Sztuczna inteligencja i inżynieria wiedzy
Lista 4

Katarzyna Fojcik, Joanna Szołomicka, Teddy Ferdinan

7 marca 2025

1 Cel listy

Celem ćwiczenia jest praktyczne zapoznanie z prostymi algorytmami uczenia maszynowego oraz podstawowymi krokami realizacji projektu opartego o uczenie maszynowe: eksploracja danych i zdefiniowanie problemu, przygotowanie danych, dobór algorytmów i hiperparametrów, ocena algorytmów, poprawa wyników, przedstawienie wyników.

2 Wprowadzenie teoretyczne

Poniżej znajdują się informacje i podstawowe definicje pomocne w wykonaniu listy zadań.

2.1 Uczenie Maszynowe

Uczenie maszynowe *(ang. Machine Learning)*, jedna z dziedzin sztucznej inteligencji, szeroka kategoria algorytmów, których celem jest automatyczne wykonanie określonych zadań. Algorytmy, które pobierają dane, uczą się na podstawie tych danych, a następnie stosują to, czego się nauczyły do wnioskowania i wykonania zadania.

Typy uczenia maszynowego:

- **Uczenie nadzorowane** *(ang. supervised learning)* — sposób uczenia, w którym zbiór danych treningowych, na których uczy się algorytm, zawiera dołączone rozwiązanie problemu, tzw. etykiety albo klasy. Dwa główne obszary wykorzystujące uczenie nadzorowane to rozwiązywanie problemu **regresji** (przewidywanie wartości) i problem **klasyfikacji** (przewidywanie klas).
- **Uczenie nienadzorowane** *(ang. unsupervised learning)* – sposób uczenia modelu, w którym dane uczące są nieoznakowane tzn. nie posiadają etykiet. Mówiąc potocznie, posiadamy surowe dane które wrzucamy do modelu i zostawiamy algorytmowi całą pracę związaną ze znalezieniem powiązań między danymi (tzw. uczenie bez nauczyciela). Należą do nich m. in. algorytmy **analizy skupień** *(ang. clustering)* – inaczej klasteryzacja albo grupowanie oraz różnego rodzaju algorytmy **wizualizujące** dane i **redukujące wymiarowość**.
- **Uczenie ze wzmocnieniem** *(ang. RL - reinforcement learning)* – jego zadaniem jest interakcja ze środowiskiem na podstawie zebranych informacji. W przeciwieństwie do wymienionych wcześniej rodzajów, w uczeniu przez wzmacnianie nie przygotowuje się zestawu danych uczących, tylko środowisko, z którego model będzie zbierał dane automatycznie. Jego celem jest zmaksymalizowanie zwracanej przez nie nagrody. Środowisko może być zależne od celu nauki. W przypadku uczenia programu grającego w gry będzie to gra, wraz z jej zasadami, lub prawdziwy świat, w przypadku programu uczącego się sterować łazikiem.

2.2 Podział danych na zbiór uczący i walidacyjny

Zbiór uczący – to taki zestaw danych, który używamy do nauki algorytmu. Na podstawie tych danych model uczy się odpowiednio klasyfikować, buduje wszelkie zależności. Można powiedzieć, że przewiduje możliwe wyniki i podejmuje decyzje na podstawie przekazanych mu danych.

Zbiór walidacyjny - to taki zestaw danych, którego używamy do przeprowadzenia nieobciążonego testu modelu, który przeszkoliliśmy na danych treningowych (uczących). Test ten przeprowadzamy podczas wyboru modelu albo dobierając zestaw hiperparametrów. Ważnym jest, żeby dane zawarte w zbiorze walidacyjnym nie były używane wcześniej do nauki modelu, ponieważ nie będą wtedy nadawać się do obiektywnego, nieobciążonego testowania.

Zbiór testowy – kiedy już wybraliśmy model, nadchodzi czas na przetestowanie go na danych ze zbioru testowego. Bardzo ważne jest, by dane te nie były wcześniej używane do uczenia czy walidacji modelu, ponieważ chcemy wiedzieć, jak wybrany algorytm sprawdza się na danych, z którymi nigdy wcześniej nie miał do czynienia.

