Projekt - dokumentacja wstępna

Uczenie Maszynowe (UMA)

Maciej Lipski, Karol Żelazowski

14 lutego 2025

Spis treści

1. Opis projektu

1.1. Temat Projektu

Zadanie projektowe miało następującą treść: *Zaimplementować inkrementacyjną indukcję reguł. Przebudowa zbioru reguł na podstawie sekwencyjnie nadchodzących porcji danych lub pojedynczych przykładów.*

1.2. Interpretacja zadania

Inkrementacyjna indukcja reguł została zinterpretowana jako proces dynamicznej modyfikacji zbioru reguł opartych na danych przychodzących sekwencyjnie. Jest szczególnie użyteczna w sytuacjach, gdy dane są dostarczane stopniowo, a nie w całości, jak w przypadku systemów strumieniowych lub uczenia maszynowego w czasie rzeczywistym. W związku z tym, jako zadanie postawiono sobie dostosowanie klasycznych algorytmów indukcji reguł, o wsadowym trybie przetwarzania danych, do budowania zbioru reguł na podstawie danych nadchodzących sekwencyjnie.

2. Ogólny opis algorytmu

W ramach realizacji zadania dostosowano klasyczny algorytm specjalizacji AQ do realizacji indukcji reguł w formie inkrementacyjnej. W klasycznej wersji algorytm ten nie jest dostosowany do formy inkrementacyjnej, a do przetwarzania statycznej listy przykładów i budowania na ich podstawie zbioru reguł poprzez sekwencyjne pokrywanie. W ramach projektu, po stosunkowo niewielkich zmianach dostosowano algorytm do wersji inkrementacyjnej.

2.1. Klasyczna realizacja algorytmu

W klasycznej wersji algorytmu specjalizacji AQ, na początku tworzony jest zbiór kompleksów G, inicjalizowany jako zbiór pusty. Następnie z zestawu danych R wybierany jest niepokrywany przykład x_s, który będzie stanowił punkt odniesienia dla dalszego działania algorytmu.

Dane są dzielone na dwa zbiory: $R^{(1)}$, zawierający przykłady o klasyfikacji zgodniej z $c(x_s)$ (pozytywne) oraz $R^{(0)}$, obejmujący przykłady o klasyfikacji różnej od $c(x_s)$ (negatywne). Zbiór G_s zawierający reguły pokrywające przykład x_s i niepokrywające analizowanych przykładów negatywnych inicjalizuje się poprzez najbardziej ogólny kompleks $<?>$. Algorytm iteracyjnie dokonuje specjalizacji kompleksów w G_s, dopóki zbiór $R_G^{(0)}$, zawierający przykłady negatywne pokrywane przez wygenerowane kompleksy, nie zostanie opróżniony. W każdej iteracji wybierany jest przykład x_n z $R_G^{(0)}$. Następnie każdy kompleks $k \in G$ jest specjalizowany poprzez dopasowanie go do niepokrywania przykładu x_n, przy jednoczesnym zachowaniu zgodności z x_s. Tak powstałe specjalizacje dodawane są do zbioru G_s, a kompleksy mniej ogólne od innych (czyli te zdominowane) są z niego usuwane.

Po dokonaniu specjalizacji algorytm ocenia jakość kompleksów w G za pomocą funkcji $v_{R^{(1)},R^{(0)}}(k)$, która uwzględnia liczbę przykładów ze zbioru trenującego pokrywanych przez kompleks o kategorii zgodnej z kategorią ziarna, liczbę przykładów klasy niezgodnej z klasą ziarna niepokrytą przez regułę oraz liczbę przykładów pokrywanych wyłącznie przez nowy kompleks. Spośród wszystkich kompleksów w G_s po specjalizacji wybierane są najlepsze m kompleksów lub pojedynczy najlepszy kompleks.

Proces ten powtarza się, aż zbiór $R^{(0)}$ zostanie opróżniony. W efekcie algorytm zwraca kompleks o najwyższej jakości, który najlepiej spełnia kryteria pokrycia ziarna. Następnie algorytm podejmuje to samo zadanie dla kolejnych ziaren, niepokrytych przez obecne reguły.

