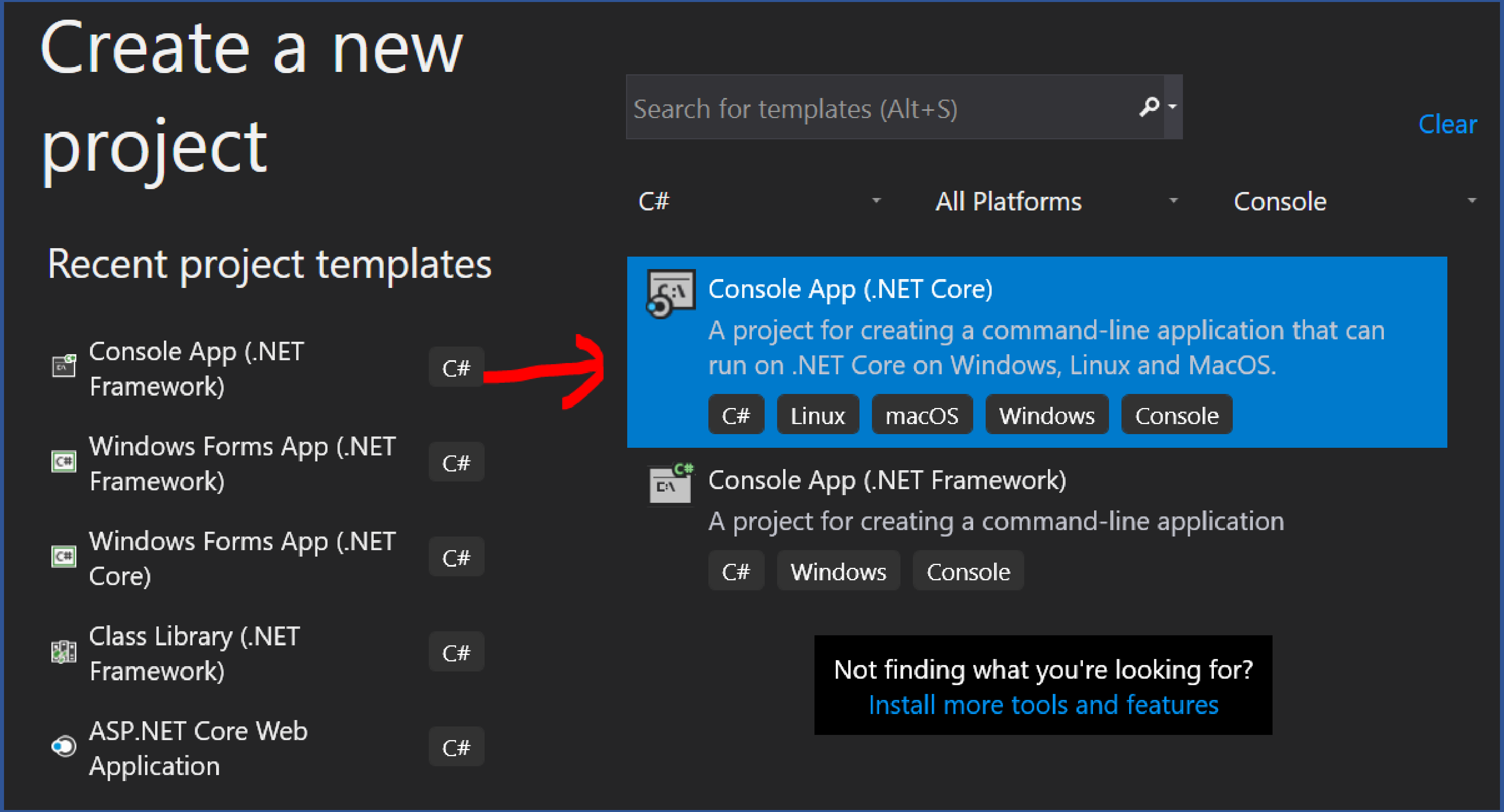
Predict from CSV file

What's in this session?

