ML.NET

Model Builder

In .net MAUI

MauiWave

Datis Almaspoor

Integrating ML.NET with .NET MAUI

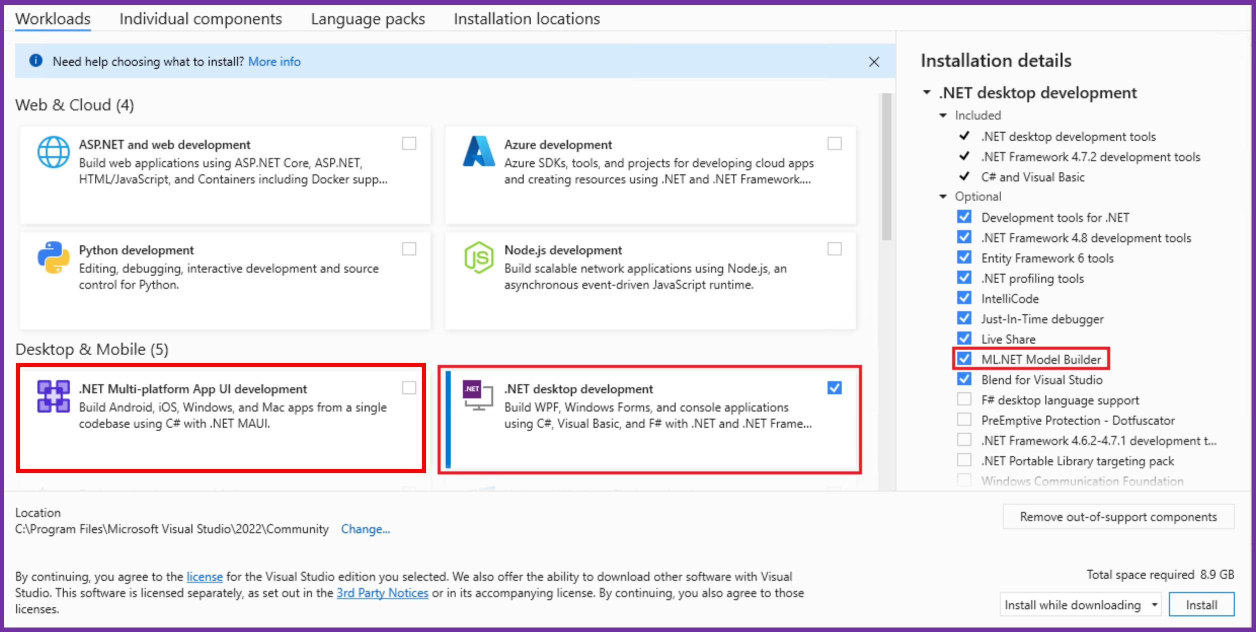
What is ml.net model builder?

ML.NET Model Builder provides an easy-to-understand visual interface within Visual Studio for creating, training, and deploying custom machine learning models.

Why API ?!

We cannot directly use the ml.net model in Android because it does not have the processing power. But it's not a problem, the alternative solution is to connect to the app and call the model in the Maui app.

Installation



Workloads | Individual components | Language packs | Installation locations

Need help choosing what to install? More info

Web & Cloud (4)

- ☐ **ASP.NET and web development** — Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
- ☐ **Azure development** — Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....
- ☐ **Python development** — Editing, debugging, interactive development and source control for Python.
- ☐ **Node.js development** — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- ☐ **.NET Multi-platform App UI development** — Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
- ✓ **.NET desktop development** — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Installation details

.NET desktop development

Included
- ✓ .NET desktop development tools
- ✓ .NET Framework 4.7.2 development tools
- ✓ C# and Visual Basic

Optional
- ✓ Development tools for .NET
- ✓ .NET Framework 4.8 development tools
- ✓ Entity Framework 6 tools
- ✓ .NET profiling tools
- ✓ IntelliCode
- ✓ Just-In-Time debugger
- ✓ Live Share
- ✓ ML.NET Model Builder
- ✓ Blend for Visual Studio
- ☐ F# desktop language support
- ☐ PreEmptive Protection - Dotfuscator
- ☐ .NET Framework 4.6.2-4.7.1 development t...
- ☐ .NET Portable Library targeting pack
- ☐ Windows Communication Foundation

Location
C:\Program Files\Microsoft Visual Studio\2022\Community   Change...

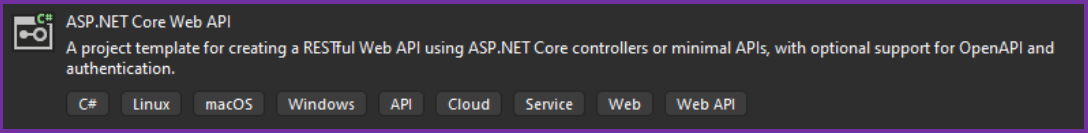
Remove out-of-support components

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 8.9 GB

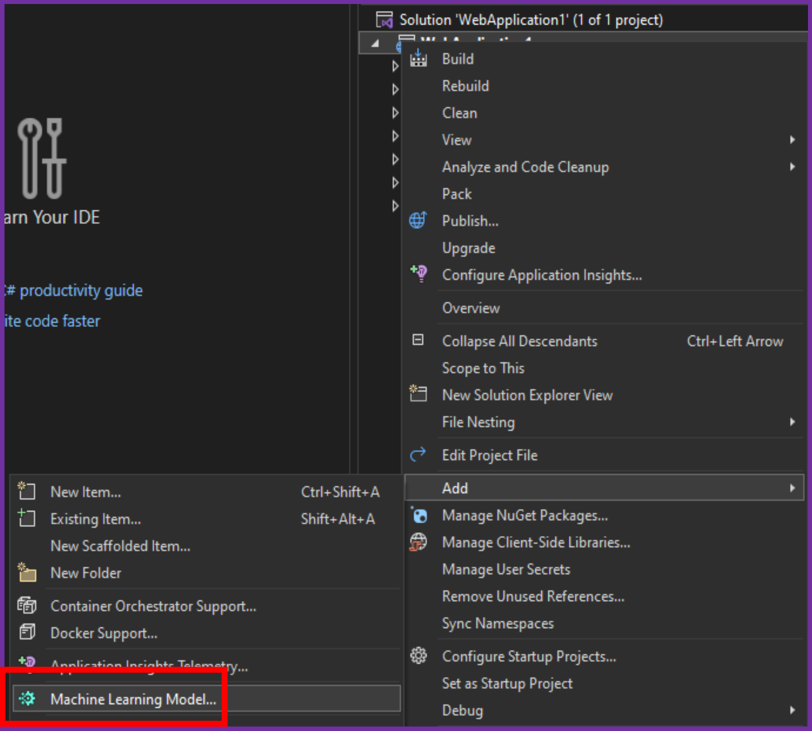
Install while downloading | Install

New Project


ASP.NET Core Web API

A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.