Algorytm dąży do uzyskania reguł, które efektywnie pokrywają przykłady należące do danej klasy, jednocześnie minimalizując pokrywanie przykładów z innych klas. Dzięki temu uzyskiwany jest uporządkowany zbiór reguł, który spełnia założenia poprawności i specyficzności.

2.2. Modyfikacja algorytmu na potrzeby inkrementacyjności

W ramach realizacji tego algorytmu w wersji inkrementacyjnej wykorzystano podstawy algorytmu AQ i zbudowano algorytm realizujący inkrementacyjną indukcję reguł, w opisanych dalszych podsekcjach krokach. Wersja inkrementacyjna algorytmu działa w taki sposób, że reguły są aktualizowane krok po kroku, dla każdego nowego przykładu, bez konieczności ponownego uruchamiania algorytmu dla całego zbioru danych. Algorytm jest jednak inicjalizowany w sposób wsadowy, z wykorzystaniem pierwszego zbioru trenującego, aby uniknąć tworzenia bardzo ogólnych reguł.

2.2.1. Przyjęcie nowego przykładu

Na podstawie wsadowego algorytmu AQ, uznano nowy, nadchodzący przykład x_s za ziarno do klasyfikacji. Warto zaznaczyć, że dane przyjmujemy w zbiorach, aby móc dokonać wsadowej inicjalizacji algorytmu oraz ze względu na złożoność obliczeniową przyjmowania pojedynczego przykładu. Po przyjęciu przykładu, jest on dodawany do zbioru przetworzonych przykładów (dalej oznaczany jako zbiór P). Zbiór ten jest przechowywany, w celu posiadania ziaren negatywnych dla tworzenia nowych reguł. Jeśli nadchodzący przykład jest poprawnie klasyfikowany przez obecne już reguły, to zbiór reguł nie wymaga zmian. Jeśli przykład nie jest pokrywany, należy utworzyć regułę zgodnie z opisem w sekcji 2.2.2.

Problemem pozostają przykłady, dla których istnieją reguły klasyfikujące je niepoprawnie (zarówno gdy są klasyfikowane jedynie błędnie, jak i gdy są klasyfikowane do więcej niż jednej klasy). Jeśli w nadchodzącym zbiorze przykładów znajduje się taki przykład, należy wykonać działania opisane w sekcji 2.2.4, które sprowadzają się do usuwania jedynie reguł o dużym błędzie.

2.2.2. Tworzenie nowej reguły

Przy tworzeniu nowej reguły, przy każdym rozważanym niepokrytym przykładzie, zbiór $R^{(1)}$ inicjalizuje się zbiorem przykładów o $c = c(x_s)$, a zbiór $R^{(0)}$ przykładami o $c \neq c(x_s)$. W zbiorach tych uwzględniamy także wszystkie przykłady ze zbioru nadchodzących, aktualnie przetwarzanych przykładów, w celu jak najlepszego pokrycia przez nową regułę całego nowego zbioru.

Zbiór rozważanych kompleksów G_s inicjalizuje się kompleksem $<?>$. Następnie dokonuje się jego specjalizacji poprzez iteracyjne wykluczanie pokrywanie kompleksów z $R^{(0)}$ przy zachowaniu pokrywania przykładu x_s. Po każdej iteracji specjalizacji, w naszej wersji algorytmu zostawiamy dwa najlepsze kompleksy, według miar oceny jakości (opisane w sekcji 2.2.5). Usuwamy także kompleksy bardziej szczegółowe od innych w G_s. Jeśli więcej niż dwa kompleksy mają taką samą ocenę, wybieramy dwa kompleksy zgodnie z ustalonym porządkiem, np. porządkiem leksykograficznym.

Taki proces powtarzamy do momentu wykluczenia pokrywania przez kompleksy w G_s wszystkich przykładów z R_0. Następnie zwracamy kompleks o najwyższej dokładności jakości (według opisu z sekcji 2.2.5). W przypadku równej jakości zwracamy regułę zgodnie z ustalonym porządkiem, np. porządkiem leksykograficznym.

