Obraz zawierający logo, tekst, symbol, Grafika

Opis wygenerowany automatycznie**Kolegium Nauk Przyrodniczych Uniwersytet Rzeszowski**

**Przedmiot:**

**Inteligentne metody eksploracji danych**

**Projekt 2**

**Wykonał:**

**Karol Dąbrowski, 125110**

**Kacper Bułaś, 127754**

**Prowadzący: dr inż. Piotr Grochowalski**

**Rzeszów 2024**

Spis treści

[Opis projektu 3](#_Toc187694422)

[Funkcje 3](#_Toc187694423)

[Użyte technologie 3](#_Toc187694424)

[Struktura projektu 4](#_Toc187694425)

[Format danych 4](#_Toc187694426)

[Instalacja 5](#_Toc187694427)

[Obsługa aplikacji 5](#_Toc187694428)

[Funkcje szczegółowe 9](#_Toc187694429)

[Własna implementacja 9](#_Toc187694430)

[Implementacja imputacji KNN 9](#_Toc187694431)

[Implementacja RandomForest 10](#_Toc187694432)

[Repozytorium zdalne 14](#_Toc187694433)

# Opis projektu

Projekt implementuje aplikację desktopową z graficznym interfejsem użytkownika (GUI) do zaawansowanego przetwarzania danych użytkowników. Wykorzystuje różne techniki przetwarzania danych, w tym imputację, normalizację, standaryzację oraz klasyfikację przy użyciu Random Forest.

# Funkcje

1. Interfejs użytkownika:

* Intuicyjny GUI oparty na tkinter
* Podgląd danych w czasie rzeczywistym
* Wizualizacja zmian w danych
* Eksport przetworzonych danych

1. Metody przetwarzania danych:

* Imputacja KNN (dwie implementacje)
* Normalizacja MinMax
* Standaryzacja
* Dyskretyzacja
* Klasyfikacja Random Forest

1. Analiza wyników:

* Statystyki zmian w danych
* Śledzenie modyfikacji decyzji
* Wizualizacja skuteczności klasyfikacji

1. Obsługa danych:

* Wczytywanie plików CSV
* Obsługa brakujących danych
* Automatyczne przetwarzanie potokowe

# Użyte technologie

* Python: język programowania
* tkinter: framework GUI
* pandas: manipulacja danymi
* scikit-learn: algorytmy uczenia maszynowego
* numpy: obliczenia numeryczne

# Struktura projektu

1. Interfejs użytkownika:

* Konfiguracja okna głównego
* Stylizacja komponentów
* Zarządzanie widokami danych

1. Przetwarzanie danych:

* Implementacja własnego algorytmu KNN
* Integracja z algorytmami scikit-learn
* Potok przetwarzania danych

1. Wizualizacja:

* Tabelaryczna prezentacja danych
* Oznaczanie zmienionych wartości
* Wyświetlanie statystyk

# Format danych

Aplikacja przetwarza dane użytkowników zawierające następujące kolumny:

* Wiek (18-65 lat)
* Płeć (0 - kobieta, 1 - mężczyzna)
* Lokalizacja (regiony A, B, C)
* Rodzaj usługi (podstawowa, premium, VIP)
* Czas subskrypcji (miesiące)
* Czas na platformie (godziny)
* Liczba reklamacji
* Średnia transakcji (miesięcznie)
* Średnie wydatki
* Zaległe płatności
* Rabaty (%)
* Porzucenie (0 - pozostał, 1 - zrezygnował)

# Instalacja

Wymagania:

* Python 3.x
* Biblioteki: numpy, pandas, scikit-learn, tkinter

Instalacja zależności:

pip install numpy pandas scikit-learn

# Obsługa aplikacji

1. Uruchomienie:

* Uruchom skrypt główny
* Po uruchomieniu skryptu wyświetli się okno aplikacji
* Następnie należy wybrać plik csv z przygotowanymi danymi

