РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. П. Лумумба

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

Дисциплина: *Компьютерный практикум по моделированию*

Студент: Королёв Иван Андреевич

Группа: НКАбд-04-22

МОСКВА

2024 г.

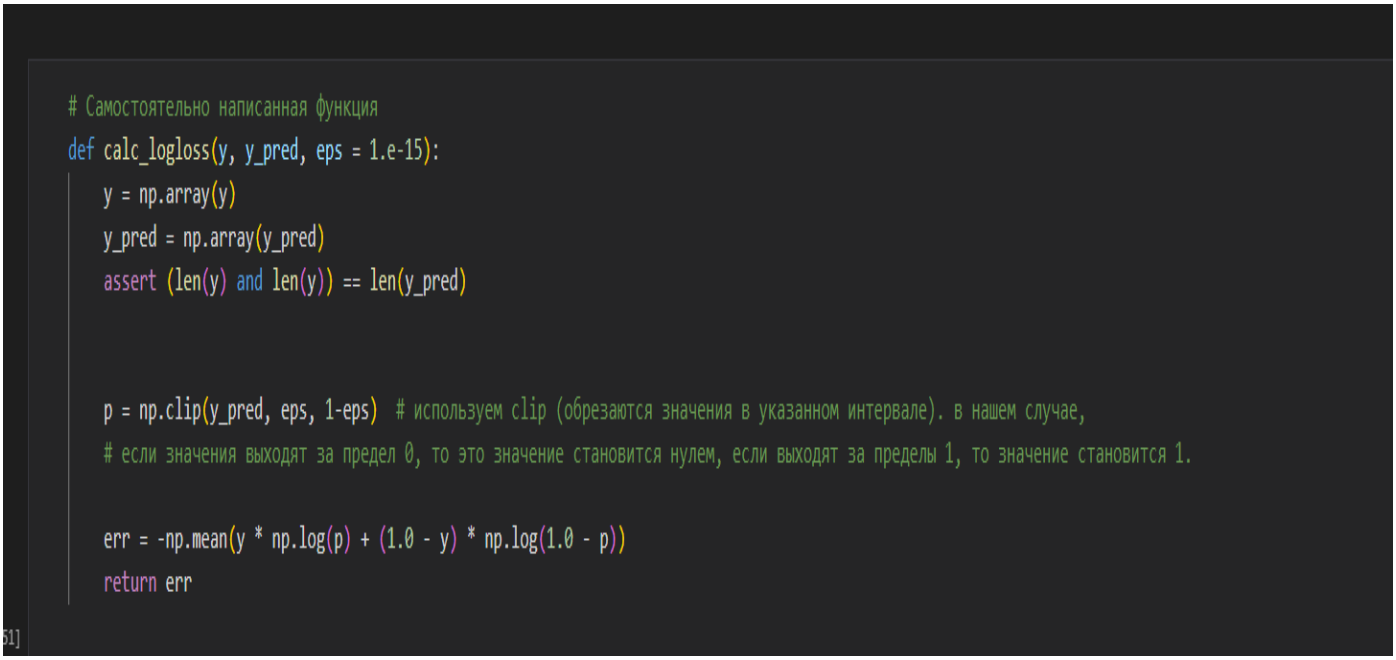
Цель работы: Логическая регрессия

Выполнение работы

Задание 1

Измените функцию calc_logloss так, чтобы нули по возможности не попадали в np.log.

Листинг программы на языке Python:



```
# Самостоятельно написанная функция
def calc_logloss(y, y_pred, eps = 1.e-15):
    y = np.array(y)
    y_pred = np.array(y_pred)
    assert (len(y) and len(y)) == len(y_pred)


    p = np.clip(y_pred, eps, 1-eps)  # используем clip (обрезаются значения в указанном интервале). в нашем случае,
    # если значения выходят за предел 0, то это значение становится нулем, если выходят за пределы 1, то значение становится 1.

    err = -np.mean(y * np.log(p) + (1.0 - y) * np.log(1.0 - p))
    return err
```

Задание 2.

