Sprawozdanie Lab 4

Jakub Kwaśniak 331396

1. Treść ćwiczenia
   - Zaimplementować algorytm regresji logistycznej.
   - Sprawdzić jakość działania algorytmu dla klasyfikacji na zbiorze danych Census Income https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic.
   - Policzyć wynik dla przynajmniej 3 różnych sposobów przygotowania danych, na przykład usuwając niektóre kolumny, dodając normalizację wartości.

2. Cel i opis eksperymentów oraz wyniki (do zdjęć dopisywany jest ostatni ruch gracza w rozgrywce)
   - Zaimplementowany został algorytm regresji logistycznej korzystający z cross entropy loss (funkcja straty)

Pochodna funkcji straty dla wag zmiennych:

Then we have the result

$$\frac{\partial}{\partial \boldsymbol{\beta}} L(\boldsymbol{\beta}) = X^T (\hat{Y} - Y).$$

(źródło: https://en.wikipedia.org/wiki/Cross-entropy)

Pochodna funkcji straty dla wyrazu wolnego:

$$\begin{aligned} \frac{\partial}{\partial \beta_0} L(\boldsymbol{\beta}) &= -\sum_{i=1}^{N} \left[\frac{y_i \cdot e^{-\beta_0 + k_0}}{1 + e^{-\beta_0 + k_0}} - (1 - y_i) \frac{1}{1 + e^{-\beta_0 + k_0}} \right] \\ &= -\sum_{i=1}^{N} [y_i - \hat{y}_i] = \sum_{i=1}^{N} (\hat{y}_i - y_i), \end{aligned}$$

   - Wyniki policzone zostały dla – w nawiasach etykiety przypadków w zestawieniu wyników:
     - Surowych danych z zbioru danych **Breast Cancer Wisconsin** **(rare data)**
     - Dancych znormalizowanych przy użyciu StandardScaler z sklearn.preprocessing **(lib_scaled)**
     - Danych przekształconych przy użyciu normalizacji min - max (własna implementacja) **(minMax_scaled)**
     - Danych z usuniętymi 10 (z 30) kolumnami i znormlizowanymi przez StandardScaler **(dropped_columns)**

- Użyte metryki (TP – true positive, FP – false positive, FN – false negative):
  - Accuracy – odsetek poprawnych klasyfikacji w stosunku do wszystkich klasyfikacji
  - F1_score = $2*\frac{precision*recall}{precision+recll}$
    - Precsion – TP/(TP + FP) – Parametr mówi jaki odsetek wszystkich pozytywnych wyników to wyniki dobrze sklasyfikowane (rzeczywiście pozytywne)
    - Recall – TP/(TP + FN) – parameter mówi ile przypadków „zgubiliśmy" podczas klasyfikacji, jak dużo przypadków pozytywnych zostało poprawnie przyporządkowanych ze wszystkich pozytywnych
  - AUROC - pole pod krzywą ROC, która przedstawia zależność między True Positive Rate (TPR) a False Positive Rate (FPR)
- Użyte normalizacje:
  - Min-max:
    - **z = (x - Min) / (Max - Min)**
    - z to wartość cechy po normalizacji
    - x to pierwotna wartość cechy
    - Min to wartości minimalna z całego zbioru wartości danej cechy
    - Max to wartość maksymalna z całego zbioru danej cechy
    - Normalizacja pozwala przeskalować wartości cechy do zbioru [0, 1], gdzie 0 to wartość minimalna w zbiorze a 1 to wartość maksymalna w zbiorze
  - StandardScaler (z sklearn.preprocessing):
    - $z=(x-\mu)/\sigma$
    - z to wartość cechy po normalizacji
    - x to pierwotna wartość cechy
    - μ to średnia wartość całego zbioru wartości danej cechy
    - σ to odchylenie standardowe całego zbioru danej cechy
    - przekształcony zbiór wartości danej cechy będzie miał rozkład o wartości średniej równej 0 i odchyleniu standardowym równym 1
- Wyniki (dla learning rate = 0.1, liczba kroków algorytmu = 100, próg odcięcia = 0.5):

```
Test data - rare data:
Accuracy: 0.392
F-score: 0.558
Auroc: 0.506
```

```
Test data - lib_scaled:
Accuracy: 0.958
F-score: 0.944
Auroc: 0.993
```

```
Test data - dropped_columns:
Accuracy: 0.944
F-score: 0.926
Auroc: 0.992
```

```
Test data - minMax_scaled:
Accuracy: 0.895
F-score: 0.845
Auroc: 0.966
```

- Najlepiej nie uruchamiać algorytmu na „surowych danych" – zgłaszany jest RuntimeWarning o przekroczeniu zakresu funkcji np.exp() – jednym z rozwiązań mogłoby być ustawienie typu danych na np.float128 ale wystarczy znormalizować wartości, wówczas również skuteczność algorytmu znacząco wzrasta

```
/Users/kwasus/Desktop/STUDIA/SEM3/WSI/wsi-jakub-kwasniak/lab4/logistic_regression.py:12: RuntimeWarning: overflow encountered in exp
  return 1 / (1 + np.exp(-x))
```

- Wykresy – algorytm otrzymał następujące wyniki i wykresy krzywej roc:

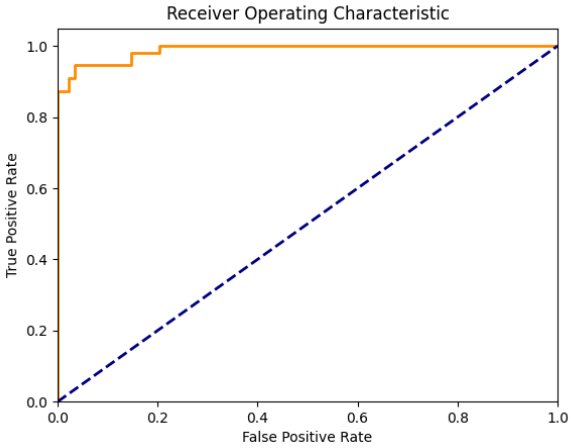
```
Test data - rare data:
Accuracy: 0.392
F-score: 0.558
Auroc: 0.506

Test data - lib_scaled:
Accuracy: 0.958
F-score: 0.944
Auroc: 0.993
```

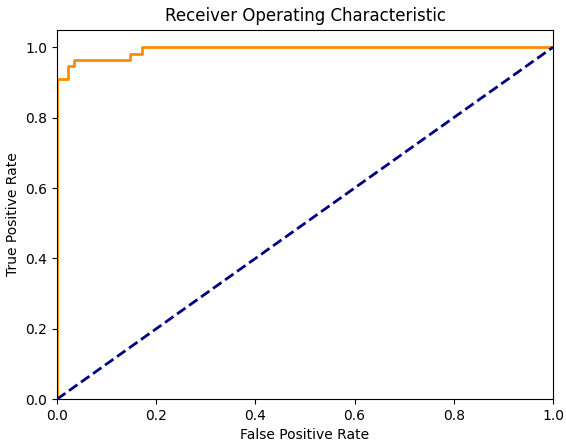
```
Test data - dropped_columns:
Accuracy: 0.951
F-score: 0.936
Auroc: 0.989

Test data - minMax_scaled:
Accuracy: 0.895
F-score: 0.845
Auroc: 0.966
```

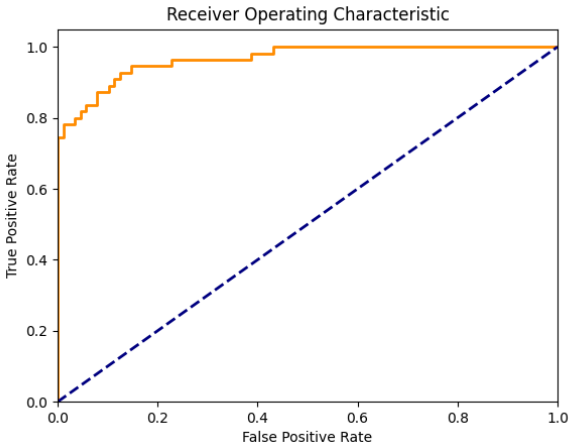
Dropped columns



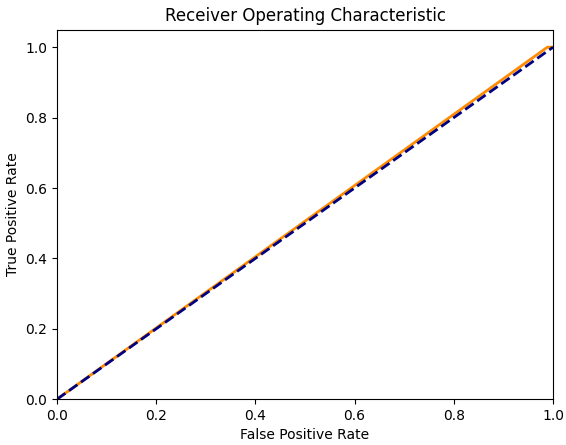
Lib scaled



minMax_scaled



Rare data



3. Instrukcja potrzebna do odtworzenia wyników (wraz z przygotowaniem środowiska, danych)

   Skrypt należy uruchomić komendą:

   - *python3 main.py -lr 0.1 -i 100 -th 0.5*

- *-lr / --learning-rate {float} -> wielkość współczynnika uczenia*
- *-I / --iters {int} -> liczba kroków algorytmu*
- *-th / --threshold {float} -> próg odcięcia dla alg. regresji logistycznej*
- *–seed {int} -> ziarno generatora liczb losowych (tu parametr potrzebny do podziału zbioru danych na zbiory uczące i testowy)*

4. Wnioski:
   - Najlepsze rezultaty klasyfikacji można osiągnąć dzięki skutecznej normalizacji danych z których tworzone są zbiory uczący i testowy
   - Algorytm regresji logistycznej jest bardzo skuteczny dla klasyfikacji binarnej - przy pracy na danych znormalizowanych osiąga wyniki metryk takich jak accuraccy, f1-score czy auroc na poziomie >80% (często blisko 100 %)
   - Krzywa Roc również pokazuje, że najlepsze wyniki osiągnąć można poprzez dobre znormalizowanie danych – wówczas nawet usunięcie części danych, na których trenujemy model jedynie minimalnie pogorszy wyniki