Walidacja krzyżowa (*ang. cross-validation*) to alternatywny sposób przygotowania zestawów treningowych i testowych dla modelu. Charakteryzuje sie on **większą skutecznością w ocenianiu faktycznej dokładności modelu**, ale jest mniej wydajny od naiwnej metody. Metoda ta polega na **podzieleniu zestawu danych na podzestawy** i utworzeniu z nich **zestawów danych walidaycjnych i treningowych**.

Istnieje wiele rodzajów walidacji krzyżowej. Bazowa K-krotna walidacja krzyżowa zakłada, że dane są niezależne, i identycznie rozłożone (IID - Independent and Identically Distributed). Dzieli wszystkie próbki na k grup, jeśli to możliwe o jednakowych rozmiarach. Algorytm ML jest nauczany przy użyciu $k-1$ grup, a pominięta grupa jest używana do walidacji. Wykonuje się zatem k treningów, a otrzymane wyniki z k przeprowadzonych walidacji uśrednia się.

Stosunek podziału danych na zestaw treningowy, walidacyjny i testowy powinien zależeć od liczności próbek. Wyróżnia się różne praktyki podziału, jednak generalnie przyjmuje się, że dla mniejszych zbiorów priorytetyzujemy konieczność trenowania algorytmu na jak największej liczbie próbek. Wówczas zestaw walidacyjny (i testowy) jest stosunkowo mniejszy (w przypadku walidacji krzyżowej wybieramy wówczas większą liczbę k, np. 10). Mając natomiast do dyspozycji kilka tysięcy próbek, możemy zdecydować się na zwiększenie części danych walidacyjnych (alterantywnie zmniejszenie liczby grup do $k = 5$).

2.3 Przeuczenie i niedouczenie

Przeuczenie następuje, gdy model uczy się wzorców zbyt specyficznych dla danych treningowych, potencjalnie ujmując zarówno istotne szczegóły, jak i szum, co sprawia, że jest on zawodny przy przewidywaniu danych niewidzianych. Przeuczony model koncentruje się zbytnio na szczegółach danych treningowych, zamiast uczyć się ogólnych wzorców. Przeuczenie może wynikać z: (1) danych treningowych, które nie w pełni reprezentują rzeczywiste dane; oraz/lub (2) zbyt złożonego modelu (tj. liczba parametrów modelu jest zbyt duża w stosunku do rozmiaru zbioru danych). Aby zaradzić przeuczeniu, stosuje się techniki takie jak regularyzacja, dropout, wczesne zatrzymanie treningu, przycinanie parametrów oraz bayesowskie priory.

Niedouczenie następuje, gdy model jest zbyt uproszczony, aby uchwycić wzorce, co prowadzi do słabego działania zarówno na danych treningowych, jak i testowych. Niedouczenie może wynikać z: (1) zbyt prostego modelu (tj. liczba parametrów modelu jest zbyt mała w stosunku do rozmiaru zbioru danych); (2) niewystarczającej ilości danych treningowych; (3) niewystarczających cech wejściowych; oraz/lub (4) nadmiernej regularyzacji, która utrudnia uczenie się.

2.4 Naiwny Klasyfikator Bayesa

Naiwny Klasyfikator Bayesa jest prostym klasyfikatorem probabilistycznym, który naiwnie zakłada niezależność i tą samą istotność cech (predyktorów) dla ustalonej etykiety klasy, opiera się o Twierdzenie Bayesa.

