Analiza danych

prof. UAM dr hab. Tomasz Górecki

tomasz.gorecki@amu.edu.pl
http://drizzt.home.amu.edu.pl

Zakład Statystyki Matematycznej i Analizy Danych
Wydział Matematyki i Informatyki
Uniwersytet im. Adama Mickiewicza w Poznaniu

Twierdzenie „No free lunch”

Jeśli uśrednimy jakość klasyfikacji po wszystkich możliwych problemach klasyfikacyjnych to wszystkie algorytmy dają ten sam błąd klasyfikacji na zbiorze testowym. To twierdzenie stanowi, że nie ma uniwersalnie najlepszego algorytmu. Zatem celem nie jest poszukiwanie takiego algorytmu, a znalezienie algorytmu optymalnego dla konkretnych danych.

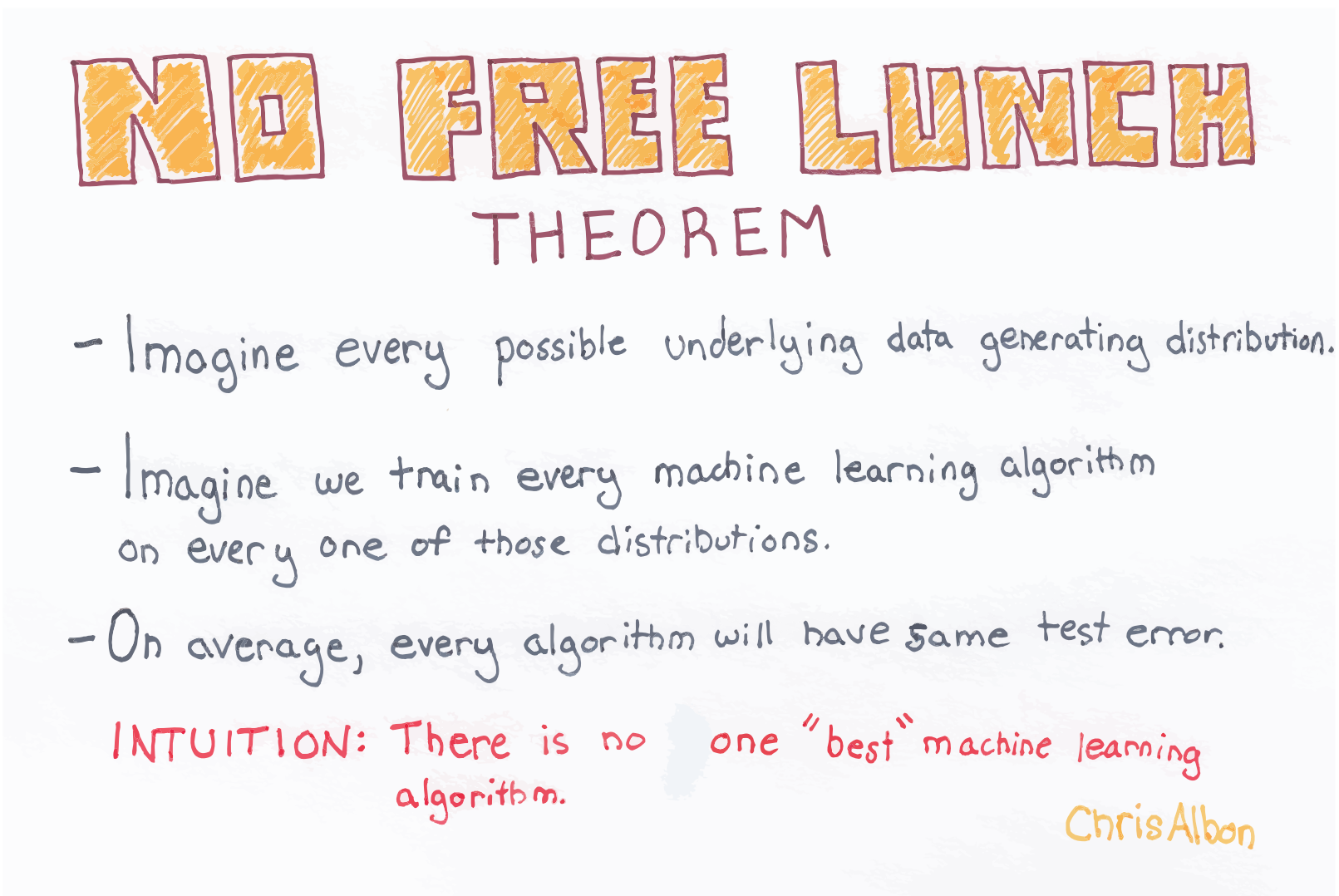
Dowód.

Rozważmy problem dwuklasowy. Jeśli Algorytm A wygrywa to zamieniamy etykiety i będzie wygrywał Algorytm B. □

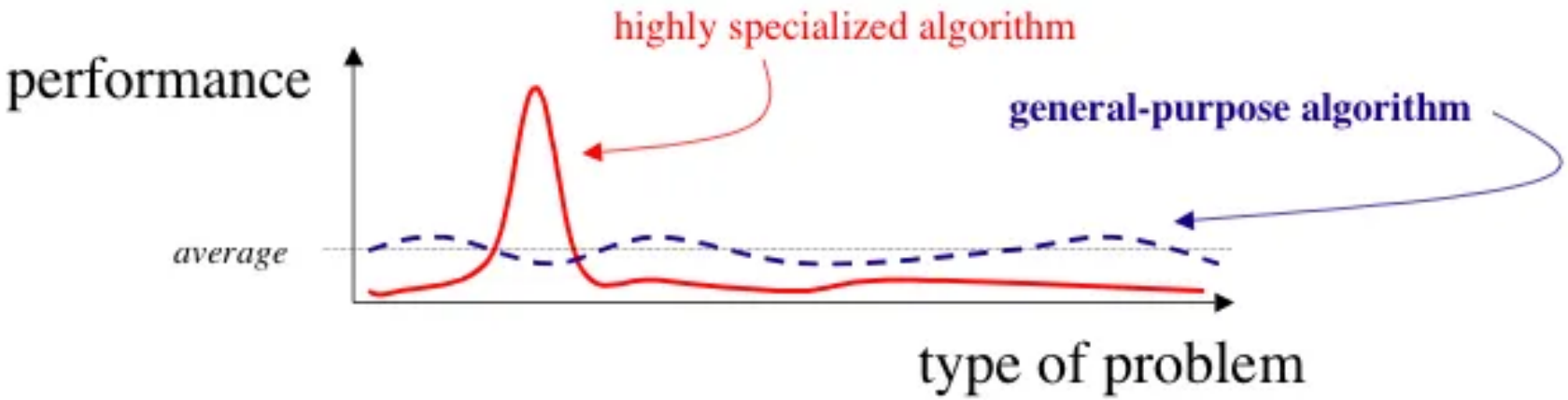

Wolpert, D. (1996). *The Lack of a Priori Distinctions between Learning Algorithms*. Neural Computation 8(7): 341–1390.


Wolpert, D. & Macready, G. (1997). *No Free Lunch Theorems for Optimization*. IEEE Transactions on Evolutionary Computation 1(1):67–82.

Twierdzenie „No free lunch”



Twierdzenie „No free lunch”



Wstęp

Problem klasyfikacji (*ang. classification*) zwany też analizą dyskryminacyjną, sprowadza się do określenia przydziału obiektów opisanych za pomocą wartości wielu cech (atrybutów) do jednej z wcześniej ustalonych klas. Klasom przypisane są kody zwane etykietami. Pierwszym elementem systemu uczącego się pod nadzorem jest próba ucząca (*ang. learning (train) sample*) składająca się z n niezależnych par zmiennych $(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)$, gdzie $\boldsymbol{X}_i$ jest wektorem zaobserwowanych cech na obiekcie pochodzącym z klasy o etykiecie Y_i, $i = 1, \ldots, n$.

Wstęp

Drugim elementem jest funkcja określona na przestrzeni wartości cech o wartościach w zbiorze etykiet, zwana klasyfikatorem (*ang. classifier*), skonstruowana na bazie próby uczącej. Trzecim elementem jest ocena skuteczności działania klasyfikatora (ocena wartości błędu klasyfikacji lub trafności klasyfikacji).

Pojęcia wstępne

Załóżmy, że dysponujemy K niezależnymi, prostymi próbami losowymi o liczebnościach, odpowiednio: $n_1, n_2, \ldots, n_K$, pobranymi z K różnych populacji (klas, grup):

$$(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n),$$

gdzie $\boldsymbol{X}_i = (X_{i1}, X_{i2}, \ldots, X_{ip})' \in \mathcal{X} \subset \mathbb{R}^p$ jest i-tą obserwacją, natomiast Y_i jest etykietą populacji, do której ta obserwacja należy, przyjmującą wartości w pewnym skończonym zbiorze $\mathcal{Y}$, $i = 1, 2, \ldots, n$, $n = n_1 + n_2 + \cdots + n_K$.
Próbę $\mathcal{L}_n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ nazywać będziemy próbą uczącą (*ang. training sample*). Interesuje nas problem predykcji etykiety Y na podstawie wektora cech $\boldsymbol{X}$. Problem ten nazywany jest klasyfikacją, dyskryminacją, uczeniem się pod nadzorem lub rozpoznawaniem wzorców.

Pojęcia wstępne

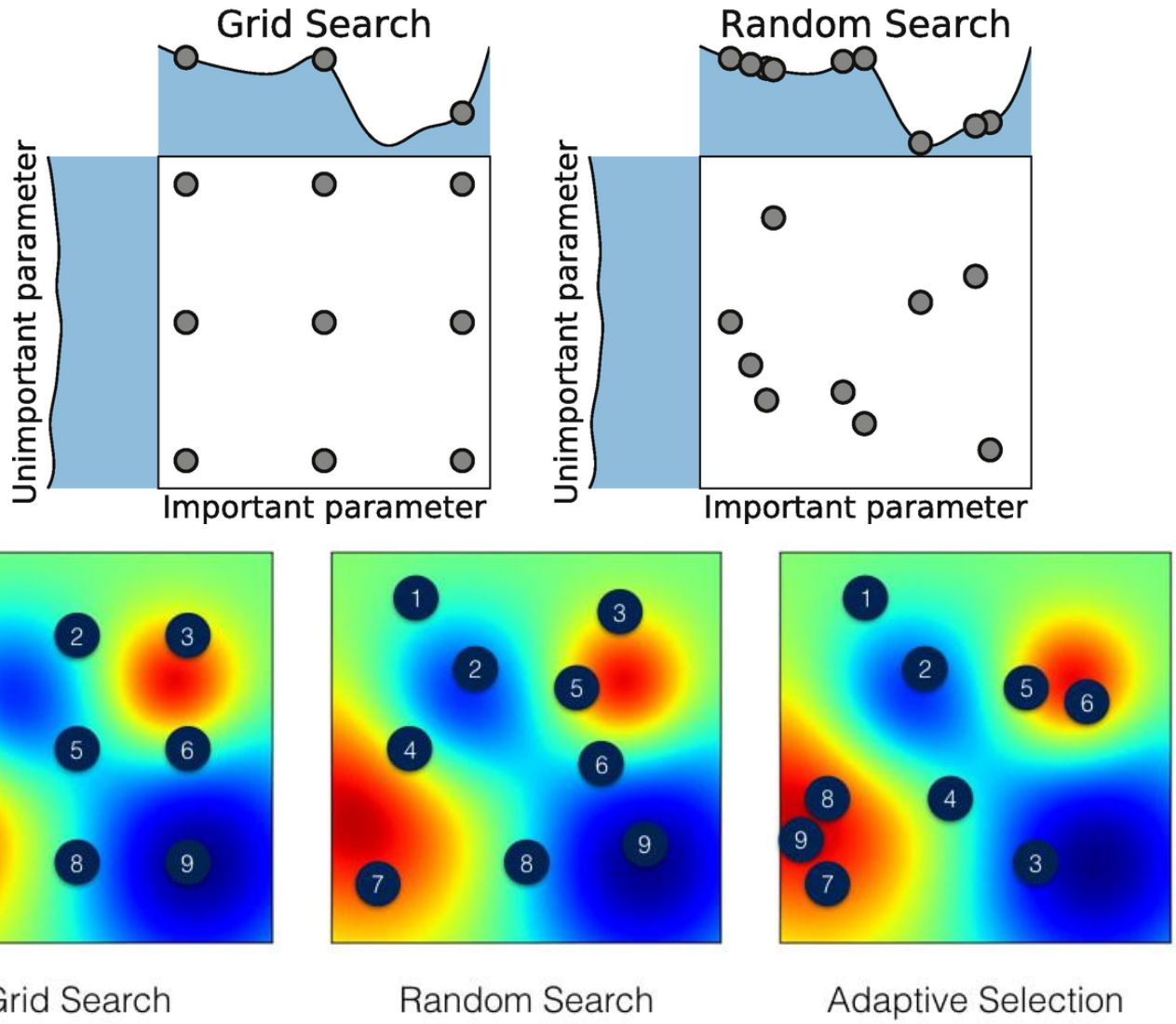
Reguła klasyfikacyjna, zwana krótko klasyfikatorem, jest funkcją $d: \mathcal{X} \to \mathcal{Y}$. Gdy obserwujemy nowy wektor $\boldsymbol{X}$, to prognozą etykiety Y jest $d(\boldsymbol{X})$.

Parametry vs hiperparametry

Parametry modelu są wyznaczane w czasie uczenia, są zatem wynikiem uczenia modelu (np. współczynniki regresji).
Hiperparametry są ustalane przed uczeniem modelu i, w przeciwieństwie do parametrów, nie mogą być wyznaczone podczas uczenia. Stosuje się dwa podstawowe podejścia do optymalizacji hiperparametrów:

- Przeszukiwanie po siatce (*ang. grid search, exhaustive search*).
- Przeszukiwanie losowe (*ang. random search*).
- Przeszukiwanie adaptacyjne (*ang. adaptive selection*).

Parametry vs hiperparametry



Parametry vs hiperparametry

Przeszukiwanie losowe jest zalecane kiedy przeszukiwana przestrzeń parametrów jest wysoko wymiarowa (więcej niż 3 wymiary). W takiej sytuacji za jego pomocą możemy przeszukać szersze spektrum wartości hiperamarametrów ponieważ przeszukiwanie po siatce jest zazwyczaj dużo bardziej złożone obliczeniowo.

Idea


CLASSIFICATION
Classification problems are when we are training a model to predict qualitative targets. For example: gender or a type of fruit.
Data point
Decision boundary
ChrisAlbon

Ocena jakości

Proces konstruowania klasyfikatora $\hat{d}$ nazywany jest uczeniem się, uczeniem pod nadzorem lub uczeniem się z nauczycielem. Jakość klasyfikatora $\hat{d}$ mierzona jest za pomocą warunkowego prawdopodobieństwa błędu

$$e(\hat{d}) = \mathbb{P}(\hat{d}(\boldsymbol{X}) \neq Y \mid \mathcal{L}_n),$$

gdzie para losowa $(\boldsymbol{X}, Y)$ jest niezależna od próby uczącej $\mathcal{L}_n$. Wielkość $e(\hat{d})$ nazywamy aktualnym poziomem błędu klasyfikatora $\hat{d}$. Ocenę błędu klasyfikacji będziemy nazywać błędem klasyfikacji (*ang. classification error rate*). Przeciwieństwem błędu klasyfikacji jest dokładność klasyfikacji (*ang. accuracy*).

Ocena jakości

W najlepszej sytuacji jesteśmy wtedy, gdy dysponujemy m-elementową próbą testową (*ang. test sample*) $\mathcal{T}_m$ niezależną od próby uczącej $\mathcal{L}_n$. W takiej sytuacji za błąd klasyfikacji przyjmujemy procent obserwacji pochodzących z próby testowej błędnie zaklasyfikowanych za pomocą reguły klasyfikacyjnej skonstruowanej na podstawie próby uczącej. W przypadku gdy nie dysponujemy niezależną próbą testową, do estymacji używamy jedynie próby uczącej.

Ocena jakości

Naturalną oceną aktualnego poziomu błędu jest wtedy wartość estymatora ponownego podstawienia (resubstytucji, *ang. resubstitution error rate*). Wartość tego estymatora uzyskuje się poprzez klasyfikację regułą $\hat{d}$ tych samych obserwacji, które służyły do jej konstrukcji. Oznacza to, iż próba ucząca jest zarazem próbą testową. Estymator ten jest więc obciążonym estymatorem wielkości $e(\hat{d})$ i zaniża jej rzeczywistą wartość.

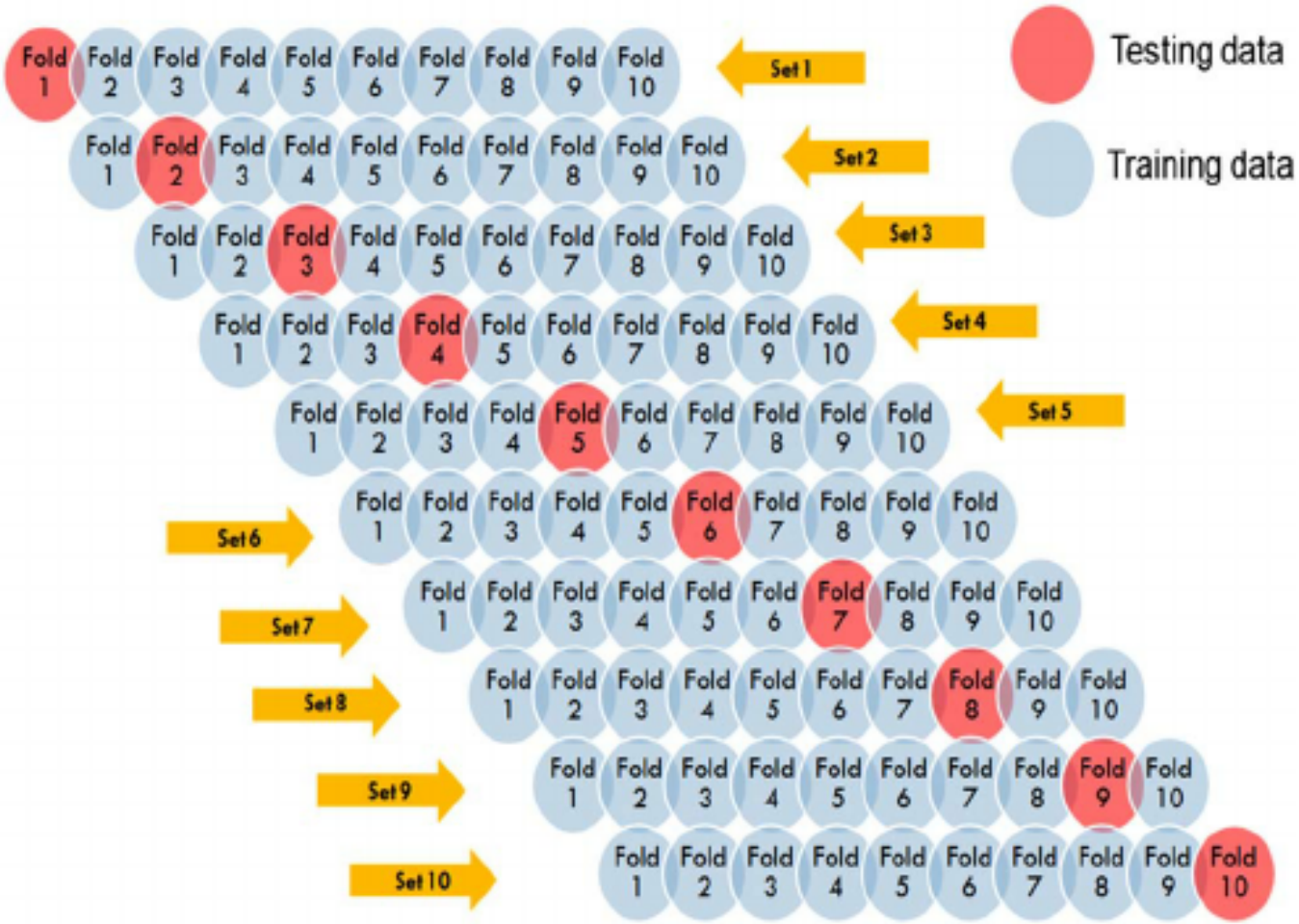
Ocena jakości

Jednym ze sposobów redukcji obciążenia powyższego estymatora przy zastosowaniu próby uczącej jako jednocześnie próby testowej jest tzw. metoda podziału próby na dwa podzbiory: próbę uczącą i próbę testową. Wówczas klasyfikator konstruuje się za pomocą pierwszego z nich, drugi natomiast służy do konstrukcji estymatora. Wykorzystanie tylko części informacji w celu uzyskania reguły klasyfikacyjnej prowadzi jednak często do zawyżenia wartości estymatora błędu. Rozwiązaniem tego problemu jest metoda sprawdzania krzyżowego z pojedynczym usuwaniem (*ang. leave-one-out cross-validation (LOOCV, nCV)*). W takiej sytuacji usuwamy kolejne obserwacji z próby uczącej i konstruujemy klasyfikator w oparciu o taką pomniejszoną o jeden próbę uczącą. Usunięty element jest następnie klasyfikowany za pomocą tak wyznaczonej metody. Frakcja błędnie zaklasyfikowanych w taki sposób obserwacji jest uznawana za błąd klasyfikacji.

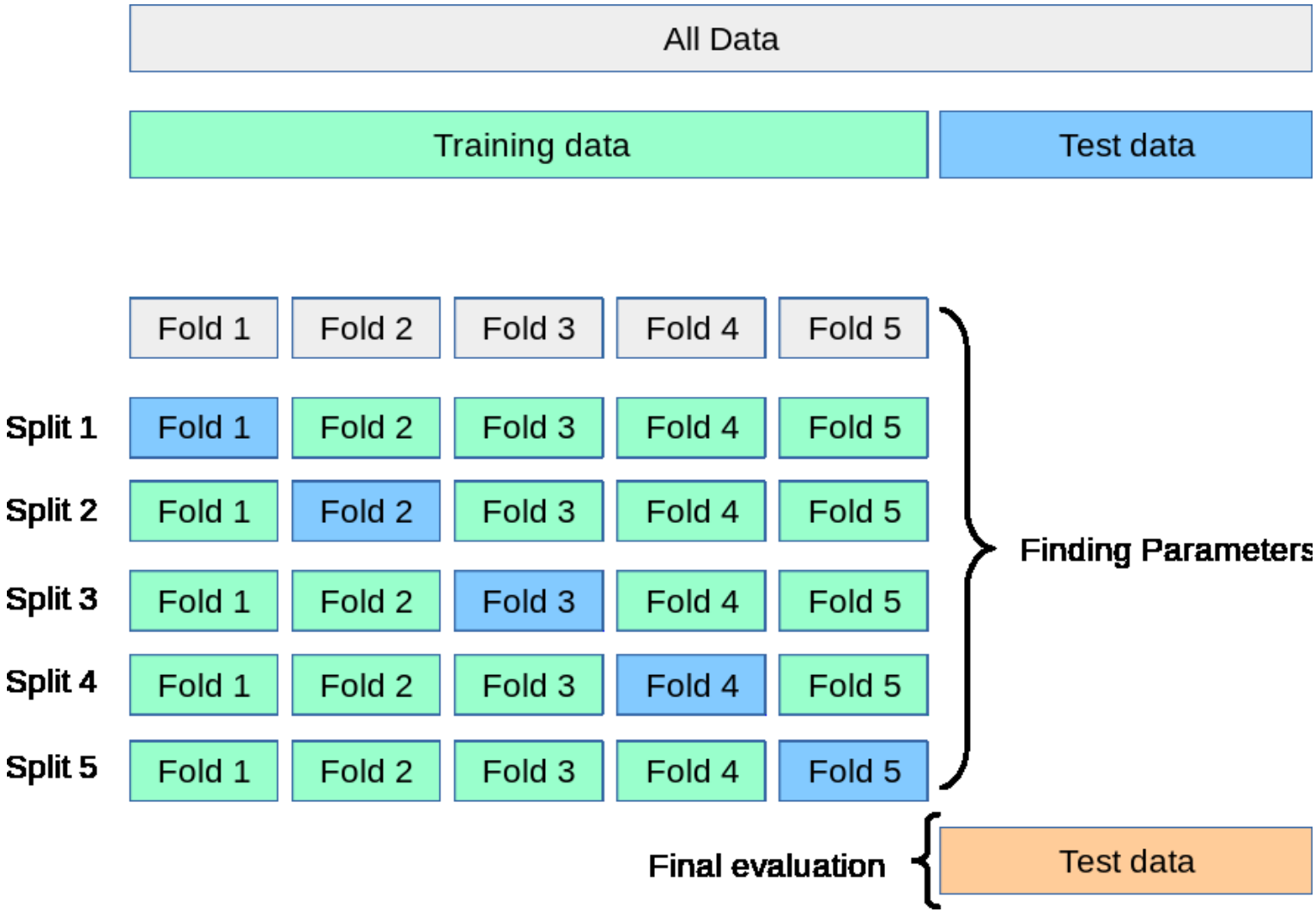
Ocena jakości

Procedura ta w każdym z n etapów jest w rzeczywistości metodą podziału próby dla przypadku jednoelementowego zbioru testowego. Każda obserwacja próby jest użyta do konstrukcji klasyfikatora $\hat{d}$. Każda z nich jest też (dokładnie jeden raz) elementem testującym. Estymator ten, choć granicznie nieobciążony, ma większą wariancję. Ponadto wymaga on konstrukcji n klasyfikatorów, co dla dużych n oznacza znaczący wzrost obliczeń. Rozwiązaniem pośrednim jest metoda rotacyjna, zwana też k-krokową metodą sprawdzania krzyżowego (*ang. k-fold cross-validation (kCV)*). Polega ona na losowym podziale próby na k podzbiorów, przy czym $k-1$ z nich tworzy próbę uczącą, natomiast pozostałe – próbę testową. Procedurę tę powtarza się k razy, dla każdego podzbioru rozpatrywanego kolejno jako zbiór testowy.

