DOI https://doi.org/10.30525/978-9934-26-506-8-106

USING ML.NET TO SOLVE A CLASSIFICATION TASK IN C#

ВИКОРИСТАННЯ ML.NET ДЛЯ ВИРІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ЗА ДОПОМОГОЮ C#

Kasianiuk O.S.,
Senior Lecturer,
LLC "Technical university "Metinvest polytechnic",
Zaporizhzhia, Ukraine

Касьянюк О.С.,
старший викладач,
ТОВ «Технічний університет «Метінвест політехніка»,
м. Запоріжжя, Україна

ML.NET [1] для C# є актуальним інструментом машинного навчання, особливо для розробників і компаній, що вже працюють у .NET екосистемі. Він забезпечує глибоку інтеграцію з .NET платформами, такими як ASP.NET та WPF, що робить впровадження моделей у реальні додатки максимально зручним і швидким. ML.NET використовує знайомий синтаксис C#, спрощуючи початок роботи для програмістів, що вже знають цю мову, та пропонує великий вибір вбудованих алгоритмів для різноманітних задач, таких як класифікація чи регресія.

Завдяки AutoML розробники можуть швидко налаштувати оптимальні параметри моделей навіть без глибоких знань у ML, а підтримка формату ONNX дозволяє використовувати моделі, створені в інших ML-екосистемах, таких як TensorFlow чи PyTorch, без необхідності переходу на інші платформи.

Проведемо дослідження ML.NET для вирішення задачі класифікації Ірисів Фішера [2]. В першу чергу потрібно підготувати класи для вхідних і вихідних даних моделі. Код для них продемонстровано на рисунку 1.

```
using Microsoft.ML.Data;

public class IrisData {
    [LoadColumn(0)]
    public float SepalLength;
    [LoadColumn(1)]
    public float SepalWidth;
    [LoadColumn(2)]
    public float PetalLength;
    [LoadColumn(3)]
    public float PetalWidth;
    [LoadColumn(4)]
    public string Label;
}
public class IrisPrediction {
    [ColumnName("PredictedLabel")]
    public string PredictedLabel;
}
```

Рис. 1. Код для збереження даних моделі

Далі вже йдуть налаштування моделі, її навчання та тестування, код для яких продемонстровано на рисунку 2. Цей код розроблявся для інтерактивного блокноту .Net Interactive.

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

// Створюємо ML контекст
var mlContext = new MLContext();

// Завантажуємо дані
string dataPath = "iris.data";
IDataView dataView = mlContext.Data.LoadFromTextFile<IrisData>(dataPath, hasHeader: false, separatorChar: ',');

// Розбиваємо дані на тренувальні та тестові набори
var splitData = mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
var trainData = splitData.TrainSet;
var testData = splitData.TestSet;

// Створюємо пайплайн для підготовки даних і навчання моделі
var pipeline = mlContext.Transforms.Conversion.MapValueToKey("Label")
    .Append(mlContext.Transforms.Concatenate("Features", "SepalLength", "SepalWidth", "PetalLength", "PetalWidth"))
.Append(mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy("Label", "Features"))
.Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel"));

// Навчаємо модель
var model = pipeline.Fit(trainData);

// Оцінюємо модель
var predictions = model.Transform(testData);
var metrics = mlContext.MulticlassClassification.Evaluate(predictions, "Label", "Score");

Console.WriteLine($"Log-loss: {metrics.LogLoss}");
Console.WriteLine($"Переконфігуровані дані точності: {metrics.MacroAccuracy}");
Console.WriteLine($"Мікро точність: {metrics.MicroAccuracy}");

// Тестуємо на новому наборі даних
var predictor = mlContext.Model.CreatePredictionEngine<IrisData, IrisPrediction>(model);
var sampleData = new IrisData()
{
    SepalLength = 5.1f,
    SepalWidth = 3.5f,
```

```
        PetalLength = 1.4f,
        PetalWidth = 0.2f
    };

    var prediction = predictor.Predict(sampleData);
    Console.WriteLine($"Прогнозований клас ірису: 
{prediction.PredictedLabel}");
```

Рис. 2. Налаштування, навчання та тестування моделі

Готову модель, яку ми навчили, можна зберегти у файл. Код для цього продемонстрований на рисунку 3.

```
    // Зберігаємо модель на диск
    string modelPath = "model.zip";
    mlContext.Model.Save(model, trainData.Schema, modelPath);
    Console.WriteLine("Модель збережено в файлі model.zip");
```

Рис. 3. Збереження моделі

Далі цю модель можна завантажити та знову використовувати для класифікації. Код для цього продемонстрований на рисунку 4.

```
    // Завантажуємо модель
    DataViewSchema modelSchema;
    ITransformer loadedModel = mlContext.Model.Load(modelPath, out 
modelSchema);

    // Створюємо PredictionEngine для використання завантаженої моделі
    var loadedPredictor = 
mlContext.Model.CreatePredictionEngine<IrisData, 
IrisPrediction>(loadedModel);

    // Виконуємо прогнозування на новому прикладі
    var newSample = new IrisData()
    {
        SepalLength = 5.1f,
        SepalWidth = 3.5f,
        PetalLength = 1.4f,
        PetalWidth = 0.2f
    };

    var newPrediction = loadedPredictor.Predict(newSample);
    Console.WriteLine($"Прогнозований клас ірису для нової моделі: 
{newPrediction.PredictedLabel}");
```

Рис. 4. Завантаження та використання моделі

Цей приклад на C# і ML.NET реалізує модель для класифікації ірисів, показуючи весь процес створення: завантаження даних, їх поділ, налаштування обробки, тренування та оцінку моделі. Код також підтримує збереження та повторне завантаження моделі, що спрощує її застосування без повторного навчання.

ML.NET є чудовим вибором для розробників C# і компаній, які вже працюють в .NET екосистемі, забезпечуючи простий спосіб інтеграції ML у наявні рішення без складних інтеграцій або змін технологічного стеку.

Перелік використаних джерел

1. ML.Net. *NuGet: веб-сайт.* URL: https://www.nuget.org/packages/microsoft.ml/ (Дата звернення 30.10.2024)

2. Iris Dataset. *UC Irvine Machine Learning Repository: веб-сайт.* URL: http://archive.ics.uci.edu/dataset/53/iris (Дата звернення 30.10.2024)

DOI https://doi.org/10.30525/978-9934-26-506-8-107

ON THE PRINCIPLES OF DEVELOPING A COMPUTER PROGRAM FOR CALCULATING THE STRENGTH OF DOUBLE-BRACED I-BEAMS

ПРО ПРИНЦИПИ РОЗРОБКИ КОМП'ЮТЕРНОЇ ПРОГРАМИ З РОЗРАХУНКУ НА МІЦНІСТЬ ДВОХОПОРНИХ ДВОТАВРОВИХ БАЛОК

Kostikov O.A.,
PhD (Physics and Mathematics), Associate Professor, LLC "Technical university "Metinvest polytechnic", Zaporizhzhia, Ukraine

Костіков О.А.,
к.ф.-м.н., доцент, ТОВ «Технічний університет «Метінвест політехніка», м. Запоріжжя, Україна

Kholodnyak Yu.S.,
PhD (Engineering), Associate Professor, Kramatorsk, Ukraine

Холодняк Ю.С.,
к.т.н., доцент, м. Краматорськ, Україна

З метою автоматизації розрахунку на міцність статично визначених двохопорних двотаврових балок було розроблено комп'ютерну програму, яка забезпечила рішення цієї задачі.