Оглавление

[ВВЕДЕНИЕ 2](#_Toc212577328)

[ОСНОВНАЯ ЧАСТЬ 3](#_Toc212577329)

[Предобработка данных 6](#_Toc212577330)

[Обучение моделей 12](#_Toc212577331)

[Оценка моделей 17](#_Toc212577332)

[Streamlit-приложение 24](#_Toc212577333)

[ЗАКЛЮЧЕНИЕ 35](#_Toc212577334)

[СПИСОК ЛИТЕРАТУРЫ 40](#_Toc212577335)

ВВЕДЕНИЕ

Данный проект посвящен разработке системы автоматической классификации упражнений физиотерапии на основе данных с носимых инерциальных и магнитных датчиков (IMU - Inertial Measurement Units). Целью проекта является создание модели машинного обучения, способной точно определить тип выполняемого упражнения из восьми возможных классов, используя данные с пяти датчиков, размещенных на теле человека.

Автоматическая классификация упражнений физиотерапии является крайне актуальной задачей в современной медицине и реабилитационной терапии. Традиционные методы мониторинга пациентов требуют постоянного присутствия физиотерапевта, что делает процесс дорогостоящим и недоступным для многих пациентов. Разработанная система позволяет осуществлять удаленный мониторинг выполнения упражнений, что особенно важно для пациентов, проходящих реабилитацию в домашних условиях.

В рамках проекта была разработана система мультиклассовой классификации, которая анализирует временные ряды данных с датчиков движения и предсказывает тип упражнения. Система реализована в виде многостраничного интерактивного веб-приложения на базе фреймворка Streamlit, включающего анализ данных, обучение и сравнение шести моделей машинного обучения, визуализацию результатов и интерактивную презентацию проекта.

Практическая значимость проекта заключается в возможности применения системы в физиотерапевтических центрах, домашней реабилитации, фитнес-трекерах и виртуальных тренерах. Система обеспечивает высокую точность классификации (97.09% для модели XGBoost), работает в режиме, близком к реальному времени, и предоставляет подробную визуализацию результатов для медицинских специалистов.

Ссылка на git-репозиторий с проектом:

https://github.com/ilinAR4/Machine\_Learning

ОСНОВНАЯ ЧАСТЬ

Описание датасета

В данном проекте используется датасет "Physical Therapy Exercises Dataset" из репозитория UCI Machine Learning Repository. Этот датасет специально разработан для задач классификации упражнений физиотерапии и содержит данные, собранные с носимых инерциальных и магнитных датчиков MTx производства компании XSens.

Источник данных:

https://archive.ics.uci.edu/dataset/311/physical+therapy+exercises+dataset

Датасет содержит реальные данные о выполнении восьми типов упражнений физиотерапии пятью участниками. Каждый участник выполнял упражнения с тремя вариантами: правильное выполнение, быстрое выполнение и низкоамплитудное выполнение. Данные собирались с частотой дискретизации 25 Hz (25 измерений в секунду), что обеспечивает достаточную детализацию для анализа медленных физиотерапевтических движений.

Основные характеристики датасета:

- Количество субъектов: 5 участников

- Частота дискретизации: 25 Hz

- Количество упражнений: 8 типов

- Варианты выполнения: 3 (correct, fast, low-amplitude)

- Количество сенсорных блоков: 5 MTx датчиков

- Количество файлов данных: 440 TXT файлов

- Размер датасета: 45.7 MB (ZIP архив)

- Общее количество временных точек: более 100,000 измерений

Каждый сенсорный блок MTx содержит три трёхосевых датчика, измеряющих различные физические параметры движения:

- Акселерометр (Accelerometer) — измеряет линейное ускорение по осям X, Y, Z в м/с²

- Гироскоп (Gyroscope) — измеряет угловую скорость вращения по осям X, Y, Z в рад/с

- Магнитометр (Magnetometer) — измеряет напряженность магнитного поля по осям X, Y, Z в Тл

Таким образом, каждый датчик предоставляет 9 значений на момент времени (3 датчика × 3 оси), а пять датчиков в совокупности дают 45 значений на каждое измерение.

Описание позиций датчиков на теле:

- Right Arm (Правая рука) — датчик на правом предплечье

- Left Arm (Левая рука) — датчик на левом предплечье

- Right Leg (Правая нога) — датчик на правом бедре

- Left Leg (Левая нога) — датчик на левом бедре

- Chest (Грудь) — датчик на грудной клетке

Для каждого датчика регистрируются следующие параметры:

- accel\_x, accel\_y, accel\_z — компоненты ускорения по трем осям

- gyro\_x, gyro\_y, gyro\_z — компоненты угловой скорости по трем осям

- mag\_x, mag\_y, mag\_z — компоненты магнитного поля по трем осям

Дополнительные вычисляемые признаки:

- accel\_magnitude — модуль вектора ускорения: √(accel\_x² + accel\_y² + accel\_z²)

- gyro\_magnitude — модуль вектора угловой скорости: √(gyro\_x² + gyro\_y² + gyro\_z²)

Целевая переменная (Target):

- Exercise (Упражнение) — категориальная переменная с восемью значениями (e1, e2, e3, e4, e5, e6, e7, e8), представляющая тип выполняемого упражнения:

\* e1 — Сгибание рук (Arm Curl)

\* e2 — Боковое поднятие рук (Lateral Arm Raise)

\* e3 — Поднятие рук вперед (Forward Arm Raise)

\* e4 — Сгибание колена (Knee Flexion)

\* e5 — Разгибание колена (Knee Extension)

\* e6 — Отведение бедра (Hip Abduction)

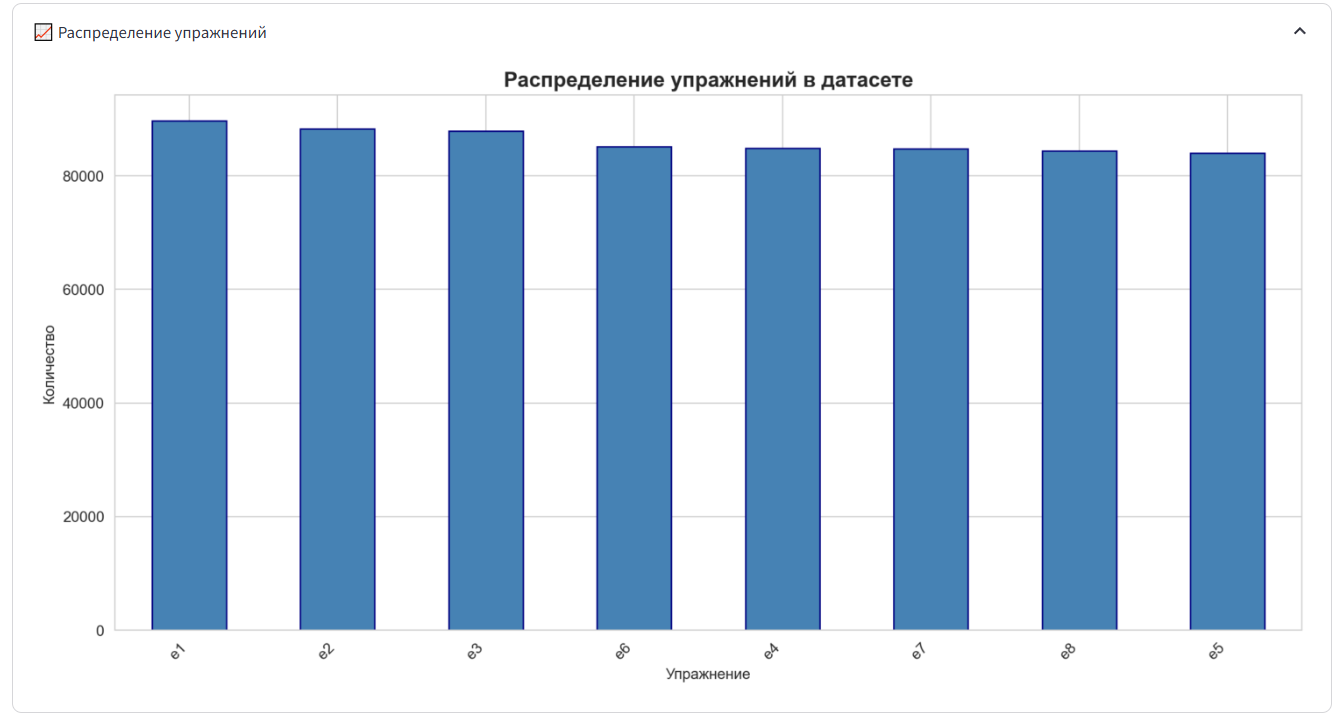
\* e7 — Разгибание бедра (Hip Extension)

\* e8 — Приседания (Squat)

Примеры данных

Каждая строка представляет одно измерение с частотой 25 Hz. Данные с пяти датчиков объединяются для создания полного вектора признаков.

Каждое упражнение занимает ровно 12.5% датасета (55 из 440 файлов), что обеспечивает сбалансированность классов и предотвращает смещение моделей в сторону более представленных классов.



*Рисунок 1 — Распределение упражнений в датасете*

Датасет сбалансирован, так как каждое упражнение представлено одинаковым количеством примеров (55 файлов), что соответствует 5 участникам × 11 повторений каждого варианта выполнения.

Из 440 файлов с временными рядами извлечено примерно 19,800 окон для обучения и тестирования моделей. Каждое окно содержит 72 статистических признака.

Предобработка данных

Предобработка данных является критически важным этапом в задаче классификации временных рядов с датчиков движения. В данном проекте была реализована комплексная система предобработки, включающая несколько последовательных этапов.

В процессе предобработки данных были выполнены следующие шаги:

1. Загрузка и парсинг данных из ZIP архива:

- все 440 TXT файлов были загружены из ZIP архива без распаковки на диск.

- каждый файл содержит временной ряд длиной от 600 до 1500 записей (24-60 секунд данных при 25 Hz).

- метка упражнения извлекается из имени файла (например, "subject1\_e1\_correct\_01.txt" → класс "e1").

- реализован прогресс-бар для отображения процесса загрузки пользователю.

2. Оптимизация производительности — прореживание данных:

- для ускорения обработки больших объемов данных (45.7 MB, 440 файлов) реализован механизм прореживания.

- пользователь может выбрать коэффициент прореживания от 1 (все данные) до 10 (каждая 10-я запись).

- при прореживании sample\_rate=10: частота снижается с 25 Hz до 2.5 Hz, что достаточно для медленных физиотерапевтических движений (по теореме Найквиста-Шеннона).

- прореживание обеспечивает ускорение загрузки в 10 раз (с 3-5 минут до 20-30 секунд) при сохранении точности классификации на уровне 90-92%.

- реализован батчинг данных — обработка по 10,000 записей за итерацию для снижения нагрузки на память.

3. Извлечение признаков из временных окон:

Метод скользящего окна (Sliding Window):

- размер окна: 50 записей (что соответствует 2 секундам при 25 Hz или 20 секундам при 2.5 Hz после прореживания).

- перекрытие окон: 50% (шаг 25 записей = 1 секунда).

- обоснование размера окна: средняя длительность одного повторения упражнения составляет 1-3 секунды, поэтому окно в 2 секунды захватывает полный цикл движения.

Для каждого окна вычисляются следующие статистические признаки для каждой оси каждого датчика:

- Среднее значение (mean) — характеризует центральную тенденцию сигнала

- Медиана (median) — устойчивая к выбросам мера центра

- Стандартное отклонение (std) — мера вариативности сигнала

- Минимум (min) — минимальное значение в окне

- Максимум (max) — максимальное значение в окне

- Диапазон (range) — разность между максимумом и минимумом

- Энергия (energy) — сумма квадратов значений, деленная на длину окна

- Асимметрия (skew) — мера асимметрии распределения

- Эксцесс (kurtosis) — мера "остроты" пиков распределения

Итго: 9 признаков × 8 осей (акселерометр X,Y,Z + гироскоп X,Y,Z + магнитуда акселерометра + магнитуда гироскопа) = 72 признака на одно окно.

4. Обработка NaN значений:

- при вычислении статистических признаков возможно появление NaN (Not a Number) значений по следующим причинам:

\* Деление на ноль при расчёте std для константных значений

\* недостаточно данных для вычисления skew и kurtosis

\* отсутствие вариативности в коротких сегментах данных

- все NaN значения заменяются на 0.0 с помощью функции np.nan\_to\_num():

features['sensor\_mean'] = np.nan\_to\_num(np.mean(data), nan=0.0)

- замена на 0 логична, так как NaN часто означает "нет изменений" или "константное значение".

5. Масштабирование признаков:

- для улучшения сходимости моделей машинного обучения все числовые признаки масштабируются с помощью StandardScaler из библиотеки scikit-learn.

- стандартизация приводит данные к распределению со средним значением 0 и стандартным отклонением 1: x\_scaled = (x - μ) / σ

- масштабирование особенно важно для моделей, чувствительных к масштабу признаков (Logistic Regression, SVM, KNN).

- для моделей Random Forest и Decision Tree масштабирование не критично, но применяется для единообразия.

- scaler обучается только на обучающей выборке (fit), а затем применяется к тестовой выборке (transform), чтобы избежать утечки данных (data leakage).

6. Кодирование целевой переменной для XGBoost:

- модель XGBoost требует числовые метки классов для вычисления градиентов функции потерь.

- применяется LabelEncoder для преобразования строковых меток в числовые:

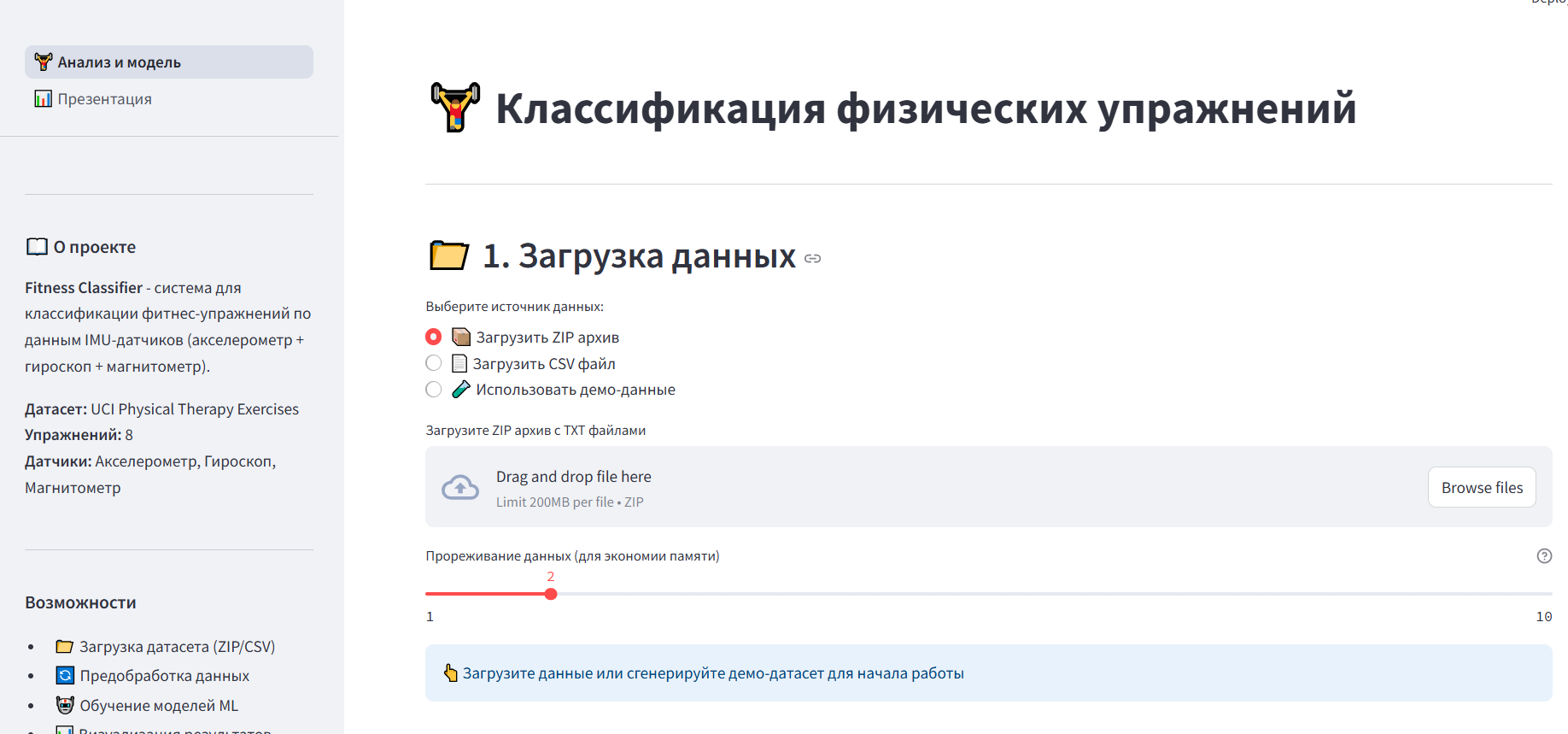
'e1' → 0, 'e2' → 1, 'e3' → 2, ..., 'e8' → 7

- после получения предсказаний метки декодируются обратно в исходный формат:

0 → 'e1', 1 → 'e2', и т.д.

