

ML.NET





Wstęp

Dzisiaj chciałbym opowiedzieć o uczeniu maszynowym oraz o technologii ML.NET.
Ta część referatu będzie się skupiać to teorii ML.NET i uczenia maszynowego



Agenda

Po co uczenie maszynowe? (też w porównaniu do tradycyjnego programowania)

Czym jest ML.NET?

Jak działa ML.NET

Pipeline

- Dane
- Ładowanie danych
- Trenowanie modelu

Rodzaje problemów w ML.NET

Model Builder

Ograniczenia ML.NET



Dlaczego potrzebujemy uczynienia maszynowego?

Tradycyjne programowanie opiera się na sztywnych instrukcjach typu „jeśli X, to Y”. Jednak w rzeczywistości często natrafiamy na problemy, gdzie liczba potrzebnych reguł jest zbyt duża, by je ręcznie opisać, lub sam koncept jest zbyt abstrakcyjny dla standardowej logiki.



Przykłady gdzie tradycyjne programowanie nasz zawodzi.

Przykładami zbyt dużej ilości reguł to filtr spamu. Ręcznie pisanie tysięcy słów, ich kombinacji oraz wyjątków jest możliwe, lecz zajęłoby to dużo za dużo czasu.

Natomiast przykładem abstrakcyjnych konceptów jest na przykład wykrywanie czy zdjęcie ma charakter wesoły lub smutny, jest to coś co ciężko sobie wyobrazić jak opisać to przy pomocy reguł if-else.



Dlaczego potrzebujemy uczynienia maszynowego?

W uczeniu maszynowym natomiast zamiast gotowych instrukcji dostarczamy przykłady wraz z wynikami z czego maszyna sama znajduje korelacje.

Na Przykład dla filtru spamu na podstawie tysięcy przykładów maszyna zauważy że większość spamu używa podobnego języka, kolorów tekstu etc

A dla oceny wesołego lub smutnego charakteru obrazka może zauważyć np że niektóre używane kolory są bardziej prawdopodobne wystąpić w wesołych niż smutnych , jak również ułożenie pikseli względem siebie.



Gdzie spotykamy ML na co dzień?

Uczenie maszynowe jest obecne w wielu aspektach naszego życia. Serwisy streamingowe polecają filmy, skrzynki mailowe filtrują spam, a banki wykrywają podejrzaną transakcje.

Wszystkie te systemy działają dzięki analizie dużych ilości danych i podejmowaniu decyzji na ich podstawie.

Przykłady

- System rekomendacji - Youtube, TikTok, instagram, itd
- Filtry NSFW - Youtube, TikTok, instagram, itd
- Weryfikacja wieku przy pomocy zdjęcia twarzy
(Na Przykład jak jest wymagane w wielkiej brytani) - Discord lub w grach np (Roblox)



Czym Jest ML.NET

ML.NET to wieloplatformowa platforma uczenia maszynowego typu open source dla programistów .NET, która umożliwia integrację niestandardowych modeli uczenia maszynowego z aplikacjami .NET. Obejmuje ona interfejs API, który składa się z różnych pakietów NuGet, rozszerzenie programu Visual Studio o nazwie Model Builder oraz interfejs wiersza poleceń instalowany jako narzędzie .NET.

Open source - oprogramowanie z publicznie dostępnym kodem źródłowym, który każdy ma prawo przeglądać, modyfikować oraz udostępniać na tych samych zasadach



Dlaczego ML.NET

Kluczową zaletą ML.NET jest eliminacja konieczności nauki języka Python, który dominuje w większości innych platform uczenia maszynowego. Jest to idealne rozwiązanie w sytuacjach, gdy reszta projektu jest już napisana w ekosystemie .NET. Dzięki temu unikamy tworzenia skomplikowanych mostów technologicznych między różnymi językami.

Inną zaletą jest też Model builder który pozwala nam stworzyć model z dostępnych scenariuszy bez potrzeby pisania żadnej linii kodu.



