Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

ИССЛЕДОВАНИЕ НЕЙРОКОМПЬЮТЕРНЫХ ИНТЕРФЕЙСОВ

Отчет о выполнении этапа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ группового

проектного обучения (ГПО)

Проект ГПО АОИ-2307

|  |  |
| --- | --- |
|  | Ответственный исполнитель  студент гр. 422-3  \_\_\_\_\_\_\_\_\_ В.Е. Заборонок  «\_\_\_\_\_»\_\_\_\_\_\_\_\_ 2025 |
|  | Проверил:  Руководитель проекта ассистент каф. АОИ  \_\_\_\_\_\_ \_\_\_\_\_\_\_ Р.С. Кульшин  (оценка)  «\_\_\_\_\_»\_\_\_\_\_\_\_\_ 2025 |
|  | Руководитель проекта ассистент каф. АОИ  \_\_\_\_\_\_ \_\_\_\_\_\_\_ И.Д. Тикшаев  (оценка)  «\_\_\_\_\_»\_\_\_\_\_\_\_\_ 2025 |
|  | Принял:  Ответственный за ГПО на  кафедре ст. преподаватель каф. АОИ  М.А. Шишанина  «\_\_\_\_\_» \_\_\_\_\_\_\_ 2025 |

Исполнители проекта ГПО АОИ-2307

|  |  |
| --- | --- |
| Студент гр.422-3 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Е. Заборонок |
| Студент гр.422-3 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Поломошнова |
| Студент гр.422-1 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Сахнов |
| Студент гр.423-2 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Мельниченко |
| Студент гр.423-1 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Сираполко |
| Студент гр.423-2 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.Д. Чернышёв |
| Студент гр.423-3 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Охина |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc199772780)

[1 ОБЗОР ТЕХНОЛОГИИ НЕЙРОИНТЕРФЕЙСА 7](#_Toc199772781)

[1.1 Нейрокомпьютерный интерфейс 7](#_Toc199772782)

[2 ИСПОЛЬЗУЕМЫЙ ИНСТРУМЕНТАРИЙ 9](#_Toc199772783)

[2.1 Neiry Headband 9](#_Toc199772784)

[2.2 Языки программирования 10](#_Toc199772785)

[2.3 Capsule API 10](#_Toc199772786)

[2.4 Используемые модели машинного обучения 15](#_Toc199772787)

[3 ЭЛЕКТРОЭНЦЕФАЛОГРАФИЯ 18](#_Toc199772788)

[3.1 Параметры ЭЭГ 18](#_Toc199772789)

[3.2 Основные ритмы мозговой активности 19](#_Toc199772790)

[3.3 Артефакты электроэнцефалограммы 20](#_Toc199772791)

[4 ПРЕДШЕСТВУЮЩАЯ РАБОТА 21](#_Toc199772792)

[5 ПОЛИГРАФ 23](#_Toc199772793)

[5.1 Применение неинвазивных нейроинтерфейсов для обнаружения лжи 24](#_Toc199772794)

[5.2 Практические примеры и перспективы использования нейроинтерфейсов в детекции лжи 24](#_Toc199772795)

[5.3 Исследуемые метрики 25](#_Toc199772796)

[5.4 Проверка релевантности данных 26](#_Toc199772797)

[5.5 Проведение экспериментов 26](#_Toc199772798)

[5.6 Вычисления и анализ 28](#_Toc199772799)

[5.6.1 Статистический анализ с использованием инструментов Excel 28](#_Toc199772800)

[5.6.2 Анализ с применением машинного обучения 31](#_Toc199772801)

[5.7 Визуализация работы полученных моделей 47](#_Toc199772802)

[5.8 Выводы 50](#_Toc199772803)

[6 НАПИСАННЫЕ СТАТЬИ 52](#_Toc199772804)

[6.1 Использование эмоционального отклика для оценки эффективности рекламы 52](#_Toc199772805)

[ЗАКЛЮЧЕНИЕ 54](#_Toc199772806)

[СПИСОК ЛИТЕРАТУРЫ 57](#_Toc199772807)

[ПРИЛОЖЕНИЕ А 60](#_Toc199772808)

[Листинг кода приложения для считывания данных 60](#_Toc199772809)

# ВВЕДЕНИЕ

Нейрокомпьютерные интерфейсы (НКИ) находят всё более широкое применение в различных областях науки и практики. Их используют в медицине для диагностики и реабилитации пациентов, в образовании — при разработке адаптивных систем обучения, а также в игровой индустрии и маркетинге — для анализа эмоционального состояния пользователя. Одним из перспективных направлений применения НКИ стало исследование когнитивных процессов, включая выявление ложных реакций по активности мозга.

Традиционные методы детекции лжи, такие как полиграф, основываются на регистрации физиологических реакций организма — изменения пульса, дыхания, кожной проводимости и других параметров. Однако такие показатели могут быть нестабильными и зависеть от множества факторов, не связанных с самой ложью. Это делает их недостаточно надёжными в научном и практическом плане. Современные исследования демонстрируют, что анализ нейронной активности позволяет получать более объективные данные о процессах принятия решений и сокрытия информации.

Актуальность темы: Анализ активности мозга с целью выявления лжи представляет собой важное направление в области нейротехнологий. С развитием неинвазивных методов регистрации мозговой активности, таких как электроэнцефалография (ЭЭГ) и нейроинтерфейсы, становится возможным создание компактных и точных систем определения достоверности ответов. Такие технологии могут найти применение в судебной практике, безопасности, психологии и других сферах, где важно отделить правду от обмана. В связи с этим разработка и тестирование алгоритмов распознавания лжи на основе данных мозговой активности является актуальной задачей.

Основная цель проекта – разработка и экспериментальная проверка методики выявления лжи на основе анализа активности мозга с использованием неинвазивного нейрокомпьютерного интерфейса.

Задачи проекта:

1. Выбор темы. На этом этапе будет проведен брейншторм с командой для выбора исследуемой темы;

2. Накопление теоретической базы. На этом этапе будет проведен сбор информации по теме исследования, а также поиск и анализ существующих статей;

3. Сбор данных. На этом этапе будет организована серия экспериментов с участием испытуемых;

4. Анализ данных и интерпретация результатов. На этом этапе будет проведен анализ полученных данных, а также будут сделаны выводы по результатам анализа.

Объект и предмет исследования: Объектом исследования являются процессы формирования ложных и правдивых ответов у человека, сопровождающиеся изменениями в нейронной активности. Предметом исследования – показатели мозговой активности, регистрируемые с помощью нейрокомпьютерного интерфейса, и их корреляция с достоверностью высказываний испытуемых.

# ОБЗОР ТЕХНОЛОГИИ НЕЙРОИНТЕРФЕЙСА

## Нейрокомпьютерный интерфейс

Нейроинтерфейс (или интерфейс «мозг – компьютер») – так называется устройство для обмена информацией между мозгом и внешним устройством. В качестве объекта управления может выступать не только компьютер, но и любое другое электронное устройство: квадрокоптер, система «умного дома», промышленный робот или боевой дрон, экзоскелет и даже искусственные органы чувств [1].

Нейроинтерфейсы представляют собой передовые технологии, которые позволяют устанавливать прямую связь между мозгом и внешними устройствами. Они способны записывать и интерпретировать электрические сигналы, генерируемые нейронами, и передавать их на различные устройства, такие как протезы и компьютеры. В активном режиме нейроинтерфейсы могут использоваться для управления протезами или другими механизмами, предоставляя пользователям возможность выполнять сложные задачи с помощью мысли. В пассивном режиме они собирают данные о мозговой активности, что позволяет анализировать состояние пользователя и принимать решения на основе этих данных [2].

Существуют три основных типа нейроинтерфейсов: неинвазивные, полуинвазивные и инвазивные. Инвазивные внедряются в ткань мозга пациента с помощью сложной хирургической операции. Из-за высоких рисков инвазивные НКИ пока что используются только когда другие опции недоступны – например, для пациентов в тяжелых состояниях, таких как паралич и другие нейромышечные нарушения. Полуинвазивные, как и инвазивные, имеют прямой контакт с мозгом, но только на его поверхности, без внедрения в ткань [3].

