|  |  |  |  |
| --- | --- | --- | --- |
|  | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji,  Informatyki i Elektrotechniki  **Zakład Systemów Teleinformatycznych** | |  |
| **Przedmiot** | Drzewa decyzyjne Sleep Monitoring / EKG | | |
| **Prowadzący** | dr inż. Michał Kruczkowski | | |
| **Temat** | *Project / Lab :* Sleep Monitoring / EKG | | |
| **Student** | Cezary Tytko | | |
| **Ocena** |  | **Data oddania spr.** |  |

Celem było przewidywanie płci na zbiorze danych HYPOLY, zawierającym 278 pomiarów badania EKG pochodzących od Polaków miedzy 19 a 30 rokiem życia w tym 149 kobiet i 129 mężczyzn

Podczas próby rozwiązania problemu przeszedłem długą drogę, poczynając od działania na surowych danych, sieciach splotowych, kończąc na danych przetworzonych przez HRAexplorer.

Dla danych nie przetworzonych, pozostawianych w szeregu czasowym o długości 700 pomiarów, 5 parametrów modele ML jak drzewa decyzje, SVM, KNN wyniki były oscylowały w okolicy 50%, czyli były losowe dla dwóch klas, były one niestabilne i zależały od grupy która przypadła na zbiór walidacyjny, oraz modele wykazywały tendencję do OverFitingu do 100% dla drzewa decyzyjnego i SVM z jądrem „linear”

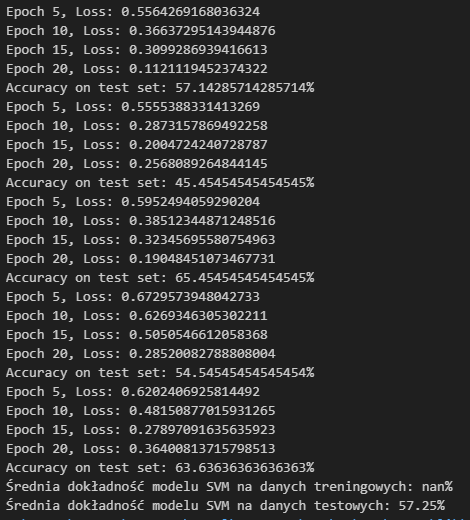
Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Dla Transfer Learningu i sieci resnet18 średnia z walidacji krzyżowej dla surowych danych wyniosła 57.25%



Po nadpróbkowaniu danych za pomocą ADASYN i zastosowaniu walidacji krzyżowej średni wynik był w okolicy 60%

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Po tym wyniku przetworzyłem dane za pomocą HRA Explorer uzyskując 50 cech dla każdego badania ekg, nie zależnie od czasu jego trwania

Te dane przetwarzałem głównie przez model SVM wykorzystując napisaną funkcje:

1. def crossValid(values, targets, splits: int, kernel: str, adasyn\_multiply: int=5):

2. skf = StratifiedKFold(n\_splits=splits, random\_state=42, shuffle=True)

3.

4. # Przeprowadź walidację krzyżową

5. train\_accuracies = []

6. test\_accuracies = []

7.

8. for train\_index, test\_index in skf.split(values, targets):

9. # Podział danych na zbiory treningowy i testowy

10. X\_train, X\_test = values[train\_index], values[test\_index]

11. y\_train, y\_test = targets[train\_index], targets[test\_index]

12.

13. # Obliczanie proporcji oversamplingu dla każdej klasy

14. unique\_classes, class\_counts = np.unique(y\_train, return\_counts=True)

15. count = np.max(class\_counts) \* adasyn\_multiply

16. target\_ratio = {cls: count for cls in unique\_classes}

17. print(f'target\_ratio: {target\_ratio}')

18. # Zastosowanie ADASYN do zwiększenia liczby danych tylko dla zbioru treningowego

19. adasyn = ADASYN(sampling\_strategy=target\_ratio, random\_state=42)

20. X\_train\_resampled, y\_train\_resampled = adasyn.fit\_resample(X\_train, y\_train)

22. # Trenowanie modelu klasyfikacji, np. klasyfikatora SVM

23. clf = svm.SVC(kernel=kernel)

24. clf.fit(X\_train\_resampled, y\_train\_resampled)

26. # Ocena dokładności na danych treningowych

27. train\_accuracy = clf.score(X\_train\_resampled, y\_train\_resampled)

28. train\_accuracies.append(train\_accuracy)

30. # Ocena dokładności na danych testowych

31. test\_accuracy = clf.score(X\_test, y\_test)

32. test\_accuracies.append(test\_accuracy)

33. print("Dokładność modelu SVM na danych treningowych: {:.2f}%".format(train\_accuracy \* 100))

34. print("Dokładność modelu SVM na danych testowych: {:.2f}%".format(test\_accuracy \* 100))

36. # Obliczenie średniej dokładności dla danych treningowych i testowych

37. avg\_train\_accuracy = np.mean(train\_accuracies)

38. avg\_test\_accuracy = np.mean(test\_accuracies)

39.

40. print("Średnia dokładność modelu SVM na danych treningowych: {:.2f}%".format(avg\_train\_accuracy \* 100))

41. print("Średnia dokładność modelu SVM na danych testowych: {:.2f}%".format(avg\_test\_accuracy \* 100))

42.

Po pierwszym uczeniu przeanalizowałem ważność cech, aby móc wyeliminować te najmniej istotne:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Powtórzyłem tą procedurę 3 razu (wyniki za każdym razem nieznacznie się poprawiały), odrzucając 20 kolumn (pozostało 50)

Ostatecznie najlepszy wynik jaki uzyskałem wyniósł 66.97%

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Wnioski:

Problem który z założenia nie wydawał się trudny, okazał się dla mnie nie do zrobienia z zadowalającym wynikiem. Uważam że 66.97% nie jest jak rzut monetą, jednak nie jest on zadowalający i nie można powiedzieć że problem został rozwiązany. Problemy jakie mógłbym wskazać, to mały zbiór danych, moim zdaniem dokładniejsze podejście do analizy danych, ekstrakcja lepszych cech i zastosowanie nadpróbkowania jest dobrym pomysłem na rozwiązanie tego problemu, z drugiej strony mając większy zbiór surowych danych, podejście z wykorzystaniem sieci neuronowych takich jak CNN, albo LSTM, może przynieść lepszy rezultat procentowy