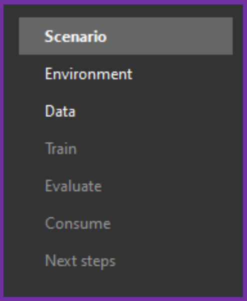
1. Add machine learning model to the project


arn Your IDE
# productivity guide
ite code faster
Solution 'WebApplication1' (1 of 1 project)
Build
Rebuild
Clean
View
Analyze and Code Cleanup
Pack
Publish...
Upgrade
Configure Application Insights...
Overview
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
File Nesting
Edit Project File
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Remove Unused References...
Sync Namespaces
Configure Startup Projects...
Set as Startup Project
Debug
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Container Orchestrator Support...
Docker Support...
Application Insights Telemetry...
Machine Learning Model...

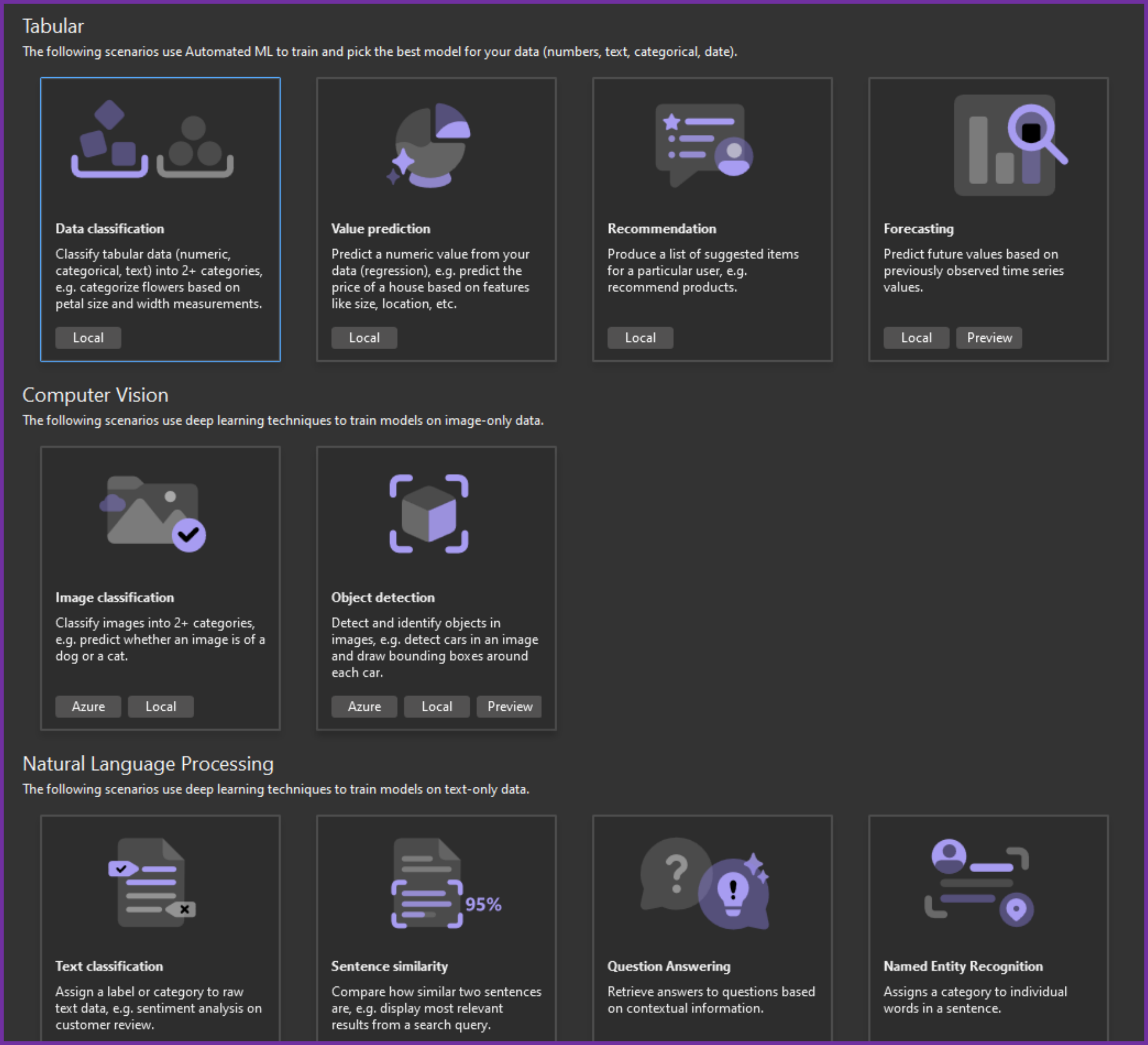
Create a Model Builder project

When you first start up Model Builder, it asks you to name the project, and then creates an mbconfig configuration file inside of the project. The mbconfig file keeps track of everything you do in Model Builder to allow you to reopen the session.

2. Complete steps of building model builder



3. Choose Scenario



Tabular

The following scenarios use Automated ML to train and pick the best model for your data (numbers, text, categorical, date).

Data classification
Classify tabular data (numeric, categorical, text) into 2+ categories, e.g. categorize flowers based on petal size and width measurements.
Local

Value prediction
Predict a numeric value from your data (regression), e.g. predict the price of a house based on features like size, location, etc.
Local

Recommendation
Produce a list of suggested items for a particular user, e.g. recommend products.
Local

Forecasting
Predict future values based on previously observed time series values.
Local Preview

Computer Vision

The following scenarios use deep learning techniques to train models on image-only data.

Image classification
Classify images into 2+ categories, e.g. predict whether an image is of a dog or a cat.
Azure Local

Object detection
Detect and identify objects in images, e.g. detect cars in an image and draw bounding boxes around each car.
Azure Local Preview

Natural Language Processing

The following scenarios use deep learning techniques to train models on text-only data.

