SMS Spam Filter
Dokumentacja do projektu

Wykonane w ramach przedmiotu PSZT przez:
Iwo Sokal, Paweł Martyniuk

Treść zadania

Napisać program, który korzystając z uczenia Bayesowskiego (Naive Bayes) będzie w stanie zaklasyfikować wiadomości SMS jako SPAM lub nie. Zbiór danych do użycia: SMS Spam Collection Dataset - https://www.kaggle.com/uciml/sms-spam-collection-dataset

Założenia

- Plik z danymi jest dzielony na zbiór uczący i zbiór testowy
- Wiadomości są klasyfikowane jako HAM lub SPAM
- Algorytm porównuje prawdopodobieństwa warunkowe $P(C=HAM|x_1,x_2,x_3,...x_n)$ i $P(C=SPAM|x_1,x_2,x_3,..., x_n)$, gdzie $x \in X$ są to słowa w wiadomości która jest klasyfikowana
- $P(C=X|x_1,x_2,x_3,...x_n) = P(C=X) * P(x_1,x_2,x_3,...x_n|C=X) / P(x_1,x_2,x_3,...x_n)$, w związku z tym nie trzeba liczyć $P(x_1,x_2,x_3,...x_n)$, ponieważ jest ono takie samo dla C = HAM jak i C = SPAM
- Korzystając z założenia naiwności używanego klasyfikatora $P(x_1,x_2,x_3,...x_n|C=X) = P(x1|C) * P(x2|C) * P(x3|C)$
- Parametry $x_1,x_2,x_3,...x_n$ są reprezentowane przez klasę Attribute, są one tworzone na podstawie danych uczących
- Do oceny naszego klasyfikatora używamy walidacji k-fold

Podział

Iwo Sokal:
- wczytywanie pliku .csv
- podział danych na zbiory uczące i testowe
- obliczanie prawdopodobieństw
- algorytm
- walidacja

Paweł Martyniuk:
- wczytywanie pliku .csv
- testy

Sposób rozwiązania

W programie użyliśmy 3 klas:
- Bayes - przedstawia konkretny model.
- Data - reprezentuje daną testową.
- Attribute - struktura do której są wczytywane dane uczące.

Naiwny klasyfikator bayesowski klasyfikuje daną testową poprzez porównanie 2 prawdopodobieństw $P(C=HAM|x_1,x_2,x_3,...x_n)$ i $P(C=SPAM|x_1,x_2,x_3,..., x_n)$, gdzie $x_1,x_2,x_3,..., x_n$ są to słowa znajdujące się w wiadomości testowej. Jeżeli $P(C=HAM|x_1,x_2,x_3,...x_n) >= P(C=SPAM|x_1,x_2,x_3,..., x_n)$, to wiadomosc jest klasyfikowana jako HAM, w przeciwnym wypadku jest klasyfikowana jako SPAM. Aby móc dokonać tego porównania, należy obliczyć następujące prawdopodobieństwa:
- P(C=HAM) - ilość wiadomości ham w danych uczących / ilość danych uczących
- P(C=SPAM) - ilość wiadomości spam w danych uczących / ilość danych uczących

