ED – Laboratorium 2

Wprowadzenie do uczenia maszynowego

1 Uczenie maszynowe – informacje wstępne

Uczenie maszynowe i **sztuczna inteligencja** to blisko powiązane ze sobą dziedziny. W obu z nich cel jest podobny: wytworzenie „czegoś", co będzie potrafiło zachowywać się inteligentnie. Jak wiemy, inteligencja nie jest prosta do zdefiniowania. Niektórzy twierdzą, że ludzie to istoty inteligentne. Inni twierdzą, że w ogólności nie.

Kiedy stworzono komputery szybko zauważono, że potrafią realizować pewnego rodzaju obliczenia szybciej niż ludzie. Tak w ogóle samo słowo 'komputer' było kiedyś nazwą na człowieka, którego praca polegała na rutynowym wykonywaniu obliczeń. Skoro więc maszyny potrafiły przeprowadzać obliczenia albo grać w gry i wygrywać z ludźmi, to może po ulepszeniach będą też potrafiły tworzyć sztukę albo rozważać bezsens życia? Może ludzie to tak naprawdę tylko bardzo skomplikowane maszyny?

Sztuczna inteligencja miała stworzyć maszyny dorównujące ludziom pod każdym względem, jednak pomimo ogromnego początkowego entuzjazmu do dzisiaj nie udało się takich maszyn stworzyć. Uzyskano jednak po drodze bardzo dużo ciekawych wyników. Na przykład programy, które potrafią znajdować regularności w ogromnych zbiorach danych, albo programy zdolne do przeprowadzania dedukcji.

Niezupełnie formalny podział jest taki, że metody „symboliczne", często mające duży związek z logiką, zaliczane są do **sztucznej inteligencji**, a te oparte przede wszystkim na statystyce do **uczenia maszynowego**.

1.1 Zadanie klasyfikacji

Zanim przejdziemy dalej musimy powiedzieć coś o tym, jakie konkretnie zadanie będzie realizować większość algorytmów uczenia maszynowego poznawanych na zajęciach. Jest to **zadanie klasyfikacji**.

O klasyfikacji możemy myśleć jako o łączeniu pewnych elementów w pary. Robimy tak bardzo często w naszym życiu. Może na przykładzie:

	Jedzenie
	Jedzenie
	Picie
	Picie
	Jedzenie

Do każdego produktu przypisaliśmy pewną etykietę mówiącą o tym, czy produkt jest do jedzenia czy do picia. W uczeniu maszynowym taką etykietę nazywa się **klasą decyzyjną**. W tym problemie mamy dwie klasy decyzyjne: 'Jedzenie' i 'Picie'.

Póki co w naszym zadaniu klasyfikacji brakuje wyzwań. Mamy obrazki produktów i wiemy odgórnie, co jest czym. Załóżmy jednak, że pojawia się kolejny element, który widzimy po raz pierwszy w życiu:

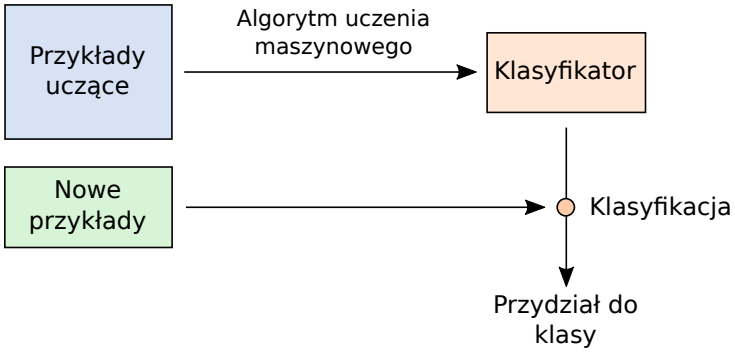
 ?

Nie widzieliśmy wcześniej dokładnie takiego samego. Czy jesteśmy na przegranej pozycji i nie posiadamy żadnych przesłanek by twierdzić, że to jedzenie albo picie? Niezupełnie. Mamy przecież powyżej naszą listę produktów i ich *klas decyzyjnych*. Spróbujmy się dowiedzieć, jakie cechy ma 'Picie', a jakie 'Jedzenie'. W tym wypadku łatwo stwierdzić, że 'Picie' generalnie jest wyższe niż szersze i jakoś tak okrągłe kształty występują w nim częściej niż w jedzeniu.

