Rafał Mironko – WSI Zadanie 4: ID3

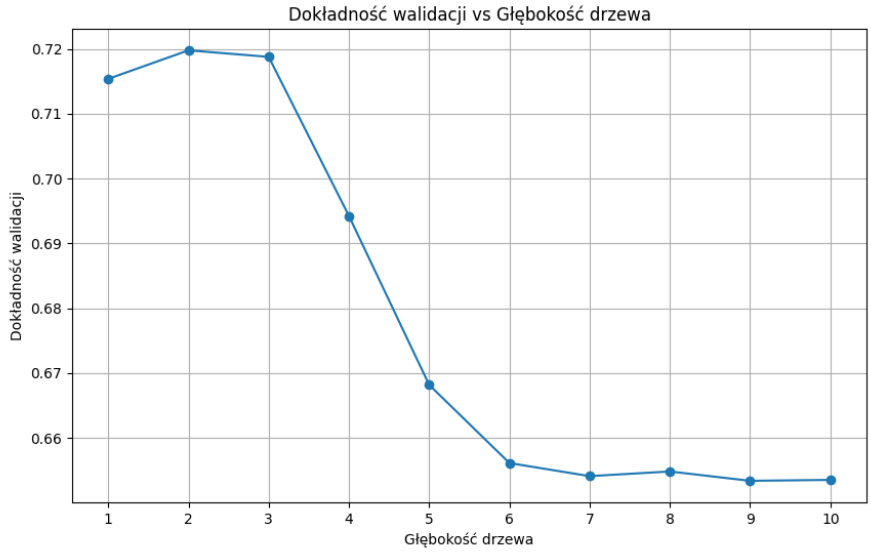
Temat maszynowego uczenia naprawdę mi się spodobał. Praca domowa wymagająca, zabrała sporo czasu, ale wszystko udało się ukończyć.

Podobnie jak poprzednio – nie mam pewności, czy jest sens rozpisywać się na 10 stron raportu. Zamiast tego, pokażę kilka wykresów i powiem, jaki parametr szczególnie zaskoczył mnie podczas testów.

Zgodnie z poleceniem, wykonam testy dla różnych głębokości drzewa [1; 10] – uwzględnię jednak też przygotowanie danych.

Po napisaniu pierwszej wersji kodu, uruchamiam go i czekam na wynik. Mieli się, mieli, czekam 10 minut, przerywam program. Po napisaniu kilku debugujących logów okazało się, że po 10 minutach jestem dopiero na głębokości 7. Dziwne.

Oczywiście, zamiast przyjrzeć się danym bliżej, postanawiam rozwiązać problem siłą. Przerobiłem kod tak, że teraz różne głębokości wykonują się wielowątkowo. Tym razem po trzech minutach doczekałem się wyniku:



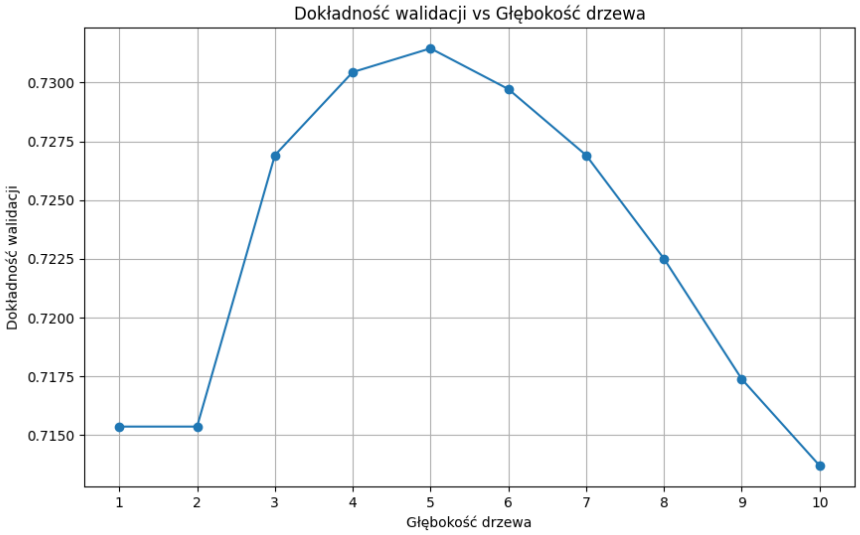
Nie ukrywam, nie jestem z niego zadowolony. Czekałem tyle czasu tylko po to, żeby zobaczyć, że głębokości ponad 4 są beznadziejne?

Co ciekawe, nawet chatgpt nie był w stanie mi pomóc. Szukałem rozwiązania, przeglądałem stack overflow, nawet przeczytałem rozdział „Sztucznej Inteligencji dla Inżynierów” o uczeniu maszynowym. W przeciwieństwie do poprzednich rozdziałów jednak, ten nie okazał się zbyt przydatny.

Odpowiedź przyszła z całkiem nieoczekiwanego źródła. Niedawno kupiłem sobie „Uczenie maszynowe z użyciem Scikit-Learn, Keras i TensorFlow”. Już nie chciało mi się szukać rozwiązania tego problemu, postanowiłem sobie po prostu poczytać ciekawą książkę o uczeniu maszynowym. I...

Dowiedziałem się, że czasem warto użyć mniejszej ilości kontenerów na dane. Ja miałem 15, autor polecił zejść nawet do trzech.

To mnie na tyle zaciekawiło, że od razu zabrałem się za sprawdzenie. Uruchomiłem algorytm na ilości przedziałów dla każdej dyskretyzowanej danej równej 3. Okazało się, że to było właśnie rozwiązanie mojego problemu.



Teraz algorytm zachowuje się tak, jak można się było tego spodziewać. Wychodzi na to, że przy zbyt wysokiej liczbie opcji, algorytm szybko nadmiernie dopasowuje się do danych treningowych i ulega przeuczeniu. Jak widać, przy zmniejszonej liczbie przedziałów, uległo to poprawie.

I jeszcze jedna rzecz. Algorytm teraz wykonuje się kilka sekund. Zastanawiałem się dlaczego, policzyłem to i okazuje się, że ograniczyłem liczbę kombinacji na dyskretyzowanych przedziałach z 15*15*15*15 do 3*3*3*3. Nic dziwnego, że to pomogło.

Podsumowując, ćwiczenie niesamowicie ciekawe. Szczególnie cieszę się, że odkryłem, że zmiana przekazania algorytmowi danych może aż tak mocno zmienić działanie algorytmu.

Na koniec puściłem sobie trenowanie z większą ilością danych na noc. Oto, co mi wyszło:

```
Kombinacja 3125/3125: a=5, b=5, c=5, d=5, e=5
Używanie 12 procesów do równoległego przetwarzania.
Dokładność dla głębokości 1: 0.7154
Dokładność dla głębokości 2: 0.7154
Dokładność dla głębokości 3: 0.7249
Dokładność dla głębokości 4: 0.7315
Dokładność dla głębokości 5: 0.7250
Dokładność dla głębokości 6: 0.7161
Dokładność dla głębokości 7: 0.7043

Najlepsza głębokość: 4 z dokładnością walidacji: 0.7315
Konfiguracja (5, 5, 5, 5, 5) nie jest lepsza od (0.7331890331890332, 5, 1, 2, 3, 2)

Najlepsza konfiguracja binów:
Dokładność walidacji: 0.7332
Biny: a=5, b=1, c=2, d=3, e=2
PS C:\wju\PYTON\WSI\Zadanko_4_Uczenie_Maszynowe>
```