Klasyfikacja wiadomości tekstowych z wykorzystaniem ML.NET w środowisku .NET

Michał Kopczyński

Styczeń 2026

Spis treści

1 Wstęp

Uczenie maszynowe (ang. *Machine Learning*) coraz częściej stanowi integralną część nowoczesnych aplikacji biznesowych. Klasyfikacja tekstu, w szczególności rozpoznawanie wiadomości typu spam, jest jednym z najbardziej klasycznych i praktycznych problemów w tej dziedzinie.

W ekosystemie platformy .NET dostępne są różne podejścia do rozwiązywania problemów opartych o sztuczną inteligencję. Z jednej strony istnieją gotowe, hostowane rozwiązania chmurowe, takie jak **Azure AI Services**, które oferują pretrenowane modele dostępne poprzez API. Z drugiej strony, biblioteka **ML.NET** umożliwia trenowanie i wykorzystywanie własnych modeli uczenia maszynowego bez opuszczania środowiska .NET oraz bez konieczności używania zewnętrznych języków programowania, takich jak Python.

Celem niniejszego projektu jest zaprojektowanie oraz implementacja prostego systemu klasyfikacji wiadomości tekstowych (e-mail oraz SMS) z wykorzystaniem biblioteki ML.NET, z zachowaniem dobrej architektury aplikacji, testów jednostkowych oraz separacji odpowiedzialności.

2 ML.NET – krótkie wprowadzenie

ML.NET jest biblioteką open-source rozwijaną przez firmę Microsoft, przeznaczoną do implementacji algorytmów uczenia maszynowego w aplikacjach .NET. Umożliwia ona między innymi:

- trenowanie modeli klasyfikacji, regresji i klasteryzacji,
- przetwarzanie danych tekstowych (NLP - Natural Language Processing),
- ewaluację jakości modeli,
- wykorzystanie wytrenowanych modeli w aplikacjach desktopowych, webowych i innych

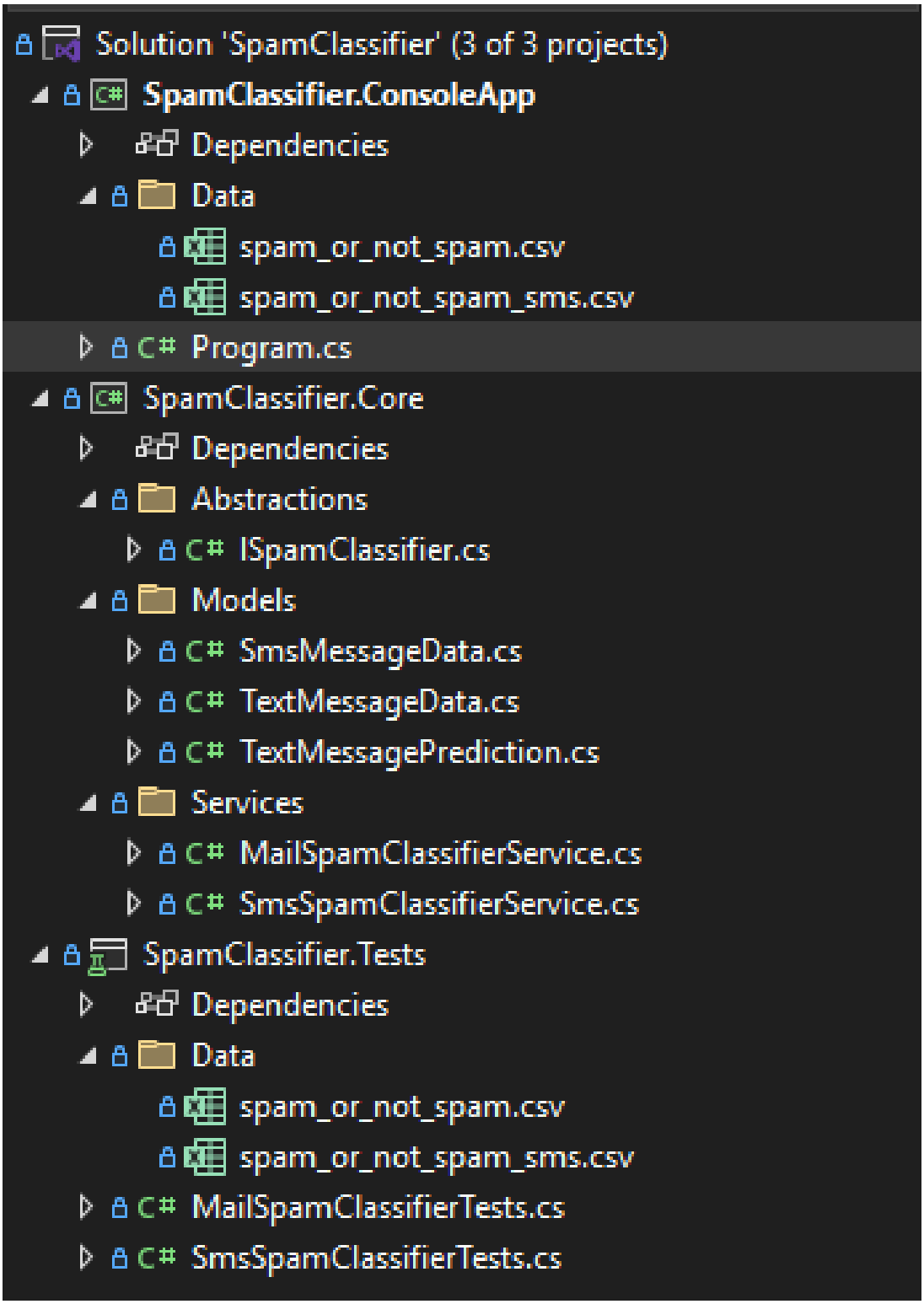
Ważną cechą ML.NET jest możliwość budowania tzw. *pipeline'ów*, czyli sekwencji transformacji danych oraz algorytmów uczących, które wspólnie tworzą kompletny model predykcyjny.

