Klasyfikacja tekstów

Klasyfikacja tekstów

- Aby umożliwić wyszukiwanie dokumentów na temat należy:
  - zgrupować dokumenty na podstawie tematów,
  - nadać tym grupom wyróżniające nazwy (etykiety),
- Klasyfikacja (kategoryzacja) tekstów:

  Proces porządkowania informacji poprzez kojarzenie dokumentów tekstowych z klasami (kategoriami)
- Uczenie maszynowe:
  - Algorytmy, które uczą się wzorców danych,
  - Po wyuczeniu wzorców można przewidywać kategorie nowych danych,
  - Algorytmy uczące używają danych testowych przy uczeniu nadzorowanym i częściowo nadzorowanym.

Klasyfikacja tekstów

- Klasyfikator definiuje się następująco:
  - *D*: zbiór dokumentów,
  - $C = \{c_1, c_2, \ldots, c_L\}$: zbiór *L* klas z odpowiednimi etykietami,
  - Klasyfikator tekstu jest funkcją binarną $F: D \times C \rightarrow \{0,1\}$ przypisującą każdej parze $[d_j, c_p]$, $d_j \in D$ i $c_p \in C$ wartość
    - 1, gdy d_j należy do klasy c_p,
    - 0, gdy gdy d_j *nie* należy do klasy c_p.

- Definicja ta objemuje algorytmy nadzorowane i nienadzorowane

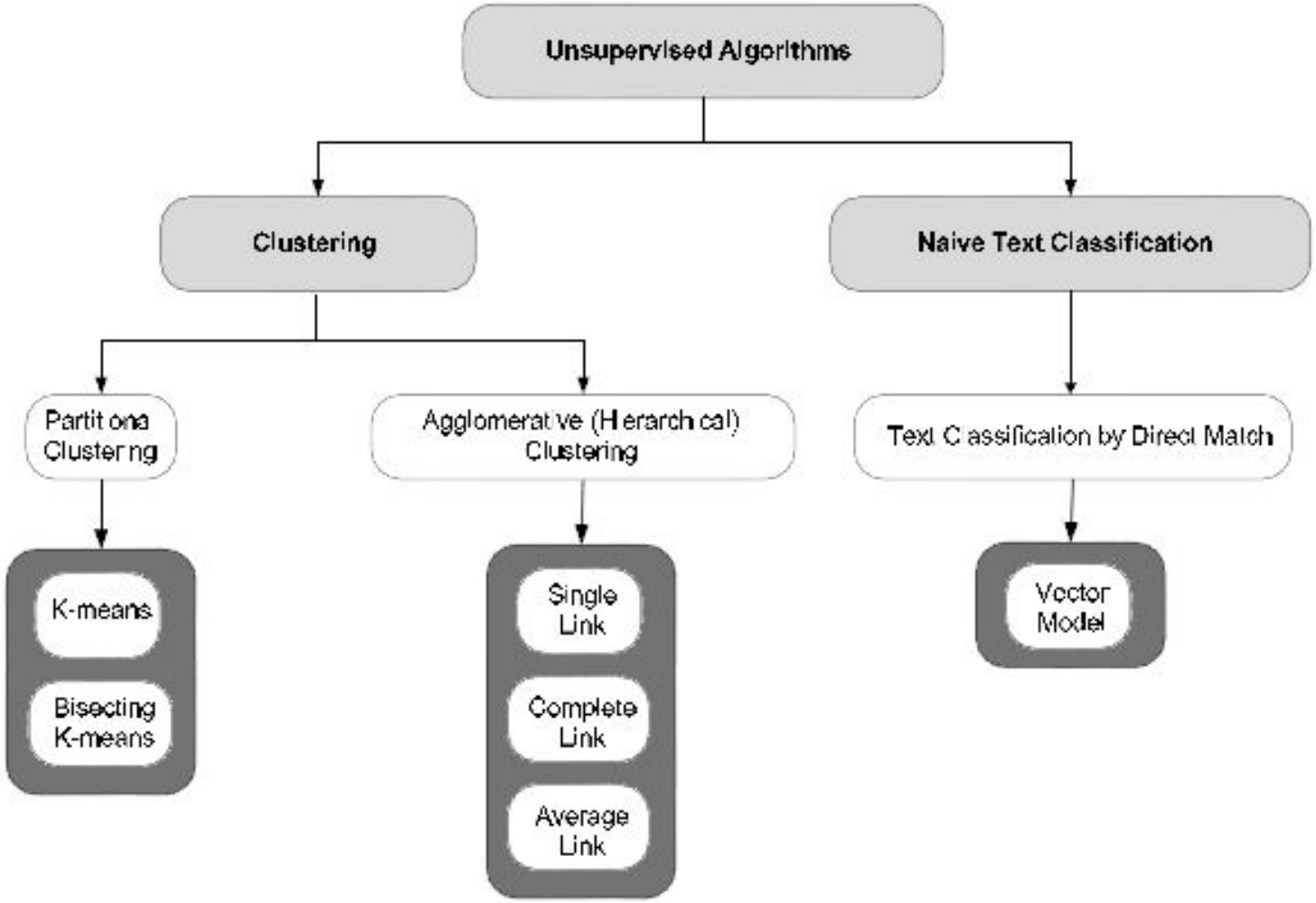
Klasyfikacja tekstów

- Dokładniejsze są algorytmy nadzorowane:
  - jednoetykietowe – pojedyncza klasa przypisana do dokumentu,
  - wieloetykietowe – jedna lub więcej klas przypisanych do dokumentu

- W ostatnim przypadku funkcja klasyfikacji $F(d_j, c_p)$ przestaje być binarna i zwraca stopień przynależności dokumentu d_j do klasy c_p.

Klasyfikacja tekstów

- Algorytmy nienadzorowane



Klasyfikacja tekstów

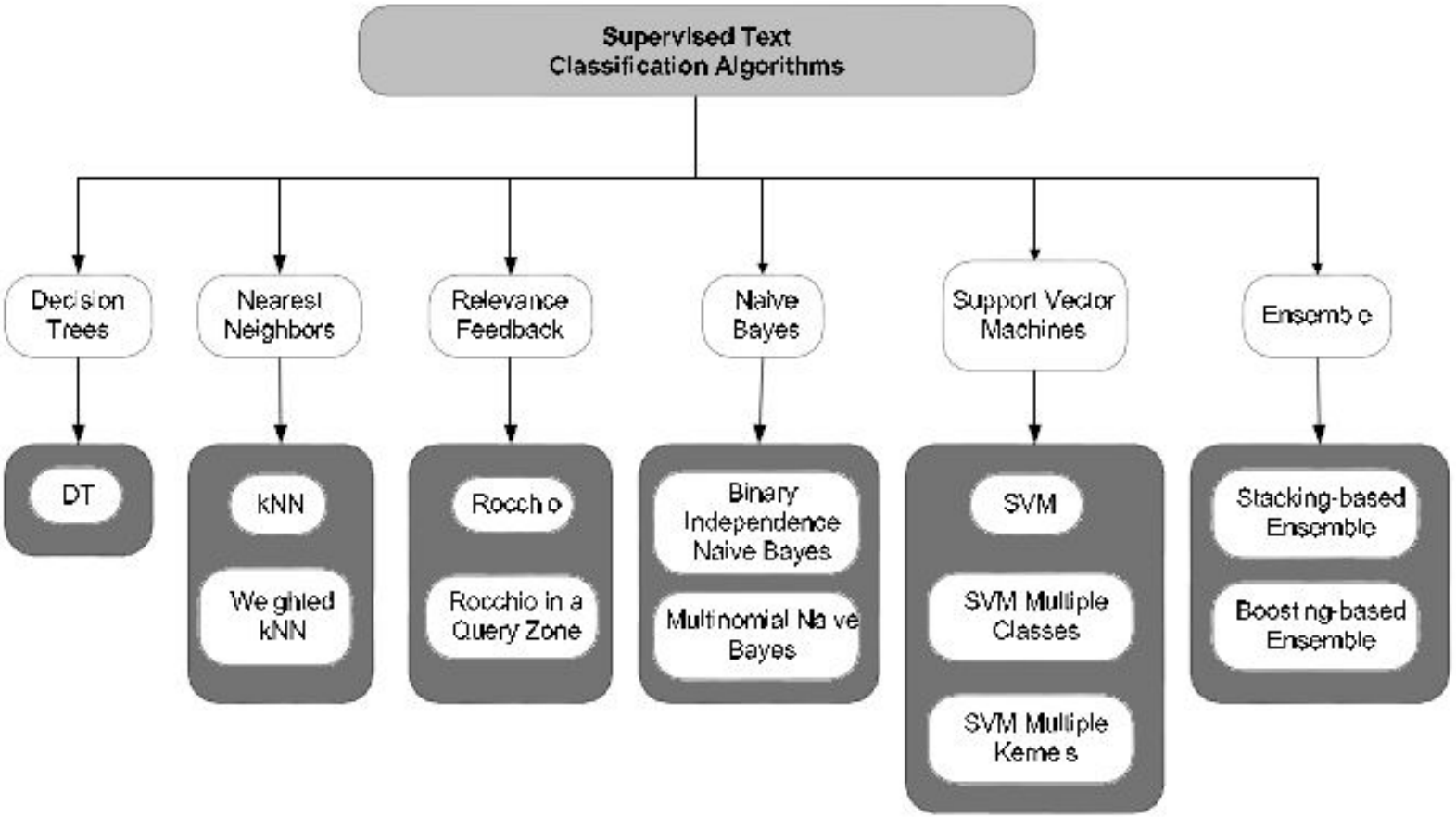
- Algorytmy nadzorowane bazują na zbiorze testowym (uczącym)
  - Zbiór klas z przykładami dokumentów w każdej z nich
  - Przynależność do klas określają specjaliści
  - Zbiór testowy służy do uczenia klasyfikatora

