Hands-On Machine Learning with ML.NET

Getting started with Microsoft ML.NET to implement popular machine learning algorithms in C#

Jarred Capellman

Packt
www.packt.com

Hands-On Machine Learning with ML.NET

Getting started with Microsoft ML.NET to implement popular machine learning algorithms in C#

Jarred Capellman

Packt

BIRMINGHAM - MUMBAI

Hands-On Machine Learning with ML.NET

Commissioning Editor: Pravin Dhandre
Acquisition Editor: Devika Battike
Content Development Editor: Joseph Sunil
Senior Editor: David Sugarman
Technical Editor: Utkarsha Kadam
Copy Editor: Safis Editing
Project Coordinator: Aishwarya Mohan
Proofreader: Safis Editing
Indexer: Manju Arasan
Production Designer: Aparna Bhagat

First published: March 2020

Production reference: 1260320

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78980-178-1

www.packt.com

To my amazing wife, Amy, for completing me.

– Jarred Capellman

Packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Jarred Capellman is a Director of Engineering at SparkCognition, a cutting-edge artificial intelligence company located in Austin, Texas. At SparkCognition, he leads the engineering and data science team on the industry-leading machine learning endpoint protection product, DeepArmor, combining his passion for software engineering, cybersecurity, and data science. In his free time, he enjoys contributing to GitHub daily on his various projects and is working on his DSc in cybersecurity, focusing on applying machine learning to solving network threats. He currently lives just outside of Austin, Texas, with his wife, Amy.

To my wife, Amy, who supported me through the nights and weekends – I devote this book to her.

About the reviewer

Andrew Greenwald holds an MSc in computer science from Drexel University and a BSc in electrical engineering with a minor in mathematics from Villanova University. He started his career designing solid-state circuits to test electronic components. For the past 25 years, he has been developing software for IT infrastructure, financial markets, and defense applications. He is currently applying machine learning to cybersecurity, developing models to detect zero-day malware. Andrew lives in Austin, Texas, with his wife and three sons.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface

Machine learning (**ML**) is widely used in many industries, such as science, healthcare, and research and its popularity is only growing. In March 2018, Microsoft introduced ML.NET to help .NET enthusiasts to work with ML. With this book, you'll explore how to build ML.NET applications with the various ML models available using C# code.

The book starts by giving you an overview of ML and the types of ML algorithms used, along with covering what ML.NET is and why you need it to build ML apps. You'll then explore the ML.NET framework, its components, and APIs. The book will serve as a practical guide to helping you build smart apps using the ML.NET library. You'll gradually become well-versed in how to implement ML algorithms such as regression, classification, and clustering with real-world examples and datasets. Each chapter will cover the practical implementation, showing you how to implement ML within .NET applications. You'll also learn how to integrate TensorFlow into ML.NET applications. Later, you'll discover how to store the regression model housing price prediction results in the database and display the real-time predicted results from the database on your web application using ASP.NET Core Blazor and SignalR.

By the end of this book, you'll have learned how to confidently perform basic to advanced-level machine learning tasks in ML.NET.

Who this book is for

If you are a .NET developer who wants to implement machine learning models using ML.NET, then this book is for you. This book will also be beneficial to data scientists and machine learning developers who are looking for effective tools to implement various machine learning algorithms. A basic understanding of C# and .NET is mandatory to grasp the concepts covered in this book effectively.

What this book covers

`Chapter 1`, *Getting Started with Machine Learning and ML.NET*, talks about what machine learning is and how important machine learning is in our society today. It also introduces ML.NET and talks in more detail about getting started with it after learning about the concepts of machine learning and how they relate.

`Chapter 2`, *Setting Up the ML.NET Environment*, talks in more detail about getting started with ML.NET, continuing the overview of machine learning and how ML.NET can assist in both developing and running models in both new and existing applications. You will ensure your development environment is set up and the chapter ends with a simple pre-trained model in a console application to demonstrate that you are ready to proceed with the training.

