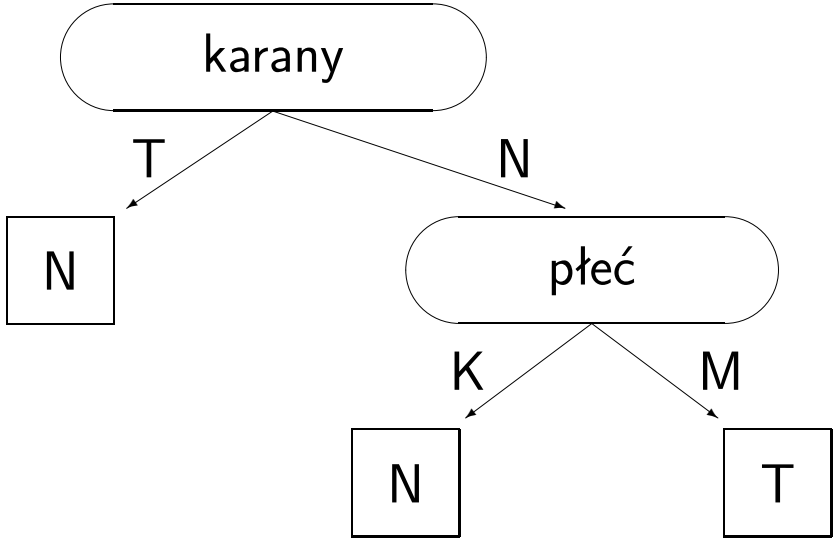
Drzewa decyzyjne — wstęp

Chcemy zautomatyzować podejmowanie wstępnych decyzji o przyznawaniu przez bank kredytów w celu ułatwienia pracy ludziom podejmującym ostateczne decyzje. Mamy pełną historię przyznanych wcześniej kredytów — czy jesteśmy w stanie czegoś się na tej podstawie nauczyć?

i.	n.	wiek	płeć	brutto	wykszt.	zatrud.	karany	...	kredyt
-	-	30	M	18,000	wyższe	2	N	...	T
-	-	42	K	12,000	wyższe	8	N	...	N
-	-	46	M	58,000	wyższe	14	N	...	T
-	-	55	M	22,500	średnie	6	T	...	N
-	-	35	M	36,000	wyższe	4	N	...	T
-	-	22	K	30,000	wyższe	< 1	N	...	N
-	-	28	M	25,000	zawod.	8	N	...	T

Mamy zbiór rekordów z wieloma atrybutami, chcemy znaleźć algorytm wyznaczenia wartości wybranego atrybutu (kredyt) na podstawie wartości pozostałych atrybutów.

Taki algorytm łatwo skonstruować przy pomocy **drzewa decyzyjnego**.



Zauważmy:

- Takie drzewo zawsze uda się skonstruować, chyba żeby istniały dwa wektory identycznych atrybutów, z różnymi decyzjami kredytowymi — nielogiczne.
- Fakt, że drzewo udało się stworzyć, nie gwarantuje jeszcze, że będzie ono prawidłowo działać, tzn. słusznie przewidywać decyzje kredytowe. Być może trzeba zatem stworzyć szereg alternatywnych drzew, i przetestować je na dodatkowym zbiorze danych testowych.
- Niektóre atrybuty należy pominąć (np. imię i nazwisko).
- Inne atrybuty mają specyficzne zbiory wartości (np. dochody brutto, lata pracy), co wtedy? Przedziały wartości?

Drzewa decyzyjne — zasady

- Za pomocą drzewa decyzyjnego można zapisać funkcję binarną wyrażoną przez dowolny zbiór przykładów: pozytywnych i/lub negatywnych, jeśli tylko jest niesprzeczny.

- Trywialna konstrukcja drzewa decyzyjnego: kolejne poziomy odpowiadają sprawdzanym atrybutom, a gałęzie drzewa wyrastające na każdym poziomie odpowiadają konkretnym wartościom tych atrybutów.

- Jednak taka konstrukcja jedynie zapamiętuje poznane przypadki: jeśli dany przypadek pojawi się ponownie to zostanie znaleziony na drzewie, ale jeśli pojawiłby się przypadek odpowiadający nieznanej kombinacji atrybutów, to drzewo nie zawierałoby odpowiedzi.

- Przydałaby się metoda konstrukcji drzewa decyzyjnego zdolna do **uogólniania** obserwacji, np. przez grupowanie przypadków jak w przedstawionym przykładzie.

- Trzeba jednak zauważyć, że w ogólności istnieje bardzo dużo funkcji binarnych (2^{2^n} dla n atrybutów), i nie wszystkie dadzą się uogólnić przez grupowanie.

  Na przykład, funkcja parzystości (sprawdzająca czy liczba przypadków z daną wartością atrybutu jest parzysta), albo funkcja większości, dadzą się wyrazić jedynie pełnym, eksponencjalnej wielkości drzewem decyzyjnym, za to obie dadzą się wyrazić stosunkowo prostymi schematami obliczeniowymi.

- Zatem drzewa decyzyjne mają sens wtedy, gdy można nimi wyrazić istotę jakiegoś pojęcia (zadanego zbiorem przykładów) w sposób zwarty, minimalny.

  Poszukiwana metoda konstrukcji drzew decyzyjnych powinna zatem utworzyć **minimalne** drzewo klasyfikujące dane zjawisko.

- Niestety, stworzenie drzewa decyzyjnego dokładnie minimalnego jest problemem trudnym — wymaga zbudowania wszystkich drzew.

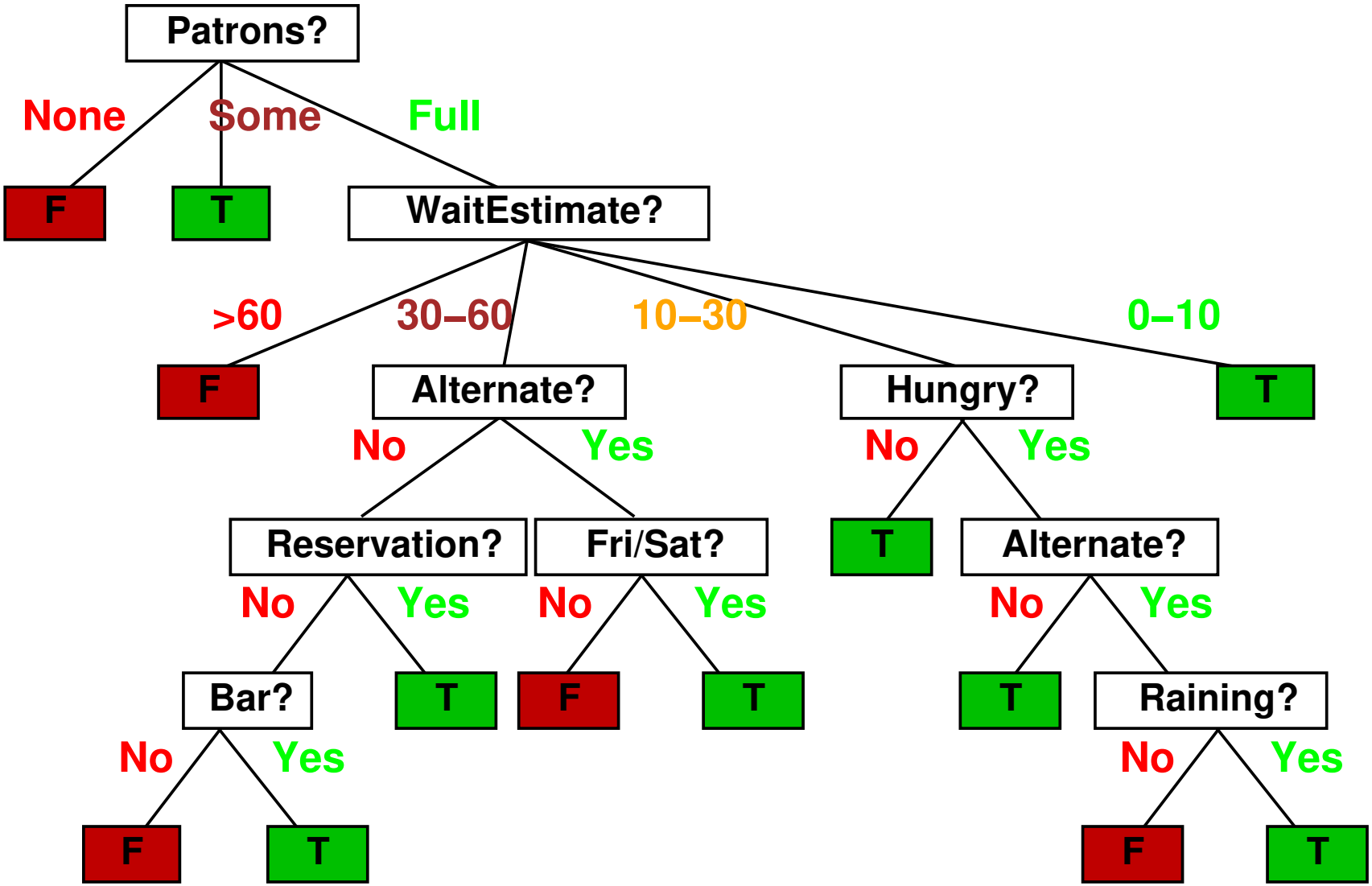
- Zamiast tego można zastosować procedurę heurystyczną wybierając po kolei najważniejsze atrybuty.

Przykład: decyzja czy czekać w restauracji

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

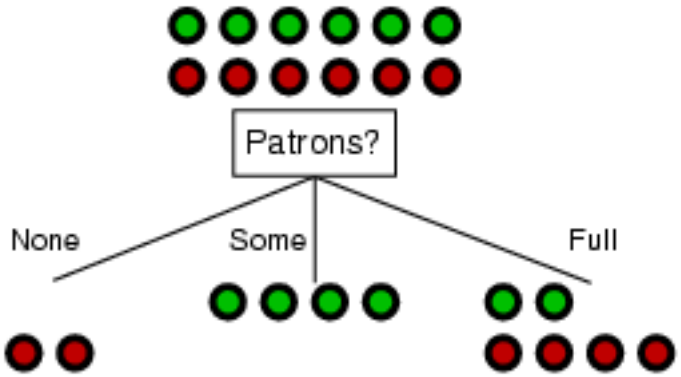
Alternate — czy w pobliżu jest inna „przyzwoita” restauracja
Bar — czy restauracja ma bar, w którym można wygodnie poczekać
Fri/Sat — czy to jest piątek albo sobota
Hungry — czy jesteśmy bardzo głodni
Patrons — jak dużo ludzi jest w restauracji (już przy stolikach)
Price — jak droga jest restauracja
Raining — czy na dworze pada
Reservation — czy mieliśmy zrobioną rezerwację
Type — rodzaj restauracji, czyli serwowana kuchnia
WaitEstimate — oszacowanie czasu oczekiwania (przez kelnera)

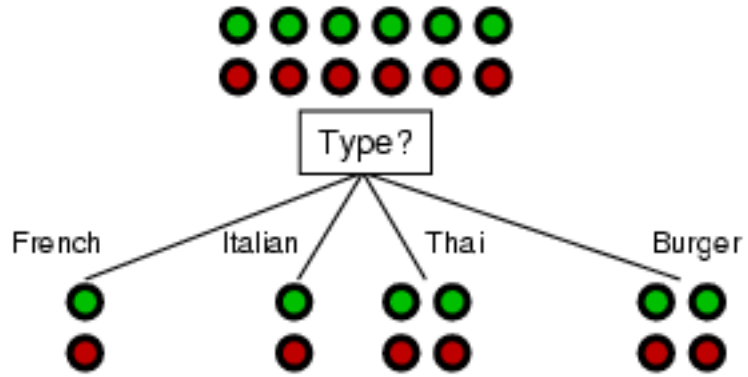
Przykład: drzewo zbudowane „ad-hoc”



Przykład: podejście systematyczne

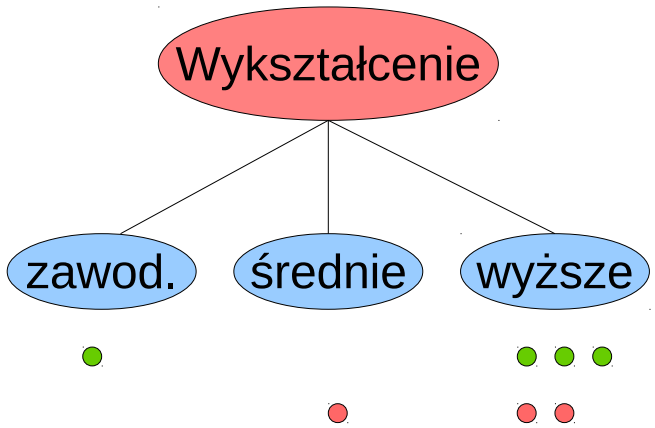
Gromadzimy zestaw przykładów uczących i „przymierzamy” różne atrybuty:

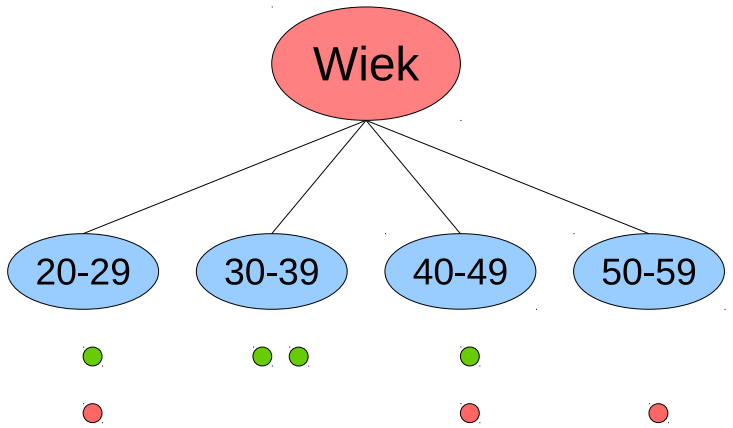




Który atrybut jest lepszym kandydatem do budowy drzewa decyzyjnego? Wydaje się, że „Patrons”. W dwóch przypadkach jednoznacznie determinuje on wynik, a w trzecim, pomimo iż wynik nie jest już jednoznaczny, to i tak jeden wynik ma przewagę ilościową.

Jednak w ogólnym przypadku wybór niekoniecznie będzie tak oczywisty jak powyżej.





Zawartość informacji

Zatem jak można sformalizować wartość podziału wieloklasowego zbioru na podzbiory?

Kluczowym pojęciem jest **informacja**. Ona jest potrzebna do określania odpowiedzi na pytania. Jeśli początkowo nie znamy odpowiedzi na jakieś pytanie, to poznając tę odpowiedź zyskujemy informację.

Pojęciem stosowanym w teorii informacji do mierzenia tego **zysku informacji** jest **entropia**. Jednostką stosowaną do pomiaru entropii jest bit. Jeśli zmienna losowa ma dwie możliwe wartości (np. wynik rzutu monetą) z równomiernym rozkładem prawdopodobieństwa $\langle \frac{1}{2}, \frac{1}{2} \rangle$, to entropia tej zmiennej, równa zyskowi informacji wynikającemu z poznania jej wartości, wynosi 1 bit.

Gdyby jednak moneta nie była równo wyważona, np. spadałaby orłem do góry w 99% przypadków, jej entropia byłaby mniejsza. W skrajnym wypadku, wynik rzutu monetą, która zawsze spada orłem do góry, ma entropię równą zero bitów, bo poznanie wyniku takiego rzutu nie wnosi żadnej nowej informacji.

Entropia

Entropia zmiennej losowej V ze zbiorem wartości v_k jest zdefiniowana jako:

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$$

W przypadku losowego wyboru ze zbioru n^+ próbek pozytywnych i n^- negatywnych, z równomiernym rozkładem prawdopodobieństwa, mamy entropię:

$$H(\langle \frac{n^+}{n^+ + n^-}, \frac{n^-}{n^+ + n^-} \rangle) = -\frac{n^+}{n^+ + n^-} \log_2 \frac{n^+}{n^+ + n^-} - \frac{n^-}{n^+ + n^-} \log_2 \frac{n^-}{n^+ + n^-}$$

Dla 12 próbek przykładu z restauracją mamy $n^+ = n^- = 6$ i entropię:

$$H(\langle \frac{6}{12}, \frac{6}{12} \rangle) = -\frac{1}{2}(-1) - \frac{1}{2}(-1) = 1$$

Jednak obliczenie entropii w pełni jednorodnego zbioru z $n^- = 12$ wyłącznie negatywnymi próbkami komplikuje się:

$$H(\langle 0, \frac{12}{12} \rangle) = -0 \cdot \log_2 0 - 1 \cdot \log_2 1 = -0 \cdot -\infty - 1 \cdot 0 = -0 \cdot -\infty = ?$$

Aby nadać sens temu obliczeniu przyjmiemy, wyłącznie dla celów obliczania entropii, że $0 \cdot \infty \equiv 0$. Zatem: $H(\langle 0, \frac{12}{12} \rangle) = 0$.

Obliczanie entropii

Obliczanie entropii wymaga wykonywania obliczeń z logarytmami o podstawie 2, co jest trudne do wykonania w pamięci, i daje wyniki w postaci niewymiernych liczb zmiennoprzecinkowych. Na przykład, dla rzutu monetą, która spada orłem do góry w 99% przypadków, entropia wynosi:

$$H = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08 \text{ bitów}$$

Poniższy wzór jest przydatny do obliczania logarytmów o podstawie 2 na kalkulatorze, który ma tylko logarytmy dziesiętne lub naturalne:

$$\log_N X = \frac{\log_M X}{\log_M N}$$

Przypomnienie kilku przydatnych wzorów do ręcznego obliczania logarytmów:

$$\begin{array}{lcl} \log_2 0 & = & -\infty \\ \log_2 1 & = & 0 \\ \log_2 2 & = & 1 \end{array} \qquad \begin{array}{lcl} \log_2 4 & = & 2 \\ \log_2 8 & = & 3 \\ \log_2 16 & = & 4 \end{array} \qquad \begin{array}{lcl} \log_N A \cdot B & = & \log_N A + \log_N B \\ \log_N A/B & = & \log_N A - \log_N B \end{array}$$

Logarytmy większych liczb można obliczać jako sumy logarytmów ich czynników, z rozbiciem w dół do liczb pierwszych. Posiadając wynotowane wartości logarytmów dwójkowych kilku początkowych liczb pierwszych można łatwo obliczać logarytmy większych wartości. Na przykład, znając $\log_2 3 \approx 1.585$:

$$\log_2 18 = \log_2(2 \cdot 3 \cdot 3) = \log_2 2 + 2 \cdot \log_2 3 \approx 1 + 2 \cdot 1.585 = 4.17$$

Budowa drzewa decyzyjnego

Zbiór próbek należący do pewnej liczby klas $c_1, c_2, ...c_n$ dzielimy na grupy według pewnego klucza (atrybutu) i instalujemy te grupy jako gałęzie drzewa, którego korzeniem jest wyjściowy zbiór próbek.

n_b^c — liczba próbek klasy c w gałęzi b

n_b — całkowita liczba próbek w gałęzi b

n_t — całkowita liczba próbek we wszystkich gałęziach

Entropię pojedynczej gałęzi obliczamy ze wzoru:

$$H_b = \sum_c \frac{n_b^c}{n_b}(-\log_2 \frac{n_b^c}{n_b})$$

Sumaryczną entropię wszystkich gałęzi wynikającą z podziału zbioru próbek ze względu na wartości danego atrybutu można obliczyć jako:

$$H = \sum_b \frac{n_b}{n_t} \times H_b = \sum_b \frac{n_b}{n_t} \times (\sum_c \frac{n_b^c}{n_b}(-\log_2 \frac{n_b^c}{n_b}))$$

Obliczając w ten sposób wartości oczekiwane entropii dla podziałów generowanych przez różne atrybuty możemy wyznaczyć atrybut zapewniający minimalną entropię, czyli maksymalny zysk informacji ułatwiającej identyfikację klasy. Jeśli dla wybranego atrybutu w niektórych gałęziach nadal pozostaną niejednorodne zbiory próbek, to tę procedurę należy powtórzyć indywidualnie dla podzbiorów w tych gałęziach, budując w ten sposób kompletne drzewo decyzyjne.

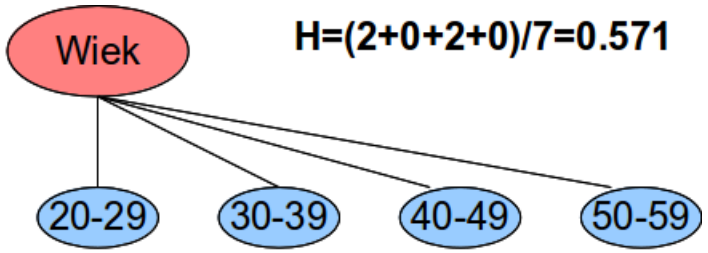
Drzewo zbudowane tą metodą jest minimalnym drzewem zapewniającym klasyfikację danego zbioru próbek. Uczenie się nieznanych pojęć tą metodą odpowiada filozofii brzytwy Ockhama: najprostsza struktura zgodna z obserwacjami danego pojęcia jest zapewne poprawna.

