Wstęp do ML. KLasyfikacja

ML Wykład 1 i 2, Tomasz Dzido

Podstawowe pojęcia i różnice

Uczenie maszynowe (Machine learning, ML) jest częścią sztucznej inteligencji lub inteligencji obliczeniowej i zajmuje się algorytmami do analizowania danych i uczenia się na podstawie tych danych, a następnie wykorzystywania zdobytej wiedzy do np. przewidywania czegoś. Jest kilka głównych rodzajów ML.

Głębokie uczenie (Deep learning) jest to podzbiór uczenia maszynowego, który zajmuje się klasyfikowaniem, rozpoznawaniem, wykrywaniem i opisywaniem danych przy pomocy głębokich sztucznych sieci neuronowych ("na głębszym poziomie" niż tradycyjne uczenie maszynowe).

Źródło: https://petrospsyllos.com/pl/prezentacje/blog/559-roznice-miedzy-inteligencja-obliczeniowa-sztuczna-inteligencja-uczeniem-maszynowym-i-glebokim-uczeniem

Data Science a AI i ML

- https://becominghuman.ai/connection-between-data-science-ml-and-ai-d1c18d89b0bd
- **Data Science** odnosi się do procesu wydobywania użytecznych informacji z danych. To interdyscyplinarne podejście łączy różne dziedziny informatyki, procesy i metody naukowe oraz statystyki w celu uzyskania danych w sposób zautomatyzowany.
- **Data Science** oznacza kompleksowy termin, który składa się z aspektów ML, które zaś jest elementem sztucznej inteligencji, w której różnorodne cele są osiągane na zupełnie nowym poziomie. ML i AI są nieodłączną częścią data science.

Jakie zagadnienia u nas?

- Wiele zagadnień z podstaw ML : klasyfikacja, grupowanie, regresja, podstawy sieci neuronowych….
- Zagadnienia ze zgłębiania danych (data science): eksploracja, oczyszczanie, przetwarzanie danych itd..
- Inne zagadnienia z ML i AI.

Podstawowa literatura

- Marcin Szeliga: Praktyczne uczenie maszynowe, PWN 2019
- Joel Grus: Data science od podstaw, Helion 2018
- Drew Conway, John Myles White: Uczenie maszynowe, Helion 2015
- Marcin Szeliga: Data Science i Uczenie Maszynowe, PWN 2017
- Sebastian Raschka, Vahid Mirjalili: Python Uczenie Maszynowe, wyd. 2, Helion 2019 (wyd. 3. 2021)
- Samouczki R, Python, kursy online University of Stanford
- Linki internetowe – podawane na bieżąco

Narzędzia

- Język Python, twórca Guido van Rossum (kod źródłowy 1991, 1 wersja 1994, 2 wersja 2000, 3 wersja 2008), projekt darmowy i otwarty, dużo dodatkowych bibliotek, najczęściej używany w ML, biblioteka NumPy, SciPy, scikit-learn, pandas i inne.
- Język R, twórcy Ross Ihak, Robert Gentleman (pierwsza wersja 1995, stabilna 2000), otwarty i darmowy język analizy danych, najczęstsze narzędzie do statystycznej analizy danych, miliony użytkowników, dziesiątki tysięcy bibliotek, The Comprehensive R Archive Network https://cran.r-project.org/
- Słabe strony R: niska wydajność, brak skalowalności, brak zintegrowanych narzędzi do udostępniania wyników. Rozwiązanie: np. produkty Microsoft (Microsoft R Open) itd.

Dystrybucja oprogramowania

- Rstudio to popularny, darmowy, graficzny edytor języka R, https://rstudio.com/products/rstudio/download
- Ładowanie i instalowanie pakietów:
- library(tm)
- print(require(XML))
- install.packages(”tm”, dependencies=TRUE)
- setwd(”~/Pobrane/”)
- install.packages(”Rcurl_1.5-0.tar.gz”, repos=NULL, type=”source”)
- (repos oznacza blokowanie próby pobrania z CRAN (binaries)

Dystrybucja oprogramowania

- Ciekawa dystrybucja Anaconda, https://www.anaconda.com/download/
- W 2025 najnowsza wersja Pythona to 3.12, ale niektórzy analitycy nadal korzystają z wersji 2.7 (nie ma wstecznej kompatybilności)
- Anaconda Navigator, narzędzia Spyder, Jupyter https://docs.anaconda.com/anaconda/user-guide/getting-started/#open-nav-win
- Anaconda Prompt (python, exit()) etc.

Wstęp do Uczenia Maszynowego

- **Uczenie maszynowe** polega na uczeniu maszyn na przykładach.
- Pozwala rozwiązać problemy, których nie potrafimy rozwiązać algorytmicznie np. rozpoznawanie obrazów.
- Radzi sobie w problemach gdy algorytm istnieje ale jest bardzo kosztowny albo nieskuteczny np. problem oszustw popełnianych przez użytkowników kart bankowych (wiele założeń, przypadków, zmieniające się zachowania użytkowników itp.)

Wstęp do uczenia maszynowego

- Powinniśmy użyć ML gdy: nie znamy rozwiązania (nie ma eksperta), rozwiązania szybko się zmieniają (giełda papierów wartościowych), rozwiązania muszą być dostosowane do poszczególnych użytkowników.
- ML nie zawsze jest możliwe. Wymaga odpowiedniej liczby przykładów. Niezbędnym jego elementem jest model. Konieczne jest zdefiniowanie funkcji kosztu użytej do oceny i poprawy jakości modelu.
- ML to zbiór różnych technik, dzielony według różnych kryteriów.

Podział technik ML ze względu na format przypadków treningowych

- 1. **Uczenie nienadzorowane (ang. unsupervised learning):** przykłady niezawierające odpowiedzi np. zdjęcia bez opisu; stosowane do: analizy skupień, redukcji wymiarów, wykrywania anomalii lub abstrakcyjnych cech przypadków (np. kształty na kodowanych obrazach).
- 2. **Uczenie nadzorowane (ang. supervised learning)** polega na uczeniu zależności między zmiennymi wejściowymi a zmienną docelową, wymaga opisania każdego przypadku. Stosowane do klasyfikacji (zmienna docelowa dyskretna) lub regresji (zmienna docelowa ciągła). Ponad 90% modeli ML to takie uczenie.
- 3. **Uczenie ze wzmocnieniem (ang. reinforcement learning)** polega na uczeniu metodą prób i błędów, model nagradzany za postępy w uczeniu (nagroda za wyprzedzenie samochodu w symulatorze, wygranie partii szachów ip.). Stosowane do nauki różnych gier (np. AlphaGo Zero), w robotyce i automatyce, do nauki działania maszyn, poszukiwania optymalnej strategii np. w negocjacjach handlowych.

Podział technik ML ze względu na rodzaj postawionej hipotezy

- 1. Modele regresji (nieco myląca nazwa): hipoteza ma postać funkcji, uczenie ma na celu znalezienie optymalnych wartości parametrów tej funkcji, przykłady: regresja liniowa, logistyczna (to raczej partycjonowanie), sztuczne sieci neuronowe, algorytm grupowania metodą k-średnich.
- 2. Modele partycjonujące: hipoteza ma postać reguł dzielących przykłady, przykłady: drzewa decyzyjne, las drzew decyzyjnych, wzmocnione drzewa decyzyjne. Modele takie mogą być także używane do klasyfikacji i regresji.

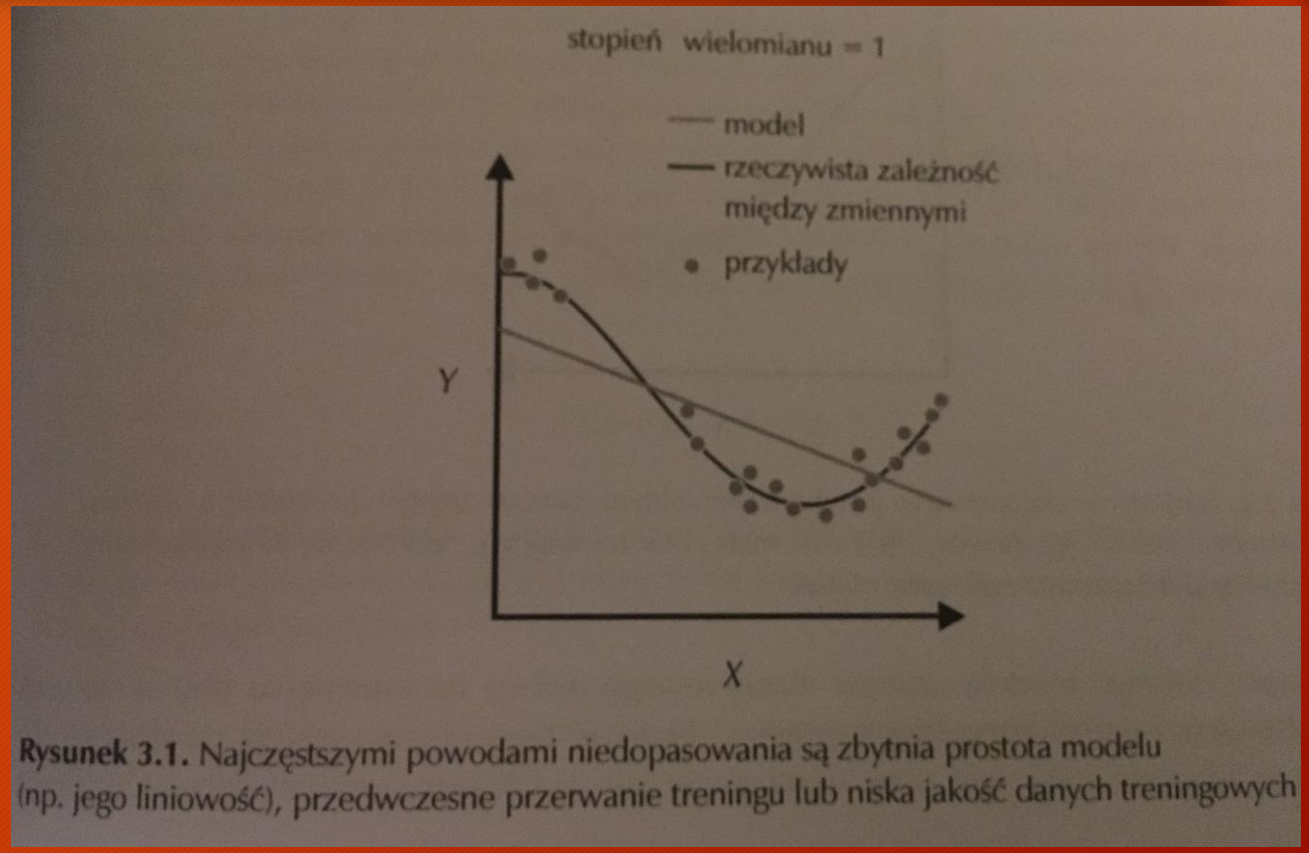
Proces uczenia i jego etapy

Uczenie maszynowe to nie jest zapamiętywanie przypadków. Przykład autobus z pękniętą szybą z niedziałającym kasownikiem i tymi samymi pasażerami nie zdarzy się szybko następny raz, a wiemy jak się zachować. Mamy bowiem zdolność do uogólniania, czyli generalizacji. To najważniejszy etap naszego uczenia się ale także uczenia maszynowego.