Text classification
Assign a label or category to raw text data, e.g. sentiment analysis on customer review.

Sentence similarity
Compare how similar two sentences are, e.g. display most relevant results from a search query.

Question Answering
Retrieve answers to questions based on contextual information.

Named Entity Recognition
Assigns a category to individual words in a sentence.

Tabular:

- Data Classification: Assigning labels or categories to data points.
- Value Prediction: Predicting a continuous value (e.g., sales, temperature).
- Recommendation: Suggesting personalized items or actions.
- Forecasting: Predicting future values based on historical data.

Computer Vision:

- Image Classification: Assigning labels to images (e.g., identifying objects in photos).
- Object Detection: Locating and classifying multiple objects within an image.

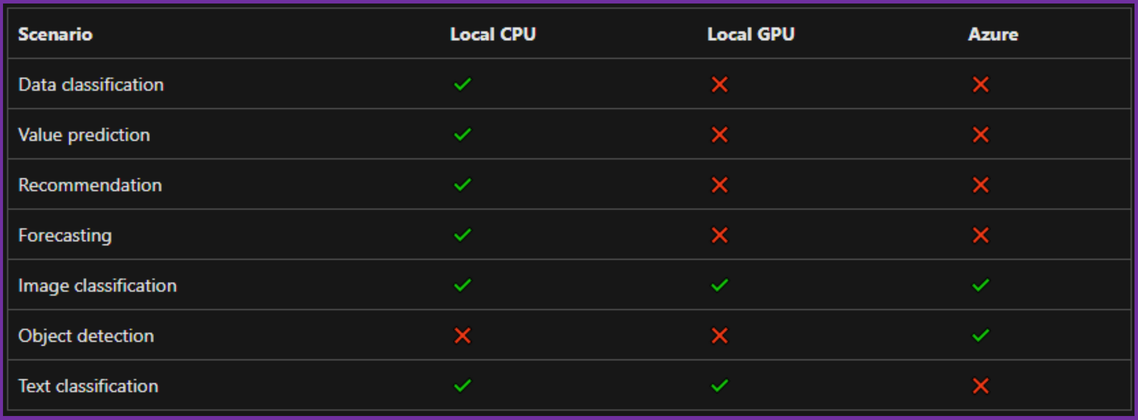
Natural Language Processing:

- Text Classification: Categorizing text into predefined classes (e.g., spam detection).
- Sentence Similarity: Measuring similarity between sentences.
- Question Answering: Generating answers based on input questions.
- Named Entity Recognition: Identifying entities (e.g., names, dates) in text.

4. Environment

You can train your machine learning model locally on your machine or in the cloud on Azure, depending on the scenario.

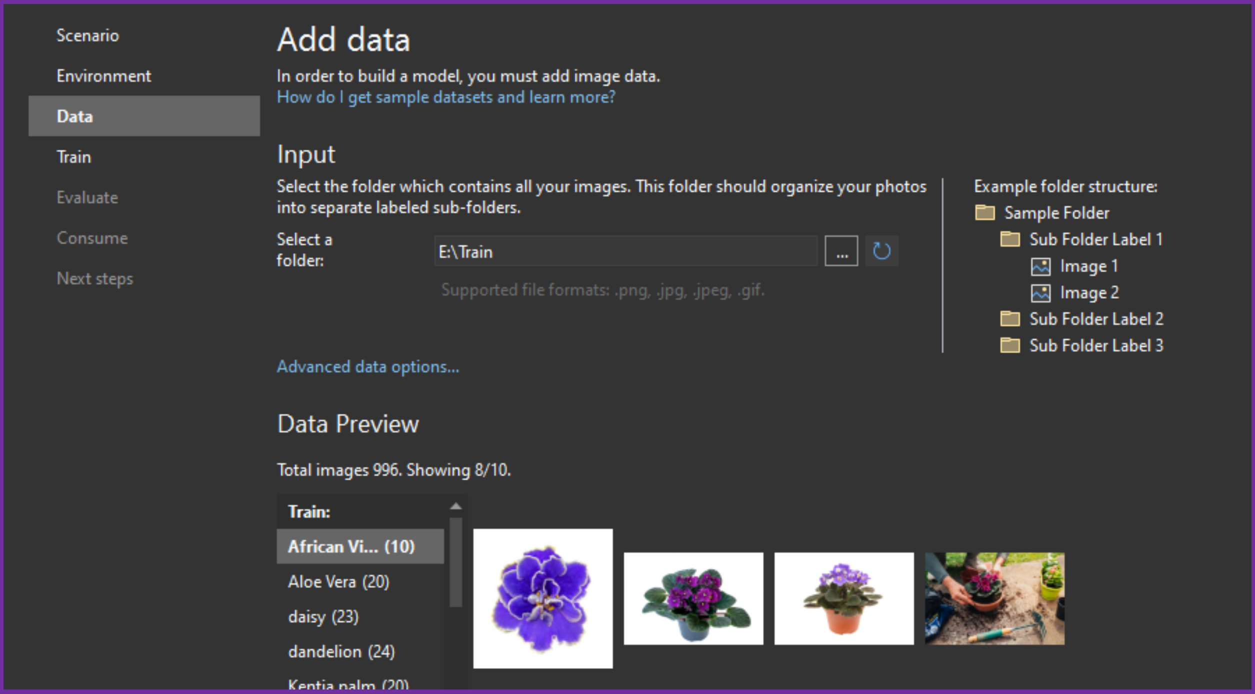
When you train locally, you work within the constraints of your computer resources (CPU, memory, and disk). When you train in the cloud, you can scale up your resources to meet the demands of your scenario, especially for large datasets.



Scenario	Local CPU	Local GPU	Azure
Data classification	✓	✗	✗
Value prediction	✓	✗	✗
Recommendation	✓	✗	✗
Forecasting	✓	✗	✗
Image classification	✓	✓	✓
Object detection	✗	✗	✓
Text classification	✓	✓	✗

5. Data

Once you've chosen your scenario, Model Builder asks you to provide a dataset. The data is used to train, evaluate, and choose the best model for your scenario.



Add data

In order to build a model, you must add image data.
How do I get sample datasets and learn more?