3 Architektura rozwiązania

Projekt został zaprojektowany jako **Solution** składająca się z trzech niezależnych projektów:

- **SpamClassifier.Core** – biblioteka zawierająca całą logikę biznesową oraz modele ML,
- **SpamClassifier.ConsoleApp** – aplikacja konsolowa pełniąca rolę prezentującą (runner),
- **SpamClassifier.Tests** – projekt z testami jednostkowymi.

Takie podejście umożliwia ponowne wykorzystanie logiki klasyfikacji w innych typach aplikacji, np. ASP.NET MVC, Blazor czy WPF.



Rysunek 1: Struktura Solution w Visual Studio

4 Uruchomienie projektu

Projekt został przygotowany w taki sposób, aby możliwe było jego pełne uruchomienie oraz odtworzenie wyników na komputerze z zainstalowanym środowiskiem .NET. Poniżej przedstawiono kolejne kroki niezbędne do uruchomienia aplikacji oraz testów jednostkowych.

4.1 Wymagania

Do poprawnego uruchomienia projektu wymagane są:

- .NET SDK w wersji 8.0 lub nowszej,
- system kontroli wersji Git.

4.2 Pobranie repozytorium

Kod źródłowy projektu został udostępniony w publicznym repozytorium GitHub. Repozytorium można pobrać za pomocą następujących poleceń:

```
git clone https://github.com/mike-314/SpamClassifier.git
cd SpamClassifier
```

4.3 Przywrócenie zależności

Po pobraniu repozytorium należy przywrócić wszystkie zależności NuGet:

```
dotnet restore
```

4.4 Uruchomienie aplikacji

Aplikację konsolową można uruchomić poleceniem:

```
dotnet run --project SpamClassifier.ConsoleApp
```

Po uruchomieniu programu użytkownik proszony jest o wybór typu klasyfikatora:

```
1 - Email
2 - SMS
```

Po zakończeniu trenowania modelu możliwe jest interaktywne wprowadzanie wiadomości tekstowych, które zostaną sklasyfikowane jako spam lub wiadomości poprawne. Wprowadzenie pustej linii kończy działanie programu.