Drzewa decyzyjne — wybór głównego atrybutu: przykład

Obliczmy entropię kolejnych atrybutów dla przykładu decyzji kredytowych:

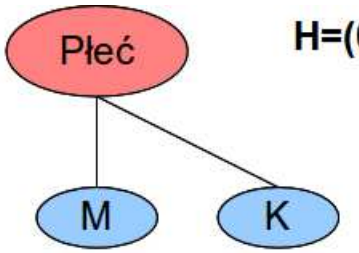
i.	n.	wiek	płeć	brutto	wykszt.	zatrud.	karany	...	kredyt
-	-	30	M	18,000	wyższe	2	N	...	T
-	-	42	K	12,000	wyższe	8	N	...	N
-	-	46	M	58,000	wyższe	14	N	...	T
-	-	55	M	22,500	średnie	6	T	...	N
-	-	35	M	36,000	wyższe	4	N	...	T
-	-	22	K	30,000	wyższe	< 1	N	...	N
-	-	28	M	25,000	zawod.	8	N	...	T

Entropia całego zbioru wynosi $-\frac{3}{7}\log_2(\frac{3}{7}) - \frac{4}{7}\log_2(\frac{4}{7}) \approx 0.985$



H=(2+0+2+0)/7=0.571

T	1	2	1	0
N	1	0	1	1
H	1	0	1	0



H=(0.722*5+0)/7=0.516

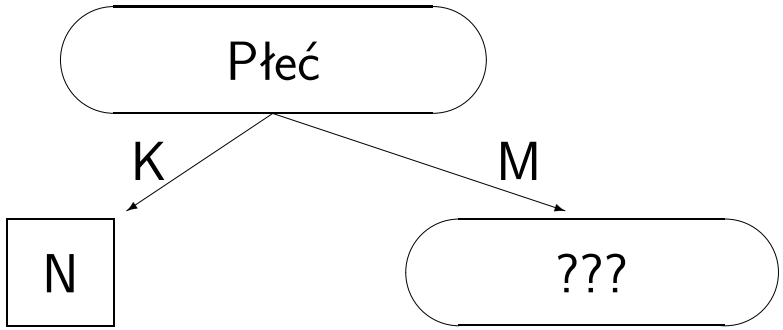
T	4	0
N	1	2
H	0.722	0

Komplet wartości entropii dla wszystkich atrybutów:

$$\begin{array}{lcll} E_{\text{wiek}} & = & 4.00/7 & = & 0.571 \\ E_{\text{płeć}} & = & 3.61/7 & = & 0.516 \\ E_{\text{brutto}} & = & 4.76/7 & = & 0.680 \\ E_{\text{wykszt.}} & = & 4.85/7 & = & 0.694 \\ E_{\text{zatrud.}} & = & 6.76/7 & = & 0.965 \\ E_{\text{karany}} & = & 5.51/7 & = & 0.787 \end{array}$$

Atrybutem o najmniejszej entropii jest Płeć z entropią 0.516. Zysk informacyjny tego atrybutu wynosi $0.985 - 0.516 = 0.469$.

Początkowy fragment drzewa decyzyjnego jest zatem następujący:



Ćwiczenie: wyznacz kolejny atrybut dla gałęzi Płeć=M.

Uwagi o entropii informacji

Pojęcie entropii, którą posługujemy się w teorii informacji pochodzi od Shannona (1948). W fizyce istnieje trochę inne pojęcie entropii termodynamicznej, której jednostką jest J/K (joul/kelvin).

W obliczaniu entropii stosuje się logarytmy, co odpowiada długości napisu potrzebnego do zakodowania danej informacji. Np. jeśli połączymy ze sobą dwa układy, z których każdy ma 1000 stanów binarnych (dla uproszczenia przyjmijmy, że jest to 1024 stanów binarnych), to łączny stan każdego z nich możemy opisać 10 bitami informacji, ale dla zapisania pełnego stanu połączonych systemów (1048576 stanów) potrzeba 20 bitów. Stąd również wynika użycie logarytmów o podstawie 2.

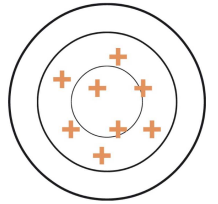
Entropia i zysk informacji jednej zmiennej może być większa niż 1. Na przykład, jeśli zmienna ma 4 możliwe wartości, to dla zapisania jej wartości potrzeba 2 bitów. Rzeczywiście, obliczając entropię równomiernego rozkładu tej zmiennej otrzymujemy:

$$H = \sum_{i=1}^{4} P(v_i)(-\log_2 P(v_i)) = \sum_{i=1}^{4} 0.25 \cdot (-\log_2 0.25) = 4 \cdot (0.25 \cdot \log_2 4) = 2$$

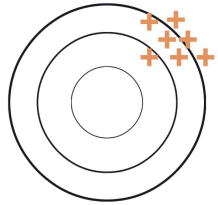
Błędy w danych

Wszystkie procesy uczenia maszynowego są wrażliwe na błędy w danych, w większym lub mniejszym stopniu. Takie błędy to: niepoprawnie, niedokładnie, lub całkowicie błędnie zarejestrowane, jak również niepoprawnie zapisane wyniki klasyfikacji. Błędy mogą być różnej natury i mieć różne przyczyny. Ale w rzeczywistym środowisku produkcyjnym są one nieuniknione podczas przetwarzania tysięcy lub setek tysięcy próbek. Wrażliwość procesu uczenia na takie błędy należy zawsze brać pod uwagę.

Błędy losowe, zwane również **szumem** (*noise*), są obecne we wszystkich procesach pomiarowych, są nieprzewidywalne i trudne do kontrolowania, i typowo wykazują dużą wariancję i małe odchylenie (*bias*).



Błędy systematyczne występują w procesach pomiarowych wskutek wad sprzętu i/lub oprogramowania, wadliwych założeń lub metodologii, ustawień parametrów, itp. Typowo mają niską wariancję i duże odchylenie.



W uczeniu maszynowych błędy systematyczne są gorsze ponieważ na ogół prowadzą do tworzenia modeli obarczonych istotnym odchyleniem. Natomiast błędy losowe często kasują się nawzajem, i wiele metod maszynowego uczenia dobrze radzi sobie z nimi, tworząc modele z niską wariancją i odchyleniem.

Problemy z brakującymi i/lub błędnymi danymi

brakujące wartości niektórych atrybutów w niektórych próbkach

W wielu rzeczywistych zbiorach danych zarejestrowanych w pewnym okresie czasu przez ludzi lub czujniki elektroniczne, brakuje pewnych danych.

Aby rozwiązać ten problem, możemy wyeliminować odpowiednie próbki, lub atrybut(y). W każdym przypadku tracimy dane, co może być akceptowalne, lub nie. Alternatywnym podejściem może być uzupełnienie brakujących wartości sztucznie generowanymi danymi. Muszą one być wygenerowane losowo, przez staranne dopasowanie rozkładu wartości danego atrybutu, z odpowiednim skalowaniem liczby takich próbek.

błędne wartości dla niektórych atrybutów

Przypadek błędnych danych jest gorszy, ponieważ często (zwykle) nie „widać" tych błędnych danych, a prowadzą one do błędnego procesu uczenia, a nawet niemożności poprawnego nauczenia się niektórych zbiorów danych.

Takie problemy można wykryć, porównując wyniki uczenia dla różnych podzbiorów danych. „Podejrzane" próbki można wyeliminować ze zbioru, lub można usunąć z nich błędne wartości, zastępując je losowo wygenerowanymi, jak powyżej.

Warunek stopu budowy drzewa decyzyjnego

Bazowa procedura budowania drzewa decyzyjnego kończy się gdy zestaw próbek otrzymany w procesie rekurencyjnego budowania drzew jest jednorodny.

Jednak biorąc pod uwagę praktyczne sytuacje, uwzględniając błędy danych, powinniśmy rozszerzyć ten warunek. Ilekroć zbiór danych w rekurencyjnym kroku budowy drzewa decyzyjnego zawiera próbki o identycznych wartościach atrybutów wejściowych, ale innej wartości atrybutu klasy, procedura powinna również zostać zatrzymana. Ale jaka powinna być zwrócona wartość, gdy klasyfikuje się nową próbka z takim wektorem atrybutów? Oczywiście, taka sytuacja oznacza błąd, więc można podjąć próbę rozwiązania problemu (np. znaleźć przyczynę i poprawić błąd).

Jeśli w podzbiorze uczącym istnieje jednoznaczna większość, wtedy można przyjąć klasę tej większości. Jeśli istnieje znaczący zestaw wartości klas z pewnym rozkładem, wtedy sytuacja jest inna. Jeśli budowany klasyfikator dopuszcza odpowiedź "Niezdecydowany", wskazując, że należy podjąć inne działanie, to jest to chyba najodpowiedniejsze rozwiązanie. Jednak gdy klasyfikator musi być w pełni zautomatyzowany, bez możliwości interwencji człowieka, wtedy możliwym rozwiązaniem jest wybrać losową wartość klasy używając oryginalnego rozkładu zbioru wartości.

Warunek stopu budowy drzewa decyzyjnego (cd.)

Powyższe poprawki nie rozwiązują definitywnie problemu stopu budowy drzewa. Mogą występować błędy w danych inne niż wiele próbek z identycznymi wartościami x i różnymi wartościami y. Na przykład, próbki o identycznych wartościach y mogą być różnicowane przez jakiś nieistotny atrybut lub przez małą różnicę numeryczną. Taki nieistotny atrybut lub różnica liczbowa, nigdy nie trafiłby do klasyfikatora drzewa decyzyjnego, gdyby go nie pojawiły się błędne próbki.

Aby wykryć i właściwie rozwiązać takie przypadki, potrzebne są bardziej zaawansowane warunki stopu. Miara zysku informacji może użyta do oceny, czy rekurencyjny węzeł drzewa decyzyjnego powinien być zbudowany, czy budowa drzewa powinna zostać zatrzymana, a wartość obliczona za pomocą powyższej procedury.

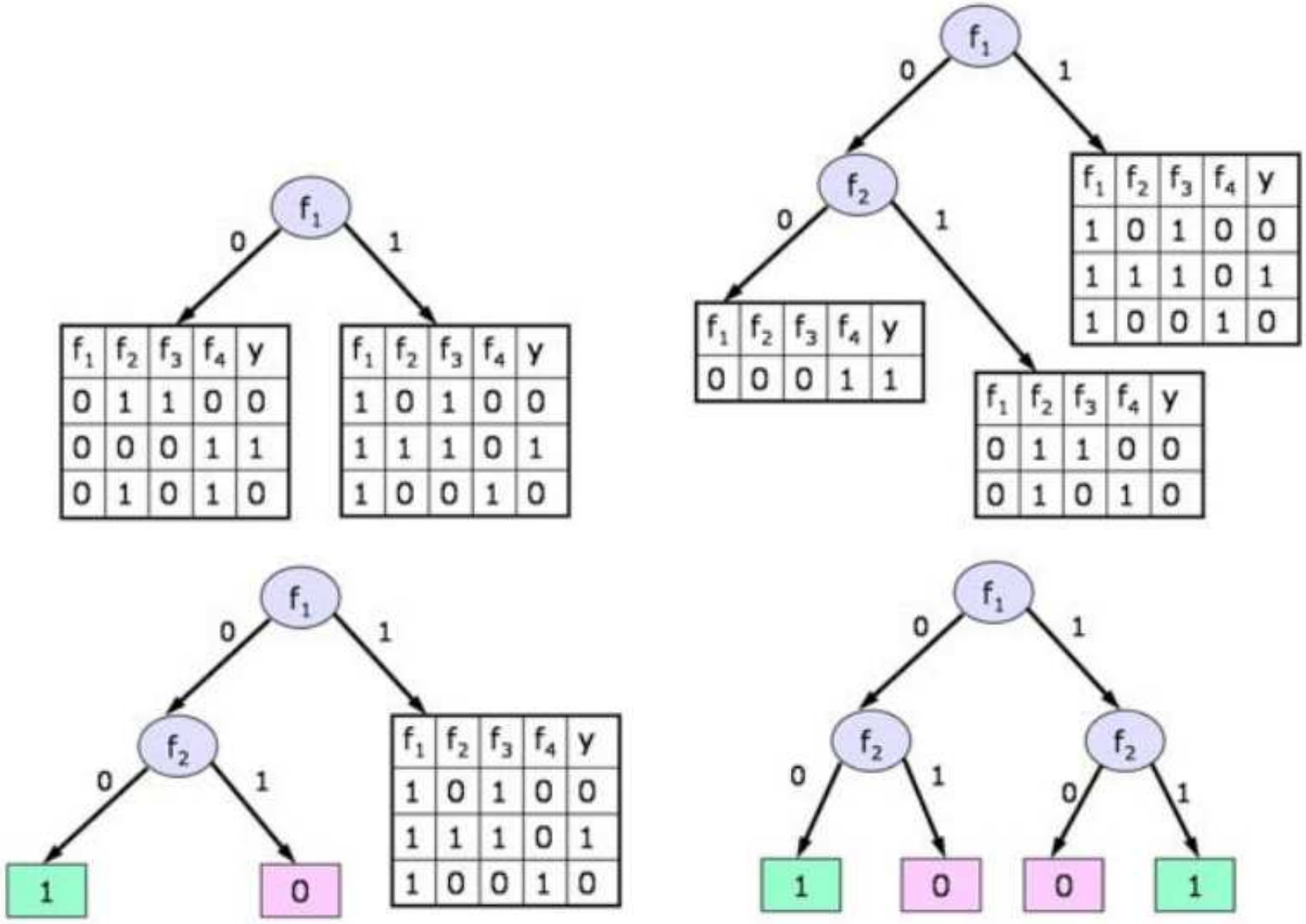
Warunek stopu budowy drzewa decyzyjnego (3)

Niestety, to też nie zawsze działa poprawnie. Niektóre trudne przypadki w klasyfikacji wynikają z wzorców danych, które są podobne do logicznego "exclusive-OR. "
Rozważmy następujący zestaw danych, otrzymany z funkcji $(f_1 \wedge \neg f_2) \vee (\neg f_1 \wedge f_2)$:

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	0	0
1	1	1	0	1
0	0	0	1	1
1	0	0	1	0
0	1	0	1	0

Entropia całego zbioru danych wynosi 0.92, ale entropie podziału na wszystkich czterech atrybutach jest również 0.92. Powyższy warunek zatrzymania sprawiłby, że algorytm przestałby budować to drzewo.

Jeśli zamiast tego dokonalibyśmy podziału na atrybucie f_1, a następnie na f_2, otrzymalibyśmy następujący wynik:



Przycinanie drzew decyzyjnych

Jak widać powyżej, może być przydatne kontynuowanie budowania drzewa decyzyjnego nawet przy niewielkim lub zerowym zysku informacji. Ale jak rozróżnić tę sytuację od przeuczenia?

Zaawansowana wersja algorytmu drzewa decyzyjnego powiększa drzewo w miarę możliwości, a po zakończeniu decyduje, czy wynikiem jest użyteczny klasyfikator, lub czy drzewo powinno zostać przycięte, i zastosowany jeden z powyższych warunków stopu.

Problemy z danymi liczbowymi

Algorytmy uczenia się klasyfikacji zakładają, że każdy atrybut ma skończony zbiór wartości. W przypadku parametrów numerycznych lub wielowartościowych pojawiają się problemy:

ciągłe lub nieskończone domeny atrybutów

Można stosować dyskretyzację, ale często konkretne zakresy wartości są krytyczne. Mogą być wybierane automatycznie, za pomocą analizy statystycznej, lub ręcznie, jeśli wiemy, jakie zakresy są ważne dla danego przypadku.

atrybuty dyskretne, ale wielowartościowe

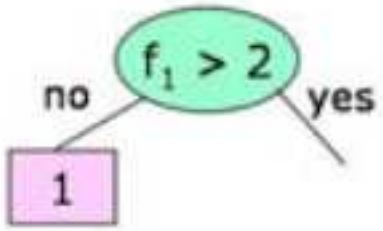
Grupy zdefiniowane przez ich wartości będą względnie małe, a zatem często bardzo jednorodne. Mogą więc wydawać się przydatne do klasyfikacji, w rzeczywistości będąc bezwartościowymi. Takie przypadki można wykryć, obliczając "względny zysk" atrybutu jako zysk względem zawartości informacji.

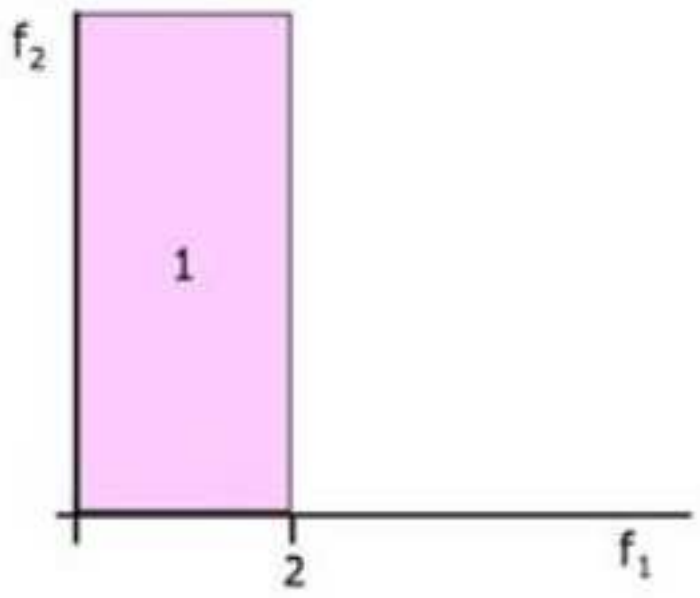
ciągły atrybut wyjściowy (klasy)

W tym przypadku nie chcemy wybierać wartości wyjściowej z małego zbioru, ale obliczyć wartość liczbową. Jest to problem **regresji**, który różni się od klasyfikacji. Istnieją odmienne metody dla tego zagadnienia (niżej).

Binarne drzewa decyzyjne

Poprzednie podejście do budowania drzew decyzyjnych polegało na dyskretyzacji atrybutów liczbowych i traktowaniu ich jak dyskretnych. Alternatywne podejście polega na użyciu w gałęziach drzewa jedynie testów binarnych w postaci $x_j > c$.

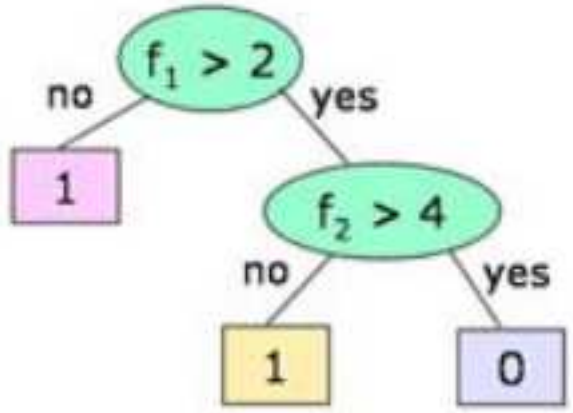


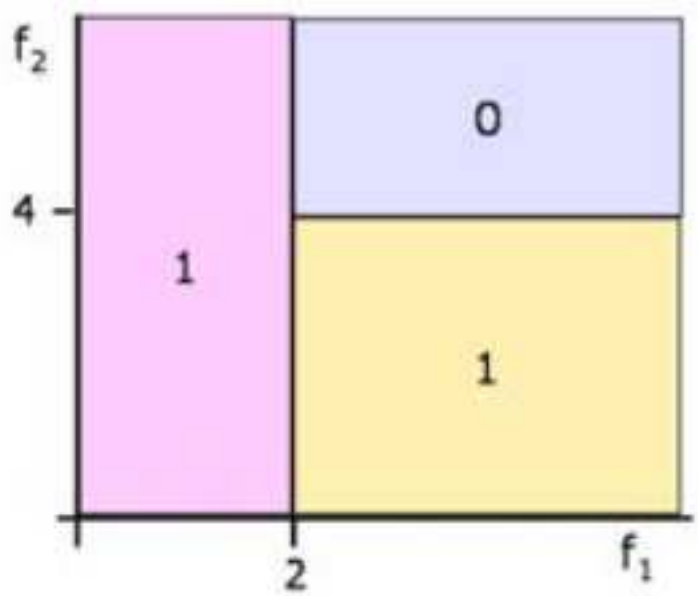


Spowoduje to podzielenie przestrzeni cech na prostokąty, a w przestrzeni wielowymiarowej na hiper-prostokąty.

Drzewa decyzyjne z binarnymi podziałami na atrybutach liczbowych

Klasa hipotez dopuszczanych w tym modelu jest dość bogata, choć może nie być w stanie wyrazić niektórych pojęć.