Input

Select the folder which contains all your images. This folder should organize your photos into separate labeled sub-folders.

Select a folder: E:\Train

Supported file formats: .png, .jpg, .jpeg, .gif.

Example folder structure:
Sample Folder
Sub Folder Label 1
Image 1
Image 2
Sub Folder Label 2
Sub Folder Label 3

Advanced data options...

Data Preview

Total images 996. Showing 8/10.

Train:
African Vi... (10)
Aloe Vera (20)
daisy (23)
dandelion (24)
Kentia palm (20)

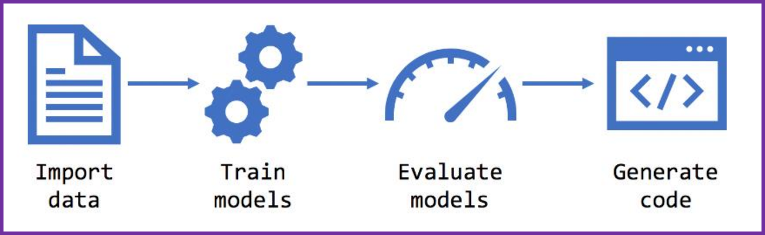
Model Builder supports datasets in .tsv, .csv, .txt, and SQL database formats. If you have a .txt file, columns should be separated with , ; \t.

If the dataset is made up of images, the supported file types are .jpg and .png.

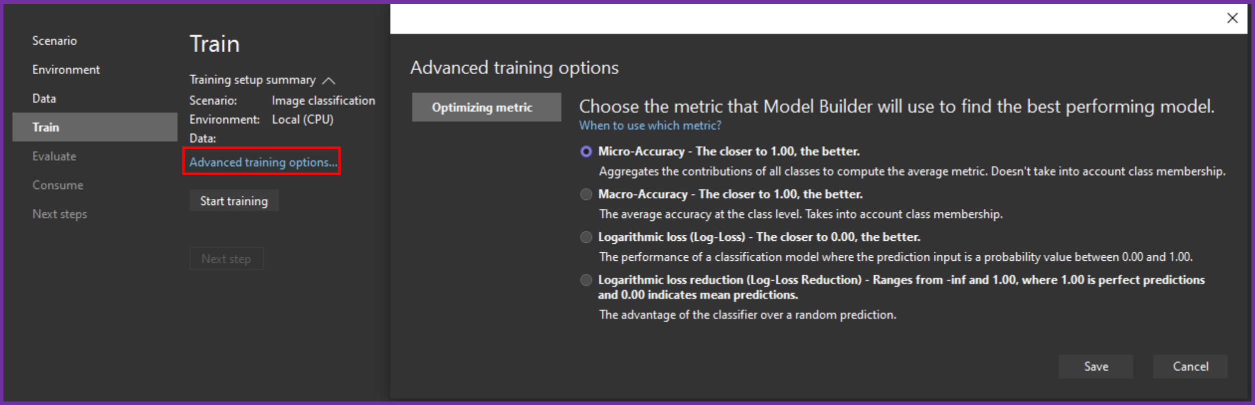
6. Train

Training is an automatic process by which Model Builder teaches your model how to answer questions for your scenario. Once trained, your model can make predictions with input data that it has not seen before. For example, if you are predicting house prices and a new house comes on the market, you can predict its sale price.

Because Model Builder uses automated machine learning (AutoML), it does not require any input or tuning from you during training.



Advanced Training options :



Train

Training setup summary

Scenario: Image classification
Environment: Local (CPU)
Data:

Advanced training options

Choose the metric that Model Builder will use to find the best performing model.

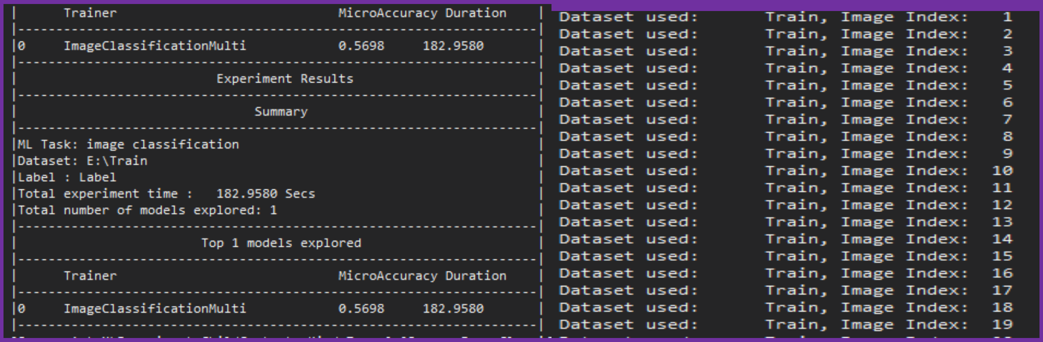
- [✓ **Micro-Accuracy - The closer to 1.00, the better.**]
  Aggregates the contributions of all classes to compute the average metric. Doesn't take into account class membership.
- **Macro-Accuracy - The closer to 1.00, the better.**
  The average accuracy at the class level. Takes into account class membership.
- **Logarithmic loss (Log-Loss) - The closer to 0.00, the better.**
  The performance of a classification model where the prediction input is a probability value between 0.00 and 1.00.
- **Logarithmic loss reduction (Log-Loss Reduction) - Ranges from -inf and 1.00, where 1.00 is perfect predictions and 0.00 indicates mean predictions.**
  The advantage of the classifier over a random prediction.

- Micro-Accuracy - The closer to 1.00, the better.
  Aggregates the contributions of all classes to compute the average metric. Doesn't take into account class membership.
- Macro-Accuracy - The closer to 1.00, the better.
  The average accuracy at the class level. Takes into account class membership.
- Logarithmic loss (Log-Loss) - The closer to 0.00, the better.
  The performance of a classification model where the prediction input is a probability value between 0.00 and 1.00.
- Logarithmic loss reduction (Log-Loss Reduction) - Ranges from -inf and 1.00, where 1.00 is perfect predictions and 0.00 indicates mean predictions.
  The advantage of the classifier over a random prediction.

