[PSZTY] – PROJEKT 2

DOKUMENTACJA

DARIA SZULIM 290643
ALEKSANDRA KONDRAT 295801

OPIS

Zaimplementować algorytm AdaBoost (Adaptive Boosting) do predykcji bazujący na sekwencji drzew decyzyjnych. Zbiór danych do użycia: Boston Housing - https://www.kaggle.com/c/boston-housing / http://lib.stat.cmu.edu/datasets/boston. Uzyskane rezultaty porównać z wynikami dla wybranej implementacji algorytmu ML z dostępnych bibliotek np. Scikit-learn, WEKA, MLlib, Tensorflow/Keras etc.

REALIZACJA ZADANIA

Podział odpowiedzialności:

Aleksandra Kondrat	**Daria Szulim**
Szkielet programu	Dane wejściowe
Implementacja algorytmu AdaBoost	
Implementacja algorytmu ML	Poprawki kodu
Testy, pomiary	Dokumentacja projektu

Opis:

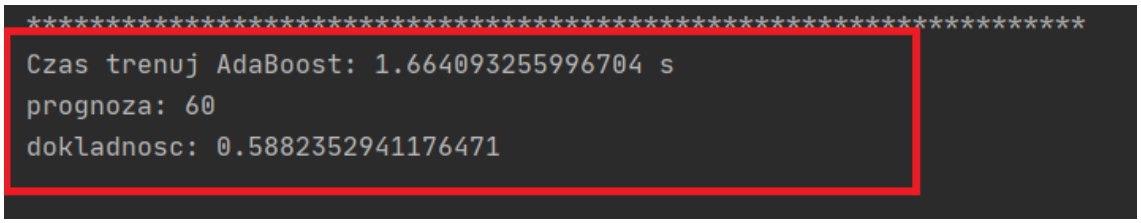
AdaBoost działa w ten sposób, że w kolejnych iteracjach trenuje a następnie mierzy błąd wszystkich dostępnych słabych klasyfikatorów. W każdej następnej iteracji "ważność" źle zakwalifikowanych obserwacji jest zwiększana, tak że klasyfikatory zwracają na nie większą uwagę. Podejścia są sekwencyjne, a kolejne klasyfikatory są ze sobą ściśle związane. Wszystkie klasyfikatory trenujemy na tym samym zbiorze danych, ale z różnym zestawem wag. Wagi te zależą od wyników poprzedniego treningu, a więc trudno jest trenować równolegle wiele klasyfikatorów. Warto podkreślić, że algorytm AdaBoost może znacznie poprawić jakość klasyfikacji, jednak owa poprawa obserwowana jest tylko wtedy, gdy jako składowych używamy “słabych” klasyfikatorów. Aby uzyskać rozsądną efektywność wystarczy, by skuteczność “słabych” uczniów była nieco wyższa niż 50%. Stosowanie algorytmu do bardziej skomplikowanych klasyfikatorów nie prowadzi na ogół do znaczącego zwiększenia skuteczności.

Struktura projektu:

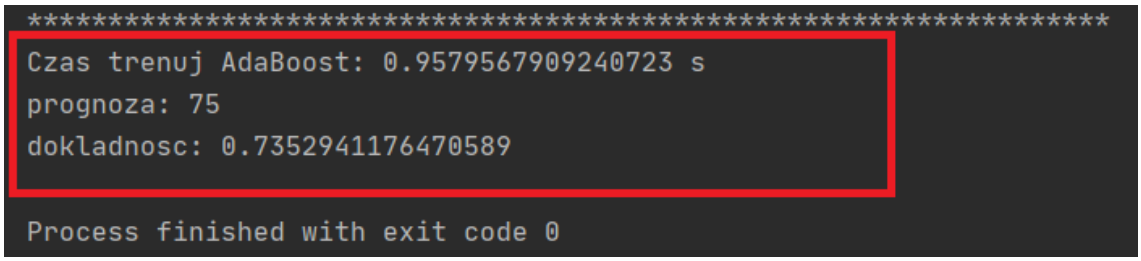
Projekt jest napisany w języku Python w środowisku PyCharm. Składa się z 3 plików o rozszerzeniu .py (AdaBoost.py, SlabyKlasyfikator.pt, main.py) i dwóch plików .txt (test.txt, train.txt) przechowujących dane do testowania. Plik SlabyKlasyfikator.py reprezentuje słaby klasyfikator algorytmu jednowarstwowego drzewa decyzyjnego. Zdefiniowana jest tu klasa DrzewoDecyzyjne oraz jej funkcje: odpowiadające za wyszukanie najniższego progu spośród wszystkich parametrów minimalnego prógu i-tego parametru, gdzie flaga progu wynosi 1 lub -1 itp. Plik AdaBoost.py w którym zdefiniowana jest klasa AdaBoost oraz jej funkcje m.in. *trenuj* w której każdy słaby klasyfikator tworzy hipotezę wyjściową dla każdej próbki w zestawie treningowym. Przy każdej iteracji t wybiera się słaby klasyfikator i przypisuje się taki współczynnik , że suma błędów treningu wynikowego klasyfikatora jest minimalizowana. Plik główny programu jest odpowiedzialny między innymi za wczytywanie i odpowiednie przygotowanie danych z plików txt. Dane są podzielone ze względu na typ train i test dzięki funkcji *train_test_split(x, y , test_size, random_size).* Otrzymujemy dwa różne zestawy danych - jeden dla niezależnych funkcji (x), a drugi dla zmiennej zależnej (y) - ostatnia kolumna zestawu danych train.txt. Następnie następuje kolejny podział danych x i y na odpowiednio xTreuj , xTestuj i yTrenuj, yTestuj. W pliku main jest również wywoływany algorytm AdaBoost i mierzony jego czas działania. Dodatkowo został dodany plik Testuj.py, który zawiera implementacje AdaBoostClasifier z biblioteki Scikit-learn.

