ML.NET 的一些例子

```
#r "nuget:Microsoft.ML"
```

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

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

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

```
MLContext mlContext = new MLContext();
```

```
HouseData[] houseData = {
    new HouseData() { Size = 1.1F, Price = 1.2F },
    new HouseData() { Size = 1.9F, Price = 2.3F },
    new HouseData() { Size = 2.8F, Price = 3.0F },
    new HouseData() { Size = 3.4F, Price = 3.7F } };
IDataView trainingData = mlContext.Data.LoadFromEnumerable(houseData);
```

```
var pipeline = mlContext.Transforms.Concatenate("Features", new[] { "Size"
}).Append(mlContext.Regression.Trainers.Sdca(labelColumnName: "Price",
maximumNumberOfIterations: 100));
```

```
var model = pipeline.Fit(trainingData);
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");
```

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

```


public class SentimentIssue
{
  [LoadColumn(0)]
  public string Text { get; set; }
  [LoadColumn(1)]
  public bool Label { get; set; }
}
```

```
public class SentimentPrediction
{
   [ColumnName("PredictedLabel")]
   public bool Prediction { get; set; }

   public float Probability { get; set; }

   public float Score { get; set; }
}
```

```
var mlContext = new MLContext(seed: 1);
```

```
IDataView trainingData = mlContext.Data.LoadFromTextFile<SentimentIssue>
(@"datasets/mlnet/train_data.tsv", hasHeader: true);
IDataView testData = mlContext.Data.LoadFromTextFile<SentimentIssue>
(@"datasets/mlnet/train_data.tsv", hasHeader: true);
```

```
var dataProcessPipeline = mlContext.Transforms.Text.FeaturizeText("Features",
nameof(SentimentIssue.Text));
```

```
var trainer =
mlContext.BinaryClassification.Trainers.LbfgsLogisticRegression("Label",
"Features");
var trainingPipeline = dataProcessPipeline.Append(trainer);
```

```
ITransformer trainedModel = trainingPipeline.Fit(trainingData);
```

```
var predictions = trainedModel.Transform(testData);
var metrics = mlContext.BinaryClassification.Evaluate(data: predictions,
labelColumnName: "Label", scoreColumnName: "Score");
```

```
Console.WriteLine($"*       Accuracy: {metrics.Accuracy:P2}");
Console.WriteLine($"*       Area Under Curve:
{metrics.AreaUnderRocCurve:P2}");
Console.WriteLine($"*       Area under Precision recall Curve:
{metrics.AreaUnderPrecisionRecallCurve:P2}");
Console.WriteLine($"*       F1Score:  {metrics.F1Score:P2}");
Console.WriteLine($"*       LogLoss:  {metrics.LogLoss:#.##}");
Console.WriteLine($"*       LogLossReduction:  {metrics.LogLossReduction:#.##}");
Console.WriteLine($"*       PositivePrecision:
{metrics.PositivePrecision:#.##}");
Console.WriteLine($"*       PositiveRecall:  {metrics.PositiveRecall:#.##}");
Console.WriteLine($"*       NegativePrecision:
{metrics.NegativePrecision:#.##}");
Console.WriteLine($"*       NegativeRecall:  {metrics.NegativeRecall:P2}");
```

```
mlContext.Model.Save(trainedModel, trainingData.Schema,
@"datasets/mlnet/SentimentModel.zip");
```

```
SentimentIssue sampleStatement = new SentimentIssue { Text = "This is a very good
film" };
```

```
var predEngine = mlContext.Model.CreatePredictionEngine<SentimentIssue,
SentimentPrediction>(trainedModel);
```

```
var resultprediction = predEngine.Predict(sampleStatement);
```

```
Console.WriteLine($"=============== Single Prediction  ===============");
Console.WriteLine($"Text: {sampleStatement.Text} | Prediction:
{(Convert.ToBoolean(resultprediction.Prediction) ? "Toxic" : "Non Toxic")}
sentiment | Probability of being toxic: {resultprediction.Probability} ");
```