Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Informatyki
Zakład Systemów Informacyjnych

Praca dyplomowa magisterska

na kierunku INFORMATYKA
w specjalności Inżynieria Systemów Informatycznych

Meta-metody służące do poprawy jakości klasyfikacji

Konrad Ziaja
nr albumu 272170

promotor
dr inż. Łukasz Skonieczny

Warszawa 2017

Warszawa, XX lutego 2017

POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa magisterska pt. Meta-metody służące do poprawy jakości klasyfikacji:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

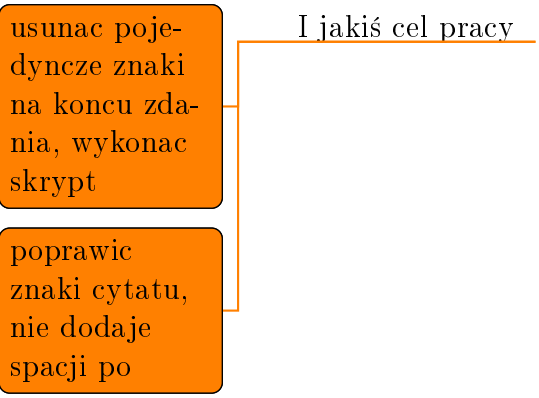
Konrad Ziaja.....

Spis treści

Wstęp

napisac wstep może tutaj napisze o eksploracji danych oraz o klasyfikacji nadzorowanej i nienadzorowanej

Cel pracy


I jakiś cel pracy
usunac pojedyncze znaki na koncu zdania, wykonac skrypt
poprawic znaki cytatu, nie dodaje spacji po

Rozdział 1

Wstęp teoretyczny

1.1 Klasyfikacja danych

może coś pisać o uczeniu maszynowym i że klasyfikacja moze byc nadzorowana i nie?

zbadac rysunki i tabele, czasami pokazuja się zbyt daleko od ich własciwiego położenia

właściwa numeracja stron, lewy prawy, rozdziały od nieparzystych

Klasyfikacja jest to proces przyporządkowania danych do jednej z predefiniowanych klas na podstawie atrybutów tych danych. Algorytm klasyfikacji na podstawie analizy danych trenujących, zawierających atrybuty oraz klasę, tworzy model klasyfikacyjny. Stworzony model klasyfikacyjny wykorzystywany jest do predykcji klasy (kategorii) nowych danych bez określonej klasy. Celem algorytmu budującego model, jest odnalezienie wzorców, w jaki sposób atrybuty obiektu wpływają na przynależność do danej klasy, tak, aby wiedza na temat analizowanych danych była możliwie ogólna oraz niezależna od próby.

Klasyfikacja danych jest procesem dwuetapowym:

- budowa modelu – proces ten polega na analizie obiektów z przyporządkowaną klasą oraz na budowie modelu opisującego predefiniowany zbiór klas danych,
- właściwa klasyfikacja – otrzymany model stosuje się do przydzielania klasy nowym obiektom.

Budowa modelu jest także procesem dwu-etapowym. Dzieli się ona na:

- uczenie – klasyfikator budowany jest w oparciu o dane treningowe,
- ocena jakości klasyfikacji – jakość klasyfikacji badana jest w oparciu o dane testowe.

W zależności od liczebności klas w zbiorze danych można wyróżnić:

- klasyfikację binarną – klasyfikator decyduje o przypisaniu obiektu do jednej z dwóch klas (np. czy człowiek jest zdrowy lub nie)

- klasyfikację wieloklasową – obiektowi przypisuje się jedną z wielu predefiniowanych klas.

Do reprezentacji danych uczących, testowych oraz do klasyfikacji najczęściej stosuje się system informacyjny.

Zachmurzenie	Temp.	Temp. wody	Opady	Wiatr	Pływać	h(x)
słonecznie	32	25	brak	słaby	tak	tak
słonecznie	31	26	brak	umiarkowany	tak	nie
pochmurnie	22	15	brak	b. mocny	nie	nie
pochmurnie	20	18	brak	słaby	tak	tak
całkowite zachmurzenie	12	6	brak	umiarkowany	nie	nie
całkowite zachmurzenie	10	8	duże	słaby	nie	nie
pochmurnie	21	10	brak	mocny	nie	tak
słonecznie	25	17	brak	umiarkowany	tak	nie
pochmurnie	23	17	przelotne	umiarkowany	nie	tak

Tablica 1.1: Przykład danych treningowych składających się z 5 atrybutów oraz klasy decyzyjnej. W ostatniej kolumnie znajduje się wynik klasyfikacji. W pięciu przypadkach, klasyfikator poprawnie wskazał klasę.

napisac o budowie modelu oraz o klasyfikacji przy wszystkich klasyfikatorach

1.2 Wybrane algorytmy klasyfikacji danych

1.2.1 Drzewo decyzyjne

Drzewo decyzyjne jest bardzo często wykorzystywane jako klasyfikator danych. Celem jest stworzenie modelu, który na podstawie danych wejściowych przewidzi poprawnie klasę. Drzewo jest acyklicznym spójnym grafem skierowanym. Korzeń (węzeł na poziomie 0) zawiera w sobie cały zbiór uczący. W każdym węźle przeprowadza się test na wartościach atrybutu, który dzieli zbiór na podzbiory. Z węzła wychodzi tyle gałęzi ile możliwych jest wyników testu z tego węzła. Pod każdym węzłem znajduje się kryterium podziału dokonywanego w danym węźle, które jest jednakowe dla wszystkich elementów zbioru. Ostatnim elementem drzewa decyzyjnego są liście, które zawierają etykiety, czyli przydział klasowy elementów z tego podzbioru. Drzewo buduje się w sposób rekurencyjny od korzenia do liścia z wykorzystaniem metody "dziel i zwyciężaj".

Proces budowy drzewa

1. Stwórz korzeń zawierający cały zbiór uczący.
2. Jeśli wszystkie przykłady należą do tej samej klasy decyzyjnej, to węzeł staje się liściem z etykietą klasy.
3. Jeżeli nie, oblicz kryterium podziału wykorzystując np. entropię, które najlepiej dzieli zbiór treningowy.
4. Dla każdego testu stwórz gałąź i podziel odpowiednio podzbiory do nowych węzłów.
5. Wywołaj rekurencyjnie algorytm dla nowych węzłów.
6. Algorytm kończy się dla kryterium stopu.

Stosuje się różne kryterium stopu:

- wszystkie przykłady należą do tej samej klasy,
- brak możliwości dalszego podziału,
- zbiór pusty,
- osiągnięto zakładany cel, np.: maksymalna głębokość drzewa, maksymalna czystość klas w liściu, minimalny przyrost informacji po podziale.

Jako kryterium podziału można stosować np. wskaźnik Giniego lub entropię. Często zdarza się, że zbudowane drzewa są zbyt duże i tworzy się nadmierne dopasowanie do danych. Wówczas należy ograniczyć wysokość drzewa. Istnieje kilka algorytmów drzew decyzyjnych takich jak: ID3, C4.5, CART, CHAID.

wstawic obrazek drzewa decyzyjnego

1.2.2 Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski opiera się na twierdzeniu o prawdopodobieństwie warunkowym stworzonym przez Thomasa Bayesa. Jest to klasyfikator probabilistyczny, zwracający prawdopodobieństwo przynależności przykładu do danej kategorii. Liczba kategorii musi być skończona i zdefiniowana a priori. Zasada działania klasyfikator opiera się na twierdzeniu:

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$$

gdzie:

- x - przykład, dla którego nieznana jest klasa, jest to n-wymiarowy wektor, a n oznacza liczbę atrybutów,

- $P(C_i|x)$ - prawdopodobieństwo a posteriori, że przykład x należy do klasy C_i,
- $P(x|C_i)$ - prawdopodobieństwo a priori, że przykład x należy do klasy C_i,
- $P(C_i)$ - prawdopodobieństwo, że dowolny przykład należy do klasy C_i
- $P(x)$ to prawdopodobieństwo a priori wystąpienia przykładu x.

Działanie naiwnego klasyfikatora bayesowskiego oparte jest na założeniu, że atrybuty wewnątrz klasy są od siebie niezależne. Bardzo często to założenie nie jest spełnione, co rzutuje na wyniki osiągane przez ten klasyfikator oraz pokazuje genezę pochodzenia nazwy klasyfikatora.

1.2.3 Klasyfikator k najbliższych sąsiadów (kNN)

moze wstawic obrazek?

Klasyfikator k najbliższych sąsiadów lub klasyfikator k-NN, (ang. *k nearest neighbours*) należy do grupy algorytmów leniwych (ang. *lazy learners*), które nie wymagają uczenia, a całe obliczenia wykonywane są w momencie pojawienia się wzorca testowego, czyli podczas klasyfikacji lub testowania. Klasyfikacja nowego obiektu x odbywa się poprzez znalezienie k najbliższych sąsiadów w danych uczących X i nadaniu mu nowej klasy poprzez głosowanie większościowe sąsiadów. Odległość pomiędzy sąsiadami można obliczyć np. takimi miarami jak: euklidesowa, Manhattan, Czebyszewa lub Minkowskiego. Dobranie idealnej wartości parametru k ma bardzo duże znaczenie w jakości klasyfikacji. Dobrze dobrane k powinno być na tyle duże, aby minimalizować prawdopodobieństwo błędnych klasyfikacji, ale jednocześnie małe, aby k najbliższych sąsiadów było rzeczywiście bliskimi sąsiadami testowanej obserwacji [4].

1.2.4 Las losowy

Jest to jeden z elementów bagging, może wspomnieć, albo dać w innym miejscu?

Las losowy (ang. *random forest*) lub losowe drzewa decyzyjne to klasyfikator złożony z drzew decyzyjnych. Algorytm trenujący działa według następującego schematu:

1. Wylosuj metodą boostrap (losowanie ze zwracaniem) n elementów ze zbioru danych,
2. Zbuduj drzewo decyzyjne, w każdym węźle:
   - wylosuj d atrybutów,
   - dla wylosowanych atrybutów, dokonaj najlepszego podziału (np. używając entropii, lub współczynnika Giniego).

3. Powtórz punkt 1 i 2 k razy (gdzie k, to liczba drzew).

Każde drzewo klasyfikuje przykład, a ostateczna klasa nadawana jest poprzez głosowanie większościowe.
Zazwyczaj, im większa liczba drzew k tym lepsze wyniki można otrzymać. Zmieniając liczbę losowanych przykładów n do zbioru uczącego, można kontrolować wariancję błędu. Im n większe, tym las losowy ma większą tendencję do nadmiernego dopasowania. Zmniejszając liczbę losowanych obserwacji można zmniejszyć szansę na przeuczenie i podnieść jakość klasyfikacji. Zazwyczaj n równa się wielkości zbioru danych. Liczbę losowanych atrybutów d, zwykle przyjmuje się na poziomie $d = \sqrt{m}$, gdzie m to ilość wszystkich atrybutów.

