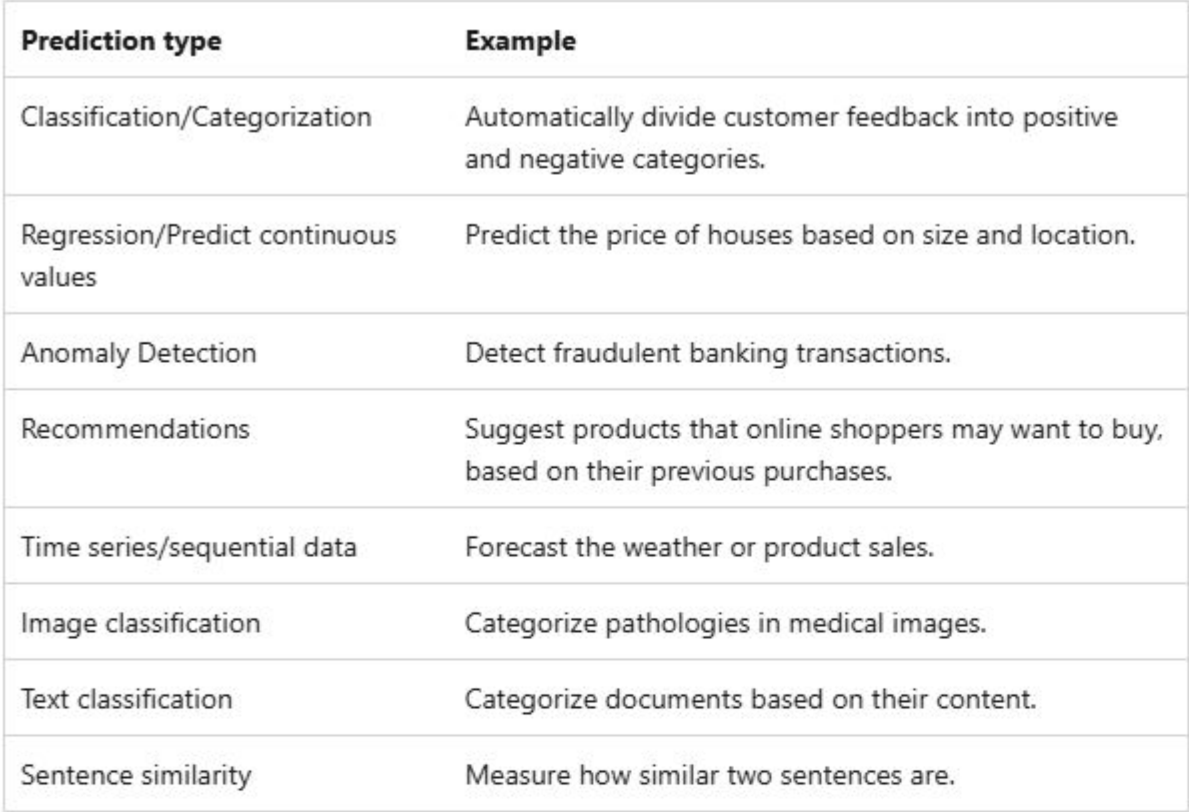
Machine learning .Net

https://learn.microsoft.com/en-us/dotnet/machine-learning

What is **ML.Net**?

- ❑ Open-source, cross-platform machine learning framework for .NET

Scenario - Machine learning task mapping

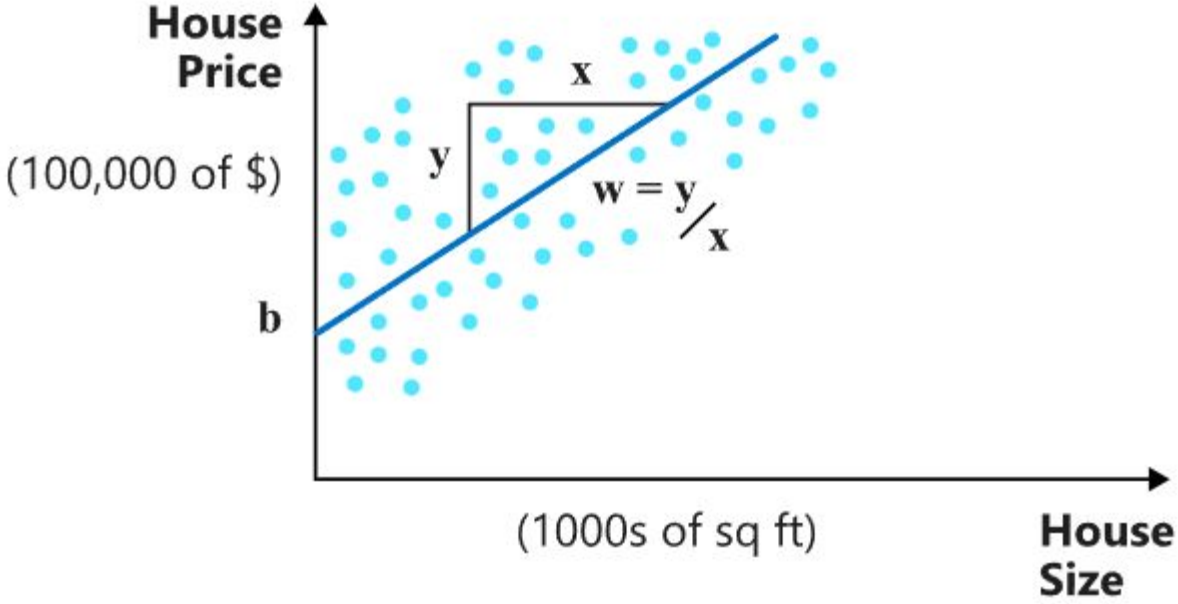


Prediction type	Example
Classification/Categorization	Automatically divide customer feedback into positive and negative categories.
Regression/Predict continuous values	Predict the price of houses based on size and location.
Anomaly Detection	Detect fraudulent banking transactions.
Recommendations	Suggest products that online shoppers may want to buy, based on their previous purchases.
Time series/sequential data	Forecast the weather or product sales.
Image classification	Categorize pathologies in medical images.
Text classification	Categorize documents based on their content.
Sentence similarity	Measure how similar two sentences are.