Definicja (Twierdzenie Bayesa) *Niech A i B są zdarzeniami, $P(B) > 0$, $P(A|B)$ oznacza prawdopodobieństwo warunkowe, wtedy:*

$$\begin{aligned} P(A \wedge B) &= P(A|B)P(B) = P(B|A)P(A) \\ P(B|A) &= \frac{P(A|B)P(B)}{P(A)} \end{aligned} \tag{1}$$

Definicja (Naiwny Klasyfikator Bayesa) *Niech $X_1, \ldots, X_n$ to cechy, wtedy:*

$$\begin{aligned} P(X_1|X_2,Y) &= P(X_1|Y) \\ P(X_1,X_2,Y) = P(X_1|X_2,Y)P(X_2,Y) &= P(X_1|X_2,Y)P(X_2|Y)P(Y) \\ P(X_1,X_2|Y) &= P(X_1|Y)P(X_2|Y) \\ P(X_1,X_2,\ldots,X_n|Y) &= \prod_{i=1}^{n} P(X_i|Y) \end{aligned} \tag{2}$$

2.5 Drzewo decyzyjne

Drzewo decyzyjne w uczeniu maszynowym to reprezentacja klasyfikatora w postaci drzewa wspomagającego proces decyzyjny. Budowanie takiego drzewa bazuje na różnych algorytmach i ich parametrach, które wyliczane są dla każdego nowego węzła drzewa decyzyjnego. Przykładowe parametry to entropia i przyrost wiedzy.

Definicja (Entropia) *Entropia jest to miara zróżnicowania danych dana wzorem 3.*

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{3}$$

gdzie S to aktualny zbiór danych dla którego liczona jest entropia (dla każdego węzła drzewa będzie to inny - odpowiednio mniejszy zbiór danych), X to zbiór klas w zbiorze S, $p(x)$ to stosunek liczby elementów z klasy x do liczby elementów w zbiorze S.

Definicja (Przyrost wiedzy - Information Gain) *Przyrost wiedzy $G(A)$ jest to miara różnicy entropii przed i po rozbiciu zbioru danych przy pomocy danego atrybutu (cechy).*

$$G(A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v) \tag{4}$$

gdzie $H(S)$ to entropia dla zbioru S, $Values(A)$ to zbiór wszystkich wartości atrybutu A, S_v to podzbiór S, taki że $S_v = \{s \in S : A(s) = v\}$, $H(S_v)$ to entropia podzbioru S_v.

Drzewa decyzyjne są bardzo podatne na zjawisko nadmiernego dopasowania (*ang. overfitting*). Wraz ze wzrostem głębokości, drzewa są w stanie bardzo dobrze dostosować się do danych uczących, ale równocześnie pogarsza się ich zdolność generalizacji (błąd na zbiorze testowym rośnie).

2.6 PCA (ang. Principal Component Analysis)

PCA (analiza głównych składowych) to statystyczna metoda analizy czynnikowej: zbiór danych składających się z N obserwacji, a każda obejmuje K zmiennych można zinterpretować jako chmurę N punktów w przestrzeni K-wymiarowej. Celem PCA jest taki obrót układu współrzędnych, aby zmaksymalizować w pierwszej kolejności wariancję pierwszej współrzędnej, następnie drugiej itd. Przekształcone wartości współrzędnych to składowe główne (początkowe wyjaśniają najwięcej zmienności). W uczeniu maszynowym służy do zmniejszania wymiaru danych.

2.7 Metryki

Aby dokonać oceny przeprowadzonej klasyfikacji wykorzystuje się wyniki różnych metryk (zazwyczaj tylko dla danych walidacyjnych):

2.7.1 Macierz pomyłek

W zależności od działania klasyfikatora wyróżnia się cztery przypadki (Tabela 1):

- **TP (ang. True Positive)** – liczba prawdziwie rozpoznanych przypadków pozytywnych

- **FP (ang. False Positive)** – liczba nieprawdziwie rozpoznanych przypadków pozytywnych
- **FN (ang. False Negative)** – liczba nieprawdziwie rozpoznanych przypadków negatywnych
- **TN (ang. True Negative)** – liczba prawdziwie rozpoznanych przypadków negatywnych

		Rzeczywistość	
		Klasa pozytywna	Klasa negatywna
Predykcja	Klasa pozytywna	TP	FP
	Klasa negatywna	FN	TN

Tabela 1: Macierz pomyłek.

