



Wydział Fizyki i Informatyki Stosowanej

---

## Praca magisterska

**Kamil Madej**

kierunek studiów: **Informatyka Stosowana**

specjalność: **Grafika Komputerowa i Przetwarzanie Obrazów**

# **Sposoby zwiększania bezpieczeństwa systemu informatycznego przy pomocy aplikacji do zarządzania zasobami systemów informatycznych**

Opiekun: **dr inż. Antoni Dydejczyk**

**Kraków, czerwiec 2018**

---

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

.....

(czytelny podpis)

---

TODO

---

Kraków, czerwiec 2018

**Tematyka pracy magisterskiej i praktyki dyplomowej Kamila Madeja,  
studenta V roku studiów kierunku informatyka stosowana, specjalności  
grafika komputerowa i przetwarzanie obrazów**

**Temat pracy magisterskiej: Sposoby zwiększania bezpieczeństwa systemu  
informatycznego przy pomocy aplikacji do zarządzania zasobami  
systemów informatycznych**

Opiekun pracy: dr inż. Antoni Dydejczyk

Recenzenci pracy: dr inż. TODO

Miejsce praktyki dyplomowej: WFiIS AGH, Kraków

**Program pracy magisterskiej i praktyki dyplomowej**

1. Omówienie realizacji pracy magisterskiej z opiekunem oraz przedstawicielem firmy IBM.
2. Zebranie i opracowanie literatury dotyczącej tematu pracy.
3. Praktyka dyplomowa:
  - Zapoznanie się z ideą Software Asset Management
  - Zapoznanie się z tematyką podatności systemów informatycznych
  - Analiza danych wejściowych
4. Opracowanie algorytmu umożliwiającego wykrywanie podatności w systemie
5. Przeprowadzenie testów, analiza wyników
6. Porównanie otrzymanych wyników z próbami przeprowadzonymi przez IBM
7. Analiza wyników, ich omówienie i zatwierdzenie przez opiekuna
8. Opracowanie redakcyjne pracy

Termin oddania w dziekanacie: TODO czerwca 2018

.....  
(podpis kierownika katedry)

.....  
(podpis opiekuna)



---

Merytoryczna ocena pracy przez opiekuna: TODO



---

Merytoryczna ocena pracy przez recenzenta: TODO





# Spis treści

<b>Wstęp</b>	<b>12</b>
<b>Cel pracy</b>	<b>13</b>
<b>1 Klasyfikacja</b>	<b>15</b>
1.1 Wstęp	15
1.2 Dostępne rozwiązania	16
1.2.1 Klasyfikator K-najbliższych sąsiadów	16
1.2.2 Drzewa decyzyjne	17
1.2.3 Maszyna wektorów nośnych	18
1.2.4 Sieci neuronowe	19
<b>2 Opis teoretyczny badanego rozwiązania</b>	<b>22</b>
2.1 Jądrowy estymator gęstości	22
2.2 Probabilistyczna sieć neuronowa	23
2.3 Dobór jądra	25
2.4 Parametr wygładzania	28
2.5 Dobór parametru wygładzania metodą plug-in	29
2.6 Modyfikacja parametru wygładzania	30
2.7 Flower Pollination Algorithm	31
<b>3 Implementacja</b>	<b>34</b>
3.1 Język programowania R	34
3.2 Implementacja PNN	35
3.3 Implementacja FPA	38
3.4 Integracja algorytmów	39
<b>4 Analiza użyteczności</b>	<b>42</b>
4.1 Badanie wpływu doboru parametrów	42
4.1.1 Wybór jądra estymacji	42
4.1.2 Stopnie swobody parametru wygładzania	43
4.1.3 Siła modyfikacji parametru wygładzania	49
4.1.4 Strojenie parametrów FPA	50
4.1.5 Wpływ normalizacji na klasyfikację	52
4.1.6 Wybór parametru wygładzania	53
4.1.7 Finalny algorytm	55

4.2 Porównanie z istniejącymi algorytmami . . . . .	55
<b>5 Podsumowanie . . . . .</b>	<b>58</b>
<b>6 Kierunki dalszego rozwoju . . . . .</b>	<b>60</b>
<b>Bibliografia . . . . .</b>	<b>62</b>
<b>Spis rysunków . . . . .</b>	<b>64</b>
<b>Spis tabel . . . . .</b>	<b>65</b>
<b>Spis listingów . . . . .</b>	<b>66</b>
<b>Dodatki . . . . .</b>	<b>66</b>

## Wstęp

Software Asset Management jest pojęciem ściśle związanym z dynamicznie rozwijającym się rynkiem oprogramowania. Pojawiające się różne rodzaje licencji określające sposób, w jaki można korzystać z poszczególnych rozwiązań, stworzyły przestrzeń dla firm takich, jak IBM, na stworzenie aplikacji ułatwiającej zarządzanie licencjami w systemach informatycznych przedsiębiorstw. W dzisiejszych realiach poważne przedsiębiorstwa muszą działać transparentnie, nie mogą sobie bowiem pozwolić na oskarżenia o nadużywanie lub nieprzestrzeganie wymogów licencyjnych. Wiąże się to nie tylko z kosztami ponoszonymi wskutek naliczanych kar, ale też ze stratami wizerunkowymi.

Każdego dnia pojawiają się nowe podatności i zagrożenia, na jakie wystawione są systemy informatyczne. Twórcom oprogramowania zależy, aby ich produkt był jak najbardziej na nie odporny. Wymusza to publikowanie nowych wersji oprogramowania, w którym poprawiono błędy i załatano luki umożliwiające niepożądane działanie systemu.

Niniejsza praca jest próbą połączenia powyższych pojęć w jednym narzędziu. Wykorzystując informacje gromadzone w aplikacji IBM Big Fix Inventory, można informować użytkownika o tym, że w jego systemie informatycznym zainstalowane jest oprogramowanie, w którym wykryta została podatność. Właściciel systemu na podstawie informacji o rodzaju podatności i o stopniu zagrożenia z nią związanego, może zdecydować jakie akcje należy podjąć, by zapewnić stabilność i bezpieczeństwo systemu.

Algorytm opracowany w ramach pracy umożliwia wykrycie w systemie informatycznym wersji oprogramowania znajdujących się na liście obciążonych podatnościami, publikowanej przez amerykańską agencję NIST. Jego działanie polega na dopasowaniu informacji o oprogramowaniu zainstalowanym i używanym w systemie, zebranych przez IBM Big Fix Inventory, z informacjami zebranymi przez NIST.

## Cel pracy

Niniejsza praca stanowi badanie możliwości wykorzystania różnych algorytmów porównania tekstu oraz ich połączeń w celu wykrywania oprogramowania, które może powodować podatność systemu informatycznego na zagrożenia. W ramach pracy zostaną wykonane następujące czynności:

- Analiza poszczególnych algorytmów pod kątem danych wejściowych.
- Na podstawie powyższej analizy zostanie opracowany algorytm pozwalający na połączenie dostępnych informacji o podatnościach występujących w oprogramowaniu informatycznym oraz informacji o oprogramowaniu zainstalowanym w badanym systemie.
- Implementacja algorytmu w języku Python
- Przeprowadzenie testów skuteczności algorytmu
- Analiza otrzymanych wyników

Podczas opracowywania algorytmu szczególna uwaga zwrócona będzie w kierunku skuteczności wykrycia zagrożonej wersji oprogramowania zainstalowanej w systemie informatycznym monitorowanym przez narzędzie IBM Big Fix Inventory. Efektem pracy będą przedstawione wnioski dotyczące efektywności poszczególnych algorytmów porównania tekstu oraz ich połączeń opracowanych w ramach pracy. Wnioski zostaną wyciągnięte na podstawie wyników testów poszczególnych rozwiązań.



# 1 Klasyfikacja

## 1.1 Wstęp

Eksploracja danych zajmuje się analizą dużych zbiorów danych dotyczących różnych zjawisk. Mogą to być dane finansowe, wyniki obserwacji fizyczny czy dane pacjentów. Danych najczęściej jest zbyt dużo aby mógł je przeanalizować człowiek, dlatego też istnieje zapotrzebowanie na automatyczne algorytmy operujące na całym zbiorze danych. Zbiór zwykle przedstawiany jest w postaci macierzy której kolumnami są cechy opisujące zjawisko, a wiersze to kolejne obserwacje. Analizowane dane mogą mieć następujące typy:

- Liczbowe - Numeryczny wynik jakiegoś pomiaru, na przykład wzrost, waga
- Kategoryczne - Wartość z ograniczonego zbioru, na przykład rozróżnienie kobiet i mężczyzn, czy grupy krwi
- Tekstowe - Bloki tekstu opisujące zjawisko, na przykład treść wiadomości e-mail

Jedną z podstawowych procedur Eksploracji danych jest klasyfikacja, polegająca na dopasowywaniu nowych obserwacji do jednej z kategorii(klas). Nieprzypisana obserwacja powinna zostać przyporządkowana do klasy w której znajdują się elementy najbardziej do niej podobne. Miarą podobieństwa może być odległość, pochodzenie z tego samego rozkładu lub inne bardziej złożone miary. Procedura ta zakłada istnienie zbioru trenującego już podzielonego na klasy, który będzie służył za przykład na podstawie którego dopasowane zostaną nowe rekordy. Przykładem klasyfikacji może być ocena czy pacjent cierpi na jakąś chorobę, czy określenie gatunku ziarna na podstawie jego wymiarów i wagi. Klasyfikacja jest przykładem uczenia nadzorowanego, procedura uczy się na podstawie dostarczonych do niej przykładów. Przykładem uczenia nienadzorowanego jest klasteryzacja, w której celem jest podział zbioru danych na grupy bez żadnych informacji podanych z zewnątrz.