Ocena jakości



Ocena jakości



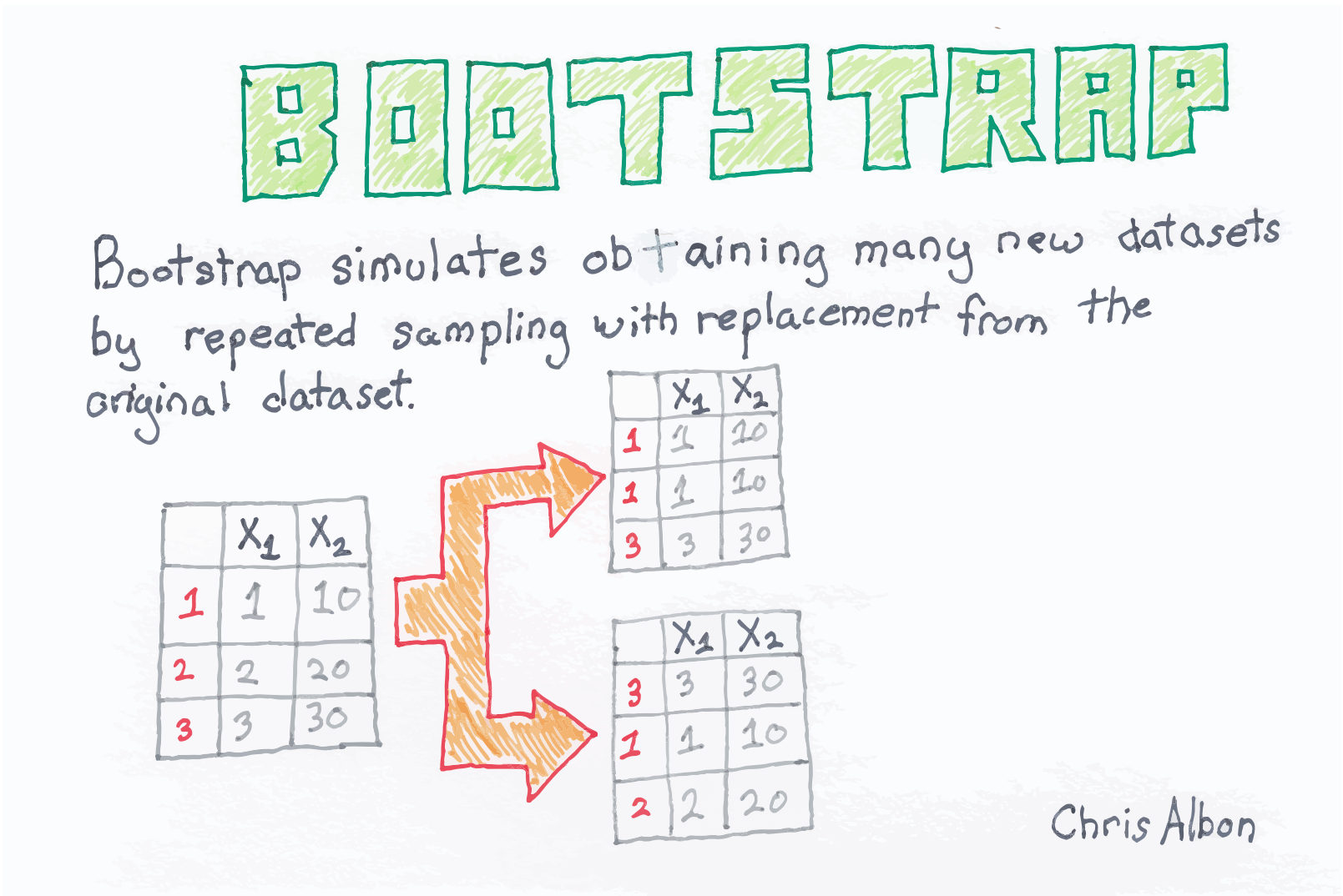
Ocena jakości

Metoda ta daje mniejsze obciążenie błędu niż metoda podziału próby i wymaga mniejszej liczby obliczeń w porównaniu ze sprawdzaniem krzyżowym (jeśli tylko $k < n$). W zagadnieniu estymacji aktualnego poziomu błędu zalecane jest $k = 5$ lub $k = 10$.
W celu redukcji losowości używa się również powtórzeń np. 5×5CV.

Ocena jakości

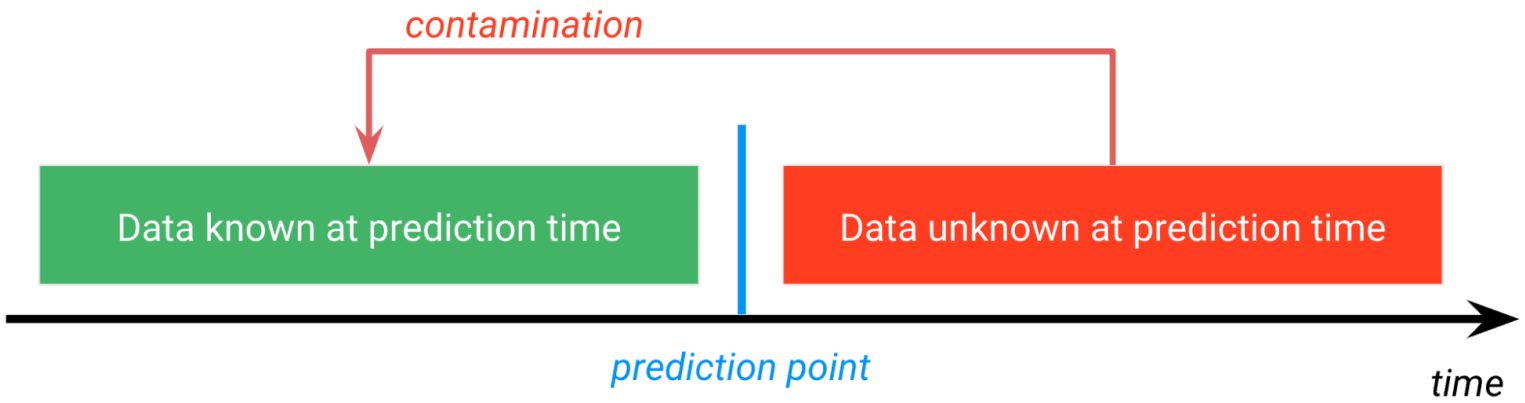
Wykorzystywana bywa również metoda bootstrap (*ang. bootstrap*) polegająca na wielokrotnym (zaleca się co najmniej 50) losowym podziale zbioru na zbiór uczący i testowy, przy czym losowanie odbywa się ze zwracaniem. Uzyskany zbiór uczący musi mieć taką samą liczebność jak cały badany zbiór. Następnie uczymy metodę na tak otrzymanym zbiorze i testujemy na tych obserwacjach, które nie zostały wylosowane.

Ocena jakości

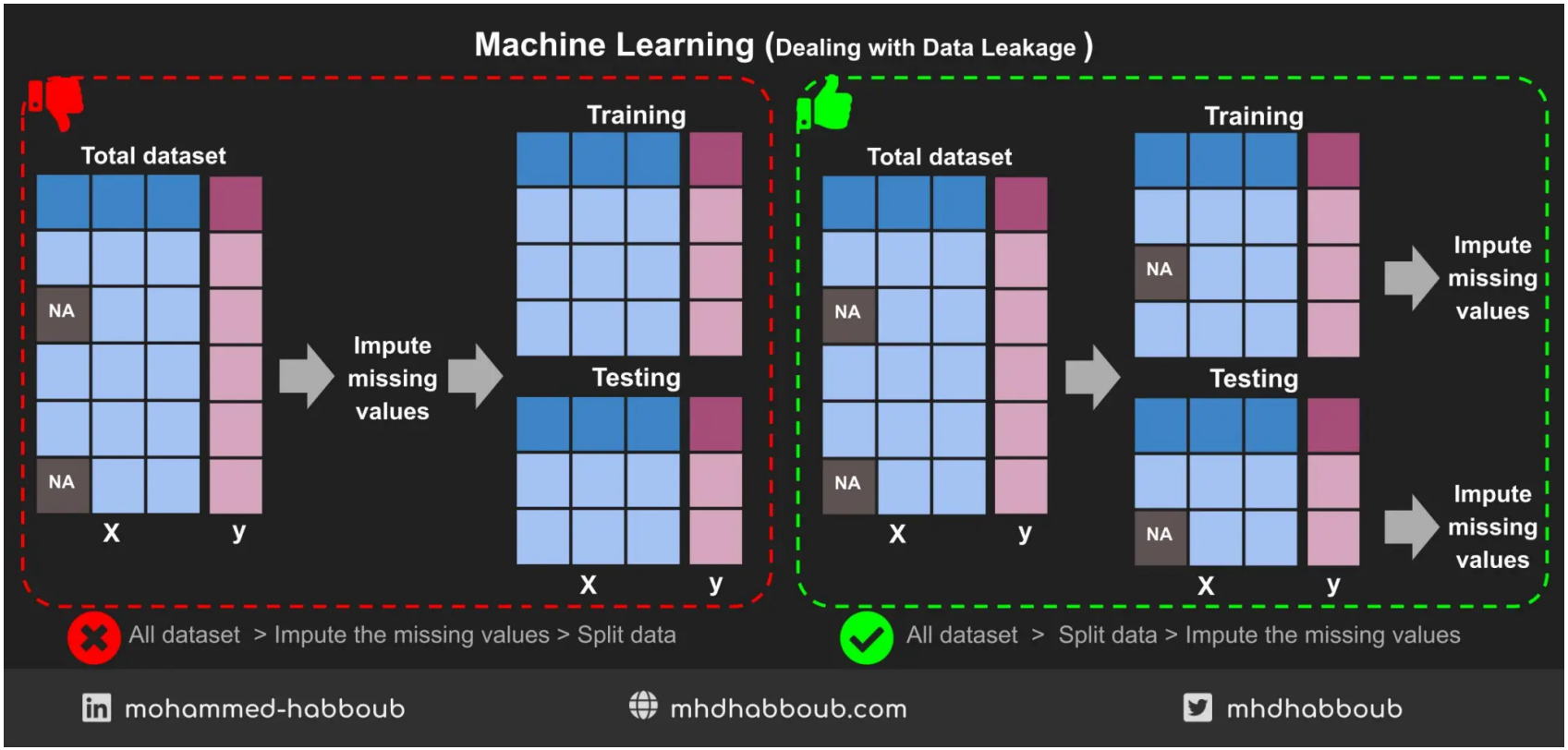


Wyciek danych

Wyciek danych (*ang. contamination, data leakage*) występuje, gdy informacje o zbiorze testowym zostaną uwzględnione w procesie uczenia, co skutkuje zbyt optymistycznym oszacowaniem jakości modelu.



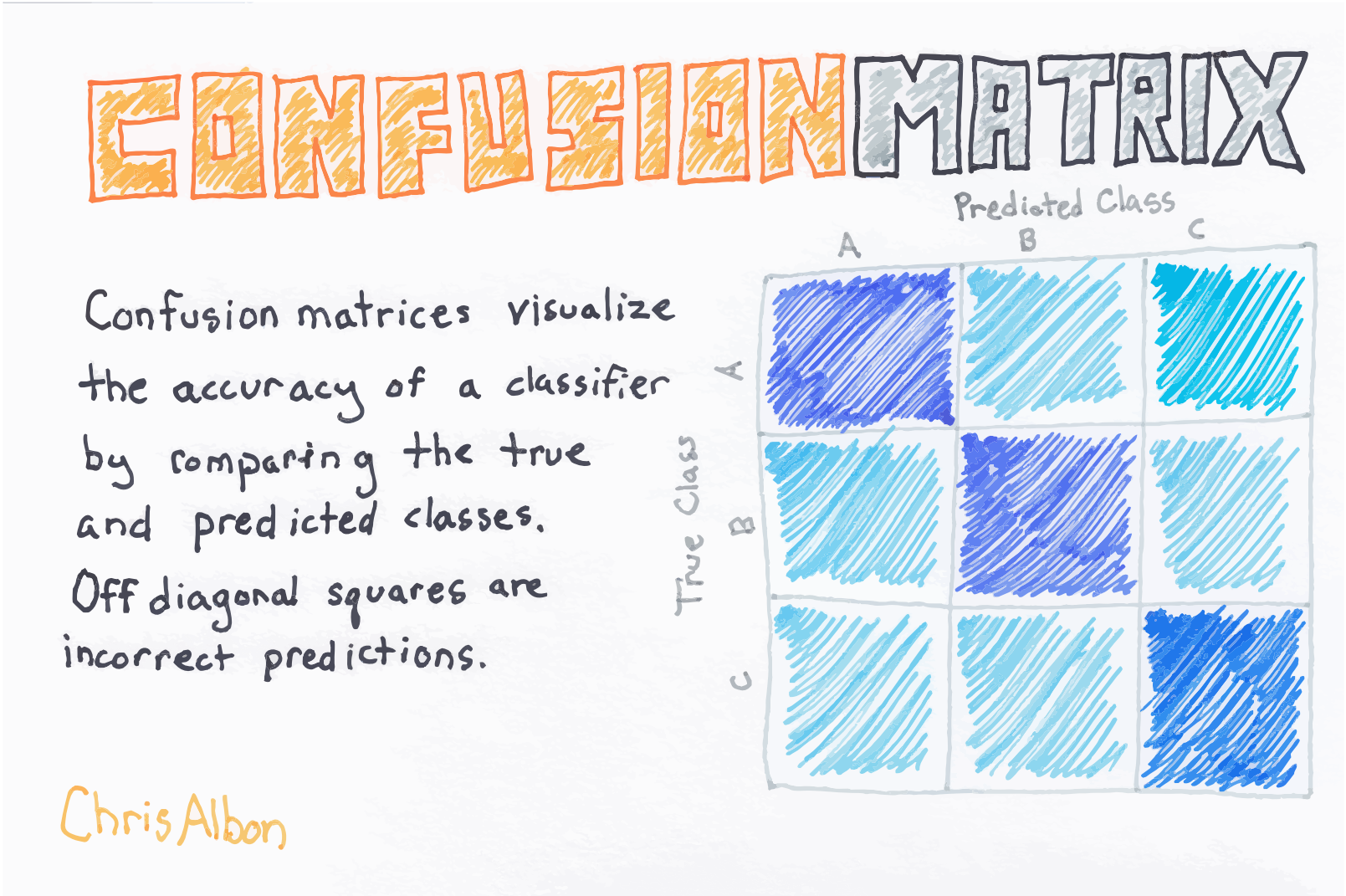
Wyciek danych



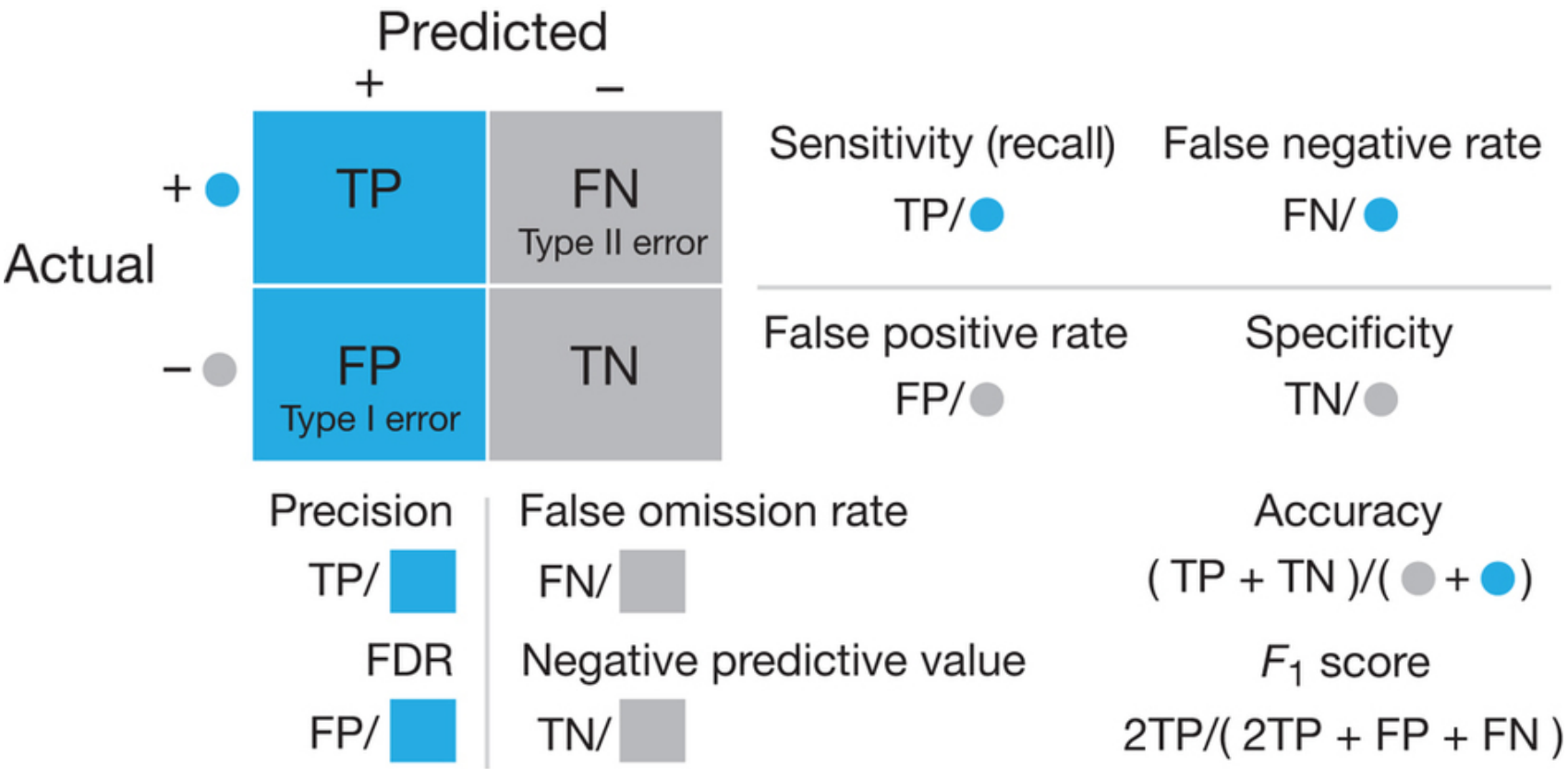
Macierz pomyłek (przynależności)

Sam błąd klasyfikacji nie wystarczy aby stwierdzić, czy klasyfikator działa dobrze. Po zakończeniu procedury oceny błędu klasyfikacji naszego modelu zostajemy z listą obserwacji testowych, gdzie dla każdej z nich znamy klasę obserwowaną oraz klasę przewidywaną przez nasz model. Zliczając liczbę przypadków dla każdej z kombinacji tych dwóch klas (obserwowanej i przewidywanej) możemy stworzyć tzw. macierz pomyłek (*ang. confusion matrix*). Możemy na jej podstawie ocenić błąd modelu: musimy zsumować wartości na głównej przekątnej (obserwacje poprawnie rozpoznane) i podzielić je przez liczbę wszystkich obserwacji testowych.

Macierz pomyłek (przynależności)



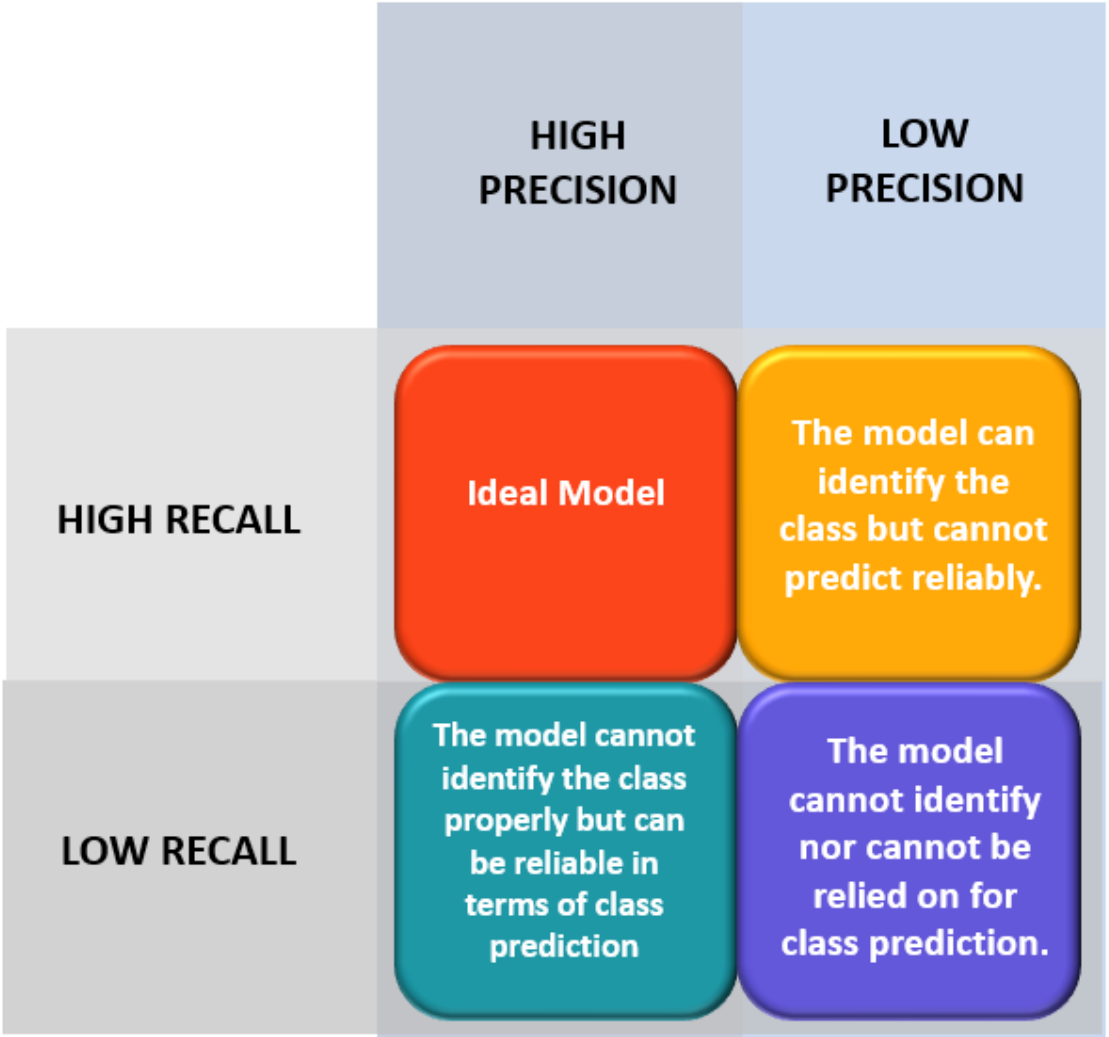
Inne miary jakości klasyfikacji



Inne miary jakości klasyfikacji

- Czułość (*ang. sensitivity, recall, hit rate*) – prawdopodobieństwo, że klasyfikacja będzie poprawna pod warunkiem, że przypadek jest pozytywny.
- Specyficzność (*ang. specificity*) – prawdopodobieństwo, że klasyfikacja będzie poprawna, pod warunkiem, że przypadek jest negatywny.
- Precyzja (*ang. precision*) – prawdopodobieństwo otrzymania poprawnej pozytywnej klasyfikacji, pod warunkiem, że otrzymaliśmy przypadek pozytywny.

Inne miary jakości klasyfikacji



Inne miary jakości klasyfikacji

- Miara F1 (*ang. F1-score*) – średnia harmoniczna z precyzji i czułości:

$$\mathrm{F1} = \frac{2}{\mathrm{precision}^{-1} + \mathrm{recall}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

  Miara ta daje ocenę balansu między czułością a precyzją. Miara ta nie uwzględnia wyników prawdziwie negatywnych. Czasami dokładność klasyfikacji (*ang. accuracy*) bywa nazywana micro-F1.
- Makro F1 (*ang. macro-F1*) – wersja dla klasyfikacji niebinarnej, liczymy F1 dla każdej klasy i uśredniamy.
- Miękkie F1 (*ang. soft-F1*) – podobnie jak w F1, ale uwzględniamy prawdopodobieństwa (zamiast 0/1).

Inne miary jakości klasyfikacji

- Współczynnik korelacji `Matthewsa` (*ang. Matthews correlation coefficient (MCC)*) – uwzględnia wszystkie elementy macierzy pomyłek i jest stosowany nawet wtedy, gdy klasy są bardzo różnej liczebności:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}.$$

Przyjmuje wartości z przedziału $[-1, 1]$. Wartość $+1$ odpowiada idealnej klasyfikacji, 0 to klasyfikator losowy, a -1 to całkowita niezgoda pomiędzy prognozą i stanem faktycznym. W przeciwieństwie do F1 jego wartość nie zależy od wyboru klasy pozytywnej. Współczynnik ten jest powiązany ze statystyką χ^2, mianowicie

$$|\text{MCC}| = \sqrt{\frac{\chi^2}{n}}.$$

.

