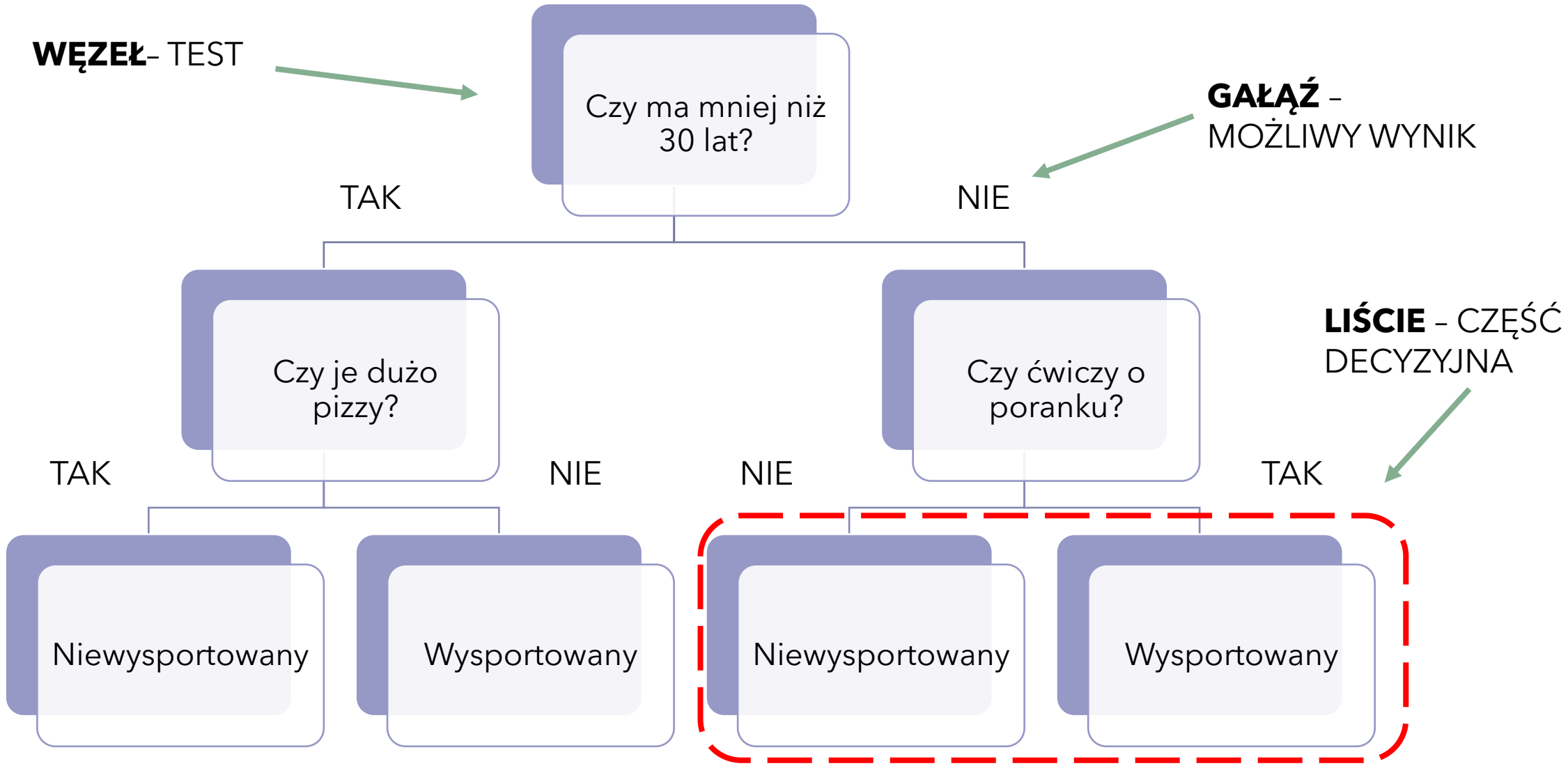

XGBOOST
Katarzyna Raca

Drzewa decyzyjne

- pierwsze wzmianki pojawiły się w pracach Morgana i Sonquista (1963)
- zyskały rozgłos dzięki pracy Breiman (1984 – analiza dyskryminacyjna i regresji, algorytm CART)
- służą do zadań klasyfikacji oraz regresji
- składowe losowych lasów
- nie jest potrzebne skalowanie cech
- odporne na wartości nietypowe

