Instytut Informatyki
Uniwersytet Rzeszowski

Przedmiot:

Sieci Rozmyte

Porównanie klasycznego algorytmu kNN z rozmytym algorytmem Fuzzy kNN

Wykonał:

Piotr Rojek, 125159

Prowadzący: pracownik UR

Rzeszów 2025

Spis treści

1. Wstęp

Celem niniejszego projektu jest porównanie skuteczności klasycznego algorytmu k-Nearest Neighbors (kNN), zwanego algorytmem k-najbliższych sąsiadów, z jego rozmytą wersją Fuzzy kNN w zadaniach klasyfikacji danych. W przeciwieństwie do klasycznego kNN, który przypisuje próbkę testową do jednej, konkretnej klasy na podstawie głosowania większościowego, algorytm Fuzzy kNN umożliwia określenie stopnia przynależności próbki do każdej z klas decyzyjnych. Pozwala to na bardziej elastyczną oraz informacyjną interpretację wyników klasyfikacji, zwłaszcza w przypadkach granicznych.

Projekt ma na celu odpowiedzieć na następujące pytania badawcze:

1. Czy zastosowanie rozmycia w algorytmie kNN poprawia jakość klasyfikacji?
2. Jak wpływają parametry rozmycia (k_init, m) na stabilność i dokładność modelu?
3. Czy Fuzzy kNN zapewnia lepszą interpretowalność decyzji klasyfikatora?
4. Jaki wpływ ma skalowanie danych na oba algorytmy?

W ramach projektu przeprowadzono eksperymenty porównawcze na rzeczywistych zbiorach danych, obejmujące analizę ilościową oraz jakościową wyników klasyfikacji.

1.1. Użyte technologie

- **Środowisko i język programowania:**
  - Środowisko: PyCharm Community Edition 2024.3.
  - Język programowania: python 3.13.
- **Użyte biblioteki:**
  - NumPy – operacje numeryczne oraz obliczanie odległości pomiędzy próbkami danych.
  - Pandas – przechowywanie i przetwarzanie zbiorów danych w strukturach DataFrame.
  - Matplotlib – wizualizacja wyników eksperymentów, w tym wykresy punktowe i macierze pomyłek.
  - scikit-learn – implementacja klasycznego algorytmu kNN, podział danych, skalowanie, walidacja krzyżowa oraz obliczanie metryk jakości klasyfikacji.
  - statistics – obliczanie odchylenia standardowego wyników walidacji krzyżowej.

1.2. Zakres prac

1. **Przygotowanie środowiska:**
   - Konfiguracja środowiska programistycznego Python i przygotowanie środowiska pracy PyCharm.
   - Instalacja i konfiguracja bibliotek NumPy, Pandas, scikit-learn oraz Matplotlib.
2. **Przygotowanie danych:**
   - Wczytanie zbiorów danych Seeds oraz Banknote Authentication.
   - Wydzielenie macierzy cech oraz wektorów etykiet klas.
   - Podział danych na zbiory treningowe i testowe w proporcji 70% / 30% z zachowaniem rozkładu klas.
   - Przygotowanie skalowanych i nieskalowanych wersji danych.
   - Standaryzacja cech przy użyciu algorytmu StandardScaler.

3. **Implementacja algorytmów:**
    - Wykorzystanie klasycznego algorytmu k-Nearest Neighbors z biblioteki scikit-learn.
    - Wykorzystanie autorskiej implementacji klasy Fuzzy kNN autorstwa Sahila Sehwaga (https://github.com/sahilsehwag/FuzzyKNN).
    - Przygotowanie metod trenowania, predykcji oraz obliczania stopni przynależności do klas decyzyjnych:
4. **Przeprowadzenie eksperymentów:**
    - Testowanie różnych wartości parametru liczby sąsiadów k.
    - Porównanie działania algorytmów dla danych skalowanych oraz nieskalowanych.
    - Analiza wpływu parametrów rozmycia k_init oraz m w algorytmie Fuzzy kNN.
    - Automatyczne wykonywanie serii eksperymentów dla obu zbiorów danych.
5. **Ewaluacja wyników:**
    - Obliczenie dokładności klasyfikacji (accuracy) dla zbioru testowego.
    - Wyznaczenie macierzy pomyłek dla poszczególnych konfiguracji algorytmów.
    - Przeprowadzenie pięciokrotnej walidacji krzyżowej z użyciem StratifiedKFold.
    - Obliczenie metryki ROC AUC dla problemów binarnych oraz wieloklasowych.
6. **Wizualizacja i analiza wyników:**
    - Wizualizacja klas rzeczywistych oraz klas przewidywanych przez algorytmy.
    - Prezentacja macierzy pomyłek w formie graficznej.
    - Wizualizacja rozmytych stopni przynależności próbek do klas w algorytmie Fuzzy kNN.
    - Porównanie jakościowe i ilościowe wyników klasycznego kNN oraz Fuzzy kNN.

2. Podstawy teoretyczne

2.1. Klasyczny algorytm kNN

Algorytm k-Nearest Neighbors (kNN) należy do grupy algorytmów klasyfikacji opartych na przykładach i jest jednym z najprostszych podejść w uczeniu maszynowym. Jego działanie polega na klasyfikacji obiektów na podstawie podobieństwa do danych uczących, bez konieczności budowania jawnego modelu w fazie uczenia.