Inne miary jakości klasyfikacji

- Współczynnik zgodności κ (*ang. Cohen's Kappa*) – miara zgodności z klasyfikatorem losowym. Przyjmuje wartości w przedziale od -1 do +1. Wartość 0 oznacza klasyfikator losowy. Wartości dodatnie oznaczają, że klasyfikator jest lepszy od losowego.

$$\kappa = \frac{\text{Accuracy} - \text{Accuracy}_{\text{Random}}}{1 - \text{Accuracy}_{\text{Random}}} = \frac{n \cdot d - \sum_{i=j}^{K} C_{i\cdot} C_{\cdot j}}{n^2 - \sum_{i=j}^{K} C_{i\cdot} C_{\cdot j}},$$

gdzie d jest liczbą poprawnych klasyfikacji, a $\boldsymbol{C}$ macierzą pomyłek i

$$C_{i\cdot} = \sum_{j=1}^{K} C_{ij} \text{ (suma wiersza)},$$

$$C_{\cdot j} = \sum_{i=1}^{K} C_{ij} \text{ (suma kolumny)}.$$

Inne miary jakości klasyfikacji

		True/Actual	
		Positive ()	Negative
Predicted	Positive ()	5 (TP)	1 (FP)
	Negative	2 (FN)	2 (TN)

Precision	$5/6 = 83\%$
Recall	$5/7 = 71\%$
Accuracy	$7/10 = 70\%$
F1	$2\frac{5/6 \cdot 5/7}{5/6+5/7} = 77\%$
MCC	$\frac{10-2}{\sqrt{7 \cdot 6 \cdot 4 \cdot 3}} = 0.3563$
κ	$\frac{10 \cdot 7-(6 \cdot 7+4 \cdot 3)}{10^2-(6 \cdot 7+4 \cdot 3)} = 35\%$

Inne miary jakości klasyfikacji

		True/Actual		
		Cat (🐱)	Fish (🐟)	Hen (🐔)
Predicted	Cat (🐱)	4	6	3
	Fish (🐟)	1	2	0
	Hen (🐔)	1	2	6

	Precision	Recall	F1	Accuracy	Macro-F1
Cat	$4/13 = 31\%$	$4/6 = 67\%$	$2\frac{4/13 \cdot 4/6}{4/13+4/6} = 42\%$	$12/25 = 48\%$	$\frac{42+31+67}{3} = =47\%$
Fish	$2/3 = 67\%$	$2/10 = 20\%$	$2\frac{2/3 \cdot 2/10}{2/3+2/10} = 31\%$		
Hen	$6/9 = 67\%$	$6/9 = 67\%$	$2\frac{6/9 \cdot 6/9}{6/9+6/9} = 67\%$		

$$\kappa = \frac{25 \cdot 12 - (13 \cdot 6 + 3 \cdot 10 + 9 \cdot 9)}{25^2 - (13 \cdot 6 + 3 \cdot 10 + 9 \cdot 9)} = 25\%$$

Inne miary jakości klasyfikacji

- Entropia krzyżowa binarna (*ang. cross-entropy loss (negative log-likelihood, logistic loss)*):

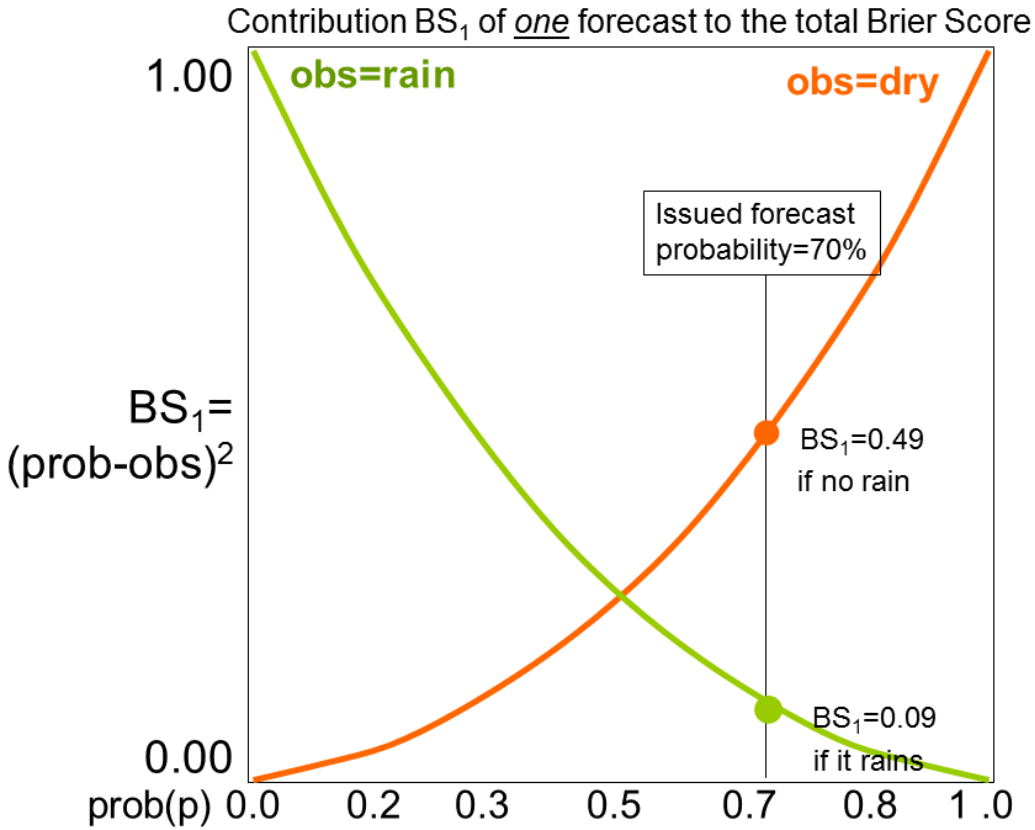
$$\text{LogLoss}(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right),$$

gdzie $y_i \in \{0,1\}$ oraz $\hat{y}_i \in (0,1)$ jest oszacowanym prawdopodobieństwem przynależności obserwacji do klasy.

- Współczynnik Briera (ang. *Brier score*):

$$\text{BS} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2.$$

Inne miary jakości klasyfikacji

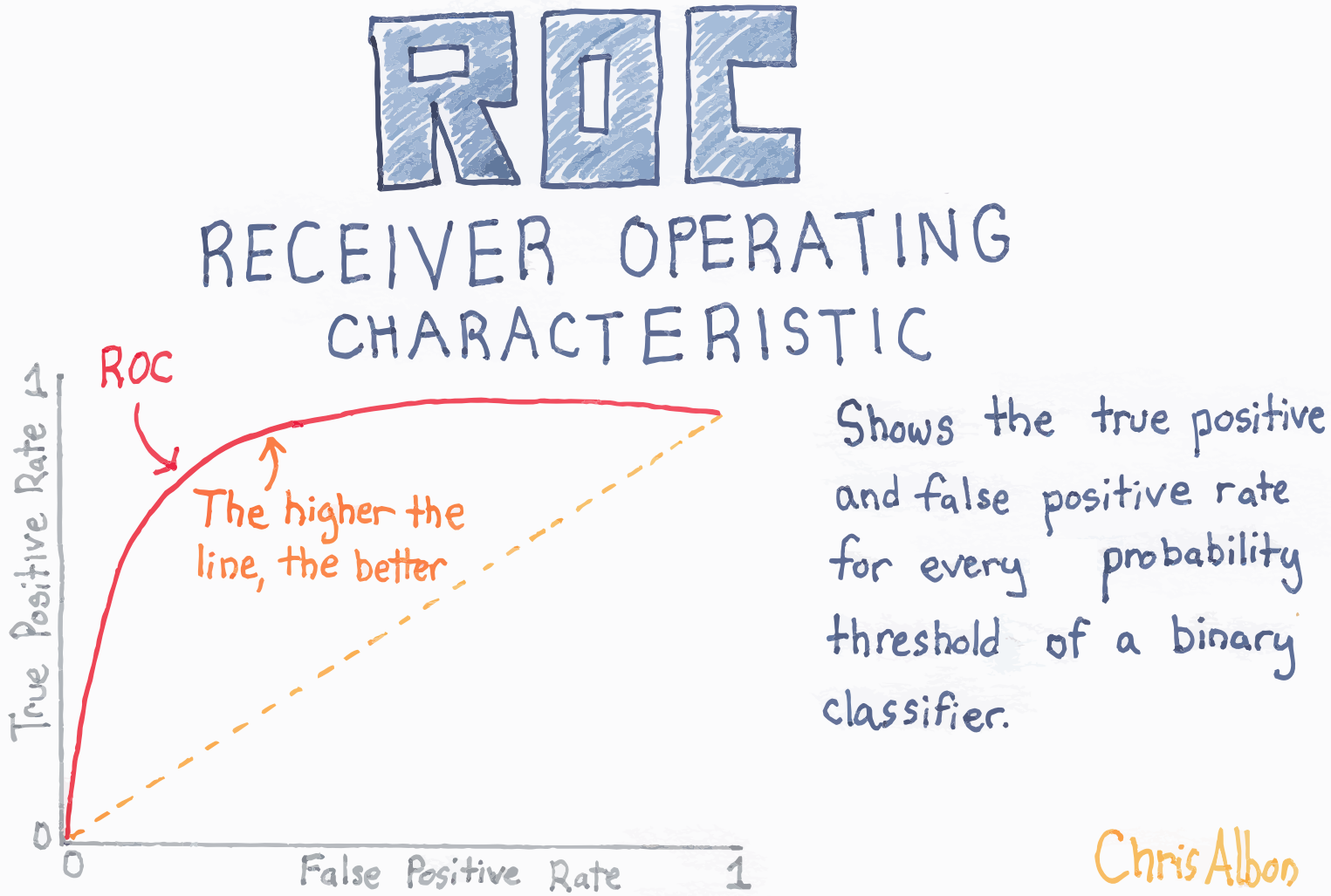


Brier, G.W. (1950). *Verification of forecasts expressed in terms of probability*. Monthly Weather Review 78(1):1–3.

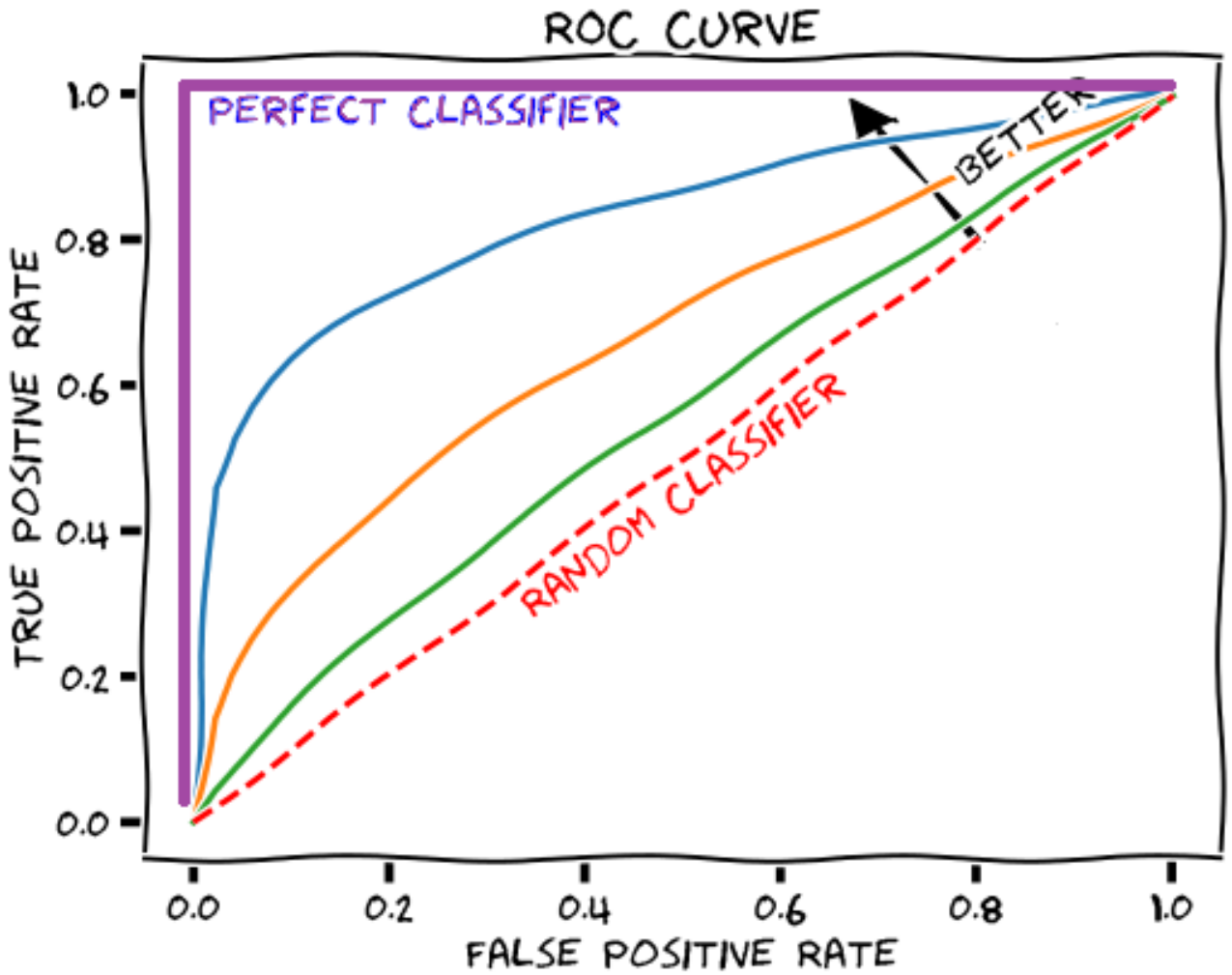
Inne miary jakości klasyfikacji

Krzywa ROC (*ang. Receiver Operating Characteristic*) jest narzędziem do oceny poprawności klasyfikatora, zapewnia ona łączny opis jego czułości i specyficzności (na osi X mamy FPR, a na osi Y mamy TPR). Krzywa ROC bywa często wykorzystywana jako narzędzie od oceny i porównywania między sobą modeli klasyfikacyjnych. Bardzo popularnym podejściem jest wyliczanie pola pod wykresem krzywej ROC, oznaczanego jako AUC (*ang. Area Under Curve*), i traktowanie go jako miarę jakości danego modelu dwuklasowego. Wartość wskaźnika AUC przyjmuje wartości z przedziału [0, 1] i im większa, tym lepszy model.

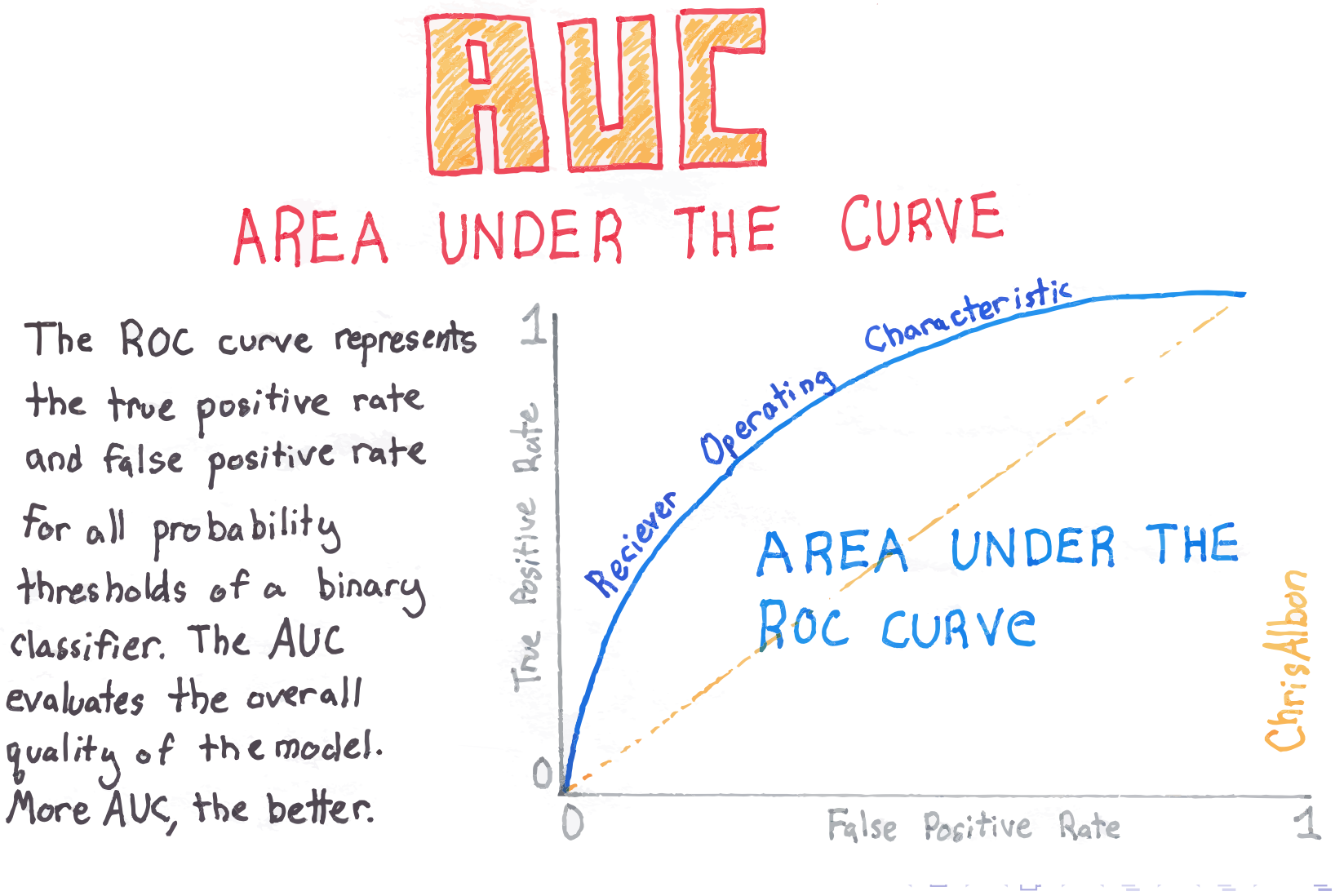
Inne miary jakości klasyfikacji



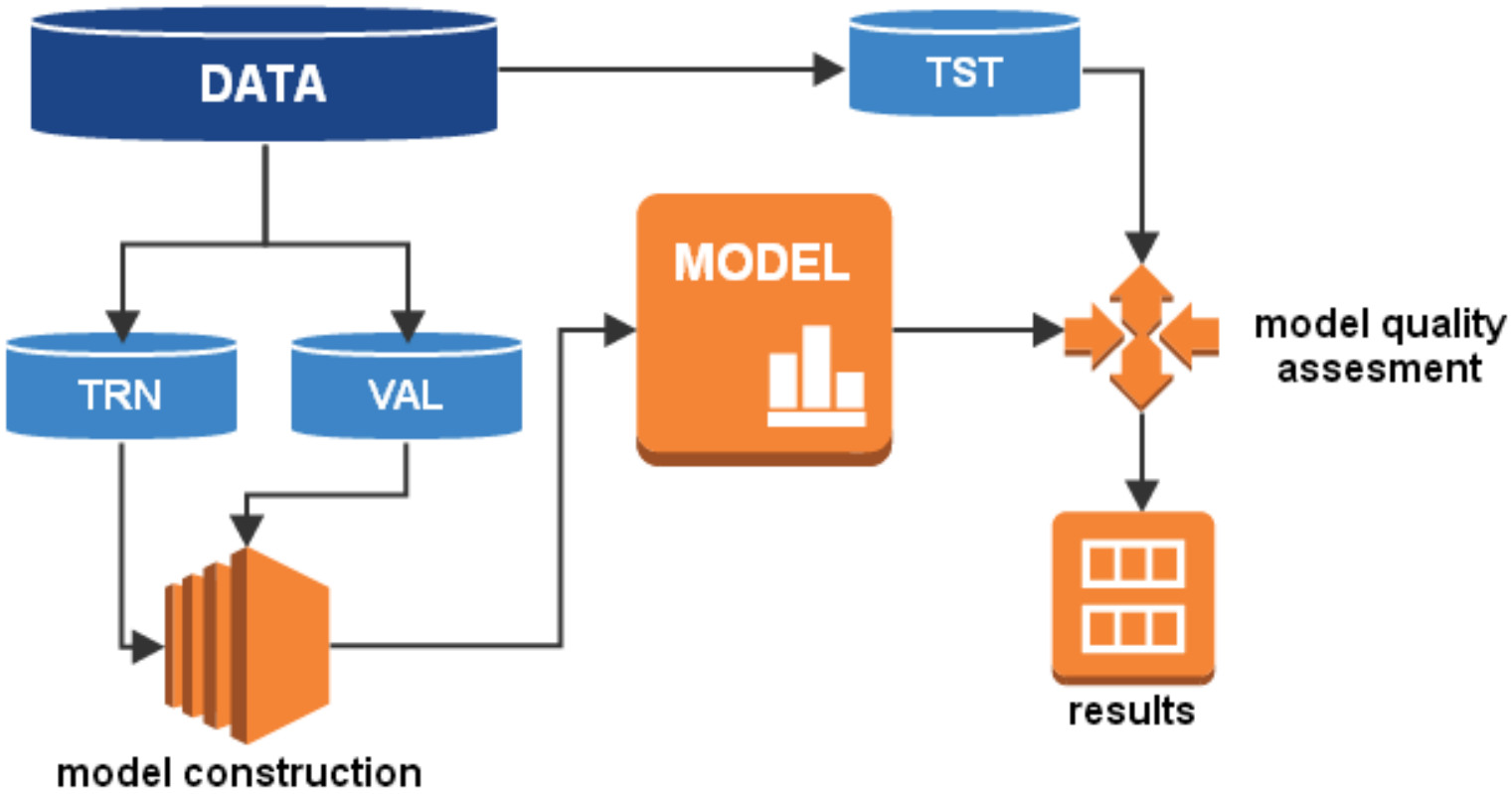
Inne miary jakości klasyfikacji



Inne miary jakości klasyfikacji

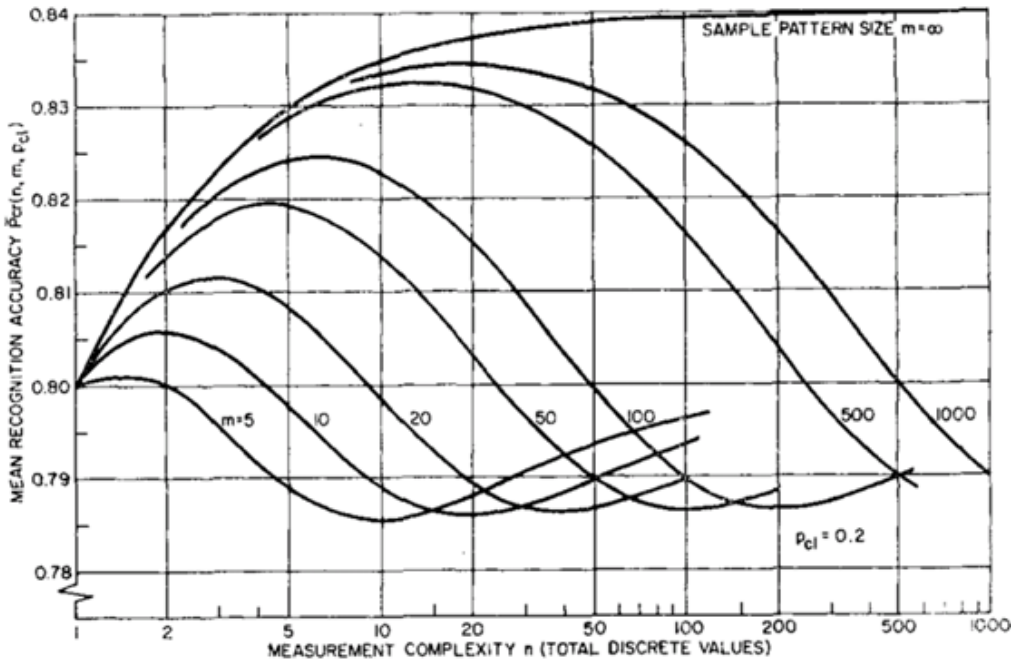


Proces konstrukcji modelu



Przekleństwo wymiarowości

Dla ustalonej wielkości próby uczącej jakość klasyfikatora (regresji) na początku wzrasta wraz ze zwiększaniem liczby cech, a następnie spada. Zjawisko to nazywane jest paradoksem Hughes'a.



Hughes, G.F. (1968). *On the mean accuracy of statistical pattern recognizers*. IEEE Transactions on Information Theory 14(1):55–63.

ZeroR – najprostszy klasyfikator

Algorytm ZeroR (*ang. Zero Rule*) ignoruje wszystkie cechy, oprócz etykiet. Na ich podstawie, sprawdza najczęściej występującą klasę i tworzy klasyfikator który zawsze będzie ją zwracać. Zatem algorytm ZeroR zawsze przypisuje nowe obserwacje do tej samej klasy (klasy większościowej), niezależnie od wartości cech.

OneR

Algorytm OneR (*ang. One Rule*) jest tylko nieco bardziej wyrafinowany niż ZeroR. Podczas gdy ZeroR ignoruje wszystkie cechy, OneR wybiera tylko jedną i ignoruje pozostałe. Wybierając najlepszą cechę sprawdza on błąd na danych uczących dla klasyfikatorów zbudowanych na każdej z cech osobno i wybiera tą z nich, która minimalizuje błąd. Dla każdej cechy dzieli on dane uczące na podzbiory ze względu na wartość tej cechy. Następnie, na każdym z nich używa algorytmu ZeroR. Pokazano, że pomimo prostoty jest to metoda jedynie nieznacznie ustępująca najlepszym klasyfikatorom.

Holte, R.C. (1993). *Very simple classification rules perform well on most commonly used datasets*. Machine Learning 11(1):63–90.