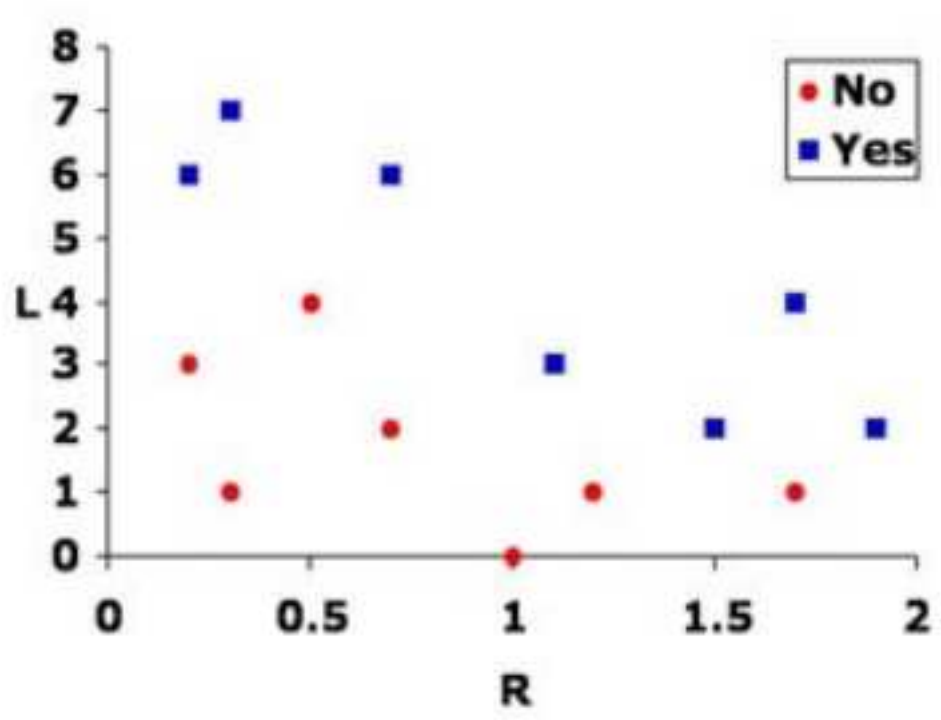




Przewidywanie bankructwa

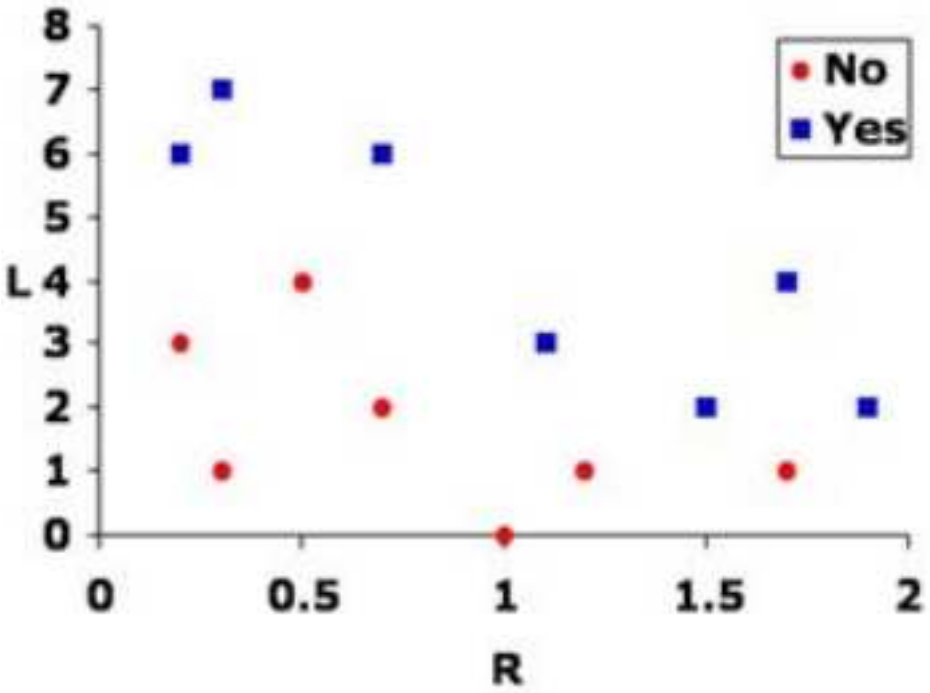
Załóżmy, że próbujemy przewidzieć bankructwo w najbliższej przyszłości danej osoby analizując dwa parametry: liczbę spóźnionych spłat kredytu w ciągu roku, oraz stosunek wydatków do dochodów kredytobiorcy. Jeśli te wartości są duże, szanse bankructwa są wysokie.

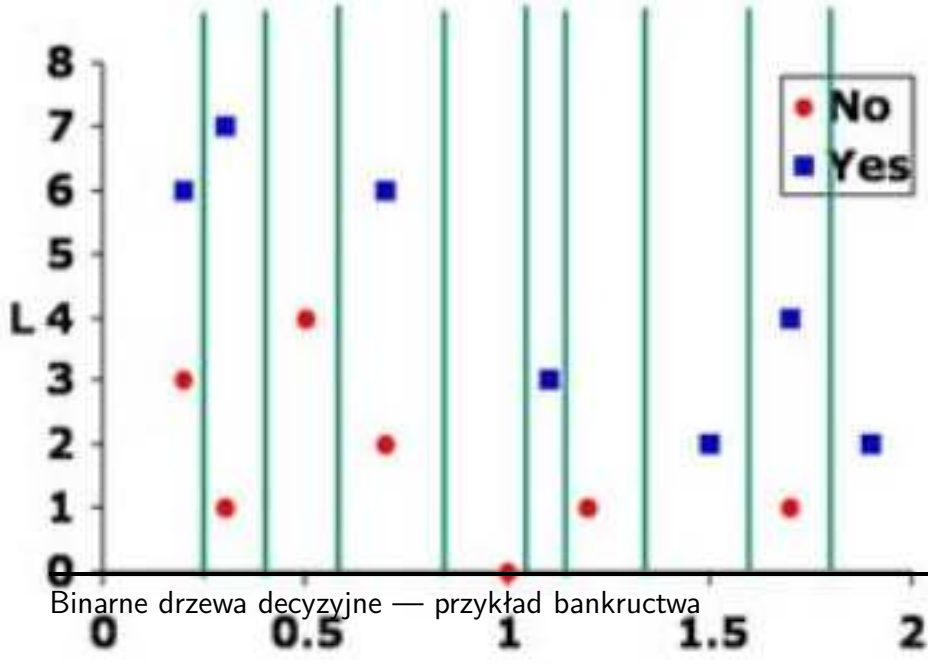
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes

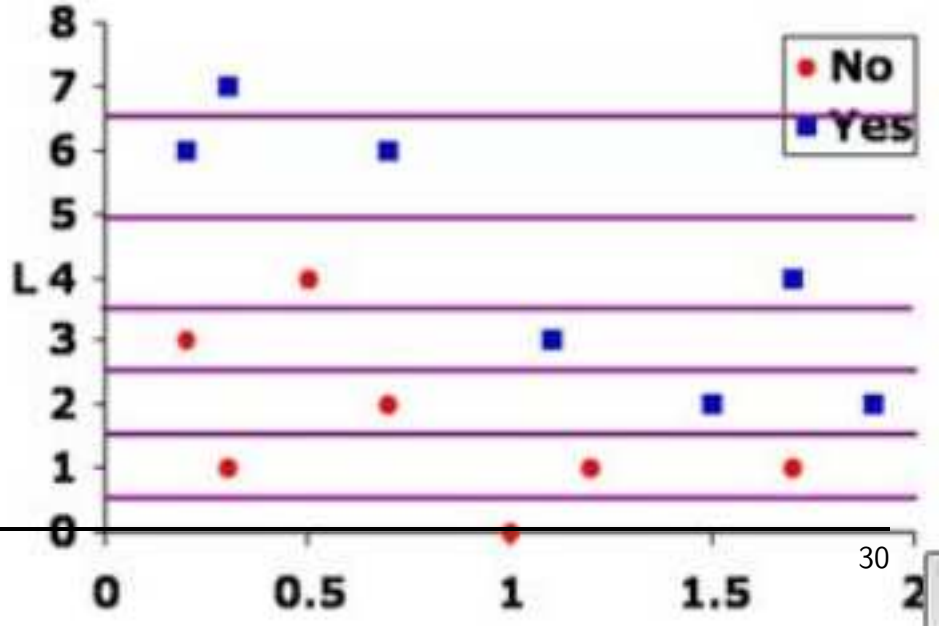


Możliwe podziały

Chcemy rozważyć podział między dowolnymi dwoma punktami w każdym z wymiarów.

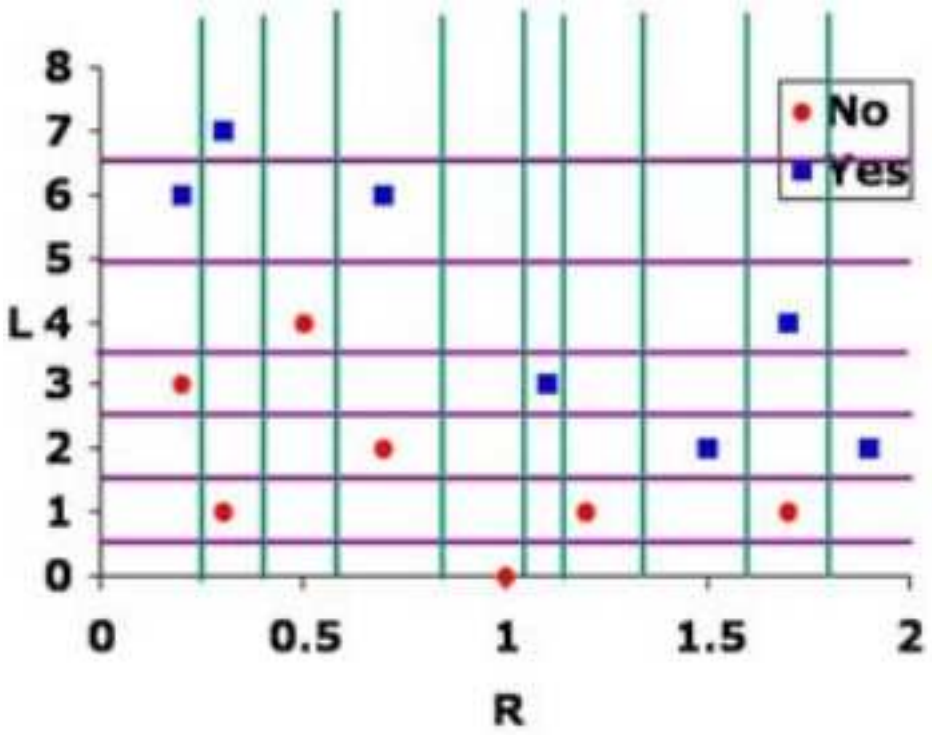






Możliwe podziały (cd.)

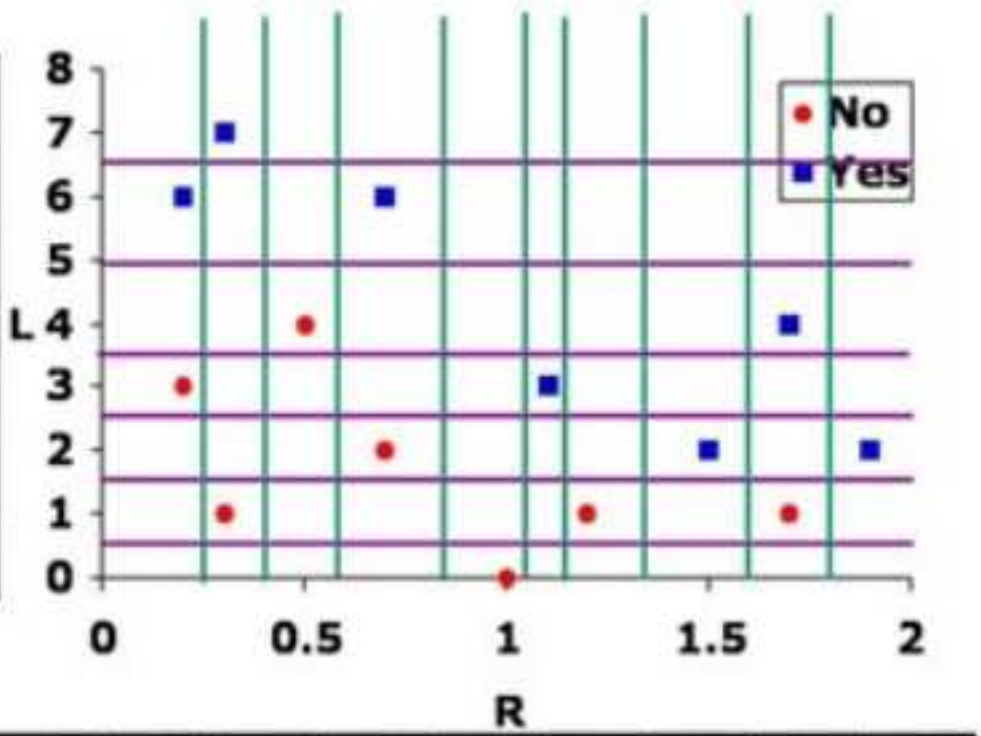
Mamy wiele możliwych podziałów do rozważenia. Musimy wybrać tylko jeden, ten który minimalizuje średnią entropię swoich węzłów potomnych.



Drzewo decyzyjne dla problemu bankructwa

Obliczanie średniej entropii dla wszystkich możliwych podziałów:

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93
	# neg to left	# pos to left	# neg to right	# pos to right	

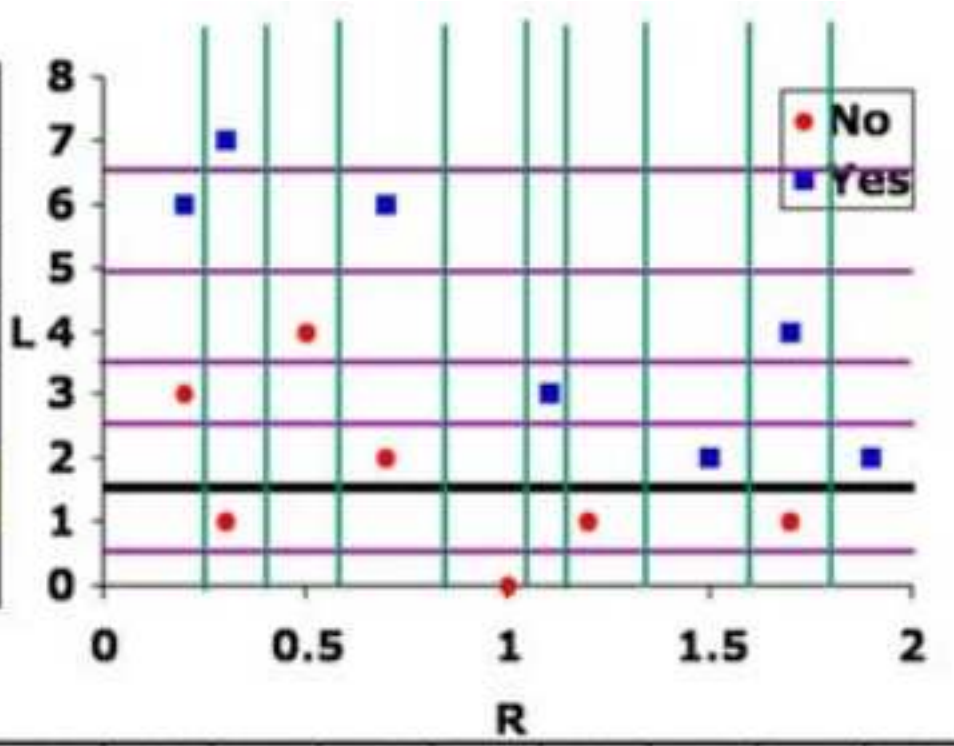


AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Drzewo decyzyjne dla problemu bankructwa (cd.)

Podział na $L > 1.5$ daje minimalną entropię wyznaczając pierwszy podział:

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

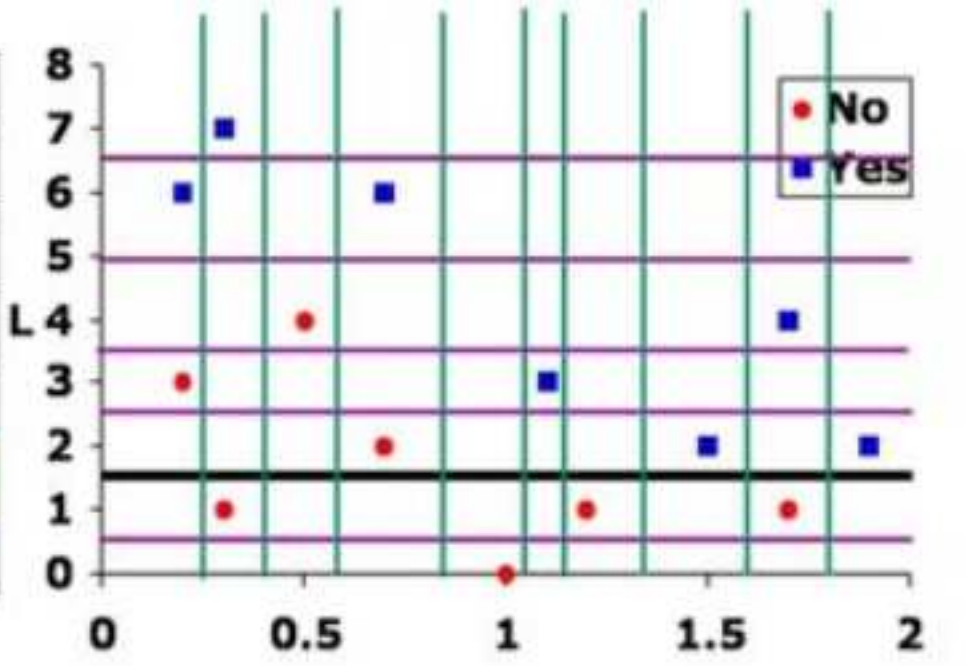


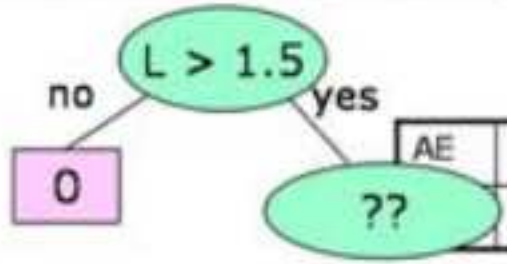
AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
R<x	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Drzewo decyzyjne dla problemu bankructwa (cd.)

Ponieważ wszystkie punkty spełniające $L < 1.5$ są w klasie czerwonej (brak bankructwa), zatem możemy utworzyć liść na drzewie.

L<y	NL	PL	NR	PR	AE
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93



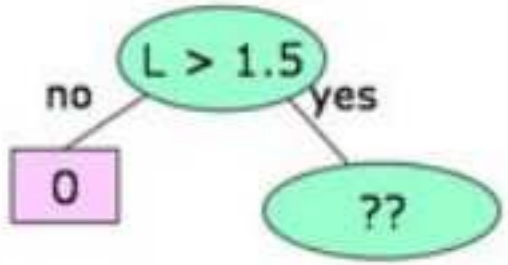


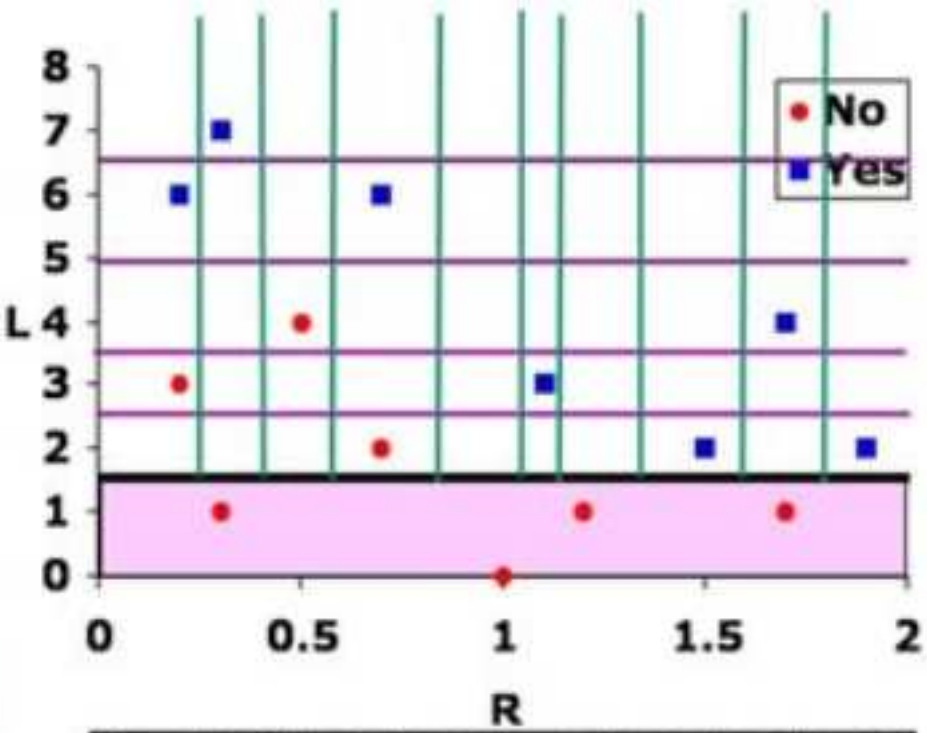
AE	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

Drzewo decyzyjne dla problemu bankructwa (cd.)