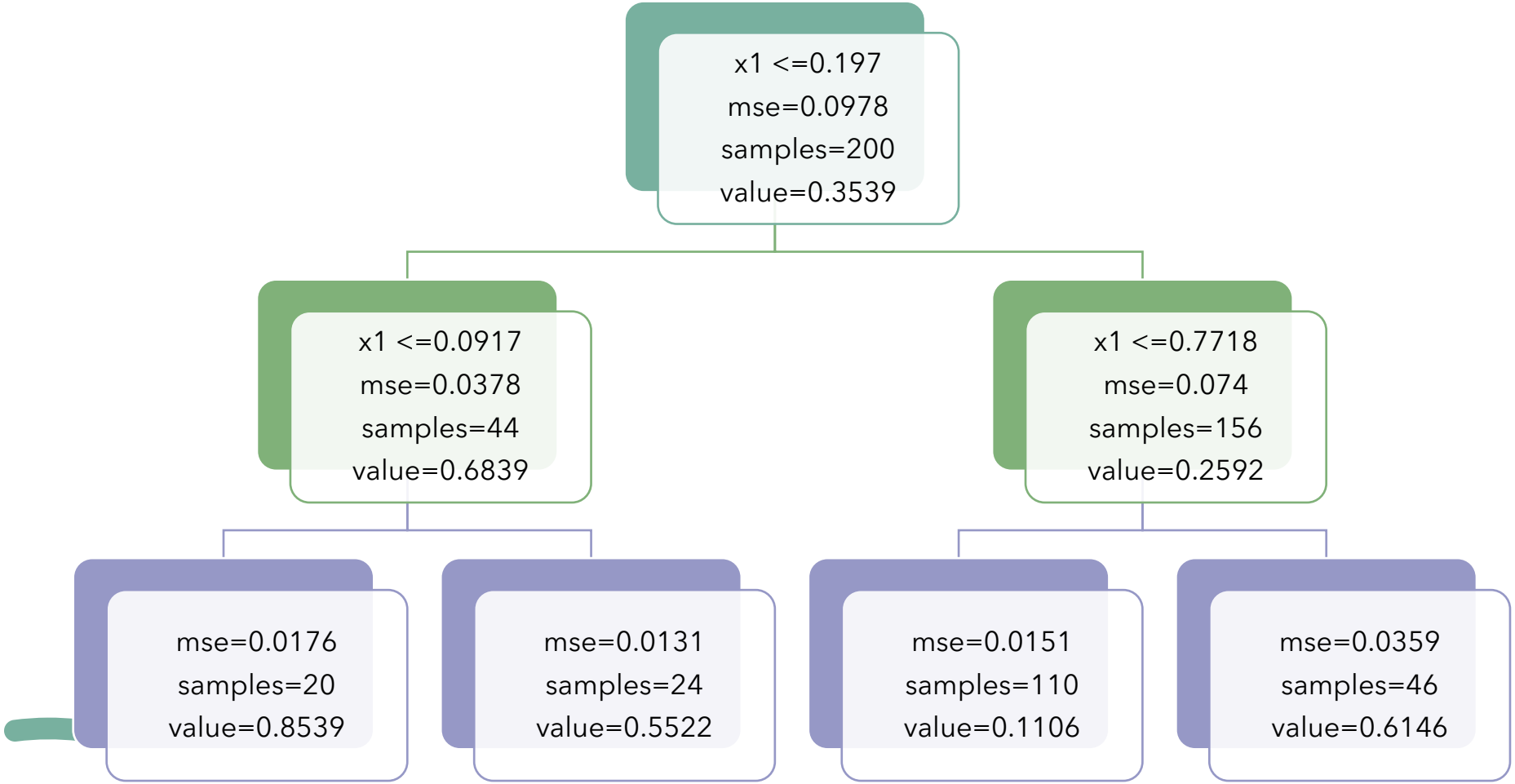
Czy osoba jest wysportowana?