1. Abstrakcja – w jej trakcie jakościowe opisy zdarzeń zamieniane są na uogólnione teorie. Wynikiem mogą być liczby albo etykiety (kolory, opisy stanów cywilnych), wartości etykiet muszą być jednak mierzalne. U ludzi proces zachodzi podświadomie, w przypadku ML możemy sterować tym procesem.
2. Generalizacja – zastosowanie doświadczeń zdobytych w przeszłości do bieżących sytuacji, dotyczy to również ML.

Przesadna generalizacja

- Mamy wtedy do czynienia z **niedopasowaniem (ang. underfitting)** modelu, np. skoro w taxi ktoś zapłacił kartą, to da napiwek.

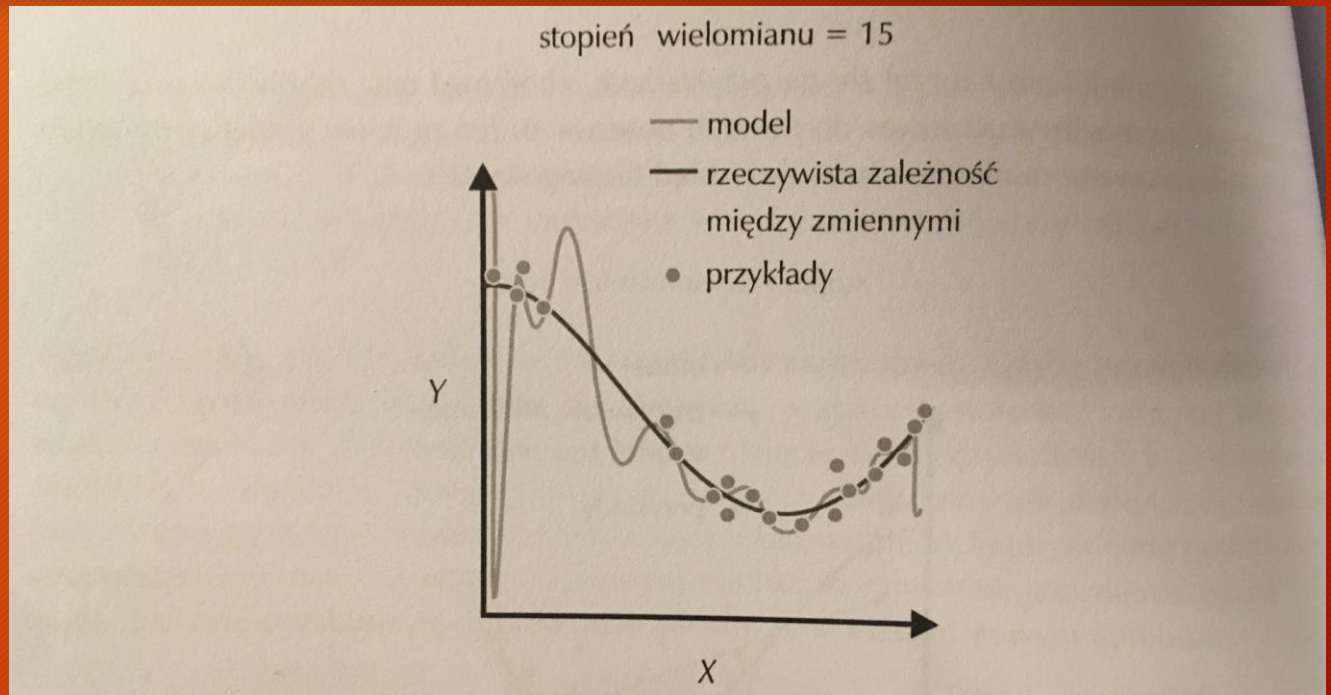


Rysunek 3.1. Najczęstszymi powodami niedopasowania są zbytnia prostota modelu (np. jego liniowość), przedwczesne przerwanie treningu lub niska jakość danych treningowych

Źródło: Marcin Szeliga: Praktyczne uczenie maszynowe, PWN 2019

Niezdolność do uogólniania

- Fantastyczna pamięć do przypadków i automatyczne postępowanie w nowej sytuacji. W AI jest to **nadmierne dopasowanie (ang. overtraining/overfitting)**, zapamiętany nawet szum. Model mało wiarygodny.

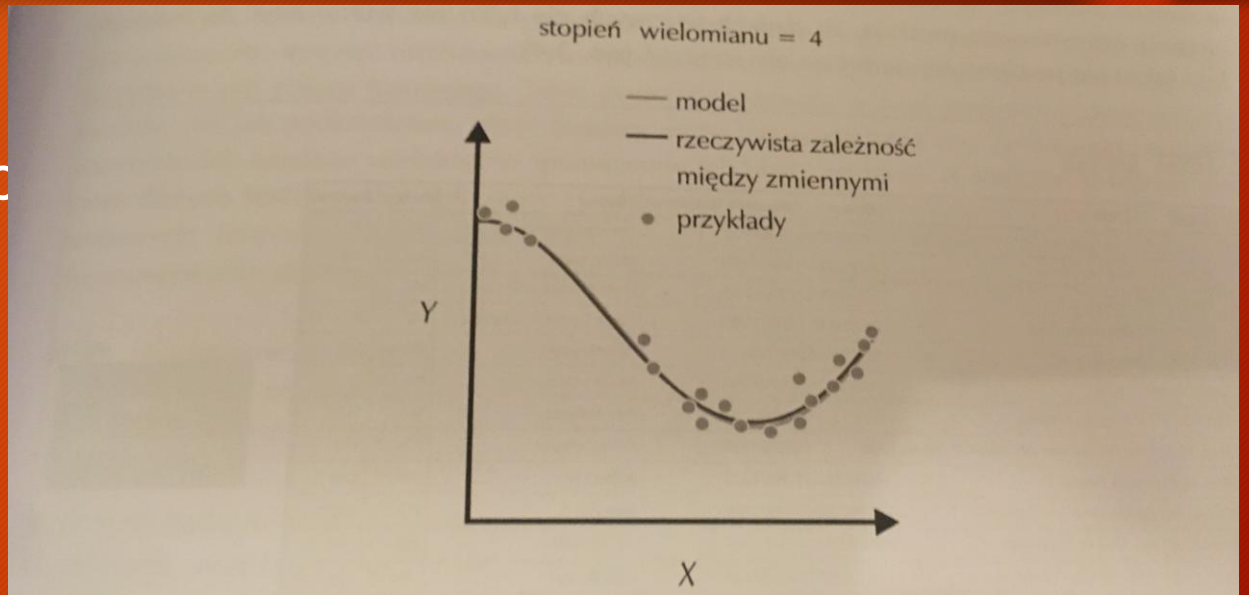


Rysunek 3.2. Nadmierne dopasowanie jest fundamentalnym, bardzo częstym problemem uczenia nadzorowanego. Wśród jego przyczyn jest zbyt mały zbiór treningowy, nadmiernie skomplikowany lub nieodpowiednio sparametryzowany model

Źródło: Marcin Szeliga: Praktyczne uczenie maszynowe, PWN 2019

Dobry model uczenia maszynowego

- To taki, w którym unikamy obu powyższych skrajności. Trudność zbudowania takiego modelu, że do treningu wykorzystywane są jedynie dane treningowe a minimalizowanie błędu może prowadzić do nadmiernego dopasowania modelu.



Rysunek 3.3. *Właściwe dopasowanie modelu wymaga znalezienia złotego środka między jego niedopasowaniem a nadmiernym dopasowaniem*

Źródło: Marcin Szeliga: Praktyczne uczenie maszynowe, PWN 2019

Praktyczne wskazówki

- 1. Do oceny modelu musimy użyć danych testowych („nie powinniśmy nawet na nie spojrzeć”).
- 2. Porównywanie stawianych przez algorytmy ML hipotez wymaga zdefiniowania funkcji kosztu. Jej wartość (błąd, koszt, ryzyko, zysk) określa rozbieżność między wynikami predykcji z rzeczywistymi wartościami zmiennej wyjściowej. Rozbieżność może być wyznaczona na wiele sposobów – będzie o tym mowa.
- 3. Błąd treningowy (empiryczny) jest średnim błędem predykcji dla danych treningowych. Można go ograniczyć do 0 (zapamiętanie wszystkich przykładów – ale czy to dobry model?).
- 4. Naszym celem jest zminimalizowanie błędu testowego (realnego), nazywanego też błędem generalizacji czyli uśrednionego błędu predykcji dla danych testowych.