Hello ML world

What is ML.NET and how does it work? - ML.NET | Microsoft Learn

Linear regression - House price prediction

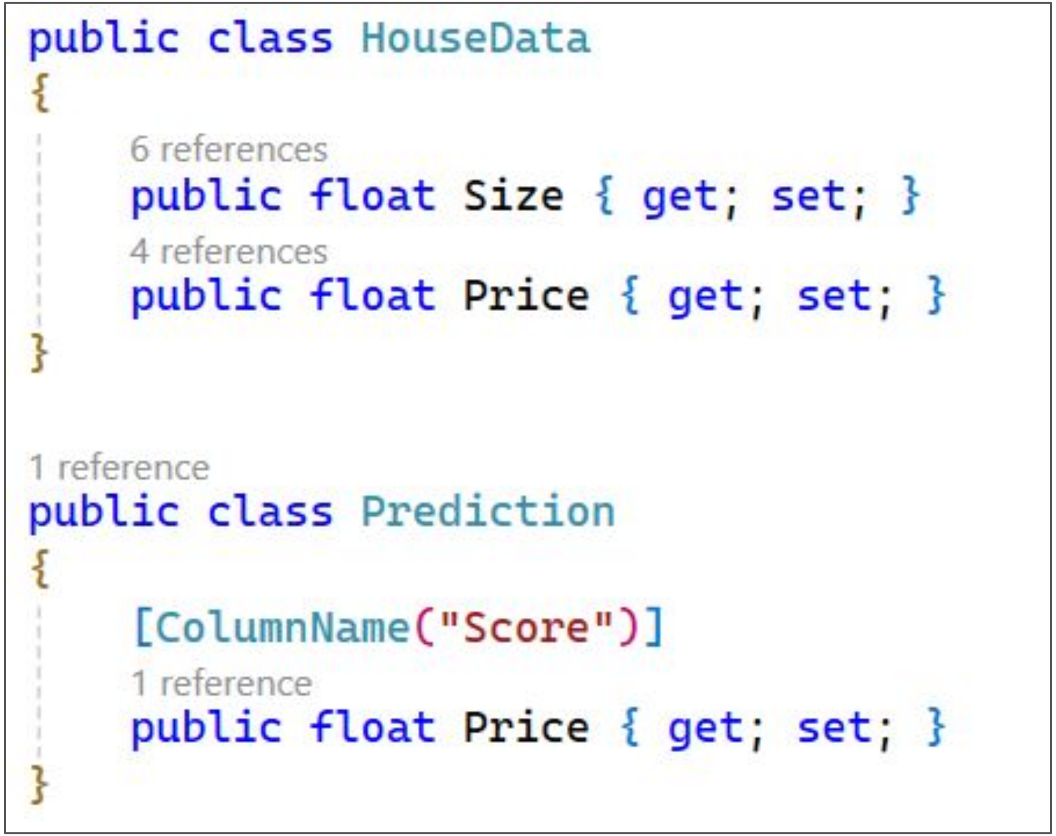


$$Price = b + Size * w.$$

Some basic terms

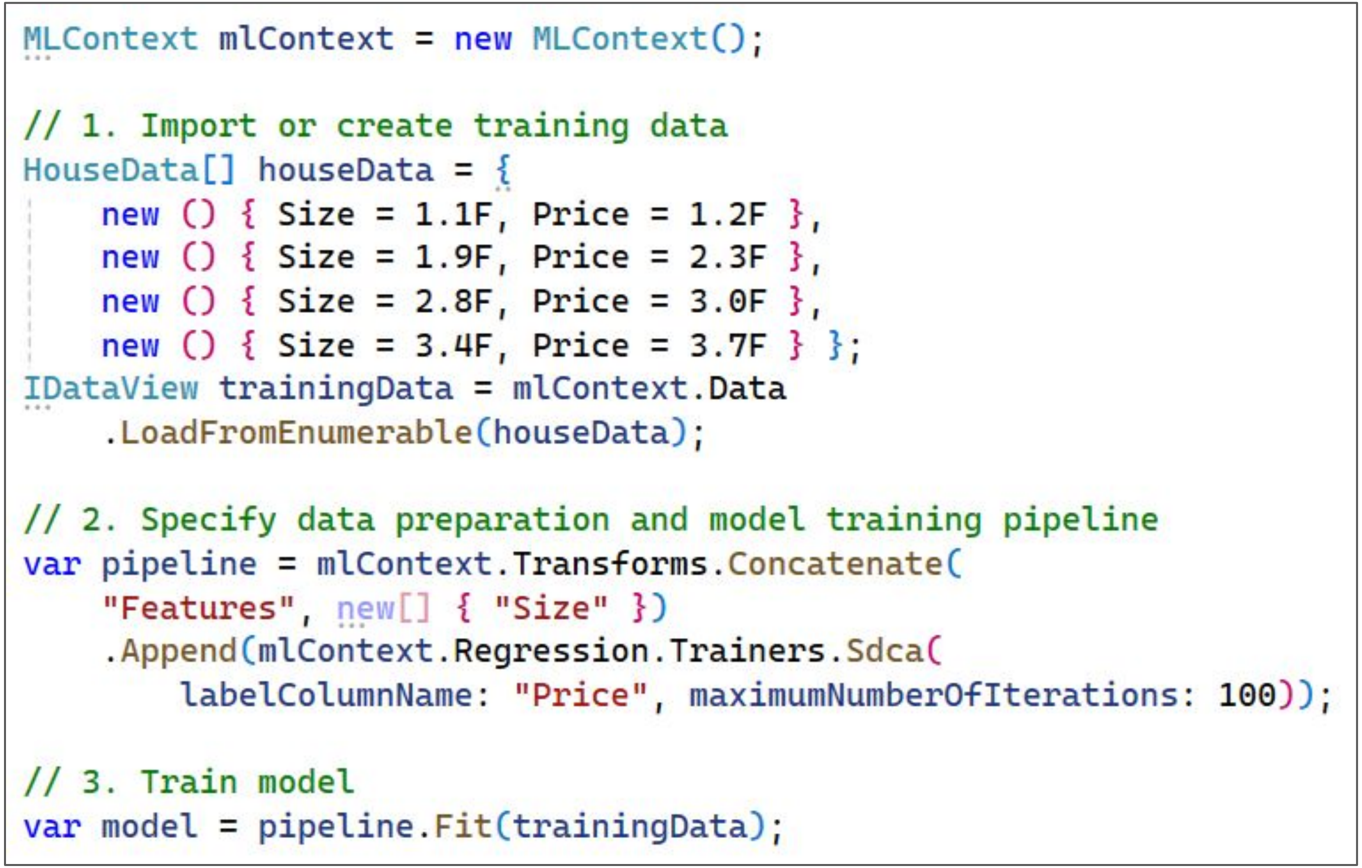
- ❑ **Training data**: The data used to find the parameters of the model
- ❑ **Features**: The inputs of a machine learning model are called.
- ❑ **Labels**: The ground-truth values used to train a machine learning model

1. Model preparation



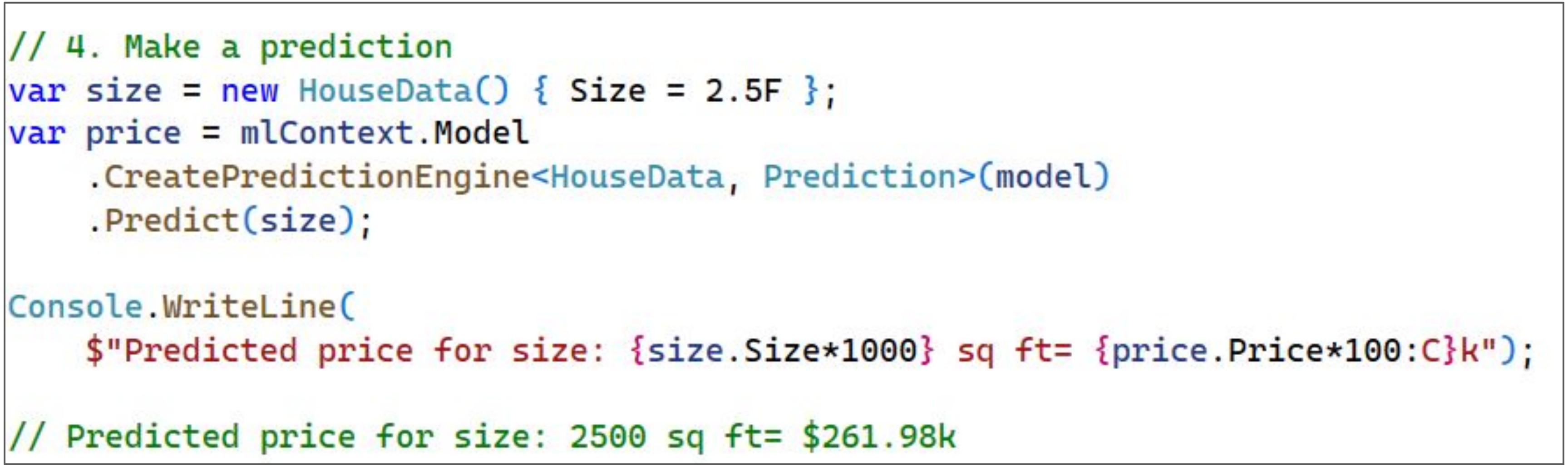
```
public class HouseData
{
    6 references
    public float Size { get; set; }
    4 references
    public float Price { get; set; }
}

1 reference
public class Prediction
{
    [ColumnName("Score")]
    1 reference
    public float Price { get; set; }
}
```

2. Prepare data & train



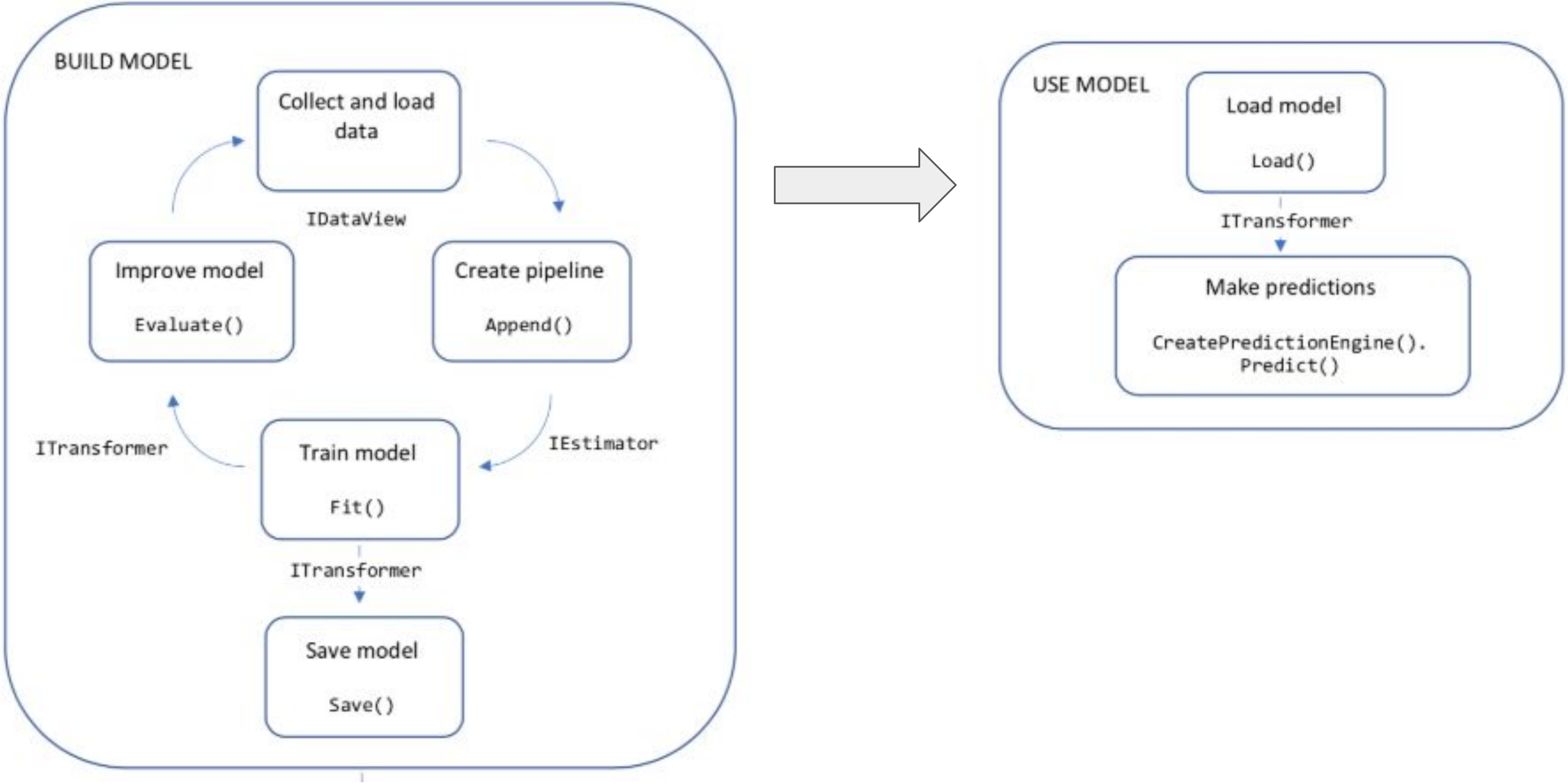
```
MLContext mlContext = new MLContext();

// 1. Import or create training data
HouseData[] houseData = {
    new () { Size = 1.1F, Price = 1.2F },
    new () { Size = 1.9F, Price = 2.3F },
    new () { Size = 2.8F, Price = 3.0F },
    new () { Size = 3.4F, Price = 3.7F } };
IDataView trainingData = mlContext.Data
    .LoadFromEnumerable(houseData);

// 2. Specify data preparation and model training pipeline
var pipeline = mlContext.Transforms.Concatenate(
    "Features", new[] { "Size" })
    .Append(mlContext.Regression.Trainers.Sdca(
        labelColumnName: "Price", maximumNumberOfIterations: 100));

// 3. Train model
var model = pipeline.Fit(trainingData);
```

3. Make prediction



```
// 4. Make a prediction
var size = new HouseData() { Size = 2.5F };
var price = mlContext.Model
    .CreatePredictionEngine<HouseData, Prediction>(model)
    .Predict(size);

Console.WriteLine(
    $"Predicted price for size: {size.Size*1000} sq ft= {price.Price*100:C}k");

// Predicted price for size: 2500 sq ft= $261.98k
```

Workflow



BUILD MODEL

Collect and load data

IDataView

Create pipeline

Append()