`Chapter 3`, *Regression Model*, talks about using a regression and logistic regression model in ML.NET in addition to the math and what problems these models can help to solve. In addition, the chapter provides a step-by-step explanation of how to create and work with both a regression model and a logistic regression model in ML.NET. The end of the chapter details a quick console application using the dataset and both the models in ML.NET.

`Chapter 4`, *Classification Model*, talks about using the classifications trainer models in ML.NET and what problems a classification model can help to solve. For this chapter, we will create two applications to demonstrate the classification trainer support in ML.NET. The first predicts whether a car is of good value based on the several attributes and comparative prices using the FastTree trainer that ML.NET provides. The second application takes email data (Subject, Body, Sender) with the SDCA trainer in ML.NET to classify the email as an Order, Spam or Friend. Through these applications, you will also learn how to evaluate classification models.

`Chapter 5`, *Clustering Model*, talks about using the k-means clustering trainer in ML.NET in addition to what problems a clustering model can help to solve. In this chapter, we will use the k-means cluster trainer that ML.NET provides in order to create an example application that will classify files as either executables, documents, or scripts. In addition, you will learn how to evaluate clustering models in ML.NET.

`Chapter 6`, *Anomaly Detection Model*, talks about using an anomaly detection model in ML.NET in addition to what problems an anomaly detection model can help to solve. For this chapter, we will create two example applications. The first uses ML.NET with SSA to detect Network Traffic anomalies, while the second example uses ML.NET with PCA to detect anomalies in a series of user logins. With these applications, we will also look at how you can evaluate your anomaly detection model once trained.

`Chapter 7`, *Matrix Factorization Model*, talks about using a matrix factorization model in ML.NET in addition to the math and what problems a matrix factorization model can help to solve. In this chapter, we will create a music recommendation application using the matrix factorization trainer that ML.NET provides. Using several data points this recommendation engine will recommend music based on the training data provided to the model. In addition, after creating this application we will learn how to evaluate a matrix factorization model in ML.NET.

`Chapter 8`, *Using ML.NET with .NET Core and Forecasting*, covers a real-world application utilizing .NET Core and utilizes both a regression and time series model to demonstrate forecasting on stock shares.

`Chapter 9`, *Using ML.NET with ASP.NET Core*, covers a real-world application utilizing ASP.NET with a frontend to upload a file to determine whether it is malicious or not. This chapter focuses on using a binary classifier and how to integrate it into an ASP.NET application.

`Chapter 10`, *Using ML.NET with UWP*, covers a real-world application utilizing UWP and ML.NET. The application will utilize ML.NET to classify whether the web page content is malicious. The chapter will also cover UWP application design and MVVM briefly to give a true production-ready sample app to build on or adapt to other applications for using UWP with ML.NET.

`Chapter 11`, *Training and Building Production Models*, covers training a model at scale with all of the considerations, along with the proper training of a production model using the DMTP project. The lessons learned include obtaining proper training sets (diversity being key), proper features, and the true evaluation of your model. The focus of this chapter is on tips, tricks, and best practices for training production-ready models.

`Chapter 12`, *Using TensorFlow with ML.NET*, talks about using a pre-trained TensorFlow model with ML.NET to determine whether a car is in a picture or not with a UWP application.

`Chapter 13`, *Using ONNX with ML.NET*, talks about using a pre-trained ONNX model with ML.NET in addition to the value added by taking a pre-existing ONNX format model into ML.NET directly.

To get the most out of this book

You will need a version of Angular installed on your computer—the latest version, if possible. All code examples have been tested using Angular 9 on Windows OS. However, they should work with future version releases too.

Software/Hardware covered in the book	OS Requirements
Microsoft Visual Studio 2019	A common Windows 10 development environment with 20-50 GB of free space (a quad core processor and 8 GB of RAM is highly recommended)

If you are using the digital version of this book, we advise you to type the code yourself or access the code via the GitHub repository (link available in the next section). Doing so will help you avoid any potential errors related to the copy/pasting of code.