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie

* Załadowane dane wyświetlą się w określonym polu

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie

1. Przetwarzanie:

* Wybierz metodę imputacji KNN

Obraz zawierający tekst, zrzut ekranu, Czcionka, Prostokąt

Opis wygenerowany automatycznie

* Podczas wykonywania kroków przetwarzania danych tabela poniżej będzie uzupełniać się na bieżąco pokazując zmiany

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Prostokąt

Opis wygenerowany automatycznie

* Istnieje też możliwość wykorzystania przycisku „Uruchom potok” który automatyzuje wykonywanie kroków

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Prostokąt

Opis wygenerowany automatycznie

1. Analiza wyników:

* Przeglądaj dane w tabelach
* Po ukończeniu potoku przetwarzania istnieje możliwość wyświetlenia statystyk dokonanych zmian

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

* Możemy też eksportować przetworzone dane do pliku csv

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

# Funkcje szczegółowe

1. Imputacja KNN:

* Implementacja własna lub scikit-learn
* Konfigurowalny parametr k
* Obsługa różnych metryk odległości

1. Normalizacja i standaryzacja:

* Skalowanie MinMax do zakresu [0,1]
* Standaryzacja do rozkładu N(0,1)

1. Dyskretyzacja:

* Podział na 4 przedziały
* Strategia równomierna

1. Klasyfikacja Random Forest:

* 100 drzew decyzyjnych
* Podział danych 80/20
* Ocena dokładności klasyfikacji

# Własna implementacja

### Implementacja imputacji KNN

**Opis działania**: Funkcja custom\_knn\_imputer umożliwia imputację brakujących danych w DataFrame na podstawie algorytmu K-Nearest Neighbors (KNN). Wykorzystuje zadaną liczbę sąsiadów (parametr k) oraz pozwala na wybór strategii ważenia (uniform lub distance). Obsługuje metrykę euklidesową jako miarę odległości.

**Kluczowe kroki**:

1. Identyfikacja brakujących wartości w danych.
2. Obliczanie odległości między wierszami na podstawie dostępnych cech.
3. Sortowanie sąsiadów względem odległości i wybór k najbliższych.
4. Uzupełnianie brakujących wartości za pomocą średniej lub ważonej średniej wartości sąsiadów.

**Ograniczenia**:

* Obsługuje jedynie metrykę euklidesową.
* Brak wsparcia dla wielowątkowości lub dużych zbiorów danych.

Obraz zawierający tekst, zrzut ekranu, Czcionka, menu

Opis wygenerowany automatycznie

### Implementacja RandomForest

**Opis działania**: Klasa CustomRandomForest implementuje uproszczony algorytm Random Forest. Składa się z wielu drzew decyzyjnych, które są trenowane na bootstrapowanych podzbiorach danych, a ostateczna klasyfikacja jest wynikiem głosowania większościowego.

**Kluczowe elementy**:

1. **Klasa DecisionTree**:
   * Obsługuje rekurencyjne budowanie drzewa decyzyjnego na podstawie współczynnika Giniego.
   * Zawiera funkcje do znajdowania najlepszego podziału, oceny współczynnika Giniego i podziału danych.
2. **Las losowy**:
   * Wykorzystuje bootstrap sampling do losowego wyboru próbek dla każdego drzewa.
   * Umożliwia kontrolę liczby drzew (n\_estimators), maksymalnej głębokości (max\_depth) oraz minimalnej liczby próbek wymaganych do podziału węzła (min\_samples\_split).
   * Wprowadza losowość w wyborze cech, co zwiększa różnorodność drzew.
3. **Głosowanie większościowe**:
   * Każde drzewo generuje predykcję, a wynik ostateczny dla każdej próbki jest wybierany na podstawie najczęściej występującej klasy.

**Ograniczenia**:

* Brak obsługi ciągłych wartości docelowych (tylko klasyfikacja).
* Brak optymalizacji wydajności (np. brak wielowątkowości czy wsparcia GPU).

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

# Repozytorium zdalne

Pliki projektu wraz z danymi znajdują się pod [linkiem](https://github.com/Poicalee/IMDB2)