Teraz obliczamy drugi podział w drzewie. Do tego należy obliczyć wszystkie entropie od nowa, ponieważ dane z liści są wyłączone z rozważań.

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88



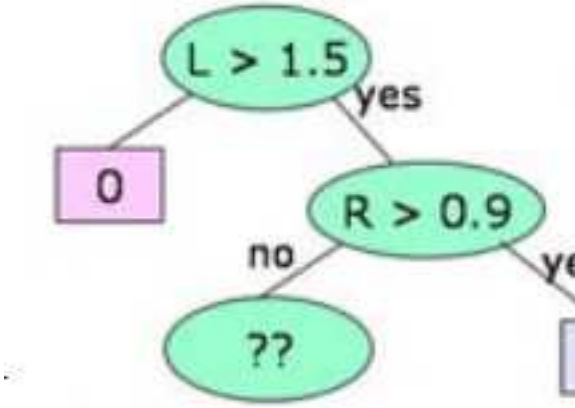


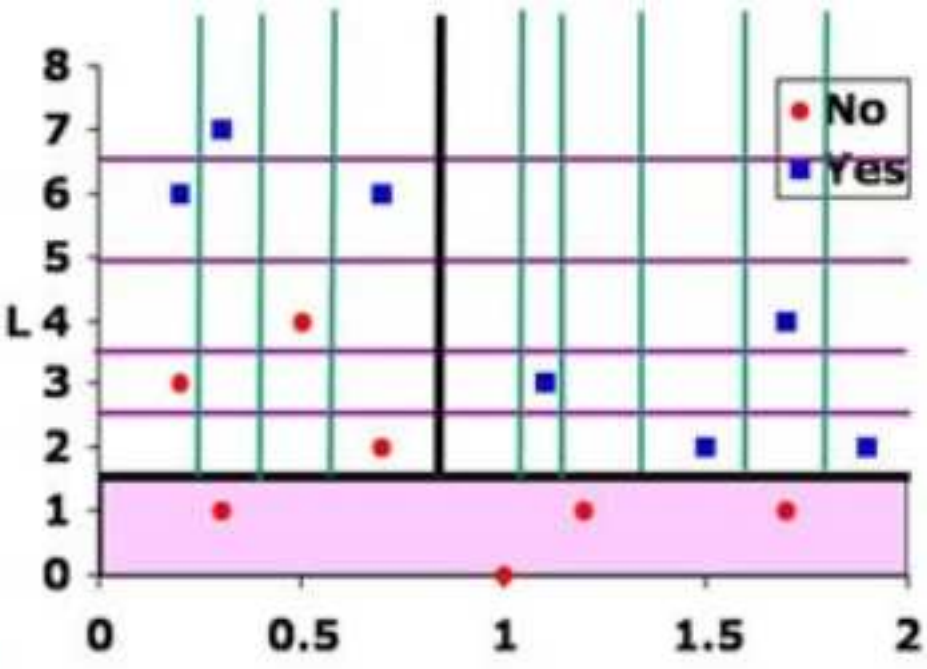
AE	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Drzewo decyzyjne dla problemu bankructwa (cd.)

W tym przypadku najmniejszą entropię ma podział na $R > 0.9$. Drzewo zostaje rozbudowane i stworzony oraz zaetykietowany jest nowy liść.

L<y	NL	PL	NR	PR	AE
6.5	6	3	0	1	0.83
5.0	4	3	0	3	0.69
3.5	3	2	4	1	0.85
2.5	2	1	5	2	0.88



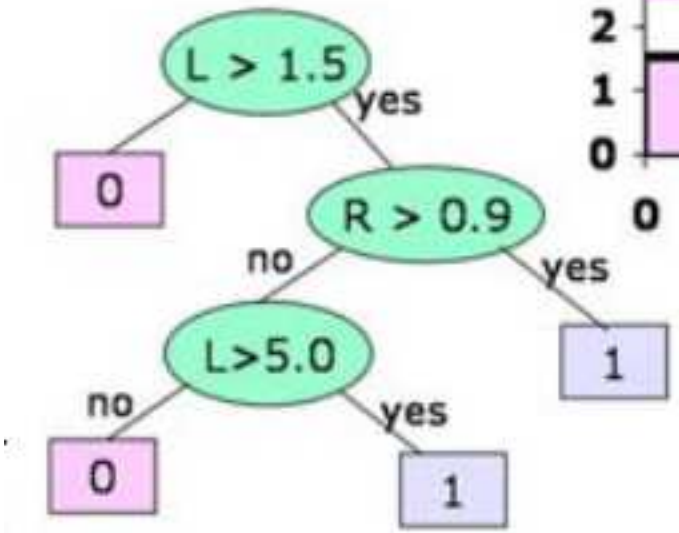


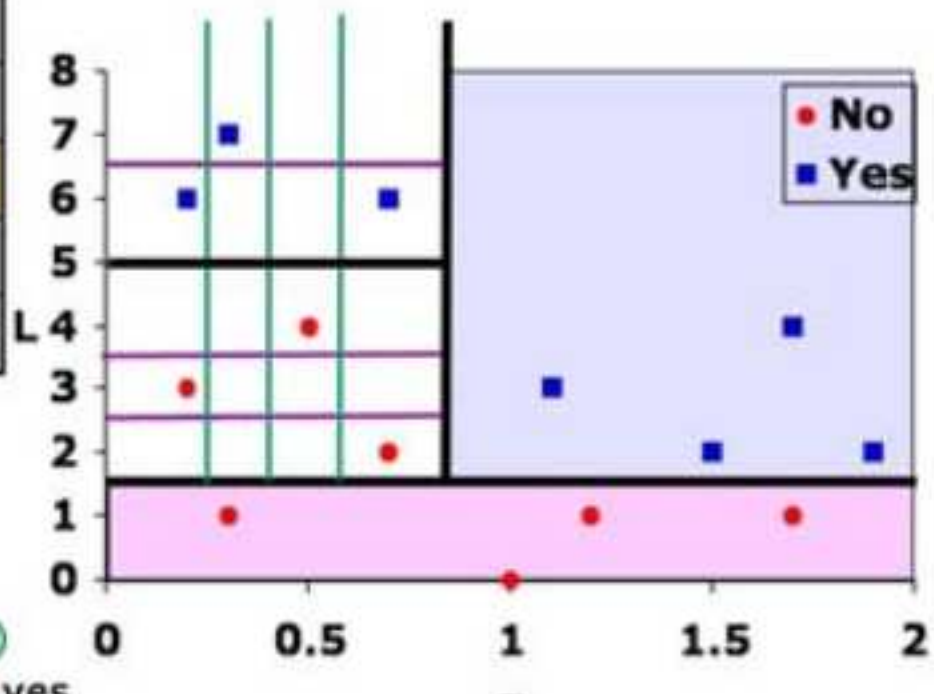
AD	0.85	0.88	0.79	0.60	0.69	0.76	0.83
R<x	0.25	0.40	0.60	0.90	1.30	1.60	1.80

Drzewo decyzyjne dla problemu bankructwa (cd.)

Ponownie obliczamy cały zestaw entropii. Kolejny podział minimalnej entropii uzyskujemy dla $L > 5$. Rozdziela on klasy pozostałych próbek kończąc budowę drzewa.

L<y	NL	PL	NR	PR	AE
6.5	3	2	0	1	0.81
5.0	3	0	0	3	0.00
3.5	2	0	1	3	0.54
2.5	1	0	2	3	0.81

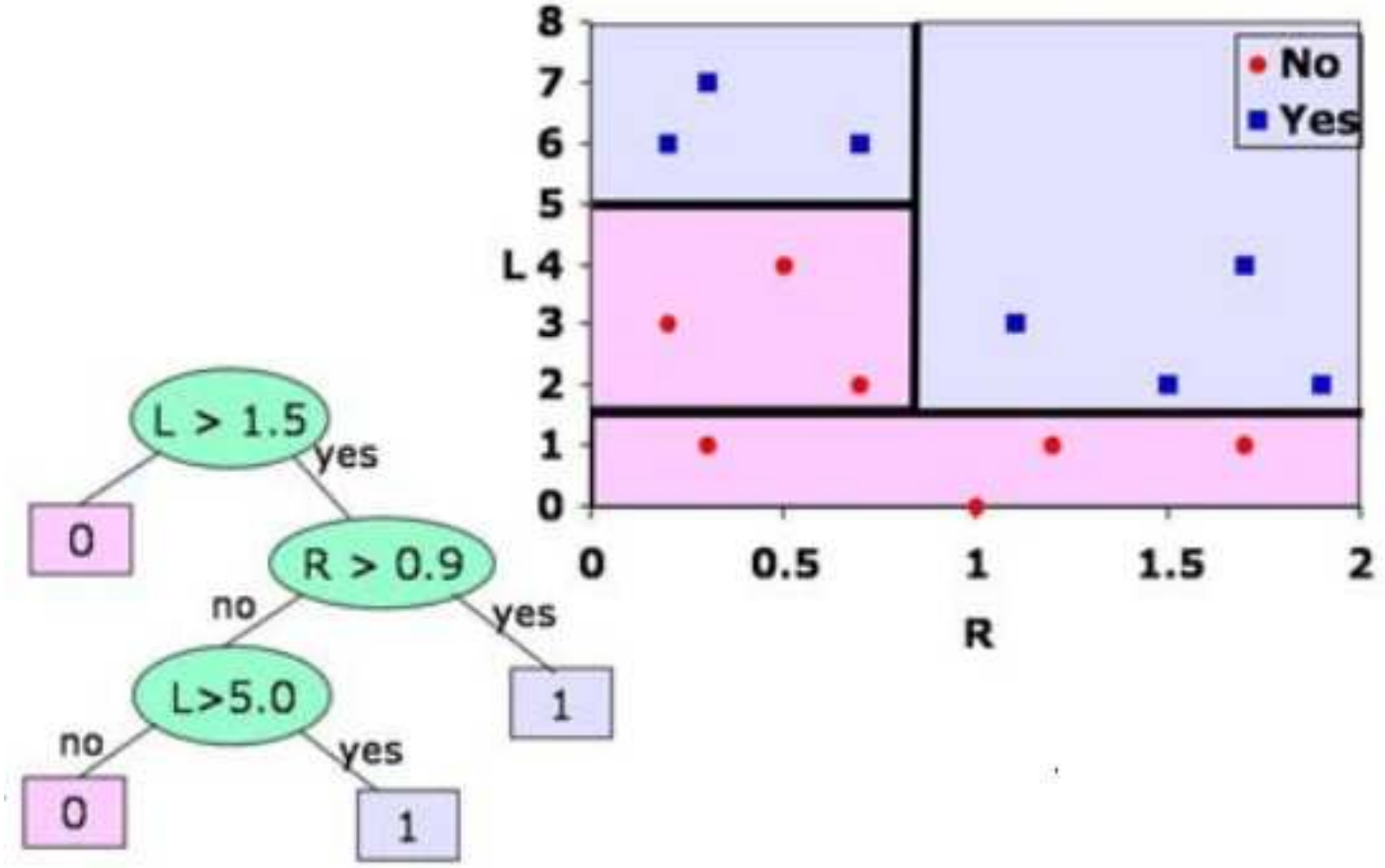




AE	1.00	0.92	1.00
R<x	0.25	0.40	0.60

Drzewo decyzyjne dla problemu bankructwa (cd.)

Kompletne drzewo ma zerowy błąd na zbiorze uczącym.



Efektywność uczenia indukcyjnego

Najprostszym sposobem oceny wyników procesu uczenia jest sklasyfikowanie wszystkich próbek z jakiegoś zbioru, dla których znana jest prawdziwa klasa, i obliczenie **Dokładności** (*Accuracy*):

$$\text{Accuracy} = \frac{\text{liczba poprawnie sklasyfikowanych próbek}}{\text{liczba wszystkich sklasyfikowanych próbek}}$$

Alternatywnie możemy obliczyć **Błąd** (*Error*) jako dopełnienie dokładności:

$$\text{Error} = 1 - \text{Accuracy}$$

Dokładność może być wykorzystana do porównania wyników uczenia tych samych danych przez różne algorytmy uczenia. Jednak nie reprezentuje ona pełnej informacji, czego algorytm się nauczył, a czego nie.

Problemy z Dokładnością

Dokładność jest pojedynczą miarą wydajności klasyfikatora na jakimś zbiorze danych. Jednak jej wartość często może być trudna do zinterpretowania lub porównania.

Bezwzględna wartość Dokładności nie ma uniwersalnego znaczenia. Dokładność=10% może być rewelacyjna przy kupowaniu losów w loterii krajowej. Ale dokładność=90% może być uważana za niską przy wyborze antidotum na ukąszenie jadowitego węża.

Dokładność=50% w klasyfikacji binarnej jest równoznaczna z przypadkowym zgadywaniem. Jednak w przypadku 13 klas te same 50% oznacza, że połowie próbek zostanie przypisana poprawna klasa, jedna z trzynastu możliwych. Dla konkretnego celu może to być wynik użyteczny lub nie, ale jest to na pewno wynik znacząco pozytywny.

Z kolei, rozważmy badanie przesiewowe w kierunku raka. Załóżmy, że pięć na 1000 ludzi ma rozwijającą się chorobę i może to być wykryte przez test przesiewowy. Klasyfikator A poprawnie identyfikuje 4 z 5 przypadków raka, ale błędnie oznacza jako nowotwór próbki 25 z 995 zdrowych osób, co daje całkowitą Dokładność=97.1%. Natomiast klasyfikator B rozpoznaje tylko 2 na 5 przypadków raka, a nieprawidłowo klasyfikuje 15 spośród 995 zdrowych ludzi, co daje Dokładność=98.2%, lepszą niż A. Jednak jego zdolność do wykrywania raka jest wyraźnie gorsza.

Zbiór uczący, walidacyjny, i testowy

Każdy algorytm uczenia nadzorowanego tworzy klasyfikator oparty na pewnym zestawie danych zwanym **zbiorem uczącym**, dla którego zarówno wejściowe wartości atrybutów jak i klasa są znane. Klasyfikator może następnie zostać oceniony przez klasyfikowanie próbek z tego samego zestawu uczącego. Alternatywnie może być testowany na próbkach z innego zestawu, zwany **zbiorem testowym**, dla którego zarówno wartości atrybutów wejściowych, jak i klasa muszą również być znane.

Jednak zbiór testowy nie może być wykorzystany na żadnym etapie procesu uczenia. Z definicji, można go użyć tylko do końcowej, jednorazowej oceny pracy klasyfikatora. W środowisku akademickim zbiór ten jest typowo dostępny dla programisty budującego klasyfikator. Jednak w prawdziwym scenariuszu przemysłowym taki zbiór powinien pozostać nieznany (a wręcz ściśle tajny), aby zapewnić obiektywność końcowego testu. Inaczej możnaby się obawiać, że producent, chcąc zwiększyć swoją konkurencyjność, podrasuje klasyfikator pod kątem osiągania nadzwyczaj dobrych wyników dla zbioru testowego, niezależnie od skuteczności jego pracy na rzeczywistych danych.

Jednocześnie, w trakcie budowy klasyfikatora, wyboru algorytmu i strojenia jego parametrów, potrzebne jest wielokrotne testowanie dla zapewnienia iteracyjnego cyklu budowy klasyfikatora. W tym celu tworzy się dodatkowy zbiór testowy, zwany **zbiorem walidacyjnym**, który powinien być różny od zbioru uczącego, lecz może być wielokrotnie wykorzystywany w procesie uczenia, do strojenia klasyfikatora.

Zbiór uczący, walidacyjny, i testowy (cd.)

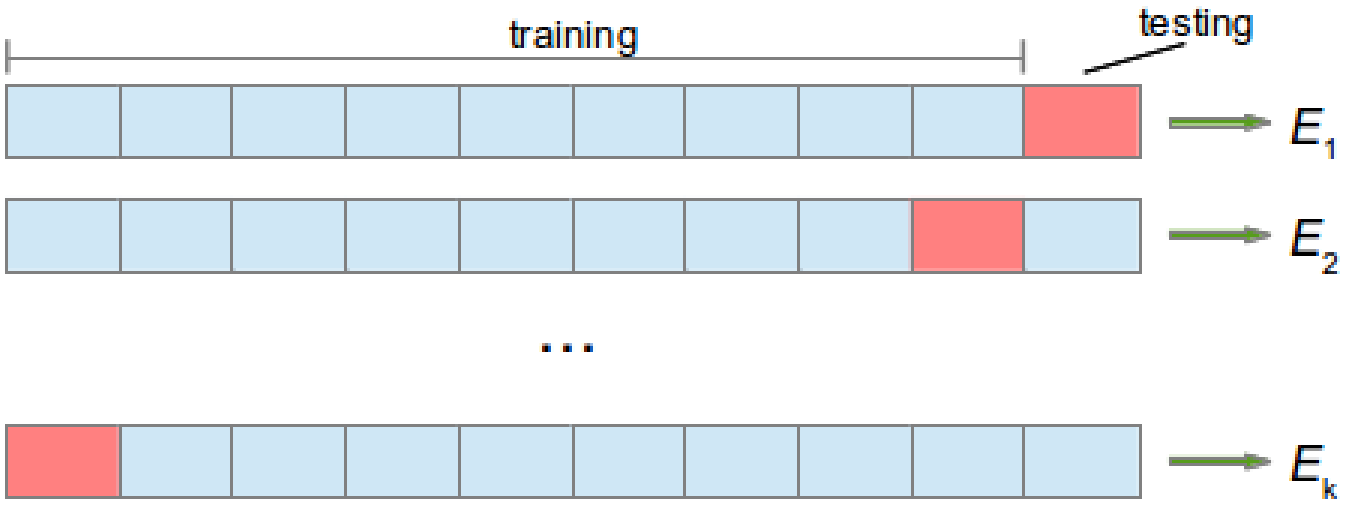
Oczywiście, dla udanego uczenia maszynowego zarówno zbiór uczący jak i walidacyjny (a do wiarygodnego testowania również zbiór testowy), powinny mieć właściwości maksymalnie podobne do rzeczywistych danych. To jest powód, dla którego próbki są losowo wybierane z populacji do testowania leków, sondowania opinii społecznych itp.

Podobnie losowo powinny być utworzone z dostępnych danych zbiory uczący i walidacyjny. Jednak każdy wybór, nawet losowy, jest obarczony pewnym odchyleniem, zarówno wartości średnich, jak i wariancji. Mówiąc obrazowo, próbki zbioru walidacyjnego mogą być albo szczególnie łatwe albo szczególnie trudne do sklasyfikowania, albo w ogóle średnio nieco inne, w porównaniu ze zbiorem uczącym. Powoduje to zaburzenie obliczanych wartości błędów walidacji, i być może mylne strojenie klasyfikatora.

Dla zapewnienia wiarygodności statystycznej, oba zbiory powinny być możliwie duże, co jest łatwe gdy dostępnych danych jest dużo, albo jest możliwe ich nieograniczone pozyskiwanie. Jednak w wielu przypadkach wielkość całego posiadanego zbioru danych jest ograniczona.

Walidacja krzyżowa

Prostą i skuteczną metodą wykorzystania ograniczonego zbioru danych zarówno do trenowania jak i walidacji jest **walidacja krzyżowa** (*cross-validation*). Dostępny zbiór danych jest dzielony losowo na k równych podzbiorów (k-*folds*). Następnie pierwsze k-1 podzbiorów jest użytych do treningu, a ostatni podzbiór do walidacji, z otrzymanym błędem walidacji E_1. Procedura jest powtarzana k razy, za każdym razem przy użyciu innego podzbioru do walidacji.



Następnie Błąd (Error) klasyfikacji jest obliczany jako średnia $E = \frac{\Sigma_i E_i}{k}$. Estymuje ona wartość Błędu na dowolnym zbiorze danych o właściwościach podobnych do posiadanego zestawu danych. Najlepszą wartość k można znaleźć eksperymentalnie dla konkretnych danych, ale często używane jest k=10.

Walidacja LOO

Walidacja krzyżowa jest podstawą pracy w uczeniu maszynowym. Jednak dla bardzo małych zbiorów danych (np. poniżej 100 próbek), szczególnie dla małej wartości k, każdy z eksperymentów walidacji np. przy k=5 jest obarczony ryzykiem uczenia z niereprezentatywnego zbioru danych. W takich przypadkach można stosować specyficzną formę walidacji zwaną **LOO** (*Leave-One-Out*), albo **LOOCV** (*Leave-One-Out Cross-Validation*). Polega ona na wybieraniu po kolei pojedynczych próbek z posiadanego zbioru danych i użyciu jej do walidacji po przeprowadzeniu uczenia pozostałymi N-1 próbkami. Błąd jest uśredniany, podobnie jak w walidacji krzyżowej.

Zauważmy, że walidacja LOO jest równoważna walidacji krzyżowej przy k=N.

