



Санкт-Петербургский  
государственный университет

# Распределённое машинное обучение (YADML)

- Осадчев Т.В, Масгутов М.А, Крылов А.С

# Цель и задачи проекта

## Цель:

Разработка распределённой системы машинного обучения для совместного обучения модели на нескольких узлах.

## Основные задачи:

- Обеспечить GUI для управления процессом.
- Создать web-сервер с REST API для управления воркерами и заданиями.
- Реализовать логику мастер-воркер взаимодействия.

## Распределение обязанностей:

- Осадчев Тимофей – (ML + Backend)
- Крылов Александр – Админ-клиент (Front + Back on Qt)
- Масгутов Михаил – Серверная часть (Backend - DevOps)



# Постановка Проблемы

Современные задачи классификации изображений требуют высокой точности и скорости обработки. Основные проблемы, которые мы стремимся решить:

- Точность Классификации: Повышение точности модели для минимизации ошибок предсказаний.
- Скорость Обработки: Ускорение процесса предсказаний для реального времени.
- Масштабируемость: Обеспечение возможности обработки больших объёмов данных.
- Удобство Использования: Создание интуитивно понятного интерфейса для конечных пользователей.



# Общая архитектура проекта

GUI админа для управления процессом обучения.

Web-сервер (REST API) для связи с мастер-нодой и воркерами.

Мастер-нода:

- Разделение датасета.
- Отправка данных воркерам по SSH.
- Сбор результатов и объединение модели.
- Повторная отправка обновлённых частей.

Воркер-ноды тренируют модель на части данных.

# Что такое MNIST?

MNIST (Modified National Institute of Standards and Technology):

Набор данных для задач классификации изображений.

Содержит 70,000 рукописных цифр (60,000 для обучения и 10,000 для тестирования).

Каждое изображение — 28×28 пикселей в градациях серого.

Используется для тестирования алгоритмов машинного обучения.

label = 5



label = 0



label = 4



label = 1



label = 9



label = 2



label = 1



label = 3



label = 1



label = 4



label = 3



label = 5



label = 3



label = 6



label = 1



label = 7



label = 2



label = 8



label = 6



label = 9



# Обзор Данных

- Количество Изображений: по 70,000 (60,000 для обучения и 10,000 для тестирования).
- Формат Изображений: Серые изображения размером 28x28 пикселей.
- Классы: 10 классов, представляющих цифры от 0 до 9 и виды одежды.
- Предварительная Обработка:
  - Нормализация пикселей до диапазона [0,1].

# Машинное Обучение (ML) - Архитектура Модели

```
class LeNet(nn.Module):  
    def __init__(self):  
        super(LeNet, self).__init__()  
        self.conv_layers = nn.Sequential(  
            nn.Conv2d(in_channels: 1, out_channels: 6, kernel_size=5, padding="same"),  
            nn.ReLU(),  
            nn.MaxPool2d(2),  
            nn.Conv2d(in_channels: 6, out_channels: 16, kernel_size=5),  
            nn.ReLU(),  
            nn.MaxPool2d(2),  
            nn.Conv2d(in_channels: 16, out_channels: 120, kernel_size=5),  
            nn.ReLU(),  
        )  
        self.fc_layers = nn.Sequential(  
            nn.Linear(in_features: 120, out_features: 84),  
            nn.ReLU(),  
            nn.Linear(in_features: 84, out_features: 10),  
            nn.LogSoftmax(dim=-1),  
        )
```

# Машинное Обучение (ML) - Процесс Обучения

Ключевые параметры обучения модели:

- Разделение Данных: 80% для обучения, 20% для валидации.
- Гиперпараметры:
  - **Оптимизатор**: Adam
  - **Функция Потерь**: Кросс-энтропия
  - **Скорость Обучения**: 0.001
  - **Количество Эпох**: 20
  - **Размер Батча**: 64
- **Метрики для Отслеживания**:
  - Точность (Accuracy)
  - Потери (Loss)

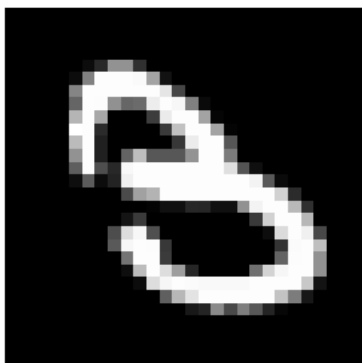


# MNIST

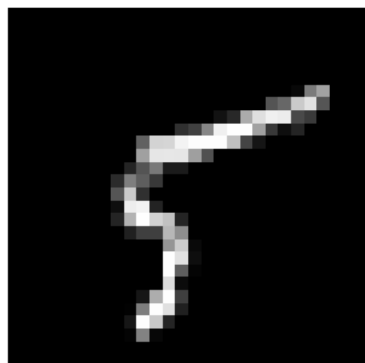
Результаты оценки набора MNIST:

- **\*\*Точность (Accuracy)\*\*: 0.9443**
- **\*\*Precision (Macro)\*\*: 0.9485**
- **\*\*Recall (Macro)\*\*: 0.9436**
- **\*\*F1-Score (Macro)\*\*: 0.9440**
- **\*\*ROC AUC (Macro)\*\*: 0.9983**
- **\*\*Average Precision (Macro)\*\*: 0.9803**

