Politechnika Poznańska
Wydział Informatyki
Instytut Informatyki

Praca dyplomowa magisterska

OPTYMALIZACJA LOSOWANIA PRZYKŁADÓW W DANYCH NIEZRÓWNOWAŻONYCH Z WYKORZYSTANIEM UCZENIA SIĘ Z KOSZTAMI

Cost-Optimal Sampling of Examples for Imbalanced Data

inż. Benedykt Jakub Jaworski

Promotor
dr hab. inż. Jerzy Stefanowski, prof. PP

Poznań, 2015 r.

Chciałbym podziękować drowi Georgowi Kremplowi z Uniwersytetu Ottona von Guerickego w Magdeburgu, bez którego praca ta nigdy by nie powstała, za pomoc przy jej tworzeniu i za bycie jej współopiekunem.

I would like to express my gratitude to Dr. Georg Krempl from Otto-von-Guericke University Magdeburg, without whom this thesis would never come to existence, for his help during its formation and being its second supervisor.

Streszczenie

W pracy opisano zaproponowaną przez M. Spiliopoulou, J. Stefanowskiego i G. Krempla metodę *COST*, służącą do wstępnego przetwarzania zbioru danych o dwóch niezrównoważonych liczebnie klasach. Metoda ta powiela przykłady ze zbioru wejściowego na podstawie kosztu popełnienia błędu typu FP w stosunku do kosztu błędu FN oraz probabilistycznej analizy danych. Przedstawiona została probabilistyczna interpretacja danych, propozycja metody uzyskiwania wynikowego zbioru danych oraz wykonana implementacja odrobinę zmodyfikowanej wersji opisanej metody. Przedstawiono również wyniki eksperymentów z klasyfikacją przetworzonych omówioną metodą danych, pokazujące, że dla odpowiednio dobranego kosztu wzmacnia ona klasę mniejszościową w zadanym zbiorze danych.

Abstract

A method of pre-processing datasets with two imbalanced classes proposed by M. Spiliopoulou, J. Stefanowski and G.Krempl, called *COST*, is described in the thesis. The method copies existing examples of input dataset in a way that depends on a ratio of FP to FN error costs and a probabilistic analysis of the data. The probabilistic interpretation of the data, suggested method of obtaining an output dataset and created working implementation of a slightly modified version of the method are presented. The thesis also shows results of experiments conducted with classifiers that learned from sets filtered by the implemented method. Those results show that for properly set cost the method strengthens the minority class.

Spis treści

Rozdział 1

Wstęp

1.1 Motywacja pracy

Od początku XXI wieku człowiekowi w wielu dziedzinach życia towarzyszą systemy automatycznego klasyfikowania danych, wykorzystujące algorytmy uczenia maszynowego. Programy komputerowe klasyfikują automatycznie wiadomości poczty elektronicznej jako przydatne lub nie (czyli tzw. *spam*), dzielą klientów banków na wypłacalnych i niewypłacalnych, rozpoznają kody pocztowe zapisane odręcznie na wysyłanych pocztą przesyłkach, pomagają lekarzom w diagnozie pacjentów.

Dane, za pomocą których uczy się komputerowe klasyfikatory, często składają się z nierównej liczby przykładów z różnych klas, co oznacza, że przykładów należących do niektórych klas jest dużo mniej niż przykładów z pozostałych i te pierwsze są silnie niedoreprezentowane w zbiorze uczącym. Dane takie zazwyczaj nazywane są *danymi niezrównoważonymi* (ang. *class imbalanced data*), a ich niezrównoważenie powoduje pewne problemy z poprawnym nauczeniem klasyfikatorów koncepcji niedoreprezentowanej klasy.

Dodatkowo koszty pomyłek mogą być inne dla różnych klas decyzyjnych – przykładowo zaklasyfikowanie przykładu z klasy A do klasy B może spowodować dwukrotnie większy koszt niż zaklasyfikowanie przykładu z klasy B do klasy A. W takim wypadku zazwyczaj bardziej opłaca się klasyfikować wszystkie wątpliwe przypadki do klasy A.

W codziennym życiu z różnymi kosztami pomyłek można spotkać się choćby przy diagnozie pacjentów w szpitalach: zaklasyfikowanie zdrowego pacjenta jako chorego może spowodować wydatki szpitala na dodatkową diagnostykę i ewentualne zbędne leczenie, ale błąd w drugą stronę – zaklasyfikowanie chorego jako osobę zdrową – może być dużo kosztowniejsze i spowodować śmierć człowieka w wyniku niepodjęcia koniecznego leczenia.

Szukając rozwiązania wspomnianych wyżej problemów, próbuje się tworzyć algorytmy, które podczas uczenia biorą pod uwagę nierówne koszty i niezrównoważenie liczności klas. Oprócz tego proponuje się algorytmy wstępnego przetwarzania zbioru uczącego, które próbują wyrównać liczbę przykładów w klasach, by klasyczne klasyfikatory lepiej radziły sobie z klasą mniejszościową. Wybrane algorytmy przetwarzania danych zostały opisane w rozdziale 2.

To drugie podejście do problemu daje bardzo obiecujące wyniki, zbiory przetworzone zazwyczaj rzeczywiście prowadzą do lepszego rozpoznawania klasy mniejszościowej

i zmniejszania kosztów popełnianych pomyłek, jednak algorytmy te zazwyczaj są bardzo heurystyczne i dokładnemu mechanizmowi wyboru przykładów do dodania lub usunięcia ze zbioru uczącego brakuje często teoretycznych podstaw.

Motywacją powstania tej pracy była chęć przyczynienia się do zmiany tej sytuacji poprzez implementację i analizę nowego, bardziej metodycznego i mniej heurystycznego, podejścia do przetwarzania wstępnego danych. Wykorzystany został algorytm *COST*, zaproponowany przez M. Spiliopoulou, J. Stefanowskiego i G. Krempla w [SSK], oparty o interpretację danych wykorzystaną w algorytmie *(O)PAL*, opisanym w [KKS14] i [KKL15].

1.2 Cel i zakres pracy

Celem pracy było stworzenie działającej implementacji *COST* (ang. *Cost-Optimized Sampling Technique*) – algorytmu nadlosowania przykładów w zbiorze uczącym o dwóch niezrównoważonych liczebnie klasach decyzyjnych, zaprezentowanego w [SSK], wykorzystującego analizę danych przedstawioną w [KKS14] – oraz przeprowadzenia analizy jego działania, zawierającej porównanie wpływu na trafność klasyfikacji z innymi metodami przetwarzania wstępnego.

Realizacja tego celu wymagała:

- zapoznania się z literaturą dotyczącą problematyki danych niezrównoważonych liczebnie,
- zapoznania się z proponowaną na potrzeby algorytmu *COST* probabilistyczną interpretacją danych,
- implementacji algorytmu *COST* w języku *Java* jako filtru pakietu uczenia maszynowego *Weka*,
- analizy poprawności działania wykonanej implementacji,
- analizy charakteru zmian wprowadzanych przez wykonaną implementację do zbioru uczącego i wpływu parametrów algorytmu na jego działanie,
- wyboru rzeczywistych zbiorów danych do późniejszych eksperymentów i przeprowadzenia filtrowania ich za pomocą wykonanej implementacji,
- wykonania eksperymentu, polegającego na ocenie klasyfikacji na wybranych zbiorach danych przed filtrowaniem i po nim, oraz porównania wyników z trafnością klasyfikatorów uczonych na zbiorach zmodyfikowanych za pomocą innych istniejących algorytmów,
- analizy i podsumowania wyników eksperymentu.

Rozdział 2

Problematyka danych liczebnie niezrównoważonych i uczenia się z kosztami

2.1 Natura problemu danych niezrównoważonych

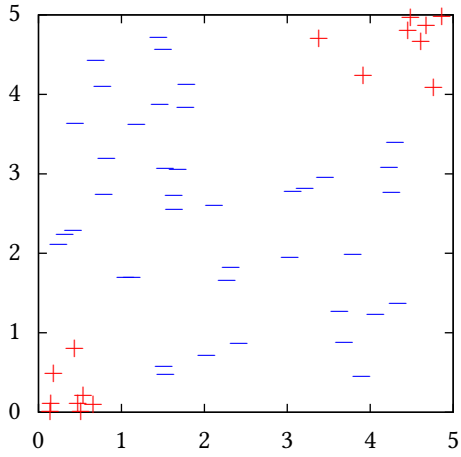
Jedną z trudności występujących przy zagadnieniu klasyfikacji w uczeniu maszynowym są dane o niezrównoważonej liczności przykładów w różnych klasach (ang. *imbalanced data*). Problem ten pojawia się w wielu dziedzinach, m.in. w danych medycznych – wśród danych nt. porodów liczba dzieci urodzonych przedwcześnie jest niewielką częścią wszystkich noworodków [Nap12], innym medycznym przykładem jest powszechnie stosowany w uczeniu maszynowym zbiór „Mammography Data Set" [HG09], będący kolekcją wyników badań mammografem, w którym mniejszość wskazuje przykłady „pozytywne" (tj. przypadki wystąpienia raka piersi), a przykładów „negatywnych" (tj. wyników osób zdrowych) jest wyraźnie więcej.

Dane niezrównoważone liczbowo pojawiają się również w innych dziedzinach, m.in. przy wykrywaniu oszustw w transakcjach przeprowadzanych za pomocą kart kredytowych [CS98] (liczba uczciwych transakcji przeważa), rozpoznawanie wycieków ropy na morzu na podstawie zdjęć satelitarnych (zdjęć z obrazem podobnym do wycieku jest dużo więcej niż przedstawiających rzeczywisty wyciek) [KHM98], czy w marketingu, gdzie celem jest określenie potencjalnych chętnych do kupienia produktu, a liczba osób, które się nim zainteresowały w skutek kampanii reklamowej to jedynie ok. $1\,\%$ tych, do których reklama dotarła [LL98].

Problem liczebnego niezrównoważenia reprezentacji klas może dotyczyć danych o dowolnej liczbie klas decyzyjnych, jednak tradycyjnie rozważa się uproszczone dane dwuklasowe, o niedoreprezentowanej klasie mniejszościowej (zwanej też często pozytywną) i nadreprezentowanej klasie większościowej (zwanej często negatywną). Również w tej pracy prowadzona będzie analiza takiego przypadku o dwóch klasach.

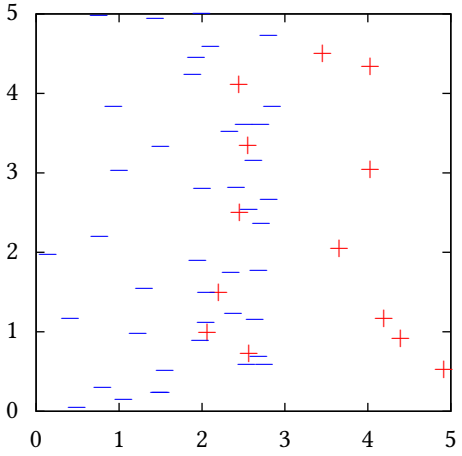
Większość współczesnych algorytmów uczenia maszynowego spodziewa się danych zrównoważonych liczbowo [HG09], [Mar00], przy ich tworzeniu zakładano (i zakłada się), że liczności przykładów uczących różnych klas są w przybliżeniu zrównoważone – z takim (choćby niejawnym) założeniem buduje się strategie uczenia klasyfikatora. W efekcie, przy niezrównoważonym zbiorze uczącym, klasyfikatory zostają w procesie uczenia mocno ukierunkowane na klasę większościową i rozpoznają ją później dużo skuteczniej niż mniejszościową w zbiorze testowym.

Gorsza trafność klasyfikowania jednak nie wynika bezpośrednio z samej różnicy liczebności przykładów z różnych klas, ale z tego, w jaki sposób przykłady te są w przestrzeni atrybutów rozmieszczone – w zbiorach można znaleźć obszary trudniejsze (w których trafność klasyfikowania jest niższa) i łatwiejsze (w których większość klasyfikatorów radzi sobie lepiej) [Jap01], [Nap12].

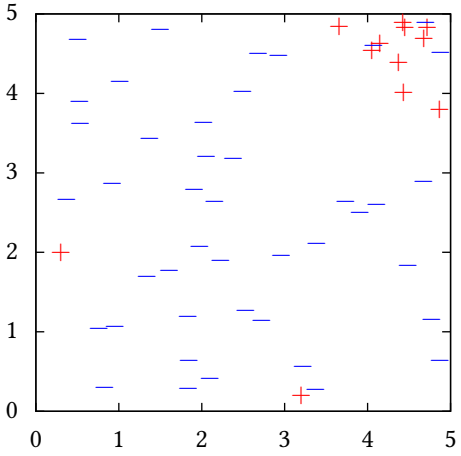


Rysunek 2.1: Dwuwymiarowy zbiór danych z separowalnymi obszarami obu klas

Wg [Nap12] globalny stopień niezrównoważenia nie ma wyraźnego wpływu na trafność klasyfikowania, jeśli klasy w zbiorze są wyraźnie odseparowane, tj. w zbiorze danych przykładów mniejszościowych jest wystarczająco dużo, by reprezentować obszary odpowiadające klasie mniejszościowej w przestrzeni atrybutów, i w obszarach tych nie występuje duży szum (ang. *noisy examples*) w postaci przykładów z klasy większościowej. Przykładowy prosty zbiór o dwóch atrybutach liczbowych z odseparowanymi obszarami klas pokazuje rys. 2.1.



Rysunek 2.2: Dwuwymiarowy zbiór danych z nachodzącymi na siebie obszarami obu klas



Rysunek 2.3: Dwuwymiarowy zbiór danych z szumem

Trudniejsze do klasyfikacji są zbiory, w których obszary z obu klas nachodzą na siebie (jak na rys. 2.2), lub występuje w nich szum (ang. *noisy examples*) w postaci kontr-

przykładów (podobnie opisanych przykładów, przypisanych do różnych klas), oraz obserwacji odstających (ang. *outliers*, przykładów znajdujących się daleko poza obszarami zdominowanymi przez ich klasę, zazwyczaj sąsiadujących z obszarami klasy przeciwnej) – taki zbiór przedstawia rys. 2.3. Współwystępowanie niezrównoważenia liczności i brak wyraźnego odseparowania klas daje w efekcie gorszą trafność klasyfikowania, przede wszystkim (lub wyłącznie) przykładów z klasy mniejszościowej.

Innym zjawiskiem związanym z danymi o niezrównoważonej liczności klas są zróżnicowane koszty błędów w klasyfikacji. Koszt błędnego zaklasyfikowania przykładu z klasy mniejszościowej do większościowej (tzw. błąd FN, ang. *false negative*) jest zazwyczaj dużo większy niż koszt błędnego zaklasyfikowania przykładu z klasy większościowej do mniejszościowej (tzw. błąd FP, ang. *false positive*).

W rozdziale 1 podano klasyfikację pacjentów w szpitalu jako przykład takiej sytuacji – w tym przypadku rzeczywiste koszty obu typów pomyłek w postaci liczbowej mogą być ciężkie do obliczenia lub oszacowania. Zdarzają się jednak przypadki, w których wartości kosztów są znane jawnie, na przykład klasyfikacja klientów banku wg ich zdolności kredytowej – bank zna dokładną kwotę pieniężną, którą straci, odrzucając dobrego klienta, oraz kwotę, którą straci, jeśli przyjmie złego.

Wspomniane zjawisko opisano dokładniej w [LS08].

2.2 Wybrane metody radzenia sobie z niezrównoważeniem danych

2.2.1 Przetwarzanie wstępne danych

Jak wspomniano we wstępie, jednym ze sposobów radzenia sobie z problemem niezrównoważenia liczebnego w danych jest wykorzystywanie metod przetwarzania wstępnego, które zmniejszają stopień niezrównoważenia liczności klas w zbiorze uczącym przed rozpoczęciem procesu uczenia klasyfikatora. Dzielą się one zazwyczaj na metody dogenerowania przykładów (ang. *oversampling*) i redukcji liczby przykładów (ang. *undersampling*). Te pierwsze dodają nowe przykłady klasy mniejszościowej do zbioru uczącego, drugie natomiast usuwają przykłady klasy większościowej.

Istnieją również metody hybrydowe, łączące oba podejścia.

Poniżej opisano pokrótce kilka wybranych metod z obu grup. Bardziej wyczerpujące opisy tych, oraz innych, algorytmów można znaleźć m.in. w [HG09], [Mac10].

Dogenerowanie przykładów

- **Random Oversampling** (nadlosowanie) – prosta metoda polegająca na wielokrotnym pseudolosowym wyborze przykładu mniejszościowego i skopiowaniu go. Parametrem metody jest stopień niezrównoważenia, do jakiego ma ona nadlosowywać przykłady, zazwyczaj nadlosowuje się je do zupełnego zrównoważenia danych. Metoda podczas nadlosowania nie bierze pod uwagę charakteru danych czy sąsiedztwa kopiowanego przykładu.

- **SMOTE** (ang. *Synthetic Minority Over-sampling Technique*, technika nadlosowania syntetycznych przykładów mniejszościowych) – metoda zaproponowana w [Cha+02]. Wybiera ona losowo przykład z klasy mniejszościowej, następnie lo-

sowo wybiera inny przykład należący do k najbliższych sąsiadów pierwszego przykładu, należących do tej samej klasy, a następnie tworzy nowy, *syntetyczny*, tj. nieistniejący w oryginalnych danych, przykład pomiędzy dwoma wybranymi wcześniej przykładami.

Nowo utworzony przykład dostaje wartości atrybutów losowane w przedziale określonym przez wartości obu wybranych wcześniej przykładów w przypadku atrybutów numerycznych, lub wartość większości przykładów z sąsiedztwa przykładu pierwszego w przypadku atrybutów nominalnych.

Podobnie jak w przypadku *Random Oversamplingu*, metodzie tej zadaje się pożądany do osiągnięcia stopień niezrównoważenia (lub, jak jest w przypadku implementacji w pakiecie *Weka* [Wekc], procent liczby przykładów klasy mniejszościowej do nadlosowania).

Ze względu na dużą popularność tej metody, powstało do niej wiele modyfikacji, biorących pod uwagę charakter danych, m.in. *Borderline-SMOTE* [HWM05], *Safe-Level-SMOTE* [BSL09], *Local-Neighbourhood SMOTE* [MS11].

- **SPIDER** (z ang. *Selective Pre-processing of Imbalanced Data*, selektywne przetwarzanie wstępne niezrównoważonych danych) – metoda hybrydowa, zaproponowana w [SW07] i [SW08]. Polega na wyborze ze zbioru uczącego przykładów stanowiących szum, a następnie usuwaniu ich, jeśli należą do klasy większościowej, lub powielaniu, gdy należą do klasy mniejszościowej. Algorytm ten dodatkowo pozwala zmienić etykietę klasy przykładom z klasy większościowej na mniejszościową.

  Wybór przykładów stanowiących szum polega na próbie zaklasyfikowania każdego przykładu algorytmem k-NN. Gdy przykład zostanie zaklasyfikowany prawidłowo, jest oznaczany jako bezpieczny i nie podlega dalszym modyfikacjom. Gdy przykład zostanie zaklasyfikowany błędnie, jest oznaczany jako szum.

  Jeśli metoda działa w trybie *wzmocnienia ze zmianą etykiety*, dla każdego przykładu klasy mniejszościowej, stanowiącego szum, wszystkie przykłady klasy większościowej stanowiące szum podlegają zmianie etykiety klasy na klasę mniejszościową, następnie przykłady mniejszościowe stanowiące szum zostają wzmocnione poprzez tylokrotne kopiowanie, ile jest bezpiecznych przykładów większościowych w ich sąsiedztwie. Następnie wszystkie pozostałe przykłady większościowe stanowiące szum zostają usunięte.

  W trybach *słabego* i *mocnego wzmocnienia* metoda jedynie wzmacnia przykłady mniejszościowe oznaczone jako szum, poprzez kopiowanie ich tyle razy, ile jest bezpiecznych przykładów większościowych w ich sąsiedztwie, przy czym *silne wzmocnienie* później powtarza proces dla większego sąsiedztwa. Na końcu przykłady większościowe stanowiące szum zostają usunięte.

  W [NSW10] opisano modyfikację tej metody, nazwaną *SPIDER2*.

Redukcja liczby przykładów

- **Random Undersampling** (odlosowanie) – metoda podobna do *Random Oversamplingu*. Wybiera ona losowo przykłady z klasy większościowej i usuwa je. Robi to

tak długo, aż zostanie osiągnięty zadany stopień niezrównoważenia, zazwyczaj robi się to do pełnego zrównoważenia danych.

- **NCR** (ang. *Neighbourhood Cleaning Rule*, reguła oczyszczania sąsiedztwa) – kolejna metoda, zaproponowana w [Lau01], wykorzystująca do oceny i wyboru przykładów algorytm *k*-NN. Klasyfikuje ona wszystkie przykłady ze zbioru uczącego za pomocą klasyfikatora *k*-NN, następnie usuwa źle zaklasyfikowane przykłady klasy większościowej, a dla źle zaklasyfikowanych przykładów z klasy mniejszościowej usuwa te przykłady z ich sąsiedztwa, które przyczyniły się do błędnej klasyfikacji.

  Może być wywoływana wielokrotnie, do momentu uzyskania stanu stabilnego, w którym żadne przykłady nie są już usuwane.

Ze względu na hybrydowy charakter można do tej grupy zaklasyfikować również algorytm *SPIDER*, opisany razem z algorytmami dogenerowania przykładów.

2.2.2 Uczenie się z kosztami

Istnieją algorytmy radzące sobie z problemem niezrównoważenia liczności klas i różnych kosztów popełnianych błedów na poziomie klasyfikator, tj. na etapie uczenia się lub klasyfikacji, biorąc w ich trakcie pod uwagę macierz kosztów pomyłek.

Jedna z metod etapu klasyfikacji, możliwa do zastosowania z algorytmami zwracającymi wiarygodne oszacowanie prawdopodobieństwa przynależności klasyfikowanego przykładu do klas – takimi jak naiwny klasyfikator bayesowski czy *k*-NN – jest dokładniej opisana w sekcji 3.2 i polega na ustaleniu zależnego od kosztów progu pewności przynależności przykładu do klasy pozytywnej. Jeśli algorytm podaje pewność klasy pozytywnej wyższą niż ten próg, należy go zaklasyfikować do klasy pozytywnej, w przeciwnym razie do negatywnej.

Algorytmy drzew decyzyjnych dostosowuje się czasem do uczenia z kosztami, modyfikując miary, wykorzystywane do podziału węzłów, minimalnej pewności klasy, czy ostatecznie modyfikując *pruning* drzewa za pomocą kosztów [HG09].

Rozdział 3

COST – kosztowa optymalizacja modyfikowanych danych

3.1 Interpretacja probabilistyczna danych uczących

Zbiór danych, składający się z przykładów opisanych wartościami określonych atrybutów, można interpretować jako realizację pewnego procesu losowego. Każdy przykład w zbiorze, wartości jego atrybutów, klasa do jakiej należy, są wynikiem pewnego rozkładu prawdopodobieństwa leżącego u podstaw tego procesu.

Oznaczmy gęstość prawdopodobieństwa w punkcie $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n)^\top$ – odpowiadająca prawdopodobieństwu pojawienia się przykładu opisanego wektorem wartości atrybutów $\mathbf{x}$ – poprzez $p(\mathbf{x})$, tak aby prawdopodobieństwo pojawienia się przykładu p, opisanego wektorem wartości atrybutów $\mathbf{x}_\mathrm{p}$, wewnątrz hiperprostopadłościanu rozciągniętego między punktami $\mathbf{a}$ i $\mathbf{b}$ wynosiło:

$$\Pr\left(x_{\mathrm{p}1} \in [a_1, b_1], x_{\mathrm{p}2} \in [a_2, b_2], \ldots, x_{\mathrm{p}n} \in [a_n, b_n]\right) = \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} p(\mathbf{x})\,\mathrm{d}x_n \cdots \mathrm{d}x_2\mathrm{d}x_1. \quad (3.1)$$

Poprzez $p(\mathbf{x}, y)$ oznaczmy gęstość łącznego prawdopodobieństwa pojawienia się przykładu opisanego wektorem $\mathbf{x}$ i należącego do klasy y.

W problemie danych niezrównoważonych rozważa się zazwyczaj zbiory z przykładami należącymi do jednej z dwóch klas – pozytywnej (mniejszościowej), oznaczonej tu symbolem „+”, i negatywnej (większościowej), oznaczonej przez „−”.

Wówczas wartość masy prawdopodobieństwa warunkowego, że przykład opisany wektorem $\mathbf{x}$ będzie należeć do klasy y to:

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})}. \quad (3.2)$$

Ponieważ wartość funkcji masy prawdopodobieństwa dyskretnego procesu losowego jest równocześnie wartością prawdopodobieństwa, to $p(y|\mathbf{x})$ jest tym samym prawdopodobieństwem tego, że przykład p znajdujący się w punkcie $\mathbf{x}$ będzie należeć do klasy y:

$$\Pr(y_\mathrm{p} = y | \mathbf{x}_\mathrm{p} = \mathbf{x}) = p(y|\mathbf{x}). \quad (3.3)$$

Przy założeniu gładkości zmian gęstości prawdopodobieństwa wraz ze zmianą wartości atrybutów, wartość z równania (3.2) jest równa prawdopodobieństwu, że przykład

należący do sąsiedztwa punktu $\mathbf{x}$ będzie w klasie y.

To, do której klasy należeć będą przykłady w sąsiedztwie $\mathbf{x}$ dyktuje zmienna losowa Y o rozkładzie zero-jedynkowym o parametrze $p = p(+|\mathbf{x})$ – prawdopodobieństwie sukcesu – tego, że przykład należeć będzie do klasy mniejszościowej, a $q = 1 - p = p(-|\mathbf{x})$ to prawdopodobieństwo niepowodzenia – tego, że przykład znajdzie się w klasie większościowej. Przy założeniu, że wartości klasy różnych przykładów są niezależne, sąsiedztwo, posiadające n przykładów, z czego k przykładów w klasie mniejszościowej, stanowi zbiór n różnych realizacji zmiennej Y i jest realizacją zmiennej losowej X o rozkładzie dwumianowym opisanym parametrami n (liczba prób) i p [Fel07]:

$$Y \sim \mathrm{B}(1,p), \tag{3.4}$$

$$X \sim \mathrm{B}(n,p). \tag{3.5}$$

Prawdopodobieństwo *a posteriori* klasy mniejszościowej to wówczas:

$$\hat{p} = \frac{k}{n}. \tag{3.6}$$

Para $(n, \hat{p})$ opisuje realizację zmiennej X w sąsiedztwie wybranego przykładu i nazywamy ją *statystyką sąsiedztwa.*

3.2 Klasyfikator optymalizujący koszty popełnianych błędów

Przy problemie klasyfikacji binarnej, z klasą mniejszościową $y_+ = 1$ oraz większościową $y_- = 0$, gdzie dane są koszty c_{FP} popełnienia błędu typu FP oraz c_{FN} błędu FN, a τ to znormalizowany koszt popełnienia błędu FP:

$$\tau = \frac{c_{\mathrm{FP}}}{c_{\mathrm{FP}} + c_{\mathrm{FN}}}, \tag{3.7}$$

klasyfikator, minimalizujący koszt błędnej klasyfikacji, powinien przydzielać przykładom wartość klasy q^* wg zależności:

$$q^* = \begin{cases} 0 & \text{dla} \quad \hat{p} < \tau, \\ 1 & \text{dla} \quad \hat{p} > \tau, \end{cases} \tag{3.8}$$

gdzie $\hat{p}$ to obserwowane przez klasyfikator prawdopodobieństwo *a posteriori* klasy mniejszościowej [LS08], [ZE01].

W przypadku, gdy $\hat{p} = \tau$, wartość oczekiwana kosztu popełnionego błędu przy zaklasyfikowaniu przykładu do którejkolwiek z klas jest jednakowa.

Takie podejście może zostać zastosowane do klasyfikacji minimalizującej koszt za pomocą klasyfikatorów zwracających oszacowanie prawdopodobieństwa wartości klasy decyzyjnej lub jej częstość, tj. takich algorytmów jak m.in. naiwny klasyfikator bayesowski, sieci bayesowskie, algorytm k najbliższych sąsiadów.