1. Add NuGet
2. Copy data models to project
3. Add base class reference
4. Add using to Program
5. Add ML Model and test data to mib folder
6. Add code to Main
7. Run program and verify the result

Create new .NET CORE console project



Create a new project

Recent project templates

Console App (.NET Framework) C#

Windows Forms App (.NET Framework) C#

Windows Forms App (.NET Core) C#

Class Library (.NET Framework) C#

ASP.NET Core Web Application C#

Search for templates (Alt+S)

Clear

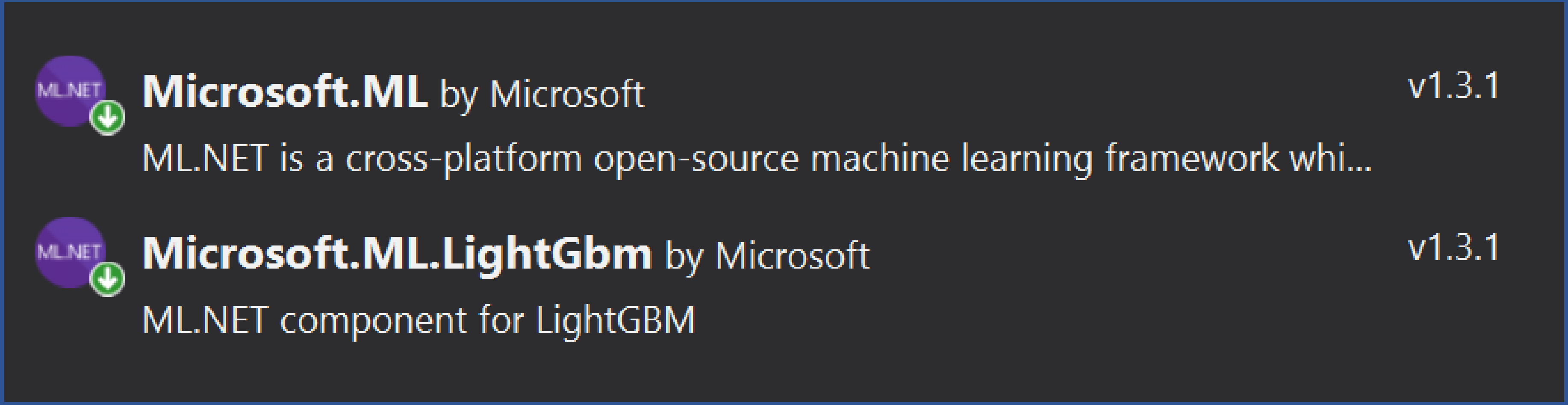
C# | All Platforms | Console

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

Console App (.NET Framework)
A project for creating a command-line application
C# Windows Console

Not finding what you're looking for?
Install more tools and features

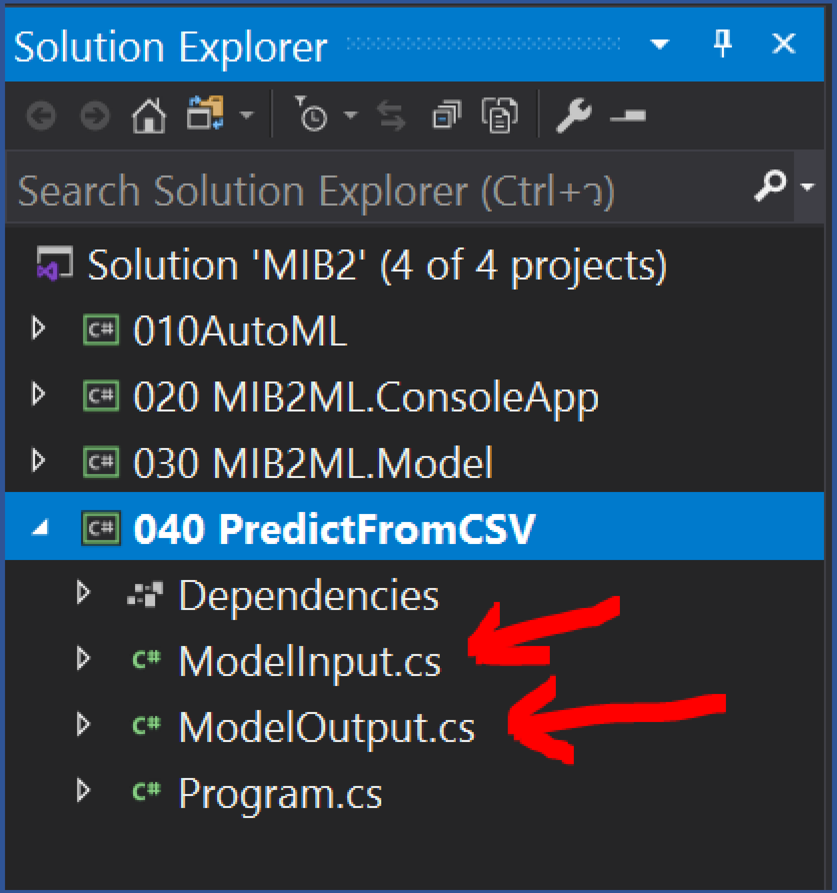
Add NuGet



Microsoft.ML by Microsoft v1.3.1
ML.NET is a cross-platform open-source machine learning framework whi...