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packtpub.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **Support** tab.
3. Click on **Code Downloads**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Hands-On-Machine-Learning-with-ML.NET`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `http://www.packtpub.com/sites/default/files/downloads/9781789801781_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "The first time the application is run, the ML.NET version of the model is trained with the images and `tags.tsv` file (to be reviewed in the next section)."

A block of code is set as follows:

```
public void Classify(string imagePath)
{
    var result = _prediction.Predict(imagePath);

    ImageClassification = $"Image ({imagePath}) is a picture of
{result.PredictedLabelValue} with a confidence of
{result.Score.Max().ToString("P2")}";
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
dotnet --version
3.0.100
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Firstly, ensure that **.NET desktop development**, **Universal Windows Platform Development**, and **ASP.NET and web development** are checked."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/support/errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Section 1: Fundamentals of Machine Learning and ML.NET

This section gives an overview of this book's audience and a short introduction to machine learning and the importance of learning how to utilize machine learning. In addition, this section introduces the reader to ML.NET. It also talks about the tools and framework needed to build the applications and gives a step-by-step explanation of how to work with ML.NET.

This section comprises the following chapters:

- `Chapter 1`, *Getting Started with Machine Learning and ML.NET*
- `Chapter 2`, *Setting Up the ML.NET Environment*

1
Getting Started with Machine Learning and ML.NET

By opening this book, you are taking the first step in disrupting your own knowledge by approaching solutions to complex problems with machine learning. You will be achieving this with the use of Microsoft's ML.NET framework. Having spent several years applying machine learning to cybersecurity, I'm confident that the knowledge you garner from this book will not only open career opportunities to you but also open up your thought processes and change the way you approach problems. No longer will you even approach a complex problem without thinking about how machine learning could possibly solve it.

Over the course of this book, you will learn about the following:

- How and when to use five different algorithms that ML.NET provides
- Real-world end-to-end examples demonstrating ML.NET algorithms
- Best practices when training your models, building your training sets, and feature engineering
- Using pre-trained models in both TensorFlow and ONNX formats

This book does assume that you have a reasonably solid understanding of C#. If you have other experience with a strongly typed object-oriented programming language such as C++ or Java, the syntax and design patterns are similar enough to not hinder your ability to follow the book. However, if this is your first deep dive into a strongly typed language such as C#, I strongly suggest picking up *Learn C# in 7 Days*, by Gaurav Aroraa, published by Packt Publishing, to get a quick foundation. In addition, no prior machine learning experience is required or expected, although a cursory understanding will accelerate your learning.

In this chapter, we will cover the following:

- The importance of learning about machine learning today
- The model-building process
- Exploring types of learning
- Exploring various machine learning algorithms
- Introduction to ML.NET

By the end of the chapter, you should have a fundamental understanding of what it takes to build a model from start to finish, providing the basis for the remainder of the book.

The importance of learning about machine learning today

In recent years, machine learning and artificial intelligence have become an integral part of many of our lives in use cases as diverse as finding cancer cells in an MRI and facial and object recognition during a professional basketball game. Over the course of just the four years between 2013 and 2017, machine learning patents alone grew 34%, while spending is estimated to grow to $57.6B by 2021 (`https://www.forbes.com/sites/louiscolumbus/2018/02/18/roundup-of-machine-learning-forecasts-and-market-estimates-2018/#794d6f6c2225`).

