POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: SYSTEMY INFORMATYKI W MEDYCYNIE

PRACA DYPLOMOWA MAGISTERSKA

Analiza porównawcza wybranych metod selekcji cech w zadaniu klasyfikacji danych niezbalansowanych.

Comparative analysis of feature selection techniques in inbalanced data classification problems.

AUTOR:
Maciej Hajduk

PROWADZĄCY PRACĘ:
dr inż. Paweł Trajdos, K02W04D03

WROCŁAW, 2021

Spis treści

Spis rysunków

Spis tablic

1 Wstęp

1.1 Wprowadzenie

Celem pracy było porównanie różnych metod selekcji cech w problemie trenowania algorytmów uczenia maszynowego na danych niezbalansowanych. W jej ramach, przedstawione i opisane zostały popularne obecnie metody selekcji oraz przeprowadzone zostały eksperymenty dla przykadowych zbiorów danych, zarówno rzeczywistych jak i syntetycznych, celem których było stworzenie rankingu algorytmów. Autor sprawdził, jak właściwie przeprowadzona selekcja wpływa na jakość wyników dostarczanych przez program klasyfikujący dane i jak przytoczone przez niego metody radzą sobie z danymi, w których występuje znaczna przewaga liczebności jednej klasy. Aspekt inżynieryjny polegał na implementacji zaproponowanych w pracy eksperymentów, co pozwoliło na kompleksowe porównanie algorytmów.

Praca swoim zakresem objęła porównanie popularnych metod selekcji cech w ramach kilku wybranych zbiorów danych. Napisana w jej ramach biblioteka pozwoliła na stworzenie eksperymentów kompleksowo porównujących wyniki różnych algorytmów testowanych na kilkuset zestawach danych. Dane te posłużyć mogą do wyboru najodpowiedniejszej metody we wszelkich problemach klasyfikacyjnych, w których elementy wykazują szczególną nadreprezentacje jednej bądź kilku klas. Zamieszczone w pracy podsumowanie zawiera wyniki przeprowadzonych przez autora badań.

Praca składa się z czterech rozdziałów:

Rozdział pierwszy: Omówienie analizy wybranego problemu, przedstawienie motywacji podjęcia tego tematu oraz przegląd literatury. Opisano w nim podstawowe metody selekcji cech i wyjaśnienie różnicy pomiędzy selekcją oraz ekstrakcją cech. Zawarto opis szczegółowej charakterystyki zagadnienia i problemu jakim jest niezrównoważony rozkład klas w algorytmie uczenia maszynowego. W rozdziale zamieszczono szczegółowy opis poszczególnych, wykorzystanych później metod selekcji, wraz z ich matematyczną interpretacją.

Rozdział drugi: Założenia i plan eksperymentu. W rozdziale drugim zawarto informacje związane z inżynieryjnym aspektem pracy, czyli projekt systemu, szczegółowy plan poszczególnych eksperymentów i opis zbiorów danych, jakie użyto podczas doświadczeń. Swoim zakresem rozdział objął krótki opis użytych przez autora bibliotek oraz wykorzystywanych funkcji.

Rozdział trzeci: Podsumowanie uzyskanych wyników. W rozdziale zawarto rezultaty eksperymentów oraz testów opisanych w rozdziale drugim. Zamieszczone table i wykresy pozwalają na czytelną interpretacje danych.

Rozdział czwarty: W czwartym czwartym zawarto interpretację wyników oraz konfrontację ich

z hipotezą postawioną na początku pracy. Wyjaśniono potencjalne zastosowanie przeprowadzonych w ramach pracy eksperymentów. Przedstawiono możliwości rozwoju projektu.

Udało się zrealizować wszystkie postawione cele.

2 Analiza problemu

Uczenie maszynowe to bardzo dynamicznie rozwijająca się gałąź informatyki. Niezwykły rozwój wynika z zapotrzebowania na wykrywanie prawidłowości, na uogólnianie oraz precyzowanie danych. Takie możliwości pozwoliły na znalezienia zastosowania dla algorytmów sztucznej inteligencji w wielu różnych branżach - począwszy od medycyny, poprzez finanse, produkcję i przemysł rozrywkowy. Tak duży przekrój różnych zastosowań wymaga ciągłego ulepszania istniejących już wzorców oraz tworzenia nowych, lepszych i bardziej efektywnych algorytmów. W większości praktycznych problemów klasyfikacji obiektów, autor operuje na dużej liczbie cech. Warto jednak pamiętać, że w tym przypadku *wiele* - nie oznacza lepszych rezultatów. Należy przytoczyć pojęcie "*przekleństwa wielowymiarowości*" [1]. Oznacza ono, że większy wymiar wymaga od programisty znacznie większej liczby danych, dodatkowo wraz ze wzrostem liczby cech wykładniczo rośnie liczba możliwych wariantów dopasowań, co znacznie zwiększa złożoność obliczeniową algorytmów.

Aby uniknąć problemów generowanych przez zbyt dużą ilość atrybutów, a jednocześnie wykorzystać te, które zapewniają jak najlepszą separowalność klas, zazwyczaj pierwszym krokiem w zadaniu klasyfikacji jest selekcja lub ekstrakcja najodpowiedniejszych cech [2].

2.1 Przegląd literatury

Zarówno sam problem selekcji cech jak i sposoby radzenia sobie z nierównomiernym rozkładem klas to - szczególnie w ostatnich latach - często poruszany problem, co skutkuje dużym przekrojem prac, również w ujęciu czysto dziedzinowym - jak wykorzystanie konkretnych algorytmów dla bardzo konkretnych zastosowań.

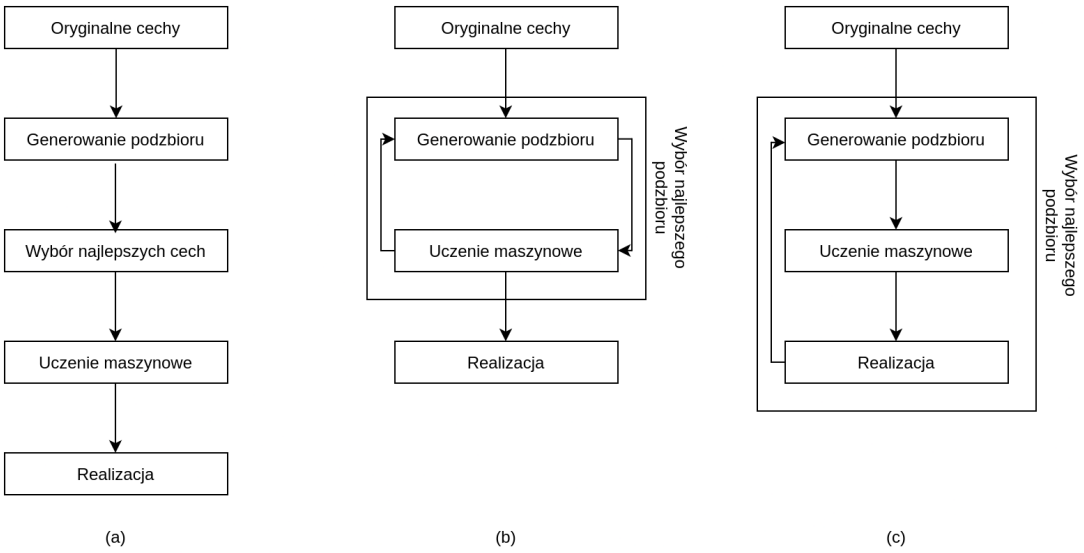
2.2 Cel selekcji cech

Selekcja cech polega na identyfikacja tych elementów puli atrybutów, które uznawane są za najlepsze deskryptory rozważanych kategorii. Zaletą selekcji jest możliwość zbadania tych deskryptorów, które są istotne z punktu widzenia danego zadania klasyfikacji, czyli jednocześnie zrozumienia różnic między analizowanymi kategoriami. Wybór najbardziej istotnych atrybutów badanych obiektów przekłada się bezpośrednio na poprawne działanie klasyfikatora [hallmark]. Dyspozycja coraz większymi bazami danych zmusza do optymalizacji procesu. Gwałtownie rosnąca liczba cech stanowi poważny problem - powoduje nie tylko wydłużenie procesu uczenia oraz wzrost złożoności klasyfikatora, ale niesie ze sobą także ryzyko spadku prawdopodobieństwa poprawnej klasyfikacji. Związane jest to z tak zwanym "przekleństwem wymiarowości" [3]. Zjawisko to zachodzi zazwyczaj, gdy liczba cech znacznie przewyższa liczebność samego zbioru danych. Zadaniem selekcji jest również lepsze zrozumienie problemu oraz zmniejszenie kosztów archiwizacji przyszłych danych. W kolejnych rozdziałach opisane zostaną trzy główne

metody tworzenia algorytmów selekcji: metody rankingowe - zwane filtrami, metody opakowane oraz metody wbudowane [4]. Dla każdej z wymienionych metod zostanie określona idea, oraz przedstawione zostaną algorytmy reprezentujące daną metodologie [2][5][3].

2.3 Podstawowy podział

W rozdziałach 2.3.1 - 2.3.3 opisane zostały trzy główne metody tworzenia algorytmów selekcji. Podział ten i działanie każdej z metod obrazuje rysunek 2.1.



Rysunek 2.1: Trzy główne podejścia do problemu selekcji cech - metody rankingowe (a), opakowane (b) oraz wbudowane (c).