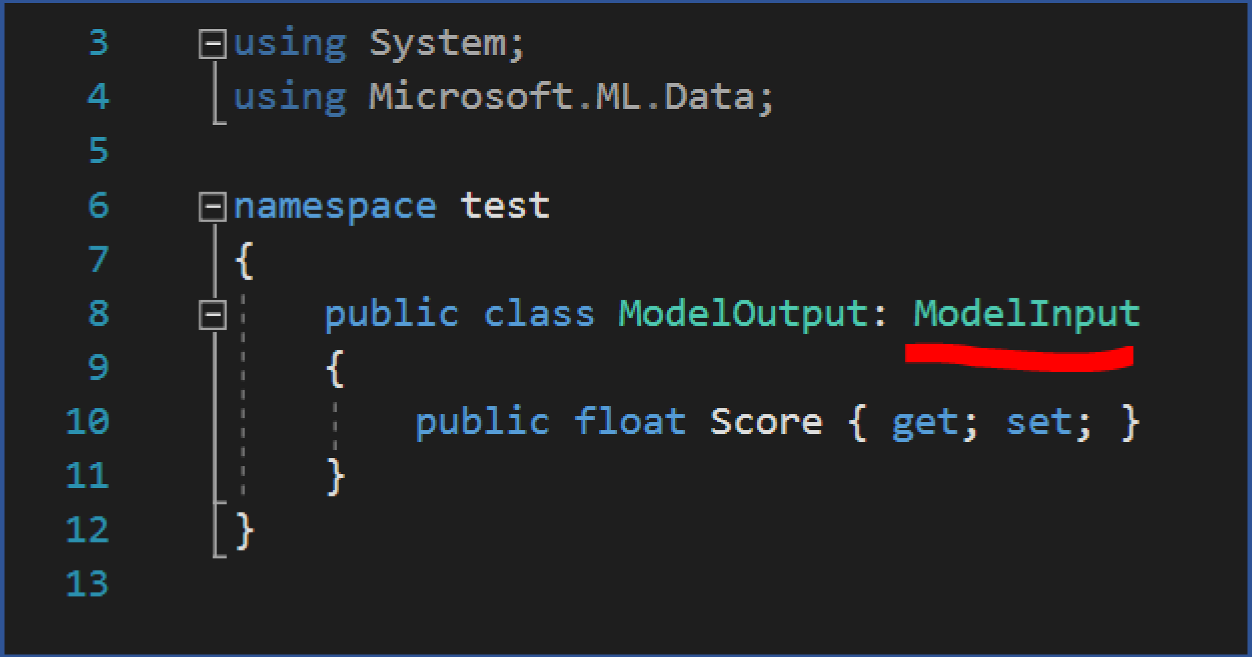
Microsoft.ML.LightGbm by Microsoft v1.3.1
ML.NET component for LightGBM

Copy data models to project



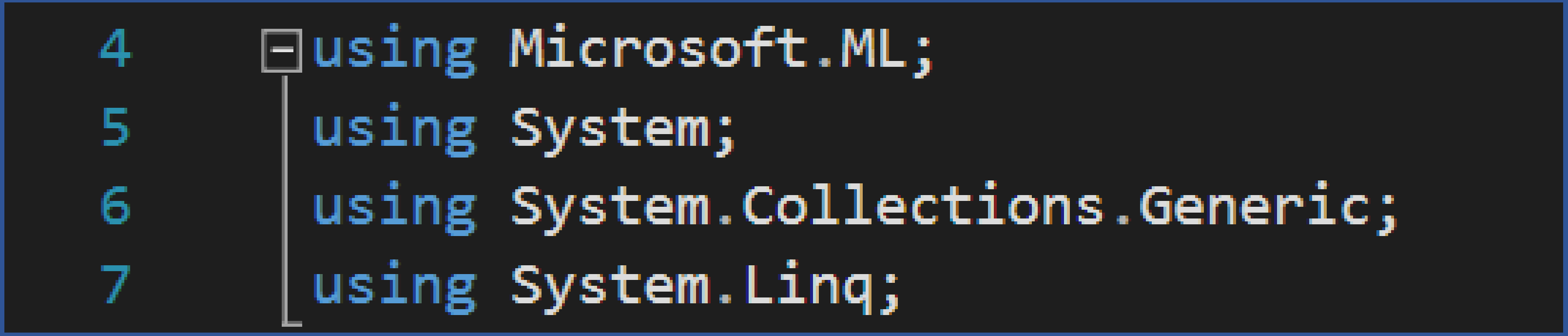
Solution 'MIB2' (4 of 4 projects)
010AutoML
020 MIB2ML.ConsoleApp
030 MIB2ML.Model
040 PredictFromCSV
Dependencies
ModelInput.cs
ModelOutput.cs
Program.cs