Budowanie drzewa

1. Wybór reguły podziału dla każdego węzła wewnątrz drzewa. Oznacza to wybór konkretnych cech oraz dla ich użycia do podziału analizowanego zestawu danych na każdym z węzłów.
2. Określenie zasad wyboru węzłów, które będą węzłami kończącymi podział, tzn. na których zostanie dokonana ostateczna klasyfikacja analizowanej próbki
3. Zastosowanie algorytmów mających na celu optymalizacje tworzonego drzewa decyzji, np. przycięcie drzewa (pruning)

Drzewo decyzyjne – zadanie regresyjne



Zwiększenie efektywności drzew decyzyjnych

- **Komitety drzew** : zbiór modeli decyzyjnych (tu drzew decyzyjnych) analizujących dany problem przy różnym podejściu. Wynik jednoczesnego działania może zostać uśredniony lub poddany pod głosowanie – może zostać stworzony oddzielny algorytmpodejmujacy ostateczna decyzję bądź to poprzez wybranie decyzji na którą wskazuje większość drzew , lub poprzez głosowanie ważone.
- **Wzmacnianie (boosting)** – iteracyjnie tworzy się coraz lepszy model procedury klasyfikacyjnej. Klasyfikowanym danym przypisane zostają odpowiednie wagi, które w kolejnych iteracjach zostają zmienione w zależności od tego jak dobrze tworzone drzewo je klasyfikuje. Zwiększone zostają wagi tych przypadków, które algorytm klasyfikuje z gorszą dokładnością, by „zachęcić" algorytm do koncentracji na nich.
- **Agregacja (bootstraping)** – generowane jest wiele modeli w oparciu o losowo wybrane zbiory próbek z dostępnego zestawu danych (zbiory mogą zawierać te same próbki). Testowanie modeli odbywa się na próbkach które nie zostały wybrane do tworzenia modeli. Ostatecznie wybierany jest model zapewniający najlepsza dokładność.

Lasy losowe

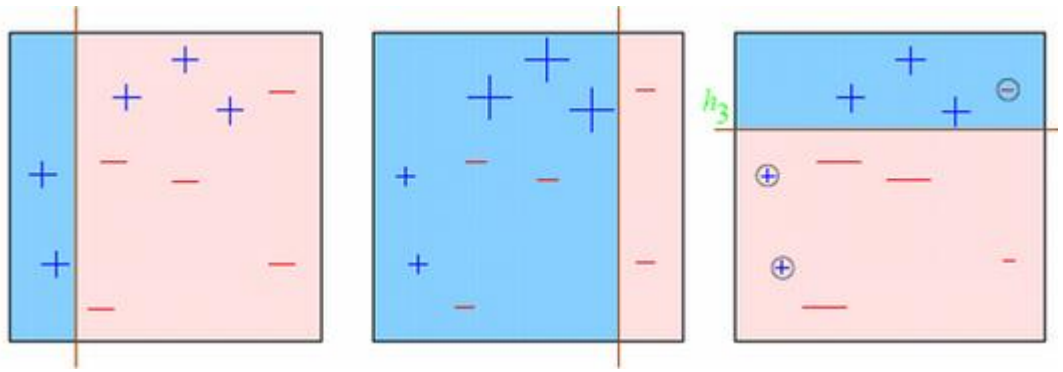
- model składający się z wielu drzew decyzyjnych, aby ograniczyć zjawisko niestabilności prognoz.
- zawiera wiele pojedynczych drzew, każde jest oparte o losową próbę z danego zbioru danych
- są one zazwyczaj bardziej dokładne niż pojedyncze drzewo decyzyjne
- prognozowanie na podstawie modelu lasu losowego polega na określeniu prognoz dla każdego drzewa wchodzącego w skład lasu oraz wyznaczeniu średniej arytmetycznej tych prognoz indywidualnych, jako prognozy całego modelu
- staramy się uzyskać drzewa jak najbardziej niezależne od siebie, aby zmniejszyć wariancję modelu
- im więcej dodajemy drzew, tym dokładniejszy i stabilniejszy się staje