Неинвазивные нейроинтерфейсы исходя из названия не требуют хирургического вмешательства, и следовательно наиболее безопасны в эксплуатации, и, как правило, менее дорогие в использовании. Однако, неинвазивные ИМК из-за естественной преграды для любого сигнала в виде костей черепа, мышц и кожи регистрируют меньше информации и, следовательно, являются менее точными для ежедневной эксплуатации [4].

Одним из неинвазивных методов исследования биоэлектрической активности головного мозга является электроэнцефалография (ЭЭГ).

# ИСПОЛЬЗУЕМЫЙ ИНСТРУМЕНТАРИЙ

## Neiry Headband

Neiry Headband – это нейроинтерфейс в виде повязки на голову, оснащенной четырьмя мягкими подвижными электродами для записи электрической активности мозга (рисунок 2.1). Электроды покрыты золотом, что делает их одним из лучших вариантов для сухого сбора ЭЭГ. Устройство работает с различными ритмами мозга, такими как Тета, Альфа, SMR и Бета. Компания Neiry предоставляет программное обеспечение Capsule и Capsule API для взаимодействия с нейроинтерфейсом.



Рисунок 2.1 – Neiry Headband

Capsule – универсальное ПО для работы с данными с устройств Neiry. Данная программа позволяет отображать данные со всех датчиков в реальном времени, выводить спектральную мощность Тета, Альфа, SMR, Бета-ритмов в диапазоне 4-35 Гц (рисунок 2.2), записывать «сырые» данные в файл формата h5 с возможностью разметки.

## Языки программирования

С++ – компилируемый, статически типизированный язык программирования общего назначения [5]. В данном проекте С++ использовался для:

* управления логикой взаимодействия с нейроинтерфейсом через CapsuleAPI;
* предварительная обработка данных, полученных от устройства;

Python – мультипарадигмальный высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ [6]. Использовался веб-фреймворк для создания API - FastAPI. В созданном API используется предобученная модель машинного обучения, классифицирующая данные.

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили [7]. Использовался со сборщиком Gulp, а также HTML5/CSS3 для структуры и стилизации интерфейса.

## Capsule API

CapsuleAPI – специализированная динамическая библиотека для взаимодействия с нейроинтерфейсами, предоставляемая компанией Neiry. Использовалась для снятия данных и их предварительной обработки. Структура API представлена на рисунках 2.1 – 2.4.

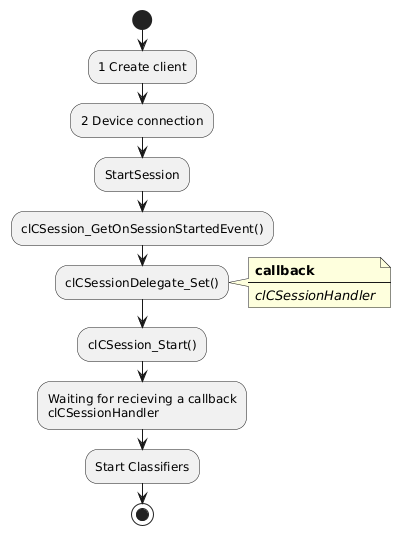


Рисунок 2.1 – Диаграмма деятельности системы

Процесс начинается с создания клиента, после чего устанавливается соединение с устройством. Затем запускается сеанс, инициируются обработчики событий с помощью cICSessionDelegate\_Set(), а также вызывается метод cICSession\_Start(). После этого система ожидает обратный вызов от cICSessionHandler, после которого запускаются классификаторы.

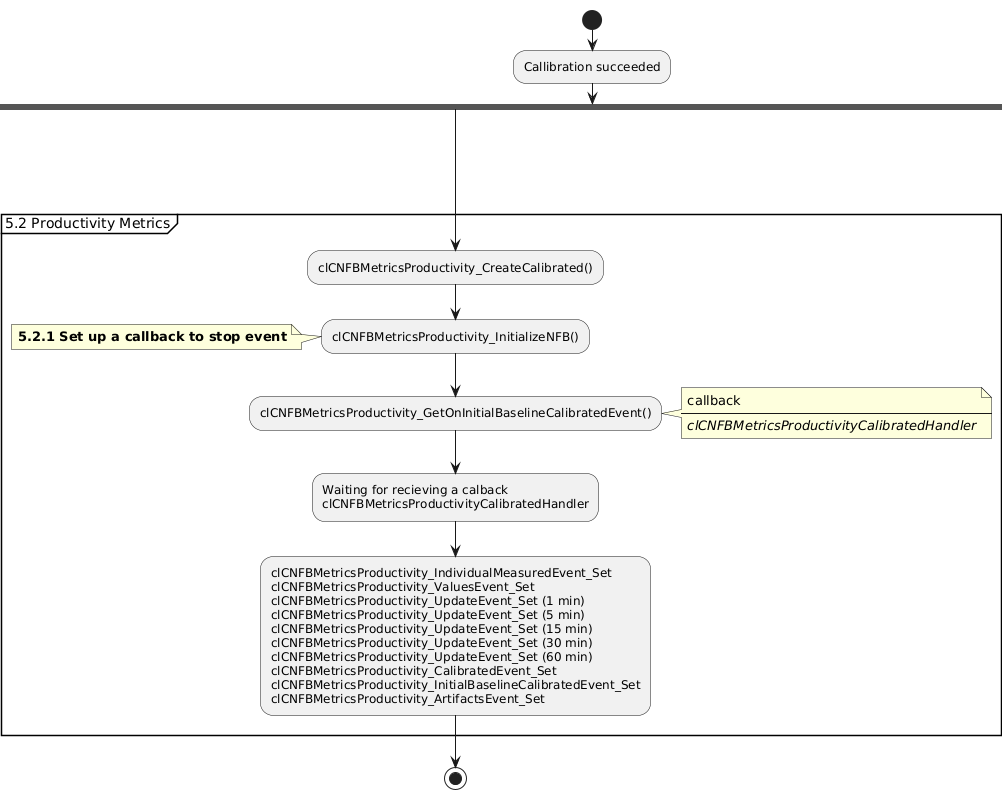


Рисунок 2.2 – Диаграмма деятельности классификатора продуктивности

После успешной калибровки создаётся калибратор, назначаются функции обратного вызова. После обратного вызова происходит настройка параметров действий с метриками производительности.