Add base class reference



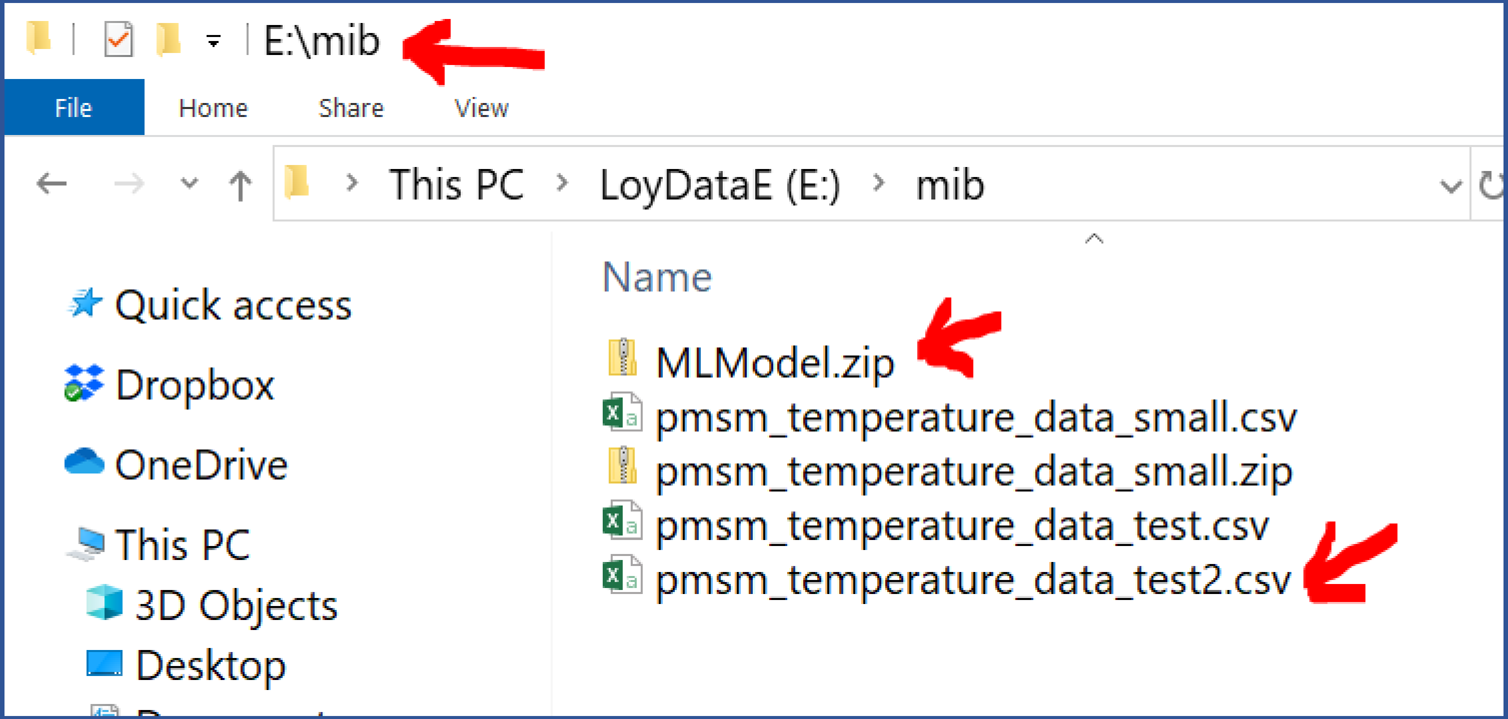
```
using System;
using Microsoft.ML.Data;

namespace test
{
    public class ModelOutput: ModelInput
    {
        public float Score { get; set; }
    }
}
```