2.3.1 Metody rankingowe

Najprostsze podejście do problemu selekcji cech reprezentowane jest poprzez metody rankingowe, nazywane też filtrami [4]. Jak sama nazwa wskazuje, do zadania selekcji przy pomocy metod rankingowych należy podejść wyróżniając w zbiorze cech następujące grupy: cechy istotne, nieistotne i redundantne. Istotne - to takie, które odróżniają od siebie klasy, nieistotne nie niosą informacji dla problemu klasyfikacji, a cechy redundantne to atrybuty których role z powodzeniem mogą przejąć inne cechy. Metody rankingowe polegają więc na znalezieniu pewnej miary pozwalającej stworzyć ranking cech, a następnie wybrać najlepsze, a odrzucić najgorsze atrybuty. Metody rankingowe zazwyczaj są najszybsze i - co istotne - nie zależą one od używanej metody analizy danych [4, 17]. Ich istotną wadą stanowi brak możliwości uwzględnienia zależności pomiędzy cechami [6]. Kolejne opisane typy metod selekcji tej wady nie posiadają. Podejście rankingowe do selekcji cech zostało zobrazowane na rysunku 2.1, w podpunkcie (a). Istnieje wiele popularnych filtrów, wśród których wyróżnić można najbardziej popularne - *chi-kwadrat*, *ANOVA* oraz *RELIEF* [6].

2.3.2 Metody opakowane

Podstawowymi i najpopularniejszymi metodami selekcji cech są metody opakowane, tak zwane wrappery [4]. W przeciwieństwie do metod rankingowych, w których algorytm selekcji i klasyfikator pozostają niezależne, w algorytmach opakowanych, ocena atrybutów dokonuje się przy użyciu konkretnego modelu. To właśnie efektywność samego klasyfikatora służy za miarę skuteczności metody. Zaletą metody jest jej uniwersalność i dokładność, natomiast wadą - wysoka złożoność obliczeniowa [4]. Dla efektywności tych algorytmów istotny jest sposób ustalania podzbioru cech [4, 17]. Wśród wielu sposobów wyszukiwania właściwego podzbioru, wyróżnić można najprostszy - przeszukanie całego zbioru podzbiorów. Jest to jednak rozwiązanie bardzo kosztowne. Wobec tego typowymi strategiami są: przeszukiwanie w przód, przeszukiwanie wstecz oraz tworzenie indywidualnego rankingu [6]. Popularnym algorytmem implementującym metodę opakowaną jest *RFE - Rekurencyjna eliminacja cech* [4]. Podejście opakowane do selekcji cech zostało zobrazowane na rysunku 2.1, w podpunkcie (b).

2.3.3 Metody wbudowane

Metody wbudowane zawierają się w algorytmie klasyfikacji i to na etapie tworzenia modelu przypisuje się poszczególnym cechom wagi lub przeprowadza się ich eliminację. Do algorytmów klasyfikacji z wbudowaną metodą selekcji zaliczyć można popularne LASSO i RIDGE [7][8][9]. W literaturze natknąć się też można na podpięcie do tej kategorii metody wektorów nośnych (SVM) czy też analizy składowych głównych (PCA) [7]. Zaletą tych metod jest ich szybkość, ponieważ użycie ich nie wiąże się z dodatkowymi operacjami na zbiorze [7][4][9]. Podejście wbudowane do selekcji cech zostało zobrazowane na rysunku 2.1, w podpunkcie (c).

2.4 Selekcja cech a ekstrakcja

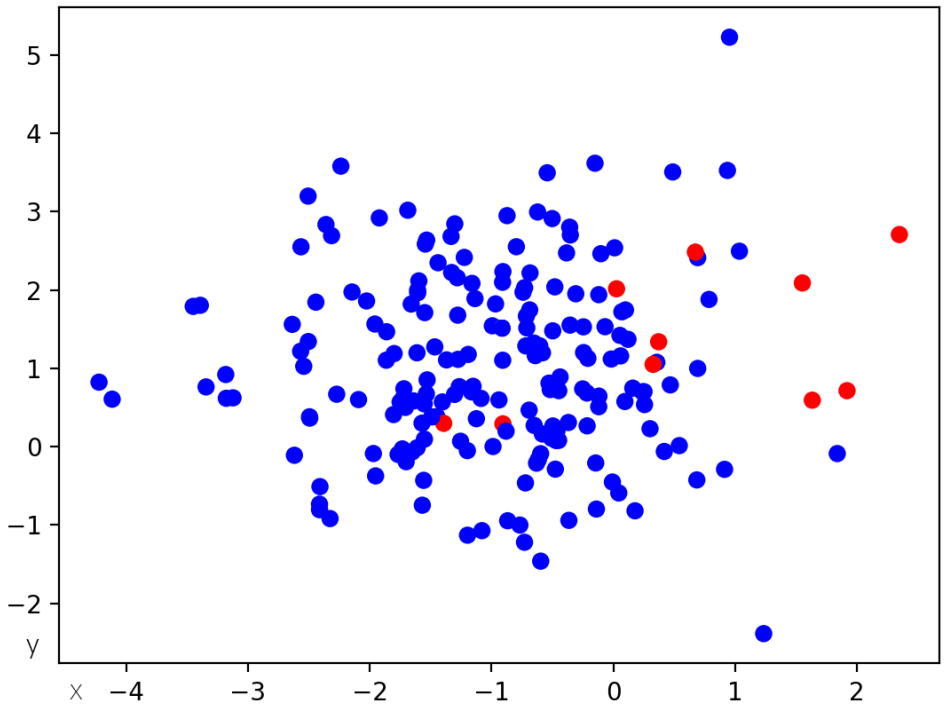
Selekcja cech ma na celu wybranie pewnych atrybutów opisujących dane pod kątem tego, czy nadają się one do dalszego wykorzystania w klasyfikacji, przy jednoczesnym odrzuceniu innych danych [10]. Zawsze rozważana jest ona w kontekście kolejnych zadań i nie można oceniać jej skuteczności w oderwaniu od wyników metody klasyfikacji wykorzystującej wybrane cechy. W większości przypadków budowany jest złożony model, który może zawierać jeden lub więcej algorytmów selekcji i co najmniej jeden klasyfikator.

Ekstrakcja cech natomiast polega na utworzeniu nowego zestawu atrybutów poprzez liniową lub nieliniową kombinację oryginalnych danych. W przeciwieństwie do selekcji, gdzie celem jest zawsze uzyskanie podzbioru wszystkich atrybutów, wykorzystanie ekstrakcji wiąże się z wymiarem przestrzennym mniejszym, równym lub nawet większym od wymiaru przestrzeni startowej [11][10]. Poprzez proces ekstrakcji, część początkowych cech zostaje utracona. Z tego powodu nie ekstrakcja, a selekcja cech jest obecnie najpowszechniejszą strategią służącą przygotowaniu reprezentacji analizowanych danych [11].

2.5 Problem niezrównoważonego rozkładu klas

Wśród wielu dobrze zbadanych i szeroko wykorzystywanych rozwiązań bazujących na uczeniu maszynowym, najbardziej obiecującymi są te, mające ratować ludzkie życie. Złożone choroby, które trudno jest wykryć w ich początkowych stadiach, stanowią poważne dla niego zagrożenie. Postęp w dziedzinie sztucznej inteligencji i metodach statystycznych stworzyły nowe możliwości klasyfikacji i diagnozy najbardziej śmiertelnych chorób, takich jak rak, choroba Alzheimera, cukrzyca itp [12]. Z przypadkami takimi wiąże się jednak problem niezrównoważonej dystrybucji klas [13].

Niezrównoważony rozkład ma miejsce, gdy co najmniej jedna z klas jest niewystarczająco reprezentowana i przytłoczona przez inne klasy. Algorytmy klasyfikacji często nie radzą sobie z niezrównoważonymi danych danymi, co stwarza wiele przeszkód w uczeniu się algorytmów i przedstawia liczne konsekwencje dla rzeczywistych zastosowań. Problem objawia się niedocenianiem przykładów klas mniejszościowych i powoduje niedokładne wyniki klasyfikacji w stosunku do przykładów klas większościowych. Klasyfikacja niezrównoważonego zbioru danych staje się trudniejsza przy ograniczonej liczbie próbek i ogromnej liczbie cech. Przykład takiej sytuacji zaobserwować można na rysunku 2.2. Ilustracja przedstawia 200 elementów z których tylko 5% należy do klasy mniejszościowej - czerwonej.



Rysunek 2.2: Przykład niezrównoważonego rozkładu klas.

Sytuacja przedstawiona na grafice jest dla algorytmu niewygodna, ponieważ większość tradycyjnych algorytmów uczenia maszynowego trenowana na podobnym zbiorze, obciążona zostanie w stosunku do klasy bardziej licznej [14]. Jednocześnie, zazwyczaj lepsze zrozumienie klas mniej licznych jest istotniejsze z punktu widzenia problemu w ujęciu biznesowym [13]. Problemem jest również określenie jakości wyników algorytmu. Jakość klasyfikacji - czyli dokładność, używana jako metryka ewaluacji może być w takim przypadku niewystarczająca, gdyż nawet model o skuteczności 95% - co jest na ogół wartością bardzo dobrą - mógłby w tym przypadku nie rozpoznawać żadnego elementu klasy mniejszościowej [14].

2.6 Metody klasyfikacji danych niezbalansowanych

Problem nierównoważnego rozkładu przyciąga w ostatnim czasie zainteresowanie dużej części społeczności zajmującej się uczeniem maszynowym i eksploracją danych, zarówno ze środowisk akademickich jak i w przemyśle, co znajduje odbicie w dużej liczbie startupów opierających swoje produkty i usługi na rozwiązaniach *machine-learningowych*. W ciągu kilkunastu ostatnich lat wyklarowały się trzy główne podejścia do uczenia modeli na danych niezbalansowanych [15][15][16][17][18]. Są to metody na poziomie danych, metody na poziomie algorytmów oraz metody hybrydowe.