1.2.5 Maszyna wektorów nośnych

Maszyna wektorów nośnych, SVM (ang. *support vector machine*) jest to nieprobabilistyczny, binarny, liniowy klasyfikator. Zbudowany model SVM, określa do której z dwóch klas należą dane wejściowe. Konstrukcja klasyfikatora opiera się na znalezieniu hiperpłaszczyzny w przestrzeni wielowymiarowej oddzielającej dwie klasy zbioru uczącego. Zadaniem algorytmu jest wybór tej z możliwie największym marginesem, tak aby odległość najbliższych punktów po obu stronach była największa. Oddzielająca hiperpłaszczyzna, opisana jest z wykorzystaniem wektora normalnego. Wektor ten jest liniową kombinacją najbliższych do hiperpłaszczyzny punktów. Jakość klasyfikacji zależy od szerokości granicy rozdzielającej klasy, im jest większa, tym błąd uogólnienia powinien być mniejszy. Modele z małym marginesem mogą wykazywać nadmierne dopasowanie. Klasyfikacja właściwa nowych przypadków, polega na określeniu po której stronie marginesu znajduje się nowy punkt i na tej podstawie wyznaczana jest klasa.

wstawic obrazek SVMA

Istnieje modyfikacja podstawowego algorytmu, dla przypadku, gdy nie istnieje hiperpłaszczyzna oddzielająca dwie klasy. Jest to maszyna wektorów nośnych o miękkim marginesie. Bierze ona pod uwagę możliwość występowania zaszumionych danych. Klasyfikator w procesie uczenia, szuka możliwie najlepszej hiperpłaszczyzny oddzielającej obie klasy. Stopień błędnej klasyfikacji mierzony jest poprzez zmienną rozluźniającą (ang. *slack variable*). Zmienną tą można traktować jako karę, dla danych znajdujących się po złej stronie granicy. Można ją kontrolować poprzez parametr C, który decyduje o szerokości rozdzielającego marginesu. Duża wartość parametru C, oznacza wysokie kary za błędną klasyfikację.
Jeżeli problem jest nieseparowalny liniowo, można zastosować tzw. trik jądra, który polega na zastąpieniu liniowego jądra, nieliniowym. Pozwala to

na stworzenie klasyfikatora nieliniowego. Najczęściej stosuje się następujące jądra:

- wielomianowe (ang. *polynomial*)
- potencjalnych funkcji bazowych RBF (ang. *radial basis function*)
- sigmoidalne

1.2.6 Zespół klasyfikatorów

W celu poprawy jakości klasyfikacji, można zastosować zespół klasyfikatorów w celu przypisania kategorii nowemu przykładowi. Zespół taki może składać się z kilku klasyfikatorów o takich samych algorytmach lub różnych. Jeżeli są to takie same klasyfikatory, to każdy model uczony jest na innych danych. Dla każdego modelu, dane trenujące mogą być wyznaczane poprzez losowanie lub losowanie ze zwracaniem. Zwykle zbiór trenujący przyjmuje nie więcej niż 2/3 wszystkich elementów. Możliwe jest także, podzielenie zbioru trenującego na $k+1$ (gdzie k to liczba klasyfikatorów) części i przekazanie każdemu klasyfikatorowi różnych k części jako zbiór trenujący. Innym sposobem jest modyfikacja przestrzeni atrybutów. Każdy z klasyfikatorów otrzymuje zbiór danych zawierający inne atrybuty.
Jeżeli są to różne algorytmy, to wszystkie modele można uczyć na takich samych danych.
Każdy klasyfikator otrzymuje nowy przykład i nadaje mu kategorię. Ostateczna kategoria wyznaczana jest poprzez:

- głosowanie większościowe (ang. *majority voting*) wybierana jest klasa najczęściej wskazywana przez modele,
- głosowanie ważone (ang. *weighted voting*) każdy klasyfikator ma przypisaną wagę lub zamiast predykcji klasy, zwraca prawdopodobieństwo z jakim przewiduje daną klasę. Docelową klasą jest ta z największym prawdopodobieństwem.

1.3 Meta-metody

1.3.1 Bagging

Metoda bagging, znana także jako bootstrap aggregating to zespół klasyfikatorów, meta-algorytm pozwalający zmniejszyć wariancję oraz uniknąć nadmiernego dopasowania. Metoda ta została zaproponowana w 1994 przez Leo Breiman'a w celu podniesienia jakości klasyfikacji poprzez połączenie

wielu klasyfikatorów tego samego typ uczących się na losowym zbiorze danych. Zbiór uczący dla każdego klasyfikatora tworzy się poprzez losowanie ze zwracaniem z głównego zbioru danych. Losowanie wykonuje się z rozkładem jednostajnym. Nowo tworzony zbiór może być mniejszy lub takiej samej wielkości. Jeżeli zbiór danych jest duży i wygenerowany zbiór ma taką samą wielkość, można spodziewać się $1 - \frac{1}{e}$ (około 63,2%) unikalnych próbek. Takie losowanie nosi nazwę próby bootstrap. Zazwyczaj łączy się wyniki wielu modeli tego samego typu. Mając zbiór uczący D o rozmiarze n, algorytm wygląda następująco:

1. Wygeneruj m nowych zbiór treningowych D_i, o rozmiarze n', poprzez losowanie ze zwracaniem ze zbioru D,
2. Trenuj m klasyfikatorów w oparciu o wygenerowane zbiory treningowe D_i.

Nowa próbka klasyfikowana jest przez wszystkie modele, a ostateczna klasa nadawana jest poprzez głosowanie większościowe. W metodzie bagging bardzo ważny jest dobór odpowiedniego klasyfikatora oraz rozważenie czy ta metoda może podnieść dokładność klasyfikacji. Jeżeli klasyfikator jest niestabilny (np. mała zmiana w danych treningowych powoduje zmianę dokładności klasyfikatora) metoda bagging może znacząco podnieść jakość i dokładność klasyfikacji. Jeżeli jednak klasyfikator jest stabilny, stosowanie metody bagging może doprowadzić do zmniejszenia jakości klasyfikacji z powodu zmniejszenia ilości danych treningowych.

1.3.2 Boosting

W 1988 i 1989 roku Micheal Kearns oraz Leslie Valiant postawili pytanie czy można stworzyć zbiór słabych klasyfikatorów, w celu stworzenia jednego silnego. Słaby klasyfikator to taki, który osiąga tylko minimalnie lepsze wyniki od losowania (zgadywania), natomiast dobry klasyfikator osiąga wyniki mocno skorelowane z rzeczywistą klasyfikacją. W 1996 roku Robert Schapire oraz Yoav Freund zaprezentowali algorytm AdaBoost. Istnieje dużo różnych algorytmów boostingu. Metoda boosting to zbiór klasyfikatorów, meta-algorytm, w którym w odróżnieniu od baggingu, słabe klasyfikatory budowane są sekwencyjnie. Większość algorytmów boostingu działa według następującego schematu:

1. Każdemu przykładowi ze zbioru początkowego przypisywana jest taka sama waga początkowa, zazwyczaj $\frac{1}{n}$.
2. Budowany jest nowy klasyfikator m w oparciu o zbiór z wagami.

3. Zbudowany klasyfikator dołączany jest do klasyfikatorów M z wagą odpowiadającą jego dokładności.
4. W zbiorze trenującym następuje aktualizacja wag. Przykładom poprawnie sklasyfikowanym zmniejsza się wagę, natomiast błędnie sklasyfikowanym zwiększa się.
5. Punkt 2-4 powtarzany jest do momentu osiągnięcia liczby estymatorów m lub do osiągnięcia zakładanego błędu.

Takie podejście nazywane jest boosting by weighting, w procesie uczenia bierze udział cały zbiór. Istnieje także boosting by sampling, w którym zbiór jest ograniczony do n elementów, a zamiast wag, używa się prawdopodobieństwa. Zwiększając prawdopodobieństwo przykładów źle sklasyfikowanych, zwiększa się szansę na wylosowanie, a w konsekwencji na poprawną klasyfikację przez model.
Algorytm boosting, skupia się na przykładach błędnie klasyfikowanych. Pozwala znacząco poprawić jakość klasyfikacji w przypadku użycia słabych klasyfikatorów (skuteczność klasyfikacji niedużo powyżej 50%). W przypadku klasyfikatorów osiągających lepsze wyniki, nie obserwuje się znaczącego przyrostu skuteczności. Boosting może wykazywać także, nadmierne dopasowanie do przykładów wielokrotnie źle klasyfikowanych.
W 2008 roku Phillip Long (Google) oraz Rocco A. Serveido (Uniwersytet Columbia), podczas 25. Międzynarodowej Konferencji poświęconej systemom uczącym, opublikowali artykuł, w którym zasugerowali, że większość stosowanych współcześnie algorytmów opartych o boosting jest wadliwa. Stwierdzili, że algorytmy AdaBoost, LogiBoost mogą wykazywać słabą odporność na losowy szum klasyfikacyjny, a zastosowanie ich w realnym świecie jest wielce wątpliwe. W artykule przedstawiony został przykład, w którym stwierdzono, że jeżeli część danych treningowych będzie miała źle oznaczone klasy, to algorytm nie będzie mógł poprawnie sklasyfikować tych danych. Skutkiem czego stworzenie modelu z dokładnością większą niż $\frac{1}{2}$ będzie niemożliwe.
Istnieje dużo algorytmów korzystających z metody boosting. Najpopularniejsze z nich to AdaBoost, LPBoost, TotalBoost, BrownBoost, MadaBoost, LogitBoost.

1.3.3 Stacking

Metoda stacking (kontaminacja modeli) to połączenie kilku lub kilkunastu klasyfikatorów o różnych algorytmach. W pierwszym etapie, klasyfikatory trenowane są na tych samych danych. Następnie zbiór treningowy klasyfikowany jest przez te modele, a przewidziane klasy tworzą nowy zbiór uczący

dla meta-klasyfikatora. Zbiór treningowy można utworzyć także, z połączenia danych wejściowych z predykowanymi klasami. Zamiast klas, jako dodatkowe wejście, można użyć prawdopodobieństwa danej klasy, o ile wszystkie klasyfikatory wspierają tą funkcję. Meta-klasyfikatorem może być każdy algorytm. Bardzo często do tego celu stosuje się jednowarstwową regresję logistyczną lub sieć neuronową.