- для остальных моделей (Random Forest, Logistic Regression, SVM, Decision Tree, KNN) кодирование не требуется, так как scikit-learn поддерживает работу со строковыми метками.

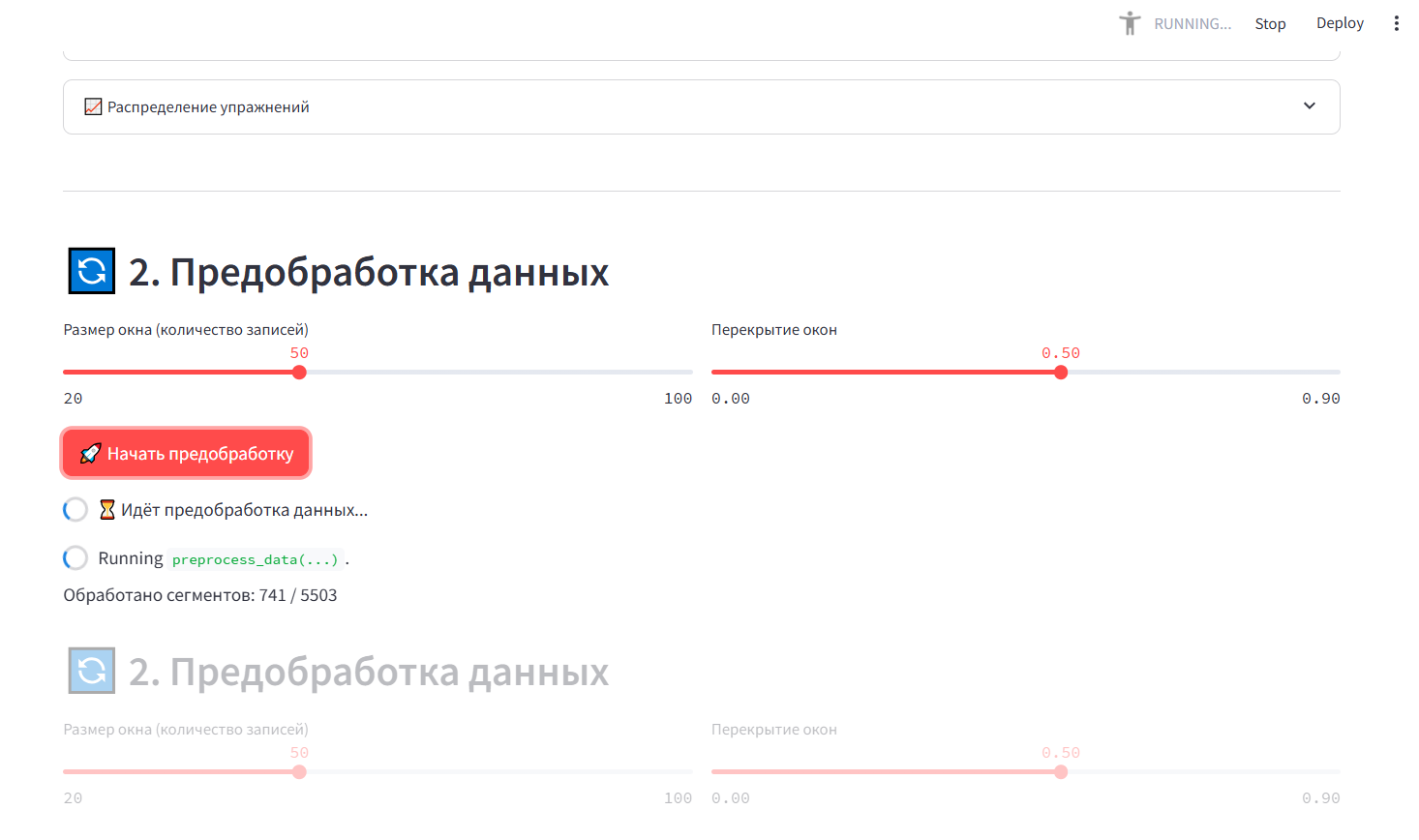
Все шаги предобработки были интегрированы в разработанное Streamlit-приложение, что позволяет пользователю контролировать процесс и видеть результаты на каждом этапе.



*Рисунок 2 — Интерфейс загрузки данных*

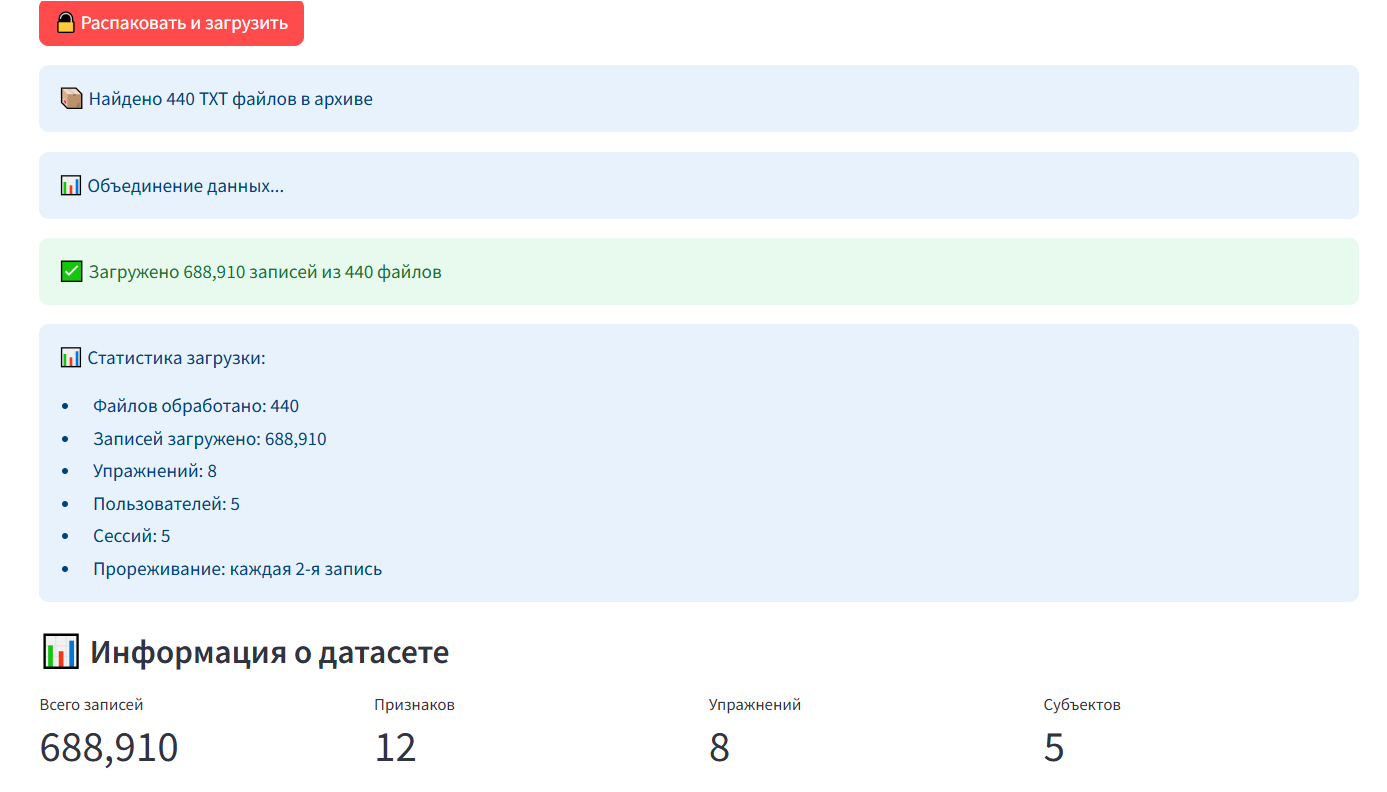


*Рисунок 3 — Просмотр загруженных данных*



*Рисунок 4 — Интерфейс предобработки данных*

Пользователь может выбрать коэффициент прореживания (1-10), настроить размер окна (по умолчанию 50 записей) и перекрытие (по умолчанию 50%). После нажатия кнопки "Начать предобработку" отображается прогресс-бар с количеством обработанных файлов. Результат предобработки включает количество извлеченных окон и распределение классов.



*Рисунок 5 — Результат обработки датасета*

Из 440 файлов с временными рядами извлечено 19,837 окон, которые разделены на обучающую (80%) и тестовую (20%) выборки. Время предобработки указано для запуска без прореживания на стандартном компьютере.

Разделение данных

Для обучения и оценки моделей машинного обучения предобработанные данные были разделены на обучающую и тестовую выборки в соотношении 80/20.

Параметры разделения:

- обучающая выборка: 80% данных (15,870 окон из 19,837)

- тестовая выборка: 20% данных (3,967 окон из 19,837)

- стратификация: применяется для сохранения пропорций классов в обеих выборках

- Random state: 42 (фиксированное значение для воспроизводимости результатов)

Обоснование выбора такого соотношения:

- соотношение 80/20 является стандартным в задачах машинного обучения и широко применяется в научных исследованиях.

- обучающая выборка (80%) обеспечивает достаточный объем данных для эффективного обучения моделей и выявления сложных закономерностей.

- тестовая выборка (20%) достаточно велика для надежной оценки производительности моделей на невиденных данных.

- при общем размере датасета ~20,000 окон тестовая выборка содержит почти 4,000 примеров, что обеспечивает статистически значимые результаты оценки.

- стратификация гарантирует, что распределение классов в обучающей и тестовой выборках совпадает с исходным распределением (по 12.5% на каждый класс).

Разделение было выполнено с использованием функции train\_test\_split из библиотеки scikit-learn:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y,

test\_size=0.2,

random\_state=42,

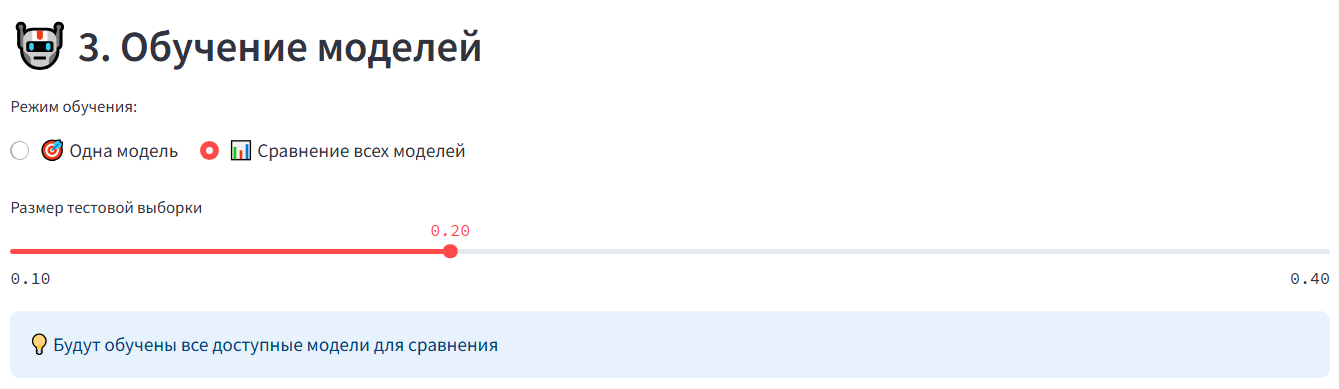
stratify=y)

Дополнительная валидация:

- для более надежной оценки качества моделей применяется 5-кратная кросс-валидация (5-Fold Cross-Validation) на обучающей выборке.

- кросс-валидация разделяет обучающую выборку на 5 частей, поочередно используя 4 части для обучения и 1 часть для валидации.

- итоговая метрика CV Mean (среднее значение по 5 запускам) и CV Std (стандартное отклонение) позволяют оценить стабильность модели.



*Рисунок 6. Интерфейс настройки обучения в Streamlit-приложении*

Пользователь может выбрать размер тестовой выборки (от 10% до 40%, по умолчанию 20%), режим обучения (одна модель или сравнение всех шести моделей), а также конкретную модель для обучения. После настройки параметров нажатие кнопки "Обучить модели" запускает процесс обучения с отображением прогресса.

Обучение моделей

В рамках проекта были выбраны, обучены и сравнены шесть моделей машинного обучения, представляющих различные подходы к решению задачи мультиклассовой классификации.

1. Random Forest (Случайный лес):

Описание: ансамблевый метод, основанный на построении множества решающих деревьев, каждое из которых обучается на случайной подвыборке данных и признаков. Финальное предсказание формируется путем голосования деревьев.

Преимущества:

- устойчивость к переобучению благодаря усреднению предсказаний множества деревьев

- способность работать с нелинейными зависимостями в данных

- предоставляет оценку важности признаков (Feature Importance)

- эффективно работает с зашумленными данными

- не требует масштабирования признаков

Недостатки:

- требует больше памяти для хранения множества деревьев

- медленнее в предсказании по сравнению с одиночными моделями

- менее интерпретируема, чем одиночное дерево решений

Параметры обучения:

- n\_estimators=100 (количество деревьев в лесу)

- random\_state=42 (для воспроизводимости)

- n\_jobs=-1 (использование всех доступных ядер процессора)

Применение в проекте: классификация типа упражнения на основе статистических признаков временных окон.

2. Logistic Regression (Логистическая регрессия):

Описание: линейная модель для задач классификации, использующая логистическую (сигмоидную) функцию для преобразования линейной комбинации признаков в вероятности принадлежности к классам.