## 1.2 Dostępne rozwiązania

### 1.2.1 Klasyfikator K-najbliższych sąsiadów

Klasyfikator K-Najbliższych sąsiadów opiera się na założeniu że obserwacje ze zbioru uczącego znajdujące się najbliżej dostarczają istotnych informacji na temat klasy do której należy klasyfikowana obserwacja. Na potrzeby tego klasyfikatora należy zdefiniować pojęcie odległości w danym zbiorze danych. W przypadku danych liczbowych odległość może zostać zdefiniowana przy pomocy metryki Euklidesowej, i taka też jest najczęściej wykorzystywana w praktyce. Inne typy danych nie mogą być bezpośrednio wykorzystywane do obliczania odległości. Dane tekstowe należy przekształcić na dane numeryczne lub kategoryczne przy pomocy procedur eksploracji tekstu. Cechy kategoryczne o więcej niż dwóch możliwych wartościach można przykładowo umieścić na sympleksie o odpowiedniej wymiarowości, tak aby odległości wszystkich wyborów były równe. Przebieg algorytmu [1]:

- Znajdź K elementów o najmniejszej odległości od klasyfikowanego
- Przypisz element do klasy z której pochodzi najwięcej sąsiadów

Dobór wartości K jest bardzo istotny dla działania algorytmu. Małe K powoduje że algorytm jest bardzo narażony na wpływ elementów nietypowych, a granicach klas pojawi się bardzo wiele źle zaklasyfikowanych elementów. Duża wartość K może spowodować że mało liczne klasy i skupienia będą ignorowane, co również negatywnie wpłynie na jakość klasyfikacji [1]. Zaletą algorytmu jest jego prostota, jednakże jest on zwykle mniej skuteczny od bardziej zaawansowanych algorytmów.



### 1.2.2 Drzewa decyzyjne

Drzewa decyzyjne to zbiór algorytmów których celem jest zbudowanie modelu w postaci drzewa umożliwiającego podjęcie decyzji na temat przynależności elementu do danej klasy. Algorytmu budowy drzew decyzyjnych polegają na rekursywnym podziale zbioru danych [2]. Celem działania drzewa decyzyjnego jest podział przestrzeni możliwych danych wejściowych na takie części aby każda z nich zawierała wyłącznie elementy jednej klasy. W węzłach drzewa umieszczane są kryteria na podstawie których dzielona jest przestrzeń danych wejściowych.

Istnieje wiele algorytmów opartych o drzewa decyzyjne, klasycznym przykładem jest algorytm Iterative Dichotomiser 3 (ID3) w którym wybór decyzji jaka zostanie dodana do drzewa kierowany jest kryterium maksymalizacji zysku informacyjnego. Zysk informacyjny to różnica pomiędzy entropią zbioru danych przed i po dodaniu decyzji. W zbiorze uporządkowanym entropia powinna wynosić 0 ( w każdym zbiorze będą znajdować się tylko elementy jednej klasy).



Rysunek 1: Struktura drzewa decyzyjnego w algorytmie ID3

### 1.2.3 Maszyna wektorów nośnych

Maszyna wektorów nośnych (ang. support vector machines, SVM), to metoda klasyfikacji binarnej której celem jest wyznaczenie hiperpłaszczyzny oddzielającej od siebie elementy dwóch klas [3]. Niestety, nie wszystkie zbiory danych można znaleźć granicę pomiędzy klasami danych, najczęściej w oryginalnej reprezentacji dane w pewnych obszarach mieszają się ze sobą. Aby rozwiązać ten problem w metodzie SVM stosuje się nieliniową transformację zbioru danych w przestrzeń o znacznie wyższej wymiarowości w której zbiory danych staną się separowalne liniowo.



Rysunek 2: Liniowy podział zbioru danych.

Algorytm SVM polega na znalezieniu takiej linii rozdzielającej punkty aby odległość od najbliższego punktu każdej z klas była jak największa, tak by zapewnić jak największy margines klasyfikacji [3]. Taki wybór linii rozdzielającej znacznie poprawia właściwości uogólniające sieci. SVM działa tylko dla przypadku dwóch klas, aby umożliwić klasyfikację większej ilości klas należy utworzyć wiele instancji SVM z których każda będzie podejmować decyzję czy dany punkt należy do danej klasy czy do pozostałych klas.

### 1.2.4 Sieci neuronowe

Sztuczne sieci neuronowe mogą być wykorzystywane do aproksymacji dowolnej funkcji, mogą więc również zostać użyte do klasyfikacji. Sztuczna sieć składa się z wielu połączonych ze sobą modeli neuronów. Neurony stanowią zwykłe matematyczny model biologicznych neuronów. Pojedynczy neuron ma za zadanie zsumować sygnały ze wszystkich swoich wejść z zadanymi wagami, a następnie dokonać pogowania [4]. Sieć neuronowa składa się z kilku warstw połączonych neuronów, zazwyczaj istnieją co najmniej dwie warstwy, warstwa wejściowa gdzie istnieje dokładnie jeden neuron dla każdej cechy zbioru danych, i warstwa wyjściowa z liczbą neuronów odpowiadającą złożoności problemu. W przypadku klasyfikacji, w warstwie wyjściowej zwykle znajduje się dokładnie tyle neuronów ile jest klas, gdzie każdy z nich daje binarną odpowiedź o przynależności do jednej z klas, bądź stosuje się dodatkową agregującą warstwę które przyporządkowuje badany element do klasy na podstawie wyjść poprzedniej warstwy.



Rysunek 3: Uproszczony schemat sieci neuronowej.

Proces uczenia sieci neuronowej polega na dobraniu takich wag aby sieć realizowała pożądaną zadanie. Wprowadza się pojęcie funkcji celu której wartość będzie optymalizowana, zwykle poprzez kwadrat odległości wyjścia sieci neuronowej od wartości faktycznej. Do optymalizacji wykorzystuje się techniki propagacji wstecznej i algorytmy spadku gradientowego [4]. Innym rodzajem sieci

neuronowej jest sieć probabilistyczna, która zostanie opisana w kolejnym rozdziale.



## 2 Opis teoretyczny badanego rozwiązania

### 2.1 Jądrowy estymator gęstości

Rozkład gęstości prawdopodobieństwa to funkcja opisująca prawdopodobieństwo wystąpienia elementu w zależności od wartości wszystkich współrzędnych [5].

$$f : \mathbb{R}^n \rightarrow [0, \infty) \quad (1)$$

Znajomość tej funkcji dostarcza wielu informacji na temat zbioru danych, na jej podstawie opiera się wiele algorytmów statystycznych, między innymi ten wykorzystany w pracy. Funkcja ta jednak nigdy nie jest dana w sposób jawny dla zbioru danych, dlatego należy ją estymować na podstawie dostępnych wartości.

Do przybliżania wartości funkcji rozkładu gęstości prawdopodobieństwa wykorzystano metodę estymatorów jądrowych. Gęstość prawdopodobieństwa często jest funkcją o wielu lokalnych maksimach i minimach, dlatego też nie można jej efektywnie estymować żadnym standardowym rozkładem prawdopodobieństwa, dlatego też wybrano metodę nieparametryczną (nie zakładającą przynależności do danego rozkładu). Metoda estymatorów jądrowych polega na sumowaniu funkcji jądrowych (ang. kernel functions) przesuniętych do każdego z elementów zbioru trenującego. Jednowymiarowy estymator gęstości opisany jest więc wzorem [5]:

$$\hat{f}(x) = \frac{1}{mh} \sum_{i=1}^m K\left(\frac{x - x_i}{h}\right) \quad (2)$$

Gdzie  $m$  to liczność zbioru zdanych,  $K$  to funkcja estymatora jądrowego, a  $h$  to parametr wygładzania. Jądro musi być funkcją symetryczną z maksimum globalnym w zerze, a jego całka musi wynosić jeden [6]. Istnieje wiele funkcji spełniających te warunki, dobór jądra zostanie omówiony w podrozdziale 2.3. Na rysunku 4 widać funkcję rozkładu gęstości prawdopodobieństwa, która przyjmuje wysokie wartości w tych przedziałach w których znajduje się duża liczba elementów, i bardzo niskie w obszarach w których elementy nie występują. Wartość tej funkcji można interpretować jako prawdopodobieństwo wystąpienia konkretnego zestawienia wartości wszystkich cech zbioru danych.



Rysunek 4: Jądrowy estymator gęstości jednowymiarowej funkcji

Dla przypadku zmiennej losowej wielowymiarowej funkcja estymatora jądrowego opisywana jest przez wzór [6]:

$$\hat{f}(x, H) = \frac{1}{m} \sum_{i=1}^m |H|^{-1/2} K(H^{-1/2}x) \quad (3)$$

Gdzie  $H$  to macierz współczynników wygładzania a  $K$  to jądro wielowymiarowe. Istnieją dwie metody generacji jąder wielowymiarowych z jąder jednowymiarowych, jądro produktywne i jądro radialne. Jądro produktywne to jądro wielowymiarowe utworzone jako iloraz jąder jednowymiarowych [6]:

$$K^n(x) = \prod_{i=1}^n K^1(x_i) \quad (4)$$

Alternatywnym podejściem jest stosowanie jądra radialnego [6]:

$$K^n(x) = c_{K,n} K^1((x^T x)^{1/2}) \quad (5)$$

W niniejszej pracy wykorzystano podejście produktywne.