1.4 Wstępne przetwarzanie danych

1.4.1 Brakujące wartości atrybutów

Często występującym problemem jest niekompletność baz danych, brak kilku wartości różnych atrybutów. Brakujące wartości mogą być wynikiem błędu człowieka, aplikacji, programu pomiarowego, nie podania danych lub z innego powodu. Zazwyczaj brakujące dane oznaczone są pustymi polami, "?lub w inny opisany sposób. Istnieje kilka sposobów na rozwiązanie tego problemu.

Usunięcie niekompletnych obserwacji

Najprostszym sposobem jest usunięcie wierszy lub kolumn, w których brakuje wartości. Przed usunięciem, należy przeanalizować dane, sprawdzić, które usunięcie będzie najbardziej korzystne (usunie najmniej danych). Może zaistnieć sytuacja, że zamiast usuwać dużą ilość przykładów (wiersze), bardziej opłaca usunąć się atrybut (kolumnę), który ma dużo pustych komórek. Opisana metoda może również mieć niepożądane konsekwencje. Usuwając niektóre obserwacje lub atrybuty, pozbywa się części informacji. Skutkiem tego zabiegu model predykcyjny może działać słabiej. Konsekwencją stosowania tego sposobu jest brak możliwości predykcji niekompletnych przykładów.

Imputacja danych

Innym pomysłem na rozwiązanie tego problemu jest imputacja danych. Brakujące dane można obliczyć lub wyznaczyć różnymi technikami na podstawie wartości pozostałych obserwacji. Jeżeli atrybut zawiera wartości ciągłe, brakujące elementy można zastąpić wartością średnią lub medianą całej kolumny. W przypadku wartości dyskretnych można uzupełnić je wartością występującą najczęściej. Stosując takie rozwiązanie można wprowadzić szum do danych. Dającym lepsze rezultaty rozwiązaniem, może być zastosowanie klasyfikatora lub regresji w celu imputacji danych.

1.4.2 Transformacja danych

Drzewo decyzyjne lub las losowy są jednymi z nielicznych algorytmów klasyfikacji, które nie wymagają skalowania danych. Atrybuty mogą mieć różne skale, jednostki i przedziały zmienności. Może to powodować dominację niektórych atrybutów nad innymi (np. w klasyfikatorze k-NN, podczas mierzenia odległości euklidesowej), a w konsekwencji do zafałszowania klasyfikacji. Ten problem można rozwiązać na kilka sposobów.

Skalowanie

Skalowanie polega na proporcjonalnym przekształceniu wartości atrybutów do nowego przedziału. Zazwyczaj jest to przedział [0,1]. Do przekształcenia wykorzystuje się następujący wzór:

$$x_i = p_{start} + (p_{stop} - p_{start})\frac{x_i - x_{min}}{x_{max} - x_{min}}$$

gdzie

- x_i to nowa wartość atrybutu,
- x_{max} i x_{min} to wartość maksymalna i minimalna atrybutu,
- p_{start} i p_{stop} to granice przedziału docelowego.

Standaryzacja

Lepszym rozwiązaniem może być standaryzacja, z powodu uwarunkowania wielu algorytmów, które rozpoczynają klasyfikację od wag równych 0 lub bliskich 0. Wykorzystując standaryzację, dane są podobne do standardowego rozkładu normalnego, a średnie wartości atrybutów ustawiane są w zerze, co pozwala na łatwiejsze uczenie się wag. Zastosowanie standaryzacji czyni klasyfikator bardziej odpornym na przykłady poboczne (tzw. outliers) w przeciwieństwie do skalowania. Standaryzację wyraża się wzorem:

$$x_i^{STD} = \frac{x_i - \mu(x)}{\sigma(x)}$$

gdzie:

- x_i^{STD} to nowa wartość atrybutu,
- x_i to wartość początkowa,
- $\mu(x)$ to wartość średnia,
- $\sigma(x)$ to odchylenie standardowe.

, , a .

1.4.3 Dane kategoryczne

Bardzo często zdarza się, że bazy danych zawierają oprócz danych numerycznych, także dane kategoryczne, zapisane w postaci łańcucha znaków. Wyróżnia się dane kategoryczne nominalne i porządkowe. Dane kategoryczne nominalne zawierają cechy wzajemnie się wykluczające. Przykładem może być płeć człowieka, kolor skóry, oczu, kolor koszulki. Takich danych nie można posortować ani też porównać. Natomiast dane kategoryczne porządkowe zawierają informacje jednoznaczne identyfikujące, które można uporządkować (np. rozmiar koszulki).

Mapowanie danych kategorycznych

Aby mieć pewność, że algorytm klasyfikacji odpowiednio zinterpretuje dane kategoryczne, należy je przekonwertować do wartości numerycznych. W poniższym przykładzie (1.2) kategoria płeć, kraj są przykładem danych kategorycznych nominalnych, a wzrost jest atrybutem porządkowym. Należy pamiętać, aby w przypadku danych porządkowych, odpowiednio odwzorować zależności.
W procesie zmieniania wystarczy wykonać odpowiednie mapowanie. Atrybut płeć:

- mężczyzna -> 0
- kobieta ->1

Atrybut kraj:

- Polska -> 0
- Niemcy ->1

Atrybut wzrost:

- niski -> 0
- średni -> 1
- wysoki -> 2

Atrybut klasa:

- klasa1 -> 0
- klasa2 -> 1

ID	Płeć	Kraj	Wzrost	Wiek	Klasa
1	Mężczyzna	Polska	Niski	25	klasa 1
2	Mężczyzna	Polska	Wysoki	52	klasa 2
3	Kobieta	Polska	Średni	19	klasa 2
4	Kobieta	Niemcy	Niski	37	klasa 1

Tablica 1.2: Przykład z kategorycznymi danymi.

Dane nominalne

Dane kategoryczne nominalne nie mogą zostać w takiej postaci. Dane te nie posiadają określonego porządku, ani nie można ich porównać. Zostawiając je w takiej formie algorytm mógłby porównać kobietę z mężczyzną. Wynik klasyfikacji mógłby być prawidłowy, ale niekoniecznie najlepszy.

Dla takich danych, należy stworzyć dodatkowe atrybuty o wartościach binarnych (nazwa tej metody w języku angielskim to *one hot encdoing*). W przypadku atrybutu płeć, należy stworzyć dwie kolumny "mężczyzna"i "kobieta". Poniżej w tabeli 1.3 przedstawiono poprawnie zakodowane dane.

ID	M	K	Pol	Nie	Wzrost	Wiek	Klasa
1	1	0	1	0	0	25	0
2	1	0	1	0	2	52	1
3	0	1	1	0	1	19	1
4	0	1	0	1	0	37	0

Tablica 1.3: Przykład z kategorycznymi danymi po odpowiednim kodowaniu.

1.4.4 Redukcja ilości atrybutów

Jednym ze sposobów uniknięcia nadmiernego dopasowania klasyfikatora do danych lub zmniejszenia jego złożoności jest redukcja atrybutów. Zdarza się, że niektóre atrybuty mają znikomy wpływ na kategorię przykładu, a mogą stanowić szum klasyfikacyjny. Atrybuty mniej ważne usuwa się, a zostawia się tylko te z największą ilością informacji, mających największy wpływ na klasyfikację. Usunięcie bezużytecznych atrybutów może zwiększyć skuteczność klasyfikacji. Kolejną możliwością jest redukcja wymiarów, czyli

poszukiwanie wzorców danych, tak aby kilka atrybutów zgrupować w jeden. Poniżej przedstawiono kilka różnych algorytmów.

Sekwencyjna selekcja postępująca oraz sekwencyjna selekcja wsteczna

Sekwencyjna selekcja postępująca (ang. *sequential forward selection* (SFS)), algorytm zaczyna działanie od pustego zbioru atrybutów. W kolejnych iteracjach dodaje atrybut, który maksymalizuje wybraną funkcję (najczęściej jest to dokładność klasyfikacji). Innymi słowy, dodaje te atrybuty, które najbardziej podnoszą jakość klasyfikacji. Algorytm działa do momentu osiągnięcia zakładanej ilości atrybutów lub do pogorszenia skuteczności klasyfikacji. Algorytm ten posiada wadę, gdyż raz dodana cecha nie może już zostać usunięta. Może zdarzyć się taka sytuacja, że dodany już atrybut powoduje obniżenie jakości po dodaniu nowego atrybutu. Rozwiązaniem tego problemu jest algorytm SFFS (ang. *sequential floating forward selection*), który po dodaniu nowej cechy usuwa inny atrybut jeśli zwiększy to wartość funkcji.
Sekwencyjna selekcja wsteczna (ang. *sequential backward selection* (SBS)), algorytm usuwa kolejne atrybuty powodujące najmniejszy spadek w jakości klasyfikatora (mające najmniejszy wpływ) do momentu osiągnięcia pożądanej ilości atrybutów. Istnieje także rozszerzenie tego algorytmu w postaci SFBS (ang. *sequential floating backward selection*)). W tym algorytmie, po usunięciu atrybutu, w kolejnym kroku sprawdza się, czy usunięte wcześniej już atrybuty, po ponownym dodaniu nie podniosą wartości wybranej funkcji. Przedstawione algorytmy są algorytmami zachłannymi i bardzo czasochłonnymi. W pesymistycznym przypadku, klasyfikator może być budowany $\frac{p(p+1)}{2}$ razy.

Analiza głównych składowych

Analiza głównych składowych PCA (ang. *principal component analysis*) jest to nienadzorowana liniowa transformacja, używana między innymi do redukcji wymiarów danych. Bardzo często stosuje się ją w statystyce oraz analizie danych. PCA identyfikuje wzorce w danych pomiędzy atrybutami. Algorytm szuka maksymalnych kierunków wariancji wielowymiarowych danych i następnie rzutuje je w nowej podprzestrzeni z mniejszą lub taką samą ilością atrybutów.

1.5 Ocena poprawności klasyfikacji

1.5.1 Miary jakości klasyfikacji danych

Jakość klasyfikacji można ocenić na podstawie kilku współczynników. Do ich obliczenia wykorzystuje się macierz pomyłek (tabela 1.4). Tworzona jest ona w oparciu o wynik klasyfikacji. Dla klasyfikacji binarnej macierz składa się z dwóch kolumn oraz dwóch wierszy. W wierszach znajdują się poprawne klasy decyzyjne, natomiast w kolumnach przewidziane przez klasyfikator. Zaklasyfikowane obiekty, umieszcza się w odpowiedniej grupie.