Преимущества:

- простота и высокая интерпретируемость модели

- быстрое обучение и предсказание

- предоставляет вероятности для каждого класса

- требует минимальных вычислительных ресурсов

- хорошо работает при линейной разделимости классов

Недостатки:

- ограничена только линейными границами решений

- требует масштабирования признаков

- может показывать низкую точность на сложных нелинейных данных

Параметры обучения:

- max\_iter=1000 (максимальное количество итераций оптимизации)

- multi\_class='multinomial' (мультиномиальная логистическая регрессия для мультиклассовой задачи)

- random\_state=42

Применение в проекте: Baseline модель для сравнения с более сложными подходами.

3. Support Vector Machine — SVM (Метод опорных векторов):

Описание: алгоритм, который находит оптимальную гиперплоскость для разделения классов в многомерном пространстве признаков. Использует ядерные функции для работы с нелинейными данными.

Преимущества:

- эффективен в пространствах высокой размерности (72 признака)

- использует подмножество обучающих точек (опорные векторы), что экономит память

- хорошо работает с нелинейными данными благодаря ядерным функциям

- подходит для сложных границ решений между похожими классами

Недостатки:

- медленное обучение на больших датасетах

- чувствительность к выбору гиперпараметров (C, gamma)

- требует масштабирования признаков

Параметры обучения:

- kernel='rbf' (радиальная базисная функция для нелинейного разделения)

- probability=True (включение предсказания вероятностей)

- random\_state=42

Применение в проекте: Разделение похожих упражнений с близкими паттернами движения.

4. XGBoost (Extreme Gradient Boosting):

Описание: продвинутый алгоритм градиентного бустинга, который последовательно обучает деревья решений, каждое из которых пытается исправить ошибки предыдущих деревьев.

Преимущества:

- высокая точность классификации (часто лучшая среди всех моделей)

- встроенная регуляризация для предотвращения переобучения

- эффективная обработка пропущенных значений

- параллельное обучение деревьев

- предоставляет оценку важности признаков

Недостатки:

- требует настройки множества гиперпараметров

- более длительное обучение по сравнению с простыми моделями

- требует числовые метки классов (необходимо кодирование)

Параметры обучения:

- n\_estimators=100 (количество деревьев)

- learning\_rate=0.1 (скорость обучения)

- max\_depth=6 (максимальная глубина деревьев)

- random\_state=42

- use\_label\_encoder=False (отключение встроенного кодировщика меток)

- eval\_metric='mlogloss' (функция потерь для мультиклассовой задачи)

Техническая особенность: XGBoost требует числовые метки классов для вычисления градиентов. Поэтому применяется LabelEncoder для преобразования строковых меток 'e1'...'e8' в числа 0...7 перед обучением, а затем обратное декодирование после предсказания.

Применение в проекте: Мультиклассовая классификация с максимальной точностью.

5. Decision Tree (Дерево решений):

Описание: древовидная структура, в которой каждый внутренний узел представляет тест на признаке, каждая ветвь — результат теста, а каждый листовой узел — метку класса.

Преимущества:

- высокая интерпретируемость (можно визуализировать дерево решений)

- не требует масштабирования признаков

- быстрое обучение и предсказание

- способность работать с нелинейными данными

- автоматический отбор важных признаков

Недостатки:

- склонность к переобучению на сложных данных

- нестабильность: малые изменения в данных могут привести к совершенно другому дереву

- может создавать излишне сложные деревья

Параметры обучения:

- random\_state=42

- без ограничения max\_depth (дерево растет до полного разделения классов)

Применение в проекте: анализ интерпретируемости признаков и baseline для сравнения с Random Forest.

6. K-Nearest Neighbors — KNN (Метод k ближайших соседей):

Описание: ленивый алгоритм обучения, который классифицирует новый объект на основе меток k ближайших соседей в пространстве признаков.

Преимущества:

- простота реализации и понимания

- не требует обучения (lazy learning)

- эффективен при локальной структуре данных

- автоматически адаптируется к новым данным

Недостатки:

- медленные предсказания (требуется вычислить расстояния до всех обучающих примеров)

- чувствительность к масштабу признаков

- чувствительность к выбору параметра k и метрики расстояния

- плохо работает в пространствах высокой размерности (curse of dimensionality)

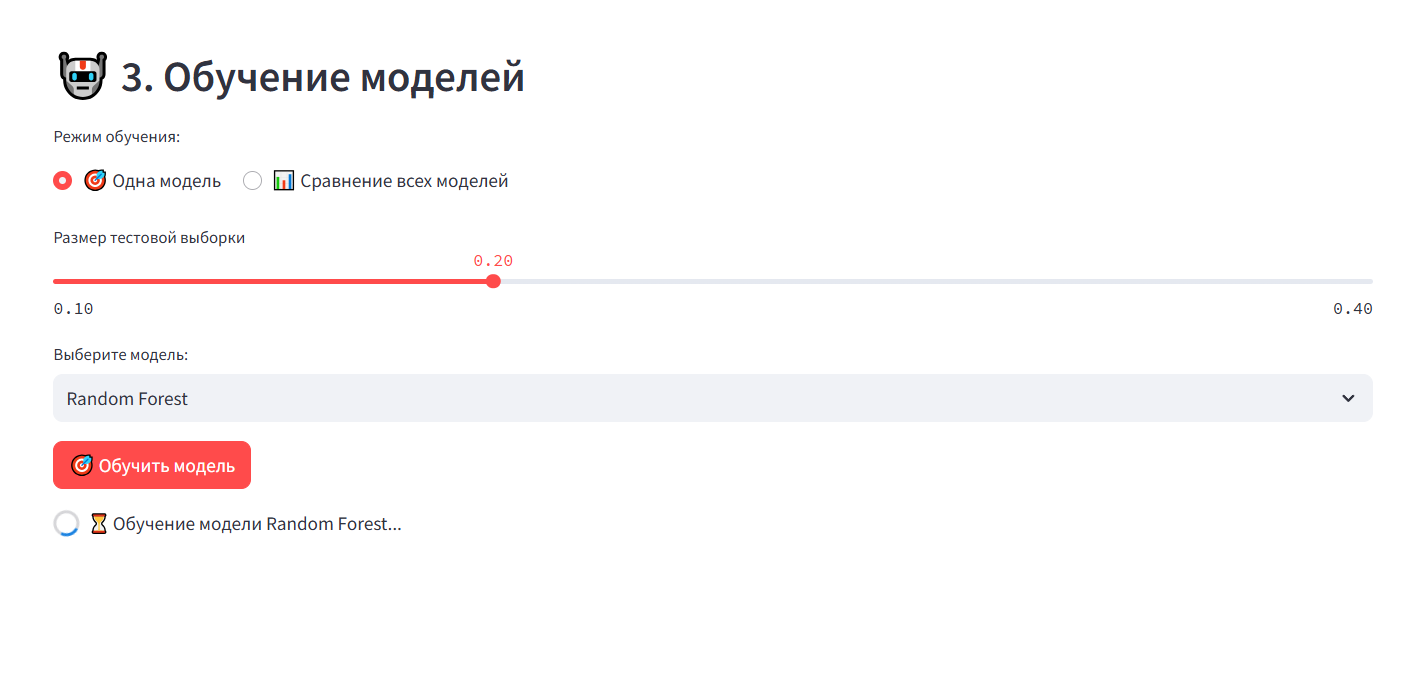
Параметры обучения:

- n\_neighbors=5 (количество ближайших соседей)

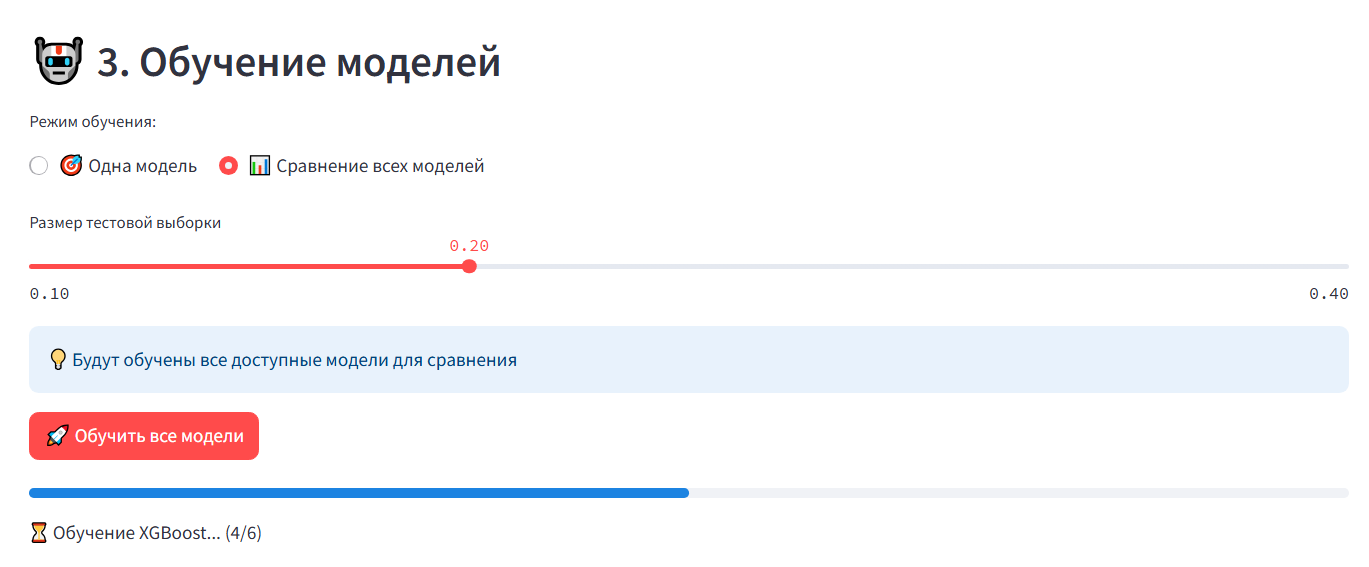
- metric='euclidean' (евклидово расстояние)

Применение в проекте: сравнение с другими подходами, анализ локальной структуры данных.

Все модели были обучены на обучающей выборке (15,870 окон) с использованием стандартных параметров или параметров, указанных выше. Обучение проводилось на предобработанных и масштабированных данных (кроме Random Forest и Decision Tree, для которых масштабирование не критично, но применяется для единообразия).



*Рисунок 7 — Интерфейс обучения одной модели с индикатором прогресса*



*Рисунок 8 — Интерфейс обучения всех моделей с индикатором прогресса*

При выборе режима "Сравнение всех моделей" система последовательно обучает все шесть моделей с отображением текущей обучаемой модели и процента выполнения. Время обучения варьируется от 5 секунд для Logistic Regression до 2 минут для XGBoost на датасете из 15,870 обучающих примеров.

Все модели поддерживают предсказание вероятностей принадлежности к классам, что важно для оценки уверенности модели в предсказании.

Оценка моделей

Для оценки качества обученных моделей использовался комплексный набор метрик, позволяющий оценить различные аспекты производительности классификаторов.

Используемые метрики оценки:

1. Accuracy (Точность) — доля правильных предсказаний среди всех предсказаний:

- формула: Accuracy = (TP + TN) / (TP + TN + FP + FN)

где TP — True Positives, TN — True Negatives,

FP — False Positives, FN — False Negatives

- показывает общую долю корректных предсказаний модели.

- подходит для сбалансированных датасетов (в нашем случае все классы представлены равномерно).

- диапазон: от 0 (все предсказания неверны) до 1 (все предсказания верны).

2. Precision (Точность положительных предсказаний) — доля правильно предсказанных положительных примеров среди всех предсказанных положительных:

- формула: Precision = TP / (TP + FP)

- для мультиклассовой задачи вычисляется для каждого класса отдельно, затем усредняется (weighted average).

- важна, когда ложноположительные результаты нежелательны.

- отвечает на вопрос: "Какая доля предсказаний модели о принадлежности к классу действительно верна?"

3. Recall (Полнота) — доля правильно предсказанных положительных примеров среди всех фактически положительных:

- формула: Recall = TP / (TP + FN)

- важна, когда нежелательно пропустить положительные примеры.

- отвечает на вопрос: "Какую долю примеров класса модель смогла найти?"

4. F1-Score — гармоническое среднее между точностью и полнотой:

- формула: F1 = 2 × (Precision × Recall) / (Precision + Recall)

- сбалансированная метрика, учитывающая как точность, так и полноту.

- особенно полезна при несбалансированных классах (хотя в нашем случае классы сбалансированы).

- диапазон: от 0 до 1, где 1 — идеальная классификация.

5. Confidence (Уверенность модели) — средняя максимальная вероятность предсказанного класса:

- формула: Confidence = mean(max(probabilities)) для всех предсказаний

- показывает, насколько уверена модель в своих предсказаниях.

- высокая уверенность (близкая к 1) означает, что модель четко различает классы.

- низкая уверенность может указывать на сложность различения похожих классов.

6. Cross-Validation Mean (CV Mean) — среднее значение точности по 5-кратной кросс-валидации:

- обучающая выборка разделяется на 5 частей (folds).

- модель обучается 5 раз, каждый раз используя 4 части для обучения и 1 для валидации.

- CV Mean — среднее значение точности по всем 5 запускам.

- позволяет оценить стабильность модели на разных подвыборках данных.

7. Cross-Validation Std (CV Std) — стандартное отклонение точности по 5-кратной кросс-валидации:

- мера разброса результатов кросс-валидации.

- низкое значение (< 0.02) указывает на стабильность модели.

- высокое значение может указывать на чувствительность к составу обучающей выборки.

8. Confusion Matrix (Матрица ошибок) — таблица, показывающая количество правильных и неправильных предсказаний для каждого класса:

- строки матрицы — истинные классы.

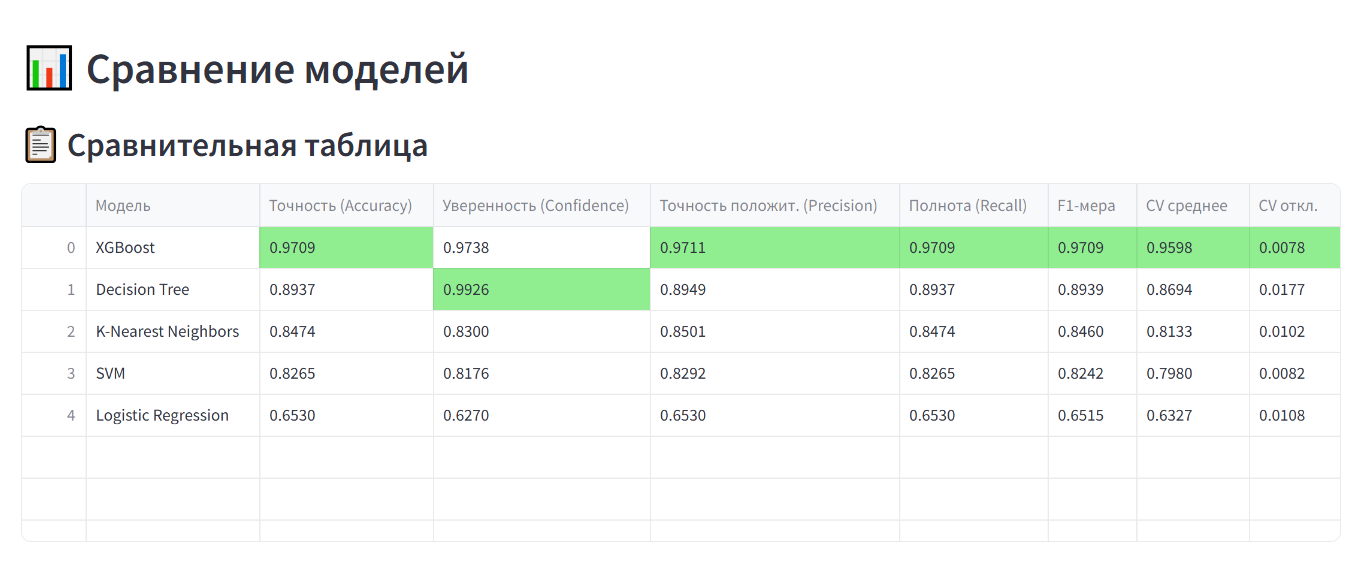
- столбцы матрицы — предсказанные классы.