Jak działa ML.NET

Proces działania ML.NET rozpoczyna się od zebrania danych. Następnie dane są przetwarzane i wykorzystywane do trenowania modelu. Po zakończeniu treningu model może wykonywać predykcje na nowych danych.

Cały ten proces jest uporządkowany w tzw. pipeline.



Pipeline : Dane

Najpierw potrzebujemy dane, ML.NET pozwala nam na
urzycie wielu różnych formatów:

Tekstowych: CSV, TSV

Bazy danych: SQL

Obrazy: JPEG, PNG

CSV - Comma-Separated Values

TSV - Tab-Separated Values



Pipeline : Dane (przykład CSV)

Przykład CSV, z dataset iris data.

kolumny od lewej do prawej to:

sepal length = 5.1

sepal width = 3.5

petal length = 1.4

petal width = 0.2

class = Iris-setosa

5.1,3.5,1.4,0.2,Iris-setosa

4.9,3.0,1.4,0.2,Iris-setosa

4.7,3.2,1.3,0.2,Iris-setosa

4.6,3.1,1.5,0.2,Iris-setosa

5.5,2.4,3.8,1.1,Iris-versicolor

5.5,2.4,3.7,1.0,Iris-versicolor

5.8,2.7,3.9,1.2,Iris-versicolor

6.0,2.7,5.1,1.6,Iris-versicolor

5.4,3.0,4.5,1.5,Iris-versicolor

6.0,3.4,4.5,1.6,Iris-versicolor

6.5,2.7,5.1,1.9,Iris-virginica

7.1,3.0,5.9,2.1,Iris-virginica

6.3,2.9,5.6,1.8,Iris-virginica



Pipeline: Ładowanie danych

Dane często wymagają oczyszczenia i przekształcenia. Często się zdarza że dane mogą zawierać błędy, brakujące wartości lub informacje zapisane w nieodpowiedniej formie.

Gdy danych brakuje trzeba albo je uzupełnić np średnią wartością z danej kolumny lub usunąć cały wiersz lub kolumnę (jeżeli duża ilość wierszy ma brakujące dane w danej kolumnie).



Pipeline: Ładowanie danych (transformowanie)

Dane muszą być też zamienione na zrozumiałej dla modelu, np tekst musi być zamieniony na listy, a zdjęcia np a macierz wartości liczbowych reprezentujących piksele.

Iris-setosa → 1

Iris-versicolor → 2

"Iris-virginica" → 3



Pipeline: Trenowanie modelu

Jak już mamy czyste dane w odpowiednim formacie, możemy zacząć trenować nasz model.

Najpierw dobrze jest podzielić dane na część testową i do uczenia, może to być np 80% do uczenia 20% do testu. Jest to potrzebne by sprawdzić czy model zgadza się jeżeli użyjemy danych których model jeszcze nie widział. Gdybyśmy użyli danych których model już widział podczas uczenia to tak jakby zobaczył już odpowiedzi do sprawdzianu , odpowie dobrze ale niekoniecznie będzie działać poprawnie.

Po wywołaniu metody `.Fit()`, otrzymujemy gotowy model. Następnie oceniamy jego jakość za pomocą dedykowanych metryk (`Evaluate()`) i – jeśli wyniki są satysfakcjonujące – zapisujemy go do pliku `.zip`



Rodzaje problemów w ML.NET

ML.NET obsługuje różne typy problemów uczenia maszynowego. Do najczęściej spotykanych należą klasyfikacja i regresja.

Każdy z tych problemów rozwiązuje inny typ zadania.



Klasyfikacja

Klasyfikacja polega na przypisaniu danych do określonej kategorii. Przykładem może być rozpoznawanie spamu lub analiza sentymentu opinii.

Model decyduje, do której klasy należy dany przykład.