		Klasa predykowana	
		pozytywna	negatywna
Klasa rzeczywista	pozytywna	prawdziwie pozytywna (TP)	fałszywie negatywna (FN)
	negatywna	fałszywie pozytywna (FP)	prawdziwie negatywna (TN)

Tablica 1.4: Macierz pomyłek

Nazwa grup inspirowana była nazewnictwem medycznym. Dla dwóch wyróżniamy następujące grupy:

- prawdziwie pozytywna (ang. *true positive*), skrót TP: są to obiekty należące klasy pozytywnej oraz zakwalifikowane przez klasyfikator jako pozytywne (trafienie, z ang. *hit*)
- fałszywie negatywna (ang. *false negative*), skrót FN: są to obiekty należące klasy pozytywnej, ale zostały błędnie zakwalifikowane przez klasyfikator jako negatywne (błąd pominięcia, z ang. *miss*)
- fałszywie pozytywna (ang. *false positive*), skrót FP: są to obiekty należące klasy negatywnej, błędnie uznane przez klasyfikator jako pozytywne (fałszywy alarm, ang. *false alarm*)
- prawdziwie negatywna (ang. *true negative*), skrót TN: są to obiekty należące klasy negatywnej, i sklasyfikowane przez klasyfikator jako negatywne (poprawnie odrzucone, ang. *correct rejection*)

Ocenę jakości klasyfikacji przeprowadza się w oparciu o współczynniki wyliczane na podstawie macierzy pomyłek.
Podstawowym kryterium służącym do oceny klasyfikacji jest dokładność (ang. *accuracy*), jest to stosunek wszystkich poprawnie sklasyfikowanych przykładów klasy pozytywnej oraz negatywnej do wszystkich przykładów. Miara ta określa dokładność z jaką klasyfikator podaje poprawny wynik.

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Można wyróżnić także błąd klasyfikatora, obliczany na podstawie dokładności.

$$Error\ rate = 1 - accuracy$$

Trzecim wskaźnikiem oceny klasyfikacji jest TPR (ang. *true positive rate*), często określany jako czułość (ang. *sensitivity* lub *recall*). Jest to stosunek obiektów poprawnie sklasyfikowanych jako pozytywne z wszystkimi pozytywnymi przykładami. Wskaźnik ten pokazuje poprawność klasyfikowania obserwacji pozytywnych. W medycynie, wykorzystując te miarę można określać skuteczność wykrywania osób chorych.

$$Sensitivity,\ Recall,\ TPR = \frac{TP}{TP + FN}$$

Kolejną miarą oceniającą klasyfikację jest TNR (ang. *true negative rate*), nazywana także specyficznością (ang. *specificity.*). Wskazuje ona efektywność klasyfikowania przykładów negatywnych. Jest to stosunek poprawnie przydzielonych przykładów negatywnych do wszystkich negatywnych obserwacji. Z jej pomocą, można ocenić celność klasyfikacji osób zdrowych.

$$Specificity,\ TNR = \frac{TN}{TN + FP}$$

Wyróżnia się także współczynnik FPR (ang. *false positive rate*), jest to iloraz przykładów fałszywie pozytywnych i sumy przykładów prawdziwie negatywnych i fałszywie negatywnych.

$$FPR = \frac{FP}{TN + FP}$$

Istotnym wskaźnikiem jest także precyzja (ang. *precision*). Określa ona jaka część przykładów uznanych za pozytywne przez klasyfikator została poprawnie oznaczona. Precyzja wyrażana jest jako stosunek prawdziwie pozytywnych przypadków do wszystkich przykładów uznanych za pozytywne. W medycynie, pokazuje procentowo ile osób uznanych za chorych, jest rzeczywiście chora.

$$precision = \frac{TP}{TP + FP}$$

Wskaźnik dokładności oraz error rate nie sprawdzają się w przypadku, gdy dane są niezrównoważone. Klasyfikator może osiągnąć wysoką dokładność np. 90% przy niskiej wykrywalności klasy mniejszościowej. Dlatego oceniając klasyfikator pracujący na niezrównoważonych danych, należy obliczyć osobno

wstawic przykład klasyfikacji dn o wysokiej skuteczności (np 95), ale o niskim wykrywaniu małej klasy

opis mierzenia w przypadku

współczynniki precyzji, czułości oraz specyficzności dla każdej kategorii danych. Jak wspomniano wcześniej, bardzo często polepszenie jakości klasyfikacji klasy mniejszościowej połączona jest z pogorszeniem rozpoznawalności klasy większościowej. Mając współczynnik czułości oraz specyficzności ciężko zdecydować, który klasyfikator jest lepszy. Kubat i Matwin zaproponowali połączenie obu tych współczynników, w postaci średniej geometrycznej czułości oraz specyficzności [1].

$$G - mean = \sqrt{precision * recall}$$

Klasyfikator z wyższym G-mean, zapewnia lepszą rozpoznawalność obu klas, jednocześnie zachowując, aby dokładność w rozpoznawaniu obu klas była zbilansowana. Współczynnik ten jest niezależny od rozkładu klas w danych [2].
Ocenę klasyfikacji danych niezrównoważonych możemy dokonać także przy pomocy F-measure. Jest to średnia harmoniczna precyzji oraz czułości. Współczynnik F-measure można obliczyć dla obu klas. β wykorzystywana jest do określenia zależności pomiędzy precyzja oraz czułością.

$$F - measure = \frac{(1+\beta)^2 * precision * recall}{\beta^2 * precision + recall}$$

Zazwyczaj $\beta = 1$, wtedy:

$$F - measure = 2 * \frac{precision * recall}{precision + recall}$$

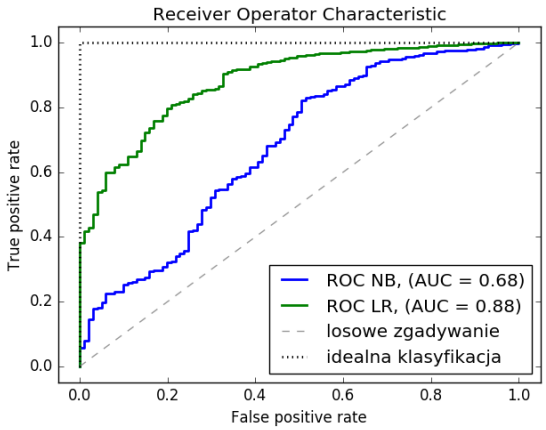
Podstawiając wzory pod precision oraz recall można otrzymać uproszczoną wersje miary F_1:

$$F_1 - measure = \frac{2 * TP}{2 * TP + FP + FN}$$

Krzywa ROC

Dla klasyfikatorów, które mogą zwracać prawdopodobieństwo klas, można zbudować wykres wartości TPR oraz FPR, które tworzą tzw. krzywą ROC (ang. *receiver operator characteristic*). Wykorzystując tą krzywą można porównać modele. Klasyfikator jest lepszy od drugiego, jeżeli jego krzywa jest powyżej drugiej krzywej. Jeżeli krzywa ROC, przebiega poniżej przekątnej, to klasyfikator ma gorszą skuteczność niż losowe zgadywanie. Na wykresie 1.1 przedstawiono porównanie naiwnego klasyfikatora bayesowskiego oraz regresji logistycznej, która osiągnęła zdecydowanie lepszy wynik w klasyfikacji.

Na wykresie zaznaczono krzywą ROC idealnego klasyfikatora, przechodzi ona przez punkty (1,0) oraz (1,1).
W celu porównania klasyfikatorów, można obliczyć pole powierzchni poniżej krzywej ROC, jest to tzw. AUC (ang. *area under curve*). Idealny klasyfikator osiąga wartość AUC = 1, natomiast losowe zgadywanie AUC = 0.5. Im wartość AUC jest wyższa, tym model jest skuteczniejszy.



Rysunek 1.1: Przykład krzywej ROC, dla naiwnego klasyfikatora bayesowskiego oraz dla regresji logistycznej dla danych ąbalone0_4_16_29

1.5.2 Nadmierne dopasowanie i wariancja klasyfikatora

napisac gdzies o nadmiernym dopasowaniu -> walidacja krzyżowa

cos o wariancji

Jeżeli model uzyskuje zdecydowanie lepsze wyniki na danych treningowych niż na danych testowych, świadczy to o nadmiernym dopasowaniu (ang. *overfitting*) klasyfikatora do danych uczących. Nadmierne dopasowanie występuje wtedy, gdy model zamiast dobrze generalizować prawdziwe dane, dopasowuje się za bardzo do danych treningowych. Powodem tego zjawiska jest zazwyczaj zbyt duża ilość parametrów (zbyt duża złożoność) modelu w stosunku do rozmiaru danych uczących. Klasyfikator może mieć wysoką skuteczność dla danych treningowych, jednak dla nowych danych będzie generował gorsze wyniki. Rozwiązaniem tego problemu może być:

- zebranie większej ilości danych uczących
- wprowadzanie kary (np. regularyzacja L1 lub L2) za zbyt dużą złożoność modelu
- wybranie prostszego algorytmu z mniejszą ilością parametrów

- zmniejszenie wymiaru danych (usunięcie niektórych atrybutów)

wstawic obrazek niedopasowania

Klasyfikator z nadmiernym dopasowaniem posiada wysoką wariancję. Wariancja mierzy zmienność przewidywań modelu dla określonego zbioru testowego, w przypadku użycia różnych zbiorów uczących (lub podzbiorów). Wysoka wariancja świadczy o dużej zmienności przewidywań klas w przypadku zmiany danych treningowych i jest to niepożądane zjawisko. Mała wariancja to mało zmian, w miarę stabilne przewidywanie klas dla tych samych próbek. Model z wysoką wariancją wrażliwy jest na losowość i szum danych.

Klasyfikator może być także niedouczony, niedopasowany dostatecznie (ang. *undefitting*), co oznacza, że model nie jest zbyt skomplikowany, aby znaleźć odpowiednie wzorce danych. W wyniku czego, osiąga słabe wyniki. Miarą towarzyszącą niedopasowaniu jest obciążenie (ang. *bias*). Klasyfikator testuje się na próbce danych, wielokrotnie budując model na tym samym zbiorze danych uczących i mierzy się jak przewidywania klas różnią się od właściwych. Wysokie obciążenie, oznacza niedotrenowanie oraz małą skuteczność klasyfikacji. Niskie obciążenie oznacza bardzo dobre przewidywania klas. Obciążenie to błąd systematyczny.

wstawic obrazek bias i wariancja

Idealny klasyfikator odnajduje wzorce w danych treningowych oraz dobrze je generalizuje, tak aby skutecznie klasyfikował nie widziane wcześniej próbki. Celem w klasyfikacji nadzorowanej, jest zbudowanie klasyfikatora z małym obciążeniem oraz niską wariancją. Niestety, zazwyczaj nie można osiągnąć obu celi. Modele parametryczne i liniowe częstą mają niskie obciążenie, a wysoką wariancję. Natomiast modele nieparametryczne i nieliniowe zazwyczaj mają duże obciążenie, ale niską wariancję. Obciążenie i wariancja są ze sobą połączone, zmniejszając jeden błąd, zwiększa się drugi. Tą zależność nazywa się przetargiem obciążenia i wariancji (ang. *bias-variance trade-off*). Zmieniając parametry klasyfikatorów, można wpływać na balans pomiędzy tymi błędami:

- w algorytmie kNN, który często ma małe obciążenie i wysoką wariancję, zwiększając liczbę sąsiadów k, można zmniejszyć wariancję, ale równocześnie zwiększyć błąd obciążenia,
- w klasyfikatorze SVM, zwiększając karę, parametr C można zwiększyć obciążenie, a zmniejszyć wariancję.