2.6.1 Metody na poziomie danych

Metody na poziomie danych (Data-level methods), modyfikują dostępne instancje problemu w celu jego zbalansowania. Można je dalej podzielić na podgrupy: metody próbkowania danych (data-sampling) i metody wyboru cech (feature selection methods) [19]. Metody nadpróbkowania i podpróbkowania stanowią dwie podgrupy metod próbkowania danych, w których próbkowanie danych z danego zbioru danych odbywa się losowo lub z wykorzystaniem określonego algorytmu. W procesie oversamplingu (nadpróbkowania) do danego zbioru danych dodawane są instancje klasy mniejszościowej (poprzez replikację), gdzie replikacja odbywa się losowo lub z wykorzystaniem algorytmów takich jak ADASYN [18]. W procesie undersamplingu (podpróbkowania) natomiast, większość wystąpień klasy zostanie usuniętych z danego zbioru danych, a usuwanie odbywa się w dużej mierze losowo. SMOTE (Synthetic Minority Over-Sampling), to technika próbkowania polegająca na sztucznym ponownym próbkowaniu zbioru danych. Końcowym jej wynikiem jest zbiór danych o zrównoważonym rozkładzie [16]. Chociaż metoda te może skutkować znacznie lepszymi wynikami w porównaniu z oryginalnym zestawem danych, istnieją poważne problemy związane z jej wykorzystaniem [1]. Po pobraniu zbyt małej liczby próbek wydajność klasyfikatora może ulec pogorszeniu z powodu potencjalnej utraty przydatnych przykładów klasy większościowej. Podobnie dodatkowe przypadki szkoleniowe wprowadzone przez nadmierne próbkowanie mogą zwiększyć złożoność obliczeniową klasyfikatora. W najgorszym przypadku dokładne kopie przykładów po nadmiernym próbkowaniu mogą prowadzić do nadmiernego dopasowania klasyfikatora [19]. Chociaż metody selekcji cech są powszechnie stosowane w celu poprawy wyników klasyfikacji, mogą one również pomóc

w wyborze najbardziej wpływowych cech w celu wygenerowania unikalnej wiedzy w ramach klasyfikacji. Zmniejsza to niekorzystny wpływ nierównowagi klas na wyniki klasyfikacji [19].

2.6.2 Metody na poziomie algorytmów

Metody na poziomie algorytmów (Algorithm-level methods), modyfikują istniejące algorytmy uczenia maszynowego. Można je podzielić na metody wrażliwe na koszty (cost-sensitive methods) i metody zintegrowane [19]. Pierwsza z nich opiera się na zasadzie przypisywania większej wagi instancjom w przypadku błędnej klasyfikacji. Na przykład fałszywie negatywnym przewidywaniom można przypisać wyższy koszt niż fałszywie dodatnim. Metody zintegrowane mogą być również stosowane jako metody wrażliwe na koszty, w przypadku których wynikiem klasyfikacji jest pewna kombinacja wielu klasyfikatorów zbudowanych na zbiorze danych [19]. Dwa powszechne typy metod uczenia zintegrowanego to Bagging i Boosting [20]. Bagging minimalizuje wariancję, generując kilka zestawów uczących z danego zestawu danych i generując klasyfikator dla każdego zestawu uczącego, a następnie łącząc ich modele w celu ostatecznej klasyfikacji. Algorytmy wykorzystujące Boosting, podobne do algorytmu *AdaBoost*, tworzą serię klasyfikatorów, wszystkie stosowane do tego samego zestawu danych [21]. *AdaBoost* wybiera tylko te cechy, o których wiadomo, że poprawiają moc predykcyjną modelu, zmniejszając wymiarowość i potencjalnie poprawiając czas wykonania, ponieważ nieistotne cechy nie muszą być obliczane. Na początku algorytm nadaje wszystkim próbkom równe wagi i prawdopodobieństwa wylosowania w kolejnej iteracji. Po każdej iteracji prawdopodobieństwa są aktualizowane. Próbka, która została poprawnie sklasyfikowana, ma mniejsze prawdopodobieństwo, że zostanie wylosowana w następnej iteracji, a błędnie sklasyfikowana próbka ma większe prawdopodobieństwo. W rezultacie klasyfikator w dalszej części serii tworzy zestaw treningowy składający się z trudnych do sklasyfikowania próbek. Metody uczenia jednoklasowego (OOC) - czyli mającego na celu identyfikacje obiektu jednej, określonej klasy, poprzez uczenie się przede wszystkim ze zbioru zawierającego tylko obiekty tej klasy, mają na celu zwalczenie problemu overfittingu. Występuje on przypadku większości klasyfikatorów uczących się na niezrównoważonych danych. Osiągane jest to poprzez podejście do tego problemu z punktu widzenia uczenia nienadzorowanego [22][23][24]. Algorytmy jednoklasowe są konstruowane w taki sposób, aby rozpoznawać próbki z danej klasy i odrzucać próbki z innych klas.

2.6.3 Podejścia hybrydowe

Metody hybrydowe mają na celu rozwiązanie znanych problemów spowodowanych metodami próbkowania danych, metodami wyboru cech, metodami wrażliwymi na koszty i podstawowymi algorytmami uczenia się (takimi jak Naive Bayes [25]). W niektórych przypadkach podgrupy metod na poziomie danych lub podgrupy metod na poziomie algorytmu można łączyć jako ogólną metodę rozwiązywania problemu niezbalansowania klas. Na przykład popularny klasyfikator losowego lasu (Random Forest) jest wersją oryginalnego algorytmu losowego lasu

decyzyjnego (Random Decision Forest) i jest zintegrowany z algorytmem uczenia się, który dodatkowo implementuje Bagging [26].

2.7 Metody oparte o selekcję cech

Pojęcie przekleństwo wymiarowości mówi, że jeśli wiele cech jest zaszumionych, koszt użycia klasyfikatora może być bardzo wysoki, a wydajność może być poważnie zaniżona [1]. Ponieważ problemowi z nierównowagą klas często towarzyszy problem dużej wymiarowości zbioru danych, zastosowanie technik selekcji cech jest koniecznym działaniem [1]. Pomysłowe techniki próbkowania i metody algorytmiczne mogą nie wystarczyć do walki z wysokowymiarowymi problemami nierównowagi klas.

Van der Putten i van Someren przeanalizowali zbiory danych z CoIL Challenge 2000 i stwierdzili, że wybór cech był bardziej istotny dla dobrych wyników niż wybór algorytmu klasyfikacji i najbardziej pomógł w walce z problemem nadmiernego dopasowania [27]. Forman odnotował podobną obserwację dotyczącą wysoce niezrównoważonych problemów klasyfikacji tekstu i stwierdził, że „żaden stopień sprytnej indukcji nie może zrekompensować braku sygnału predykcyjnego w przestrzeni wejściowej" [28][29][30]. Badania te pokazują, że w wielowymiarowych zbiorach danych, sam dobór cech może zwalczyć problem nierównowagi klas.

W ostatnich latach, radzenie sobie z niezrównoważonymi zbiorami danych za pomocą selekcji cech stało się popularne wśród społeczności zajmujących się eksploracją danych i uczeniem maszynowym [31]. Wspomniane wcześniej techniki koncentrują się na próbkowaniu danych uczących w celu przezwyciężenia niezrównoważonego rozkładu klas. Metoda redukcji cech, taka jak selekcja cech, przyjmuje inne podejście do przezwyciężenia problemu. Ogólna koncepcja polega na uzyskaniu podzbioru cech, które optymalnie korygują dysproporcje między klasami w zbiorze danych i wybierają najlepsze cechy, które reprezentują obie klasy.

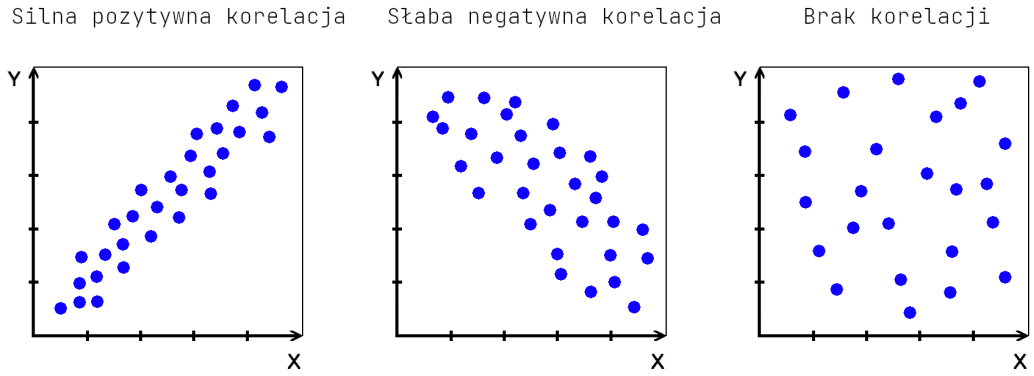
Przedstawione w rozdziałach 2.7.1 - 2.7.5 algorytmy należą do tradycyjnych, szeroko używanych metod selekcji cech [32]. Omówione strategie należą do tak zwanych filtrów, czyli metod rankingowych. Jest to najbardziej naturalne podejście do rozpatrywanego tematu, gdyż opisane algorytmy nie są zależne od wbudowanego klasyfikatora. Pozwala to również na ich kompleksowe i obiektywne porównanie.

2.7.1 Correlation coefficient

Korelacja to miara liniowej zależności pomiędzy dwoma zmiennymi losowymi. Jest to więc poniekąd miara tego, jak silnie jedna zmienna zależy od drugiej. Jest to zazwyczaj bardzo użyteczna właściwość - w przypadku dwóch, silnie skorelowanych zmiennych, posiadając informacje o jednej zmiennej można przewidzieć wartości drugiej. W przypadku liniowych modeli uczenia maszynowego, częstym celem będzie znalezienie elementów silnie skorelowanych ze zmienną losową opisującą przynależność do klasy. Jednocześnie, dwie silnie skorelowane ze sobą zmienne

dostarczają też redundantnych informacji. Zasadniczo można dokonać poprawnego sklasyfikowania z pomocą tylko jednego z tych atrybutów. Usunięcie drugiego może wręcz pomóc w zmniejszeniu wymiarowości i zbędnego szumu [32].