- Duża liczność zbioru uczącego lepiej dostraja klasyfikator i zapobiega przeuczeniu (*overfitting*)

- Klasyfikator podlega ocenie jakości (walidacja krzyżowa)

Klasyfikacja tekstów

- Nadzorowane algorytmy klasyfikacji



Klasyfikacja tekstów

Algorytmy nienadzorowane (klastering)

Klasyfikacja K-średnich

- W metodach nienadzorowanych etykiety klas są generowane automatycznie poprzez ustalenie środków skupień danych w odpowiednio dobranej przestrzeni atrybutów (termów) opisujących dokumenty

- Wyniki mogą być czasami niezadowalające lub różne od oczekiwanych przez użytkownika

- Metoda K-średnich:
  - Wejście – zadana liczba K klastrów,
  - Każdy klaster jest reprezentowany przez centroidę dokumentów,
  - Algorytm:
    - Przypisanie dokumentu do najbliższej centroidy,
    - Przeliczenie centroid,
    - Powtórzenie poprzednich kroków aż do stabliizacji położenia centroid

K-średnich – tryb wsadowy

- Wszystkie dokumenty są klasyfikowane przed przeliczeniem centroid
- Dokument d_j reprezentuje wektor $\vec{d}_j$

$$\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

- gdzie
  - $w_{i,j}$: waga termu k_i w dokumencie d_j,
  - t: rozmiar słownika.
- Krok początkowy:
  - Wybierz losowo K dokumentów jako centroidy

$$\vec{\triangle}_p = \vec{d}_j$$

K-średnich – tryb wsadowy

- Krok przypisania:
  - przypisz każdy dokument do najbliższej centroidy
  - oblicz funkcję odległości jako odwrotność funkcji podobieństwa d_j i c_p (formuła kosinusowa)

$$sim(d_j, c_p) = \frac{\vec{\triangle}_p \bullet \vec{d}_j}{|\vec{\triangle}_p| \times |\vec{d}_j|}$$

- Krok poprawy – przeliczenie centroid każdego klastra

$$\vec{\triangle}_p = \frac{1}{size(c_p)} \sum_{\vec{d}_j \in c_p} \vec{d}_j$$

- Powtarzaj kroki przypisania i poprawy, aż centroidy przestaną się zmieniać

Bisekcja K-średnich

- Algorytm:
  - buduje hierarchię klastrów,
  - w każdym kroku dzieli na 2 klastry.
- Wielokrotne powtórzenie K-średnich dla K=2
- Krok początkowy – przypisanie dokumentów do jednego klastra,
- Krok podziału:
  - wybierz najliczniejszy klaster,
  - zastosuj do niego metodę K-średnich (K=2)
- Krok wyboru (decyzji):
  - zatrzymaj, jeśli wszystkie klastry mniejsze niż zadany rozmiar,
  - w przeciwnym razie wróć do kroku podziału

Klastering hierarchiczny

- Algorytm podstawowy:
- Początek:
  - start: zbiór *N* dokumentów do klasyfikacji
  - macierz podobieństwa (odległości) $N \times N$
- Przypisz każdy dokument do swojego klastra
  - Utwórz *N* klastrów, po jednym dla każdego dokumentu
- Znajdź dwa najbliższe klastry
  - połącz je w jeden klaster,
  - zredukuj liczbę klastrów do *N*-1
- Przelicz odległości między nowym klastrem i każdym ze starych
- Powtórz 2 ostatnie kroki, aż powstanie jeden klaster o rozmiarze *N*.

Klastering hierarchiczny

- Sposób obliczania odległości klastrów definiuje 3 warianty algorytmu:
  - *single-link*
  - *complete-link*
  - *average-link*
- *dist*(c_p, c_r): odległość klastrów c_p and c_r
- *dist*(d_j , d_l): odległość dokumentów d_j and d_l

- **Algorytm *Single-Link***

$$dist(c_p, c_r) = \min_{\forall\ d_j \in c_p, d_l \in c_r} dist(d_j, d_l)$$

Klastering hierarchiczny

- Algorytm *Complete-Link*

$$dist(c_p, c_r) = \max_{\forall\ d_j \in c_p, d_l \in c_r} dist(d_j, d_l)$$

- Algorytm *Average-Link*

$$dist(c_p, c_r) = \frac{1}{n_p + n_r} \sum_{d_j \in c_p} \sum_{d_l \in c_r} dist(d_j, d_l)$$

Prosta klasyfikacja tekstu (naive)

- Wejście:
  - *D*: zbiór dokumentów,
  - $C = \{c_1, c_2, \ldots, c_L\}$: zbiór *L* klas z odpowiednimi etykietami

- Algorytm: przypisz jedną lub więcej klas *C* do każdego dokumentu *D*.
  - dopasuj termy dokumentów do etykiet klas,
  - dopuść częściowe dopasowanie do klas
  - popraw pokrycie przez zdefiniowanie alternatywnych etykiet klas – synonimów

Prosta klasyfikacja tekstu (naive)

- **Klasyfikacja przez dopasowanie bezpośrednie**
- Wejście:
  - *D*: zbiór dokumentów,
  - $C = \{c_1, c_2, \ldots, c_L\}$: zbiór *L* klas z odpowiednimi etykietami
- Reprezentacja:
  - dokument d_j jako ważony wektor $\vec{d}_j$
  - klasa c_p jako ważony wektor $\vec{c}_p$
- Dla każdego dokumentu $d_j \in D$
  - wyszukaj klasy $c_p \in C$ których etykiety zawierają termy d_j,
  - dla każdej wyszukanej pary $[d_j, c_p]$ oblicz wektor rankingu

$$sim(d_j, c_p) = \frac{\vec{d}_j \bullet \vec{c}_p}{|\vec{d}_j| \times |\vec{c}_p|}$$

Prosta klasyfikacja tekstu (naive)

- Wybierz dla d_j klasy c_p z najwyższymi wartościami $sim(d_j, c_p)$

Algorytmy nadzorowane

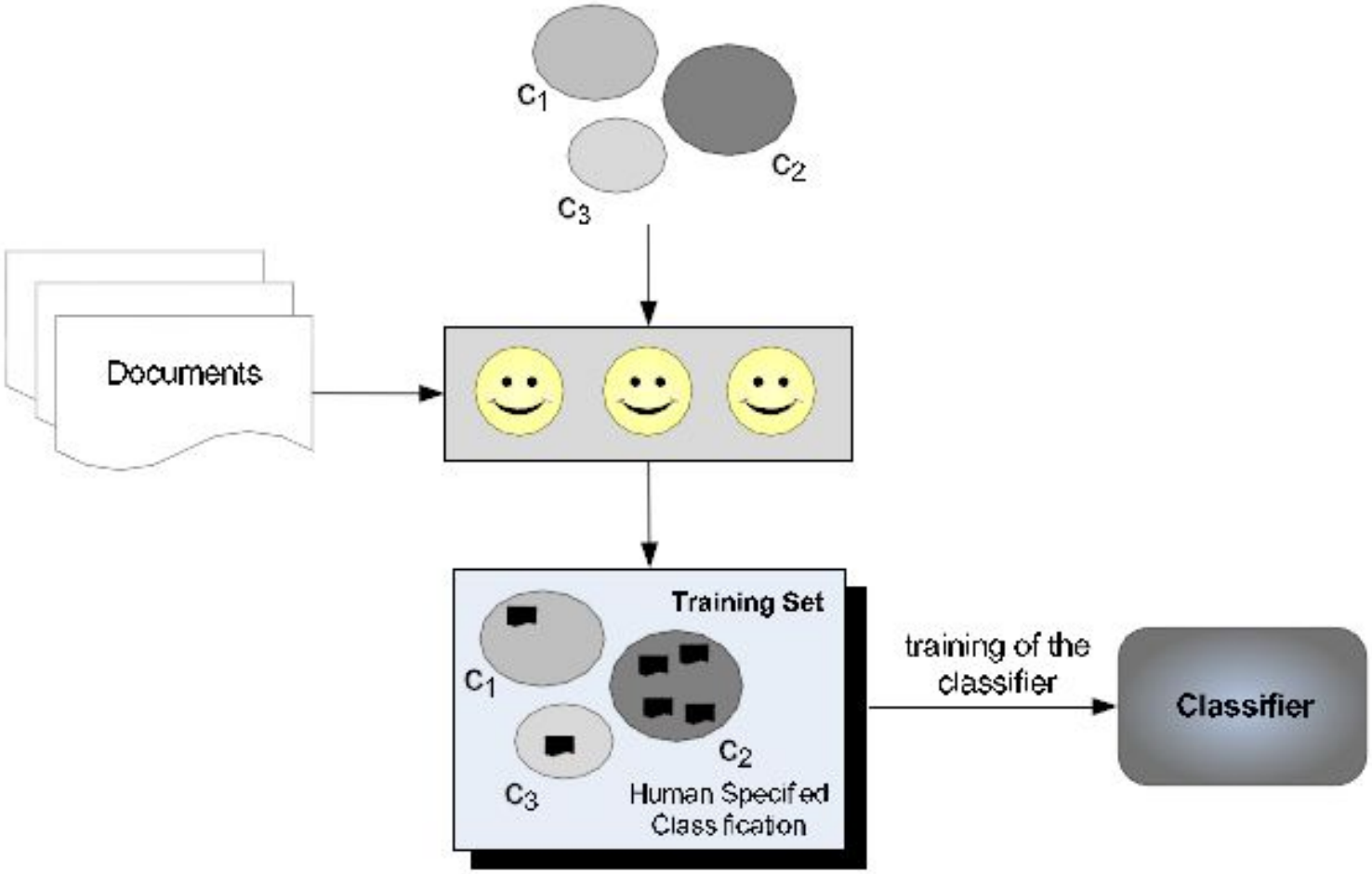
Algorytmy nadzorowane

- Wymagają zbiorów uczących
- $D_t \in D$: zbiór dokumentów uczących
- T: $D_t \times C \rightarrow \{0,1\}$: funkcja trenująca