Despite its status as a growing technology, the term machine learning was coined back in 1959 by Arthur Samuel—so what caused the 60-year gap before its adoption? Perhaps the two most significant factors were the availability of technology able to process model predictions fast enough, and the amount of data being captured every minute digitally. According to DOMO Inc, a study in 2017 concluded that 2.5 quintillion bytes were generated daily and that at that time, 90% of the world's data was created between 2015 and 2017 (`https://www.domo.com/learn/data-never-sleeps-5?aid=ogsm072517_1sf100871281=1`). By 2025, it is estimated that 463 exabytes of data are going to be created daily (`https://www.visualcapitalist.com/how-much-data-is-generated-each-day/`), much of which will come from cars, videos, pictures, IoT devices, emails, and even devices that have not made the transition to the smart movement yet.

The amount that data has grown in the last decade has led to questions about how a business or corporation can use such data for better sales forecasting, anticipating a customer's needs, or detecting malicious bytes in a file. Traditional statistical approaches could potentially require exponentially more staff to keep up with current demands, let alone scale with the data captured. Take, for instance, Google Maps. With Google's acquisition of Waze in 2013, users of Google Maps have been provided with extremely accurate routing suggestions based on the anonymized GPS data of its users. With this model, the more data points (in this case GPS data from smartphones), the better predictions Google can make for your travel. As we will discuss later in this chapter, quality datasets are a critical component of machine learning, especially in the case of Google Maps, where, without a proper dataset, the user experience would be subpar.

In addition, the speed of computer hardware, specifically specialized hardware tailored for machine learning, has also played a role. The use of **Application-Specific Integrated Circuits** (**ASICs**) has grown exponentially. One of the most popular ASICs on the market is the Google **Tensor Processing Unit** (**TPU**). Originally released in 2016, it has since gone through two iterations and provides cloud-based acceleration for machine learning tasks on Google Cloud Platform. Other cloud platforms, such as Amazon's AWS and Microsoft's Azure, also provide FPGAs.

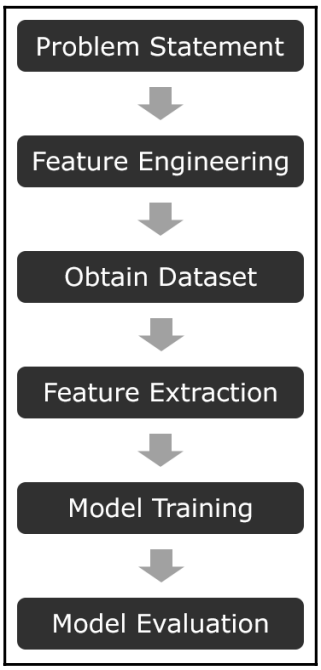
Additionally, **Graphics Processing Units** (**GPUs**) from both AMD and NVIDIA are accelerating both cloud-based and local workloads, with ROCm Platform and CUDA-accelerated libraries respectively. In addition to accelerated workloads, typical professional GPUs offered by AMD and NVIDIA provide a much higher density of processors than the traditional CPU-only approach. For instance, the AMD Radeon Instinct MI60 provides 4,096 stream processors. While not a full-fledged x86 core, it is not a one-to-one comparison, and the peak performance of double-precision floating-point tasks is rated at 7.373 TFLOPs compared to the 2.3 TFLOPs in AMD's extremely powerful EPYC 7742 server CPU. From a cost and scalability perspective, utilizing GPUs in even a workstation configuration would provide an exponential reduction in training time if the algorithms were accelerated to take advantage of the more specialized cores offered by AMD and NVIDIA. Fortunately, ML.NET provides GPU acceleration with little additional effort.

From a software engineering career perspective, with this growth and demand far outpacing the supply, there has never been a better time to develop machine learning skills as a software engineer. Furthermore, software engineers also possess skills that traditional data scientists do not have – for instance, being able to automate tasks such as the model building process rather than relying on manual scripts. Another example of where a software engineer can provide more value is by adding both unit tests and efficacy tests as part of the full pipeline when training a model. In a large production application, having these automated tests is critical to avoid production issues.

Finally, in 2018, for the first time ever, data was considered more valuable than oil. As industries continue to adopt the use of data gathering and existing industries take advantage of the data they have, machine learning will be intertwined with the data. Machine learning to data is what refining plants are to oil.

The model building process

Before diving into ML.NET, an understanding of core machine learning concepts is required. These concepts will help create a foundation for you to build on as we start building models and learning the various algorithms ML.NET provides over the course of this book. At a high level, producing a model is a complex process; however, it can be broken down into six main steps:



Over the next few sections, we will go through each of these steps in detail to provide you with a clear understanding of how to perform each step and how each step relates to the overall machine learning process as a whole.

Defining your problem statement