Dla każdej próbki testowej algorytm oblicza odległość (najczęściej euklidesową, choć mogą być stosowane również inne metryki, takie jak odległość Manhattan) do wszystkich próbek ze zbioru treningowego, a następnie wybiera k najbliższych sąsiadów. Klasa decyzyjna przypisywana jest na podstawie głosowania większościowego wśród wybranych sąsiadów, co oznacza, że próbka testowa zostaje przypisana do klasy występującej najczęściej wśród k najbliższych obserwacji. W przypadku remisu głosów możliwe jest losowe przypisanie próbki do jednej z klas zremisowanych.

Do głównych zalet algorytmu kNN należy jego prostota, intuicyjność oraz brak etapu uczenia w klasycznym rozumieniu. Algorytm ten dobrze sprawdza się w zadaniach o niewielkiej liczbie cech oraz przy odpowiednio dobranej metryce odległości. Do jego wad zalicza się wysoką złożoność obliczeniową w fazie predykcji, wrażliwość na dobór parametru k oraz brak informacji o niepewności podjętej decyzji klasyfikacyjnej, szczególnie w przypadkach granicznych.

2.2. Rozmyty algorytm Fuzzy kNN

Klasyfikacja rozmyta opiera się na pojęciu stopnia przynależności, który określa, w jakim stopniu dany obiekt należy do poszczególnych klas decyzyjnych. W przeciwieństwie do klasycznej klasyfikacji, gdzie obiekt przypisywany jest wyłącznie do jednej klasy, podejście rozmyte umożliwia jednoczesne przypisanie obiektu do wielu klas z różnym stopniem przynależności. Takie rozwiązanie jest szczególnie użyteczne w problemach, w których granice pomiędzy klasami są nieostre lub częściowo się nakładają.

Algorytm Fuzzy kNN stanowi rozszerzenie klasycznego algorytmu kNN o elementy logiki rozmytej. W pierwszym etapie inicjalizowane są rozmyte przynależności próbek treningowych na podstawie parametru k_init, określającego liczbę sąsiadów branych pod uwagę przy inicjalizacji. W fazie klasyfikacji algorytm uwzględnia zarówno odległość pomiędzy próbkami, jak i stopnie przynależności sąsiadów do poszczególnych klas.

Wynikiem działania algorytmu Fuzzy kNN jest nie tylko klasa decyzyjna, ale również wektor stopni przynależności do wszystkich klas, co umożliwia bardziej informacyjną interpretację wyniku klasyfikacji. Istotnym parametrem algorytmu jest współczynnik m, który kontroluje stopień rozmycia. Wraz ze wzrostem jego wartości zmniejsza się wpływ dalszych sąsiadów na końcową decyzję klasyfikatora. Dzięki temu algorytm pozwala na analizę nie tylko ostatecznej decyzji klasyfikacyjnej, ale również stopnia pewności tej decyzji dla poszczególnych klas. Takie podejście znajduje bezpośrednie zastosowanie w przeprowadzonych eksperymentach, gdzie wektory przynależności wykorzystywane są do porównania działania klasycznego kNN oraz rozmytego Fuzzy kNN.

3. Zbiory danych

W celu przeprowadzenia eksperymentów porównawczych wykorzystano dwa publicznie dostępne zbiory danych pochodzące z UCI Machine Learning Repository, które są często stosowane w badaniach z zakresu uczenia maszynowego. Zbiory te zostały dobrane w taki sposób, aby obejmowały zarówno problem klasyfikacji wieloklasowej, jak i binarnej, co pozwala na wszechstronną ocenę skuteczności oraz właściwości algorytmów kNN i Fuzzy kNN w różnych scenariuszach klasyfikacyjnych.

3.1. Seeds (Ziarna Pszenicy)

Źródło danych:

- **Tytuł:** UCI Machine Learning Repository – Seeds
- **Link:** https://archive.ics.uci.edu/dataset/236/seeds

Opis problemu:

Zbiór danych Seeds opisuje problem klasyfikacji wieloklasowej, którego celem jest rozróżnienie trzech odmian pszenicy na podstawie ich cech geometrycznych. Dane zostały pozyskane na podstawie analizy obrazu ziaren pszenicy i są często wykorzystywane w badaniach porównawczych algorytmów klasyfikacji.

Charakterystyka zbioru danych:

- **Liczba próbek:** 120
- **Liczba cech:** 7
- **Liczba klas:** 3
- **Typ problemu:** klasyfikacja wieloklasowa

Klasy decyzyjne:

- Kama
- Rosa
- Canadian

W implementacji etykiety klas zostały przeskalowane z zakresu {1, 2, 3} do {0, 1, 2} w celu ujednolicenia zapisu oraz zapewnienia pełnej kompatybilności z biblioteką scikit-learn.

Cechy opisujące ziarna:

- Pole powierzchni
- Obwód
- Zwartość
- Długość ziarna
- Szerokość ziarna
- Współczynnik asymetrii
- Długość bruzdy

Zbiór danych Seeds charakteryzuje się niewielkim rozmiarem, dobrą separowalnością klas oraz brakiem brakujących wartości, co czyni go szczególnie odpowiednim do analizy porównawczej działania algorytmów klasyfikacji oraz oceny wpływu parametrów modelu na jakość uzyskiwanych wyników.

3.2. Banknote Authentication (Uwierzytelnianie Banknotów)

Źródło danych:

- **Tytuł:** UCI Machine Learning Repository – Banknote Authentication
- **Link:** https://archive.ics.uci.edu/dataset/267/banknote+authentication

Opis problemu:

Zbiór danych Banknote Authentication dotyczy problemu klasyfikacji binarnej, którego celem jest określenie, czy dany banknot jest prawdziwy, czy fałszywy. Dane zostały wygenerowane na podstawie analizy obrazów banknotów z wykorzystaniem transformacji falkowej.