Współczynnik korelacji Pearsona to algorytm, który określa poziom zbieżności liniowej pomiędzy zmiennymi. Wynikiem tej metody są wartości od -1 do 1. Bezwzględna wartość współczynnika określa siłę zależności liniowej - wartości bliższe 1 wskazują na silniejszy związek [33]. Znak współczynnika wskazuje kierunek zależności: znak dodatni wskazuje, że dwie zmienne rosną lub maleją względem siebie (pod względem korelacji), a znak ujemny wskazuje, że jedna zmienna rośnie, a druga maleje [32]. Interpretacje wyników korelacji na przykładowych zbiorach danych zostały przedstawione na rysunku 2.3.



Rysunek 2.3: Interpretacje współczynnika korelacji dla różnych zbiorów danych.

Ogólna postać współczynnika przedstawiona została w równaniu 2.1:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}, \tag{2.1}$$

gdzie:

- n jest liczebnością próbki
- x_i oraz y_i są indywidualnymi próbkami indeksowanymi po i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ - jest średnią arytmetyczną (podobnie dla $\bar{y}$).

Współczynnik korelacji Pearsona można wykorzystać do oceny związku między więcej niż dwiema zmiennym obliczając macierz relacji między każdą parą zmiennych w zbiorze danych. Rezultatem jest symetryczna macierz zwana macierzą korelacji.

2.7.2 Chi-square

Chi-square (chi-kwadrat lub χ^2) jest testem statystycznym mierzącym niezależność cechy od etykiety klasy. Mierzy on zależność między zmiennymi stochastycznymi, więc użycie tej metody

"*usuwa*" cechy, które z największym prawdopodobieństwem są niezależne od klasy, a zatem nie mają znaczenia dla klasyfikacji. Metoda polega na obliczeniu metryki χ^2 pomiędzy wartością docelową a cechą i wyborze zmiennej o maksymalnym wyniku testu [33].

Ogólna postać testu została przedstawiona w równaniu 2.2:

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{O_i - E_i}{E_i} \right)^2, \tag{2.2}$$

gdzie:

- O_i jest wartością mierzoną
- E_i jest wartością oczekiwaną
- n jest liczbą pomiarów.

Forman zauważył, że ten test może zachowywać się nieprawidłowo, gdy spodziewana jest niewielka liczba cech. Jest to dość powszechne w przypadku niezrównoważonych zbiorów danych [29]. Chociaż test chi-kwadrat zadowalająco uogólnia dane dyskretne, nie radzi sobie dobrze się podczas testowania danych ciągłych [6].

2.7.3 Information Gain

Entropia warunkowa to entropia po podziale zbioru przy pomocy danego atrybutu [6]. Dla danego atrybutu a, entropia warunkowa wyraża się wzorem 2.3:

$$Ent(S|a) = \sum_{j=1}^{p} \frac{n_{s_j}}{n} Ent(S_j), \tag{2.3}$$

gdzie:

- p to liczba wartości atrybutu a
- S_j to zbiór przykładów z wartością atrybutu v_j
- n_{s_j} to liczebność zbioru S_j
- $Ent(S_j)$ to entropia zbioru S_j, wyrażona wzorem 2.4:

$$Ent(S) = -\sum_{i=1}^{k} p_i log_2 p_i, \tag{2.4}$$

gdzie:

- p_i to prawdopodobieństwo przynależności do klasy $i - tej$
- S to zbiór przykładów
- k liczba klas.

Im mniejsza wartość entropii warunkowej, tym większa jednorodność podziału [32]. Information Gain mierzy różnicę między entropią etykiet klas a entropią warunkową etykiet klas dla danej cechy [6]. Metoda ta ocenia przyrost informacji przy użyciu atrybutu. Dla danego atrybutu a:

$$IG(S, a) = Ent(S) - Ent(S|a) \tag{2.5}$$

Podobnie jak test chi-kwadrat, metoda *Information Gain* dobrze uogólnia atrybuty dyskretne, ale nie radzi sobie atrybutami z danymi ciągłymi. Ponadto preferuje atrybuty o dużej liczbie wartości i może prowadzić do przeuczenia [32]. Problemy te rozwiązuje zmodyfikowana wersja algorytmu - *Gain Ratio*, który korzysta z tak zwanej *wartości wewnętrznej*, która koryguje obciążenie poprzez dostarczenie danych o wielkości zbioru.

2.7.4 Relief i ReliefF

Grupa algorytmów *Relief* jest jednym z najskuteczniejszych przedstawicieli opracowanych dotychczas metod filtrujących [6]. Większość metod filtrowania opracowanych w celach eksploracji danych i uczenia maszynowego zakłada warunkową niezależność atrybutów. Algorytmy Relief są bardzo pod tym względem rewolucyjne, ponieważ nie zakładają, że atrybuty są od siebie niezależne. Te algorytmy są zależne od kontekstu. Kiedy istnieje silny związek między atrybutami, jakość atrybutów może być poprawnie oszacowana, co sprawia, że algorytm ten jest jednym z najbardziej efektywnych algorytmów przetwarzania wstępnego [6].

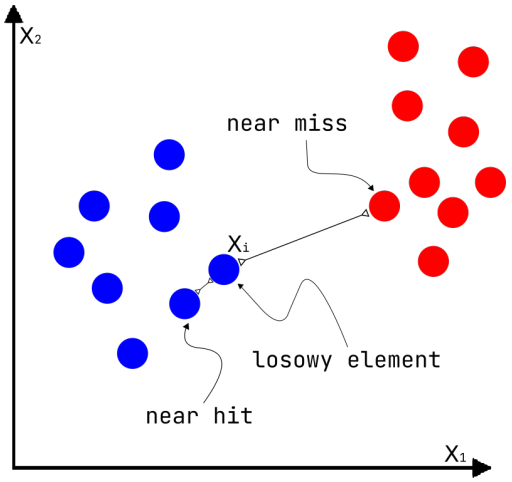
Podstawową ideą algorytmów Relief jest oszacowanie jakości cech na podstawie tego, jak dobrze cecha może rozróżniać instancje, które są blisko siebie. Relief oblicza punktację dla każdej cechy, którą można następnie zastosować do selekcji cech. Punktacja opiera się na wyliczeniu różnic wartości cechy między parami najbliższych sąsiadów. Jeśli różnica wartości cechy zostanie zaobserwowana w sąsiedniej parze instancji z tą samą klasą, wynik funkcji maleje. Jeżeli natomiast różnica wartości cechy zostanie zaobserwowana w sąsiedniej parze instancji z różnymi wartościami klas, wynik funkcji rośnie [34]. Algorytm został pierwotnie zaprojektowany do zastosowania w problemach klasyfikacji binarnej. Relief zainspirował rodzinę algorytmów wyboru cech - RBA (Relief Based Alghoritms), w tym algorytm ReliefF [34]. Jest on dodatkowo przystosowany do działania w problemach wieloklasowych lub zbiorach z niekompletnymi danymi.

Metoda na wejściu przyjmuje wektor wartości cechy i klasy. Algorytm będzie powtarzany m razy i rozpoczyna się z tablicą wag W o długości równej ilości cech, wypełnioną zerami. Podczas każdej iteracji, algorytm rozpatruje wektor cech X należący do losowej instancji i wektory cech instancji najbliższe X (według odległości euklidesowej) z każdej cechy. Najbliższa instancja tej samej klasy nazywana jest *prawie trafioną* (*near hit*), natomiast najbliższa instancja innej klasy - *prawie spudłowaną* (*near miss*). Wektor wag aktualizowany jest według wzoru 2.6:

$$W_i = W_i - (x_i - nearHit_i)^2 + (x_i - nearMiss_i)^2, \tag{2.6}$$

gdzie:

- W_i to $i-ty$ element wektora wag W
- x_i to $i-ty$ element wektora X
- $nearHit_i$ to $i-ty$ element wektora *near hit*
- $nearMiss_i$ to $i-ty$ element wektora *near miss*.



Rysunek 2.4: Wybór elementów *near hit* oraz *near miss* w każdej iteracji działania algorytmu.

Waga danej cechy maleje, jeżeli różni się ona od tej cechy w pobliskich instancjach tej samej klasy bardziej, niż pobliskie instancje innych klas, a wzrasta w przeciwnym przypadku. Po m iteracjach, każdy element wektora W jest dzielony przez m. Tworzy on wtedy ranking cech [34].

Zaletą metod RBA jest to, że nie są zależne od heurystyki, działają w czasie wielomianowym niskiego rzędu, są odporne na zakłócenia, a także nadają się do danych binarnych lub ciągłych [35][36].

2.7.5 ANOVA

ANOVA pochodzi od angielskiego “analysis of variance,” czyli analiza wariancji. Jest to metoda, która wyjaśnia, z jakim prawdopodobieństwem wyodrębnione czynniki mogą być powodem różnic między obserwowanymi średnimi grupowymi. Algorytm polega na porównaniu wariancji międzygrupowej do wariancji wewnątrzgrupowej. ANOVA to jeden z algorytmów statystyki F (F-test) - od nazwiska twórcy, Ronalda Fishera [37].