True: 3  
Pred: 8



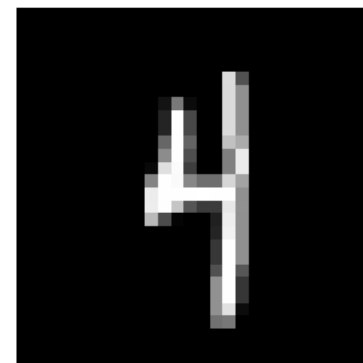
True: 5  
Pred: 7



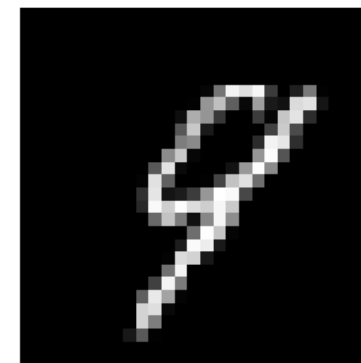
True: 8  
Pred: 2

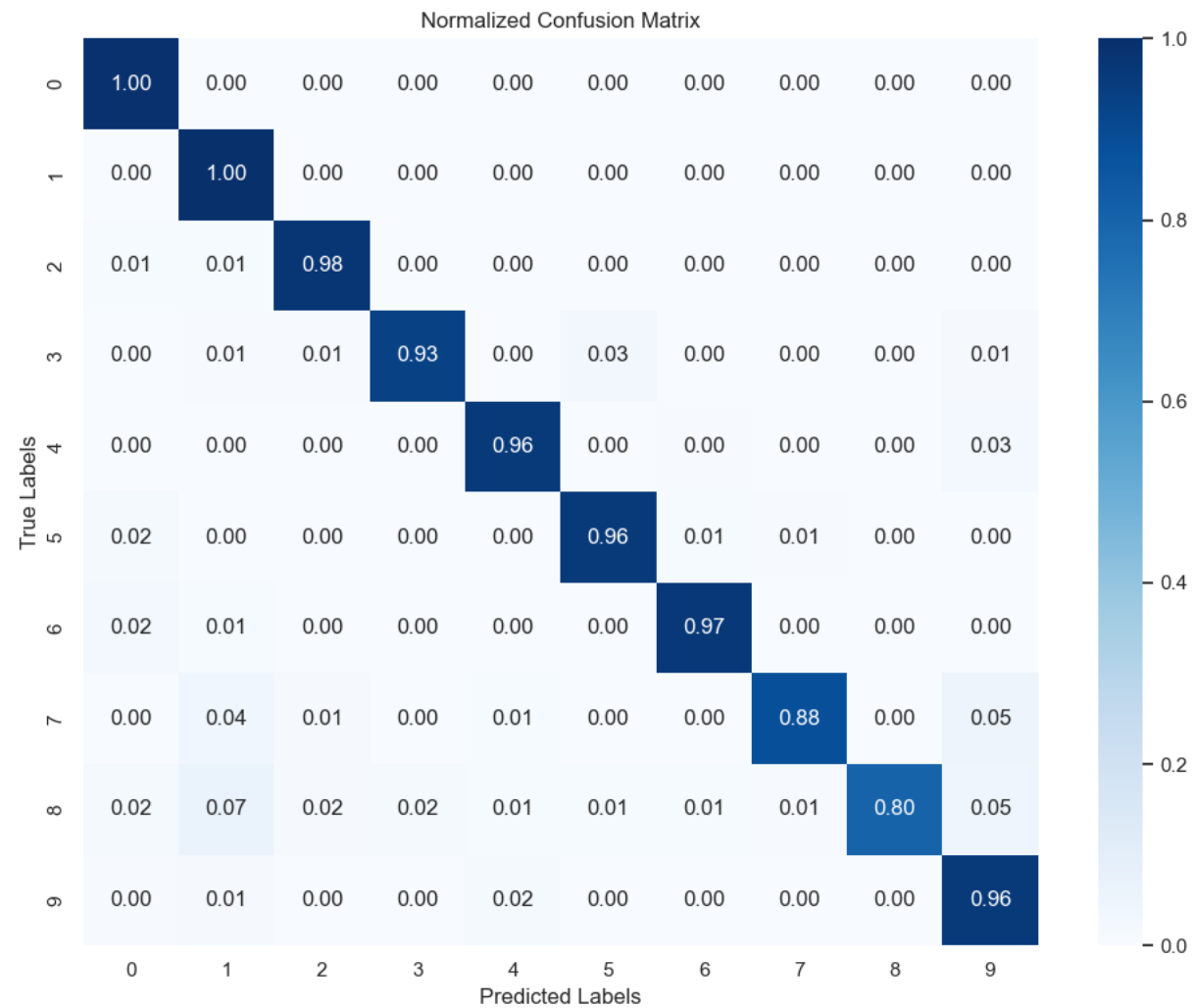
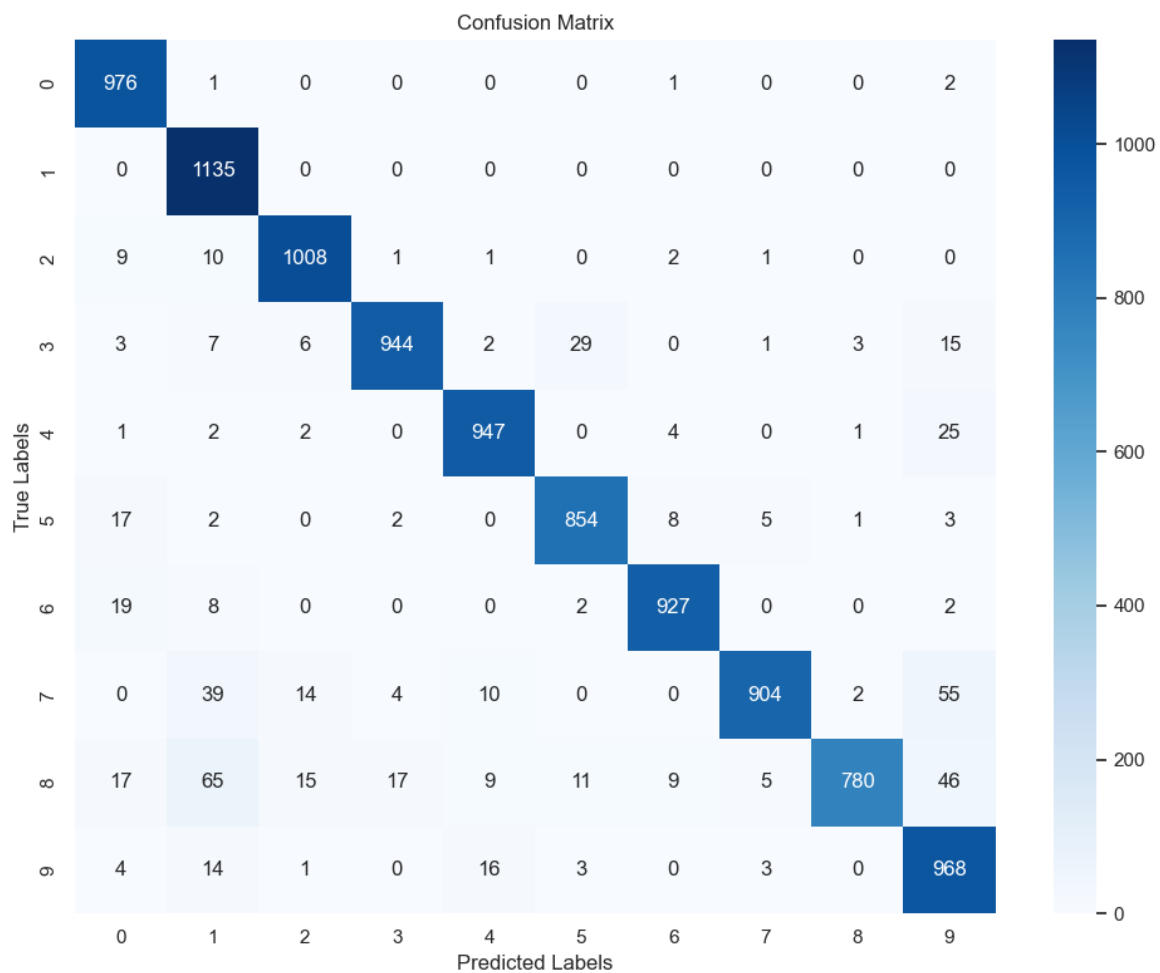