2.7.2 Dokładność (inaczej trafność, ang. accuracy)

Dokładność ACC, to najczęściej podawany wskaźnik, który pozwala nam ocenić jakość klasyfikacji danych. Dowiadujemy się, jaka ich część ze wszystkich zaklasyfikowanych, została zaklasyfikowana poprawnie. Czyli sumę poprawnych klasyfikacji z danej kategorii (TP) oraz poprawnej z innych kategorii (TN) dzielimy przez liczbę wszystkich klasyfikowanych przypadków.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} \tag{5}$$

2.7.3 Czułość (inaczej kompletność, ang. recall/sensitivity/true positive rate)

Czułość TPR mówi nam o tym, jaki jest udział prawidłowo zaprognozowanych przypadków pozytywnych (TP) wśród wszystkich przypadków pozytywnych (również tych, które błędnie zostały zaklasyfikowane do negatywnych – FN).

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{6}$$

2.7.4 Precyzja (inaczej dokładność, ang. precision/positive predictive rate)

Dzielimy liczbę prawidłowo zaprognozowanych pozytywnych wartości (TP) przez sumę wszystkich zaprognozowanych pozytywnie (również tych błędnie zaprognozowanych w ten sposób), czyli TP+FP. W efekcie dowiadujemy się, ile wśród przykładów zaprognozowanych pozytywnie jest rzeczywiście pozytywnych.

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{7}$$

2.7.5 F1-score

F1-score to średnia harmoniczna pomiędzy precyzją (precision) i czułością (recall). Im bliższa jest jedynki, tym lepiej to świadczy o algorytmie klasyfikującym. W najlepszym przypadku przyjmuje wartość 1, kiedy mamy do czynienia z idealną czułością i precyzją.

$$\mathrm{F}_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{8}$$

3 Zadanie

Zmierz się z problemem identyfikacji typu wzoru morfologicznego płodu na podstawie cech diagnostycznych kardiotokogramu (CTG, z *ang. cardiotocogram*). W tym celu wykorzystaj dostarczony plik `cardiotocography_v2.csv`. Te dane zostały zaadaptowane z oryginalnego zbioru danych Cardiotocography[1] do celów tego ćwiczenia laboratoryjnego.

Plik zawiera 2126 wierszy danych kardiotokogramów płodu (CTG). Celem jest przeprowadzenie klasyfikacji 10 typów wzorców morfologicznych płodu na podstawie 21 cech diagnostycznych. W cechach występują pewne brakujące wartości. Kolumny w pliku to:

[1] D. Campos and J. Bernardes. "Cardiotocography,"UCI Machine Learning Repository, 2000. [Online]. URL: https://doi.org/10.24432/C51S4N

- LB - [cecha] wartość bazowa FHR (uderzeń na minutę)
- AC - [cecha] liczba przyspieszeń na sekundę
- FM - [cecha] liczba ruchów płodu na sekundę
- UC - [cecha] liczba skurczów macicy na sekundę
- DL - [cecha] liczba łagodnych spowolnień na sekundę
- DS - [cecha] liczba poważnych spowolnień na sekundę
- DP - [cecha] liczba długotrwałych spowolnień na sekundę
- ASTV - [cecha] odsetek czasu z nieprawidłową zmiennością krótkoterminową
- MSTV - [cecha] średnia wartość zmienności krótkoterminowej
- ALTV - [cecha] odsetek czasu z nieprawidłową zmiennością długoterminową
- MLTV - [cecha] średnia wartość zmienności długoterminowej
- Width - [cecha] szerokość histogramu FHR
- Min - [cecha] minimum histogramu FHR
- Max - [cecha] maksimum histogramu FHR
- Nmax - [cecha] liczba pików histogramu
- Nzeros - [cecha] liczba zer histogramu
- Mode - [cecha] modus histogramu
- Mean - [cecha] średnia histogramu
- Median - [cecha] mediana histogramu
- Variance - [cecha] wariancja histogramu
- Tendency - [cecha] tendencja histogramu
- CLASS - [cel] kod klasy wzorca morfologicznego płodu (od 1 do 10)

Zadania powinny być wykonane z pomocą Pythona i/lub WEKI.