2.2.3. Aktualizacja zbioru reguł i przycinanie

Zwracanym kompleksem aktualizujemy zbiór reguł. W tym miejscu dokonujemy także przycinania zbioru reguł, aby uniknąć nadmiernego dopasowania. Wstępnie uznano, że zostawiane będą minimum 3 najlepsze, pod względem jakości opisanej w sekcji 2.2.5, reguły klasyfikujące do danej klasy lub więcej reguł, jeśli ich jakość będzie wysoka. Słuszność założeń zostanie potwierdzona eksperymentalnie. Ze zbioru reguł usuwamy także reguły o dużym wskaźniku błędów, oceniającym jaki odsetek przykładów jest niepoprawnie klasyfikowany przez reguły w zbiorze reguł.

2.2.4. Problem reguł niepoprawnie klasyfikujących

Po przyjęciu nowego zbioru dokonujemy sprawdzenia, czy są w nim przykłady niepoprawnie klasyfikowane przez reguły w zbiorze reguł. Jeśli znajdują się takie przykłady, dokonujemy obliczenia

wskaźnika błędów dla reguł, oceniającego jaki odsetek przykładów jest przez nie niepoprawnie klasyfikowany, uwzględniając także obecnie przyjmowane przykłady. Jeśli wskaźnik błędu dla danej reguły będzie miał zbyt wysoką wartość (przykładowo 10%, wartość ta zostanie ustalona w trakcie eksperymentów), regułę tą usuwamy ze zbioru reguł, a dla przykładów, które w wyniku usunięcia reguły zostaną niepokryte, generujemy nowe reguły zgodnie z opisem tworzenia nowych reguł (sekcja 2.2.2). W wypadku, gdy odsetek błędów danej reguły jest niewielki, godzimy się na to, by przyjmowany przykład był błędnie klasyfikowany.

2.2.5. Ocena jakości pokrywania

Ze względu na specyfikę algorytmu AQ, jako miarę oceny reguł przy specjalizacji i przycinaniu zbioru reguł, wybrano sumę liczby przykładów ze zbioru trenującego pokrywanych przez kompleks o klasie zgodnej z klasą ziarna (lub przy przycinaniu z klasą reguły), liczby przykładów klasy niezgodnej z klasą ziarna (lub przy przycinaniu z klasą reguły) niepokrytą przez regułę oraz liczby przykładów pokrywanych wyłącznie przez kompleks. Definiuje to następujący wzór:

$$v_{k(x_s,P)} = |x \in P_k \mid c(x) = c(x_s)| + |x \in P - P_k \mid c(x) \neq c(x_s)| + |x \in P_k \mid c(x_s) = c(x_s)|$$

$|x \in P_k \mid c(x) = c(x_s)|$ – liczba przykładów ze zbioru trenującego pokrywanych przez kompleks o klasie zgodnej z klasą ziarna (lub przy przycinaniu z klasą reguły)

$|x \in P - P_k \mid c(x) \neq c(r)|$ – liczba przykładów klasy niezgodnej z klasą ziarna (lub przy przycinaniu z klasą reguły) niepokrytych przez regułę

$|x \in P_k \mid c(x) = c(r)|$ – liczba przykładów pokrywanych wyłącznie przez kompleks/ regułę

W wypadku, gdy obliczamy jakość na potrzeby klasyfikacji, nie uwzględniamy liczby przykładów pokrywanych wyłącznie przez daną regułę. W takich wypadkach jakość reguły obliczamy ze wzoru:

$$v_{k(r,P)} = |x \in T_k \mid c(x) = c(x_s)| + |x \in T - T_k \mid c(x) \neq c(x_s)|$$

$|x \in T_k \mid c(x) = c(r)|$ – liczba przykładów ze zbioru trenującego poprawnie pokrywanych przez regułę

$|x \in T - T_k \mid c(x) \neq c(r)|$ – liczba przykładów poprawnie niepokrywanych przez regułę

2.2.6. Klasyfikacja

Klasyfikacja przykładów następuje na podstawie aktualnego zbioru reguł. W wypadku, gdy nie można zdecydować do jakiej klasy przyporządkować przykład - gdy nie jest on pokrywany przez żadną regułę lub gdy jest pokrywany przez reguły klasyfikujące do różnych klas, rozstrzyga zmodyfikowana ocena jakości pokrywania, opisana w sekcji 2.2.5. Jak opisano w sekcji 2.2.5, przy klasyfikacji rozstrzyga suma liczby przykładów ze zbioru trenującego poprawnie klasyfikowanych przez regułę oraz liczby przykładów klasy niezgodnej z klasą reguły przez nią niepokrytych, bez uwzględniania liczby przykładów pokrywanych wyłącznie przez konkretną regułę. Dla klasyfikacji nieuwzględniony element oceny jakości nie jest istotny.