Charakterystyka zbioru danych:

- **Liczba próbek:** 280
- **Liczba cech:** 4
- **Liczba klas:** 2
- **Typ problemu:** klasyfikacja binarna

Klasy decyzyjne:

- Prawdziwy
- Fałszywy

Cechy opisujące banknoty:

- Wariancja
- Skośność
- Kurtoza
- Entropia

Zbiór danych Banknote Authentication charakteryzuje się stosunkowo dobrą separacją klas, zróżnicowanymi skalami cech oraz brakiem brakujących wartości, co czyni go dobrym przykładem do analizy wpływu skalowania danych na skuteczność algorytmów klasyfikacyjnych.

4. Opis działania programu

Głównym celem programu jest przeprowadzenie kompleksowego porównania skuteczności klasycznego algorytmu k-Nearest Neighbors (kNN) z jego rozmytą wersją Fuzzy kNN dla dwóch zbiorów danych: Seeds oraz Banknote Authentication. Program został zaprojektowany w sposób sekwencyjny, tak aby umożliwić automatyczne testowanie wielu konfiguracji algorytmów, ich ewaluację z wykorzystaniem różnych metryk oraz wizualizację uzyskanych wyników. Działanie programu można podzielić na kilka logicznych etapów.

4.1. Inicjalizacja środowiska i import bibliotek

Na początku programu importowane są wszystkie niezbędne biblioteki wykorzystywane w dalszych etapach przetwarzania danych. Obejmują one biblioteki numeryczne (NumPy, Pandas), narzędzia do wizualizacji wyników (Matplotlib), a także moduły biblioteki scikit-learn odpowiedzialne za klasyfikację, walidację krzyżową oraz obliczanie metryk jakości. Dodatkowo importowana jest implementacja klasy FuzzyKNN autorstwa Sahila Sehwaga. Etap ten przygotowuje środowisko do dalszych obliczeń oraz zapewnia dostęp do wszystkich funkcji wykorzystywanych w kolejnych częściach programu.

4.2. Definicja funkcji pomocniczych

Program wykorzystuje zestaw funkcji pomocniczych, które porządkują strukturę kodu oraz zwiększają jego czytelność i modularność. Należą do nich:

- Funkcja one_line formatująca tablice NumPy do czytelnego wypisu w jednej linii.
- Funkcja add_legend odpowiedzialna za tworzenie legend na wykresach.
- Funkcja get_data, która w zależności od wybranego zbioru danych oraz informacji o skalowaniu zwraca odpowiednie dane treningowe i testowe.
- Funkcja test_classifier realizująca pełen cykl testowania klasyfikatora, obejmujący trenowanie modelu, predykcję na zbiorze testowym, obliczenie dokładności oraz wyznaczenie macierzy pomyłek.
- Funkcja print_result wypisująca szczegółowe wyniki testowania klasyfikatora.
- Funkcja print_best_or_worst_classifier prezentująca najlepszą oraz najgorszą konfigurację klasyfikatora dla obu analizowanych zbiorów danych.

Zastosowanie funkcji pomocniczych pozwala uniknąć powielania kodu oraz ujednolica sposób testowania wszystkich konfiguracji algorytmów.

4.3. Wczytanie i przygotowanie danych

W kolejnym etapie program wczytuje dwa zbiory danych: Seeds oraz Banknote Authentication. Dane są rozdzielane na macierze cech wejściowych oraz wektory etykiet klas. Następnie wykonywany jest podział na zbiór treningowy i testowy w proporcji 70% do 30%, z zachowaniem rozkładu klas przy użyciu mechanizmu stratyfikacji.

Dla obu zbiorów danych przygotowywane są dwie wersje danych:

- Wersja nieskalowana.
- Wersja skalowana z wykorzystaniem standaryzacji (StandardScaler).

Dzięki temu możliwe jest bezpośrednie porównanie wpływu skalowania danych na działanie algorytmów kNN oraz Fuzzy kNN.

4.4. Definicja przestrzeni eksperymentów

Program definiuje zestawy hiperparametrów, które będą testowane w trakcie eksperymentów. Obejmują one:

- Różne wartości liczby sąsiadów k.
- Różne wartości parametrów k_init oraz m w algorytmie Fuzzy kNN.
- Dwie wersje danych (skalowane i nieskalowane).
- Dwa zbiory danych.

Na tej podstawie tworzona jest pełna przestrzeń eksperymentów, w której każda konfiguracja algorytmu jest testowana niezależnie.

4.5. Pętla eksperymentalna klasycznego algorytmu kNN

Dla każdej kombinacji zbioru danych, liczby sąsiadów k oraz wersji danych program:

- Tworzy obiekt klasycznego klasyfikatora kNN.
- Trenuje go na zbiorze treningowym.
- Wykonuje predykcję na zbiorze testowym.
- Oblicza dokładność klasyfikacji.
- Wyznacza macierz pomyłek.

Następnie przeprowadzana jest pięciokrotna walidacja krzyżowa z wykorzystaniem StratifiedKFold. W zależności od wersji danych stosowany jest:

- Mechanizm Pipeline (dla danych skalowanych).
- Bezpośrednie użycie klasyfikatora (dla danych nieskalowanych).

Uzyskane wartości dokładności oraz metryki ROC AUC są zapisywane do dalszej analizy.