Taki proces rozumowania, czyli wyciąganie wniosku tłumaczącego dostępne obserwacje, nazywa się **indukcją**. Nie posiadamy żadnej gwarancji, że nasz wniosek będzie obiektywnie poprawny, jednak jak już musimy strzelać to raczej w to, co daje nam wyższe prawdopodobieństwo sukcesu. Algorytmy uczenia maszynowego indukują więc pewną wiedzę z podanego im **zbioru przykładów**, by móc następnie klasyfikować, czyli przypisywać etykiety (*klasy decyzyjne*) elementom spoza tego zbioru.

1.2 Klasyfikator

Poniższy diagram przedstawia sposób wykorzystania algorytmu uczenia maszynowego:



Jak widać, algorytm uczenia maszynowego (górna strzałka) działa na pewnym zbiorze **przykładów uczących**. Algorytmy uczenia maszynowego produkują na podstawie tego zbioru **klasyfikator**. Klasyfikator to taki program komputerowy, który dostając jakiś wcześniej nieznany przykład potrafi przypisać mu klasę decyzyjną (etykietę), czyli mówimy, że potrafi go **zaklasyfikować**. W poprzedniej podsekcji to właśnie odpowiedni klasyfikator niejawnie posłużył nam do nadania etykiety szklance z jakimś płynem (oczywiście bezalkoholowym).

1.3 Przykłady uczące

Rozumiemy już mniej więcej ideę stojącą za przykładami uczącymi. Musimy jednak powiedzieć o nich trochę więcej, ponieważ poprzednio pokazany przykład był trochę uproszczony. Uproszczenie to polegało na tym, że nie rozróżnialiśmy bezpośrednio **cech** obiektów. Standardowe algorytmy uczenia maszynowego potrzebują klarownie wskazanych cech, które mają brać pod uwagę – nie są tak jak my zdolne do ich automatycznego wytworzenia. Musimy im je więc podać na tacy.

Cechy te, nazywane formalnie **atrybutami**, są zdefiniowane dla każdego przykładu uczącego. Mogą przyjmować wartości na dowolnej ze skal opisanych w sekcji 2. Z racji tego, że każdy przykład uczący ma jakieś wartości na wszystkich atrybutach, możemy je przedstawić w tabelce. Poniższa tabelka przedstawia dość popularny zbiór danych uczących dotyczący określania tego, czy pogoda jest dobra do gry w tenisa:

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	sunny	hot	high	weak	NO
D2	sunny	hot	high	strong	NO
D3	overcast	hot	high	weak	YES
...	...	...	...	...	...
D13	overcast	hot	normal	weak	YES
D14	rain	mild	high	strong	NO

Jak widzimy, pewien „ekspert" wyróżnił pewne cechy pogody, które uważał za potencjalnie istotne. Atrybut *PlayTennis* jest specjalny, ponieważ zawiera nasze etykiety, czyli klasy decyzyjne (tutaj mamy dwie klasy decyzyjne: YES i NO). Nazywany on jest **atrybutem decyzyjnym**. Zbiór przykładów uczących ma zawsze tylko jeden taki atrybut. Atrybut *Day* jest z kolei **identyfikatorem**, czyli unikalną nazwą pojedynczego przykładu uczącego. Identyfikatory nie biorą udziału w procesie uczenia. Identyfikatory nie są też obowiązkowe, w przeciwieństwie do atrybutu decyzyjnego, który przed rozpoczęciem uczenia zawsze musi zostać wyróżniony.

1.4 Więcej o atrybutach

1.4.1 Brakujące wartości

Czasami dla jakiegoś przykładu nie znamy jego wartości na pewnym atrybucie. Może to wynikać z różnych powodów. Przykładowo, pewnego dnia mógł nam się zepsuć miernik wilgotności i nie mamy żadnego jej pomiaru na tamten dzień.

Możliwe sposoby radzenia sobie:

- wstawienie wartości występującej najczęściej dla danego atrybutu,
- wstawienie specjalnej wartości atrybutu dla nieokreślonych przypadków (wartość 'unknown'),
- odrzucenie przypadku z brakującą wartością.