## 2.2 Probabilistyczna sieć neuronowa

Probabilistyczna sieć neuronowa to struktura wykorzystująca funkcję rozkładu gęstości prawdopodobieństwa do celu klasyfikacji. Struktura sieci przedstawiona jest na rysunku 5.



Rysunek 5: Struktura probabilistycznej sieci neuronowej

Pierwsza warstwa to warstwa wejściowa, każdy neuron warstwy wejściowej reprezentuje jeden z atrybutów zbioru danych. Wartości atrybutów klasyfikowanego elementu przekazywane są na wejścia neuronów wejściowych. W drugiej warstwie (warstwa wzorów) obliczane są wartości funkcji jądrowej przesuniętej do punktu ze zbioru uczącego w badanym punkcie dla każdego elementu trenującego. Warstwa ta zawiera wszystkie informacje ze zbioru trenującego. Powstałe wartości są sumowane i skalowane w warstwie sumacyjnej. Sumowanie przeprowadzane jest po wszystkich elementach klasy, więc neuronów w warstwie sumacyjnej jest dokładnie tyle ile klas w zbiorze danych. Wyjścia warstwy sumacyjnej są odpowiednikami obliczenia wartości jądrowego estymatora gęstości dla każdej z klas w badanym punkcie. Ostatnia warstwa podejmuje decyzję o klasyfikacji. Wybrana zostaje klasa której wartość wyjścia warstwy sumacyjnej jest największa.

W przeciwieństwie do klasycznych sieci neuronowych, sieć probabilistyczna nie wymaga procesu optymalizacji wag wszystkich połączeń. Proces uczenia PNN polega na budowie struktury i umieszczeniu w niej wszystkich elementów zbioru trenującego, oraz na odpowiednim dobraniu parametrów jądrowego estymatora



gęstości, które zostaną opisane w kolejnych podrozdziałach. Wadą probabilistycznych sieci neuronowych jest jednakże wysoki czas klasyfikacji, który uwarunkowany jest umieszczeniem całego zbioru danych w strukturze sieci.

## 2.3 Dobór jądra

Funkcja jądrowa (ang. kernel) powinna spełniać poniższe warunki [6]:

$$\int K(x)dx = 1 \tag{6}$$

$$\int xK(x)dx = 0 \tag{7}$$

$$\int x^2K(x)dx < \inf \tag{8}$$

$$K(x) > 0 \tag{9}$$

$$K(x) = K(-x) \tag{10}$$

Pierwsze wymaganie podyktowane jest chęcią modelowania funkcji gęstości prawdopodobieństwa, której całka również musi wynosić jeden. Pozostałe warunki wynikają ze specyfiki procesu estymacji, warto aby funkcja jądra była symetryczna aby nie kierować danych w jednym z kierunków. Istnieje wiele funkcji spełniających powyższe warunki, nie można również stwierdzić które z okien daje najlepsze rezultaty. W tabeli 1 wypisano jądra badane w niniejszej pracy.

Tabela 1: Jądra







Wykres	Wzór	Nazwa
	$K(x) = 1/2,  x  \leq 1$	Jednostajne
	$K(x) = (1 -  x ),  x  \leq 1$	Trójkątne
	$K(x) = 3/4(1 - x^2),  x  \leq 1$	Epanecznikowa

Tabela 2: Jądra cz.2

Wykres	Wzór	Nazwa
	$K(x) = 15/16(1 - x^2)^2,  x  \leq 1$	Dwówagowe
	$K(x) = 1/\sqrt{2\pi}e^{-1/2u^2}$	Normalne
	$K(x) = 2/\pi(x^2 + 1)^2$	Cauchiego

Jądro Epanecznikowa jest optymalne pod względem błędu średniokwadratowego estymacji [8], jednakże wykorzystywana jest znaczna ilość innych okien, przede wszystkim dlatego że teoretyczna różnica dokładności nie jest duża, a często wybiera się okna o dobrych właściwościach matematycznych, ułatwiając tym samym przeprowadzanie obliczeń. Z tego właśnie powodu często wykorzystywane są jądra Cauchiego i normalne. Można też zauważyć że istnieją dwa typy jąder:

- O skończonej szerokości: Jednostajne, Trójkątne, Epanecznikowa, Dwówagowe
- O nieskończonej szerokości: Normalne, Cauchiego

Różnice pomiędzy tymi typami okien oraz wyniki uzyskane z różnymi oknami zostaną omówione w części praktycznej pracy.

### 2.4 Parametr wygładzania

Bardzo istotny w procesie estymacji jądrowej jest parametr wygładzania. Parametr ten kontroluje szerokość jądra, duże wartości powodują mocne wygładzenie estymatora rozkładu, natomiast małe wartości prowadzą do bardzo dużej zmienności estymatora. We wzorze 3 symbol  $H$  oznacza macierz symetryczną parametrów wygładzania, można analizować kilka stopni dokładności doboru parametru wygładzania:

- Parametr wygładzania taki sam dla wszystkich wymiarów i klas -  $H$  to macierz powstała przez przemnożenie składową przez macierz jednostkową, ta sama dla każdej klasy.
- Parametr wygładzania taki sam dla wszystkich klas, zmienny dla wymiarów -  $H$  to macierz diagonalna, ta sama dla każdej klasy.
- Parametr wygładzania taki sam dla wszystkich wymiarów, zmienny dla klas -  $H$  to macierz powstała przez przemnożenie składową przez macierz jednostkową, różna dla każdej klasy.
- Parametr wygładzania zmienny dla wymiarów i klas -  $H$  to macierz diagonalna, różna dla każdej klasy.

W pracy zbadane zostaną wszystkie wymienione podejścia.

## 2.5 Dobór parametru wygładzania metodą plug-in

Istnieje wiele metod doboru parametru wygładzania, w przypadku jądra radialnego najczęściej wykorzystywana jest metoda kros-walidacji, natomiast w przypadku podejścia produktywnego najczęściej wykorzystuje się metodę podstawiania (ang. plug-in) [5]. W niniejszej pracy wykorzystana i omówiona zostanie metoda plug-in. Jakość estymacji oceniana jest miarą całkowitego błędu średnio kwadratowego MISE:

$$MISE(\hat{f}) = E \left( \int_{R^n} [\hat{f}(x) - f(x)]^2 dx \right) \quad (11)$$

Gdzie  $E$  oznacza wartość oczekiwaną. Celem metody plug-in jest znalezienie  $h$  minimalizującego powyższe wyrażenie.

Można pokazać że wartością minimalizującą powyższe wyrażenie dla przypadku jednowymiarowego z błędem zmierzającym do zera wraz ze wzrostem rozmiaru zbioru danych jest [6]:

$$h_{MISE} = \left[ \frac{R(K)}{mP(K)^2\Psi_4} \right]^{\frac{1}{5}} \quad (12)$$

Gdzie:

$$R(K) = \int_{R^n} K(x)^2 dx \quad (13)$$

$$P(K) = \int_{R^n} x^T x K(x) dx \quad (14)$$

$$\Psi_4 = \int_{-\infty}^{\infty} f^{(4)}(x) f(x) dx \quad (15)$$

Jedynym nieznanym elementem równania 12  $\Psi_4$ . W metodzie plug-in do znalezienia tej zmiennej wykorzystuje się proces estymacji jądrowej pochodnej rozkładu prawdopodobieństwa. Do estymacji pochodnej użyty może zostać dowolny inny estymator jądrowy  $K_1$  i dowolny inny parametr wygładzania  $g$ . Proces wyznaczania estymatora jądrowego wymaga znajomości jego drugiej pochodnej, natomiast proces wyznaczania estymatora jądrowego drugiej pochodnej wymaga znajomości pochodnej czwartego rzędu. W ogólności wyznaczenie  $h$  dla pochodnej rzędu  $N$  wymaga znajomości pochodnej rzędu  $N+2$ . Aby uzyskać dokładną wartość  $h_{MISE}$  należało by przeanalizować nieskończoną ilość pochodnych, dlatego też stosuje się pewne przybliżenie. Proces podstawiania kończy się po skończonej ilości iteracji ( w niniejszej pracy 3), a do estymacji pochodnej najwyższego rzędu wykorzystuje się przybliżenie nie oparte o

estymatory jądrowe. Estymator  $\Psi_r$  zdefiniowany jest następująco [5]:

$$\hat{\Psi}_r(g) = \frac{1}{m^2 g^{r+1}} \sum_{i=1}^m \sum_{j=1}^m K_1^{(r)}\left(\frac{x_i - x_j}{g}\right) \quad (16)$$

Po zastosowaniu estymatora wzór na  $h_{MISE}$  wygląda następująco:

$$h_{MISE} = \left[ \frac{R(K)}{mP(K)^2 \hat{\Psi}_4(g_4)} \right]^{\frac{1}{5}} \quad (17)$$

Wartość  $g$  stanowi parametr wygładzania estymatora pochodnej rozkładu gęstości prawdopodobieństwa, i można go obliczyć ze wzoru:

$$g_r = \left[ \frac{2K^{(r)}(0)}{mP(K) \hat{\Psi}_{r+2}(g_{r+2})} \right]^{\frac{1}{r+3}} \quad (18)$$