  Przypisuje do każdej pary $[d_j, c_p]$, $d_j \in D_t$ oraz $c_p \in C$ wartość:
  - 1, gdy $d_j \in c_p$ wg. oceny specjalistów,
  - 0, gdy $d_j \notin c_p$ wg. oceny specjalistów
- Funkcja ucząca *T* jest stosowana do dostrojenia klasyfikatora

Algorytmy nadzorowane

- Faza treningu (uczenia się)



Algorytmy nadzorowane

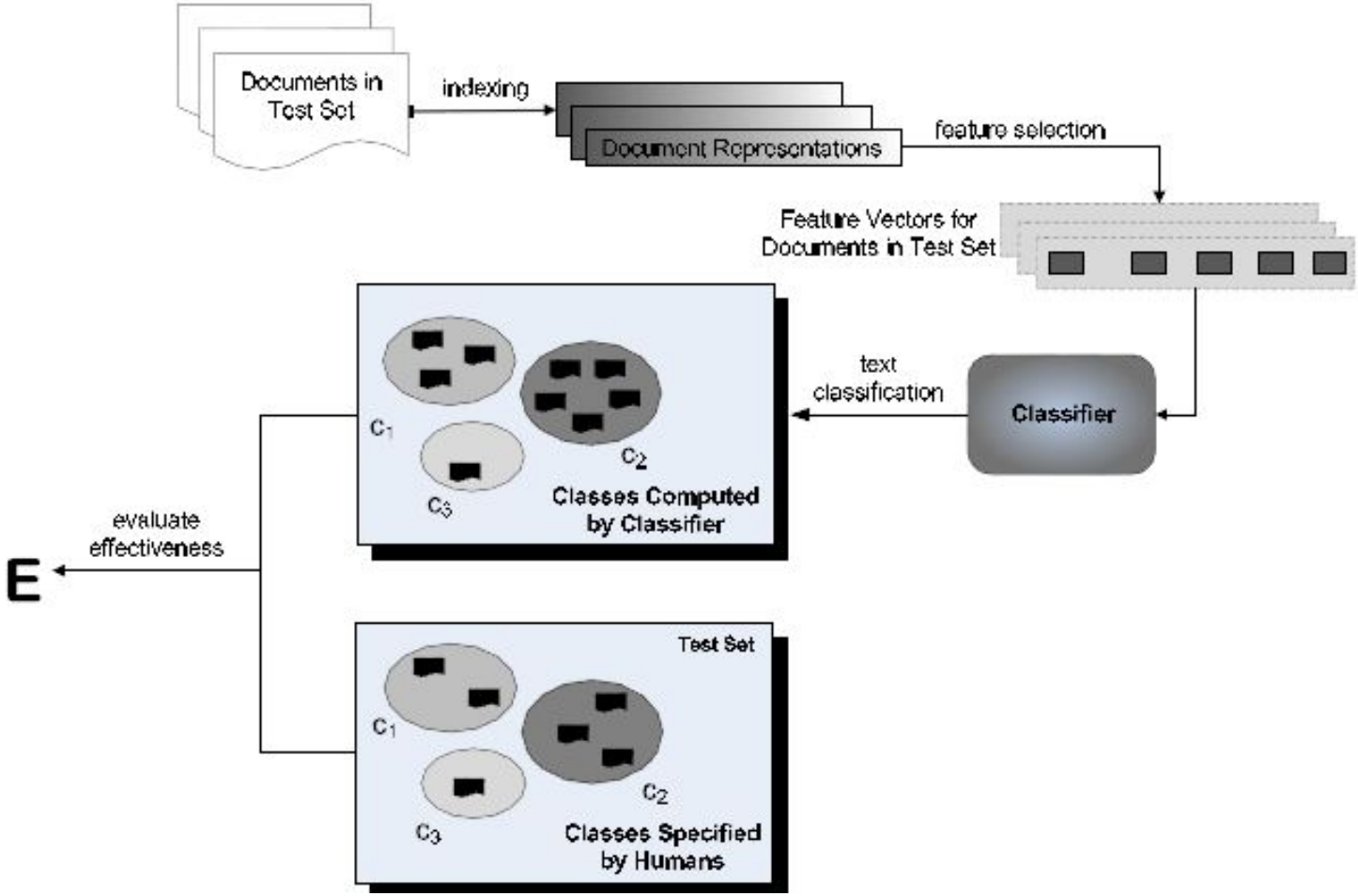
- Aby ocenić klasyfikator używa się zbioru testowego – podzbioru dokumentów nie występujących w zbiorze uczącym

- Klasy dokumentów określają specjaliści

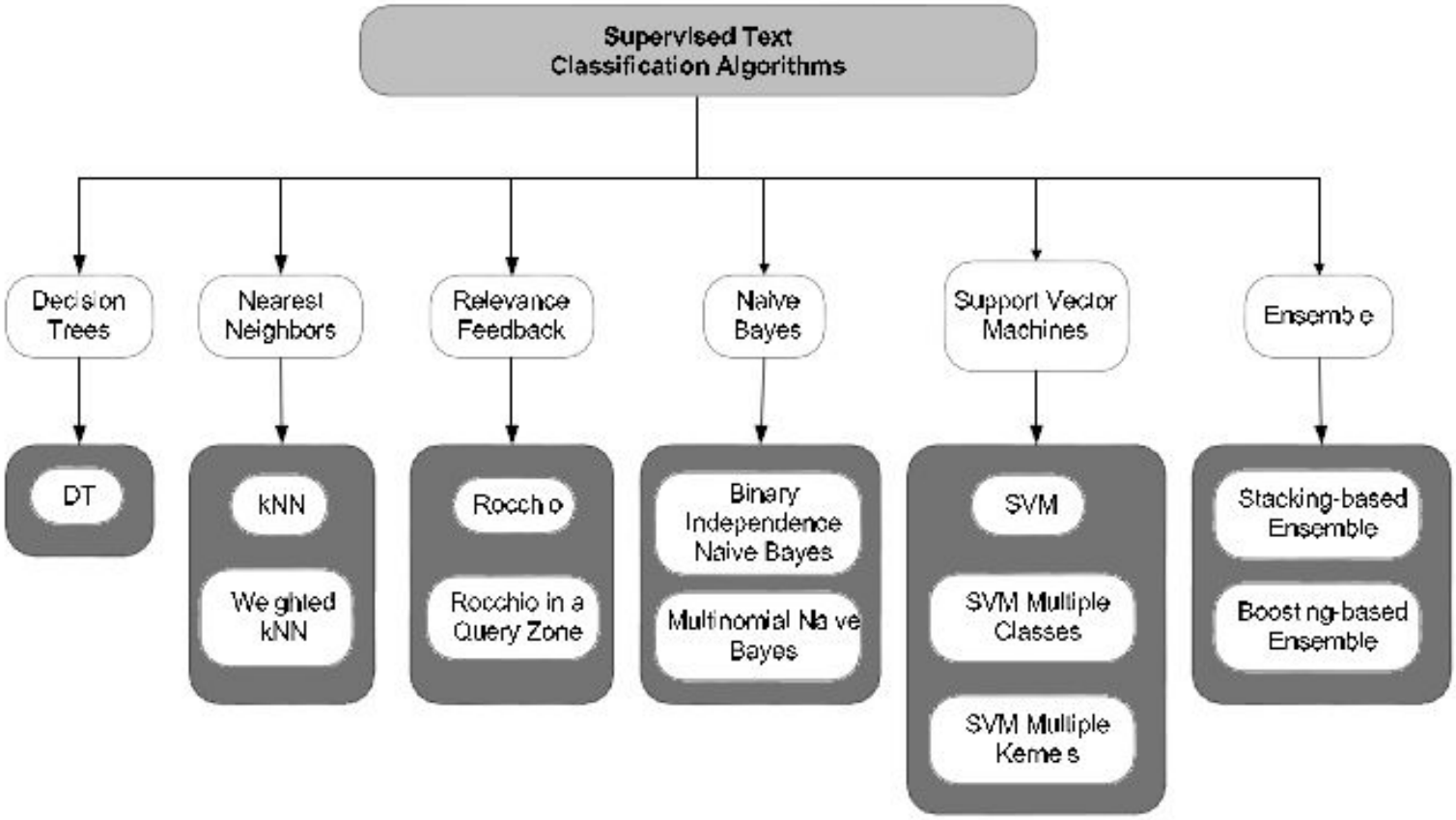
- Weryfikacja algorytmu polega na:
  - klasyfikacji zbioru testowego przy pomocy algorytmu,
  - porównaniu uzyskanych klas z podanymi przez specjalistów.