1.4.2 Dyskretyzacja

Dyskretyzacja jest przejściem z ciągłej dziedziny (np. liczby rzeczywiste) do dyskretnej (liczby całkowite lub uporządkowane etykiety). Ciągła dziedzina może przyjmować dowolne wartości pośrednie, podczas gdy dziedziny dyskretne mają wyróżnione „kroki", pomiędzy którymi wiemy, że żadnych wartości nie ma (np. dla liczb całkowitych nie ma żadnej liczby między 1 i 2). Dyskretyzacja wiąże się z pewną utratą informacji.

Przykład 1.1 Zamiana pomiaru temperatury w °C na jakościowe określenia, takie jak: *'hot'*, *'mild'*, *'cold'*. Możemy na przykład uznać, że wartości $< 15^oC$ to *'cold'*, następnie $< 30^oC$ to *'mild'*, a $\geqslant 30^oC$ to *'hot'*. ■

Przykład 1.2 Zaokrąglanie do liczb całkowitych. Przykładowo: $25.5 \to 26$, $10.2 \to 10$, $31.9 \to 32$. ■

1.5 Rodzaje algorytmów uczenia maszynowego

Algorytmy uczenia maszynowego można w naturalny sposób podzielić na dwa rodzaje: **uczenie nadzorowane** i **uczenie nienadzorowane**. To rozróżnienie jest bardzo istotne.

1.5.1 Uczenie nadzorowane

Jest to taka sytuacja, w której algorytm uczenia maszynowego dostaje przykłady wraz z ich przypisaniem do klasy decyzyjnej. Dzięki temu możemy łatwo sprawdzić, jak dobry jest nasz algorytm, gdyż znamy optymalne przypisania poszczególnych przykładów uczących. Jest

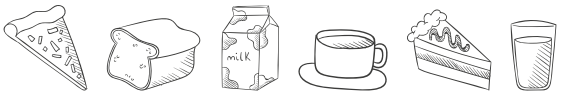
to najbardziej standardowy sposób uczenia i tak właściwie nasze rozważania wcześniej dotyczyły właśnie jego – o ile pamiętasz, mieliśmy klasy decyzyjne 'Jedzenie' i 'Picie' albo 'YES' i 'NO' przypisane odgórnie do przykładów uczących.

Nazwa wzięła się stąd, że możemy sobie wyobrazić, że to pewnego rodzaju „nauczyciel" podaje właściwe przypisania do klas. Nie wnikając w ontologiczny status nauczyciela, ktoś/coś nadzoruje naszym uczeniem podając poprawne odpowiedzi dla przykładów uczących. Zazwyczaj algorytm uczenia maszynowego dostaje je od jakiegoś eksperta albo znamy je z przeszłości.

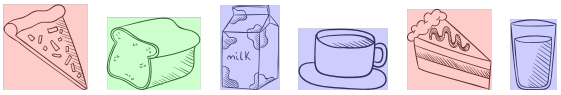
1.5.2 Uczenie nienadzorowane

Tutaj mamy nieco odmienną sytuację. Mamy różne przykłady uczące, ale nie dysponujemy ich przypisaniem do klasy decyzyjnej. Naszym celem będzie de facto właśnie wytworzenie „klas decyzyjnych", nazywanych **skupieniami** (ew. klastrami), przez odpowiednie pogrupowanie obiektów. Obiekty musimy pogrupować na podstawie ich wzajemnego podobieństwa. Jest to zasadniczo jedyna informacja, której możemy do tego celu użyć. Zadaniem realizowanym przez uczenie nienadzorowane nie jest więc klasyfikacja, a **grupowanie**.

Nasz zbiór przykładów uczących wygląda następująco (zwróć uwagę na brak przypisanych etykiet):



W wyniku działania algorytmu uczenia nienadzorowanego może powstać na przykład taki podział na grupy:



Kolor jest tu oznaczeniem danej grupy (skupienia) przykładów zwróconej przez algorytm. Przykłady te są dość podobne w obrębie danej grupy i raczej się dość mocno różnią od przykładów z innych grup.

Algorytmy uczenia nienadzorowanego są w pewnym sensie wyjątkowe, gdyż nie tworzą żadnego klasyfikatora. Po prostu zadanie jest inne – chcemy tak naprawdę sprawdzić, które elementy są podobne do innych. Następnie ekspert może spojrzeć na podział zaproponowany przez algorytm i nadać interpretacje różnym grupom (np. stwierdzić, że niebieska grupa powyżej dotyczy napojów, a czerwona jedzenia w kształcie trójkątów).