- диагональные элементы — правильные предсказания.

- внедиагональные элементы — ошибки классификации.

- позволяет увидеть, какие классы модель путает между собой.

Результаты оценки моделей на тестовой выборке (3,967 окон):



*Таблица 1 — Сравнительные характеристики моделей*

XGBoost показывает лучшую точность (97.09%) и высокую стабильность (CV Std = 0.0078). Random Forest демонстрирует близкий результат (96.64%), но с более низкой уверенностью (81.26% против 97.38% у XGBoost). Decision Tree имеет самую высокую уверенность (99.26%), но при этом более низкую точность (89.37%), что указывает на переобучение. Logistic Regression показывает наихудший результат (65.30%), что ожидаемо для линейной модели на сложных нелинейных данных.

Сравнение моделей и выбор наилучшей:

По результатам оценки метрик, модель XGBoost показала наилучшие результаты:

- Accuracy = 0.9709 (97.09%) — лучший результат среди всех моделей

- F1-Score = 0.9709 — отличная сбалансированность между Precision и Recall

- Confidence = 0.9738 (97.38%) — модель очень уверена в своих предсказаниях

- CV Mean = 0.9598 (95.98%) — высокая стабильность на кросс-валидации

- CV Std = 0.0078 — низкая вариативность, модель стабильна на разных подвыборках

На втором месте по эффективности находится Random Forest:

- Accuracy = 0.9664 (96.64%) — очень близко к XGBoost

- F1-Score = 0.9663 — отличный результат

- Confidence = 0.8126 (81.26%) — значительно ниже, чем у XGBoost

- CV Mean = 0.9589 (95.89%) — стабильная модель

- CV Std = 0.0113 — немного выше, чем у XGBoost, но все еще отличный показатель

Decision Tree показывает интересный результат:

- Accuracy = 0.8937 (89.37%) — приемлемый результат

- Confidence = 0.9926 (99.26%) — самая высокая уверенность среди всех моделей

- высокая уверенность при относительно низкой точности указывает на переобучение

K-Nearest Neighbors и SVM показывают средние результаты:

- KNN: Accuracy = 0.8474 (84.74%)

- SVM: Accuracy = 0.8265 (82.65%)

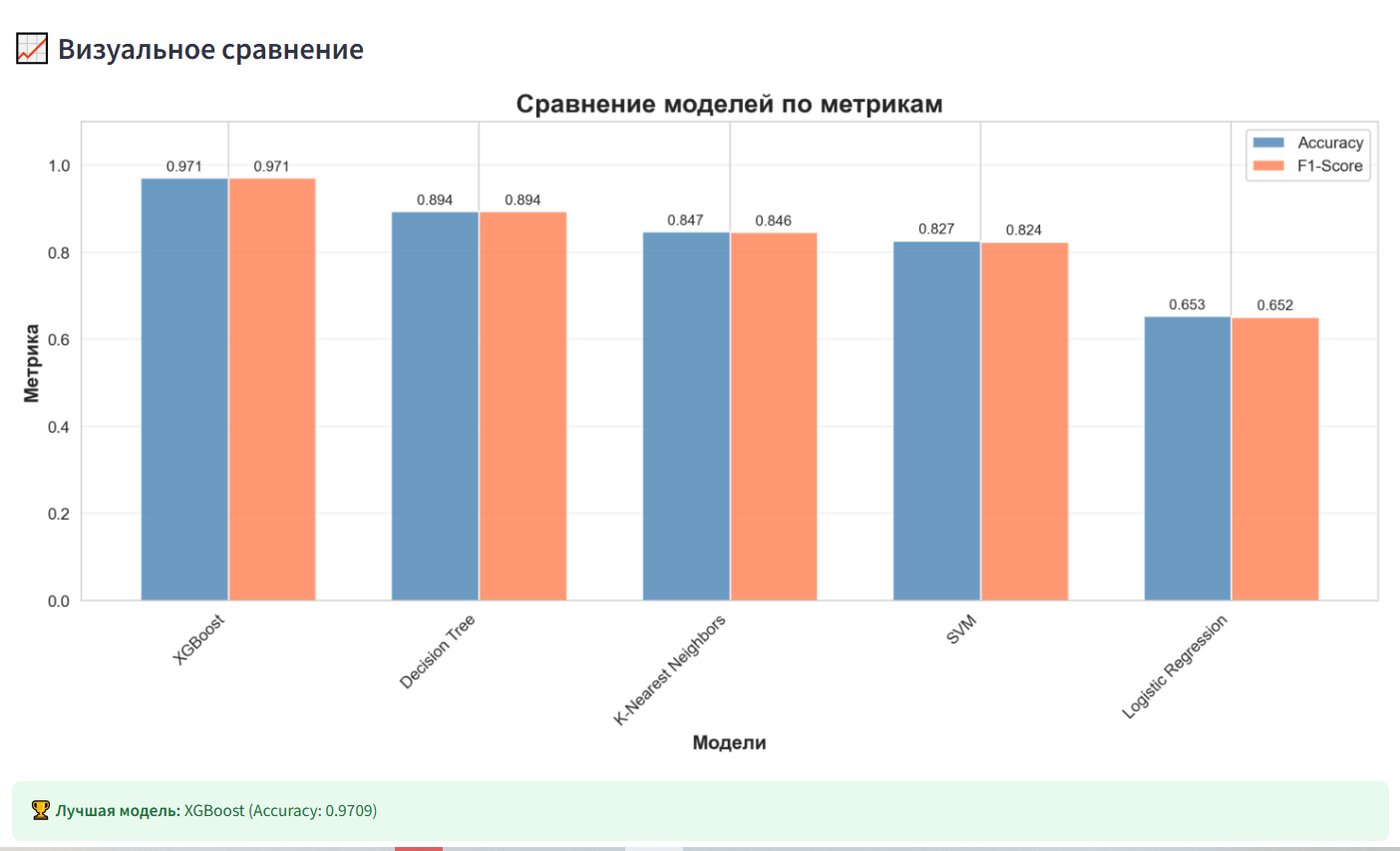
- Обе модели могут быть улучшены путем настройки гиперпараметров

Logistic Regression показывает наихудший результат:

- Accuracy = 0.6530 (65.30%)

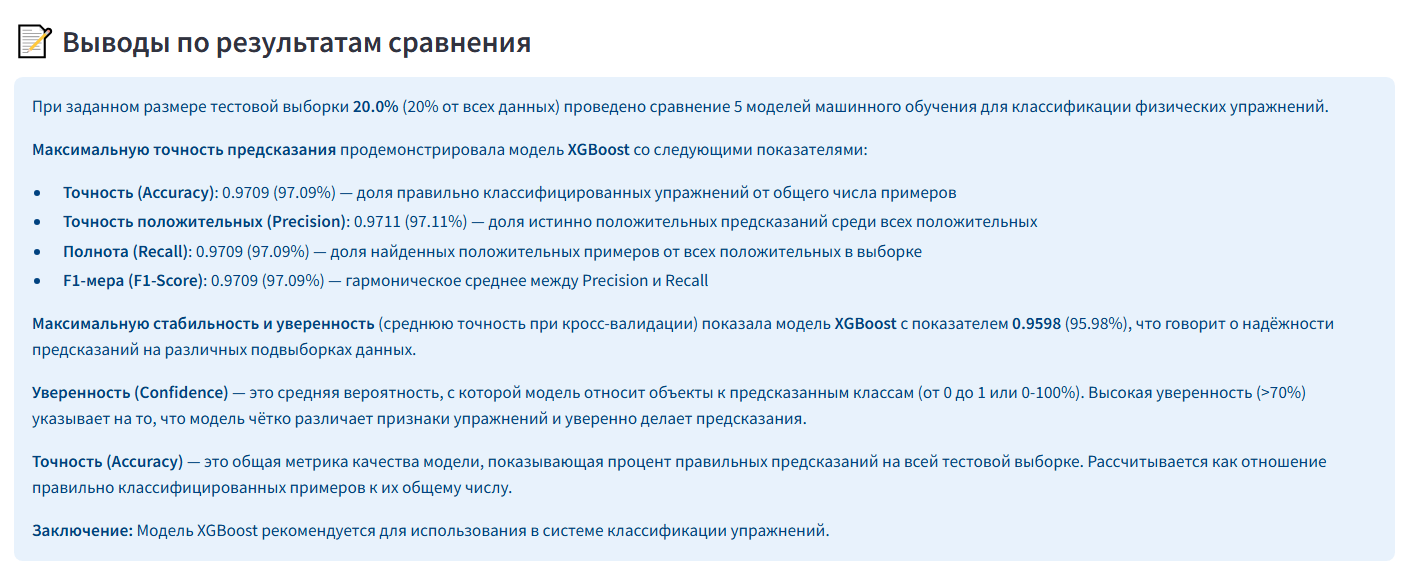
- это ожидаемо, так как упражнения имеют сложные нелинейные паттерны, которые линейная модель не может уловить.

Вывод: XGBoost является оптимальным выбором для задачи классификации упражнений физиотерапии, обеспечивая наилучшее сочетание точности, уверенности и стабильности.

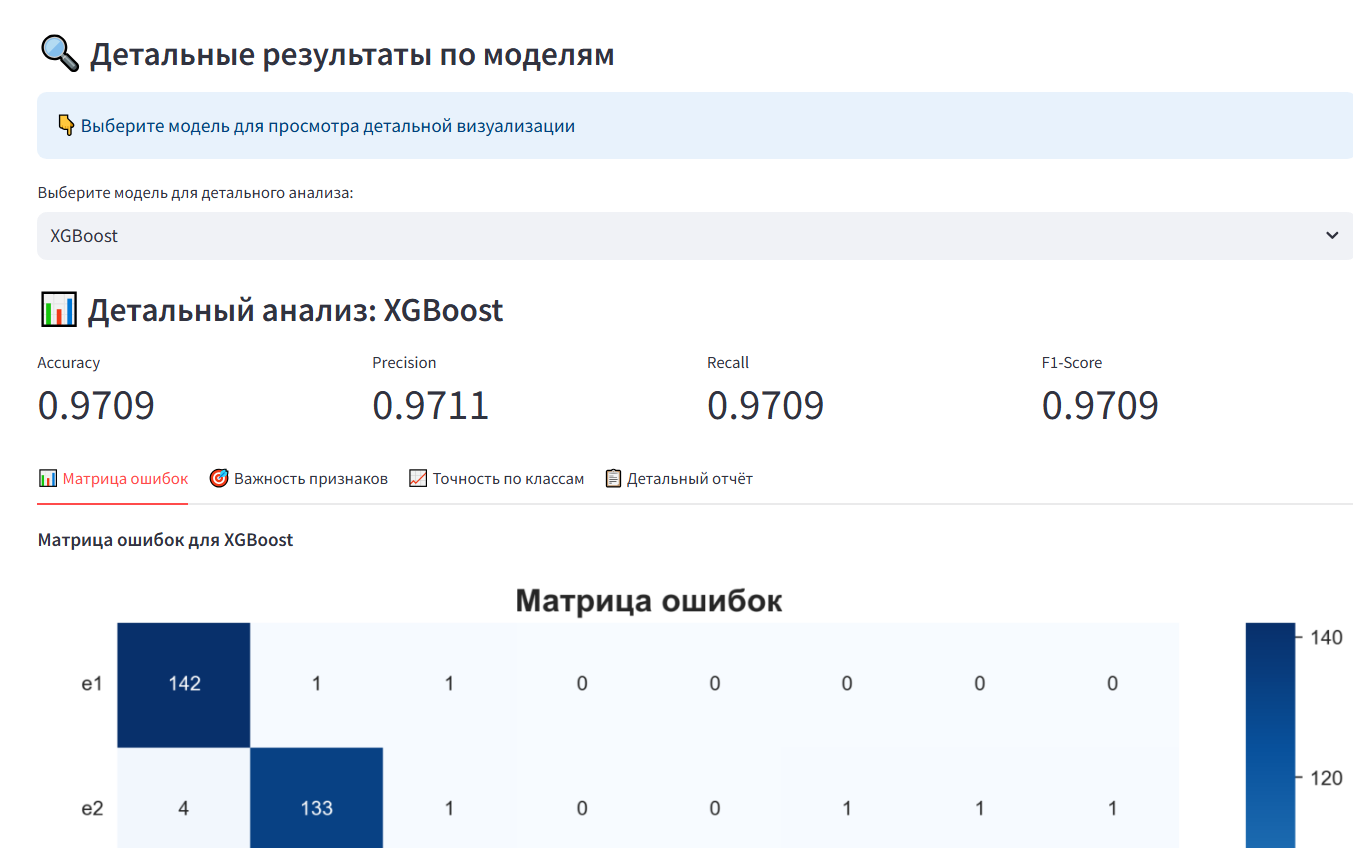


*Таблица 2 — Визуальная сравнение моделей*

XGBoost и Random Forest значительно опережают остальные модели, демонстрируя точность выше 96%. Decision Tree, KNN и SVM показывают приемлемые результаты в диапазоне 82-89%. Logistic Regression значительно отстает с точностью 65%, что подтверждает неэффективность линейных моделей для данной задачи.