IEstimator

Train model

Fit()

ITransformer

Improve model

Evaluate()

ITransformer

Save model

Save()

USE MODEL

Load model

Load()

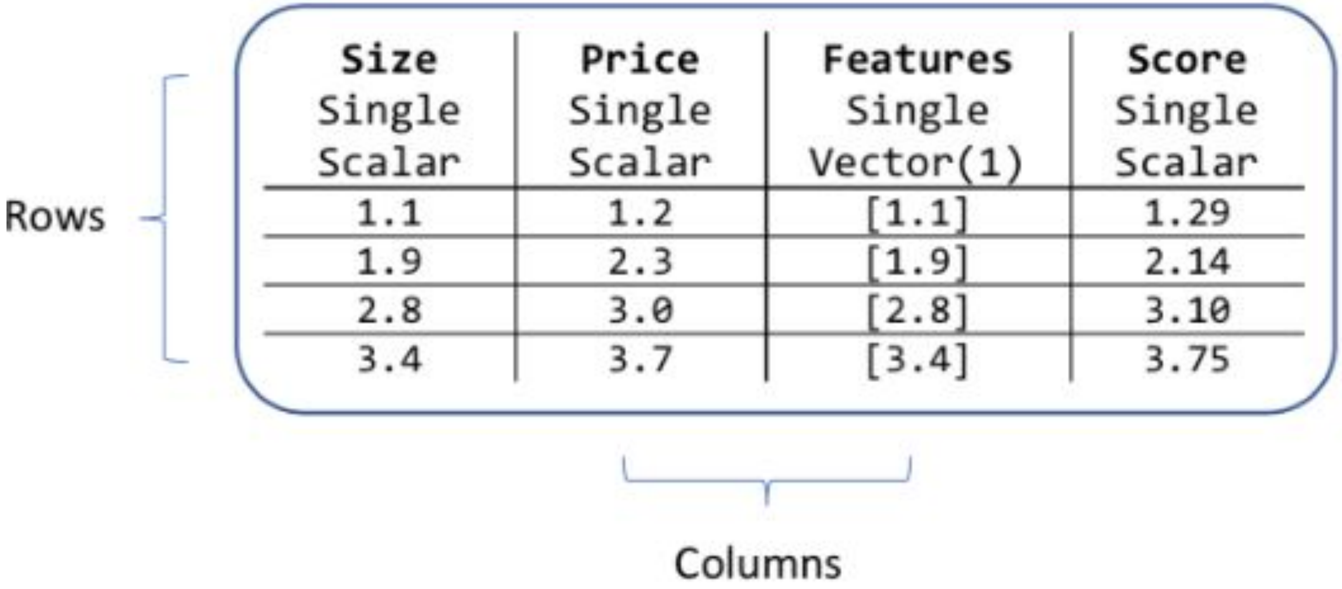
ITransformer

Make predictions

CreatePredictionEngine().Predict()

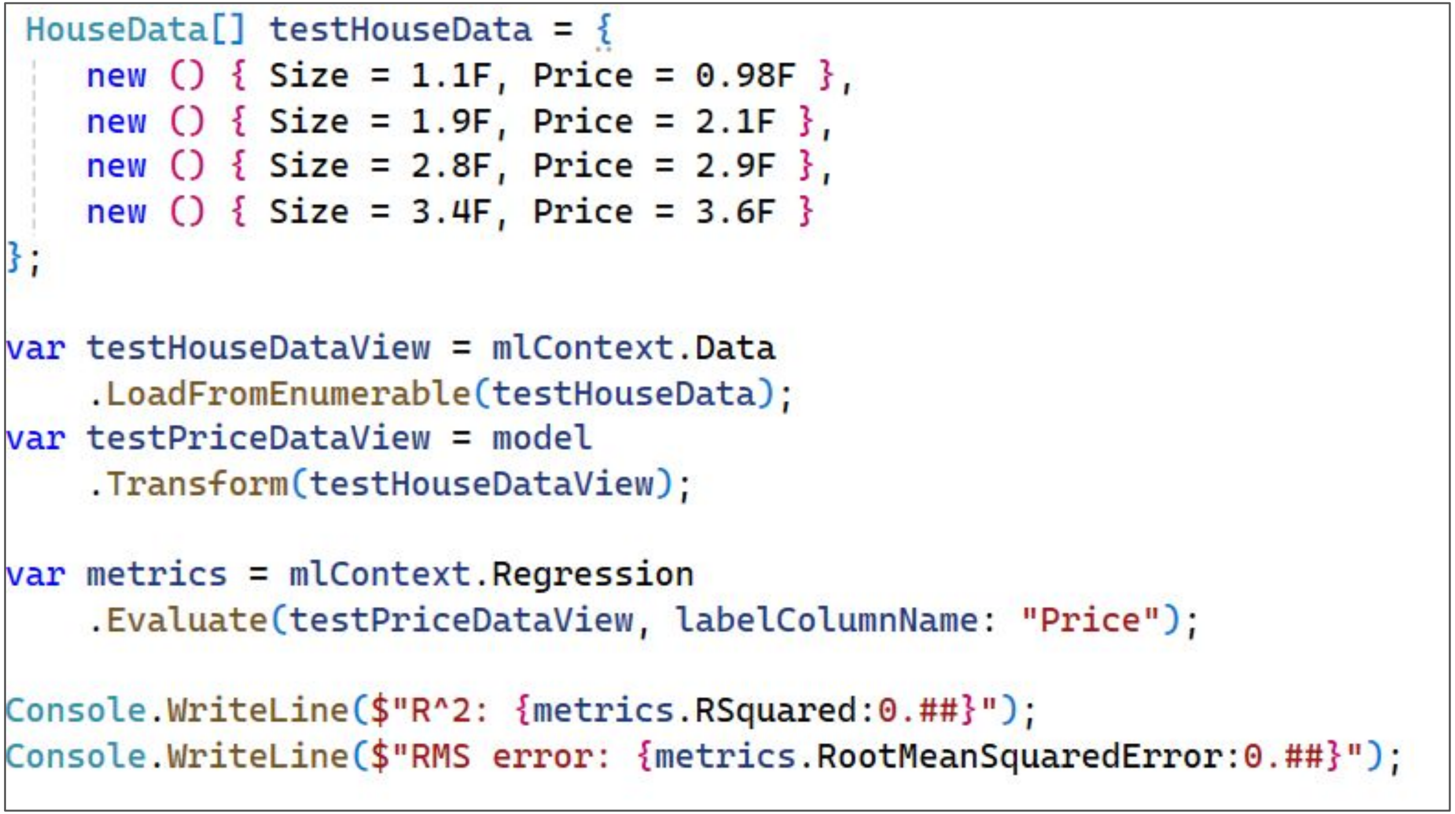
Data models & schema

- ❏ DataView



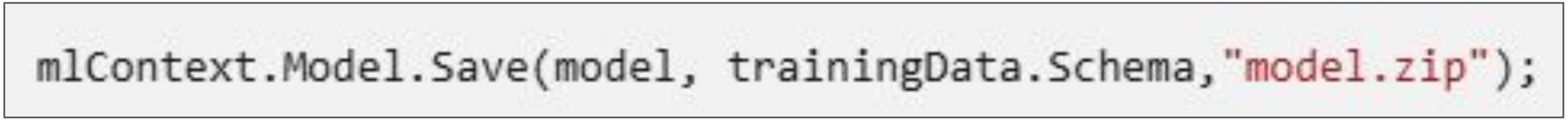
Size Single Scalar	Price Single Scalar	Features Single Vector(1)	Score Single Scalar
1.1	1.2	[1.1]	1.29
1.9	2.3	[1.9]	2.14
2.8	3.0	[2.8]	3.10
3.4	3.7	[3.4]	3.75

4. Model evaluation using test data

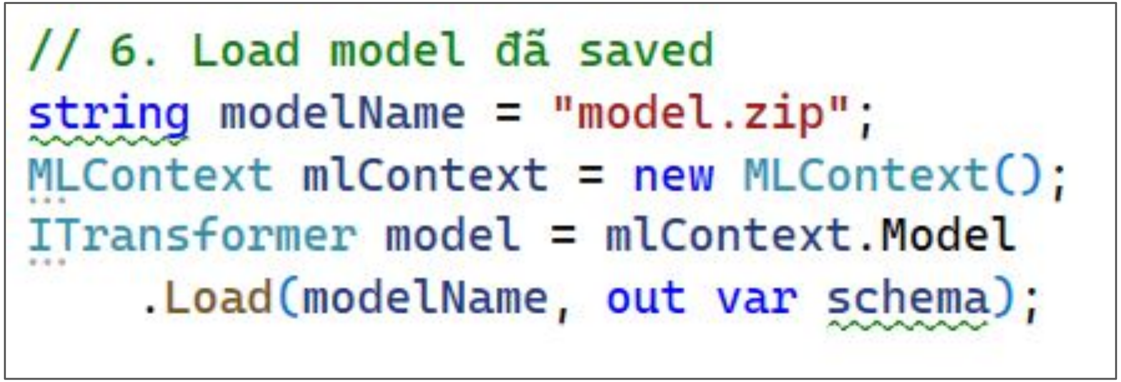


```
 HouseData[] testHouseData = {
    new () { Size = 1.1F, Price = 0.98F },
    new () { Size = 1.9F, Price = 2.1F },
    new () { Size = 2.8F, Price = 2.9F },
    new () { Size = 3.4F, Price = 3.6F }
};

var testHouseDataView = mlContext.Data
    .LoadFromEnumerable(testHouseData);
var testPriceDataView = model
    .Transform(testHouseDataView);

var metrics = mlContext.Regression
    .Evaluate(testPriceDataView, labelColumnName: "Price");

Console.WriteLine($"R^2: {metrics.RSquared:0.##}");
Console.WriteLine($"RMS error: {metrics.RootMeanSquaredError:0.##}");
```