3.3 Przekształcenie prawdopodobieństw, uwzględniające koszty

Gdy zasady opisanej w sekcji 3.2 nie da się wbudować bezpośrednio do klasyfikatora (np. z tego powodu, że ów nie ocenia prawdopodobieństw przynależności do klasy lub szacuje je mało wiarygodnie – np. klasyfikatory regułowe czy drzewa decyzyjne

dążą do podejmowania decyzji na podstawie niewielkiej liczby przykładów, należących do tej samej klasy, przez co nie dają wiarygodnych oszacowań [HG09]), można przekształcić zbiór uczący w taki sposób, by zmienić obserwowane w nim prawdopodobieństwa *a posteriori* tak, by klasyfikacja optymalizująca koszt przypisywała klasę mniejszościową zawsze, gdy $\hat{p} > 0{,}5$, a większościową, gdy $\hat{p} < 0{,}5$ (domyślna reguła stosowana przez bezkosztowe klasyfikatory, zwracające prawdopodobieństwo klas [LS08]).

Problem ten można rozwiązać poprzez przekształcenie prawdopodobieństw wystąpienia przykładów, jak opisano w [SSK, sekcja 3.2.1]. Poprzez $c(\mathbf{x})$ oznaczona będzie kosztowa gęstość w punkcie $\mathbf{x}$, dająca optymalną klasyfikację w klasyfikatorze nieuwzględniającym kosztów.

Kosztowa gęstość prawdopodobieństwa łącznego dla wektora atrybutów i wartości klasy może zostać obliczona jako:

$$c(\mathbf{x},+) = (1-\tau)\cdot p(\mathbf{x},+), \tag{3.9}$$

$$c(\mathbf{x},-) = \tau \cdot p(\mathbf{x},-). \tag{3.10}$$

Ogólne prawdopodobieństwo wystąpienia danej wartości klasy w przykładzie ze zbioru przekształca się podobnie:

$$c(+) = (1-\tau)\cdot p(+), \tag{3.11}$$

$$c(-) = \tau \cdot p(-). \tag{3.12}$$

Na podstawie (3.9) i (3.10) można wyznaczyć przekształconą gęstość prawdopodobieństwa w punkcie $\mathbf{x}$:

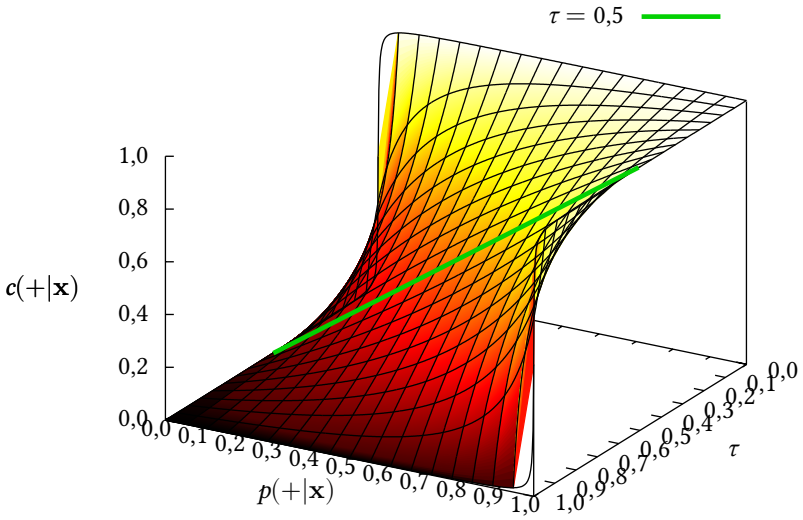
$$\begin{aligned} c(\mathbf{x}) &= c(\mathbf{x},+) + c(\mathbf{x},-) = (1-\tau)\cdot p(\mathbf{x},+) + \tau \cdot p(\mathbf{x},-) \\ &= \tau \cdot p(\mathbf{x}) + (1-2\tau)\cdot p(\mathbf{x},+). \end{aligned} \tag{3.13}$$

Mając (3.13), można wyznaczyć przekształcenie prawdopodobieństw warunkowych w sąsiedztwie $\mathbf{x}$:

$$\begin{aligned} c(+|\mathbf{x}) &= \frac{c(\mathbf{x},+)}{c(\mathbf{x})} = \frac{(1-\tau)\cdot p(\mathbf{x},+)}{\tau\cdot p(\mathbf{x}) + (1-2\tau)\cdot p(\mathbf{x},+)} \\ &= \frac{p(\mathbf{x})\cdot(1-\tau)\cdot p(+|\mathbf{x})}{p(\mathbf{x})\cdot\Big(\tau + (1-2\tau)\cdot p(+|\mathbf{x})\Big)} \\ &= \frac{(1-\tau)\cdot p(+|\mathbf{x})}{\tau + (1-2\tau)\cdot p(+|\mathbf{x})}, \end{aligned} \tag{3.14}$$

$$\begin{aligned} c(-|\mathbf{x}) &= \frac{c(\mathbf{x},-)}{c(\mathbf{x})} = \frac{\tau\cdot p(-|\mathbf{x})}{\tau + (1-2\tau)\cdot p(+|\mathbf{x})} = \frac{\tau\cdot(1-p(+|\mathbf{x}))}{\tau + (1-2\tau)\cdot p(+|\mathbf{x})} \\ &= 1 - c(+|\mathbf{x}). \end{aligned} \tag{3.15}$$

Wykres z rys. 3.1 pokazuje, jak przekształcane jest prawdopodobieństwo klasy mniejszościowej w prawdopodobieństwo kosztowe przy użyciu różnych wartości τ. Linia zielona pokazuje, że dla $\tau = 0{,}5$ przekształcenie z równania (3.14) jest odwzorowaniem tożsamościowym. Oznacza to, że $\tau = 0{,}5$ (dla którego koszty błędów FN i FP są jednakowe) jest wartością graniczną, poniżej której wartości $p(+|\mathbf{x})$ są podnoszone, a powyżej której są obniżane.



Rysunek 3.1: Przekształcenie prawdopodobieństwa warunkowego klasy mniejszościowej $\hat{p}(+|\mathbf{x})$ w zależności od kosztu τ

Z wykresu widać również, że dla $\tau = 0$ (sytuacji odpowiadającej błędom FN nieskończenie bardziej kosztownym od błędów FP) dowolne wartości $p(+|\mathbf{x})$ przekształcane są w $c(+|\mathbf{x}) = 1$. Analogicznie dla $\tau = 1$ (koszt błędu FP nieskończenie wyższy od kosztu FN) dowolne wartości $p(+|\mathbf{x})$ przekształcane są w $c(+|\mathbf{x}) = 0$.

Ponieważ praca ta zajmuje się analizą algorytmu *COST* jako metody wzmacniania klasy mniejszościowej, w dalszej części skupia się ona na działaniu tego algorytmu dla wartości $\tau < 0{,}5$.

3.3.1 Podejście naiwne

Obliczenia te można w prosty naiwny sposób zastosować do przekształcania zbioru danych – obserwować w różnych sąsiedztwach wartość prawdopodobieństwa *a posteriori* klasy mniejszościowej $\hat{p}$ i przekształcać ją wg (3.14), traktując wartość *a posteriori* jako rzeczywiste $p(+|\mathbf{x})$, a następnie tak dodawać (lub usuwać) do zbioru przykłady, by obserwowane prawdopodobieństwa zbliżyły się wartościami do obliczonych $c(+|\mathbf{x})$.

Wystarczy to zrobić dla jednej klasy, ponieważ zgodnie z (3.15), zbliżając się do optymalnego $c(+|\mathbf{x})$, zbliża się również do $c(-|\mathbf{x})$.

3.3.2 Estymacja w oparciu o probabilistyczną interpretację danych

Zamiast traktować stosunek liczby przykładów klasy mniejszościowej do wszystkich w sąsiedztwie jako oszacowanie prawdopodobieństwa warunkowego wystąpienia takiego przykładu, można estymować rzeczywiste prawdopodobieństwa w sposób mniej naiwny.

Rzeczywistego prawdopodobieństwa $p(+|\mathbf{x})$, będącego parametrem rozkładu dwumianowego, opisanego w sekcji 3.1, nie da się bezpośrednio obliczyć przy skończonej liczbie przykładów. Można je natomiast zamodelować zmienną losową P.

Wartość funkcji masy prawdopodobieństwa rozkładu dwumianowego $f_{\mathrm{B}\,(n,p)}(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$ mówi jakie jest prawdopodobieństwo, że realizacja zmiennej rozkładu $\mathrm{B}\,(n,p)$ wygeneruje k sukcesów (przykładów mniejszościowych w sąsiedztwie) [Fel07], czyli jest proporcjonalna do szansy, że p jest rzeczywistym prawdopodobieństwem, leżącym u podstaw tego sąsiedztwa. Można zatem użyć tej wartości jako *wagi* do obliczenia wartości oczekiwanej modelowanego rzeczywistego prawdopodobieństwa $p(+|\mathbf{x})$.

Ponieważ n i k są nieujemne, można zapisać, że:

$$f_{\mathrm{B}\,(n,p)}(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} = \frac{\Gamma(n+1)}{\Gamma(k+1)\cdot\Gamma(n-k+1)} \cdot p^k \cdot (1-p)^{n-k}. \quad (3.16)$$

Przy czym całka tej wartości dla $p \in [0,1]$ wynosi:

$$\int_0^1 \frac{\Gamma(n+1)}{\Gamma(k+1)\cdot\Gamma(n-k+1)} \cdot p^k \cdot (1-p)^{n-k}\,\mathrm{d}p = \frac{1}{n+1}. \quad (3.17)$$

Zatem wartości te wymagają normalizacji poprzez $n+1$, aby przedstawiały właściwą gęstość prawdopodobieństwa, mówiącego o szansie, że argument p jest równy rzeczywistemu $p(+|\mathbf{x})$. Tak znormalizowane dają funkcję wag dla p:

$$\omega_{n,k}(p) = \frac{\Gamma(n+1)\cdot(n+1)}{\Gamma(k+1)\cdot\Gamma(n-k+1)} \cdot p^k \cdot (1-p)^{n-k} \quad (3.18)$$

$$= \frac{\Gamma(n+2)}{\Gamma(k+1)\cdot\Gamma(n-k+1)} \cdot p^k \cdot (1-p)^{n-k}, \quad (3.19)$$

która jest funkcją gęstości prawdopodobieństwa zmiennej losowej P. Równocześnie jest to funkcja gęstości prawdopodobieństwa rozkładu Beta $(k+1,\, n-k+1)$. Oznaczmy ją przez $f_{\beta_{k+1,\,n-k+1}}(p)$, z definicji:

$$f_{\beta_{k+1,\,n-k+1}}(p) = \frac{\Gamma\Big((k+1)+(n-k+1)\Big)}{\Gamma(k+1)\cdot\Gamma(n-k+1)} \cdot p^{(k+1)-1} \cdot (1-p)^{(n-k+1)-1} \quad (3.20)$$

$$= \frac{\Gamma(n+2)}{\Gamma(k+1)\cdot\Gamma(n-k+1)} \cdot p^k \cdot (1-p)^{n-k} \quad (3.21)$$

$$= \omega_{n,k}(p). \quad (3.22)$$

Zatem P jest zmienną losową o rozkładzie beta (por. [KKS14]):

$$P \sim \mathrm{Beta}\,(k+1, n-k+1) = \mathrm{Beta}\,(n\hat{p}+1, n\cdot(1-\hat{p})+1). \quad (3.23)$$

Wartość oczekiwaną $p(+|\mathbf{x})$ można obliczyć, z definicji wartości oczekiwanej [JS01]:

$$\mathrm{E}_p\Big[p(+|\mathbf{x})\Big] = \int_0^1 p\cdot\omega_{n,k}(p)\,\mathrm{d}p, \quad (3.24)$$

natomiast docelową wartość oczekiwaną kosztowego prawdopodobieństwa warunkowego $c(+|\mathbf{x})$ analogicznie, po zmianie p na wartość przekształconą zgodnie z równaniem (3.14) – por. [SSK]:

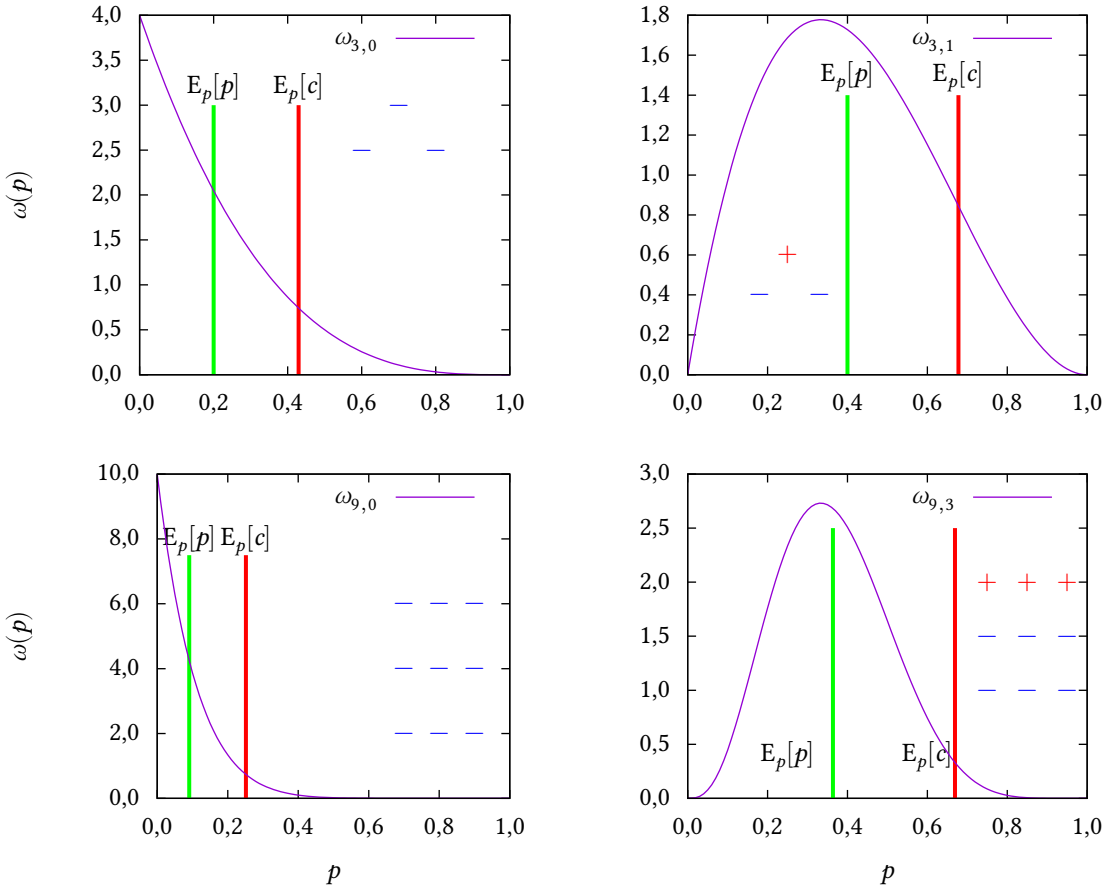
$$\mathrm{E}_p\Big[c(+|\mathbf{x})\Big] = \int_0^1 \frac{(1-\tau)\cdot p}{\tau + p\cdot(1-2\cdot\tau)} \cdot \omega_{n,k}(p)\,\mathrm{d}p \tag{3.25}$$

$$= \int_0^1 \frac{(1-\tau)\cdot p}{p+\tau-2\cdot p\tau} \cdot f_{\beta_{k+1,\,n-k+1}}(p)\,\mathrm{d}p \tag{3.26}$$

$$= \frac{1-\tau}{\tau}\cdot\frac{1+k}{2+n}\cdot {}_2\mathrm{F}_1\left(1,\,2+k;\;3+n;\;2-\frac{1}{\tau}\right), \tag{3.27}$$

gdzie ${}_2\mathrm{F}_1(a,b;\,c;\,z)$ to funkcja hipergeometryczna Gaussa.

3.3.3 Wizualizacja przekształceń prawdopodobieństw dla przykładowych sąsiedztw



Rysunek 3.2: Wykresy funkcji wag $\omega_{n,k}(p)$ z wartościami oczekiwanymi p i c dla $\tau = 0{,}2$

Na rys. 3.2 pokazano wykresy funkcji wag $\omega_{n,k}$ (czyli gęstości rozkładu Beta $(k+1, n-k+1)$) dla czterech różnych sąsiedztw. W górnym wierszu widać sąsiedztwa o trzech przykładach ($n = 3$), w dolnym o dziewięciu ($n = 9$). Po lewej stronie są wykresy sąsiedztw bez żadnych przykładów klasy mniejszościowej ($k = 0$, $\hat{p} = 0$), po prawej stronie są sąsiedztwa o $1/3$ przykładów z klasy mniejszościowej (na górze $k = 1$, $\hat{p} = 1/3$, na dole $k = 3$, $\hat{p} = 3/9 = 1/3$).

Dodatkowo na wykresach zaznaczono liniami pionowymi wartości oczekiwane kosztowego prawdopodobieństwa obliczane wg równania (3.27) – linią zieloną dla $\tau = 0{,}5$ (czyli równą $\mathrm{E}_p[p(+|\mathbf{x})]$), a linią czerwoną dla $\tau = 0{,}2$.

Widać stąd, że dla większych liczebnie sąsiedztw prawdopodobieństwo *a posteriori* przy obliczaniu wartości oczekiwanej jest traktowane jako pewniejsze i wartość oczekiwana jest mu bliższa niż w przypadku mniejszych sąsiedztw. Widać to szczególnie na wykresach z lewej strony, gdzie dla sąsiedztwa z trzema przykładami wartość $\mathrm{E}_p[p] = 0{,}2$ i jest wyraźnie dalej od $\hat{p} = 0$, niż w przypadku sąsiedztwa z dziewięcioma przykładami, gdzie $\mathrm{E}_p[p] \approx 0{,}09$.

3.4 Metoda COST

3.4.1 Wybór sąsiedztwa

Pozostaje problem tego, czym właściwie jest sąsiedztwo danego punktu w przestrzeni atrybutów zbioru danych i skąd wziąć liczbę przykładów należących do tego sąsiedztwa. Rozważono trzy metody:

- dyskretyzacja atrybutów w siatkę,
- sąsiedztwo k najbliższych sąsiadów,
- estymacja jądrowa gęstości.

Dyskretyzacja polega na ustaleniu pewnych granicznych wartości atrybutów, z których wydzielonoby hiperkostki w przestrzeni atrybutów. Każda hiperkostka byłaby wtedy jednym sąsiedztwem, do którego należałyby wszystkie przykłady znajdujące się w jej wnętrzu.

Zaletą takiego sąsiedztwa byłaby łatwość obliczania jego statystyk – dowolna modyfikacja pojedynczego przykładu zmieniałaby rozkład jedynie wewnątrz jednego sąsiedztwa i nie miałaby żadnego wpływu na pozostałe, co upraszczałoby przeliczanie statystyk sąsiedztw po modyfikacji zbioru.

Wady tego rozwiązania to:

- konieczność doboru wielu nieoczywistych parametrów – liczba podziałów dla każdego atrybutu, metoda doboru miejsca podziału, szerokość podziałów,
- w rzeczywistości zmiana liczby przykładów wewnątrz jednej hiperkostki może mieć duży wpływ na klasyfikację wewnątrz pozostałych kostek, szczególnie, gdy kopiuje się przykład z wartością bliską wartości granicznej (leżący blisko granicy dwóch hiperkostek),
- możliwość wystąpienia pustych sąsiedztw, w których nie da się obliczyć $\hat{p}$.

Sąsiedztwo k najbliższych sąsiadów jest proste w użyciu, istnieją gotowe implementacje wyszukiwania k najbliższych sąsiadów (np. [Wekb]). Jest ono również wykorzystywane przez niektóre algorytmy klasyfikujące (np. [Weka], implementacja algorytmu k-NN w pakiecie *Weka*) i przez wiele innych metod przetwarzania wstępnego danych (por. rozdział 2). Zmiany w jednym sąsiedztwie mogą mieć silny wpływ na statystykę w sąsiedztwach okolicznych. Ma ono jednak kilka wad:

- stała liczba przykładów w sąsiedztwie (o ile kilka przykładów nie znajduje się w tym samym punkcie) – uniemożliwia to dobrą ocenę niepewności prawdopodobieństwa *a posteriori*, co eliminuje jedną z zalet estymacji wartości oczekiwanej kosztowego prawdopodobieństwa za pomocą rozkładu beta,
- skwantowane wartości prawdopodobieństw *a posteriori* – jeśli w każdym sąsiedztwie jest k_{NN} przykładów, to zawsze $\hat{p} \in \left\{0, \frac{1}{k_{\mathrm{NN}}}, \frac{2}{k_{\mathrm{NN}}}, \frac{3}{k_{\mathrm{NN}}}, \ldots, 1\right\}$,
- wszystkie przykłady w sąsiedztwie mają jednakowy wpływ na obliczone prawdopodobieństwo, mimo że potencjalnie mogą się znajdować arbitralnie daleko od badanego punktu – ich odległość nie ma znaczenia, jeśli tylko są k-tym lub bliższym sąsiadem.

Estymacja jądrowa gęstości polega na obliczaniu przybliżonej gęstości występowania przykładów w danym punkcie przestrzeni danych za pomocą funkcji jądrowej. Jest to metoda wybrana w [SSK], do jej zrealizowania wybrany jest wielowymiarowy estymator gęstości Gaussa, opisany funkcją:

$$\mathrm{K}(\mathbf{x}) = \exp\left(-\frac{1}{2}\mathbf{x}^{\top}\Sigma^{-1}\mathbf{x}\right), \tag{3.28}$$

gdzie Σ jest macierzą kowariancji atrybutów.

Statystykę sąsiedztwa $\mathbf{x}$ uzyskuje się wg:

$$n = \sum_{\mathbf{x}_i \in S} \mathrm{K}(\mathbf{x} - \mathbf{x}_i) = \sum_{\mathbf{x}_i \in S} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_i)^{\top}\Sigma^{-1}(\mathbf{x} - \mathbf{x}_i)\right), \tag{3.29}$$

$$k = \sum_{\mathbf{x}_{i+} \in S} \mathrm{K}(\mathbf{x} - \mathbf{x}_{i+}) = \sum_{\mathbf{x}_{i+} \in S} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_{i+})^{\top}\Sigma^{-1}(\mathbf{x} - \mathbf{x}_{i+})\right), \tag{3.30}$$

$$\hat{p} = \frac{k}{n}, \tag{3.31}$$

gdzie $\mathbf{x}_i \in S$ oznacza wszystkie przykłady w zbiorze uczącym S, a $\mathbf{x}_{i+} \in S$ wszystkie przykłady klasy mniejszościowej w tym zbiorze.

3.4.2 Minimalizowana miara odmienności rozkładów

Celem algorytmu powinno być takie powielanie przykładów oryginalnego zbioru danych, aby zbliżyć do siebie obserwowane prawdopodobieństwa w sąsiedztwach do wartości oczekiwanych prawdpodobieństwa kosztowego. Powinno to pozwolić później, niedostosowanym do uczenia z kosztami, klasyfikatorom nauczonym na przetworzonym zbiorze lepiej minimalizować koszt pomyłek.

W tym celu oznaczmy docelową wartość prawdopodobieństwa klasy mniejszościowej w sąsiedztwie, wyliczaną wg równania (3.27):

$$t(+|\mathbf{x}) = \mathrm{E}_p\Big[c(+|\mathbf{x})\Big]. \tag{3.32}$$

Algorytm powinien tak powielać przykłady, aby minimalizować odmienność między $\hat{p}(+|\mathbf{x})$ obserwowanym w sąsiedztwach, a $t(+|\mathbf{x})$ obliczonym dla tych sąsiedztw.

W [SSK] sugeruje się użycie dywergencji Kullbacka-Leiblera, będącej miarą dopasowania do siebie dwóch różnych rozkładów prawdopodobieństwa, jako miary użytej

przez algorytm do minimalizacji. Miara ta wymaga, do porównania, funkcji masy lub gęstości prawdopodobieństw [CT06].

Ponieważ prawdopodobieństwa warunkowe klas w niezależnych sąsiedztwach są również od siebie niezależne, nie można użyć ich bezpośrednio do obliczania dywergencji Kullbacka-Leiblera, zatem ostatecznie miarą proponowaną do użycia wg [SSK] jest suma dywergencji dla obu klas rozkładów łącznych znalezienia przykładu z danej klasy w danym sąsiedztwie, oznaczmy ją przez d:

$$d = \sum_{\mathbf{x} \in T} \left[t(\mathbf{x}, -) \cdot \log \frac{t(\mathbf{x}, -)}{\hat{p}(\mathbf{x}, -)} + t(\mathbf{x}, +) \cdot \log \frac{t(\mathbf{x}, +)}{\hat{p}(\mathbf{x}, +)} \right], \tag{3.33}$$

gdzie T oznacza zbiór punktów, w których obliczane są prawdopodobieństwa *a posteriori* oraz docelowe, obliczane na podstawie wartości docelowych warunkowych (obliczonych wg równania (3.27)):

$$t(\mathbf{x}, y) = t(y \,|\mathbf{x}) \cdot \hat{p}(\mathbf{x}), \tag{3.34}$$

natomiast prawdopodobieństwa wystąpienia przykładu w sąsiedztwie ($\hat{p}(\mathbf{x})$) oraz prawdopodobieństwa łączne ($\hat{p}(\mathbf{x}, y)$) estymowane są – podobnie jak statystyka sąsiedztwa (por. równania (3.29, 3.30)) – metodą jądrowej estymacji, znormalizowanym estymatorem Gaussa:

$$\begin{aligned} \hat{p}(\mathbf{x}) &= \frac{1}{|S|} \sum_{\mathbf{x}_i \in S} \frac{1}{\sqrt{(2\pi)^D \cdot |\Sigma|}} \mathrm{K}(\mathbf{x} - \mathbf{x}_i) \\ &= \frac{1}{|S|} \sum_{\mathbf{x}_i \in S} \frac{1}{\sqrt{(2\pi)^D \cdot |\Sigma|}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{x}_i)^\top \Sigma^{-1} (\mathbf{x} - \mathbf{x}_i) \right), \end{aligned} \tag{3.35}$$

$$\hat{p}(\mathbf{x}, y) = \frac{1}{|S|} \sum_{\mathbf{x}_{iy} \in S} \frac{1}{\sqrt{(2\pi)^D \cdot |\Sigma|}} \mathrm{K}(\mathbf{x} - \mathbf{x}_{i+}), \tag{3.36}$$

gdzie D jest liczbą atrybutów, a $|S|$ jest liczbą przykładów w zbiorze S, $\mathbf{x}_{iy} \in S$ oznacza zbiór wszystkich przykładów z S z klasy y.

Aby obliczyć opisaną wyżej dywergencję między rozkładami, należy jeszcze wybrać zbiór T punktów, w których sąsiedztwach prawdopodobieństwa będą estymowane. W momencie pisania tej pracy, w [SSK] proponuje się użycie siatki przykładów równoodległych na wszystkich atrybutach.

3.4.3 Metoda minimalizacji odmienności między rozkładami

Dla pierwszej wersji algorytmu *COST* wybrano prostą metodę optymalizacji z rodziny metaheurystyk przeszukiwania lokalnego – strome przeszukiwanie lokalne (ang. *steepest descent*). Algorytm inicjuje rozwiązanie problemu jako pusty zbiór danych $S' = \varnothing$, następnie kolejno bada efekt, jaki na wartość dywergencji d opisanej w poprzedniej sekcji, będzie mieć dodanie do rozwiązania kolejnych przykładów z oryginalnego zbioru danych S. Następnie dodaje do rozwiązania przykład minimalizujący d i rozważa ponownie dodanie, kolejno, każdego z przykładów z S, jak i bada wpływ usunięcia poprzednio dodanych przykładów z S'. Iteracyjnie powtarza operację dodania i ew. usunięcia przykładu, aż osiągnie punkt, w którym żadna możliwa do wykonania operacja

dodania lub usunięcia przykładu nie zmniejsza dywergencji między rozkładami prawdopodobieństw obserwowanych i docelowych – w tym momencie zwraca wynikowy zbiór S'.

3.4.4 Algorytm

Algorytmy 1, 2 i 3 przedstawiają zapis metody *COST* taki, jaki został zaproponowany w [SSK].