Aby wychwycić nadmierne dopasowanie lub niedotrenowanie należy zbiór danych podzielić na zbiór uczący oraz na zbiór testowy. W ten sposób można sprawdzić z jaką skutecznością klasyfikator będzie pracował na nowych danych. Istnieją różne metody pomiarowe, które zostały opisane w podrozdziale 1.5.3

1.5.3 Metody pomiaru jakości klasyfikacji danych

W celu prawidłowej oceny możliwości klasyfikatora należy zbiór danych podzielić na zbiór uczący oraz na zbiór testowy. Należy pamiętać, że zwiększając zbiór testowy, zmniejsza się zbiór uczący i tym samym mniej informacji jest stosowanych do tworzenia modelu. Może to prowadzić do zaniżenia ogólnej oceny klasyfikatora. Jednocześnie mały zbiór testowy, może być niewystarczający do prawidłowej oceny, która może być obarczona dużym błędem. Proces oceny należy rozpocząć od budowy modelu w oparciu o dane treningowe, a następnie wykonać klasyfikację testową wykorzystując do tego zbiór testowy. Następnie buduje się macierz pomyłek w o parciu o sklasyfikowane przypadki. Kolejnym krokiem jest obliczenie opisanych wcześniej współczynników (zob. podrozdział 1.5.1) w oparciu o tą macierz. Istnieją różne schematy postępowania, służące do oceny zbudowanego modelu.

napisac cos jeszcze o celu CV)

Metoda z jednym zbiorem

Do budowy klasyfikatora wykorzystywany jest cały zbiór dostępnych danych. W procesie testowania, bierze udział także cały zbiór danych. Metoda ta, nie jest zbyt wartościowa i prowadzi do zawyżenia jakości klasyfikatora. W przypadku nowych danych, taki model osiągnie gorsze wyniki niż wskazywałyby na to obliczone współczynniki.

Metoda z wydzielonym zbiorem testowym (ang. *the holdout method*)

W tej metodzie, zbiór danych dzielony jest w sposób losowy na dwie części. Użytkownik dobiera rozmiar zbioru uczącego (np. 80%) oraz zbioru testowego (np. 20%). Wadą takiego rozwiązania, jest losowy rozkład klasw zbiorze testowym oraz zmniejszenie zbiór uczącego. Może to doprowadzić do sytuacji nadmiernego dopasowania (zawyżonych wyników) lub do niedoszacowania klasyfikatora. Ważne jest, aby nie używać ciągle tego samego zbioru testowego do wyboru modeli, ale dokonywać losowania przed każdą oceną. Ulepszeniem tej metody, może być równy rozkład klas w obu zbiorach, tak aby zostały zachowane proporcje z oryginalnego zbioru.

Sprawdzian krzyżowy z p przykładami (ang. *leave-p-out cross-validation*)

Sprawdzian krzyżowy z p przykładami wykorzystuje p obserwacji jako zbiór testowy, pozostałe elementy tworzą zbiór uczący. Cały proces jest powtarzany do momentu stworzenia i przetestowania wszystkich możliwych

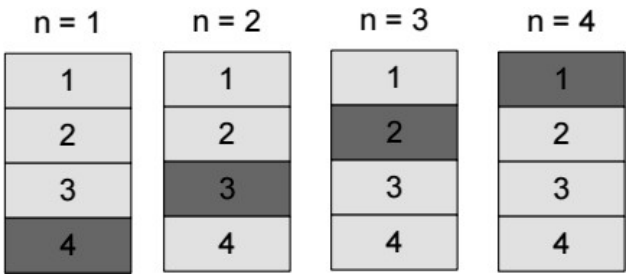
kombinacji p przykładów ze zbioru n. Ten rodzaj metody wymaga uczenia i testowania klasyfikatora $\binom{n}{p}$ razy, gdzie n to liczebność całego zbioru danych. W przypadku dużego zbioru danych oraz p>1, obliczenia mogą być czasochłonne, a nawet ze względu na duża ilość kombinacji, obliczenie ich może być niemożliwe.

Sprawdzian krzyżowy minus jeden element (ang. *leave-one-out cross-validation*)

Jest to specjalny przypadek sprawdzianu krzyżowego z p przykładami, dla p = 1. W tej metodzie zbiór testowy tworzy jeden element, pozostałe tworzą zbiór uczący. Testowanie klasyfikatora trwa do momentu użycia wszystkich obserwacji jako zbioru testowego. W przeciwieństwie do poprzedniej metody, ta jest wolna od czasochłonnych obliczeń, gdyż $\binom{n}{1}$=n, gdzie n to liczba wszystkich obserwacji. Zazwyczaj ta metoda wykorzystywana jest tylko do małych zbiorów danych.

Sprawdzian krzyżowy k-krotny (ang. *k-fold cross-validation*)

Zbiór danych jest losowo dzielony na k równych podzbiorów. Następnie każdy z podzbiorów w kolejnych k iteracjach staje się kolejno zbiorem testowym, pozostałe zbiory tworzą zbiór uczący, na podstawie, którego buduje się model. Klasyfikacja i testowanie wykonywane są k-krotnie. Otrzymane wyniki łączy się i uśrednia w celu uzyskania jednego wyniku. Zaletą tej metody jest mały błąd estymacji oraz niższa wariancja błędu niż w przypadku metody minus jednego elementu. Zwykle stosuje się k=3..10, dla których koszt czasowy jest umiarkowany.



Rysunek 1.2: Przykład sprawdzianu krzyżowego k-krotnego, k=4.

Równomierny sprawdzian krzyżowy k-krotny (ang. *Stratified k-fold cross-validation*)

Jest to specjalny przypadek sprawdzianu krzyżowego k-krotnego. Podzbiory tworzone są z zachowaniem proporcji wszystkich klas. Każdy podzbiór powinien zawierać w przybliżeniu podobny procent obserwacji z każdej kategorii.

napisac o tym, że niektóre zbiory są multiklasowe, natomiast dane sprowadzone są do 2 klas

napisac o podejsci one vs all

Rozdział 2

Klasyfikacja danych niezrównoważonych

wspomnieć o 2 podejścia -> preprecesing oraz algorytmy (algorytmamie się nie zajmuje)

Większość istniejących algorytmów klasyfikacji, nastawiona jest na poprawną klasyfikację zbiorów o zrównoważonej liczebności wszystkich klas. Niestety w rzeczywistych problemach, bardzo często zdarza się, że zbiory są mocno niezbilansowane.

2.1 Dane niezrównoważone

dodac odnosnik do one vs all

Dane są niezrównoważone (ang. *imbalanced data* jeśli klasy decyzyjne nie są przybliżeniu tak samo liczebne. Najmniejsza klasa, nazywana jest klasą mniejszościową (ang. *minority class*), natomiast klasa dominująca, lub pozostałe połączone klasy (można połączyć pozostałe klasy w jedną, doprowadzając do klasyfikacji binarnej, one vs all), nazywana jest klasą większościową (ang. *majority class*). W praktyce klasa mniejszościowa, zazwyczaj liczy około 10-20% wszystkich przykładów. Często zdarzają się jednak takie problemy, gdzie to zróżnicowanie jest większe np.:

- około 2% transakcji kartami kredytowymi w GOCARDLESS to oszustwa [3].

dodac przyklady danych mniejszosciowych

W przytoczonych przykładach ważniejsza jest klasa mniejszościowa i wykrycie jej stanowi priorytet. Niezrównoważenie klas w zbiorze danych stanowi problem w fazie uczenia i znacząco obniża jakość klasyfikacji. Ze względu na częstość występowania klasy dominującej, klasyfikator preferuje tą klasę, dążąc do optymalizacji i obniżenia błędu error rate (1.5.1) nie biorąc pod uwagę rozłożenia klas w zbiorze. Klasyfikator może osiągnąć wysoką skuteczność

klasyfikacji np. 95% przy niskiej lub zerowej wykrywalności klasy mniejszościowej. Należy oczekiwać od klasyfikatora wysokiej skuteczności wykrywania klasy mniejszościowej, nawet kosztem pogorszenia rozpoznawania klasy większościowej. Poddając analizie sąsiedztwa przykłady z klasy zdominowanej, można wyróżnić przykłady bezpieczne i niebezpieczne w klasyfikacji:

- safe - przykład bezpieczny, w jego sąsiedztwie zdecydowana większość obserwacji jest z tej samej klasy,
- borderline - graniczny, przykład niebezpieczny, w jego sąsiedztwie ilość przykładów z obu klas jest podobna
- outlier - poboczny, przykład niebezpieczny, w jego sąsiedztwie większość obserwacji jest z klasy przeciwnej, dominującej,
- rare - rzadki, przykład niebezpieczny, w jego sąsiedztwie występują tylko przykłady z klasy przeciwnej, większościowej.

dodac obrazek danyych safe border itd.

2.2 !!preprocessing danych niezrównoważonych

podac wykresy przykłady danych niezrównoważonych

W celu zrównoważenia rozkładu danych niezbilansowanych wprowadzono różne metody usuwania przykładów klasy dominującej lub tworzenia sztucznych obserwacji klasy mniejszościowej. Poniżej zostaną omówione metody, które zostały użyte podczas badań.

2.2.1 Metody undersampling

Jest to cała rodzina różnych metod, które usuwają przykłady z klasy większościowej. **Losowe usuwanie** (ang. *random undersampling*), jak sama nazwa wskazuje losowo usuwa przykłady z klasy dominującej. Rozwiązanie to ma niestety wadę. Jeśli usunie się zbyt dużo przykładów danego przypadku, można pozbawić klasyfikator bardzo ważnej informacji.