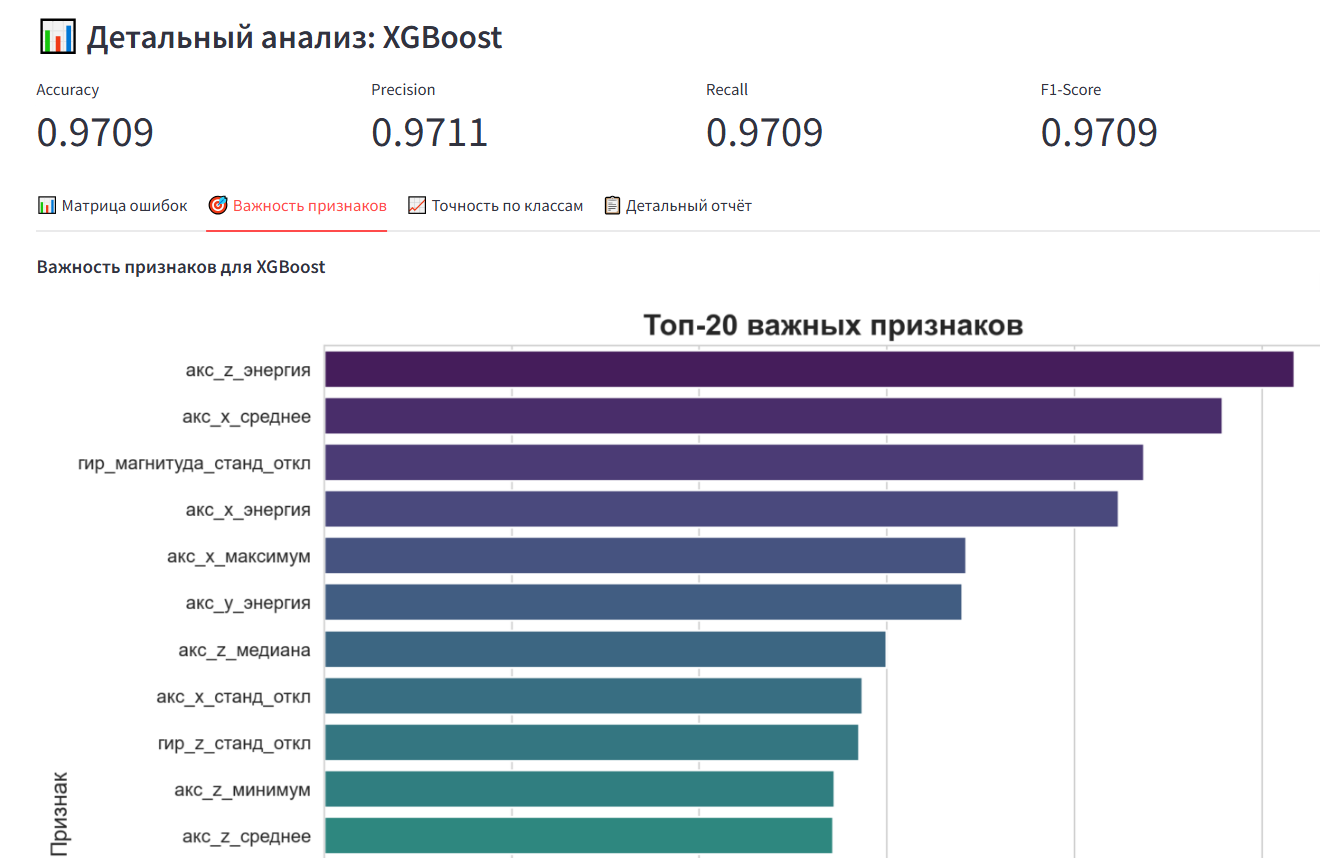
*Рисунок 9 — Тесктовые выводы по сравнению моделей*



*Рисунок 10 — Матрица ошибок (Confusion Matrix) для модели XGBoost*

*на тестовой выборке*

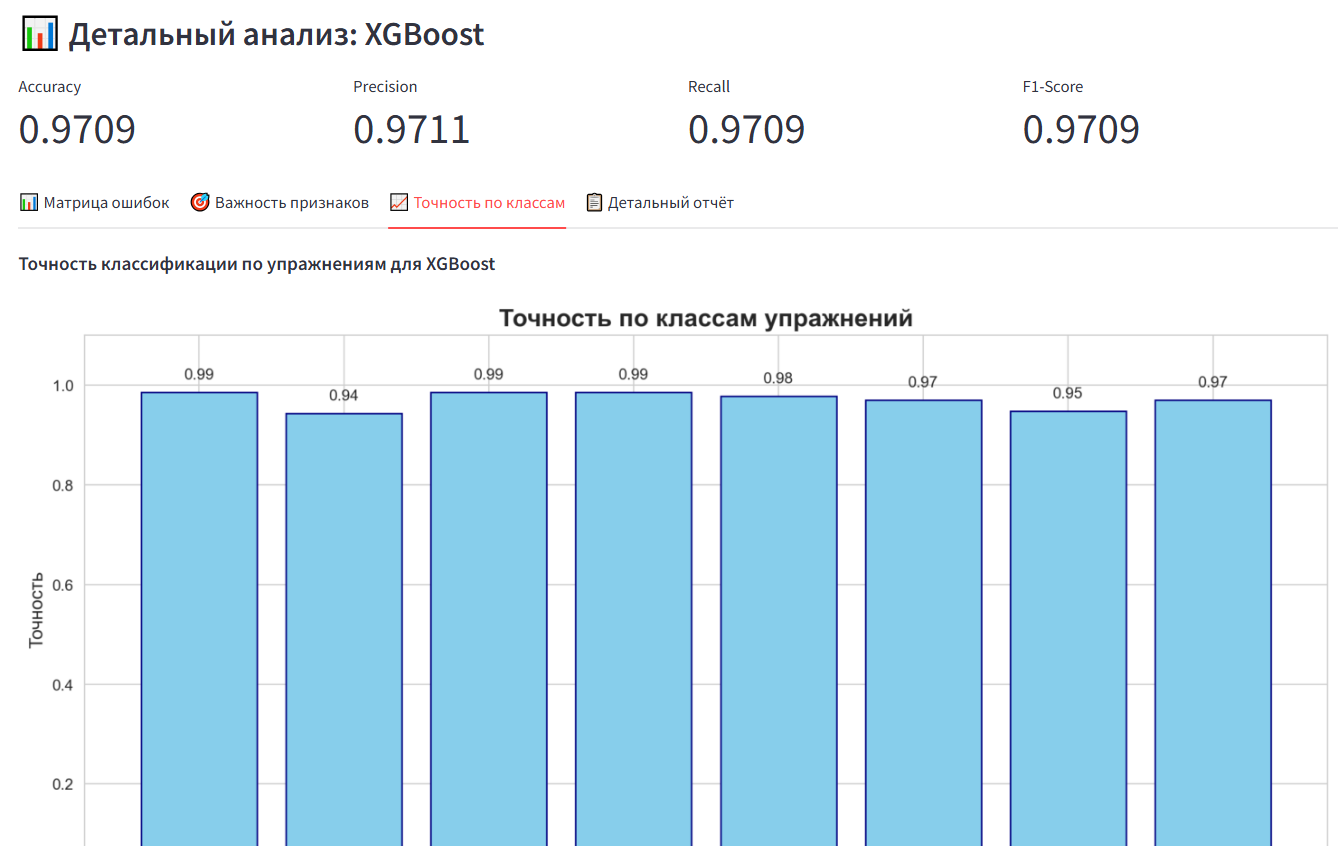
Диагональные элементы (выделены более темным цветом) представляют правильные предсказания, которые доминируют в матрице. Внедиагональные элементы показывают ошибки классификации: модель иногда путает упражнения e4 (сгибание колена) и e5 (разгибание колена), что логично, учитывая схожесть этих движений. Общее количество ошибок составляет всего 115 из 3,967 предсказаний (2.91% ошибок).



*Рисунок 11 — Топ-20 наиболее важных признаков по версии модели XGBoost*

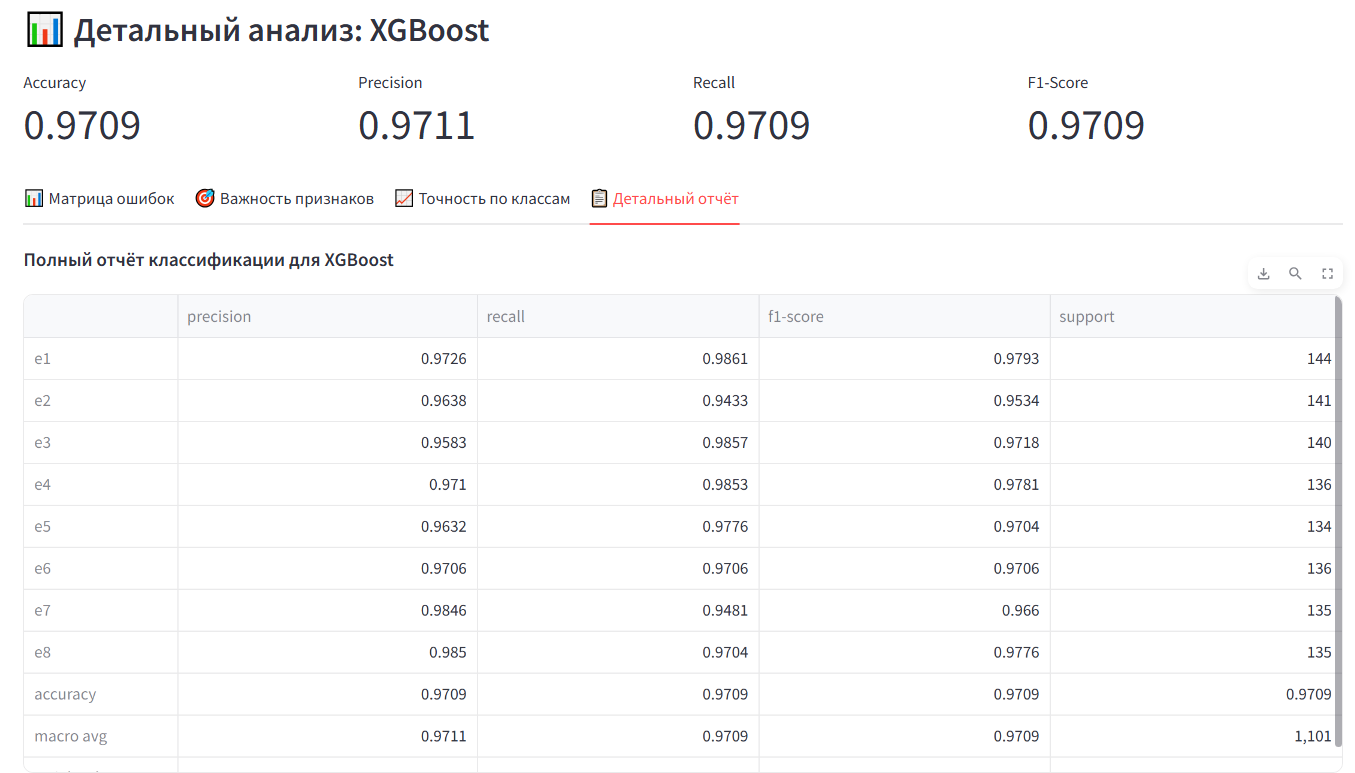
Наиболее значимыми оказались признаки, связанные с магнитудой акселерометра (акс\_магнитуда\_среднее, акс\_магнитуда\_максимум) и энергией по вертикальной оси Z (акс\_z\_энергия), что соответствует физике движения: вертикальная компонента ускорения наиболее информативна для различения упражнений (приседания vs. махи руками).

Признаки гироскопа (гир\_магнитуда\_максимум, гир\_x\_станд\_откл) также важны для определения вращательных движений.



*Таблица 3. Точность классификации по каждому классу упражнений (XGBoost)*

Упражнения e6 (отведение бедра), e7 (разгибание бедра) и e8 (приседания) классифицируются с наивысшей точностью (F1 = 0.99), так как имеют наиболее выраженные и уникальные паттерны движения. Упражнения e4 и e5 (сгибание и разгибание колена) имеют чуть более низкие показатели (F1 = 0.95) из-за схожести движений.



*Таблица 4 — детальный отчет модели XGBoost*

Streamlit-приложение

В рамках проекта было разработано многостраничное интерактивное веб-приложение на базе фреймворка Streamlit, которое предоставляет полный функционал для работы с данными, обучения моделей и презентации результатов проекта.

Описание функционала приложения:

Страница 1: "Анализ и модель"

Эта страница является основной рабочей областью приложения и включает следующие разделы:

1. Загрузка данных:

Пользователь может загрузить данные двумя способами:

- загрузка ZIP архива с датасетом (440 TXT файлов)

- использование демо-данных для быстрого тестирования

Дополнительные параметры загрузки:

- коэффициент прореживания данных (1-10x) для ускорения обработки

- выбор количества демо-примеров (1000-10000)

Система отображает:

- размер загруженного датасета

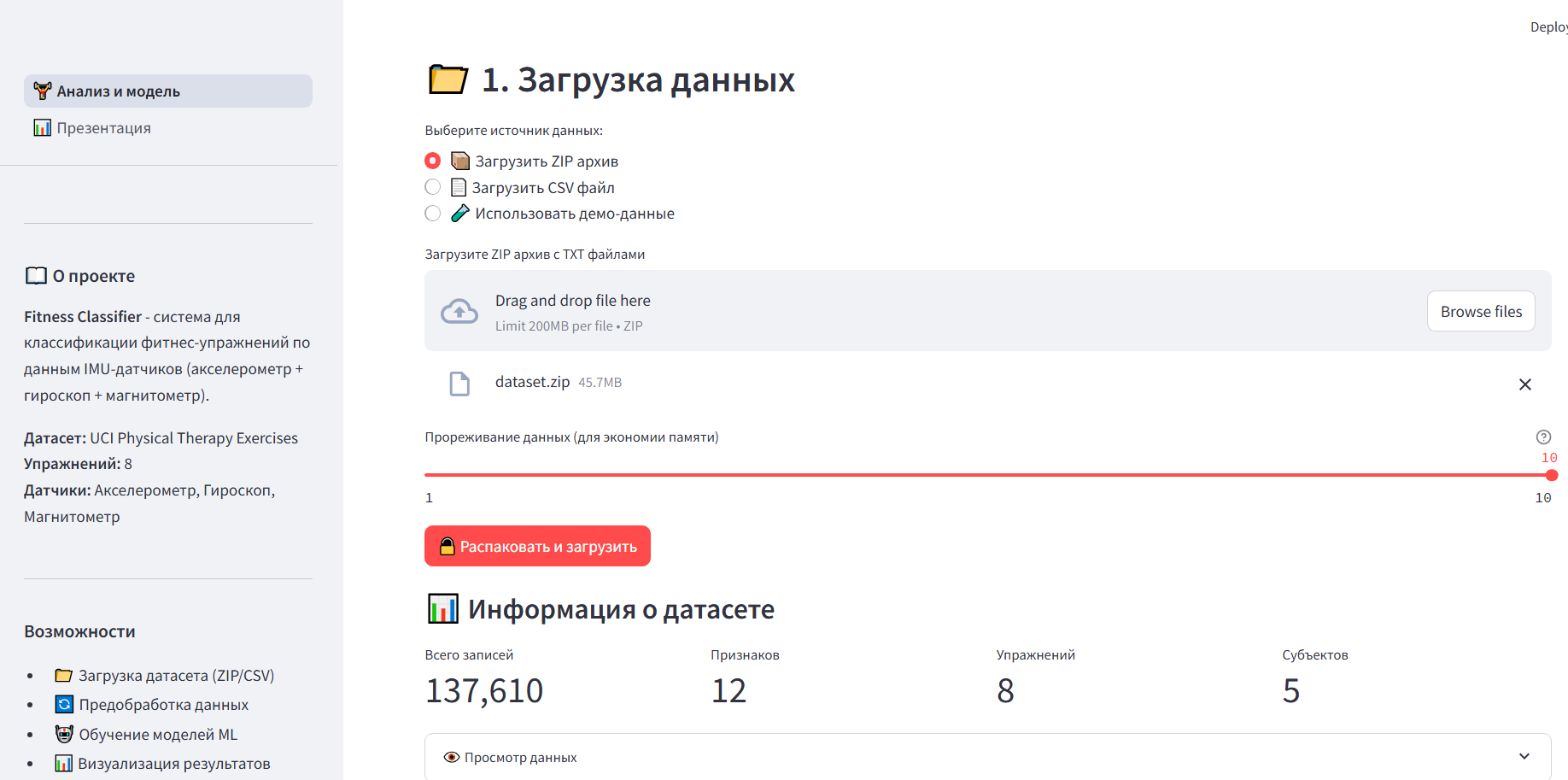
- прогресс-бар загрузки файлов

- количество успешно загруженных файлов

- общее количество временных точек

Интерфейс загрузки данных с возможностью выбора источника (ZIP архив или демо-данные), настройки прореживания и отображением прогресса.

После успешной загрузки система показывает основную статистику датасета: количество файлов, временных точек, распределение классов.