Add using to Program



```
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.Linq;
```

Add ML Model and test data to mib folder

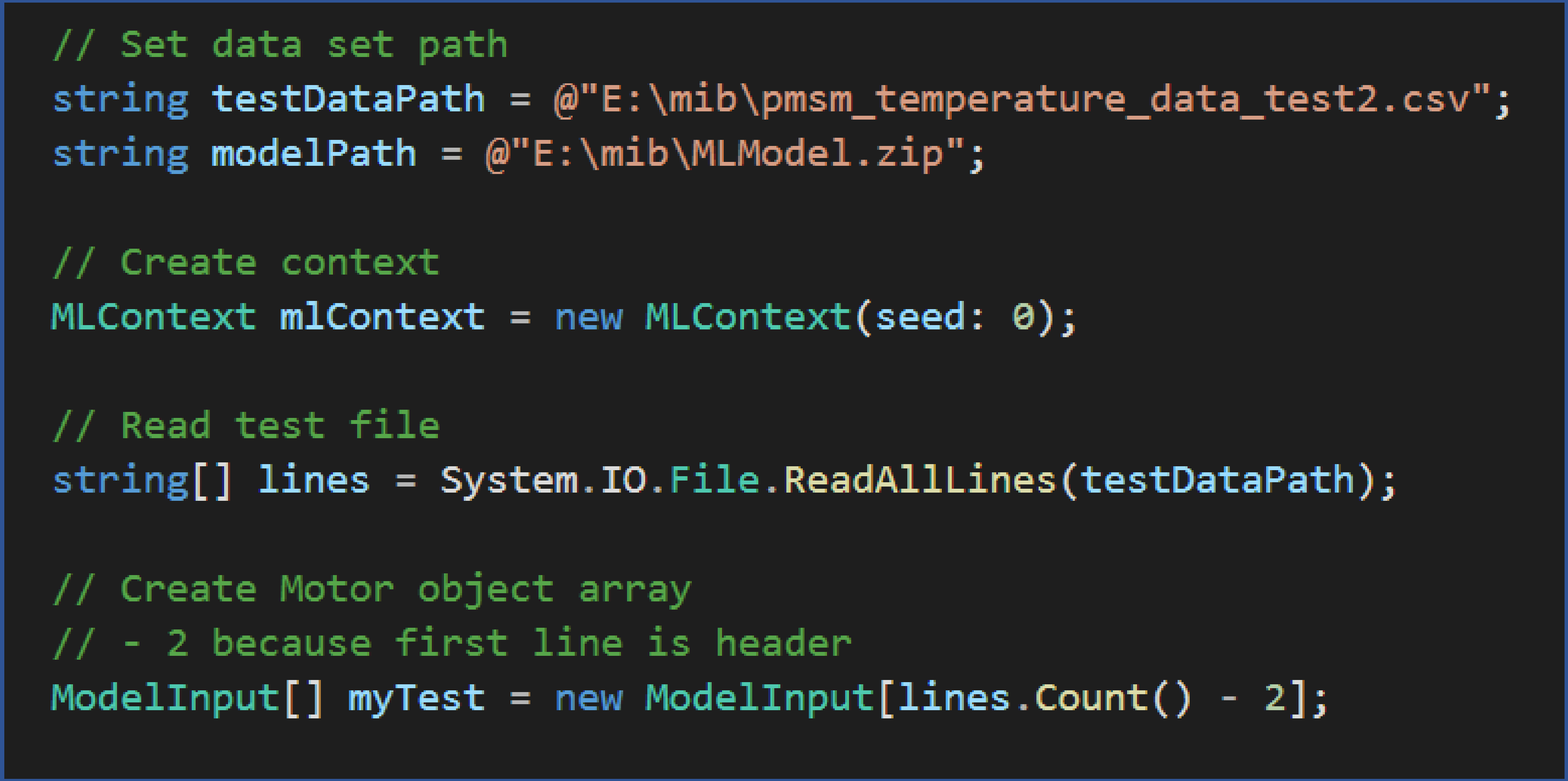

E:\mib

This PC › LoyDataE (E:) › mib

Name

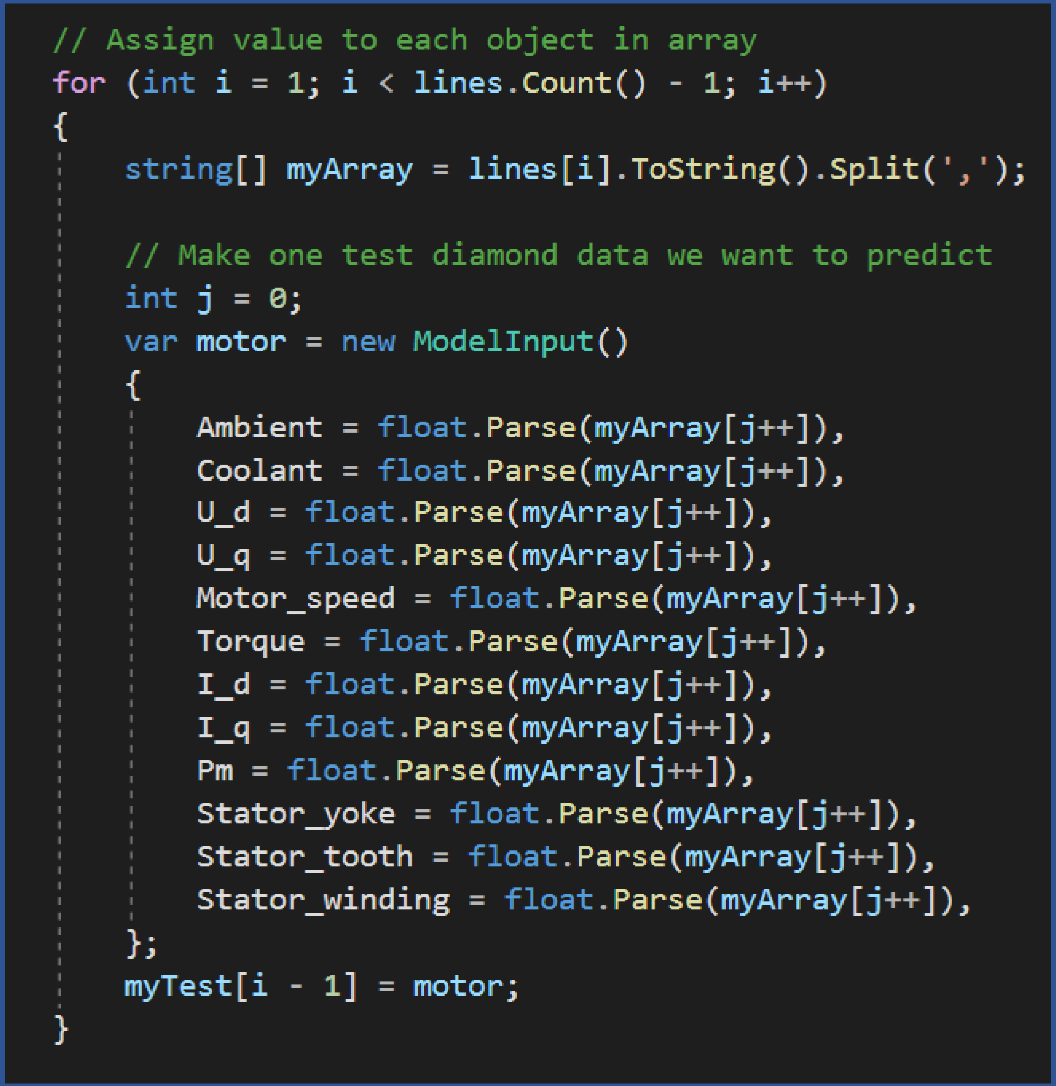
- MLModel.zip
- pmsm_temperature_data_small.csv
- pmsm_temperature_data_small.zip
- pmsm_temperature_data_test.csv
- pmsm_temperature_data_test2.csv

Add code to Main



```
// Set data set path
string testDataPath = @"E:\mib\pmsm_temperature_data_test2.csv";
string modelPath = @"E:\mib\MLModel.zip";

// Create context
MLContext mlContext = new MLContext(seed: 0);

// Read test file
string[] lines = System.IO.File.ReadAllLines(testDataPath);

// Create Motor object array
// - 2 because first line is header
ModelInput[] myTest = new ModelInput[lines.Count() - 2];
```