Wzmacnianie drzew

- sekwencyjne uczenie predyktorów w taki sposób, że każdy następny próbuje korygować poprzednika.
- łączy kilka klasyfikatorów słabych w klasyfikator silny,
- rodzaj algorytmów uczenia się, które próbują dopasować dane za pomocą wielu prostszych modeli lub tak zwanego podstawowego ucznia / słabego ucznia. Sposób polega na adaptacyjnym dopasowaniu danych przy użyciu tego samego lub nieco innego w podstawowych uczących się ustawieniach parametrów.

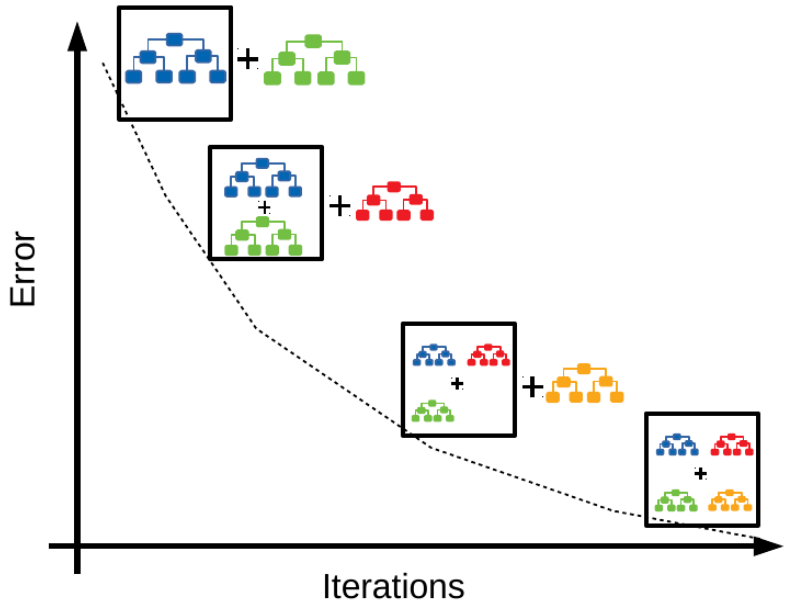
Najpopularniejsze algorytmy to:

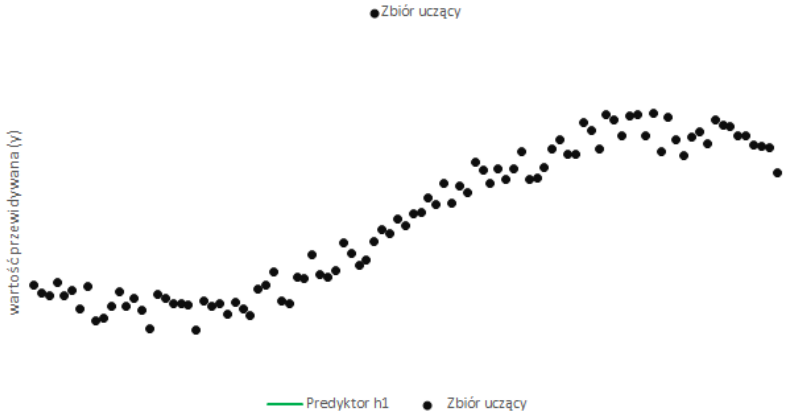
- wzmacnianie adaptacyjne – AdaBoost – Adaptive Boosting
- wzmacnianie gradientowe – Gradient Boosting

