WSTĘP DO SZTUCZNEJ INTELIGENCJI

Ćwiczenie 4 – Regresja logistyczna

MICHAŁ MIZIA 331407

SPIS TREŚCI

Wstęp

Do policzenia metryk algorytmu użyta została biblioteka scikit-learn, a dokładniej funkcje accuracy_score, f1_score oraz roc_auc_score. Wykorzystanie stworzonego modelu regresji logistycznej wygląda następująco:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=420
)

model = LogisticRegression(learn_rate=0.05, iters=100)
model.fit(X_train, y_train)
predicted = model.predict(X_test)

print("Accuracy: ", accuracy_score(y_test, predicted))
print("F1: ", f1_score(y_test, predicted))
print("AUROC: ", roc_auc_score(y_test, predicted))
```

Funkcja kosztu użyta w modelu wygląda następująco:

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

A tak wygląda jej gradient:

$$J'(\theta) = \begin{bmatrix} \frac{dJ}{dw} \\ \frac{dJ}{db} \end{bmatrix} = [\dots] = \begin{bmatrix} \frac{1}{N}\sum 2x_i(\hat{y} - y_i) \\ \frac{1}{N}\sum 2(\hat{y} - y_i) \end{bmatrix}$$

Wszystkie wartości w zbiorze są liczbowe, ciągłe i bez brakujących danych a więc przygotowanie danych nie jest w stu procentach konieczne.

Użyte zostały funkcje F1 oraz Auroc.

F1 używa średniej harmonicznej precision oraz recall gdzie precision to stosunek true_positives/(true_positives + false_positives) a recall czyli czułość to true_positives/(true_positives + false_negatives).

AUROC to pole pod krzywą ROC (Receiving characteristic curve). Na osi OY jest recall a na osi OX false positive rate czyli false_positives/(false_positives + true_negatives). Obie wartości są znajdywane dla różnych progów decyzyjnych. Finalna wartość mieści się między 0 a 1 gdzie 0.5 to losowe zgadywanie.

Wyniki

Bez znormalizowania danych X, model napotyka błąd w funkcji sigmoid która używa np.exp, przez co wyniki funkcji nie są poprawne, co za tym idzie wyniki modelu również na tym cierpią.

```
RuntimeWarning: overflow encountered in exp
 return 1 / (1 + np.exp(-x))
```

```
Accuracy:  0.3986013986013986
F1:  0.5656565656565656
AUROC:  0.5057471264367817
```

- Dodanie znormalizowania danych OX dało dużo lepsze rezultaty. Dane są normalizowane przed podziałem na testowe i treningowe

```
Accuracy:  0.972027972027972
F1:  0.9636363636363636
AUROC:  0.9674671592775042
```

```
scaler = preprocessing.StandardScaler().fit(X)
X = scaler.transform(X)
```

- Usunięcie wszystkich kolumn zawierających w nazwie 1 z zachowaniem normalizacji prawie nie miało wpływu na wyniki pomiarów.

```
Accuracy:  0.965034965034965
F1:  0.954954954954955
AUROC:  0.9617200328407225
```

```
X.drop(
    [
        "radius1",
        "texture1",
        "perimeter1",
        "area1",
        "smoothness1",
        "compactness1",
        "concavity1",
        "concave_points1",
        "symmetry1",
        "fractal_dimension1",
    ],
    axis=1,
)
```

Wnioski

Normalizacja danych jest ważna przy trenowaniu modeli regresji logistycznej, bez niej model trenowany jest mniej wydajnie a jeśli dane są odpowiednio duże, może dojść do overflow funkcji exp co całkowicie psuje wyniki modelu.

Usunięcie niektórych kolumn nie miało wpływu na skuteczność przewidywań modelu, może to mieć związek z dużą liczbą kolumn danych które prawdopodobnie są ze sobą na tyle powiązane, że pozostałe kolumny są w stanie równie dobrze przewidzieć stopień raka pacjenta.