Algorytmy nadzorowane

- Klasyfikacja i ocena



Algorytmy nadzorowane



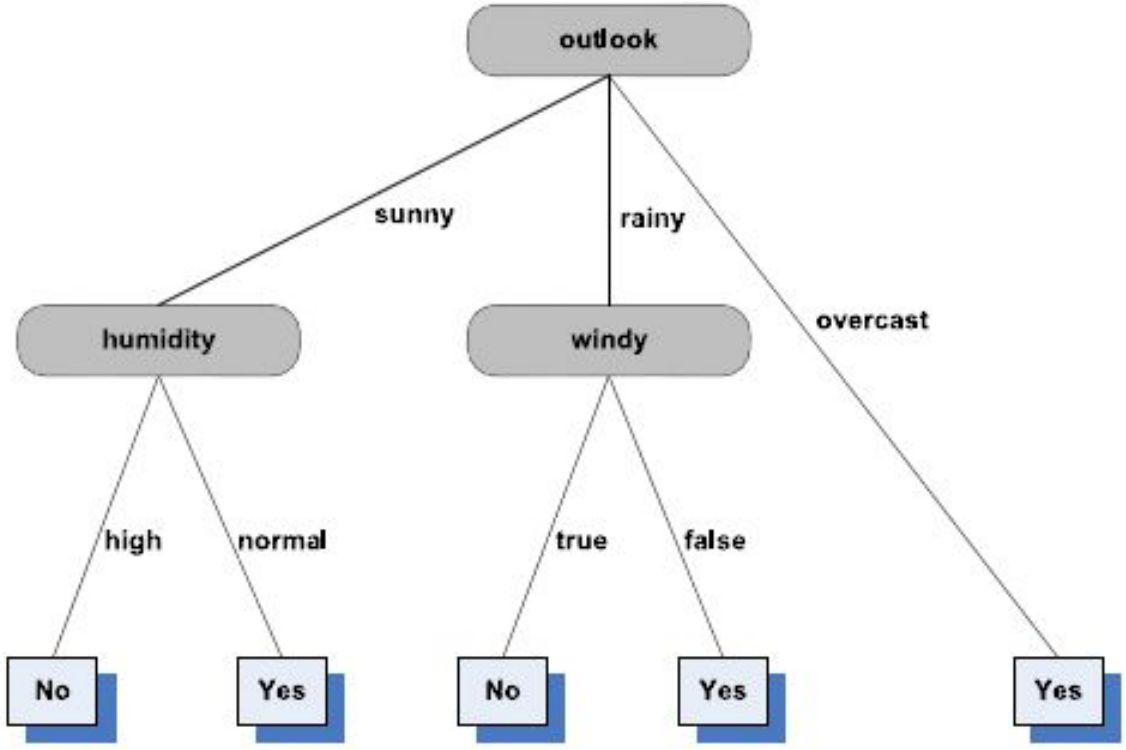
Drzewa decyzyjne

- Zbiór uczący jest używany do budowy reguł klasyfikacji
- Organizacja klasyfikatora w formie drzewa grafu
- Reguły klasyfikacji łatwe do interpretacji przez człowieka
- Przykładowa baza danych:

	Id	Play	Outlook	Temperature	Humidity	Windy
Training set	1	yes	rainy	cool	normal	false
	2	no	rainy	cool	normal	true
	3	yes	overcast	hot	high	false
	4	no	sunny	mild	high	false
	5	yes	rainy	cool	normal	false
	6	yes	sunny	cool	normal	false
	7	yes	rainy	cool	normal	false
	8	yes	sunny	hot	normal	false
	9	yes	overcast	mild	high	true
	10	no	sunny	mild	high	true
Test Instance	11	?	sunny	cool	high	false

Drzewa decyzyjne

- Przewidywanie atrybutu Play



Drzewa decyzyjne

- Węzły wewnętrzne → nazwy atrybutów
- Krawędzie → wartości atrybutów
- Trawers drzewa decyzyjnego → wartość atrybutu “Play”.
- (Outlook = sunny) $\wedge$ (Humidity = high) → (Play = no)

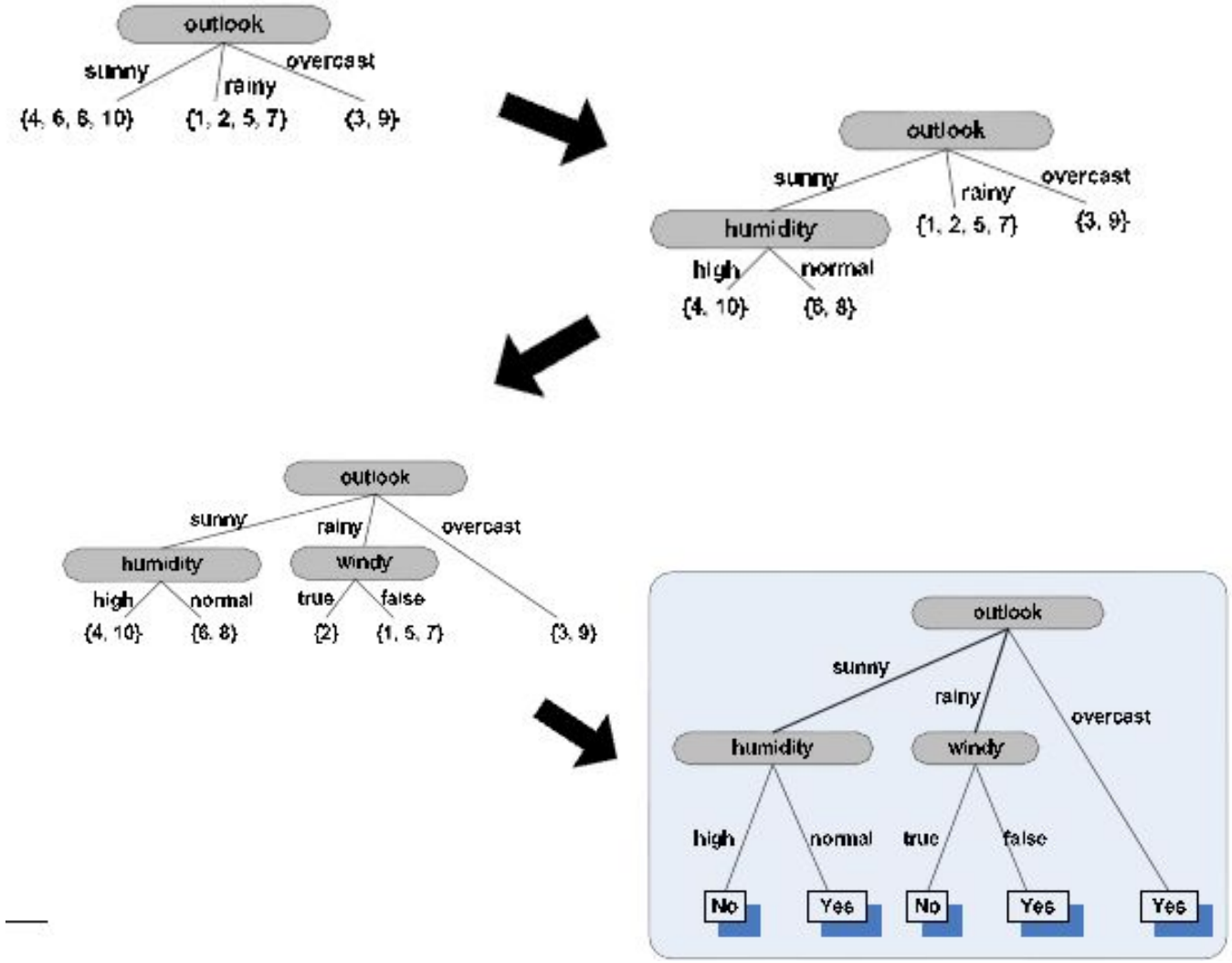
	Id	Play	Outlook	Temperature	Humidity	Windy
Test Instance	11	?	sunny	cool	high	false

- Predykcje bazują na zadanych przykładach
- Nowe przykłady naruszające wzorzec prowadzą do błędnych przewidywań
- Przykładowa baza danych stanowi zbiór uczący, determinujący drzewo decyzyjne DT (Decision Tree)

Drzewa decyzyjne

- DT buduje się z bazy danych poprzez rekursywne podziały
- Wybiera się dowolny atrybut różny od „Play” jako korzeń drzewa
- Pozostałe atrybuty dzielą krotki danych na podzbiory,
- Dla każdego podzbioru krotek wybiera się kolejny atrybut dzielący

Drzewa decyzyjne



Drzewa decyzyjne

- Proces podziału zależy od kolejności rozpatrywania atrybutów
- Drzewa decyzyjne mogą być niezrównoważone; drzewa zrównoważone lepiej przewidują wartości atrybutów
- Reguła kciuka: wybieraj atrybuty, które pozwalają zmniejszyć średnią długość ścieżki

- Drzewa do klasyfikacji dokumentów:
  - term indeksujący jest skojarzony z węzłami wewnętrznymi,
  - klasy dokumentów są skojarzone z liśćmi,
  - binarne predykaty wskazujące obecność/nieobecność termów indeksujących są skojarzone z krawędziami.

Drzewa decyzyjne

- Niech:
  - $K = \{k_1, k_2, \ldots, k_t\}$: zbiór termów indeksujących dokumentów
  - C: zbiór klas dokumentów
  - P: zbiór predykatów logicznych termów indeksujących