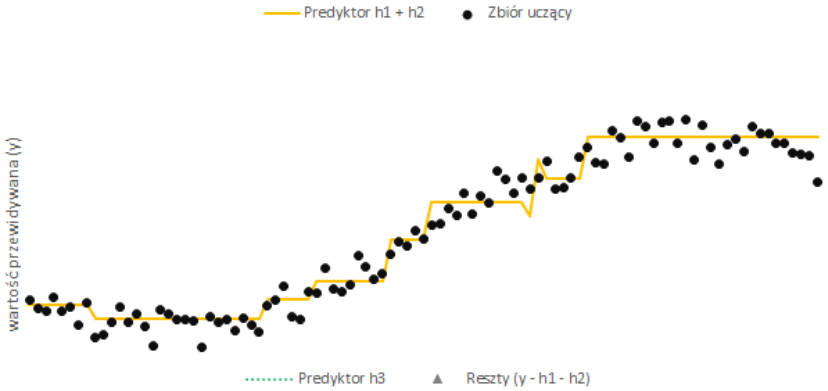


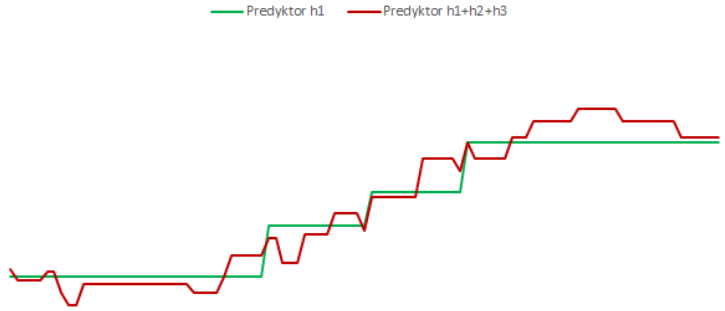
Gradient Boosting

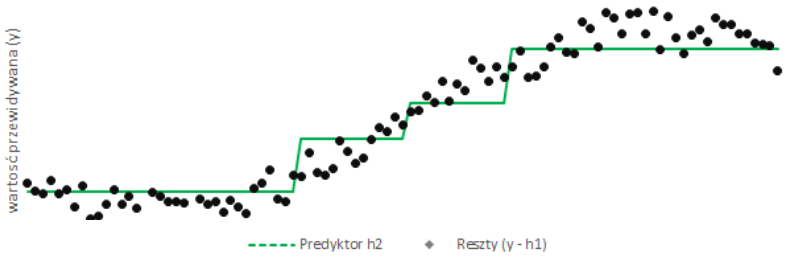
Dodaje kolejne predyktory do zespołu w sposób sekwencyjny, gdzie następny poprawia poprzednika. Jednak nie aktualizujemy tu wag przykładów po każdym przebiegu, lecz próbujemy dopasować predyktor do błędu resztowego popełnionego przez poprzedni predyktor

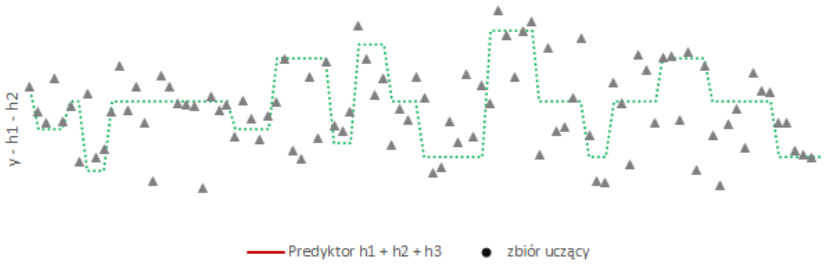



Zbiór uczący
wartość przewidywana (y)
Predyktor h1
Zbiór uczący

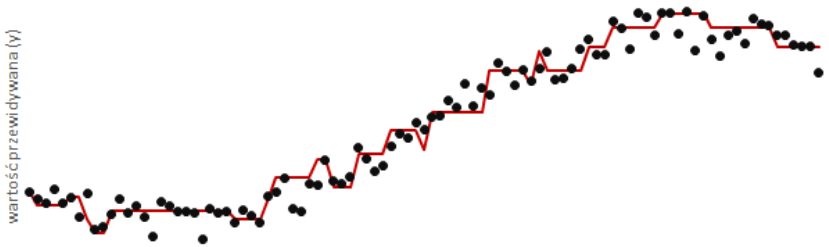

Predyktor h1 + h2
Zbiór uczący
wartość przewidywana (y)
Predyktor h3
Reszty (y - h1 - h2)