Wymaga on na następujących danych:

- $S = (x_i, y_i)$ – oryginalny zbiór danych
- $S' = (x_i, y_i)$ – wyjściowy przetworzony zbiór danych
- $T = (x_i, y_i, t_i)$ – zbiór punktów i wartości docelowych
- d – miara odmienności między S' i T
- D – liczba atrybutów
- Σ – macierz kowariancji atrybutów $D \times D$

Algorytm 1 Algorytm COST

function COST(S, τ)
  $T \leftarrow$ EQUIDISTANTSAMPLE(S) ▷ Todo: Should be equidistant sampling points
  for $\mathbf{x}_T \in T$ **do**
    $s_{\text{joint}}, s_{\text{marg}} \leftarrow 0$ ▷ compute $\Pr(\mathbf{x}_T, +)$ and $\Pr(\mathbf{x}_T)$ at $S(\mathbf{x}_T, +)$
    for $(\mathbf{x}_S, y_S) \in S$ **do**
      $s_{\text{marg}} \leftarrow s_{\text{marg}} + \frac{1}{|S|} \cdot \text{KERNEL}(\mathbf{x}_S - \mathbf{x}_T)$
      if $y_S = +$ **then**
        $s_{\text{joint}} \leftarrow s_{\text{joint}} + \frac{1}{|S|} \cdot \text{KERNEL}(\mathbf{x}_S - \mathbf{x}_T)$
      end if
    end for
    $s_{\text{post}} \leftarrow \frac{s_{\text{joint}}}{s_{\text{marg}}}$ ▷ compute $\Pr(+|\mathbf{x}_T)$
    $c_{\text{post}} \leftarrow \frac{(1-\tau) \cdot s_{\text{post}}}{\tau + s_{\text{post}} \cdot (1 - 2 \cdot \tau)}$ ▷ Cost-weighting posterior $c(+|\mathbf{x}_T)$ using Eq. (3.14)
    $t(\mathbf{x}, +) \leftarrow \mathrm{E}_p\left[c_{\text{post}}\right] \cdot s_{\text{marg}}$ ▷ Compute cost-weighted target joint using Eq. (3.27)
    $t(\mathbf{x}, -) \leftarrow \mathrm{E}_p\left[1 - c_{\text{post}}\right] \cdot s_{\text{marg}}$
  end for
  $S' \leftarrow$ RESAMPLE(S, T)
  return S'
end function

Funkcja KERNEL użyta w funkcjach z algorytmów 1 i 3 obliczana jest wg:

$$\text{KERNEL}(\mathbf{x}) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1} \mathbf{x}\right). \tag{3.37}$$

Algorytm 2 Funkcja dogenerowania przykładów

function RESAMPLE(S,T)
  $S' \leftarrow \emptyset$
  $d,d' \leftarrow \infty$
  $i \leftarrow 0$
  repeat
    if $i < 0$ **then** ▷ removal
      $S' \leftarrow S' \setminus S'(-i)$ ▷ remove instance
    else if $i > 0$ **then** ▷ addition
      $S' \leftarrow S' \cup S(i)$ ▷ add instance
    end if
    $d \leftarrow d'$
    for $i^* = -|S'|, -1, 1, \ldots, |S|$ **do**
      if $i^* < 0$ **then**
        $d'_{i^*} \leftarrow$ DISSIMILARITY$(T, S' \setminus S'(-i^*))$ ▷ check for removal of instance i^*
      else if $i^* > 0$ **then**
        $d'_{i^*} \leftarrow$ DISSIMILARITY$(T, S' \cup S(i^*))$ ▷ check for adding instance i^*
      end if
    end for
    $i \leftarrow \arg\min_{i^*} d'_{i^*}$
    $d' \leftarrow d'_i$
  until $d - d' \leqslant 0$
  return S', d
end function

Algorytm 3 Funkcja obliczania miary odmienności rozkładów

function DISSIMILARITY(T,S)
  $d_{T||S} \leftarrow 0$
  for $(\mathbf{x}_T, y_T) \in T$ **do** ▷ iterate over all positions $(\mathbf{x}_S, y_S) \in T$
    $s \leftarrow 0$ ▷ compute density at $S(\mathbf{x}_T, y_T)$
    for $(\mathbf{x}_S, y_S) \in S$ **do**
      if $y_S = y_T$ **then**
        $s \leftarrow s + \frac{1}{|S|} \cdot \text{KERNEL}(\mathbf{x}_S - \mathbf{x}_T)$
      end if
    end for
    $d_{T||S} \leftarrow d_{T||S} + t(\mathbf{x}_T, y_T) \cdot \log\left(\frac{t(\mathbf{x}_T, y_T)}{s}\right)$ ▷ add divergence at $(\mathbf{x}_T, y_T)$
  end for
  return $d_{T||S}$
end function

Rozdział 4

Implementacja

Na potrzeby niniejszej pracy stworzono implementację algorytmu wstępnego przetwarzania niezrównoważonego zbioru danych, opartego o koszty błędów typu FP i FN, jako klasę `COSTFilter`, będącą filtrem danych dla pakietu *Weka*. Implementacja ta w pewnych szczegółach różni się od zaproponowanej w [SSK] wersji metody *COST* (dokładne różnice opisane są w sekcji 4.2).

4.1 Wykorzystane narzędzia

4.1.1 Weka

Podczas implementacji korzystano z wielu komponentów pakietu *Weka* – napisanego w *Javie* rozbudowanego systemu uczenia maszynowego o otwartych źródłach, udostępniającego wiele metod przetwarzania danych, klasyfikacji, oceny klasyfikacji, wizualizacji zbiorów danych. *Weka* jest systemem modułowym, umożliwiającym względnie łatwe rozbudowywanie go poprzez implementację nowych algorytmów[1] [Hal+09].

Dzięki tym cechom była dostatecznie dobrym narzędziem do wykorzystania podczas implementacji nowego algorytmu przetwarzania wstępnego danych – umożliwiała wykorzystanie gotowych implementacji wyszukiwania najbliższych sąsiadów czy obliczania odległości między sąsiadami, a do tego pozwalała na łatwe uzyskanie wartości wielu miar oceny klasyfikacji w eksperymencie porównującym klasyfikację zbiorów, na których zastosowano różne algorytmy filtrujące.

Dodatkowo, dla porównania wyników działania stworzonej implementacji *COST*, wykorzystano wiele implementacji innych filtrów niezrównoważonych danych, napisanych dla *Weki* w Instytucie Informatyki na Wydziale Informatyki Politechniki Poznańskiej – wykorzystanie tego pakietu pozwoliło oszczędzić czas, który byłby konieczny w innym przypadku na implementację tych algorytmów za pomocą innych narzędzi.

4.1.2 Biblioteki matematyczne

Ze względu na dużą ilość nietrywialnych obliczeń wymaganych przez implementowany algorytm konieczne było wykorzystanie w nim pewnych bibliotek matematycznych.

Sporym problemem okazało się znalezienie napisanej w języku *Java*, lub innym umożliwiającym wykonanie kodu w nim napisanego z poziomu programu w *Javie*, implemen-

[1] Strona internetowa projektu Weka: `http://www.cs.waikato.ac.nz/ml/weka/`

tacji funkcji hipergeometrycznej Gaussa ${}_2\mathrm{F}_1(a, b; c; z)$, obecnej w równaniu (3.27), koniecznym do obliczenia wartości oczekiwanej docelowego, przekształconego kosztem, prawdopodobieństwa wystąpienia przykładu klasy mniejszościowej w danym sąsiedztwie.

Rozważano użycie dwóch gotowych implementacji:

- Funkcja `gsl_sf_hyperg_2F1()` z *GNU Scientific Library* [Gal+09], napisanej w języku *C* biblioteki będącej częścią projektu *GNU*. Umożliwia obliczanie wartości hipergeometrycznej funkcji Gaussa dla argumentów rzeczywistych, jednak z ograniczeniem takim, że $|z| < 1$.
- Funkcja `sf_hypergeom_2f1()` z projektu *Special functions for Octave*[2], implementującego szereg brakujących specjalnych funkcji matematycznych dla systemu obliczeń numerycznych *GNU Octave*. Jednak tutaj również występuje ograniczenie argumentu $|z| < 1$, oraz wywołanie funkcji *Octave'a* z programu w *Javie* byłoby trudne.

Ponieważ w równaniu (3.27) argument z funkcji geometrycznej wynosi $z = 2 - \frac{1}{\tau}$, to znaczy, że $|z| \geqslant 1$ dla $\tau \leqslant \frac{1}{3}$. Powoduje to, że wyżej opisane biblioteki nie mogły zostać wykorzystane w implementacji algorytmu *COST*.

Ponadto rozważano użycie biblioteki `org.nevec.rjm`, opisanej w [Mat09], implementującej szereg funkcji matematycznych w *Javie* dla typu `BigDecimal` – przechowującego liczby rzeczywiste dowolnej dokładności. Dokumentacja tej biblioteki nie wspomina o ograniczeniach wartości argumentów implementowanej przez nią funkcji hipergeometrycznej Gaussa.

Z uwagi na opisane wyżej ograniczenia zrezygnowano zupełnie z używania funkcji hipergeometrycznej, na rzecz zaimplementowania metody obliczania wartości oczekiwanej z równania (3.26) – wykorzystującej funkcję gęstości prawdopodobieństwa rozkładu beta.

Do tego użyto popularnej biblioteki *Apache Commons Math*[3]. Jest to otwartoźródłowa biblioteka matematyczna, napisana w *Javie*, implementująca m.in. algorytmy algebry liniowej, statystyki, generatory liczb pseudolosowych, funkcje związane z wieloma rozkładami prawdopodobieństwa, i wiele innych.

Biblioteka ta posiada klasę `BetaDistribution`, implementującą metodę `density()`, zwracającą wartość gęstości prawdopodobieństwa rozkładu beta o zadanych parametrach w zadanym punkcie.

Dodatkowo biblioteki tej użyto do implementacji funkcji jądrowej estymatora gęstości Gaussa, opisanej równaniem (3.28), wymagającej mnożenia, transpozycji i odwracania macierzy. Można jej również użyć do obliczania wyznacznika macierzy na potrzeby równania (3.37).

[2] W czasie pisania tej pracy jest dostępny jako projekt zarchiwizowany pod URL: `https://code.google.com/p/special-functions-octave-library/`

[3] Strona projektu: `https://commons.apache.org/proper/commons-math/`

4.2 Uproszczenia w stosunku do oryginalnie zaproponowanego algorytmu

Na potrzeby niniejszej pracy stworzono implementację algorytmu wstępnego przetwarzania niezrównoważonego zbioru danych, opartego o koszty błędów typu FP i FN. Implementacja ta różni się w następujących punktach od zaproponowanej w [SSK].

Przede wszystkim zamiast estymować wartości gęstości prawdopodobieństwa $p(\mathbf{x})$, $p(\mathbf{x},+)$ i $p(\mathbf{x},-)$ w wybranych punktach, używane są jedynie prawdopodobieństwa warunkowe klasy mniejszościowej ($p(+|\mathbf{x})$) w sąsiedztwach tych punktów. Zdecydowano się na to, ponieważ w momencie wykonywania implementacji metoda szacowania wartości łącznych nie została jeszcze zaproponowana.

Ze względu na wyżej opisaną różnicę niemożliwe było zastosowanie dywergencji Kullbacka-Leiblera jako miary odmienności między rozkładami. Wynika to, jak opisano w sekcji 3.4.2, z tego, że prawdopodobieństwa warunkowe w różnych sąsiedztwach są niezależne i nie tworzą wspólnie rozkładu prawdopodobieństwa. Wybrano, w związku z tym, inny powszechny sposób porównywania obserwowanych wartości prawdopodobieństwa *a posteriori* z wartościami oczekiwanymi rozkładu docelowego – w wybranych sąsiedztwach istnieje możliwość porównywania sumy wartości bezwzględnych błędów (SAE, ang. *Sum of Absolute Errors*):

$$d = \sum_{\mathbf{x} \in T} \left| t(+|\mathbf{x}) - \hat{p}(+|\mathbf{x}) \right|, \tag{4.1}$$

lub sumy kwadratów błędów (SSE, ang. *Sum of Squared Errors*):

$$d = \sum_{\mathbf{x} \in T} \left(t(+|\mathbf{x}) - \hat{p}(+|\mathbf{x}) \right)^2. \tag{4.2}$$

Tę drugą miarę wykorzystano w eksperymentach opisanych w dalszej części pracy.

Następnym uproszczeniem jest wybór punktów, w których obliczane są statystyki sąsiedztwa i docelowe wartości prawdopodobieństw. We właściwym algorytmie *COST* proponuje się utworzenie siatki równoodległych przykładów w całym obszarze przestrzeni atrybutów i w nich estymowanie rozkładów prawdopodobieństw występowania przykładów obu klas. W stworzonej na potrzeby niniejszej pracy implementacji wykorzystuje się punkty, w których znajdują się rzeczywiste przykłady z oryginalnego zbioru danych (zbiór T inicjalizowany jest przykładami ze zbioru oryginalnego S). Eliminuje to konieczność ustalania wartości dodatkowych parametrów (jak np. gęstość siatki na różnych wymiarach) i daje większą liczbę wartości w obszarach gęsto reprezentowanych w zbiorze danych niż w rzadziej zapełnionych przykładami. Z drugiej strony powoduje to konieczność wykonywania ogromnej ilości obliczeń dla dużych zbiorów danych.

Zaimplementowano dwa typy sąsiedztwa: zarówno proponowaną oryginalnie estymację jądrową gęstości, jak i rozważane wcześniej (i popularniejsze wśród innych algorytmów, por. algorytmy opisane w rozdziale 2) sąsiedztwo k najbliższych sąsiadów. Z powodów opisanych w rozdziale 5 w pracy tej skupiono się na analizie wyników uzyskanych z eksperymentów z sąsiedztwem k najbliższych sąsiadów.

Ostatnią istotną różnicą w wykonanej implementacji jest to, że wykonuje ona jedynie dogenerowanie przykładów, nie umożliwia usunięcia przykładów istniejących. W oryginalnym *COST* wyjściowy, przefiltrowany, zbiór danych tworzony jest od zbioru pustego

poprzez dodawanie przykładów ze zbioru wejściowego (i ewentualne późniejsze usuwanie przykładów wcześniej dodanych), jednak nie jest zagwarantowane, że wszystkie przykłady oryginalne znajdą się w zbiorze wyjściowym (co odpowiada *undersamplingowi*, redukcji liczby przykładów) – opisany algorytm jest zatem hybrydowy. Opisywana implementacja jednak inicjalizuje rozwiązanie jako oryginalny zbiór i pozwala usuwać jedynie zduplikowane przykłady, zatem zawsze zwraca nadzbiór oryginalnego zbioru danych.

Możliwość tworzenia nowego zbioru, wychodząc od zbioru pustego, również została zaimplementowana, jednak w momencie pisania tej pracy nie jest przetestowana i możliwe jest, że nie działa w pełni poprawnie.

4.3 Zapis i szczegóły techniczne zaimplementowanej wersji *COST*

Algorytmy 4, 5 i 6 przedstawiają wersję algorytmu *COST*, jaką zaimplementowano na potrzeby tej pracy, ze zmianami w stosunku do oryginalnej propozycji opisanymi wyżej w tym rozdziale.

Algorytm 4 Zaimplementowany algorytm COST

function COST(S, τ)
  $T \leftarrow S$ ▷ inicjalizacja punktów do porównania oryginalnym zbiorem
  for $\mathbf{x}_T \in T$ **do**
    $n, k \leftarrow 0$ ▷ oblicz statystykę sąsiedztwa: n, k, $\hat{p}$
    for $(\mathbf{x}_S, y_S) \in S$ **do**
      $n \leftarrow n + \text{KERNEL}(\mathbf{x}_S - \mathbf{x}_T)$ ▷ lub $n+1$ dla sąsiedztwa k_{NN} jeśli $\mathbf{x}_S$ należy ▷ do k_{NN} najbliższych sąsiadów $\mathbf{x}_T$, $n+0$, gdy nie
      if $y_S = +$ **then**
        $k \leftarrow k + \text{KERNEL}(\mathbf{x}_S - \mathbf{x}_T)$ ▷ lub $k + \{0,1\}$ dla sąsiedztwa k_{NN}
      end if
    end for
    $\hat{p} \leftarrow \frac{k}{n}$ ▷ oblicz $\hat{p}(+|\mathbf{x}_T)$
    $t(+|\mathbf{x}_T) \leftarrow \text{E}_p\left[\frac{1-\tau}{p+\tau-2\cdot\tau} p\right]$ ▷ Oblicz docelową wartość wg (3.26)
  end for
  $S' \leftarrow \text{RESAMPLE}(S, T)$
  return S'
end function

Funkcja KERNEL użyta w funkcjach z algorytmów 4 i 6 definiowana jest – w przeciwieństwie do wersji z [SSK] – wg równania 3.28 jako:

$$\text{KERNEL}(\mathbf{x}) = \exp\left(-\frac{1}{2}\mathbf{x}^\top \Sigma^{-1} \mathbf{x}\right). \tag{4.3}$$

Dodatkowo utworzona klasa `COSTFilter`, gdy wywoływana z sąsiedztwem z estymatorem jądrowym, domyślnie buduje diagonalną macierz kowariancji z wariancjami atrybutów na przekątnej na podstawie podanej jej wartości odchylenia standardowego σ:

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix}. \tag{4.4}$$

Algorytm 5 Zaimplementowana funkcja dogenerowania przykładów

function RESAMPLE(S,T)
  $S' \leftarrow \emptyset$
  $d,d' \leftarrow \infty$
  $i \leftarrow 0$
  repeat
    if $i < 0$ **then** ▷ jeśli usuwanie
      $S' \leftarrow S' \setminus S'(-i)$ ▷ usuń dodany wcześniej przykład
    else if $i > 0$ **then** ▷ jeśli dodawanie
      $S' \leftarrow S' \cup S(i)$ ▷ skopiuj nowy przykład
    end if
    $d \leftarrow d'$
    for $i^* = -|S'|, -1, 1, \ldots, |S|$ **do**
      if $i^* < 0$ **then**
        $d'_{i^*} \leftarrow$ DISSIMILARITY$(T, S' \cup S \setminus S'(-i^*))$ ▷ usuń przykład z S' i oblicz odmienność
        ▷ dla **sumy** nowego i oryginalnego zbioru
      else if $i^* > 0$ **then**
        $d'_{i^*} \leftarrow$ DISSIMILARITY$(T, S' \cup S \cup S(i^*))$ ▷ to samo dla kopiowania
      end if
    end for
    $i \leftarrow \arg\min_{i^*} d'_{i^*}$
    $d' \leftarrow d'_i$
  until $d - d' \leqslant 0$
  return $S' \cup S, d$ ▷ zwróć sumę nowych i oryg. przykładów
end function

Algorytm 6 Zaimplementowana funkcja obliczania miary odmienności rozkładów

function DISSIMILARITY(T,S)
  $d_{T||S} \leftarrow 0$
  for $\mathbf{x}_T \in T$ **do** ▷ dla wszystkich przykładów z T
    $n,k \leftarrow 0$ ▷ oblicz statystykę sąsiedztwa $\mathbf{x}_T$
    for $(\mathbf{x}_S, y_S) \in S$ **do**
      $n \leftarrow n +$ KERNEL$(\mathbf{x}_S - \mathbf{x}_T)$ ▷ lub $n + \{0,1\}$ dla sąsiedztwa k_{NN}
      if $y_S = +$ **then**
        $k \leftarrow k +$ KERNEL$(\mathbf{x}_S - \mathbf{x}_T)$ ▷ lub $k + \{0,1\}$ dla sąsiedztwa k_{NN}
      end if
    end for
    $\hat{p} \leftarrow \frac{k}{n}$
    $d_{T||S} \leftarrow d_{T||S} + (t(+|\mathbf{x}_T) - \hat{p})^2$ ▷ lub wart. bezwzględna
  end for
  return $d_{T||S}$
end function

Dla sąsiedztwa k najbliższych sąsiadów algorytm wymaga podania mu wielkości sąsiedztwa.

W przypadku obu sąsiedztw (k najbliższych sąsiadów, jak i estymacji gęstości) do porównywania przykładów użyta została miara odległości euklidesowej:

$$d_{\text{euklid}}(\mathbf{x}_\text{a}, \mathbf{x}_\text{b}) = \sqrt{\sum_{i=1}^{D} (x_{\text{a}i} - x_{\text{b}i})^2}, \tag{4.5}$$

gdzie D jest liczbą atrybutów. Wartości wszystkich numerycznych atrybutów występujące w zbiorze danych są normalizowane do wartości z zakresu $[0;\ 1]$, porównanie na atrybutach nominalnych daje wartość 1 dla różnych wartości i 0 dla jednakowych.

W przypadku statystyk sąsiedztw z estymacją gęstości wykonano pewną optymalizację, nieujętą w algorytmie 5. Zamiast obliczać od nowa statystyki sąsiedztw we wszystkich punktach z T na podstawie wszystkich punktów z $S' \cup S$ przy próbie dodania lub usunięcia przykładu, oblicza się dla tych sąsiedztw poprawkę, zależną jedynie od dodawanego lub usuwanego przykładu. Pozwala to zmniejszyć złożoność obliczeniową jednej iteracji z $O(|S|^3)$ do $O(|S|^2)$. Z powodu trudności określenia dokładnego wpływu dodania lub usunięcia przykładu na wszystkie sąsiedztwa w przypadku używania k najbliższych sąsiadów, dla tego sąsiedztwa nie udało się przeprowadzić takiej optymalizacji.

Rozdział 5

Wyniki eksperymentów z użyciem wykonanej implementacji

W celu analizy działania i określenia przydatności zaimplementowanej klasy `COSTFilter` przeprowadzono opisane w tym rozdziale eksperymenty. Podzielone są one na eksperymenty sprawdzające poprawność działania elementów składowych algorytmu (opisane w sekcji 5.1.1), eksperymenty służące analizie działania całego algorytmu – tego w jaki sposób powiela on przykłady ze zbioru uczącego (opisane w sekcji 5.1.2), oraz eksperymenty wykonane w celu porównania klasyfikacji na sztucznych i rzeczywistych zbiorach danych przetworzonych wybranymi zbiorami danych (sekcja 5.2).

5.1 Analiza działania filtru `COSTFilter`

5.1.1 Testy poprawności działania elementów algorytmu

Jednym z elementów wymagających sprawdzenia poprawności działania był komponent obliczający docelowe prawdopodobieństwa warunkowe w badanych sąsiedztwach, $t(+|\mathbf{x})$ – ponieważ, jak opisano w rozdziale 4, z powodu braku łatwo dostępnej implementacji funkcji hipergeometrycznej Gaussa, posłużono się numerycznym całkowaniem wartości z użyciem funkcji gęstości prawdopodobieństwa rozkładu beta z biblioteki *Apache Commons Math*.

Wymagało to sprawdzenia, czy wyniki uzyskiwane numerycznym całkowaniem są zgodne z rzeczywistymi wartościami. Z powodu braku dostępności odpowiedniej implementacji funkcji ${}_2F_1(a, b;\ c;\ z)$ w pakietach *Matlab*, *Octave* czy w programistycznych bibliotekach matematycznych napisanych w języku *C* lub w *Javie*, posłużono się stroną internetową *Wolfram|Alpha*, której zadawano zapytania postaci:

`((1 - τ)/τ) ((1 + k)/(2 + n)) Hypergeometric2F1[1, 2 + n, 3 + k, 2 - 1/τ]`,

podstawiając pod zmienne odpowiednie wartości do zapytania[1]. Wyniki podobne do tych z *Wolframu* można uzyskać również za pomocą otwartoźródłowego systemu algebry komputerowej *Maxima*[2].

Sprawdzono obliczenia dla niskiej liczności przykładów w sąsiedztwie ($n = 1$, niewielka pewność prawdopodobieństwa *a posteriori*), dla wartości niecałkowitej ($n = 5{,}3$,

[1] Przykładowe zapytanie i jego wynik można obejrzeć pod adresem URL: `http://www.wolframalpha.com/input/?i=((1 - 0.25)/0.25) ((1 + 2)/(2 + 5)) Hypergeometric2F1[1, 2 + 2, 3 + 5, 2 - 1/0.25]`

[2] Strona projektu: `http://maxima.sourceforge.net/`

taką można uzyskać za pomocą jądrowej estymacji gęstości) oraz dla większej liczności przykładów ($n = 20$, wyższa pewność prawdopodobieństwa *a posteriori*).

Tabela 5.1: Porównanie wartości $t(+|\mathbf{x})$ obliczonych przez wykonaną implementację oraz *Wolfram|Alpha*

τ	n	k	$\hat{p}$	$t(+\|\mathbf{x})$ `COSTFilter`	$t(+\|\mathbf{x})$ *Wolfram|Alpha*
0,0	1,0	0,000	0,000	0,999 512	–
	5,3	1,767	0,333	1,000 000	–
	20,0	20,000	1,000	0,994 881	–
0,25	1,0	0,000	0,000	0,528 122	0,528 122
		0,333	0,333	0,645 282	0,645 283
		1,000	1,000	0,823 471	0,823 959
	5,3	0,000	0,000	0,286 793	0,286 793
		1,767	0,333	0,614 492	0,614 492
		5,300	1,000	0,943 297	0,944 835
	20,0	0,000	0,000	0,117 190	0,117 191
		6,667	0,333	0,603 854	0,603 854
		20,000	1,000	0,978 769	0,983 888
0,5	1,0	0,000	0,000	0,333 333	0,333 333
		0,333	0,333	0,444 443	0,444 444
		1,000	1,000	0,666 178	0,666 667
	5,3	0,000	0,000	0,136 986	0,136 986
		1,767	0,333	0,378 995	0,378 995
		5,300	1,000	0,861 476	0,863 014
	20,0	0,000	0,000	0,045 454	0,045 455
		6,667	0,333	0,348 485	0,348 485
		20,000	1,000	0,949 427	0,954 545
0,75	1,0	0,000	0,000	0,176 041	0,176 041
		0,333	0,333	0,257 407	0,257 408
		1,000	1,000	0,471 390	0,471 878
	5,3	0,000	0,000	0,055 165	0,055 165
		1,767	0,333	0,185 927	0,185 927
		5,300	1,000	0,711 670	0,713 207
	20,0	0,000	0,000	0,016 112	0,016 112
		6,667	0,333	0,156 385	0,156 385
		20,000	1,000	0,877 692	0,882 809

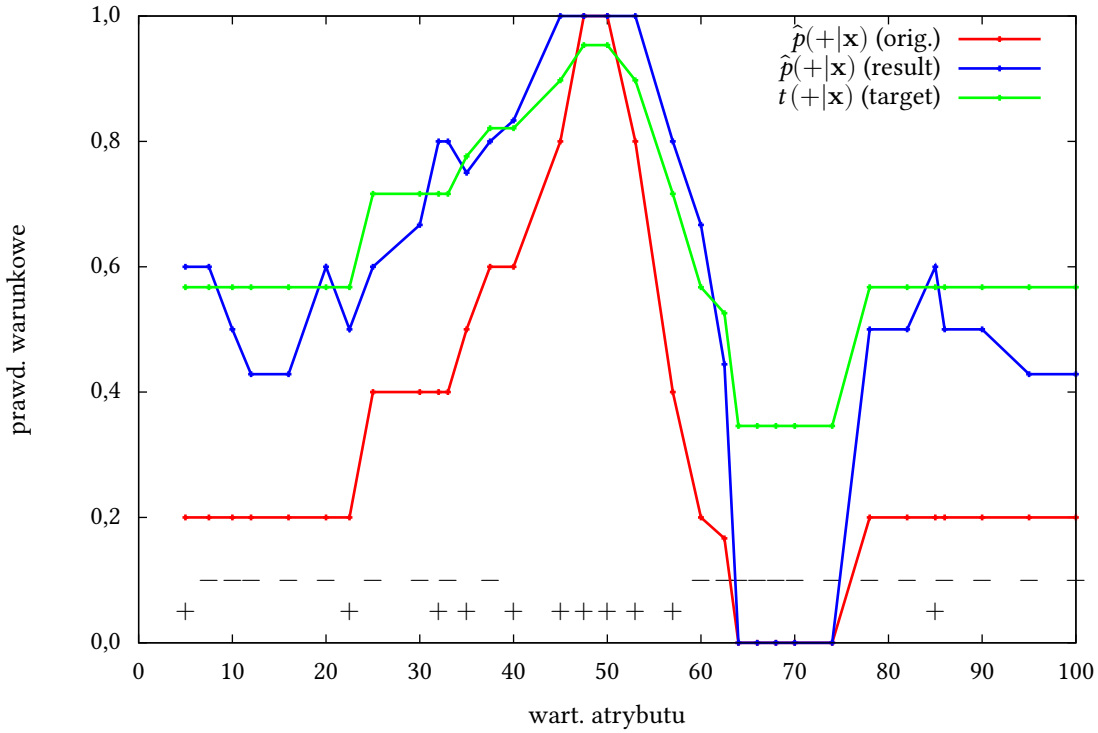
Porównanie wartości uzyskanych z opisywanej tu implementacji oraz ze strony *Wolfram|Alpha* przedstawiono w tabeli 5.1. Widać z niej, że dla przypadków z niskim $\hat{p}$ różnice wyników całkowania numerycznego różnią się nieznacznie (w najgorszym przypadku na szóstym miejscu po przecinku) od wartości rzeczywistych. Większe różnice występują dla przypadków o $\hat{p} = 1$, jednak i tu nie występują różnice większe niż 0,006.