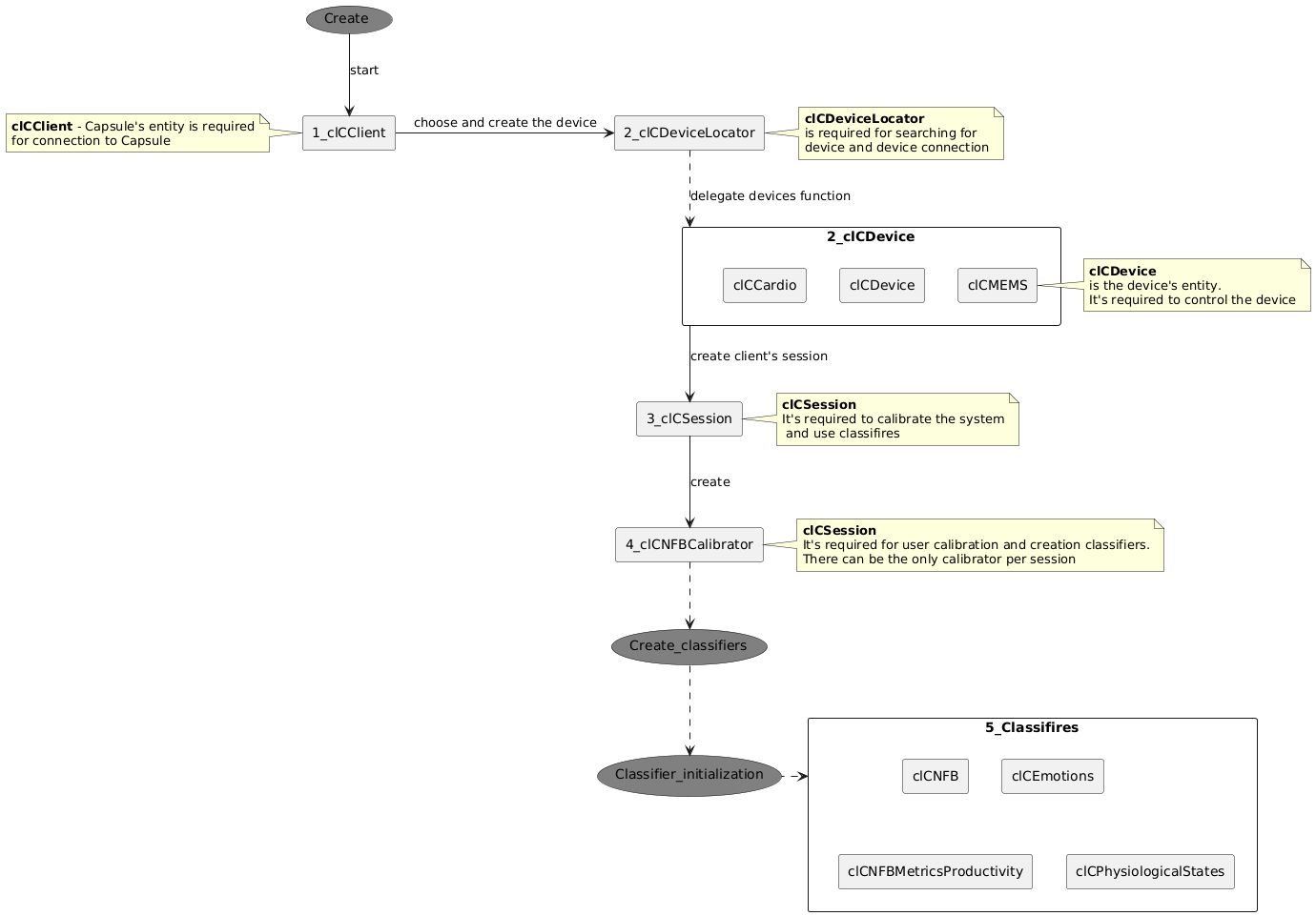


Рисунок 2.3 – Блок-схема работы API

Общая схема устройства API. Процесс начинается с создания клиента, после происходит обнаружение девайса, создание сессии, калибровка данных и прочее.

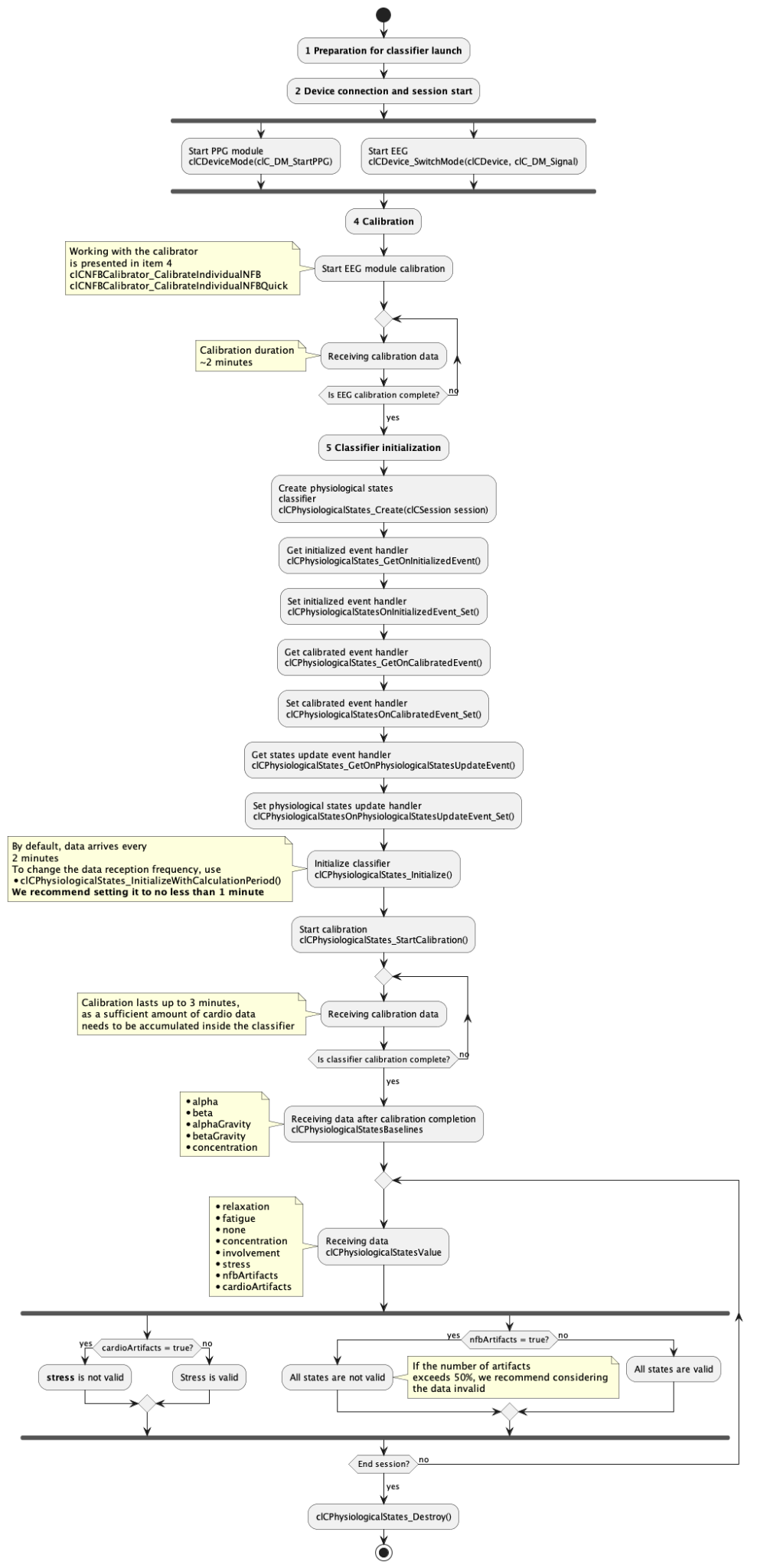


Рисунок 2.4 – Блок-схема получения физиологических данных

Процесс начинается с подготовки и запуска устройства, после чего происходит калибровка модулей PPG и EEG. Далее инициализируется классификатор физиологических состояний, который получает данные, проверяет их на артефакты и передает результаты, завершается освобождением ресурсов при необходимости.

Помимо функциональных возможностей, вместе с API предоставляются три готовых решения, которые позволяют снимать сырые и отфильтрованные данные. В качестве основы для написания кода под нужды данного группового проекта было взято одно из готовых решений, в котором был реализован следующий функционал:

* подключение нейроинтерфейса;
* калибровка нейроинтерфейса;
* начало сессии снятия данных;
* вывод сырых данных в консоль;

В это решение был добавлен следующий функционал:

* считывание сердечного ритма;
* считывание индекса Каплана;
* считывание показателя стресса;
* запись ритмов головного мозга, сердечного ритма, индекса Каплана, показателя стресса в файл;
* функционал для отладки и отслеживания ошибок работы программы.

Весь этот функционал позволил достичь цели данного группового проекта. Листинг кода программы представлен в приложении А.

## Используемые модели машинного обучения

Logistic Regression (Логистическая регрессия) – статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события, она выдаёт ответ в виде числа в промежутке от 0 до 1 [8]. Бинарная логистическая регрессия применяется в случае, когда зависимая переменная является бинарной (может принимать только два значения). С помощью логистической регрессии можно оценивать вероятность наступления события.  
 Random forest (Случайный лес) – метод случайного леса строится на основе ансамбля деревьев решений. Основная идея заключается в том, чтобы создать множество деревьев решений и объединить их результаты для получения более точного прогноза [9].

XGBoost (Extreme Gradient Boosting) – это реализация градиентного бустинга, ориентированная на высокую производительность, масштабируемость и точность. Этот алгоритм машинного обучения активно используется на практике в задачах классификации, регрессии и ранжирования. Его популярность обусловлена стабильностью, скоростью работы и превосходной способностью обобщать данные даже в условиях ограниченного набора признаков [10].

CatBoost – открытая программная библиотека, разработанная компанией Яндекс и реализующая уникальный патентованный алгоритм построения моделей машинного обучения, использующий одну из оригинальных схем градиентного бустинга [11].