Add code to Main

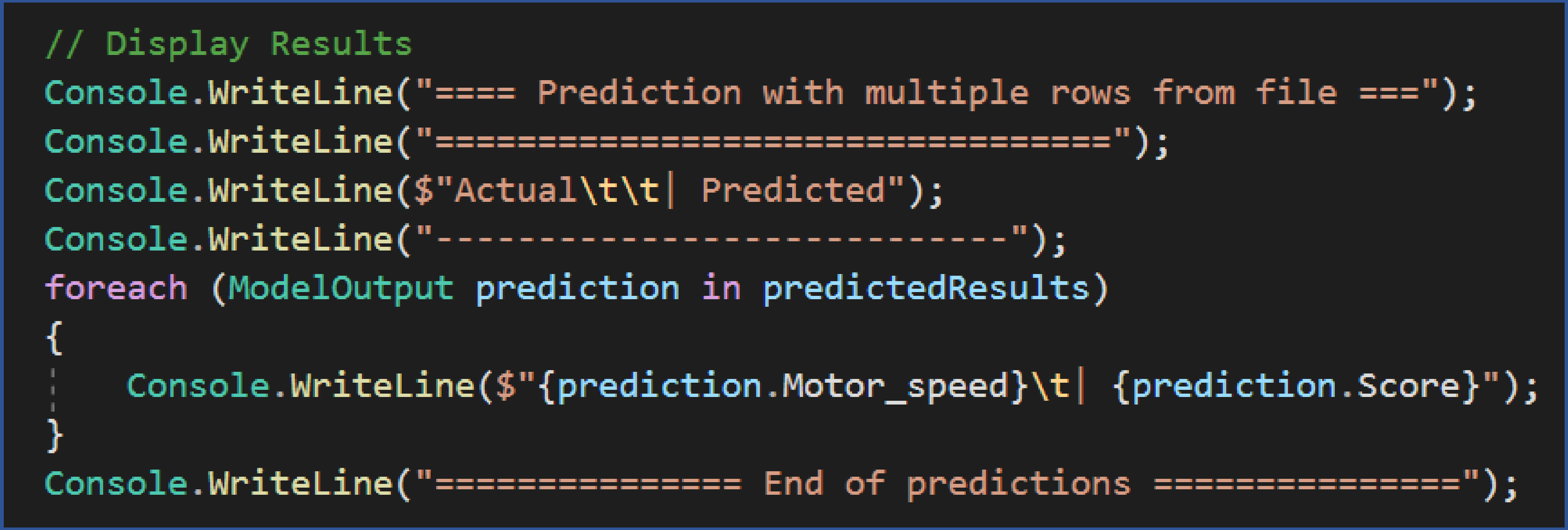


```
// Assign value to each object in array
for (int i = 1; i < lines.Count() - 1; i++)
{
    string[] myArray = lines[i].ToString().Split(',');

    // Make one test diamond data we want to predict
    int j = 0;
    var motor = new ModelInput()
    {
        Ambient = float.Parse(myArray[j++]),
        Coolant = float.Parse(myArray[j++]),
        U_d = float.Parse(myArray[j++]),
        U_q = float.Parse(myArray[j++]),
        Motor_speed = float.Parse(myArray[j++]),
        Torque = float.Parse(myArray[j++]),
        I_d = float.Parse(myArray[j++]),
        I_q = float.Parse(myArray[j++]),
        Pm = float.Parse(myArray[j++]),
        Stator_yoke = float.Parse(myArray[j++]),
        Stator_tooth = float.Parse(myArray[j++]),
        Stator_winding = float.Parse(myArray[j++]),
    };
    myTest[i - 1] = motor;
}
```