Dla $\tau = 0$ oczekiwaną poprawną wartością byłoby $t(+|\mathbf{x}) = 1$, czyli sytuacja, w której wszystkie przykłady powinny mieć pewność bycia klasyfikowanymi jako mniejszościowe – i widać, że wykonana implementacja daje wyniki bliskie jedności. *Wolfram|Alpha* nie podaje tutaj skończonych wartości ze względu na występujące w równaniu (3.27) dzielenie przez $\tau = 0$.

5.1.2 Analiza charakteru dodawanych przykładów

W celu analizy działania minimalizacji różnicy między wartością oczekiwaną kosztowego prawdopodobieństwa warunkowego $t(+|\mathbf{x})$ a obserwowanym prawdopodobieństwem *a posteriori* przez zaimplementowany algorytm *COST*, przeprowadzono szereg eksperymentów, polegających na przefiltrowaniu zestawu jedno- i dwuwymiarowych zbiorów danych. Wyniki eksperymentów zostały opisane poniżej.

Zbiory jednowymiarowe



Rysunek 5.1: Efekt filtrowania jednowymiarowego zbioru *onedim1* metodą *COST* z sąsiedztwem 5-NN, dla kosztu $\tau = 0{,}2$

Na wykresie z rys. 5.1 przedstawiono efekt filtrowania prostego jednowymiarowego sztucznego zbioru danych *onedim1*. Filtrowany zbiór składał się z 33 przykładów, z czego 11 mniejszościowych i 22 większościowych.

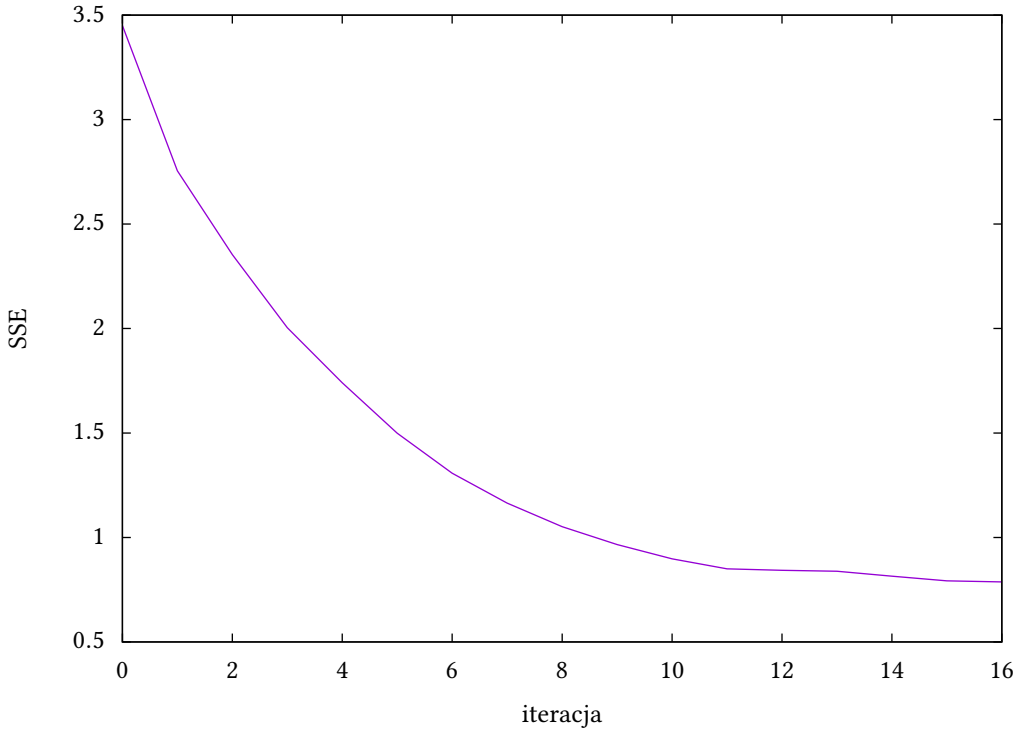
Przykłady mniejszościowe są skupione wokół wartości atrybutu 50, choć kilka przykładów tej klasy przyjmuje wartości wyraźnie mniejsze. Jeden mniejszościowy przykład odstający (ang. *outlier*) ma wartość 85.

Przykłady większościowe znajdują się po obu stronach regionu mniejszościowy, przyjmując wartości z zakresu $[0;\ 40] \cup [60;\ 100]$.

Na wykresie przedstawiono wartości prawdopodobieństwa *a posteriori* $\hat{p}(+|\mathbf{x})$ przed filtrowaniem i po nim, obserwowane w sąsiedztwach punktów z oryginalnymi przykładami. Dodatkowo linia zielona przedstawia wartości docelowe, do których algorytm starał się dopasować prawdopodobieństwa *a posteriori*.

Można z tego wykresu zauważyć, że niskie wartości prawdopodobieństwa $\hat{p}(+|\mathbf{x})$ zostają po filtrowaniu podniesione, to wynik niskiej wartości τ (czyli niskiego kosztu FP w stosunku do FN) – klasa mniejszościowa zostaje wzmocniona. Ze względu na niewielką pewność obserwowanego $\hat{p}(+|\mathbf{x})$ (sąsiedztwa składające się z jedynie pięciu instancji), algorytm próbuje złagodzić wartości ekstremalne – widać to w punktach, w których $\hat{p}(+|\mathbf{x}) = 1$ – wartość docelowa prawdopodobieństwa jest tam niższa. Równocześnie można zauważyć, że w miejscach, w których oryginalne obserwowane $\hat{p}(+|\mathbf{x})$ przyjmuje wartości ekstremalne (1 lub 0), algorytm nie jest w stanie zmienić tej sytuacji. Wynika to z tego, że wszystkie przykłady w takim sąsiedztwie należą do tej samej klasy i brakuje w nim przykładów klasy przeciwnej do powielenia, by przesunąć wartość *a posteriori* w stronę mniejszej pewności (zmniejszyć, gdy równa 1, zwiększyć, gdy równa 0).

Kolejną obserwacją jest, że obserwowane prawdopodobieństwa *a posteriori* przyjmują wartości skwantowane, gdy obliczane są w sąsiedztwach o jednakowej liczności przykładów – wśród $\hat{p}(+|\mathbf{x})$ obserwowanych w oryginalnym zbiorze tylko raz wartość niebędąca wielokrotnością $\frac{1}{5} = 0{,}2$ – oznacza to, że w jej sąsiedztwie musiały się znaleźć przykłady o identycznej wartości atrybutu, przez co sąsiedztwo 5-NN było w rzeczywistości większe niż 5 sąsiadów. Widać to wyraźnie wśród $\hat{p}(+|\mathbf{x})$ obserwowanych już po filtrowaniu – wiele przykładów zostało powielonych, zatem liczności sąsiedztw zwiększają się i obserwowane prawdopodobieństwa warunkowe klasy mniejszościowej często przyjmują w nich wartości spoza wielokrotności $\frac{1}{5}$.



Rysunek 5.2: Wartość błędu SSE w kolejnych iteracjach filtrowania jednowymiarowego zbioru *onedim1* metodą *COST* z sąsiedztwem 5-NN, dla kosztu $\tau = 0{,}2$

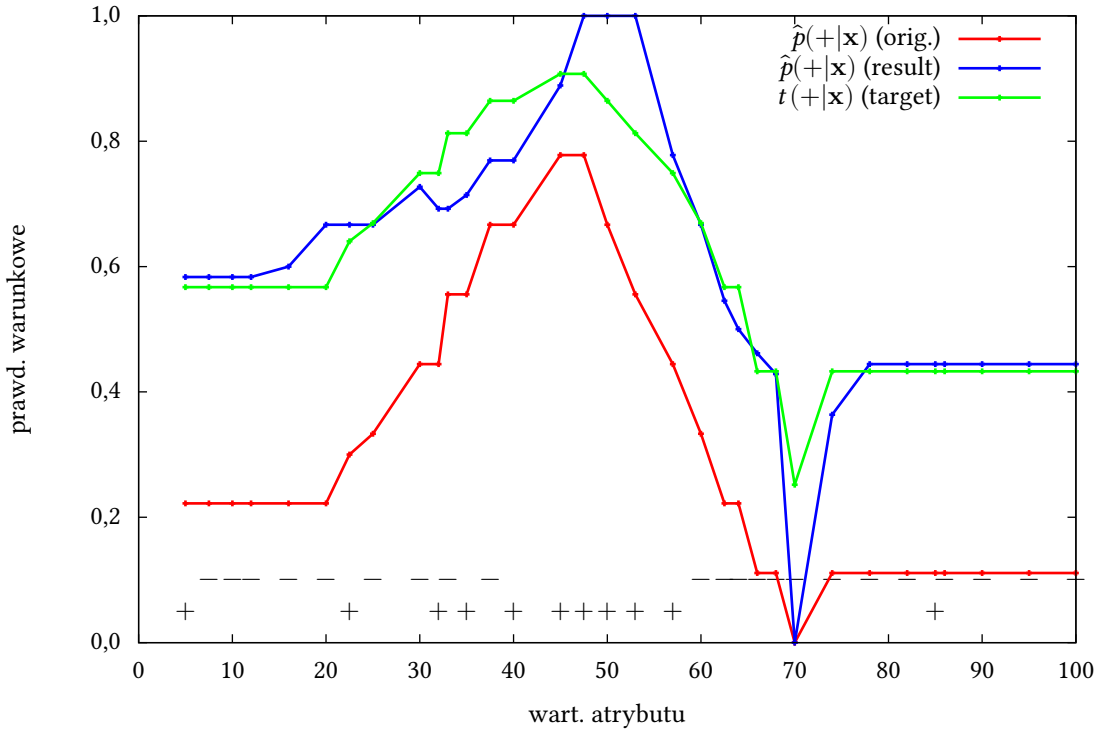
Algorytm podczas filtrowania tego zbioru skopiował 16 instancji w 16 iteracjach.

Wartość sumy kwadratów błędów (SSE) w kolejnych iteracjach przedstawia wykres 5.2 (iteracja 0 oznacza wartość błędu przed pierwszym nadlosowaniem). Widać poprawne stałe zmniejszanie błędu. Ze względu na wspomniane wcześniej występowanie sąsiedztw, w których wartość $\hat{p}(+|\mathbf{x})$ jest niemożliwa do zmiany, występuje pewna minimalna wartość błędu, poniżej której nie da się zejść za pomocą prostego kopiowania, i bez usuwania, istniejących przykładów. Ta minimalna wartość dla danego zbioru może być obliczona w sposób:

$$e_{\min} = \sum_{\mathbf{x}_\mathrm{T}\,:\,\hat{p}(+|\mathbf{x}_\mathrm{T})\in\{0;\,1\}} \left(t\left(+|\mathbf{x}_\mathrm{T}\right) - \hat{p}(+|\mathbf{x}_\mathrm{T})\right)^2, \tag{5.1}$$

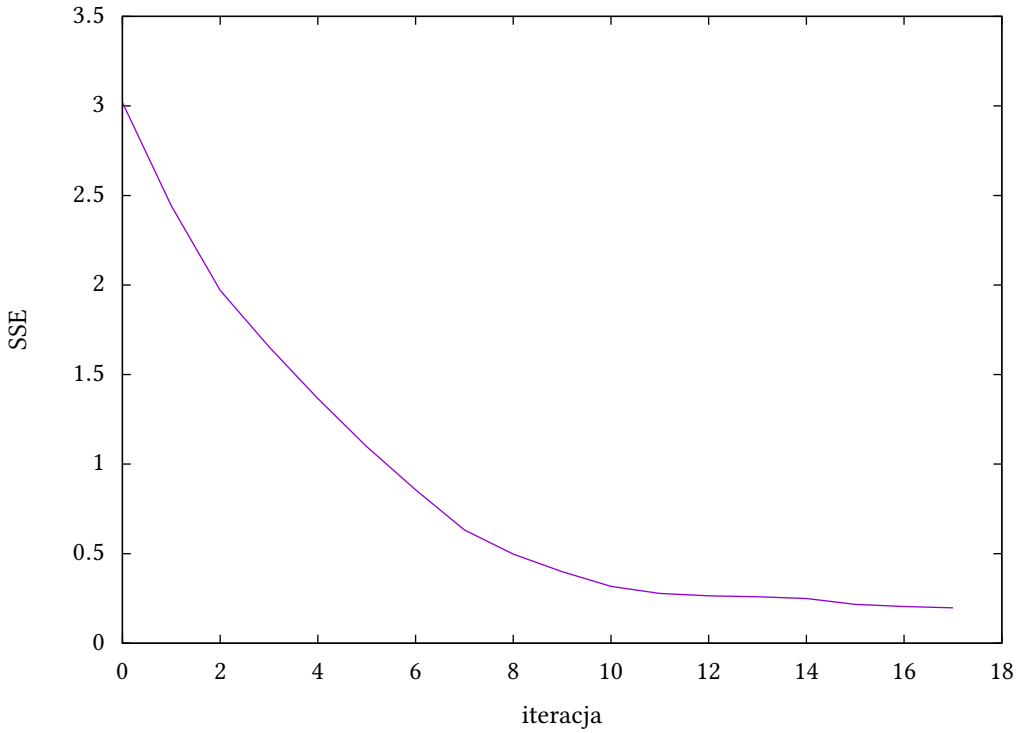
tj. sumując oryginalne wartości błędu dla wszystkich sąsiedztw, w których prawdopodobieństwo *a posteriori* równe jest jedności lub zeru. W przypadku przedstawianym przez rys. 5.1 i 5.2 wartość ta wynosiła $e_{\min} = 0{,}603$.

Na rys. 5.3 przedstawiono wykres, pokazujący efekt filtrowania tego samego zbioru, ale z sąsiedztwem poszerzonym do dziewięciu najbliższych sąsiadów (9-NN).



Rysunek 5.3: Efekt filtrowania jednowymiarowego zbioru *onedim1* metodą *COST* z sąsiedztwem 9-NN, dla kosztu $\tau = 0{,}2$

Widać na nim, że zmniejszyły się minimalne różnice między wartościami $\hat{p}(+|\mathbf{x})$ w różnych sąsiedztwach, zgodnie z oczekiwaniami prawdopodobieństwo warunkowe przyjmuje teraz wartości, będące wielokrotnościami $\frac{1}{9}$. Ze względu na większą liczbę sąsiadów rzadziej występują wartości ekstremalne obserwowanego prawdopodobieństwa klasy mniejszościowej (jedynie w punkcie odpowiadającym wartości 70 atrybutu występuje $\hat{p} = 0$), a co za tym idzie, zmniejszyła się też minimalna możliwa do uzyskania wartość błędu, i wynosi dla tego przypadku $e_{\min} = 0{,}064$.



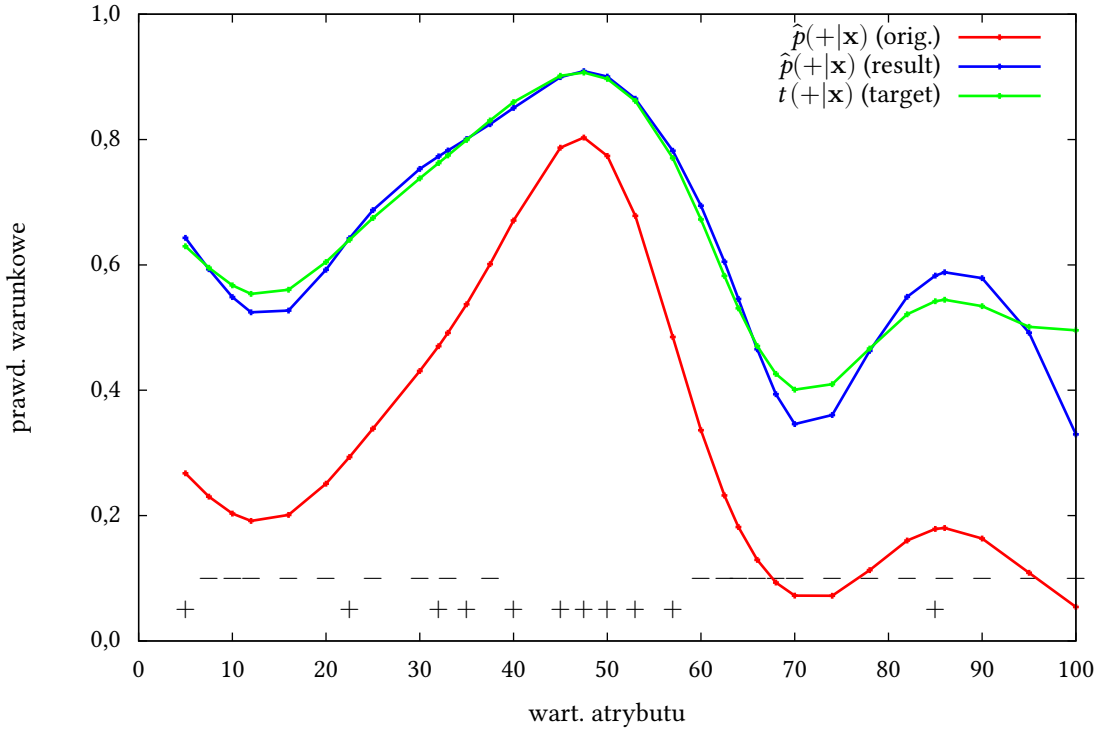
Rysunek 5.4: Wartość błędu SSE w kolejnych iteracjach filtrowania jednowymiarowego zbioru *onedim1* metodą *COST* z sąsiedztwem 9-NN, dla kosztu $\tau = 0{,}2$

Wykres z rys. 5.4 przedstawia zmianę błędu przy filtrowaniu z sąsiedztwem 9-NN. Algorytm w tym przypadku wykonał 17 iteracji.

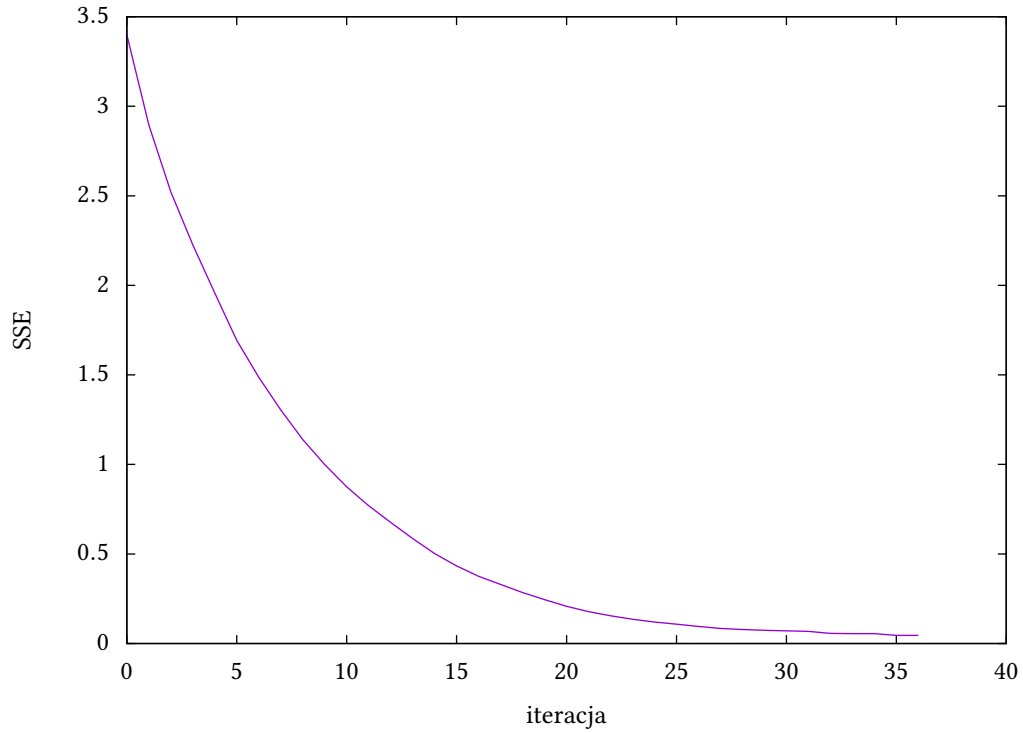
Na rys. 5.5 widać wykres pokazujący efekt filtrowania zbioru *onedim1* ze statystykami sąsiedztw uzyskiwanymi poprzez estymację gęstości funkcją jądrową Gaussa, jako macierz kowariancji atrybutu została użyta domyślna macierz, generowana wg równania (4.4) na podstawie podanej wartości odchylenia standardowego σ. W tym wypadku podano $\sigma = 0{,}08$, czyli macierz kowariancji w tym jednowymiarowym przypadku miała postać $\Sigma = [0{,}64]$, przy czym należy pamiętać, że wartości atrybutów na potrzeby analizowanej implementacji *COST* normalizowane są do wartości z zakresu $[0;\ 1]$.

Jak widać z wykresu, w przypadku używania jądra Gaussa do uzyskiwania statystyk sąsiedztw, nie pojawiają się sąsiedztwa, w których niemożliwa jest zmiana wartości obserwowanej prawdopodobieństwa klasy mniejszościowej poprzez duplikowanie przykładów. Dzięki temu uzyskuje się również wyraźnie większy spadek wartości błędu niż w przypadku sąsiedztw k-NN, co potwierdza wykres z rys. 5.6. Niestety, uzyskanie niższego błędu wymagało również wykonania większej liczby iteracji, dodano 36 przykładów.

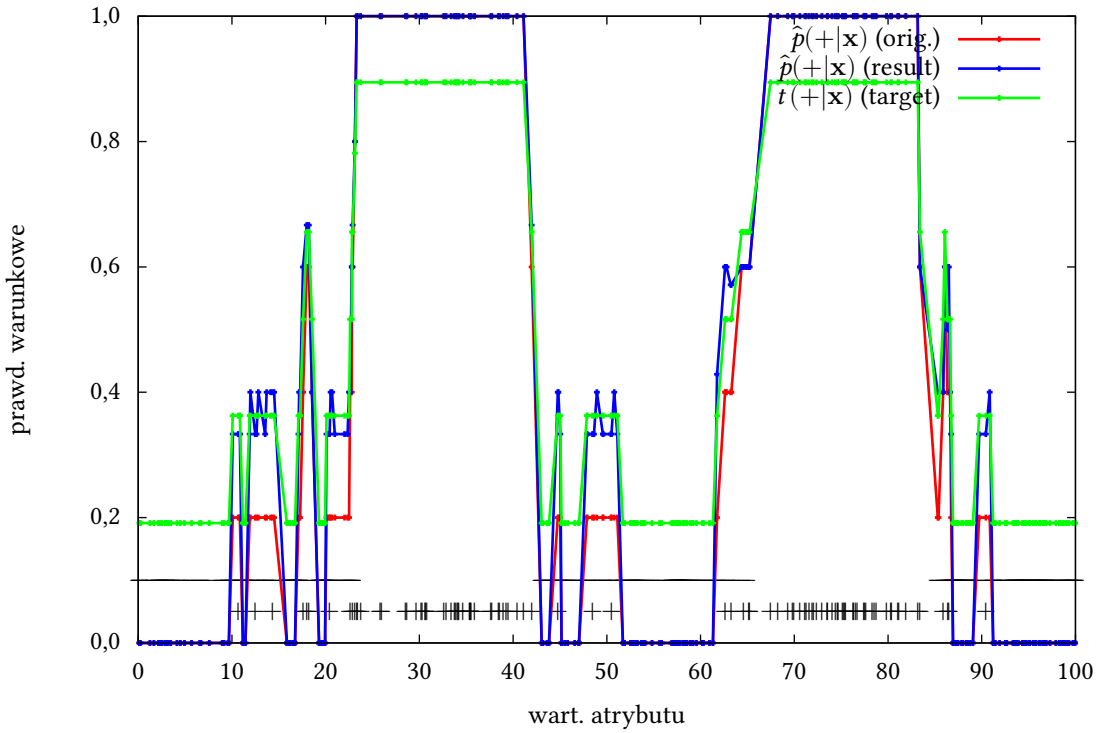
Na rysunkach 5.7 i 5.8 przedstawiono wykresy uzyskane po filtrowaniu sztucznego jednowymiarowego zbioru *onedim2*, utworzonego z grup 200 przykładów większościowych, rozmieszczonych po 66–67 przykładów równomiernie w przedziałach $[0;\ 23]$, $[42;\ 65]$ i $[85;\ 100]$ oraz 100 przykładów mniejszościowych, umieszczonych losowo wg dwóch rozkładów normalnych, jednego o $\mu = 30$ i $\sigma = 10$, drugiego o $\mu = 75$ i $\sigma = 6{,}7$, w efekcie obszary obu klas wyraźnie na siebie nachodzą.



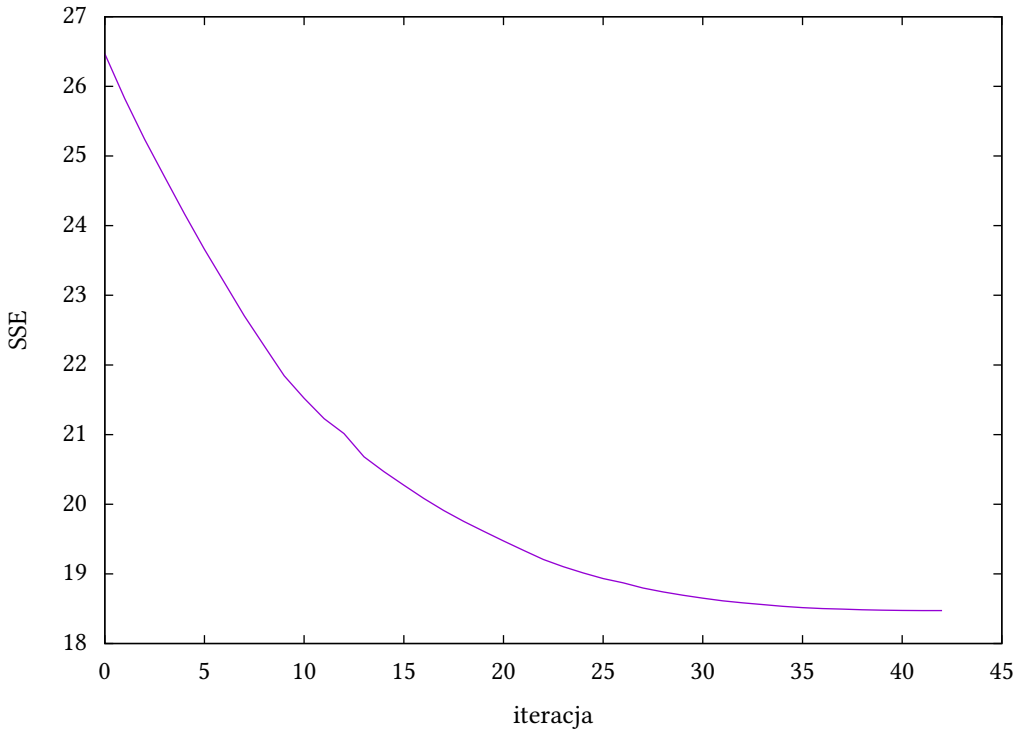
Rysunek 5.5: Efekt filtrowania jednowymiarowego zbioru *onedim1* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}08$, $\tau = 0{,}2$



Rysunek 5.6: Wartość błędu SSE w kolejnych iteracjach filtrowania jednowymiarowego zbioru *onedim1* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}08$, $\tau = 0{,}2$



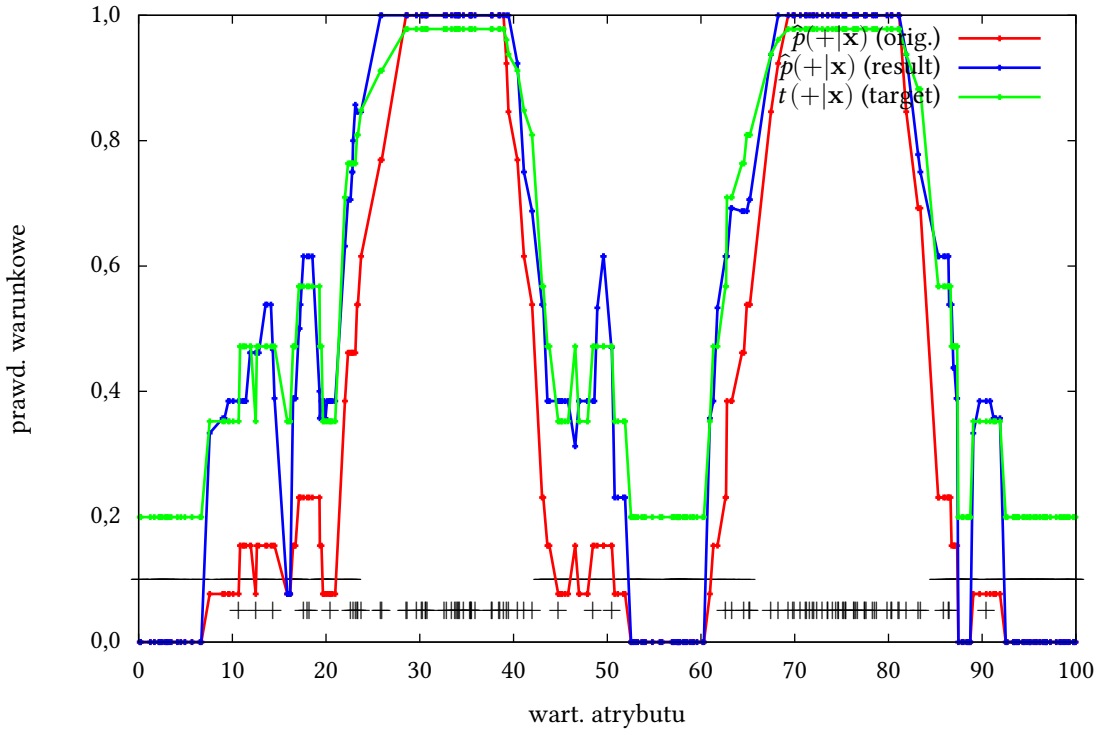
Rysunek 5.7: Efekt filtrowania jednowymiarowego zbioru *onedim2* metodą *COST* z sąsiedztwem 5-NN, $\tau = 0{,}2$



Rysunek 5.8: Wartość błędu SSE w kolejnych iteracjach filtrowania jednowymiarowego zbioru *onedim2* metodą *COST* z sąsiedztwem 5-NN, $\tau = 0{,}2$

Nachodzenie klas przy dość dużym zagęszczeniu przykładów powoduje, że sąsiedztwo 5-NN staje się w tym wypadku mało skuteczne. Wykres $\hat{p}(+|\mathbf{x})$ jest mocno poszarpany, w obszarach, gdzie klasy na siebie nachodzą, wartości *a posteriori* skaczą z zera (gdzie są niemożliwe do zmiany) do wyższych wartości i z powrotem.