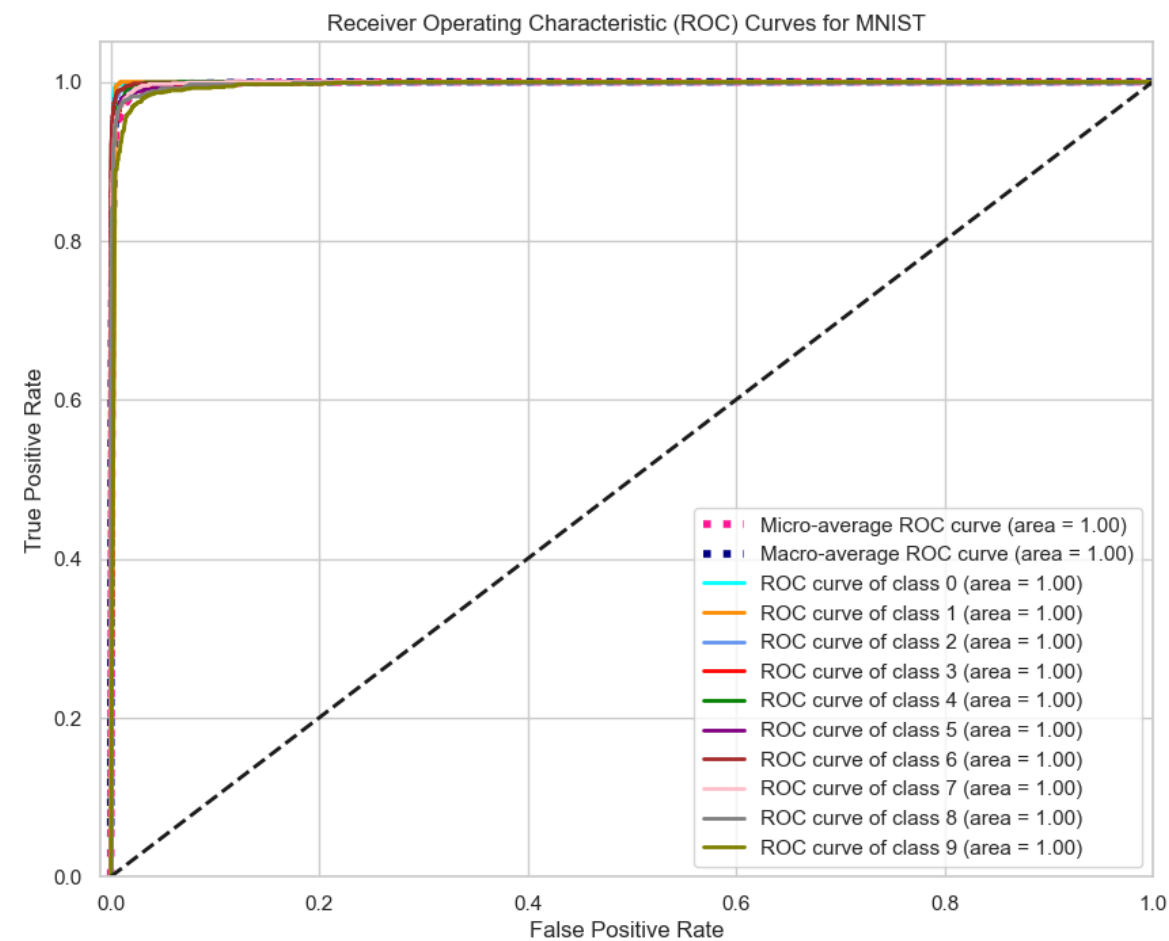
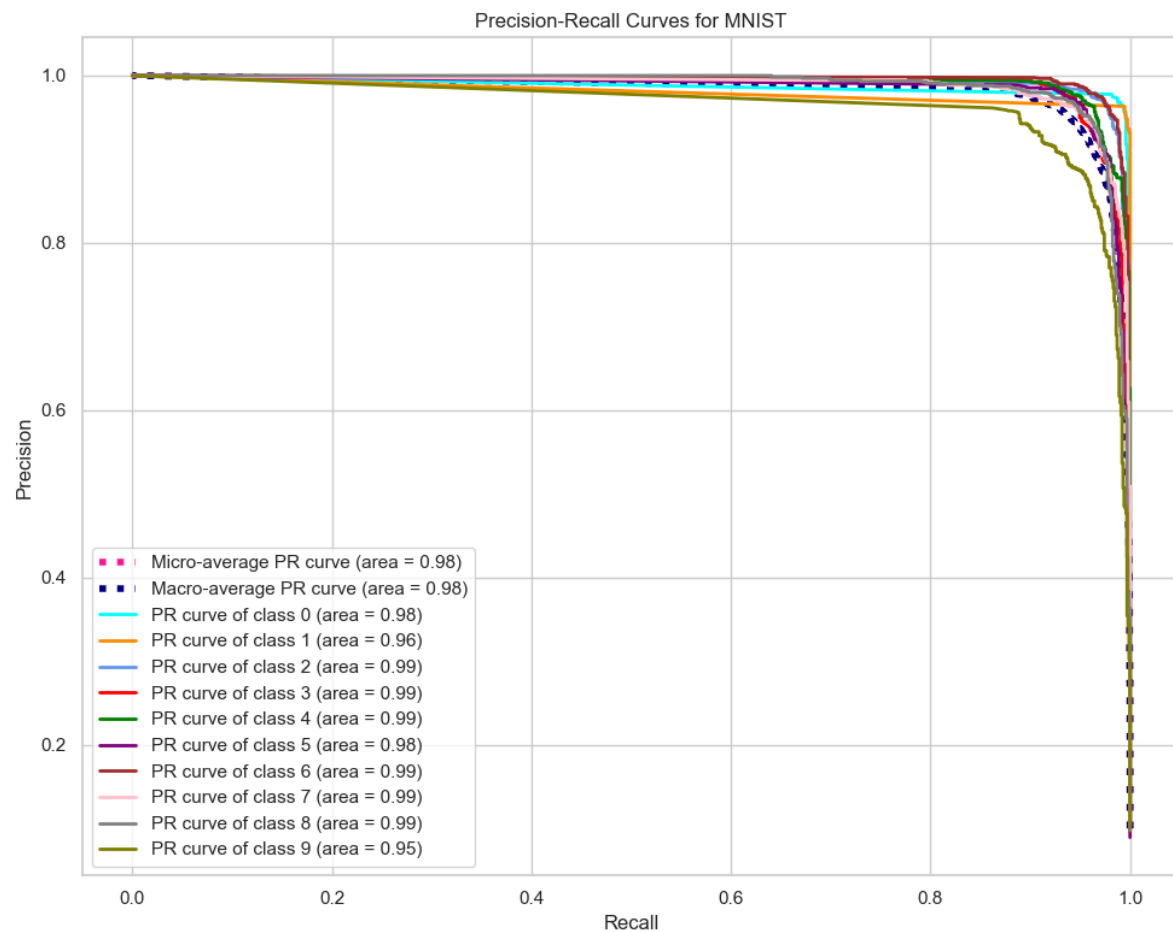
True: 4  
Pred: 9