5. Model deployment

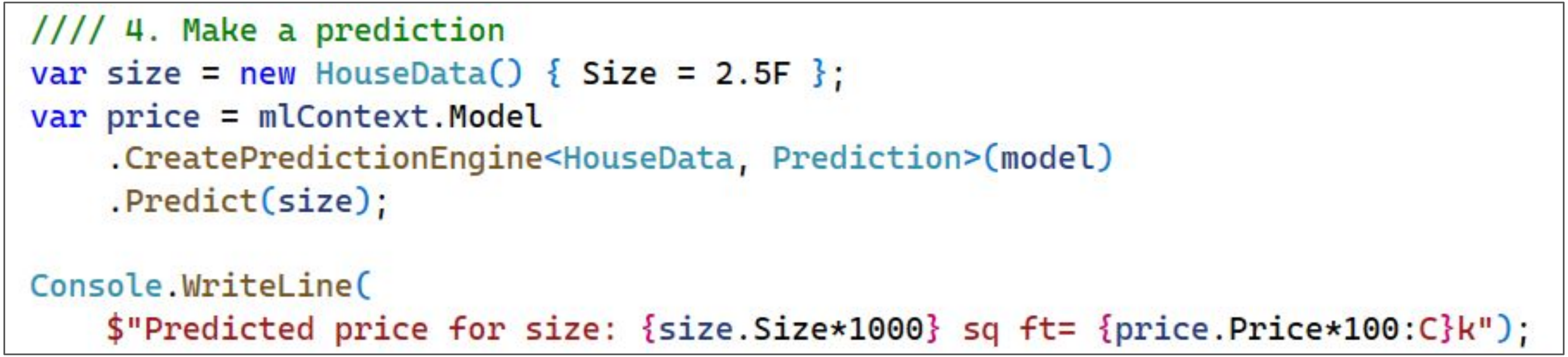


```
mlContext.Model.Save(model, trainingData.Schema,"model.zip");
```



```
// 6. Load model đã saved
string modelName = "model.zip";
MLContext mlContext = new MLContext();
ITransformer model = mlContext.Model
    .Load(modelName, out var schema);
```

6. Make prediction after loading model

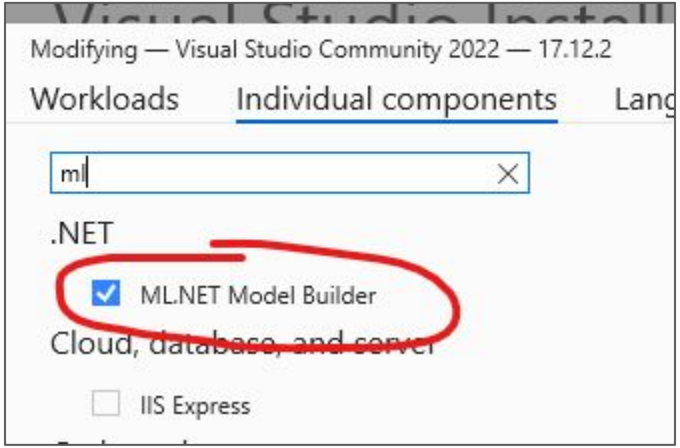


```
//// 4. Make a prediction
var size = new HouseData() { Size = 2.5F };
var price = mlContext.Model
    .CreatePredictionEngine<HouseData, Prediction>(model)
    .Predict(size);

Console.WriteLine(
    $"Predicted price for size: {size.Size*1000} sq ft= {price.Price*100:C}k");
```

Model Builder

ML.NET Tutorial | Get started in 10 minutes | .NET

1. Installation



Visual Studio Install

Modifying — Visual Studio Community 2022 — 17.12.2

ml

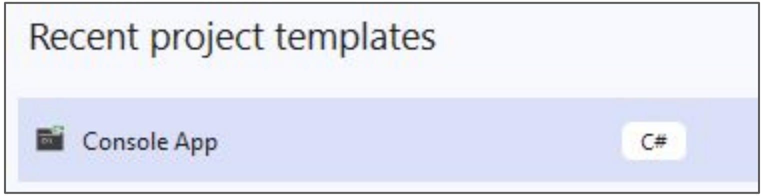
.NET

☑ ML.NET Model Builder

Cloud, database, and server

□ IIS Express

2. Create a C# console app

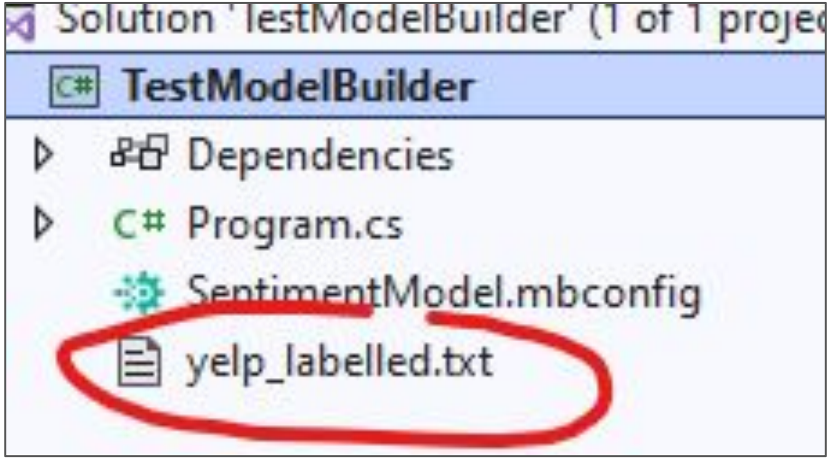


Recent project templates

Console App C#

3. Download data

- ❑ Dataset: [Sentiment Labelled Sentences]
- ❑ Author: UCI Machine Learning Repository
- ❑ Uncompress, add file **yelp_labelled.txt** to your project



Solution 'TestModelBuilder' (1 of 1 projec

TestModelBuilder

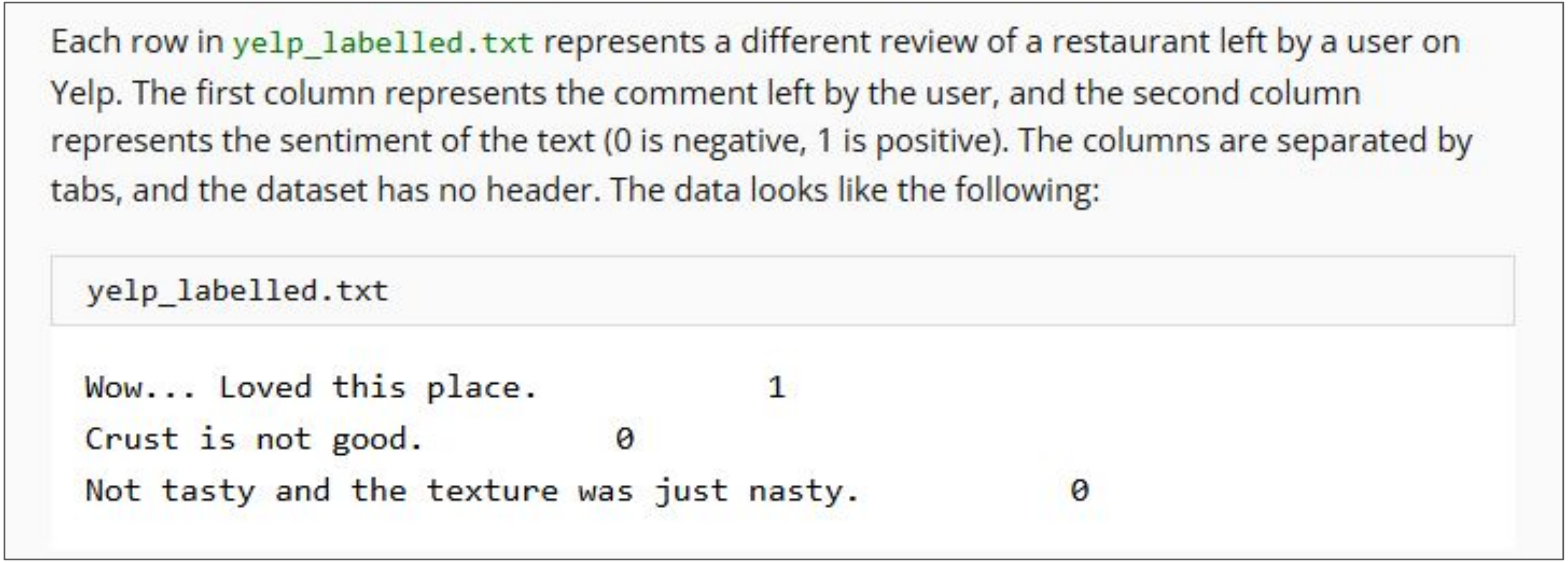
Dependencies

Program.cs

SentimentModel.mbconfig

yelp_labelled.txt

Data exploration



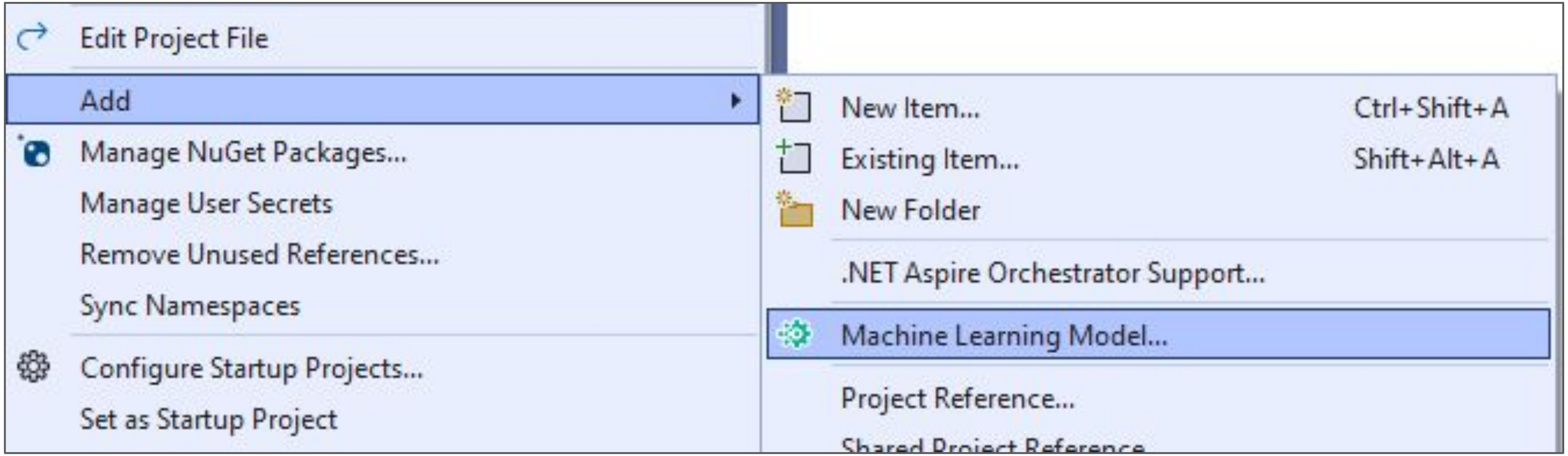
Each row in `yelp_labelled.txt` represents a different review of a restaurant left by a user on Yelp. The first column represents the comment left by the user, and the second column represents the sentiment of the text (0 is negative, 1 is positive). The columns are separated by tabs, and the dataset has no header. The data looks like the following:

```
yelp_labelled.txt
```