2.3. Realizacja algorytmu w pseudokodzie

W celu ilustracji działania zaprojektowanego algorytmu, stworzono jego realizację w pseudokodzie. Pseudokod ten zamieszczono poniżej. Skupia się on przede wszystkim na ilustracji działania algorytmu krok po kroku.

```
// Krok 1: Inicjalizacja wsadowa
INPUT: Pierwszy zbiór przykładów T, zbiór przetworzonych przykładów P = ⌀, zbiór reguł
G = ⌀

FUNCTION inicjalizacja_wsadowa(T):
    FOR każdy przykład x IN T:
      P = P + x // Dodanie przykładów do zbioru przetworzonych, w celu uwzględnienia
ich w ocenie jakości
    FOR każdy przykład x IN T:
        IF x nie jest pokrywany przez istniejące reguły w G:
            nowa_reguła = utwórz_nową_regułę(x, P)
            G = G + nowa_reguła
    RETURN G, P

// Krok 2: Iteracyjnie przyjmowanie nowych zbiorów przykładów trenujących
FUNCTION przyjmowanie_przykładów(nowe_przykłady, P, G):
    P = P + nowe_przykłady
    FOR każdy przykład x_s IN nowe_przykłady:
        // Obsługa reguł klasyfikujących x_s niepoprawnie
        IF x_s jest niepoprawnie pokrywany przez reguły w G:
                R = obsłuż_niepoprawną_klasyfikację(x_s, G, P)
        IF x_s jest pokrywany poprawnie przez reguły w G:
                CONTINUE // Nie wymaga zmian
        // Nowy przykład nie jest pokrywany
        ELSE IF: x_s nie jest pokrywany przez żadną regułę:
                nowa_reguła = utwórz_nową_regułę(x_s, P)
                G = aktualizuj_zbiór_reguł(nowa_reguła, G)
    RETURN G, P

//Tworzenie nowych reguł
FUNCTION utwórz_nową_regułę(x_s, P):
    R1 = {x | x ∈ P AND c(x) = c(x_s)} // Przykłady o klasie zgodnej z klasą ziarna
    R0 = {x | x ∈ P AND c(x) ≠ c(x_s)} // Przykłady o klasie niezgodnej z klasą ziarn
    G_s = {<?>} // Inicjalizacja kompleksu jako pustego

    WHILE istnieją przykłady w R0 pokrywane przez G_s:
        G_s = specjalizuj(G_s, x_s, R0) // Wyklucz pokrycie negatywnych przykładów
         G_s = wybierz_najlepsze_kompleksy(G_s, P , n = 2) // Zostaw dwa najlepsze
kompleksy

    // Zwracamy kompleks o najwyższej jakości
    RETURN kompleks_o_najwyzszej_jakosci(G_s, P, n = 1)

// Krok 4: Aktualizacja zbioru reguł
FUNCTION aktualizuj_zbiór_reguł(nowa_reguła, G):
    G = G + nowa_reguła
```

```
    G = przytnij_reguły(G)
    RETURN G

// Krok 5: Przycinanie reguł
FUNCTION przytnij_reguły(G):
    IF |G| > 3: // Minimalna liczba reguł
        G_dokładne = wybierz_najlepsze_reguły(G,P, n = 3)
    RETURN G_dokładne

//Obsługa reguł błędnie klasyfikujących
FUNCTION obsłuż_niepoprawną_klasyfikację(x_s, G, P):