4.6. Pętla eksperymentalna rozmytego algorytmu Fuzzy kNN

Po przetestowaniu klasycznego algorytmu kNN program przechodzi do testowania rozmytego algorytmu Fuzzy kNN wewnątrz pętli testowania klasycznego kNN. Dla każdej kombinacji parametrów k_init oraz m program:

- Tworzy obiekt rozmytego klasyfikatora Fuzzy kNN.
- Trenuje go na zbiorze treningowym.
- Wykonuje predykcję na zbiorze testowym.
- Oblicza dokładność klasyfikacji.
- Wyznacza macierz pomyłek.

Analogicznie jak w przypadku klasycznego kNN wykonywana jest pięciokrotna walidacja krzyżowa z wykorzystaniem StratifiedKFold. W zależności od wersji danych stosowany jest:

- Mechanizm Pipeline (dla danych skalowanych).
- Bezpośrednie użycie klasyfikatora (dla danych nieskalowanych).

Dodatkowo algorytm Fuzzy kNN umożliwia wyznaczenie rozkładu stopni przynależności do klas, co pozwala na analizę niepewności decyzji klasyfikatora. Uzyskane wartości dokładności oraz metryki ROC AUC są zapisywane do dalszej analizy i porównań wyników.

4.7. Agregacja i analiza wyników

Wyniki wszystkich eksperymentów zapisywane są w strukturach danych zawierających:

- Nazwę algorytmu i konfiguracji.
- Nazwę zbioru danych.
- Informację o skalowaniu.
- Predykcje.
- Dokładność klasyfikacji.
- Macierze pomyłek.

Na tej podstawie program identyfikuje najlepsze i najgorsze konfiguracje algorytmów, porównuje klasyczny kNN z rozmytym Fuzzy kNN osobno dla każdego zbioru danych oraz wypisuje szczegółowe podsumowanie wyników w konsoli.

4.8. Wizualizacja wybranych wyników

Ostatnim etapem działania programu jest wizualizacja wyników dla konfiguracji parametrów k = 3, scaled = True, k_init = 3, m = 2. Obejmuje ona:

- Porównanie macierzy pomyłek klasycznego kNN i rozmytego Fuzzy kNN.
- Wizualizację klas rzeczywistych oraz przewidywanych na wykresach punktowych.
- Prezentację rozmytych przynależności próbek testowych do poszczególnych klas.

Wizualizacje umożliwiają jakościową ocenę działania algorytmów oraz lepsze zrozumienie różnic pomiędzy podejściem ostrym i rozmytym.

4.9. Podsumowanie działania kodu

Zaimplementowany program realizuje kompletny proces eksperymentalny, od wczytania i przygotowania danych, przez trenowanie i testowanie modeli, aż po ewaluację oraz wizualizację wyników. Struktura kodu umożliwia łatwe rozszerzenie projektu o kolejne algorytmy lub zbiory danych oraz zapewnia pełną powtarzalność przeprowadzonych eksperymentów.

5. Eksperymenty i wyniki

Eksperymenty przeprowadzono w celu porównania skuteczności klasycznego algorytmu kNN oraz jego rozmytej wersji Fuzzy kNN dla dwóch zbiorów danych: Seeds oraz Banknote Authentication. Zbiory te reprezentują odpowiednio problem klasyfikacji wieloklasowej oraz binarnej, co umożliwia ocenę działania algorytmów zarówno w przypadku rozróżniania wielu klas decyzyjnych, jak i w prostszym scenariuszu klasyfikacji dwuklasowej. Dzięki temu możliwe było przeanalizowanie wpływu zastosowania podejścia rozmytego na jakość klasyfikacji w różnych typach problemów.

Dla obu zbiorów danych zastosowano:

- Podział na zbiór treningowy (70%) oraz testowy (30%) z zachowaniem proporcji klas.
- Dwie wersje danych: skalowaną oraz nieskalowaną.
- Pięciokrotną walidację krzyżową z wykorzystaniem mechanizmu StratifiedKFold.

Jako miary jakości klasyfikacji wykorzystano:

- Dokładność klasyfikacji (accuracy).
- Pole pod krzywą ROC (ROC AUC) w wariancie one-vs-one dla problemu wieloklasowego oraz klasycznym dla problemu binarnego.
- Macierze pomyłek.

Eksperymenty wykonano dla różnych konfiguracji parametrów:

- Liczby sąsiadów k oraz danych skalowanych i nieskalowanych dla klasycznego kNN oraz rozmytego Fuzzy kNN.
- Parametrów k_init oraz m w algorytmie Fuzzy kNN.

5.1. Wyniki dla zbioru danych Seeds

Dla zbioru danych Seeds, po zastosowaniu standaryzacji cech, klasyczny algorytm kNN osiągnął najwyższą skuteczność klasyfikacji dla parametru k = 5.

Klasyczny kNN – najlepsza konfiguracja (k = 5, scaled = True):

- Dokładność (accuracy) na zbiorze testowym: 0.94444.
- Liczba błędnych klasyfikacji: 2 próbki.
- Macierz pomyłek wskazuje na bardzo dobrą separację klas, z pojedynczymi pomyłkami pomiędzy klasami skrajnymi.

Najgorsze wyniki klasycznego kNN uzyskano dla danych nieskalowanych oraz mniejszej liczby sąsiadów.

Klasyczny kNN – najgorsza konfiguracja (k = 3, scaled = False):

- Dokładność (accuracy) na zbiorze testowym: 0.86111.
- Liczba błędnych klasyfikacji: 5 próbek.
- Pomyłki dotyczą głównie klas Kama oraz Canadian.