Analizę wariancji można podzielić na trzy grupy analiz:

- **jednoczynnikowa analiza wariancji** - wpływ jednego czynnika międzygrupowego na zmienną zależną
- **wieloczynnikowa analiza wariancji** - wpływ kliku czynników międzygrupowych na zmienną zależną

- **analiza wariancji dla czynników wewnątrzgrupowych** - wpływ czynnika wewnątrzgrupowego na zmienną zależną, tzw. “powtarzane pomiary”

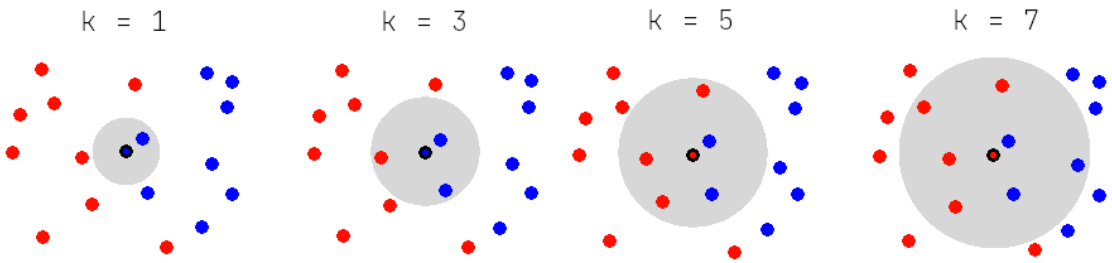
Analiza wariancji to stosunek wariancji, obliczona pomiędzy badanymi grupami a średnią wariancją, zaobserwowaną wewnątrz grup. Analiza ta jest metodą statystyczną pozwalającą na podział zaobserwowanej wariancji wyników na oddzielne części. Analizowana jest wariancja przypadająca na każdy z analizowanych czynników jak również wariancja błędu. Idealna sytuacja ma miejsce wtedy, gdy wariancja międzygrupowa jest duża, natomiast wariancja wewnątrzgrupowa - mała [38][37].

3 Założenia i plan eksperymentu

Obiektywne porównanie przytoczonych algorytmów cech wymaga przeprowadzenia szeregu eksperymentów porównujących skuteczność popularnych metod selekcji cech. Praca swoim zakresem objęła eksperymenty przeprowadzone na kilkuset wybranych zbiorach danych. Użyto zarówno zbiorów rzeczywistych - to znaczy zebranych w ramach rzeczywistych pomiarów jak i syntetycznych - wygenerowanych przez algorytm. Bazy danych, użyte w ramach badań implikują skupienie na problemach dwuklasowych. Hipoteza, z którą twórca konfrontuje wyniki eksperymentów, to założenie że wszystkie, badane metody selekcji poradzą sobie podobnie z postawionym zadaniem, a poza względami wydajnościowymi, nie ma znaczenia funkcja, która zostanie użyta. Technologia, w jakiej zostaną przeprowadzone doświadczenie to jezyk Python w wersji `3.8` oraz biblioteki `sckit-learn` (`https://scikit-learn.org`), `numpy` (`https://numpy.org`). i `pandas` (`https://pandas.pydata.org/`)

3.1 Generowanie wyników

Wybrane przez autora metody selekcji cech należą do grupy tak zwanych filtrów. Tworzą one ranking cech, przydatny do zdefiniowania atrybutów, które będą używane przez algorytm w celu przeprowadzenia klasyfikacji. W celu prawidłowego porównania badanych algorytmów należy sprawdzić ich wyniki w połączeniu z całym procesem klasyfikowania. Zdecydowano się na klasyfikator KNN - K Najbliższych Sąsiadów. W metodzie tej, badany obiekt przydzielany jest do klasy, do której należy większość z jego sąsiadów [37][39].



Rysunek 3.1: Graficzna reprezentacja działania algorytmu KNN - przydzielenie badanej instancji do jednej z dwóch klas na podstawie jego sąsiadów. Obiekt może zostać oetykietowany inaczej w zależności od parametrów, w tym przypadku promienia obszaru.

Algorytm przyjmuje zbiór danych uczących zawierający elementy, z których każda ma przypisaną klasę oraz wektor cech. Dany jest element C z przypisanym wektorem cech $X_1 \dots X_n$ dla którego prognoza odbywa się w następujący sposób:

1. Porównanie wartości cech dla elementu C z wartościami cech dla każdego elementu w zbiorze uczącym.

2. Wybór k (ustalona z góry liczba) najbliższych do C elementów zbioru uczącego.
3. Uśrednienie wartości klasy dla wybranych elementów, w wyniku czego uzyskiwana jest prognoza.

3.2 Ocena działania algorytmów

Określenie jakości działania algorytmu stanowi w badanych przypadkach problem. Dokładność (accuracy) używana jako metryka ewaluacji może być w takim niewystarczająca, gdyż nawet model o skuteczności 95% - co jest na ogół wartością bardzo dobrą - w przypadku rozkładu 5/95 mógłby nie rozpoznawać żadnego elementu klasy mniejszościowej. Metrykami, które dostarczą bardziej wartościowe dane są:

- Macierz konfuzji: tabela pokazująca prawidłowe prognozy i typy nieprawidłowych przewidywań.
- Precyzja: liczba prawdziwie pozytywnych wyników podzielona przez wszystkie pozytywne przewidywania. Precyzja jest również nazywana pozytywną wartością predykcyjną. Jest miarą dokładności klasyfikatora. Niska precyzja wskazuje na dużą liczbę fałszywych wyników.
- Czułość: liczba prawdziwie pozytywnych wyników podzielona przez liczbę dodatnich wartości w danych testowych. Jest miarą kompletności klasyfikatora. Niska czułość wskazuje na dużą liczbę fałszywie negatywnych wyników.
- F1 Score: średnia harmoniczna precyzji i czułości, wyrażona wzorem 3.1:

$$F1Score = \frac{2 * (Recall * Precision)}{Recall + Precision}. \tag{3.1}$$

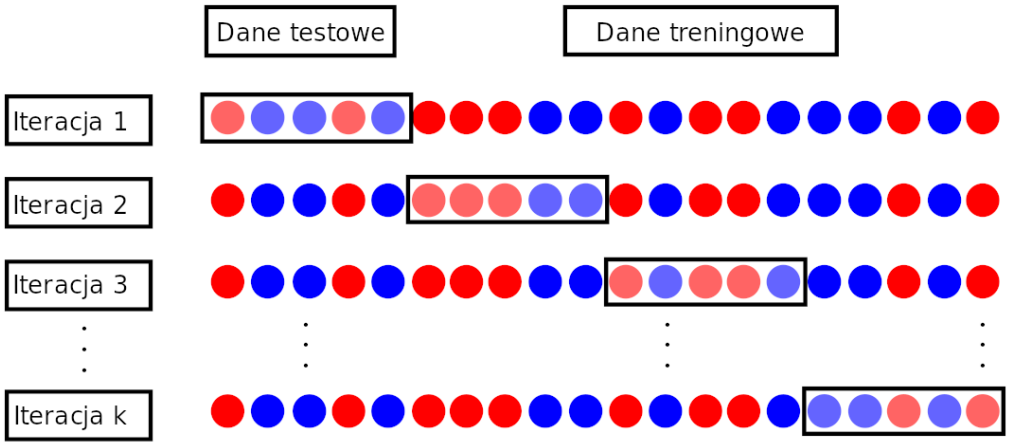
Pod uwagę brane są funkcje straty (loss), która informuje o dopasowaniu modelu do danych, oraz dokładności (accuracy), która określa skuteczność klasyfikacji. W celu określenia, która z testowanych metod daje najlepsze wyniki klasyfikacji wykorzystany zostanie test statystyczny - test Wilcoxona [40]. Do jego wykonania użyto wartości dokładności uzyskanych dla każdej z badanych metod.

3.3 Generowanie zbioru treningowego i testowego

W początkowym etapie uczenia maszynowego, programista dysponuje jedynie spójnym zbiorem danych. Jednym z kluczowych kroków w procesie jest podział zbioru na podzbiory: treningowy i testowy. Jest to konieczne w celu wydzielenia fragmentów, na których klasyfikator będzie się uczył i tych, na których nastąpi testowanie wyuczonego już klasyfikatora. Pominięcie tego kroku może skutkować błędnymi, wysokimi wynikami dokładności (accuracy). Istnieje kilka metod podziału zbioru. Najprostszym możliwym podejściem jest losowy podział zbiory na dwie części z zachowaniem proporcji ilościowej. Pozwala to na uzyskanie dwóch podzbiorów, lecz istnieje przy tym ryzyko, że losowość dokonała się w sposób, który wykaże błędną, zawyżoną dokładność.

Rozwiązaniem tego problemu jest wielokrotny podział. Metodą, która zapewnia wielokrotny, sprawiedliwy podział, eliminujący prawdopodobieństwo występowania tych samych próbek w różnych zbiorach uczących jest *K-Fold Cross-Validation*, czyli tak zwana *walidacja krzyżowa* [41].

W algorytmie walidacji krzyżowej, zbiór dzielony jest losowo na k równych podzbiorów. Następnie kolejno każdy z nich używa się jako zbiór testowy, a połączoną resztę - jako zbiór uczący. Finalnie, rezultaty uśrednia się w celu uzyskania jednorodnego wyniku.



Rysunek 3.2: Graficzna reprezentacja działania algorytmu walidacji krzyżowej.

W ramach przeprowadzonych eksperymentów posłużono się funkcją *KFold* z biblioteki *scikit-learn* [42]. Podobnie jak inne funkcje z tej biblioteki, umożliwia ona zadeklarowanie ziarna losowości, co zapewnia możliwość powtórzenia użyskanych w ten sposób wyników.