OneR

Day	Outlook	Temperature	Humidity	PlayTennis
D1	sunny	hot	high	NO
D2	sunny	hot	high	NO
D3	overcast	hot	high	YES
D4	overcast	hot	normal	YES
D5	rain	mild	high	NO

Skonstruujmy klasyfikatory dla różnych atrybutów:

Outlook	YES	NO	trafność
sunny	0	2	$5/5 = 100\%$
overcast	2	0	
rain	0	1	

Temperature	YES	NO	trafność
hot	2	2	$3/5 = 60\%$
mild	0	1	

Humidity	YES	NO	trafność
high	1	3	$4/5 = 80\%$
normal	1	0	

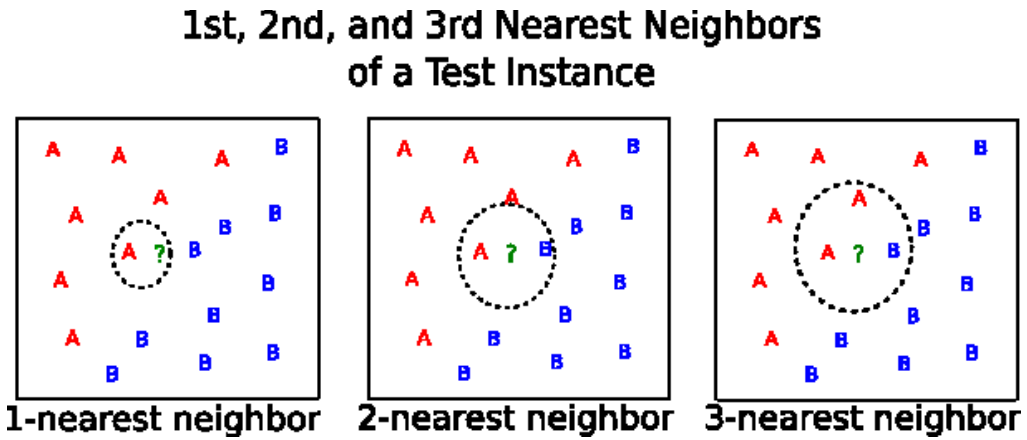
Jak widzimy, najwyższą trafność osiągamy korzystając z atrybutu *Outlook*. Algorytm stworzy więc następujący klasyfikator:

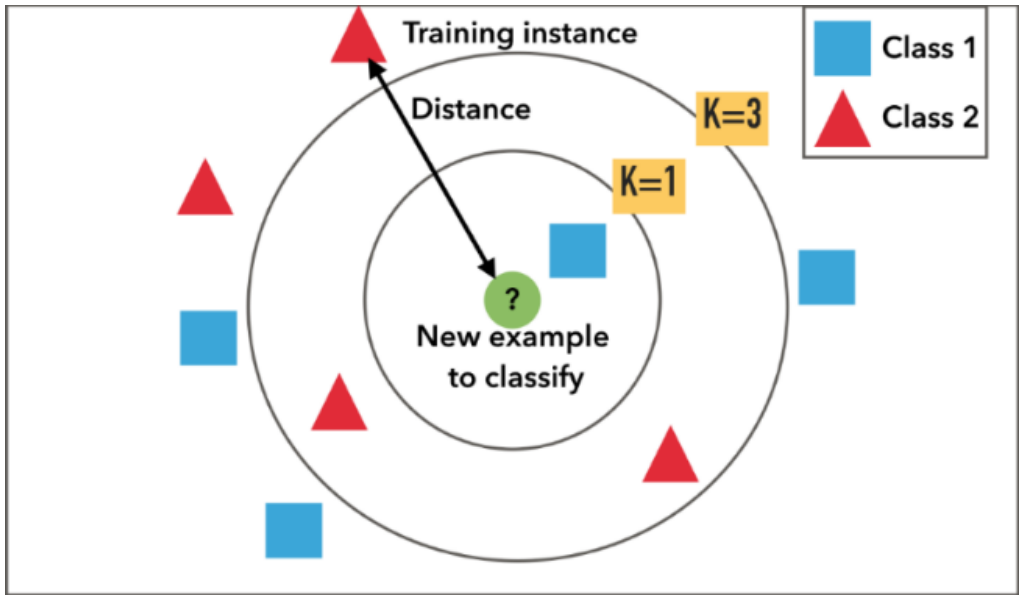
IF $Outlook = sunny$ THEN $PlayTennis = NO$
IF $Outlook = overcast$ THEN $PlayTennis = YES$
IF $Outook = rain$ THEN $PlayTennis = NO$

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)

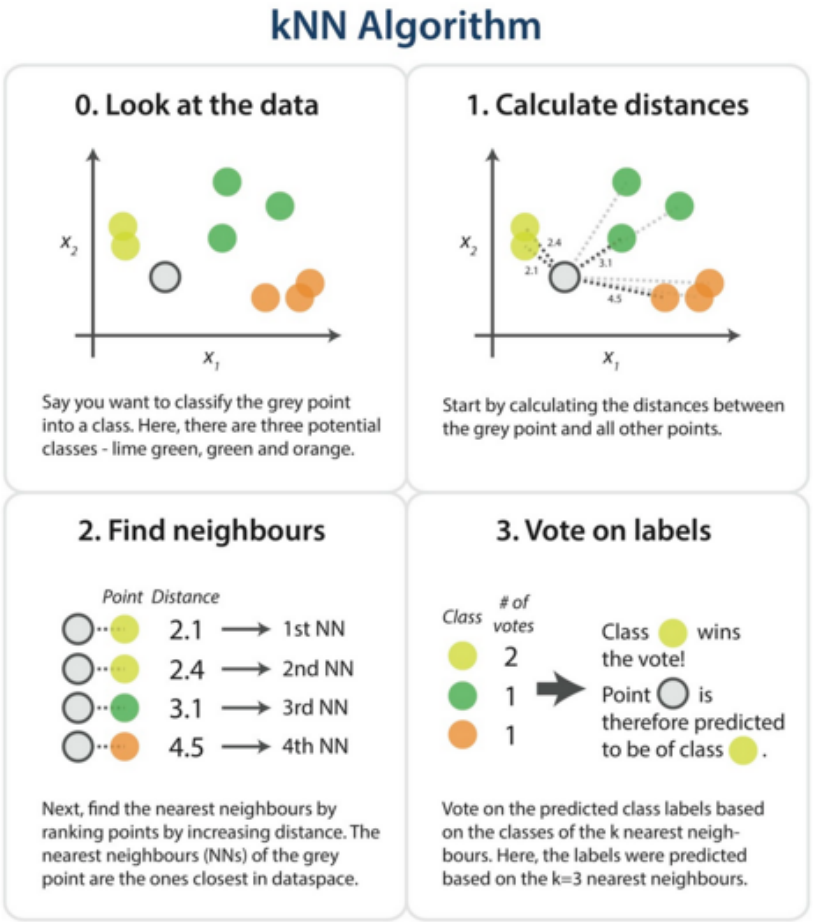
Metoda najbliższego sąsiada (1NN) jest jedną z najpopularniejszych metod klasyfikacji. Jej idea jest prosta i intuicyjna. Nowy obiekt otrzymuje klasę obiektu, który jest najbliżej (w uogólnieniu tej metody, czyli metodzie k najbliższych sąsiadów, klasę która występuje najczęściej pośród jego k sąsiadów). Należy podkreślić, że do oceny odległości może zostać wykorzystana bardzo szeroka klasa funkcji. Jest to metoda nieparametryczna, nie wymaga zatem żadnych założeń co do rozkładów danych w klasach.

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)



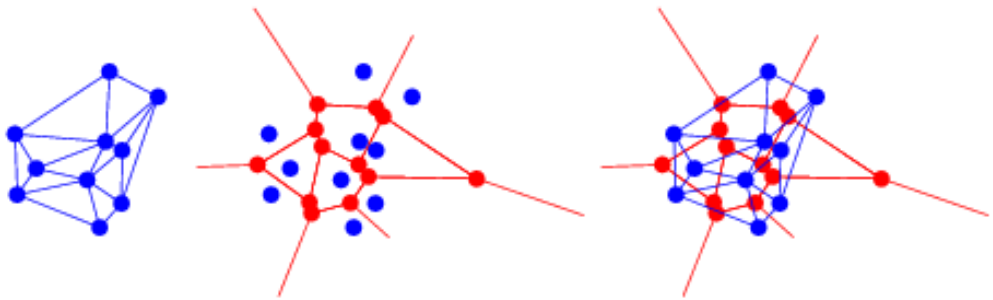


Metoda najbliższego sąsiada (*ang. nearest neighbor method*)



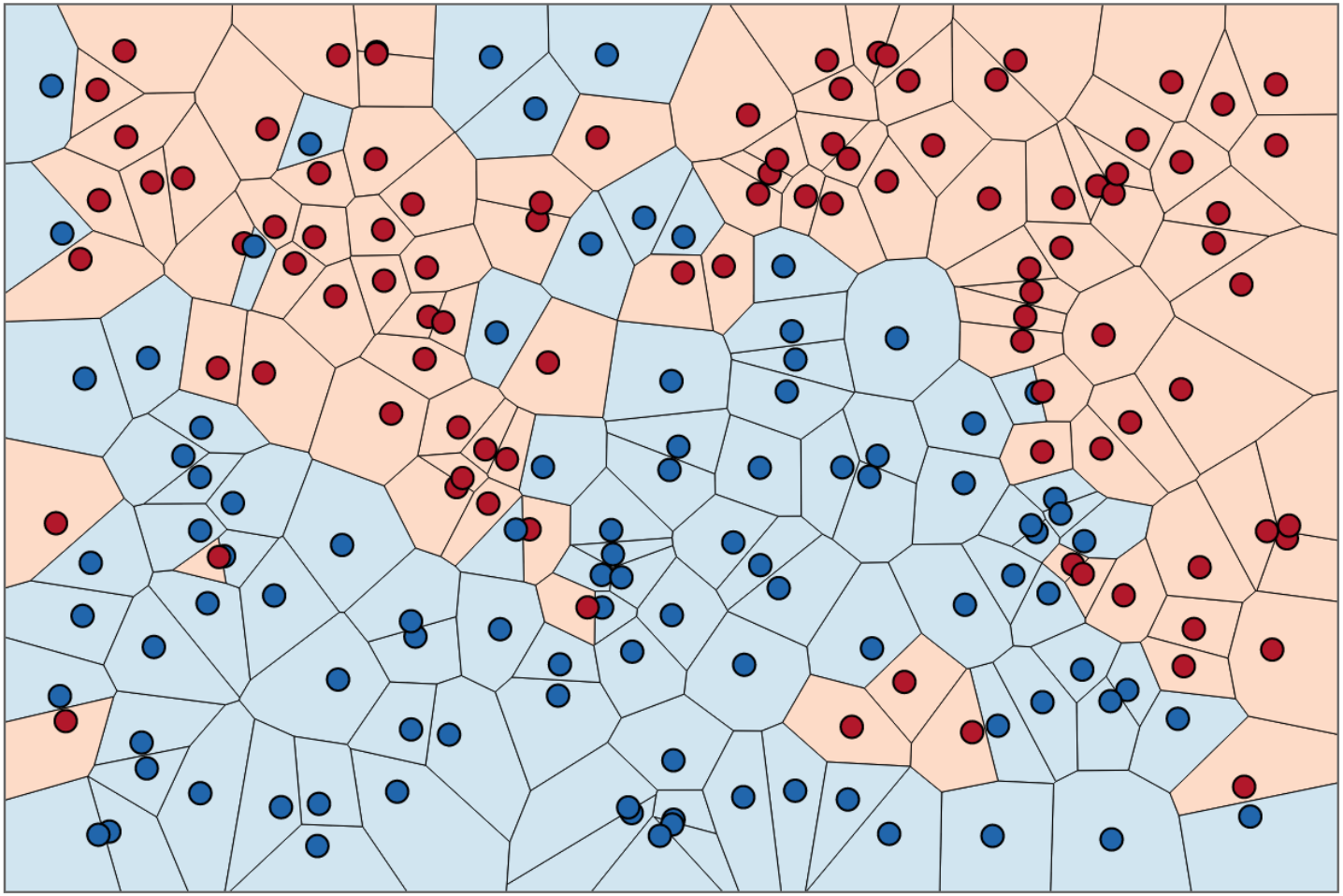
Metoda najbliższego sąsiada (*ang. nearest neighbor method*)

Idea metody może zostać zwizualizowana za pomocą diagramów Voronoia. Diagram Voronoia dzieli płaszczyznę na rozłączne, wypukłe komórki, które wewnątrz posiadają tylko jeden punkt z danych wejściowych. Krawędź komórki jest w równej odległości od dwóch sąsiednich punktów, tzn. że linia oddzielająca dwa punkty leży w połowie odległości między nimi. Łącząc punkty, których komórki mają wspólną krawędź otrzymujemy triangulację Delaunaya



Delaunay triangulation *Voronoi diagram* *Delaunay and Voronoi*

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)



Metoda najbliższego sąsiada (*ang. nearest neighbor method*)

Gieorgij Fieodosjewicz Woronoj
(1868-1908)

Borys Mikołajewicz Delaunay
(1890-1980)

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)

Metoda ta ma bardzo wysoką efektywność, gdy liczba obserwacji rośnie nieskończenie. Jednakże w wielu sytuacjach praktycznych liczba dostępnych obserwacji jest niewielka, co często prowadzi do drastycznego spadku efektywności metody najbliższych sąsiadów. Metoda najbliższych sąsiadów nie wymaga estymacji warunkowych funkcji gęstości, jest więc zdecydowanie prostsza w implementacji. Okazuje się, że nawet jeśli mamy metodę, która jest zgodna (tzn. asymptotycznie daje optymalny błąd bez względu na rozkłady cech w klasach), np. taką jak metoda najbliższych sąsiadów, jej efektywność na zbiorze skończonym może być zupełnie niewystarczająca.

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)

W praktyce wartość parametru k (liczba sąsiadów) dobiera się eksperymentalnie, stosując bądź to próbę testową bądź metodę sprawdzania krzyżowego. Metoda $1NN$ ma tendencję do bycia zbyt czułą na pewną losowość (błędne sklasyfikowanie lub błędny pomiar cechy) ukrytą w danych. Ponieważ pewne cechy, których wartości są duże mogą niwelować wpływ innych dlatego niezbędna jest normalizacja wartości cech. Dla zmiennych ciągłych używamy:

- normalizacja min-max

$$x = \frac{x - \min(x)}{\max(x) - \min(x)},$$

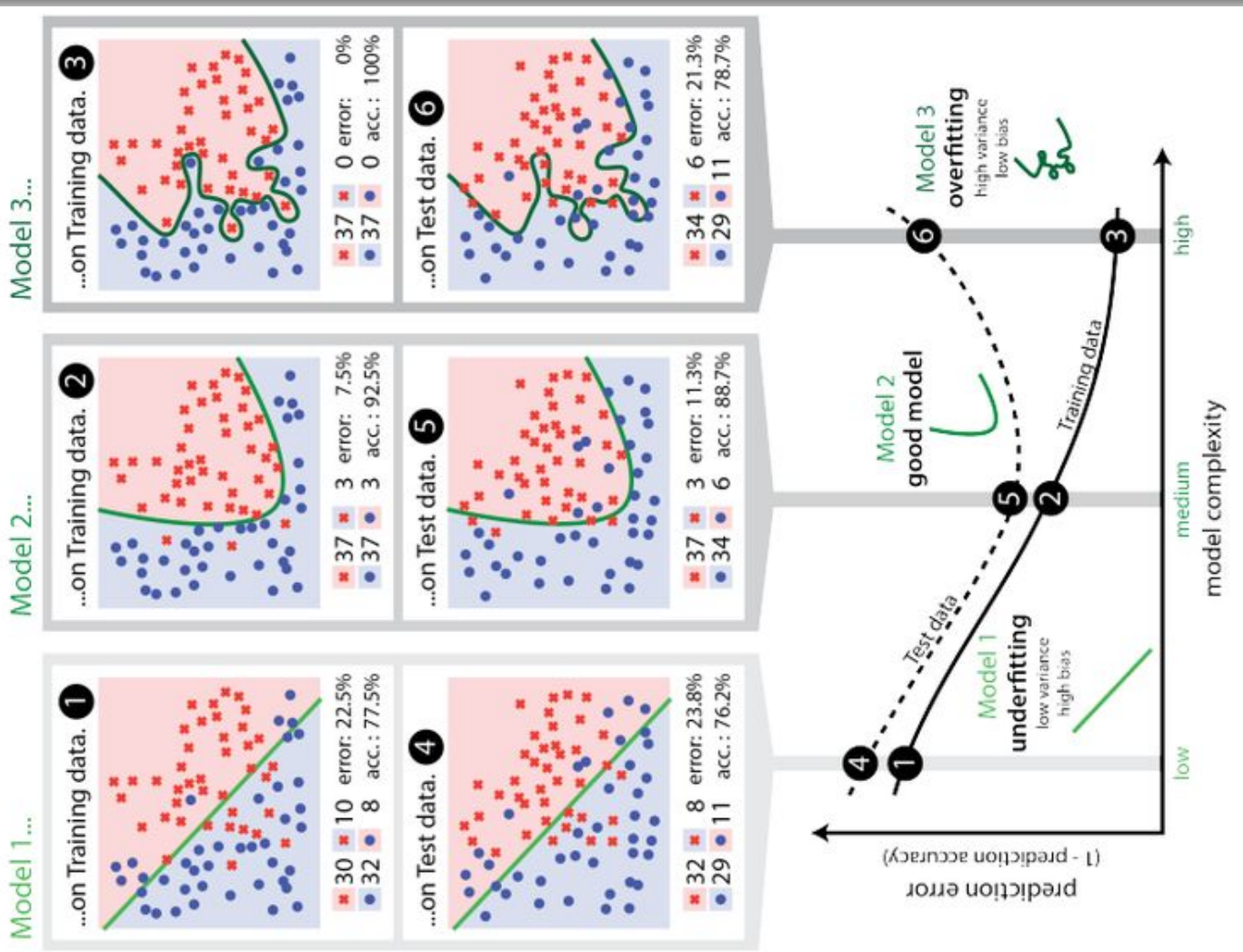
- standaryzacja

$$x = \frac{x - \bar{x}}{s(x)}$$

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)

Normalizacja jest niezbędna, ponieważ przeprowadza zmienne o różnych jednostkach w wielkości niemianowane i porównywalne. Metoda najbliższego sąsiada jest odporna na występowanie obserwacji odstających oraz zakłóceń. Niestety jej główną wadą jest długi czas obliczeń, który rośnie bardzo szybko wraz ze wzrostem liczby obserwacji. Nie istnieje również możliwość wykorzystywania zmiennych jakościowych, a wyniki klasyfikacji mocno zależą od wybranej miary odległości.

Metoda najbliższego sąsiada (*ang. nearest neighbor method*)



Przekleństwo wymiarowości (wymiar danych)

Miary odległości tracą swoją użyteczność w wysokich wymiarach. Zilustrujemy to na poniższym przykładzie. Niech hiperkula o promieniu r będzie wpisana w hipersześcian o boku $2r$. Wszystko umieszczone jest w przestrzeni p-wymiarowej. Wtedy:

$$\lim_{p\to\infty} \frac{V_{\text{hiperkula}}}{V_{\text{hipersześcian}}} = \lim_{d\to\infty} \frac{\pi^{p/2}}{p2^{p-1}\Gamma(p/2)} = 0.$$

Zatem jeśli wymiar rośnie to hiperkula staje się nieznaczącą częścią hipersześcianu. Ponadto, odległość pomiędzy środkiem i wierzchołkami ($r\sqrt{p}$) zwiększa się wraz ze wzrostem wymiaru dla ustalonego r. W tym sensie prawie wszystkie punkty są daleko od środka. Mówiąc inaczej prawie wszystkie punkty hipersześcianu znajdują się w rogach, a nie w środku.

LDA, QDA

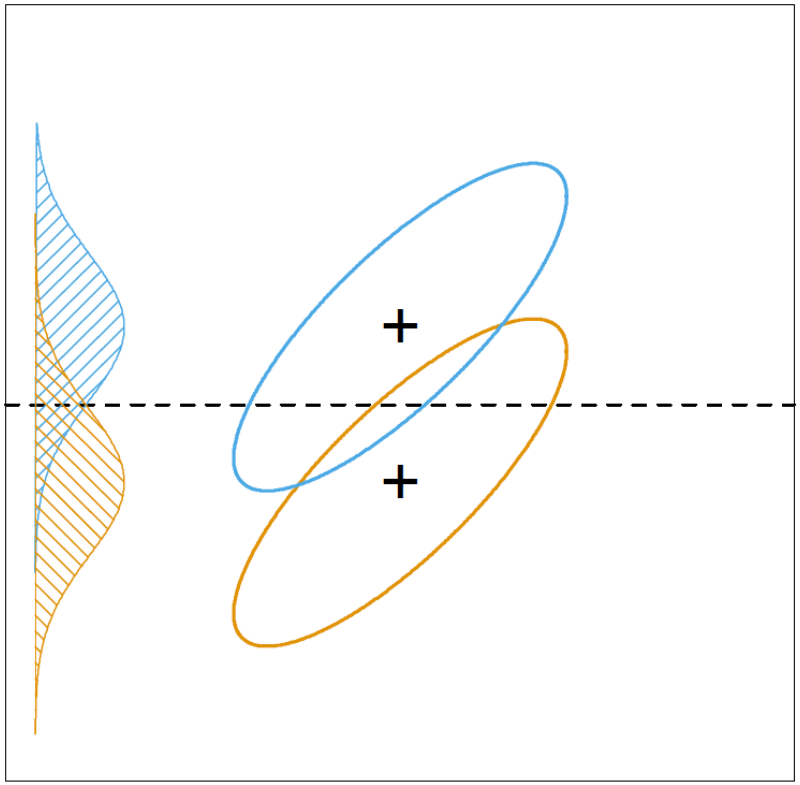
Przy założeniu, że dane we wszystkich klasach mają wielowymiarowy rozkład normalny najlepsze do klasyfikacji są metody bayesowskie. W przypadku gdy macierze kowariancji we wszystkich klasach są takie same powinniśmy użyć metody zwanej liniową analizą dyskryminacyjną – LDA (w przypadku gdy prawdopodobieństwa a priori przynależności do klas są równe i mamy do czynienia jedynie z dwoma klasami, metoda ta nazywana jest dyskryminacją liniową Fishera), w takiej sytuacji powierzchnie rozgraniczające są hiperpłaszczyznami. W przeciwnym razie, tzn. gdy macierze kowariancyjne różnią się, należy użyć kwadratowej analizy dyskryminacyjnej – QDA, która jako powierzchnie rozdzielające daje hiperpowierzchnie stopnia drugiego.

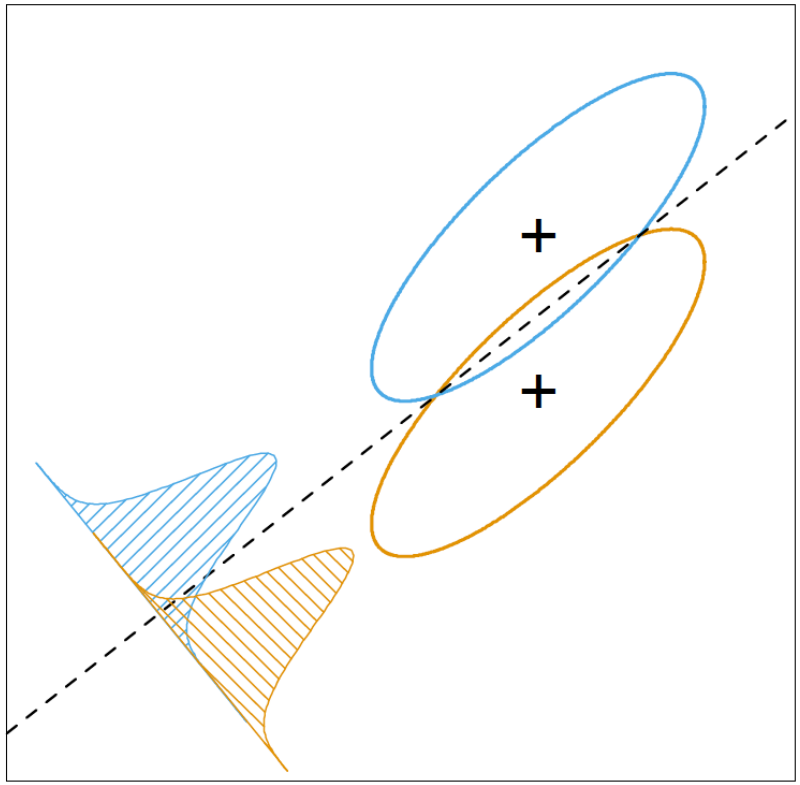
LDA, QDA

Pierwsza wersja liniowej analizy dyskryminacyjnej w przypadku dwóch grup pochodzi od FISHERA (1936). Metoda ta polegała na zredukowaniu wektora cech do jednego wymiaru poprzez zrzutowanie danych na prostą. Metodę tę można uogólnić i w efekcie otrzymać nowe zmienne, zwane zmiennymi dyskryminacyjnymi (*ang. discriminant coordinates*). Zmienne dyskryminacyjne są nieskorelowane, jednakże nie są one ortogonalne. W praktyce jednak rysuje się zmienne dyskryminacyjne w prostokątnym układzie współrzędnych. Liczba zmiennych dyskryminacyjnych jest równa $\min(K-1, p)$. Można ich używać w podobny sposób jak składowych głównych.

LDA, QDA

Graficzna prezentacja idei zmiennych dyskryminacyjnych.





Zbiór danych `iris`

Zbiór danych `iris flower` jest wielowymiarowym zbiorem danych wprowadzonym przez FISHERA (1936). Składa się ze 150 obserwacji, po 50 z trzech gatunków kwiatów irysa (setosa, virginica and versicolor). Mierzone były 4 cechy (w centymetrach): the length and the width of the sepals and petals, in centimeters. Bazując na nim Fisher zaproponował liniowy model klasyfikacji odróżniający te trzy gatunki irysa.