    FOR każda reguła r IN G:
        IF wskaźnik_błędów(r) > 10%: //Wartość 10% wstępna, będąca obiektem
        eksperymentów
            G = G - r // Usuń regułę o dużym błędzie
            P_niepokryte = {x | x ∈ P AND x niepokrywany przez żadną z reguł w G}
            FOR każdy x IN P_niepokryte:
                nowa_reguła = utwórz_nową_regułę(x, P)
                G = G + nowa_reguła
    RETURN G

// Krok 7: Klasyfikacja przykładów
FUNCTION klasyfikuj_przykład(x, G):
    pokrywające_reguły = {r ∈ G | x jest pokrywany przez r}
    //Gdy żadna z reguł nie pokrywa przykładu, wybieramy regułę o najwyższej jakości
ze zbioru reguł
    IF |pokrywające_reguły| = 0:
        RETURN wybierz_najlepszą_regułę(G,P)
    ELSE:
        klasyfikacje = {c(r) | r ∈ pokrywające_reguły}
        IF |klasyfikacje| = 1:
            RETURN c(r)
        ELSE:
            najlepsza_reguła = wybierz_najlepszą_regułę(pokrywające_reguły, P, n = 1,
czy_klasyfikacja = TRUE)
            RETURN c(najlepsza_reguła)

// Wybór najlepszej reguły
FUNCTION wybierz_najlepszą_regułę(reguły, P):
    najlepsza_reguła = NULL
    najlepsza_ocena = -∞
    FOR każda reguła r IN pokrywające_reguły:
        ocena = licz_ocenę_jakości(r, P)
        IF ocena > najlepsza_ocena:
            najlepsza_ocena = ocena
            najlepsza_reguła = r
```

```
    RETURN najlepsza_reguła

// Funkcja liczenia oceny jakości reguły
// Zmienna czy_klasyfikacja to bool oznaczający czy stosujemy ocenę jakości przy klasyfikacji, domyślnie FALSE
FUNCTION licz_ocenę_jakości(reguła, P, czy_klasyfikacja = FALSE):
    // Inicjalizacja zmiennych
    pokryte_poprawne = 0 // Liczba przykładów z P pokrywanych poprawnie
    niepokryte_błędne = 0 // Liczba przykładów z P klasy niezgodnej z klasą reguły przez nią niepokrywanych
    wyłącznie_pokryte = 0 // Liczba przykładów z P pokrywanych wyłącznie przez regułę

    FOR każdy przykład x IN P:
        IF x jest pokrywany przez regułę:
            IF c(x) = c(reguła): // Klasa przykładu zgodna z klasą reguły
                pokryte_poprawne += 1
        ELSE:
            IF c(x) ≠ c(reguła): // Klasa przykładu niezgodna z klasą reguły
                niepokryte_błędne += 1
        //Nieuwzględnienie wyłącznego pokrywania przez regułę, gdy liczmy jakość przy klasyfikacji
        IF !czy_klasyfikacja:
            IF x jest pokrywany wyłącznie przez regułę:
                wylacznie_pokryte += 1

    // Suma wszystkich składników oceny jakości
    ocena = pokryte_poprawne + niepokryte_błędne + tylko_nowe_pokryte
    RETURN ocena
```

