Sztuczna Inteligencja – Projekt  
Sprawozdanie

M\* S\* 188\*, Z\* S\* 188\*, J\* T\* 188\*

1. Wstęp

Celem projektu była implementacja kilku algorytmów służących do budowy drzew decyzyjnych, porównanie ich między sobą oraz sprawdzenie jak radzą sobie z publicznie dostępnymi zbiorami danych służącymi do testowania i uczenia modeli SI rozwiązujących problemy klasyfikacji oraz regresji.

Wszystkie algorytmy zaimplementowane zostały w języku Python.

1. Opis problemu i metod
   1. Klasyfikacja

Problem klasyfikacji polega na przypisaniu obiektów składających się z wielu atrybutów (ciągłych bądź dyskretnych) do najlepiej opisujących je klas ustalonych wcześniej. Na przykład przypisaniu obiektu „Pacjenta” składającego się z atrybutów „Wiek”, „Czy jest palaczem?”, itd. do klasy „Tak” lub „Nie” odpowiadającej na pytanie czy pacjent ten dożyje 90 roku życia. Klasyfikacja gdzie istnieją tylko dwie klasy wynikowe jest nazywana binarną, ale oczywiście klas może być dużo więcej.

* + 1. Drzewa decyzyjne

Drzewo decyzyjne jest modelem, który potrafi udzielić odpowiedzi na problemy klasyfikacyjne. Składa się z węzłów oraz liści. Każdy węzeł rozdziela logikę na osobne ścieżki w zależności od wartości jednego z atrybutów. Liście natomiast zawierają informację o ostatecznej klasyfikacji obiektu do poszczególnej klasy. Drzewa decyzyjne można w łatwy sposób przedstawić w formie graficznej lub w postaci wyrażenia if/else.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Reprezentacja drzewa decyzyjnego odpowiadającego na problem klasyfikacji irysów w postaci wyrażenia if/else

* + 1. ID3

ID3 był jednym z pierwszych algorytmów służących do budowania drzew decyzyjnych. Algorytm ten zaczyna budowę drzewa od korzenia i rekurencyjnie wybiera najlepszy atrybut do utworzenia węzła i podzielenia danych treningowych. Jeśli w danym przebiegu wszystkie instancje danych treningowych należą do jednej klasy to w tym miejscu tworzony jest liść reprezentujący tą klasę.

ID3 wybiera najlepszy atrybut do podziału danych licząc „zysk informacyjny” (information gain) dla każdego z nich a następnie wybierając ten z największą wartością.

Algorytm jest podatny na wybieranie atrybutów o większej liczbie wartości, co może prowadzić do przetrenowania drzewa, ponadto w domyślnej specyfikacji nie obsługuje on atrybutów ciągłych.

* + 1. C4.5

Algorytm ten jest ulepszeniem ID3 wprowadzającym obsługę atrybutów o wartościach ciągłych; mechanizmów przycinania drzewa, aby ograniczyć przetrenowanie; lepszą obsługą brakujących danych oraz zmianą algorytmu, który decyduje o wyborze atrybutu do podziału.

Główną zmianą, której się przyjrzymy będzie właśnie nowy algorytm podziału – „współczynnik zysku informacyjnego”, który bierze pod uwagę ilość podzbiorów na które dany atrybut podzieli dane. Im mniej podzbiorów – tym większa wartość współczynnika.

* 1. Regresja

TODO

* 1. Las Losowy (Random Forest)

Random Forest jest algorytmem uczenia maszynowego, który wykorzystuje technikę zwaną "baggingiem" (bootstrap aggregating), aby zbudować zespół drzew decyzyjnych. Algorytm ten polega na tworzeniu wielu drzew decyzyjnych na podstawie losowych próbek danych treningowych, a następnie łączeniu ich w celu podjęcia ostatecznej decyzji.

Każde drzewo w lesie losowym jest trenowane na innym podzbiorze danych treningowych, pobranym losowo ze zwracaniem. Oznacza to, że jedna instancja danych może wystąpić wielokrotnie w różnych drzewach. Każde drzewo jest trenowane niezależnie, a ostateczna decyzja jest podejmowana na podstawie głosowania większościowego.