LightGBM (Light Gradient Boosted Machine) – библиотека с открытым исходным кодом, которая предоставляет эффективную и действенную реализацию алгоритма градиентного бустинга. LightGBM расширяет алгоритм градиентного бустинга, добавляя тип автоматического выбора объектов, а также фокусируясь на примерах бустинга с большими градиентами [12].

AdaBoost (сокращение от Adaptive Boosting – адаптивный бустинг) – модель ансамблевого машинного обучения, которая пытается создать сильный классификатор из слабых классификаторов. Алгоритм работает путём итеративного обучения слабых учеников (обычно простых моделей, таких как пни решений) на выборке, корректируя веса неправильно классифицированных экземпляров на каждой итерации [13].

BaggingClassifier – это ансамблевый метод, реализующий технику бэггинга. Он объединяет несколько моделей и обучает их на различных подвыборках исходных данных с возвращением. Полученные предсказания агрегируются, что позволяет снизить дисперсию модели и повысить устойчивость к переобучению [14].

Autogluon – библиотека, которая делает автоматическими такие операции, как настройка гиперпараметров и поиск нейронной архитектуры. AutoGluon построен на основе ML-библиотеки Gluon, которая создает модели из набора готовых оптимизированных элементов. Он работает с данными таблиц, изображениями и текстом, способен выполнять задачи классификации, регрессии, а также более сложные задания [15].

# 

# ЭЛЕКТРОЭНЦЕФАЛОГРАФИЯ

Основным способом получения данных о работе мозга является измерение электроэнцефалографии (ЭЭГ).

Электроэнцефалография (ЭЭГ) – это метод исследования функциональной активности головного мозга путем регистрации его электрических импульсов. ЭЭГ проводится с помощью электродов, размещаемых на поверхности головы, которые фиксируют электрические сигналы, возникающие в нейронах мозга. Эти сигналы усиливаются и отображаются в виде графика – электроэнцефалограммы.

В области исследований, основанных на физиологических сигналах, ЭЭГ – это спонтанный, несубъективный физиологический сигнал, который может объективно отражать эмоциональные состояния человека.

## Параметры ЭЭГ

Основными видами активности, которые записываются в ходе процедуры и впоследствии подвергают интерпретации, а также дальнейшему изучению считаются волновые частота, амплитуда и фаза.

Частота– это количество колебаний электрического сигнала мозга за одну секунду. Она измеряется в герцах (Гц) и характеризует скорость изменения электрической активности нейронов. Частота ритмов указывает на состояние мозга, например, низкие частоты связаны с глубоким сном, а высокие – с активным бодрствованием.

Амплитуда– это разница потенциалов между максимумом и минимумом электрического сигнала, регистрируемого в процессе ЭЭГ. Измеряется расстоянием между пиками волн в противоположных фазах и выражается в микровольтах (мкВ).

Фаза– это положение сигнала в рамках одного цикла колебания, которое измеряется относительно начала этого цикла. Она описывает относительное состояние волны (например, пик, спад, нулевое пересечение) в конкретный момент времени. Фаза используется для анализа временных соотношений между разными ритмами или областями мозга.

## Основные ритмы мозговой активности

Ритмы мозговой активности – это электрические колебания, которые отражают характерные состояния мозга. При расшифровке показателей ритма ЭЭГ головного мозга вносятся его частота, соответствующая состоянию участка мозга, амплитуда, и характерные его изменения при функциональных сменах активности. Каждый ритм связан с определёнными физиологическими и когнитивными процессами.

Тета-ритм определяется на частотах 4-8 Гц. Ассоциируется с состоянием расслабления, мечтательности и лёгкой сонливости. Этот ритм также играет роль в процессе обработки памяти и эмоциональных реакциях. У детей тета-ритм часто регистрируется в бодрствующем состоянии, тогда как у взрослых его выраженность может свидетельствовать о стрессовых состояниях, нарушениях внимания или патологии мозга.

Альфа-ритм определяется на частотах 8-13 Гц. проявляется в состоянии спокойного бодрствования с закрытыми глазами. Он локализуется преимущественно в затылочной области и подавляется при открытии глаз или активной умственной деятельности. Альфа-ритм связан с состоянием расслабления, отсутствием внешних стимулов и внутренним самовосприятием. Нарушения в альфа-ритме могут указывать на тревожность, депрессию или органические повреждения мозга.

Бета-ритм определяется на частотах 13-30 Гц. Он характеризуется высокой частотой и относительно низкой амплитудой. Этот ритм ассоциируется с активной умственной деятельностью, концентрацией и эмоциональным возбуждением. Он наиболее выражен в лобных областях. Усиление бета-ритма может быть связано с тревожностью, употреблением некоторых медикаментов или реакцией на стимуляцию.

## Артефакты электроэнцефалограммы

Артефакты электроэнцефалограммы – это нежелательные сигналы, которые появляются в записи ЭЭГ и не связаны с электрической активностью мозга. Они могут искажать данные, затрудняя анализ и интерпретацию. Артефакты могут быть вызваны внешними источниками, движениями пациента, активностью мышц или техническими проблемами с оборудованием.

Различают несколько типов артефактов. Физиологические артефакты возникают из-за активности, связанной с телом человека. Примеры включают артефакты движения глаз (например, при моргании), которые проявляются как медленные волны в передних отведениях, и мышечные артефакты, связанные с напряжением мышц лица, создающие высокочастотные колебания. Также сюда относятся артефакты, вызванные дыханием, движением языка или сердечной активностью, которая может проявляться как низкочастотный сигнал, синхронизированный с пульсом. Нефизиологические артефакты связаны с внешними источниками помех. Примером являются электрические помехи от оборудования, например, от мобильных телефонов или других электронных устройств. Также сюда относятся помехи от неплотного контакта электродов с кожей или изменения сопротивления из-за потоотделения.

Для минимизации артефактов принимаются различные меры. Это может включать правильную установку электродов, инструктаж участника эксперимента о минимизации движений, а также использование специализированного программного обеспечения для фильтрации артефактов. Однако полностью исключить их невозможно, поэтому необходимо учитывать их влияние при анализе результатов ЭЭГ.

# ПРЕДШЕСТВУЮЩАЯ РАБОТА

Предыдущий семестр для команды проекта начался с поиска новой темы для исследования. Для этого был проведён мозговой штурм – каждый участник проекта предлагал несколько тем, которые потенциально можно было развить с использованием нейроинтерфейса. После выдвижения тем был произведён анализ реализуемости каждой. Таким образом было выбрано две темы – изучение эмоциональной реакции на аудиовизуальные стимулы при помощи нейроинтерфейса была взята в работу как основная, дополнительно было проведено поверхностное исследование темы детекции лжи с использованием нейроинтерфейсов.

В предыдущем семестре команда занималась оценкой эффективности рекламы через изучение эмоциональной реакции зрителей. В рамках этого исследования использовались нейрокомпьютерные интерфейсы для регистрации электроэнцефалограммы (ЭЭГ), что позволяло фиксировать неврологические реакции участников эксперимента в режиме реального времени. Основной задачей являлась идентификация положительных и отрицательных эмоций, а также уровня внимания и вовлечённости аудитории при просмотре рекламных материалов. Подход обеспечил объективное измерение эмоционального восприятия контента и позволил минимизировать влияние субъективных факторов, характерных для традиционных методов маркетингового анализа. Итогом работы с оценкой эффективности рекламы стала соответствующая статья, более подробно описанная в разделе 6.1.

Параллельно с основной темой исследований также рассматривалась возможность использования реакций мозга для определения лжи. В рамках этой темы были рассмотрены методики определения лжи с использованием технологий ЭЭГ. На основе проанализированных материалов получилось разработать методику исследования вопроса использования портативных неинвазивных нейроинтерфейсов, использующих технологию ЭЭГ, для регистрации сигналов, указывающих на ложь, а также на наличие в памяти испытуемого какой-либо информации.

В рамках текущего семестра было принято решение завершить цикл исследования эмоций, начатый в предыдущем семестре, и углубиться в новую для команды область – выявление лжи на основе данных мозговой активности. Такое развитие проекта стало логичным продолжением работы с нейроинтерфейсами и позволило расширить границы применения технологий анализа ЭЭГ за счет исследования не только эмоциональных, но и когнитивных реакций испытуемого.