1.5.3 Uwagi końcowe

Poniższa tabelka przedstawia nazwy zadań realizowanych przez poszczególne rodzaje uczenia oraz przykładowe algorytmy.

Rodzaj uczenia	Zadanie	Przykłady
uczenie nadzorowane	klasyfikacja, regresja	ZeroR, OneR, k-NN, drzewa decyzyjne, klasyfikator Bayesa, sztuczne sieci neuronowe (backpropagation)
uczenie nienadzorowane	grupowanie, odkrywanie asocjacji	k-means, k-medoids, Apriori

2 Skale pomiarowe

Skala jest pewnym sposobem interpretacji pomiarów/danych. Możemy o niej myśleć jako o dodatkowej (meta-)informacji, która opisuje wzajemne relacje i możliwe operacje między wartościami w niej wyrażonymi. Omówimy różne rodzaje skal na przykładzie.

2.1 Przykład i omówienie

Scenariusz: dokonujemy pomiarów temperatury w ciągu pięciu kolejnych dni. Za każdym razem notujemy odczyty czterech różnych termometrów, z których *przypadkiem* każdy daje wyniki na skali innego rodzaju.

(Termometr 1) Skala nominalna

Zanotowane pomiary: 'jakaśA', 'jakaśB', 'jakaśC', 'jakaśD', 'jakaśD'

Jak widzimy, termometr nie zaszalał z podawaną nam informacją. Potrafimy tylko rozpoznać, kiedy temperatury są różne, a kiedy równe. Nie mamy pojęcia, jak ma się *'jakaśC'* do *'jakaśD'*. Innym przykładem skali nominalnej może być płeć ('K' i 'M'). Wartości na takiej skali to w zasadzie etykiety, nawet jeżeli wyglądają jak liczby (gdyby ten termometr zwracał zamiast *'jakaśX'* „liczby", i zwracałby '1', '2', '3', '4', '4', to mając wiedzę o nominalności wiemy, że twierdzenie o wyższości odczytu jednego z pomiarów nad innym jest nieuprawnione).

(Termometr 2) Skala porządkowa

Zanotowane pomiary: 'bardzo wysoka', 'średnia', 'niska', 'wysoka', 'wysoka'

Drugi termometr jest nieco lepszy. Wiemy, jak się mają temperatury między poszczególnymi dniami, to znaczy czy były wyższe czy niższe. Mamy określony z góry porządek ('bardzo niska', 'niska', 'średnia', 'wysoka', 'bardzo wysoka'). W skali nominalnej takiego porządku nie było. Nie mamy jednak niestety pojęcia, o ile dokładnie wysoka temperatura jest wyższa od średniej.

(Termometr 3) Skala przedziałowa

Zanotowane pomiary: 40°C, 20°C, 10°C, 30°C, 30°C

Z takiego termometru korzysta większość z nas. Skala przedziałowa (zwana też interwałową) pozwala, poza uporządkowaniem pomiarów, obliczać również różnice między nimi.

(Termometr 4) Skala ilorazowa

Zanotowane pomiary: 313 K, 293 K, 283 K, 303 K, 303 K

Jednostką temperatury rozpatrywaną w fizyce nie są °C a Kelviny (K). 0°C to w przybliżeniu 273 K. Kelviny mają pewną bardzo dobrą własność: przy braku *jakiegokolwiek* ciepła obiekt ma temperaturę 0 Kelvinów. Nie można mieć niższej temperatury[1]. Nazywamy ją *zerem bezwzględnym*. Na każdej skali ilorazowej możemy wyróżnić jakieś zero bezwzględne.

Skąd się zatem wzięła nazwa *skala ilorazowa*? Mianowicie możemy przy zerze bezwzględnym zacząć mówić o tym, że coś jest ileś tam razy większe od innego. Jak coś ma 1 K, a coś innego ma 20 K, to jest dwadzieścia razy bardziej ciepłe. Przy skali wyłącznie przedziałowej to nie ma sensu. Jak mamy temperaturę 20°C i 40°C, to ta druga w rzeczywistości wcale nie jest 2 razy większa (czy potrafisz powiedzieć, dlaczego?). A są jeszcze w skali Celsjusza temperatury ujemne...

