## Front matter

title: "Отчёт по лабораторной работе №2" subtitle: "1 вариант" author: "Тимур Ринатович Каримов"

## Generic otions

lang: ru-RU toc-title: "Содержание"

## Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt

## I18n polyglossia

polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs: name: english

## I18n babel

babel-lang: russian babel-otherlangs: english

## Fonts

mainfont: IBM Plex Serif romanfont: IBM Plex Serif sansfont: IBM Plex Sans monofont: IBM Plex Mono mathfont: STIX Two Math mainfontoptions: Ligatures=Common,Ligatures=TeX,Scale=0.94 romanfontoptions: Ligatures=Common,Ligatures=TeX,Scale=0.94 sansfontoptions: Ligatures=Common,Ligatures=TeX,Scale=MatchLowercase,Scale=0.94 monofontoptions: Scale=MatchLowercase,Scale=0.94,FakeStretch=0.9 mathfontoptions:

## Biblatex

biblatex: true biblio-style: "gost-numeric" biblatexoptions:

- parentracker=true
- backend=biber
- hyperref=auto
- language=auto
- autolang=other*
- citestyle=gost-numeric

## Pandoc-crossref LaTeX customization

figureTitle: "Рис." tableTitle: "Таблица" listingTitle: "Листинг" lofTitle: "Список иллюстраций" lotTitle: "Список таблиц" lolTitle: "Листинги"

## Misc options

indent: true header-includes:

- \usepackage{indentfirst}
- \usepackage{float} # keep figures where there are in the text
- \floatplacement{figure}{H} # keep figures where there are in the text

---

# Цель работы

Умение технически реализовывать стохастический градиентный спуск, его отличие от наивного. Понимание оптимальных случаев для применения масштабирования. Изучение нового метода для оптимизации обучения - регуляризации.

# Задание

1. Изменить функцию calc_logloss так, чтобы нули по возможности не попадали в np.log
2. Подобрать аргументы функции eval_model для логистической регрессии таким образом, чтобы log loss был минимальным
3. Создать функцию calc_pred_proba, возвращающую предсказанную вероятность класса 1
4. Создать функцию calc_pred, возвращающую предсказанный класс
5. Реализовать функции для подсчета Accuracy, матрицы ошибок, точности и полноты, а также F1 score

# Выполнение лабораторной работы

## Задание 1

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

**Решение:** Задача состояла в том, чтобы функция попадала под область определения, в которых логарифм не имеет конечное значение равное const. Для этого необходимо ограничить наш y_pred.

`y_pred = np.clip(y_pred, epsilon, 1 - epsilon)`. Работа команды clip - приравнивает значения больше максимума к максимуму.

**Код:**

```
def calc_logloss(y, y_pred, epsilon=1e-15):
    y_pred = np.clip(y_pred, epsilon, 1 - epsilon) # ставим диапазон возможных
чисел
    err = -np.mean(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))
    return err
```

## Задание 2

**Формулировка:** Подберите аргументы функции eval_model для логистической регрессии таким образом, чтобы log loss был минимальным.

**Решение:** Я воспользовался циклом, который будет подбирать количество итераций и скорость обучения (eta) до тех пор, пока log loss не перестанет уменьшаться.