Jak widać na wykresie błędu, algorytm wykonał tu 42 iteracje i obniżył sumę kwadratów błędów do ok. 18,5. Wartość minimalna wynosiła w tym przypadku $e_{\min} = 18{,}127$.

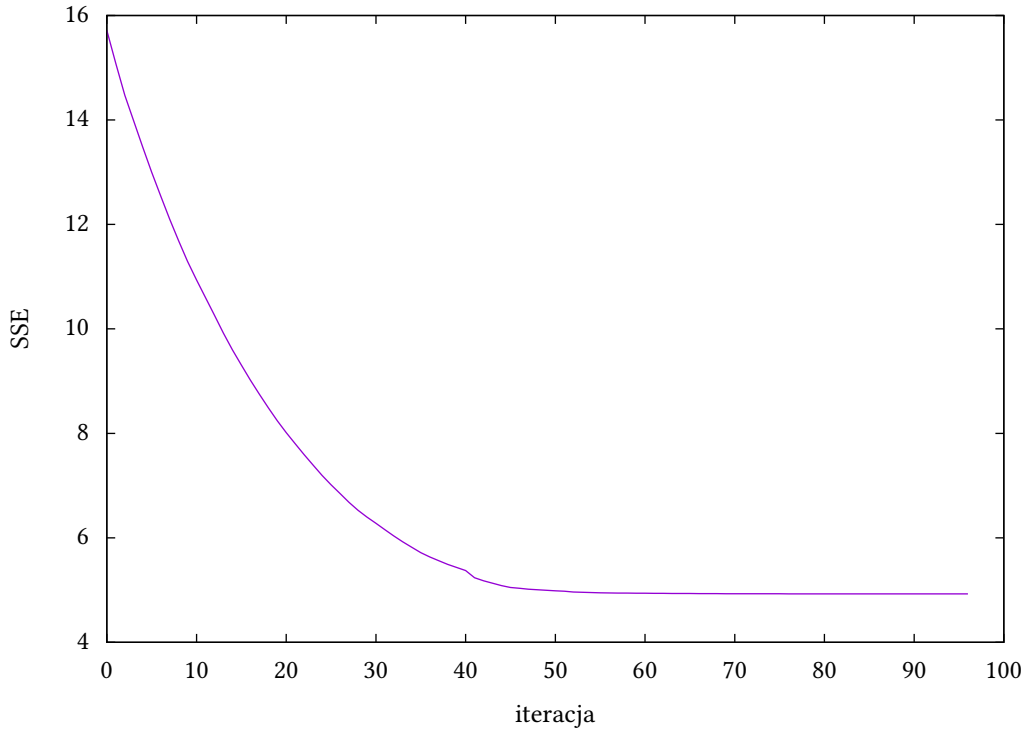


Rysunek 5.9: Efekt filtrowania jednowymiarowego zbioru *onedim2* metodą *COST* z sąsiedztwem 13-NN, $\tau = 0{,}2$

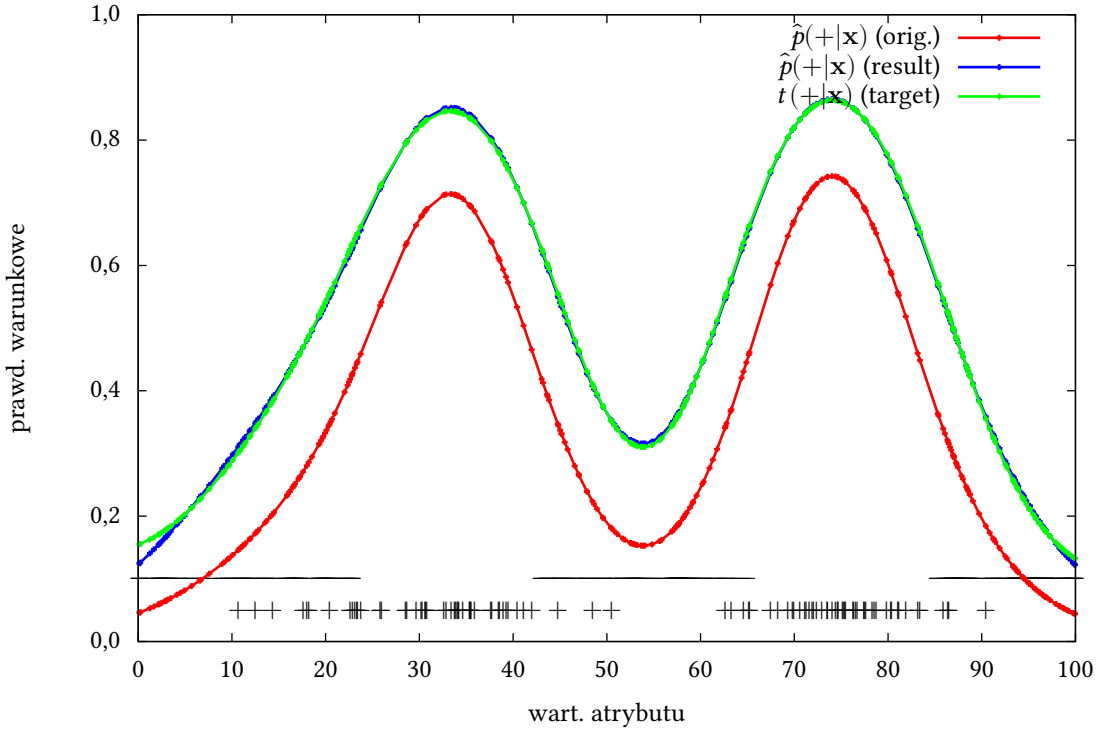
Lepszy efekt udało się uzyskać dla tego zbioru za pomocą większego sąsiedztwa – 13-NN. Przypadek ten pokazują wykresy z rys. 5.9 i 5.10. Minimalna wartość błędu wynosiła $e_{\min} = 4{,}044$. Porównując wykres efektu działania *COST* w tym przypadku oraz dla sąsiedztwa 5-NN (rys. 5.7), jak liczba przykładów (a za tym wiarygodność estymacji) wpływa na wartość docelową – dla sąsiedztwa 11-NN w miejscach, gdzie $\hat{p} = 1$, algorytm ustala wartość docelową $t(+|\mathbf{x})$ wyraźnie bliżej jedności, niż w przypadku sąsiedztwa 5-NN.

Zbiór *onedim2* przefiltrowano również z sąsiedztwami uzyskanymi estymacją gęstości, podobnie jak przypadku zbioru *onedim1* z parametrem $\sigma = 0{,}08$, z kosztem $\tau = 0{,}3$ (wartość na tyle odbiegająca od 0,5, że pokazuje jak zachowuje się filtr dla tego zbioru, a jednocześnie pozwoliła na szybkie uzyskanie wynika przy relatywnie niskiej liczbie iteracji).

Wynik tego filtrowania przedstawia wykres z rysunku 5.11. Jak widać, przy tym sąsiedztwie algorytm radzi sobie niemal doskonale – rys. 5.12 przedstawia wykres błędu w kolejnych iteracjach, widać z niego, że wartość błędu spada prawie do zera. Algorytm wykonał 151 iteracji, w tym 146 powieleń przykładów i 6 usunięć skopiowanych

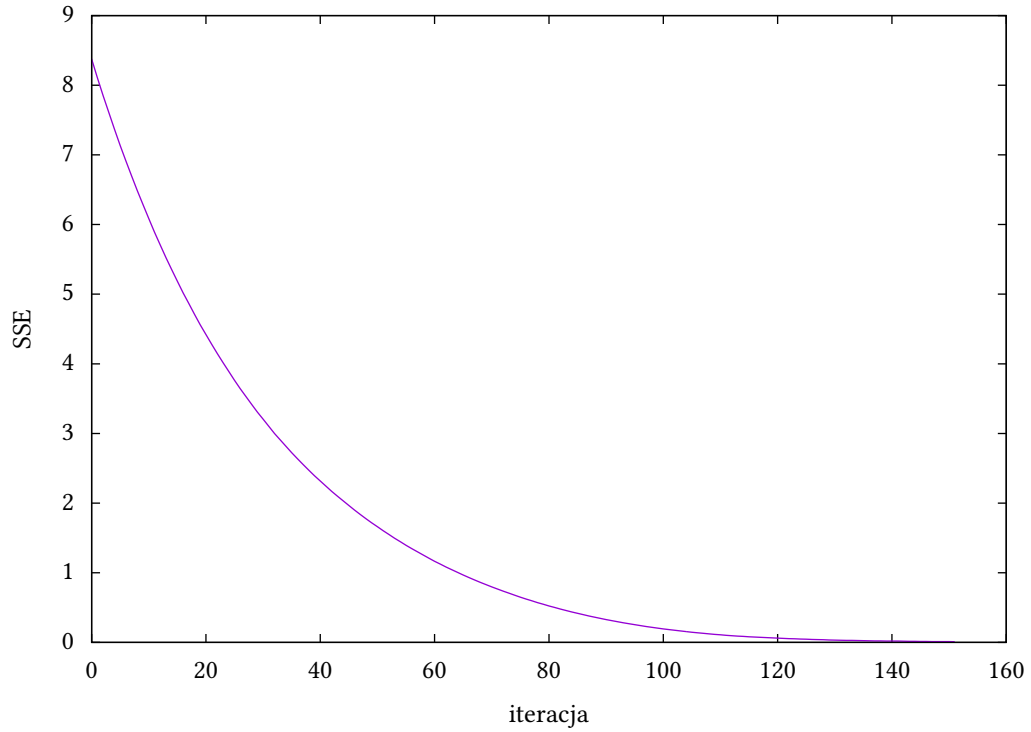


Rysunek 5.10: Wartość błędu SSE w kolejnych iteracjach filtrowania jednowymiarowego zbioru *onedim2* metodą *COST* z sąsiedztwem 13-NN, $\tau = 0{,}2$



Rysunek 5.11: Efekt filtrowania jednowymiarowego zbioru *onedim2* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}08$, $\tau = 0{,}3$

wcześniej przykładów.



Rysunek 5.12: Wartość błędu SSE w kolejnych iteracjach filtrowania jednowymiarowego zbioru *onedim2* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}08$, $\tau = 0{,}3$

Ze względu na brak sąsiedztw o całkowitym udziale jedynie przykładów z jednej klasy (ponieważ przy estymacji gęstości do obliczania sąsiedztw dodaje się udział *każdego* przykładu w zbiorze), nie da się dla tego sąsiedztwa prosto określić granicy minimalnego błędu e_{min}, niemożliwej do pokonania duplikowaniem istniejących przykładów – mimo to widać, że taka również istnieje dla tego sąsiedztwa – dla wartości atrybutu z zakresu ok. $[0; 3]$ wynikowe obserwowane wartości $\hat{p}(+|\mathbf{x})$ wyraźnie odstają od docelowych $t(+|\mathbf{x})$, ponieważ w obszarze tym brakuje przykładów klasy mniejszościowej do powielenia, a powielanie przykładów bardziej oddalonych nie modyfikuje odpowiednio kształtu wykresu.

Zbiory dwuwymiarowe

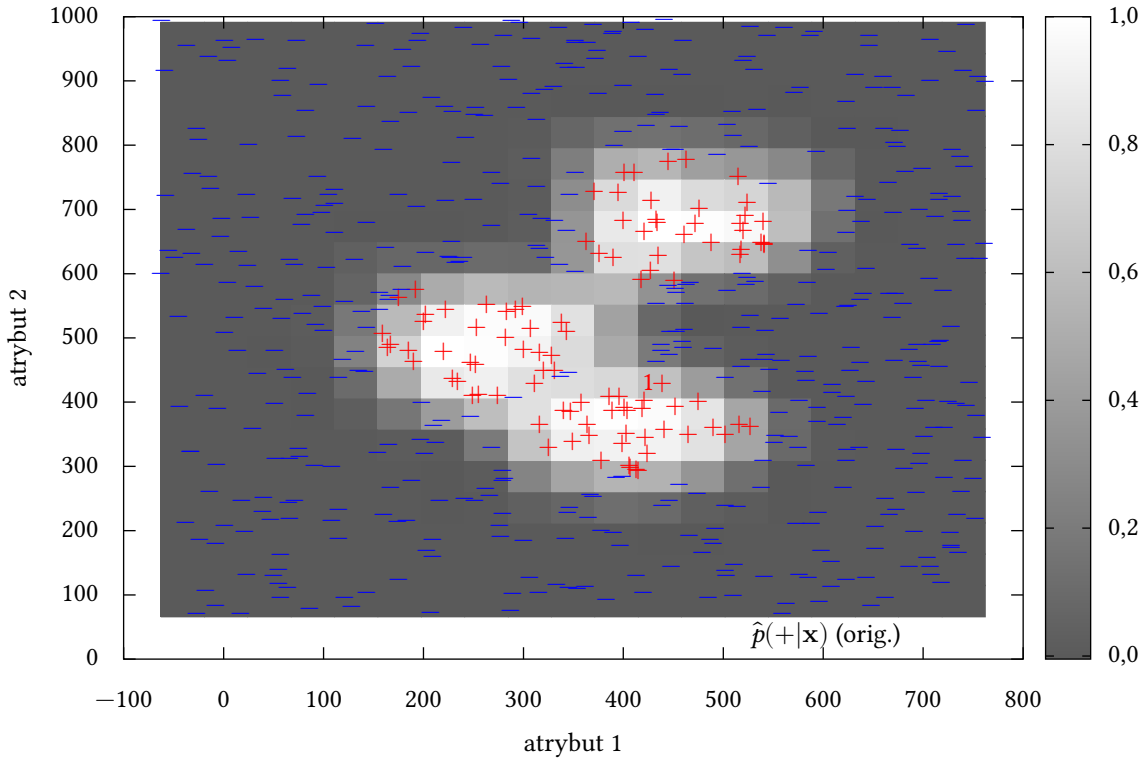
Na wykresie z rys. 5.13 widać efekt filtrowania metodą *COST* zbioru o dwóch atrybutach liczbowych, z 600 przykładami, w tym 100 mniejszościowymi. Obszary klas w tym zbiorze są odseparowane – nie nachodzą na siebie, ani nie występuje w nich szum. Na wykresie jasność tła w odcieniach szarości odpowiada wartościom obserwowanego prawdopodobieństwa klasy mniejszościowej w sąsiedztwach – wartości z sąsiedztw punktów odpowiadających przykładom ze zbioru zostały interpolowane funkcjami sklejanymi do siatki 20×20 wartości na potrzeby umieszczenia ich na wykresie. Przykłady klasy mniejszościowej zostały oznaczone czerwonymi symbolami $+$, przykłady klasy większościowej niebieskimi symbolami $-$. Przykłady, które zostały powielone zaznaczone są liczbą, oznaczającą ile razy dany przykład został powielony, w kolorze odpowiadającym jego klasie (czerwony dla klasy mniejszościowej, niebieski dla klasy większościowej).

Rysunek 5.14 przedstawia wykres zmiany błędu podczas filtrowania zbioru *02a-600-5-0-BI*, widać z niego, że algorytm przeprowadził tylko jedną iterację, dodał jeden przykład mniejszościowy (co widać na rys. 5.13). Wynika to stąd, że filtrowano z kosztem $\tau = 0{,}5$, co oznacza równy koszt FP i FN, algorytm zatem nie powinien ingerować zbytnio w ułożenie klas. Przeprowadzona zmiana wynika ze zmniejszania pewności wartości prawdopodobieństwa przez algorytm – nawet dla $\tau = 0{,}5$ wartości $t(+|\mathbf{x})$ nie odpowiadają dokładnie wartościom $\hat{p}(+|\mathbf{x})$.

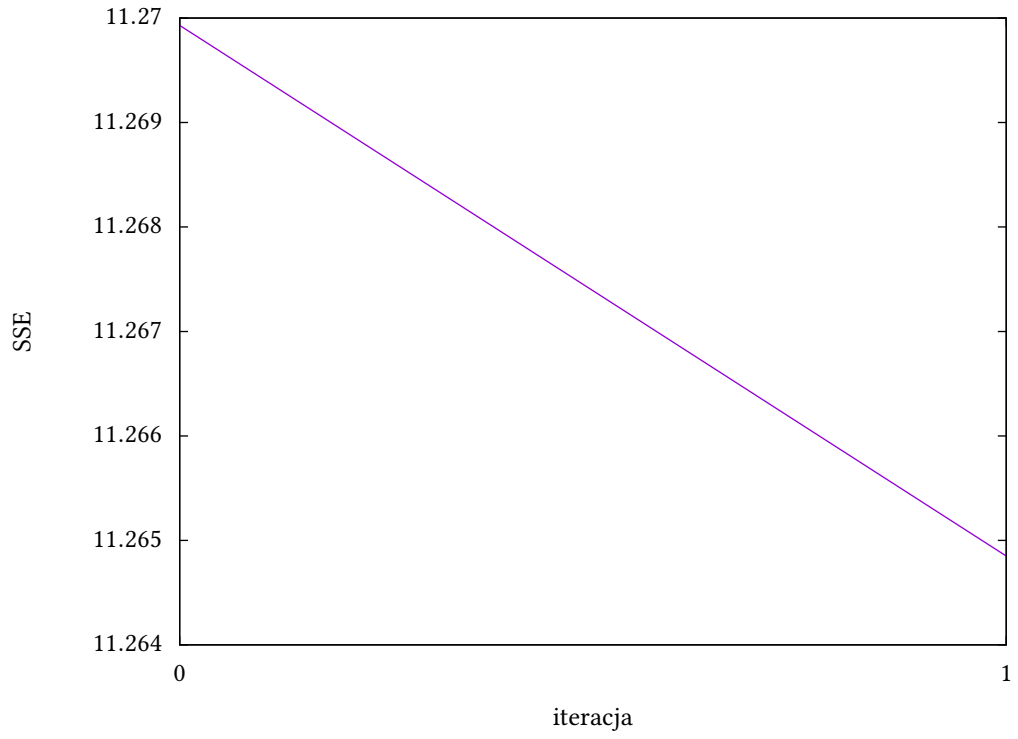
Analogiczny eksperyment przeprowadzono z sąsiedztwem z estymowanymi gęstościami przykładów, co prezentują wykresy z rys. 5.15 i 5.16. Jak łatwo z nich zauważyć, dodano w tym przypadku dużo więcej przykładów. Wynika to stąd, że sąsiedztwo z jądrową estymacją gęstości pozwala dużo lepiej zmniejszać odmienność rozkładu obserwowanego od docelowego, więc niewielkie zmiany wymagane dla osiągnięcia wartości oczekiwanych kosztowego prawdopodobieństwa w sąsiedztwach dla jednakowych kosztów obu klas powodują i tak wykonanie 51 iteracji. Dodatkowo z wykresu błędu widać, że algorytm trafia na pewno minimum lokalne ze względnie wysoką wartością błędu, bo nie schodzi ona nawet poniżej 0,82.

Dodanie lub usunięcie jednego przykładu w tym wypadku trwało ok. 0,5 s. Ze względu na dużą liczbę dodawanych przykładów i czas na to potrzebny nie wykonano eksperymentów z jądrową estymacją w przypadku dwuwymiarowych danych dla $tau < 0{,}5$. Próba takiego filtrowania dla $\tau = 0{,}4$ nie zakończyła się po 24 godzinach, co oznacza, że algorytm wykonał ponad 150 tysięcy iteracji i kontynuował działanie. Trwało to tak długo mimo optymalizacji dla tego typu sąsiedztwa, opisanej w sekcji 4.3, na stronie 24. Dla porównania filtrowanie z sąsiedztwem 5-NN potrzebowało ok. pół minuty na dodanie jednej instancji, algorytm jednak wykonywał dla niego znacznie mniej operacji, w efekcie czego przefiltrowanie zbioru dla niższego τ z tym sąsiedztwem trwało znacznie krócej.

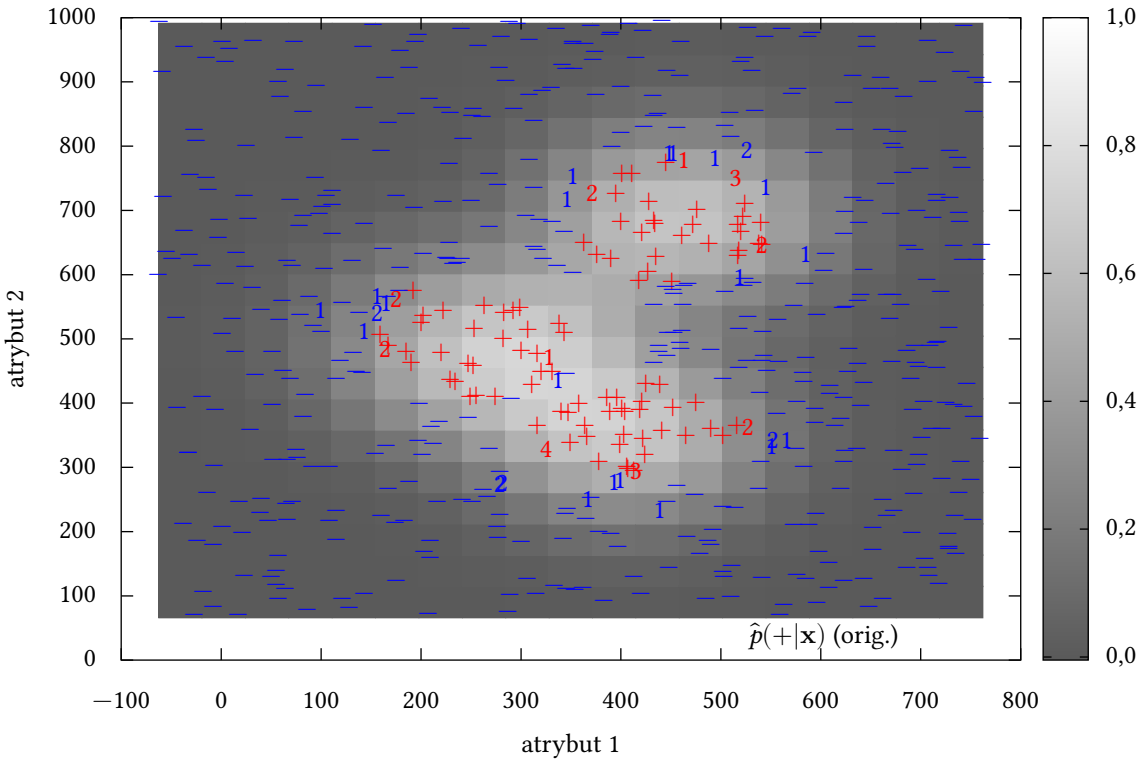
Na wykresach z rys. 5.17 przedstawiono zmiany błędów w kolejnych iteracjach filtrowania zbioru *02a-600-5-0-BI* z kosztem $\tau = 0{,}3$ z sąsiedztwami 5-NN i 9-NN. Widać z nich, że większe sąsiedztwo pozwala na lepsze zmniejszenie wartości błędu (wynika to m.in. z tego, że dla większych sąsiedztw mniej jest przypadków, gdzie $\hat{p} \in \{0;\ 1\}$, w któ-



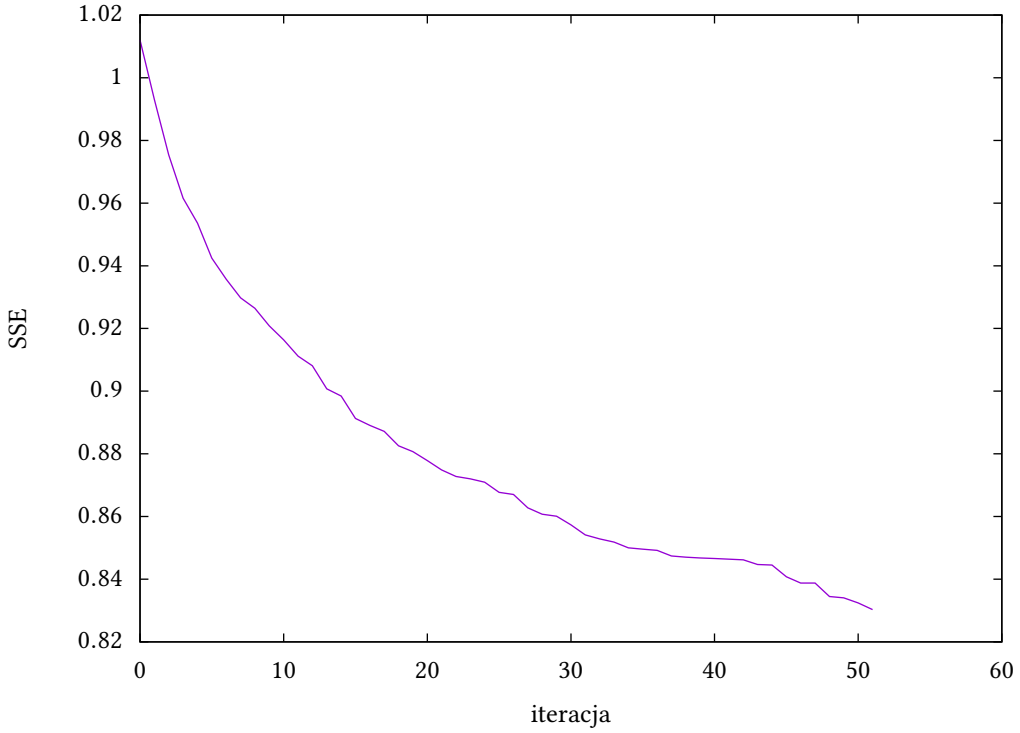
Rysunek 5.13: Efekt filtrowania zbioru *02a-600-5-0-BI* metodą *COST* z sąsiedztwem 5-NN, dla kosztu $\tau = 0{,}5$



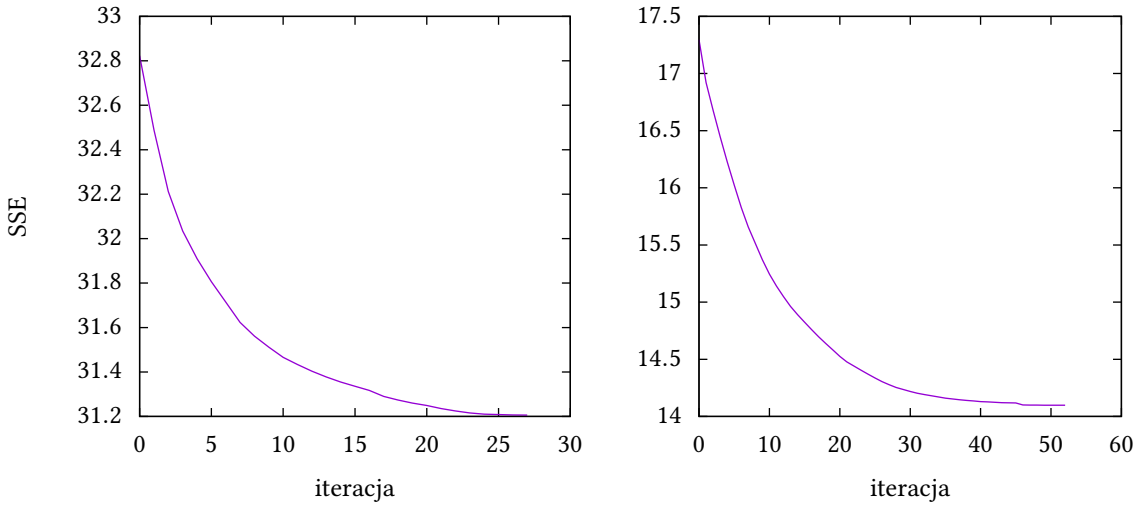
Rysunek 5.14: Wartość błędu SSE w kolejnych iteracjach filtrowania zbioru *02a-600-5-0-BI* metodą *COST* z sąsiedztwem 5-NN, dla kosztu $\tau = 0{,}5$



Rysunek 5.15: Efekt filtrowania zbioru *02a-600-5-0-BI* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}08$, $\tau = 0{,}5$



Rysunek 5.16: Wartość błędu SSE w kolejnych iteracjach filtrowania zbioru *02a-600-5-0-BI* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}08$, $\tau = 0{,}5$



Rysunek 5.17: Wartość błędu SSE w kolejnych iteracjach filtrowania zbioru *02a-600-5-0-BI* metodą *COST* z sąsiedztwem 5-NN (po lewej) i 9-NN (po prawej), $\tau = 0{,}3$

rych nie da się zmienić statystyki). Dla większego sąsiedztwa jednak algorytm wykonał prawie dwa razy więcej iteracji.