Sir Ronald Aylmer Fisher (1890-1962)

Zbiór danych `iris`

Iris Versicolor **Iris Setosa** **Iris Virginica**

Fisher R. A. (1936). *The use of multiple measurements in taxonomic problems*. Annals of Eugenics 7(2):179–188.

Naiwny klasyfikator bayesowski (*ang. naïve Bayes classifier*)

Naïve Bayes Classifier

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

Thomas Bayes
1702 - 1761

Naiwny klasyfikator bayesowski (*ang. naïve Bayes classifier*)

Jest to prosty klasyfikator probabilistyczny oparty na założeniu, że cechy opisujące obiekty są wzajemnie niezależne. Założenie to nie ma raczej nic wspólnego z rzeczywistością i właśnie z tego powodu metoda nazywana jest naiwną. Pomimo tego klasyfikator ten często działa lepiej niż można się po nim było spodziewać (zwłaszcza jeśli jest dużo cech). W praktyce estymuje się gęstość każdej cechy w każdej klasie, a następnie bada iloczyn takich gęstości dla każdej klasy. Obserwacja klasyfikowana jest do klasy, dla której ten iloczyn jest największy.

$$\bigvee_{k=1,\ldots,K} \mathbb{P}(G_k \mid \boldsymbol{x}) = \frac{\mathbb{P}(G_k)\mathbb{P}(\boldsymbol{x} \mid G_k)}{\mathbb{P}(\boldsymbol{x})} = \frac{\mathbb{P}(G_k, \boldsymbol{x})}{\mathbb{P}(\boldsymbol{x})} \propto \mathbb{P}(x_1 \mid x_2, \ldots, x_p, G_k) \cdot$$

$$\cdot \mathbb{P}(x_2 \mid x_3, \ldots, x_p, G_k) \cdots \mathbb{P}(x_p \mid G_k) \cdot \mathbb{P}(G_k) = \mathbb{P}(G_k) \prod_{i=1}^{p} \mathbb{P}(x_i \mid G_k).$$

Naiwny klasyfikator bayesowski (*ang. naïve Bayes classifier*)

Chcemy zaklasyfikować obserwację:

$$\boldsymbol{x} = (\text{Age} = \text{21-30}, \text{Income} = \text{Medium}, \text{Status} = \text{Married}).$$

Age	Income	Status	Buy
<=20	low	students	yes
<=20	high	students	yes
<=20	medium	students	no
<=20	medium	married	no
<=20	high	married	yes
21 - 30	low	married	yes
21 - 30	low	married	no
21 - 30	medium	students	no
21 - 30	medium	married	no
21 - 30	high	students	yes
>30	high	married	no
>30	high	married	yes
>30	medium	married	yes
>30	medium	married	no
>30	low	students	no

Naiwny klasyfikator bayesowski (*ang. naïve Bayes classifier*)

1. Prawdopodobieństwa a priori

$$\mathbb{P}(\text{Buy} = \text{yes}) = \mathbb{P}(G_1) = 7/15$$
$$\mathbb{P}(\text{Buy} = \text{no}) = \mathbb{P}(G_2) = 8/15$$

2. Prawdopodobieństwa warunkowe cech

$$\mathbb{P}(\text{Age} = \text{21-30}|\text{Buy} = \text{yes}) = \mathbb{P}(x_1|G_1) = 2/7$$
$$\mathbb{P}(\text{Age} = \text{21-30}|\text{Buy} = \text{no}) = \mathbb{P}(x_1|G_2) = 3/8$$
$$\mathbb{P}(\text{Income} = \text{Medium}|\text{Buy} = \text{yes}) = \mathbb{P}(x_2|G_1) = 1/7$$
$$\mathbb{P}(\text{Income} = \text{Medium}|\text{Buy} = \text{no}) = \mathbb{P}(x_2|G_2) = 5/8$$
$$\mathbb{P}(\text{Status} = \text{Married}|\text{Buy} = \text{yes}) = \mathbb{P}(x_3|G_1) = 4/7$$
$$\mathbb{P}(\text{Status} = \text{Married}|\text{Buy} = \text{no}) = \mathbb{P}(x_3|G_2) = 5/8$$

Naiwny klasyfikator bayesowski (*ang. naïve Bayes classifier*)

3. Prawdopodobieństwa całkowite

$$\mathbb{P}(\boldsymbol{x}|\text{Buy} = \text{yes}) = \prod_{i=1}^{3} \mathbb{P}(x_i|G_1) = 2/7 \cdot 1/7 \cdot 4/7 = 0.0233$$

$$\mathbb{P}(\boldsymbol{x}|\text{Buy} = \text{no}) = \prod_{i=1}^{3} \mathbb{P}(x_i|G_2) = 3/8 \cdot 5/8 \cdot 5/8 = 0.1464$$

4. Prawdopodobieństwa a posteriori

$$\mathbb{P}(\text{Buy} = \text{yes}|\boldsymbol{x}) = \mathbb{P}(G_1|\boldsymbol{x}) = \mathbb{P}(G_1) \prod_{i=1}^{3} \mathbb{P}(x_i|G_1) = 7/15 \cdot 0.0233 = 0.0108$$

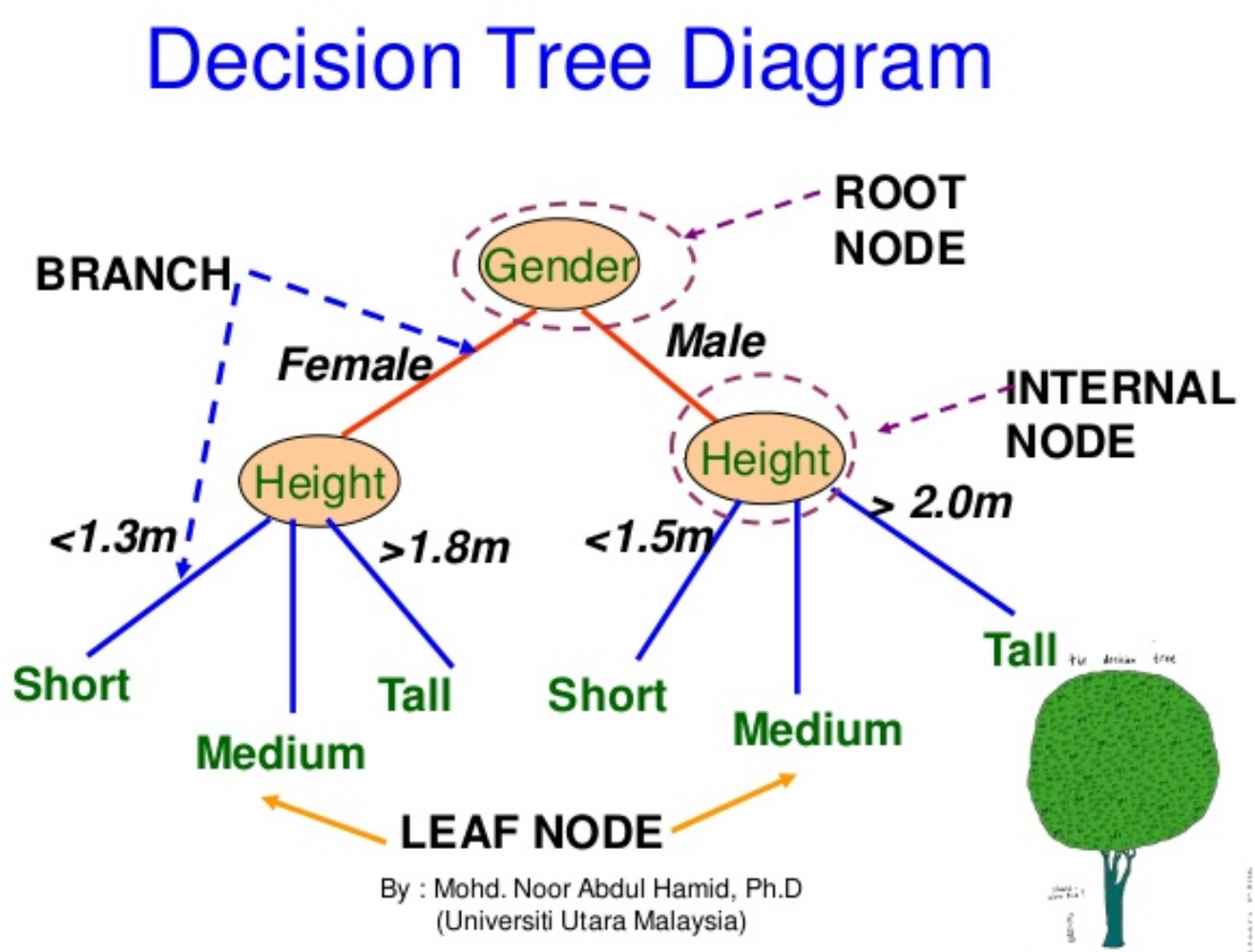
$$\mathbb{P}(\text{Buy} = \text{no}|\boldsymbol{x}) = \mathbb{P}(G_2|\boldsymbol{x}) = \mathbb{P}(G_2) \prod_{i=1}^{3} \mathbb{P}(x_i|G_2) = 8/15 \cdot 0.1464 = \mathbf{0.0781}$$

Zatem obserwacja powinna zostać zaklasyfikowana jako `Buy = no`.

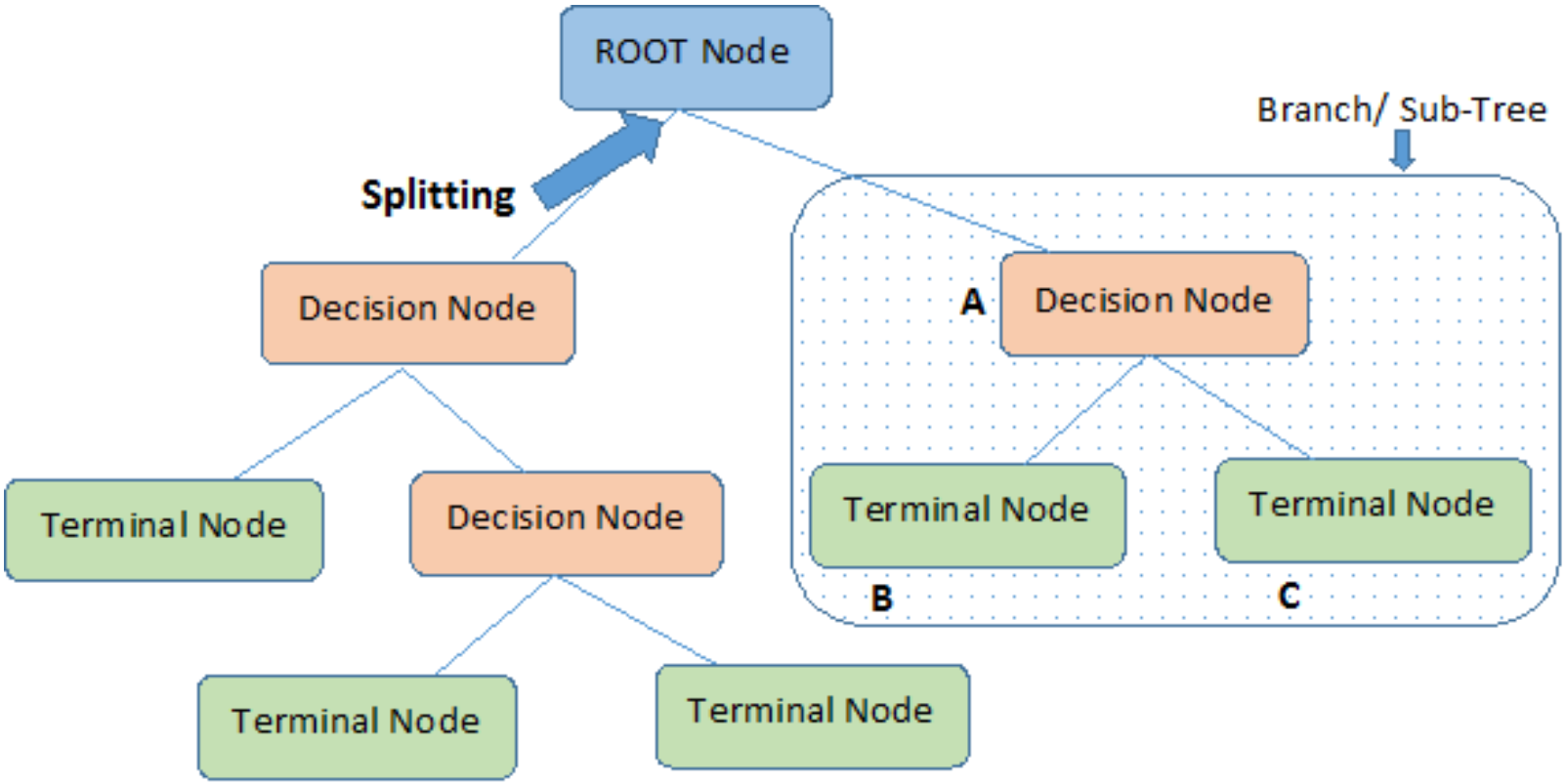
Drzewa klasyfikacyjne (*ang. classification trees*)

Drzewa klasyfikacyjne – rodzina metod statystycznych z zakresu eksploracji danych, dokonujących, za pomocą diagramów zwanych drzewami, klasyfikacji danych. Drzewo składa się z korzenia oraz gałęzi prowadzących z korzenia do kolejnych węzłów. W każdym węźle sprawdzany jest pewien warunek dotyczący danej obserwacji, i na jego podstawie wybierana jest jedna z gałęzi prowadząca do kolejnego węzła piętro niżej. Na dole znajdują się liście, w których odczytujemy do której z klas należy przypisać daną obserwację. Klasyfikacja danej obserwacji polega na przejściu od korzenia do liścia i przypisaniu do tej obserwacji klasy zapisanej w danym liściu.

Drzewa klasyfikacyjne (*ang. classification trees*)

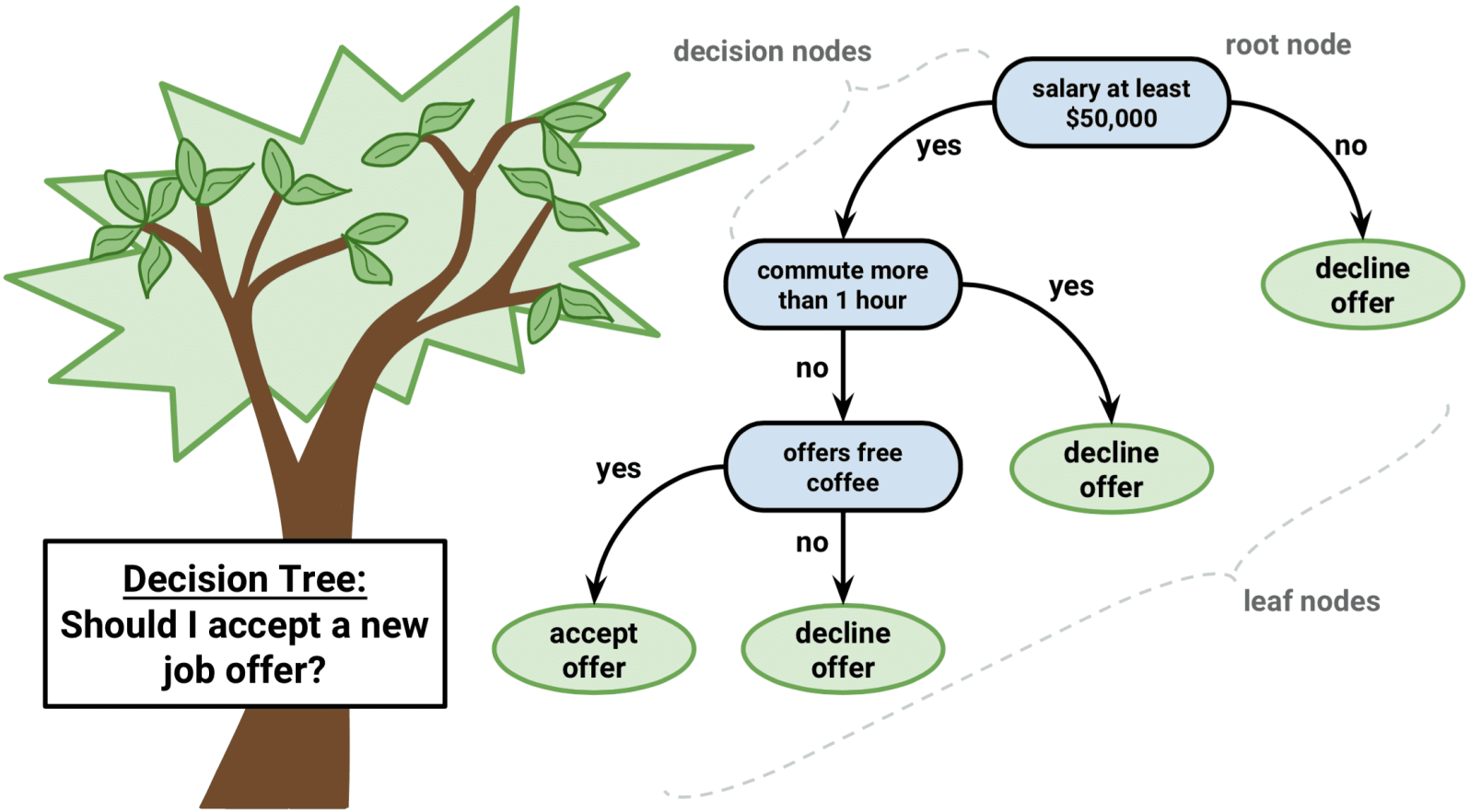


Drzewa klasyfikacyjne (*ang. classification trees*)



Note:- A is parent node of B and C.

Drzewa klasyfikacyjne (*ang. classification trees*)



Drzewa klasyfikacyjne (*ang. classification trees*)

Każdy podział s jest uwarunkowany przez obserwacje ze zbioru uczącego $\mathcal{L}_n$, należące do danego węzła t. Praktycznie trzeba zatem rozdzielić je na dwa podzbiory możliwie jednorodne ze względu na etykietę klasy. Gdyby zatem w węźle t znajdowały się obserwacje tylko z dwóch klas, to idealnym byłby taki podział, który przypisałby obserwacje uczące z jednej klasy do węzła t_L, a z drugiej klasy do węzła t_R. Dla każdego węzła t określmy pewną miarę $I(t)$ niejednorodności elementów w tym węźle. Stąd, dla każdego podziału s węzła t będziemy mogli zmierzyć niejednorodność elementów w tym węźle oraz w jego potomkach t_L i t_R.

Drzewa klasyfikacyjne (*ang. classification trees*)

Niech ϕ oznacza funkcję określoną dla wszystkich K-elementowych ciągów prawdopodobieństw $(p_1, p_2, \ldots, p_K)$ takich, że

$$\sum_{i=1}^{K} p_i = 1, \;\; p_i \geq 0, \;\; i = 1, 2, \ldots, K,$$

spełniającą następujące warunki:

1. funkcja ϕ osiąga maksimum tylko w punkcie $(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$,
2. funkcja ϕ osiąga minimum tylko w punktach: $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$, $\ldots$, $(0, 0, \ldots, 1)$,
3. funkcja ϕ jest symetryczną funkcją swoich argumentów.

Miarę niejednorodności $I(t)$ w węźle t definiujemy następująco:

$$I(t) = \phi(p(1 \mid t), \ldots, p(K \mid t)).$$

Drzewa klasyfikacyjne (*ang. classification trees*)

1. Błąd klasyfikacji:

$$\phi_1(p_1, \ldots, p_K) = 1 - \max\{p_1, \ldots, p_K\}.$$

2. Entropia:

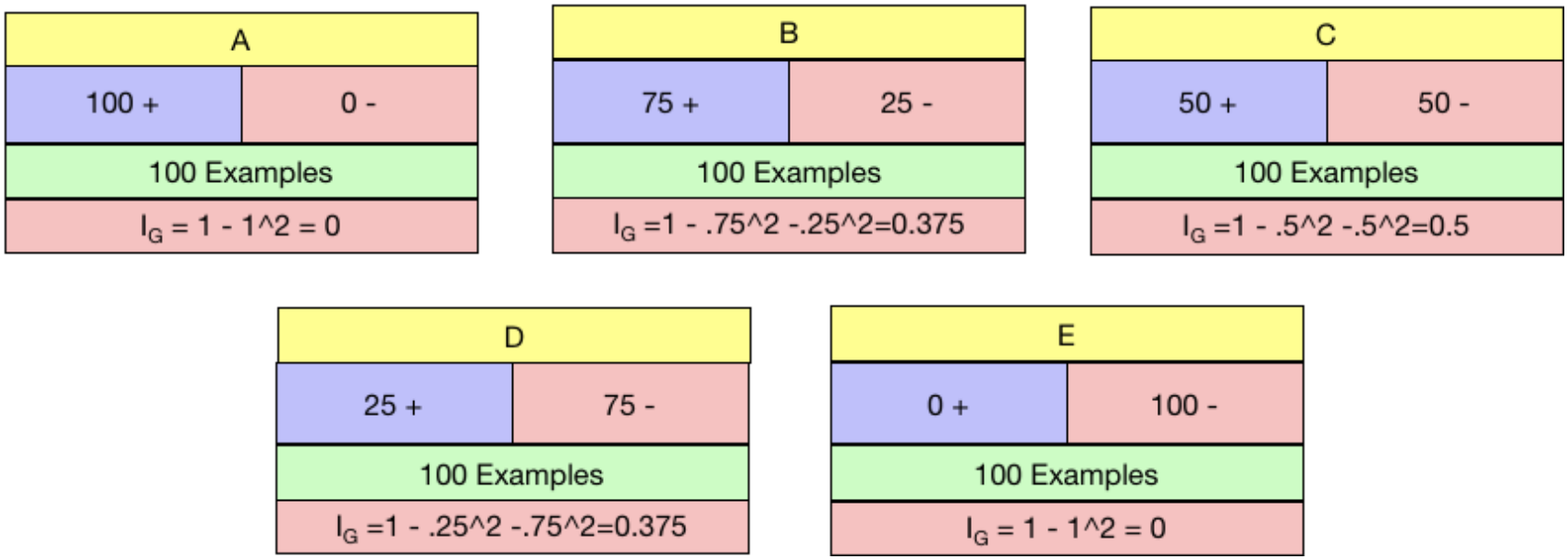
$$\phi_2(p_1, \ldots, p_K) = E(p_1, \ldots, p_K) = -\sum_{i=1}^{K} p_i \log p_i.$$

3. Indeks Giniego:

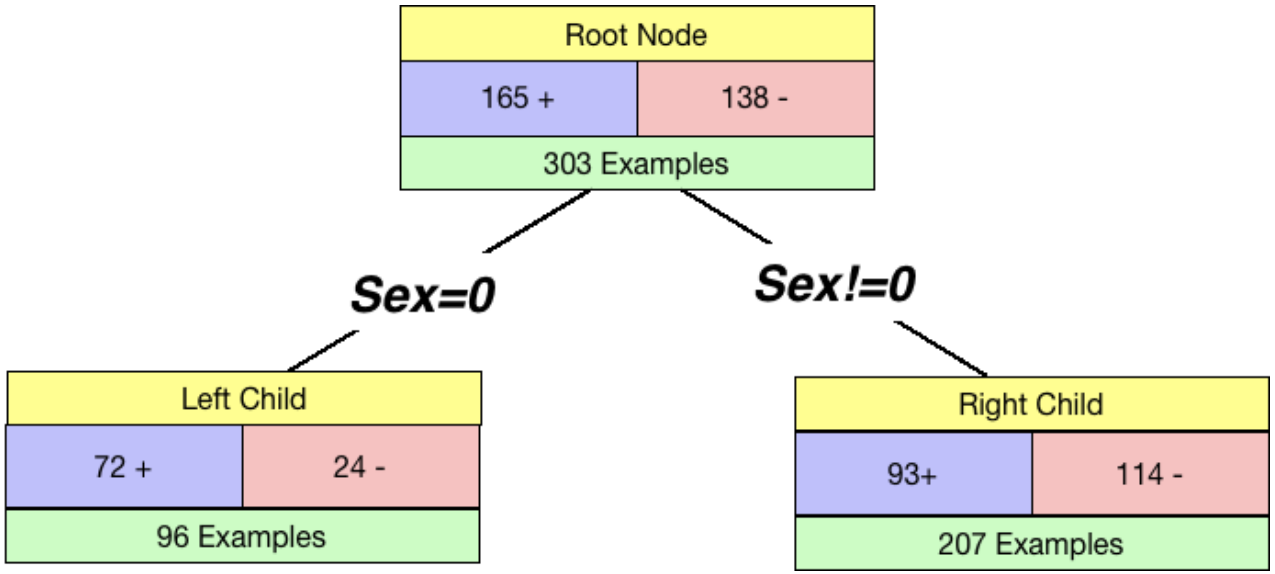
$$\phi_3(p_1, \ldots, p_K) = I_G(p_1, \ldots, p_K) = 1 - \sum_{i=1}^{K} p_i^2.$$

Wybierany jest taki podział, który daje maksymalną redukcję niejednorodności indeksu przynależności do klasy w węźle.

