# Podstawy Sztucznej Inteligencji - projekt II
# Dokumentacja

**Michał Berliński- 290432, Daniel Borowski - 292803**

### Opis problemu

Zaimplementować algorytm AdaBoost (Adaptive Boosting) do klasyfikacji bazujący na sekwencji drzew decyzyjnych. Zbiór danych do użycia: Heart Disease - https://archive.ics.uci.edu/ml/datasets/Heart+Disease. Uzyskane rezultaty porównać z wynikami dla wybranej implementacji algorytmu ML z dostępnych bibliotek np. Scikit-learn, WEKA, MLlib, Tensorflow/Keras etc.

### Przyjęte założenia oraz doprecyzowanie treści:

Zaimplementowaliśmy w języku Java z użyciem standardowych bibliotek algorytm Adaptive Boosting bazujący na sekwencji drzew decyzyjnych. Jego zadaniem jest oszacowanie prawdopodobieństwa zapadnięcia pacjentów na chorobę serca. W tym celu najpierw analizuje on jeden konkretny zestaw danych pacjentów posiadających 14 atrybutów (m.in wiek, cholesterol, a także finalnie fakt czy dana osoba jest chora). Następnie wykorzystuje on zdobytą wiedzę do próby przewidzenia faktu wystąpienia choroby u pacjentów w innych zestawach danych ( na podstawie pierwszych 13 atrybutów ). Wówczas sprawdzana jest jego skuteczność, poprzez porównywanie przewidywań z faktem czy dana osoba jest zdrowa czy chora w rzeczywistości.

### Podział obowiązków:

Implementacja algorytmu AdaBoost była wykonywana wspólnie.

Wczytywanie danych wejściowych, dokumentacja - Michał Berliński

Testy algorytmu, konfiguracja środowiska WEKA - Daniel Borowski

### Zwięzły opis algorytmu / architektura i uzasadnienie sposobu realizacji:

Projekt został napisany w języku Java z wykorzystaniem środowiska Inteliij IDEA. Pliki AdaBoost.java oraz DecisionStump.java odpowiadają za realizowanie algorytmu wykorzystywanego do klasyfikacji, którym jest klasyczny Adaptive Boosting. Został on zaimplementowany w następujący sposób. Tworzone są instancje klasy DecisionStump odwzorowywujące słabe klasyfikatory, dla każdego z nich wyliczany jest optymalny próg podziału. Następnie wybierany jest najlepszy klasyfikator, czyli taki, który dzieli rekordy w najlepszy sposób (najniższa wartość gini impurity). W kolejnym kroku wyliczana jest wartość zmiennej amountOfSay aktualnie trenowanego klasyfikatora. Jest to wartość potrzebna do finalnej klasyfikacji danego rekordu, informująca w jakim stopniu brana jest pod uwagę decyzja pojedynczego klasyfikatora. AmountOfSay wyliczana jest na podstawie sumy wag wszystkich błędnych decyzji, które podjął klasyfikator. Następnie aktualizowane są wartości wag poszczególnych rekordów w taki sposób, że próbki źle zaklasyfikowane dostają większą wagę od reszty, a co za tym idzie będą brane pod uwagę w większym stopniu w kolejnych

iteracjach programu niż te zaklasyfikowane dobrze. Przy wyliczeniu wag istotną funkcję pełni również amountOfSay. W ostatnim kroku treningu słabego klasyfikatora tworzony jest nowy zestaw danych, z większą ilością próbek wcześniej sklasyfikowanych niepoprawnie dzięki czemu następca aktualnego klasyfikatora będzie brał te próbki pod uwagę w większym stopniu.

**Testowanie:**

Do testowania programu wykorzystaliśmy dane ze strony https://archive.ics.uci.edu/ml/datasets/Heart+Disease.

Poniższy zrzut ekranu przedstawia rezultat działania algorytmu wytrenowanego na zestawie danych processed.cleveland.data zawierającego 303 pozycje. Wykorzystany został następnie do przewidzenia wystąpienia choroby serca u pacjentów z zestawów kolejno: processed.cleveland.data, processed.hungarian.data (294 pozycje), processed.switzerland.data (123 rekordy), processed.va.data (200 pozycji) oraz zestawu zawierającego dane zebrane ze wszystkich wcześniej wymienionych plików (w sumie 820). Ogółem testowanie w naszym programie polega na wytrenowaniu algorytmu na wybranym z zestawów, a następnie wykorzystaniu go do przewidywania rezultatów we wszystkich zestawach.

```
Czas trenowania: 170ms
Skutecznosc przewidywania wynosi: 83.82838283828383%
Skutecznosc przewidywania wynosi: 81.29251700680273%
Skutecznosc przewidywania wynosi: 51.21951219512195%
Skutecznosc przewidywania wynosi: 59.5%
Skutecznosc przewidywania wynosi: 73.36956521739131%
```