Lepszym rozwiązaniem jest świadome usuwanie przykładów spełniających określone kryteria. Taką metodą jest **undersampling z "Tomek links"**. Parę punktów Tomek link, definiuje się jako dwa punkty należące do różnych klas, z odległością równą $d(E_i, E_j)$, jeśli nie istnieje inny punkt E_l, taki, że $d(E_i, E_l) < d(E_i, E_j)$ lub $d(E_j, E_l) < d(E_i, E_j)$. Punkty tworzące Tomek link to szum lub punkt graniczny. Po znalezieniu takich punktów, usuwa się przykład z klasy dominującej. Usunięcie takiej obserwacji, powoduje rozszerzenie granicy klasy mniejszościowej.

opisac reszte te ktore wykorzystam

2.2.2 Metody oversampling

Rozdział 3

Przeprowadzone badania

3.1 Projekt klasyfikatora

3.2 Opis platformy i w jaki sposób zrealizowano badania

3.2.1 Język python

Wszystkie badania i testy zostały napisane z wykorzystaniem języka python. Jest to język programowania interpretowany, wysokiego poziomu z dużą ilością dostępnych bibliotek. Python[9] posiada dynamiczne zarządzanie typami oraz automatyczne zarządzanie pamięcią. Wspiera kilka paradygmatów programowania, takich ja: obiektowy, imperatywny, funkcyjny i proceduralny. Został zaprojektowany z myślą o czytelności kodu oraz składnią pozwalającą napisać program z mniejszą ilością kodu niż w językach C++ lub Java. Implementacje języka python dostępna jest na wiele systemów operacyjnych. Często wykorzystywany jest jako język skryptowy. Python jest projektem typu Open Source.

W pracy wykorzystano język python w wersji 2.7.11. Język ten wybrano ze względu na łatwość pisania w nim kodu, szybką możliwość nauki oraz na szeroki wachlarz dostępnych bibliotek. Ważnym argumentem w wyborze były gotowe biblioteki z klasyfikatorami oraz do pracy z klasyfikacją danych. Dostępność bibliotek do wizualizacji dla tego języka, pozwoliła na przedstawienie wyników testów w formie graficznej. Napisane testy, można w łatwy sposób rozbudować, zmodyfikować lub dodać nowe elementy.

3.2.2 Biblioteka scikit-learn

Scikit-learn[10] to proste i wydajne narzędzie do analizy i eksploracji danych. Jest to biblioteka uczenia maszynowego dla języka python. Rozpowszechnianie oparta jest na licencji BSD. W Scikt-learn zaimplementowane są (lub napisany jest kod obsługujący) różne algorytmy klasyfikacji, regresji, analizy skupień takie jak: maszyna wektorów nośnych, algorytmy najbliższego sąsiada, naiwny Bayes, drzewa decyzyjne, sieć neuronowa, zespoły klasyfikatorów. Z wykorzystaniem tej biblioteki można przygotować oraz przetworzyć odpowiednio dane. Możliwe jest także, ocenianie i wizualizacja wyników.

W testach użyto biblioteki scikit-learn w wersji 0.18.1. Wykorzystano z niej algorytmy klasyfikacji oraz wstępnego przetwarzania danych.

3.2.3 Biblioteka imbalanced-learn

Biblioteka imbalanced-learn[11] zawiera zestaw narzędzi do wstępnego przetwarzania danych niezrównoważonych. Posiada ona zaimplementowane różne metody under- oraz over-sampling do równoważenia zbiorów danych. W pracy wykorzystano te metody z biblioteki w wersji 0.2.1.

3.2.4 Pozostałe użyte biblioteki

Mlxtend

Mlxtend (machine learning extensions)[12] jest to biblioteka zawierająca różne narzędzia do pracy z danymi. W badaniach wykorzystano z niej algorytm Stacking.

Numpy

Numpy to pakiet umożliwiający obliczenia naukowe. Szczególnym elementem jest możliwość wykonywania obliczeń na tablicach N-wymiarowych.

Maptolib

Maptolib to biblioteka pythona, która tworzy różnego rodzaju wykresy 2D oraz interaktywne na różnych platformach.

Texttable

Texttable to prosty moduł napisany w języku python, służący do produkcji prostych tabel ASCII. Został wykorzystany do prezentacji wyników w

konsoli.

Pylatex

Pylatex to biblioteka pythona, służąca do tworzenia i kompilacji plików LaTeX. W pracy została wykorzystana do zapisu wyników w badań w plikach .tex oraz .pdf.

3.2.5 opisac co zaimplementowane?

napisac o roznych F i o tym jak wyniki wyglada, pokazać test

ze wlasna krosswalidacja, ze wlasne miary, klasyfikator. itd

inline

3.3 Opis danych użytych w badaniach

Do przeprowadzenia badań użyto 26 różnych prawdziwych zbiorów danych (tabela 3.1) pochodzących z repozytorium serwisu "The UCI Machine Learning Repository"[6]. Dane wybrano ze względu na różnorodność typów danych, ilości rekordów, atrybutów oraz zróżnicowanie rozkładu klas. Większość z danych była używana w publikacjach podobnych tematycznie[7][8].

Wszystkie dane zostały zapisane w skrypcie, w folderze $praca/data/files$, a opis szczegółowy danych znajduje się w folderze $praca/data/files/data\_descrytpion$. Do importu danych służą funkcje z pliku $praca/data/import\_data.py$. Do ogólnego importu danych z pliku, służy funkcja $importfile$, zaś wczytywanie danych użytych w projekcie odbywa się poprzez funkcje zaczynające się od $load\_$. Import danych z pliku odbywa się z wykorzystaniem funkcji z pakietu $numpy$ $genfromtext$ oraz $load\_txt$. Atrybuty posiadające dane kategoryczne zapisane w postaci łańcuchów znaków zostały zamienione na dane numeryczne. Cechy nominalne zostały zakodowane metodą $onehotencoding$. Dla danych zawierających więcej niż dwie klasy, klasa z najmniejszą liczebnością została wybrana jako klasa mniejszościowa, pozostałe klasy utworzyły klasę większościową. We wszystkich zbiorach danych, kategorie reprezentowane są w systemie binarnym. 5 zbiorów danych posiadało brakujące wartości. Zostały one zastąpione wartościami środkowymi zbioru (medianą).

Nazwa danych	L. el.	Atrybuty	Rozklad klas	% kl. mn.	IR
abalone0_4	4177	8	4103/74	1.77	55.45
abalone041629	4177	8	3842/335	8.02	11.47
abalone16_29	4177	8	3916/261	6.25	15.0
balance_scale	625	4	576/49	7.84	11.76
breast_cancer	286	9	201/85	29.72	2.36
bupa	341	6	200/141	41.35	1.42
car	1728	6	1663/65	3.76	25.58
cmc	1473	9	1140/333	22.61	3.42
ecoli	336	7	301/35	10.42	8.6
german	1000	24	700/300	30.0	2.33
glass	214	9	197/17	7.94	11.59
haberman	306	3	225/81	26.47	2.78
heart_cleveland	303	13	268/35	11.55	7.66
hepatitis	155	19	123/32	20.65	3.84
horse_colic	368	22	232/136	36.96	1.71
ionosphere	351	34	225/126	35.9	1.79
new_thyroid	215	5	185/30	13.95	6.17
postoperative	90	8	66/24	26.67	2.75
seeds	210	7	140/70	33.33	2.0
solar_flare	1066	10	1023/43	4.03	23.79
transfusion	748	4	569/179	23.93	3.18
vehicle	846	18	647/199	23.52	3.25
vertebal	310	6	210/100	32.26	2.1
yeastME1	1484	8	1440/44	2.96	32.73
yeastME2	1484	8	1433/51	3.44	28.1
yeastME3	1484	8	1321/163	10.98	8.1

Tablica 3.1: Dane użyte w badaniach wraz z charakterystyką.

3.4 Sposób mierzenia w sprawdzianie krzyżowym

3.5 Ocena klasyfikatora w sprawdzianie krzyżowym k-krotnym.

W większości publikacji naukowych dotyczących klasyfikacji, ocena klasyfikatora mierzona jest z wykorzystaniem sprawdzianu krzyżowego (zwykle k=10) oraz przedstawionych wcześniej miar. Jednakże, w tych publikacjach nie został opisany sposób obliczania współczynników w czasie sprawdzianu krzyżowego. Wykorzystanie różnych sposobów, prowadzi do różnych wyników. Niektóre metody są mniej lub bardziej obciążone błędem. Różnice w wynikach, wynikające z przyjętej metody obliczeniowej, są szczególnie widoczne w sprawdzianie krzyżowym z losowym rozkładem danych oraz w klasyfikacji danych niezrównoważonych. Są dwie główne możliwości obliczania współczynników:

- obliczanie wartości współczynników dla każdej k-iteracji (klasyfikatora), a następnie obliczenie średniej z tych iteracji
- stworzenie jednej wspólnej macierzy pomyłek dla każdej k-iteracji, a następnie obliczenie wskaźników.

W przypadku drugiego sposobu, poszczególne elementy macierzy pomyłek będą wynosić odpowiednio:

$$TP := \sum_{i=1}^{k} TP^{(i)}$$

$$FP := \sum_{i=1}^{k} FP^{(i)}$$

$$TN := \sum_{i=1}^{k} TN^{(i)}$$

$$FN := \sum_{i=1}^{k} FN^{(i)}$$

3.5.1 Test sposobów oceny klasyfikatora

W celu wyboru najlepszego sposobu oceny klasyfikatora, z najmniejszym błędem oraz wariancją wykonano pięć porównujących testów dla różnych metod obliczania miar. Wszystkie testy miały takie same założenia oraz sposób

wykonania. Testy wykonano na wygenerowanych losowo danych dla różnej ilości przykładów pozytywnych (od 1% do 10%). Jakość klasyfikacji była oceniana dla równomiernego sprawdzianu krzyżowego oraz dla losowego. Symulacje odbyły się one w następujący sposób:

1. Wygeneruj zbiór 1500 losowych próbek z 2 atrybutami, 2 klasami o rozkładzie 4:1
2. Wykonaj niezrównoważenie zbioru z ratio 0.1
3. Wybierz m=[1..10]*10 przykładów klasy mniejszościowej oraz 1000-m przykładów klasy dominującej
   (a) Wykonaj N iteracji:
      i. Wymieszaj dane
      ii. Wykonaj sprawdzian krzyżowy k-krotny, k=10
      iii. Oblicz współczynniki dla obu metod
   (b) Oblicz odchylenie standardowe oraz średnie wartości współczynników
4. Przedstaw wyniki odchylenia standardowego oraz średnie wartości współczynników dla różnego rozkładu klas.

