Paweł Świątkowski

Grzegorz Aleksiuk

**Podstawy sztucznej inteligencji – Projekt 2**

**MM.DD.R3 Random Forest – Forest Fires**

1. **Treść zadania**

Zaimplementować algorytm lasu losowego (ang. Random Forest) do predykcji. Zbiór danych do użycia: Forest Fires - https://archive.ics.uci.edu/ml/datasets/Forest+Fires. Uzyskane rezultaty porównać z wynikami dla wybranej implementacji algorytmu ML z dostępnych bibliotek np. Scikit-learn, WEKA, MLlib, Tensorflow/Keras etc.

1. **Przyjęte założenia, doprecyzowanie treści**

Projekt został zaimplementowany z wykorzystaniem Javy, natomiast dla porównania jakości rozwiązania została zastosowana biblioteka pythona Scikit-learn (RandomForestRegressor).

1. **Podział odpowiedzialności**
   * Paweł Świątkowski:
     + Stworzenie struktur danych do implementacji zadania w Javie
     + Przygotowanie danych (dummy variables)
     + Implementacja tworzenia drzewa decyzyjnego
     + Zapisanie struktury zbudowanego drzewa
   * Grzegorz Aleksiuk:
     + Implementacja budowy lasu losowego
     + Implementacja przycinania drzewa decyzyjnego
     + Implementacja k-krotnej walidacji krzyżowej
     + Implementacja zadania z wykorzystaniem pythona (Scikit-learn)
2. **Opis algorytmu**
   * Przygotowanie danych – dane zawierały pola tekstowe, reprezentujące dni tygodnia oraz miesiące. Kolumny te zostały zastąpione dummy varaibles. Przykładowo kolumna tekstowa WeekDay została zastąpiona przez 7 kolumn binarnych, każda reprezentująca odpowiedni dzień tygodnia, analogicznie w przypadku kolumny Month.
   * Budowanie drzewa decyzyjnego – algorytm CART – jako wejście algorytmu przyjmowane są próbki danych oraz argumenty względem których będzie odbywał się podział.

W trakcie budowania drzewa decyzyjnego wybierany jest argument oraz jego wartość względem jakiego będzie odbywał się podział – ten wybór następuje poprzez sprawdzenie każdego możliwego podziału i wybraniu takiego, dla którego dostajemy najmniejszy błąd średniokwadratowy. Takie podziały następują aż do czasu kiedy osiągnęliśmy maksymalną zdefiniowaną głębokość drzewa, bądź w danym wierzchołku drzewa mam już tylko 3 próbki danych. Dodatkowy warunek wynika z implementacji przycinania drzewa w trakcie jego budowy tj. jeżeli błąd średniokwadratowy przed podziałem był mniejszy niż po podziale.

* + Budowanie lasu losowego – jako wejście przyjmujemy wszystkie próbki danych.

Tworzonych jest zbiorów. W każdym ze zbiorów znajduje się B próbek danych, losowanych z zbioru początkowego ze zwracaniem (próbki w danym tworzonym zbiorze mogą się powtarzać). Następnie dla każdego z tych zbiorów losowane różnych argumentów (losowanie bez zwracania).

Dla każdego ze stworzonych zbiorów budowane jest drzewo decyzyjne.

Zbiór stworzonych drzew stanowi las losowy.

Wynik dla nowo wprowadzonej próbki uzyskuje się, poprzez wzięcie średniej wartości z wyników zwracanych przez drzewa decyzyjne.

* + K-krotna walidacja krzyżowa – jest stosowana do oceny jakości stworzonego modelu przy zachowaniu jak największej ilości danych do zbioru trenującego.

Dane są dzielone losowo na K możliwie równych zbiorów T1, T2 … Tk. Następnie model lasu losowego jest uczony na zbiorze T – Ti (zbiór danych treningowych) natomiast zbiór danych testowych to Ti. Na podstawie zbioru testowego wyliczany jest błąd średniokwadratowy modelu Ei który zapisujemy. Taka operacja jest powtarzana dla i = 1, 2 … K. Po tym model jest uczony na pełnym zbiorze danych T, a jego błąd średniokwadratowy to średnia wyliczonych błędów E1,E2 … Ek.

1. **Raport z przeprowadzonych eksperymentów**

Stworzyliśmy model w Javie oraz pythonie. Oba modele zostały nauczone tymi samymi danymi przy pomocy 10-krotnej walidacji krzyżowej. Poniższy wykres prezentuje jakość każdego z modeli (błąd predykcji jaki uzyskujemy dla danych próbek). Im mniejszy błąd tym przewidziana wartość jest bliższa realnemu wynikowi.

|  |  |  |
| --- | --- | --- |
| **Kategoria** | **Python** | **Java** |
| Średni Błąd Kwadratowy | 788.7654195 | 558.391 |
| Średni Błąd | 15.55 | 11.79 |

Dodatkowo jest załączona tabela pokazująca błędy każdego z modeli.

(szczegóły znajdują się w załączonym pliku Excela Wyniki.xlsx)

1. **Wnioski**

Jakość naszego modelu w Javie jest statystycznie lepsza od jakości modelu pythona – co można łatwo wywnioskować z załączonego wykresu oraz tabeli. Mimo to, nie jest ona zadowalająca w pełni, wynika to z małej ilości próbek (około 500).

W calu usprawnienia modelu, można by zastosować feature engineering.

Z powodu małej ilości danych, nasze modele drzew decyzyjnych nie są w stanie nauczyć się zależności miedzy tymi danymi.

Dodatkowo problemem jest sam algorytm budowania lasu losowego, konkretnie chodzi o losowanie argumentów które są brane pod uwagę podczas budowania drzewa decyzyjnego. W naszym przypadku wybierane jest 5 argumentów, przez co może się zdarzyć że wybrane zostaną argumenty oznaczające dni tygodnia, w związku z czym tak powstałe drzewo decyzyjne będzie bardzo niskiej jakości.

Jednym ze sposobów poprawy wyników zwracanych przez las losowy może być przypisanie do drzewa współczynnika proporcjonalnego do jego jakości. Współczynnik ten będzie uwzględniany podczas wyliczania średniej wartości dla lasu tj, jeżeli dane drzewo decyzyjne jest niskiej jakości to spowoduje, że jego współczynnik będzie niski, przez co wynik zwracany przez to drzewo będzie mniej wpływał na wartość zwracaną przez las.