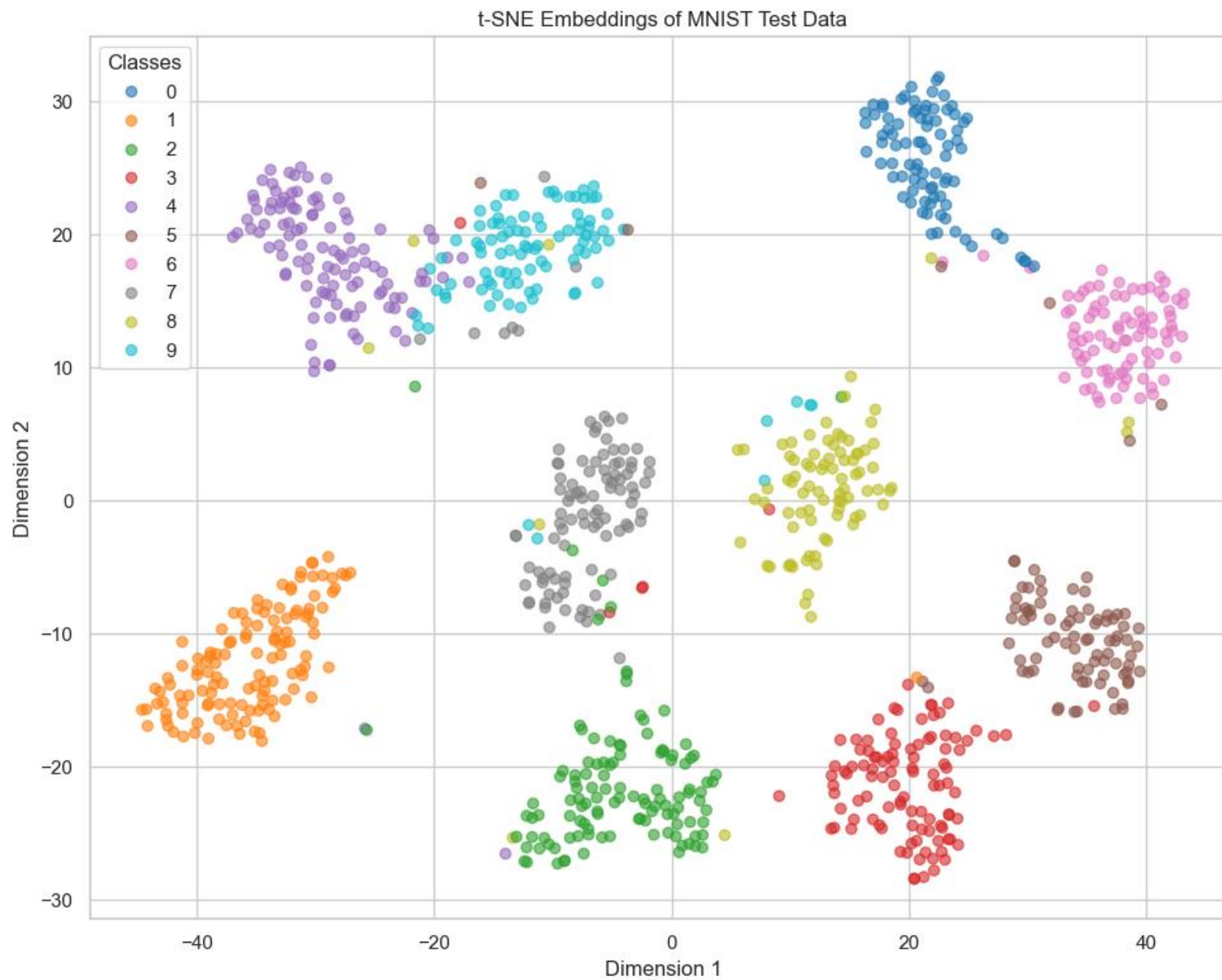


True: 9  
Pred: 4









# Fashion-MNIST

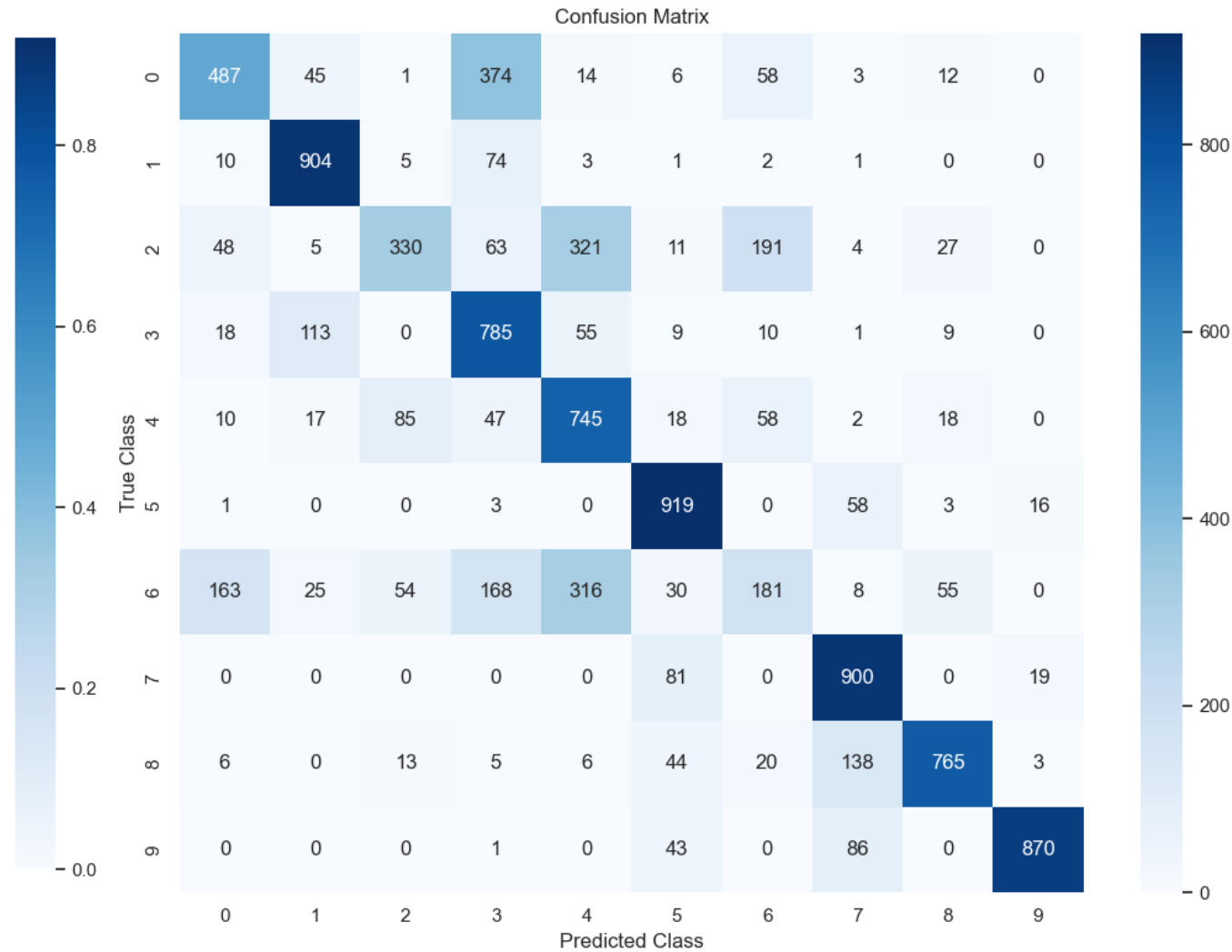
Результаты оценки набора

Fashion-MNIST:

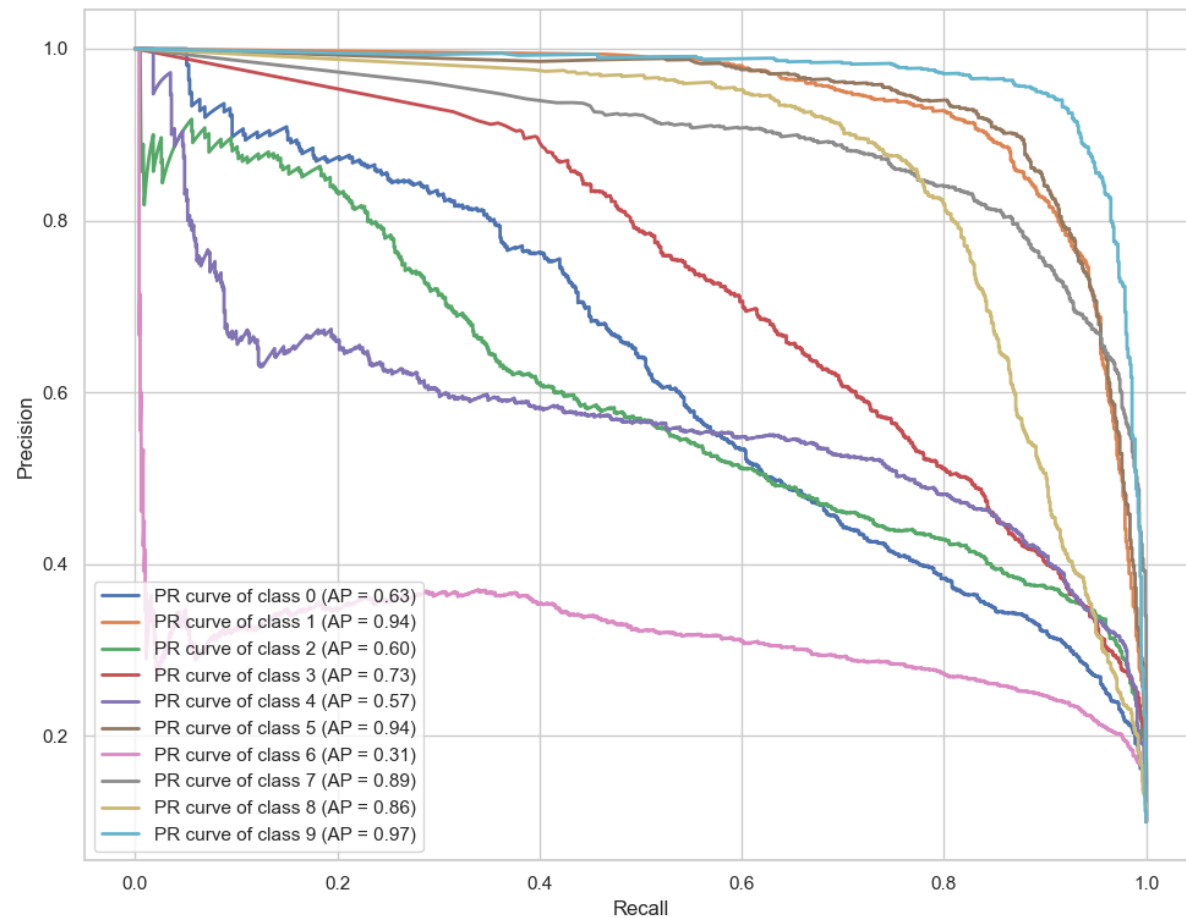
- **\*\*Точность (Accuracy)\*\*:** 0.6886
- **\*\*Precision (Macro)\*\*:** 0.6881
- **\*\*Recall (Macro)\*\*:** 0.6886
- **\*\*F1-Score (Macro)\*\*:** 0.6716
- **\*\*ROC AUC (Macro)\*\*:** 0.9527
- **\*\*Average Precision (Macro)\*\*:** 0.7432