Drzewa klasyfikacyjne (*ang. classification trees*)



Drzewa klasyfikacyjne (*ang. classification trees*)

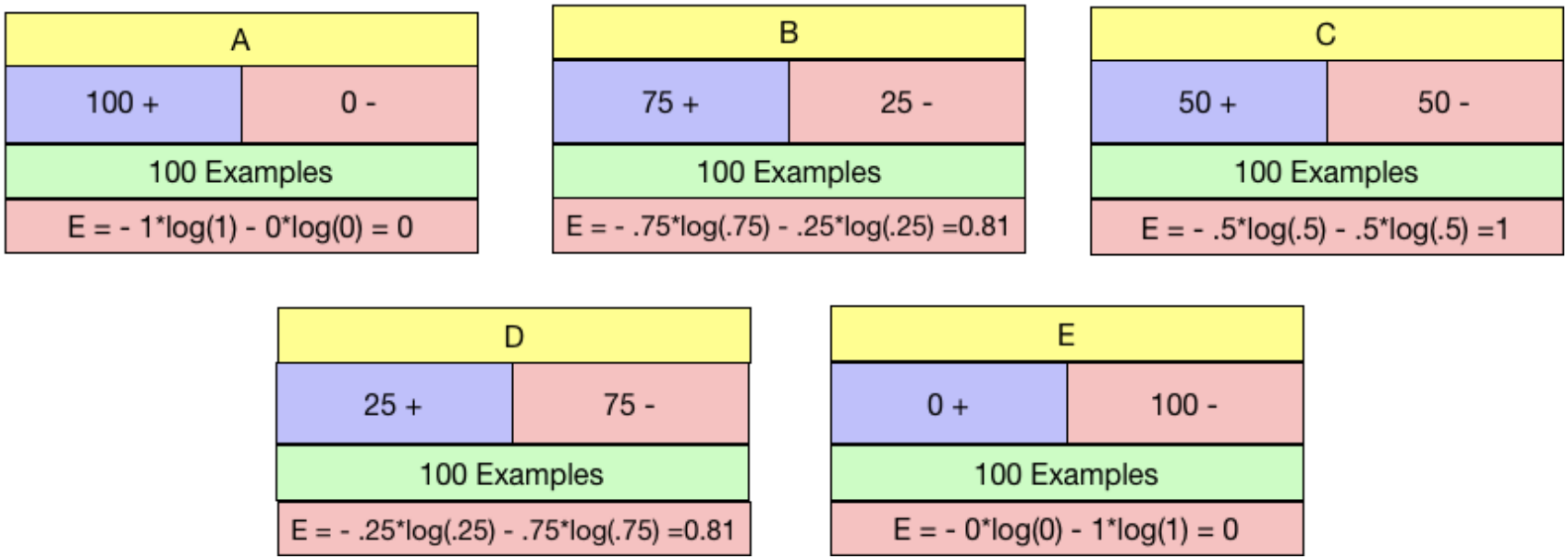


$$I_G(\text{Left}) = 1 - (72/96)^2 - (24/96)^2$$

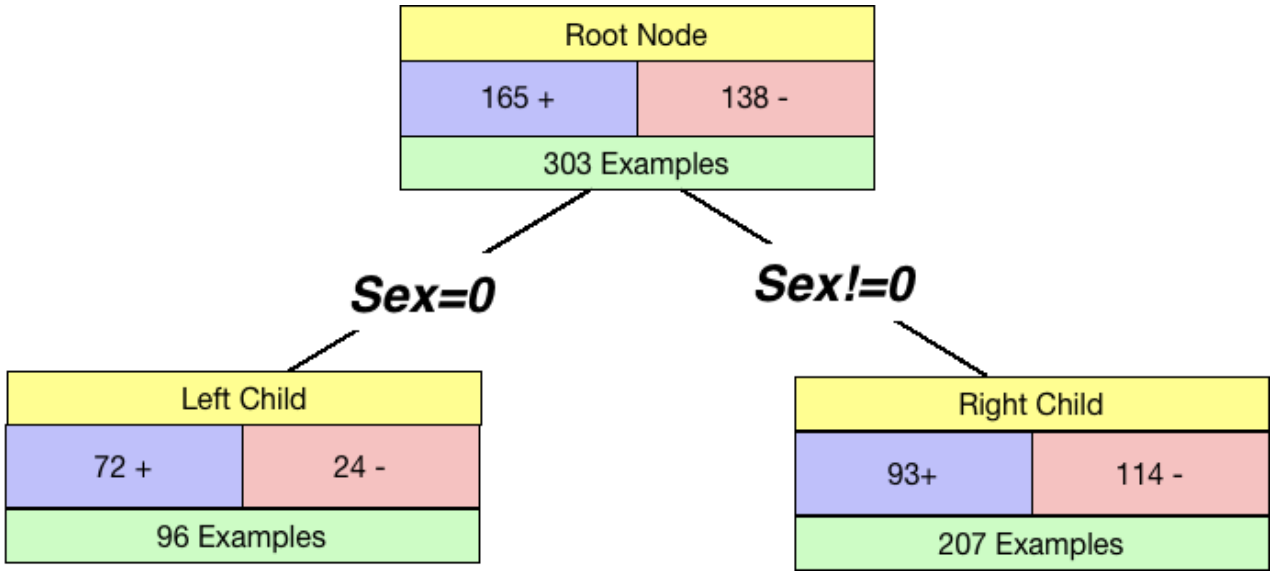
$$I_G(\text{Right}) = 1 - (93/207)^2 - (114/207)^2$$

$$I_G = 96/(96+207) * I_G(\text{Left}) + 207/(96+207) * I_G(\text{Right})$$

Drzewa klasyfikacyjne (*ang. classification trees*)



Drzewa klasyfikacyjne (*ang. classification trees*)

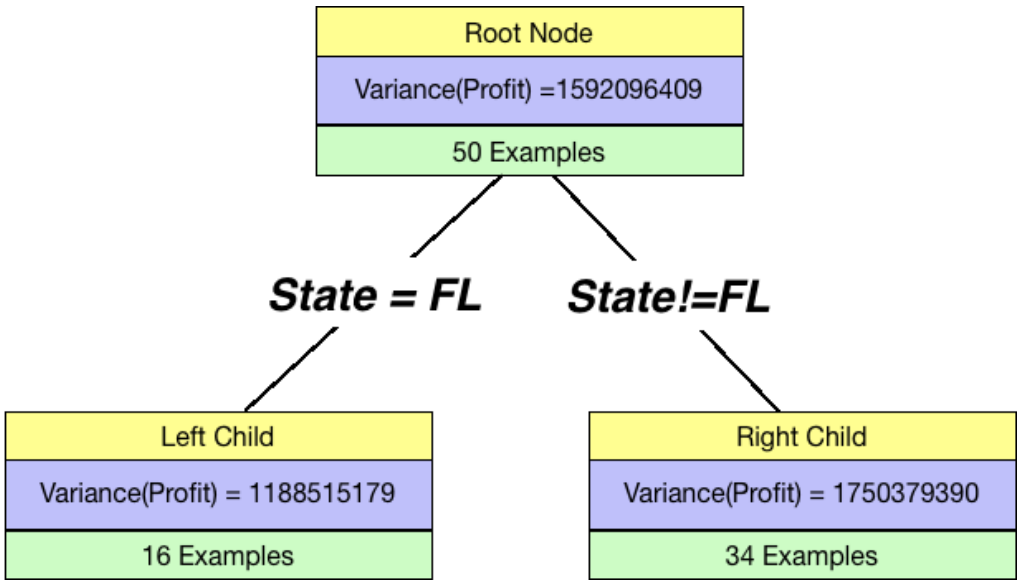


$$E(\text{Left}) = -(72/96) * \log(72/92) - (24/96) * \log(24/96)$$
$$E(\text{Right}) = -(93/207) * \log(93/207) - (114/207) * \log(114/207)$$
$$E = 96/(96 + 207) * E(\text{Left}) + 207/(96 + 207) * E(\text{Right})$$

Drzewa klasyfikacyjne (*ang. classification trees*)

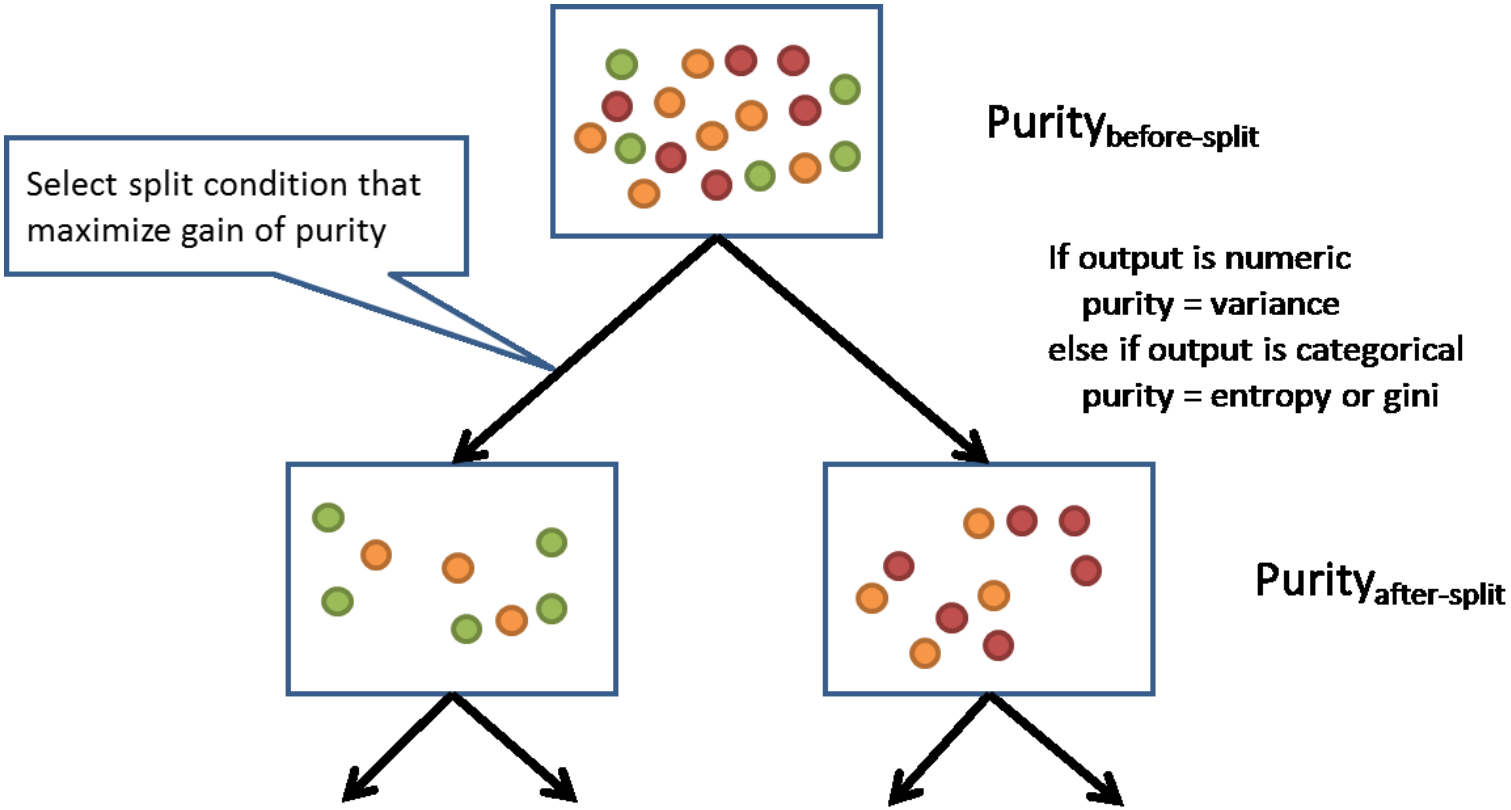
Regresja

Powyższe miary dobrze działają w przypadku klasyfikacji. Dla regresji często używana jest wariancja



$$\mathbb{V}\text{ar} = 14/(14+36) * \mathbb{V}\text{ar}(\text{Left}) + 36/(14+36) * \mathbb{V}\text{ar}(\text{Right})$$

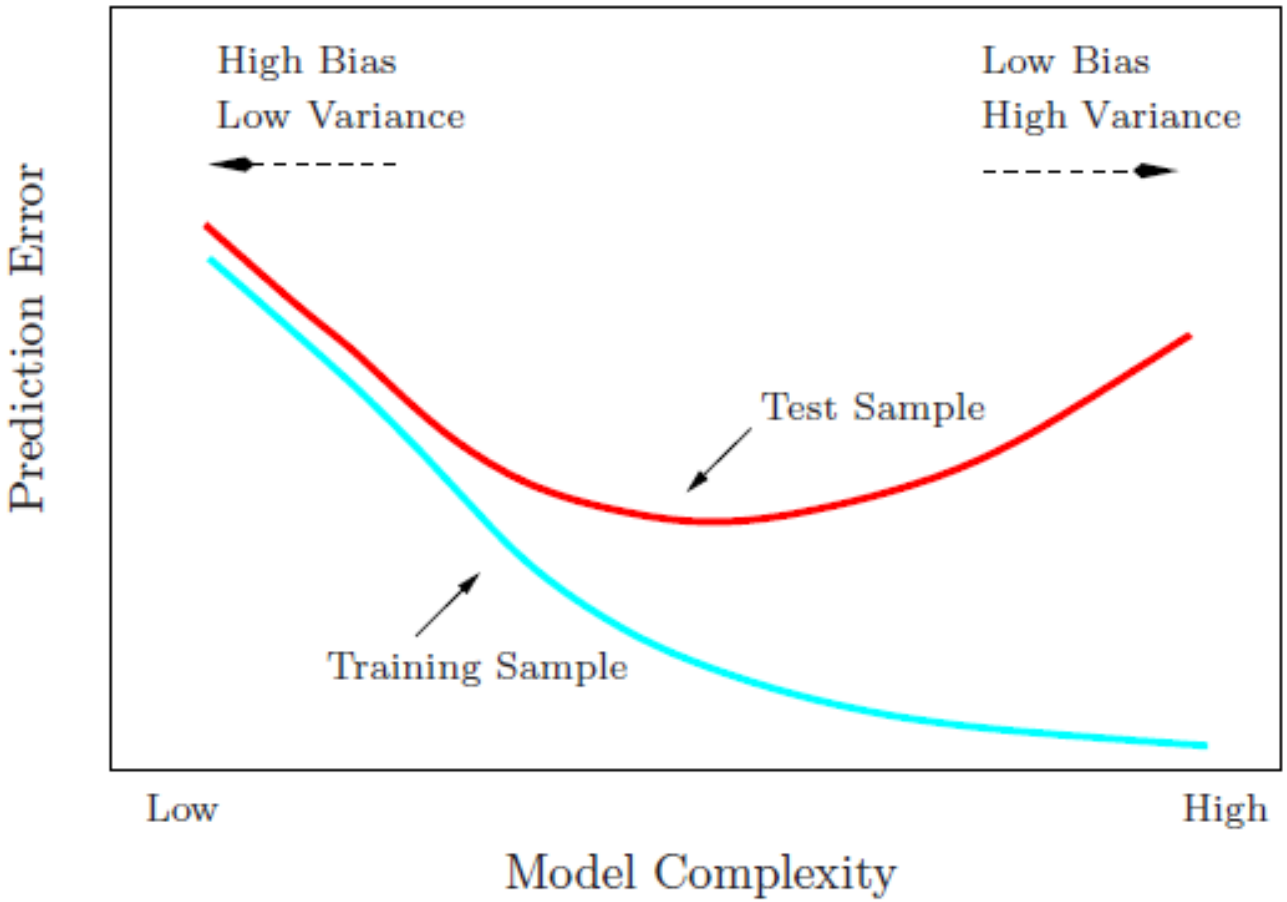
Drzewa klasyfikacyjne (*ang. classification trees*)



Drzewa klasyfikacyjne (*ang. classification trees*)

Przy tworzeniu drzewa należy unikać zbytniego rozbudowania struktury drzewa, ponieważ wzrasta złożoność opartego na nim modelu, co w konsekwencji prowadzi do trudności w jego interpretacji oraz utraty właściwości generalizacji. Budując zbyt duże drzewo mamy do czynienia z tzw. efektem przeuczenia (*ang. overfitting*). Polega on na tym, że drzewo doskonale klasyfikuje obiekty z próby uczącej lecz coraz słabiej (w miarę zwiększania liczby liści) nowe elementy. Aby tego uniknąć wpierw konstruuje się drzewa maksymalnie złożone, a następnie stosuje technikę zwaną przycinaniem (*ang. pruning*), która zmniejsza drzewo.

Drzewa klasyfikacyjne (*ang. classification trees*)

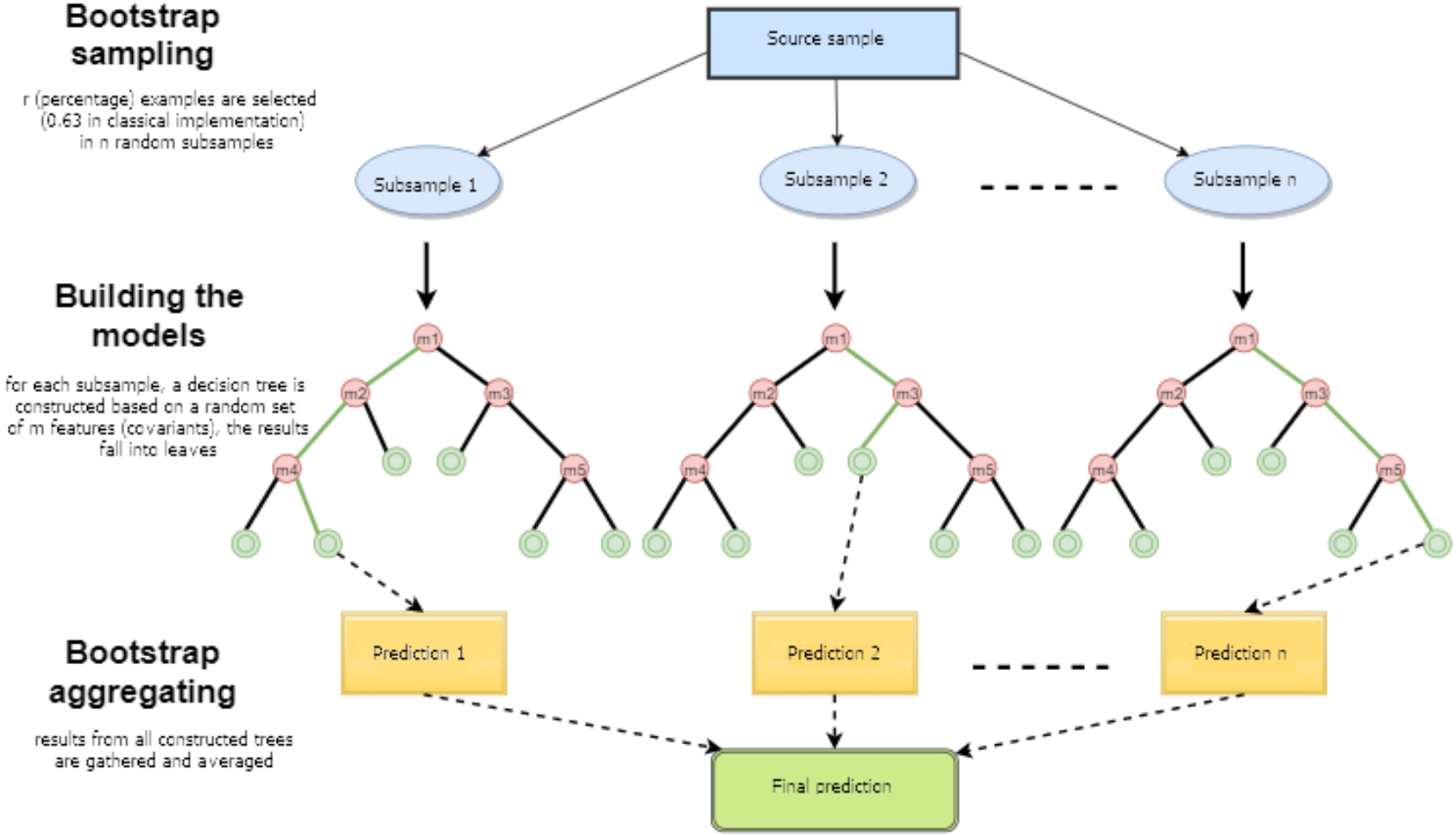


Las losowy (*ang. random forest*)

Jest to metoda łączenia wielu drzew klasyfikacyjnych. Wpierw losujemy K prób bootstrapowych (im więcej tym lepiej), dla każdej z nich konstruujemy drzewo klasyfikacyjne w taki sposób, że w każdym węźle losujemy m cech, które będą uczestniczyły w wyborze najlepszego podziału. Drzewa budowane są bez przycinania. Ostatecznie obserwacja klasyfikowana jest poprzez metodę głosowania (*ang. majority voting*). Parametr m powinien być znacznie mniejszy od wymiaru danych p, przyjmuje się najczęściej jego wartość równą $\sqrt{p}$ (klasyfikacja) lub $p/3$ (regresja). Lasy losowe implementują ideę baggingu (*ang. bagging*).

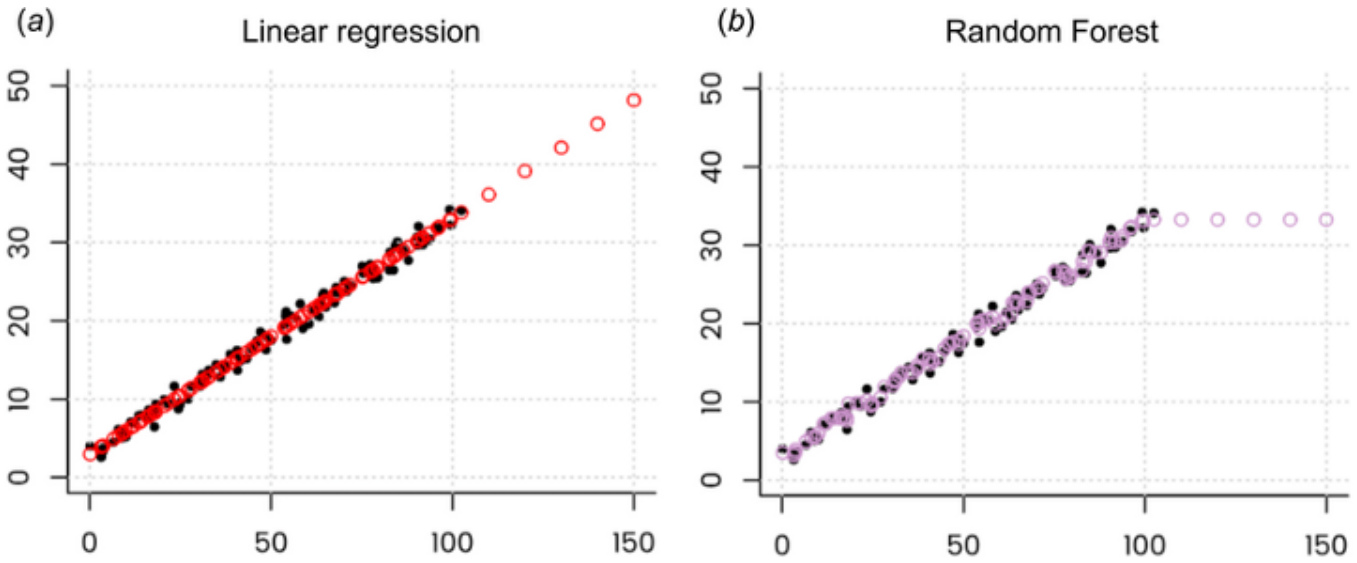
Breiman, L. (2001). Random Forests. Machine Learning 45(1):5–32.

Las losowy (*ang. random forest*)



Las losowy (*ang. random forest*)

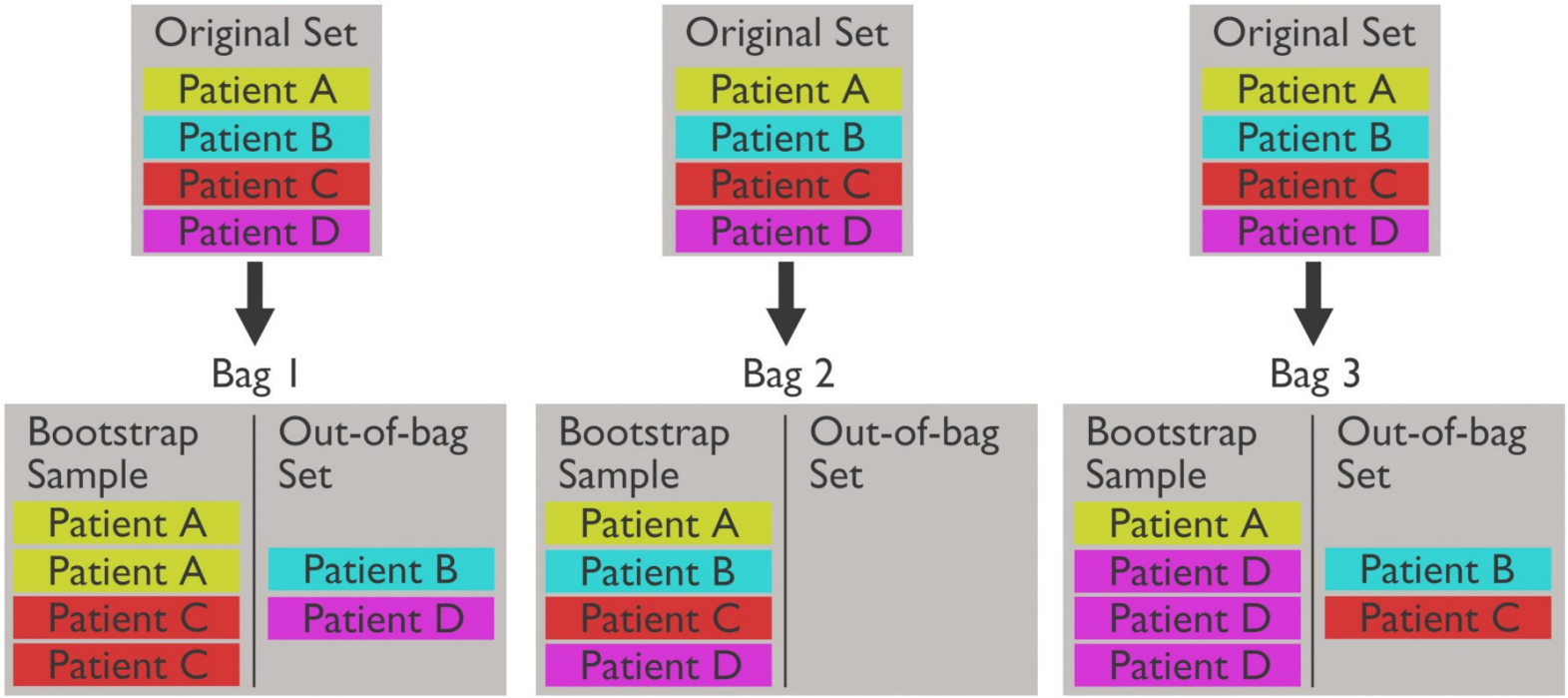
RF nie jest w stanie odkryć trendów, które zmusiłyby go do przewidywania wartości spoza znanego przedziału (ekstrapolacji). Wynika to z tego, że w drzewie losowym w celu wykonania prognozy uśredniamy wartości w liściu. W celu uniknięcia tego problemu można zastosować rozszerzone lasy losowe (*ang. Regression-Enhanced Random Forests (RERFs)*).