Wykonanie testu N-krotnie (najlepiej n>100000) pozwala na obliczenie "prawdziwych"wartości miar oceny klasyfikacji. Powtórzenie sprawdzianu krzyżowego wielokrotnie pozwala na ocenę błędu oraz wariancji miar dla każdej metodyki. Przeprowadzenie testu dla danych zawierających tylko 1% obserwacji klasy mniejszościowej (przypadek ekstremalny, w zbiorze danych znajduje się wtedy tylko 10 takich przykładów) oznacza, że w niektórych iteracjach sprawdzianu krzyżowego nie będzie przykładów poprawnie sklasyfikowanych z tej klasy. Brak dobrze sklasyfikowanych przykładów klasy mniejszościowej może mieć także miejsce w losowym sprawdzianie krzyżowym. Wynika to z braku równomiernego rozkładu obu klas.

Dokładność oraz błąd klasyfikatora

Dokładność klasyfikacji oraz błąd klasyfikatora, korzystając z metody pierwszej, będzie wynosić:

$$accuracy_{avg} := \frac{1}{k} \sum_{i=1}^{k} accuracy^{(i)}$$

a błąd klasyfikatora:

$$error\ rate_{avg} = 1 - accuracy_{avg}$$

Obliczając drugim sposobem, korzysta się z podstawowego wzoru z wykorzystaniem wspólnej macierzy pomyłek.

W przypadku dokładności oraz błędu klasyfikatora, niezależnie od przyjętej metodyki otrzymane wartości będą takie same, nieobciążone błędem.

Czułość, specyficzność, FPR oraz precyzja

Czułość, specyficzność, FPR oraz precyzja w metodzie pierwszej obliczą się wg. wzorów:

$$Sensitivity_{avg},\ Recall_{avg},\ TPR_{avg} := \sum_{i=1}^{k} TPR_{avg}^{(i)}$$

$$Specificity_{avg},\ TNR_{avg} := \sum_{i=1}^{k} TNR_{avg}^{(i)}$$

$$FPR_{avg} := \sum_{i=1}^{k} FPR_{avg}^{(i)}$$

$$Precision_{avg} := \sum_{i=1}^{k} Precision_{avg}^{(i)}$$

W drugim sposobie korzysta się z wspólnej macierzy pomyłek oraz z podstawowych wzorów.

Testy wyżej wymienionych współczynników, zostały przeprowadzone z wykorzystaniem skryptu *test_wsk.py*. Zauważono, że w przypadku równomiernego sprawdzianu krzyżowego, różnica w wynikach jest bardzo mała, poniżej 0.5%. Obie metody obarczone sa małym błędem i wariancją. Natomiast w przypadku sprawdzianu krzyżowego z rozkładem losowym, metoda druga okazała się lepsza. Obliczona czułość oraz precyzja sposobem drugim, uzyskały wyniki z mniejszym błędem, bliższe wartości "prawdziwej". Natomiast specyficzność w obu metodach wyszła taka sama, ze względu na dużą ilość przykładów z tej klasy. Zostało sprawdzone, że w momencie odwrócenia liczebności klas, specyficzność posiada taką samą charakterystykę jak czułość. Różnice w wynikach obu sposobów, zmniejszają się wraz ze wzrostem przykładów klasy większościowej. Zazwyczaj przy 10% zawartości danych klasy zdominowanej w zbiorze, wyniki są takie same.

wstawic wykres i opisac

Miara F_1

W niniejszej pracy oraz w dużej ilości publikacjach miara F-measure obliczana jest dla $\beta = 1$, dlatego testy przeprowadzono tylko dla tej wartości.

Miarę F, można obliczyć na trzy różne sposoby. Pierwszy polega na obliczeniu F dla każdego k klasyfikatora, a następnie uśrednienie wyników:

$$F_{avg} := \frac{1}{k}\sum_{i=1}^{k} F_1^{(i)}$$

Drugi sposób, to obliczenie średniej czułości i precyzji, a następnie miary F z podstawowego wzoru:

$$Pre_{avg} := \frac{1}{k}\sum_{i=1}^{k} Pre^{(i)}$$

$$Re_{avg} := \frac{1}{k}\sum_{i=1}^{k} Re^{(i)}$$

$$F_{pre,re} = 2 * \frac{Pre_{avg} * Re_{avg}}{Pre_{avg} + Re_{avg}}$$

Ostatnio sposób, to obliczenie współczynnika F ze wspólnej, końcowej macierzy pomyłek:

$$F_{tp,fp,fn} = \frac{2 * TP}{2 * TP + FP + FN}$$

Do powyższych wzorów można dodać jeszcze sposób odrzucający oceny klasyfikatorów, dla których precyzja lub czułość są niezdefiniowane. Ta metoda została odrzucona ze względu na zawyżanie końcowej oceny.

Skrypt testujący powyższe trzy sposoby znajduje się w pliku $test\_f1.py$. Analizując otrzymane wyniki, zauważono, że najbardziej powtarzalne wyniki otrzymano korzystając z wzoru $F_{tp,fp,fn}$. W przypadku obu sprawdzianów krzyżowych, metoda ta generowała najmniejszy błąd. Zauważono, że niezależnie od ilości danych niezrównoważonych, otrzymane wyniki tą metodą różniły się nieznacznie, w przeciwieństwie do pozostałych sposobów. Podobnie jak w przypadku poprzednich miar, wraz ze wzrostem ilości danych niezrównoważonych, uzyskiwane wyniki były prawie takie same, niezależnie od sposobu obliczania.

Miara G-mean

Miara G-mean również może być obliczona na trzy różne sposoby. Pierwszy z nich to średnia z wszystkich klasyfikatorów:

$$G - mean_{avg} := \frac{1}{k}\sum_{i=1}^{k} G - mean^{(i)}$$

W drugim sposobie należy najpierw obliczyć średnią wartość czułości oraz specyficzności, a końcowy wynik G-mean oblicza się z głównego wzoru:

$$G - mean_{Se,Sp} = \sqrt{Sensitivity_{avg} * Specificity_{avg}}$$

W ostatniej metodzie obliczania G-mean, za czułość oraz specyficzność, wstawia się właściwie wzory, a wartość oblicza się na podstawie zsumowanej macierzy pomyłek.

$$G - mean_{tp,fp,fn} = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}}$$

Skrypt testujący powyższe wzory znajduje się w pliku *test_g_mean.py*. Analizując wyniki równomiernego sprawdzianu krzyżowego, zaobserwowano, że wyniki $G-mean_{tp,fp,fn}$ oraz $G-mean_{Se,Sp}$ pokrywają się. Natomiast w zwykłym sprawdzianie krzyżowym, pomiarem najmniej obarczonym błędem był $G - mean_{tp,fp,fn}$. Z wykorzystaniem tego wzoru, dla różnej zawartości klasy mniejszościowej w zbiorze danych otrzymano wyniki różniące się jedynie o kilka procent pomiędzy sobą, podczas gdy wyniki pozostałych metod różniły się aż o 15%-30%

Krzywa ROC i miara AUC

Wartość AUC w sprawdzianie krzyżowym można skonstruować na dwa sposoby. W pierwszej metodzie, konstruuje się krzywą ROC oraz oblicza się AUC dla każdego k klasyfikatora. Następnie oblicza się AUC_{AVG} poprzez obliczenie średniej:

$$AUC_{AVG} := \frac{1}{k} \sum_{i=1}^{k} AUC^{(i)}$$

W przypadku sprawdzianu krzyżowego z losowym rozkładem klas, może okazać się, że nie sklasyfikowano żadnego przykładu pozytywnego. Wtedy skonstruowanie krzywej ROC oraz obliczenie AUC będzie niemożliwe. W takich przypadkach podczas obliczania AUC_{AVG} można pominąć taki wynik.

Drugim sposobem jest połączenie prawdopodobieństwa przykładów testowych z każdej iteracji. Z połączonych obserwacji, konstruuje się jedną krzywą ROC i oblicza AUC. Korzystając z tego sposobu, zakłada się, że klasyfikator ma dobrze skalibrowane określanie prawdopodobieństwa. Test obu metod zostały przeprowadzone z użyciem skryptu *test_roc.py*.

Podsumowanie

Analizując przeprowadzone testy, najlepsze wyniki osiągnęły miary obliczone metodami opartymi na zsumowanej macierzy pomyłek oraz na łą-

czeniu wyników testów z każdej iteracji sprawdzianu krzyżowego. Obliczone w ten sposób współczynniki miały najbardziej stabilne wyniki, najmniejszy błąd oraz wariancję. Poniżej przedstawiono tabelę (3.2) z obliczonymi współczynnikami na różne sposoby. W dalszej części pracy, wszystkie miary w sprawdzianie krzyżowym, będą obliczane nad podstawie wspólnej macierzy pomyłek.

k-fold	**Pos**	**Neg**	**TP**	**FP**	**FN**	**TN**	**Se**	**Sp**	**Pre**	**G-Mean**	**F_1**
1	3	97	2	0	1	97	0,67	1,00	1,00	0,82	0,80
2	3	97	0	0	3	97	0,00	1,00	0,00	0,00	0,00
3	3	97	3	4	1	93	0,75	0,96	0,43	0,85	0,55
						AVG	**0,47**	**0,99**	**0,48**	**0,55**	**0,45**
						tp,fp,tn	**0,50**	**0,99**	**0,56**	**0,70**	**0,53**
										$\mathbf{G_{Se,Sp} = 0,68}$	$\mathbf{F_{Pre,Re} = 0,47}$

Tablica 3.2: Przykład obliczonych miar dla równomiernego sprawdzianu krzyżowego. Dla k=2, gdzie nie było pozytywnie sklasyfikowanych przykładów, wartości sensitivity, precision, F_1 zostały ustawione na 0, aby uniknąć dzielenia przez zero. W wierszu oznaczonym jako "tp,fp,tn", wskaźniki zostały obliczone na podstawie wspólnej macierzy pomyłek.

3.6 Sprawdzian krzyżowy, a oversampling

3.7 Porównanie klasyfikatorów

przetestsowa wszystkie klasyfkatory, z metodami pre i bez

Porównanie klasyfikatorów z domyślnymi ustawieniami, na takich samych typach danych. Jeżeli w tabelce występuje 0, to klasyfikator całkowicie błędnie klasyfikuje klasę mniejszościową.