W przypadku rozmytego algorytmu Fuzzy kNN najlepsze wyniki uzyskano dla konfiguracji z większą liczbą sąsiadów oraz danych skalowanych.

Rozmyty Fuzzy kNN – najlepsza konfiguracja (k = 5, scaled = True, k_init = 5, m = 2):

- Dokładność (accuracy) na zbiorze testowym: 0.94444.
- Liczba błędnych klasyfikacji: 2 próbki.
- Macierz pomyłek identyczna jak dla najlepszego klasycznego kNN.

Najgorsza konfiguracja Fuzzy kNN odpowiadała przypadkowi danych nieskalowanych i najmniejszej liczby sąsiadów.

Rozmyty Fuzzy kNN – najgorsza konfiguracja (k = 3, scaled = False, k_init = 3, m = 2):
- Dokładność (accuracy) na zbiorze testowym: 0.86111.
- Liczba błędnych klasyfikacji: 5 próbek.
- Macierz pomyłek identyczna jak w najgorszym przypadku klasycznego kNN.

Uzyskane wyniki pokazują, że dla zbioru Seeds skalowanie danych ma kluczowe znaczenie dla jakości klasyfikacji, natomiast zastosowanie rozmycia nie zwiększa bezpośrednio dokładności klasyfikatora, lecz dostarcza dodatkowej informacji o niepewności decyzji klasyfikatora.

5.2. Wyniki dla zbioru danych Banknote Authentication

Zbiór danych Banknote Authentication charakteryzuje się bardzo dobrą separacją klas, co przełożyło się na wysoką skuteczność klasyfikacji dla obu algorytmów.

Klasyczny kNN – najlepsza konfiguracja (k = 5, scaled = False):
- Dokładność (accuracy) na zbiorze testowym: 0.98810.
- Liczba błędnych klasyfikacji: 1 próbka.
- Macierz pomyłek wskazuje na niemal idealną separację klas.

Najgorsza konfiguracja klasycznego kNN wystąpiła dla mniejszej liczby sąsiadów oraz również nieskalowanych danych.

Klasyczny kNN – najgorsza konfiguracja (k = 3, scaled = False):
- Dokładność (accuracy) na zbiorze testowym: 0.95238.
- Liczba błędnych klasyfikacji: 4 próbki.
- Większa liczba fałszywych klasyfikacji banknotów prawdziwych.

W przypadku rozmytego algorytmu Fuzzy kNN najlepsze wyniki uzyskano dla konfiguracji z większą liczbą sąsiadów oraz wyższym stopniem rozmycia.

Rozmyty Fuzzy kNN – najlepsza konfiguracja (k = 5, scaled = False, k_init = 5, m = 10):
- Dokładność (accuracy) na zbiorze testowym: 0.98810.
- Liczba błędnych klasyfikacji: 1 próbka.
- Identyczna skuteczność jak w najlepszym przypadku klasycznego kNN oraz stabilne wyniki niezależnie od parametrów rozmycia.

Najgorsza konfiguracja Fuzzy kNN odpowiadała przypadkowi najmniejszej liczby sąsiadów, małemu stopniu rozmycia i początkowej inicjalizacji, ale nadal charakteryzowała się wysoką skutecznością.

Rozmyty Fuzzy kNN – najgorsza konfiguracja (k = 3, scaled = False, k_init = 3, m = 2):

- Dokładność (accuracy) na zbiorze testowym: 0.96429.
- Liczba błędnych klasyfikacji: 3 próbki.
- Wyniki lepsze niż najgorszy klasyczny kNN.

Dla zbioru Banknote Authentication różnice pomiędzy klasycznym kNN a Fuzzy kNN są niewielkie, co wynika z bardzo dobrej separowalności klas. Zastosowanie rozmycia nie daje istotnego wzrostu dokładności, lecz potwierdza stabilność algorytmu w różnych konfiguracjach parametrów.