3. Plan eksperymentów

Aby ocenić jakość powstałych modelów wytwarzanych przez nasz algorytm planujemy zrealizować serię eksperymentów. Pierwszy z eksperymentów zakłada podział zbioru danych w proporcjach 70% na dane trenujące oraz 30% na dane testowe. W taki sposób będziemy mogli ocenić jakoś powstałego modelu za pomocą posiadanych danych. W ten sposób będziemy chcieli przetestować:

- Inkrementalne uczenie się - sprawdzenie, czy algorytm poprawnie aktualizuje zbiór reguł. Miarą jakości takiego eksperymentu może być stabilność reguł bądź odsetek przykładów poprawnie pokrywanych przez regułę.
- Obsługa konfliktów reguł i błędnej kwalifikacji - przetestowanie poprawności i zasadności wybierania i usuwania reguł w przypadku konfliktu lub błędnej kwalifikacji.
- Test przycinania reguł - przetestowanie sposobu przycinania reguł w przypadku wyprodukowania za dużej ilości reguł.

Po ogólnym przetestowaniu naszych modeli danych będziemy mogli je ocenić poprzez dane testowe. Miarami jakości będą:

- Dokładność - Procent poprawnych klasyfikacji na zbiorze testowym
- F1-score - Harmoniczna średnia precyzji i czułości. Dla zbiorów z klasami niebinarnymi zastosujemy Macro-averaging, polegający na policzeniu precyzji, czułość i F1-score uwzględniając osobno każdą

klasę jako pozytywną a resztę jako negatywną, a następnie wyciągnięciu średniej arytmetycznej. Definiują to następujące wzory:

$$\text{Macro-Precyzja} = \frac{1}{K}\sum_{i=1}^{k} \text{Precyzja}_i; K - \text{liczba klas}; \text{Precyzja}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}$$

$$\text{Macro-Czułość} = \frac{1}{K}\sum_{i=1}^{k} \text{Czułość}_i; K - \text{liczba klas}; \text{Precyzja}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}$$

$$\text{Macro-F1} = \frac{1}{K}\sum_{i=1}^{k} \text{F1}_i; \text{F1}_i = 2 * \frac{\text{Precyzja}_i * \text{Czułość}_i}{\text{Precyzja}_i + \text{Czułość}_i}$$

- Wskaźnik Laplaca: $\frac{n_c+1}{n_t+k}$, gdzie n_c - to liczba przykładów pokrywanych przez regułę, n_c - liczba poprawnie pokrywanych, k - liczba klas

W przypadku sprawdzenia nadmiernego dopasowania podzielimy zestaw danych na trzy zbiory. Zbiór treningowy, który będzie zawierał 70% danych, będzie on służył do wytrenowania modelu. Dalej zbiór walidacyjny składający się z 15% danych, dzięki niemu będziemy mogli monitorować proces uczenia się i ocenić czy algorytm zaczyna się przeuczać. Na dostaniemy zbiór testowego, w którego skład będzie wchodziło 15% danych początkowych. Będzie on służyć do ostatecznej oceny jakości naszego modelu na danych, które nie były używane ani do treningu, ani do walidacji. Następnie w procesie monitorowania będziemy mierzyć miary jakości na każdym zestawie. W zbiorze treningowym oczekujemy, że jakość klasyfikacji będzie rosła. Na zbiorze walidacyjnym, gdy algorytm zacznie się przeuczać, jakość zacznie spadać mimo poprawy wyników na zbiorze treningowym. Do ustalenia przeuczania się algorytmu użyjemy krzywej uczenia się, która będzie mierzyła dokładność na zbiorze treningowym i walidacyjnym co kolejną porcję podanych danych.

W ramach eksperymentów zostanie wykonane także porównanie naszego algorytmu do klasycznych, wsadowych algorytmów indukcji reguł. Jednym z nich będzie wsadowa implementacja algorytmu CN2 w bibliotece Orange Data Mining ([1]). Porównanie do innego algorytmu indukcji reguł ma uzasadnienie, ponieważ nie znaleziono implementacji algorytmu AQ w popularnych bibliotekach. Jedyną znalezioną implementacją AQ w języku Python, był projekt znaleziony na Githubie ([2]), do którego także nasz inkrementacyjny algorytm zostanie porównany.

4. Zbiory dane używane do eksperymentów

Aby sprawdzić poprawność działania naszego rozwiązania musieliśmy zebrać dane, które posłużą nam do eksperymentów. W zebranych zbiorach danych dzielimy je w losowy sposób na dane trenujące i dane sprawdzające poprawność. W przypadku natrafienia na puste pole w danych będziemy traktować je jako wartość średnią.

4.1. Zbiór danych „Iris”

Zbiór danych dotyczący klasyfikacji irysów ([3]). Zawiera on dane na temat stu pięćdziesięciu kwiatów, które są klasyfikowane na trzy klasy: „Iris Setosa”, „Iris Versicolour”, „Iris Virginica”. Każdy z kwiatów jest opisany za pomocą czterech cech:

- Sepal length - Długość działek kielicha kwiatu irysa (zielonej struktury przypominającej liść, która otacza pąk kwiatowy). Ciągła wartość podana w centymetrach.
- Sepal width - Szerokość działek kielicha kwiatu irysa. Ciągła wartość podana w centymetrach
- Petal length - Długość płatków kwiatu irysa (kolorowa struktura kwiatu). Ciągła wartość podana w centymetrach.
- Petal width - Szerokość płatków kwiatu irysa. Ciągła wartość podana w centymetrach.