5.2 Klasyfikacja przetworzonych zbiorów

Pełniejsze eksperymenty zostały przeprowadzone na wybranych 5 zbiorach rzeczywistych danych oraz jednym zbiorze utworzonym sztucznie.

Każdy ze zbiorów rzeczywistych został najpierw losowo podzielony na 10 par – zbiór uczący i testowy – na potrzeby 10-krotnej warstwowej kroswalidacji. Zbiór utworzony sztucznie miał już przygotowany podział na zbiór uczący i testowy.

Następnie poddano każdy ze zbiorów uczących przetwarzaniu następującymi filtrami:

- **COST** – opisana w tej pracy implementacja *Cost-Optimized Sampling Technique* z sąsiedztwem k-NN, z parametrami $k \in \{5, 9\}$, $\tau \in \{0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5\}$,
- **ROS** – *Random Oversampling*, nadlosowanie przykładów klasy mniejszościowej do wyrównania liczności obu klas decyzyjnych,
- **NCR** – *Neighbourhood Cleaning Rule* z sąsiedztwem 5-NN,
- **SPIDER** – metoda *SPIDER* z sąsiedztwem 5-NN,
- **SMOTE** – dogenerowywanie przykładów metodą *SMOTE* z sąsiedztwem 5-NN do wyrównania liczności klas.

A później każdy z uzyskanych zestawów zbiorów poddano 10-krotnej kroswalidacji, poza wspomnianym zbiorem sztucznym, na którym przeprowadzono pojedynczy eksperyment klasyfikacji przykładów ze zbioru testowego, za pomocą kilku klasyfikatorów.

5.2.1 Charakterystyka wykorzystanych zbiorów

Do przeprowadzenia eksperymentów klasyfikacji użyto następujących zbiorów danych:

- **paw-4-600-5-30-10-10-0-U1** – utworzony sztucznie trudny zbiór o dwóch liczbowych atrybutach, 600 przykładach, z tego 100 w klasie mniejszościowej. 30 % przykładów mniejszościowych znajdowało się na granicy obu klas, 10 % przykładów rzadkich (zgrupowanych z kilkoma innymi z klasy mniejszościowej, ale otoczonych klasą większościową), 10 % było przykładami odstającymi (pojedynczymi przykładami klasy mniejszościowej wśród przykładów większościowych). Zbiorowi temu towarzyszył zbiór testowy *paw-4-600-5-30-10-10-0-T1* z 300 przykładami o podobnym rozkładzie. Rozmieszczenie przykładów tego zbioru można obejrzeć na wykresie z rys. 5.18, prezentującym efekt jego filtrowania. Zbiór ten pochodzi z [Nap12].
- **new-thyroid** – dość łatwy zbiór o pięciu atrybutach numerycznych. Opisuje on stan pacjentów z podziałem na chorych na nadczynność tarczycy i innych. Składa się z 215 przykładów, w tym 35 w klasie mniejszościowej.
- **haberman** – zbiór opisujący przeżywalność pacjentów po operacji raka piersi w szpitalu Billings Hospital Uniwersytetu Chicagowskiego w latach 1958–1970, przykłady opisane są trzema liczbowymi atrybutami (wiek w momencie operacji, rok, w którym operacja się odbyła, liczba węzłów chłonnych, do których dostały się komórki rakowe), przynależność do klasy większościowej mówi o tym, że pacjent przeżył co najmniej 5 lat po operacji, zaklasyfikowani do klasy mniejszościowej nie dożyli piątego roku po operacji. Zbiór ma 306 przykładów, w tym 81 z klasy mniejszościowej.
- **ecoli** – zbiór z dziedziny biologii molekularnej, opisujący położenie białek w bakteriach *e. coli*, przykłady opisane są 7 atrybutami liczbowymi. Składa się z 336 przykładów, z czego 35 należy do klasy mniejszościowej.
- **transfusion** – dość duży zbiór z czterema atrybutami liczbowymi. Składa się z 748 przykładów, w tym 178 z klasy mniejszościowej.
- **yeast-ME2** – największy z filtrowanych zbiorów, z największym stopniem niezrównoważenia. Przykłady opisane są ośmioma atrybutami liczbowymi. Składa się z 1484 przykładów, w tym zaledwie 51 mniejszościowych.

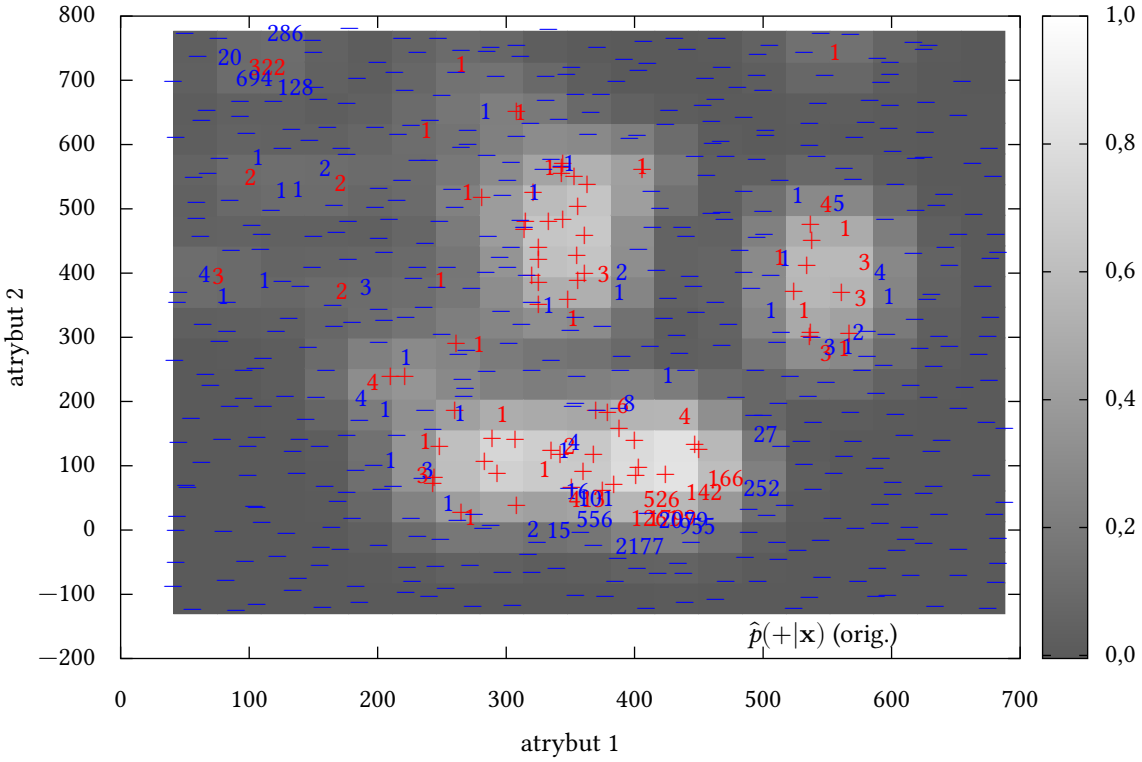
5.2.2 Wynik przetwarzania wstępnego

Każdy ze zbiorów poddano, jak napisano wyżej, filtrowaniu pięcioma różnymi metodami przetwarzania wstępnego. Dla metody *COST* przeprowadzono filtrowanie z sąsiedztwem 5-NN i 9-NN, oraz dla wartości kosztu τ w zakresie od 0,1 do 0,5 z krokiem co 0,1.

Podczas, gdy algorytmy *SMOTE* i *Random Oversampling* wyrównywały liczność obu klas, *COST* dodawał mniejszą liczbę przykładów, mocno zależą od zadanego kosztu τ – im wartość bardziej odbiegała od 0,5, tym więcej przykładów zostawało powielonych. Było tak przynajmniej dla sąsiedztw typu *k* najbliższych sąsiadów. Dla sąsiedztwa 5-NN i $\tau = 0{,}3$ każdy z 10 zbiorów uczących powstałych ze zbioru *haberman* miał powielonych średnio 73,3 przykładu, przy czym nie wszystkie były z klasy mniejszościowej – uzyskane zbiory miały średnio 344,2 przykładu, w tym 132,6 z klasy mniejszościowej, co daje 38,5 % przykładu z klasy mniejszościowej. Dla $\tau = 0{,}1$ uzyskiwano przekroczenie zrównoważenia i doprowadzenie do sytuacji, w której przykłady mniejszościowe stanowiły większość przykładów w zbiorze.

Podobnie zachowywały się pozostałe zbiory dla sąsiedztw typu k-NN, choć to, jak wyrównane będą liczności klas zależało mocno od początkowego stopnia niezrównoważenia. Nawet po filtrowaniu z $\tau = 0{,}1$ w zbiorze *yeast-ME2* przykłady mniejszościowe stanowiły jedynie ok. 14 %.

Dodatkowo udało się przefiltrować dwuwymiarowy zbiór *paw-4-600-5-30-10-10-0-U1* z użyciem jądrowej estymacji gęstości za pomocą funkcji Gaussa, z diagonalną macierzą kowariancji o wartościach σ^2 na głównej przekątnej dla $\sigma \in \{0{,}04; 0{,}08\}$ dla kosztu $\tau = 0{,}5$. Dla $\sigma = 0{,}08$ wykonano 33 iteracje, dla $\sigma = 0{,}04$ wykonano iteracji aż 12 607. Wykres z rys. 5.18 przedstawia wynik tego filtrowania. Ze względu na liczbę dodawanych przykładów i potrzebny na to czas nie przefiltrowano żadnego ze zbiorów przy użyciu estymacji gęstości dla niższych wartości τ, ani nie uwzględniono filtrowania z tym typem sąsiedztwa w późniejszej klasyfikacji przefiltrowanych zbiorów.



Rysunek 5.18: Efekt filtrowania zbioru *paw-4-600-5-30-10-10-0-U1* metodą *COST* z sąsiedztwem GK, $\sigma = 0{,}04$, $\tau = 0{,}5$

Z wykresu tego wyraźnie widać, że *COST* dodaje przykłady praktycznie jedynie w obszarach wątpliwych, w których blisko siebie występują przykłady obu klas. Widać jednak również, że przy sąsiedztwie estymacji gęstości algorytm usiłuje zbyt dobrze przybliżyć do siebie porównywane rozkłady, przez co tworzy niepotrzebnie tysiące kopii niektórych przykładów. Do tego, aby uzyskać wartość prawdopodobieństwa w sąsiedztwie równej docelowej, musi z różną częstością wielokrotnie powielać przykłady obu klas. Widać to szczególnie w lewym górnym rogu wykresu, gdzie jeden przykład mniejszościowy został powielony 322 razy, a otaczające go przykłady większościowe były powielane od 20 do aż 694 razy.

5.2.3 Porównanie skuteczności klasyfikatorów na przetworzonych danych

Przetworzone zbiory wykorzystano później do eksperymentu 10-krotnej kroswalidacji w przypadku rzeczywistych zbiorów i prostego testu klasyfikacji zbioru testowego dla zbioru sztucznego.

Wyniki klasyfikacji pokazano niżej w tabelach 5.2–5.31. Tabele są zgrupowane wg zbiorów, których dotyczą, jedna tabela dla jednego użytego klasyfikatora. W kolumnie **parametry** podano parametry użytych filtrów i klasyfikatorów – parametr k oznacza wielkość sąsiedztwa użytego przez filtr (gdy używa on sąsiedztwa k-NN) jak i sąsiedztwa użytego przez klasyfikator w przypadku klasyfikatora k-NN, parametr *ratio* mówi o docelowym stopniu (nie-)zrównoważenia danych, zadanym filtrowi, parametr τ mówi o koszcie zadanym algorytmowi *COST*.

Użyto opisanych poniżej klasyfikatorów:

- ***k*-NN** – klasyfikator k najbliższych sąsiadów, wykorzystano implementację z klasy `IBk` pakietu *Weka*. Dokonywano klasyfikacji zbioru nieprzetworzonego z sąsiedztwem 5-NN oraz 9-NN, w przypadku zbiorów przetworzonych metodą *COST* użyto sąsiedztwa takiego, jak w metodzie *COST*, w przypadku pozostałych zbiorów używano 5-NN. Wykorzystaną miarą odległości była odległość euklidesowa z wartościami atrybutów normalizowanych do zakresu $[0;\ 1]$.
- **J48** – algorytm budujący drzewo decyzyjne, implementacja algorytmu *C4.5* z pakietu *Weka*. Wykorzystano domyślne ustawienia.
- **PART** – algorytm budujący zestaw reguł decyzyjnych na podstawie częściowych drzew *C4.5*. Włączono opcję *unpruned*, tj. bez usuwania reguł o zbyt małym poparciu decyzji w danych, aby poprawić klasyfikację klasy mniejszościowej nawet w przypadku bez filtrowania.
- **Naïve Bayes** – naiwny klasyfikator bayesowski. Wykonano za jego pomocą eksperyment dwukrotnie – ponieważ algorytm ten szacuje prawdopodobieństwa dla różnych atrybutów, może w różne sposoby radzić sobie z atrybutami liczbowymi – użyto dwóch metod:
  - *Naïve Bayes* z jądrową estymacją gęstości – buduje funkcję jądrową Gaussa dla wszystkich atrybutów numerycznych i za jej pomocą oblicza gęstości klas w sąsiedztwie testowanego przykładu,
  - metoda domyślna – traktuje każdy atrybut liczbowy osobno tak, jakby miał on rozkład normalny i oblicza gęstości na tym atrybucie jednowymiarową funkcją Gaussa.

  Wyniki dla obu metod w większości wypadków są podobne, jednak w kilku miejscach pojawiają się wyraźne różnice.

Podczas eksperymentów zbierano następujące miary klasyfikacji:

- **TP** – liczba poprawnie zaklasyfikowanych przykładów klasy mniejszościowej (ang. *true positives*),
- **FP** – liczba przykładów większościowych błędnie zaklasyfikowanych jako mniejszościowe (ang. *false positives*),

- **FN** – liczba przykładów mniejszościowych błędnie zaklasyfikowanych jako większościowe (ang. *false negatives*),
- **TN** – liczba poprawnie zaklasyfikowanych przykładów większościowych (ang. *true negatives*),
- **TPR** – czułość, tj. odsetek prawdziwie mniejszościowych (ang. *true positive ratio*), $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$,
- **TNR** – specyficzność, tj. odsetek prawdziwie większościowych (ang. *true negative ratio*), $\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$,
- **precyzja** – odsetek prawdziwie mniejszościowych wśród wszystkich zaklasyfikowanych jako mniejszościowe, $\mathrm{prec} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$,
- **G-mean** – też *G-measure*, średnia geometryczna TPR i precyzji – miara ta jest często używana do porównywania klasyfikacji niezrównoważonych danych [Nap12], $G = \sqrt{\mathrm{TPR} \cdot \mathrm{prec}}$,
- **AUC** – pole powierzchni pod krzywą ROC, uśrednione z 10 eksperymentów kroswalidacji, obliczone poprzez *Wekę* (ang. *Area Under Curve*).

Dyskusja wyników przedstawiona jest po umieszczonych tabelach, na s. 58.

paw-4-600-5-30-10-10-0

Tabela 5.2: Wyniki klasyfikacji klasyfikatorem ***k*-NN**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak	$k = 5$		34	8	16	242	0,680	0,968	0,810	0,742	0,949
	$k = 9$		26	7	24	243	0,520	0,972	0,788	0,640	0,928
COST	$k = 5$	$\tau = 0{,}5$	32	8	18	242	0,640	0,968	0,800	0,716	0,943
		$\tau = 0{,}4$	40	19	10	231	0,800	0,924	0,678	0,736	0,954
		$\tau = 0{,}3$	42	33	8	217	0,840	0,868	0,560	0,686	0,942
		$\tau = 0{,}2$	50	61	0	189	1	0,756	0,450	0,671	0,952
		$\tau = 0{,}1$	50	67	0	183	1	0,732	0,427	0,653	0,945
	$k = 9$	$\tau = 0{,}5$	26	7	24	243	0,520	0,972	0,788	0,640	0,934
		$\tau = 0{,}4$	32	15	18	235	0,640	0,940	0,681	0,660	0,932
		$\tau = 0{,}3$	40	29	10	221	0,800	0,884	0,580	0,681	0,933
		$\tau = 0{,}2$	44	44	6	206	0,880	0,824	0,500	0,663	0,939
		$\tau = 0{,}1$	49	93	1	157	0,980	0,628	0,345	0,581	0,935
Random OS	$k = 5$		49	73	1	177	0,980	0,708	0,402	0,628	0,942
	$k = 9$		46	77	4	173	0,920	0,692	0,374	0,587	0,931
NCR	$k = 5$		40	33	10	217	0,800	0,868	0,548	0,814	0,921
SPIDER	$k = 5$		35	17	15	233	0,700	0,932	0,673	0,686	0,939
SMOTE	$k = 5$	ratio 1:1	48	37	2	213	0,960	0,852	0,565	0,736	0,951

Tabela 5.3: Wyniki klasyfikacji klasyfikatorem **J48**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			14	5	36	245	0,280	0,980	0,737	0,454	0,751
COST	$k=5$	$\tau=0{,}5$	14	5	36	245	0,280	0,980	0,737	0,454	0,751
		$\tau=0{,}4$	14	5	36	245	0,280	0,980	0,737	0,454	0,751
		$\tau=0{,}3$	32	33	18	217	0,640	0,868	0,492	0,561	0,786
		$\tau=0{,}2$	29	26	21	224	0,580	0,896	0,527	0,553	0,793
		$\tau=0{,}1$	32	27	18	223	0,640	0,892	0,542	0,589	0,833
	$k=9$	$\tau=0{,}5$	14	5	36	245	0,280	0,980	0,737	0,454	0,751
		$\tau=0{,}4$	14	5	36	245	0,280	0,980	0,737	0,454	0,751
		$\tau=0{,}3$	28	21	22	229	0,560	0,916	0,571	0,565	0,804
		$\tau=0{,}2$	34	24	16	226	0,680	0,904	0,586	0,631	0,809
		$\tau=0{,}1$	36	34	14	216	0,720	0,864	0,514	0,608	0,823
ROS		ratio 1:1	36	31	14	219	0,720	0,876	0,537	0,622	0,818
NCR	$k=5$		38	25	12	225	0,760	0,900	0,603	0,677	0,820
SPIDER	$k=5$		14	5	36	245	0,280	0,980	0,737	0,454	0,751
SMOTE	$k=5$	ratio 1:1	38	51	12	199	0,760	0,796	0,427	0,570	0,815

Tabela 5.4: Wyniki klasyfikacji klasyfikatorem **PART** (bez pruningu)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			14	5	36	245	0,280	0,980	0,737	0,454	0,807
COST	$k=5$	$\tau=0{,}5$	14	5	36	245	0,280	0,980	0,737	0,454	0,807
		$\tau=0{,}4$	14	5	36	245	0,280	0,980	0,737	0,454	0,775
		$\tau=0{,}3$	0	0	50	250	0	1	0	0	0,762
		$\tau=0{,}2$	34	29	16	221	0,680	0,884	0,540	0,606	0,857
		$\tau=0{,}1$	37	42	13	208	0,740	0,832	0,468	0,588	0,832
	$k=9$	$\tau=0{,}5$	13	5	37	245	0,260	0,980	0,722	0,433	0,800
		$\tau=0{,}4$	14	5	36	245	0,280	0,980	0,737	0,454	0,775
		$\tau=0{,}3$	14	5	36	245	0,280	0,980	0,737	0,454	0,802
		$\tau=0{,}2$	38	44	12	206	0,760	0,824	0,463	0,593	0,842
		$\tau=0{,}1$	30	19	20	231	0,600	0,924	0,612	0,606	0,834
ROS		ratio 1:1	41	67	9	183	0,820	0,732	0,380	0,558	0,838
NCR	$k=5$		37	26	13	224	0,740	0,896	0,587	0,659	0,843
SPIDER	$k=5$		29	14	21	236	0,580	0,944	0,674	0,625	0,834
SMOTE	$k=5$	ratio 1:1	39	71	11	179	0,780	0,716	0,355	0,526	0,830

Tabela 5.5: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes** (z jądrową estymacją gęstości)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			1	1	49	249	0,020	0,996	0,500	0,100	0,846
COST	$k = 5$	$\tau = 0{,}5$	1	1	49	249	0,020	0,996	0,500	0,100	0,846
		$\tau = 0{,}4$	6	1	44	249	0,120	0,996	0,857	0,321	0,850
		$\tau = 0{,}3$	17	11	33	239	0,340	0,956	0,607	0,454	0,856
		$\tau = 0{,}2$	23	23	27	227	0,460	0,908	0,500	0,480	0,859
		$\tau = 0{,}1$	42	61	8	189	0,840	0,756	0,408	0,585	0,862
	$k = 9$	$\tau = 0{,}5$	1	1	49	249	0,020	0,996	0,500	0,100	0,844
		$\tau = 0{,}4$	13	3	37	247	0,260	0,988	0,813	0,460	0,853
		$\tau = 0{,}3$	24	17	26	233	0,480	0,932	0,585	0,530	0,857
		$\tau = 0{,}2$	31	38	19	212	0,620	0,848	0,449	0,528	0,860
		$\tau = 0{,}1$	40	71	10	179	0,800	0,716	0,360	0,637	0,865
ROS		ratio 1:1	42	74	8	176	0,840	0,704	0,362	0,551	0,863
NCR	$k = 5$		23	22	27	228	0,460	0,912	0,511	0,485	0,833
SPIDER	$k = 5$		12	1	38	249	0,240	0,996	0,923	0,471	0,845
SMOTE	$k = 5$	ratio 1:1	42	70	8	180	0,840	0,720	0,375	0,561	0,869

Tabela 5.6: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			0	0	50	250	0	1	0	0	0,787
COST	$k = 5$	$\tau = 0{,}5$	0	0	50	250	0	1	0	0	0,786
		$\tau = 0{,}4$	0	0	50	250	0	1	0	0	0,782
		$\tau = 0{,}3$	0	0	50	250	0	1	0	0	0,786
		$\tau = 0{,}2$	0	0	50	250	0	1	0	0	0,782
		$\tau = 0{,}1$	26	51	24	199	0,520	0,796	0,338	0,419	0,784
	$k = 9$	$\tau = 0{,}5$	0	0	50	250	0	1	0	0	0,778
		$\tau = 0{,}4$	0	0	50	250	0	1	0	0	0,784
		$\tau = 0{,}3$	0	0	50	250	0	1	0	0	0,775
		$\tau = 0{,}2$	2	5	48	245	0,040	0,980	0,286	0,107	0,772
		$\tau = 0{,}1$	35	68	15	182	0,700	0,728	0,340	0,488	0,785
ROS		ratio 1:1	39	84	11	166	0,780	0,664	0,317	0,497	0,785
NCR	$k = 5$		0	0	50	250	0	1	0	0	0,782
SPIDER	$k = 5$		0	0	50	250	0	1	0	0	0,787
SMOTE	$k = 5$	ratio 1:1	38	79	12	171	0,760	0,684	0,325	0,497	0,785

new-thyroid

Tabela 5.7: Wyniki klasyfikacji klasyfikatorem ***k*-NN**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak	$k = 5$		31	0	4	180	0,886	1,000	1,000	0,941	0,999
	$k = 9$		25	0	10	180	0,714	1,000	1,000	0,845	0,997
COST	$k = 5$	$\tau = 0{,}5$	31	0	4	180	0,886	1,000	1,000	0,941	0,999
		$\tau = 0{,}4$	31	0	4	180	0,886	1,000	1,000	0,941	0,999
		$\tau = 0{,}3$	33	2	2	178	0,943	0,989	0,943	0,943	0,999
		$\tau = 0{,}2$	33	3	2	177	0,943	0,983	0,917	0,930	0,998
		$\tau = 0{,}1$	33	3	2	177	0,943	0,983	0,917	0,930	0,992
	$k = 9$	$\tau = 0{,}5$	26	0	9	180	0,743	1,000	1,000	0,862	0,997
		$\tau = 0{,}4$	29	1	6	179	0,829	0,994	0,967	0,895	0,997
		$\tau = 0{,}3$	32	1	3	179	0,914	0,994	0,970	0,942	0,998
		$\tau = 0{,}2$	33	3	2	177	0,943	0,983	0,917	0,930	0,997
		$\tau = 0{,}1$	33	6	2	174	0,943	0,967	0,846	0,893	0,997
ROS		ratio 1:1	35	4	0	176	1,000	0,978	0,897	0,947	0,996
NCR	$k = 5$		32	1	3	179	0,914	0,994	0,970	0,942	0,999
SPIDER	$k = 5$		31	0	4	180	0,886	1,000	1,000	0,941	0,999
SMOTE	$k = 5$	ratio 1:1	35	4	0	176	1,000	0,978	0,897	0,947	0,994

Tabela 5.8: Wyniki klasyfikacji klasyfikatorem **J48**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			32	3	3	177	0,914	0,983	0,914	0,914	0,944
COST	$k = 5$	$\tau = 0{,}5$	32	3	3	177	0,914	0,983	0,914	0,914	0,944
		$\tau = 0{,}4$	32	2	3	178	0,914	0,989	0,941	0,928	0,963
		$\tau = 0{,}3$	30	3	5	177	0,857	0,983	0,909	0,883	0,933
		$\tau = 0{,}2$	29	4	6	176	0,829	0,978	0,879	0,853	0,918
		$\tau = 0{,}1$	32	4	3	176	0,914	0,978	0,889	0,901	0,929
	$k = 9$	$\tau = 0{,}5$	31	3	4	177	0,886	0,983	0,912	0,899	0,931
		$\tau = 0{,}4$	31	4	4	176	0,886	0,978	0,886	0,886	0,927
		$\tau = 0{,}3$	30	3	5	177	0,857	0,983	0,909	0,883	0,921
		$\tau = 0{,}2$	32	3	3	177	0,914	0,983	0,914	0,914	0,946
		$\tau = 0{,}1$	31	2	4	178	0,886	0,989	0,939	0,912	0,916
ROS		ratio 1:1	30	2	5	178	0,857	0,989	0,938	0,896	0,925
NCR	$k = 5$		33	4	2	176	0,943	0,978	0,892	0,917	0,964
SPIDER	$k = 5$		31	1	4	179	0,886	0,994	0,969	0,926	0,960
SMOTE	$k = 5$	ratio 1:1	33	4	2	176	0,943	0,978	0,892	0,917	0,970