Jak widzimy, równania wykazują zależność rekurencyjną. W przypadku algorytmu trzeciego stopnia dokonujemy obliczeń  $\Psi_4$ ,  $\Psi_6$  i  $\Psi_8$  według podanych wzorów, natomiast  $\Psi_{10}$  potrzebne do obliczenia  $\Psi_8$  zostaje estymowane przy pomocy wzoru normalnej skali (ang. normal scale) [6]:

$$\hat{\Psi}_r^{NS} = \left[ \frac{-1^{r/2} r!}{(2\hat{\sigma})^{r+1} (r/2)! \pi^{1/2}} \right]^{\frac{1}{9}} \quad (19)$$

## 2.6 Modyfikacja parametru wygładzania

Modyfikacja parametru wygładzania to proces uzależnienia stopnia wygładzania od lokalnej gęstości występowania elementów w estymatorze jądrowym. Algorytm modyfikacji przebiega następująco [5]:

- Obliczenie estymatora gęstości  $\hat{f}_*$  nie używając modyfikacji parametru wygładzania, wykorzystując parametry wygładzania obliczone na przykład przy pomocy metody plug-in
- Obliczenie parametru modyfikacji  $s_i$  dla każdego elementu zbioru danych  $x_i$  według wzoru:

$$s_i = \left( \frac{\hat{f}_*(x_i)}{s} \right)^{-c} \quad (20)$$

Gdzie  $s$  to średnia geometryczna wartości estymatora  $\hat{f}_*$  dla wszystkich elementów zbioru danych, a  $c$  to większa lub równa od zera zmienna oznaczająca siłę modyfikacji.

- Obliczenie estymatora gęstości wykorzystując wyznaczony parametr modyfikacji korzystając ze wzoru:

$$\hat{f}(x) = \frac{1}{mh^n} \sum_{i=1}^m \frac{1}{s_i^n} K\left(\frac{x - x_i}{hs_i^n}\right) \quad (21)$$

Modyfikacja parametru wygładzania ma korzystny wpływ z dwóch przyczyn, po pierwsze zwiększone wygładzanie w rejonach niskiej gęstości znacznie poprawia odporność klasyfikatora na elementy nietypowe. Rekord bardzo oddalony od jakiegokolwiek innego zostanie bardzo mocno wygładzony, dzięki czemu jego wpływ na wynik będzie minimalny. W rejonach wysokiej gęstości nie potrzebne jest duże wygładzanie, a każde wygładzanie prowadzi do utraty informacji (wysokie częstotliwości w funkcji rozkładu gęstości zostają odfiltrowane), dlatego też zmniejszenie stopnia wygładzania w rejonach wysokiej gęstości pozytywnie wpływa na jakość estymacji. Wpływ stosowania modyfikacji wygładzania zostanie zbadany w niniejszej pracy.

## 2.7 Flower Pollination Algorithm

W niniejszej pracy badany będzie wpływ zastosowania algorytmu optymalizacyjnego do doboru parametru wygładzania w probabilistycznej sieci neuronowej. Algorytm jaki został wybrany to oparty o schematy biologiczne Flower Pollination Algorithm (FPA, pol. Algorytm zapylania kwiatów). FPA to algorytm metaheurystyczny, nie istnieją mocne teoretyczne podstawy dla jego skuteczności, proces optymalizacji przebiegać będzie w ten sam sposób niezależnie od optymalizowanej funkcji. Inspiracją do stworzenia tego algorytmu jest proces zapylania kwiatów w którym możemy wyróżnić dwa oddzielne podprocesy [9]:

- Zapylanie lokalne, odbywające się w obrębie jednego kwiatu bądź w obrębie kwiatów tej samej rośliny lub poprzez czynniki abiotyczne jak na przykład wiatr.
- Zapylanie globalne, odbywające się pod wpływem działania organizmów zewnętrznych jak na przykład pszczoły, które mogą pokonywać duże dystanse pomiędzy roślinami

Obydwa procesy wykorzystane zostają w algorytmie, aby je wykorzystać należy przedstawić je w postaci równań aktualizacji pozycji wektora rozwiązań. W

przypadku zapyłania globalnego aktualizacja przebiega według wzoru [9]:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(g_* - x_i^t) \quad (22)$$

Gdzie  $g_*$  to rozwiązanie optymalne w poprzedniej generacji,  $L(\lambda)$  to krok pochodzący z rozkładu Lévy'ego a  $\gamma$  to współczynnik skalujący. Wzór ten zakłada że poruszanie się organizmów zapyłających może być modelowane poprzez loty Lévy'ego [9]. Zapyłanie lokalne modelowane jest przy pomocy wzoru [9]:

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t) \quad (23)$$

Gdzie  $x_j^t$  i  $x_k^t$  to inne wektory rozwiązań, a  $\epsilon$  to rozmiar kroku który losowany jest z rozkładu jednorodnego w przedziale od 0 do 1. Niniejszy wzór modeluje stałość kwiatów (flower constancy), czyli zjawisko w którym obserwuje się chęć organizmów zapyłających do odwiedzania wyłącznie kwiatów tego samego gatunku. Zjawisko to ma uzasadnienie ewolucyjne zarówno dla kwiatów (więcej zapylnych osobników) jak i dla organizmów, które dzięki dużej liczbie zapylnych kwiatów uzyskują dostęp do większej ilości nektaru [9]. Wybór pomiędzy rodzajami zapyłania kontrolowany jest przez prawdopodobieństwo przełączenia (ang. switch probability), faworyzujące zapyłanie lokalne, którego sugerowana w literaturze wartość to 0.8. Przebieg algorytmu:

1. Wylosowanie początkowych wektorów rozwiązań.
2. Znalezienie rozwiązania optymalnego w obecnej generacji
3. Wybór typu zapylenia dla każdego rozwiązania zgodnie z prawdopodobieństwem przełączenia
4. Wyznaczenie nowych rozwiązań korzystając z odpowiedniego z wzorów 22 i 23
5. Wyznaczenie wartości optymalizowanej funkcji w nowych rozwiązaniach, aktualizacja rozwiązań jeżeli nowe rozwiązanie osiągnęło lepszy wynik
6. Powtórzenie kroków 2-5 przez pewną liczbę iteracji





## 3 Implementacja

### 3.1 Język programowania R

W pracy wykorzystano interpretowany język programowania R. R to język służący do przeprowadzania obliczeń i analiz statystycznych i wizualizacji ich wyników. Obecnie jest jednym z najczęściej wykorzystywanych języków w analizie danych i uczeniu maszynowym zaraz po Pythonie. Największą zaletą tego języka jest ogromna ilość dostępnych bibliotek i narzędzi, duża ilość algorytmów z dziedzin statystyki i analizy danych jest już zaimplementowana i dostępna w postaci pakietów. Kolejną zaletą języka R jest fakt że jest dostępny na otwartej licencji GNU. Ilość dostępnych bibliotek i wbudowanych funkcjonalności umożliwiają szybkie pisanie algorytmów, ich testowanie i prototypownie, jednakże niska wydajność języka i brak funkcjonalności związanych z bezpieczeństwem powodują że R nie nadaje się do budowy finalnych aplikacji. R jest językiem który znacznie lepiej sprawdzi się w pracy naukowej. Środowiskiem programistycznym jakie wykorzystano jest darmowe RStudio.



Rysunek 6: Logo języka R

## 3.2 Implementacja PNN

Probabilistyczna sieć neuronowa została zrealizowana jako klasa referencyjna języka R. Taki zabieg ma na celu umożliwienie przechowywania stanu sieci, aby pozwolić na wielokrotne wykorzystanie gotowego modelu do klasyfikacji wielu elementów. Zaimplementowana sieć posiada możliwość wyboru następujących parametrów:

- Parametr wygładzania:
  1. W formie skalara, jeden parametr wygładzania dla wszystkich elementów.
  2. W formie wektora o długości odpowiadającej ilości wymiarów, aby uzyskać inne wygładzanie dla każdego stopnia swobody.
  3. W formie wektora o długości równej liczbie klas, aby zapewnić różne wygładzanie dla każdej klasy.
  4. W formie macierzy aby uzyskać różne parametry dla każdej klasy i każdego wymiaru.
- Jądro estymacji, wybierane jako argument znakowy z listy dostępnych jąder wymienionych w tabeli 2
- Siła modyfikacji parametru wygładzania. W przypadku podania siły 0 obliczenia związane z modyfikacją nie zostają wykonane.

Oprócz wspomnianych parametrów do konstruktora sieci neuronowej przekazujemy również zbiór danych w postaci macierzy oraz wektor etykiet klas. Po inicjalizacji klasa reprezentująca sieć udostępnia metodę `classify` zwracającą etykiety klas umożliwiającą klasyfikację nowych danych. Funkcja `classify` jest zwektoryzowana, można przekazywać do niej całe wektory elementów do zaklasyfikowania, w takim wypadku zostanie zwrócony wektor etykiet klas do których przyporządkowano odpowiednie elementy. Wektoryzacja funkcji znacznie przyspiesza proces klasyfikacji dla większej ilości elementów. Funkcja `classify` na każdym z elementów wywołuje funkcję `get_class_score`, a następnie wybiera etykietę klasy dla której zwrócony został najwyższy wynik. Funkcja `get_class_score` oblicza wartość estymatora gęstości prawdopodobieństwa stworzonego dla danej klasy w badanym punkcie. Wykorzystanie klasy do implementacji algorytmu umożliwia zmniejszenie ilości argumentów przekazywanych pomiędzy funkcjami, co optymalizuje ilość wykorzystywanej pamięci.