[1] Aczkolwiek fizycy umówili się, by w pewnych szczególnych sytuacjach przypisywać cząsteczkom ujemną liczbę Kelvinów. Nie bierzemy tego pod uwagę.

2.2 Uwagi

Skale tworzą pewnego rodzaju hierarchię, w której dana skala ma „dostęp” do wszystkich operacji możliwych na skalach położonych niżej w hierarchii. Nietrudno zgadnąć, że na szczycie hierarchii znajduje się skala ilorazowa. Następnie jest skala przedziałowa, potem porządkowa, a na końcu nominalna.

3 Miary jakości modelu

Jak już wiemy, uczenie maszynowe służy do tworzenia (między innymi) klasyfikatorów, które mogą być następnie wykorzystywane do klasyfikacji nowych przykładów. Nie każde dwa klasyfikatory są jednak sobie równe – niektóre będą działać lepiej, podczas gdy inne gorzej. W tym rozdziale omówimy więc różne miary oceniania ich jakości.

3.1 Trafność klasyfikacji

Najprostszą miarą jakości klasyfikatora jest jego **trafność klasyfikacji**, tj. procent przypadków, w których trafnie rozpoznaje on klasę decyzyjną. Jakkolwiek najczęściej wykorzystywana, to należy mieć na uwadze, iż nie zawsze jest to najlepsza możliwa miara: wyobraźmy sobie lekarza badającego grupę osób pod względem obecności jakiejś rzadkiej choroby (np. występującej jedynie u 1% populacji). Jeżeli taki lekarz zawsze będzie mówił pacjentom, że są zdrowi, osiągnie on bardzo wysoką trafność (aż 99%)! Mimo to, nazwalibyśmy go szarlatanem – nie rozpoznał on ani jednego przypadku choroby! Podobnie jest z klasyfikatorami: sama trafność klasyfikacji nie wystarczy aby stwierdzić, czy dobrze one działają, zwłaszcza w przypadku dużego niezbalansowania liczebności klas decyzyjnych.

Przykład 3.1 — Liczenie trafności klasyfikacji. Załóżmy, że lekarz poprawnie zdiagnozował 99 pacjentów, a 26 niepoprawnie. Łącznie miał on więc 125 pacjentów. Trafność klasyfikacji wynosić będzie:

$$\frac{99}{125} = 0.792 = 79.2\%$$

■

3.2 Macierz pomyłek

Po zakończeniu procedury testowania trafności naszego modelu zostajemy z listą przykładów testowych, gdzie dla każdego z nich znamy klasę rzeczywistą oraz klasę przewidywaną przez nasz model. Zliczając liczbę przypadków dla każdej z kombinacji tych dwóch klas (rzeczywistej i przewidywanej) możemy stworzyć tzw. **macierz pomyłek**.

	a	b	c
a	10	2	3
b	0	8	5
c	1	1	8

(a) Macierz pomyłek.

	a	¬ a
a	10	5
¬ a	1	22

(b) Macierz pomyłek dla klasy a.

	a	¬ a
a	TP	FN
¬ a	FP	TN

(c) Oznaczenia wartości macierzy pomyłek.

Tabela 1: W wierszach: klasa rzeczywista; w kolumnach: klasa przewidywana.

Tabela 1a przedstawia przykładową macierz pomyłek. Możemy na jej podstawie obliczyć trafność modelu: musimy zsumować wartości na głównej przekątnej (przypadki poprawnie rozpoznane) i podzielić je przez liczbę wszystkich przypadków testowych. W tym wypadku trafność wynosi $\frac{26}{38} \approx 68\%$. Możemy ponadto wyciągnąć z niej dodatkowe informacje: widzimy m.in. że klasa b często była przez nasz klasyfikator mylona z klasą c, podczas gdy nigdy nie pomyliliśmy jej z klasą a.

Tabela 1b przedstawia tą samą macierz pomyłek dla klasy a – tym razem połączyliśmy klasy b i c jako klasę $\neg a$: możesz porównać wartości w obu macierzach aby się upewnić czy dobrze rozumiesz w jaki sposób przeszliśmy z macierzy 1a do 1b.

