УДК 004.8

АНАЛИЗ РАБОТЫ ТЕХНОЛОГИИ ML.NET

♦♦♦♦♦

ANALYSIS OF THE WORK OF ML.NET TECHNOLOGIES

Тотухов К.Е.
кандидат технических наук, доцент кафедры «Информационные системы и программирование»,
Кубанский государственный технологический институт
101ke@mail.ru

Ковалев Н.С.
студент,
Кубанский государственный технологический институт
kovalov_1998@mail.ru

Боярко А.Э.
Студентка,
Кубанский государственный технологический институт
Merts1411@mail.ru

Аннотация. В 2018 году Microsoft разработали ML.NET – фреймворк машинного обучения для .NET разработчиков. За прошедшее время эта библиотека претерпела существенные изменения и обзавелась новыми функциями, в этой статье мы на основе примеров познакомимся с возможностями данного фреймворка.

Ключевые слова: анализ работы технологии ML.NET, SDCA, метод Fit, MLContext, DataView.

Totukhov K.E.
Candidate of Technical Sciences,
Associate Professor,
Department of Information Systems and Programming
Kuban State Technological Institute
101ke@mail.ru

Kovalev N.S.
Student,
Kuban State Technological Institute
kovalov_1998@mail.ru

Boyarko A.E.
Student,
Kuban State Technological Institute
Merts1411@mail.ru

Annotation. In 2018, Microsoft developed ML.NET, a machine learning framework for .NET. In this article, we will get acquainted with this framework based on examples.

Keywords: analysis of the work of ML.NET technologies, SDCA, Fit method, MLContext, DataView.

Под ML.NET понимают систему, которая позволяет добавлять в приложения .NET возможности машинного обучения в автономном и подключенном режимах. Используя эту технологию, вы сможете получать автоматические прогнозы на основе входных данных, заготовленных для вашего приложения.

В основе ML.NET лежит модель машинного обучения. Эта модель определяет шаги, которые необходимо пройти для получения прогнозов на основе заготовленных входных данных. С помощью ML.NET мы сможем обучить пользовательскую модель, указав соответствующий алгоритм, а также импортировать предварительно обученные модели TensorFlow и ONNX.

Основная цель ML.NET заключается в том, чтобы обеспечить простой способ построения сложных сквозных конвейеров, от этапа преобразования и дополнения необработанных данных, до обучения моделей машинного обучения и развертывания их в других системах.

Давайте на примере разберемся как работает ML.NET. Как уже было упомянуто ML.NET был разработан чтобы быть интуитивно понятным для разработчиков .NET платформы. Именно поэтому мы столкнемся с концепциями и шаблонами, которые можно найти в других фреймворках, таких как ASP.NET и Entity framework. Самыми важными классами ML.NET являются MLContext и DataView, являющиеся обязательными во всех проектах на базе ML.NET.

Класс MLContext является одноэлементным классом, и его объект предоставляет доступ к большинству функциональных возможностей ML.NET, такие как различные алгоритмы машинного обучения, которые называются тренажерами (trainers) в контексте ML.NET.

Класс DataView – это абстракция, заимствованная из систем управления реляционными базами данных. Этот класс обеспечивает композиционную обработку схема-

тизированных данных, в то же время позволяя изящно и эффективно обрабатывать многомерные данные в любых их наборах, размер которых превышает объем основной памяти. В двух словах, этот класс является причиной того, почему ML.NET настолько быстрый в вопросах обучения машинных систем.

Создание приложения с помощью ML.NET состоит из нескольких этапов.

Самым первым этапом является сбор и загрузка данных – необработанные данные должны быть загружены в память программы, для этого используется интерфейс IDataView.

Следующим этапом является создания конвейера преобразования загруженных данных и последующего обучения модели на базе этих данных. ML.NET предоставляет различные этапы преобразования, такие как одноразовое кодирование и различные алгоритмы машинного обучения.

Дальше идет этап обучения модели – это делается с помощью метода MLContext.Fit(), который поддерживается во всех алгоритмах машинного обучения внутри ML.NET.

После процесса обучения всегда следует оценить модель обучения и на основе полученных результатов внести необходимые дополнительные изменения в вышеописанных этапах.

Затем, после обучения модель сохраняется в файл, для того чтобы полученный алгоритм прогнозирования мог быт имплементирован в другой сервис.

На рисунке 1 наглядно представлен алгоритм проектирования приложения на базе ML.NET.