1. Opis realizacji
   1. ID3 oraz C4.5

Oba drzewa zostały zaimplementowane jako klasy, odpowiednio ID3Tree oraz C45Tree. Przyjęliśmy, że drzewo C4.5 od ID3 różni się jedynie algorytmem podziału, jak wyżej napisane, inaczej drzewa ID3 nie moglibyśmy przetrenować na wielu ze zbiorów danych, dlatego w naszej implementacji działa ono także z atrybutami ciągłymi. Interfejs obu drzew jest identyczny i prezentuje się następująco:

tree = C45Tree(X, Y, max\_depth, min\_samples\_split)

gdzie *X* – macierz wartości atrybutów danych treningowych,

*Y* – wektor wynikowych klas danych treningowych

*max\_depth* – hiperparametr określający jak głębokie może być drzewo

*min\_samples\_split* – hiperparametr określający przy ilu próbkach można podzielić węzeł drzewa

Oba hiperparametry posiadają domyślne wartości, odpowiednio ∞ i 1. Podanie własnych wartości może pomóc zwalczyć efekt przetrenowania drzewa, gdzie drzewo zbyt dokładnie dopasowuje się do danych treningowych przez co osiąga gorsze wyniki w ogólnym rozrachunku.

Po konstrukcji drzewa, aby użyć go do sklasyfikowania danej instancji testowej należy użyć metody tree.predict(x\_row) która jako argument przyjmuje wektor reprezentujący daną instancję i zwraca przewidzianą klasę.

* 1. Drzewo regresyjne

TODO

* 1. Las losowy (Random Forest)

Obiekt RandomForest jest tworzony z możliwością ustalenia wartości trzech hiperparametrów: num\_trees odpowiadający za ilość drzew w lesie losowym oraz max\_depth i min\_sample\_split, odpowiadające za maksymalną głębokość drzewa oraz minimalną liczbę próbek wymaganych do podziału węzła.

Budowa lasu losowego odbywa się w metodzie fit() i polega na tworzeniu drzew na podstawie próbek losowych, które zwraca metoda sample(). W metodzie tej wybieramy indeksy z jednakowym prawdopodobieństwem i możliwością powtarzania się, co sprawia, że losowo odrzucamy pewne zestawy cech i próbek. Po zbudowaniu każde drzewo dodawane jest do listy.

Przewidywanie w metodzie predict() polega na zebraniu przewidywań każdego drzewa przechowywanego w liście drzew oraz wybranie w głosowaniu etykiety, którą przewidziano najwięcej razy. Sprawdzenie, jaka etykieta przewidywana była najczęściej, realizuje metoda most\_common\_label().

1. Porównanie algorytmów na zbiorach danych

Zaimplementowane drzewa decyzyjne przetestowaliśmy na kilku zbiorach danych dostępnych na stronie <https://archive-beta.ics.uci.edu/>. Dla każdego z nich 80% danych przeznaczyliśmy do treningu, a pozostałe 20% do testowania. Hiperparametry zostały dobrane tak, aby test wykonywał się w sensownym czasie.

Do sporządzenia wykresów użyliśmy biblioteki *matplotlib*.

Obraz zawierający tekst, zrzut ekranu, diagram, Prostokąt

Opis wygenerowany automatycznieDry Beans:

Zbiór składający się z kilkuset instancji zawierających dane kilku gatunków fasoli opisanych za pomocą atrybutów ciągłych takich jak np. rozmiar. Zadaniem jest przewidzenie gatunku fasoli.

Obraz zawierający tekst, zrzut ekranu, diagram, Prostokąt

Opis wygenerowany automatycznie

Adult:

Zbiór w którym instancją jest człowiek, to gdzie mieszka, skąd pochodzi, wykształcenie, wiek, itd. . Zadaniem jest przewidzieć czy zarabia on >50k$ rocznie. Zbiór zawiera kilka atrybutów ciągłych i sporo dyskretnych o wielu możliwych wartościach.

Obraz zawierający tekst, zrzut ekranu, diagram, linia

Opis wygenerowany automatycznieIonosphere:

Zbiór reprezentujący dane z radarów obserwujących wolne elektrony w jonosferze. Zadaniem jest ocena czy dany sygnał jest „dobry” czy „zły”. Zbiór składa się z atrybutów ciągłych.