Listing 1: Kod funkcji `classify` i `get_class_score`

```
get_class_score = function(x, class_index) {  
  kernel_density=function(cls, mod){  
    univariate=kernel((x - cls)/(smooth[,class_index] * mod))  
    return(colProds(univariate)/mod)  
  }  
  kernels = mapply(kernel_density,  
                    data.frame(test_set[[class_index]]),  
                    modification[[class_index]])  
  m=ncol(test_set[[class_index]])  
  return(rowSums2(to_mat(kernels))/(prod(smooth[,class_index]) * m))  
}  
  
classify = function(x) {  
  x <- t(to_mat(x))  
  class_scores =sapply(seq_along(test_set),  
                       Curry(get_class_score, x = x))  
  return(names(test_set)[max.col(to_mat(class_scores))])  
}
```

Klasa PNN posiada również implementację algorytmu modyfikacji parametrów wygładzania. Jeżeli podana zostanie siła modyfikacji większa od zera, wykonane zostaną obliczenia współczynników modyfikacji dla każdego z elementów zbioru trenującego, a następnie zostaną one użyte do budowania estymatorów jądrowych poszczególnych klas.

Poza wspomnianymi funkcjonalnościami do wykorzystania z klasą PNN zaimplementowano funkcję pozwalającą na wyznaczenie parametrów wygładzania metodą plug-in dla dowolnego jądra wymienionego w tabeli 2. Wykorzystano trzystopniowy algorytm plug-in, do estymacji pochodnych funkcji gęstości wykorzystano jądro Cauchy'ego. Parametry wygładzania obliczane mogą zostać dla całego zbioru danych, bądź dla każdej z klas z osobna.

Dodatkowo w celu ułatwienia prezentacji wyników zaimplementowano funkcję umożliwiającą rysowanie wykresów estymatorów jądrowych wykorzystywanych w klasie PNN. Funkcja wyświetla trójwymiarowy przekrój estymatora dla dwóch zmiennych, a pozostałe zostają ustawioną na średnią wartość w obrębie klasy.



Rysunek 7: Przykład wykresu przekroju estymatora jądrowego

### 3.3 Implementacja FPA

Flower pollination algorithm zaimplementowano w postaci funkcji przyjmującej funkcję do optymalizacji, wymiarowość optymalizowanej zmiennej, rozmiar populacji i maksymalną ilość iteracji które należy wykonać. Do generacji liczb losowych z rozkładu jednostajnego wykorzystano funkcje wbudowane, natomiast do generacji lotów Levy’ego wykorzystano algorytm Mantegna zaproponowany w [9].

Listing 2: Algorytm Mantegana do generacji lotów Levy’ego

```
Levy <- function(d, l, scale) {  
  sigma <- (gamma(1 + l) * sin(pi * l/2)  
            /(gamma((1 + l)/2) * l * 2^((1 - l)/2)))^(1/l)  
  u <- rnorm(d, sd = sigma)  
  v <- rnorm(d)  
  step <- u/(abs(v))^(1/l)  
  return(scale * step)  
}
```

Generacja odbywa się poprzez obliczenie ilorazu zmiennych pochodzących z dwóch rozkładów normalnych,  $u$  i  $v$ . Rozkład  $u$  ma odchylenie standardowe określone w parametrze  $\sigma$ , natomiast rozkład  $v$  ma odchylenie standardowe równe 1.

### 3.4 Integracja algorytmów

Do oceny jakości klasyfikacji w niniejszej pracy wykorzystano dziesięciokrotną walidację krzyżową. W procesie tym zbiór danych dzielony jest na dziesięć równych części, a następnie dziesięciokrotnie przeprowadzana jest klasyfikacja w której jako zbiór treningowy wykorzystuje się dziewięć z dziesięciu części, i sprawdza się jakość klasyfikacji pozostałej jednej dziewiętej. Jakość klasyfikacji mierzona jest jako iloczyn ilości poprawnie zaklasyfikowanych elementów i liczności zbioru klasyfikowanego.

FPA wymaga sformułowania funkcji celu, zaimplementowano wersję algorytmu służącą do maksymalizacji wartości funkcji. Algorytm optymalizacyjny użyty został do automatycznego dobierania wartości parametru wygładzania tak aby osiągnąć najlepszą jakość klasyfikacji. Stworzono więc funkcję której jedynym parametrem wejściowym jest macierz wygładzania, a parametrem wyjściowym jest wynik 10-cio krotnej walidacji krzyżowej procesu klasyfikacji zbioru danych. Optymalizacji można poddać wszystkie możliwości doboru zmienności parametru wygładzania.

FPA jest algorytmem optymalizacji globalnej przeznaczonym do pracy z funkcjami gdzie przestrzeń rozwiązań jest nieograniczona. W przypadku optymalizacji doboru parametrów wygładzania przestrzeń rozwiązań ograniczona jest od dołu przez zero, i teoretycznie nie jest ograniczona z góry. Bardzo wysokie wartości parametru wygładzania bardzo rzadko mają jednak zastosowanie praktycznie, dlatego też można stosować również ograniczenia odgórne. Należało więc zmodyfikować algorytm FPA w taki sposób aby umożliwić podanie wartości granicznych w obrębie których ma zostać przeprowadzona optymalizacja. Aby to osiągnąć rozszerzono sygnaturę funkcji o dwa dodatkowe parametry, ograniczono zakres losowania populacji początkowej tak aby wszystkie wyniki znalazły się wewnątrz ograniczonego przedziału. Kolejnym krokiem modyfikacji było dopasowanie procesu losowania nowych położeń w taki sposób aby zwracane były tylko nowe lokalizacje w obszarze optymalizowanym. Osiągnięto to poprzez powtarzanie losowania tak długo jak otrzymany wynik nie spełnia założonych kryteriów. Powyższa modyfikacja znacznie poprawiła szybkość zbieżności algorytmu, odpowiednie dobranie granic optymalizacji pozwala znacznie przyspieszyć proces.

Ostatnią modyfikacją algorytmu w celu lepszej integracji z zadaniem optymalizacji parametrów wygładzania jest dodanie możliwości wprowadzenia znanego dobrego rozwiązania. Jeżeli chcemy optymalizować funkcję, ale mamy rozwiązanie pochodzące z innego algorytmu możemy podać to rozwiązanie do funkcji FPA, co zapewni że jedno z rozwiązań z populacji początkowej znajdzie się w tym punkcie. W przypadku optymalizacji parametrów wygładzania, jako znane dobre rozwiązanie można wykorzystać na przykład wynik metody plug-in. Taka modyfikacja gwarantuje otrzymanie wyniku nie gorszego niż podany, jednakże może negatywnie wpłynąć na zdolności optymalizacyjne algorytmu. Wpływ modyfikacji zostanie zbadany w kolejnym rozdziale.





## 4 Analiza użyteczności

Do oceny użyteczności algorytmu wykorzystana zostanie dziesięciokrotna walidacja krzyżowa na 4 standardowych zbiorach danych:

Tabela 3: Zbiory danych

Nazwa	Ilość elementów	Wymiarowość
Seeds	210	8
Cardiotocography (CTG)	2126	35
Breast Cancer Wisconsin	569	32
Iris	150	5

Ocenę będzie stanowić stosunek próbek poprawnie zaklasyfikowanych do liczności zbioru danych.

### 4.1 Badanie wpływu doboru parametrów

#### 4.1.1 Wybór jądra estymacji

Istnieje wiele jąder które można wykorzystać do estymacji jądrowej i klasyfikacji przy użyciu PNN, nie można jednoznacznie stwierdzić które jest najlepsze. Bardzo często wybór jądra zależy od zbioru danych, często jest też uwarunkowywany łatwością przeprowadzania obliczeń z wykorzystaniem danego jądra. W tabeli 4 przedstawiono wyniki klasyfikacji osiągnięte przez sieć przy użyciu wszystkich jąder z tabeli 2. Do wykonania klasyfikacji wykorzystano stały parametr wygładzania wynoszący 1 dla wszystkich klas i wymiarów.