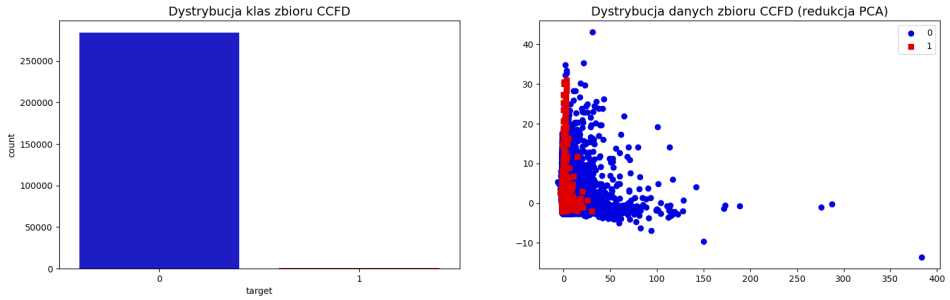
3.4 Zbiory danych

W ramach przeprowadzonych doświadczeń posłużono się 115 zbiorami danych. Wszystkie charakteryzowały się wysokim niezbalansowaniem i wykazywały nadreprezentację jednej bądź wielu cech. Rozdziały 3.4.1 - 3.4.4 zawierają szczegółowy opis dużych, starannie przygotowanych w ramach pracy zbiorów z różnych dziedzin nauki. Rozdział 3.4.5 skrótowo opisuje pozostałe zbiory danych, uzyskane za pośrednictwem platformy *KEEL*.

3.4.1 Credit Card Fraud Detection

Zbiór danych zawiera informacje o transakcjach dokonanych kartami kredytowymi we wrześniu 2013 roku, przez europejskich posiadaczy kart. Dataset składa się z zapisów transakcji, które miały miejsce w ciągu dwóch dni, w których miały miejsce 492 oszustwa z 284 807 wszystkich operacji. Zbiór jest wysoce niezbalansowany, a klasa pozytywna (oszustwa) stanowi 0,172% wszystkich transakcji [43].

Elementy zbioru składają się tylko z danych liczbowych, które są wynikiem transformacji PCA. Jest to podyktowane względami poufności - twórcy nie są w stanie dostarczyć oryginalnych wartości ani dodatkowych informacji o danych. Cechy $V1, V2, \ldots V28$ są głównymi składnikami uzyskanymi za pomocą analizy składowych głównych, jedynymi cechami, które nie zostały przekształcone są "*Time*" i "*Amount*." Warość "*Time*" zawiera sekundy, które upłynęły między każdą transakcją a pierwszą transakcją w zbiorze danych. Funkcja "*Amount*" to kwota transakcji. Cecha "*Class*" jest zmienną odpowiedzi i przyjmuje wartość 1 w przypadku oszustwa i 0 w innym przypadku. Dystrybucja klas oraz danych została ukazana na rysunku 3.3. W celu ukazania dystrybucji danych na dwuwymiarowym wykresie, zastosowano ekstrakcję cech metodą *2-PCA*.



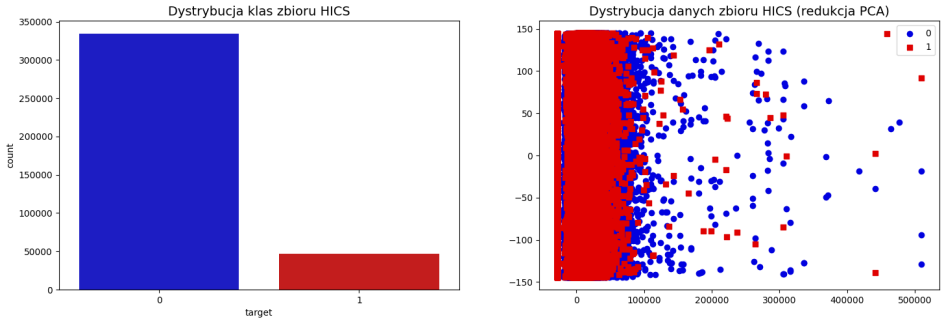
Rysunek 3.3: Dystrybucja klas i danych dla zbioru Credit Card Fraud Detection.

Zbiór został pozyskany za pośrednictwem platformy `Kaggle` (`https://www.kaggle.com/`).

3.4.2 Health Insurance Cross Sell

Zbiór został pierwotnie stworzony w ramach konkursu. Celem wyzwania było przewidywanie prawdopodobieństwa, że klienci firmy ubezpieczeniowej będą zainteresowani ubezpieczeniem komunikacyjnym oferowanym przez firmę [44]. Każdy wiersz odpowiada określonemu posiadaczowi polisy, a kolumny opisują ich cechy. Zmienna docelowa jest tu dogodnie nazywana wynikiem (*result*) i wskazuje, czy ubezpieczający będzie zainteresowany zakupem polisy.

Zbiór danych został dobrze udokumentowany przez twórców. Każdy klient został opisany dwunastoma cechami i są to: płeć (gender), wiek (age), posiadanie prawa jazdy (driving_license), region zamieszkania klienta (region_code), informacja o poprzednim ubezpieczeniu (previously_insured), wiek pojazdu (vehicle_age), informacja o uszkodzeniach pojazdu (vehicle_damage), koszty ubezpieczenia (annual_premium), kanał kontaktu z klientem (policy_sales_channel) i czas jaki klient jest związany z firmą (vintage). Dane o płci, wieku samochodu i uszkodzeniach pojazdu są danymi tekstowymi. Dystrybucja klas oraz danych została ukazana na rysunku 3.4. W celu ukazania dystrybucji danych na dwuwymiarowym wykresie, zastosowano ekstrakcję cech metodą *2-PCA*.



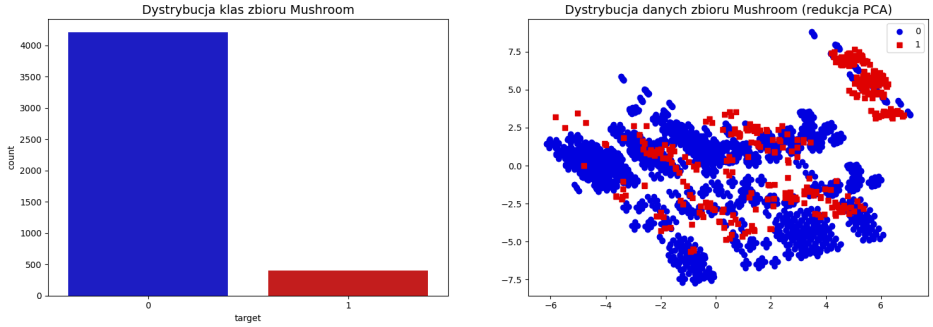
Rysunek 3.4: Dystrybucja klas i danych dla zbioru Health Insurance Cross Sell.

Zbiór został pozyskany za pośrednictwem platformy `Kaggle` (`https://www.kaggle.com/`).

3.4.3 Mushroom Classification

Zestaw danych zawiera opisy próbek różnych gatunków grzybów, zaczerpnięte z The Audubon Society Field Guide to North American Mushrooms (1981) [45]. Każdy gatunek jest określony jako zdecydowanie jadalny, zdecydowanie trujący lub o nieznanej jadalności (niezalecany). Ta ostatnia klasa została połączona z klasą trującą. Przewodnik jasno stwierdza, że nie ma prostej zasady określania jadalności grzyba co powinno uniemożliwić skuteczną klasyfikację. Każdy gatunek grzyba opisany jest za pomocą 23 cech, wśród których znajdują się cechy takie jak rozmiar i kształt kapelusza, rozmiar blaszek i przerw pomiędzy nimi, kolor, wysokość, typ pierścienia wokół pnia itd. Wszystkie cechy opisane są zmiennymi tekstowymi. Sam zbiór nie wykazuje cech niezbalansowania. Został odpowiednio dostosowany do celów eksperymentu, zostało to opisane w rozdziale 3.5.3.

Dystrybucja klas oraz danych została ukazana na rysunku 3.5. W celu ukazania dystrybucji danych na dwuwymiarowym wykresie, zastosowano ekstrakcję cech metodą *2-PCA*.

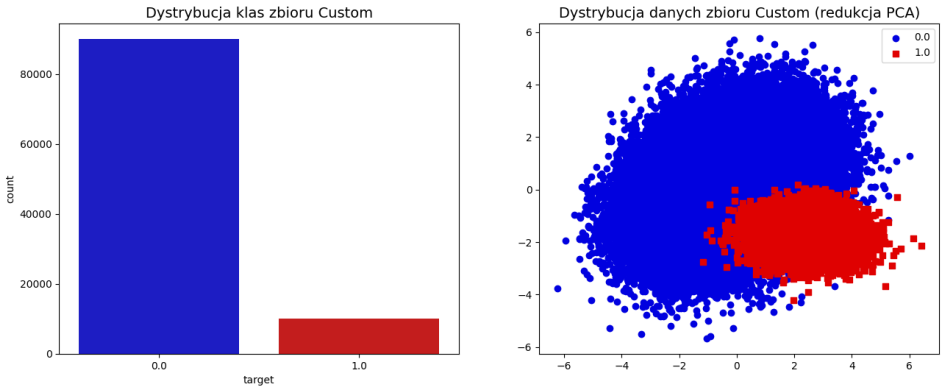


Rysunek 3.5: Dystrybucja klas i danych dla zbioru Mushroom.

Zbiór został pozyskany za pośrednictwem platformy `Kaggle` (`https://www.kaggle.com/`).

3.4.4 Zbiór 'Custom'

Zbiór custom został wygenerowany sztucznie z pomocą funkcji *make_classification* biblioteki *scikit-learn* [46]. Pozwoliło to na dobranie parametrów w taki sposób, aby zbiór odpowiadał jak najlepiej założeniom projektowym. Dataset składa się 100000 elementów. Kolumna *target* wskazuje na przypasowanie elementu do jednej z dwóch klas. Zbiór jest niezbalansowany, jedynie 10% elementów należy do klasy pozytywnej. Każdy element opisany jest za pomocą 20 cech, z czego trzy z nich są najbardziej informatywne, a jedna jest zupełnie zbędna. Zbiór nie zawiera pustych danych. Dystrybucja klas oraz danych została ukazana na rysunku 3.6. W celu ukazania dystrybucji danych na dwuwymiarowym wykresie, zastosowano ekstrakcję cech metodą *2-PCA*.