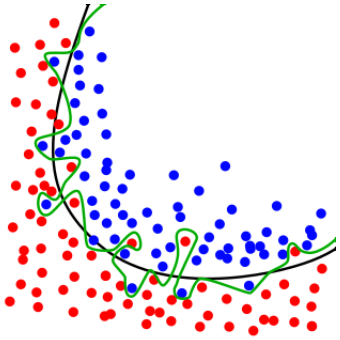
Błędy w uczeniu — przeuczenie

Przeuczenie (*overfitting*) jest jednym z najważniejszych problemów w uczeniu maszynowym. Objawia się w wielu postaciach i może prowadzić do drastycznie błędnych wyników. Typowym scenariuszem jest otrzymanie modelu, który wydaje się poprawny na zbiorze danych uczących. Jednak testowany na innym (testowym) zbiorze danych, daje znacznie większe błędy, albo wyniki kompletnie nieakceptowalne.

Jeden z możliwych scenariuszy jest taki, że uczenie całkowicie regularnego zjawiska jest prowadzone z niewielkiego zbioru danych, i otrzymany model jest zaburzony przez jakieś szczególne zjawiska, takie jak **obserwacje odstające** (*outliers*) (pojedyncze obserwacje silnie różniące się od reszty populacji) albo przypadkowością (np. podzbiór wykazujący pewne cechy wspólne, ale również nietypowe).

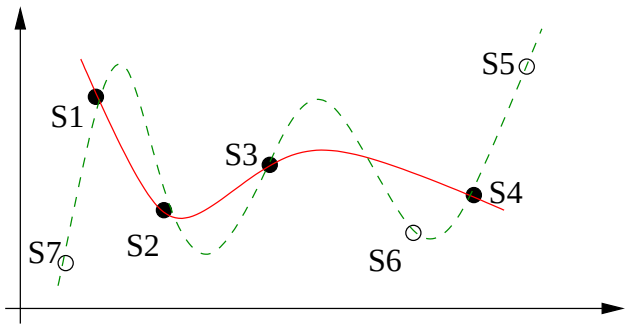
Inną możliwością jest szkolenie modelu tak intensywnie, że w końcu odzwierciedla nie tylko regularne właściwości, ale także szum zawarty w zbiorze danych. Jest to możliwe zwłaszcza w przypadku korzystania z bardzo zaawansowanych algorytmów maszynowego uczenia zdolnych do tworzenia bardzo złożonych i elastycznych modeli.

W tym przykładzie wydaje się, że czarna krzywa dobrze dopasowuje dany zbiór punktów. Jednak intensywne uczenie może doprowadzić do wygenerowania zielonej linii, która wykazuje zerowy błąd na zbiorze uczącym.



Błędy w uczeniu — niedouczenie

Jednak nie zawsze duże błędy uzyskane na danych testowych świadczą o przeuczeniu. Ogólnie, im większy jest zbiór uczący w stosunku do liczby atrybutów wejściowych i wielkości przestrzeni hipotez, tym mniejsze zagrożenie przeuczeniem. Duże błędy na zbiorze testowym, wraz z niewielkimi błędami na zbiorze uczącym, mogą równie dobrze być wynikiem **niedouczenia**, które może być skutkiem zbyt małego zbioru danych uczących, albo zbyt uproszczonego modelu, zastosowania nieadekwatnego algorytmu uczenia maszynowego, niedostatecznego uczenia, itp.



Uczenie się punktów S1,...,S4 pozwala wygenerować czerwoną krzywą ciągłą. Punkty testowe S5,S6,S7 dają znaczne błędy, i w tym przypadku jest to wynik niedouczenia. Ponowne uczenie z wszystkimi punktami daje zieloną krzywą przerywaną.

Określenie czy błędy danych testowych wynikają z przeuczenia czy niedouczenia, albo jeszcze z czegoś innego, stanowi trudny element uczenia maszynowego i często wymaga przeprowadzenia szeregu eksperymentów.

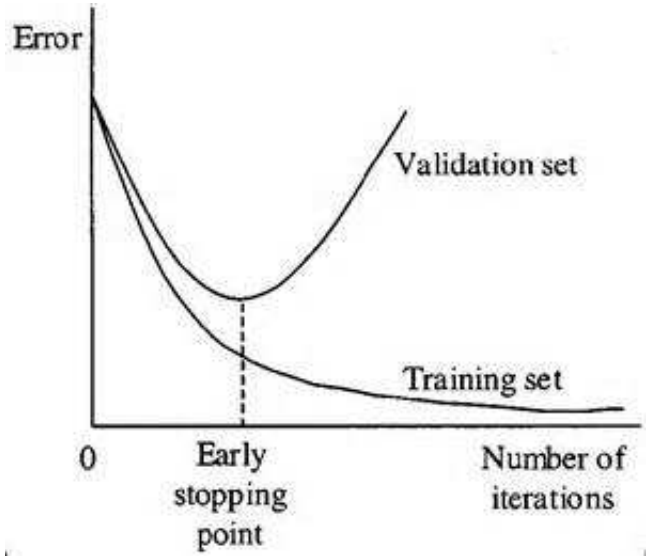
Błędy w uczeniu — wykrywanie przeuczenia

W uczeniu maszynowym opracowano wiele podejść do walki z przeuczeniem. Niektóre z nich są specyficzne i/lub wbudowane w określony algorytm ML, podczas gdy inne są ogólne i mogą być stosowane uniwersalnie. Tutaj poznamy kilka z nich.

Prostym i często skutecznym podejściem jest pozyskanie większej ilości danych dobrej jakości. Jak widzieliśmy, jednym z objawów przeuczenia jest obciążony model uzyskany na zaszumionym zbiorze danych. Przy większym zbiorze danych, nawet z takim samym szumem, wiele algorytmów ML będzie w stanie stworzyć modele lepszej jakości, ponieważ w dużym zbiorze szumy często mają tendencję do znoszenia się.

Inną metodą unikania przeuczenia jest **wczesne zatrzymanie** (*early stopping*).

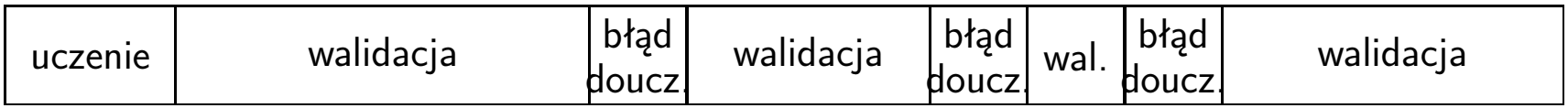
Działa ono w trakcie iteracyjnego ulepszania modelu, dzięki obserwacji jego efektywności przez porównanie błędów obliczonych na zbiorze treningowym i drugim zbiorze walidacyjnym. Każda runda optymalizacji klasyfikatora zmniejsza oba błędy. Jest rzeczą naturalną i oczekiwaną, że błąd walidacji będzie nieco większy, ale też malejący, w ślad za błędem zbioru uczącego. Jednak moment, gdy błąd walidacji przestaje się zmniejszać i zaczyna rosnąć, podczas gdy błąd zbioru uczącego nadal maleje, wskazuje, że wystąpiło przeuczenie.



Uczenie i walidacja

Uczenie maszynowe typowo wykonywane jest na przemian z walidacją, która służy do oceny skuteczności procesu uczenia, i jednocześnie wykrywania przeuczenia. Standardowym podejściem jest zastosowanie walidacji krzyżowej, szczególnie skutecznym gdy dostępnych danych jest względnie mało.

Alternatywnie, można zastosować następujące podejście **okna danych**, przydatne zwłaszcza gdy danych jest bardzo dużo. Wtedy można użyć tylko niewielkiej części danych do uczenia, a resztę wykorzystać do walidacji.



Gdy walidacja w pewnym momencie zawodzi, uczenie jest ponawiane dla zbioru uczącego z dodatkiem próbek dających błędy. Po tym, walidacja jest wznawiana na pozostałych danych.

Naiwny klasyfikator bayesowski

Jedną z najciekawszych metod maszynowego uczenia, na pewno jedną z najprostszych, a jednocześnie bardzo skuteczną metodę Naiwnego klasyfikatora bayesowskiego (*Naive Bayes Classifier, NBC*) przedstawimy na przykładzie. Dla zestawu próbek obliczamy następujące ułamki dla każdej z cech:

X_1	X_2	X_3	X_4	Y
0	1	1	0	1
0	0	1	1	1
1	0	1	0	1
0	0	1	1	1
0	0	0	0	1
1	0	0	1	0
1	1	0	1	0
1	0	0	0	0
1	1	0	1	0
1	0	1	1	0

$R_1(1,1) = 1/5$ - ułamek próbek pozytywnych, które mają $X_1 = 1$

$R_1(0,1) = 4/5$ - ułamek próbek pozytywnych, które mają $X_1 = 0$

$R_1(1,0) = 5/5$ - ułamek próbek negatywnych, które mają $X_1 = 1$

$R_1(0,0) = 0/5$ - ułamek próbek negatywnych, które mają$X_1 = 0$

$R_1(1,1) = 1/5 \quad R_1(0,1) = 4/5$
$R_1(1,0) = 5/5 \quad R_1(0,0) = 0/5$

$R_2(1,1) = 1/5 \quad R_2(0,1) = 4/5$
$R_2(1,0) = 2/5 \quad R_2(0,0) = 3/5$

$R_3(1,1) = 4/5 \quad R_3(0,1) = 1/5$
$R_3(1,0) = 1/5 \quad R_3(0,0) = 4/5$

$R_4(1,1) = 2/5 \quad R_4(0,1) = 3/5$
$R_4(1,0) = 4/5 \quad R_4(0,0) = 1/5$

Obliczone ułamki odpowiadają prawdopodobieństwom, że próbka z konkretną wartością klasy będzie miała określoną wartość atrybutu.

$$
\begin{array}{llll}
R_1(1,1) = 1/5 & R_2(1,1) = 1/5 & R_3(1,1) = 4/5 & R_4(1,1) = 2/5 \\
R_1(1,0) = 5/5 & R_2(1,0) = 2/5 & R_3(1,0) = 1/5 & R_4(1,0) = 4/5 \\
R_1(0,1) = 4/5 & R_2(0,1) = 4/5 & R_3(0,1) = 1/5 & R_4(0,1) = 3/5 \\
R_1(0,0) = 0/5 & R_2(0,0) = 3/5 & R_3(0,0) = 4/5 & R_4(0,0) = 1/5
\end{array}
$$

Aby przewidzieć wartość klasy nowej próbki, sprawdzamy jej wartości atrybutów, i obliczamy prawdopodobieństwa posiadania każdej wartości klasy przez pomnożenie prawdopodobieństw posiadania określonej wartości atrybutu z tą wartością klasy.

Dodatkowo, uwzględniamy pierwotne prawdopodobieństwo danej wartości klasy, np.:

$$
\begin{aligned}
\text{nowe } X &= \langle 0,0,1,1 \rangle \\
S(1) &= P(1) * R_1(0,1) * R_2(0,1) * R_3(1,1) * R_4(1,1) = 0.1024 \\
S(0) &= P(0) * R_1(0,0) * R_2(0,0) * R_3(1,0) * R_4(1,0) = 0
\end{aligned}
$$

Ponieważ $S(1) > S(0)$ określamy wartość klasy nowej próbki jako 1.

Bardziej formalnie, obliczamy następujące wartości R_j dla każdego atrybutu X_j przez zliczanie próbek i z określonymi wartościami X_j^i i Y^i:

$$\begin{aligned} R_j(1,1) &= \frac{\#(X_j^i = 1 \wedge Y^i = 1)}{\#(Y^i = 1)} \\ R_j(0,1) &= 1 - R_j(1,1) \end{aligned}$$

$$\begin{aligned} R_j(1,0) &= \frac{\#(X_j^i = 1 \wedge Y^i = 0)}{\#(Y^i = 0)} \\ R_j(0,0) &= 1 - R_j(1,0) \end{aligned}$$

Następnie, mając nową próbkę X^k, obliczamy:

$$\begin{aligned} S(1) &= P(1) * \prod_j R_j(X_j^k, 1) \\ S(0) &= P(0) * \prod_j R_j(X_j^k, 0) \end{aligned}$$

Odpowiedź klasyfikatora jest 1 iff $S(1) > S(0)$

Pracę klasyfikatora można znacznie usprawnić stosując logarytmy, ponieważ łatwiej jest dodawać niż mnożyć małe liczby. Obliczenie logarytmów jest konieczne tylko jednorazowo, w czasie budowy klasyfikatora.

$$\begin{aligned} \log S(1) &= \log P(1) + \sum_j \log R_j(X_j^k, 1) \\ \log S(0) &= \log P(0) + \sum_j \log R_j(X_j^k, 0) \end{aligned}$$

Teraz odpowiedź klasyfikatora jest 1 iff $\log S(1) > \log S(0)$

Poprawki Laplace'a

Powyższy schemat działa dobrze, chyba że w przypadku jednego z atrybutów nie ma próbek z pewną wartością atrybutu i klasy. Wtedy odpowiedni współczynnik R_j wynosi 0, i ta klasa nigdy nie będzie wybrana dla tej wartości atrybutu, niezależnie od innych atrybutów.

Można tego uniknąć dodając pewną stałą wartość liczbową l do każdej liczby próbek (w liczniku wyrażenia R_j). W celu znormalizowania wynikowego prawdopodobieństwa, każdy mianownik R_j zwiększamy odpowiednio o $2l$. Taką operacja nazywa się **wygładzaniem** (*smoothing*) estymaty, a wartość l stanowi siłę tego wygładzania. Często stosowany przypadek l=1 nazywa się wygładzaniem Laplace'a.

$$\begin{aligned} R_j(1,1) &= \frac{\#(X_j^i = 1 \wedge Y^i = 1) + 1}{\#(Y^i = 1) + 2} \\ R_j(0,1) &= 1 - R_j(1,1) \end{aligned}$$

$$\begin{aligned} R_j(1,0) &= \frac{\#(X_j^i = 1 \wedge Y^i = 0) + 1}{\#(Y^i = 0) + 2} \\ R_j(0,0) &= 1 - R_j(1,0) \end{aligned}$$

X_1	X_2	X_3	X_4	Y
0	1	1	0	1
0	0	1	1	1
1	0	1	0	1
0	0	1	1	1
0	0	0	0	1
1	0	0	1	0
1	1	0	1	0
1	0	0	0	0
1	1	0	1	0
1	0	1	1	0

$R_1(1,1) = 2/7 \quad R_1(0,1) = 5/7$

$R_1(1,0) = 6/7 \quad R_1(0,0) = 1/7$

$R_2(1,1) = 2/7 \quad R_2(0,1) = 5/7$

$R_2(1,0) = 3/7 \quad R_2(0,0) = 4/7$

$R_3(1,1) = 5/7 \quad R_3(0,1) = 2/7$

$R_3(1,0) = 2/7 \quad R_3(0,0) = 5/7$

$R_4(1,1) = 3/7 \quad R_4(0,1) = 4/7$

$R_4(1,0) = 5/7 \quad R_4(0,0) = 2/7$

Naiwny klasyfikator bayesowski — teoria

Spróbujmy wyprowadzić pół-formalne uzasadnienie powyższej procedury. Możemy uważać atrybuty $X_1, \ldots, X_n$ za zmienne losowe. Próbujemy na podstawie serii próbek nauczyć się prawdopodobieństwa $P(Y = 1|X_1, \ldots, X_n)$. Następnie, biorąc pod uwagę nową próbkę, używamy wyuczonej dystrybucji do obliczenia prawdopodobieństwa, że Y ma wartość 1. Jeśli to prawdopodobieństwo wynosi > 0.5, jako klasę wytypujemy 1, w przeciwnym razie 0.

Zatem musimy oszacować rozkład $P(Y = 1|X_1, \ldots, X_n)$ z danych treningowych. W metodzie Naiwnego klasyfikatora bayesowskiego zrobimy to wykorzystując regułę Bayesa.

Naiwne założenie

Reguła Bayesa w ogólnym przypadku:

$$P(A|B) = P(B|A)\frac{P(A)}{P(B)}$$

W szczególności z n atrybutami:

$$P(Y = 1|X_1...X_n) = P(X_1...X_n|Y = 1)\frac{P(Y = 1)}{P(X_1...X_n)}$$

Wyrażenie $P(X_1...X_n)$ jest charakterystyczne dla całego zbioru danych, a dla danej próbki jest stałe. Zatem dla celów wyznaczenia klasy próbki wystarczy obliczyć:

$$P(Y = 1) * P(X_1...X_n|Y = 1)$$

Zakładając, że wszystkie atrybuty są niezależne, możemy przyjąć:

$$P(X_1, ...X_n|Y = 1) = \prod_j P(X_j|Y = 1)$$

Takie założenie jest często fałszywe, ale ta metoda działa dobrze w wielu domenach. Dlatego klasyfikator nazywa się Naiwnym.

Regresja logistyczna: wstęp

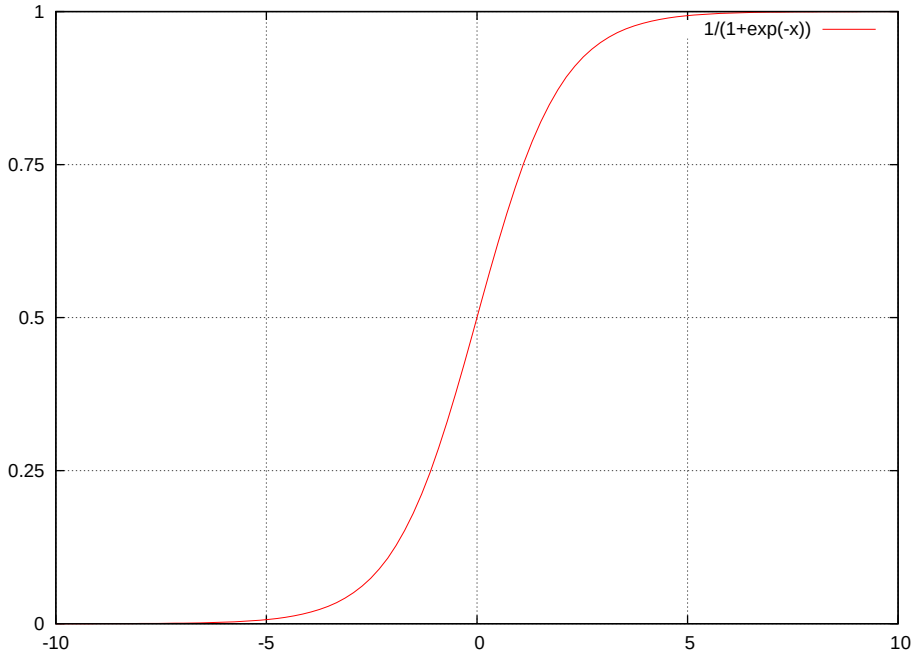
Regresja logistyczna jest formą klasyfikacji. Jest to podejście do uczenia się funkcji postaci $f : X \to Y$, lub $P(Y|X)$ w przypadku, gdy Y jest wartością dyskretną, a $X = \langle X_1 ... X_n \rangle$ jest wektorem zawierającym zmienne dyskretne lub ciągłe.

Do modelowania rozkładu prawdopodobieństwa wykorzystujemy **funkcję logistyczną**. „Esowaty" kształt tej funkcji nazywany jest także **krzywą sigmoidalną**. Jest ona dana następującym równaniem:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

gdzie: x_0 to wartość x punktu środkowego sigmoidy, L jest maksymalną wartością funkcji, a k daje nachylenie krzywej.

```
$ gnuplot
set grid ytics lt 0 lw 1 lc rgb "#444444"
set grid xtics lt 0 lw 1 lc rgb "#444444"
set ytics .25
plot 1/(1+exp(-x))
```

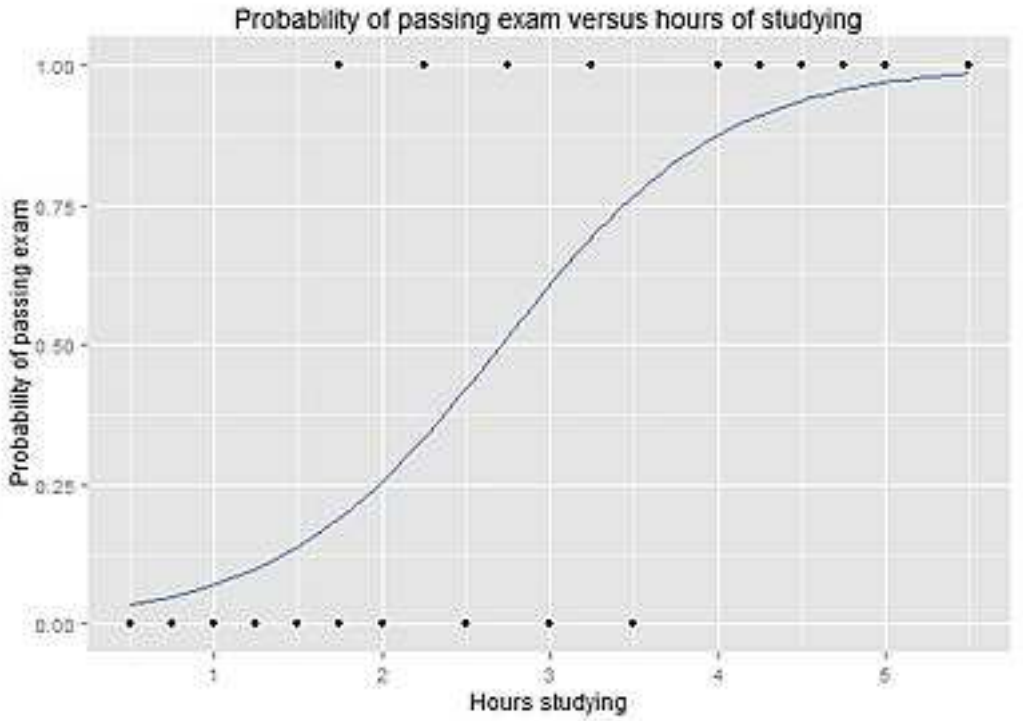


Regresja logistyczna: przykład

Rozważmy następujący przykład (Wikipedia): grupa 20 studentów spędziła od 0 do 6 godzin ucząc się do egzaminu; potem niektórzy zdali a niektórzy oblali.

