GitHub

Write Code

(Multiclass Classification – Sdca Maximum Entropy)

What's in this session?

1. Question and Data
2. Create project
3. Add NuGet packages
4. Add using name space
5. Create data set input/output scheme
6. Set data set path
7. Load data

8. Add algorithm
9. Train the model
10. Save model
11. Evaluate the model and show accuracy stats
12. Predict single item

Question and Data

Question: The issue is belong to witch category?

Dataset:

issues_train.tsv

https://raw.githubusercontent.com/laploy/ML.NET/master/GitHub-Issue/issues_train.tsv

issues_test.tsv

https://raw.githubusercontent.com/laploy/ML.NET/master/GitHub-Issue/issues_test.tsv

Dataset description

ID:	Issue Identification Number	Must be dropped
Area:	Issue area	This is the label
Title:	Issue title	This is the first feature
Description:	Issue description	This is the second feature

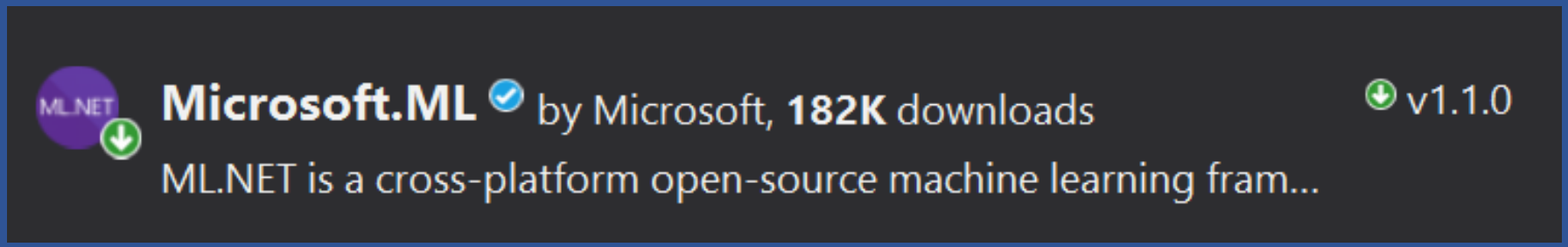
	A	B	C	
1	ID	Area	Title	Description
2	24597	area-System.Net	HttpWebRequest Not Sup	``` HttpRequest = (HttpWe
3	24598	area-System.Diagnostics	System.Diagnostics.Tests.	Failed test: System.Diagnos
4	24599	area-System.Diagnostics	System.Diagnostics.Tests.	Failed test: System.Diagnos
5	24600	area-System.Diagnostics	System.Diagnostics.Tests.	Failed test: System.Diagnos
6	24601	area-System.Diagnostics	System.Diagnostics.Tests.	Failed tests: * System.Dia
7	24602	area-System.Diagnostics	System.Diagnostics.Tests.	Failed test: System.Diagnos
8	24603	area-System.Diagnostics	System.Diagnostics.Tests.	Failed test: System.Diagnos
9	24606	area-System.Memory	System.Memory package	*Steps to Reproduce*: 1.
10	24608	area-System.Data	sni.dll bug or problem usi	I think there's a bug where

Create New Project

Create new .NET CORE console app project name = “Taxi”