Rysunek 3.6: Dystrybucja klas i danych dla zbioru Custom.

3.4.5 Inne zbiory danych

W celu poprawnego i obiektywnego porównania wszystkich metod selekcji wykonano eksperymenty na dodatkowych 111 zbiorach udostępnionych przez platformę KEEL [47]. W tym celu stworzono program automatycznie importujący i przygotowujący zbiór do wykorzystania przez testowane algorytmy. Każdy ze zbiorów posiada pewne cechy wspólne - wszystkie one zostały stworzone w podobnym formacie, zawierają w większości cechy opisane numerycznie i jedynie dwie klasy - opisane za pomocą etykiet "*positive*" oraz "*negative*." Ponadto, klasą mniejszościową jest zawsze klasa "*positive*."

3.5 Przygotowanie danych

Dane wysokiej jakości są wymagane wstępnie dla modeli predykcyjnych. Wstępne przetwarzanie i czyszczenie danych to ważne zadania, które należy wykonać, zanim zestaw danych będzie mógł zostać użyty do uczenia modelu. Nieprzetworzone dane są zwykle zaszumione i mogą zawierać błędne wartości lub luki. Modelowanie przy użyciu tego typu danych może dawać mylące wyniki.

Szczególnie na takie zagrożenia narażone są dane rzeczywiste i zbiory zbierane z różnych źródeł. Typowe problemy z jakością danych to:

- Niekompletność: dane nie mają atrybutów lub zawierają brakujące wartości.
- Zakłócenia: dane zawierają błędne rekordy lub elementy odstające.
- Niespójność: dane zawierają rekordy powodujące niezgodności.

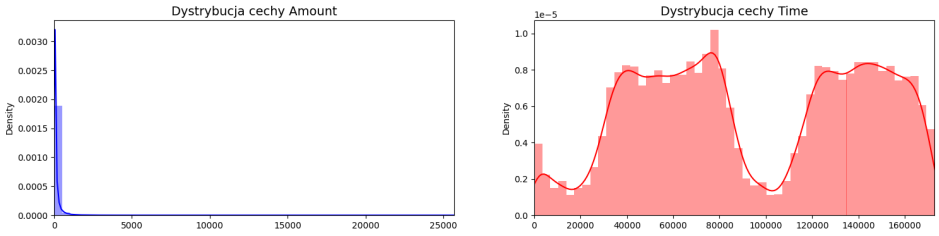
W celu odpowiedniego, wstępnego przetworzenia danych, do każdego zbioru należy podejść indywidualnie. Są jednak metody, które pozwalają poradzić sobie z większością napotkanych problemów. Do popularnych metod radzenia sobie z problemami wynikającymi ze źle przygotowanymi bazami danych należą:

- Uzupełnienie brakujących wartości.
- Likwidowanie wartości odstających (ang. outliers).
- Standaryzacja poprzez normalizację lub dyskretyzację.
- Redukcja wymiarów poprzez selekcję i ekstrakcję cech.
- Równoważenie danych poprzez usuwanie przypadków klas większościowych lub nadpróbkowanie klas mniejszościowych.
- Transformacja zmiennych (np. liniowa).
- Dodawanie nowych zmiennych w celu zwiększenia liczby cech (np. iloczynów istniejących zmiennych).
- Podział danych na dane treningowe i dane testowe.

Zbiory użyte w ramach eksperymentów należało w większości poddać wstępnej obróbce. Rozdziały 3.5.1 - 3.5.5 opisują działania podjęte w celu poprawienia jakości danych.

3.5.1 Przygotowanie zbioru CCFD

W zbiorze *Credit Card Fraud Detection* nie występują puste wartości. Większość cech została wcześniej poddana transformacji metodą PCA czego efektem ubocznym jest ich wyskalowanie, które konieczne jest w przypadku użycia tej metody. Cechami wyróżniającymi się są *"Time"* oraz *"Amount"*. Dystrybucja wartości dla tych cech ukazana jest na wykresach 3.7.



Rysunek 3.7: Dystrybucja wartości dla cech *Time* oraz *Amount*.

Wartości te należało przeskalować, aby nie odstawały od innych danych. Użyto w tym celu metody *RobustScaler* z biblioteki *scikit-learn* [48].

3.5.2 Przygotowanie zbioru HICS

W zbiorze *Health Insurance Cross Sell* nie występują puste wartości. Większość cech to cechy binarne, które nie potrzebują obróbki. Jak wspomniano w rozdziale 3.4.2, dane o płci, uszkodzeniach pojazdu i wieku samochodu są danymi tekstowymi. Kolumna *gender* przyjmuje wartości *Male* oraz *Female*, natomiast kolumna *vehicle_damage - yes* oraz *no*. Wartości te zostały zmienione na dane binarne: "*0*" i "*1*." Kolumna opisująca wiek pojazdu posiada 3 możliwe wartości tekstowe, które zostały przepisane na wartości liczbowe: "*0*," "*1*" i "*2*."

3.5.3 Przygotowanie zbioru Mushroom

Zbiór *Mushroom* był początkowo zbiorem zbalanowanym. W celu utworzenia zbioru niezbalasowanego, usunięto z niego 90% elementów zakwalifikowanych jako grzyby trujące (zostawiając co dziesiąty element oznaczony klasą '*p*'). Ponadto, wszystkie cechy zbioru opisane zostały etykietami słownymi. Stworzyło to konieczność przepisania wszystkich elementów zbioru na etykiety numeryczne. Posłużono się w tym celu funkcją *LabelEncoder* ze zbioru *scikit-learn* [49]. Metoda ta enkoduje każdy element listy wartością pomiędzy 0 a n-1 gdzie n to liczba wariantów cechy.

3.5.4 Przygotowanie zbioru Custom

Ponieważ zbiór *Custom* został wygenerowany sztucznie, wszystkie parametry zostały dobrane tak, by nie zachodziła potrzeba jego ponownej obróbki.

3.5.5 Przygotowanie pozostałych zbiorów

Zbiory z bazy danych *KEEL* zostały przez twórców przygotowane do testów, bez konieczności uprzedniego przygotowywania danych. Z pośród 111 zbiorów, istniało kilka, zawierających cechy w formacie tekstowym. Funkcja przygotowująca sprawdza każdy zbiór pod kątem takich przypadków i w razie potrzeby podmienia takie atrybuty na dane numeryczne. Użyta została w tym celu funkcja *LabelEncoder* z biblioteki *scikit-learn* [49]. Preprocessing pozostałych zbiorów ograniczył się do podmiany wartości "*positive*" oraz "*negative*" na wartości "*1*" i "*0*" na etapie parsowania zbioru.

3.6 Eksperymenty

W celu wykonania analizy, przy pomocy bibliotek *scikit-learn* oraz *pandas*, zaimplementowano metody opisane w rozdziałach 2.7.1 - 2.7.5. Algorytmy *ANOVA*, *chi*^{2} oraz *Information Gain* mają gotową implementację w bibliotece *scikit-learn* - kolejno funkcje *f_classif* [50], *chi2* [51] i *mutual_info_classif* [52]. Zostały one wykorzystane w ramach funkcji *SelectKBest*, umożliwiającej

zdefiniowanie metody selekcji cech oraz docelowej ilości atrybutów. Algorytm *Relieff* wdrożony został przy użyciu biblioteki *sklearn_relief* [53]. Metoda *Correlation Coefficient* zrealizowana została przy użyciu funkcji *corr* z pakietu *pandas*, zwracającej korelację Pearsona pomiędzy parami wszystkich kolumn danego zbioru [54]. Obie funkcje dostosowano tak, by odpowiadały swoim działaniem pozostałym trzem metodom. Eksperymenty zostały powtórzone kilka razy z różnymi ustawieniami w celu ustalenia optymalnej liczby cech dla każdego zbioru.

W ramach wszystkich zbiorów danych przeprowadzony został eksperyment badający skuteczność klasyfikacji bez wykonywania wcześniejszej selekcji cech oraz biorący pod uwagę różną ilość cech w procesie klasyfikacji.

W celu ułatwienia sprawnego i obiektywnego porównania wszystkich metod, w ramach pracy stworzono bibliotekę zawierającą fukcje przygotowujące dane, dzielące zbiór na podzbiory, przeprowadzające klasyfikacje oraz wyliczające wyniki eksperymentów. Rezultatem każdego doświadczenia jest plik w formacie *csv*. Plik ten zawierają nazwę zbioru użytego do klasyfikacji, nazwę metody selekcji sumaryczną ilość cech oraz ilość cech po przeprowadzonej selekcji, oraz dokładność, precyzję, macierz konfuzji, wartość *FPT*, wartość *TPR* i wynik testu *F1 Score* w 2 wariantach - przed selekcją i po selekcji cech.

Łącznie przeprowadzono 690 eksperymentów, po 115 dla każdej opisanej metody selekcji cech. Eksperymenty zawierają każdą permutację z zadanych zagadnień:

- Porównanie wyników dla każdego z 115 zbiorów danych.
- Porównanie wyników dla danych poddanych selekcji cech i klasyfikacji dla całego zbioru.
- Porównanie wyników dla każdej z 5 metod selekcji.

Wyniki eksperymentów opisane zostały w rozdziale 4 - *Wyniki*.

4 Wyniki

5 Wnioski

6 Bibliografia

[1] G. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *J. Artif. Intell. Res. (JAIR)*, vol. 19, pp. 315–354, Jul. 2003, doi: 10.1613/jair.1199.