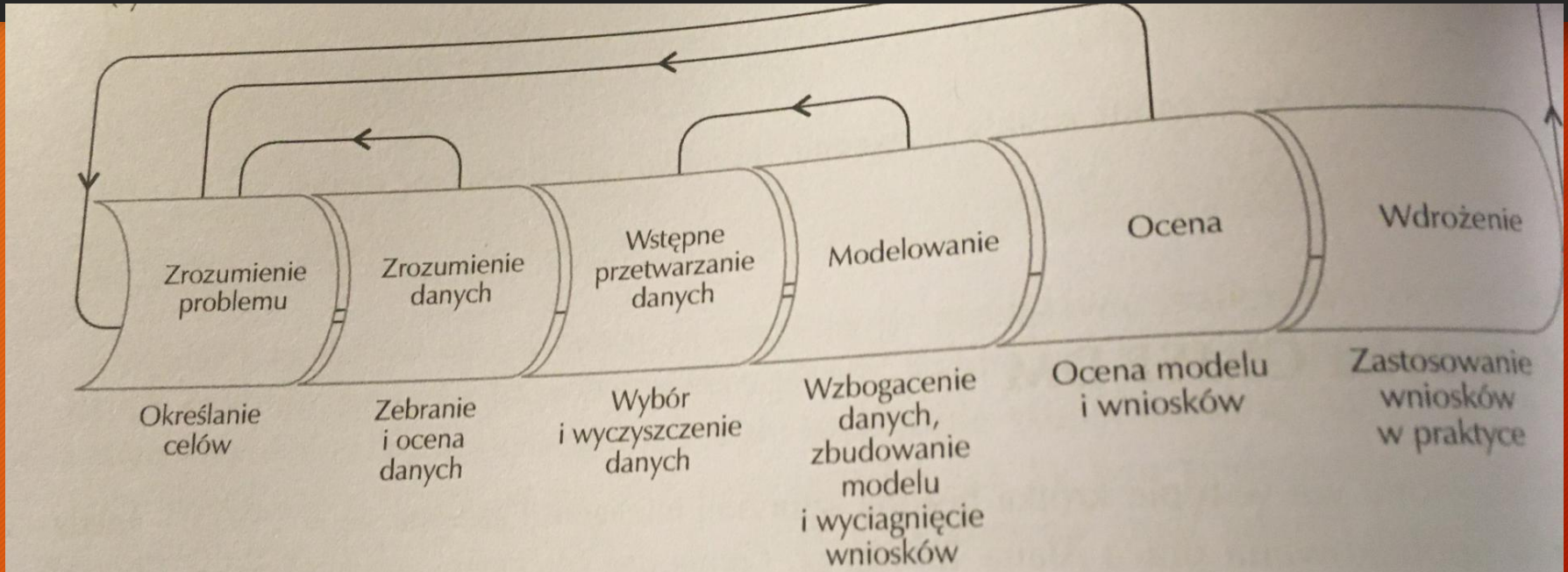
Praktyczne wskazówki – c.d.

- 5. Modele powinniśmy wybierać na podstawie ich błędu realnego. Możemy odkryć nadmierne dopasowanie ale porównywanie modeli oznacza wielokrotne sprawdzanie ich błędu realnego. Z każdym kolejnym użyciem danych testowych zwiększamy ryzyko nadmiernego dopasowania modelu do danych testowych. Może to być niebezpieczne.
- 6. Wykrycie nadmiernego dopasowania modelu do danych testowych wymaga sprawdzenia go na nowych, nieużytych danych. Dlatego zbiór treningowy dzieli się na: dane treningowe i walidacyjne.
- 7. Dane najczęściej dzieli się losowo. W zależności od ilości danych: kilkadziesiąt tysięcy: 70:20:10, kilkaset tysięcy: 80:15:5, kilkadziesiąt milionów: 95:4:1.
- 8. Błąd testowy ma 2 składowe: błąd systematyczny (estymacji) oraz wynikający ze zmienności modelu błąd aproksymacji. Ten pierwszy mierzy o ile gorszy jest model od modelu idealnego, ten drugi mierzy jakość modelu (różnie wyznaczany).

Metodyka CRISP-DM (Cross Industry Standard Process for Data Mining)

- Ta opisowa metodyka stworzona w 1997 określa kolejne etapy projektu wraz z opisami zadań do wykonania w każdym z nich. Pozwala budować wiarygodne modele czyli zapobiega próbom naginania rzeczywistości (danych) do postawionych hipotez. Wiele opublikowanych prac naukowych zawiera fałszywe wnioski (słynna krzywa Philipsa, negatywna korelacja bezrobocia i inflacji, i wiele innych).

Metodyka CRISP-DM (Cross Industry Standard Process for Data Mining)



Rysunek 3.22. *Metodyka CRISP-DM podkreśla iteracyjny i zwinny charakter procesu odkrywania wiedzy z danych – wyniki każdego etapu są oceniane i na tej podstawie jest podejmowana decyzja o ewentualnym powtórzeniu któregoś z wcześniejszych kroków*

Źródło: Marcin Szeliga: Praktyczne uczenie maszynowe, PWN 2019

Metodyka TDSP (Team Data Science Process)

Zwinna, opracowana przez Microsoft z myślą o pracy zespołowej, metodyka prowadzenia projektów ML. Jej założenia są zgodne z metodyką CRISP-DM, ale kładzie nacisk na udokumentowanie poszczególnych zadań. Cykl życia projektu został podzielony na cztery etapy:

- zrozumienie problemu biznesowego,
- zdobycie i zrozumienie danych,
- modelowanie,
- wdrożenie.

Więcej na https://www.datascience-pm.com/tdsp/

Klasyfikator

- Najstarsza i najczęstsza metoda eksploracji danych
- Celem jest znalezienie funkcji klasyfikującej *f* (klasyfikatora) , która odwzorowuje rekord (x_i, y_i) na klasę *y*.
- Model klasyfikacyjny nauczony na podstawie danych historycznych (treningowych) będzie przypisywał nowe przypadki do jednej z klas.
- Dane treningowe stanowią zbiór przypadków (obserwacji), z których każdy jest opisany za pomocą listy atrybutów (zmiennych wejściowych, objaśniających, zmiennych *x*) i etykiety klas (zmiennej wyjściowej, objaśnianej , zmiennej *y*)

Klasyfikator

Tabela 5.1. W pierwszych czterech kolumnach zapisane są zmienne wejściowe, w piątej – etykiety klas

Długość kielicha	Szerokość kielicha	Długość płatka	Szerokość płatka	Typ kwiatu
5,1	3,5	1,4	0,2	Iris-setosa
4,9	3,1	0,4	0,2	Iris-setosa
6,3	3,3	4,7	1,6	Iris-versicolor
7,7	3,8	6,7	2,2	Iris-virginica

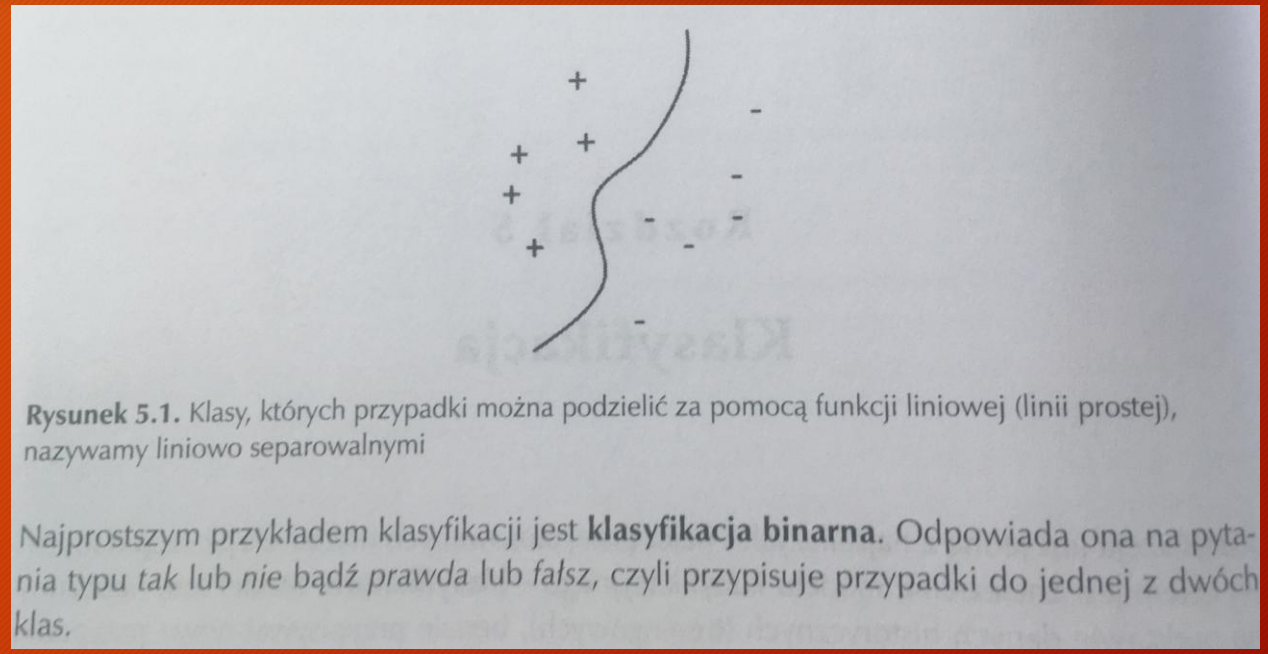
Zmienne wejściowe klasyfikatorów mogą być numeryczne lub kategoryczne, zmienna wyjściowa zawsze jest zmienną kategoryczną.

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Zmienne kategoryczne to zmienne, które przyjmują skończoną liczbę arbitralnie ustalonych stanów (kategorii). Stany nie są w żaden sposób uporządkowane a ich porównywanie nie jest możliwe.

Klasyfikator

- Klasyfikacja jest metodą uczenia nadzorowanego – zbiór danych treningowych zawiera przykładowe wyniki szukanej funkcji. Znaleziona na ich podstawie funkcja f powinna separować od siebie przypadki należące do poszczególnych klas.



Rysunek 5.1. Klasy, których przypadki można podzielić za pomocą funkcji liniowej (linii prostej), nazywamy liniowo separowalnymi

Najprostszym przykładem klasyfikacji jest **klasyfikacja binarna**. Odpowiada ona na pytania typu *tak* lub *nie* bądź *prawda* lub *fałsz*, czyli przypisuje przypadki do jednej z dwóch klas.

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Klasyfikacja - zastosowanie

- Bankowość- czy dane podanie o udzielenie kredytu jest obarczone dużym czy małym ryzykiem, czy dana transakcja kartą jest oszustwem,
- Edukacja - umieszczenie studenta w odpowiedniej grupie z uwzględnieniem jego potrzeb,
- Medycyna - diagnozowanie, czy występuje dana choroba,
- Telekomunikacja - czy dany klient zrezygnuje z usług firmy telekomunikacyjnej
- Informatyka - czy wiadomość pocztowa jest spamem…….