Predyktor h1
Predyktor h1+h2+h3


wartość przewidywana (y)
Predyktor h2
Reszty (y - h1)

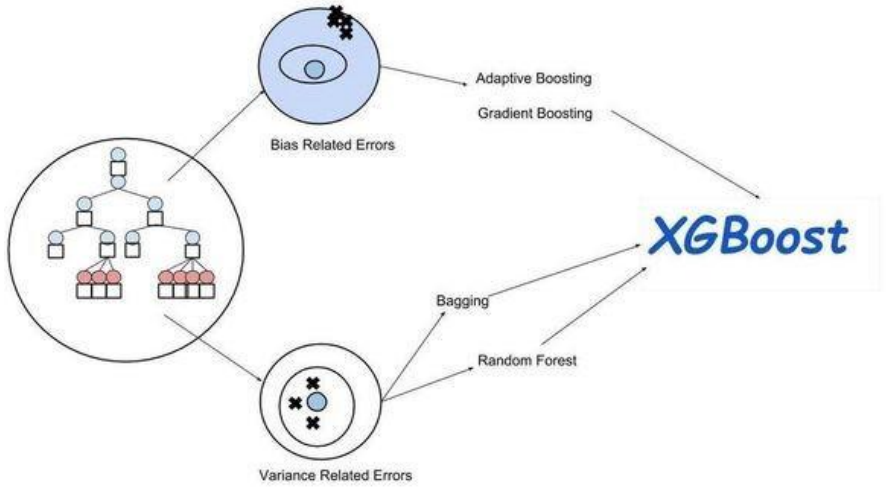

y - h1 - h2
Predyktor h1 + h2 + h3
zbiór uczący


y - h1


wartość przewidywana (y)

XGBoost

- ogólna postać algorytmu XGBoost składa się z dwóch części. Pierwszy składnik to ten odpowiadający za minimalizację błędu, nazywany funkcją straty, inaczej funkcją kosztu. Drugi zaś, czyli regularyzacja, pomaga zapobiec przetrenowaniu i kontroluje złożoność modelu.
- jest to specyficzna implementacja metody Gradient Boosting, która wykorzystuje dokładniejsze przybliżenia w celu znalezienia najlepszego modelu drzewa.



Polecam obejrzeć: https://www.youtube.com/watch?v=A53Vf_GD9H8

Gradient Boosting/XGBoost

1. XGBoost oblicza gradient drugiego rzędu, tj. drugich pochodnych cząstkowych funkcji straty (podobnej do metody Newtona), która dostarcza więcej informacji na temat kierunku gradientów i jak dojść do minimum naszej funkcji straty. Podczas gdy gradient boosting wykorzystuje funkcję straty naszego modelu podstawowego (np. Drzewa decyzyjnego) do minimalizacji błędu całego modelu, XGBoost używa pochodnej drugiego rzędu jako przybliżenia.
2. XGBooost używa zaawansowaną regularyzacja (L1 i L2), która poprawia generalizację modelu. XGBoost ma dodatkowe zalety: szkolenie jest bardzo szybkie i może być równoległe / rozproszone między klastrami.

$$w_{jm} = \begin{cases} -\frac{G_{jm}}{n_{jm}} & GBM, \\ -\frac{G_{jm}}{H_{jm}} & XGBoost. \end{cases} \qquad Gain = \begin{cases} \frac{G_{jmL}^2}{n_{jmL}} + \frac{G_{jmR}^2}{n_{jmR}} - \frac{G_{jm}^2}{n_{jm}} & GBM, \\ \frac{1}{2}[\frac{T_\alpha(G_{jmL})^2}{H_{jmL}+\lambda} + \frac{T_\alpha(G_{jmR})^2}{H_{jmR}+\lambda} - \frac{T_\alpha(G_{jm})^2}{H_{jm}+\lambda}] - \gamma & XGBoost. \end{cases}$$