Effectively, what problem are you attempting to solve? Being specific at this point is crucial as a less concise problem can lead to considerable re-work. For example, take the following problem statement: *Predicting the outcome of an election*. My first question upon hearing that problem statement would be, at what level? County, state, or national? Each level more than likely requires considerably more features and data to properly predict than the last. A better problem statement, especially early on in your machine learning journey, would be for a specific position at a county level, such as *Predicting the 2020 John Doe County Mayor*. With this more direct problem statement, your features and dataset are much more focused and more than likely attainable. Even with more experience in machine learning, proper scoping of your problem statement is critical. The five Ws of Who, What, When, Where, and Why should be followed to keep your statement concise.

Defining your features

The second step in machine learning is defining your features. Think of features as components or attributes of the problem you wish to solve. In machine learning – specifically, when creating a new model – features are one of the biggest impacts on your model's performance. Properly thinking through your problem statement will promote an initial set of features that will drive differentiation between your dataset and model results. Going back to the Mayor example in the preceding section, what features would you consider data points for the citizen? Perhaps start by looking at the Mayor's competition and where he/she sits on issues in ways that differ from other candidates. These values could be turned into features and then made into a poll for citizens of John Doe County to answer. Using these data points would create a solid first pass at features. One aspect here that is also found in model building is running several iterations of feature engineering and model training, especially as your dataset grows. After model evaluation, *feature importance* is used to determine what features are actually driving your predictions. Occasionally, you will find that gut-instinct features can actually be inconsequential after a few iterations of model training and feature engineering.

In `Chapter 11`, *Training and Building Production Models*, we will deep dive into best practices when defining features and common approaches to complex problems to obtain a solid first pass at feature engineering.

Obtaining a dataset

As you can imagine, one of the most important aspects of the model building process is obtaining a high-quality dataset. A dataset is used to train the model on what the output should be in the case of the aforementioned case of supervised learning. In the case of unsupervised learning, labeling is required for the dataset. A common misconception when creating a dataset is that bigger is better. This is far from the truth in a lot of cases. Continuing the preceding example, what if all of the poll results answered the same way for every single question? At that point, your dataset is composed of all the same data points and your model will not be able to properly predict any of the other candidates. This outcome is called *overfitting*. A diverse but representative dataset is required for machine learning algorithms to properly build a production-ready model.

In `Chapter 11`, *Training and Building Production Models*, we will deep dive into the methodology of obtaining quality datasets, looking at helpful resources, ways to manage your datasets, and transforming data, commonly referred to as data wrangling.

Feature extraction and pipeline

Once your features and datasets have been obtained, the next step is to perform feature extraction. Feature extraction, depending on the size of your dataset and your features, could be one of the most time-consuming elements of the model building process.

For example, let's say that the results from the aforementioned fictitious John Doe County Election Poll had 40,000 responses. Each response was stored in a SQL database captured from a web form. Performing a SQL query, let's say you then returned all of the data into a CSV file, using which your model can be trained. At a high level, this is your feature extraction and pipeline. For more complex scenarios, such as predicting malicious web content or image classification, the extraction will include binary extraction of specific bytes in files. Properly storing this data to avoid having to re-run the extraction is crucial to iterating quickly (assuming the features did not change).

In Chapter 11, *Training and Building Production Models,* we will deep dive into ways to version your feature-extracted data and maintain control over your data, especially as your dataset grows in size.