Tabela 4: Wybór jądra,  $h=1$

	Normalne	Trójkątne	Cauchyego	Epanechnikowa	Jednostajne	Dwuwagowe
Seeds	0.9095238	0.8904762	0.9	0.8857143	0.8952381	0.8857143
CTG	0.6622265	0.3283063	0.9388542	0.3344406	0.3494707	0.3551023
Breast Cancer	0.4920739	0.4974311	0.9402569	0.5184211	0.5007832	0.4955514
Iris	0.9266667	0.96	0.9533333	0.9666667	0.9133333	0.9666667

Wyniki klasyfikacji w zbiorach o niskiej wymiarowości są zbliżone dla wszystkich jąder, natomiast w zbiorach o wyższej wymiarowości zauważalna jest znaczna

różnica pomiędzy jądrami Cauchiego i Normalnym a pozostałymi. Te dwa jądra odróżniają się od pozostałych faktem nieskończonej szerokości. Wszystkie pozostałe jądra dla wartości o module większym od jedności przyjmują wartość zero. Dokładniejsza analiza wyników gęstości prawdopodobieństwa dla poszczególnych klas pokazuje że w przypadku jąder o skończonym czasie trwania większość badanych elementów uzyskuje gęstość prawdopodobieństwa równą zero. Zjawisko to wynika z klątwy wysokiej wymiarowości (ang. curse of dimensionality). W przestrzeniach o dużej liczbie wymiarów uzyskanie dobrego pokrycia dostępnego miejsca wymaga ogromnej ilości elementów. W przypadku jąder o skończonej szerokości, niskie pokrycie skutkuje powstawaniem przerw w estymatorze. Jeżeli klasyfikowany element trafi na tego rodzaju przerwę, istnieje duże prawdopodobieństwo że nie zostanie poprawnie zaklasyfikowany. Jądra o nieskończonej szerokości, jak na przykład jądro cauchiego radzą sobie z tą sytuacją znacznie lepiej, ponieważ pokrywają one w teorii całą przestrzeń rozwiązań za każdym razem, dlatego też teoretycznie zawsze powinniśmy uzyskać niezerową wartość gęstości prawdopodobieństwa, na podstawie której można dokonać klasyfikacji. W praktyce ograniczona dokładność obliczeń komputerowych powoduje że nawet okna Cauchiego i normalne mają ograniczoną szerokość, jednakże jest ona nadal znacznie większa od szerokości okien skończonych. Próba rozwiązania problemu skończonej szerokości zostanie przeprowadzona w kolejnych podrozdziałach przy pomocy odpowiedniego dostrojenia parametru wygładzania.

W prawie wszystkich przypadkach najlepiej sprawdza się okno Cauchiego. Teoretycznie najlepsze okno Epanechnikowa wypadło najlepiej tylko w jednym przypadku. Powyższy wynik jest pierwszą sugestią że stosowanie metod optymalnych asymptotycznie do estymacji rozkładu prawdopodobieństwa może nie być najlepszym rozwiązaniem do zastosowania w PNN.

#### 4.1.2 Stopnie swobody parametru wygładzania

Parametr wygładzania można dobierać z różną dokładnością. Można dobierać tylko jeden globalny parametr, jak również osobny parametr dla każdego wymiaru zbioru danych czy dla każdej klasy osobno.

Stosowanie doboru osobnego parametru wygładzania dla każdego z wymiarów jest uzasadnione zróżnicowaniem zmienności danych. Bardzo często zdarza się że w

jednym zbiorze danych znajdują się kolumny o odchyleniu standardowym różnych rzędów wielkości. W takiej sytuacji, gdyby dobrano wyłącznie jeden parametr wygładzania jeden z wymiarów musiałby być wy-estymowany niepoprawnie ze względu na zbyt duże lub zbyt małe wygładzenie. Przykład takiej sytuacji wykryto badając zbiór `seeds` wykorzystując okno Epanechnikowa, przy zastosowaniu parametru wygładzania 2 dla wszystkich wymiarów osiągnięto wynik 0.9047619, jednakże część wymiarów przy takim ustawieniu została za bardzo wygładzona i część informacji została utracona. Ustawienie wygładzania jednego z wymiarów na 0.5 przyniosło poprawę wyniku do 0.9142857. Metoda plug-in umożliwia dobór parametrów wygładzania dla każdego z wymiarów.

Osobny parametr wygładzania dla każdej z klas również może być przydatny, można zaistnieć sytuacja w której w jednym zbiorze danych znajdują się klasy bardzo szerokie, charakteryzujące się dużą zmiennością wielu atrybutów jak i klasy bardzo specyficzne, o bardzo niskiej zmienności. W sytuacji kiedy używalibyśmy tylko jednego parametru wygładzania niezależnie od klasy, mogło by się zdarzyć tak że poprawne wygładzenie klasy szerokiej spowodowało by że dwie zbliżone do siebie klasy wąskie zostały by ze sobą połączone. Jednakże dobór osobnych parametrów wygładzania dla każdej z klas niesie ze sobą pewne problemy. Przede wszystkim, wartość parametru wygładzania wpływa na wartość estymowanej funkcji gęstości prawdopodobieństwa. Wysokie wygładzanie prowadzi do niskich wartości gęstości prawdopodobieństwa ponieważ zawsze zachowana musi zostać wartość całki gęstości prawdopodobieństwa. Taki wpływ może powodować błędną klasyfikację, elementy mogą być rzadziej klasyfikowane do klasy bardziej wygładzonej. Kolejnym problemem jest możliwość pojawienia się zerowego odchylenia standardowego. Może zdarzyć się tak że w obrębie klasy dana kolumna będzie miała stałą wartość, a w większości algorytmów wyznaczania parametru wygładzania wygładzanie jest odwrotnie proporcjonalne do odchylenia standardowego. Taka sytuacja prowadzi do wyznaczenia ogromnych bądź nieskończonych wartości parametru wygładzania, co zwykle uniemożliwia klasyfikację.



Rysunek 8: Estymator jądrowy, wygładzanie równe jeden dla każdego wymiaru i klasy



Rysunek 9: Estymator jądrowy, wygładzania 1.5 dla jednej z klas



Rysunek 10: Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów



Rysunek 11: Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów i 1.5 dla jednej z klas



Należy również pamiętać że większa ilość parametrów wygładzania to również większa złożoność obliczeniowa. Algorytmy wyznaczania parametrów wygładzania mają zwykle wysokie wymagania obliczeniowe, metoda plug-in ma złożoność kwadratową. W przypadku optymalizacji przy użyciu FPA większa ilość parametrów do optymalizowania znacznie spowalnia proces optymalizacji.

Tabela 5: Wpływ ilości stopni swobody parametru wygładzania na proces optymalizacji FPA

Wariant \ Ilość iteracji	10	20	100
Jeden parametr	0.9047	0.9047	0.9047
Różne dla wymiarów	0.9523	0.9571	0.9619
Różne dla klas	0.895	0.9047	0.9142
Różne dla wymiarów i klas	0.9238	0.94761	0.9571

Tabela 5 pokazuje że optymalizacja dla parametru wygładzania zmiennego dla każdego wymiaru daje najlepsze rezultaty najszybciej. Można podejrzewać, że optymalizacja dla każdego wymiaru i klasy dała by lepsze rezultaty, ale wymagało by do bardzo dużej ilości iteracji. Z powyższych przyczyn do dalszych rozważań wybrano wariant wyboru parametru wygładzania dla każdego wymiaru, lecz taki sam dla wszystkich klas.

#### 4.1.3 Siła modyfikacji parametru wygładzania

Literatura najczęściej podaje że dobrą wartością siły modyfikacji parametru wygładzania jest 0.5, w niniejszym podrozdziale zbadana zostanie zależność jakości klasyfikacji od wartości parametru siły modyfikacji. Ponadto podjęta zostanie decyzja czy algorytm FPA należy uruchamiać z już ustawionym wygładzaniem, czy wystarczy znaleźć optymalne parametry bez wygładzania.

Pierwszym eksperymentem jaki przeprowadzono jest porównanie wyników uzyskanych przez klasyfikator z parametrem wygładzania wybranym przez algorytm FPA w poprzednim podrozdziale. Zastosowanie parametru modyfikacji spowodowało spadek wyniku z 0.9619 do 0.9285. Można więc wnioskować, że stosowanie parametru wygładzania po zastosowaniu FPA nie jest dobrym rozwiązaniem. Drugim przetestowanym podejściem była optymalizacja parametrów wygładzania z modyfikacją ustawioną na 0.5. Wadą takiego rozwiązania jest

znacznie dłuższy czas wykonywania, ze względu na konieczność obliczenia parametrów modyfikacji dla każdego wektora rozwiązań. Rozwiązanie to również nie okazało się skuteczne, wynik jaki udało się osiągnąć w 100 iteracjach to tylko 0.9095.

Tabela 6: Wpływ siły modyfikacji na jakość klasyfikacji

Wariant \ Siła wygładzania	0	0.5	1
FPA bez modyfikacji, Epanechnikow, seeds	0.9619	0.9047	0.9047
plug-in, Cauchy, seeds	0.9142	0.9142	0.9142
h=1, Cauchy, seeds	0.9	0.8952	0.8952
plug-in, Cauchy, breast cancer	0.9825	0.9825	0.9821

Wyniki zaprezentowane w tabeli 8 wskazują że stosowanie modyfikacji wygładzania ma neutralny lub negatywny wpływ na proces klasyfikacji. Oczywiście powyższy wniosek nie musi być prawdziwy dla innych zbiorów danych, jednakże w niniejszej pracy modyfikacja parametru wygładzania nie zostanie wykorzystana.

#### 4.1.4 Strojenie parametrów FPA

Algorytm Flower Pollination posiada parametry wejściowe, które należy odpowiednio dobrać aby uzyskać najlepszą efektywność. Efektywność algorytmu wyraża się przez szybkość z jaką wynik zbiega do optymalnego, jak również zdolność algorytmu do odnajdywania ekstremów globalnych. W przypadku FPA w klasycznej implementacji zaproponowanej w [9] do inicjalizacji algorytmu należy wybrać jedynie licznosc populacji. Wybór ten ma bardzo duży wpływ na zbieżność algorytmu, czym większa populacja tym szybciej algorytm znajduje coraz lepsze rozwiązania, jednakże większa ilość elementów to również więcej obliczeń do wykonania. Należy również pamiętać, że duża liczba elementów populacji daje większą szanse na znalezienie globalnego celu optymalizacji, niska liczba elementów może prowadzić do zatrzymania algorytmu w lokalnym ekstremum. Wysoka licznosc populacji jest również pożądana w celu zwiększenie pokrycia badanej przestrzeni. Klątwa wymiarowości oznacza że przestrzenie o wysokich wymiarowościach bardzo trudno jest wypełnić, dlatego też zwłaszcza w przypadkach zbiorów o wysokiej wymiarowości istotny jest duży rozmiar populacji początkowej.

