**Sieci neuronowe i neurosterowniki**

**Projekt nr.1**

|  |  |
| --- | --- |
| **Prowadzący:**  Dr. Inż. Krzysztof Halawa | **Termin:**  Środa 13:15 TP |

**Autorzy:**

Armand Piecyk 235293

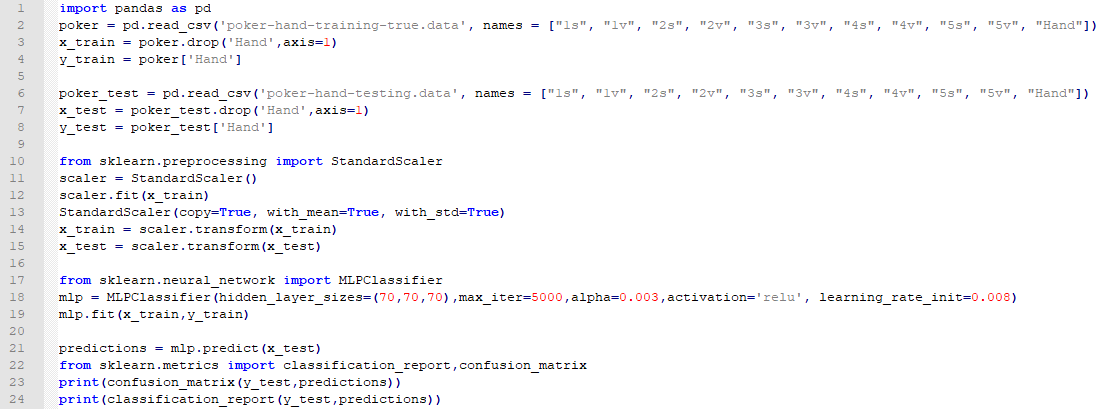
Radosław Bugiel 235118

1.Wstęp

Zadanie polegało na stworzeniu sieci neuronowej- wielowarstwowego perceptronu. Dane pobrano ze strony <https://archive.ics.uci.edu/ml/index.php>, oznaczone jako Poker Hand. Dane składają się z 11 atrybutów zawierających informację o pięciu kartach pobranych ze standardowego zestawu zawierającego 52 karty. Pierwsze 10 atrybutów opisuje kolejno kolor i figurę pięciu kart a ostatni atrybut opisuje układ pokerowy tworzony przez zestaw tych pięciu kart. Zestaw danych zawiera 25 010 instancji treningowych i 1 000 000 instancji testowych.

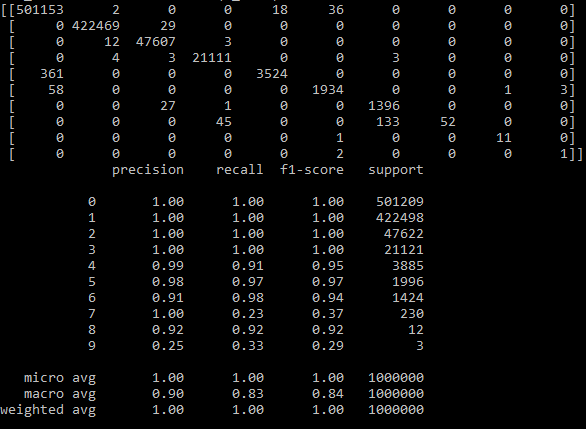
2.Program i testy

Do stworzenia sieci wykorzystano język Python i bibliotekę Scikit-learn**.**



*1.Kod programu w języku Python*

Głównym celem testów było uzyskanie zadowalającej precyzji ocenianych danych. Wyniki wahają się na granicy 70-80% poprawnej klasyfikacji. Na poniższym obrazku proszę zwrócić uwagę na kolumnę recall oraz wartość macro avg.



*2.Wyniki działania programu.*

Testy przeprowadzono poprzez zmianę lub dodawanie dodatkowych argumentów w funkcji MLPClassifier.

Zmiana argumentu :

1. **hidden\_layer\_sizes**

dla 2 warstw wyniki są w przedziale 60-80%

dla 3/4 warstw wyniki są takie same(80-90 %) dla wartości 40-80, zwiększanie nie powoduje zmian

1. **max\_iter**

dla wartości w granicach 250-15000 nie ma różnicy, z powodów technicznych nie można sprawdzić większych wartości.

1. **Alpha**

Dla alpha w przedziale 0.0001-0.03 wyniki są bardzo zbliżone w granicach 85% precyzji.

Dla wartości 0.04 i wyższych otrzymano błąd obliczeń.

1. **Learning\_rate\_init**

Wartości w przedziale 0.001-0.01 dają zbliżone wyniki w granicach 80-90%.

Po uruchomieniu programu z wartościa 0.0001 oraz 0.05 otrzymano 20% precyzji.

1. **Activation**

Sprawdzono wszystkie możliwości, czyli identity, logistic, tanh i relu z czego ostatni dawał najlepsze wyniki.

1. **Shuffle**

Domyślnie true. Zmiana wartości na false nie zmienia wyników.

1. **Epsilon**

Domyślnie 1e-8. Wartości w przedziale 1e-3 -- 1e-11 nie powodują widocznych zmian precyzji. Dla wartości 1e-1 otrzymano 20% precyzji, dla 2e-1 40% ,a dla 2e-8 86%.

3. Wnioski

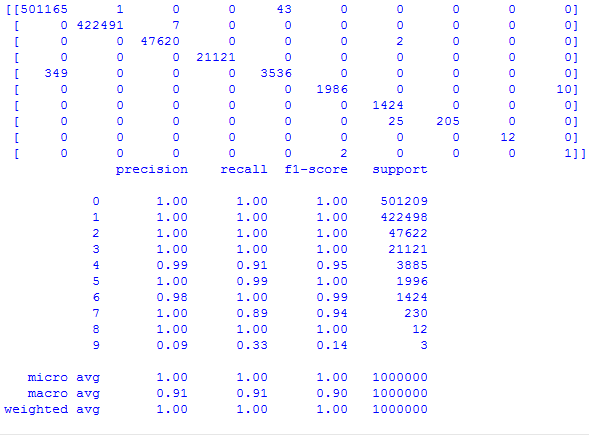
Przetestowano także argumenty tol,beta\_1 oraz **n\_iter\_no\_change i nie zauważono żadnych znaczących różnic precyzji. Precyzja na poziomie 80% < jest zadowalająca biorąc pod uwagę, że z większą ilością danych treningowych wynik byłby znacznie wyższy.**

**Komentarz**

**Informacje na temat interpretacji classification\_report I confiusion matrix. Recall jest zdaje się wyznacznikiem dokładności.**

[**https://stackoverflow.com/questions/30746460/how-to-interpret-scikits-learn-confusion-matrix-and-classification-report**](https://stackoverflow.com/questions/30746460/how-to-interpret-scikits-learn-confusion-matrix-and-classification-report)

Screen dla moich danych(tych z gita). Wygląda to lepiej choć nie zawsze.

****

Poprawiłem trochę, głównie wstęp. Najlepiej używać formy bezosobowej. Poprawiłbym wnioski i może o każdym teście coś więcej napisać.

Usiądę do tego jeszcze wieczorem.

Z drugiej strony Halawa i tak chyba się tym wszystkim nie przejmuje.