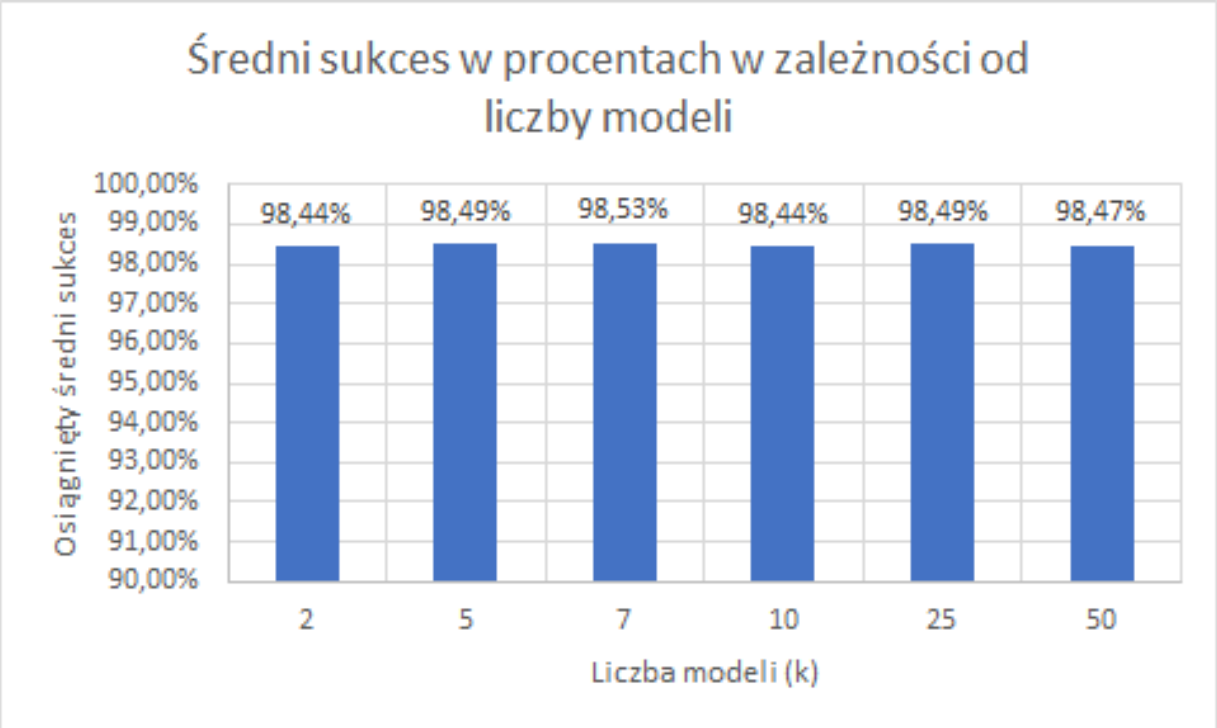
- $P(x_i|C=HAM)$ - ilość wystąpień danego słowa w wiadomościach ham (bez powtórzeń na poziomie tej samej wiadomości) w danych uczących / ilość wiadomości ham w danych uczących
- $P(x_i|C=SPAM)$ - ilość wystąpień danego słowa w wiadomościach spam (bez powtórzeń na poziomie tej samej wiadomości) w danych uczących / ilość wiadomości spam w danych uczących

Prawdopodobieństwa te są zapisywane w odpowiednich klasach. W ramach eksperymentów próbowaliśmy dzielić słowa na odpowiednie grupy, takie jak linki i numery telefonów oraz próbowaliśmy zmniejszać litery. Do oceny rozwiązania używamy walidacji krzyżowej, polegającej na przetestowaniu naszego algorytmu dla k różnych modeli i wyznaczeniu z nich średniej, przy czym każdy z modeli ma inny zbiór testowy i zbiór uczący.
Taki sposób rozwiązania wynika z tego że wydawał się najbardziej intuicyjny biorąc pod uwagę wzory z których korzystamy oraz wydaje się być zgodny z tym co wyczytaliśmy w ramach analizy zadania.

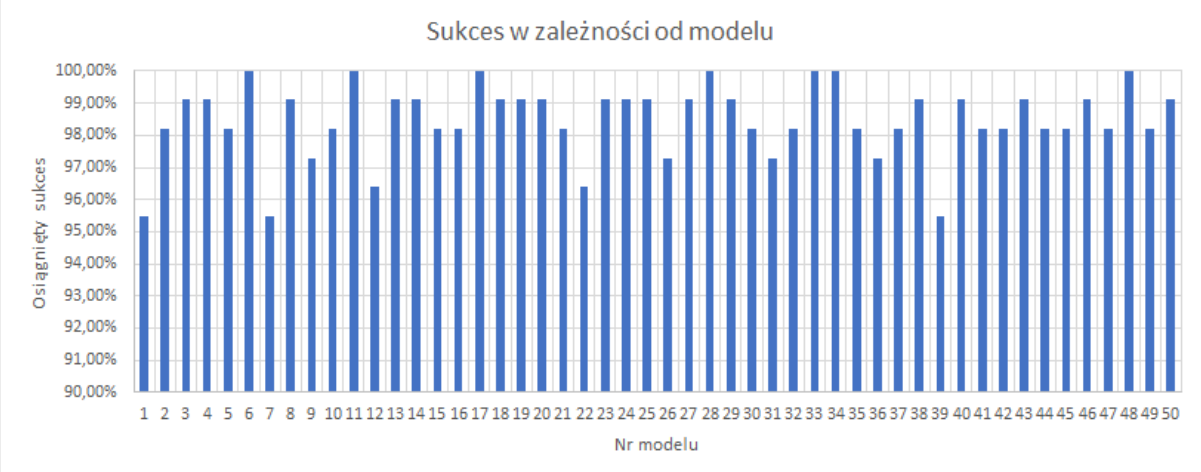
Uruchomienie programu

Program można wygodnie uruchomić w terminalu na Linuxie przy użyciu narzędzia Make, a następnie ./bin/program