Important :

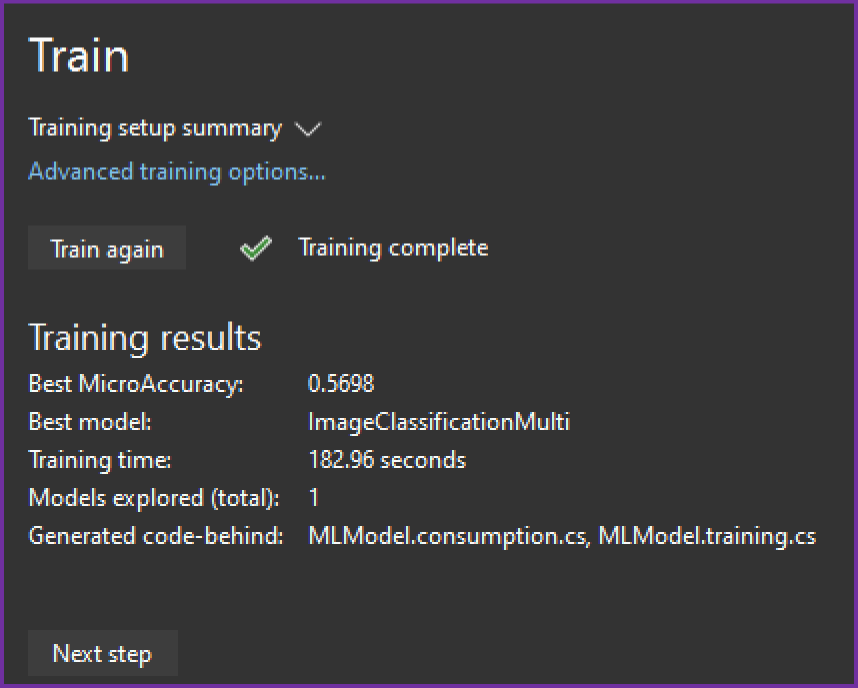
After clicking on start, make sure that all the desired files are installed and placed in the right place, if there is any deficiency, the relevant information and download link will be placed in the output section.

After starting(train), you can check the progress in the output section.



```
|      Trainer                                MicroAccuracy Duration  |
|---------------------------------------------------------------------|
|0     ImageClassificationMulti               0.5698      182.9580    |
|---------------------------------------------------------------------|
|                         Experiment Results                          |
|---------------------------------------------------------------------|
|                              Summary                                |
|---------------------------------------------------------------------|
|ML Task: image classification                                        |
|Dataset: E:\Train                                                    |
|Label : Label                                                        |
|Total experiment time :   182.9580 Secs                              |
|Total number of models explored: 1                                   |
|---------------------------------------------------------------------|
|                        Top 1 models explored                        |
|---------------------------------------------------------------------|
|      Trainer                                MicroAccuracy Duration  |
|---------------------------------------------------------------------|
|0     ImageClassificationMulti               0.5698      182.9580    |
|---------------------------------------------------------------------|
```

```
Dataset used:      Train, Image Index:   1
Dataset used:      Train, Image Index:   2
Dataset used:      Train, Image Index:   3
Dataset used:      Train, Image Index:   4
Dataset used:      Train, Image Index:   5
Dataset used:      Train, Image Index:   6
Dataset used:      Train, Image Index:   7
Dataset used:      Train, Image Index:   8
Dataset used:      Train, Image Index:   9
Dataset used:      Train, Image Index:  10
Dataset used:      Train, Image Index:  11
Dataset used:      Train, Image Index:  12
Dataset used:      Train, Image Index:  13
Dataset used:      Train, Image Index:  14
Dataset used:      Train, Image Index:  15
Dataset used:      Train, Image Index:  16
Dataset used:      Train, Image Index:  17
Dataset used:      Train, Image Index:  18
Dataset used:      Train, Image Index:  19
```



Train

Training setup summary

Advanced training options...

Train again ✓ Training complete

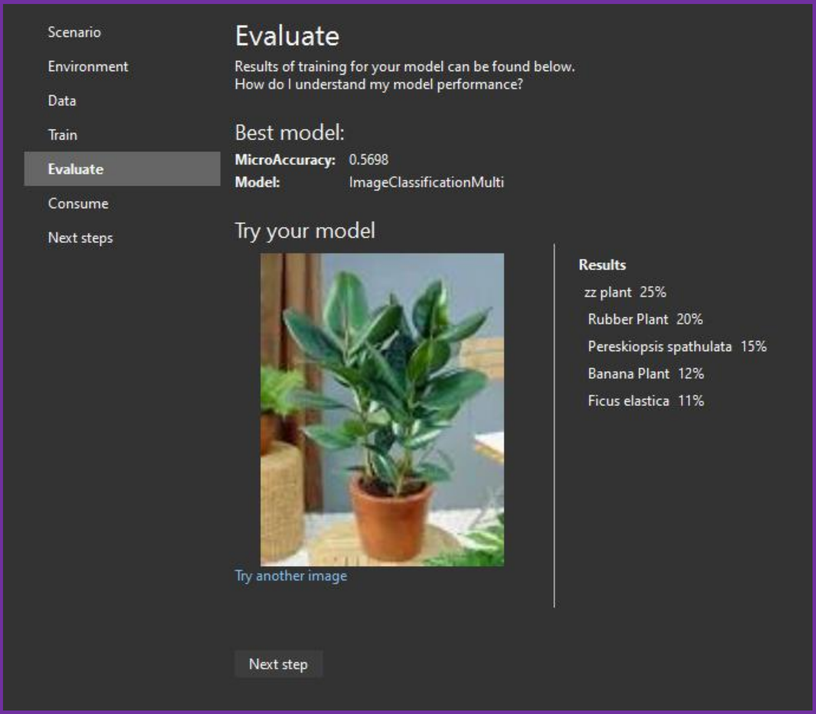
Training results

Best MicroAccuracy:	0.5698
Best model:	ImageClassificationMulti
Training time:	182.96 seconds
Models explored (total):	1
Generated code-behind:	MLModel.consumption.cs, MLModel.training.cs

Next step

7. Evaluate

Evaluation is the process of measuring how good your model is. Model Builder uses the trained model to make predictions with new test data, and then measures how good the predictions are.



Evaluate

Results of training for your model can be found below.
How do I understand my model performance?