Подберите аргументы функции eval_model для логистической регрессии таким образом, чтобы log loss был минимальным

Листинг программы на языке Python:

```python
def eval_model(X, y, iterations, eta=1e-4):
    np.random.seed(42)
    W = np.random.randn(X.shape[1])
    n = X.shape[0]

    best_err = float('inf')
    best_eta = None
    best_iter = None
    best_W = None

    for iter in iterations:
        for e in eta:
            for i in range(iter):
                z = np.dot(X, W)
                y_pred = sigmoid(z)
                err = calc_logloss(y, y_pred)

                dQ = 1/n * X.T @ (y_pred - y)
                W -= e * dQ
                # if i % (iterations / 10) == 0:
                    # print(i, W, err)
            if err < best_err:
                best_err = err
                best_eta = e
                best_iter = iter
                best_W = W

    print(f'Количество итераций: = {best_iter}, \nСкорость обучения = {best_eta}, \nОшибка = {best_err}')
    return best_W
```

Задание 3.

Создайте функцию calc_pred_proba, возвращающую предсказанную вероятность класса 1 (на вход подаются W, который уже посчитан функцией eval_model и X, на выходе - массив y_pred_proba).

Листинг программы на языке Python:

```
def calc_pred_proba(W, X):

    m = X.shape[0]

    y_pred_proba = np.zeros(m)

    A = np.squeeze(sigmoid(np.dot(X, W)))

    for i in range(A.shape[0]):
        if (A[i] > 0.5):
            y_pred_proba[i] = 1
        elif (A[i] <= 0.5):
            y_pred_proba[i] = 0

    return y_pred_proba
```

Задание 4.

Создайте функцию calc_pred, возвращающую предсказанный класс (на вход подаются W, который уже посчитан функцией eval_model и X, на выходе - массив y_pred)

Листинг программы на языке Python:

```
def calc_pred(W, X):

    y_pred_class = 0

    m = X.shape[0]

    y_pred_proba = np.zeros(m)

    A = np.squeeze(sigmoid(np.dot(X, W)))

    for i in range(A.shape[0]):
        if (A[i] > 0.5):
            y_pred_proba[i] = 1
        elif (A[i] <= 0.5):
            y_pred_proba[i] = 0
    for i in y_pred_proba:
        if i >= 0.5:
            y_pred_class = 1
        else:
            y_pred_class = 0


    print(f'Класс: {y_pred_class}')
```

Задание 5.

Реализуйте функции для подсчета Accuracy, матрицы ошибок, точности и полноты, а также F1 score.

Листинг программы на языке Python:

```python
import pandas as pd

def confusion_matrix(y, y_pred):

    y = pd.Series (y, name='Actual')
    y_pred = pd.Series(y_pred, name='Predicted')
    result = pd.crosstab(y, y_pred)

    print(result, '\n')

def metrics_calc(y, y_pred):
    true_positive = np.logical_and(y_pred, y)  # и в прогнозе, и в реальности было "да"
    false_positive = np.logical_and(y_pred, np.logical_not(y))  # прогноз сказал "да", а в реальности было "нет"
    true_negative = np.logical_and(np.logical_not(y_pred), np.logical_not(y))  # прогноз – "нет" и он прав
    false_negative = np.logical_and(np.logical_not(y_pred), y)  # прогноз – "нет" и ошибся

    tp, fp, tn, fn = (x.sum() for x in (true_positive, false_positive, true_negative, false_negative))

    accuracy = (tp + tn) / (tp + fp + fn + tn)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
    f1_score = 2 * (recall * precision) / (recall + precision)

    print(f"Accuracy: {accuracy}, \nPrecision: {precision}, \nRecall: {recall}, \nF1_score: {f1_score}.")
```