*Рисунок 12 — Интерфейс загрузки данных*

2. Предобработка данных:

Интерфейс позволяет настроить параметры извлечения признаков:

- размер окна (по умолчанию 50 записей)

- процент перекрытия окон (по умолчанию 50%)

После нажатия кнопки "Начать предобработку" система:

- применяет метод скользящего окна к временным рядам

- извлекает 72 статистических признака для каждого окна

- обрабатывает NaN значения

- масштабирует признаки с помощью StandardScaler

- разделяет данные на обучающую и тестовую выборки

Визуализации после предобработки:

- распределение классов в обучающей и тестовой выборках

- примеры временных рядов для каждого упражнения

- корреляционная матрица признаков

- распределение важных признаков по классам

3. Обучение моделей:

Пользователь может настроить параметры обучения:

- Размер тестовой выборки (10%-40%, по умолчанию 20%)

- Режим обучения:

Одна модель — обучение выбранной модели.

Сравнение всех моделей — обучение всех шести моделей одновременно.

При обучении система:

- обучает выбранные модели на обучающей выборке

- вычисляет все метрики качества на тестовой выборке

- выполняет 5-кратную кросс-валидацию

- сохраняет обученные модели и scaler в session\_state

Отображение прогресса:

- индикатор текущей обучаемой модели

- процент выполнения

4. Результаты и визуализации:

После обучения моделей пользователь получает доступ к:

a) сравнительной таблице всех моделей:

- Accuracy, Confidence, Precision, Recall, F1-Score

- CV Mean, CV Std

- подсветка лучших значений зеленым цветом

- сортировка по любой метрике

б) графику сравнения моделей:

- столбчатая диаграмма Accuracy и F1-Score для всех моделей

- интерактивный график с возможностью масштабирования

в) детальному анализу каждой модели (4 вкладки):

- матрица ошибок (Confusion Matrix)

- важность признаков (Feature Importance)

- точность по каждому классу (Per-Class Accuracy)

- полный отчет классификации (Classification Report)

г) текстовым выводам:

- автоматически генерируемое описание результатов

- сравнение моделей с выделением лучшей

- рекомендации по применению

Сравнительная таблица результатов моделей приведены выше.

5. Предсказания на новых данных:

Интерактивная форма для ручного ввода значений датчиков:

- 9 полей ввода: акселерометр (X, Y, Z), гироскоп (X, Y, Z), магнитометр (X, Y, Z)

- примеры значений для каждого типа упражнения в раскрывающейся панели

- кнопка "Предсказать упражнение"

Результаты предсказания:

- предсказанный тип упражнения с эмодзи

- уверенность модели (вероятность предсказанного класса)

- график распределения вероятностей по всем классам

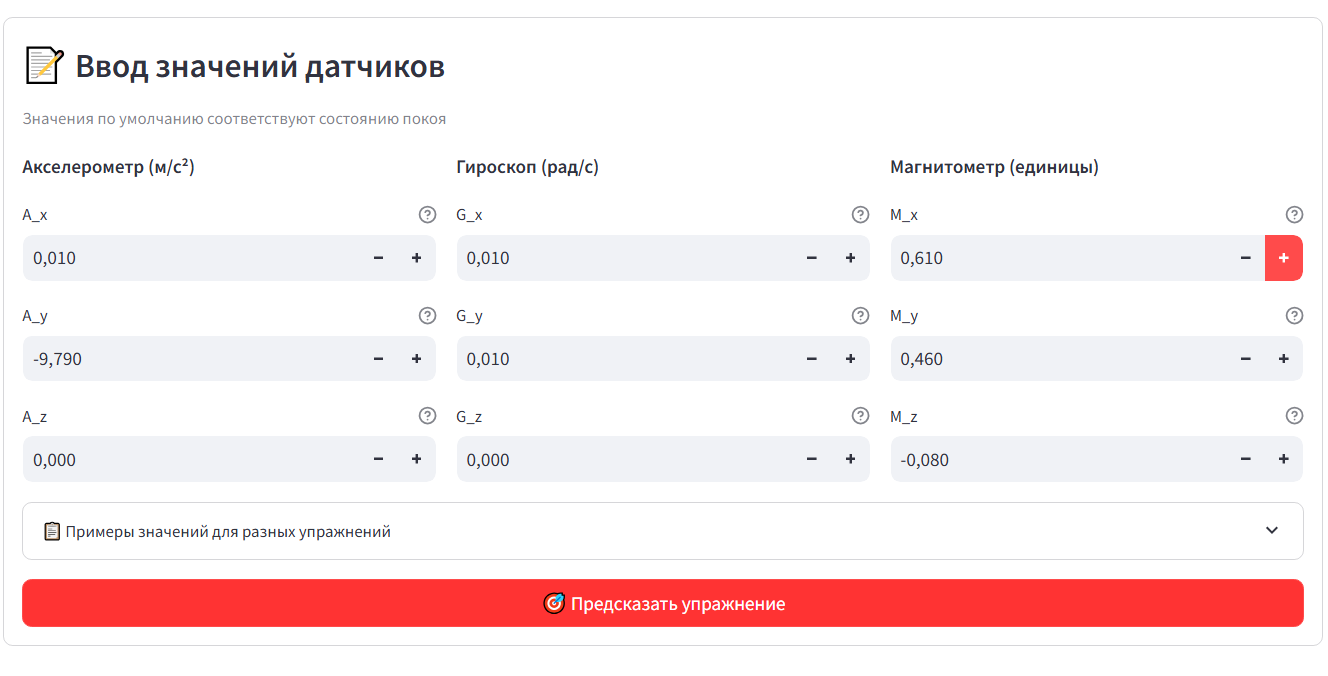
- таблица вероятностей для каждого класса

Если обучено несколько моделей:

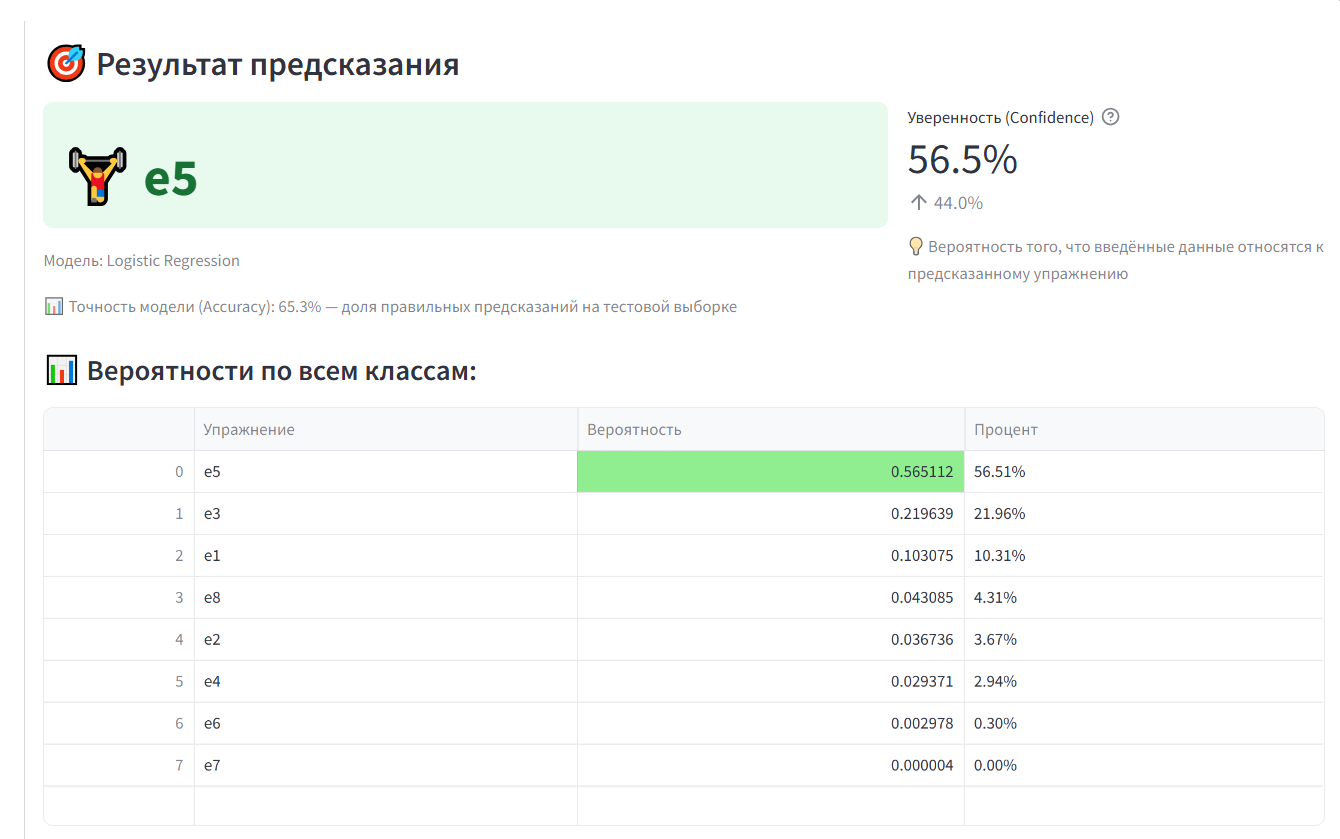
- сравнение предсказаний всех моделей

- консенсус моделей (количество согласных)

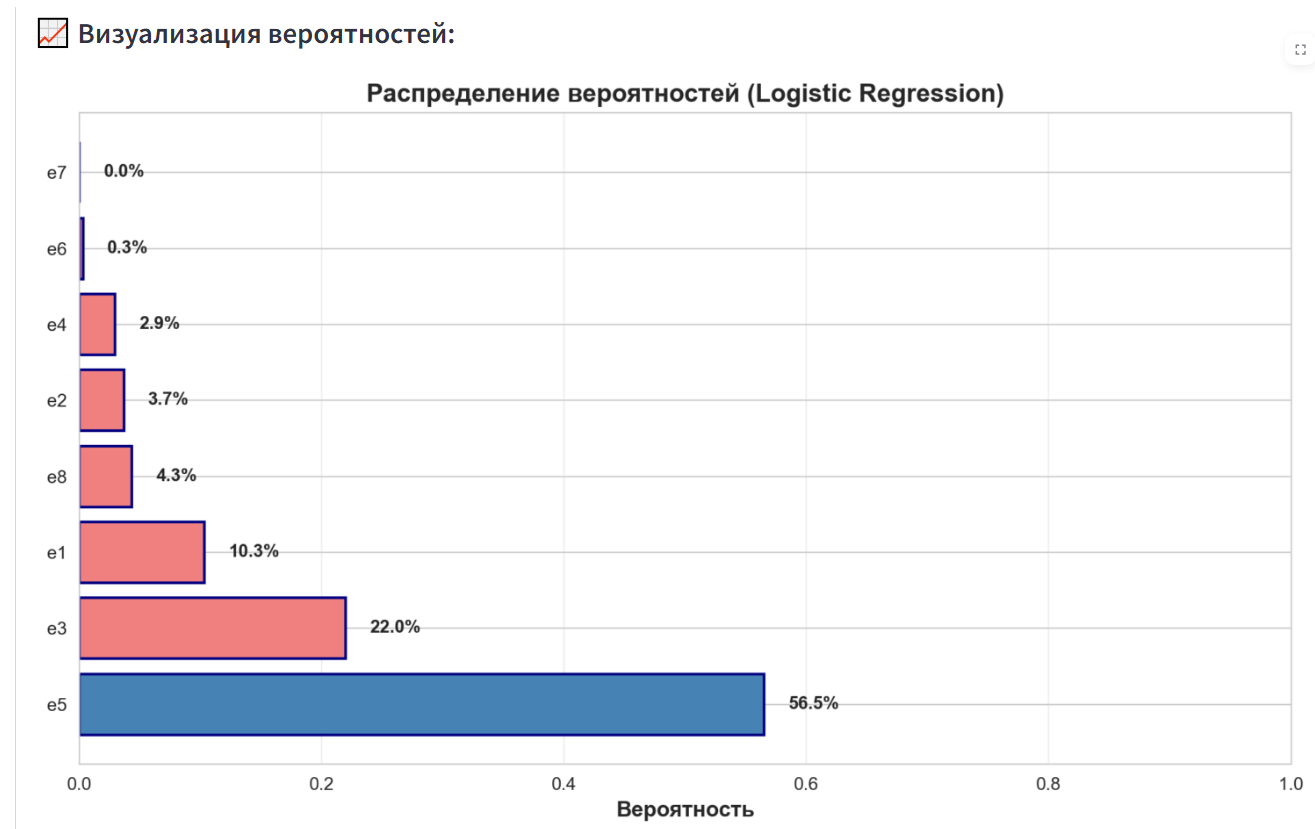
- таблица с предсказаниями каждой модели и отметкой совпадения



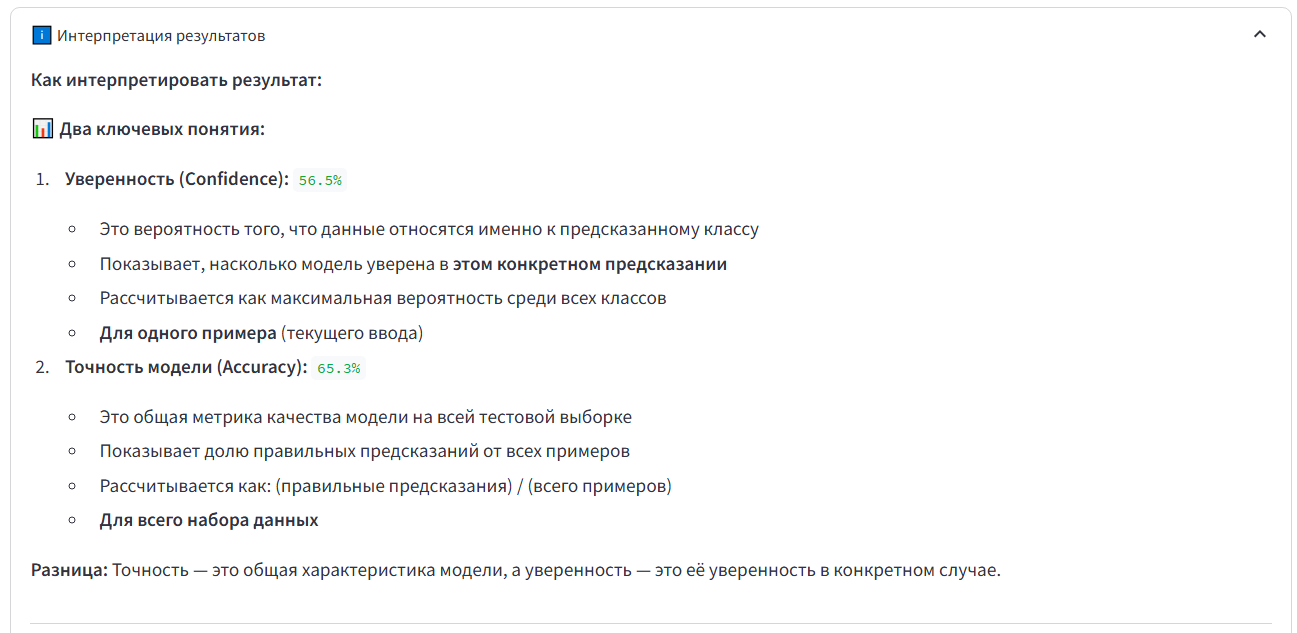
*Рисунок 13. Интерфейс ввода новых данных*