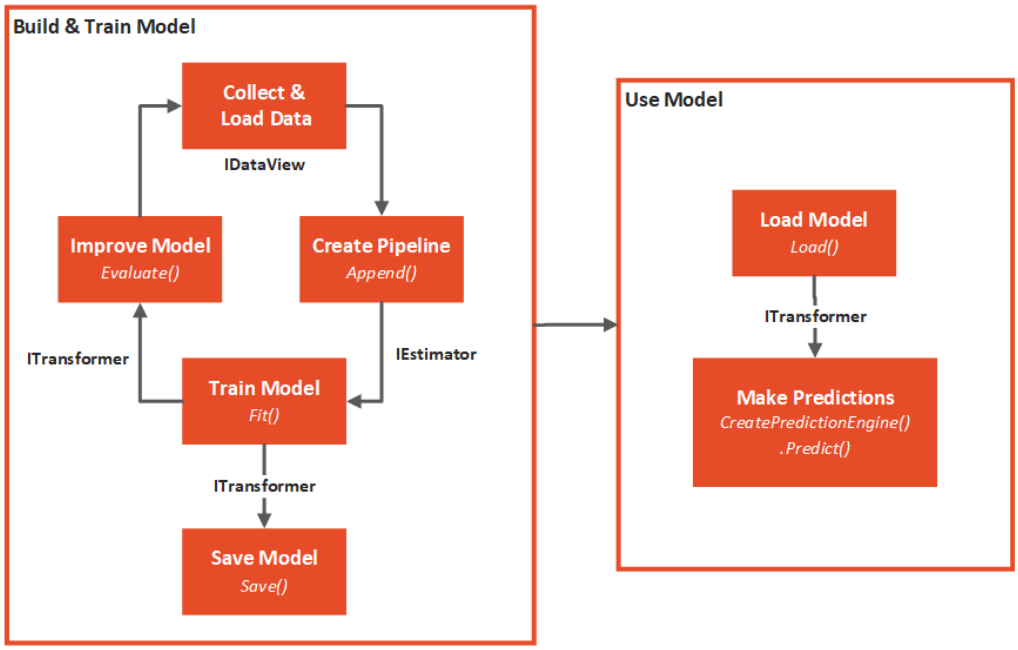

Рисунок 1 – Схема проектирования приложения ML.NET

Рассмотрим фрагмент кода, на котором показано простейшее приложения ML.NET. Код в этом примере создает модель линейной регрессии для прогнозирования цен на дома, исходя из их размера и стоимости. На рисунке 2 представлен этот пример.

В этом примере мы сначала создаем экземпляр MLContext. Затем создается массив houseData, базирующийся на классе HouseData, определенный заранее float параметрами Size и Price. Так как это пример, то данные домов мы загружаем именно таким образом, в реальных ML.NET приложениях все параметры загружаются из CSV\TSV файлов либо из файлов баз данных, таких как MSSQL или MySQL. После этого мы загружаем данные в память посредством вызова метода LoadFromEnumerable(), что на самом деле не является единственным способом загрузки данных, их можно загрузить множеством других методов в зависимости от степени интеграции программы с другими системами.

```
static void Main(string[] args)
{
    MLContext mlContext = new MLContext();

    // 1. Import or create training data
    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);

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

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

    // 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
}
```

Рисунок 2 – Пример для прогнозирования цен на дома

На следующем шаге мы создаем конвейер. Здесь мы используем метод Append() для добавления различных преобразований данных и алгоритмов машинного обучения. В этом конкретном случае мы использовали алгоритм регрессии SDCA, который является версией алгоритма Линейной регрессии.

Затем мы обучаем модель посредством метода Fit() на базе подготовленных данных и сохраняем модель для создания новых прогнозов.

После всех этих шагов полученную модель можно использовать для создания новых прогнозов.

Давайте поближе познакомимся с методом обучения модели, использованном в данном примере. Метод SDCA является вариацией алгоритма линейной регрессии, где одно непрерывное количество пропорционально другому. На рисунке 3 схематически изображен данный алгоритм.

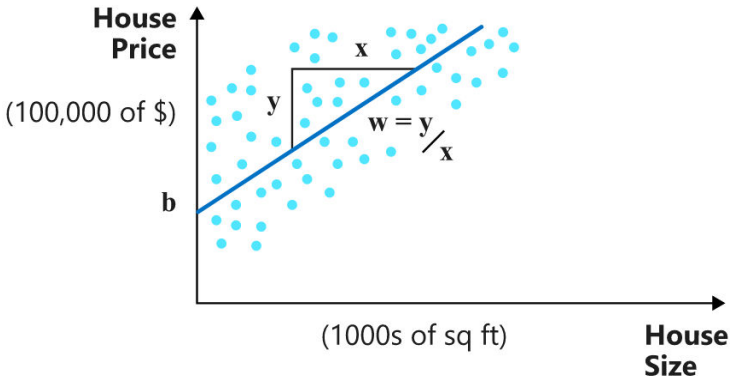

Рисунок 3 – Схематическое отображение алгоритма, описанного в примере

Модель проста: Цена = b + Размер * w. Параметры b и w рассчитываются с помощью линии, проведенной через набор пар данных (размер, цена). Данные, используемые для поиска параметров модели, называются обучающими. Входные данные модели машинного обучения называются компонентами. В этом примере Размер (Size) – единственный компонент. Эталонные значения, которые используются для обучения модели машинного обучения, называются метками. В этом примере метками служат значения параметра Цена в обучающем наборе данных.

Литература

1. Библиотека ML.NET [Электронный ресурс]. – URL : https://ru.wikipedia.org/wiki/ML.NET
2. Building Machine-Learning Models with ML.NET [Электронный ресурс]. – URL : https://www.atmosera.com/blog/building-machine-learning-models-with-ml-net/
3. Документация Microsoft. Руководство по ML.NET [Электронный ресурс]. – URL : https://learn.microsoft.com/ru-ru/dotnet/machine-learning/automate-training-with-model-builder

References

1. ML.NET library [Electronic resource]. – URL : https://ru.wikipedia.org/wiki/ML.NET
2. Building Machine-Learning Models with ML.NET [Electronic resource]. – URL : https://www.atmosera.com/blog/building-machine-learning-models-with-ml-net/
3. Microsoft documentation. ML.NET manual [Electronic resource]. – URL : https://learn.microsoft.com/ru-ru/dotnet/machine-learning/automate-training-with-model-builder