Best model:

MicroAccuracy: 0.5698
Model: ImageClassificationMulti

Try your model

Results
zz plant 25%
Rubber Plant 20%
Pereskiopsis spathulata 15%
Banana Plant 12%
Ficus elastica 11%

Improve

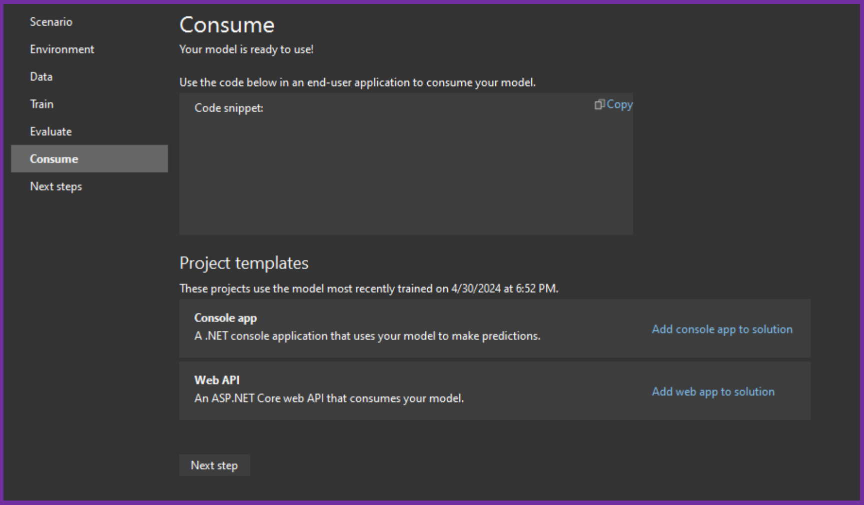
If your model performance score is not as good as you want it to be, you can:

- Train for a longer period of time. With more time, the automated machine learning engine experiments with more algorithms and settings.
- Add more data. Sometimes the amount of data is not sufficient to train a high-quality machine learning model.This is especially true with datasets that have a small number of examples.
- Balance your data. For classification tasks, make sure that the training set is balanced across the categories. For example, if you have four classes for 100 training examples, and the two first classes (tag1 and tag2) are used for 90 records, but the other two (tag3 and tag4) are only used on the remaining 10 records, the lack of balanced data may cause your model to struggle to correctly predict tag3 or tag4.

8. Consume

After the evaluation phase, Model Builder outputs a model file, and code that you can use to add the model to your application. ML.NET models are saved as a zip file. The code to load and use your model is added as a new project in your solution. Model

Builder also adds a sample console app that you can run to see your model in action.In addition, Model Builder gives you the option to create projects that consume your model. Currently, Model Builder will create the following projects:



Consume

Your model is ready to use!

Use the code below in an end-user application to consume your model.

Code snippet:

Project templates

These projects use the model most recently trained on 4/30/2024 at 6:52 PM.

Console app
A .NET console application that uses your model to make predictions.

Web API
An ASP.NET Core web API that consumes your model.

Console app: Creates a .NET console application to make predictions from your model.

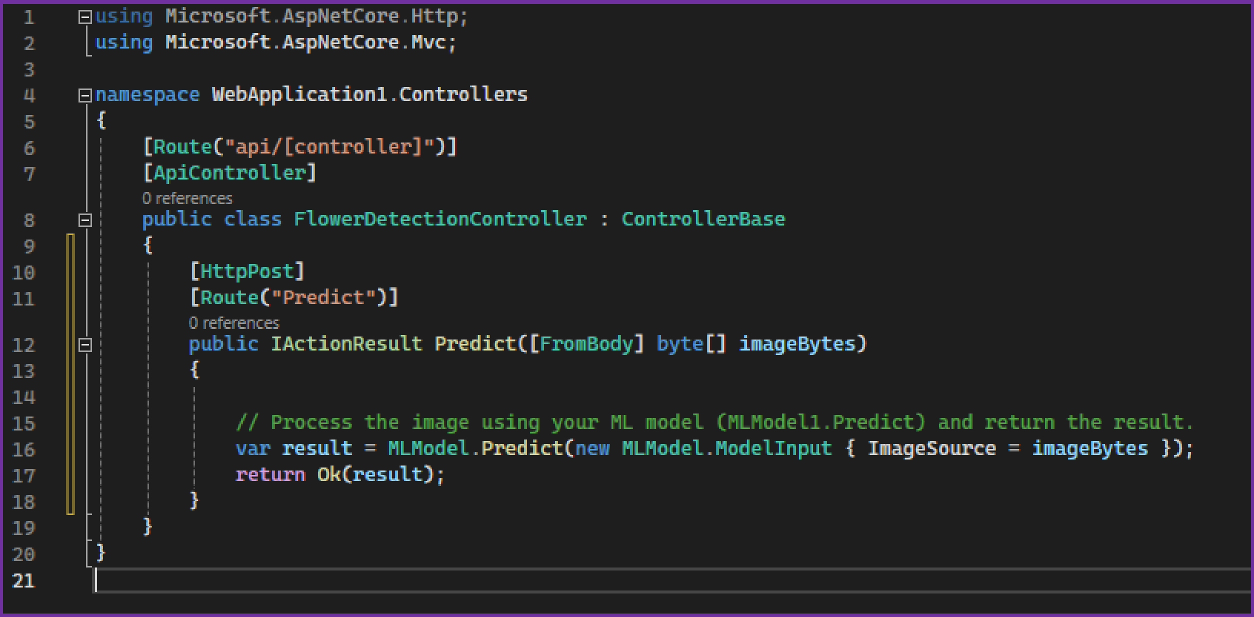
Web API: Creates an ASP.NET Core Web API that lets you consume your model over the internet.

After training, three files are generated under the *.mbconfig file:

- **Model.consumption.cs** : This file contains the ModelInput and ModelOutput schemas as well as the Predict function generated for consuming the model.
- **Model.training.cs** : This file contains the training pipeline (data transforms, algorithm, algorithm hyperparameters) chosen by Model Builder to train the model. You can use this pipeline for re-training your model.
- **Model.zip** : This is a serialized zip file which represents your trained ML.NET model.