W celu doboru optymalnej wartości parametru liczności populacji przeprowadzono test w którym porównano jakość klasyfikacji i czas wykonywania algorytmu przy użyciu liczności 5 i 50 elementów. Test przeprowadzono wykorzystując optymalizację wyniku pięciokrotnej walidacji krzyżowej. Wykorzystując pięcioelementową populację po 100 generacjach osiągnęła wynik 0.9047, obliczenia zajęły 125 sekund. Populacja pięćdziesięciu elementów osiągnęła wynik 0.9619 a czas wykonywania wyniósł 1250 sekund. Różnica w czasie wykonywania jest zrozumiała, złożoność obliczeniowa FPA zależy liniowo od wielkości populacji. FPA wykorzystujące populację 50-ciu elementów osiągnęło wynik lepszy od przypadku pięcioelementowego już w pierwszej generacji, której obliczenie zajęło zaledwie 30 sekund.

Kolejnym elementem mającym wpływ na zbieżność algorytmu FPA jest ograniczenie domeny poszukiwań. Z definicji FPA jest algorytmem optymalizacji globalnej funkcji o nieskończonej dziedzinie, jednakże po zastosowaniu modyfikacji opisanej w podrozdziale 3.4 można wykorzystać go do optymalizacji funkcji ograniczonych. Funkcja jakości klasyfikacji w zależności od wartości parametrów wygładzania jest funkcją ograniczoną, parametr wygładzania nie powinien mieć wartości mniejszej od zera, a wartości wygładzania rzędu dziesiątek czy setek tysięcy w praktyce nigdy nie są wykorzystywane. Dlatego też można ograniczać dziedzinę poszukiwań optymalnego parametru wygładzania. Przeprowadzono test wpływu modyfikacji na zbieżność algorytmu. Usunięcie ograniczeń spowodowało ogromne pogorszenie jakości klasyfikacji i zbieżności algorytmu. W poprzednim teście wykorzystano ograniczenie zakresu parametru wygładzania do przedziału  $(0, 10]$ , po zmianie przedziału na  $(0, 10^6]$  wynik algorytmu w pierwszej iteracji spadł z 0.93 do 0.38, a finalny wynik wyniósł zaledwie 0.9095.

Ostatnią możliwością modyfikacji jest wprowadzona możliwość podania punktu startowego do algorytmu FPA. W badanych zbiorach algorytm FPA niemalże natychmiast znajduje rozwiązanie lepsze od proponowanego przez metodę plug-in, dlatego też rozwiązanie to nie zostało wykorzystane przede wszystkim ze względu na wysoką złożoność obliczeniową metody plug-in. Jednakże, możliwość taka może być przydatna w innych zbiorach danych. Nie zaobserwowano żadnego wpływu stosowania tej modyfikacji w badanych zbiorach danych.

#### 4.1.5 Wpływ normalizacji na klasyfikację

W pracy badane są zbiory danych z atrybutami o bardzo różnych charakterystykach zmienności co wymusza dopasowanie parametrów klasyfikacji mocno różniących się dla każdego wymiaru. Można analizować przykładowo dane na temat wzrostu i na temat zarobków, obydwie zmienne będą miały zarówno różną wartość oczekiwaną jak i odchylenie standardowe. Przeskalowanie zbioru danych w taki sposób aby uniknąć tego typu różnic mogłoby teoretycznie uprościć analizę danych i procedury wyznaczania parametrów do algorytmów, jak na przykład parametr wygładzania. Przeprowadzono test sprawdzający wpływ zastosowania normalizacji przed przystąpieniem do analizy danych na wyniki klasyfikacji.

Bardzo ważny problemem podczas wykorzystywania algorytmów optymalizacyjnych jest nadmierne dopasowanie danych. Zjawisko to wynika ze zbyt długiego treningu, z czasem optymalizowana struktura traci swoje możliwości generalizacji. Zbyt długi trening prowadzi do wyciągania wniosków z danych w sposób nie dający większego zrozumienia problemu, ale pozwalający na osiągnięcie lepszego wyniku. Aby rozwiązać ten problem należy tak dobrać ilość iteracji algorytmu aby uzyskać najbardziej zoptymalizowany wynik nie pozwalając na przetrenowanie. Aby to osiągnąć przeprowadzono testy wykorzystujące pięciokrotną walidację krzyżową. Należy przypomnieć, że algorytm FPA optymalizuje wynik pięciokrotnej walidacji krzyżowej, wykorzystane zostaną dwie walidacje krzyżowe, jedna w procesie tworzenia modelu z wykorzystaniem FPA, a druga do walidacji algorytmu tworzenia modeli. Przyczyną przetrenowania może być zarówno duża ilość iteracji jak i duża populacja. W celu doboru optymalnej ilości iteracji i wielkości populacji przeprowadzono porównanie, wyniki przedstawiono w tabeli 7.

Tabela 7: Wpływ ilości iteracji i wielkości populacji na jakość klasyfikacji dla zbioru seeds.

Liczność populacji \ Ilość iteracji	Ilość iteracji		
	10	20	50
5	0.9285	0.9378	0.9047
10	0.9476	0.939	0.941
20	0.9333	0.8952	0.8952

Najlepszym wynikiem jaki udało się uzyskać jest wynik 0.9476 dla przypadku dziesięciu iteracji i dziesięciu elementów populacji FPA. Takie ustawienia zostaną wykorzystane w niniejszej pracy, jednakże dla każdego zbioru danych wybór tych parametrów należy przeprowadzić osobno.

Tabela 8: Wpływ normalizacji na jakość klasyfikacji

Wariant \ Siła wygładzania	0	0.5	1
FPA bez wygładzania, Epanechnikow, seeds	0.9619	0.9047	0.9047
plug-in, Cauchy, seeds	0.9142	0.9142	0.9142
h=1, Cauchy, seeds	0.9	0.8952	0.8952
plug-in, Cauchy, breast cancer	0.9825	0.9825	0.9821

Można zaobserwować że ze wszystkich badanych przypadkach neutralizacja ma wpływ pozytywny lub neutralny na wyniki klasyfikacji, dlatego też zostanie użyta w finalnym algorytmie. Dodatkowo, zastosowanie normalizacji umożliwia określenie mniejszego rejonu optymalizacji dla algorytmu FPA, który będzie uniwersalny niezależnie od zbioru danych. Ogranicza to liczbę parametrów wejściowych algorytmu, i poprawia czas zbieżności.

#### 4.1.6 Wybór parametru wygładzania

W niniejszej pracy zbadane zostaną dwie metody automatycznego doboru parametru wygładzania, metoda plug-in, oraz optymalizacji przy użyciu FPA. Dobierane będą osobne parametry wygładzania dla każdego z wymiarów. Do analizy wykorzystane zostanie jedno okno o nieskończonej szerokości (Cauchiego) i jedno jądro o skończonej szerokości (Epanechnikowa). Dokonano porównania jakości klasyfikacji dla przypadku wykorzystania stałego wygładzania, dla metody plug-in i dla algorytmu FPA uruchomionego na 100 generacji.

Tabela 9: Dobór parametrów wygładzania dla jądra Cauchiego

	1	plug-in	FPA
Seeds	0.9	0.9142	0.9714
Cardiotocography (CTG)	0.9388	0.9825	0.98824
Breast Cancer Wisconsin	0.9402	0.94558	0.9595
Iris	0.9533	0.9333	0.97333

W tabeli 9 można zauważyć że we wszystkich przypadkach algorytm FPA był w stanie osiągnąć lepsze wyniki niż metoda plug-in. Często już losowo wygenerowana populacja osiąga wynik znacznie lepszy od tego osiągniętego przez metodę plug-in. Dodatkowo w tabeli ??obserwujemy że w przypadku użycia jąder o skończonej szerokości algorytm FPA świetnie radzi sobie z doбором wygładzania pozwalającego na dostateczne wypełnienie przestrzeni, i w przeciwieństwie do algorytmu plug-in umożliwia użycie jąder tego typu. Wynik taki jest kolejnym argumentem potwierdzającym tezę postulującą że stosowanie estymatora gęstości o najmniejszym błędzie średnio kwadratowym i w ogólności dążenie do jak najlepszego odwzorowania funkcji gęstości prawdopodobieństwa nie jest najbardziej optymalne dla zadania klasyfikacji przy użyciu PNN. Przykład jąder o skończonej szerokości pokazuje że dokładne pokrycie pewnych obszarów przestrzeni przez estymator jest ważniejsze niż błąd średnio kwadratowy. Kolejnym argumentem za skutecznością FPA jest możliwość optymalizacji oceniającej istotność konkretnych wymiarów w procesie klasyfikacji. Proces optymalizacji prowadzi do znalezienia takiego rozwiązania gdzie kolumny wprowadzające nieistotne bądź szkodliwe informacje mogą zostać wygładzone znacznie bardziej od pozostałych. Można zauważyć że różnica jakości klasyfikacji jest większa w małych i nisko-wymiarowych zbiorach danych. Fakt ten wynika z mniejszej przestrzeni rozwiązań którą łatwiej jest przeszukać. W tych przypadkach zastosowanie większej populacji i większej liczby iteracji mogło by dać jeszcze lepsze rezultaty.