# ПОЛИГРАФ

Первые попытки выявления лжи с помощью анализа активности мозга появились почти одновременно с развитием методов нейровизуализации. Основная гипотеза заключалась в том, что процесс сознательного обмана требует больших когнитивных усилий, чем честный ответ, что должно отражаться в изменениях показателей работы мозга, регистрируемых с помощью ЭЭГ или фМРТ [4].

Один из подходов к детекции лжи основан на регистрации мозговых реакций на специфические раздражители – слова, изображения или звуки. Эта процедура известна как «нейрофизиологический отпечаток» или «мозговой след». Испытуемому предъявляются различные стимулы, часть из которых связана с событием, подлежащим расследованию, а другие – нет. В случае, если человек узнаёт релевантные стимулы, в его мозге возникает характерный паттерн электрической активности, называемый P300-MERMER (Memory and Encoding Related Multifaceted Electroencephalographic Response) [16]. Таким образом, можно определить, хранится ли в памяти испытуемого конкретная информация.

Ещё один метод выявления лжи использует функциональную магнитно-резонансную томографию (фМРТ), которая регистрирует гемодинамические изменения, связанные с нейронной активностью [17]. ФМРТ позволяет фиксировать увеличение кровотока в тех участках мозга, которые задействованы при выполнении когнитивных задач, таких как принятие решения говорить правду или лгать. Исследование [18] демонстрирует возможность применения фМРТ в качестве детектора лжи: испытуемому задают вопросы с ограниченным выбором ответа, и анализируется уровень активации различных областей мозга. По сравнению с полиграфом, который измеряет такие параметры, как пульс, давление и потоотделение, фМРТ позволяет зафиксировать непосредственно когнитивные процессы, а не эмоциональные реакции. Это делает метод более устойчивым к попыткам манипуляции результатами. Кроме того, благодаря компьютеризации процесса, интерпретация полученных данных становится объективной и не зависит от эксперта.

## Применение неинвазивных нейроинтерфейсов для обнаружения лжи

Нейрокомпьютерный интерфейс (brain-computer interface, BCI) представляет собой устройство, способное считывать электрическую активность мозга и преобразовывать её в сигналы, понятные компьютеру. В контексте детекции лжи такой интерфейс может служить миниатюрной версией ЭЭГ-аппарата, позволяя регистрировать потенциал P300-MERMER. Например, это используется в экспериментах, где участник должен распознавать цвет объекта.

Принцип действия такого детектора лжи аналогичен описанному выше. Перед началом теста испытуемому предлагается провести время в определённом помещении и запомнить как можно больше деталей. Позднее он проходит проверку на устройстве, основанном на нейроинтерфейсе. Анализ записанных данных осуществляется с помощью математических моделей или нейросетей, способных выявлять моменты появления сигнала P300-MERMER. Параллельно также фиксируется пульс испытуемого.

## Практические примеры и перспективы использования нейроинтерфейсов в детекции лжи

На данный момент данные, собранные с помощью полиграфа, ЭЭГ и фМРТ, не признаются в большинстве судебных систем в качестве доказательств. Однако существуют успешные случаи применения этих технологий. Например, в Индии в 2008 году некая женщина по имени А. Шарма была приговорена к пожизненному заключению за отравление бывшего жениха. При расследовании использовалась разработка индийских учёных – метод BEOS (Brain Electrical Oscillations Signature), схожий с ЭЭГ. В Великобритании традиционный полиграф применяется при досрочном освобождении лиц, осуждённых за сексуальные преступления. Также полиграф иногда используется компаниями при найме сотрудников, но только с согласия кандидата.

Создание детектора лжи на основе неинвазивного нейроинтерфейса выглядит перспективным, поскольку такие устройства компактны и не требуют сложной подготовки испытуемого. Однако точность исследования лжи даже на полноразмерном аппарате ЭЭГ не является абсолютной, поэтому о допустимой достоверности портативных решений говорить пока преждевременно.

## Исследуемые метрики

Для исследования активности мозга во время выдачи правды или лжи были рассмотрены различные показатели, снятые с помощью нейроинтерфейса, такие как:

* ритмы волновой активности мозга: альфа-, бета-, тета- и дельта-ритмы. Они позволяют определить конкретные состояния, в которых может находиться испытуемый, такие как усталость, злость, раздражённость, радость и так далее;
* интенсивность электромагнитной активности мозга: данный показатель может охарактеризовать нагрузку, которой подвергается мозг испытуемого во время выдачи той или иной информации, а также от внешних раздражителей;
* показатель P300-MERMER: метрика, показывающая реакцию мозга испытуемого на распознавание какого-либо образа или информации, уже находящейся в его памяти. По данной реакции мозга можно узнать, например, видел ли раньше испытуемый, показанный ему предмет, либо слышал ли он голос с предоставленной ему записи.

Все показатели были записаны с помощью разработанного через Capsule API приложения.

## Проверка релевантности данных

В ходе исследования были собраны три категории данных:

* ложные ответы – участники сознательно давали неверные ответы на заданные вопросы;
* правдивые ответы – участники отвечали честно и точно;
* тестовые данные (произвольные ответы) – участники выбирали способ ответа самостоятельно, без указаний говорить только правду или лгать.

Такая структура позволила сравнить нейронные паттерны, возникающие при различных типах ответов, а также протестировать эффективность алгоритма в условиях неопределённости.

## Проведение экспериментов

Для углубленного исследования детекции лжи с помощью нейроинтерфейсов команда провела анализ медицинских и научных публикаций, описывающих изменения активности мозга при лжи. Изучив существующие работы, было выяснено, что во время лжи у человека наблюдаются характерные изменения в основных биоритмах, таких как бета- и альфа-ритмы, что связано с повышенной когнитивной нагрузкой и эмоциональным напряжением.

Чтобы подтвердить эту теорию на практике, была начата разработка экспериментальной методики. Первым этапом стал подбор теста, который мог бы позволить наиболее эффективно выявлять изменение активности мозга во время лжи. было протестировано несколько типов вопросов:

1. Объективные вопросы с однозначными ответами (например, «Сколько будет два плюс два?»).
2. Персональные вопросы, касающиеся личных данных испытуемого и не нарушающие его личные границы (например, «Как вас зовут?», «Сколько вам лет?»). Такие вопросы позволяли наблюдать реакцию мозга на правдивые и ложные ответы в нейтральном контексте.
3. Эмоционально заряженные вопросы, затрагивающие личные переживания или неудобные темы (например, «Вы когда-нибудь обманывали близких?», «Вы курите?»). Эти вопросы могут вызывать у испытуемых стресс, стеснение или неловкость, что усиливает изменения в биоэлектрической активности.

Кроме того, был рассмотрен вариант тестирования, где испытуемые не отвечали на вопросы, а зачитывали вслух заранее подготовленные верные и ложные утверждения. Это позволило исключить фактор спонтанности, но исключало фактор эмоциональности, влияющий на степень изменения показателей активности мозга.

В результате тестирования различных методик было решено, что вопросы личного характера наиболее эффективны для детекции лжи, так как вызывают выраженные эмоциональные реакции, которые четко фиксируются нейроинтерфейсом. На основе этого вывода был составлен финальный вариант теста, включающий комбинацию персональных и эмоционально напряженных вопросов. Также было принято остановиться на том, чтобы испытуемый отвечал на вопросы письменно, это позволило исключить влияние речевой активности на показатели ЭЭГ и сосредоточиться исключительно на анализе мозговых процессов, связанных с ложью.

Также для дальнейшей проверки работы алгоритма был проведен дополнительный эксперимент. Испытуемый загадывал цифру от 1 до 10, фиксировал на листе бумаги и отвечал отрицательно на 10 типовых вопросов: «Ваша цифра 1?», «Ваша цифра 2?» и т.д. Данный эксперимент позволяет однозначно определить ложь, а также быстро снять данные с большого количества человек. Всего в экспериментах участвовало 24 человека, включая команду проекта.