- Drzewo klasyfikacji:

$$DT = (V, E; r; l_I, l_L, l_E)$$

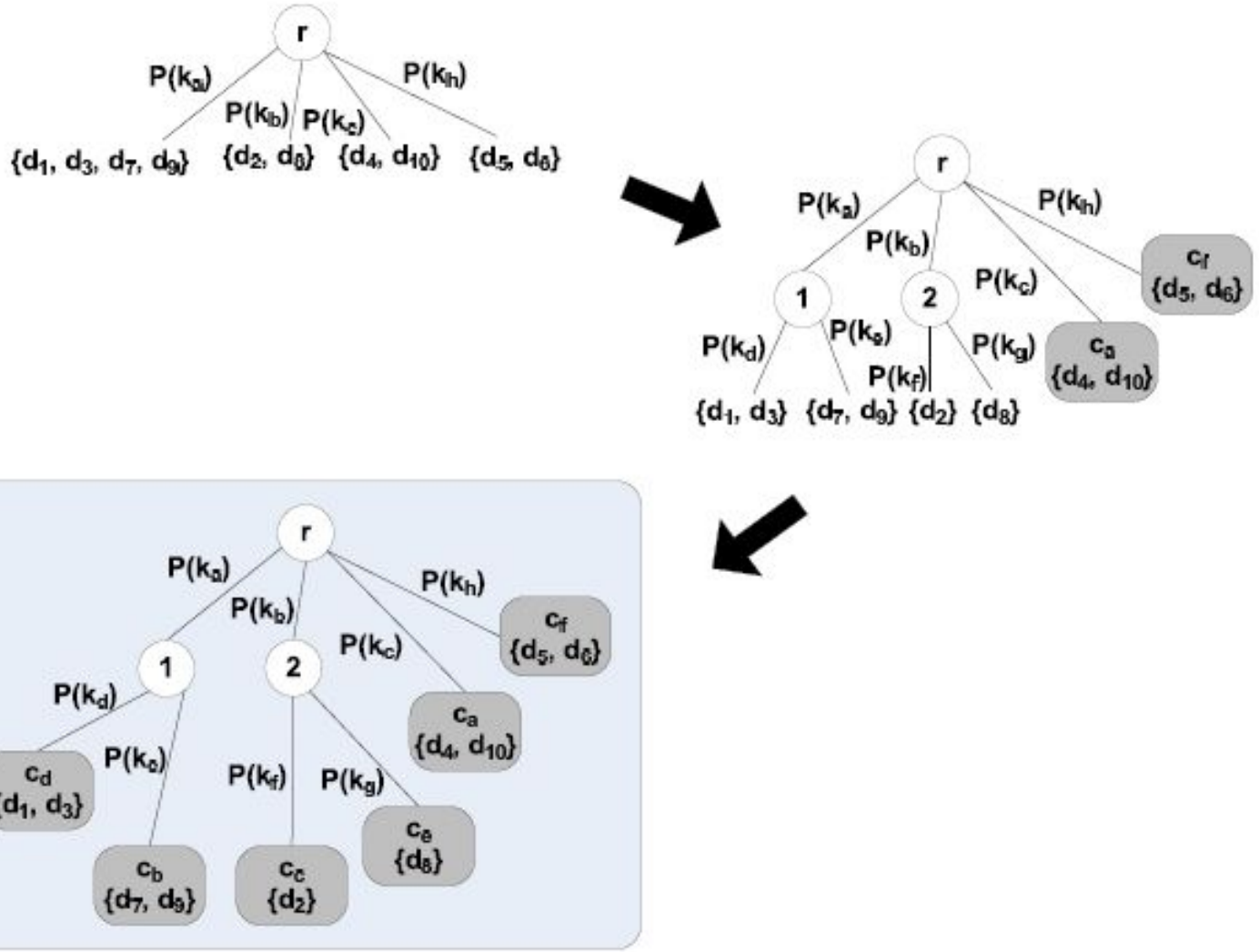
  - (V, E, r): drzewo z korzeniem r,
  - $l_I : I \rightarrow K$: funkcja kojarząca termy indeksowe K z węzłami wewnętrznymi I,
  - $l_L : \bar{I} \rightarrow C$: funkcja kojarząca klasy $c_p \in C$ z liśćmi drzewa,
  - $l_E : E \rightarrow P$: funkcja kojarząca predykaty logiczne P z krawędziami E.

Drzewa decyzyjne

- Tworzenie drzewa przez rekursywne podziały:
  - **krok 1:** skojarzenie dokumentów z korzeniem,
  - **krok 2:** wybierz termy indeksujące zapewniające dobry podział
  - **krok 3:** powtarzaj aż drzewo rozwinie się w pełni

Drzewa decyzyjne

- Termy k_a, k_b, k_c, k_h - termy wybrane do 1 podziału



Drzewa decyzyjne

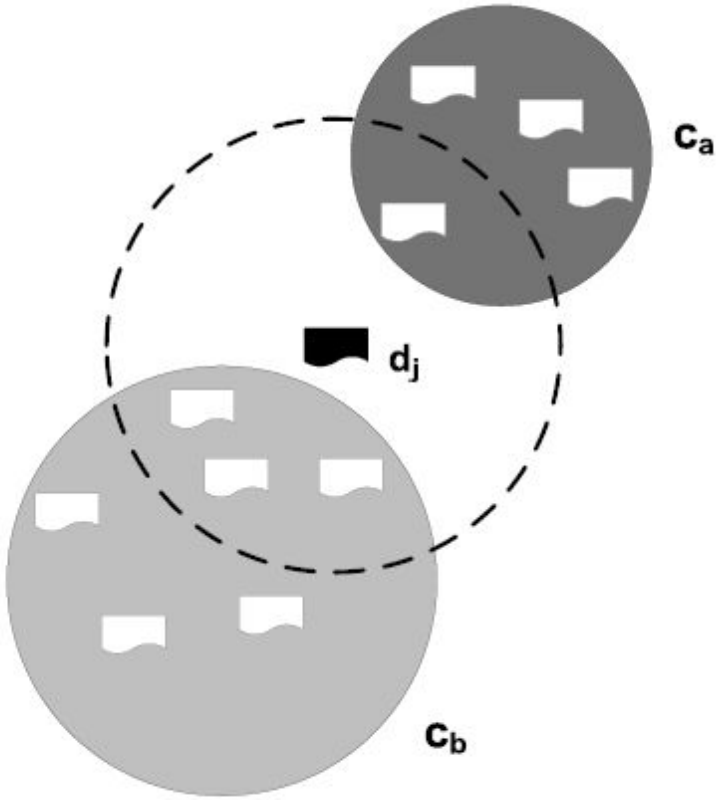
- Wybór termów rozdzielających odbywa się w oparciu o entropię lub tzw. wzmocnienie informacyjne

- Wybór termów z dużym wzmocnieniem informacyjnym
  - zwiększa liczbę gałęzi na danym poziomie drzewa,
  - zmniejsza liczbę dokumentów w podzbiorach wynikowych,
  - buduje mniejsze i mniej złożone drzewa decyzyjne.

- Jeżeli jakoś dokument nie zawiera termów używanych w budowie drzewa decyzyjnego nie wykorzystuje się go przy tej budowie

Klasyfikator k-NN

- Klasyfikacja na żądanie (zwłoczna) – wykonywana gdy prezentowany jest nowy dokument d_j.



Klasyfikator k-NN

- Określa się k najbliższych sąsiadów dokumentu d_j w zbiorze uczącym.
- Klasy tych sąsiadów używa się do ustalenia klasy dokumentu d_j.
- Każdej parze dokument-klasa $[d_j, c_p]$ przypisuje się wartość:

$$S_{d_j,c_p} = \sum_{d_t \in N_k(d_j)} similarity(d_j, d_t) \times \mathcal{T}(d_t, c_p)$$

gdzie:

- $N_k(d_j)$: zbiór k najbliższych sąsiadów dokumentu d_j w zbiorze uczącym
- *similarity*(d_j, d_t): formuła podobieństwa dokumentów,
- $T(d_t, c_p)$: funkcja zbioru uczącego równa 1 gdy $d_t \in c_p$ albo 0 w przeciwnym wypadku

Klasyfikator k-NN

- Klasyfikator przypisuje dokumentowi d_j klasę (klasy) c_p z najwyższym wynikiem
- Klasyfikator musi obliczyć odległości dokumentu d_j do każdego z elementów zbioru uczącego w zadanym otoczeniu
- Innym problemem jest wybór najlepszej wartości k

Klasyfikator Rocchio

- Klasyfikator Rocchio:
  - modyfikuje zapytanie na bazie odpowiedzi użytkownika
  - tworzy nowe zapytanie lepiej wyrażające potrzeby użytkownika
  - może być adaptowany do klasyfikacji tekstu

- Zbiór uczący pełni rolę informacji zwrotnej od użytkownika
  - termy dokumentów uczących z danej klasy c_p dają dodatnie sprzężenie zwrotne,
  - termy dokumentów uczących spoza danej klasy c_p dają ujemne sprzężenie zwrotne.

- Sprzężenie informacyjne zbierane jest przez wektor centroidy
- Nowy dokument jest klasyfikowany na bazie odległości od tej centroidy

Klasyfikator Rocchio

- Dokument d_j:

$$\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

  - $w_{i,j}$: waga termu k_i w dokumencie d_j,
  - t: rozmiar słownika.

- Klasyfikator Rocchio dla klasy c_p:

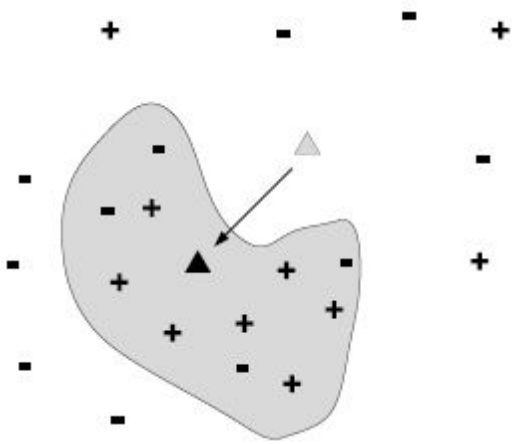
$$\vec{c}_p = \frac{\beta}{n_p} \sum_{d_j \in c_p} \vec{d}_j - \frac{\gamma}{N_t - n_p} \sum_{d_l \notin c_p} \vec{d}_l$$

gdzie

  - n_p: liczba dokumentów w klasie c_p,
  - N_t: całkowita liczba dokumentów w zbiorze uczącym.

Klasyfikator Rocchio

- +: termy w klasie c_p,
- -: termy poza klasą c_p.