Punktacja:

1. eksploracja danych – przedstaw podstawowe dane statystyczne i uwagi dotyczące cech i etykiet zbioru danych. (10 punktów)
2. przygotowanie danych – podziel dane na zestaw uczący i walidacyjny (alternatywnie użyj walidacji krzyżowej). Przygotuj dane przy użyciu metody radzenia sobie z brakującymi wartościami (sugerowane: usunięcie, imputacja, interpolacja). Następnie użyj 2 różnych metod (osobno) do przetwarzania danych (sugerowane: normalizacja, standaryzacja, dyskretyzacja, selekcja cech, PCA). Porównaj wyniki klasyfikacji bez przetwarzania danych z wynikami uzyskanymi przy użyciu różnych metod przetwarzania danych.
3. klasyfikacja – przetestuj klasyfikatory i zbadaj wpływ na wyniki: naiwny klasyfikator Bayesa oraz drzewo decyzjne używając przynajmniej 3 różnych zestawów hiperparametrów. (40 punktów)

   Bonus – Przetestuj (ze zrozumieniem!) bardziej zaawansowane algorytmy, takie jak Las losowy czy Klasyfikator wektorów nośnych (SVM, z *ang. Support Vector Machines*). (5 punktów)

   Bonus – Dla jednego wybranego klasyfikatora zaproponuj jedną metodę łagodzenia przeuczenia. Przeprowadź eksperyment z wykorzystaniem tego podejścia i porównaj wyniki z eksperymentem bez niego. (5 punktów)

4. ocena klasyfikacji – do porównania wyników różnego typu przygotowania danych oraz wykorzystanego klasyfikatora użyj poznanych metryk oceny klasyfikacji i zinterpretuj wyniki. (20 punktów)

Do zadania przygotuj raport zawierający krótki opis wszystkich wykonywanych kroków oraz rezultatów zadania (najlepiej zebranych tabeli) wraz z interpretacją. W raporcie wskaż wykorzystane materiały źródłowe oraz krótko opisz biblioteki wykorzystane przy implementacji. Raport wyślij prowadzącemu przynajmniej na 24 godziny przed oddaniem listy.

4 Dodatek - filtry i klasyfikatory w Weka Explorer

4.1 Filtry

- **dyskretyzacja atrybutu:** filters → supervised → attribute → Discretize lub filters → unsupervised → attribute → Discretize
- **normalizacja atrybutu:** filters → unsupervised → attribute → Normalize
- **PCA:** filters → unsupervised → attribute → PrincipalComponents
- **usuwanie atrybutu:** filters → unsupervised → attribute → Remove
- **uzupełnianie wartości atrybutu:** filters → unsupervised → attribute → ReplaceMissingValues
- **standaryzacja atrybutu:** filters → unsupervised → attribute → Standardize
- **usuwanie instancji o zadanych wartościach atrybutów:** filters → unsupervised → instance → RemoveWithValues

4.2 Selekcja atrybutów

Karta *Select attributes* - wybór przyciskiem *Attribute Evaluator*:

- GainRatioAttributeEval
- InfoGainAttributeEval
- PrincipalComponents

4.3 Klasyfikatory

Karta *Classify*:

- **naiwny klasyfikator Bayesa:** classifiers → bayes → NaiveBayes lub NaiveBayesMultinomial (w zależności od potrzeb)
- **drzewo decyzyjne C4.5:** classifiers → trees → J48
- **zespół klasyfikatorów:**
  - **boosting:** classifiers → meta → AdaBoostM1
  - **bagging:** classifiers → meta → Bagging
  - **różne rodzaje klasyfikatorów:** classifiers → meta → Stacking

5 Dodatek - przydatne bilbioteki dla Pythona

- from sklearn.preprocessing import Normalizer, StandardScaler
- from sklearn.decomposition import PCA
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.naive_bayes import GaussianNB