Rodzaje klasyfikacji:

- Klasyfikacja binarna (Binary Classification): Wybór między dwiema opcjami (Tak/Nie, Spam/Odebrane, Zdrowe/Chore).
- Klasyfikacja wieloklasowa (Multiclass Classification): Przypisanie do jednej z wielu kategorii (np. rozpoznawanie gatunków roślin, sortowanie zgłoszeń do działów: IT, HR, Finanse).



Regresja

Regresja służy do przewidywania wartości liczbowych, takich jak cena, sprzedaż lub czas.

Model na podstawie danych historycznych przewiduje konkretną wartość dla nowego przypadku.

Model podczas uczenia szuka funkcji matematycznej która jak najbliżej pasuje do danych na których się uczył.

Wynikiem jest zawsze konkretna liczba np cena za serwis 1100 zł



Model Builder

Model Builder to narzędzie z graficznym interfejsem dostępne w Visual Studio, które umożliwia tworzenie modeli ML bez konieczności pisania dużej ilości kodu.

Użytkownik wybiera dane oraz typ problemu, a narzędzie automatycznie trenuje model.

Build your machine learning model

1. Scenario

2. Data
3. Train
4. Evaluate
5. Code

Pick a Scenario

Select a template that best matches your scenario.
[Which Machine Learning scenario is right for me?](#)



Classify data into 2 categories (binary classification), e.g. predict positive or negative sentiment of comments.



Classify data into 3+ categories (multi-class classification), e.g. predict labels of GitHub issues.



Predict a numeric value from your data (regression), e.g. predict the price of a house.



Classify images into 2+ categories (image-classification) e.g. predict type of flowers.



Build custom models with your data using classification, regression and other tasks.

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'ConsoleApp7' (1 of 1 project)

- ConsoleApp7
 - Dependencies
 - Program.cs



- ✓ 1. Scenario
- ✓ 2. Data
- ✓ 3. Train
- ✓ 4. Evaluate
- 5. Code

Evaluate

Results of training for your model can be found below.
[How do I understand my model performance?](#)

Output

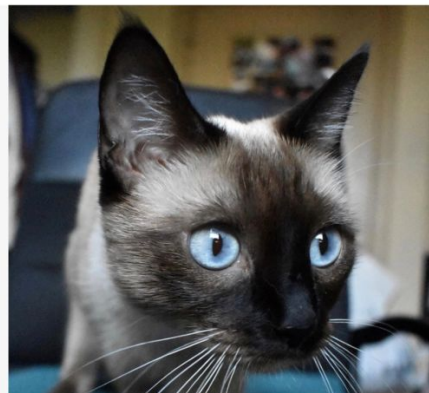
[Overview](#) [Details](#)

ML Task: image-classification
Training Time: 323.1 seconds
Models Explored (Total): 1 | [View Top 1 model explored](#)

Overall accuracy:

50%

Try your model



Results:

Cat	51%
Dog	49%

Predict

[Test another image](#)

Next Step: [Code](#)



Przykład gdzie przewiduje koszt
przejazdu taksówką

ML.NET Model Builder

✓ 1. Scenario

✓ 2. Data

✓ 3. Train

✓ 4. Evaluate

5. Code

Evaluate

Results of training for your model can be found below.
[How do I understand my model performance?](#)

Output

OverviewDetails

ML Task:	regression
Training Time:	9.6 seconds
Models Explored (Total):	8 View Top 5 models explored

Try your model

Note: Fields below are pre-filled by row 1 of your data.

passenger_count

2

trip_time_in_secs

2500

trip_distance

5.8

payment_type

CRD

Predict

Next Step: [Code](#)

Overall accuracy:

0.8751

Result

fare_amount: 28.62



Ograniczenia ML.NET

ML.NET ma mniejsze możliwości w zakresie zaawansowanego deep learningu w porównaniu z rozwiązaniami pythonowymi.

Mimo to bardzo dobrze sprawdza się w wielu rzeczywistych projektach.

Dziękuję za uwagę