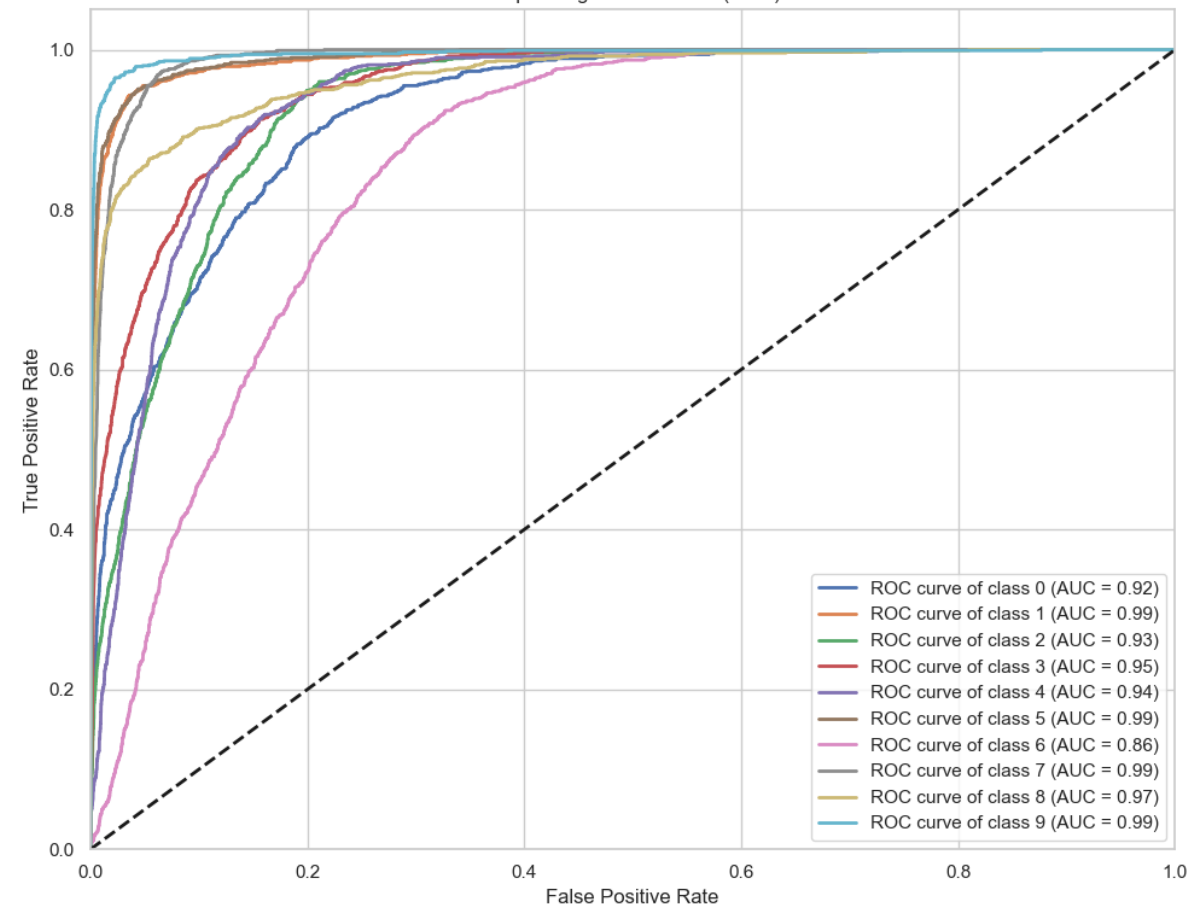


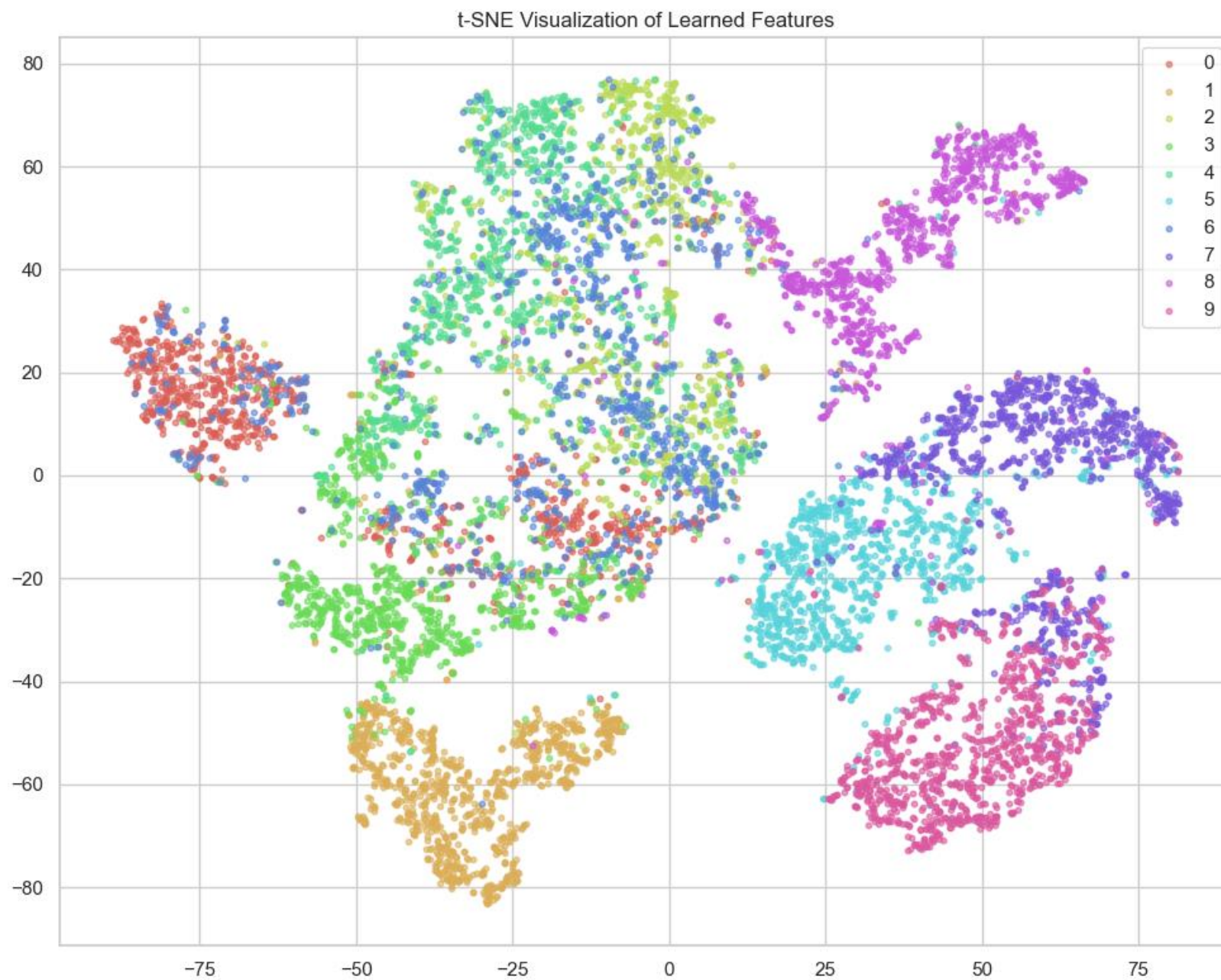


Precision-Recall Curves



Receiver Operating Characteristic (ROC) Curves





# Результаты и Выводы

## **\*\*Достигнутые Результаты:\*\***

- Высокая точность классификации изображений.
- Эффективная работа модели на различных классах с минимальными ошибками.

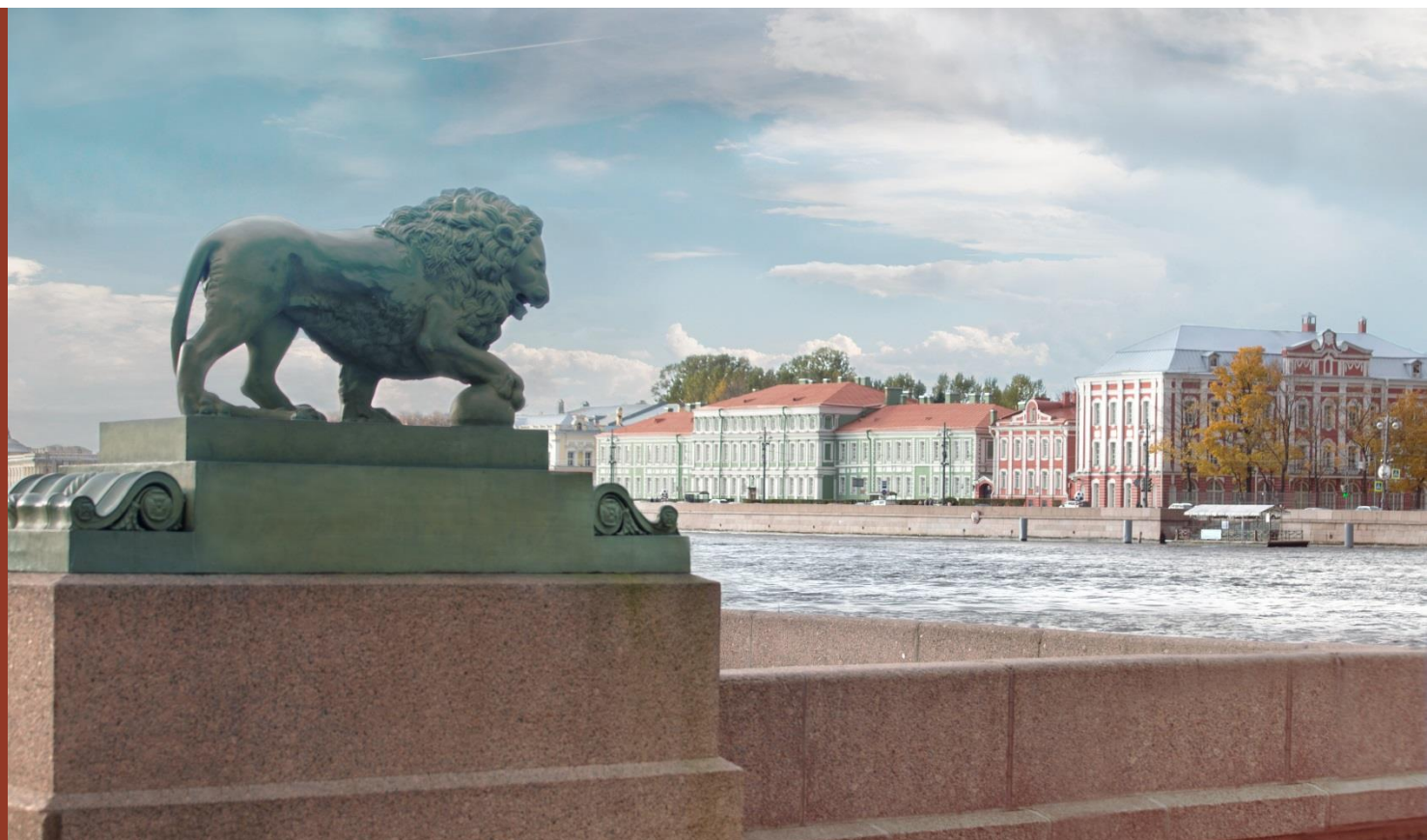
## **\*\*Направления для Улучшения:\*\***

- Внедрение более сложных архитектур (например, ResNet) для повышения точности.
- Использование методов регуляризации для ещё большего уменьшения переобучения.
- Расширение набора данных для повышения обобщающей способности модели.
- Тестирование на других наборах данных.



Санкт-Петербургский  
государственный университет

## Блок 2: Распределённые Вычисления





# Необходимость Распределённых Вычислений в Проекте

- Причины Внедрения:

- Большие Объёмы Данных: Обработка и хранение множества изображений.
- Скорость Обработки: Необходимость быстрого обучения и предсказаний.
- Масштабируемость: Возможность увеличения ресурсов по мере роста данных.

- Цели:

- Оптимизация процесса обучения модели.
- Ускорение времени предсказаний для реального времени.
- Обеспечение устойчивости и доступности системы.