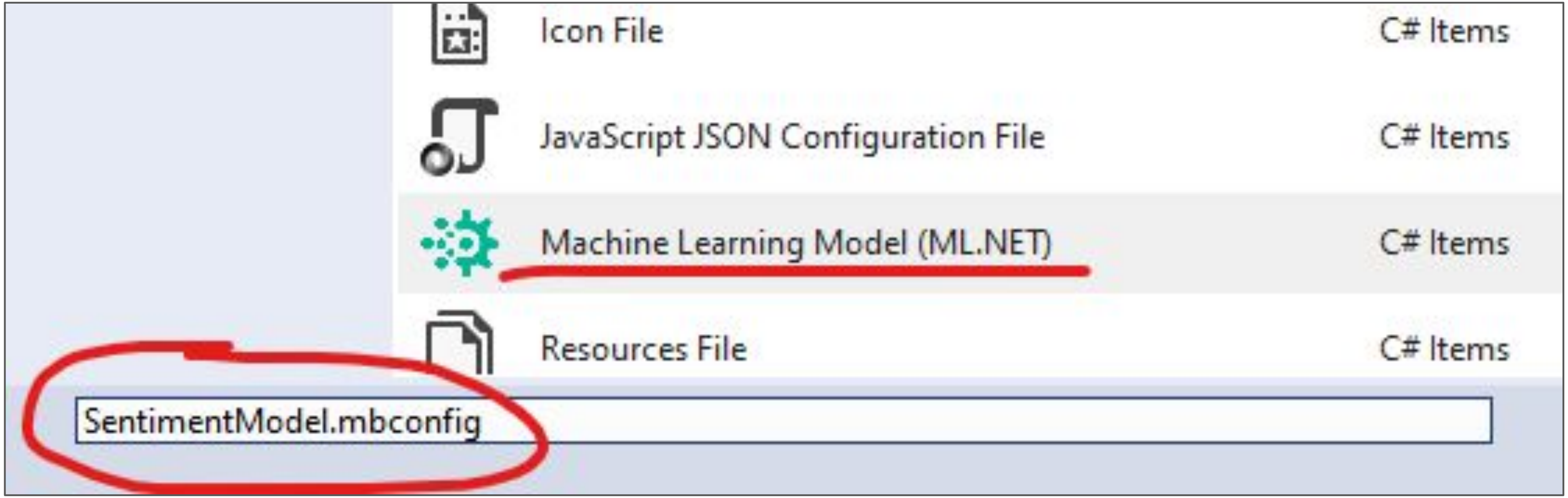
```
Wow... Loved this place.          1
Crust is not good.         0
Not tasty and the texture was just nasty.         0
```

4. Add machine learning

- ❑ Right click on the project
- ❑ Add > Machine learning model...



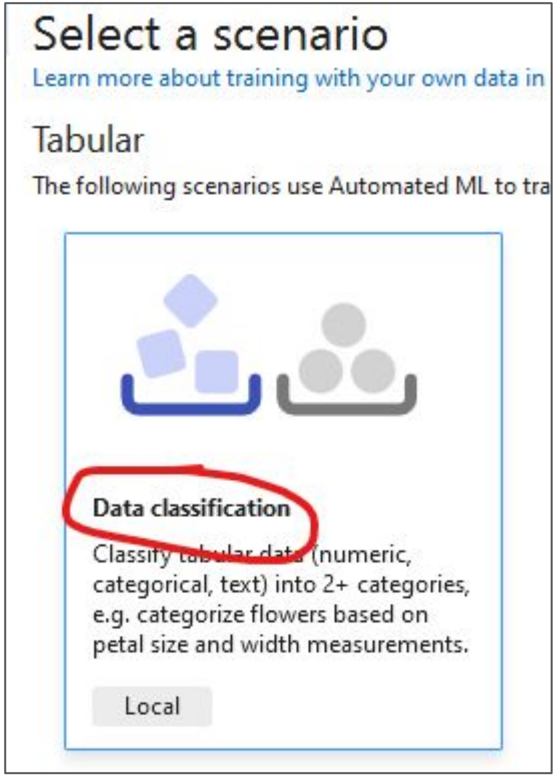
5. Name your model: SentimentModel



Icon File
C# Items
JavaScript JSON Configuration File
C# Items
Machine Learning Model (ML.NET)
C# Items
Resources File
C# Items

SentimentModel.mbconfig

6. Config your training



Select a scenario

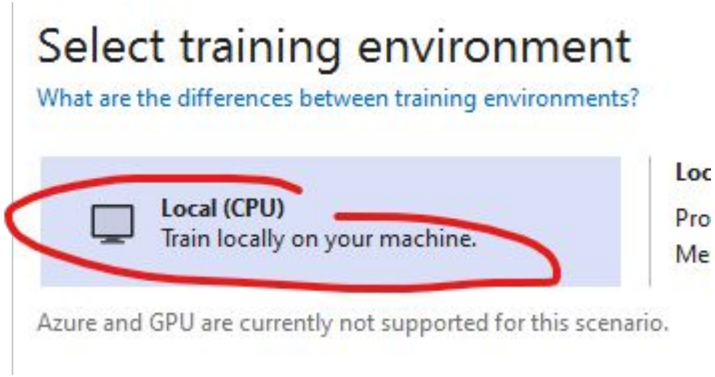
Learn more about training with your own data in

Tabular

The following scenarios use Automated ML to tra

[✓ Data classification]

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



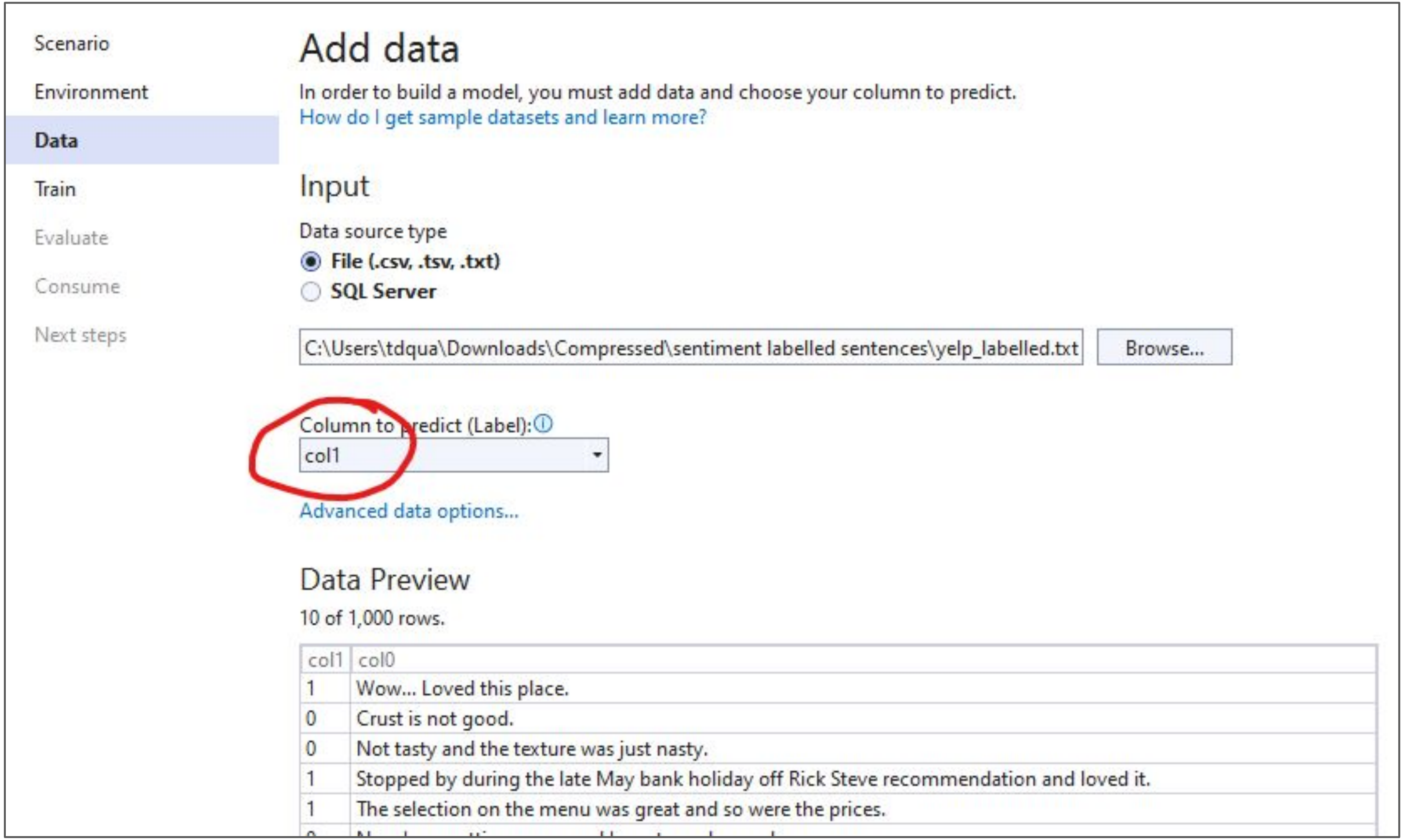
Select training environment

What are the differences between training environments?

[✓ Local (CPU)
Train locally on your machine.]

Azure and GPU are currently not supported for this scenario.

7. Add data



Add data

In order to build a model, you must add data and choose your column to predict.

How do I get sample datasets and learn more?