## Вычисления и анализ

### Статистический анализ с использованием инструментов Excel

Первым шагом для подтверждения научных и медицинских фактов и теорий стало использование однофакторного дисперсионного анализа для выявления зависимостей между показаниями ритмов и правдивости ответов испытуемого. Использовался однофакторный дисперсионный анализ (ANOVA) в Excel для проверки статистической значимости различий между группами. Рассчитывались средние значения, дисперсия, F-критерий и p-value.

Была выдвинута следующая гипотеза: ложь влияет на показатели, фиксируемый нейроинтерфейсом, в частности, на биоритмы организма. Для каждого из ритмов был проведен отдельный анализ, были взяты данные, снятые с одного из испытуемых. Сформулируем гипотезы:

H0: 𝐷𝐹 = 𝐷𝜀 – влияние всех уровней фактора одно и тоже.

H1: 𝐷𝐹 > 𝐷𝜀 – влияние всех уровней фактора статистически значимо.

Примем уровень значимости, равный 0,05. Результаты анализа показателей альфа-ритма (рис. 5.1):

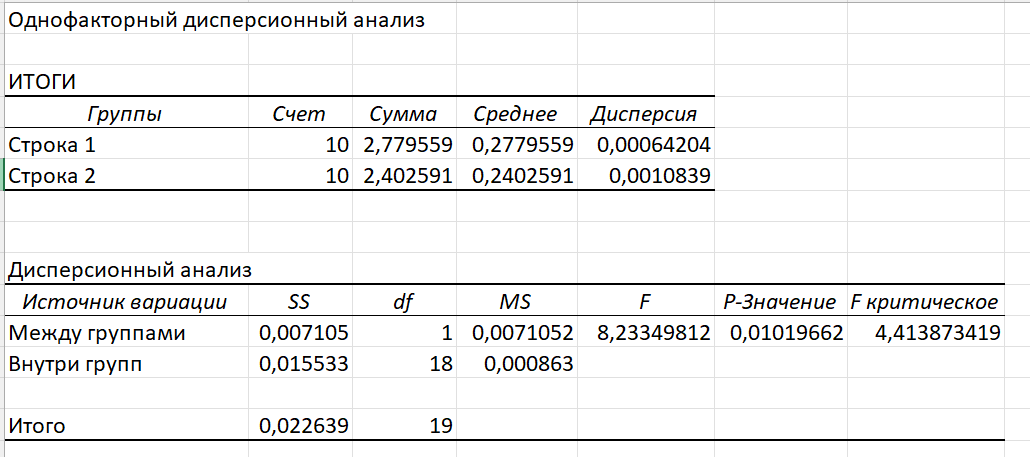


Рисунок 5.1 – Результаты анализа для альфа-ритма

Так как 𝐹набл<𝐹табл и P-Значение меньше принятого уровня значимости, то гипотезу H0 отклоняем в пользу альтернативной гипотезы, то есть различие между правдивостью ответа оказывает существенное влияние на показатели альфа-ритма.

Результаты анализа показателей бета-ритма (рис. 5.2):

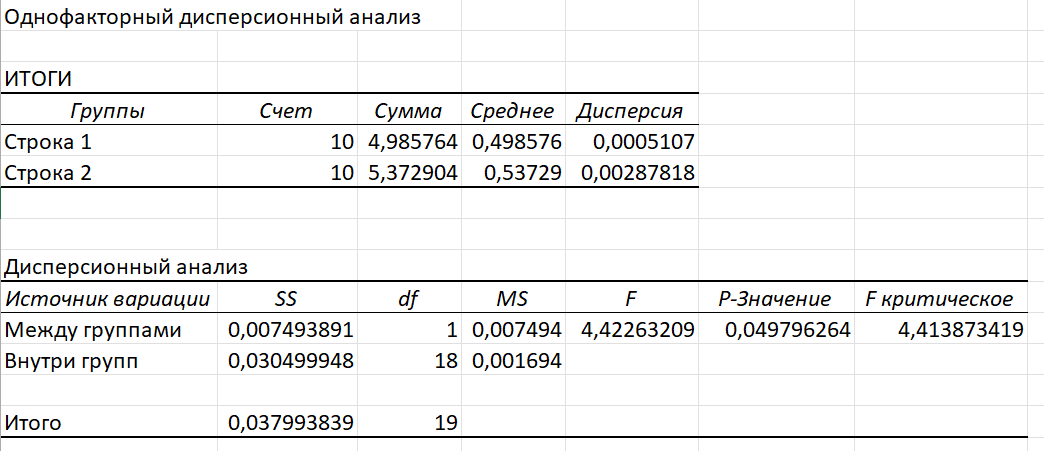


Рисунок 5.2 – Результаты анализа для бета-ритма

Так как 𝐹набл<𝐹табл и P-Значение меньше принятого уровня значимости, то гипотезу H0 отклоняем в пользу альтернативной гипотезы, то есть различие между правдивостью ответа оказывает существенное влияние на показатели бета-ритма.

Результаты анализа показателей тета-ритма (рис. 5.3):

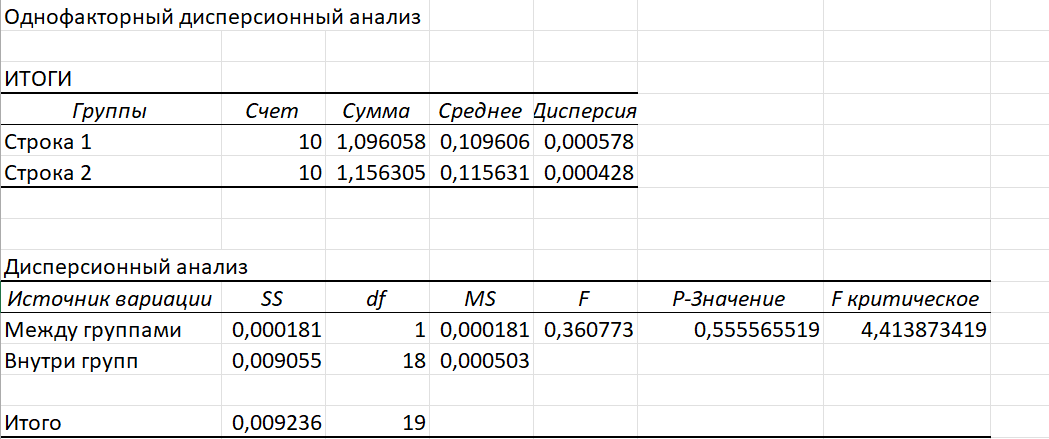


Рисунок 5.3 – Результаты анализа для тета-ритма

Так как 𝐹набл>𝐹табл и P-Значение больше принятого уровня значимости, то гипотезу принимаем, то есть различие между правдивостью ответов не оказывает влияния на показания тета-ритма.

Результаты анализов можно интерпретировать следующим образом:

1. Так как альфа-ритм отражает состояние расслабленного бодрствования, а его низкие показатели связаны с когнитивной нагрузкой, эмоциональным напряжением и концентрацией внимания, результат анализа подтверждает его роль в детекции лжи: чем ниже показатели, тем с большей вероятностью человек говорит ложь.
2. Повышение бета-ритма ассоциировано с активным мышлением и тревожностью. Следовательно, результат анализа подтверждает факт того, что имеется тенденция повышения показателей бета-ритма при лжи.
3. Изменение показателей тета-ритма менее предсказуемо в связи с зависимостью от индивидуальных различий темперамента испытуемых, их объемом памяти. Нельзя наблюдать какую-либо выраженную тенденцию изменения данных.

Проведение данных вычислений подтолкнуло к более глубокому анализу данных: было принято решение использовать технологии машинного обучения и применять такие методы исследования, как корреляционный, кластерный, однофакторный и двухфакторный дисперсионные анализы.

### Анализ с применением машинного обучения

На рисунке 5.1 представлен датасет, полученный с помощью API Neiry. Он содержит информацию о времени записи данных, а также числовые данные по двум каналам channel\_0 и channel\_1.