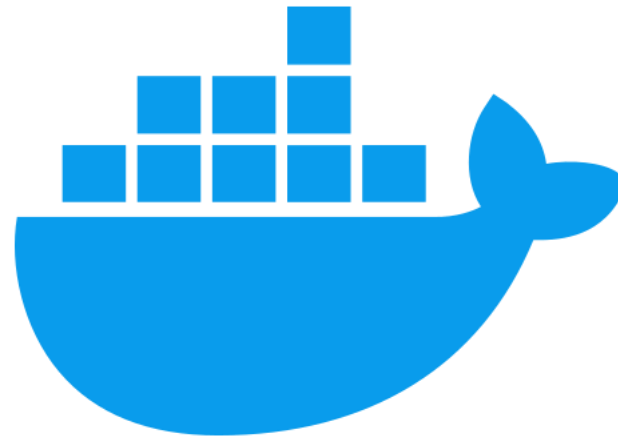
# Архитектура системы

## Центральная нода:

- Сервер на **FastAPI**, авторизация через единственного администратора.
- Главная задача — управление воркерами, координация выполнения задач.

## Воркеры:

- Работают в **Docker-контейнерах**, каждый из которых изолирован.
- На каждом контейнере запущен **OpenSSH сервер** и размещены необходимые Python-скрипты.
- Выполняют задачи независимо, поддерживая устойчивость системы.



# Авторизация и веб-интерфейс

## Авторизация:

- Главная нода допускает доступ только для администратора.
- Администратор управляет процессами через веб-интерфейс.

## Функции веб-интерфейса:

1. Запуск скрипта обучения.
2. Выбор метода разбиения датасета.
3. Контроль конфликта процессов (гарантия, что два обучения не запускаются одновременно).

# Работа главной ноды

Разбиение датасета:

- Датасет разделяется на несколько частей выбранным пользователем методом.
- Каждая часть передаётся на выбранные воркеры.

Управление процессами:

- Асинхронный запуск обучения на всех выбранных воркерах.
- Сбор результатов обучения с каждого воркера.

Агрегация:

- Главная нода объединяет локальные модели из воркеров в единую глобальную модель.
- Финальная модель отправляется обратно на воркеры.

# Устройство воркеров

## Docker-контейнеры:

- Полная изоляция каждого воркера.
- Скрипты для обучения и OpenSSH-сервер предустановлены.

## Работа:

- Получение задачи от главной ноды через SSH.
- Обучение локальной модели на своей части данных.
- Передача результата обратно на главную ноду.



# Преимущества и масштабируемость

## Масштабируемость:

- Добавление нового воркера не требует изменений в архитектуре.
- Количество воркеров неограниченно.

## Устойчивость:

- Падение отдельного воркера не нарушает работу системы.
- Обучение не зависит от состояния других воркеров.

## Контроль конфликтов:

- Обеспечена безопасность выполнения задач на главной ноде.

**Таким образом система обеспечивает распределённое обучение моделей с контролем конфликтов. Устойчива к отказам отдельных элементов. Легко расширяется за счёт добавления новых воркеров.**

# Начало обучения

```
INFO:      127.0.0.1:44006 - "POST /run HTTP/1.1" 200 OK
Часть 0 сохранена в splits/split_0.pt
Часть 1 сохранена в splits/split_1.pt
Uploaded splits/split_0.pt to 192.168.31.190:/app/dataset_parts/split_0.pt
Uploaded splits/split_1.pt to 192.168.31.40:/app/dataset_parts/split_1.pt
=== Global Epoch 1/4 ===
Running training on node 192.168.31.190...
Running training on node 192.168.31.40...
192.168.31.190 STDOUT: Эпоха 1/10, Потери: 0.5636203289031982
Эпоха 2/10, Потери: 0.9736637473106384
Эпоха 3/10, Потери: 0.9510632157325745
Эпоха 4/10, Потери: 0.7306684851646423
Эпоха 5/10, Потери: 0.5757642984390259
Эпоха 6/10, Потери: 1.077193260192871
Эпоха 7/10, Потери: 1.020348072052002
Эпоха 8/10, Потери: 0.5709155797958374
Эпоха 9/10, Потери: 0.3049222230911255
Эпоха 10/10, Потери: 0.594040036201477
Модель сохранена в /app/models/model_0.pt
```

# Промежуточная агрегация модели

```
192.168.31.40 STDOUT: Эпоха 1/10, Потери: 0.6114419102668762
Эпоха 2/10, Потери: 0.6845095753669739
Эпоха 3/10, Потери: 0.37140750885009766
Эпоха 4/10, Потери: 0.7772022485733032
Эпоха 5/10, Потери: 0.7756518125534058
Эпоха 6/10, Потери: 0.8413106799125671
Эпоха 7/10, Потери: 0.6111480593681335
Эпоха 8/10, Потери: 0.632580041885376
Эпоха 9/10, Потери: 0.43630170822143555
Эпоха 10/10, Потери: 0.422661155462265
Модель сохранена в /app/models/model_0.pt

192.168.31.40 STDERR:
Checking files in /app/models on node 192.168.31.190
['..', '.', 'model_1.pt', 'model_0.pt', 'model_2.pt', 'global_model.pt']
Downloaded /app/models/model_0.pt to local_models/model_192_168_31_190_7777.pt
Checking files in /app/models on node 192.168.31.40
['..', '..', 'model_0.pt', 'model_2.pt', 'model_9.pt', 'model_4.pt', 'model_5.pt',
Downloaded /app/models/model_0.pt to local_models/model_192_168_31_40_7777.pt
Объединённая модель сохранена в final_model.pt с использованием метода 'regularize
Uploaded final_model.pt to 192.168.31.190:/app/models/global_model.pt
Uploaded final_model.pt to 192.168.31.40:/app/models/global_model.pt
=== Global Epoch 2/4 ===
Running training on node 192.168.31.190...
Running training on node 192.168.31.40...
```



# Финальная агрегация модели

```
INFO: 192.168.31.88:40298 - "POST /run HTTP/1.1" 200 OK
Часть 0 сохранена в splits/split_0.pt
Часть 1 сохранена в splits/split_1.pt
Uploaded splits/split_0.pt to 192.168.31.190:/app/dataset_parts/split_0.
Uploaded splits/split_1.pt to 0.0.0.0:/app/dataset_parts/split_1.pt
=== Global Epoch 1/1 ===
Running training on node 192.168.31.190...
Running training on node 0.0.0.0...
192.168.31.190 STDOUT: Эпоха 1/1, Потери: 2.30436635017395
Модель сохранена в /app/models/model_0.pt

192.168.31.190 STDERR: /app/train.py:10: FutureWarning: You are using `t
t malicious pickle data which will execute arbitrary code during unpickl
for `weights_only` will be flipped to `True`. This limits the functions
llowlisted by the user via `torch.serialization.add_safe_globals`. We re
GitHub for any issues related to this experimental feature.
    data = torch.load(args.data_path)

0.0.0.0 STDOUT: Эпоха 1/1, Потери: 0.7858152389526367
Модель сохранена в /app/models/model_0.pt

0.0.0.0 STDERR:
Checking files in /app/models on node 192.168.31.190
['..', '.', 'model_1.pt', 'model_0.pt', 'model_2.pt', 'global_model.pt',
Downloaded /app/models/model_0.pt to local_models/model_192_168_31_190_7
Checking files in /app/models on node 0.0.0.0
['.', '..', 'model_0.pt', 'global_model.pt', 'model_1.pt', 'model_2.pt',
Downloaded /app/models/model_0.pt to local_models/model_0_0_0_0_7777.pt
Объединённая модель сохранена в final_model.pt с использованием метода '
Uploaded final_model.pt to 192.168.31.190:/app/models/global_model.pt
Uploaded final_model.pt to 0.0.0.0:/app/models/global_model.pt
Финальная модель сохранена в final_model.pt
```



Санкт-Петербургский  
государственный университет

## Блок 3: Разработка Интерфейса





# Введение в Разработку Интерфейса

## •Значение Интерфейса:

- Обеспечивает удобный доступ пользователей к функциональности модели.
- Упрощает процесс взаимодействия с системой, делая её интуитивно понятной.