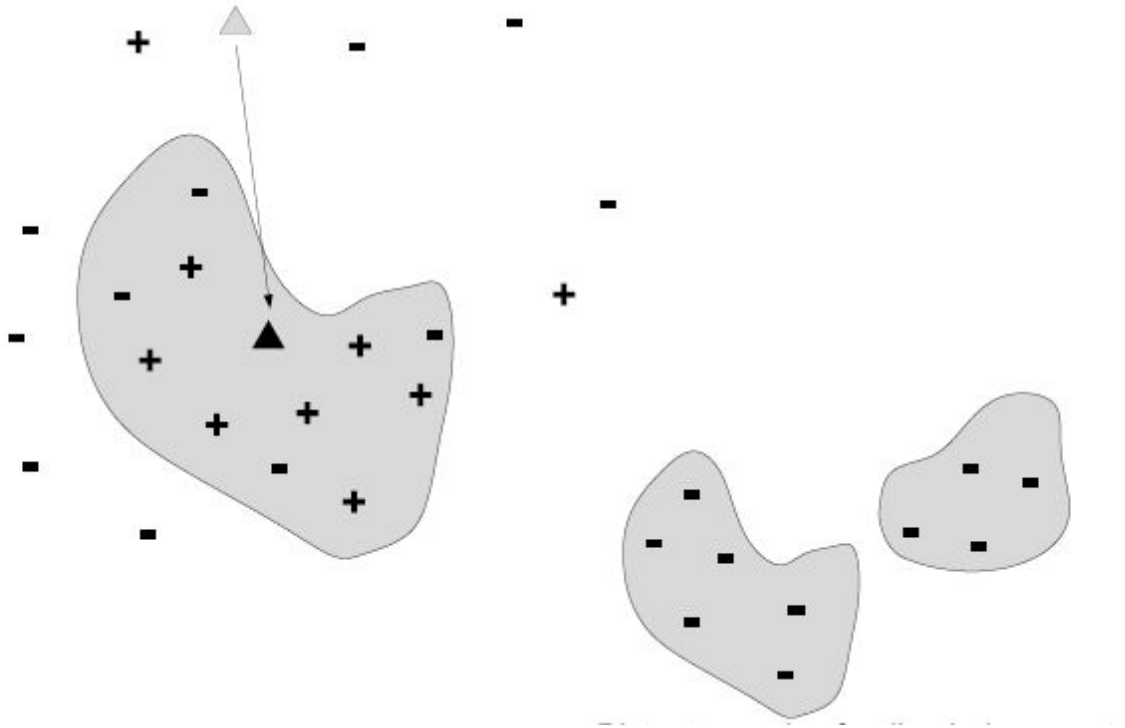
- Klasyfikator przypisuje każdej parze $[d_j, c_p]$ wartość:

$$S(d_j, c_p) = |\vec{c}_p - \vec{d}_j|$$

- Klasa z najmniejszą wartością S jest przypisana do d_j.

Klasyfikator Rocchio

- Problem: ujemne sprzężenie przemieszcza niekorzystnie centroidę



- Wykorzystuje się jedynie „najbardziej pozytywne” dokumenty z ujemnym sprzężeniem informacyjnym

Prosty klasyfikator Bayesa

- Każdej parze $[d_j, c_p]$ przypisuje się prawdopodobieństwo $P(c_p|\vec{d}_j)$ (tw. Bayesa)

$$P(c_p|\vec{d}_j) = \frac{P(c_p) \times P(\vec{d}_j|c_p)}{P(\vec{d}_j)}$$

  - $P(\vec{d}_j)$: prawdopodobieństwo wylosowania dokumentu $\vec{d}_j$
  - $P(c_p)$: prawdopodobieństwo wylosowania dokumentu w klasie c_p
- Nowym dokumentom przypisuje się klasy o największym prawdopodobieństwie
- Aby uprościć wyznaczenie $P(\vec{d}_j|c_p)$ zakłada się niezależność termów indeksujących

Prosty klasyfikator Bayesa

- Dokument d_j jest reprezentowany przez wektor wag binarnych

$$\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

$$w_{i,j} = \begin{cases} 1 & \text{if term } k_i \text{ occurs in document } d_j \\ 0 & \text{otherwise} \end{cases}$$

- Do każdej pary [d_j, c_p] przypisuje się wartość

$$S(d_j, c_p) = \frac{P(c_p|\vec{d}_j)}{P(\overline{c}_p|\vec{d}_j)} \sim \frac{P(\vec{d}_j|c_p)}{P(\vec{d}_j|\overline{c}_p)}$$

Prosty klasyfikator Bayesa

- Przy założeniu niezależności termów

$$P(\vec{d}_j|c_p) = \prod_{k_i \in \vec{d}_j} P(k_i|c_p) \times \prod_{k_i \notin \vec{d}_j} P(\overline{k}_i|c_p)$$

$$P(\vec{d}_j|\overline{c}_p) = \prod_{k_i \in \vec{d}_j} P(k_i|\overline{c}_p) \times \prod_{k_i \notin \vec{d}_j} P(\overline{k}_i|\overline{c}_p)$$

- Stąd

$$S(d_j, c_p) \sim \sum_{k_i} w_{i,j} \left(\log \frac{p_{iP}}{1 - p_{iP}} + \log \frac{1 - q_{iP}}{q_{iP}} \right)$$

$$p_{iP} = P(k_i|c_p)$$

$$q_{iP} = P(k_i|\overline{c}_p)$$

Prosty klasyfikator Bayesa

- Prawdopodobieństwa p_{iP} oraz q_{iP} są wyznaczane ze zbioru D_t dokumentów uczacych.

$$p_{iP} = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(c_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)} = \frac{1 + n_{i,p}}{2 + n_p}$$

$$q_{iP} = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(\overline{c}_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(\overline{c}_p|d_j)} = \frac{1 + (n_i - n_{i,p})}{2 + (N_t - n_p)}$$

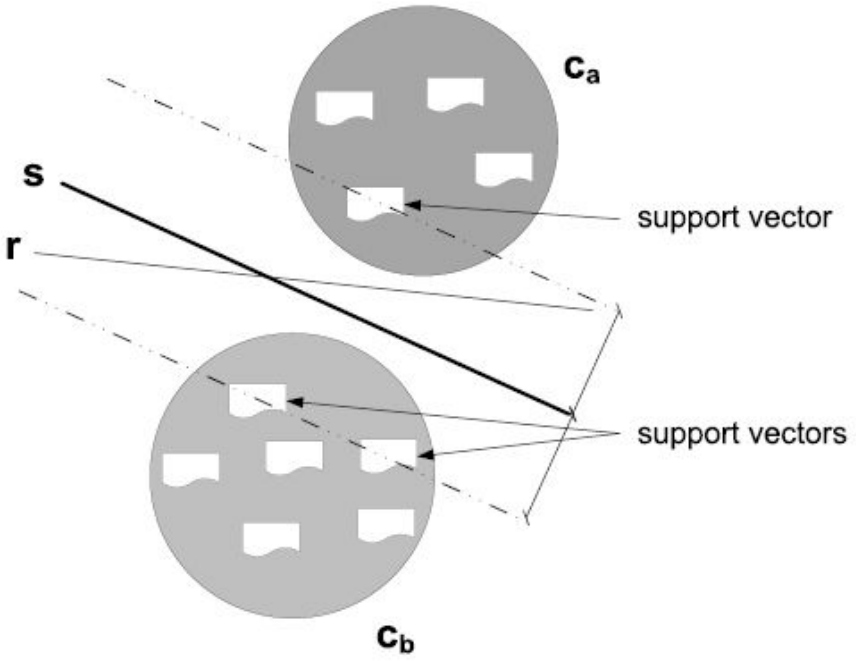
  - $n_{i,p}$, n_i, n_p ,N_t: jak w modelu probablilistycznym (N_t – *liczba dokumentów testowych*, n_i – liczba dokumentów z i-tym termem, $n_{i,p}$ – liczba dokumentów z i-tym termem w klasie c_p.
  - $P(c_p|d_j) \in \{0,1\}$ $oraz$ $P(\bar{c}_p|d_j) \in \{0,1\}$: wzięte ze zbioru uczącego.

- Klasyfikator przypisuje każdemu dokumentowi d_j klasę z największą wartością $S(d_j, c_p)$.

Klasyfikator SVM

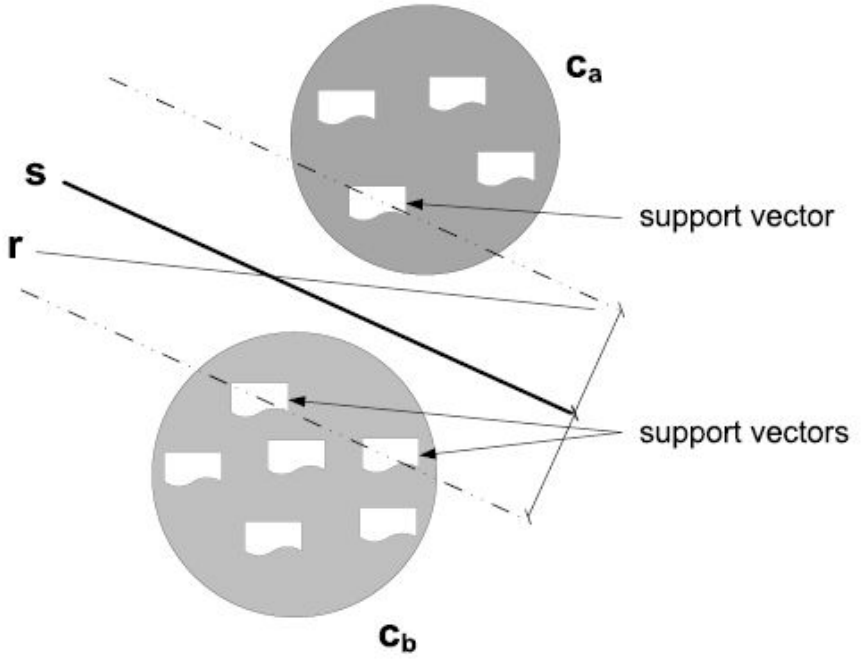
- SVM – metoda wektorów nośnych (*Support Vector Machine*)
- Hiperpłaczczyzna *s* rozdzielająca klasy maksymalizuje odległości do najbliższych dokumentów w każdej z rozdzielanych klas (c_a, c_b na rysunku).