Jakie jest prawdopodobieństwo zdania egzaminu przy danej liczbie godzin nauki?

hours	pass	hours	pass
0.50	n	2.75	t
0.75	n	3.00	n
1.00	n	3.25	t
1.25	n	3.50	n
1.50	n	4.00	t
1.75	n	4.25	t
1.75	t	4.50	t
2.00	n	4.75	t
2.25	t	5.00	t
2.50	n	5.50	t



Regresja logistyczna: teoria

Dla uproszczenia najpierw rozważymy przypadek, w którym Y jest zmienną binarną. Później rozszerzymy prezentację na przypadek wielu wartości dyskretnych.

Regresja logistyczna przyjmuje następującą parametryczną dystrybucję $P(Y|X)$:

$$P(Y=1|X) = \frac{1}{1+\exp(w_0+\Sigma_{i=1}^{n} w_i X_i)}$$

$$P(Y=0|X) = \frac{\exp(w_0+\Sigma_{i=1}^{n} w_i X_i)}{1+\exp(w_0+\Sigma_{i=1}^{n} w_i X_i)}$$

Parametry w_i zostaną wyuczone z danych treningowych.

Regresja logistyczna: teoria (2)

Ponieważ wyuczona funkcja logistyczna będzie wykorzystana do klasyfikacji, interesuje nas jedynie następujący warunek:

$$\frac{P(Y=0|X)}{P(Y=1|X)} > 1$$

Jeśli warunek jest spełniony, określamy $Y = 0$. Po podstawieniu ogólnych wzorów parametrycznych powyższy warunek przybiera postać:

$$\exp(w_0 + \sum_{i=1}^{n} w_i X_i) > 1$$

a po zastosowaniu logarytmu naturalnego do obu stron staje się:

$$w_0 + \sum_{i=1}^{n} w_i X_i > 0$$

Regresja logistyczna: uczenie parametrów

Jedno możliwe podejście do uczenia parametrów regresji logistycznej polega na wybraniu wartości, które maksymalizują warunkowe prawdopodobieństwo danych uczących. Jest ono prawdopodobieństwem wystąpienia zaobserwowanych wartości Y w danych treningowych, uwarunkowane odpowiadającymi im wartościami X. Zatem wybieramy parametry W, które spełniają:

$$W \leftarrow \arg\max_W \prod_s P(Y^s|X^s, W)$$

gdzie $W = \langle w_0, w_1 ... w_n \rangle$ jest wektorem parametrów, które mają być oszacowane, a Y^s, X^s oznaczają zaobserwowane wartości Y, X w s-ej próbce uczącej. Wyrażenie po prawej stronie arg max jest warunkowym prawdopodobieństwem wystąpienia danych. W występuje w części warunkowej dla podkreślenia, że wyrażenie, które chcemy maksymalizować, jest funkcją W.

Równoważnie, możemy pracować z logarytmem warunkowego prawdopodobieństwa:

$$W \leftarrow \arg\max_W \sum_s \ln P(Y^s|X^s, W)$$

Regresja logistyczna: uczenie parametrów (2)

To logarytmiczne prawdopodobieństwo warunkowe danych $\Sigma_s \ln P(Y^s|X^s, W)$, które będziemy oznaczać $l(W)$, można zapisać jako

$$l(W) = \sum_s Y^s \ln P(Y = 1|X^s, W) + (1 - Y^s) \ln P(Y^s = 0|X^s, W)$$

ponieważ Y^s może przyjmować jedynie wartości 0 lub 1, tylko jeden z dwóch składników sumy będzie niezerowy dla danego Y^s. To można przepisać jako:

$$l(W) = \sum_s Y^s \ln \frac{P(Y^s = 1|X^s, W)}{P(Y^s = 0|X^s, W)} + \ln P(Y^s = 0|X^s, W)$$

i przez włączenie parametrycznego wzoru na P otrzymujemy:

$$l(W) = \sum_s Y^s (w_0 + \sum_i^n w_i X_i^s) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^s))$$

Regresja logistyczna: uczenie parametrów (3)

Niestety, nie ma zamkniętej formuły rozwiązania maksymalizującego $l(W)$ względem W. Na szczęście funkcja $l(W)$ jest **wklęsła** względem W. Dlatego naturalnym podejściem jest wykorzystanie przeszukiwania gradientowego w przestrzeni częściowych pochodnych $l(W)$. Składnik i-ty wektora gradientu ma postać:

$$\frac{\partial l(W)}{\partial w_i} = \sum_s X_i^s (Y^s - \hat{P}(Y^s = 1 | X^s, W))$$

gdzie $\hat{P}(Y^s | X^s, W)$ jest regresją logistyczną poszukiwanego prawdopodobieństwa. Aby uwzględnić wagę w_0, wprowadzamy sztuczną zmienną $X_0 = 1$ dla wszystkich s.

To wyrażenie na pochodną ma intuicyjną interpretację: wyrażenie wewnątrz nawiasów jest po prostu błędem przewidywania; to jest różnica między obserwowanym Y^s a jego przewidywanym prawdopodobieństwem! Zauważ: jeśli $Y^s = 1$ to chcemy aby $\hat{P}(Y^s = 1 | X^s, W)$ było bliskie 1, a jeśli $Y^s = 0$ wtedy wolimy, żeby $\hat{P}(Y^s = 1 | X^s, W)$ było równe 0 (czyli $\hat{P}(Y^s = 0 | X^s, W)$ równe 1). Ten czynnik błędu jest mnożony przez wartość X_i, określającą wielkość $w_i X_i$ w tej predykcji.

Regresja logistyczna: uczenie parametrów (4)

Aby przeprowadzić standardowe przeszukiwanie gradientowe dla zoptymalizowania wagi W, możemy zacząć od początkowych wag równych zero i wielokrotnie aktualizować wagi w kierunku gradientu, każdorazowo zmieniając każdą wagę w_i zgodnie z:

$$w_i \leftarrow w_i + \eta \sum_s X_i^s (Y^s - \hat{P}(Y^s = 1 | X^s, W))$$

gdzie η jest małą stałą (np. 0.01) określającą rozmiar kroku. Ponieważ logarytmiczne prawdopodobieństwo $l(W)$ jest funkcją wklęsłą w W, ta procedura przeszukiwania gradientowego będzie zbieżna do globalnego maksimum.

Regresja logistyczna: regularyzacja

Regularyzacja to technika stosowana w wielu algorytmach statystycznych dla eliminacji przeuczenia, które jest możliwe zwłaszcza w przypadku wielowymiarowych, ale nielicznych danych. Regularyzacja działa poprzez wprowadzenie dodatkowego wyrażenia w formułach jako kary dla dużych wartości W, przy założeniu, że duże współczynniki występują w wysoce przeuczonych funkcjach. Jednym z możliwych podejść jest karanie funkcji wiarygodności logarytmicznej:

$$W \leftarrow \arg\max_W \sum_s \ln P(Y^s|X^s, W) - \frac{\lambda}{2}|W|^2$$

przez dodanie kary proporcjonalnej do kwadratu wielkości W. λ określa siłę tego karnego składnika. Taki cel maksymalizacji odpowiada wyznaczaniu estymaty MAP (Maximum A Posteriori) dla W przy założeniu rozkładu normalnego (Gaussa) dla $P(W)$ ze średnią równą zero i wariancją (σ^2) związaną z $\frac{1}{\lambda}$.

Nie jest trudno wyprowadzić następującą regułę aktualizacji gradientowej, podobną do przypadku braku regularyzacji:

$$w_i \leftarrow w_i + \eta \sum_s X_i^s (Y^s - \hat{P}(Y^s = 1|X^s, W)) - \eta\lambda w_i$$

Regresja logistyczna: regularyzacja (2)

Powyższa procedura używa modelu normalnego (Gaussa) jako rozkładu $P(W)$, co prowadzi do tak zwanej **regularyzacji $\mathbf{L}_2$**. Dąży ona do minimalizacji kwadratowej normy $|W|^2$.

Innym sposobem jest regularyzacja L_1, która minimalizuje $|W|$.

Regresja logistyczna: funkcje wielowartościowe

W przypadku wielowartościowej funkcji Y, z pewną liczbą dyskretnych wartości $y_1, ..., y_K$, rozkład $P(Y = y_k|X)$ dla $Y = Y_1, Y = y_2, ..., Y = y_{K-1}$ ma postać:

$$P(Y = y_k|X) = \frac{\exp(w_{k0} + \Sigma_{i=1}^{n} w_{ki}X_i)}{1 + \Sigma_{j=1}^{K-1} \exp(w_{j0} + \Sigma_{i=1}^{n} w_{ji}X_i)}$$

a dla ostatniego $Y = y_K$:

$$P(Y = y_K|X) = \frac{1}{1 + \Sigma_{j=1}^{K-1} \exp(w_{j0} + \Sigma_{i=1}^{n} w_{ji}X_i)}$$

Tutaj w_{ji} oznacza wagę związaną z j-tą klasą $Y = y_j$ i z próbką X_i. Łatwo zauważyć, że wcześniejsze wyrażenia dla przypadku, w którym Y była wartością logiczną, są szczególnym przypadkiem powyższych wyrażeń. Zauważ także, że forma wyrażenia dla $P(Y = y_K|X)$ zapewnia $[\Sigma_{k=1}^{K} P(Y = y_k|X)] = 1$.

Podstawowa różnica między tymi wyrażeniami i tymi dla logicznego Y jest to, że gdy Y przyjmuje K możliwych wartości, tworzymy $K - 1$ różnych wyrażeń liniowych do modelowania rozkładów dla różnych wartości Y. Rozkład dla końcowej K-ej wartości Y jest po prostu jedynką minus suma prawdopodobieństw pierwszych $K - 1$ wartości.

Regresja logistyczna: funkcje wielowartościowe (2)

W przypadku funkcji wielowartościowej, reguła przeszukiwania gradientowego staje się:

$$w_{ji} \leftarrow w_{ji} + \eta \sum_s X_i^s (\delta(Y^s = y_j) - \hat{P}(Y^s = y_j | X^s, W))$$

gdzie $\delta(Y^s = y_j) = 1$, jeśli s-ta wartość, Y^s, jest równa y_j i $\delta(Y^s = y_j) = 0$ w przeciwnym przypadku. Wcześniejsza reguła uczenia jest szczególnym przypadkiem tej nowej reguły, gdy $K = 2$. Podobnie jak w przypadku dla $K = 2$, wyrażenie wewnątrz nawiasów może być interpretowane jako czynnik błędu, który zmierza do zera, jeśli oszacowane prawdopodobieństwo warunkowe $P(Y^s = y^j | X^s, W))$ idealnie pasuje do obserwowanej wartości Y^s.

Z uwzględnieniem regularyzacji powyższa formuła przybiera postać:

$$w_{ji} \leftarrow w_{ji} + \eta \sum_s X_i^s (\delta(Y^s = y_j) - \hat{P}(Y^s = y_j | X^s, W)) - \eta \lambda w_{ji}$$

Macierz pomyłek

Widzieliśmy, że podstawową miarą skuteczności pracy klasyfikatora może być Dokładność/Błąd. Jednocześnie widzieliśmy, że nie opisują one w pełni jakości pracy, oraz przydatności do danego problemu decyzyjnego, wyuczonego klasyfikatora.

Posiadanie wiarygodnej miary błędu klasyfikacji jest pożądane. Niestety, żadna pojedyncza statystyczna miara błędu nie jest wystarczająca. Dlatego w użyciu jest pewna liczba takich wskaźników. Większość z nich można obliczyć ze struktury zwanej **macierzą pomyłek** (*confusion matrix*), reprezentująca liczbę próbek poszczególnych klas sklasyfikowanych jako każda z klas. W ogólnym przypadku ma ona postać:

	sklasyfikowane jako klasa:			
	1	2	...	m
rzeczywista klasa: 1	$N_{1,1}$	$N_{1,2}$	...	$N_{1,m}$
rzeczywista klasa: 2	$N_{2,1}$	$N_{2,2}$	...	$N_{2,m}$
...	...	...	...	...
rzeczywista klasa: m	$N_{m,1}$	$N_{m,2}$	...	$N_{m,m}$

Widać że: Accuracy $= \frac{\sum_i N_{i,i}}{\sum_i \sum_j N_{i,j}}$

Macierz pomyłek klasyfikacji binarnej

W klasyfikacji binarnej klasyfikator powinien jedynie wybrać próbki pozytywne i pominąć negatywne. W tym przypadku stosuje się zatem terminologię:

$$\begin{aligned} \text{TP } (\textit{true positives}) &= N_{1,1} \\ \text{TN } (\textit{true negatives}) &= N_{0,0} \\ \text{FN } (\textit{false negatives}) &= N_{1,0} \\ \text{FP } (\textit{false positives}) &= N_{0,1} \end{aligned}$$

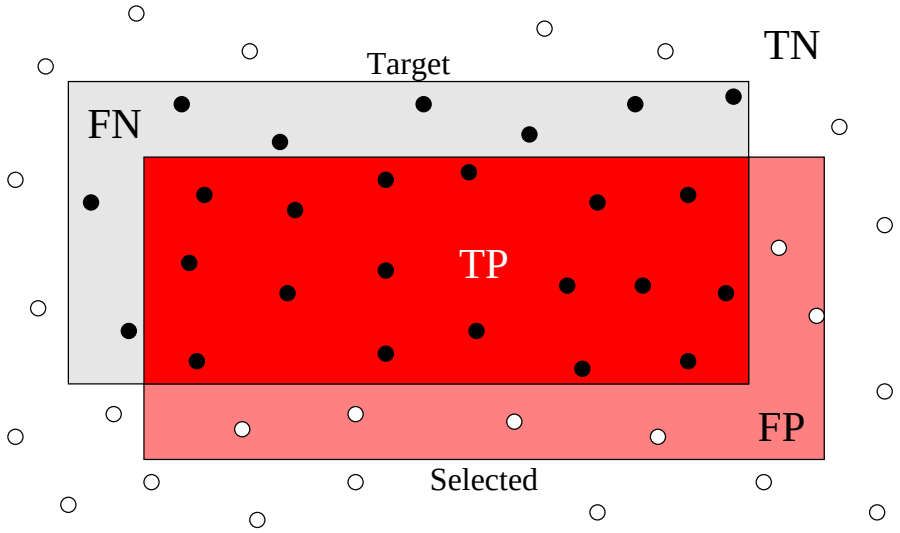
	sklasyfikowane jako: klasa 0	klasa 1
rzeczywista klasa 0	TN	FP
rzeczywista klasa 1	FN	TP

Widać że:

$$\text{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Precyzja i Czułość

Precyzja (*Precision*) jest miarą błędu pochodzącą z klasyfikacji binarnej i wyraża ułamek próbek oznaczonych przez klasyfikator jako pozytywne które rzeczywiście są pozytywne. **Czułość** (*Recall, Sensitivity*) jest ułamkiem pozytywnych przykładów, które są sklasyfikowane jako takie przez klasyfikator.



$$\text{Precyzja} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{Czułość} = \frac{\text{TP}}{\text{TP+FN}}$$

Częstym kompromisem pomiędzy Precyzją a Czułością jest, że maksymalizacja jednej może spowodować zmniejszenie drugiej i odwrotnie. Rozważmy chirurga mającego wyciąć guza z mózgu pacjenta (przykład z Wikipedii). Powinien on usunąć wszystkie komórki nowotworu, ponieważ pozostawienie części może spowodować jego odnowienie. Jednak nie może wyciąć żadnego fragmentu zdrowej tkanki, bo mogłoby to zaburzyć funkcje mózgu. Wycinając bardziej liberalnie, chirurg zwiększa czułość kosztem precyzji. Postępując bardziej konserwatywnie, zwiększa precyzję, ale pogarsza czułość.

Precyzja i Czułość (cd.)

Zauważmy, że Precyzja i Czułość są obliczane ściśle dla klasy przykładów pozytywnych; odnosi się ona do wybierania lub wytypowania elementów z populacji. Gdy poprawność wyboru przykładów negatywnych jest równie istotna, możemy obliczać Precyzję i Czułość dla klasy negatywnej.

W ogólnym przypadku wielu klas możemy obliczać Precyzję i Czułość dla każdej z klas:

$$\text{Precision(C)} = \frac{N_{C,C}}{\sum_i N_{i,C}} \qquad \text{Recall(C)} = \frac{N_{C,C}}{\sum_i N_{C,i}}$$

Precyzja klasyfikacji równa 1.0 dla klasy C oznacza, że każda próbka sklasyfikowana jako klasa C rzeczywiście należy do klasy C, ale nic nie mówi o elementach klasy C, które nie zostały prawidłowo sklasyfikowane.

Czułość 1.0 dla klasy C oznacza, że wszyscy jej członkowie są poprawnie sklasyfikowani jako C, ale nie mówi nic o tym, ile innych elementów było niepoprawnie również oznaczonych jako należący do klasy C.

W celu agregacji wielu takich wielkości dla wielu klas możemy obliczać średnią Precyzję lub Czułość ważoną przez liczność klas.

Statystyka $\hat{\kappa}$ (Kappa)

Parametr Kappa jest często określany jako **interrater agreement**. Odzwierciedla on zgodność dwóch sposobów klasyfikacji obiektów. W rzeczywistości określa, o ile lepsza niż losowa jest zgodność między dwoma klasyfikatorami. Jest obliczany według wzoru:

$$\hat{\kappa} = \frac{NT \, \Sigma_i \, N_{i,i} - \Sigma_i (NR_i \times NC_i)}{NT^2 - \Sigma_i (NR_i \times NC_i)}$$

gdzie:

$$\begin{aligned} NT &= \sum_i \sum_j N_{i,j} \quad \text{(całkowita liczba próbek)} \\ NR_i &= \sum_j N_{i,j} \quad \text{(całkowita liczba próbek w wierszu } i\text{)} \\ NC_i &= \sum_j N_{j,i} \quad \text{(suma próbek w kolumnie } i\text{)} \end{aligned}$$

Wartość Kappa może wynosić od -1 do 1, ale dla korelacji dodatniej jest > 0. Wartości powyżej 0.5 wskazują na znacząco poprawną klasyfikację.

Jest również w użyciu ważona wersja Kappa z wagami przypisanymi do każdej z klas.

Sąsiedztwo w przestrzeni cech

Mając wiele zapamiętanych punktów w przestrzeni cech, i nowy punkt z nieznaną klasą, możemy zbadać jego sąsiedztwo w przestrzeni.

Zlokalizujemy najbliższego sąsiada nowego punktu i użyjemy jego klasy dla nowego punktu. To jest algorytm **Najbliższego Sąsiada** (*Nearest Neighbor*).

Obliczanie odległości w przestrzeni cech

Dobrym pytaniem jest, jak określimy „najbliższego.” Potrzebujemy miary odległości dla przestrzeni cech. Typowe jest użycie odległości euklidesowej:

$$D(x_i, x_k) = \sqrt{\sum_j (x_{i_j} - x_{k_j})^2}$$

Jednak poszczególne wymiary przestrzeni odpowiadają różnym cechom, które mogą być różnymi wielkościami, wyrażonymi w różnych jednostkach, być może o różnych rzędach wielkości.

Skalowanie wejść

Aby przybliżyć wymiary w przestrzeni cech, można zastosować skalowanie. Typowym podejściem jest obliczenie wartości średniej $\overline{x}$ i odchylenia standardowego σ_x każdej cechy wejściowej x, a następnie przeskalowanie takiej cechy x jako:

$$x' = \frac{x - \overline{x}}{\sigma_x}$$

To jest uniwersalny schemat skalowania, zwany **normalizacją**. Zamiast niego można użyć jakiejś innej metody, na przykład w celu zwiększenia wpływu niektórych cech w stosunku do innych. Można to uzyskać eksperymentalnie, procedurą poszukiwania gradientowego (hill-climbing) z walidacją krzyżową, do zdobycia wektora wag cech, który najlepiej sprawdza się dla danego zbioru danych.