TESTOWANIE

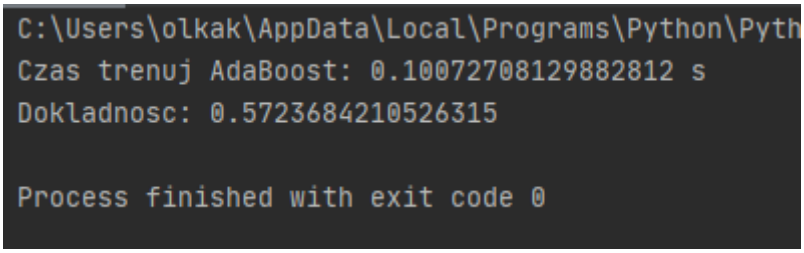
Program został przetestowany na danych udostępnionych w linku umieszczonym w treści zadania : http://lib.stat.cmu.edu/datasets/boston.
Otrzymany rezultat prezentuje poniższe zdjęcie ekranu:



Wynik działania programu na zestawie danych przygotowanych przez nas na potrzeby projektu:



Wynik działania programu z użyciem biblioteki Scikit-learn:



	CZAS	PROGNOZA	DOKŁADNOŚĆ
AdaBoost - Boston	1.5658 s	60	0.58823
AdaBoost - dane.txt	0.7353 s	75	0.73529
Adaboost Scikit-learn - Boston	0.1007 s	-	0.57237

WNIOSKI

Klasyfikator AdaBoost buduje silny klasyfikator o wysokiej dokładności, łącząc wiele słabo działających klasyfikatorów. Ostateczna prognoza (60) została uzyskana poprzez zsumowanie ważonych prognoz każdego klasyfikatora. Dokładność (58,82%) modelu danych mówi nam, ile razy przewiduje on prawidłowe klasy. Nie jest to wysoka dokładność ,ale przewyższa 50 % i znacząco poprawia dokładność słabych klasyfikatorów. Należy pamiętać , że technika wspomagająca uczy się stopniowo, wartość otrzymanej dokładności zależy od jakości przetwarzanych danych. Adaboost jest (niestety) bardzo wrażliwy na dane odstające, więc przed użyciem zaleca się ich wyeliminowanie w celu otrzymania bardziej satysfakcjonujących wyników. Zestaw danych (przygotowany na podstawie zamieszczonych w poleceniu linków), na którym było przedstawione zadanie mógł zawierać wyżej wymienione dane odstające, co może skutkować niezbyt wysoką wartością ostatecznej dokładności. Aby sprawdzić dokładność działania programu został przygotowany zestaw odpowiednio dobranych danych, co skutkowało znaczne lepszą dokładnością (73,52%) i prognozą (75). Przy skorzystaniu z biblioteki Scikit-learn dla danych Boston wartość dokładności jest porównywalna do implementacji projektu, jednak czas jest znacząco krótszy. Wynika to najprawdopodobniej z mniejszej złożoności algorytmu biblioteki.

URUCHOMIENIE PROGRAMU

Uruchomienie programu następuje poprzez run pliku main.py.