Jeżeli mamy już macierz pomyłek dla klasy a, możemy nadać nazwy jej czterem wartościom zgodnie z tabelą 1c. W tej macierzy skupiamy się na klasie a, więc jako *positive* będziemy rozumieli rozpoznanie przykładu jako a, a jako *negative* rozpoznanie przykładu jako coś innego niż a. Rozpoznania mogą być jednak poprawne – *true* oraz błędne – *false*. Jak już się pewnie spodziewasz, te cztery wartości będziemy nazywali odpowiednio:

TP – *true positive*,

FP – *false positive*,

FN – *false negative*,

TN – *true negative*.

3.3 TPrate, FPrate, Precision, Recall

Dla każdej z klas możemy obliczyć wiele wartości o różnych interpretacjach. Tutaj skupimy się na czterech z nich: **TPrate**, **FPrate**, **Precision** oraz **Recall**, przy czym *Recall* jest jedynie inną nazwą na *TPrate*. Pamiętaj, że te miary wyliczane są zawsze osobno dla poszczególnych klas (w poniższych przykładach, dla klasy a)!

A więc idąc po kolei:

- TPrate(a) / Recall(a)
  - **Wzór**: $\frac{TP}{TP + FN}$
  - **Interpretacja**: Jaką część przypadków klasy a udało nam się poprawnie wykryć? Miejmy nadzieję, że jak najwięcej!
- FPrate(a)
  - **Wzór**: $\frac{FP}{FP + TN}$
  - **Interpretacja**: Jaką część przypadków negatywnych ($\neg a$) błędnie rozpoznaliśmy jako klasę a? Miejmy nadzieję, że jak najmniej!
- Precision(a)
  - **Wzór**: $\frac{TP}{TP + FP}$
  - **Interpretacja**: Jaka część naszych rozpoznań klasy a była poprawna (jak często rozpoznając klasę a mieliśmy rację)? Miejmy nadzieję, że jak największa!

Jeśli obliczymy powyższe miary dla klasy *chory* i powyżej wspominanego przypadku szarlatana, możemy łatwo wykryć jego oszustwo: $TPrate(chory) = 0\%$, $FPrate(chory) = 0\%$, a $Precision(chory)$ nie jesteśmy w stanie obliczyć ponieważ nikt nie został rozpoznany jako chory (a więc byśmy musieli dzielić przez zero). Czy potrafisz wskazać, która z tych miar go demaskuje?

4 Testowanie

A więc nasz algorytm uczenia maszynowego wyprodukował jakiś klasyfikator, co teraz? Skąd możemy wiedzieć, czy będzie on dobrze działać dla zupełnie nowych danych?

Pierwsza odpowiedź, która przychodzi do głowy, to: „sprawdźmy czy jego przewidywania zgadzają się z rzeczywistością!". Tutaj jednak pojawia się problem: jakich danych użyć do przeprowadzenia takich testów? Pierwszą myślą może być użycie tych samych przykładów na podstawie których się uczyliśmy... co zwykle będzie bardzo złym pomysłem.

Czemu? Jak temu zaradzić? Czy istnieją różne metody przeprowadzania testów? Odpowiedzi na te i inne pytania znajdziemy poniżej.

4.1 Test na zbiorze uczącym

Jedną z opcji jest pójście za pierwotnym instynktem i przetestowanie naszego modelu na danych uczących. Dlaczego jest to zły pomysł?

Wyobraź sobie drobny zbiór uczący zawierający trzy przykłady: $(pizza, food)$, $(cake, food)$, $(spaghetti, food)$. Praktycznie każdy algorytm zauważy, iż poprawną klasą decyzyjną jest zawsze $food$, więc wszystkie nowe przykłady będą również klasyfikowane w ten sposób. Testując model na zbiorze uczącym, okaże się, że ma on 100% trafność. Hurra! Hurra?

Nie całkiem. Wyobraź sobie, że zadowoleni z rezultatów naszego klasyfikatora zaczynamy stosować go w praktyce. Nagle się okazuje, że wszystkie napoje są również niepoprawnie klasyfikowane jako jedzenie! Testy nas oszukały!

Oczywiście jest to dość drastyczny przykład, ale pokazuje główny problem z testowaniem modeli na danych uczących: będą one zawsze zwracać zawyżoną trafność. Chociaż nasz klasyfikator może być (i zazwyczaj będzie) dobrze dopasowany do danych na których się uczył, może on mieć problem z uogólnieniem wiedzy na przykłady z którymi się jeszcze nie zetknął. A jeżeli nie potrafi uogólnić wiedzy to znaczy, że w praktyce jego skuteczność będzie niska.