6. Analiza wizualna wyników eksperymentów

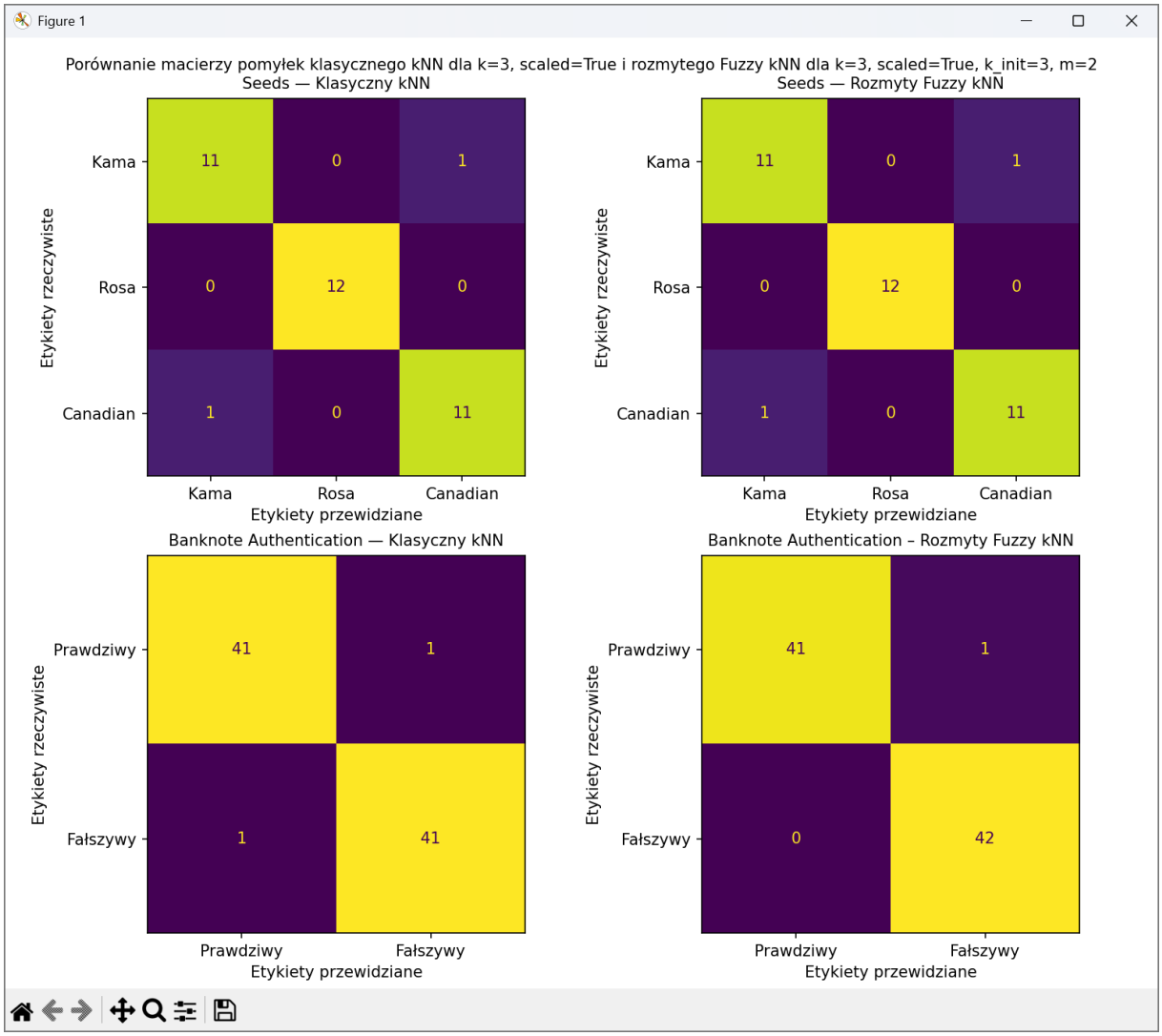
W celu pogłębionej analizy działania klasycznego algorytmu kNN oraz jego rozmytej wersji Fuzzy kNN wykorzystano wizualizacje wyników klasyfikacji. Obejmują one porównanie macierzy pomyłek, rozkładu etykiet rzeczywistych i przewidywanych oraz wizualizację stopni przynależności próbek testowych do poszczególnych klas. Analiza wizualna pozwala na jakościową ocenę skuteczności algorytmów oraz lepsze zrozumienie różnic pomiędzy podejściem ostrym i rozmytym. Analiza dotyczy przypadków z użyciem parametrów k = 3, scaled = True, k_init = 3, m = 2.

6.1. Porównanie macierzy pomyłek dla klasycznego kNN i rozmytego Fuzzy kNN

Na rysunku poniżej przedstawiono porównanie macierzy pomyłek dla klasycznego algorytmu kNN oraz rozmytego Fuzzy kNN dla zbiorów danych Seeds oraz Banknote Authentication, przy identycznych parametrach klasyfikacji.

Dla zbioru Seeds macierze pomyłek obu algorytmów są identyczne, co potwierdza, że w tej konfiguracji zastosowanie rozmycia nie wpłynęło na końcową decyzję klasyfikacyjną. Błędne klasyfikacje występują sporadycznie i dotyczą głównie pomyłek pomiędzy klasami Kama oraz Canadian, co jest zgodne z ich częściowo nachodzącymi na siebie cechami geometrycznymi.

W przypadku zbioru Banknote Authentication macierze pomyłek wskazują na niemal idealną separację klas. Zarówno klasyczny kNN, jak i Fuzzy kNN popełniają bardzo niewielką liczbę błędów, co potwierdza wysoką skuteczność algorytmów dla danych o wyraźnym podziale klas decyzyjnych. Jednak w przypadku rozmytego algorytmu zaobserwowano nieco mniejszą liczbę błędnych klasyfikacji, co wskazuje na jego większą stabilność decyzyjną w obszarach granicznych przestrzeni cech.