Główne parametry XGBoost

- **learning_rate / eta** [default=0.3]: parametr mówiący po każdej wyliczonej iteracji jaki krok chcemy dać do przodu. Im większy krok tym szybciej zbliżamy się do celu, ale jeśli będzie zbyt duży to możemy nie dojść do najlepszego wyniku.
- **max_depth** [default=6]**:** maksymalna głębokość prostych drzew. Im głębsze drzewa tym model jest mocniejszy, ale trzeba uważać by nie przeuczyć modelu.
- **n_estimators** : liczba tych prostych drzewek, które chcemy zbudować.
- **min_child_weight**: mówi o minimalnej liczbie obserwacji w każdym liściu drzewa. Im większa waga tym model bardziej konserwatywny – potrzebujemy większej wagi by dokonać danego podziału.
- **gamma** [default=0]: odpowiada za zmniejszenie strat wymaganych do utworzenia kolejnego węzła liści. Im większa waga tym model jest bardziej konserwatywny.
- **seed** [default=0]: nasionko, które służy do generowania liczb losowych.

Do walki z przeuczeniem

- **subsample** [default=1]: Informacja jaki procent obserwacji chcemy brać do budowy prostego drzewka
- **colsample_bytree** [default=1]: Informacja jaki procent charakterystyk chcemy brać do budowy prostego drzewka
- **colsample_bylevel** [default=1]: Informacja jaki procent charakterystyk chcemy losować do budowy prostego drzewka po każdym kolejnym podziale danych (split)
- **max_delta_step** [default=0]: Maksymalny krok delta pozwalający na wyjście z każdego liścia. Wartość 0 oznacza brak ograniczeń. Jeśli jest ustawiona na wartość dodatnią, może pomóc w uczynieniu kroku aktualizacji bardziej konserwatywnym. Zwykle ten parametr nie jest potrzebny, ale może pomóc w regresji logistycznej, gdy klasa jest wyjątkowo niezrównoważona. Ustawienie wartości 1-10 może pomóc w kontrolowaniu aktualizacji.

Regularyzacja

- **alpha** [default=0]: odpowiada za parametr alfa przy regularyzacji L1 (Lasso). Im większa waga tym model jest bardziej konserwatywny.
- Można myśleć, że im większa alpha tym mniej charakterystyk jest branych pod uwagę i wyłapywane są bardziej istotne. Stopniowo odrzuca współliniowe atrybuty i pozostawia zbiór najbardziej istotnych.
- **lambda** [default=1]: odpowiada za parametr lambda przy regularyzacji L2 (Ridge regression). Im większa waga tym model jest bardziej konserwatywny.
- Intuicyjnie im większe lambda to wagi będą bliskie zeru.

Dodatkowe

- **eval_metric**: metryka walidacji modelu. Domyślnie dla problemu regresji jest błąd średniej kwadratowej (RMSE) a dla klasyfikacji stosunek dobrych przypisani do złych (accurency).
- **objective:** możliwość ustawienia rodzaju regresji do wyliczeń algotymów dla XGBoosta. Domyślnie ustawiona jest regresja liniowa.
- **tree_method**: po wybraniu **gpu_hist** wykorzystujemy pełną moc kart graficznych i znacznie przyśpieszamy wyliczenia. Ostatnio podczas modelu liczenia ma moim laptopie trwało to jedynie 30 minut a na komputerach dostosowanych do obliczeń z kartami trwało zaledwie kilkanaście sekund.

Więcej informacji…

- https://medium.com/syncedreview/tree-boosting-with-xgboost-why-does-xgboost-win-every-machine-learning-competition-ca8034c0b283
- https://medium.com/implodinggradients/benchmarking-lightgbm-how-fast-is-lightgbm-vs-xgboost-15d224568031
- https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/


XGBOOST
Katarzyna Raca