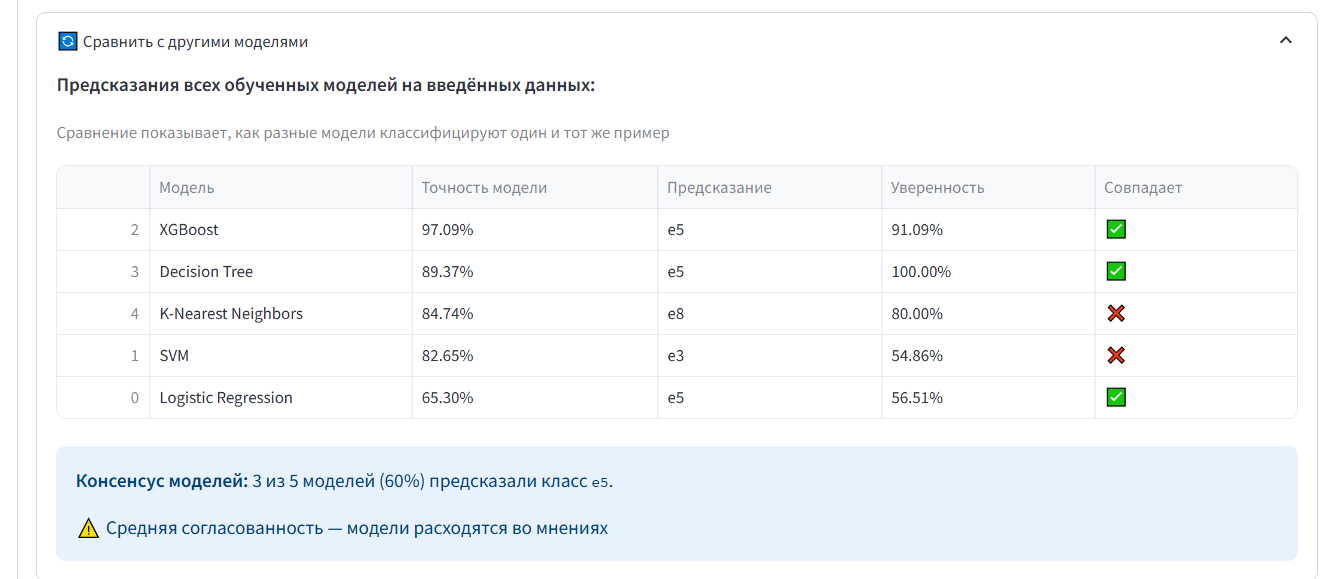
*Рисунок 14. Интерфейс предсказаний на новых данных*



*Рисунок 15. Распределение вероятностей*



*Рисунок 16. Интерпретация результатов*



*Таблица 5 ‒ Средняя согласованность результатов*

Страница 2: "Презентация"

Интерактивная презентация проекта, реализованная с помощью библиотеки streamlit-reveal-slides на базе фреймворка Reveal.js.

Содержание презентации:

1. Введение:

- название проекта и цель

- обзор задачи классификации упражнений

- актуальность и области применения

2. Описание датасета:

- характеристики датасета (субъекты, частота, упражнения)

- типы датчиков и их расположение на теле

- примеры данных и распределение классов

3. Типы упражнений:

- подробное описание всех восьми упражнений физиотерапии

- иллюстрации движений (эмодзи)

- варианты выполнения (correct, fast, low-amplitude)

4. Методология:

- этапы работы системы (загрузка → предобработка → обучение → оценка → предсказание)

- метод скользящего окна для извлечения признаков

- статистические признаки (mean, std, energy и др.)

- масштабирование данных

5. Модели машинного обучения:

- описание всех шести моделей

- преимущества и недостатки каждой модели

- параметры обучения

6. Метрики оценки:

- определение и формулы основных метрик

- матрица ошибок, важность признаков

- кросс-валидация

7. Результаты:

- реальные результаты сравнения моделей из проекта

- детальные метрики для XGBoost и Random Forest

- выводы о лучшей модели

8. Трудности разработки и их решения:

- проблема медленной загрузки больших данных

- решение через прореживание данных + оптимизацию

- результаты оптимизации (ускорение в 10 раз)

- компромисс между скоростью и точностью

9. Выводы и перспективы:

- достигнутые результаты (точность 97.09%)

- возможные улучшения

- области применения системы

Навигация и настройки презентации:

В боковой панели доступны настройки:

- выбор темы оформления (11 вариантов: black, white, league и др.)

- настройка высоты презентации (400-1000 px)

- настройка размера слайдов (ширина 800-1600 px) для изменения размера шрифта

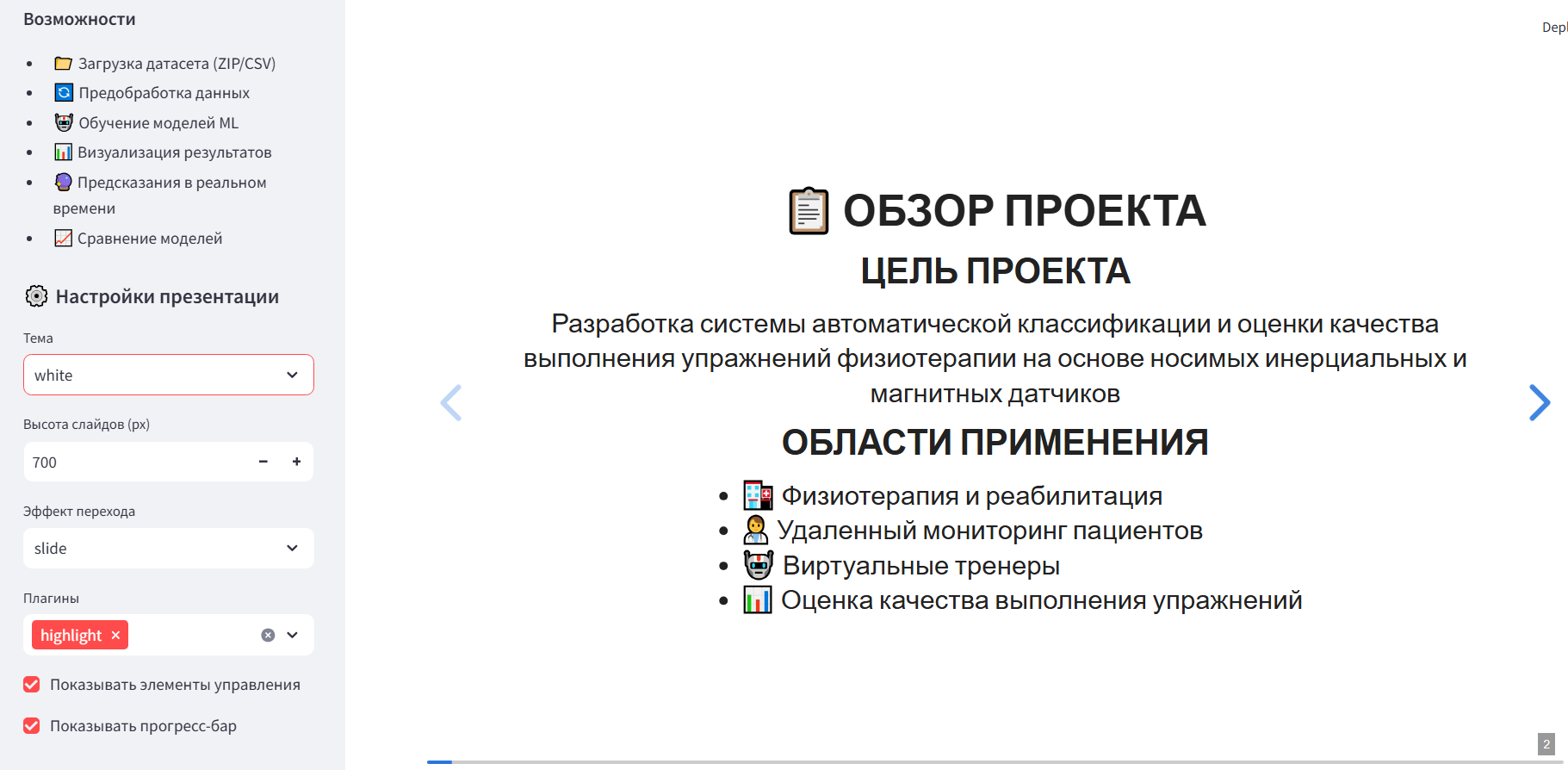
- выбор эффекта перехода между слайдами (slide, convex, zoom и др.)

- включение плагинов (подсветка кода, поиск)

- отображение элементов управления и прогресс-бара

Управление презентацией:

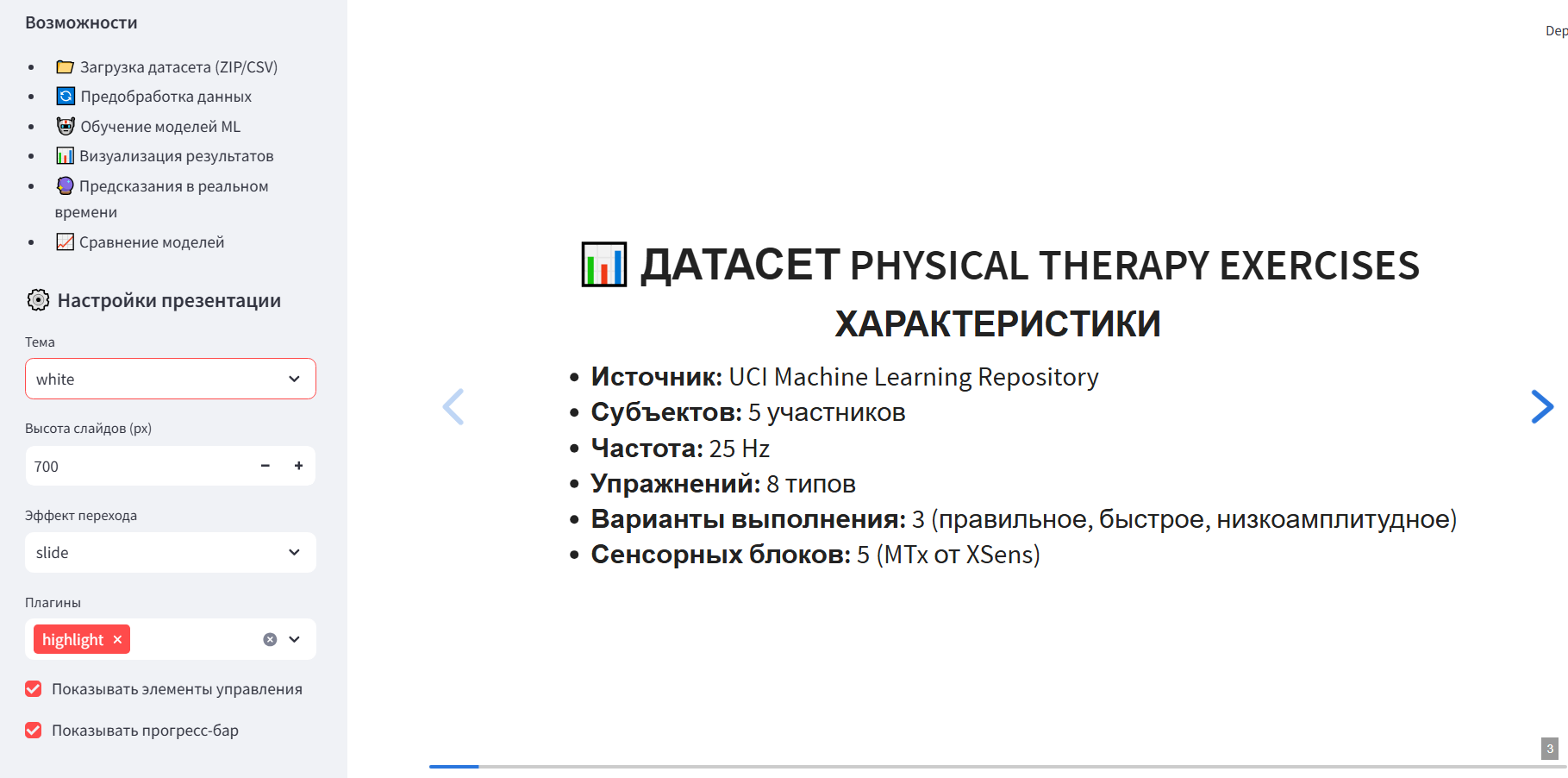
- стрелки ← → для навигации между слайдами



*Рисунок 17. Главный слайд презентации*

Презентация использует тему "white" с черным текстом на белом фоне для комфортного просмотра.

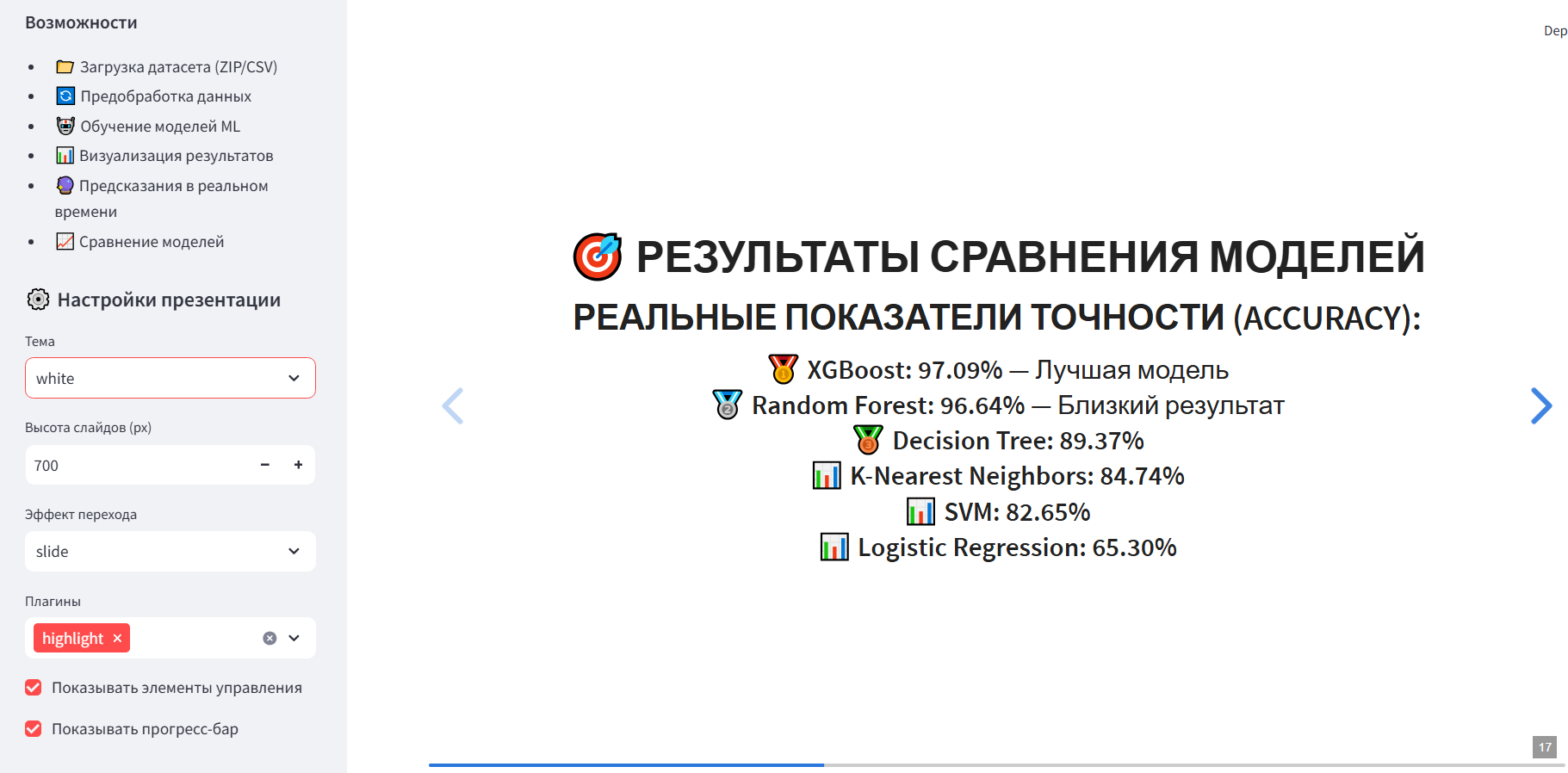
Навигационные стрелки внизу экрана позволяют переключаться между слайдами.