## •Цели Разработки:

- Создание пользовательского интерфейса для загрузки изображений и получения предсказаний.
- Визуализация результатов классификации и метрик модели.
- Обеспечение масштабируемости и отзывчивости интерфейса.

# Дизайн и UX/UI

## Принципы Дизайна:

- Простота: Минималистичный дизайн с акцентом на основные функции.
- Интуитивность: Лёгкая навигация и понятные элементы управления.

## Компоненты Интерфейса:

- Окно логина: Вход в программу по идентификатору админа, получение токена доступа.
- Основное окно: Основной функционал, настройки и отображение параметров.
- Окно настройки воркеров: Настройки воркеров.


# Используемые Технологии:

## Frontend:

- - Qt со стилем css: Для придания приятного вида интерфейсу программы. Визуальная составляющая реализована на низком уровне, сугубо через код, без использования Qt designer.

## Backend:

- - Qt Framework (C++) – вся реализация клиента, включая взаимодействие с сервером и базой данных, написана на Qt.
- **QtWidgets (QVBoxLayout, QLabel, QPushButton и др.):** Используются для создания графического интерфейса пользователя (GUI). Обеспечивают расположение элементов интерфейса, таких как кнопки, текстовые поля и метки.
- **QSqlDatabase и QSqlQuery:** Предоставляют доступ к базе данных SQLite. Используются для хранения информации о воркерах (настройки, IP, порт, учетные данные).
- **QFileDialog, QInputDialog:** Упрощают взаимодействие пользователя с файловой системой (например, выбор и сохранение файлов) и ввод данных.
- **QNetworkAccessManager, QNetworkRequest:** Реализуют HTTP-запросы к серверу для отправки/получения данных и управления распределенным обучением.

 Админ-авторизация

**Вход администратора**

Войти

**Выберите файл матрицы:**[Загрузить файл](#)**Выберите машины для обучения:**

tima  
sasha

[Добавить воркер](#)[Удалить воркер](#)**Выберите метод деления датасета:**

- ☒ Случайное деление  
☐ Стратифицированное деление

**Выберите метод объединения весов:**

- ☒ simple\_average  
☐ median  
☐ regularized

[Открыть настройки воркеров](#)[Получить данные](#)[Отправить данные](#)[Начать обучение](#)**Количество глобальных эпох:****Количество локальных эпох:**

**Выберите файл матрицы:**[Загрузить файл](#)**Выберите машины для обучения:**

tima  
sasha

[Добавить воркер](#)[Удалить воркер](#)**Выберите метод деления датасета:**

- ☒ Случайное деление  
☐ Стратифицированное деление

**Выберите метод объединения весов:**

- ☒ simple\_average  
☐ median  
☐ regularized

[Открыть настройки воркеров](#)[Получить данные](#)[Отправить данные](#)[Начать обучение](#)**Количество глобальных эпох:****Количество локальных эпох:**

W Настройки воркера: tima

IP адрес:

Порт:

Имя пользователя:

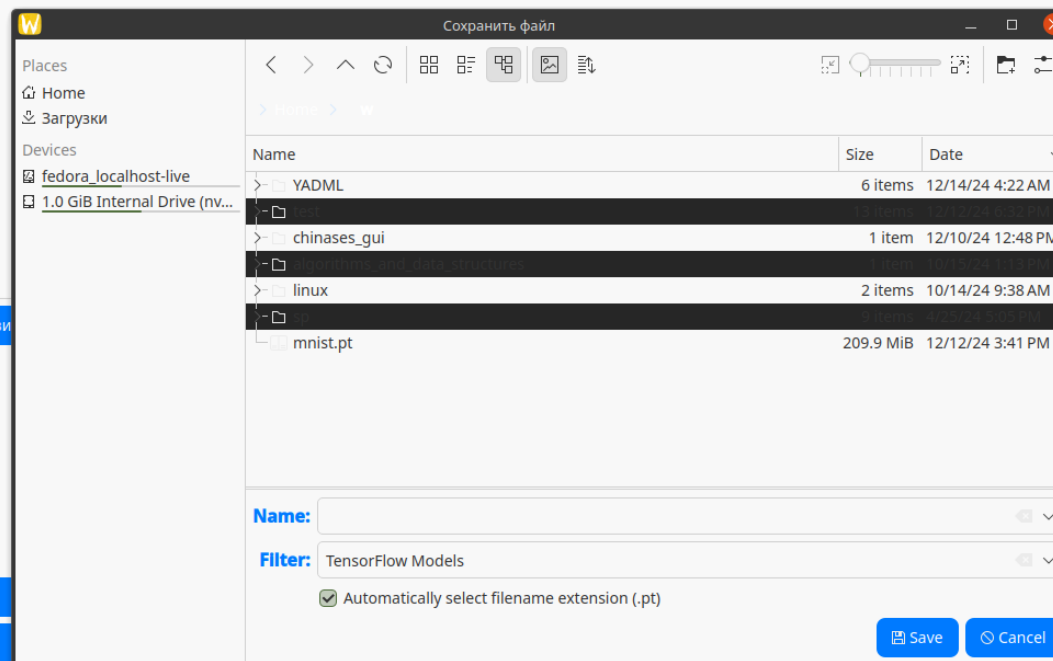
Пароль:

✓ OK

⌛ Cancel

**Выберите файл матрицы:**[Загрузить файл](#)**Выберите машины для обучения:**

tima  
sasha

[Добави](#)**Выберите метод деления датасета:**

- ☒ Случайное деление
- ☐ Стратифицированное деление

**Выберите метод объединения весов:**

- ☒ simple\_average
- ☐ median
- ☐ regularized

[Отправить данные](#)[Начать обучение](#)**Количество глобальных эпох:****Количество локальных эпох:**

**Выберите файл матрицы:**[Загрузить файл](#)**Выберите машины для обучения:**

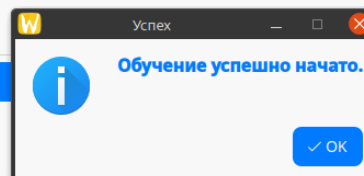
tima  
sasha

[Добавить воркер](#)[Удалить воркер](#)**Выберите метод деления датасета:**

- ☒ Случайное деление  
☐ Стратифицированное деление

**Выберите метод объединения весов:**

- ☒ simple\_average  
☐ median  
☐ regularized

[Открыть настройки воркеров](#)[Получить данные](#)[Отправить данные](#)[Начать обучение](#)**Количество глобальных эпох:****Количество локальных эпох:**



**Выберите файл матрицы:**[Загрузить файл](#)**Выберите машины для обучения:**

tima  
sasha

[Добавить воркер](#)[Удалить воркер](#)**Выберите метод деления датасета:**

- ☒ Случайное деление  
☐ Стратифицированное деление

**Выберите метод объединения весов:**

- ☒ simple\_average  
☐ median  
☐ regularized

[Открыть настройки воркеров](#)[Получить данные](#)[Отправить данные](#)[Начать обучение](#)**Количество глобальных эпох:****Количество локальных эпох:**