Las losowy (*ang. random forest*)

W przypadku lasów losowych istnieje specyficzna metoda oceny jakości zwana błędem out of bag (OOB). Zauważmy, że każda obserwacja jest używana do uczenia jedynie przez około 63.2% drzew. Jeśli teraz zaklasyfikujemy obserwacje, które nie należą do zbioru uczącego dla konkretnego drzewa, to otrzymamy dla każdej obserwacji przewidywane klasy prze te drzewa, które nie wykorzystywały jej w procesie uczenia. Ostateczną oceną jest klasa, która została wybrana przez większość drzew. W taki sposób dla każdej obserwacji otrzymamy ocenę klasy, z której pochodzi. Wyliczając teraz procent błędnych decyzji otrzymamy właśnie błąd OOB. Błąd OOB zbiega do błędu LOO CV, ale jest dużo tańszy obliczeniowo.

Las losowy (*ang. random forest*)



Boosting

Metoda przekształcania słabych klasyfikatorów w mocne:

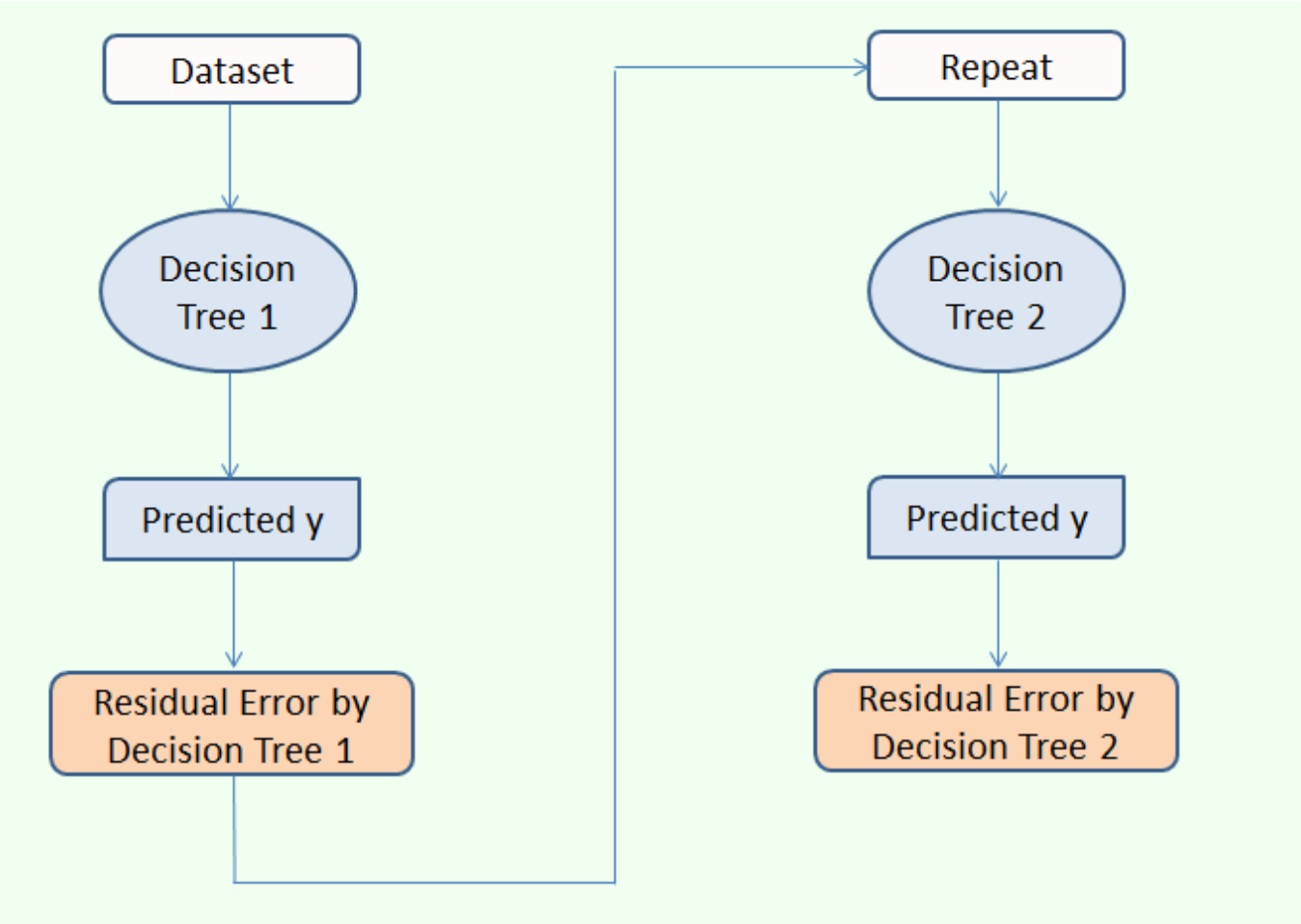
1. Konstruujemy pierwszy klasyfikator na wszystkich dostępnych obserwacjach.
2. Jeśli, któraś z obserwacji została błędnie zaklasyfikowana będzie miała większe wagi w kolejnych iteracjach. Stosujemy kolejny klasyfikator (może być ponownie ten sam).
3. Powtarzamy krok drugi aż do osiągnięcia reguły stopu.
4. Łączymy decyzje z wszystkich klasyfikatorów.

Boosting

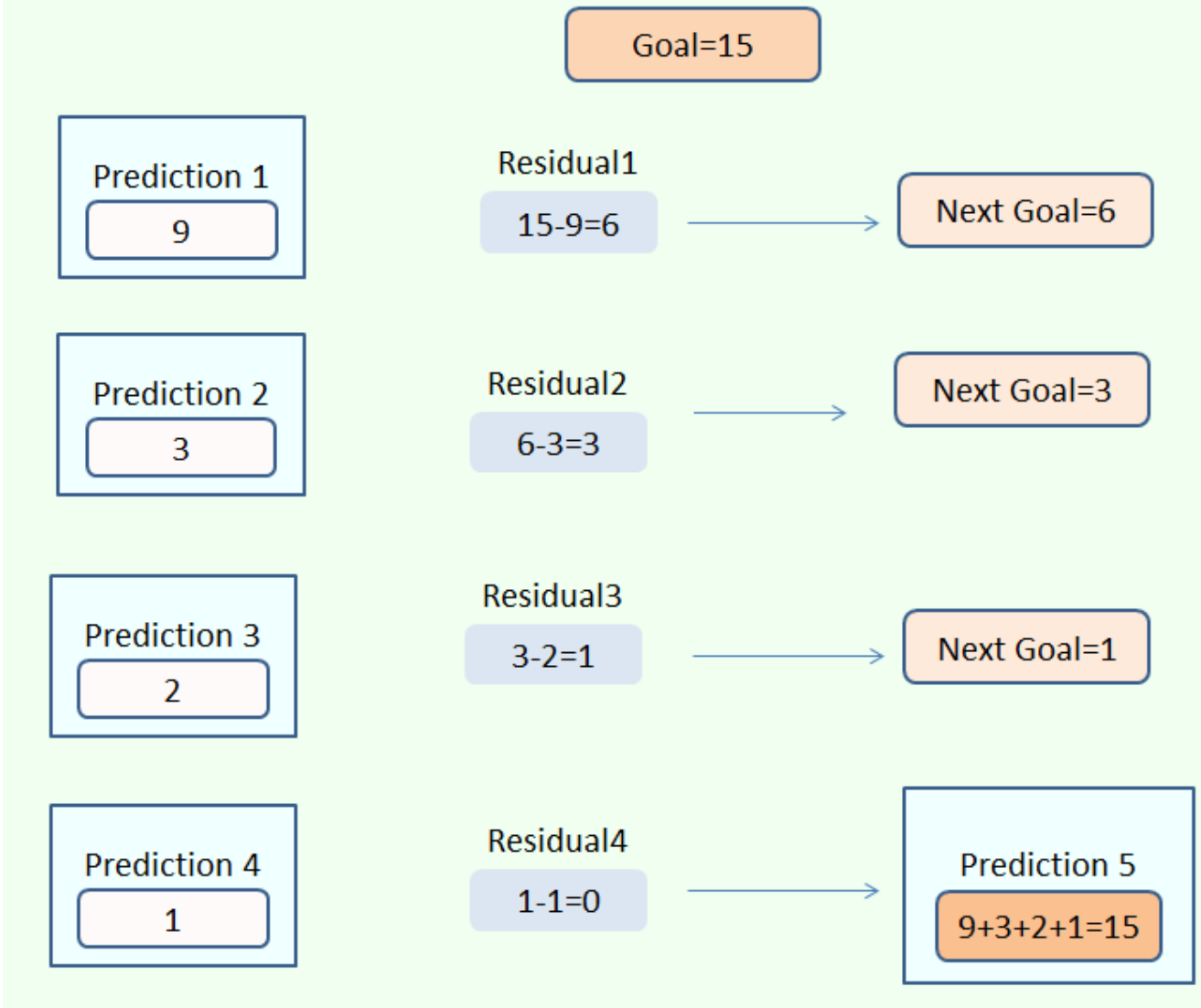
Istnieje wiele wariantów tej metody:

- AdaBoost,
- Gradient Boosting (GBM),
- Extreme Gradient Boosting (XGboost) – szybsza (10 razy) wersja GBM,
- LightGBM – jeszcze szybsza wersja GBM (stworzony przez Microsoft),
- CatBoost – wersja dobrze radząca sobie z cechami jakościowymi (stworzony przez Yandex).

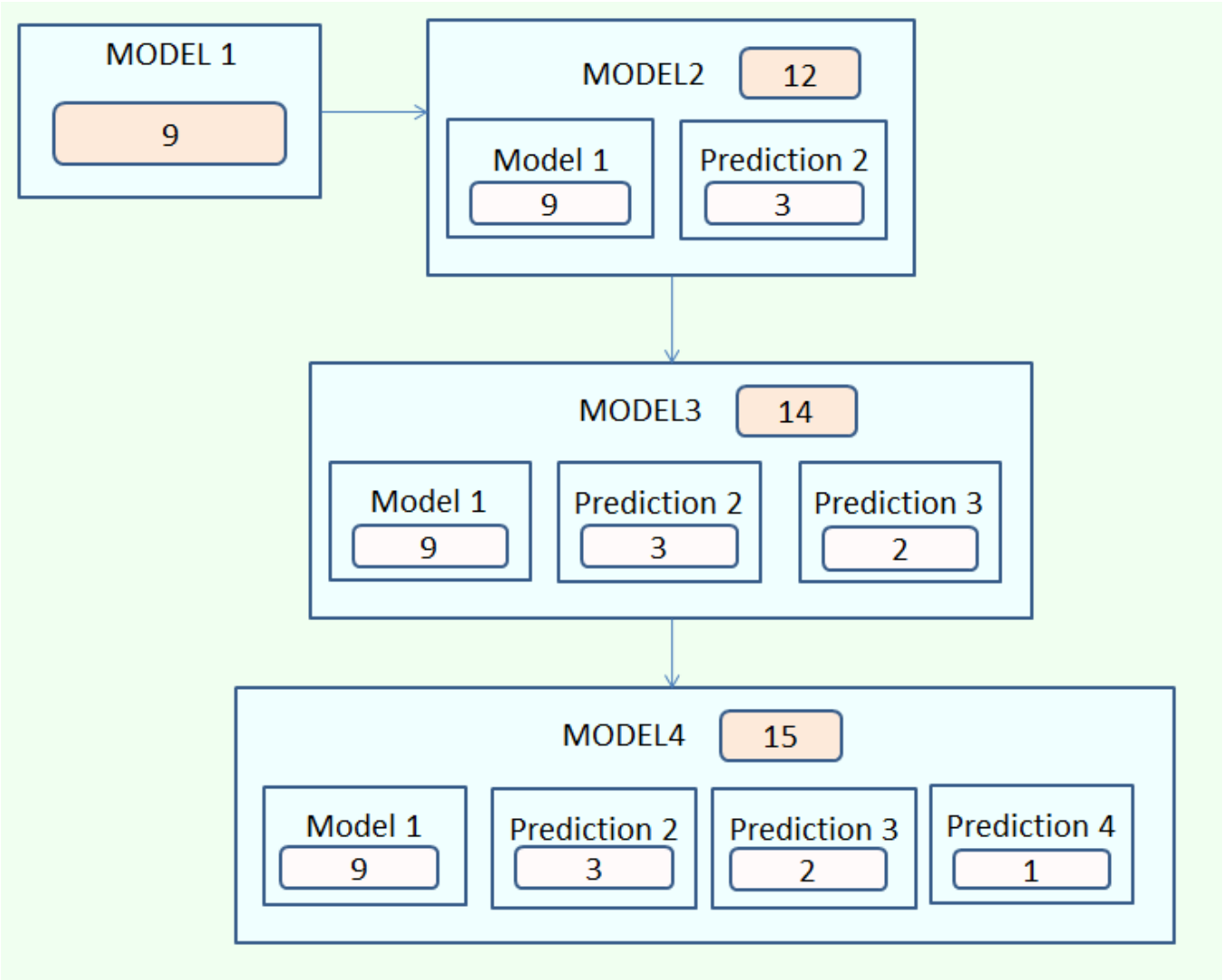
Boosting


Dataset
Decision Tree 1
Predicted y
Residual Error by Decision Tree 1
Repeat
Decision Tree 2
Predicted y
Residual Error by Decision Tree 2

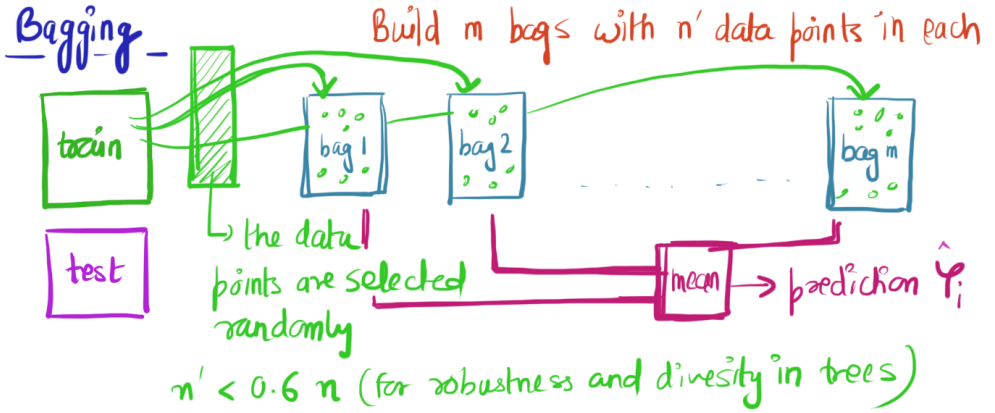
Boosting

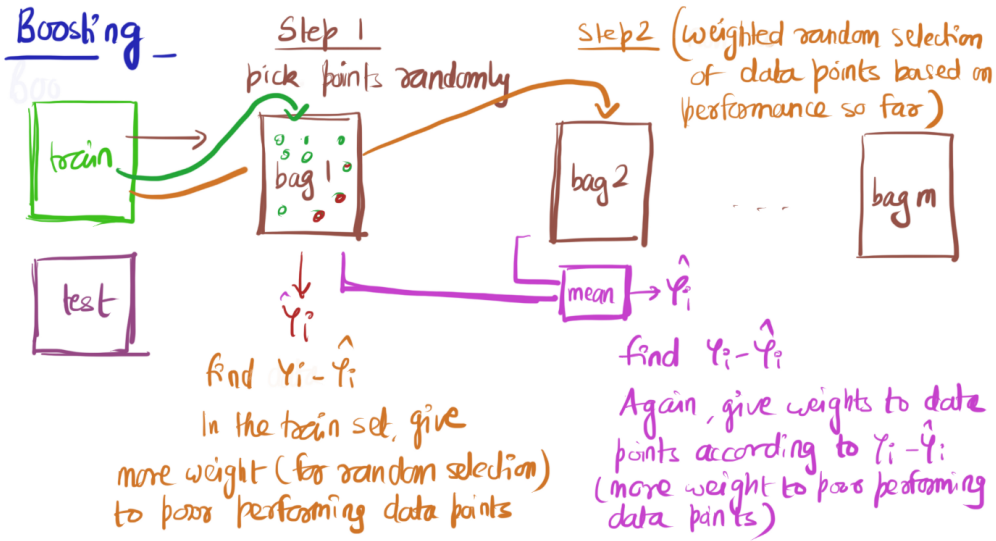

Goal=15
Prediction 1
9
Residual1
15-9=6
Next Goal=6
Prediction 2
3
Residual2
6-3=3
Next Goal=3
Prediction 3
2
Residual3
3-2=1
Next Goal=1
Prediction 4
1
Residual4
1-1=0
Prediction 5
9+3+2+1=15

Boosting

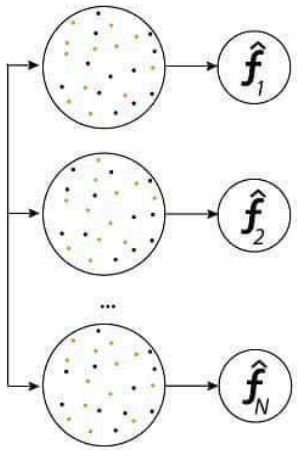
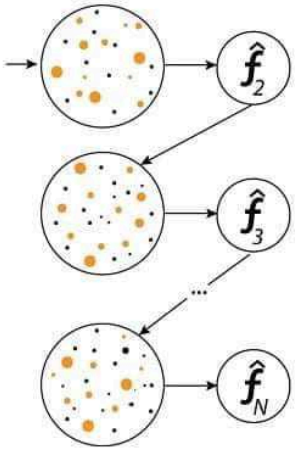

MODEL 1
9
MODEL2
12
Model 1
9
Prediction 2
3
MODEL3
14
Model 1
9
Prediction 2
3
Prediction 3
2
MODEL4
15
Model 1
9
Prediction 2
3
Prediction 3
2
Prediction 4
1

Bagging vs boosting





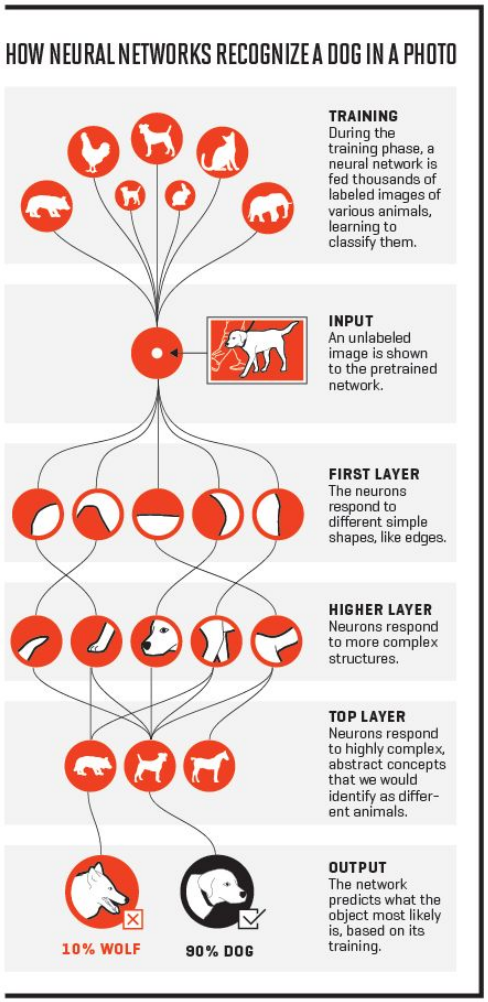
Bagging vs boosting

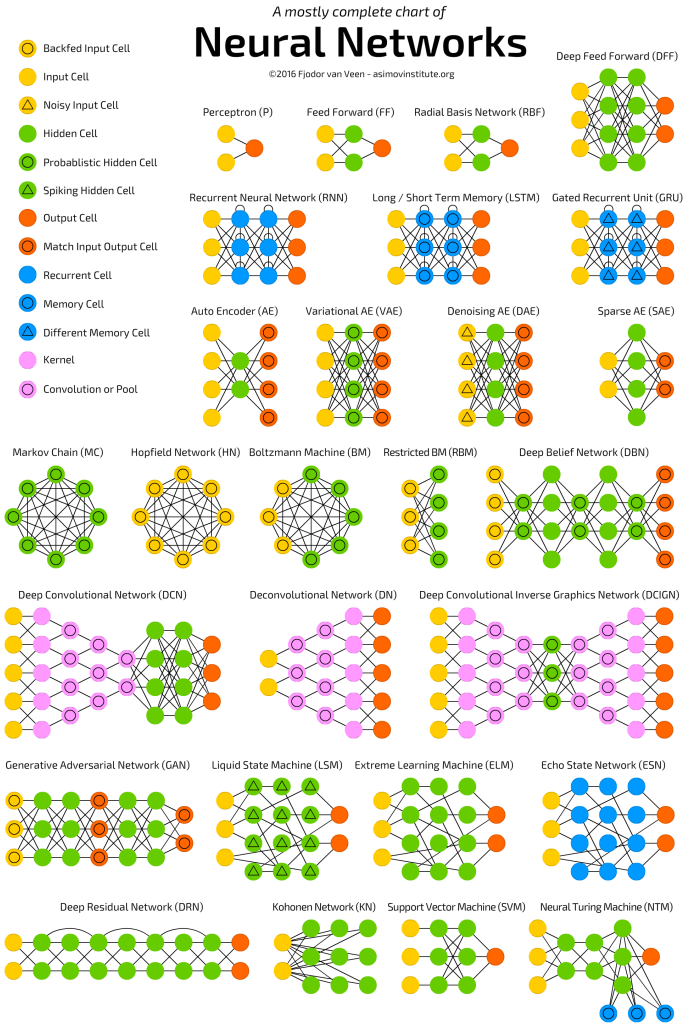
Bagging	Boosting
Bootstrapped samples	Fit enitre dataset
Base trees created independently	Base tree created successively
Only data points considered	Use residuals from previous models
No weighting used	Up-weighting misclassified points
Excess trees will not overfit	Beware of overfitting
Parallel Ensemble Learning	Sequential Ensemble Learning
	

Sieci neuronowe (*ang. neural networks*)

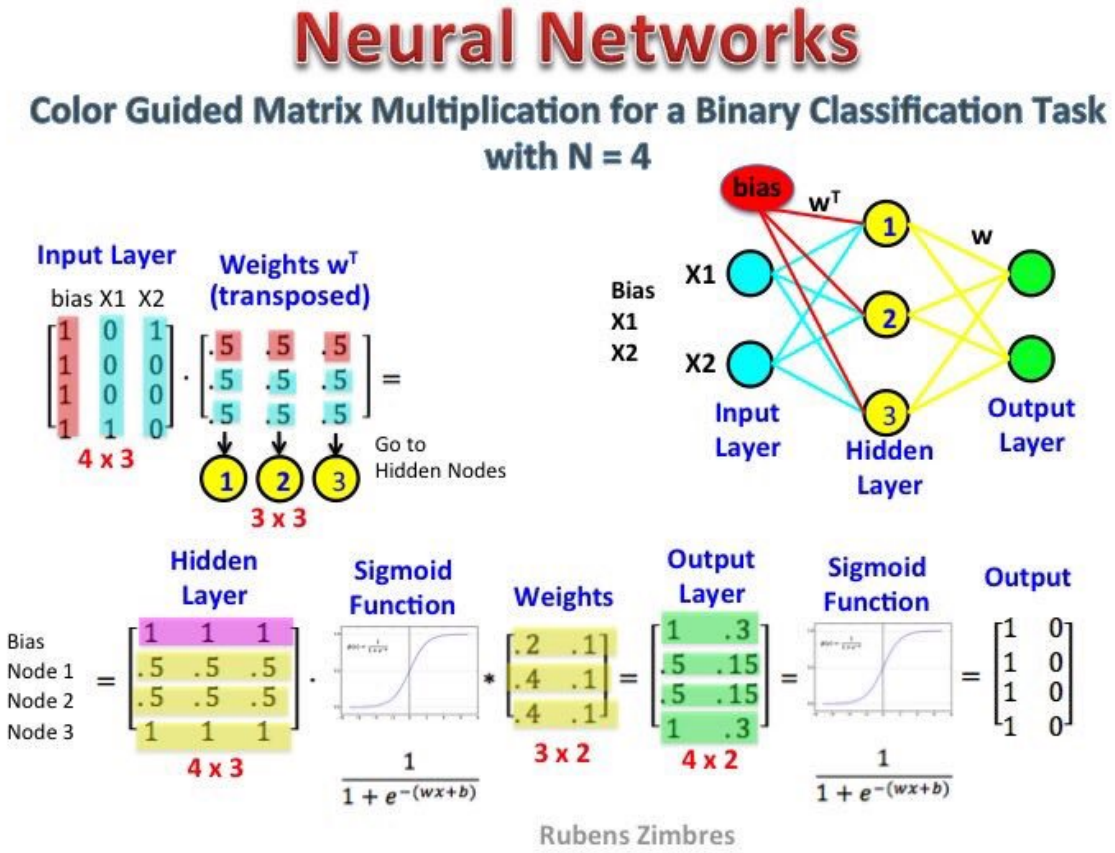
Sieć neuronowa to połączona grupa sztucznych komórek nerwowych, która wykorzystuje model matematyczny w celu przetwarzania informacji. Istotną cechą sieci jest możliwość uczenia się, czyli modyfikowania parametrów charakteryzujących poszczególne neurony w taki sposób, by zwiększyć efektywność sieci. Pomysł tej techniki wywodzi się z badań bioelektrycznych sieci utworzonych w mózgu przez neurony i synapsy. W modelu sieci neuronowej węzły (zwane neuronami) są połączone między sobą tworząc sieć węzłów – stąd określenie „sieć neuronowa". Sztuczne sieci neuronowe składają się najczęściej z trzech warstw: warstwa wejściowa, ukryta i wyjściowa.