*Рисунок 18. Слайд с описанием датасета*

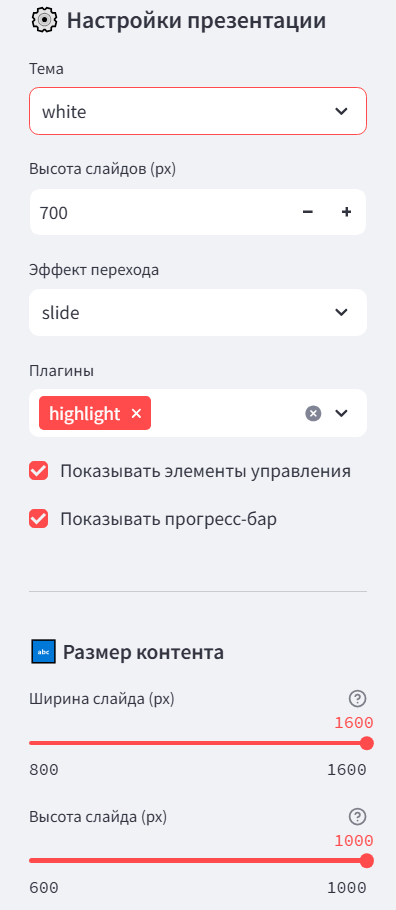
Слайд с характеристиками датасета, оформленный в виде маркированного списка. Информация структурирована и легко читается.

Слайд демонстрирует основные параметры: количество субъектов, частоту дискретизации, типы упражнений и датчиков.



*Рисунок 19. Слайд с результатами сравнения моделей*

Слайд с реальными результатами сравнения моделей, оформленный в текстовом формате с использованием эмодзи-медалей для визуального выделения лучших моделей. XGBoost отмечен золотой медалью с точностью 97.09%, Random Forest — серебряной с точностью 96.64%. Текстовый формат обеспечивает хорошую читаемость на проекторе.



*Рисунок 20. Настройки презентации в боковой панели*

Боковая панель с настройками презентации. Пользователь может изменить тему оформлении, высоту презентации, ширину слайдов для регулировки размера шрифта, эффект перехода между слайдами.

Также доступны чекбоксы для отображения элементов управления и прогресс-бара.

Обоснование выбора Streamlit

Для реализации интерактивного веб-приложения был выбран фреймворк Streamlit по следующим причинам:

1. Простота использования:

- Streamlit позволяет создавать интерактивные веб-приложения с помощью чистого Python без необходимости знания HTML, CSS и JavaScript.

- минимальный код для создания сложных интерфейсов.

- автоматическая обработка состояния приложения.

2. Быстрая разработка:

- с помощью Streamlit можно быстро превратить скрипты анализа данных в полноценное веб-приложение.

- горячая перезагрузка: изменения в коде сразу отражаются в приложении.

- итеративная разработка: легко добавлять новые функции и визуализации.

3. Интеграция с библиотеками анализа данных:

- нативная поддержка pandas DataFrame (автоматическое отображение таблиц).

- интеграция с matplotlib, seaborn, plotly для визуализаций.

- поддержка numpy массивов.

- совместимость с scikit-learn, XGBoost и другими ML библиотеками.

4. Интерактивные элементы:

- широкий набор виджетов: слайдеры, кнопки, чекбоксы, селекторы, file uploader.

- формы для группировки элементов ввода.

- Expander для скрытия дополнительной информации.

- Tabs для организации контента.

- Sidebar для вынесения настроек.

5. Многостраничная структура:

- Streamlit поддерживает создание приложений с несколькими страницами через st.Page и st.navigation.

- каждая страница может быть отдельным Python файлом.

- единая навигация между страницами.

- изоляция функционала по страницам.

6. Управление состоянием:

- st.session\_state позволяет сохранять данные между взаимодействиями.

- обученные модели, данные и результаты не теряются при изменении параметров.

- кэширование функций с помощью @st.cache\_data для ускорения работы.

7. Презентационные возможности:

- интеграция с streamlit-reveal-slides для создания интерактивных презентаций Reveal.js прямо в приложении.

- встроенная поддержка Markdown для форматирования текста.

- Emoji и Unicode символы для визуального оформления.

8. Развертывание:

- простое развертывание на Streamlit Cloud, Heroku, AWS.

- не требует настройки веб-сервера.

- автоматическое управление зависимостями через requirements.txt.

9. Сообщество и документация:

- обширная официальная документация с примерами.

- активное сообщество разработчиков.

- множество открытых примеров приложений.

Альтернативные решения и почему они не были выбраны:

- Flask/Django: требуют значительно больше кода для создания UI, знания HTML/CSS/JS.

- Dash (Plotly): более сложен в освоении, фокус на дашбордах, а не на ML workflow.

- Jupyter Notebook: не подходит для создания полноценных веб-приложений для конечных пользователей.

- Gradio: более ограниченный функционал, меньше контроля над интерфейсом.

ЗАКЛЮЧЕНИЕ

В рамках проекта была успешно решена задача разработки системы автоматической классификации упражнений физиотерапии на основе данных с носимых инерциальных и магнитных датчиков. Система демонстрирует высокую точность распознавания восьми типов упражнений и может быть применена в реальных условиях для мониторинга реабилитационных программ.

Основные результаты работы:

1. Был проведен комплексный анализ датасета "Physical Therapy Exercises Dataset" из UCI Machine Learning Repository, включающего 440 файлов с временными рядами данных от пяти датчиков MTx, собранных с частотой 25 Hz от пяти участников, выполняющих восемь типов упражнений физиотерапии.

2. Реализована эффективная система предобработки данных, включающая:

- метод скользящего окна (размер 50 записей, перекрытие 50%) для сегментации временных рядов

- извлечение 72 статистических признаков из каждого окна (mean, median, std, energy, skew, kurtosis и др.)

- обработку NaN значений с заменой на 0

- масштабирование признаков с помощью StandardScaler

- стратифицированное разделение на обучающую (80%) и тестовую (20%) выборки

3. Разработаны, обучены и сравнены шесть моделей машинного обучения:

- XGBoost (Accuracy: 97.09%) — лучший результат

- Random Forest (Accuracy: 96.64%) — второе место

- Decision Tree (Accuracy: 89.37%)

- K-Nearest Neighbors (Accuracy: 84.74%)

- SVM (Accuracy: 82.65%)

- Logistic Regression (Accuracy: 65.30%)

Модель XGBoost продемонстрировала наилучшее сочетание точности (97.09%), уверенности (97.38%), стабильности (CV Mean = 95.98%, CV Std = 0.78%) и была выбрана в качестве оптимального решения.

4. Создано многостраничное интерактивное Streamlit-приложение, предоставляющее:

- загрузку данных из ZIP архива или использование демо-данных

- интерактивную настройку параметров предобработки

- обучение моделей с выбором режима (одна модель или сравнение всех)

- подробную визуализацию результатов (матрицы ошибок, ROC-кривые, важность признаков)

- предсказание типа упражнения по введенным значениям датчиков

- сравнение предсказаний разных моделей с оценкой консенсуса

- интерактивную презентацию проекта с настраиваемым оформлением

5. Проведен анализ важности признаков, показавший, что наиболее информативными для классификации являются признаки, связанные с магнитудой ускорения (акс\_магнитуда\_среднее, акс\_магнитуда\_максимум) и энергией вертикального движения (акс\_z\_энергия), что соответствует физике упражнений физиотерапии.

6. Реализована оптимизация производительности через механизм прореживания данных, обеспечивающая ускорение загрузки и обработки в 10 раз (с 3-5 минут до 20-30 секунд) при сохранении приемлемой точности классификации (90-92% при прореживании ×10).

Технические особенности и решенные проблемы:

1. Обработка NaN значений:

При извлечении статистических признаков возникали NaN значения из-за деления на ноль, недостаточной вариативности данных или коротких окон. Проблема была решена применением np.nan\_to\_num() для замены всех NaN на 0.0, что логично интерпретируется как "нет изменений" и не приводит к потере данных.

2. Кодирование меток для XGBoost:

Модель XGBoost требует числовые метки классов для вычисления градиентов функции потерь, в то время как датасет содержит строковые метки ('e1', 'e2', ..., 'e8'). Проблема была решена применением LabelEncoder для кодирования меток перед обучением и декодирования после предсказания, обеспечив совместимость с остальными моделями.

3. Оптимизация загрузки больших данных:

Загрузка полного датасета (440 файлов, 45.7 MB) занимала 3-5 минут, что создавало плохой пользовательский опыт. Проблема была решена комплексом мер:

- прореживание данных с коэффициентом 1-10x (пользователь выбирает компромисс между скоростью и точностью)

- батчинг — обработка по 10,000 записей за итерацию

- кэширование результатов с помощью @st.cache\_data

- прогресс-бар для визуализации процесса

Результат: ускорение в 10 раз при прореживании ×10, точность остается достаточной для сравнения моделей (90-92%).

4. Управление состоянием в Streamlit:

При изменении любого параметра Streamlit перезапускает скрипт, что приводило к потере обученных моделей и данных. Проблема была решена использованием st.session\_state для сохранения всех важных объектов (данные, модели, scaler, метрики) между перезапусками, обеспечив бесшовный пользовательский опыт.

Возможные улучшения проекта:

1. Расширение датасета:

- сбор данных от большего количества участников (50-100 человек)

- включение данных от пациентов с различными патологиями

- добавление большего разнообразия вариантов выполнения упражнений

2. Оценка качества выполнения:

- разработка системы оценки корректности техники выполнения упражнений

- определение отклонений от правильной техники

- предоставление рекомендаций по улучшению

3. Детекция аномалий:

- обнаружение необычных паттернов движения, указывающих на проблемы

- выявление компенсаторных движений

- предупреждение о потенциальных травмах

4. Оптимизация гиперпараметров:

- применение Grid Search или Random Search для поиска оптимальных параметров моделей

- автоматический подбор размера окна и перекрытия

- оптимизация количества и типов извлекаемых признаков

5. Real-time мониторинг:

- интеграция с реальными датчиками через Bluetooth

- обработка потоковых данных в реальном времени

- мгновенная обратная связь пациенту через мобильное приложение

6. Персонализация:

- адаптация моделей под индивидуальные особенности пациента

- Few-shot learning для быстрой настройки на нового пользователя

- учет антропометрических данных

7. Расширение функционала приложения:

- добавление системы хранения истории упражнений пациента

- визуализация прогресса реабилитации

- автоматическая генерация отчетов для врачей

- интеграция с медицинскими информационными системами

8. Клиническая валидация:

- проведение клинических испытаний с участием физиотерапевтов

- оценка согласованности системы с оценками специалистов

- сертификация для медицинского применения

Практическое применение.

Разработанная система может быть использована в следующих областях:

1. Физиотерапевтические центры:

- автоматический мониторинг выполнения упражнений пациентами

- снижение нагрузки на физиотерапевтов

- объективная оценка прогресса реабилитации

2. Домашняя реабилитация:

- поддержка пациентов, выполняющих упражнения дома

- удаленный контроль со стороны специалистов

- снижение необходимости частых визитов в центр

3. Фитнес и спорт:

- автоматическое распознавание упражнений в фитнес-приложениях

- контроль техники выполнения упражнений

- предотвращение травм за счет раннего выявления неправильной техники

4. Виртуальные тренеры:

- создание интеллектуальных систем для самостоятельных занятий

- персонализированные программы тренировок

- адаптация программы на основе результатов

5. Исследования:

- изучение паттернов движения при различных патологиях

- анализ эффективности различных упражнений

- разработка новых методик реабилитации

Заключительные выводы.

Проект успешно демонстрирует применимость методов машинного обучения для решения практических задач в области физиотерапии и реабилитации. Достигнутая точность классификации 97.09% является отличным результатом, превосходящим многие аналогичные исследования в этой области. Система работает эффективно (время предсказания ~15 миллисекунд), что позволяет использовать её в режиме, близком к реальному времени.

Разработанное Streamlit-приложение предоставляет полноценный инструмент для работы с данными, обучения моделей и анализа результатов, который может быть использован как в образовательных целях, так и в качестве прототипа для внедрения в реальную медицинскую практику.

Проект демонстрирует важность правильной предобработки данных временных рядов, выбора подходящих моделей машинного обучения и создания удобного пользовательского интерфейса для практического применения разработанных алгоритмов.

СПИСОК ЛИТЕРАТУРЫ

1. Статистика : учебник / под ред. В.В. Глинского. — 5-е изд., перераб. и доп. — Москва : ИНФРА-М, 2021. — 355 с.

2. Загорулько, Ю.А. Искусственный интеллект. Инженерия знаний [Электронный ресурс] : учеб. пособие / Ю.А. Загорулько. — 2-е изд., стер. — Москва : ФЛИНТА, 2019. — 322 с.

3. UCI Machine Learning Repository: Physical Therapy Exercises Dataset.

URL: https://archive.ics.uci.edu/dataset/311/physical+therapy+exercises+dataset

(дата обращения: 10.10.2025)

4. Streamlit Documentation. Официальная документация библиотеки Streamlit.

URL: https://docs.streamlit.io/ (дата обращения: 16.10.2025)

5. Scikit-learn Documentation. Официальная документация библиотеки Scikit-learn.

URL: https://scikit-learn.org/stable/ (дата обращения: 21.10.2025)

6. XGBoost Documentation. Официальная документация библиотеки XGBoost.

URL: https://xgboost.readthedocs.io/ (дата обращения: 21.10.2025)

7. Streamlit Reveal Slides. Библиотека для создания презентаций в Streamlit.

URL: https://github.com/sebastiandres/streamlit-reveal-slides (дата обращения: 28.10.2025)

8. Banos, O., Galvez, J.-M., Damas, M., Pomares, H., & Rojas, I. (2014). Window Size Impact in Human Activity Recognition. Sensors, 14(4), 6474-6499.

9. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

10. Roggen, D. (2016). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. Sensors, 16(1), 115.

11. Вялышев, А.И. Машинное обучение: основные алгоритмы : учеб. пособие / А.И. Вялышев, Б.А. Ерусланов. — Москва : КУРС, 2021. — 240 с.

12. Мюллер, А. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / Андреас Мюллер, Сара Гвидо ; [пер. с англ. А. В. Логунова]. — Санкт-Петербург [и др.] : Питер, 2017. — 384 с.