Add NuGet Package

- Microsoft.ML



Prepare data

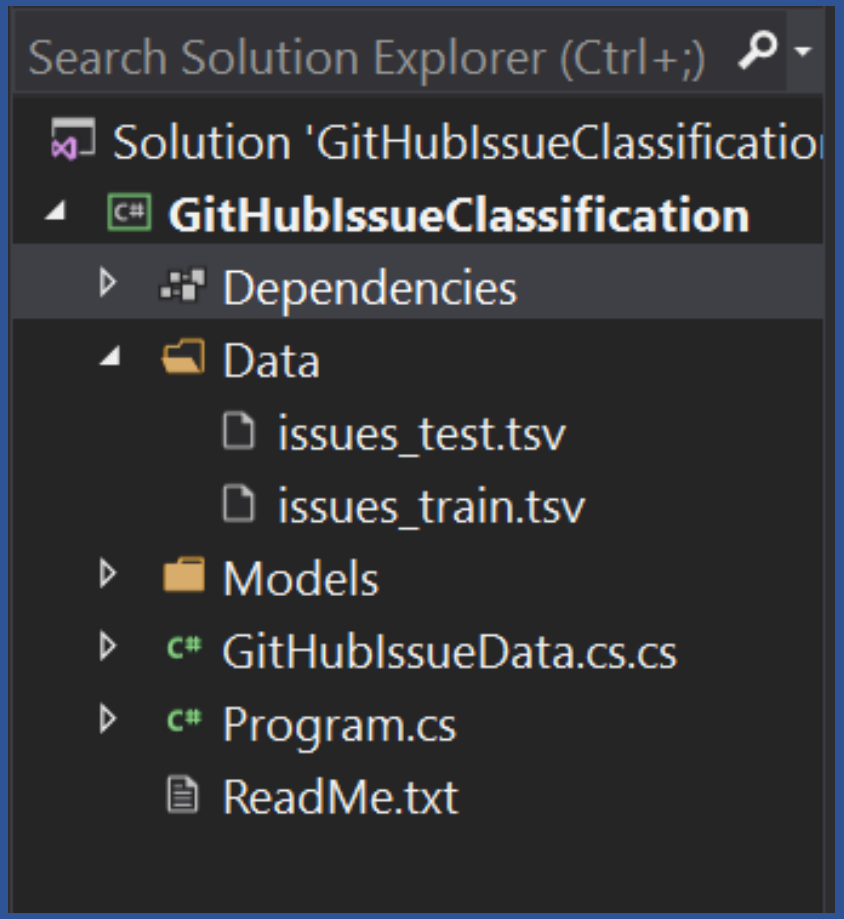
Train Data Set

https://raw.githubusercontent.com/laploy/ML.NET/master/Sentiment/yelp_labelled.txt