Dokładność klasyfikacji

Jak liczyć dokładność klasyfikatora:

- budujemy klasyfikator w oparciu o zbiór treningowy - dla każdego rekordu określona jest jego klasa,
- używając zbudowanego klasyfikatora, dla każdego wiersza ze zbioru testowego wyznaczamy klasę oraz porównujemy z rzeczywistą klasą,
- dokładność określamy jako stosunek liczby poprawnie sklasyfikowanych wierszy ze zbioru testowego do mocy tego zbioru.

Rodzaje klasyfikacji

- Klasyfikacja probabilistyczna
- Klasyfikacja metodą k najbliższych sąsiadów
- Klasyfikacja metodą drzew decyzyjnych
- Klasyfikacja z użyciem wektorów podparcia
- Wiele innych trochę mniej popularnych typu regułowe, asocjacyjne i inne.

Klasyfikacja probabilistyczna

- Modelowane jest prawdopodobieństwo zależności między zmiennymi wejściowymi a wyjściową. Powszechnie stosowane np. w prognozach pogody (gdy w przeszłości w takich samych warunkach deszcz padał 6 razy na 10, to teraz prawd. wynosi 60%).
- Zaletami wydajność i niewrażliwość na przeuczenie
- Podstawą konstrukcji klasyfikatorów bayesowskich (w tym naiwny klasyfikator Bayesa) jest twierdzenie matematyka i duchownego brytyjskiego Thomasa Bayesa (rok 1763).

Klasyfikator probabilistyczny

- Prawdopodobieństwo wszystkich możliwych stanów wynosi 100% (czyli 1). Np. dla 2 stanów (spam lub ham) jeśli P(spam)=20%, to P(ham)=80%. Wartość tego prawdopodobieństwa zależy od częstości występowania klas i jest nazywane prawdopodobieństwem **a priori** (bezwzględnym).
- **Prawdopodobieństwo** warunkowe zależy od kontekstu. Przykładowo jeżeli słowo okazja występuje w 5% wszystkich wiadomości, ale częściej w niechcianych wiadomościach, to jego obecność to sygnał, że wiadomość może być spamem.

Twierdzenie Bayesa

Prawdopodobieństwo warunkowe $P(\text{spam} \mid \text{okazja})$, czyli prawdopodobieństwo zdarzenia, że wiadomość jest spamem, ponieważ zawiera słowo *okazja*, obliczymy na podstawie **twierdzenie Bayesa**:

$$P(\text{spam} \mid \text{okazja}) = \frac{P(\text{okazja, spam}) \cdot P(\text{spam})}{P(\text{okazja})}$$

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Pozwala obliczyć prawdopodobieństwo **warunkowe (a posteriori)**. Potrzebujemy jedynie informacji jak często słowo okazja wystąpiło w niechcianych wiadomościach. Przyjmijmy wartości jak na kolejnym slajdzie.

Twierdzenie Bayesa – c.d.

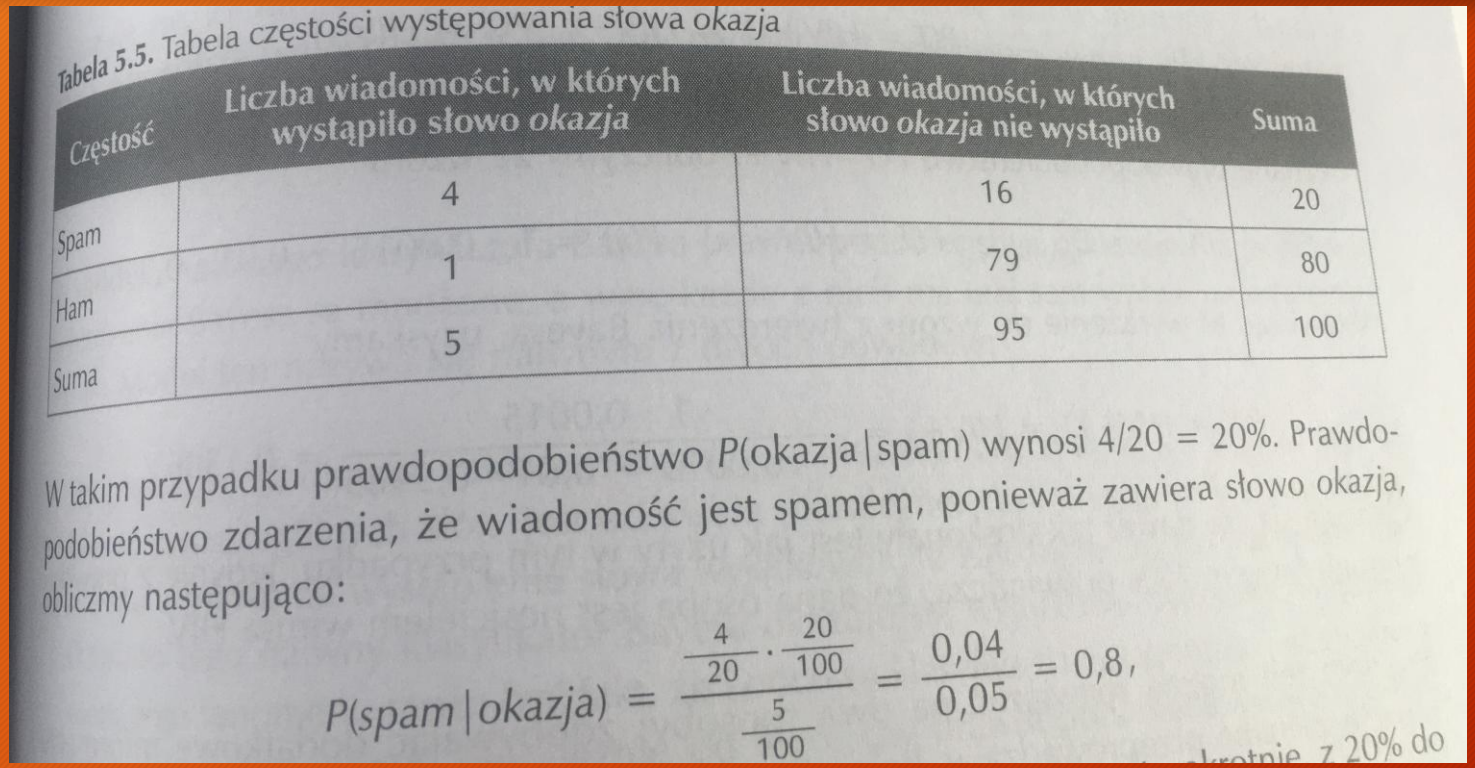


Tabela 5.5. Tabela częstości występowania słowa okazja

Częstość	Liczba wiadomości, w których wystąpiło słowo *okazja*	Liczba wiadomości, w których słowo *okazja* nie wystąpiło	Suma
Spam	4	16	20
Ham	1	79	80
Suma	5	95	100

W takim przypadku prawdopodobieństwo $P(\text{okazja}|\text{spam})$ wynosi 4/20 = 20%. Prawdopodobieństwo zdarzenia, że wiadomość jest spamem, ponieważ zawiera słowo okazja, obliczmy następująco:

$$P(spam|okazja) = \frac{\frac{4}{20}\cdot\frac{20}{100}}{\frac{5}{100}} = \frac{0{,}04}{0{,}05} = 0{,}8,$$

...krotnie z 20% do

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Prawdopodobieństwo, że mamy spam, wzrosło z 20% do 80%, gdy wiadomość zawiera słowo *okazja*.

Naiwny klasyfikator Bayesa

Dokonując kolejnych obserwacji, w tym przypadku polegających na obliczeniu częstości występowania innych słów w spamie, możemy modyfikować prawdopodobieństwo warunkowe, uwzględniając we wzorze z twierdzenia Bayesa ich wyniki. Otrzymamy w ten sposób **naiwny klasyfikator Bayesa**:

$$P(D_i|V_1, \ldots, V_n) = \frac{P(V_1, \ldots, V_n)|D_i}{P(V_1, \ldots, V_n)}$$

Ponieważ interesuje nas wynik porównania prawdopodobieństw, a nie ich wartości, mianownik można uznać za stałą i wyeliminować go z modelu.

Klasyfikacja w tym modelu polega na wyznaczeniu prawdopodobieństw a *posteriori* dla każdej kombinacji zbioru zmiennych wejściowych ze zmienną wyjściową i przypisaniu przypadku do tej klasy, dla której wartość prawdopodobieństwa będzie największa (**zasada maksymalizacji prawdopodobieństwa *a posteriori***).

A dokładniej:

$$P(D_i|V_1, \ldots, V_n) = \frac{P(V_1, \ldots, V_n|D_i)P(D_i)}{P(V_1, \ldots, V_n)}$$

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Przykład działania

- Zaklasyfikowanie pacjenta jako zarażonego lub nie wirusem HIV.
- Załóżmy, że mamy test, którego skuteczność w wykrywaniu wirusa wynosi 100%, jednak błędnie klasyfikuje jako nosicieli 1% zdrowych osób. Przyjmijmy, że 0,15% populacji jest nosicielami.
- Jeśli wyniki testu są negatywne, to mamy pewność że osoba ta jest zdrowa. Jeśli jednak test wypadł pozytywnie, to prawdopodobieństwo, że dana osoba jest nosicielem wirusa wynosi:
- $$P(X = HIV + | T = HIV +) = \frac{P(T = HIV + | X = HIV +) P(X = HIV +)}{P(T = HIV +)}$$

Przykład – c.d.

- Mamy $P(T = HIV+) = \sum_{x\{HIV-,HIV+\}} P(T = HIV+|x)$=

  = 1· 0,0015+0,01 ·0,9985, a stąd otrzymujemy

  $P(X = HIV+|T = HIV+) = \frac{1 \cdot 0{,}0015}{1 \cdot 0{,}0015+0{,}01 \cdot 0{,}9985} = 0{,}1306$

Ojej, nawet tak dobry test poświadcza jedynie z prawdopodobieństwem 13%, że dana osoba jest nosicielem wirusa HIV. Co można zrobić? Zdobyć dodatkowe informacje o pacjencie lub przeprowadzić kolejny test. Oba wymagają kolejnych prawdopodobieństw warunkowych.