Input

Data source type

● File (.csv, .tsv, .txt)

○ SQL Server

C:\Users\tdqua\Downloads\Compressed\sentiment labelled sentences\yelp_labelled.txt

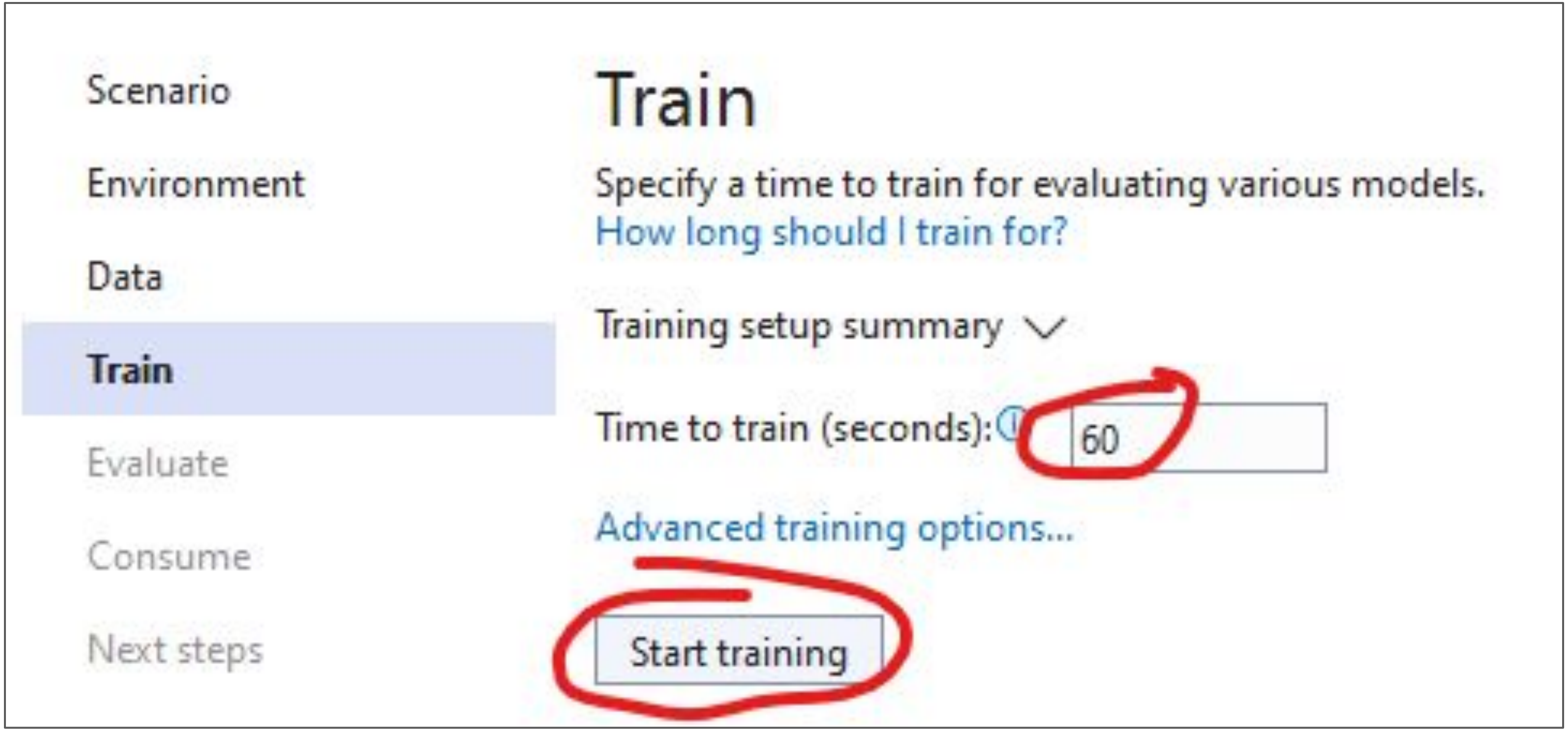
Column to predict (Label): col1

Data Preview

10 of 1,000 rows.

col1	col0
1	Wow... Loved this place.
0	Crust is not good.
0	Not tasty and the texture was just nasty.
1	Stopped by during the late May bank holiday off Rick Steve recommendation and loved it.
1	The selection on the menu was great and so were the prices.

8. Train your model



Train

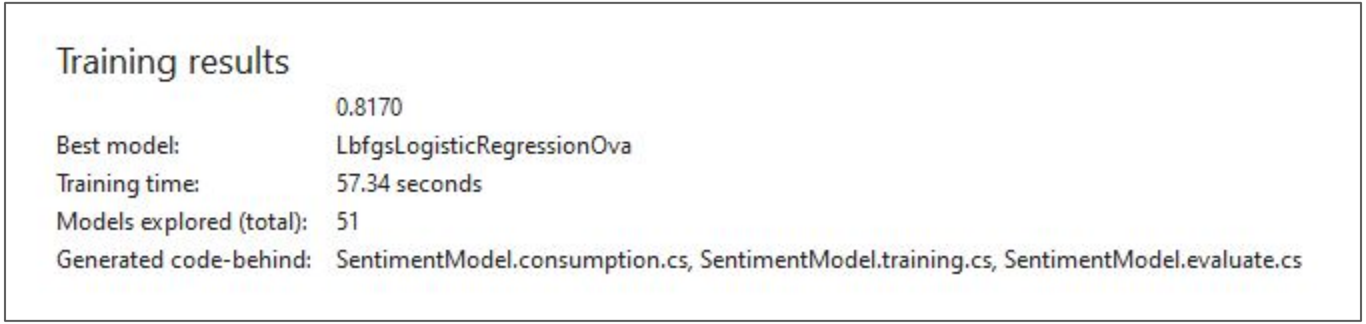
Specify a time to train for evaluating various models.
How long should I train for?

Training setup summary

Time to train (seconds): 60

Advanced training options...

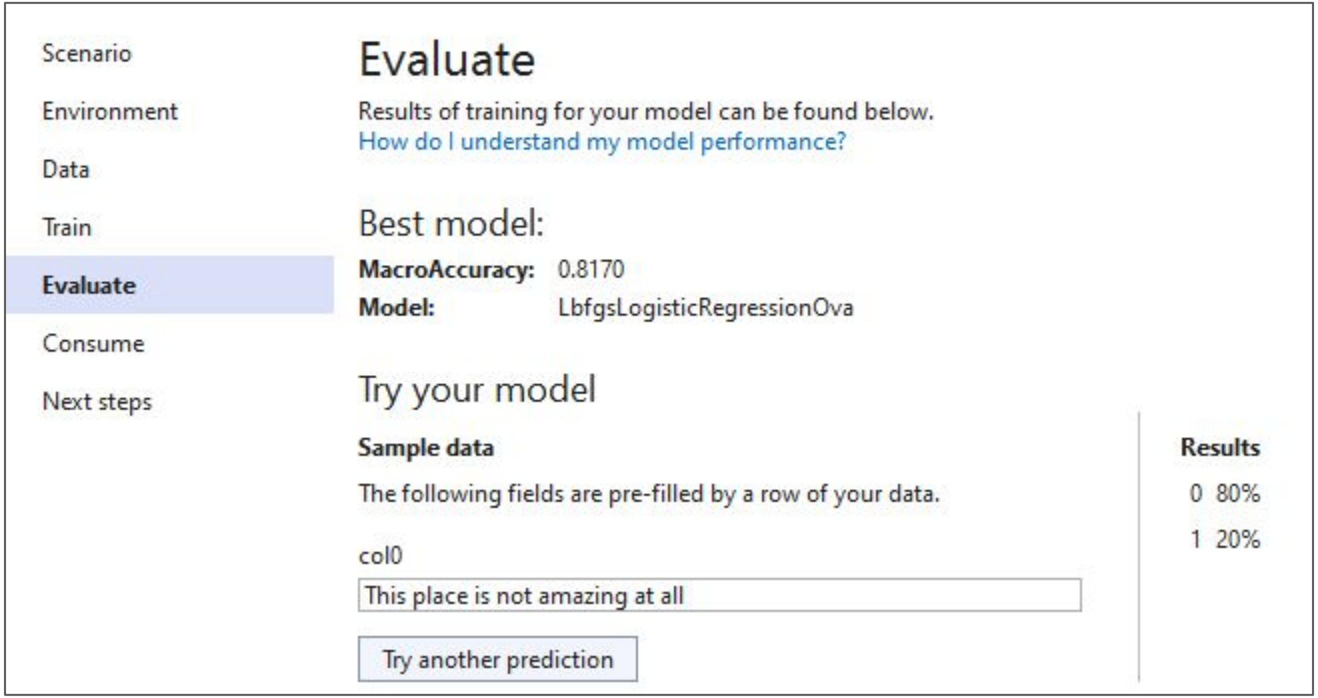
9. Training result



Training results

	0.8170
Best model:	LbfgsLogisticRegressionOva
Training time:	57.34 seconds
Models explored (total):	51
Generated code-behind:	SentimentModel.consumption.cs, SentimentModel.training.cs, SentimentModel.evaluate.cs

10. Evaluate the model



Evaluate

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

Best model:

MacroAccuracy: 0.8170
Model: LbfgsLogisticRegressionOva

Try your model

Sample data

The following fields are pre-filled by a row of your data.