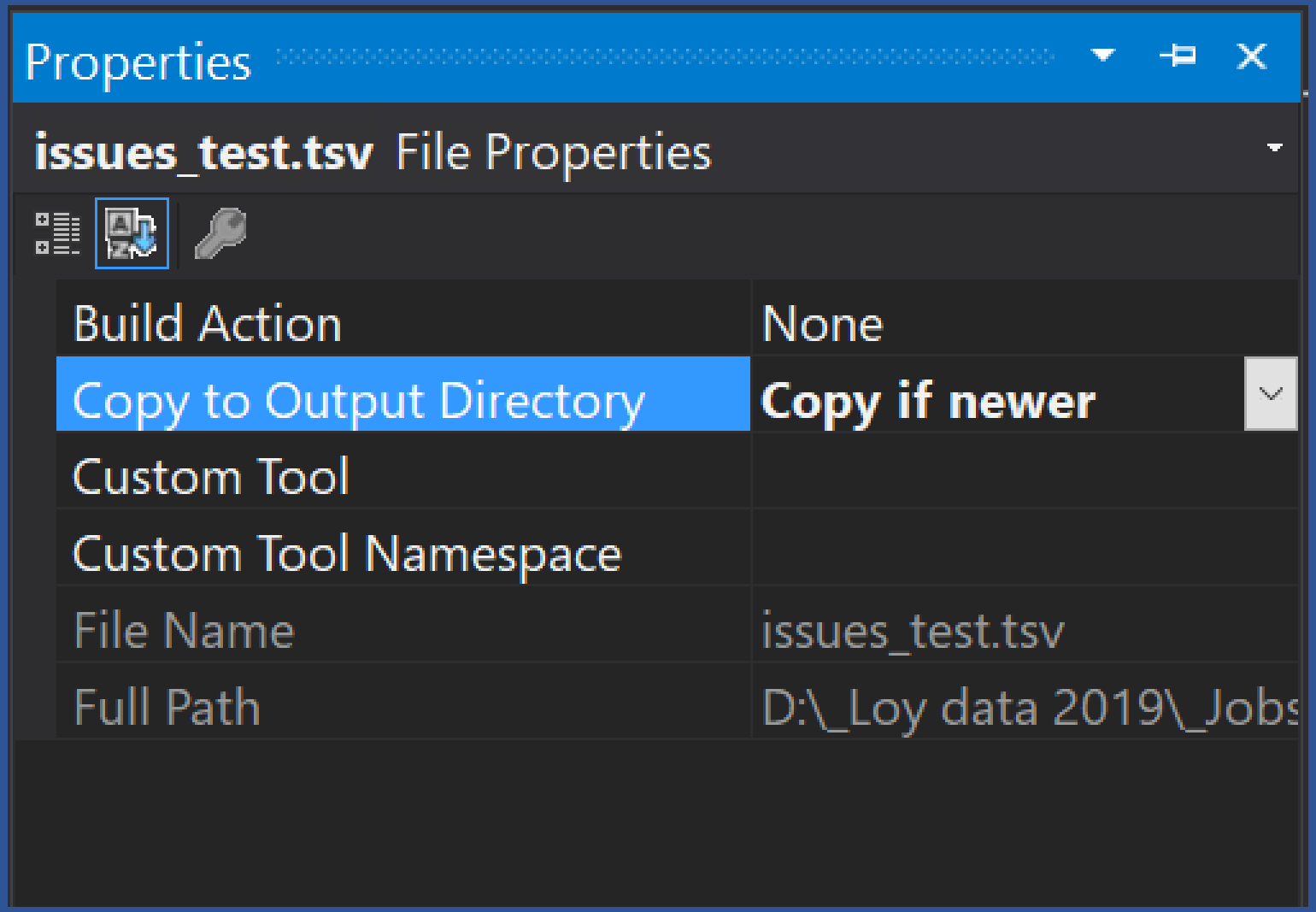
Test Data Set

https://raw.githubusercontent.com/laploy/ML.NET/master/Sentiment/BatchSentiments.txt

Add new folder “Data”
Copy datasets to this folder



Set property of each datasets to “Copy if newer”



Create data set input/output scheme

```
4   using System;
5   using System.Collections.Generic;
6   using System.Text;
7   using Microsoft.ML.Data;
8
9   namespace GitHubIssueClassification
10  {
11      // input dataset class
12      public class GitHubIssue...
23
24      // class used for prediction after th
25      // and a PredictedLabel ColumnName at
26      public class IssuePrediction
27      {
28          [ColumnName("PredictedLabel")]
29          public string Area;
30      }
31  }
```

Set data set paths

```
private static string _trainDataPath = Path.Combine(Environment.CurrentDirectory, "Data", "issues_train.tsv");
private static string _testDataPath = Path.Combine(Environment.CurrentDirectory, "Data", "issues_test.tsv");

private static string _appPath => Path.GetDirectoryName(Environment.GetCommandLineArgs()[0]);
private static string _modelPath => Path.Combine(_appPath, "..", "..", "..", "Models", "model.zip");
```

Declare class fields

```
private static MLContext _mlContext;
private static PredictionEngine<GitHubIssue, IssuePrediction> _predEngine;
private static ITransformer _trainedModel;
static IDataView _trainingDataView;
```

Create ML Context, Load data, and Process Data

```
static void Main(string[] args)
{
    // random seed ( seed: 0 ) for repeatable / deterministic
    // results across multiple trainings
    _mlContext = new MLContext(seed: 0);

    // Load the data
    _trainingDataView = _mlContext.Data.LoadFromTextFile<GitHubIssue>
        (_trainDataPath, hasHeader: true);

    // The ProcessData method executes the following tasks:
    //      Extracts and transforms the data.
    //      Returns the processing pipeline.
    var pipeline = ProcessData();
```

Build, train, and predict

```
var trainingPipeline = BuildAndTrainModel(_trainingDataView, pipeline);


// Evaluate the model
/*
The Evaluate method executes the following tasks:
        Loads the test dataset.
        Creates the multiclass evaluator.
        Evaluates the model and create metrics.
        Displays the metrics.
 */
// model created in BuildAndTrainModel is passed in to be evaluated
Evaluate(_trainingDataView.Schema);

// save model for use in later Prediction
_mlContext.Model.Save(_trainedModel, _trainingDataView.Schema, _modelPath);

// Deploy and Predict with a model
/*
The PredictIssue method executes the following tasks:
    * Creates a single issue of test data.
    * Predicts Area based on test data.
    * Combines test data and predictions for reporting.
    * Displays the predicted results.
 */
PredictIssue();
```

The program output result

```
========== Training Start 07:42:34.183  ==========
========== Training Done 07:43:12.153  ==========

************************************************************
* Metrics for Multi-class Classification model - Test Data
*-----------------------------------------------------------
* MicroAccuracy: 0.739
* MacroAccuracy: 0.676
* LogLoss: .92
* LogLossReduction: .643
************************************************************

=============== Single Prediction - Result: area-System.Data =========
```

Exercise: Make batch prediction

```
==== Prediction with multiple rows from file ===
-----------------------------------------------
Actual price: area-System.Net           | Predicted price: area-System.Net
Actual price: area-System.Diagnostics           | Predicted price: area-System.Diagnostics
Actual price: area-System.Memory                | Predicted price: area-System.Memory
Actual price: area-System.Data          | Predicted price: area-System.Data
Actual price: area-System.Net           | Predicted price: area-System.Net
Actual price: area-Serialization                | Predicted price: area-Serialization
Actual price: area-Infrastructure               | Predicted price: area-Infrastructure
Actual price: area-System.IO            | Predicted price: area-System.IO
=============== End of predictions ===============
```

What's next?