4.2. Chemiczna analiza wina

Dane te są wynikami analizy chemicznej win uprawianych w tym samym regionie Włoch, ale pochodzących z trzech różnych odmian ([4]). W wyniku analizy określono różne ilości trzynastu składników występujących w każdym z trzech rodzajów win.

- Klasa wina: odmiana pierwsza, druga, trzecia
- Alcohol - Zawartość alkoholu.
- Malic acid - Zawartość kwasu jabłkowego.
- Ash - Zawartość wszystkich produktów powstałych po odparowaniu wina.
- Alcalitnity of ash - Suma kationów innych niż jony amonowe połączone z kwasami organicznymi w winie
- Magnesium - zawartość magnezu w winie
- Total phenols - Zawartość całkowita polifenoli i związków fenolowych w winie
- Flavanoids - Zawartość flawanoidów w winie
- Nonflavanoid phenols - Zawartość fenoli, które nie są flawanoidami
- Proanthocyanins - Zawartość proantocyjanidynów w winie
- Color intensity - Intensywność koloru wina
- Hue - Odcień koloru wina
- OD280/OD315 of diluted wines - Absorbancja OD280/OD315 określająca zawartość białka w winie
- Proline - Zawartość proliny w winie

4.3. Dane dotyczące raka piersi

Zbiór danych pochodzących z badań Dr. Williama H. Wolberg z szpitala uniwersyteckiego Wisconsin [5]. Każda instancja danych klasyfikowany jest jako nowotwór łagodny bądź złośliwy. Dodatkowo do opisania każdego przypadku użyte jest dziewięć cech zawierających wartości od jeden do dziesięć

- Clump Thickness - Grubość skupisk komórek. Ocena wielkości i grubości skupisk komórkowych.
- Uniformity of Cell Size - Jednorodność rozmiaru komórek. Ocena, czy komórki mają jednolitą wielkość.
- Uniformity of Cell Shape - Jednorodność kształtu komórek. Analiza podobieństwa kształtów komórek.
- Marginal Adhesion - Przyleganie komórek na brzegach. Sprawdzanie, jak mocno komórki przylegają do siebie.
- Single Epithelial Cell Size - Rozmiar pojedynczych komórek nabłonkowych. Pomiar wielkości indywidualnych komórek nabłonkowych.
- Bare Nuclei - Nagie jądra komórkowe. Ocena obecności jąder niezawierających cytoplazmy.
- Bland Chromatin - Chromatyna o łagodnym wyglądzie. Sprawdzanie struktury chromatyny.
- Normal Nucleoli - Obecność jąderka. Ocena obecności i wielkości jąderka w komórkach.
- Mitoses - Podziały komórkowe. Liczba podziałów komórkowych.

4.4. Zbiór danych dotyczących gry w golfa

Zbiór danych nie opierający się na atrybutach numerycznych, który zbiera dane na temat możliwości gry golfa w różnych warunkach pogodowych ([6]). Jest to mały zbiór użyty będzie on przez nas do ocenienia inkrementalnego uczenia się na małych zbiorach danych. Cechy danej prognozy:

- Outlook: Rainy, Sunny, Overcast
- Temperature: Hot, Cold, Mild
- Humidity: High, Normal
- Windy: True, False

Bibliografia

[1] Demsar J, „Orange: Data Mining Toolbox in Python”. [Online]. Dostępne na: https://www.varonis.com/blog/arp-poisoning/

[2] Patrick Canny, „Implementation of the AQ (Max Star) Data Mining Algorithm”. [Online]. Dostępne na: https://github.com/patrickcanny/AQ

[3] Fisher R, „Iris [Dataset]n”. [Online]. Dostępne na: https://doi.org/10.24432/C56C76.

[4] Cortez, „Wine Quality”. [Online]. Dostępne na: https://doi.org/10.24432/C56S3T

[5] Wolberg WIlliam, „Breast Cancer Wisconsin (Original)”. [Online]. Dostępne na: https://doi.org/10.24432/C5HP4Z

[6] Aditya Rahman, „Golf weather dataset”. [Online]. Dostępne na: https://gist.github.com/kudaliar032/b8cf65d84b73903257ed603f6c1a2508

[7] Paweł Cichosz, *Systemy uczące się*. 2000.

[8] Paweł Cichosz, „Uczenie Maszynowe - Prezentacje wykładowe”. 2024.