Przykład – c.d.

- Znamy wiek pacjenta, a aktualne statystyki mówią, że prawdopodobieństwo, że 25-letnia osoba jest nosicielem wynosi 0,8%. Jeżeli ponadto test jest niezależny od wieku, to otrzymujemy:
- $P(X = HIV + |A) = \frac{1 \cdot 0{,}008}{1 \cdot 0{,}008 + 0{,}01 \cdot 0{,}992}$=0,4462
- Gdzie A oznacza T=HIV+, W=25
- Dodatkowa wiedza a posteriori pozwoliła ponad 3-krotnie zwiększyć uzyskanie prawidłowego wyniku. Tak duży wpływ wiedzy a posteriori i wynikającego z niej prawdopodobieństwa warunkowego jest podstawą skuteczności klasyfikatorów Bayesa.

Przykład – c.d.

- Drugi sposób pewności predykcji polega na powtórzeniu testu w taki sposób, aby wyniki obu testów były od siebie niezależne. Załóżmy, że drugi test jest mniej trafny i błędnie klasyfikuje jako nosicieli 5% zdrowych osób. Prawdopodobieństwo, że osoba u której oba testy wypadły pozytywnie jest nosicielem, wynosi:
- $P(X = HIV + | T1 = HIV+, T2 = HIV +) = \frac{P(X+)P(T_1,T_2|X+)}{P(T_1,T_2)} = \frac{P(X+)P(T_1|X+)P(T_2|X+)}{P(T_1,T_2|X+)P(X+)+P(T_1,T_2|X-)P(X-)} = \frac{1 \cdot 1 \cdot 0{,}008}{1 \cdot 1 \cdot 0{,}008 + 0{,}01 \cdot 0{,}992 \cdot 0{,}05} = 0{,}9416$
- Iloczyn prawdopodobieństw a posteriori zdominował silną wiedzę a priori o tym, że zaledwie 0,8% osób w tym wieku jest nosicielem.

Dlaczego model nazywamy naiwnym?

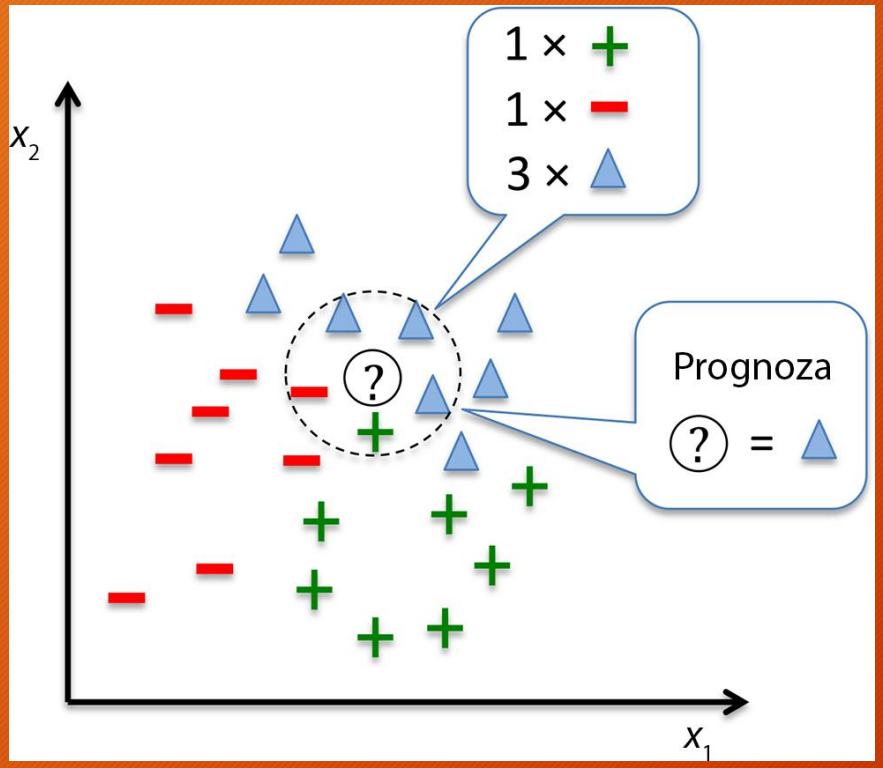
- 1. Zakłada warunkową niezależność zmiennych (np. wystąpienie słowa *okazja* jest niezależne od wystąpienia słowa *wyjątkowa*), co jest rzadkie. Mimo tego klasyfikator jest dokładny – zauważmy, że końcowa klasyfikacja (spam\ham) jest taka sama, niezależnie od tego czy prawdopodobieństwo warunkowe wynosiło 51% czy 99%.
- 2. Założenie, że każda zmienna ma taki sam wpływ na końcowy wynik. Ten problem się rozwiązuje wprowadzając wagi prawdopodobieństw.
- Wad tych są pozbawione sieci Bayesa, które określają zależności przyczynowe między wszystkimi zmiennymi zbioru treningowego.

Klasyfikator k-najbliższych sąsiadów KNN

- K-nearest neighbor classifier KNN to algorytm uczenia nadzorowanego, jest **leniwym klasyfikatorem** (ang. lazy learner) bo nie uczy się on funkcji dyskryminacyjnej na podstawie danych uczących, ale stara się zapamiętać cały zbiór próbek.
- 3 etapy:

  1. wybierz jakąś wartość parametru k i metrykę odległości
  2. znajdź k najbliższych sąsiadów próbki, którą chcesz sklasyfikować
  3. przydziel etykietę klasy poprzez głosowanie większościowe.

KNN - przykład



KNN wyszukuje zgodnie z wybraną metryką w zestawie danych uczących k próbek znajdujących się najbliżej klasyfikowanego punktu lub wykazujących najbliższe prawdopodobieństwo do niego. Etykieta klasy tej próbki zostaje określona poprzez większościowe głosowanie przeprowadzone pomiędzy k najbliższych sąsiadów.

Źródło: Raschka, Mirjalili: Python. Uczenie Maszynowe, Helion 2019

KNN – zalety i wady

- Zalety: natychmiastowe adoptowanie się klasyfikatora w trakcie pobierania nowych danych uczących
- Wady: liniowy wzrost złożoności obliczeniowej wraz z liczbą próbek, nie możemy odrzucać żadnych danych uczących bo nie występuje tak naprawdę proces uczenia (!), może być więc problem pojemnością nośników.

KNN – dalsze informacje

- W przypadku remisu w głosowaniu algorytm KNN zaimplementowany w scikit-learn faworyzuje sąsiadów bliżej próbki.
- Ważny jest właściwy dobór parametru k w zależności od danych.
- Ważny jest również dobór metryki. Gdy euklidesowa, to często stosuje się standaryzację danych (każda cecha będzie wtedy odpowiednio wyskalowana do odległości).
- Popularna odległość Minkowskiego (uogólnienie metryki Euklidesa):

$$d\left(x^{(i)}, x^{(j)}\right) = \sqrt[p]{\sum_k |x_k^{(i)} - x_k^{(j)}|^p}$$

Gdy p=2, to metryka E., gdy p=1, to metryka Manhattan (miejska). Interfejs scikit-learn zawiera dużo metryk:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.DistanceMetric.html

Techniki ewaluacji modelu

- Określamy następujące 4 wartości:
- TP (true positive) - liczba rekordów prawidłowo zaklasyfikowanych jako pozytywne,
- FP (false positive) - liczba rekordów błędnie zaklasyfikowanych jako pozytywne,
- TN (true negative) - liczba rekordów prawidłowo zaklasyfikowanych jako negatywne,
- FN (false negative) - liczba rekordów błędnie zaklasyfikowanych jako negatywne.

- Macierz błędu klasyfikacji (ang. confusion matrix)
- Wartości na przekątnej dotyczą dobrej klasyfikacji.

		Predicted class: *P*	Predicted class: *N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

KNN – przykłady (Python)

- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
>>> print(neigh.predict([[1.1]]))
>>> print(neigh.predict_proba([[0.9]]))
```

- **Kolejny przykład – ekran (irysy)**

KNN – przykład (język R)

- Wykorzystamy plik Prostate_Cancer.csv ze strony https://www.kaggle.com/
- A sam przykład: https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/

  setwd("C:/Users/Tomek/Desktop/ważne/wykladIO/wyklad2")
- prc <- read.csv("Prostate_Cancer.csv",stringsAsFactors = FALSE)

  By za szybko nie faktoryzować łańcucha znaków, przykład:

  Freud: Czasami cygaro jest tylko cygarem i niczym więcej…

  d <- data.frame(label = rep("tbd", 5))

  d$label[[2]] <- "north "

  d <- data.frame(label = rep("tbd", 5), stringsAsFactors = FALSE)
- str(prc) #czy dane mają odpowiednią strukturę
- prc <- prc[-1] #usuwamy pierwszą kolumnę
- table(prc$diagnosis_result)
- prc$diagnosis <- factor(prc$diagnosis_result, levels = c("B", "M"), labels = c("Benign", "Malignant"))

KNN – przykład (język R – c.d)

- round(prop.table(table(prc$diagnosis)) * 100, digits = 1) #1 cyfra po przecinku
- normalize <- function(x) { return ((x - min(x)) / (max(x) - min(x))) }
- prc_n <- as.data.frame(lapply(prc[2:9], normalize))
- summary(prc_n$radius)
- prc_train <- prc_n[1:65,] # wzięte wszystkie wiersze i kolumny
- prc_test <- prc_n[66:100,]
- prc_train_labels <- prc[1:65, 1] # y jest wartością pierwszej kolumny
- prc_test_labels <- prc[66:100, 1]
- install.packages("class") library(class)
- prc_test_pred <- knn(train = prc_train, test = prc_test, cl = prc_train_labels, k=10)
- install.packages("gmodels") library(gmodels)
- CrossTable(x = prc_test_labels, y = prc_test_pred, prop.chisq=FALSE)

Drzewa decyzyjne

- Opracowane niezależnie przez Johna Quinlana oraz Breimana, Friedmana, Olshena i Stona w latach 80tych XX wieku.
- Obecnie często używane w modelach będących kombinacjami wielu klasyfikatorów.
- Przykład: kolacja w restauracji, w obcym mieście. Czy miejscowi tam chodzą? Czy kelnerka miła? Czy liczba stron menu nie jest za duża? Jaka kuchnia? Zostać w niej, czy może poszukać zupełnie innej? Dużo pytań do rozważenia.
- **Drzewa decyzyjne konstruowane są przez rekurencyjne dzielenie danych treningowych na podzbiory na podstawie tego, w jaki sposób ten podział wpłynie na stany zmiennej wyjściowej.**

Drzewa decyzyjne - przykład

W naszym przykładzie zmienną wyjściową jest ocena restauracji, a wejściowymi: liczba klientów, obsługa, liczba stron menu i kuchnia (tab. 5.2).

Tabela 5.2. Zestawienie opinii na temat 200 restauracji z podziałem na atrybuty

Opinia	Czy popularna		Obsługa			Liczba stron menu		Kuchnia	
	tak	nie	doskonała	dobra	zła	dużo	mało	francuska	lokalna
Dobra	60	20	30	50	20	20	80	50	60
Zła	10	110	20	40	40	60	40	40	50

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Budowanie DD zaczynamy od znalezienia zmiennej, której podział wyznaczy korzeń drzewa. Można użyć poziomu entropii, prawd. Bayesa, zysku informacyjnego czy współczynnika Giniego.

Entropia

- Jeden bit niesie ze sobą co najwyżej jedną informację, dwa bity pozwalają przekazać 4 informacje itd. Ogólnie:

  Informacja w bitach = log_2(liczba możliwych stanów).

Entropia (ang. entropy, Shannon 1948) to miara niepewności (co będzie w kolejnym zdarzeniu) czyli średnia ilość informacji przypadająca na pojedynczą wiadomość ze źródła informacji. Liczymy ją ze wzoru: $\mathrm{E}(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$ dla zmiennej losowej X o wartościach $\{x_1, \dots x_n\}$.

Rozważmy 2 gry, w których mamy liczby od 1 do 10. W jednej mamy odgadnąć czy wylosowana liczba jest parzysta, a w drugiej tą liczbę. W pierwszej grze entropia wynosi 1,0 (duża niepewność w kolejnym ruchu) a w drugiej 0,47 ($-0{,}1 \cdot log_2 0{,}1 - 0{,}9 \cdot log_2 0{,}9$). Gdzie jest ukryte więcej informacji? (w wygranej czy przegranej?)

Entropia – przykłady w R

- Przykłady – patrz Rstudio – kiedy entropia 0, kiedy 1 itd.
- Zmienne, które nie niosą ze sobą żadnej informacji, należy usunąć ze zbioru danych treningowych.
- Często okazuje się, że zmienne, dla których entropia jest bardzo wysoka jedynie komplikują model (usunąć je po konsultacji z ekspertem).
- Ocena poziomu wiedzy wymaga znajomości jej kontekstu. Gdyby wygrana była 10 złotych to informacja o wygranej byłaby inna niż gdyby wygrana była 10000000 złotych….

Zysk informacyjny

- To ilość informacji, jaką zdobędziemy o stanie zmiennej B, jeśli poznamy stan zmiennej A (jest symetryczny). Czyli jest to ilość entropii, jaka została usunięta z jednej zmiennej dzięki poznaniu wartości drugiej.
- Załóżmy, że (w ustalonym węźle) zbiór danych *S* składa się z *s* wierszy. Załóżmy, że zbiór klas (wartości zmiennej celu) składa się z *m* róznych elementów $C_1, \dots, C_m$. Niech s_i oznacza liczbę wierszy z klasy C_i. Ilość informacji niezbędna do zaklasyfikowania rekordu z danego zbioru:

$$I(s_1, \dots, s_m) = -\sum_{i=1}^{m} p_i \cdot log p_i, \text{ gdzie } p_i = \frac{s_i}{s}$$

Zysk informacyjny – c.d.

- Jeżeli *A* jest atrybutem, który przyjmuje *v* możliwych wartości $a_1, \dots, a_v$, to atrybut *A* dzieli zbiór *S* na *v* podzbiorów $S_1, \dots, S_v$.

Niech s_{ij} oznacza liczbę wierszy z klasy C_i należących do podzbioru S_j (czyli wartość atrybutu *A* jest w tym podzbiorze równa a_j). Entropia:

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \dots + s_{mj}}{s} \cdot I(s_{1j}, \dots, s_{mj}),$$ gdzie:

$$I(s_{1j}, \dots, s_{mj}) = -\sum_{i=1}^{m} p_{ij.} \cdot log p_{ij},$$ oraz $p_{ij} = \frac{s_{ij}}{|S_j|}$ Wtedy:

Zysk informacji dla atrybutu *A*:

Gain(A) = $I(s_1, \dots, s_m)$ - *E(A)*

Zysk informacyjny - przykład

wiek	dochod	stud	zdolKred	czyKupi
<=30	duży	nie	umiark	nie
<=30	duży	nie	doskonała	nie
31..40	duży	nie	umiark	tak
>40	sredni	nie	umiark	tak
>40	mały	tak	umiark	tak
>40	mały	tak	doskonała	nie
31..40	mały	tak	doskonała	tak
<=30	sredni	nie	umiark	nie
<=30	mały	tak	umiark	tak
>40	sredni	tak	umiark	tak
<=30	sredni	tak	doskonała	tak
31..40	sredni	nie	doskonała	tak
31..40	duży	tak	umiark	tak
>40	sredni	nie	doskonała	nie

Źródło: Han, Jiawei/ Kamber, Micheline/ Pei, JianPublisher: Data Mining, Elsevier Science 2011

Mamy 4 atrybuty, niech atrybut decyzyjny przyjmuje 2 wartości: $C_1 =' tak'$ lub $C_2 =' nie'$.

Przykład – c.d.

- $I(s_1, s_2) = -\frac{9}{14} log \frac{9}{14} - \frac{5}{14} log \frac{5}{14} = 0{,}94$
- Teraz należy przeprowadzić obliczenia entropii podziału zbioru S na partycje wg wszystkich atrybutów deskryptorów. Najpierw dla wieku.