SVM – zakłada liniową separowalność klas dokumentów



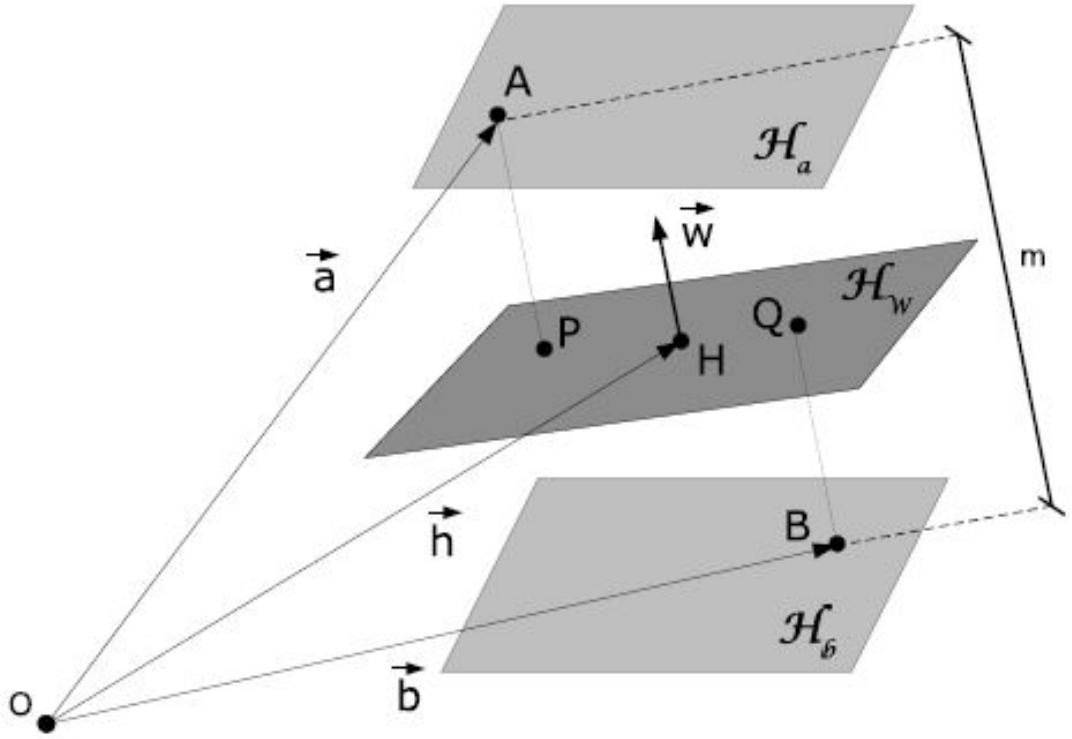
Klasyfikator SVM

- Hpperpłaszczyznę decyzyjną *s* wybiera się ze zbioru płaszczyzn równoległych do hiperpłaszczyzn ograniczających (support vectors) i umieszczonych pomiędzy nimi.



Klasyfikator SVM

- Problem optymalizacji SVM: dla danych wektorów $\vec{a}\ i\ \vec{b}$ znaleźć płaszczyznę H_w, która maksymalizuje margines m.



Klasyfikator SVM

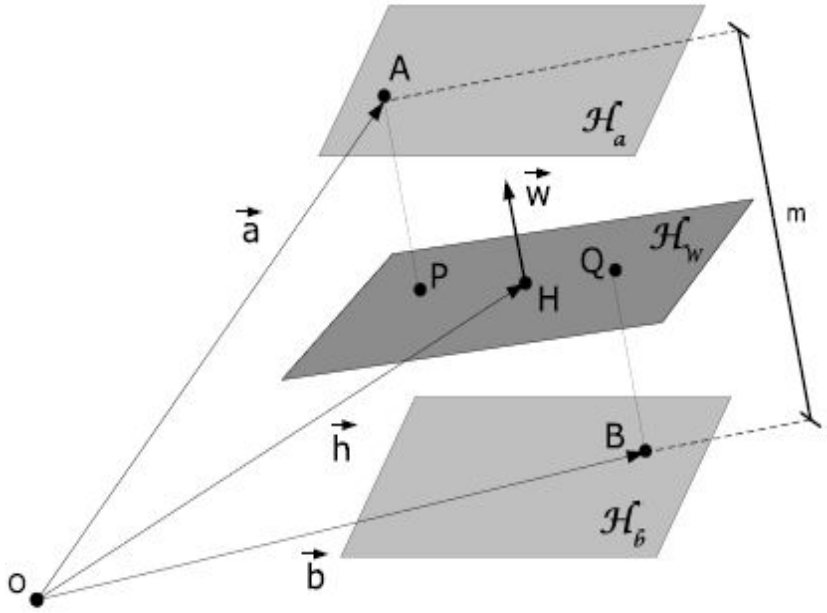
- Hiperpłaszczyzna H_w jest określona przez wektor $\vec{h}$ i prostopadły wektor $\vec{w}$, które nie są z góry znane

$\overline{AP}$: Odległość punktu *A* do hiperpłaszczyzny H_w.

$$\overline{AP} = \frac{\vec{a}\vec{w} + k}{|\vec{w}|}$$

$\overline{BQ}$: Odległość punktu *B* do hiperpłaszczyzny H_w.

$$\overline{BQ} = -\frac{\vec{b}\vec{w} + k}{|\vec{w}|}$$



Klasyfikator SVM

- Margines *m* niezależny od rozmiaru $\vec{w}$:

$$m = \overline{AP} + \overline{BQ}$$

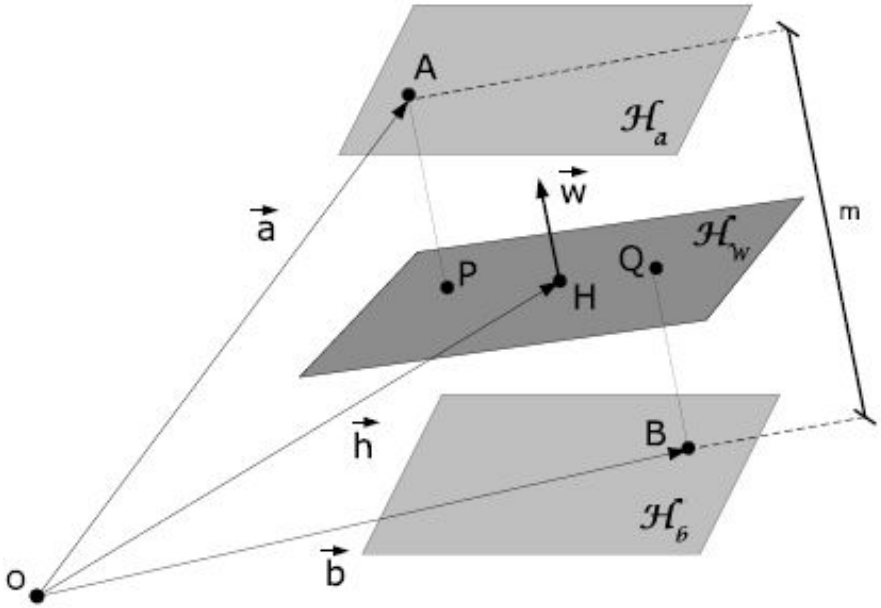
- Założenia ograniczające dla $\vec{w}$:

$$\begin{aligned} \vec{a}\vec{w} + k &= 1 \\ \vec{b}\vec{w} + k &= -1 \end{aligned}$$

- Stad:

$$m = \frac{1}{|\vec{w}|} + \frac{1}{|\vec{w}|}$$

$$m = \frac{2}{|\vec{w}|}$$



Klasyfikator SVM

- Oznaczenia:
  - $T = \{\ldots, [c_j, \vec{z}_j\ ], [c_{j+1}, \vec{z}_{j+1}\ ], \ldots\}$: zbiór uczący,
  - c_j : klasa związana z punktem $\vec{z}_j$ reprezentującym dokument d_j.

- Problem optymalizacji SVM:
- maksymalizacja $m = 2/|\vec{w}|$

  przy ograniczeniach

$$\vec{w}\vec{z}_j + b \geq +1 \text{ if } c_j = c_a$$
$$\vec{w}\vec{z}_j + b \leq -1 \text{ if } c_j = c_b$$

- wektory nośne spełniają warunki równości w powyższym układzie równań

Klasyfikator SVM

- Przykład:
- maksymalizować wartość

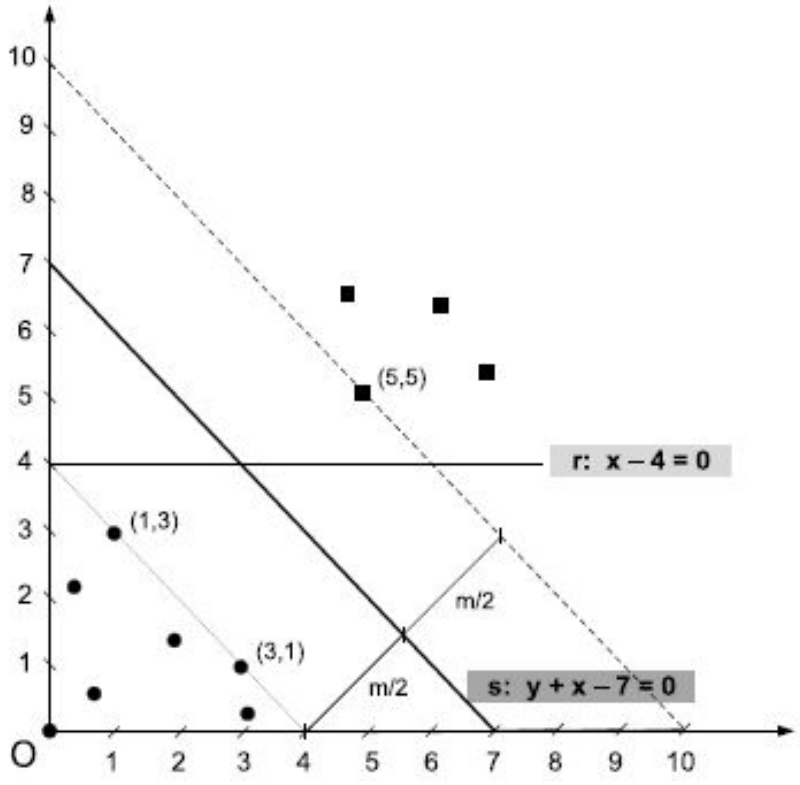
$$m = 2/|\vec{w}|$$

- przy ograniczeniach

$$\vec{w} \cdot (5,5) + b = +1$$
$$\vec{w} \cdot (1,3) + b = -1$$

- $m = 3\sqrt{2}$ - odległość między hiperpłaszczyznami,
- $|\vec{w}| = \sqrt{x^2 + y^2}$
- Stąd:

$$\begin{aligned} 3\sqrt{2} &= 2/\sqrt{x^2+y^2} \\ 5x + 5y + b &= +1 \\ x + 3y + b &= -1 \end{aligned}$$