**Код:**

```
def find_min_logloss(X, y, max_iterations=100000, learning_rates=[0.0001, 0.001,
0.01, 0.1]):
    np.random.seed(42)
    best_W = None
    best_loss = float('inf')
    best_eta = None
    best_iterations = 0
    history = []

    for eta in learning_rates:
        W = np.random.randn(X.shape[1])
        losses = []
        for i in range(max_iterations):
            z = np.dot(X, W)
            y_pred = sigmoid(z)
            loss = calc_logloss(y, y_pred)
            losses.append(loss)
            gradient = 1/len(X) * X.T @ (y_pred - y)
            W -= eta * gradient

            if loss < best_loss:
                best_loss = loss
                best_W = W.copy()
                best_eta = eta
                best_iterations = i + 1

            # Остановка
            if i > 1000 and abs(losses[-1] - losses[-100]) < 1e-8:
                break

        history.append({
            'eta': eta,
            'losses': losses,
            'final_loss': losses[-1]
        })

    # Визуализация
    plt.figure(figsize=(15, 5))

    plt.subplot(1, 3, 1)
    for hist in history:
        plt.plot(hist['losses'][:1000], label=f'eta={hist["eta"]}')
    plt.xlabel('Итерации')
    plt.ylabel('Log Loss')
```

```
    plt.legend()
    plt.title('Сходимость по итерациям')
    plt.yscale('log')

    plt.subplot(1, 3, 2)
    etas = [hist['eta'] for hist in history]
    final_losses = [hist['final_loss'] for hist in history]
    plt.plot(etas, final_losses, 'o-')
    plt.xscale('log')
    plt.xlabel('Learning Rate (eta)')
    plt.ylabel('Final Log Loss')
    plt.title('Зависимость loss от learning rate')

    plt.subplot(1, 3, 3)
    z_best = np.dot(X, best_W)
    y_pred_best = sigmoid(z_best)
    predictions = (y_pred_best > 0.5).astype(int)
    accuracy = np.mean(predictions == y)
    plt.bar(['Accuracy'], [accuracy], color='green' if accuracy > 0.9 else
'orange')
    plt.ylim(0, 1)
    plt.title(f'Точность: {accuracy:.1%}')

    plt.tight_layout()
    plt.show()

    return best_W, best_loss, best_eta, best_iterations

best_W, best_loss, best_eta, best_iterations = find_min_logloss(X, y)

print(f"Минимальный Log Loss: {best_loss:.6f}")
print(f"Лучший learning rate: {best_eta}")
print(f"Достигнут за {best_iterations} итераций")
print(f"Оптимальные веса: {best_W}")
```

**График:**

![График сходимости](image/Pasted image 20251016001048.png){#fig:001 width=70%}

## Задание 3

**Формулировка:** 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).

**Решение:**

```
def calc_pred_proba(W, X):
    z = np.dot(X, W)
    return sigmoid(z)
```

Так как функция `calc_pred_proba` совпадает с сигмоидой потому, что **в логистической регрессии сигмоидная функция как раз и вычисляет вероятность принадлежности к классу 1**.

## Задание 4

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

**Решение:**

```
def calc_pred(W, X, porog=0.5):
    y_pred_proba = calc_pred_proba(W, X)
    y_pred = (y_pred_proba >= porog).astype(np.float64) # Для того, чтобы тип был
не bool
    return y_pred
```

В общем, порог классификации - это значение, от которого мы отталкиваемся, чтобы решить, к какому классу отнести объект. В связи, с чем можно задать классификацию с помощью условия.

## Задание 5

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

**Код:**

```
def accuracy_score(y, y_pred):
    correct = np.sum(y == y_pred)
    total = len(y)
    return correct / total

def confusion_matrix(y, y_pred):
    tp = np.sum((y == 1) & (y_pred == 1))
    tn = np.sum((y == 0) & (y_pred == 0))
    fp = np.sum((y == 0) & (y_pred == 1))
    fn = np.sum((y == 1) & (y_pred == 0))
    return np.array([[tn, fp], [fn, tp]])

def precision_score(y, y_pred):
    cm = confusion_matrix(y, y_pred)
    tp = cm[1, 1]
    fp = cm[0, 1]
    if tp + fp == 0:
        return 0.0
    return tp / (tp + fp)

def recall_score(y, y_pred):
    cm = confusion_matrix(y, y_pred)
    tp = cm[1, 1]
    fn = cm[1, 0]
```

```
    if tp + fn == 0:
        return 0.0
    return tp / (tp + fn)

def f1_score(y, y_pred):
    precision = precision_score(y, y_pred)
    recall = recall_score(y, y_pred)
    if precision + recall == 0:
        return 0.0
    return 2 * (precision * recall) / (precision + recall)

if __name__ == "__main__":
    y = np.array([1, 0, 1, 1, 0, 1, 0])
    y_pred = np.array([1, 0, 0, 1, 0, 1, 1])
    print("Accuracy:", accuracy_score(y, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y, y_pred))
    print("Precision:", precision_score(y, y_pred))
    print("Recall:", recall_score(y, y_pred))
    print("F1-score:", f1_score(y, y_pred))
```

- F1-мера — это гармоническое среднее точности и полноты.

**Вывод:**

![Результаты метрик](image/Pasted image 20251016003928.png){#fig:002 width=70%}

# Выводы

В ходе выполнения лабораторной работы были успешно реализованы и протестированы основные функции для работы с логистической регрессией. Были получены навыки оптимизации параметров модели, работы с функциями потерь и оценки качества классификации с помощью различных метрик.

# Список литературы{.unnumbered}