9. Connect model to API

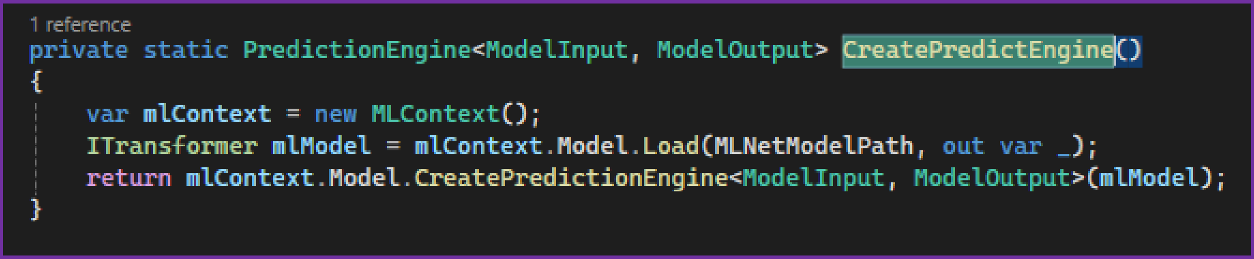
Add Empty API Controller to your project, and add endpoint like this (only for image classification, customize it by your scenario) :



```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    0 references
    public class FlowerDetectionController : ControllerBase
    {
        [HttpPost]
        [Route("Predict")]
        0 references
        public IActionResult Predict([FromBody] byte[] imageBytes)
        {

            // Process the image using your ML model (MLModel1.Predict) and return the result.
            var result = MLModel.Predict(new MLModel.ModelInput { ImageSource = imageBytes });
            return Ok(result);
        }
    }
}
```

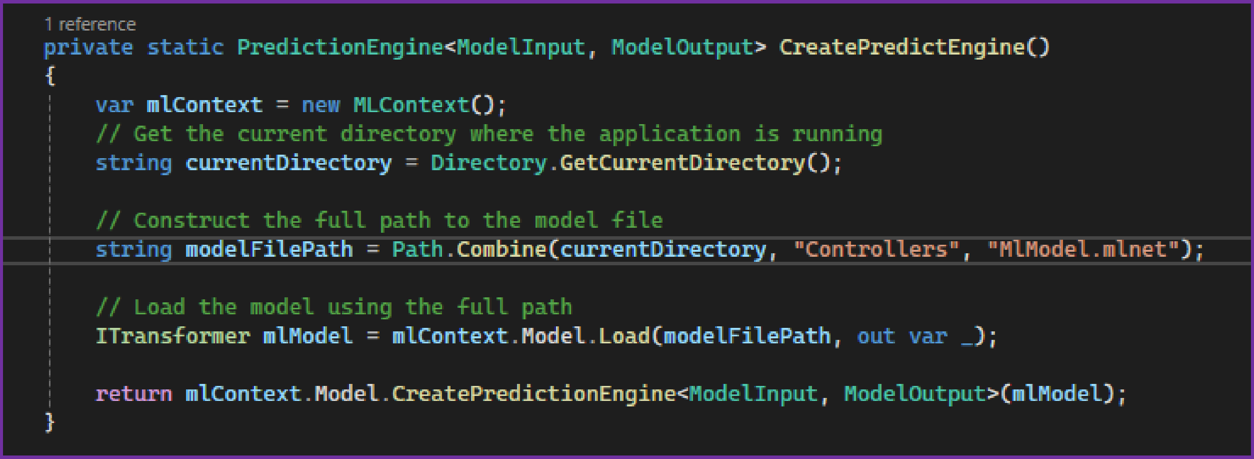
Next step is edit consumption.cs file, we should edit CreatePredictEngine like this :

Before :



```
1 reference
private static PredictionEngine<ModelInput, ModelOutput> CreatePredictEngine()
{
    var mlContext = new MLContext();
    ITransformer mlModel = mlContext.Model.Load(MLNetModelPath, out var _);
    return mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);
}
```

After :

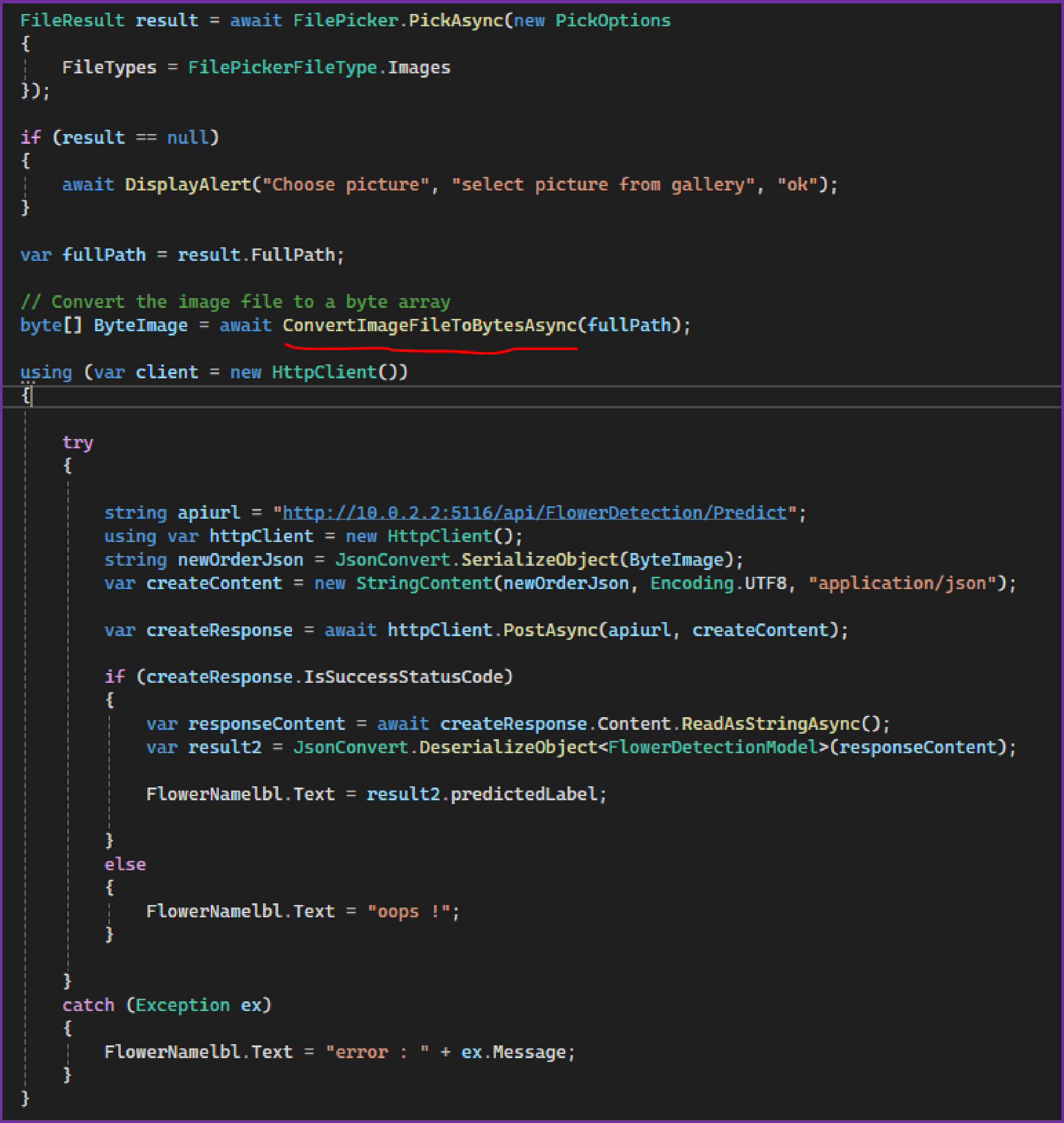


```
1 reference
private static PredictionEngine<ModelInput, ModelOutput> CreatePredictEngine()
{
    var mlContext = new MLContext();
    // Get the current directory where the application is running
    string currentDirectory = Directory.GetCurrentDirectory();

    // Construct the full path to the model file
    string modelFilePath = Path.Combine(currentDirectory, "Controllers", "MlModel.mlnet");

    // Load the model using the full path
    ITransformer mlModel = mlContext.Model.Load(modelFilePath, out var _);

    return mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);
}
```