Tabela 5.9: Wyniki klasyfikacji klasyfikatorem **PART** (bez pruningu)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			32	2	3	178	0,914	0,989	0,941	0,928	0,949
COST	$k = 5$	$\tau = 0{,}5$	32	2	3	178	0,914	0,989	0,941	0,928	0,948
		$\tau = 0{,}4$	32	2	3	178	0,914	0,989	0,941	0,928	0,962
		$\tau = 0{,}3$	31	0	4	180	0,886	1,000	1,000	0,941	0,942
		$\tau = 0{,}2$	29	1	6	179	0,829	0,994	0,967	0,895	0,901
		$\tau = 0{,}1$	33	4	2	176	0,943	0,978	0,892	0,917	0,963
	$k = 9$	$\tau = 0{,}5$	31	2	4	178	0,886	0,989	0,939	0,912	0,935
		$\tau = 0{,}4$	31	3	4	177	0,886	0,983	0,912	0,899	0,933
		$\tau = 0{,}3$	29	3	6	177	0,829	0,983	0,906	0,867	0,908
		$\tau = 0{,}2$	31	2	4	178	0,886	0,989	0,939	0,912	0,936
		$\tau = 0{,}1$	31	2	4	178	0,886	0,989	0,939	0,912	0,916
ROS		ratio 1:1	28	2	7	178	0,800	0,989	0,933	0,864	0,894
NCR	$k = 5$		31	4	4	176	0,886	0,978	0,886	0,886	0,944
SPIDER	$k = 5$		31	0	4	180	0,886	1,000	1,000	0,941	0,964
SMOTE	$k = 5$	ratio 1:1	33	4	2	176	0,943	0,978	0,892	0,917	0,960

Tabela 5.10: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes** (z jądrową estymacją gęstości)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			35	5	0	175	1,000	0,972	0,875	0,935	1,000
COST	$k = 5$	$\tau = 0{,}5$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau = 0{,}4$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau = 0{,}3$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau = 0{,}2$	35	9	0	171	1,000	0,950	0,795	0,892	1,000
		$\tau = 0{,}1$	35	13	0	167	1,000	0,928	0,729	0,854	1,000
	$k = 9$	$\tau = 0{,}5$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau = 0{,}4$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau = 0{,}3$	35	6	0	174	1,000	0,967	0,854	0,924	1,000
		$\tau = 0{,}2$	35	10	0	170	1,000	0,944	0,778	0,882	1,000
		$\tau = 0{,}1$	35	13	0	167	1,000	0,928	0,729	0,854	0,998
ROS		ratio 1:1	35	11	0	169	1,000	0,939	0,761	0,872	1,000
NCR	$k = 5$		35	6	0	174	1,000	0,967	0,854	0,924	1,000
SPIDER	$k = 5$		35	5	0	175	1,000	0,972	0,875	0,935	1,000
SMOTE	$k = 5$	ratio 1:1	35	9	0	171	1,000	0,950	0,795	0,892	1,000

Tabela 5.11: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			35	5	0	175	1,000	0,972	0,875	0,935	1,000
COST	$k=5$	$\tau=0{,}5$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau=0{,}4$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau=0{,}3$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau=0{,}2$	35	9	0	171	1,000	0,950	0,795	0,892	1,000
		$\tau=0{,}1$	35	13	0	167	1,000	0,928	0,729	0,854	1,000
	$k=9$	$\tau=0{,}5$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau=0{,}4$	35	5	0	175	1,000	0,972	0,875	0,935	1,000
		$\tau=0{,}3$	35	6	0	174	1,000	0,967	0,854	0,924	1,000
		$\tau=0{,}2$	35	10	0	170	1,000	0,944	0,778	0,882	1,000
		$\tau=0{,}1$	35	13	0	167	1,000	0,928	0,729	0,854	0,998
ROS		ratio 1:1	35	11	0	169	1,000	0,939	0,761	0,872	1,000
NCR	$k=5$		35	6	0	174	1,000	0,967	0,854	0,924	1,000
SPIDER	$k=5$		35	5	0	175	1,000	0,972	0,875	0,935	1,000
SMOTE	$k=5$	ratio 1:1	35	9	0	171	1,000	0,950	0,795	0,892	1,000

haberman

Tabela 5.12: Wyniki klasyfikacji klasyfikatorem ***k*-NN**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak	$k=5$		15	26	66	199	0,185	0,884	0,366	0,260	0,613
	$k=9$		17	17	64	208	0,210	0,924	0,500	0,324	0,609
COST	$k=5$	$\tau=0{,}5$	15	27	66	198	0,185	0,880	0,357	0,257	0,617
		$\tau=0{,}4$	28	43	53	182	0,346	0,809	0,394	0,369	0,615
		$\tau=0{,}3$	36	68	45	157	0,444	0,698	0,346	0,392	0,617
		$\tau=0{,}2$	55	114	26	111	0,679	0,493	0,325	0,470	0,601
		$\tau=0{,}1$	58	123	23	102	0,716	0,453	0,320	0,479	0,594
	$k=9$	$\tau=0{,}5$	17	19	64	206	0,210	0,916	0,472	0,315	0,607
		$\tau=0{,}4$	23	37	58	188	0,284	0,836	0,383	0,330	0,605
		$\tau=0{,}3$	35	66	46	159	0,432	0,707	0,347	0,387	0,602
		$\tau=0{,}2$	43	104	38	121	0,531	0,538	0,293	0,394	0,571
		$\tau=0{,}1$	59	140	22	85	0,728	0,378	0,296	0,465	0,592
ROS		ratio 1:1	44	93	37	132	0,543	0,587	0,321	0,418	0,600
NCR	$k=5$		51	108	30	117	0,630	0,520	0,321	0,449	0,623
SPIDER	$k=5$		26	41	55	184	0,321	0,818	0,388	0,353	0,629
SMOTE	$k=5$	ratio 1:1	34	51	47	174	0,420	0,773	0,400	0,410	0,637

Tabela 5.13: Wyniki klasyfikacji klasyfikatorem **J48**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			31	41	50	184	0,383	0,818	0,431	0,406	0,590
COST	$k = 5$	$\tau = 0{,}5$	36	45	45	180	0,444	0,800	0,444	0,444	0,627
		$\tau = 0{,}4$	44	56	37	169	0,543	0,751	0,440	0,489	0,639
		$\tau = 0{,}3$	46	65	35	160	0,568	0,711	0,414	0,485	0,617
		$\tau = 0{,}2$	51	90	30	135	0,630	0,600	0,362	0,477	0,617
		$\tau = 0{,}1$	46	81	35	144	0,568	0,640	0,362	0,454	0,606
	$k = 9$	$\tau = 0{,}5$	41	46	40	179	0,506	0,796	0,471	0,488	0,629
		$\tau = 0{,}4$	40	56	41	169	0,494	0,751	0,417	0,454	0,620
		$\tau = 0{,}3$	45	72	36	153	0,556	0,680	0,385	0,462	0,623
		$\tau = 0{,}2$	54	85	27	140	0,667	0,622	0,388	0,509	0,642
		$\tau = 0{,}1$	43	70	38	155	0,531	0,689	0,381	0,449	0,595
ROS		ratio 1:1	47	67	34	158	0,580	0,702	0,412	0,489	0,613
NCR	$k = 5$		57	86	24	139	0,704	0,618	0,399	0,530	0,648
SPIDER	$k = 5$		35	36	46	189	0,432	0,840	0,493	0,462	0,637
SMOTE	$k = 5$	ratio 1:1	39	52	42	173	0,481	0,769	0,429	0,454	0,617

Tabela 5.14: Wyniki klasyfikacji klasyfikatorem **PART** (bez pruningu)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			26	31	55	194	0,321	0,862	0,456	0,383	0,605
COST	$k = 5$	$\tau = 0{,}5$	28	31	53	194	0,346	0,862	0,475	0,405	0,630
		$\tau = 0{,}4$	37	53	44	172	0,457	0,764	0,411	0,433	0,650
		$\tau = 0{,}3$	43	69	38	156	0,531	0,693	0,384	0,451	0,605
		$\tau = 0{,}2$	39	49	42	176	0,481	0,782	0,443	0,462	0,653
		$\tau = 0{,}1$	53	109	28	116	0,654	0,516	0,327	0,463	0,631
	$k = 9$	$\tau = 0{,}5$	29	34	52	191	0,358	0,849	0,460	0,406	0,626
		$\tau = 0{,}4$	38	49	43	176	0,469	0,782	0,437	0,453	0,652
		$\tau = 0{,}3$	47	71	34	154	0,580	0,684	0,398	0,481	0,641
		$\tau = 0{,}2$	49	56	32	169	0,605	0,751	0,467	0,531	0,690
		$\tau = 0{,}1$	61	140	20	85	0,753	0,378	0,303	0,478	0,606
ROS		ratio 1:1	53	85	28	140	0,654	0,622	0,384	0,501	0,639
NCR	$k = 5$		51	75	30	150	0,630	0,667	0,405	0,505	0,656
SPIDER	$k = 5$		36	44	45	181	0,444	0,804	0,450	0,447	0,634
SMOTE	$k = 5$	ratio 1:1	38	54	43	171	0,469	0,760	0,413	0,440	0,653

Tabela 5.15: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes** (z jądrową estymacją gęstości)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			17	12	64	213	0,210	0,947	0,586	0,351	0,675
COST	$k = 5$	$\tau = 0{,}5$	17	12	64	213	0,210	0,947	0,586	0,351	0,660
		$\tau = 0{,}4$	19	13	62	212	0,235	0,942	0,594	0,373	0,665
		$\tau = 0{,}3$	20	16	61	209	0,247	0,929	0,556	0,370	0,666
		$\tau = 0{,}2$	26	19	55	206	0,321	0,916	0,578	0,431	0,659
		$\tau = 0{,}1$	54	113	27	112	0,667	0,498	0,323	0,464	0,659
	$k = 9$	$\tau = 0{,}5$	18	13	63	212	0,222	0,942	0,581	0,359	0,664
		$\tau = 0{,}4$	20	14	61	211	0,247	0,938	0,588	0,381	0,658
		$\tau = 0{,}3$	22	16	59	209	0,272	0,929	0,579	0,397	0,669
		$\tau = 0{,}2$	28	22	53	203	0,346	0,902	0,560	0,440	0,651
		$\tau = 0{,}1$	52	98	29	127	0,642	0,564	0,347	0,472	0,666
ROS		ratio 1:1	28	19	53	206	0,346	0,916	0,596	0,454	0,662
NCR	$k = 5$		39	48	42	177	0,481	0,787	0,448	0,465	0,664
SPIDER	$k = 5$		31	28	50	197	0,383	0,876	0,525	0,448	0,662
SMOTE	$k = 5$	ratio 1:1	23	17	58	208	0,284	0,924	0,575	0,404	0,655

Tabela 5.16: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			17	12	64	213	0,210	0,947	0,586	0,351	0,675
COST	$k = 5$	$\tau = 0{,}5$	17	12	64	213	0,210	0,947	0,586	0,351	0,660
		$\tau = 0{,}4$	19	13	62	212	0,235	0,942	0,594	0,373	0,665
		$\tau = 0{,}3$	20	16	61	209	0,247	0,929	0,556	0,370	0,666
		$\tau = 0{,}2$	26	19	55	206	0,321	0,916	0,578	0,431	0,659
		$\tau = 0{,}1$	54	113	27	112	0,667	0,498	0,323	0,464	0,659
	$k = 9$	$\tau = 0{,}5$	18	13	63	212	0,222	0,942	0,581	0,359	0,664
		$\tau = 0{,}4$	20	14	61	211	0,247	0,938	0,588	0,381	0,658
		$\tau = 0{,}3$	22	16	59	209	0,272	0,929	0,579	0,397	0,669
		$\tau = 0{,}2$	28	22	53	203	0,346	0,902	0,560	0,440	0,651
		$\tau = 0{,}1$	52	98	29	127	0,642	0,564	0,347	0,472	0,666
ROS		ratio 1:1	28	19	53	206	0,346	0,916	0,596	0,454	0,662
NCR	$k = 5$		39	48	42	177	0,481	0,787	0,448	0,465	0,664
SPIDER	$k = 5$		31	28	50	197	0,383	0,876	0,525	0,448	0,662
SMOTE	$k = 5$	ratio 1:1	23	17	58	208	0,284	0,924	0,575	0,404	0,655

ecoli

Tabela 5.17: Wyniki klasyfikacji klasyfikatorem ***k*-NN**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak	$k = 5$		21	10	14	291	0,600	0,967	0,677	0,638	0,921
	$k = 9$		21	10	14	291	0,600	0,967	0,677	0,638	0,924
COST	$k = 5$	$\tau = 0{,}5$	21	10	14	291	0,600	0,967	0,677	0,638	0,921
		$\tau = 0{,}4$	24	15	11	286	0,686	0,950	0,615	0,650	0,909
		$\tau = 0{,}3$	26	25	9	276	0,743	0,917	0,510	0,615	0,900
		$\tau = 0{,}2$	26	37	9	264	0,743	0,877	0,413	0,554	0,898
		$\tau = 0{,}1$	27	44	8	257	0,771	0,854	0,380	0,542	0,875
	$k = 9$	$\tau = 0{,}5$	21	10	14	291	0,600	0,967	0,677	0,638	0,921
		$\tau = 0{,}4$	26	17	9	284	0,743	0,944	0,605	0,670	0,920
		$\tau = 0{,}3$	28	23	7	278	0,800	0,924	0,549	0,663	0,908
		$\tau = 0{,}2$	30	30	5	271	0,857	0,900	0,500	0,655	0,917
		$\tau = 0{,}1$	30	51	5	250	0,857	0,831	0,370	0,563	0,913
ROS		ratio 1:1	32	51	3	250	0,914	0,831	0,386	0,594	0,886
NCR	$k = 5$		30	29	5	272	0,857	0,904	0,508	0,660	0,917
SPIDER	$k = 5$		25	12	10	289	0,714	0,960	0,676	0,695	0,927
SMOTE	$k = 5$	ratio 1:1	30	35	5	266	0,857	0,884	0,462	0,629	0,910

Tabela 5.18: Wyniki klasyfikacji klasyfikatorem **J48**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			20	10	15	291	0,571	0,967	0,667	0,617	0,806
COST	$k = 5$	$\tau = 0{,}5$	21	11	14	290	0,600	0,963	0,656	0,627	0,815
		$\tau = 0{,}4$	21	15	14	286	0,600	0,950	0,583	0,592	0,809
		$\tau = 0{,}3$	20	18	15	283	0,571	0,940	0,526	0,548	0,733
		$\tau = 0{,}2$	23	20	12	281	0,657	0,934	0,535	0,593	0,825
		$\tau = 0{,}1$	24	10	11	291	0,686	0,967	0,706	0,696	0,840
	$k = 9$	$\tau = 0{,}5$	22	11	13	290	0,629	0,963	0,667	0,647	0,834
		$\tau = 0{,}4$	20	14	15	287	0,571	0,953	0,588	0,580	0,712
		$\tau = 0{,}3$	23	17	12	284	0,657	0,944	0,575	0,615	0,840
		$\tau = 0{,}2$	22	17	13	284	0,629	0,944	0,564	0,595	0,832
		$\tau = 0{,}1$	23	16	12	285	0,657	0,947	0,590	0,623	0,830
ROS		ratio 1:1	23	17	12	284	0,657	0,944	0,575	0,615	0,815
NCR	$k = 5$		29	27	6	274	0,829	0,910	0,518	0,655	0,892
SPIDER	$k = 5$		24	11	11	290	0,686	0,963	0,686	0,686	0,899
SMOTE	$k = 5$	ratio 1:1	29	19	6	282	0,829	0,937	0,604	0,708	0,895

Tabela 5.19: Wyniki klasyfikacji klasyfikatorem **PART** (bez pruningu)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			16	4	19	297	0,457	0,987	0,800	0,605	0,857
COST	$k = 5$	$\tau = 0{,}5$	16	5	19	296	0,457	0,983	0,762	0,590	0,833
		$\tau = 0{,}4$	15	13	20	288	0,429	0,957	0,536	0,479	0,823
		$\tau = 0{,}3$	22	18	13	283	0,629	0,940	0,550	0,588	0,849
		$\tau = 0{,}2$	21	18	14	283	0,600	0,940	0,538	0,568	0,796
		$\tau = 0{,}1$	18	13	17	288	0,514	0,957	0,581	0,546	0,834
	$k = 9$	$\tau = 0{,}5$	17	18	18	283	0,486	0,940	0,486	0,486	0,813
		$\tau = 0{,}4$	16	13	19	288	0,457	0,957	0,552	0,502	0,864
		$\tau = 0{,}3$	16	15	19	286	0,457	0,950	0,516	0,486	0,809
		$\tau = 0{,}2$	18	17	17	284	0,514	0,944	0,514	0,514	0,776
		$\tau = 0{,}1$	21	16	14	285	0,600	0,947	0,568	0,584	0,831
ROS		ratio 1:1	22	21	13	280	0,629	0,930	0,512	0,567	0,787
NCR	$k = 5$		28	28	7	273	0,800	0,907	0,500	0,632	0,886
SPIDER	$k = 5$		20	11	15	290	0,571	0,963	0,645	0,607	0,860
SMOTE	$k = 5$	ratio 1:1	30	31	5	270	0,857	0,897	0,492	0,649	0,879

Tabela 5.20: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes** (z jądrową estymacją gęstości)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			30	32	5	269	0,857	0,894	0,484	0,644	0,937
COST	$k = 5$	$\tau = 0{,}5$	30	32	5	269	0,857	0,894	0,484	0,644	0,936
		$\tau = 0{,}4$	31	38	4	263	0,886	0,874	0,449	0,631	0,931
		$\tau = 0{,}3$	31	42	4	259	0,886	0,860	0,425	0,613	0,929
		$\tau = 0{,}2$	32	42	3	259	0,914	0,860	0,432	0,629	0,932
		$\tau = 0{,}1$	32	43	3	258	0,914	0,857	0,427	0,625	0,936
	$k = 9$	$\tau = 0{,}5$	30	35	5	266	0,857	0,884	0,462	0,629	0,930
		$\tau = 0{,}4$	31	38	4	263	0,886	0,874	0,449	0,631	0,929
		$\tau = 0{,}3$	32	42	3	259	0,914	0,860	0,432	0,629	0,930
		$\tau = 0{,}2$	32	43	3	258	0,914	0,857	0,427	0,625	0,928
		$\tau = 0{,}1$	31	43	4	258	0,886	0,857	0,419	0,609	0,938
ROS		ratio 1:1	34	49	1	252	0,971	0,837	0,410	0,631	0,935
NCR	$k = 5$		31	42	4	259	0,886	0,860	0,425	0,613	0,933
SPIDER	$k = 5$		31	36	4	265	0,886	0,880	0,463	0,640	0,938
SMOTE	$k = 5$	ratio 1:1	32	42	3	259	0,914	0,860	0,432	0,629	0,945

Tabela 5.21: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			30	32	5	269	0,857	0,894	0,484	0,644	0,937
COST	$k=5$	$\tau=0{,}5$	30	32	5	269	0,857	0,894	0,484	0,644	0,936
		$\tau=0{,}4$	31	38	4	263	0,886	0,874	0,449	0,631	0,931
		$\tau=0{,}3$	31	42	4	259	0,886	0,860	0,425	0,613	0,929
		$\tau=0{,}2$	32	42	3	259	0,914	0,860	0,432	0,629	0,932
		$\tau=0{,}1$	32	43	3	258	0,914	0,857	0,427	0,625	0,936
	$k=9$	$\tau=0{,}5$	30	35	5	266	0,857	0,884	0,462	0,629	0,930
		$\tau=0{,}4$	31	38	4	263	0,886	0,874	0,449	0,631	0,929
		$\tau=0{,}3$	32	42	3	259	0,914	0,860	0,432	0,629	0,930
		$\tau=0{,}2$	32	43	3	258	0,914	0,857	0,427	0,625	0,928
		$\tau=0{,}1$	31	43	4	258	0,886	0,857	0,419	0,609	0,938
ROS		ratio 1:1	34	49	1	252	0,971	0,837	0,410	0,631	0,935
NCR	$k=5$		31	42	4	259	0,886	0,860	0,425	0,613	0,933
SPIDER	$k=5$		31	36	4	265	0,886	0,880	0,463	0,640	0,938
SMOTE	$k=5$	ratio 1:1	32	42	3	259	0,914	0,860	0,432	0,629	0,945

transfusion

Tabela 5.22: Wyniki klasyfikacji klasyfikatorem ***k*-NN**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak	$k=5$		59	57	119	513	0,331	0,900	0,509	0,411	0,705
	$k=9$		52	45	126	525	0,292	0,921	0,536	0,396	0,703
COST	$k=5$	$\tau=0{,}5$	60	62	118	508	0,337	0,891	0,492	0,407	0,700
		$\tau=0{,}4$	83	102	95	468	0,466	0,821	0,449	0,457	0,703
		$\tau=0{,}3$	94	137	84	433	0,528	0,760	0,407	0,464	0,690
		$\tau=0{,}2$	122	268	56	302	0,685	0,530	0,313	0,463	0,682
		$\tau=0{,}1$	141	345	37	225	0,792	0,395	0,290	0,479	0,673
	$k=9$	$\tau=0{,}5$	52	47	126	523	0,292	0,918	0,525	0,392	0,689
		$\tau=0{,}4$	74	86	104	484	0,416	0,849	0,463	0,438	0,694
		$\tau=0{,}3$	94	146	84	424	0,528	0,744	0,392	0,455	0,692
		$\tau=0{,}2$	126	257	52	313	0,708	0,549	0,329	0,483	0,684
		$\tau=0{,}1$	148	392	30	178	0,831	0,312	0,274	0,477	0,680
ROS		ratio 1:1	122	233	56	337	0,685	0,591	0,344	0,485	0,696
NCR	$k=5$		112	176	66	394	0,629	0,691	0,389	0,495	0,711
SPIDER	$k=5$		75	92	103	478	0,421	0,839	0,449	0,435	0,696
SMOTE	$k=5$	ratio 1:1	103	157	75	413	0,579	0,725	0,396	0,479	0,695

Tabela 5.23: Wyniki klasyfikacji klasyfikatorem **J48**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			70	54	108	516	0,393	0,905	0,565	0,471	0,719
COST	$k = 5$	$\tau = 0{,}5$	70	54	108	516	0,393	0,905	0,565	0,471	0,719
		$\tau = 0{,}4$	78	62	100	508	0,438	0,891	0,557	0,494	0,704
		$\tau = 0{,}3$	83	79	95	491	0,466	0,861	0,512	0,489	0,723
		$\tau = 0{,}2$	115	169	63	401	0,646	0,704	0,405	0,511	0,705
		$\tau = 0{,}1$	122	269	56	301	0,685	0,528	0,312	0,462	0,645
	$k = 9$	$\tau = 0{,}5$	71	61	107	509	0,399	0,893	0,538	0,463	0,694
		$\tau = 0{,}4$	81	82	97	488	0,455	0,856	0,497	0,476	0,706
		$\tau = 0{,}3$	80	93	98	477	0,449	0,837	0,462	0,456	0,671
		$\tau = 0{,}2$	115	209	63	361	0,646	0,633	0,355	0,479	0,657
		$\tau = 0{,}1$	112	245	66	325	0,629	0,570	0,314	0,444	0,621
ROS		ratio 1:1	114	200	64	370	0,640	0,649	0,363	0,482	0,656
NCR	$k = 5$		111	133	67	437	0,624	0,767	0,455	0,533	0,704
SPIDER	$k = 5$		73	75	105	495	0,410	0,868	0,493	0,450	0,721
SMOTE	$k = 5$	ratio 1:1	110	159	68	411	0,618	0,721	0,409	0,503	0,664

Tabela 5.24: Wyniki klasyfikacji klasyfikatorem **PART** (bez pruningu)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			77	71	101	499	0,433	0,875	0,520	0,474	0,720
COST	$k = 5$	$\tau = 0{,}5$	78	71	100	499	0,438	0,875	0,523	0,479	0,725
		$\tau = 0{,}4$	81	62	97	508	0,455	0,891	0,566	0,508	0,721
		$\tau = 0{,}3$	69	66	109	504	0,388	0,884	0,511	0,445	0,710
		$\tau = 0{,}2$	121	202	57	368	0,680	0,646	0,375	0,505	0,696
		$\tau = 0{,}1$	128	275	50	295	0,719	0,518	0,318	0,478	0,663
	$k = 9$	$\tau = 0{,}5$	74	73	104	497	0,416	0,872	0,503	0,457	0,703
		$\tau = 0{,}4$	80	88	98	482	0,449	0,846	0,476	0,463	0,701
		$\tau = 0{,}3$	73	97	105	473	0,410	0,830	0,429	0,420	0,663
		$\tau = 0{,}2$	115	185	63	385	0,646	0,675	0,383	0,498	0,680
		$\tau = 0{,}1$	105	193	73	377	0,590	0,661	0,352	0,456	0,656
ROS		ratio 1:1	107	177	71	393	0,601	0,689	0,377	0,476	0,673
NCR	$k = 5$		97	111	81	459	0,545	0,805	0,466	0,504	0,702
SPIDER	$k = 5$		76	68	102	502	0,427	0,881	0,528	0,475	0,724
SMOTE	$k = 5$	ratio 1:1	86	119	92	451	0,483	0,791	0,420	0,450	0,695

Tabela 5.25: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes** (z jądrową estymacją gęstości)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			34	37	144	533	0,191	0,935	0,479	0,302	0,713
COST	$k = 5$	$\tau = 0{,}5$	32	38	146	532	0,180	0,933	0,457	0,287	0,706
		$\tau = 0{,}4$	39	44	139	526	0,219	0,923	0,470	0,321	0,704
		$\tau = 0{,}3$	42	57	136	513	0,236	0,900	0,424	0,316	0,704
		$\tau = 0{,}2$	52	82	126	488	0,292	0,856	0,388	0,337	0,697
		$\tau = 0{,}1$	143	274	35	296	0,803	0,519	0,343	0,525	0,699
	$k = 9$	$\tau = 0{,}5$	33	40	145	530	0,185	0,930	0,452	0,289	0,704
		$\tau = 0{,}4$	42	50	136	520	0,236	0,912	0,457	0,328	0,707
		$\tau = 0{,}3$	46	65	132	505	0,258	0,886	0,414	0,327	0,709
		$\tau = 0{,}2$	74	116	104	454	0,416	0,796	0,389	0,402	0,712
		$\tau = 0{,}1$	141	282	37	288	0,792	0,505	0,333	0,514	0,705
ROS		ratio 1:1	58	88	120	482	0,326	0,846	0,397	0,360	0,705
NCR	$k = 5$		55	95	123	475	0,309	0,833	0,367	0,337	0,704
SPIDER	$k = 5$		43	62	135	508	0,242	0,891	0,410	0,315	0,715
SMOTE	$k = 5$	ratio 1:1	72	87	106	483	0,404	0,847	0,453	0,428	0,714

Tabela 5.26: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			33	37	145	533	0,185	0,935	0,471	0,296	0,709
COST	$k = 5$	$\tau = 0{,}5$	32	38	146	532	0,180	0,933	0,457	0,287	0,706
		$\tau = 0{,}4$	39	44	139	526	0,219	0,923	0,470	0,321	0,704
		$\tau = 0{,}3$	42	57	136	513	0,236	0,900	0,424	0,316	0,704
		$\tau = 0{,}2$	52	82	126	488	0,292	0,856	0,388	0,337	0,697
		$\tau = 0{,}1$	143	274	35	296	0,803	0,519	0,343	0,525	0,699
	$k = 9$	$\tau = 0{,}5$	33	40	145	530	0,185	0,930	0,452	0,289	0,704
		$\tau = 0{,}4$	42	50	136	520	0,236	0,912	0,457	0,328	0,707
		$\tau = 0{,}3$	46	65	132	505	0,258	0,886	0,414	0,327	0,709
		$\tau = 0{,}2$	74	116	104	454	0,416	0,796	0,389	0,402	0,712
		$\tau = 0{,}1$	141	282	37	288	0,792	0,505	0,333	0,514	0,705
ROS		ratio 1:1	58	88	120	482	0,326	0,846	0,397	0,360	0,705
NCR	$k = 5$		55	95	123	475	0,309	0,833	0,367	0,337	0,704
SPIDER	$k = 5$		43	62	135	508	0,242	0,891	0,410	0,315	0,715
SMOTE	$k = 5$	ratio 1:1	72	87	106	483	0,404	0,847	0,453	0,428	0,714

yeast-ME2

Tabela 5.27: Wyniki klasyfikacji klasyfikatorem ***k*-NN**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak	$k = 5$		10	11	41	1422	0,196	0,992	0,476	0,306	0,816
	$k = 9$		5	6	46	1427	0,098	0,996	0,455	0,211	0,855
COST	$k = 5$	$\tau = 0{,}5$	10	11	41	1422	0,196	0,992	0,476	0,306	0,816
		$\tau = 0{,}4$	15	19	36	1414	0,294	0,987	0,441	0,360	0,768
		$\tau = 0{,}3$	21	31	30	1402	0,412	0,978	0,404	0,408	0,814
		$\tau = 0{,}2$	29	100	22	1333	0,569	0,930	0,225	0,358	0,815
		$\tau = 0{,}1$	31	114	20	1319	0,608	0,920	0,214	0,360	0,813
	$k = 9$	$\tau = 0{,}5$	4	8	47	1425	0,078	0,994	0,333	0,162	0,829
		$\tau = 0{,}4$	16	16	35	1417	0,314	0,989	0,500	0,396	0,852
		$\tau = 0{,}3$	19	29	32	1404	0,373	0,980	0,396	0,384	0,852
		$\tau = 0{,}2$	26	50	25	1383	0,510	0,965	0,342	0,418	0,840
		$\tau = 0{,}1$	33	143	18	1290	0,647	0,900	0,188	0,348	0,852
ROS		ratio 1:1	35	126	16	1307	0,686	0,912	0,217	0,386	0,815
NCR	$k = 5$		22	29	29	1404	0,431	0,980	0,431	0,431	0,844
SPIDER	$k = 5$		12	12	39	1421	0,235	0,992	0,500	0,343	0,816
SMOTE	$k = 5$	ratio 1:1	41	151	10	1282	0,804	0,895	0,214	0,414	0,892