4.5 Uruchamianie testów jednostkowych

Projekt zawiera testy jednostkowe weryfikujące poprawność działania klasyfikatorów. Testy można uruchomić poleceniem:

```
dotnet test
```

Testy obejmują zarówno przypadki wiadomości spam, jak i wiadomości poprawnych dla klasyfikatora e-mail oraz SMS.

5 Modele danych

Do reprezentacji danych wejściowych oraz wyników predykcji wykorzystano następujące klasy.

5.1 Model danych tekstowych

Dla wiadomości e-mail wykorzystywany jest model:

- `Text` – treść wiadomości,
- `Label` – wartość logiczna określająca, czy wiadomość jest spamem.

Dla wiadomości SMS zastosowano osobny model, w którym etykieta zapisana jest jako tekst (`"ham"` lub `"spam"`) i mapowana na wartość logiczną podczas trenowania modelu.

5.2 Model predykcji

Wynik predykcji reprezentowany jest przez klasę zawierającą:

- `PredictedLabel` – przewidywana klasa,
- `Probability` – estymowane prawdopodobieństwo przynależności do klasy spam.

Wartość `Probability` mieści się w przedziale od 0 do 1 i wynika bezpośrednio z zastosowanego algorytmu regresji (wykorzystano BinaryClassification.Trainers.SdcaLogisticRegression).

6 Interfejs klasyfikatora

Wspólny kontrakt dla wszystkich klasyfikatorów został zdefiniowany za pomocą interfejsu:

- `Train(string dataPath)` – trenowanie modelu na podstawie danych z pliku,
- `Predict(string text)` – klasyfikacja pojedynczej wiadomości.

Dzięki temu możliwe jest łatwe przełączanie się pomiędzy różnymi implementacjami klasyfikatorów (e-mail, SMS).

7 Proces trenowania modelu

Proces trenowania modelu w ML.NET składa się z następujących etapów:

1. Wczytanie danych z pliku CSV do struktury `IDataView`,
2. Podział danych na zbiór treningowy i testowy,
3. Przekształcenie tekstu na wektory cech (`FeaturizeText`),
4. Trenowanie modelu klasyfikacji binarnej,
5. Ewaluacja jakości modelu.

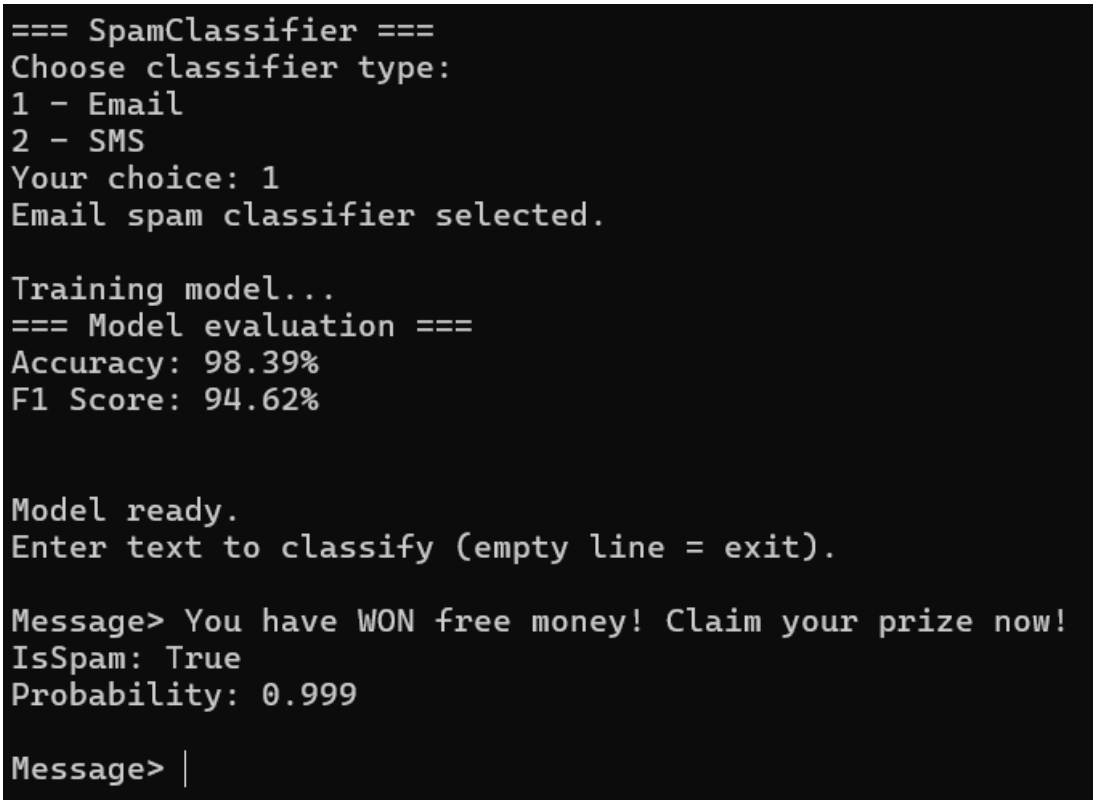
Zastosowanym algorytmem jest **SDCA Logistic Regression**, który dobrze sprawdza się w problemach klasyfikacji.