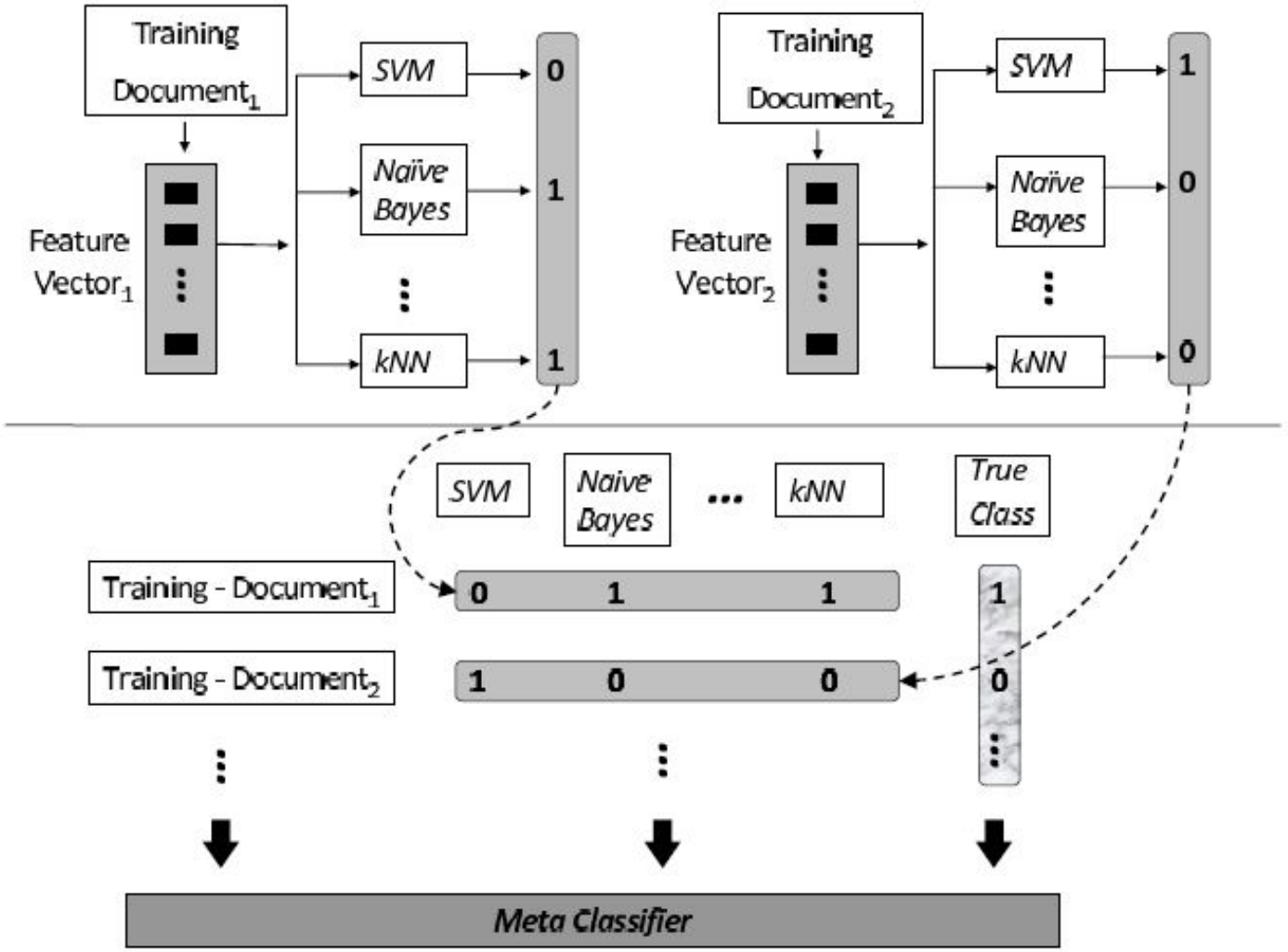
Rozwiązanie:

$$y + x - 7 = 0$$

Klasyfikator SVM

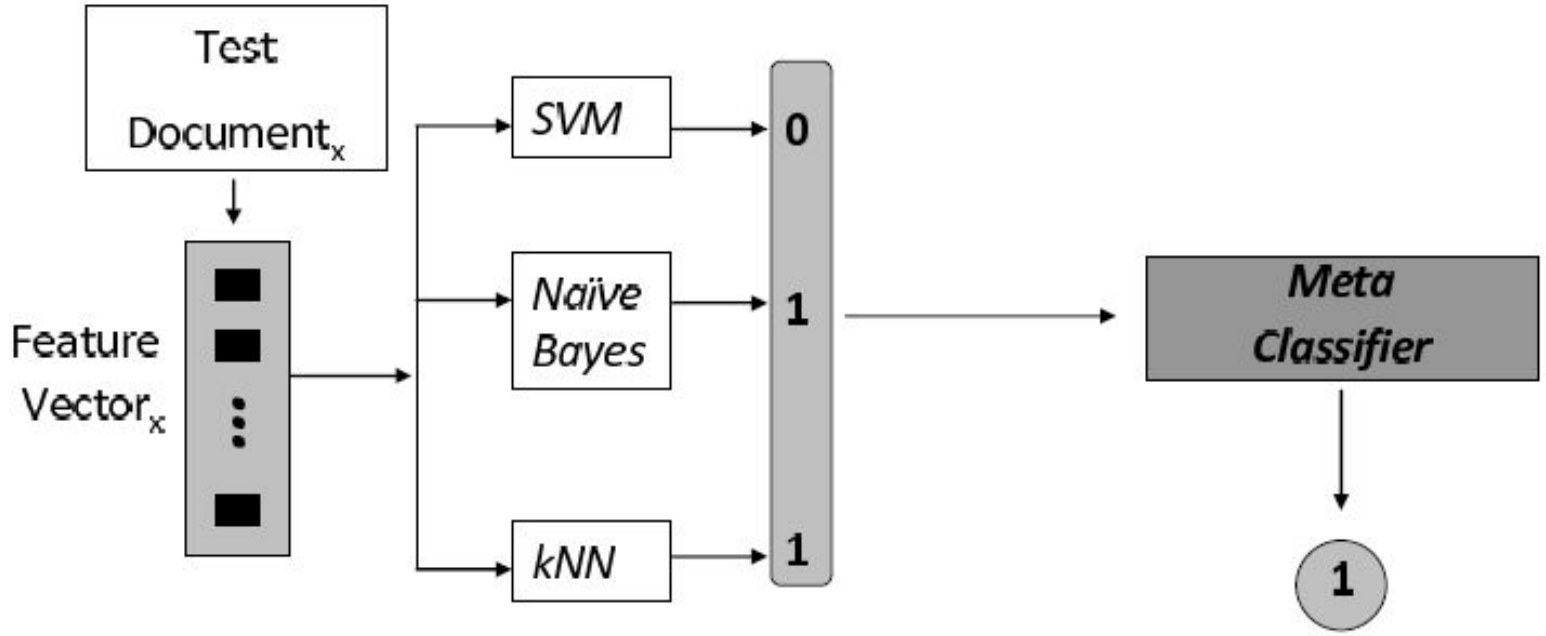
- Klasyfikacja dokumentu d_i (wektora $\vec{z}_j$)
  $$f(\vec{z}_j) = sign(\vec{w}\vec{z}_j + b)$$
- Nowy dokument d_j jest klasyfikowany
  - do klasy c_a: gdy $\vec{w}\vec{z}_j + b > 1$,
  - do klasy c_b: gdy $\vec{w}\vec{z}_j + b < -1$.
- Przy wielu klasach każda wybrana klasa c_p jest oddzielana od pozostałych
- Wybrane klasy d_j – te, które zapewniają największe marginesy m względem innych klas

Klasyfikatory zespołowe (złożone)



Klasyfikatory zespołowe (złożone)

- Metoda stosowa: funkcja ucząca składa wyniki przewidywań poszczególnych klasyfikatorów składowych z określonymi wagami lub wybiera najlepszy



Klasyfikatory zespołowe (złożone)

- Metoda wzmacniania (boosting): łączone klasyfikatory są budowane w wielu iteracjach tą samą metodą uczenia
- W każdej iteracji
  - dokument w zbiorze uczącym ma przypisaną wagę,
  - wagi źle sklasyfikowanych dokumentów zwiększają się po kolejnych iteracjach
- Po n iteracjach
  - wyjścia klasyfikatorów są łączone w sumę ważoną,
  - wagi są związane z błędami estymacji każdego klasyfikatora