Zaawansowane metody klasyfikacji

Wstęp

W celu poprawy jakości pojedynczych modeli jednym z ciekawych rozwiązań jest zastąpienie pojedynczego eksperta konsylium ekspertów wspólnie podejmujących decyzję.

Najprostszym tego przykładem jest wspólne głosowanie. Polega ono na podjęciu niezlaeżnej decyzji przez każdego z ekspertów osobno, a następnie na podjęciu kolektywnej decyzji przez niezależne głosowanie.

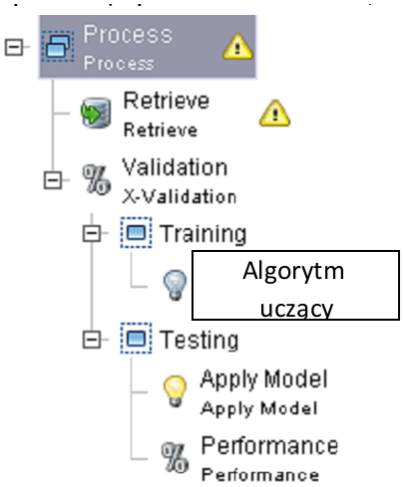
Inną formą budowania konsylium ekspertów jest wykorzystanie jednego z ekspertów do podjęcia decyzji, a następnie powołanie kolejnego eksperta specjalizującego się w obszarach w których pierwszy zwykle popełniał błędy. Powtarzając tą procedurę kilkukrotnie dostajemy konsylium uzupełniających się ekspertów, którzy na koniec podejmują w sposób ważony. Ideę tą implementują algorytmy uczone ze wzmocnieniem (ang. Boosted) , czego przykładem jest algorytm *AdaBoost*. Jego idea polega na tym, iż każdemu wektorowi przypisana jest odpowiednia waga określająca pewność zaufanie z jakim algorytm podejmuje decyzję (odpowiedź) w stosunku do tego wektora. Początkowo każda z wag ma równą wartość i wynosi ona *1/n,* gdzie *n*- to liczba wektorów w zbiorze, następnie buduje się wstępny model. Model ten wykorzystywany jest następnie do wyznaczenia/aktualizacji wag tak, iż każdy wektor gorzej klasyfikowany dostaje większą wartość wagi. W następnym kroku proces uczenia jest powtarzany i tworzony jest nowy model, jednak tak, aby skoncentrować się na tych wektorach które mają większą wagę. Na samym końcu podczas podejmowania decyzji, wszystkie z modeli biorą udział proporcjonalny do ich znaczenia.

Kolejną z idei budowy konsylium ekspertów jest budowa ekspertów dla podzbioru zadań. W tym przypadku ze wszystkich problemów do rozwiązania losowany jest ze zwracaniem podzbiór problemów a następnie dla tego podzbioru szukany jest ekspert. Powyższą koncepcję implementuje algorytm *Bugging.* W algorytmie tym z całego zbioru danych uczących losowany jest podzbiór (losowanie ze zwracaniem) i dla tego podzbioru budowany jest model predykcyjny, następnie po raz kolejny ze zwracaniem losowany jest inny podzbiór wektorów i dla niego budowany jest kolejny model. Całość powtarzana jest *k* krotnie a na koniec wszystkie zbudowane modele użyte są do głosowania.

Szczególnym przykładem koncepcji *Bugging* jest algorytm *losowych lasów drzew* (ang. *Random Forest*) jego idea polega na zbudowaniu konsylium ekspertów z *losowych drzew* (ang. *Random tree*) gdzie w odróżnieniu od klasycznych drzew decyzji losowe drzewa budowane sa na tej zasadzie iż podzbiór analizowanych cech w węźle dobierany jest losowo, ponadto poszczególne drzewa z *losowych lasów drzew* budowane są zgodnie z koncepcją *Bugging.*

Do wykonania w RapidMiner

1. Zbuduj Schemat do testów jak na rysunku Pamiętaj aby wczytać zbiór danych w odpowiedni sposób (Operator read CSV, read ARFF, inne). Do testów wykorzystaj test krzyżowy.
   Uwaga: Pamiętaj aby odpowiednio oznaczyć kolumnę typu „LABEL”
   Testów dokonaj na zbiorze Zbiór5.csv



2. Jako model bazowy wykorzystaj drzewo decyzji. Spróbuj zoptymalizować jego parametry, tak aby uzyskać jak największą dokładność. W tym zadaniu dobrym rozwiązaniem jest wykorzystanie operatora *Loop Parameters*
   Pytanie: Dla jakich parametrów drzewo decyzji uzyskało największą jakość predykcji?
   Uwaga: Optymalizując drzewo włącz opcję *„no pre prunning”.* Optymalizacji poddaj parametry *„minimal gain*” oraz *„confidence*”.
3. Zgodnie z instrukcją najprostszym rozwiązaniem poprawy jakości poszczególnych modeli jest głosowanie co można zrealizować wykorzystując operator *Vote.* W tym celu dodaj ten operator do kroswalidacji. Operator ten posiada podproces pozwalający na określenie podmodeli użytych do głosowania. W tym miejscu podłącz kilka różnych modeli: najlepsze drzewo decyzji (zad 2), model kNN oraz modele sieci neuronowych 8 neuronach w warstwie ukrytej.
   Pytanie: Czy uzyskane wyniki są lepsze od każdego z modeli składowych uzyskanych niezależnie?
   Pytanie: Czy jeśli zmienisz parametry konfiguracyjne modeli składowych zawsze głosowanie będzie poprawiało wyniki? Uzasadnij swoją wypowiedź
   Pytanie: Czy jeśli wstawisz trzy takie same modele, np. trzy sieci neuronowe o różnej liczbie neuronów w poszczególnych warstwach też uzyskasz poprawę w stosunku do każdego z modeli z osobna?
4. W miejsce *Algorytmu uczącego* z zadania 1 wstaw operator *AdaBoost* a jako podproces wstaw najlepsze uzyskane drzewo. Przeprowadź obliczenia i sprawdź jak wpływa dodanie tego operatora na uzyskane wyniki.
   Pytanie: Czy zastosowanie operatora AdaBoost spowodowało poprawę jakości predykcji
   Pytanie: Jak zastosowanie operatora AdaBoost wpłynęło na wariancję uzyskanych wyników
5. W podobny sposób jak w zad 5 dokonaj weryfikacji rozwiązania opartego o operator Bugging
   Pytanie: Czy zastosowanie operatora Bugging spowodowało poprawę jakości predykcji
   Pytanie: Jak zastosowanie operatora Bugging wpłynęło na wariancję uzyskanych wyników
6. Zastąp drzewo decyzji lasem drzew. W tym celu wykorzystaj operator *Random Forest* Jego ustawienia określ identycznie jak najlepszego drzewa decyzji uzyskanego z poprzedniego zadania.
   Pytanie: Jak zastosowanie drzewa wpłynęło na jakość wyników?
   Pytanie: Czy drzewa składowe lasu są identyczne, czy może różnią się, jeśli się różnią to skąd wynikają różnice?