	Drzewo	kNN	NKB	SVM	RForest	BAGGING	BOOSTING	STACKING
abalone0_4	0.98	0.99	0.96	0.98	0.99	0.96	0.96	0.99
abalone0_4_16_29	0.89	0.92	0.87	0.92	0.93	0.87	0.71	0.92
abalone16_29	0.9	0.93	0.7	0.94	0.94	0.71	0.66	0.94
balance_scale	0.86	0.91	0.92	0.92	0.91	0.92	0.92	0.92
breast_cancer	0.62	0.64	0.73	0.66	0.71	0.72	0.37	0.7
bupa	0.65	0.64	0.53	0.58	0.71	0.55	0.52	0.64
car	0.99	0.97	0.88	0.98	0.98	0.89	0.98	0.99
cmc	0.7	0.75	0.68	0.77	0.76	0.68	0.63	0.77
ecoli	0.89	0.9	0.77	0.9	0.92	0.78	0.93	0.9
german	0.68	0.69	0.73	0.72	0.77	0.71	0.68	0.73
glass	0.89	0.9	0.45	0.92	0.92	0.49	0.85	0.92
haberman	0.58	0.71	0.75	0.71	0.67	0.76	0.63	0.75
heart_cleveland	0.8	0.88	0.8	0.88	0.87	0.8	0.78	0.88
hepatitis	0.78	0.75	0.77	0.79	0.83	0.75	0.55	0.81
horse_colic	0.79	0.72	0.79	0.68	0.86	0.79	0.61	0.82
ionosphere	0.87	0.83	0.88	0.94	0.93	0.89	0.78	0.92
new_thyroid	0.97	0.97	0.97	0.86	0.98	0.97	0.97	0.97
postoperative	0.61	0.63	0.72	0.73	0.64	0.68	0.38	0.71
seeds	0.9	0.93	0.92	0.95	0.92	0.92	0.9	0.93
solar_flare	0.94	0.95	0.67	0.96	0.94	0.67	0.52	0.96
transfusion	0.62	0.67	0.74	0.7	0.66	0.74	0.64	0.72
vehicle	0.94	0.93	0.66	0.77	0.97	0.67	0.75	0.93
vertebal	0.79	0.82	0.78	0.68	0.82	0.77	0.71	0.81
yeastME1	0.98	0.98	0.66	0.97	0.98	0.7	0.92	0.98
yeastME2	0.94	0.96	0.17	0.97	0.97	0.17	0.1	0.96
yeastME3	0.93	0.95	0.32	0.89	0.95	0.29	0.89	0.94

Tablica 3.3: Dokładność algorytmów

	Drzewo	kNN	NKB	SVM	RForest	BAGGING	BOOSTING	STACKING
abalone0_4	0.43	0.58	0.97	0.0	0.51	0.97	0.42	0.42
abalone0_4_16_29	0.36	0.25	0.3	0.0	0.23	0.29	0.28	0.16
abalone16_29	0.29	0.15	0.59	0.0	0.13	0.57	0.45	0.08
balance_scale	0.08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
breast_cancer	0.41	0.2	0.45	0.07	0.38	0.44	0.6	0.24
bupa	0.6	0.46	0.77	0.0	0.52	0.78	0.48	0.45
car	0.92	0.62	1.0	0.77	0.71	1.0	0.82	0.92
cmc	0.34	0.29	0.6	0.15	0.28	0.6	0.35	0.15
ecoli	0.6	0.46	0.94	0.0	0.49	0.94	0.63	0.4
german	0.51	0.32	0.62	0.13	0.41	0.64	0.01	0.18
glass	0.41	0.12	0.76	0.0	0.06	0.71	0.12	0.0
haberman	0.32	0.35	0.21	0.01	0.25	0.25	0.2	0.17
heart_cleveland	0.23	0.0	0.57	0.0	0.0	0.51	0.31	0.0
hepatitis	0.53	0.06	0.72	0.0	0.5	0.69	0.56	0.31
horse_colic	0.73	0.57	0.76	0.25	0.76	0.76	0.56	0.64
ionosphere	0.8	0.56	0.74	0.84	0.87	0.76	0.73	0.83
new_thyroid	0.9	0.77	0.87	0.0	0.9	0.87	0.8	0.83
postoperative	0.21	0.0	0.12	0.0	0.0	0.12	0.62	0.0
seeds	0.89	0.94	0.94	0.94	0.9	0.94	0.9	0.91
solar_flare	0.14	0.0	0.93	0.0	0.07	0.93	0.56	0.0
transfusion	0.24	0.27	0.17	0.06	0.27	0.18	0.34	0.1
vehicle	0.85	0.88	0.83	0.01	0.95	0.83	0.74	0.81
vertebal	0.66	0.76	0.88	0.0	0.73	0.85	0.61	0.7
yeastME1	0.68	0.66	1.0	0.0	0.64	1.0	0.55	0.55
yeastME2	0.33	0.14	0.96	0.0	0.14	0.96	0.94	0.0
yeastME3	0.68	0.68	0.98	0.0	0.66	0.98	0.03	0.61

Tablica 3.4: Specyficzność algorytmów

	Drzewo	kNN	NKB	SVM	RForest	BAGGING	BOOSTING	STACKING
abalone0_4	0.66	0.76	0.97	0.0	0.71	0.97	0.64	0.65
abalone0_4_16_29	0.58	0.49	0.52	0.0	0.48	0.52	0.46	0.4
abalone16_29	0.52	0.38	0.65	0.0	0.36	0.64	0.55	0.28
balance_scale	0.27	0.0	0.0	0.0	0.0	0.0	0.0	0.0
breast_cancer	0.54	0.41	0.61	0.25	0.57	0.6	0.41	0.46
bupa	0.64	0.59	0.53	0.0	0.66	0.56	0.52	0.59
car	0.96	0.78	0.94	0.87	0.84	0.94	0.9	0.96
cmc	0.52	0.51	0.65	0.38	0.5	0.65	0.5	0.38
ecoli	0.75	0.66	0.84	0.0	0.69	0.85	0.78	0.62
german	0.62	0.52	0.69	0.36	0.62	0.69	0.1	0.42
glass	0.62	0.34	0.57	0.0	0.24	0.58	0.33	0.0
haberman	0.47	0.54	0.45	0.11	0.45	0.48	0.4	0.41
heart_cleveland	0.45	0.0	0.69	0.0	0.0	0.65	0.51	0.0
hepatitis	0.67	0.24	0.75	0.0	0.68	0.72	0.55	0.54
horse_colic	0.77	0.68	0.78	0.48	0.84	0.78	0.6	0.77
ionosphere	0.85	0.74	0.84	0.91	0.92	0.86	0.77	0.89
new_thyroid	0.94	0.88	0.93	0.0	0.94	0.93	0.89	0.91
postoperative	0.4	0.0	0.34	0.0	0.0	0.33	0.42	0.0
seeds	0.89	0.94	0.92	0.95	0.91	0.92	0.9	0.93
solar_flare	0.37	0.0	0.78	0.0	0.26	0.78	0.54	0.0
transfusion	0.42	0.46	0.4	0.22	0.46	0.41	0.5	0.3
vehicle	0.91	0.91	0.71	0.1	0.97	0.72	0.75	0.88
vertebal	0.75	0.8	0.8	0.0	0.8	0.79	0.68	0.77
yeastME1	0.82	0.81	0.81	0.0	0.8	0.83	0.71	0.74
yeastME2	0.57	0.37	0.36	0.0	0.37	0.37	0.26	0.0
yeastME3	0.81	0.82	0.49	0.0	0.81	0.45	0.17	0.77

Tablica 3.5: G-mean

	Drzewo	kNN	NKB	SVM	RForest	BAGGING	BOOSTING	STACKING
abalone0_4	0.48	0.62	0.45	0.0	0.58	0.45	0.29	0.53
abalone0_4_16_29	0.34	0.34	0.27	0.0	0.34	0.27	0.14	0.25
abalone16_29	0.27	0.22	0.2	0.0	0.2	0.2	0.14	0.14
balance_scale	0.08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
breast_cancer	0.39	0.25	0.49	0.11	0.44	0.48	0.36	0.32
bupa	0.58	0.51	0.58	0.0	0.6	0.59	0.45	0.51
car	0.86	0.61	0.39	0.76	0.72	0.4	0.79	0.85
cmc	0.34	0.35	0.46	0.23	0.34	0.46	0.3	0.23
ecoli	0.54	0.49	0.46	0.0	0.57	0.47	0.64	0.46
german	0.49	0.38	0.58	0.22	0.52	0.58	0.02	0.29
glass	0.37	0.16	0.18	0.0	0.11	0.18	0.11	0.0
haberman	0.29	0.39	0.31	0.02	0.28	0.35	0.22	0.27
heart_cleveland	0.21	0.0	0.39	0.0	0.0	0.37	0.24	0.0
hepatitis	0.5	0.1	0.56	0.0	0.55	0.53	0.34	0.41
horse_colic	0.71	0.6	0.73	0.37	0.8	0.73	0.51	0.72
ionosphere	0.81	0.7	0.82	0.91	0.91	0.83	0.7	0.88
new_thyroid	0.89	0.87	0.9	0.0	0.92	0.9	0.87	0.89
postoperative	0.22	0.0	0.19	0.0	0.0	0.17	0.35	0.0
seeds	0.85	0.9	0.89	0.92	0.88	0.89	0.85	0.9
solar_flare	0.15	0.0	0.19	0.0	0.09	0.18	0.09	0.0
transfusion	0.23	0.28	0.24	0.08	0.28	0.25	0.31	0.15
vehicle	0.87	0.86	0.54	0.02	0.94	0.54	0.58	0.84
vertebal	0.67	0.73	0.72	0.0	0.73	0.71	0.57	0.7
yeastME1	0.62	0.7	0.15	0.0	0.7	0.16	0.28	0.66
yeastME2	0.29	0.21	0.07	0.0	0.22	0.07	0.07	0.0
yeastME3	0.67	0.73	0.24	0.0	0.73	0.23	0.06	0.69

Tablica 3.6: F1

Bibliografia

[1] M. Kubat i S. Matwin. Addresing the curse of imbalanced training sets: one-side selection.

[2] H. He i E. A. Garcia. Learning from imbalanced data. IEEE Transactions on Data and Knowledge Engineering, 2009.

[3] N. Hockham: Machine learning with imbalanced data sets https://www.youtube.com/watch?v=X9MZtvvQDR4

[4] C. M. Bishop. Neural Networks for Pattern Recognition. Claredon press. Oxford, 1995.

[5] Long P, Servedio R. Random Classification Noise Defeats All Convex Potential Boosters

[6] The UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/

[7] F. Hu, X. Liu, J. Dai i H. Yu. A Novel Algorithm for Imbalance Data Classification Based on Neighborhood Hypergraph

[8] J. Stefanowski. Dealing with Data Difficulty Factors while Learning from Imbalanced Data.

[9] Python Software Foundation. https://www.python.org/

[10] scikit-learn Machine Learning in Python. http://scikit-learn.org/

[11] imbalanced-learn. http://contrib.scikit-learn.org/imbalanced-learn/

[12] S. Raschka. Mlxtend (machine learning extensions), https://github.com/rasbt/mlxtend