Dla porównania poniżej widnieje rezultat wykorzystania klasyfikacji AdaBoost oprogramowania WEKA. Algorytm wytrenowany na zestawie z Cleveland, przewidywał szanse choroby dla pacjentów tegoż zestawu z wykorzystaniem 10 krotnej walidacji krzyżowej. Jak widać uzyskany rezultat jest niemal identyczny, co potwierdza skuteczność naszego programu. Na jego niekorzyść różnią się natomiast czasy wykonania.

```
Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         254               83.8284 %
Incorrectly Classified Instances        49               16.1716 %
Kappa statistic                          0.6742
Mean absolute error                      0.2351
Root mean squared error                  0.359
Relative absolute error                 47.3295 %
Root relative squared error             72.0491 %
Total Number of Instances              303
```

Poniżej zostały zaprezentowane rezultaty wykorzystania algorytmu wytrenowanego na zestawie stanowiącym połączenie wszystkich zestawów. Kolejność wypisanych danych jest analogiczna jak w poprzednim przypadku. Znacząco zwiększyła się trafność przewidywań dla zestawów processed.switzerland.data i processed.va.data.

```
Czas trenowania: 1051ms
Skutecznosc przewidywania wynosi: 82.50825082508251%
Skutecznosc przewidywania wynosi: 84.6938775510204%
Skutecznosc przewidywania wynosi: 78.86178861788618%
Skutecznosc przewidywania wynosi: 71.5%
Skutecznosc przewidywania wynosi: 80.32608695652173%
```

Skuteczność klasyfikacji pojedynczego, słabego klasyfikatora obliczana na tym samym zestawie danych:

```
numer klasyfikatora: 12 Skutecznosc przewidywania wynosi: 76.0 %
numer klasyfikatora: 2 Skutecznosc przewidywania wynosi: 75.0 %
numer klasyfikatora: 11 Skutecznosc przewidywania wynosi: 74.0 %
numer klasyfikatora: 8 Skutecznosc przewidywania wynosi: 71.0 %
numer klasyfikatora: 7 Skutecznosc przewidywania wynosi: 70.0 %
numer klasyfikatora: 10 Skutecznosc przewidywania wynosi: 68.0 %
numer klasyfikatora: 9 Skutecznosc przewidywania wynosi: 68.0 %
numer klasyfikatora: 1 Skutecznosc przewidywania wynosi: 61.0 %
numer klasyfikatora: 0 Skutecznosc przewidywania wynosi: 57.0 %
numer klasyfikatora: 6 Skutecznosc przewidywania wynosi: 58.0 %
numer klasyfikatora: 4 Skutecznosc przewidywania wynosi: 57.0 %
numer klasyfikatora: 3 Skutecznosc przewidywania wynosi: 56.0 %
numer klasyfikatora: 5 Skutecznosc przewidywania wynosi: 53.0 %
```

**Wnioski:**

Wyniki są zadowalające, skuteczność przewidywania jest na tyle wysoka, że klasyfikator AdaBoost na podstawie dostarczonych danych jest w stanie sensownie oszacować wystąpienie choroby. Wyraźnie widać, że nie jest to kwestią przypadku (wyniki znacząco różne od 50%). Rezultaty są mocno zbliżone do tych zwracanych przez oprogramowanie WEKA.

Można również zaobserwować ewidentną wrażliwość na jakość dostarczonych danych. Skuteczność przewidywania dla zestawów processed.switzerland.data oraz processed.va.data jest zauważalnie niższa. Spowodowane jest to gorszą jakością tych danych. Niemal wszyscy pacjenci z tych zestawów posiadają kilka atrybutów o niewiadomych wartościach. Możliwa byłaby poprawa tych wyników poprzez nie branie pod uwagę tych “stumpów” gdzie brak danych był szczególnie częsty.

Testy pokazują również to, że AdaBoost tworzy inteligentny duży klasyfikator na podstawie mniejszych, słabych klasyfikatorów. Chociaż każdy z klasyfikatorów ma skuteczność co najwyżej 76% razem wykorzystane przez algorytm AdaBoost ich skuteczność sięga prawie 84 %.