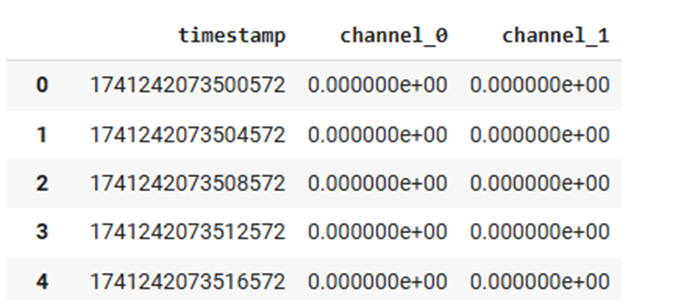
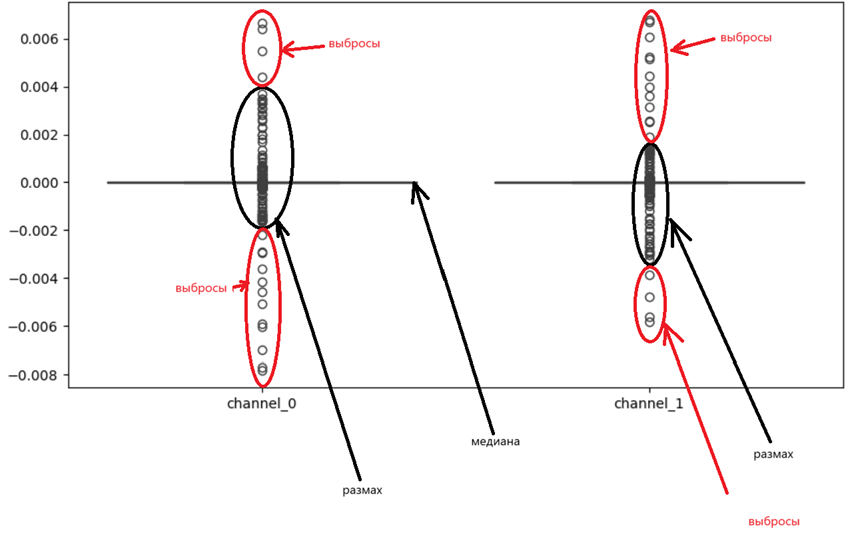
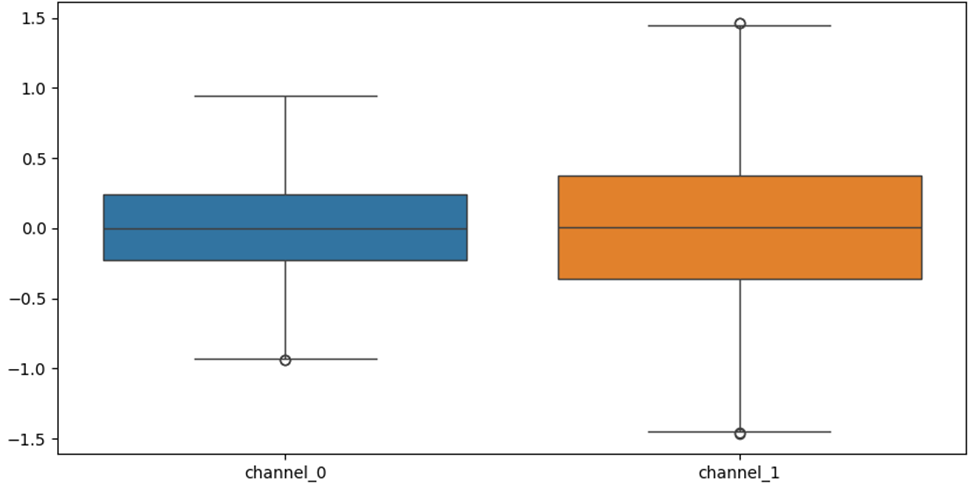


Рисунок 5.1 – Датасет, полученный с помощью API Neiry

Для выявления выбросов было использовано правило межквартильного размаха. Так как сигнал не очищался от шумов при записи, было выявлено 3454 выброса в channel\_0 и 1810 выбросов в channel\_1 (рисунок 5.2).

Рисунок 5.2 – Диаграмма боксплот до очистки от выбросов

В данном исследовании некорректно заменять выбросы на среднее или минимальное значение, так как велика вероятность потери зависимости лжи от ритмов мозга, поэтому значения, лежащие за пределами первого и третьего квартилей были удалены (рисунок 5.3).

Рисунок 5.2 – Диаграмма боксплот после очистки от выбросов

Далее из каналов были выделены alpha, beta и theta волны с помощью полосового фильтра с использованием фильтра Баттерворта. Фильтр Баттерворта – это тип электронного или цифрового фильтра, используемого для обработки сигналов, который характеризуется максимально плоской амплитудно-частотной характеристикой в полосе пропускания [19]. Это означает, что он минимизирует искажения сигнала в заданном частотном диапазоне, обеспечивая плавное усиление или ослабление сигнала без резких колебаний.

На рисунке 5.3 представлена таблица корреляции признаков с целевой переменной, по которой можно сделать вывод, что при снятии данных были допущены ошибки, которые повлекли за собой удаление трети полученных данных.

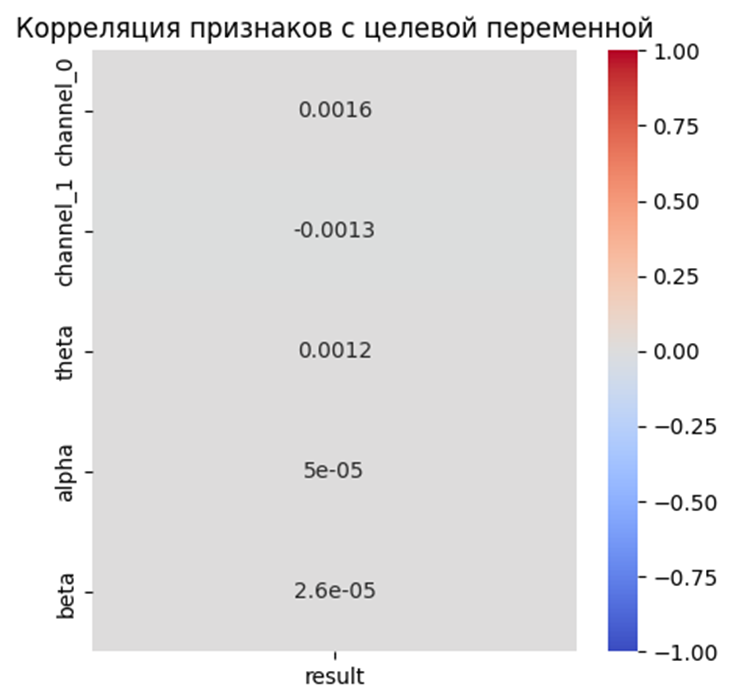


Рисунок 5.3 – Таблица корреляции

На рисунке 5.4 представлен датасет, полученный с помощью разработанного API. Он содержит информацию о ЭЭГ-ритмах.

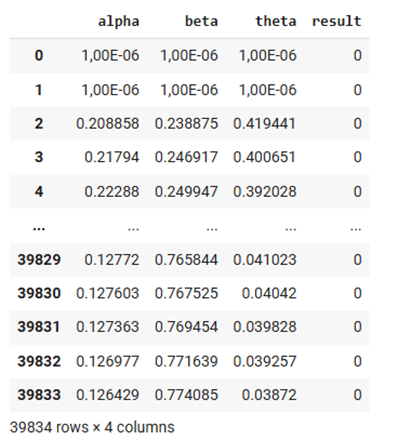


Рисунок 5.4 – Датасет, полученный с помощью разработанного API

Матрица корреляции признаков (рисунок 5.5) показывает отсутствие сильной линейной зависимости с целевой переменной.