Sieci neuronowe (*ang. neural networks*)





Sieci neuronowe (*ang. neural networks*)

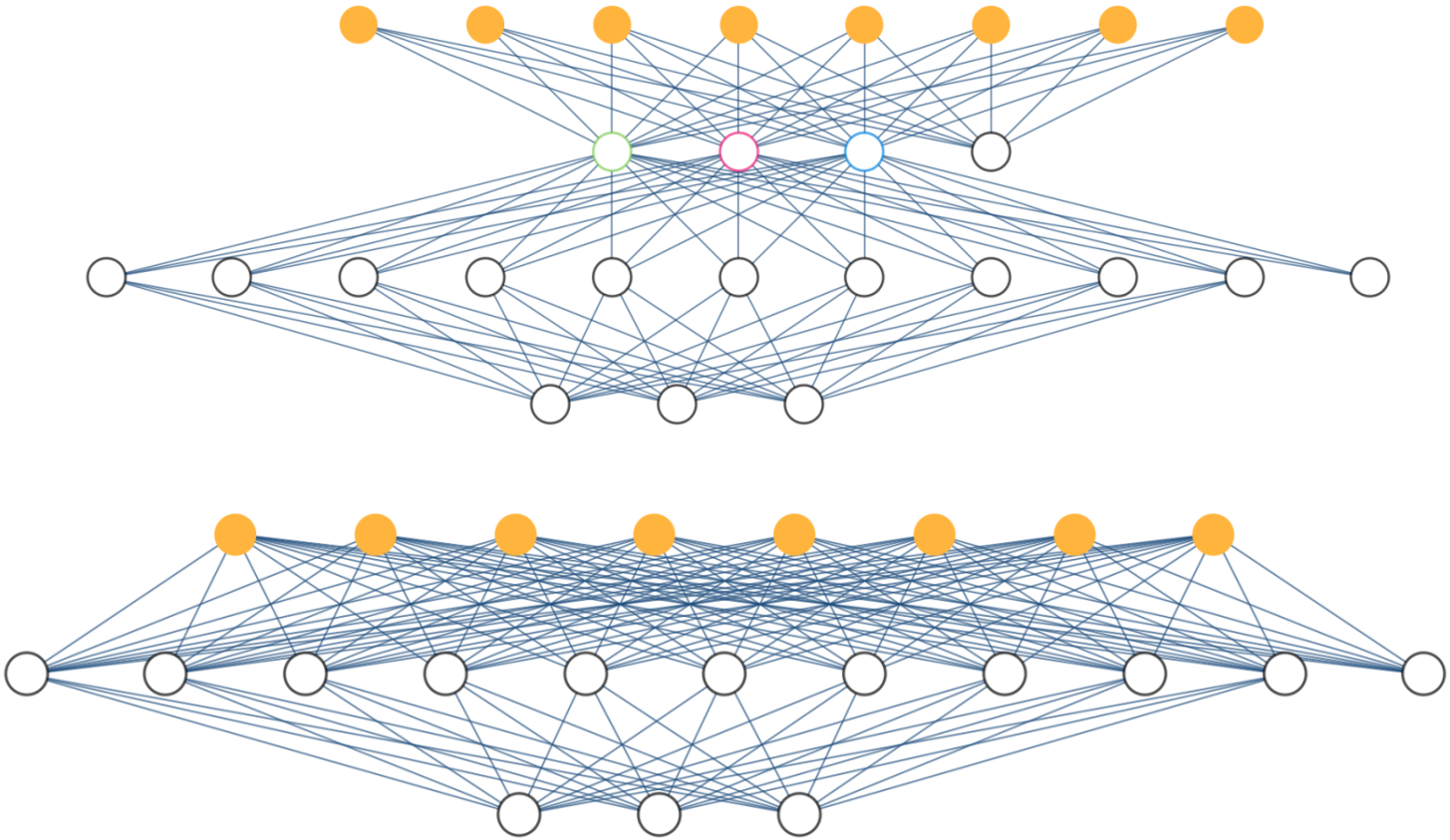


http://playground.tensorflow.org/

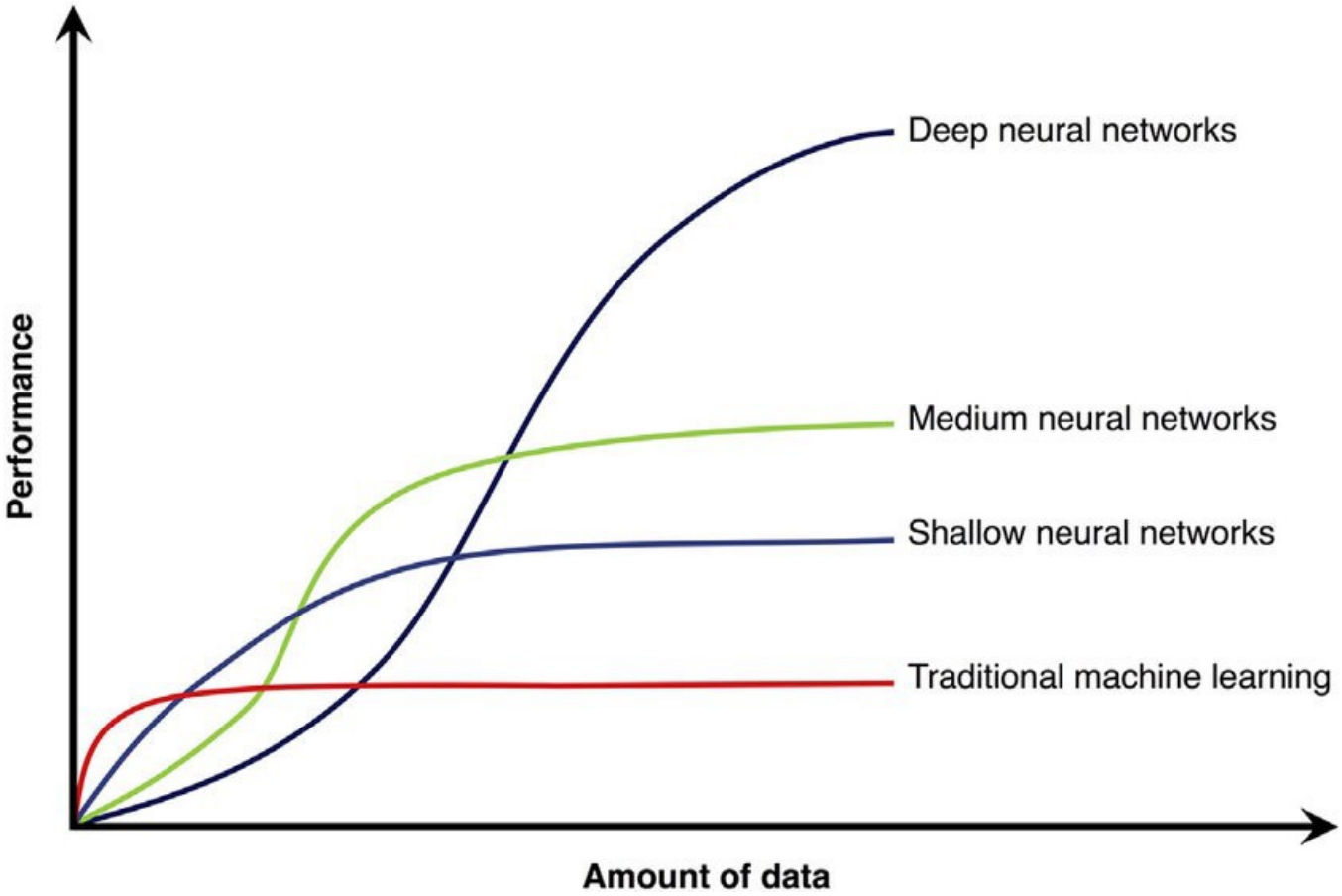
Sieci neuronowe (*ang. neural networks*)

Głębsze architektury pozwalają na rozwiązywanie trudniejszych problemów. Cała ta złożoność może być jednak zwinięta w jedną dużą warstwę ukrytą. Sieć trójwarstwowa ma taką samą moc jak sieć czterowarstwowa, ale nie jest tak samo efektywna: sieć trójwarstwowa o konfiguracji $3 - 10 - 8$ ma 118 parametrów $(3 \times 10 + 10 \times 8 + 8)$, podczas gdy sieć czterowarstwowa o konfiguracji $3 - 10 - 3 - 8$ ma jedynie 95 parametrów $(3 \times 10 + 10 \times 3 + 3 \times 8 + 3 + 8)$. Na tym właśnie polega siła głębokich sieci neuronowych (*ang. deep neural network*) – umożliwiają one badanie złożonych modeli stosując stosunkowo niewiele parametrów.

Sieci neuronowe (*ang. neural networks*)



Sieci neuronowe (*ang. neural networks*)



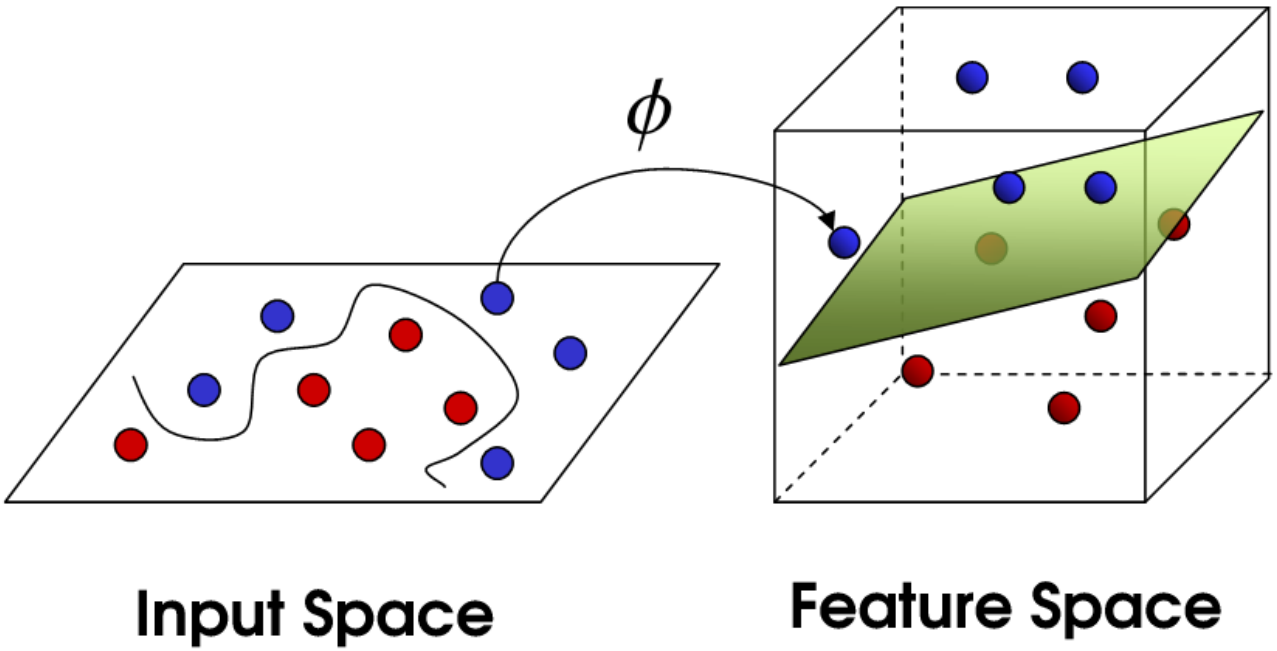
Metoda wektorów nośnych (*ang. support vector machines*)

Metoda wektorów nośnych jest metodą, która polega na transformacji obserwacji (za pomocą tzw, triku jądrowego (*ang. kernel trick*)), które nie są liniowo separowalne (czyli nie da się ich podzielić na klasy hiperpłaszczyznami), do przestrzeni o wyższym wymiarze, w której być może staną się już separowalne liniowo. Dzięku trikowi jądrowego nie ma potrzeby transformacji punktów, wystarczy jedynie mierzyć odległości w wyższym wymiarze.

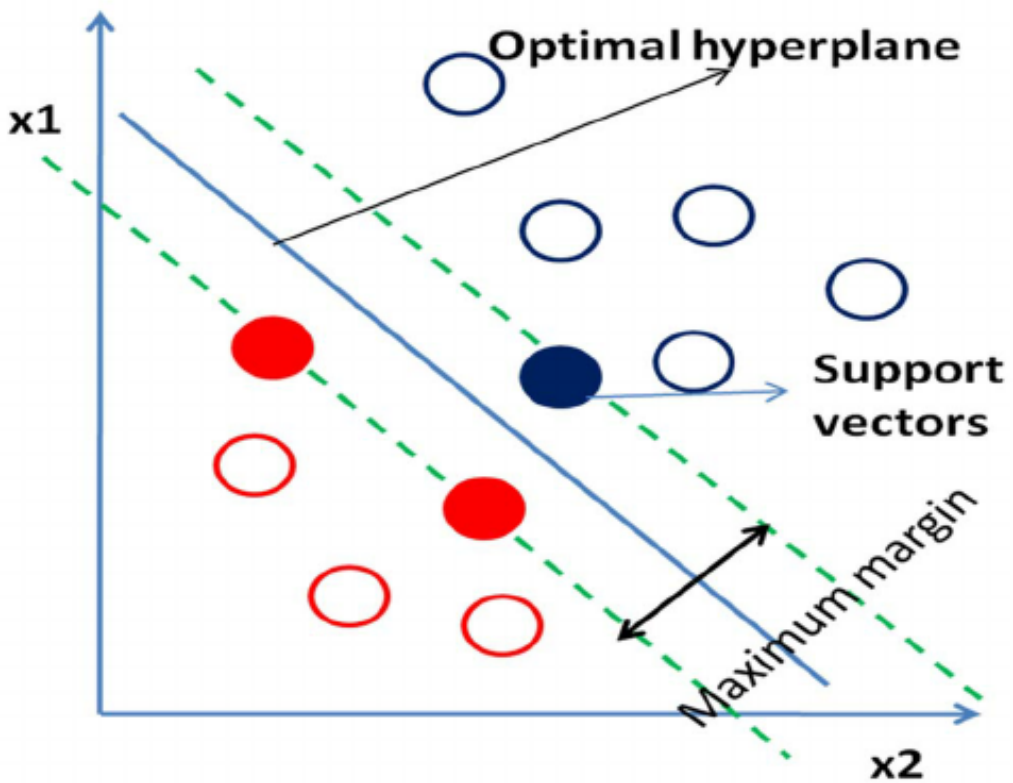
 Aizerman, M.A., Braverman, E.M., Rozonoer, L.I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 25:821–837.

 Cortes, C., Vapnik, V.N. (1995). Support-vector networks. Machine Learning. 20(3):273–297.

Metoda wektorów nośnych (*ang. support vector machines*)



Metoda wektorów nośnych (*ang. support vector machines*)



caret – karta pomocy

caret Package

Cheat Sheet

Specifying the Model

Possible syntaxes for specifying the variables in the model:

```
train(y ~ x1 + x2, data = dat, ...)
train(x = predictor_df, y = outcome_vector, ...)
train(recipe_object, data = dat, ...)
```

- rfe, sbf, gafs, and safs only have the x/y interface.
- The train formula method will **always** create dummy variables.
- The x/y interface to train will not create dummy variables (but the underlying model function might).

Remember to:

- Have column names in your data.
- Use factors for a classification outcome (not 0/1 or integers).
- Have valid R names for class levels (not "0"/"1")
- Set the random number seed prior to calling train repeatedly to get the same resamples across calls.
- Use the train option na.action = na.pass if you will being imputing missing data. Also, use this option when predicting new data containing missing values.

To pass options to the underlying model function, you can pass them to train via *the ellipses*:

```
train(y ~ ., data = dat, method = "rf",
      # options to `randomForest`:
      importance = TRUE)
```

Parallel Processing

The foreach package is used to run models in parallel. The train code does not change but a "do" package must be called first.

```
# on MacOS or Linux
library(doMC)
registerDoMC(cores=4)
```

```
# on Windows
library(doParallel)
cl <- makeCluster(2)
registerDoParallel(cl)
```

The function parallel::detectCores can help too.

Preprocessing

Transformations, filters, and other operations can be applied to the *predictors* with the preProc option.

```
train(, preProc = c("method1", "method2"), ...)
```

Methods include:

- "center", "scale", and "range" to normalize predictors.
- "BoxCox", "YeoJohnson", or "expoTrans" to transform predictors.
- "knnImpute", "bagImpute", or "medianImpute" to impute.
- "corr", "nzv", "zv", and "conditionalX" to filter.
- "pca", "ica", or "spatialSign" to transform groups.

train determines the order of operations; the order that the methods are declared does not matter.

The recipes package has a more extensive list of preprocessing operations.

Adding Options

Many train options can be specified using the trainControl function:

```
train(y ~ ., data = dat, method = "cubist",
      trControl = trainControl(<options>))
```

Resampling Options

trainControl is used to choose a resampling method:

```
trainControl(method = <method>, <options>)
```

Methods and options are:

- "cv" for K-fold cross-validation (number sets the # folds).
- "repeatedcv" for repeated cross-validation (repeats for # repeats).
- "boot" for bootstrap (number sets the iterations).
- "LGOCV" for leave-group-out (number and p are options).
- "LOO" for leave-one-out cross-validation.
- "oob" for out-of-bag resampling (only for some models).
- "timeslice" for time-series data (options are initialWindow, horizon, fixedWindow, and skip).

Performance Metrics

To choose how to summarize a model, the trainControl function is used again.

```
trainControl(summaryFunction = <R function>,
             classProbs = <logical>)
```

Custom R functions can be used but caret includes several: defaultSummary (for accuracy, RMSE, etc), twoClassSummary (for ROC curves), and prSummary (for information retrieval). For the last two functions, the option classProbs must be set to TRUE.

Grid Search

To let train determine the values of the tuning parameter(s), the tuneLength option controls how many values **per tuning** parameter to evaluate.

Alternatively, specific values of the tuning parameters can be declared using the tuneGrid argument:

```
grid <- expand.grid(alpha = c(0.1, 0.5, 0.9),
                    lambda = c(0.001, 0.01))

train(x = x, y = y, method = "glmnet",
      preProc = c("center", "scale"),
      tuneGrid = grid)
```

Random Search

For tuning, train can also generate random tuning parameter combinations over a wide range. tuneLength controls the total number of combinations to evaluate. To use random search:

```
trainControl(search = "random")
```

Subsampling

With a large class imbalance, train can subsample the data to balance the classes them prior to model fitting.

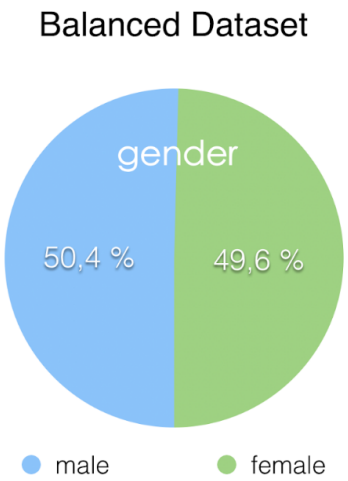
```
trainControl(sampling = "down")
```

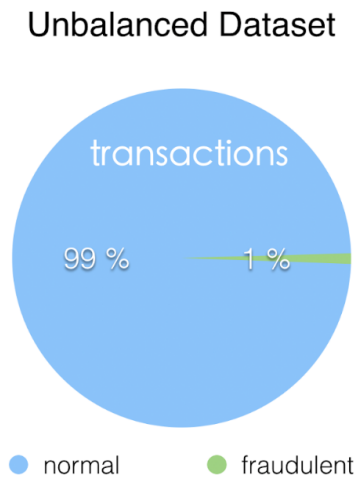
Other values are "up", "smote", or "rose". The latter two may require additional package installs.

Learn more at https://topepo.github.io/caret/ • Updated: 9/17

Definicja

Zbiór danych nazywamy niezbalansowanym (*ang. unbalanced*) jeżeli zawiera dużo więcej obserwacji z jednej klasy (*ang. majority class*) niż z pozostałych lub jedna z klas (*ang. minority class*) zawiera dużo mniej obserwacji niż pozostałe. W takiej sytuacji dokładność klasyfikacji może być bardzo dobra na klasach licznych i bardzo słaba na klasach mniejszościowych.

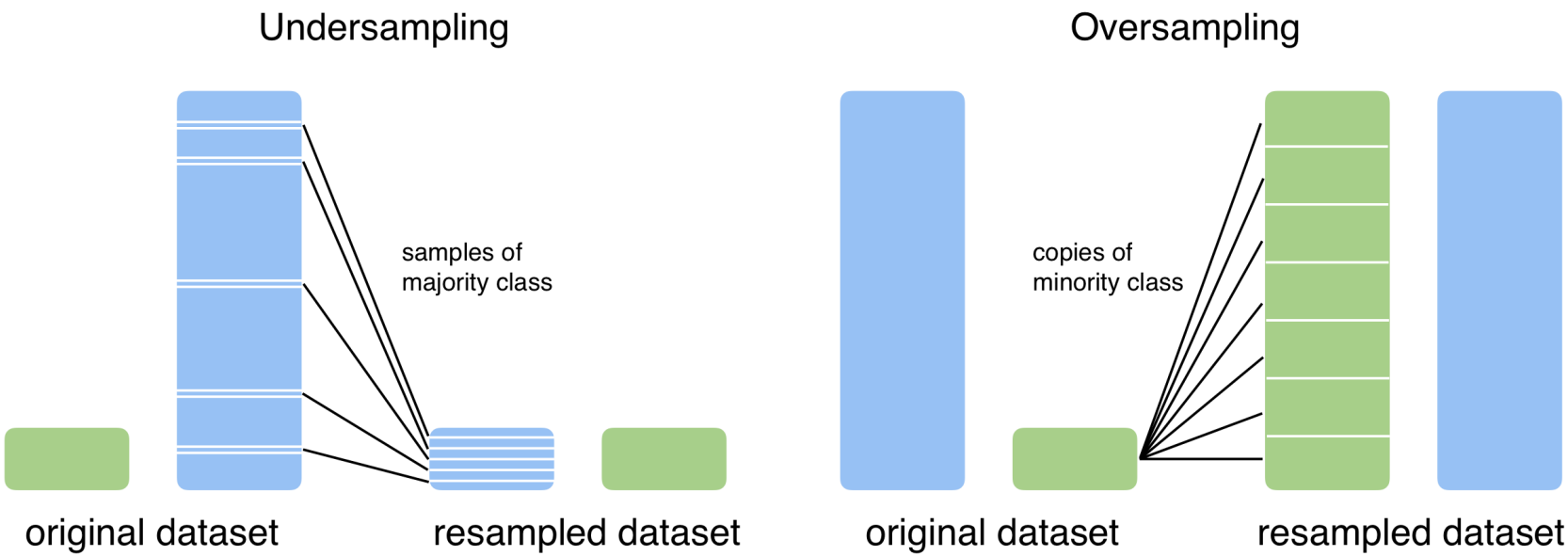




Metody rozwiązania problemu niezbalansowania

- Undersampling majority class – losowe (lub wybieramy pewne obserwacje, które reprezentują klasy – prototypy) zmniejszanie liczebności dużych klas.
- Oversampling minority class – zwiększanie liczebności małych klas (dodajemy losowo kopie klas mniejszościowych). Najpopularniejsza jest metoda ROSE (*ang. Random Over-Sampling Examples*)
- Dodanie sztucznych danych – podobne do oversamplingu, ale nowe dane nie są kopiami starych. Najpopularniejszą metodą jest SMOTE (*ang. Synthetic Minority Oversampling Technique*), jej wariant ADASYN (*ang. Adaptive Synthetic Sampling Method*).

Metody rozwiązania problemu niezbalansowania



Metody rozwiązania problemu niezbalansowania

Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). *SMOTE: synthetic minority over-sampling technique*. Journal of Artificial Intelligence Research 16(1):321–357.

He, H., Bai, Y., Garcia, E.A., Li, S. (2008). ADASYN: adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 1322–1328.

Menardi, G. & Torelli, N. (2014). *Training and assessing classification rules with imbalanced data*. Data Mining and Knowledge Discovery 28:92–122.

Algorytm SMOTE

1. Znajdź dla każdej obserwacji (dla każdej cechy z osobna) z klasy mniejszościowej k-najbliższych sąsiadów.
2. Wyznacz proste z wybranego punktu do wszystkich k sąsiadów.
3. Wygeneruj losowo punkty na tej linii.

ADASYN zamiast brać punkty na prostej dodaje do nich jeszcze nieznaczny szum.

Algorytm SMOTE