Tabela 5.28: Wyniki klasyfikacji klasyfikatorem **J48**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			15	18	36	1415	0,294	0,987	0,455	0,366	0,817
COST	$k = 5$	$\tau = 0{,}5$	14	18	37	1415	0,275	0,987	0,438	0,347	0,817
		$\tau = 0{,}4$	21	37	30	1396	0,412	0,974	0,362	0,386	0,776
		$\tau = 0{,}3$	16	31	35	1402	0,314	0,978	0,340	0,327	0,755
		$\tau = 0{,}2$	18	46	33	1387	0,353	0,968	0,281	0,315	0,685
		$\tau = 0{,}1$	18	40	33	1393	0,353	0,972	0,310	0,331	0,710
	$k = 9$	$\tau = 0{,}5$	16	23	35	1410	0,314	0,984	0,410	0,359	0,804
		$\tau = 0{,}4$	17	35	34	1398	0,333	0,976	0,327	0,330	0,778
		$\tau = 0{,}3$	20	46	31	1387	0,392	0,968	0,303	0,345	0,730
		$\tau = 0{,}2$	16	37	35	1396	0,314	0,974	0,302	0,308	0,716
		$\tau = 0{,}1$	21	37	30	1396	0,412	0,974	0,362	0,386	0,740
ROS		ratio 1:1	22	39	29	1394	0,431	0,973	0,361	0,394	0,702
NCR	$k = 5$		20	24	31	1409	0,392	0,983	0,455	0,422	0,702
SPIDER	$k = 5$		17	16	34	1417	0,333	0,989	0,515	0,414	0,830
SMOTE	$k = 5$	ratio 1:1	26	71	25	1362	0,510	0,950	0,268	0,370	0,747

Tabela 5.29: Wyniki klasyfikacji klasyfikatorem **PART** (bez pruningu)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			15	18	36	1415	0,294	0,987	0,455	0,366	0,777
COST	$k = 5$	$\tau = 0{,}5$	14	19	37	1414	0,275	0,987	0,424	0,341	0,797
		$\tau = 0{,}4$	6	21	45	1412	0,118	0,985	0,222	0,162	0,776
		$\tau = 0{,}3$	17	55	34	1378	0,333	0,962	0,236	0,281	0,829
		$\tau = 0{,}2$	18	55	33	1378	0,353	0,962	0,247	0,295	0,729
		$\tau = 0{,}1$	16	38	35	1395	0,314	0,973	0,296	0,305	0,710
	$k = 9$	$\tau = 0{,}5$	7	13	44	1420	0,137	0,991	0,350	0,219	0,870
		$\tau = 0{,}4$	5	21	46	1412	0,098	0,985	0,192	0,137	0,807
		$\tau = 0{,}3$	9	26	42	1407	0,176	0,982	0,257	0,213	0,744
		$\tau = 0{,}2$	20	43	31	1390	0,392	0,970	0,317	0,353	0,732
		$\tau = 0{,}1$	16	56	35	1377	0,314	0,961	0,222	0,264	0,726
ROS		ratio 1:1	24	46	27	1387	0,471	0,968	0,343	0,402	0,719
NCR	$k = 5$		20	29	31	1404	0,392	0,980	0,408	0,400	0,783
SPIDER	$k = 5$		12	22	39	1411	0,235	0,985	0,353	0,288	0,751
SMOTE	$k = 5$	ratio 1:1	31	83	20	1350	0,608	0,942	0,272	0,407	0,823

Tabela 5.30: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes** (z jądrową estymacją gęstości)

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			24	78	27	1355	0,471	0,946	0,235	0,333	0,861
COST	$k = 5$	$\tau = 0{,}5$	25	80	26	1353	0,490	0,944	0,238	0,342	0,863
		$\tau = 0{,}4$	26	85	25	1348	0,510	0,941	0,234	0,346	0,853
		$\tau = 0{,}3$	28	95	23	1338	0,549	0,934	0,228	0,354	0,856
		$\tau = 0{,}2$	31	104	20	1329	0,608	0,927	0,230	0,374	0,860
		$\tau = 0{,}1$	34	133	17	1300	0,667	0,907	0,204	0,368	0,851
	$k = 9$	$\tau = 0{,}5$	23	80	28	1353	0,451	0,944	0,223	0,317	0,856
		$\tau = 0{,}4$	27	88	24	1345	0,529	0,939	0,235	0,353	0,859
		$\tau = 0{,}3$	29	98	22	1335	0,569	0,932	0,228	0,360	0,865
		$\tau = 0{,}2$	33	105	18	1328	0,647	0,927	0,239	0,393	0,866
		$\tau = 0{,}1$	36	126	15	1307	0,706	0,912	0,222	0,396	0,859
ROS		ratio 1:1	41	263	10	1170	0,804	0,816	0,135	0,329	0,862
NCR	$k = 5$		31	99	20	1334	0,608	0,931	0,238	0,381	0,864
SPIDER	$k = 5$		25	80	26	1353	0,490	0,944	0,238	0,342	0,863
SMOTE	$k = 5$	ratio 1:1	42	283	9	1150	0,824	0,803	0,129	0,326	0,844

Tabela 5.31: Wyniki klasyfikacji klasyfikatorem **Naïve Bayes**

filtr	parametry		TP	FP	FN	TN	TPR	TNR	precyzja	G-mean	AUC
brak			25	80	26	1353	0,490	0,944	0,238	0,342	0,862
COST	$k = 5$	$\tau = 0{,}5$	25	80	26	1353	0,490	0,944	0,238	0,342	0,863
		$\tau = 0{,}4$	26	85	25	1348	0,510	0,941	0,234	0,346	0,853
		$\tau = 0{,}3$	28	95	23	1338	0,549	0,934	0,228	0,354	0,856
		$\tau = 0{,}2$	31	104	20	1329	0,608	0,927	0,230	0,374	0,860
		$\tau = 0{,}1$	34	133	17	1300	0,667	0,907	0,204	0,368	0,851
	$k = 9$	$\tau = 0{,}5$	23	80	28	1353	0,451	0,944	0,223	0,317	0,856
		$\tau = 0{,}4$	27	88	24	1345	0,529	0,939	0,235	0,353	0,859
		$\tau = 0{,}3$	29	98	22	1335	0,569	0,932	0,228	0,360	0,865
		$\tau = 0{,}2$	33	105	18	1328	0,647	0,927	0,239	0,393	0,866
		$\tau = 0{,}1$	36	126	15	1307	0,706	0,912	0,222	0,396	0,859
ROS		ratio 1:1	41	263	10	1170	0,804	0,816	0,135	0,329	0,862
NCR	$k = 5$		31	99	20	1334	0,608	0,931	0,238	0,381	0,864
SPIDER	$k = 5$		25	80	26	1353	0,490	0,944	0,238	0,342	0,863
SMOTE	$k = 5$	ratio 1:1	42	283	9	1150	0,824	0,803	0,129	0,326	0,844

Dyskusja wyników

Z tabel łatwo zauważyć, że *COST* dla $\tau = 0{,}5$, niezależnie od wyboru sąsiedztwa, nie zmienia znacznie miar jakości klasyfikacji, czasem nieznacznie je pogarsza – potrafi równocześnie pogorszyć TPR, precyzję, G-mean i pole pod krzywą ROC – ale wszystkie te wartości pozostają blisko tych uzyskanych dla nieprzetworzonego zbioru. Jest to zgodne z oczekiwaniami, ponieważ dla$\tau = 0{,}5$ algorytm nie powinien zmieniać rozkładów przykładów poza drobnym dostosowaniem prawdopodobieństw, uwzględniającym niepewność oszacowań.

Równocześnie zazwyczaj dla niskich τ (między 0,1 a 0,3) poprawia TPR kosztem TNR i precyzji. Oznacza to, że algorytm ten wzmacnia klasę mniejszościową dla $\tau < 0{,}5$. Zależnie od zbioru i klasyfikatora ma to ostatecznie różny wpływ na wartość G-mean.

W przypadku sztucznego zbioru *paw-4-600-5-30-10-10-0* widać, że dla klasyfikatora k-NN jedynie filtr *NCR* wyraźnie poprawił wartość G-mean. Reszta, choć poprawiała TPR, czyli wzmacniała klasę mniejszościową, przy okazji tak bardzo osłabiała precyzję, że zmniejszała miarę G-mean. Metoda *COST* działała w tym wypadku dla niskich τ porównywalnie do prostego nadlosowania, przy czym dodawała mniej przykładów, natomiast filtrowanie trwało dużo dłużej. W przypadku pozostałych klasyfikatorów, uzyskuje się wyraźne poprawienie zarówno TPR, jak i G-mean.

Zbiór *new-thyroid*, jak wspomniano wcześniej, jest zbiorem łatwym, co widać po tym, że wszystkie klasyfikatory radzą sobie z nim dość dobrze nawet bez filtrowania. Większość metod przetwarzania odrobinę pogarsza miary klasyfikacji takie jak TPR czy G-mean. Warto zauważyć, że *COST* dla sąsiedztwa 5-NN i $\tau \leqslant 0{,}2$ wypada porównywalnie z metodą *SMOTE*.

Na trudniejszym zbiorze *haberman* udaje się zauważyć wyraźne podniesienie warto-

ści TPR oraz poprawę G-mean. Przy klasyfikacji za pomocą pięciu najbliższych sąsiadów metoda *COST* radzi sobie lepiej od pozostałych filtrów już dla $\tau = 0{,}2$. W przypadku pozostałych klasyfikatorów wypada zazwyczaj nieco gorzej niż *NCR* i porównywalnie z pozostałymi metodami.

Inaczej jest w przypadku zbioru *ecoli* – przy użyciu klasyfikatora 5-NN *COST* pogarsza G-mean, a TPR polepsza gorzej niż *NCR*, *SMOTE*, czy losowe dogenerowanie przykładów, które uzyskało najwyższą wartość TPR, kosztem niższej G-mean. Dla pozostałych klasyfikatorów jest podobnie – *COST* poprawia nieco czułość kosztem G-mean, i wypada odrobinę gorzej niż pozostałe zastosowane filtry – które jednak z reguły również pogarszają wartość G-mean. Dodatkowo widać tutaj trudność parametryzowania *COST* – zależnie od klasyfikatora lepszy okazuje się zbiór przefiltrowany z kosztem τ o wartości 0,1, 0,2 lub 0,3.

Dla zbioru *transfusion* metoda *COST*, przy użyciu niskich τ, uzyskuje zazwyczaj drobne polepszenie G-mean. Warto zauważyć, że w poprawie G-mean wypada zazwyczaj nieco gorzej niż *NCR*, choć uzyskuje od niej wyraźnie lepszą czułość. Dodatkowo *COST* to jedyna metoda przetwarzania wstępnego, która wyraźnie poprawiła klasyfikację w przypadku użycia naiwnego klasyfikatora bayesowskiego. Może to wynikać z probabilistycznej natury tego algorytmu i tego, że *COST* jawnie stara się poprawić rozkład prawdopodobieństwa używany w regule klasyfikacji implementowanej przez ten klasyfikator. Problemem przy tym zbiorze jednak był jego rozmiar – filtrowanie dla $\tau - 0{,}1$ trwało ponad 10 godzin dla każdego podzbioru uczącego dla kroswalidacji. Przygotowanie zbioru do eksperymentu zajęło ok. tygodnia.

Najtrudniejszym i największym zbiorem był *yeast-ME2*. W jego przypadku najlepszą poprawę TPR uzyskuje metoda *SMOTE*, jednak z powodu dodawania syntetycznych przykładów psuje ona mocno precyzję i w efekcie G-mean. Również dobrą poprawę czułości wykazuje tutaj *Random Oversampling*, co może wynikać z tego, że, podobnie jak *SMOTE*, nadlosowuje on przykłady aż do zupełnego zrównoważenia obu klas decyzyjnych, a w zbiorze tym występuje bardzo duże niezrównoważenie klas (przykłady mniejszościowe stanowią zaledwie 3,4 % całego zbioru). *COST* w tym wypadku dodawał relatywnie mało przykładów (średnio 166 dodanych dla $\tau = 0{,}1$, w porównaniu do 1243 dodawanych przez *SMOTE* i nadlosowanie), osiągając G-mean porównywalne do pozostałych metod. W wypadku klasyfikatora *PART* wypada lepiej od *SPIDER*-a, choć gorzej od pozostałych metod, w przypadku *Naïve Bayesa* osiąga G-mean lepsze od pozostałych testowanych algorytmów przetwarzania wstępnego z wyjątkiem algorytmu *NCR*, choć dla najniższego τ uzyskuje nieco lepszą czułość.

Można zatem podsumować, że gdy czas i moc obliczeniowa wymagane do przetworzenia danych przed nauczeniem klasyfikatora nie są problemem, użycie metody *COST* z kosztem z zakresu $[0{,}1;\ 0{,}3]$ może być opłacalne, szczególnie, jeśli zamierza się później stosować klasyfikator *k*-NN lub *Naïve Base*. Dodatkową zaletą tego rozwiązania jest również to, że wynikowy zbiór, choć większy od oryginalnego, to nie dąży do pełnego zrównoważenia klas, dzięki czemu jego rozmiar jest mniejszy niż po losowym dogenerowaniu przykładów.

Jednak, ze względu na złożoność obliczeniową *COST*, w większości sytuacji bardziej opłacalne wydaje się użycie metody oczyszczenia sąsiedztwa – *NCR*, która zazwyczaj

powoduje lepszą poprawę miary G-mean od pozostałych badanych filtrów. Natomiast przy dużym nacisku na czułość podczas klasyfikacji, najszybszą metodą pozostaje losowe dogenerowanie przykładów klasy mniejszościowej.

Rozdział 6

Podsumowanie

W pracy opisano zaproponowaną w [SSK] metodę przetwarzania wstępnego niezrównoważonych liczebnie zbiorów danych w oparciu o ich probabilistyczną interpretację, nazywaną *COST*, z ang. *Cost-Optimized Sampling Technique*.

Przedstawiono wykonaną implementację nieco zmodyfikowanej wersji metody *COST*. Jak pokazują wyniki eksperymentów z rozdziału 5, zbliża ona, za pomocą zwielokrotniania wybieranych przykładów, rozkład obserwowanych prawdopodobieństw do docelowych wartości obliczanych sposobem przedstawionym w rozdziale 3. Eksperymenty przeprowadzone na zbiorach dwuwymiarowych pokazują, że dodaje przykłady głównie w obszarach na granicy klas i wokół przykładów odstających. Dla niewielkich wartości τ wzmacnia klasę mniejszościową i poprawia trafność klasyfikacji przykładów tej klasy w rzeczywistych zbiorach danych przy użyciu różnych klasyfikatorów.

Metoda *COST* nie tworzy żadnych sztucznych przykładów, natomiast modyfikuje zbiór poprzez kopiowanie przykładów istniejących, wybierając zazwyczaj takie, które znajdują się na granicy klas, czy w innych obszarach, sprawiających trudność klasyfikatorom. Dodatkowo, ze względu na opisaną metodę estymacji wartości oczekiwanych docelowego rozkładu klas, algorytm bierze pod uwagę pewność, z jaką może określić prawdopodobieństwo pojawienia się w danym sąsiedztwie przykładu z danej klasy.

Zaimplementowana klasa filtru *Weki* charakteryzuje się kilkoma różnicami w stosunku do wersji metody zaproponowanej w [SSK]. Nie oblicza ona dywergencji Kullbacka-Leiblera między rozkładami prawdopodobieństw wystąpienia przykładów, a jedynie sumę kwadratów różnic między prawdopodobieństwami warunkowymi w branych pod uwagę sąsiedztwach. Oblicza ona również tę miarę w sąsiedztwach punktów odpowiadających przykładom z oryginalnego zbioru danych, zamiast generowania siatki równoodległych punktów. Dodatkowo służy jedynie do dogenerowania nowych przykładów w filtrowanym zbiorze – podczas gdy wersja zaproponowana w [SSK] ma charakter hybrydowy (z elementami *undersamplingu* – buduje nowy zbiór od zera na podstawie oryginalnych przykładów), przygotowany na potrzeby tej pracy algorytm buduje *nadzbiór* zbioru oryginalnego.

Z wyników eksperymentów widać, że *COST* może poprawić czułości klasyfikacji przykładów mniejszościowych podobnie do metody *SMOTE* (a dla niektórych klasyfikatorów lepiej), przy jednocześnie mniejszym wynikowym zbiorze uczącym. Poprawia również często miarę G-mean, choć poprawa ta nie jest tak silna jak w przypadku metod usuwania przykładów większościowych, np. metodą *NCR*.

Niestety, czas obliczeń wykonanej implementacji nie jest zadowalający. Ponieważ czas wykonania jednej iteracji algorytmu zależy od rozmiaru zbioru danych w trzeciej potędze, filtrowanie zbiorów składających się z więcej niż trzystu przykładów może zająć od kilku do kilkunastu godzin. W przypadku sąsiedztwa opartego o jądrową estymację gęstości funkcją Gaussa pomaga optymalizacja, zmniejszająca złożoność obliczeniową, dzięki czemu czas dodania lub usunięcia przykładu zależy od kwadratu rozmiaru zbioru, jednak w tym przypadku algorytm ma tak duże możliwości zmniejszania odmienności między obserwowanym a docelowym rozkładem prawdopodobieństwa, że wykonuje bardzo dużą liczbę iteracji, co również powoduje bardzo długi czas filtrowania – problem ten prawdopodobnie dałoby się rozwiązać, dodając kolejny warunek zatrzymania algorytmu, np. gdy poprawka błędu w kolejnej iteracji jest mniejsza od zadanego progu.

Pozwala to na dalszy rozwój tak implementacji, jak i samego algorytmu. W celu przyśpieszenia jego działania można modyfikować warunki zatrzymania, lub zupełnie zmienić metodę optymalizacji rozkładu prawdopodobieństwa, by nie opierać się na zachłannym, heurystycznym, algorytmie wspinaczki po najbardziej stromym zboczu. Być może również inny wybór punktów, w których sąsiedztwach działa algorytm, mógłby go wyraźnie przyśpieszyć – może siatka 100 równoodległych punktów by wystarczyła dla zbioru o kilku tysiącach przykładach opisanych kilkoma atrybutami?

W ramach dalszej pracy można również wykonać pełną implementację metody *COST*, zaproponowanej w [SSK], bez zmian i uproszczeń występujących w wykonanej klasie `COSTFilter`, by przetestować jaki wpływ miały one na działanie zaproponowanego algorytmu.

Dodatek A

Załączniki

Do pracy dołączona jest płyta CD, na której znajduje się praca w wersji elektronicznej i pliki źródłowe stworzonego projektu.

Na płycie znajdują się również pliki z przetworzonymi metodą *COST* zbiorami danych oraz plik z podsumowaniem filtrowania – zawierający informację o liczbie przykładów dodanych do zbiorów rzeczywistych i czasach wykonywania programu podczas filtrowania tych zbiorów.

Spis rysunków

Spis algorytmów

Literatura

[BSL09] Chumphol Bunkhumpornpat, Krung Sinapiromsaran i Chidchanok Lursinsap. „Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem". W: *Advances in Knowledge Discovery and Data Mining*. Thanaruk Theeramunkong i in. (red.) T. 5476. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, s. 475–482. ISBN: 978-3-642-01306-5. DOI: 10.1007/978-3-642-01307-2_43.

[CS98] P. L. Chan i S. J. Stolfo. „Toward Scalable Learning with Non-uniform Class and Cost Distributions: A Case Study in Creidt Card Fraud Detection". W: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*. Rakesh Agrawal i Paul Stolorz (red.) AAAI Press, 1998, s. 164–168. ISBN: 978-1-57735-070-5.

[Cha+02] Nitesh V. Chawla i in. „SMOTE: Synthetic Minority Over-sampling Technique". W: *Journal of Artificial Intelligence Research* 16.1 (2002), s. 321–357. ISSN: 1076-9757.

[CT06] Thomas M. Cover i Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 2006, s. 19. ISBN: 978-0-471-24195-9.

[Fel07] William Feller. *Wstęp do rachunku prawdopodobieństwa*. Wyd. szóste. Warszawa: Wydawnictwo Naukowe PWN, 2007. ISBN: 978-8301146849.

[Gal+09] M. Galassi i in. *GNU Scientific Library Reference Manual*. 3rd Ed. Network Theory Ltd., 2009. ISBN: 0954612078, 9780954612078. URL: http://www.gnu.org/software/gsl/.

[Hal+09] Mark Hall i in. „The WEKA Data Mining Software: An Update". W: *SIGKDD Explorations* 11.1 (list. 2009), s. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[HWM05] Hui Han, Wen-Yuan Wang i Bing-Huan Mao. „Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning". W: *Advances in Intelligent Computing*. De-Shuang Huang, Xiao-Ping Zhang i Guang-Bin Huang (red.) T. 3644. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, s. 878–887. ISBN: 978-3-540-28226-6. DOI: 10.1007/11538059_91.

[HG09] Haibo He i Edwardo A. Garcia. „Learning from Imbalanced Data". W: *IEEE Transactions on Knowledge and Data Engineering* 9 (wrz. 2009), s. 1263–1284.

[JS01] Jacek Jakubowski i Rafał Sztencel. *Wstęp do teorii prawdopodobieństwa*. SCRIPT, 2001. ISBN: 83-904564-5-1.

[Jap01] Nathalie Japkowicz. „Concept-Learning in the Presence of Between-Class and Within-Class Imbalances". W: *Advances in Artificial Intelligence*. Eleni Stroulia i Stan Matwin (red.) T. 2056. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, s. 67–77. ISBN: 978-3-540-42144-3. DOI: 10.1007/3-540-45153-6_7.

[KKL15] Georg Krempl, Daniel Kottke i Vincent Lemaire. „Optimised probabilistic active learning (OPAL)". W: *Machine Learning* 100.2-3 (2015), s. 449–476. ISSN: 0885-6125. DOI: 10.1007/s10994-015-5504-1.

[KKS14] Georg Krempl, Daniel Kottke i Myra Spiliopoulou. „Probabilistic Active Learning: Towards Combining Versatility, Optimality and Efficiency". W: *Discovery Science*. Sašo Džeroski i in. (red.) T. 8777. Lecture Notes in Computer Science. Springer International Publishing, 2014, s. 168–179. ISBN: 978-3-319-11811-6. DOI: 10.1007/978-3-319-11812-3_15.

[KHM98] Miroslav Kubat, Robert C. Holte i Stan Matwin. „Machine Learning for the Detection of Oil Spills in Satellite Radar Images". W: *Machine Learning* 30.2-3 (1998), s. 195–215. ISSN: 0885-6125. DOI: 10.1023/A:1007452223027.

[Lau01] Jorma Laurikkala. „Improving Identification of Difficult Small Classes by Balancing Class Distribution". W: *Artificial Intelligence in Medicine*. Silvana Quaglini, Pedro Barahona i Steen Andreassen (red.) T. 2101. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, s. 63–66. ISBN: 978-3-540-42294-5. DOI: 10.1007/3-540-48229-6_9.

[LL98] Charles X. Ling i Chenghui Li. „Data Mining for Direct Marketing: Problems and Solutions". W: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*. Rakesh Agrawal i Paul Stolorz (red.) AAAI Press, 1998, s. 73–79. ISBN: 978-1-57735-070-5.

[LS08] Charles X. Ling i Victor S. Sheng. „Cost-Sensitive Learning and the Class Imbalance Problem". W: *Encyclopedia of Machine Learning*. Claude Sammut i Geoffrey Webb (red.) Springer US, 2008, s. 231–235. ISBN: 978-0-387-30768-8. DOI: 10.1007/978-0-387-30164-8_181.

[MS11] T. Maciejewski i J. Stefanowski. „Local Neighbourhood Extension of SMOTE for mining imbalanced data". W: *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*. Kw. 2011, s. 104–111. DOI: 10.1109/CIDM.2011.5949434.

[Mac10] Tomasz Maciejewski. „Metody z rodziny SMOTE dla selektywnego przetwarzania danych z niezrównoważonymi liczebnie klasami decyzyjnymi". Prac. mag. Poznań, Polska: Politechnika Poznańska, 2010.

[Mar00] Dragos D. Margineantu. „When Does Imbalanced Data Require more than Cost-Sensitive Learning?" W: *Workshop on Learning from Imbalanced Data, National Conference on Artificial Intelligence*. AAAI Technical Report WS-00-05. 2000.

[Mat09] R. J. Mathar. „A Java Math.BigDecimal Implementation of Core Mathematical Functions". W: *ArXiv e-prints* (2009). arXiv: 0908.3030v3 [math.NA]. URL: http://arxiv.org/abs/0908.3030v3.

[Nap12] Krystyna Napierała. „Improving Rule Classifiers For Imbalanced Data". Prac. dokt. Poznań, Polska: Politechnika Poznańska, 2012.

[NSW10] Krystyna Napierała, Jerzy Stefanowski i Szymon Wilk. „Learning from Imbalanced Data in Presence of Noisy and Borderline Examples". W: *Rough Sets and Current Trends in Computing*. Marcin Szczuka i in. (red.) T. 6086. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, s. 158–167. ISBN: 978-3-642-13528-6. DOI: 10.1007/978-3-642-13529-3_18.

[SSK] Myra Spiliopoulou, Jerzy Stefanowski i Georg Krempl. „COST: A Cost-Optimal Sampling Technique". Maszynopis wewnętrzny, wersja z 02.06.2015.

[SW07] Jerzy Stefanowski i Szymon Wilk. „Improving rule based classifiers induced by MODLEM by selective pre-processing of imbalanced data". W: *Proceedings of the RSKD Workshop at ECML/PKDD, Warsaw*. 2007, s. 54–65.

[SW08] Jerzy Stefanowski i Szymon Wilk. „Selective Pre-processing of Imbalanced Data for Improving Classification Performance". W: *Data Warehousing and Knowledge Discovery*. Il-Yeol Song, Johann Eder i ThoManh Nguyen (red.) T. 5182. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, s. 283–292. ISBN: 978-3-540-85835-5. DOI: 10.1007/978-3-540-85836-2_27.

[Weka] Dokumentacja pakietu Weka. *Class IBk*. URL: http://weka.sourceforge.net/doc.stable/weka/classifiers/lazy/IBk.html (data dostępu: 01. 09. 2015).

[Wekb] Dokumentacja pakietu Weka. *Class LinearNNSearch*. URL: http://weka.sourceforge.net/doc.stable/weka/core/neighboursearch/LinearNNSearch.html (data dostępu: 01. 09. 2015).

[Wekc] Dokumentacja pakietu Weka. *Class SMOTE*. URL: http://weka.sourceforge.net/doc.packages/SMOTE/weka/filters/supervised/instance/SMOTE.html (data dostępu: 01. 09. 2015).

[ZE01] Bianca Zadrozny i Charles Elkan. „Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". W: *In Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann, 2001, s. 609–616.

Instytut Informatyki, Wydział Informatyki
Politechnika Poznańska

Skład przy użyciu systemu XeLaTeX.

BibTeX:

```
@mastersthesis{ key,
    author = "Benedykt Jakub Jaworski",
    title = "Optymalizacja losowania przykładów w danych niezrównoważonych z wykorzystaniem uczenia się
z kosztami",
    school = "Poznań University of Technology",
    address = "Poznań, Poland",
    year = "2015",
}
```