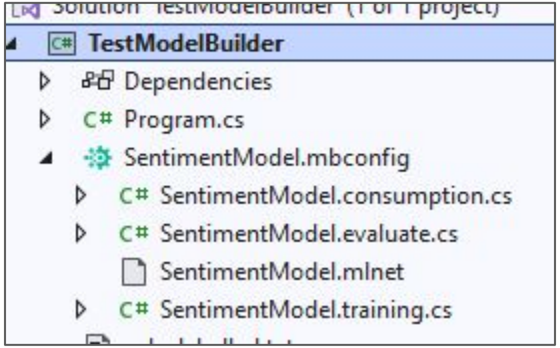
col0

This place is not amazing at all

Results

0	80%
1	20%

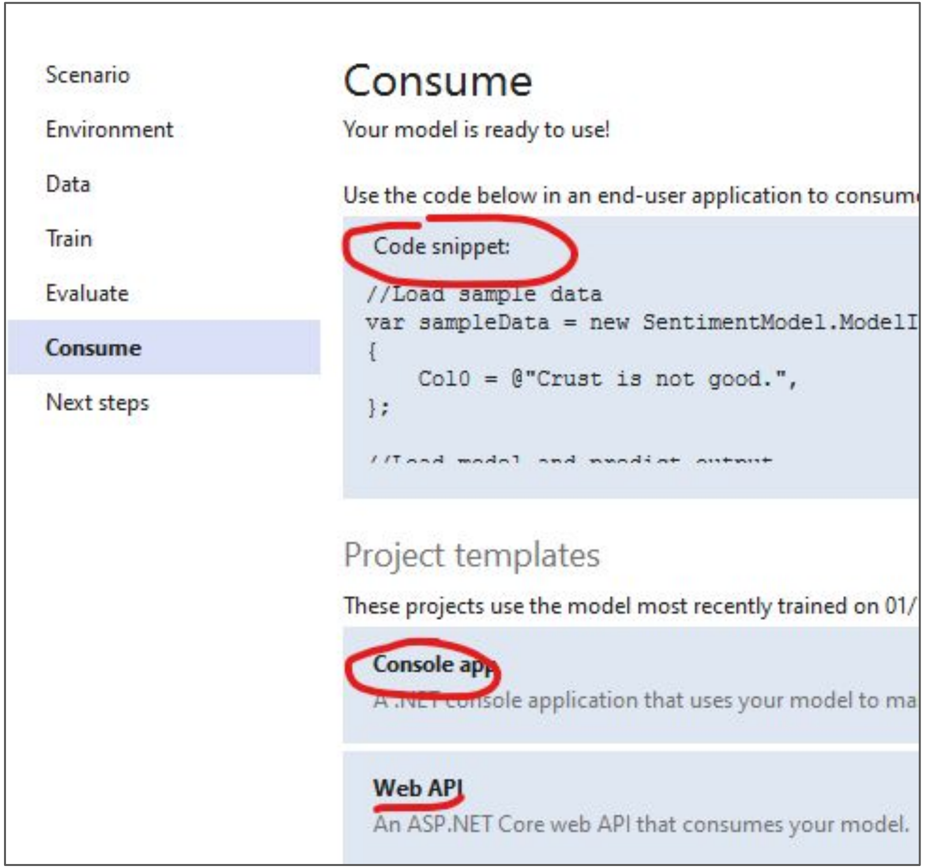
11. Auto generated code



Solution 'TestModelBuilder' (1 of 1 project)
- TestModelBuilder
  - Dependencies
  - Program.cs
  - SentimentModel.mbconfig
    - SentimentModel.consumption.cs
    - SentimentModel.evaluate.cs
    - SentimentModel.mlnet
    - SentimentModel.training.cs

- ❑ **SentimentModel.consumption.cs**: model input & output classes & a Predict method that can be used for model consumption.
- ❑ **SentimentModel.evaluate.cs**: CalculatePFI method that uses the Permutation Feature Importance (PFI) technique to evaluate which features contribute most to the model predictions
- ❑ **SentimentModel.mlnet:** This file is the trained ML.NET model, which is a serialized zip file.
- ❑ **SentimentModel.training.cs:** code to understand the importance input columns have on your model predictions

12. Consume your model



Consume

Your model is ready to use!

Use the code below in an end-user application to consum

Code snippet:

```
//Load sample data
var sampleData = new SentimentModel.ModelI
{
    Col0 = @"Crust is not good.",
};

//Load model and predict output
```

Project templates

These projects use the model most recently trained on 01/

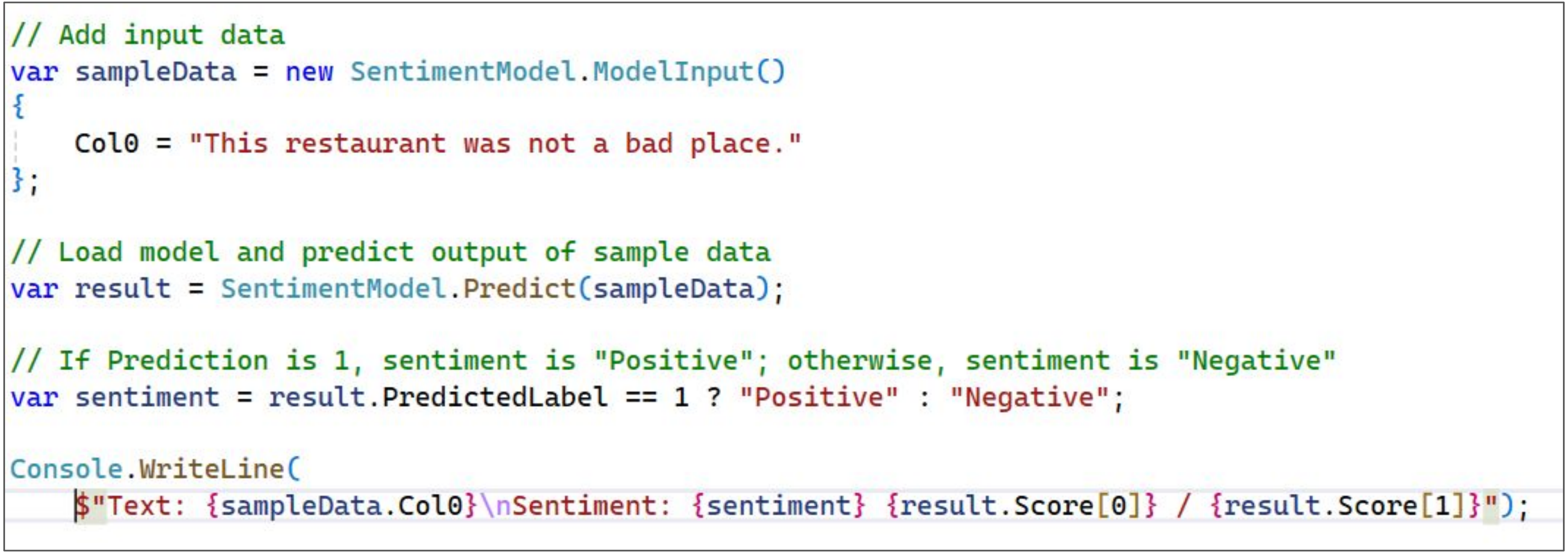
Console app

A .NET console application that uses your model to ma

Web API

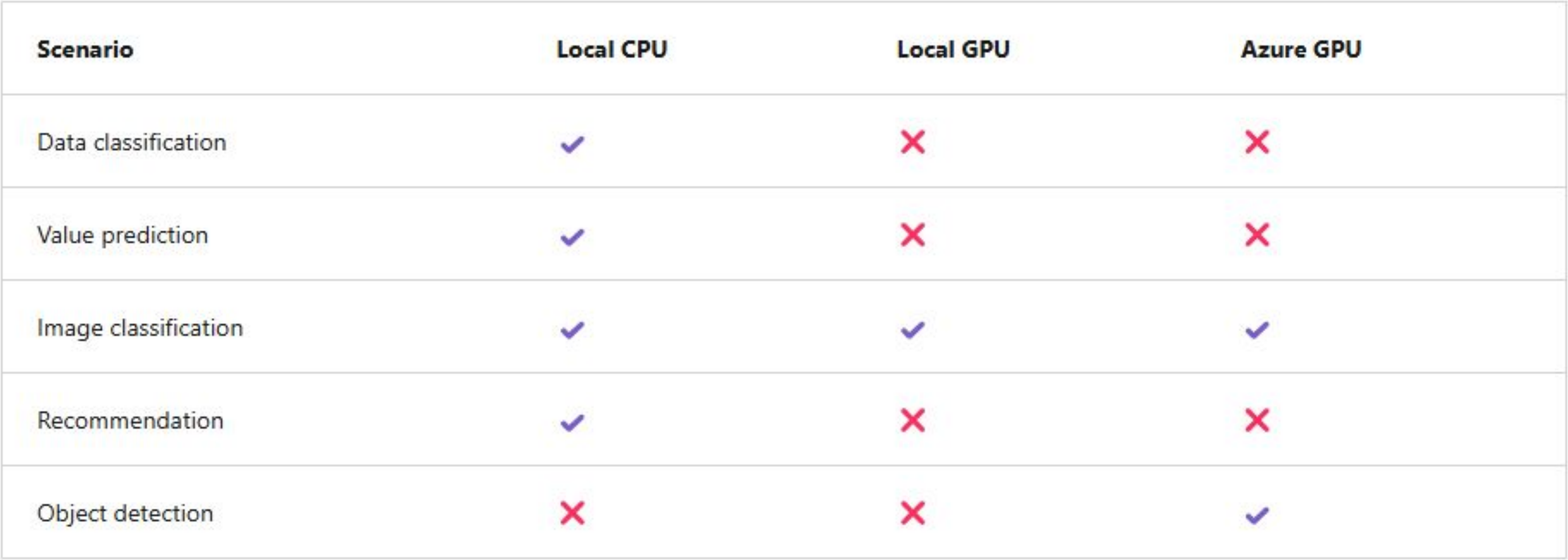
An ASP.NET Core web API that consumes your model.

Testing with generated code snippet



```
// Add input data
var sampleData = new SentimentModel.ModelInput()
{
    Col0 = "This restaurant was not a bad place."
};

// Load model and predict output of sample data
var result = SentimentModel.Predict(sampleData);

// If Prediction is 1, sentiment is "Positive"; otherwise, sentiment is "Negative"
var sentiment = result.PredictedLabel == 1 ? "Positive" : "Negative";

Console.WriteLine(
    $"Text: {sampleData.Col0}\nSentiment: {sentiment} {result.Score[0]} / {result.Score[1]}");
```

Supported environments in Model Builder



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

Types of problem to use with Model Builder

- ❑ **Categorizing data**: Organize news articles by topic.
- ❑ **Predicting a numerical value**: Estimate the price of a home.
- ❑ **Grouping items with similar characteristics**: Segment customers.
- ❑ **Recommending items**: Recommend movies.
- ❑ **Classifying images**: Tag an image based on its contents.
- ❑ **Detecting objects in an image**: Detect pedestrians and bicycles at an intersection