Be sure to replace MlModel.mlnet with your model name.

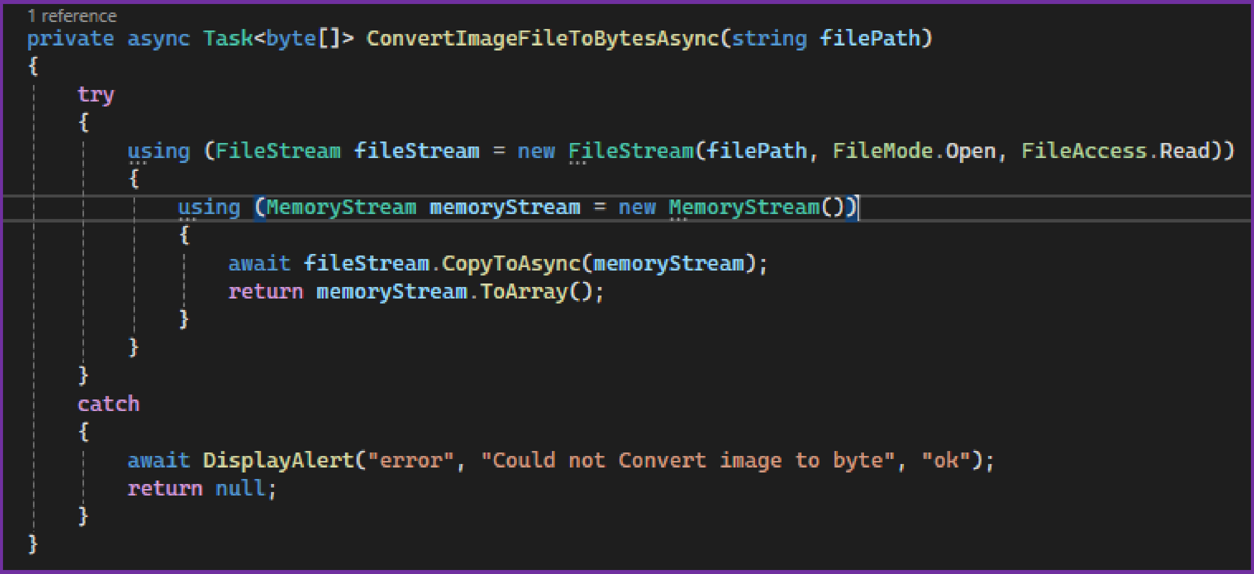
10.Connect .Net MAUI to API

Here, our scenario is image classification, so we need to select the photo in the phone (or, for example, take a photo at the same moment, but we will not pay much attention because this is not the topic of the book), if your scenario is different, you can easily take it from user, for example a text box.



```
FileResult result = await FilePicker.PickAsync(new PickOptions
{
    FileTypes = FilePickerFileType.Images
});

if (result == null)
{
    await DisplayAlert("Choose picture", "select picture from gallery", "ok");
}

var fullPath = result.FullPath;

// Convert the image file to a byte array
byte[] ByteImage = await ConvertImageFileToBytesAsync(fullPath);

using (var client = new HttpClient())
{

    try
    {

        string apiurl = "http://10.0.2.2:5116/api/FlowerDetection/Predict";
        using var httpClient = new HttpClient();
        string newOrderJson = JsonConvert.SerializeObject(ByteImage);
        var createContent = new StringContent(newOrderJson, Encoding.UTF8, "application/json");

        var createResponse = await httpClient.PostAsync(apiurl, createContent);

        if (createResponse.IsSuccessStatusCode)
        {
            var responseContent = await createResponse.Content.ReadAsStringAsync();
            var result2 = JsonConvert.DeserializeObject<FlowerDetectionModel>(responseContent);

            FlowerNamelbl.Text = result2.predictedLabel;

        }
        else
        {
            FlowerNamelbl.Text = "oops !";
        }

    }
    catch (Exception ex)
    {
        FlowerNamelbl.Text = "error : " + ex.Message;
    }
}
```

Here is ConvertImageFileToByteAsync code :



```
1 reference
private async Task<byte[]> ConvertImageFileToBytesAsync(string filePath)
{
    try
    {
        using (FileStream fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read))
        {
            using (MemoryStream memoryStream = new MemoryStream())
            {
                await fileStream.CopyToAsync(memoryStream);
                return memoryStream.ToArray();
            }
        }
    }
    catch
    {
        await DisplayAlert("error", "Could not Convert image to byte", "ok");
        return null;
    }
}
```

Thank you for taking the time to read my digital book on programming. I hope you found it informative and enjoyable. Your support and interest mean a lot to me. Happy coding!