Przykład bankructwa jeszcze raz

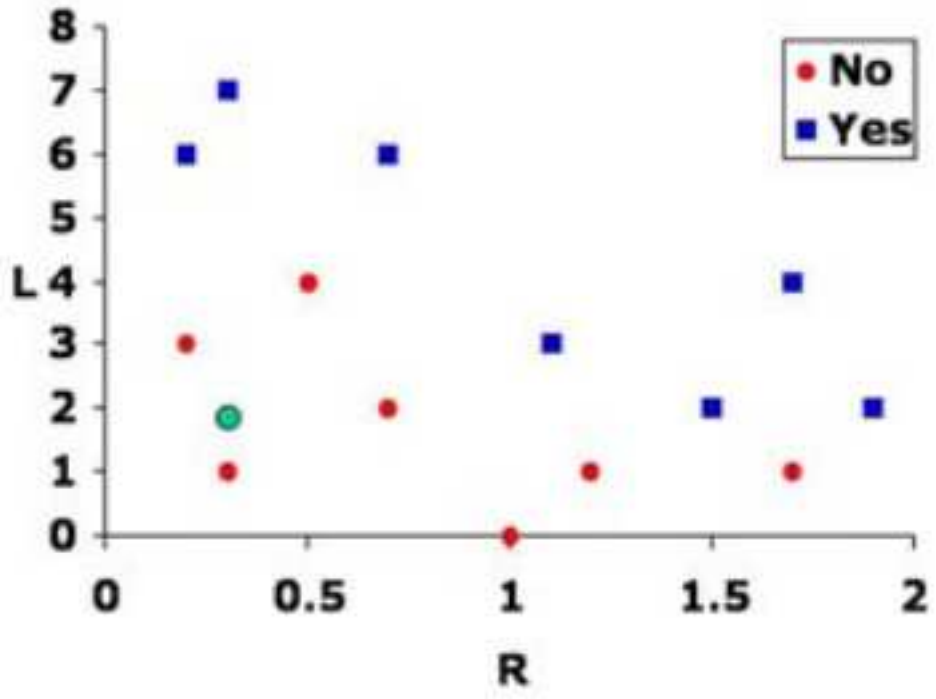
Załóżmy, że taką procedurę przeprowadzono w odniesieniu do zbioru danych dotyczących bankructwa rozważanego wcześniej, a wynikiem było to, że cecha R (stopa wydatków względem dochodu) powinna być przeskalowana $5\times$, w stosunku do L (liczba spóźnionych spłat miesięcznych kredytu w ciągu roku).

$$D(x_i, x_k) = \sqrt{(L_{x_i} - L_{x_k})^2 + (5R_{x_i} - 5R_{x_k})^2}$$

(W przybliżeniu odpowiada to skali osi na obrazku poniżej.)

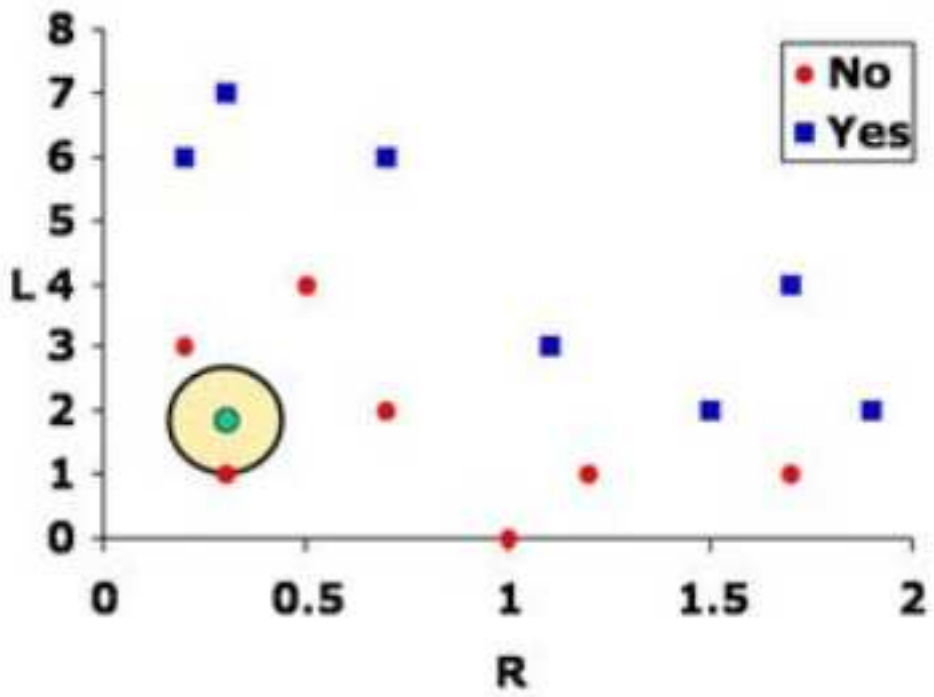
Załóżmy, że jest nowa osoba do sklasyfikowania, z $R = 0.3$ i $L = 2$.

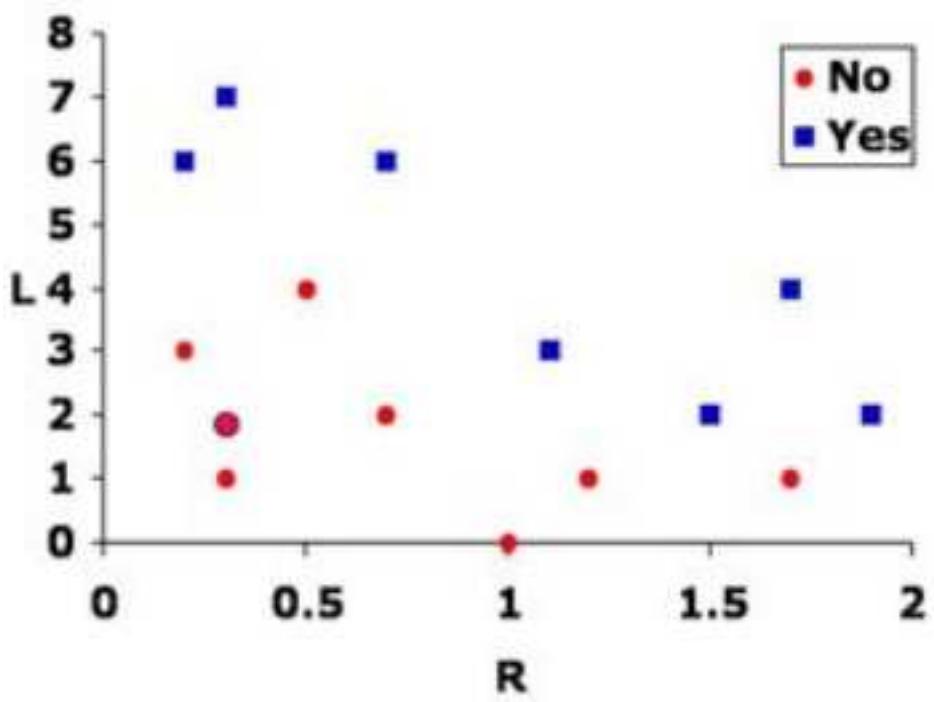
Jaka powinna być wartość jej klasy?



Przykład bankructwa jeszcze raz (cd.)

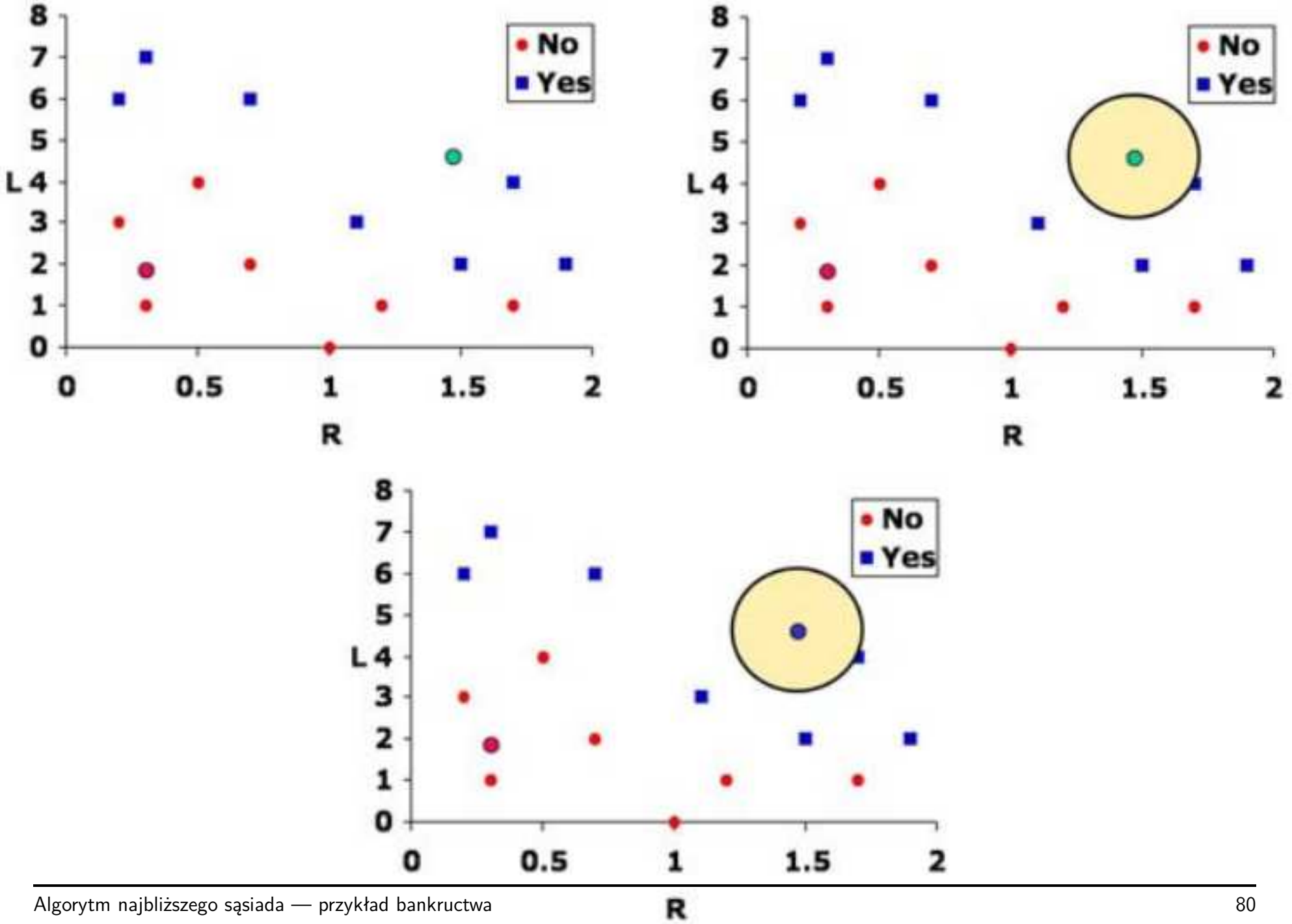
Możemy obliczyć najbliższego sąsiada ...





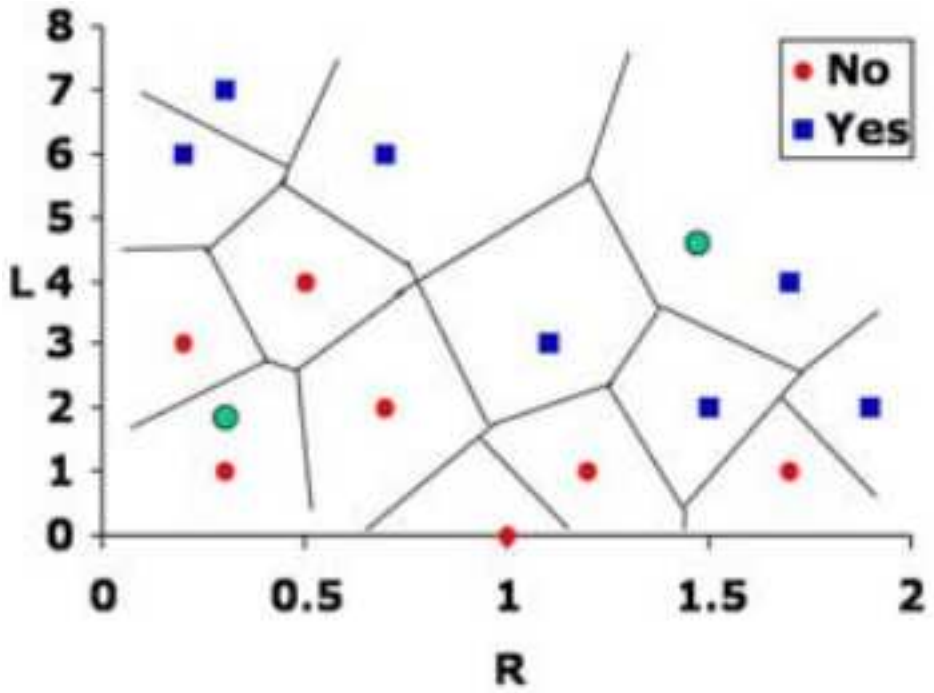
... i wyciągnąć wniosek, że klasa nowej osoby powinna być czerwona (brak bankructwa).

I podobnie dla innej nowej próbki:



Klasyfikacja najbliższego sąsiada

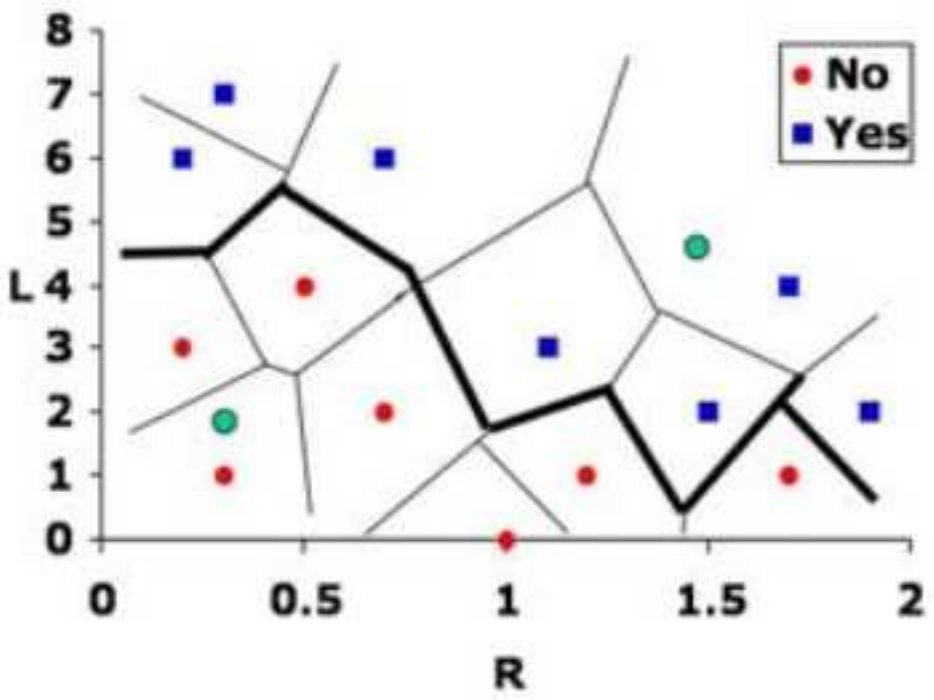
Przedstawiony tu algorytm najbliższego sąsiada efektywnie dzieli przestrzeń cech na regiony:



Podział (dwuwymiarowej) przestrzeni cech segmentami linii równoodległych od zbioru punktów nazywa się **diagramem Voronoia**.

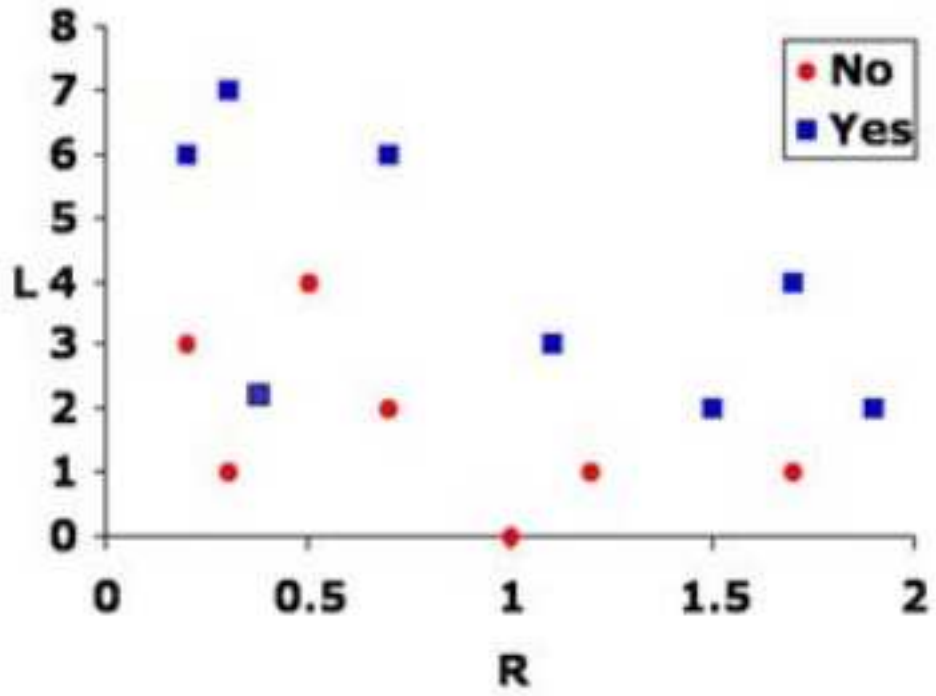
Czas i pamięć

Pamiętając wszystkie próbki ze zbioru danych, który może być bardzo duży, a następnie obliczenie odległości od wszystkich tych punktów dla dowolnej nowej próbki może być intensywne obliczeniowo. Dobra struktura danych (drzewo K-D) pozwala na szybkie obliczenia, średnio $O(\log(m) * n)$ z n cechami i m próbkami w zbiorze uczącym. Próbki uczące odległe od linii granicznych (lub k-powierzchni) można dodatkowo pominąć, zmniejszając zużycie pamięci.



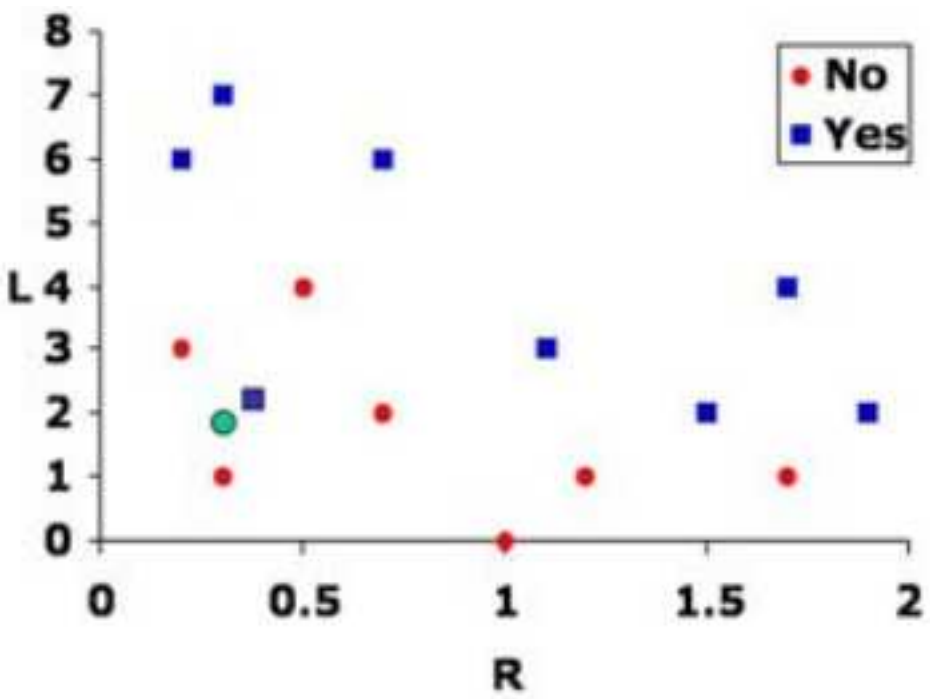
Szumy

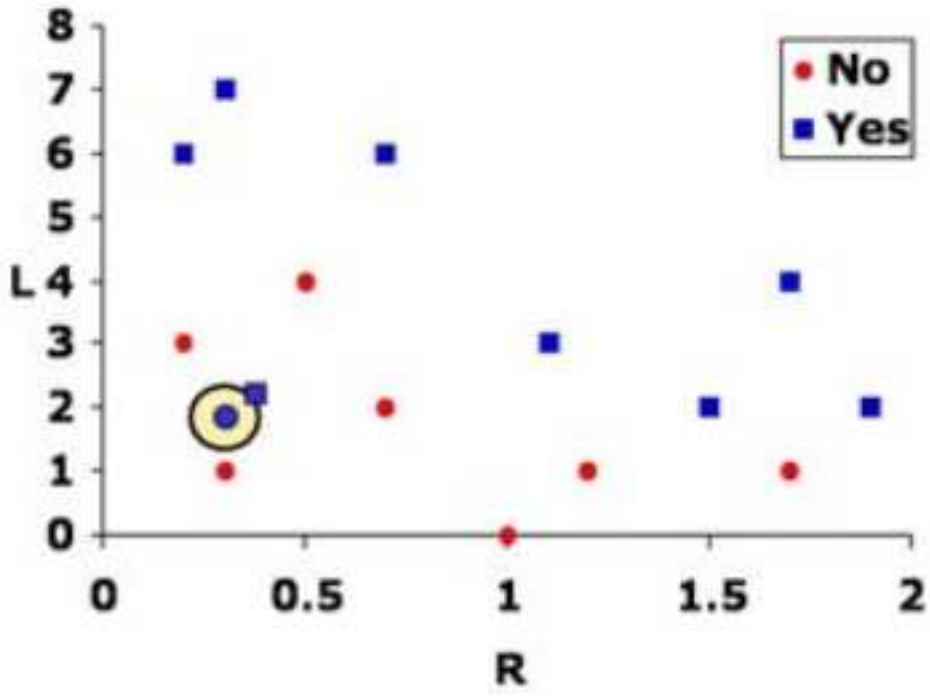
Gdy zbiór danych uczących jest wiarygodny, granica pomiędzy dwoma klasami wyjściowymi jest dobrze określona. Jednak powstaje problem, gdy zbankrutowała osoba z dobrymi parametrami finansowymi.