Add code to Main

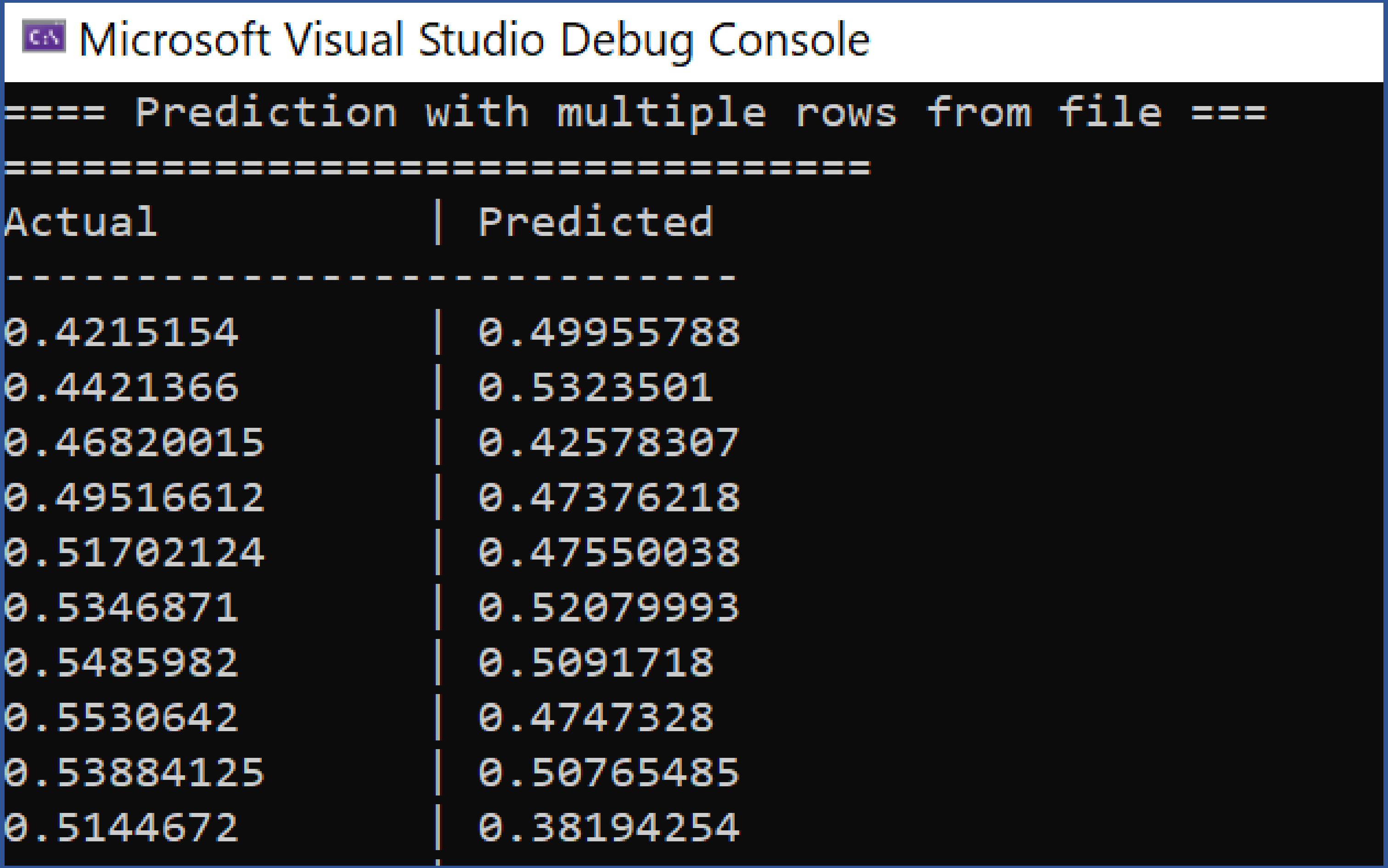
```
// Create IDataView from enumberable
IDataView batchView = mlContext.Data.LoadFromEnumerable(myTest);

// Load Model
ITransformer loadedModel = mlContext.Model.Load(modelPath, out var modelInputSchema);

// Create prediction view
IDataView predictions = loadedModel.Transform(batchView);

// Create enumerable prediction
IEnumerable<ModelOutput> predictedResults = mlContext.Data.CreateEnumerable<ModelOutput>
    (predictions, reuseRowObject: false);
```

Add code to Main



```
// Display Results
Console.WriteLine("==== Prediction with multiple rows from file ===");
Console.WriteLine("=================================");
Console.WriteLine($"Actual\t\t| Predicted");
Console.WriteLine("----------------------------");
foreach (ModelOutput prediction in predictedResults)
{
    Console.WriteLine($"{prediction.Motor_speed}\t| {prediction.Score}");
}
Console.WriteLine("=============== End of predictions ===============");
```

Run program and verify the result



```
==== Prediction with multiple rows from file ===
=================================
Actual          | Predicted
----------------------------
0.4215154       | 0.49955788
0.4421366       | 0.5323501
0.46820015      | 0.42578307
0.49516612      | 0.47376218
0.51702124      | 0.47550038
0.5346871       | 0.52079993
0.5485982       | 0.5091718
0.5530642       | 0.4747328
0.53884125      | 0.50765485
0.5144672       | 0.38194254
```

What's next?