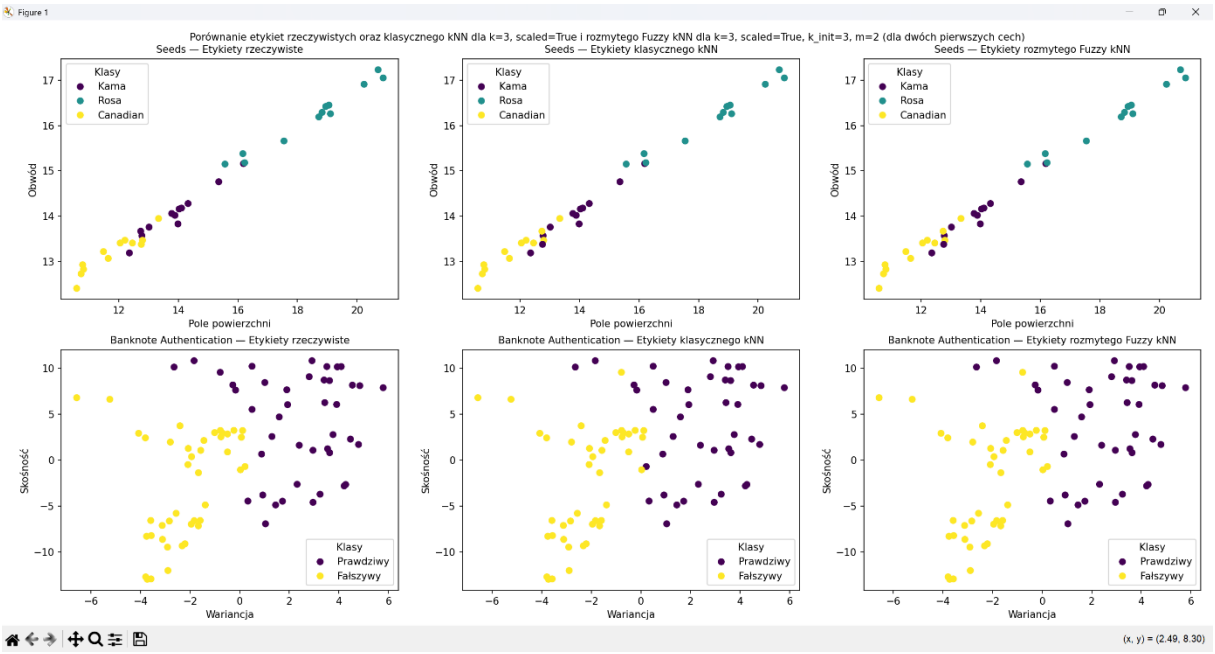
6.2. Wizualizacja etykiet rzeczywistych i przewidywanych

Kolejna wizualizacja przedstawia porównanie etykiet rzeczywistych oraz etykiet przewidywanych przez klasyczny kNN i rozmyty Fuzzy kNN dla obu zbiorów, z wykorzystaniem dwóch pierwszych cech.

W przypadku zbioru Seeds wykorzystano cechy pole powierzchni oraz obwód ziarna. Rozkład punktów pokazuje wyraźne skupienia odpowiadające poszczególnym klasom pszenicy. W obu algorytmach granice decyzyjne są bardzo zbliżone, a większość próbek została sklasyfikowana poprawnie. Ograniczona liczba błędów pojawia się w obszarach granicznych pomiędzy klasami, gdzie cechy próbek są do siebie najbardziej zbliżone. Porównanie wykresów dla klasycznego i rozmytego kNN wskazuje, że Fuzzy kNN nie zmienia etykiet końcowych, jednak jego przewaga ujawnia się na etapie analizy stopni przynależności, co zostało przedstawione w kolejnych wizualizacjach.

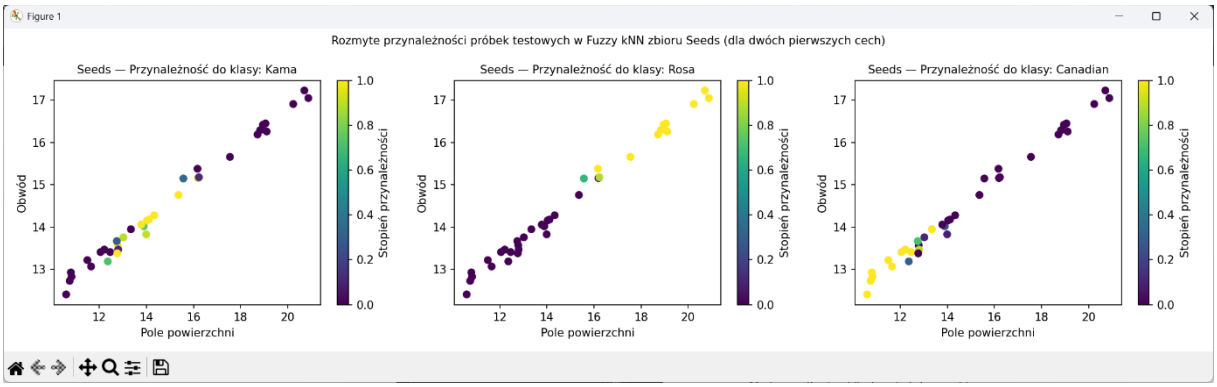
Analogiczna wizualizacja wykonana dla zbioru Banknote Authentication, z wykorzystaniem cech wariancji i skośności, potwierdza bardzo dobrą separowalność klas. Zarówno klasyczny

kNN, jak i Fuzzy kNN poprawnie klasyfikują większość próbek, a błędne decyzje występują sporadycznie. Rozkład punktów wskazuje na wyraźny podział przestrzeni cech pomiędzy banknoty prawdziwe i fałszywe. Różnice pomiędzy wynikami klasycznego i rozmytego kNN są minimalne, co potwierdza, że w przypadku danych binarnych o dobrej separacji klas zastosowanie rozmycia nie wpływa istotnie na końcową skuteczność klasyfikacji.