Taki punkt można potraktować dwojako: albo przestrzegamy go ściśle, i bierzemy pod uwagę w przyszłej klasyfikacji, albo traktujemy jako nietypowy przypadek, który należy zignorować. Jednak w rzeczywistym zbiorze danych, z wieloma cechami i wieloma wartościami, możemy nie być w stanie rozpoznać takich elementów zestawu uczącego.

Szumy — klasyfikowanie nowej próbki

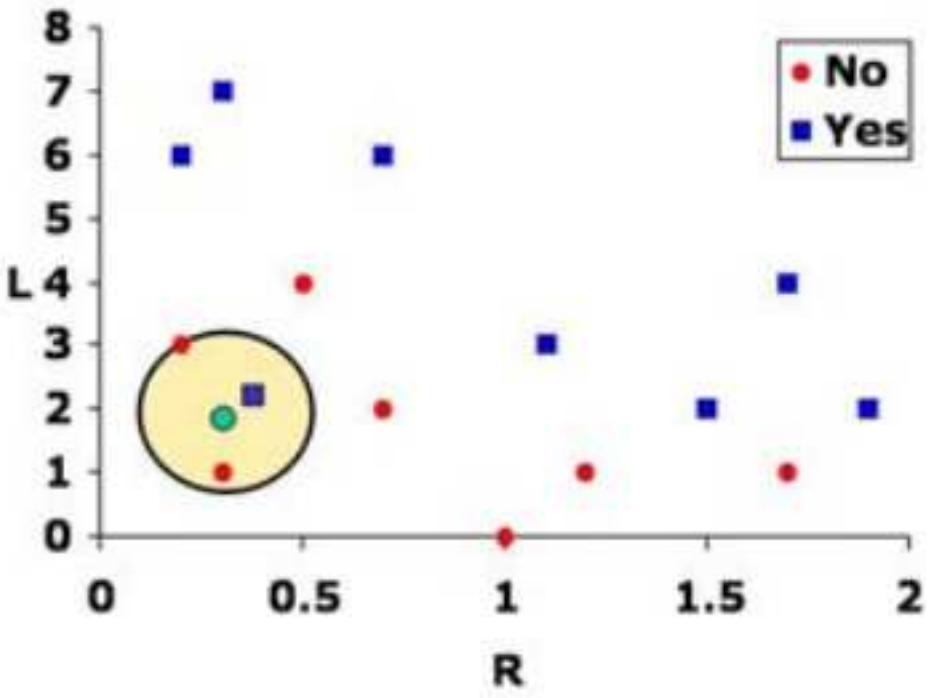


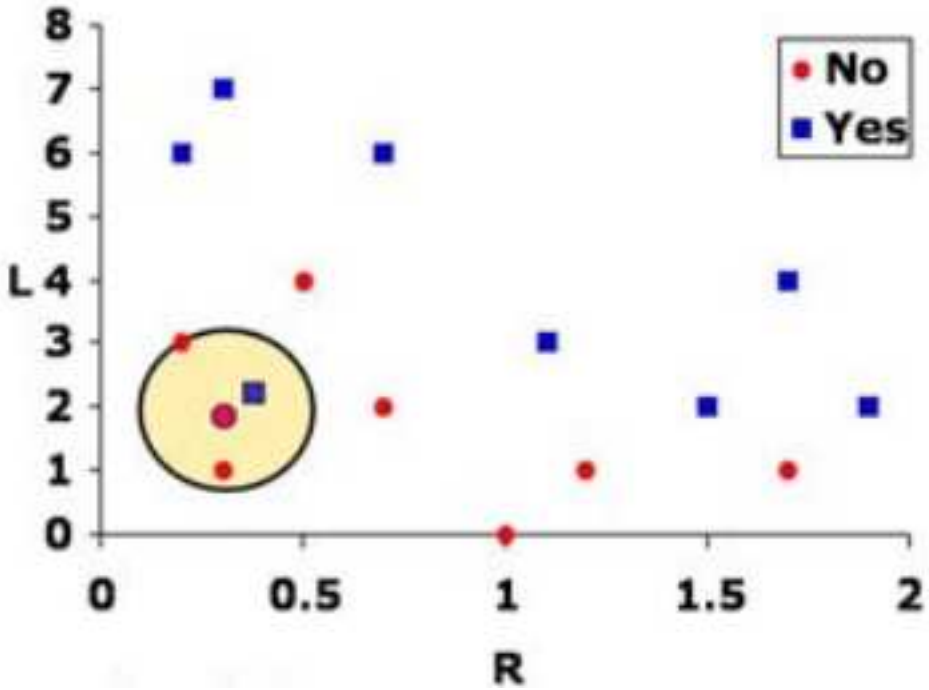


Standardowy algorytm najbliższego sąsiada może sklasyfikować nowy punkt na podstawie próbki uczącej stanowiącej „szum.”

Algorytm k-Najbliższych Sąsiadów

Proste rozszerzenie algorytmu uwzględnia pewną liczbę k najbliższych punktów i wyznacza klasę większości elementów.





Przy większych wartościach k klasyfikator staje się bardziej odporny na szumy, ale mniej dokładny. Można określić optymalną wartość k eksperymentalnie za pomocą walidacji krzyżowej. Z oczywistych względów praktycznych ma sens używanie nieparzystych wartości k: 3, 5, 7, 9, ...

Klątwa wymiarowości

Algorytm najbliższego sąsiada działa bardzo dobrze i jest jednym z najskuteczniejszych algorytmów uczenia maszynowego.

Czynnikiem ograniczającym tej metody są wielowymiarowe przestrzenie cech. W wysokich wymiarach, prawie wszystkie punkty są odległe od siebie, jeśli nie względem jednego to innego wymiaru. To zjawisko nazywa się **klątwą wymiarowości** (*curse of dimensionality*), często spotykaną w uczeniu maszynowym.

Klątwę wymiarowości można intuicyjnie wyjaśnić następująco. Wyobraźmy sobie, że $\mathcal{D}$-wymiarowa przestrzeń cech jest jednostkowym hiper-sześcianem z równomiernie rozłożonymi próbkami zbioru uczącego. Aby przypisać klasę nowej próbce rozszerzamy równomiernie mały hiper-sześcian cech wokół tego punktu, do momentu, aż będzie on zawierał określony ułamek p punktów zbioru uczącego. Wartość oczekiwana rozmiaru boku tego hiper-sześcianu wyniesie $e_{\mathcal{D}}(p) = p^{1/\mathcal{D}}$.

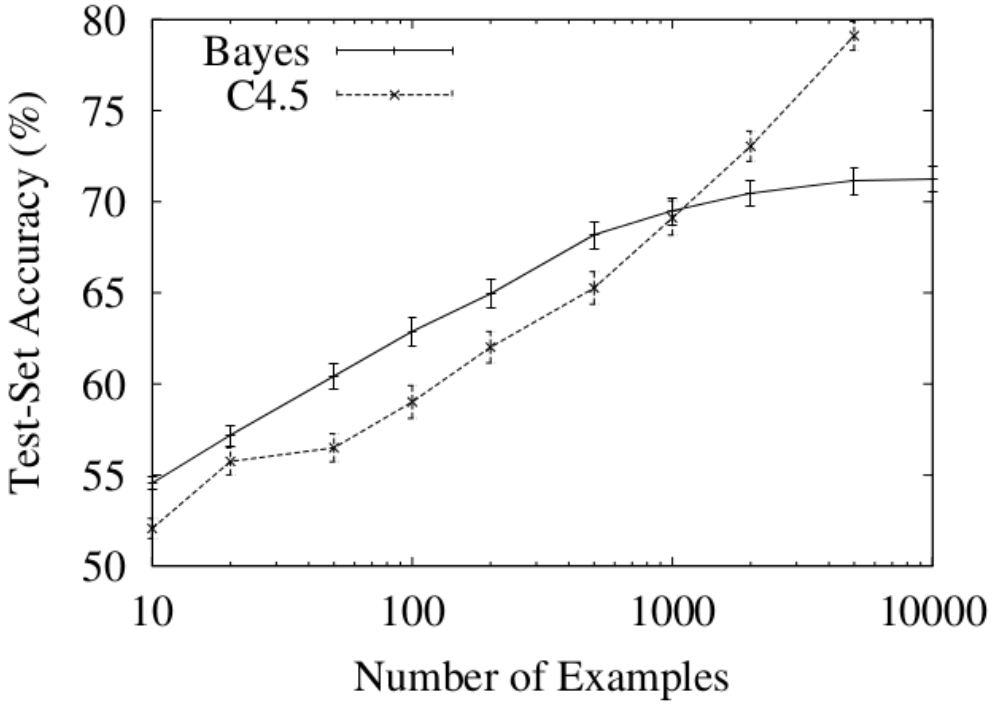
Dla $\mathcal{D}$=10 i p=1% mamy $e_{10}(0.01) = 0.63$, dla jednostkowego hiper-sześcianu. Oznacza to, że metoda najbliższego sąsiada nie jest już wcale lokalna, skoro wymaga sprawdzania punktów tak odległych (praktycznie, w drugiej połówce hiper-sześcianu). Punkty te mogą nie być dobrymi predyktorami dla nowo klasyfikowanego punktu.

Klątwa wymiarowości — redukcja wymiaru

Ze względu na klątwę wymiarowości często pożądane jest zmniejszenie wymiarowości przestrzeni, przez eliminację części cech. Może to się wydawać szkodliwe, ponieważ wiele cech może zawierać ważne informacje o próbkach. Jednak jest to zgodne z zasadą brzytwy Ockhama: jeśli krótszy wektor cech działa równie dobrze (lub prawie tak samo dobrze), to należy go użyć. Daje to szybsze uczenie, bardziej wiarygodną klasyfikację i umożliwia łatwiejszą wizualizację lub interpretację procesu klasyfikacji.

Silniejszy algorytm nie zawsze jest lepszy

Wykres pokazuje wyniki eksperymentu, w którym dane oryginalnie sklasyfikowane klasyfikatorem regułowym został następnie poddane uczeniu naiwnym klasyfikatorem bayesowskim oraz zaawansowanym algorytmem regułowym C4.5rules.



Mimo iż oryginalnym klasyfikatorem jest zestaw reguł, to nawet przy 1000 próbkach naiwny klasyfikator bayesowski jest lepszy niż algorytm uczący się reguł. Dzieje się tak pomimo fałszywego założenia bayesowskiego, że granica jest linią prostą! Takie sytuacje są częste w uczeniu maszynowym: bardzo ograniczony niepoprawny model może być lepszy niż poprawny, ale słabo określony, ponieważ ten drugi potrzebuje znacznie więcej danych by uniknąć przeuczenia.

Inżynieria cech

Najważniejszym czynnikiem, który może sprawić, że projekt uczenia maszynowego będzie sukcesem lub porażką, jest zastosowany zestaw cech. Jeśli mamy wiele niezależnych cech, z których każda koreluje dobrze z klasą, uczenie jest łatwe. Jeśli klasa jest bardzo złożoną funkcją cech, to może nie być możliwe jej automatyczne nauczenie.

Często surowe dane mają postać nie pozwalającą na uczenie, ale można przekształcić je na cechy, które pozwalają. To jest kolejny element, w którym można wykorzystać wiedzę w procesie uczenia, i gdzie można włożyć najwięcej wysiłku w projekt maszynowego uczenia. Często jest to również jeden z najbardziej interesujących elementów, gdzie intuicja, kreatywność i twórczość są równie ważne jak elementy techniczne.

Inżynieria cech (2)

Inżynieria cech może być w pewnym stopniu zautomatyzowana, przez automatyczne generowanie dużej liczby zmiennych stanu, i wybieranie najlepszych ze względu na zysk informacji w odniesieniu do klasy. Ale cechy, które pojedynczo sprawiają wrażenie nieistotnych, mogą być wartościowe w połączeniu. Na przykład, jeśli klasa jest funkcją XOR k cech wejściowych, to żadna z nich sama w sobie nie nosi żadnej informacji o klasie.

Z drugiej strony, uczenie klasyfikatora z bardzo dużą liczbą cech aby wyłonić takie, które są użyteczne w połączeniu, może być zbyt czasochłonne, i/lub powodować przeuczenie. Tak więc w sumie nic nie jest w stanie zastąpić inteligentnej inżynierii cech.

Duża ilość danych znaczy więcej niż „mocny” algorytm

Załóżmy, że skonstruowaliśmy najlepszy możliwy zestaw cech, ale wygenerowany klasyfikator nadal nie jest dostatecznie dokładny. Co jeszcze można zrobić?

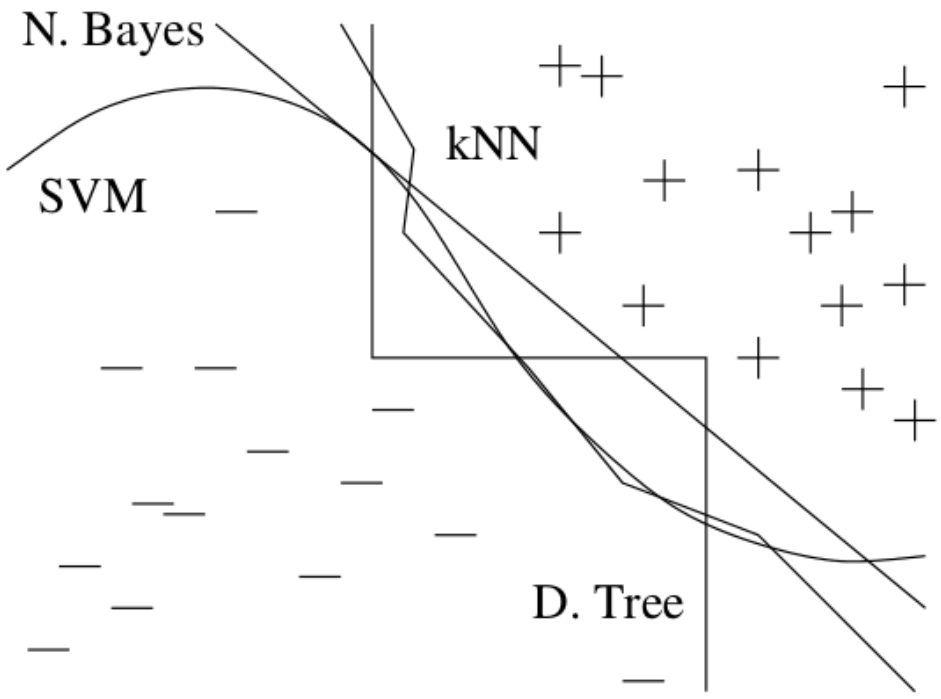
Istnieją dwie główne opcje: lepszy algorytm maszynowego uczenia lub zgromadzenie więcej danych: więcej próbek, ale również więcej nowych cech, z uwzględnieniem klątwy wymiarowości. Specjaliści od maszynowego uczenia koncentrują się na pierwszym podejściu, ale w rzeczywistości najszybszą drogą do sukcesu często jest pozyskanie dodatkowych danych. Ogólnie, słabszy algorytm z dużą ilością danych ma większe szanse powodzenia niż zaawansowany algorytm z mniejszą ich ilością. Uczenie maszynowe w istocie polega na wykorzystaniu dużej ilości danych do rozwiązywania problemów.

Duża ilość danych znaczy więcej niż „mocny” algorytm (2)

Jednym z powodów, dla których stosowanie wyrafinowanych algorytmów daje słabsze efekty niż możnaby się spodziewać jest, że do pewnego stopnia wszystkie algorytmy robią to samo. Działanie wszystkich algorytmów zasadniczo polega na grupowaniu bliskich próbek w jedną klasę; główna różnica polega na tym co znaczy „bliskich”.

W przypadku nierównomiernego rozkładu danych algorytmy mogą tworzyć znacząco różne granice, jednocześnie tak samo klasyfikując dane w regionach gdzie jest dużo danych uczących (i gdzie zapewne pojawi się większość danych testowych).

Ten efekt jest jeszcze silniejszy w przestrzeniach wielowymiarowych.



Silniejszy algorytm nie zawsze jest lepszy

Próba rozpoczęcia uczenia maszynowego „najsilniejszymi" metodami zwykle nie jest najlepszą strategią. Nic nie zastąpi samodzielnego przeprowadzenia wstępnych eksperymentów, i poszukiwania metod polepszenia wyniku.

Z reguły opłaca się najpierw spróbować najprostsze algorytmy, np.:

- naiwny klasyfikator bayesowski przed regresją logistyczną
- k-najbliższych sąsiadów przed maszynami wektorów nośnych
- itd.

Bardziej wyrafinowane algorytmy są kuszące, ale zwykle trudniejsze w użyciu, ponieważ mają więcej ustawień, które trzeba poprawnie dobrać, aby uzyskać dobre wyniki, oraz ponieważ ich wewnętrzne działanie jest bardziej nieprzejrzyste.

Powinniśmy odwołać się do nich, gdy jesteśmy pewni, że zbadaliśmy wszystkie opcje prostszych metod, i mamy zasoby niezbędne do głębszej eksploracji: czas i umiejętności.

Tworzenie wielu modeli zamiast jednego

Dawniej w uczeniu maszynowym specjaliści mieli swoje ulubione algorytmy. Większość wysiłku szła w próbowanie wielu odmian jednej metody i wybranie najlepszej.

Późniejsze badania wykazały, że wybór najlepszego algorytmu zależy od zastosowania. Zaczęły pojawiać się systemy implementujące wiele algorytmów. Teraz wysiłek zaczął iść w próbowanie wielu odmian wielu metod, nadal w celu wybrania najlepszych.

Ale końcu naukowcy zauważyli, że zamiast wybierać jedną najlepszą odmianę, lepiej łączyć wiele odmian. W ten sposób uzyskuje się lepsze, a często znacznie lepsze, wyniki.

Takie podejście nazywa się: **ensemble learning**.

Tworzenie wielu modeli zamiast jednego (2)

Istnieje szereg metod pozwalających łączyć ze sobą oddzielnie budowane modele.

Bagging, inaczej **bootstrap aggregating**, generuje losowe wariacje zestawu uczącego przez próbkowanie (duplikację i usuwanie próbek), uczenie klasyfikatora na każdej oddzielnie, i łączenie wyników przez głosowanie większościowe. Metoda działa, ponieważ znacznie zmniejsza wariancję wyniku, tylko nieco zwiększając odchylenie.

> Algorytm **random forest** łączy losowe drzewa decyzyjne z *bagging* dla osiągnięcia bardzo wysokiej dokładności klasyfikacji.

Boosting tworzy kolejne klasyfikatory i nadaje wagi próbkom uczącym i są one różnicowane w taki sposób aby każdy kolejny klasyfikator skupiał się na próbkach, na których poprzednie klasyfikatory dawały złe wyniki. Metoda działa dobrze w sytuacjach, gdy tworzone klasyfikatory wykazują małą wariancję i duże odchylenie.

> Algorytm **AdaBoost** był jednym z pierwszych przykładów tego podejścia, i pozostaje bardzo popularną metodą.

Stacking tworzy wiele klasyfikatorów, również różnymi metodami, i wyjścia poszczególnych klasyfikatorów stają się wejściami klasyfikatora „wyższego poziomu," którego zadaniem jest optymalne wygenerowanie ostatecznej predykcji klasy próbki.

Materiały

W tej prezentacji wykorzystane zostały materiały z następujących prac:

1. Stuart J. Russell, Peter Norvig: Artificial Intelligence A Modern Approach (Third Edition), Prentice-Hall, 2010

2. Ian H. Witten, Eibe Frank, Mark A. Hall: Data Mining Practical Machine Learning Tools and Techniques, Third Edition, Morgan Kaufman, 2011

3. Kevin P. Murphy: Machine Learning A Probabilistic Perspective, MIT Press, 2012 (rysunki udostępnione przez MIT Press)

4. Leslie Kaelbling, Tomás Lozano-Pérez: M.I.T. 6.034 Artificial Intelligence Spring 2005 `http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-spring-2005/lecture-notes/`

5. Tom M. Mitchell: Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression, draft chapter intended for inclusion in the upcoming second edition of the textbook Machine Learning, 2017 `http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf`

6. Pedro Domingos: A Few Useful Things to Know about Machine Learning, Communications of the ACM, 2012