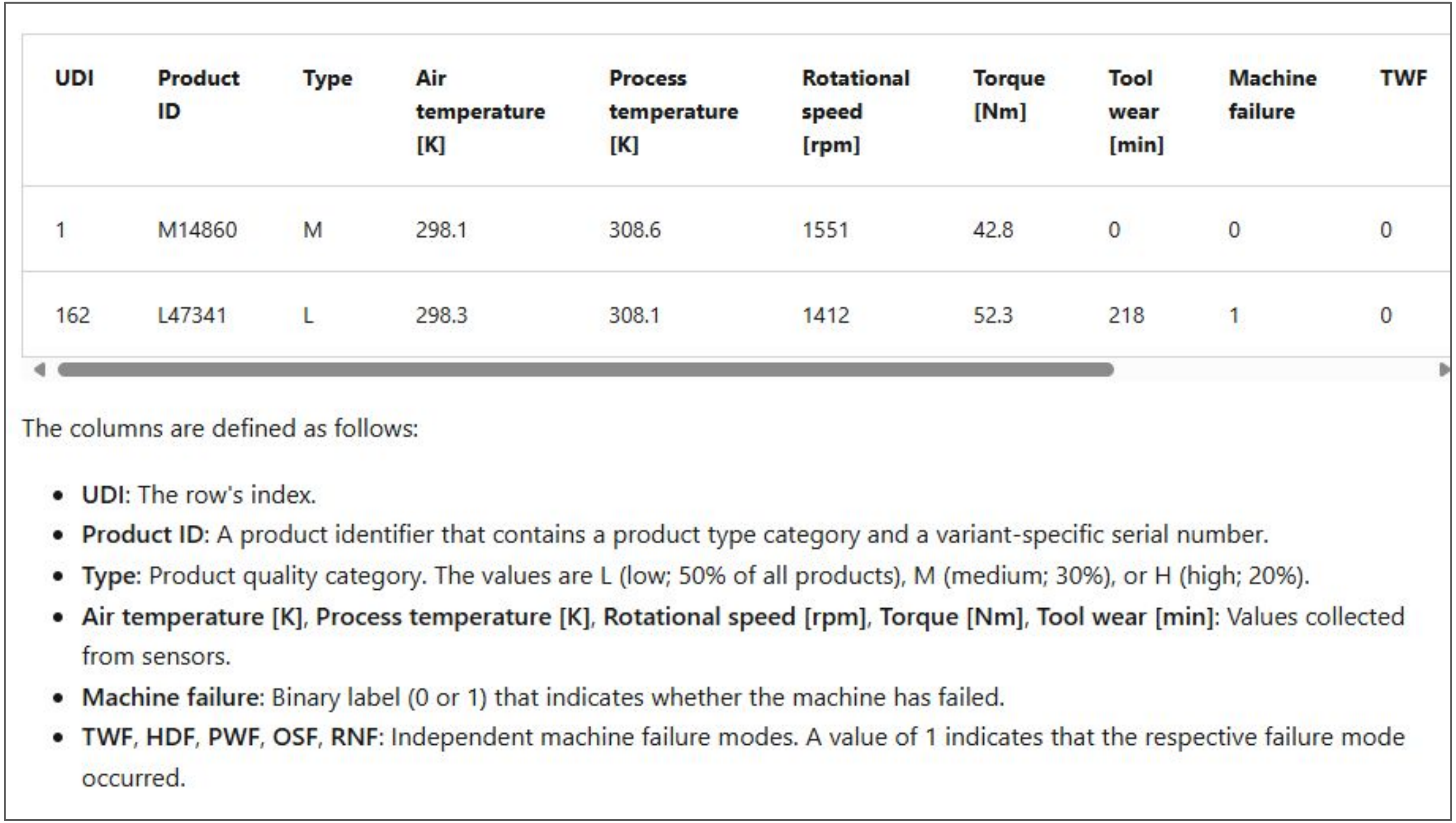
Train a predictive maintenance model

Train a machine learning model for predictive maintenance by using ML.NET Model Builder - Training | Microsoft Learn

1. Prepare dataset

- ❑ AI4I 2020 Predictive Maintenance
- ❑ 10,000 data points and 14 columns.

Data understanding



UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure	TWF
1	M14860	M	298.1	308.6	1551	42.8	0	0	0
162	L47341	L	298.3	308.1	1412	52.3	218	1	0

The columns are defined as follows:

- **UDI**: The row's index.
- **Product ID**: A product identifier that contains a product type category and a variant-specific serial number.
- **Type**: Product quality category. The values are L (low; 50% of all products), M (medium; 30%), or H (high; 20%).
- **Air temperature [K]**, **Process temperature [K]**, **Rotational speed [rpm]**, **Torque [Nm]**, **Tool wear [min]**: Values collected from sensors.
- **Machine failure**: Binary label (0 or 1) that indicates whether the machine has failed.
- **TWF**, **HDF**, **PWF**, **OSF**, **RNF**: Independent machine failure modes. A value of 1 indicates that the respective failure mode occurred.

Data analysis

Not all the columns in the dataset are needed

Because you want to be able to predict whether a machine will fail or not, the Machine failure column is the label. In Model Builder, for features, you can use data from the Product ID, Type, and various sensor columns.

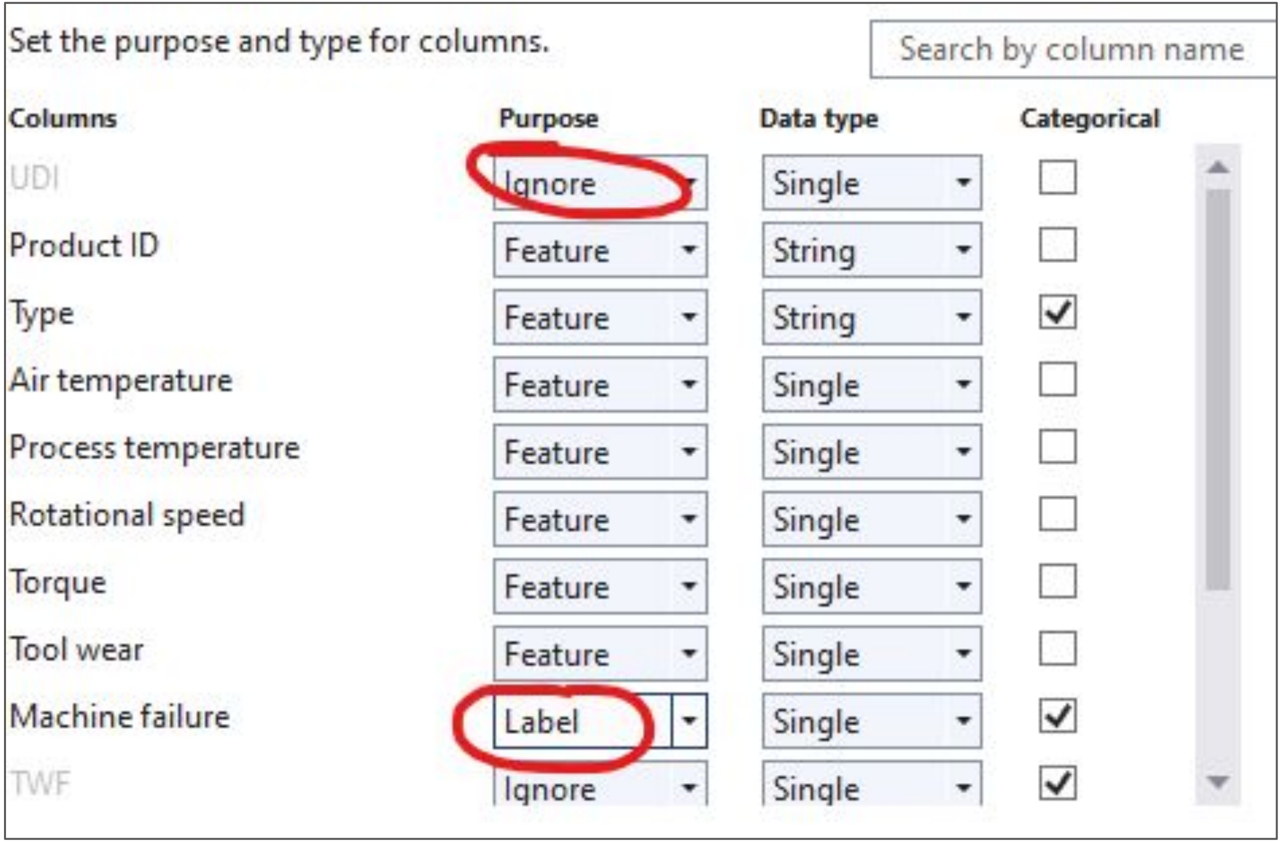
Although the failure modes are useful in diagnosing the root cause of a failure, they aren't useful for your use case.

2. Add ML Model to a new project

- ❑ PredictiveMaintenanceModel
- ❑ Remove special characters in dataOriginal header:
    - ❑ UDI,Product ID,Type,Air temperature **[K]**,Process temperature **[K]**,Rotational speed **[rpm]**,Torque **[Nm]**,Tool wear **[min]**,Machine failure,TWF,HDF,PWF,OSF,RNF

- ❑ Updated header:
    - ❑ UDI,Product ID,Type,Air temperature,Process temperature,Rotational speed,Torque,Tool wear,Machine failure,TWF,HDF,PWF,OSF,RNF

3. Column settings



Set the purpose and type for columns.

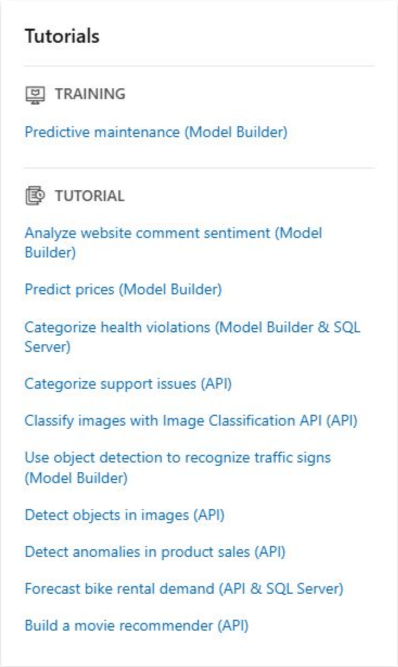
Columns	Purpose	Data type	Categorical
UDI	Ignore	Single	☐
Product ID	Feature	String	☐
Type	Feature	String	☑
Air temperature	Feature	Single	☐
Process temperature	Feature	Single	☐
Rotational speed	Feature	Single	☐
Torque	Feature	Single	☐
Tool wear	Feature	Single	☐
Machine failure	Label	Single	☑
TWF	Ignore	Single	☑

4. Train

Time: 30s

What’s next?

https://learn.microsoft.com/en-us/dotnet/machine-learning



Tutorials

TRAINING

Predictive maintenance (Model Builder)

TUTORIAL

Analyze website comment sentiment (Model Builder)

Predict prices (Model Builder)

Categorize health violations (Model Builder & SQL Server)

Categorize support issues (API)

Classify images with Image Classification API (API)

Use object detection to recognize traffic signs (Model Builder)

Detect objects in images (API)

Detect anomalies in product sales (API)

Forecast bike rental demand (API & SQL Server)

Build a movie recommender (API)