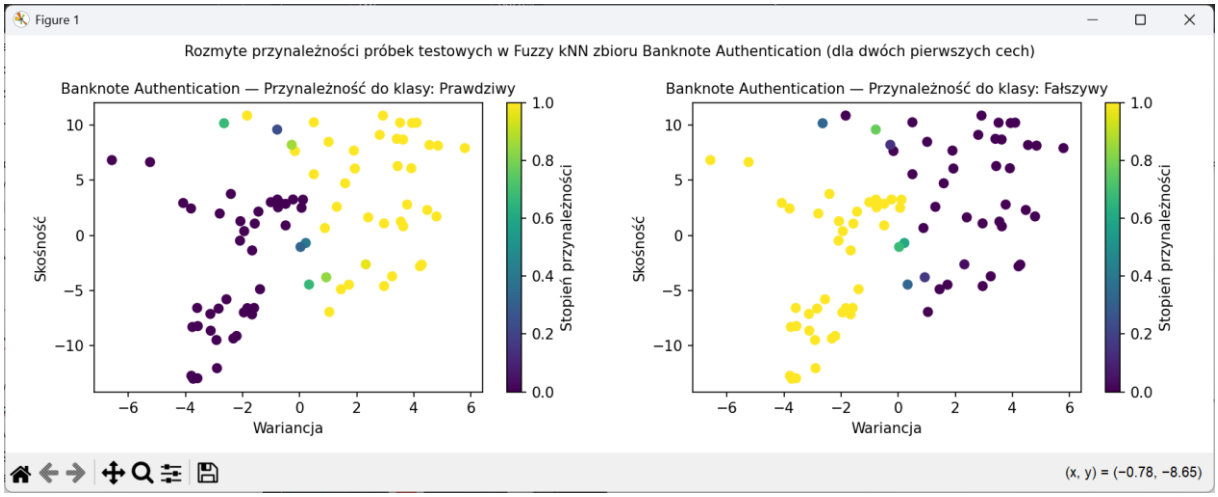


6.3. Rozmyte przynależności próbek

Wizualizacja rozmytych przynależności próbek testowych zbioru Seeds do klas Kama, Rosa oraz Canadian przedstawia stopnie przynależności obliczone przez algorytm Fuzzy kNN. Kolor punktów odzwierciedla wartość przynależności do danej klasy w zakresie od 0 do 1. Dla większości próbek obserwuje się wysokie wartości przynależności do jednej klasy, co oznacza dużą pewność decyzji klasyfikatora. Jednocześnie w obszarach granicznych pomiędzy klasami pojawiają się próbki o rozłożonych stopniach przynależności, co wskazuje na niejednoznaczność klasyfikacyjną, jednak tych próbek nie jest duża ilość. Tego typu informacja nie jest dostępna w klasycznym kNN i stanowi istotną zaletę podejścia rozmytego, szczególnie w analizie danych o nieostrych granicach decyzyjnych.



Ostatnia wizualizacja przedstawia rozmyte przynależności próbek testowych do klas Prawdziwy oraz Fałszywy dla zbioru Banknote Authentication. W większości przypadków wartości przynależności są bliskie 0 lub 1, co potwierdza wysoką pewność decyzji klasyfikatora. Jedynie nieliczne próbki charakteryzują się pośrednimi wartościami przynależności, co odpowiada przypadkom granicznym, trudniejszym do jednoznacznej klasyfikacji. Potwierdza to stabilność algorytmu Fuzzy kNN oraz jego zdolność do identyfikacji próbek o podwyższonej niepewności decyzyjnej.



7. Podsumowanie

Celem niniejszego projektu było porównanie skuteczności klasycznego algorytmu k-Nearest Neighbors oraz jego rozmytej wersji Fuzzy kNN w zadaniach klasyfikacji danych. Analizę przeprowadzono na dwóch zbiorach danych o odmiennej charakterystyce: wieloklasowym zbiorze Seeds oraz binarnym zbiorze Banknote Authentication.

Uzyskane wyniki wskazują, że w obu zbiorach danych klasyczny algorytm kNN oraz rozmyty Fuzzy kNN osiągają porównywalną skuteczność klasyfikacji. W przypadku zbioru Seeds kluczowy wpływ na jakość wyników miało skalowanie danych, natomiast samo zastosowanie rozmycia nie prowadziło do bezpośredniego wzrostu dokładności klasyfikacji. Algorytm Fuzzy kNN nie poprawił wartości accuracy względem klasycznego kNN, jednak umożliwił dodatkową analizę stopni przynależności próbek do klas, co zwiększa interpretowalność wyników, szczególnie w obszarach granicznych przestrzeni cech.

Dla zbioru Banknote Authentication oba algorytmy osiągnęły bardzo wysoką skuteczność, co wynika z dobrej separacji klas w danych. Różnice pomiędzy klasycznym kNN a rozmytym Fuzzy kNN były niewielkie, jednak rozmyta wersja algorytmu wykazywała większą stabilność

decyzyjną w wybranych konfiguracjach parametrów, co potwierdziła analiza macierzy pomyłek oraz stopni przynależności.

Przeprowadzona analiza ilościowa oraz wizualna potwierdza, że Fuzzy kNN stanowi wartościowe rozszerzenie klasycznego algorytmu kNN, szczególnie w kontekście interpretacji wyników i identyfikacji próbek o podwyższonej niepewności decyzyjnej. Choć nie zawsze prowadzi on do wzrostu dokładności klasyfikacji, dostarcza dodatkowych informacji niedostępnych w klasycznym podejściu ostrym.

Podsumowując, zrealizowany projekt spełnia założone cele badawcze i potwierdza, że wybór pomiędzy klasycznym kNN a Fuzzy kNN powinien być uzależniony od charakteru danych oraz wymagań dotyczących interpretowalności wyników. Praca stanowi również solidną podstawę do dalszych badań, obejmujących analizę innych zbiorów danych, alternatywnych metryk odległości oraz rozszerzenie eksperymentów o kolejne algorytmy klasyfikacji rozmytej.