Można by tu wysunąć analogię do studentów uczących się na kolokwium: wykucie się odpowiedzi do pytań z poprzedniego roku niewiele pomoże, jeżeli w roku następnym pytania się zmienią. Gdyby pytania się nie zmieniły, studenci otrzymaliby wysokie oceny sugerujące dobrą znajomość tematu. Jeżeli jednak pytania się zmienią, dobre oceny uzyskają jedynie ci z nich, którzy rzeczywiście zrozumieli temat. W tym sensie, testowanie algorytmu na danych uczących jest niczym sprawdzanie wiedzy w oparciu o stare pytania o znanych odpowiedziach.

4.2 Test na wydzielonym zbiorze testowym

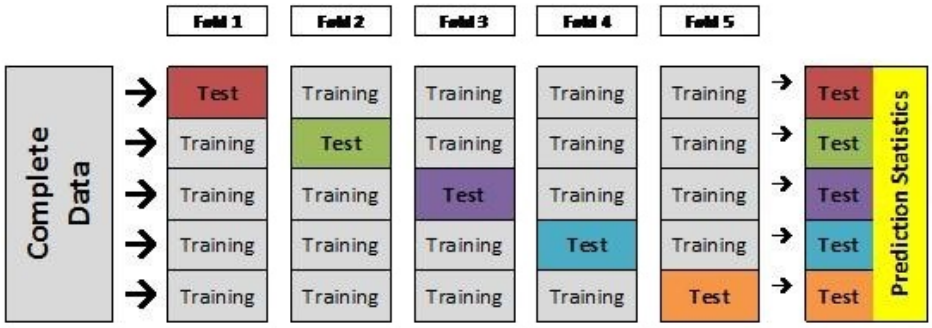
Wiemy już, że (i czemu) nie należy używać danych uczących do testowania modelu. Oczywistym więc będzie próba znalezienia innych danych na których możemy sprawdzić trafność naszego modelu. Możemy użyć osobnego pliku z danymi, albo wydzielić z danych uczących część testową. Jaka jest różnica między tymi podejściami?

Finalny klasyfikator jest zawsze uczony na pełnych danych uczących. Jeśli używamy osobnego pliku z danymi testowymi, wyniki testów będą dotyczyć dokładnie tego klasyfikatora, który został nam zaprezentowany. W przypadku wydzielenia fragmentu testowego z danych uczących, wyniki testów nie będą dotyczyły bezpośrednio klasyfikatora który został nam zaprezentowany, ale klasyfikatora nauczonego na mniejszej liczbie danych!

4.3 K-fold cross validation

Zakładając, że chcemy przetestować jakość naszego modelu ale nie posiadamy drugiego pliku z danymi, czy wydzielenie stałego fragmentu testowego z danych uczących jest najlepszą opcją? Nie, lepszym podejściem jest **k-fold cross validation** (znane również jako: kroswalidacja, sprawdzian krzyżowy).

Powiedzmy, że wyznaczyliśmy 33% zbioru uczącego jako dane testowe. Zauważmy, że to 33% nie zostało jeszcze użyte do nauki, a jednocześnie pozostałe 66% nie było użyte do testowania.



Rysunek 1: Wizualizacja kroswalidacji. Źródło: http://blog-test.goldenhelix.com/wp-content/uploads/2015/04/B-fig-1.jpg

Jest to marnotrawstwo perfekcyjnie dobrych danych, zwłaszcza jeżeli pechowo to 33% będzie zawierać szczególnie istotne przypadki z granic między klasami.

Kroswalidacja działa następująco. Zbiór uczący dzielony jest na k równych części (ergo k-fold). Wybierana jest jedna z nich i następnie używana jest jako zbiór testowy dla modelu nauczonego na pozostałych $k-1$ fragmentach. Procedura ta powtarzana jest $k-1$ razy, przy czym za każdym razem wybierany jest inny fragment jako dane testowe (czyli sumarycznie mamy k iteracji – tyle ile części na które podzieliśmy zbiór). Otrzymane k wyników (np. trafności klasyfikacji) jest uśredniane. W ten sposób użyliśmy całego zbioru uczącego jako danych testowych, nie napotykając się jednocześnie na problemy związane z testowaniem modelu na zbiorze uczącym.