1 Opis problemu

Celem algorytmu jest klasyfikacja czy pacjent, który doświadczył ataku serca przeżyje najbliższy rok.

2 Dane wejściowe

2.0.1 Opis

Zbiór danych przedstawia informacje o pacjętach, którzy w swoim życiu doświadczyli ataku serca. Liczba próbek: 132

2.0.2 Atrybuty

Dane wejściowe składały się z 13 atrybutów:

- survival - integer, liczba miesięcy, którą pacjęt przeżył od momentu ataku
- still-alive - boolean, 0 - pacjent nieżywy, 1 - pacjent żywy (pod koniec okresu z survival
- age-at-heart-attack - integer, wiek pacjenta podczas ataku serca
- pericardial-effusion - boolean, 1 - plyn obecny w okolicy serca, 0 - płyn nieobecny w okolicy serca
- fractional-shortening - float, miara kurczliwości serca
- epss - float, separacja przegrody serca w punkcie E, miara kurczliwosci serca
- lvdd - float, wymiar końcowo-rozkurczowy lewej komory. To jest miara wielkości serca na końcu rozkurczu.
- wall-motion-score - float, miara ruchu lewej przegrody
- wall-motion-index - wall-motion-score podzielona przez liczbe zauważonych segmentów.
- mult - zmienna do zignorowania
- name - string, imie pacjenta
- group - string, zmeinna do zignorowania
- alive-at-1 - boolean, oznaczenie czy pacjent przeżył przynajmniej rok

2.0.3 Brakujące dane

Liczba brakujących danych:

- survival - 2
- still-alive - 1
- age-at-heart-attack - 5
- pericardial-effusion - 1
- fractional - 8
- epss - 15
- lvdd - 11
- wall-motion-score - 4
- wall-motion-index - 1
- mult - 4
- name - 0
- group - 22
- alive-at-1 - 58

3 Model

3.1 Preprocessing

3.1.1 Pierwszy etap - usunięcie niepotrzebnych wartości

Z danych wejściowych odrzuciłem parametry name, group oraz mult, ponieważ nie wnoszą przydatnych informacji do naszego problemu. Odrzuciłem również parametr wall-motion-index, ponieważ użyłem miary wall-motion-score. Według opisu danych, powinno używać się tylko 1 z tych 2.

3.1.2 Drugi etap - uzupełnienie brakujących wartości

Brakujące dane w kolumnach: 'age-at-heart-attack', 'fractional-shortening', 'epss','lvdd','wall-motion-score' oraz 'pericardial-effusion' uzupełniłem medianą ze wszystkich wartości. Nasępnie za pomocą wartości w survival oraz still-alive uzupełniłem brakujące wartości w alive-at-1, oraz usunąłem pacjętów, którzy przetrwali mniej niż rok, ale wciąż są żywi.

3.1.3 Trzeci etap - usunięcie niepotrzebnych wartości

Finalnie usunąłem parametry survival oraz still-alive, które momgłyby sfałoszować skuteczność algorytmu. Przewidywanie czy ktoś przeżyje rok byłoby łatwiejsze wiedząc ile przeżył czasu od ataku. Po sprawdzeniu korelacji parametrów nie stwierdziłem obecności parametrów, które warto by było usunąć w celu zredukowania wymiarowości problemu.

Finalnie zostało 6 parametrów i 97 rekordów.

3.2 Wybrany model

Wybrany model to Support Vector Machine. Jest to model działający jak klasyfikator, którego nauka ma na celu wyznaczenie liniowej hiperpłaszczyzny rozdzielającej z maksymalnym marginesem wartości należące do dwóch klas. Problem z pozoru wydaje się prosty, lecz często nie ma jednego sposobu podziału, więc algorytm stara się wybrać najlepszy możliwy sposób. Algorytm znajduje "marginesy", czyli punkty (nazywane support vectors) leżące najbliżej takiej linii podziału z obu klas. Algorytm dąży do maksymalizacji odległości między tymi pomocniczymi wektorami. Dla bardziej skomplikowancyh problemów nie da się rozdzielić danych zwykłą prostą lub płaszczyzną w wymiarach danych wejściowych, stąd wprowadza się dodatkowe wymiary w celu umożliwienia takiego podziału, hiperpłaszczyzną.

3.3 Wybór zbioru testowego

Zbiór testowy wybrałem za pomocą metody train test split dzieląc 67% danych jako zbiór uczący i 33% danych jako zbiór testowy, jednocześnie dbając o rozkład klas wewnątrz zbioru treningowego oraz testowego. Nie wprowadziłem trójpodziału na zbiór treningowy, walidacyjny oraz testowy ze względu na niewystarczającą ilość rekordów.

3.4 Sposób wyboru hiperparametrów - grid search

Hiperparametry modelu wybrałem za pomocą metody Grid Search optymalizując model pod kątem metryki AUC. Metoda Grid Search buduje model dla każdej możliwej kombinacji zadanych hiperparametrów i wybiera najlepszy pod względem metryki przekazanej w parametrze score. Jest to dość efektywny sposób, lecz może on być bardzo kosztowny obliczeniowo. Optymalizowałem parametry: kernel (liniowy, czy rbf) oraz parametr C (mowiący jak bardzo chce uniknąć błędnego sklasyfikowania)