using kernel density estimation (KDE).

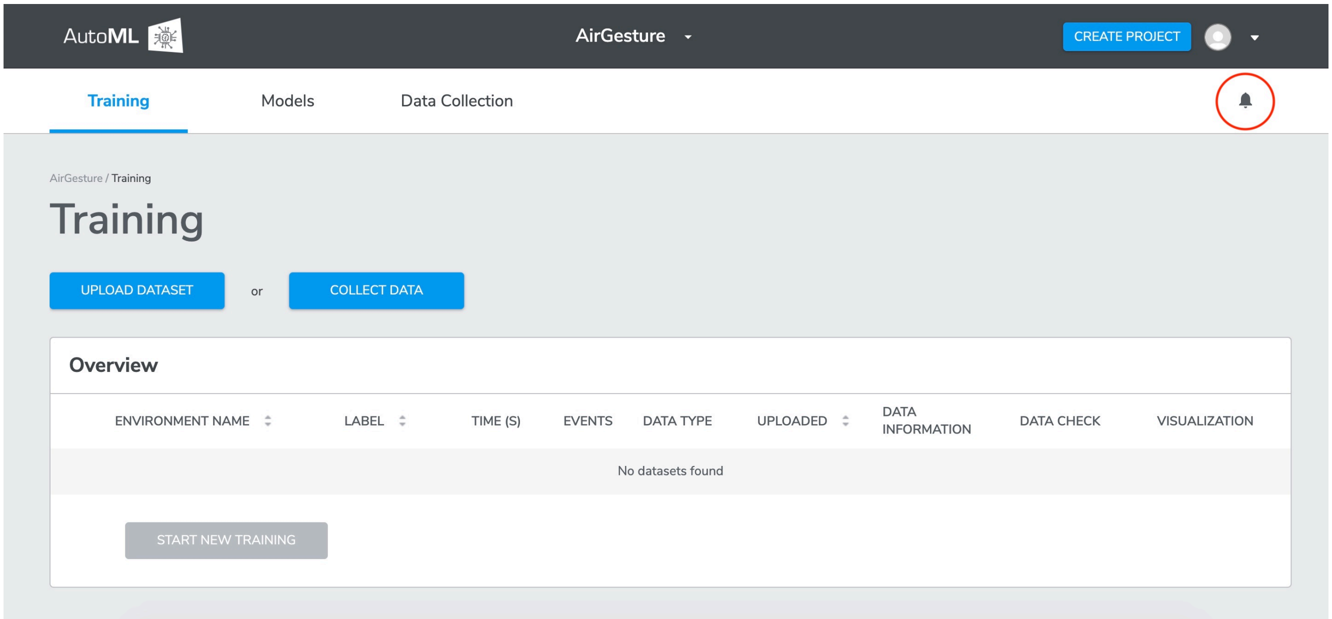
KDE plots provide detailed insights into error analysis. The example below is a KDE plot for a gesture recognition problem. All instances are collected as "Gesture2". Ideally, the scores for the other classes should be distributed around zero. However, the mode of "Gesture1" distribution (the blue line) is 0.3, and its tail extends beyond 0.5. These signify potential issues with the live-data or the model used for the analysis. We want to see the distribution of "Gesture1" and "Gesture2" as separate as possible with almost no to little overlap for both the classes. While we can observe that the "Stationary" class is quite well separated from "Gesture1" and "Gesture2". "Stationary" class has a peak around zero and also very narrow compared to "Gesture1" and "Gesture2" very well isolates it.

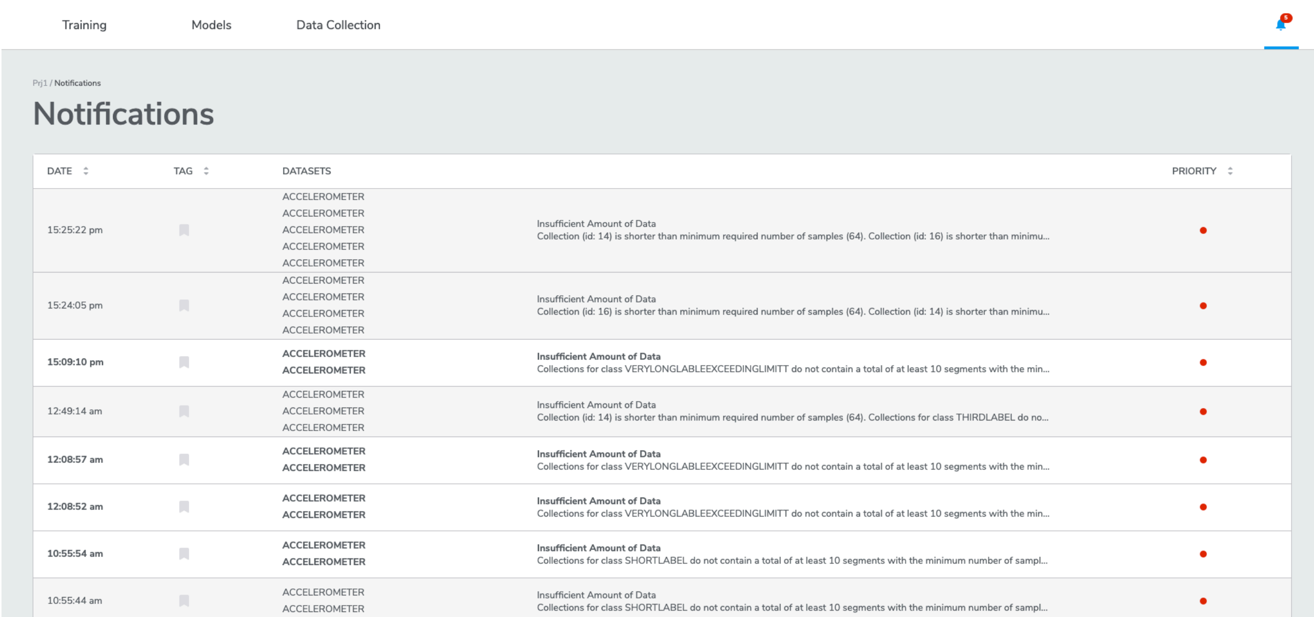


Notification Center

If you wish to be more aware of the current status of your projects, you can go to the Notification Center in which AutoML can provide detailed information of your projects, data, models and training runs.

By simply clicking the bell icon located at the rightmost of the sub-navigation-bar, you will be redirected to the Notification Center.





---

1. Choose only one microphone but not both. ↩