[2] M. Hall, "Correlation-based feature selection for machine learning," *Department of Computer Science*, vol. 19, Jun. 2000.

[3] P. Ziemba, "Redukcja wymiarowości i selekcja cech w zadaniach klasyfikacji i regresji z wykorzystaniem uczenia maszynowego," *Zeszyty Naukowe Uniwersytetu Szczecińskiego. Studia Informatica*, pp. 221–236, 2012.

[4] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[5] I. Guyon and A. Elisseeff, "An introduction of variable and feature selection," *J. Machine Learning Research Special Issue on Variable and Feature Selection*, vol. 3, pp. 1157–1182, Jan. 2003, doi: 10.1162/153244303322753616.

[6] D. Tiwari, "Handling class imbalance problem using feature selection," Apr. 2014.

[7] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016, doi: 10.1109/TEVC.2015.2504420.

[8] J. M. Pereira, M. Basto, and A. F. da Silva, "The logistic lasso and ridge regression in predicting corporate failure," *Procedia Economics and Finance*, vol. 39, pp. 634–641, 2016, doi: https://doi.org/10.1016/S2212-5671(16)30310-0.

[9] L. E. Melkumova and S. Ya. Shatskikh, "Comparing ridge and LASSO estimators for data analysis," *Procedia Engineering*, vol. 201, pp. 746–755, 2017, doi: https://doi.org/10.1016/j.proeng.2017.09.615.

[10] H. Motoda and H. Liu, "Feature selection, extraction and construction," *Communication of IICM (Institute of Information and Computing Machinery, Taiwan)*, vol. 5, pp. 67–72, Jan. 2002.

[11] A. Sophian, G. Y. Tian, D. Taylor, and J. Rudlin, "A feature extraction technique based on principal component analysis for pulsed eddy current NDT," *NDT & E International*, vol. 36, no. 1, pp. 37–41, 2003, doi: https://doi.org/10.1016/S0963-8695(02)00069-5.

[12] G. Abdulrauf Sharifai and Z. Zainol, "Feature selection for high-dimensional and imbalanced biomedical data based on robust correlation based redundancy and binary grasshopper optimization algorithm," *Genes*, vol. 11, no. 7, 2020, doi: 10.3390/genes11070717.

[13] L. Mena and J. Gonzalez, "Machine learning for imbalanced datasets: Application in medical diagnostic." in *FLAIRS 2006 - Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, Jan. 2006, vol. 2006, pp. 574–579.

[14] D. A. N. S. Silva, L. C. Souza, and G. H. M. B. Motta, "An instance selection method for large datasets based on markov geometric diffusion," *Data & Knowledge Engineering*, vol. 101, pp. 24–41, 2016, doi: https://doi.org/10.1016/j.datak.2015.11.002.

[15] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Advances in intelligent computing*, 2005, pp. 878–887.

[16] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[17] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, pp. 5718–5727, Jan. 2006, doi: 10.1016/j.eswa.2008.06.108.

[18] H. He, Y. Bai, E. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of the International Joint Conference on Neural Networks*, Jul. 2008, pp. 1322–1328, doi: 10.1109/IJCNN.2008.4633969.

[19] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data*, vol. 5, no. 1, p. 42, Nov. 2018, doi: 10.1186/s40537-018-0151-6.

[20] J. R. Quinlan, "Bagging, boosting, and C4.5," in *In proceedings of the thirteenth national conference on artificial intelligence*, 1996, pp. 725–730.

[21] T. Chengsheng, L. Huacheng, and X. Bing, "AdaBoost typical algorithm and its application research," *MATEC Web of Conferences*, vol. 139, p. 00222, Jan. 2017, doi: 10.1051/matecconf/201713900222.

[22] K. Hempstalk, E. Frank, and I. Witten, "One-class classification by combining density and class probability estimation," Sep. 2008, doi: 10.1007/978-3-540-87479-9_51.

[23] H. J. Shin, D.-H. Eom, and S.-S. Kim, "One-class support vector machines—an application in machine fault detection and classification," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 395–408, 2005, doi: https://doi.org/10.1016/j.cie.2005.01.009.

[24] S. Ertekin, J. Huang, L. Bottou, and C. Giles, "Learning on the border: Active learning in imbalanced data classification," in *International Conference on Information and Knowledge Management, Proceedings*, Jan. 2007, pp. 127–136, doi: 10.1145/1321440.1321461.

[25] M. Langarizadeh and F. Moghbeli, "Applying naive bayesian networks to disease prediction: A systematic review," *Acta Informatica Medica*, vol. 24, p. 364, Oct. 2016, doi: 10.5455/aim.2016.24.364-369.

[26] M. Bader-El-Den, E. Teitei, and T. Perry, "Biased random forest for dealing with the class imbalance problem," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–10, Nov. 2018, doi: 10.1109/TNNLS.2018.2878400.

[27] P. van der Putten and M. van Someren, "A bias-variance analysis of a real world learning problem: The CoIL challenge 2000," *Machine Learning*, vol. 57, no. 1, pp. 177–195, Oct. 2004, doi: 10.1023/B:MACH.0000035476.95130.99.

[28] D. Mladenić and M. Grobelnik, "Feature selection for unbalanced class distribution and naive bayes." Jan. 1999, pp. 258–267.

[29] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, no. null, pp. 1289–1305, Mar. 2003.

[30] Z. Zheng, X. Wu, and R. Srihari, "Feature selection for text categorization on imbalanced data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 80–89, Jun. 2004, doi: 10.1145/1007730.1007741.

[31] C. Elkan, "Magical thinking in data mining: Lessons from CoIL challenge 2000," *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul. 2001, doi: 10.1145/502512.502576.

[32] I. Jamali, M. Bazmara, and S. Jafari, "Feature selection in imbalance data sets," *International Journal of Computer Science Issues*, vol. 9, p. 42, May 2013.

[33] J. Biesiada and W. Duch, "Feature selection for high-dimensional data — a pearson redundancy based filter," in *Computer recognition systems 2*, 2007, pp. 242–249.

[34] I. Kononenko, E. Šimec, and M. Robnik-Sikonja, "Overcoming the myopia of inductive learning algorithms with RELIEFF," *Applied Intelligence*, vol. 7, pp. 39–55, Jan. 1997, doi: 10.1023/A:1008280620621.

[35] R. J. Urbanowicz, M. Meeker, W. La Cava, R. S. Olson, and J. H. Moore, "Relief-based feature selection: Introduction and review," *Journal of Biomedical Informatics*, vol. 85, pp. 189–203, 2018, doi: https://doi.org/10.1016/j.jbi.2018.07.014.

[36] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Machine learning proceedings 1992*, D. Sleeman and P. Edwards, Eds. San Francisco (CA): Morgan Kaufmann, 1992, pp. 249–256.

[37] M. Kumar, N. Rath, A. Swain, and S. Rath, "Feature selection and classification of microarray data using MapReduce based ANOVA and k-nearest neighbor," *Procedia Computer Science*, vol. 54, pp. 301–310, Dec. 2015, doi: 10.1016/j.procs.2015.06.035.

[38] M. G. Larson, “Analysis of variance,” pp. 115–121, Jun. 2008, doi: https://doi.org/10.1161/CIRCULATIONAHA.107.654335.

[39] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “KNN model-based approach in classification,” in *On the move to meaningful internet systems 2003: CoopIS, DOA, and ODBASE*, 2003, pp. 986–996.

[40] F. Nahm, “Nonparametric statistical tests for the continuous data: The basic concept and the practical use,” *Korean Journal of Anesthesiology*, vol. 69, p. 8, Feb. 2016, doi: 10.4097/kjae.2016.69.1.8.

[41] Y. Jung, “Multiple predicting k-fold cross-validation for model selection,” *Journal of Nonparametric Statistics*, vol. 30, no. 1, pp. 197–215, 2018, doi: 10.1080/10485252.2017.1404598.

[42] “Sklearn model_selection KFold,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html.

[43] M. L. G.-. ULB, “Credit card fraud detection,” *Kaggle*. Mar. 2018, [Online]. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud.

[44] A. Kumar, “Health insurance cross sell prediction,” *Kaggle*. Sep. 2020, [Online]. Available: https://www.kaggle.com/anmolkumar/health-insurance-cross-sell-prediction.

[45] U. M. Learning, “Mushroom classification,” *Kaggle*. Dec. 2016, [Online]. Available: https://www.kaggle.com/uciml/mushroom-classification.

[46] “Sklearn datasets make_classification,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html.

[47] “KEEL: Software tool. Evolutionary algorithms for data mining,” *SCI2S*. [Online]. Available: https://sci2s.ugr.es/keel/imbalanced.php.

[48] “Sklearn preprocessing RobustScaler,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[49] “Sklearn preprocessing LabelEncoder,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.

[50] “Sklearn feature_selection f_classif,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html.

[51] “Sklearn feature_selection chi2,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html.

[52] “Sklearn feature_selection mutual_info_classif,” *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html.

[53] A. Mungo, “Sklearn-relief,” *GitLab - sklearn-relief*. [Online]. Available: https://gitlab.com/moongoal/sklearn-relief.

[54] “Python: Pandas dataframe.corr(),” *GeeksforGeeks*. Apr. 2020, [Online]. Available: https://www.geeksforgeeks.org/python-pandas-dataframe-corr/.

Zawartość płyty CD

Do pracy dołączono płytę CD o następującej zawartości:

- kod źródłowy programu znajdujący się w folderze `/src`
- otwartoźródłowe dane użyte do trenowania sieci neuronowych w katalogu `/data`
- katalog `/docs` zawierający kod źródłowy tej pracy
- katalog `/results` zawierający pliki wynikowe przeprowadzanych eksperymentów
- plik w formacie `pdf` zawierający tę pracę