8 Aplikacja konsolowa

Aplikacja konsolowa umożliwia użytkownikowi:

- wybór typu klasyfikatora (e-mail lub SMS),
- trenowanie odpowiedniego modelu,
- interaktywne wprowadzanie wiadomości,
- uzyskanie wyniku klasyfikacji wraz z prawdopodobieństwem.

Program działa w pętli, umożliwiając klasyfikację wielu wiadomości bez ponownego trenowania modelu.



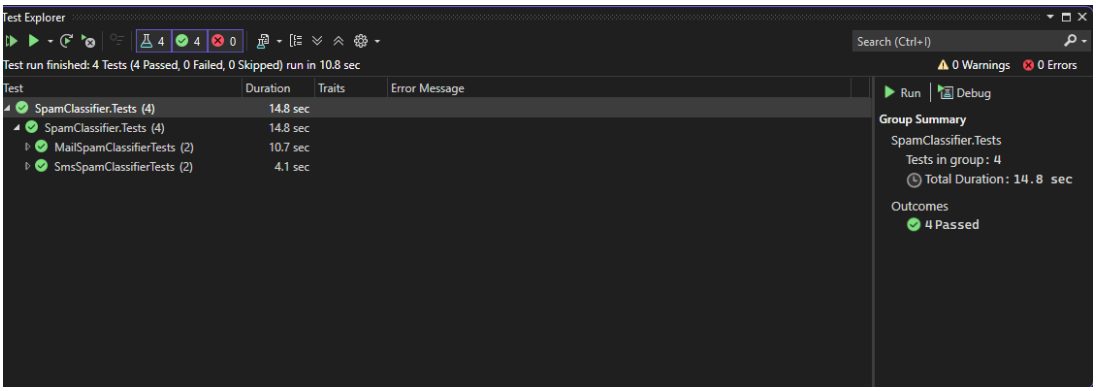
Rysunek 2: Przykładowe uruchomienie aplikacji konsolowej

9 Testy jednostkowe

Projekt zawiera podstawowe testy jednostkowe napisane z wykorzystaniem frameworka **xUnit**. Testy te sprawdzają poprawność działania klasyfikatorów zarówno dla wiadomości spam, jak i wiadomości poprawnych.

Testy:

- nie sprawdzają dokładnej wartości prawdopodobieństwa,
- weryfikują jedynie poprawność klasyfikacji logicznej.



Rysunek 3: Przykładowe uruchomienie wszystkich testów

10 Możliwe rozszerzenia

Projekt może zostać w przyszłości rozszerzony o:

- zapis i odczyt wytrenowanych modeli z plików,
- zmianę progu decyzyjnego klasyfikacji,
- integrację z aplikacją webową (ASP.NET),
- wykorzystanie bardziej zaawansowanych algorytmów NLP.

11 Podsumowanie

W ramach projektu zaimplementowano system klasyfikacji wiadomości tekstowych z wykorzystaniem biblioteki ML.NET. Przedstawione rozwiązanie pokazuje, że możliwe jest tworzenie własnych modeli uczenia maszynowego w środowisku .NET bez konieczności korzystania z zewnętrznych narzędzi czy języków programowania.