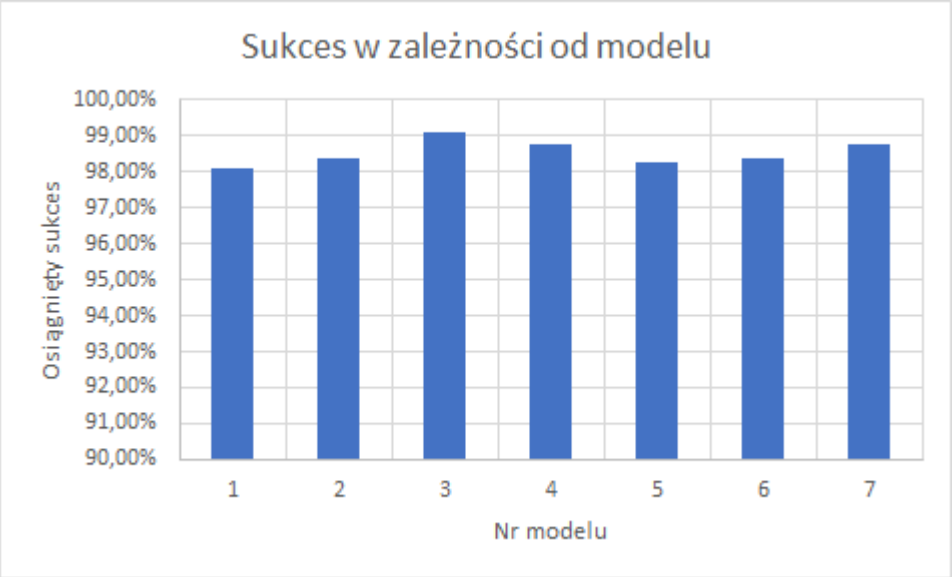
Testy



Wykres sukcesu dla 50 modeli:



Wykres sukcesu dla 7 modeli:



Średni sukces dla 7 modeli	
Średni sukces	Test
98.5284%	Bez zmiany wielkich liter na małe, grupowanie w linki i numery telefonów
98.1335%	Zmiana wszystkich wielkich liter na małe, grupowanie w linki i numery telefonów
98.4925%	Bez zmiany wszystkich wielkich liter na małe, grupowanie w linki, bez grupowania w numery telefonów

Wnioski

- Testy pokazały że przy użyciu naszego algorytmu jesteśmy w stanie filtrować wiadomości SMS z sukcesem ~98%. Jest to naszym zdaniem dość dobry wynik biorąc pod uwagę stosunkowo dużą liczbę wiadomości testowych.
- Wraz ze wzrostem k zwiększa się ilość danych testowych, a zmniejsza liczba danych uczących. W związku z tym rozbieżność między najlepszym, a najgorszym spośród k modeli jest tym większa im większe jest k.
- Najlepszy wynik uzyskujemy dla k = 7, natomiast dla każdego spośród k wynik jest dość podobny, z czego wynika że k nie ma dużego wpływu na działanie modelu.
- Zastosowanie grupowania ciągów cyfr i oznaczanie ich wewnątrz programu jako numer telefonu pozwala zwiększyć średni sukces poprawnego zakwalifikowania wiadomości o 0.03%. Natomiast zamiana wszystkich liter w danych na małe okazała się dużym zaskoczeniem, ponieważ średni sukces w tym przypadku zmalał o 0.3948%. Grupowanie w linki nie ma dużego znaczenia, głównie dlatego, że funkcja wykrywająca linki jest niewystarczająco selektywna. W związku z tym na podstawie testów najlepszym ustawieniem dla nas jest włączenie rozpoznawania linków i numerów telefonów oraz brak zamiany liter na małe.

Przemyślenia i możliwe usprawnienia

- wprowadzenie mieszania danych przed wyborem zbiorów uczących i testowych
- poprawa doboru atrybutów przez lepszy podział danych uczących na słowa