- Dla wartosci '<=30': $s_{11} = 2$, $s_{21} = 3$,

$$I(s_{11}, s_{21}) = -\frac{2}{5} log \frac{2}{5} - \frac{3}{5} log \frac{3}{5} = 0{,}971,$$

- Dla wartosci '31..40': $s_{12} = 4$, $s_{22} = 0, I(s_{12}, s_{22}) = 0$,
- Dla wartosci '>=40': $s_{13} = 3$, $s_{23} = 2$,
- $I(s_{13}, s_{23}) = -\frac{2}{5} log \frac{2}{5} - \frac{3}{5} log \frac{3}{5} = 0{,}971.$

Przykład – c.d.

- Entropia atrybutu „wiek”wynosi:

$$E(\text{„wiek”}) = \frac{5}{14} \cdot I(s_{11}, s_{21}) + \frac{4}{14} \cdot I(s_{12}, s_{22}) + \frac{5}{14} \cdot I(s_{13}, s_{23}) = 0{,}694.$$

- Zysk informacyjny wynikający z podziału zbioru S według atrybutu wiek wynosi:

$$Gain(\text{„wiek”}) = I(s_1, s_2) - E(\text{„wiek”}) = 0{,}246.$$

- Analogicznie obliczamy zysk informacyjny dla pozostałych atrybutów:

  Gain(„dochód”) = 0,029

  Gain(„student”) = 0,151

  Gain(„zdolKred”) = 0,048

- A zatem tworzymy drzewo decyzyjne zaczynając od wierzchołka „wiek”.

Konstrukcja drzewa – patrz tablica.

Współczynnik Giniego

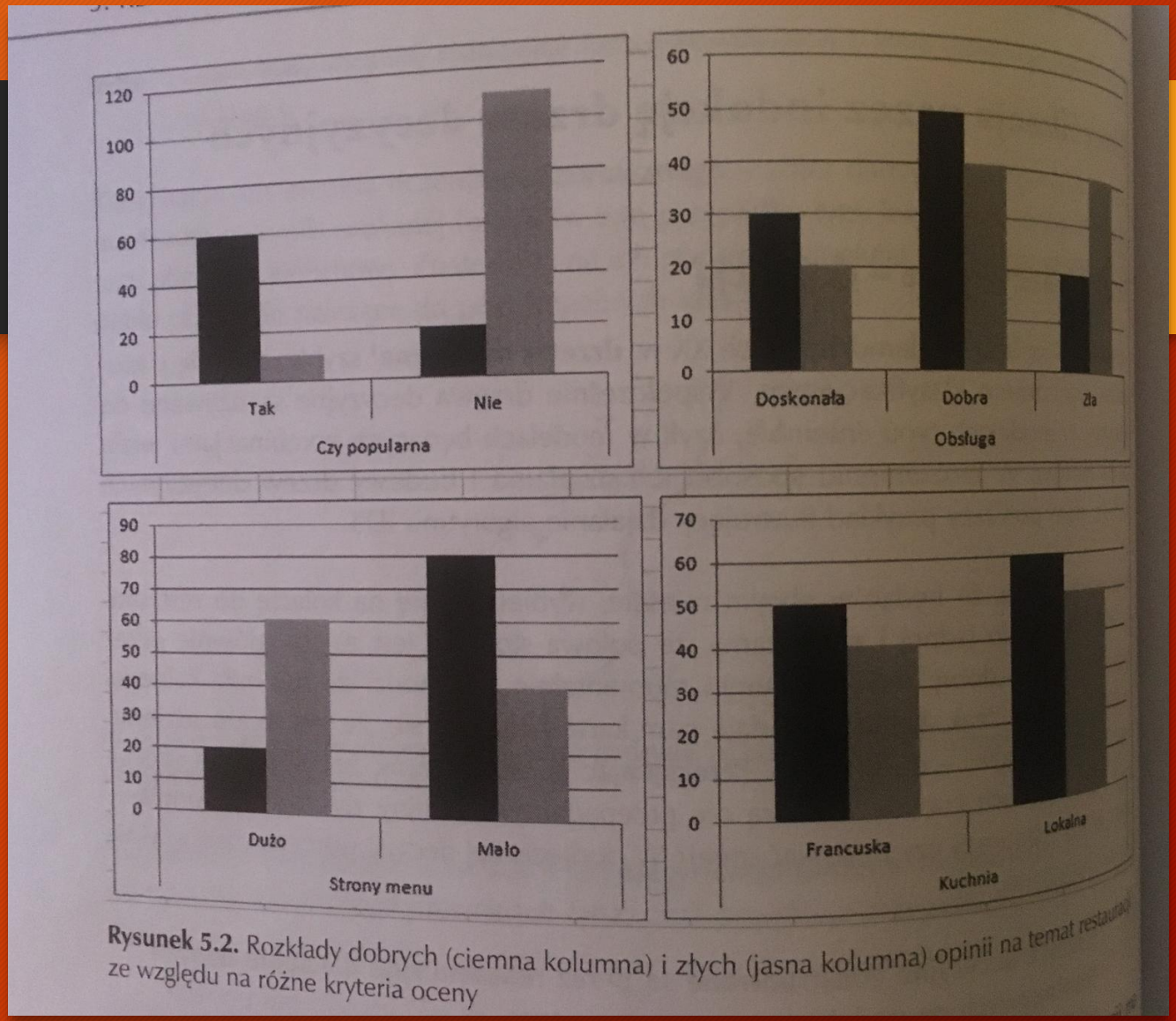
- **Współczynnik Giniego (włoski statystyk 1884-1965)**, **wskaźnik Giniego**, indeks Giniego – stosowana w statystyce miara koncentracji (nierównomierności) rozkładu zmiennej losowej.

  Własności:
- współczynnik G. przyjmuje wartości z przedziału [0; 1], często jednak wyraża się go w procentach,
- wartość zerowa współczynnika wskazuje na pełną równomierność rozkładu,
- wzrost wartości współczynnika oznacza wzrost nierówności rozkładu,
- współczynnik G. przyjąłby wartość 1, gdyby tylko jedna obserwacja uzyskała dodatnią wartość zmiennej (na przykład tylko jedno gospodarstwo domowe posiadało dochody) przy nieskończonej liczbie obserwacji.

W ekonomii to częsta miara wskaźnika nierówności społecznej.

Powrót do przykładu z restauracją i własności drzew decyzyjnych

- Jaka cecha ma największy wpływ na ocenę restauracji?
- Wybierając tę zmienną do podziału przypadków, minimalizujemy ilość informacji potrzebnej do zakwalifikowania przypadków do którejś z otrzymanych grup.
- Wiemy więc co w korzeniu drzewa. Dzielimy potem przypadki na 2 podzbiory (tak lub nie), a każdy z nich znowu dzielimy itd.
- Podział może być binarny (obsługa doskonała lub nie) albo zupełny (obsługa na 3 podzbiory).



Rysunek 5.2. Rozkłady dobrych (ciemna kolumna) i złych (jasna kolumna) opinii na temat restauracji ze względu na różne kryteria oceny

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Drzewa decyzyjne – c.d.

- Kłopot ze zmiennymi numerycznymi: wartości ich są dynamicznie dzielone na równe przedziały, oddzielnie dla każdego węzła DD. Potem sąsiednie przedziały są łączone, o ile takie połączenie daje możliwość lepszego podzielenia przypadków. Powtarzane to jest gdy mamy przedziały najlepiej nadające się do podzielenia przypadków w danym węźle DD.
- Problem ze zmiennymi kategorycznymi: są faworyzowane przez algorytmy, w których do podziału wykorzystano entropię lub zysk informacyjny. Dostajemy dużą liczbę małych podzbiorów – przeuczenie modelu. Rozwiązanie: uogólniamy zmienne o dużej liczbie stanów lub stosujemy współczynnik G. do oceny podziałów.

Drzewa decyzyjne – c.d.

- Aby zmniejszyć ryzyko nadmiernego dopasowania DD do danych treningowych to buduje się je w 2 etapach: konstrukcja (indukcja) na podstawie danych treningowych, potem etap przycinania.
- Metoda przycinania wstępnego (ang. prepruning) : podczas konstrukcji, lepiej skalowalna, szybsza i łatwiejsza ale dd może być nieoptymalne (gdy konstrukcja zakończyła się za szybko – pozornie bezwartościowe podziały).
- Metoda przycinania końcowego (ang. postpruning) po pełnej konstrukcji dd usuwamy węzły, które mają niewielkie znaczenie dla klasyfikacji. Lepsze rezultaty ale kosztowna.
- Do oceny DD używamy danych walidacyjnych, oceniamy każdy podział, co może prowadzić do nadmiernego dopasowania przyciętych drzew do danych walidacyjnych.

Algorytm ML wykorzystujący DD

- jeżeli zmienna wejściowa jest numeryczna, zostaje ona poddana dyskretyzacji;
- ze zmiennych wejściowych wybierana jest ta, według której podział zminimalizuje ilość informacji potrzebnej do zaklasyfikowania przypadków względem zmiennej wyjściowej do otrzymanych w ten sposób podzbiorów;
- podzbiory są tworzone dla wszystkich stanów wybranej zmiennej lub metodą binarną, tak aby każdy podzbiór był maksymalnie jednorodny względem zmiennej wyjściowej i jednocześnie maksymalnie odrębny względem zmiennej wyjściowej od pozostałych podzbiorów;
- budowane są poddrzewa, których danymi treningowymi są otrzymane w poprzednim kroku podzbiory; jeżeli stosowana jest metoda wstępnego przycinania, poddrzewa, których podzbiory liczą zbyt mało przypadków lub dla których rozkład klas jest za mało wyraźny, są usuwane.

Źródło: M. Szeliga: Praktyczne uczenie maszynowe, PWN 2019

Algorytm ML dla DD – c.d.

- Powyższe kroki są powtarzane dla każdego poddrzewa, aż do:

1. wyczerpania przypadków w zbiorze danych treningowych,
2. przypisania wszystkich przypadków do tej samej klasy,
3. przekroczenia zadanego czasu lub zadanej, maksymalnej wielkości drzew.

Gdy stosujemy metodę przycinania końcowego, to najpierw usuwane są poddrzewa najdalsze od korzenia (czasowo), potem porównujemy jakość klasyfikatora z modelem, w którym to poddrzewo jeszcze było. Możliwe przywrócenie poddrzewa lub zastąpienie jego pojedynczym zbiorem danych treningowych.

Drzewo decyzyjne – przykład i konsekwencje

- W języku R, funkcja *ctree* z pakietu *party*. Patrz ekran.
- W języku Python, funkcja DecisionTreeClassifier, patrz ekran
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

- Bardzo proste drzewo, ale widać jego ograniczenia: algorytm nie jest w stanie wykryć zależności między kombinacjami zmiennych wyjściowych a zmienną wyjściową np. typ kwiatu może zależeć od sumy długości i szerokości płatków itp. Skuteczność jest więc nienajlepsza. Poza tym DD mogą być optymalne jedynie lokalnie oraz nie zawierać wszystkich zmiennych wejściowych i tych o znacznej sile predykcyjnej (silnie skorelowanych z innymi – brak długości i szerokości kielicha w przykładzie). Często występuje też przeuczenie modelu.

- Co stosuje się więc by poprawić skuteczność drzew decyzyjnych?

Kombinacje drzew decyzyjnych

- Buduje się modele zawierające wiele niezależnych klasyfikatorów (modele złożone). Ich wynikowa predykcja jest rezultatem głosowania większościowego. Są one bardziej odporne na błędy w danych, co wynika ze sposobów ich konstruowania. 2 podstawowe sposoby:

  a) agregacja bootstrapowa,

  b) wzmocnione, losowe wybieranie ze zwracaniem.

Metody konstruowania modeli złożonych mogą polegać na modyfikowaniu danych treningowych, listy zmiennych i algorytmu eksploracji danych (inne sparametryzowanie przy tych samych danych treningowych).

Co ważne: nie tylko drzewa decyzyjne mogą wchodzić w skład kombinacji klasyfikatorów, te same metody ich konstrukcji dotyczą też modeli, w których zastosowano inne algorytmy.

Agregacja bootstrapowa

- To metoda modyfikowania danych treningowych, która polega na wielokrotnym losowaniu ze zwracaniem przypadków – po n losowaniach dostaniemy n-elementowy zbiór danych treningowych z ewentualnymi powtórzeniami. Wynikowa predykcja zwracana przez model złożony jest ustalana metodą głosowania większościowego poszczególnych modeli bazowych. Zwiększa się stabilność całego modelu a zmniejsza ryzyko przeuczenia modelu.

- Proces uczenia modelu lasu dd przebiega następująco:

- Z danych treningowych jest losowana próba bootstrap, czyli wybiera się z powtórzeniami *k* przypadków.
- Próba ta jest używana do uczenia pojedynczego drzewa decyzyjnego. Sposób konstruowania drzewa nie różni się od standardowego, z jednym istotnym wyjątkiem – dla każdego tworzonego drzewa decyzyjnego jest wybieranych losowo *x* atrybutów z listy wszystkich dostępnych atrybutów. Oznacza to, że poszczególne węzły utworzonego w ten sposób drzewa decyzyjnego będą analizowały zależności między losowym podzbiorem atrybutów wejściowych a atrybutem wyjściowym.
- Zbudowane drzewo jest oceniane za pomocą przypadków niewybranych do próby bootstrap. Mierzone są między innymi: dokładność predykcji (liczba błędnych klasyfikacji) i wpływ poszczególnych atrybutów wejściowych na dokładność predykcji.
- Punkty od 1 do 3 powtarzane są *n* razy.
- Odpowiedzi na zapytania predykcyjne (wyniki zwracane przez model) są ustalane przez głosowanie przy użyciu wchodzących w jego skład drzew decyzyjnych. Na przykład jeżeli 67% drzew sklasyfikuje daną transakcję jako oszustwo, wynikiem będzie jej ocena jako oszustwo [5].

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Drzewa decyzyjne ze wzmacnianiem (ang. Boosted decision trees)

- Kolejne zbiory treningowe nie są wybierane losowo, ale w sposób, który faworyzuje przypadki błędnie sklasyfikowane przez wcześniej utworzone modele bazowe. Czyli prawdopodobieństwo wybrania do nich przypadków nie jest takie samo, jak w przypadku lasu dd. Powstały w ten sposób model, złożony z wielu słabszych klasyfikatorów, jest co najmniej tak dobry, jak najlepszy z modeli bazowych, a często znacznie od niego lepszy.

- W tym celu stosuje się metodę wzmocnionego wybierania losowego ze zwracaniem (boosting). Schemat jej działania:

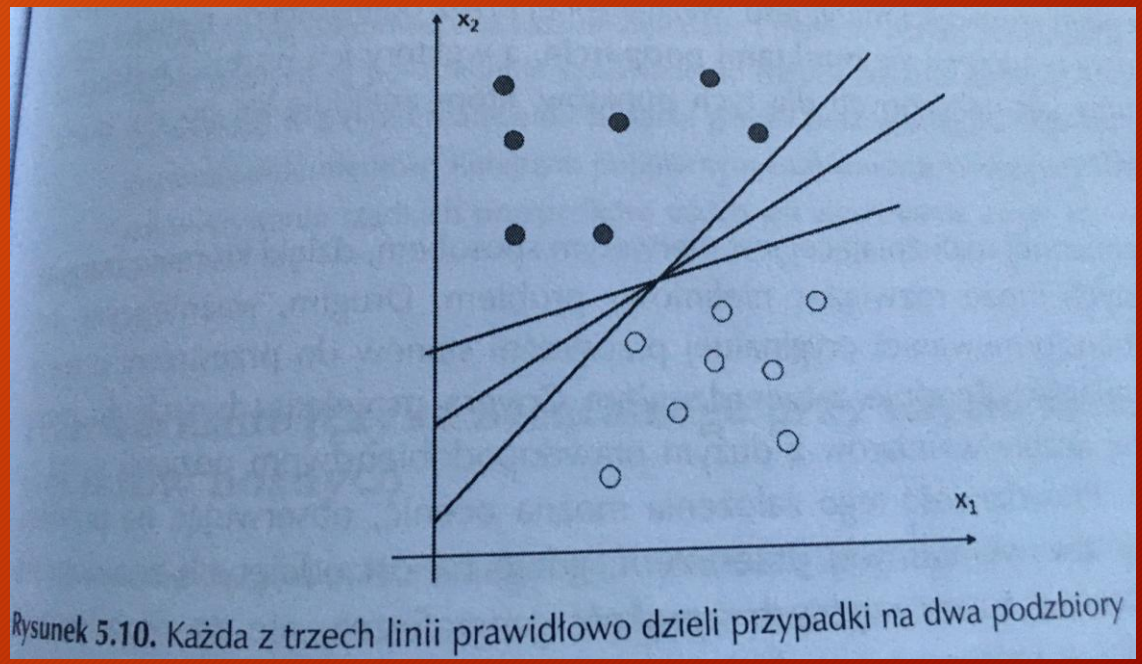
Drzewa decyzyjne ze wzmacnianiem – c.d.

- Przypisanie przypadkom wag. Początkowo wagi wszystkich przypadków w zbiorze danych treningowych D są takie same.
- Wybranie próby D_1 metodą losowania ze zwracaniem. Przy wybieraniu próby są uwzględniane wagi, ale jako że dla pierwszego zbioru treningowego są one takie same dla wszystkich przypadków, losowanie odbywa się z rozkładu jednostajnego.
- Konstruowanie modelu C_1 oraz ocena jego dokładności przy użyciu danych treningowych D.
- Modyfikowanie wag rekordów w zbiorze danych treningowych D na podstawie dokładności modelu C_1 – wagi błędnie sklasyfikowanych przypadków są mnożone przez ustalony dla iteracji współczynnik, a wagi przypadków poprawnie sklasyfikowanych są zmniejszane, tak aby suma wszystkich wag wyniosła 1.
- Wybieranie próby D_2 metodą losowania ze zwracaniem – prawdopodobieństwa wylosowania do niej poszczególnych przypadków są już różne, ponieważ uwzględniane są wagi przypadków.
- Konstruowanie modelu C_2 oraz ocena jego dokładności przy użyciu danych treningowych D. Na podstawie dokładności modelu C_2 są modyfikowane (w taki sam sposób) wagi rekordów w zbiorze danych treningowych D.
- Kroki te powtarza się aż do utworzenia k modeli bazowych.

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

Klasyfikator SVM (Support Vector Machine)- skrótowo

- Liniowy klasyfikator binarny. Polega na podzieleniu zbioru treningowego na 2 podzbiory i oddzieleniu ich od siebie możliwie jak najszerszym marginesem.



Rysunek 5.10. Każda z trzech linii prawidłowo dzieli przypadki na dwa podzbiory

Jak wybrać najlepszą linię? Możemy je przesuwać w bok aż do pierwszych punktów z obu klas wyznaczając marginesy. Zadaniem modelu SVM w dwuwymiarowej przestrzeni jest znalezienie linii, która maksymalizuje margines.

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017

SVM – c. d.

- W praktyce mamy do czynienia ze znajdowaniem hiperpłaszczyzny (l. wymiarów o 1 mniejszej od przestrzeni, w której się znajduje), która w optymalny sposób oddzieli należące do różnych klas n-wymiarowe wektory.
- Co gdy nie da się danych podzielić za pomocą liniowego klasyfikatora? W modelach SVM stosuje się: wprowadzenie zmiennej rozluźniającej lub przetransformowanie punktów do przestrzeni o większej liczbie wymiarów.
- Za pomocą zmiennej rozluźniającej (ang. slack variable) wprowadzamy karę dla punktów znajdujących się na marginesie lub po złej stronie granicy. Teraz szukamy najlepszej hiperpłaszczyzny z uwzględnieniem szerokości marginesu wyznaczonej przez odległości do najbliżej położonych – punktów podparcia i związanymi z nimi wektorami podparcia – i sumy kar naliczonych dla tych punktów, które znajdują się po złej stronie hiperpłaszczyzny.

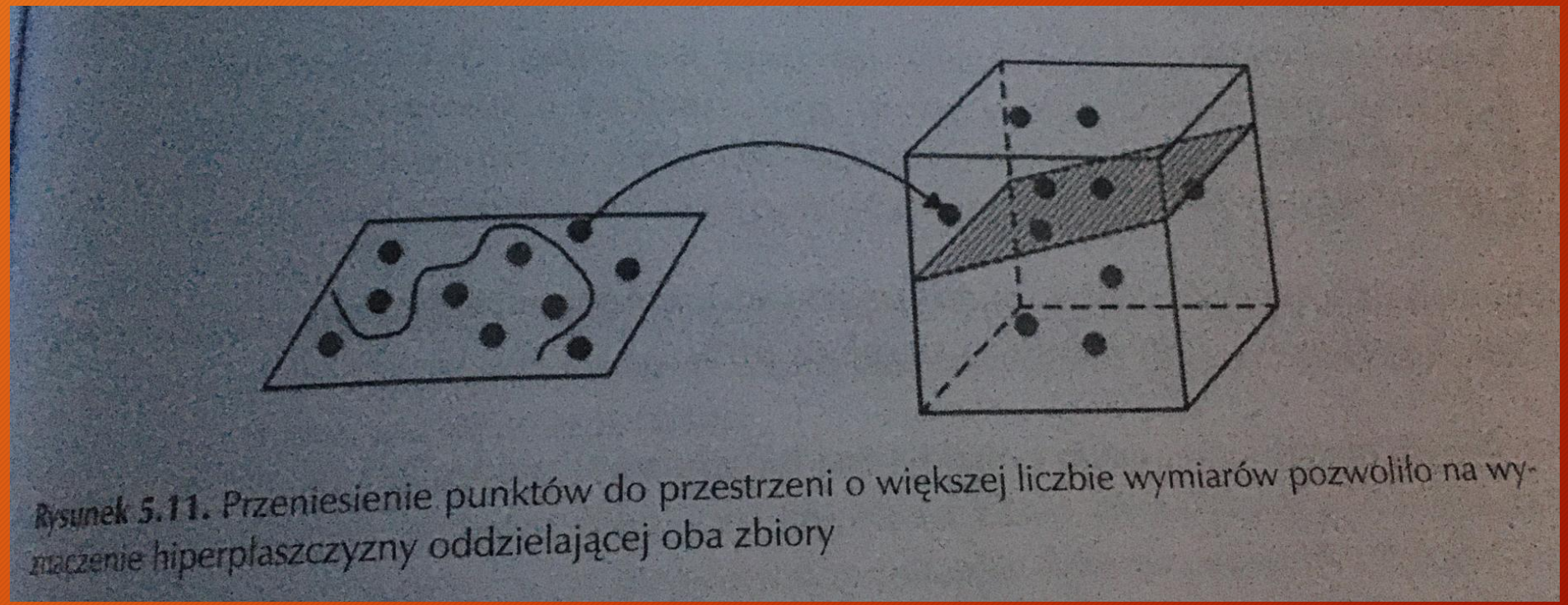
SVM – ciąg dalszy

- Twierdzenie Covera mówi, że projekcja danych do przestrzeni o większej liczbie wymiarów z dużym prawdopodobieństwem pozwoli odseparować je liniowo. Przykład mapa pogody i wysokość n.p.m. a nie tylko długość i szerokość geograficzna.

- Do przetransformowania stosuje się funkcje jądrowe, wielomiany, funkcje sigmoidalne, radialne symetryczne funkcje (RBF), wielomiany sklejane.

- Czy występuje tu klątwa wielowymiarowości? Jeśli uda nam się znaleźć liniową funkcję modelującą dane, to do określenia hiperpłaszczyzny wystarczy nam jedynie tyle punktów, ile wymiarów ma przestrzeń. Nie znamy jednak docelowego rozmiaru przestrzeni…

SVM – ilustracja przetransformowania



Rysunek 5.11. Przeniesienie punktów do przestrzeni o większej liczbie wymiarów pozwoliło na wyznaczenie hiperpłaszczyzny oddzielającej oba zbiory

Źródło: M. Szeliga, Data Science i Uczenie Maszynowe, PWN 2017