Jeżeli klasyfikacja pojedynczego elementu zostanie uznana za operację jednostkową to złożoność obliczeniowa algorytmu plug-in jest kwadratowa względem ilości elementów w zbiorze danych, natomiast złożoność obliczeniowa algorytmu FPA jest liniowa. Jednakże, w przypadku algorytmu FPA liniowy czas wykonywania przemnożony zostaje przez stałą wynoszącą iloraz ilości iteracji i wielkości populacji początkowej. Dlatego też w małych zbiorach danych algorytm FPA wykonuje się znacznie wolniej niż algorytm plug-in, jednakże w zbiorach znacznie większych, jeżeli ilość iteracji i wielkość populacji pozostanie na tym samym poziomie algorytm FPA będzie wykonywał się szybciej. Zestawienie czasów wykonania przedstawiono w tabeli 12.

Tabela 10: Czas wykonania algorytmów

	1	plug-in	FPA
Seeds	0.24 s	1.12 s	1250 s
Cardiotocography (CTG)	19 s	336 s	89988 s
Breast Cancer Wisconsin	1.95 s	30.30 s	8793 s
Iris	0.18 s	0.47 s	906.91 s

#### 4.1.7 Finalny algorytm

Na podstawie wszystkich poprzednich obserwacji przygotowano algorytm służący do klasyfikacji który zostanie porównany z innymi dostępnymi metodami klasyfikacji w kolejnym podrozdziale. Założenia algorytmu:

- Wykorzystanie Probabilistycznej sieci neuronowej jako klasyfikatora
- Dobór parametru wygładzania przy pomocy Flower Pollination Algorithm
- Brak modyfikacji parametru wygładzania
- Okno Cauchi'ego
- Strojenie parametrów dla każdego wymiaru osobno, tak samo dla każdej z klas.
- Dokonano normalizacji zbioru danych przed podejściem do klasyfikacji

## 4.2 Porównanie z istniejącymi algorytmami

Do porównania z innymi algorytmami wykorzystana zostanie dziesięciokrotna walidacja krzyżowa. Porównane zostaną wyniki oraz czas wykonywania algorytmu dla wszystkich zbiorów danych wymienionych w tabeli 3. Do porównania wykorzystaną popularne metody klasyfikacji:

- SVM - maszyna wektorów nośnych, jej działanie zostało opisane w podrozdziale 1.2.3. Do porównania wyników wykorzystany zostanie pakiet e1071 i wchodząca w jego skład funkcja svm. Parametry modelu cost i gamma zostały dobrane przy użyciu funkcji svm::tune, następnie stworzono model i przeprowadzono klasyfikację.

- Gradient Boosting (XGBoost) - algorytm polegający na tworzeniu modelu poprzez składanie mniej dokładnych modeli, najczęściej drzew decyzyjnych. Biblioteka XGBoost implementuje algorytm parallel tree boosting, umożliwiając rozwiązywanie problemów klasyfikacyjnych i regresyjnych. Biblioteka ta jest obecnie bardzo popularna w świecie analizy danych. Parametry modelu zostały dobrane eksperymentalnie.
- KNN - algorytm k najbliższych sąsiadów opisany w podrozdziale 1.2.1. Współczynnik k

Tabela 11: Czas wykonania algorytmów

	SVM	XGBoost	kNN	FPA PNN
Seeds	0.9380	0.9333	0.9142	
Cardiotocography (CTG)	0.9901	0.9901	0.9891	
Breast Cancer Wisconsin	0.9771	0.9666	0.97189	
Iris	0.9466	0.94	0.96	0.9333

Tabela 12: Jakość klasyfikacji algorytmów

	SVM	XGBoost	kNN	FPA PNN
Seeds	24.28	2.92	14.82	
Cardiotocography (CTG)	3764	8.30	121	
Breast Cancer Wisconsin	188.4	3.16	20.43	
Iris	12.73	2.22	12.75	





## 5 Podsumowanie

asd



## 6 Kierunki dalszego rozwoju

Niniejsza praca stanowi jedynie badanie użyteczności algorytmu FPA do strojenia probabilistycznych sieci neuronowych, jednakże analizowana dziedzina jest niezwykle szeroka i istnieje jeszcze wiele elementów które należało by zbadać. Dużym problemem prezentowanego algorytmu jest jego czas wykonywania. Przede wszystkim, niniejszy algorytm zaimplementowano w języku R, który świetnie nadaje się do szybkiego prototypowania, jednakże jest znacznie wolniejszy od języków niższego poziomu. Implementacja zarówno FPA jak i PNN w kompilowanym języku jak na przykład C++ przyniosłaby znaczny zysk wydajnościowy. Wartym rozważenia było by również przeniesienie obliczeń na GPU. Należało by również przetestować proponowane modyfikacje algorytmu PNN zakładające zastąpienie umieszczania wszystkich elementów zbioru danych w strukturze sieci umieszczaniem jedynie reprezentantów klastrów. Zwiększenie szybkości wykonywania algorytmu pozwoliło by na wykorzystanie optymalizacji wygładzania dla każdej z klas z osobna, jak również na zwiększenie populacji i liczby iteracji. Dopracowanie samego algorytmu mogło by polegać na zbadaniu innych metod przeszukiwania przestrzeni w celach optymalizacyjnych. Można również przetestować możliwości strojenia parametru modyfikacji wygładzania przy użyciu FPA przy jednoczesnym strojeniu pozostałych parametrów. Zainteresowanie sieciami typu PNN jest obecnie bardzo duże, dlatego też powstaje wiele algorytmów mających na celu poprawę architektury sieci tego typu. Istnieją modyfikację wprowadzające dodatkowe wagi i/lub warstwy, warto było by przeprowadzić próbę wykorzystania algorytmu FPA do optymalizacji wszystkich parametrów takich zmodyfikowanych sieci.



## Bibliografia

- [1] Kramer O. (2013) K-Nearest Neighbors. In: Dimensionality Reduction with Unsupervised Nearest Neighbors. Intelligent Systems Reference Library, vol 51. Springer, Berlin, Heidelberg
- [2] Berk R.A. (2008) Classification and Regression Trees (CART). In: Statistical Learning from a Regression Perspective. Springer Series in Statistics. Springer, New York, NY
- [3] Cortes, C. Vapnik, V. Mach Learn (1995) 20: 273. <https://doi.org/10.1007/BF00994018>
- [4] Evans J.P. (1990) Classifying Artificial Neural Network Architecture. In: International Neural Network Conference. Springer, Dordrecht
- [5] Kulczycki P. (2005) Estymatory jądrowe w analizie systemowej. WNT, Warszawa
- [6] Gramacki A. (2018) Kernel Density Estimation. In: Nonparametric Kernel Density Estimation and Its Computational Aspects. Studies in Big Data, vol 37. Springer, Cham
- [7] Kowalski, P.A. Kulczycki, P. Neural Comput Applic (2017) 28: 817. <https://doi.org/10.1007/s00521-015-2109-3>
- [8] Epanechnikov, V.A. (1969). "Non-parametric estimation of a multivariate probability density". Theory of Probability and its Applications. 14: 153–158.
- [9] Xin-She Yang, *Nature-Inspired Optimization Algorithms*, Rozdział 11 – Flower Pollination Algorithms 2014.
- [10] Oficjalna dokumentacja języka R, <https://www.r-project.org/>, ostatni dostęp: 03.05.2018

## Spis rysunków

1	Struktura drzewa decyzyjnego w algorytmie ID3 . . . . .	17
2	Liniowy podział zbioru danych. . . . .	18
3	Uproszczony schemat sieci neuronowej. . . . .	19
4	Jądrowy estymator gęstości jednowymiarowej funkcji . . . . .	23
5	Struktura probabilistycznej sieci neuronowej . . . . .	24
6	Logo języka R . . . . .	34
7	Przykład wykresu przekroju estymatora jądrowego . . . . .	37
8	Estymator jądrowy, wygładzanie równe jeden dla każdego wymiaru i klasy . . . . .	45
9	Estymator jądrowy, wygładzania 1.5 dla jednej z klas . . . . .	46
10	Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów . . . . .	47
11	Estymator jądrowy, wygładzanie 2 dla jednego z wymiarów i 1.5 dla jednej z klas . . . . .	48

## Spis tabel

1	Jądra . . . . .	26
2	Jądra cz.2 . . . . .	27
3	Zbiory danych . . . . .	42
4	Wybór jądra, $h=1$ . . . . .	42
5	Wpływ ilości stopni swobody parametru wygładzania na proces optymalizacji FPA . . . . .	49
6	Wpływ siły modyfikacji na jakość klasyfikacji . . . . .	50
7	Wpływ ilości iteracji i wielkości populacji na jakość klasyfikacji dla zbioru seeds. . . . .	52
8	Wpływ normalizacji na jakość klasyfikacji . . . . .	53
9	Dobór parametrów wygładzania dla jądra Cauchiego . . . . .	53
10	Czas wykonania algorytmów . . . . .	55
11	Czas wykonania algorytmów . . . . .	56
12	Jakość klasyfikacji algorytmów . . . . .	56



## Spis listingów

1	Kod funkcji <code>classify</code> i <code>get_class_score</code> . . . . .	36
2	Algorytm Mantegana do generacji lotów Levy'ego . . . . .	38

## Dodatki