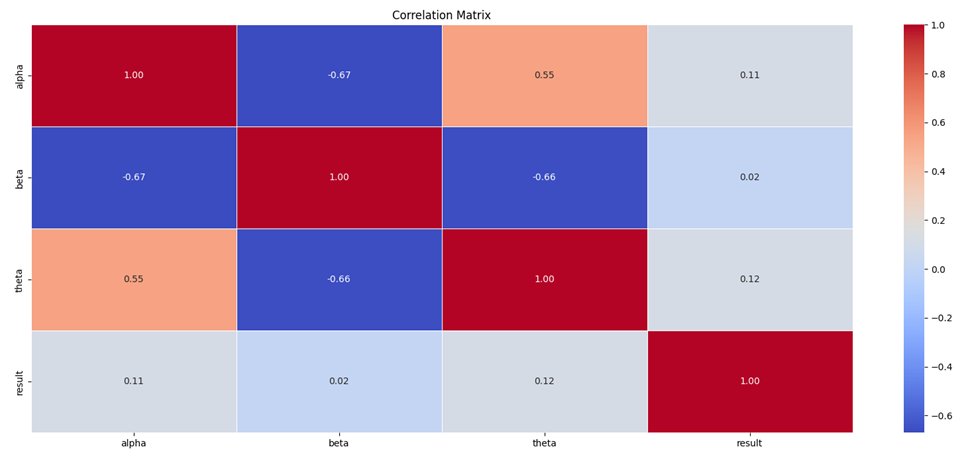


Рисунок 5.5 – Матрица корреляции

На рисунке 5.6 представлена матрица phik, отображающая нелинейные зависимости. По ней можно сделать вывод, что признаки имеют умеренную и слабую зависимость с целевой переменной. Также можно заметить мультиколлинеарность – признаки alpha, beta и theta сильно зависят друг от друга.

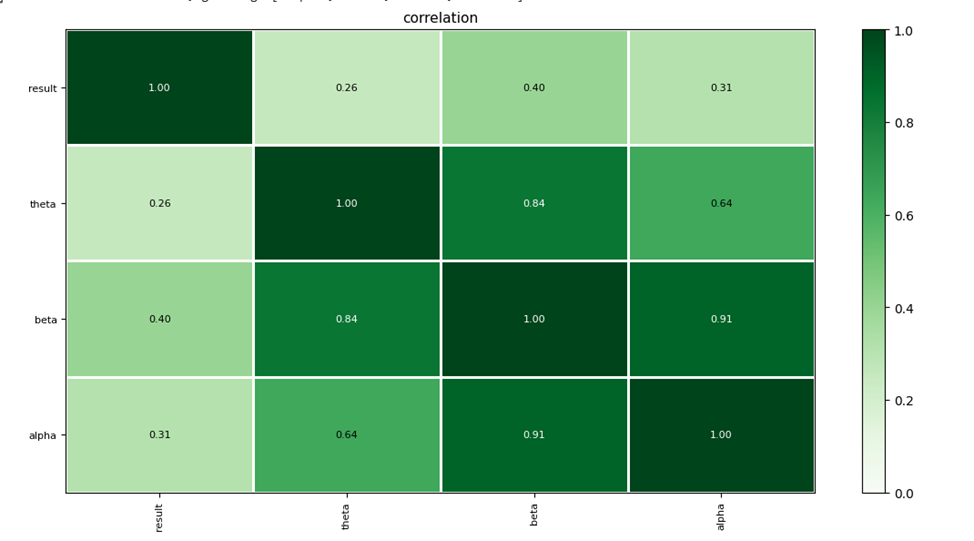


Рисунок 5.6 – Матрица phik

Для решения данной задачи были выбраны: логистическая регрессия, 6 ансамблевых моделей и автоматизированное машинное обучение. Для оценки точности классификационных моделей были выбраны метрики F1-score и ROC\_AUC.

F1-Score – это метрика производительности, используемая в машинном обучении для оценки того, насколько хорошо модель классификации работает на наборе данных. Вычисляется как гармоническое среднее значение точности и полноты, формула (5.1)

|  |  |
| --- | --- |
| , | (5.1) |

где:

* Precision (точность) – доля истинно положительных среди всех объектов, предсказанных как положительные, формула (5.2)

|  |  |
| --- | --- |
|  | (5.2) |
|  |  |

где:

* TP – истинно положительные объекты.
* FP – ложно положительные объекты.
* Recall (полнота) – доля истинно положительных среди всех фактически положительных объектов, формула (5.3).

|  |  |
| --- | --- |
|  | (5.3) |
|  |  |

где:

* TP – истинно положительные объекты.
* FN – ложно отрицательные объекты.

ROC\_AUC – это метрика, оценивающая способность модели различать классы, основанная на построении кривой ошибок (ROC-кривой). Площадь под ROC-кривой (AUC) представляет собой интегральную характеристику качества модели: она показывает вероятность того, что модель присвоит более высокую оценку (вероятность) случайно выбранному положительному примеру, чем случайно выбранному отрицательному [20].

В таблице 1 представлено сравнение плюсов и минусов алгоритмов.

Таблица 1 – Сравнение плюсов и минусов алгоритмов.

|  |  |  |
| --- | --- | --- |
| Алгоритм | Плюсы | Минусы |
| Logistic Regression | Простой алгоритм.  Быстро работает даже на больших данных.  Может использоваться как базовая модель для оценки других алгоритмов. | Плохо работает с нелинейными зависимостями.  Чувствителен к мультиколлинеарности. |
| Random Forest | Устойчива к переобучению.  Хорошо справляется с пропущенными значениями | Может требовать много памяти.  Медленнее логистической регрессии. |
| XGBoost | Один из самых эффективных алгоритмов градиентного бустинга.  Высокая точность и гибкость.  Поддерживает регуляризацию. | Требует тщательной настройки гиперпараметров.  Чувствителен к выбросам и шуму. |
| CatBoost | Эффективно работает с категориальными признаками;  Минимизирует переобучение;  Высокая скорость обучения. | Требует больше времени на обучение, чем линейная регрессия. |
| LightGBM | Очень высокая скорость обучения и предсказания.  Эффективен на больших объемах данных.  Высокая точность. | Может переобучаться при неправильной настройке.  Плохо работает с небольшими данными.  Требует только числовые форматы. |
| BaggingClassifier | Снижает переобучение базовой модели за счет усреднения.  Работает параллельно.  Простая реализация и настройка. | Зависит от качества базового классификатора.  Может проигрывать бустинговым алгоритмам по точности. |
| AdaBoostClassifier | Усиливает слабые модели.  Хорошо работает с шумными данными. | Чувствителен к выбросам.  Может переобучаться.  Меньшая устойчивость к несбалансированным данным по сравнению с градиентным бустингом. |
| Autogluon | Полная автоматизация процесса: подбор модели, обработка данных и гиперпараметров.  Использует стекинг, ансамблирование, обучение по времени. | Требует много ресурсов и времени.  Не всегда лучше ручной настройки. |

Все алгоритмы имеют свои особенности и применимы в зависимости от условий задачи. Базовые модели, такие как логистическая регрессия, хороши для первичного анализа, просты в интерпретации, но часто уступают ансамблевым методам. Модели на основе бэггинга и бустинга обеспечивают более высокую точность за счёт объединения слабых моделей, однако требуют больше вычислительных ресурсов и времени на настройку. AutoGluon предлагает компромисс между качеством и автоматизацией, обеспечивая высокую производительность без глубокого вовлечения пользователя в детали моделирования.

Логистическая регрессия (Logistic Regression) – это базовый алгоритм бинарной классификации, который моделирует вероятность принадлежности наблюдаемого объекта к одному из классов, формула (5.4)

|  |  |
| --- | --- |
|  | (5.4) |
|  |  |

где *βi* – параметры модели, *xi* – признаки.

Ансамблевые модели [21]:

* + - 1. Random Forest.

Случайный лес – это ансамблевый алгоритм, основанный на методе бэггинга и использующий множество независимых деревьев решений для повышения устойчивости и точности модели. Каждое дерево обучается на случайной подвыборке обучающего набора, а также на случайном подмножестве признаков. Итоговое предсказание классификатора принимается по принципу большинства голосов формула (5.5)

|  |  |
| --- | --- |
|  | (5.5) |
|  |  |

где *yk* – предсказание k-го дерева.

* + - 1. XGBoost

XGBoost – это эффективная реализация градиентного бустинга над деревьями решений, разработанная с целью обеспечения высокой производительности и масштабируемости. Алгоритм обучает модели поэтапно, где каждая новая модель минимизирует остаточную ошибку предыдущих с использованием градиентного спуска формула (5.6)

|  |  |
| --- | --- |
|  | (5.6) |
|  |  |

где:

* Ft-1(x) – предыдущая модель;
* ht(x) – новое дерево, уменьшающее ошибку;
* η – коэффициент обучения
  + - 1. CatBoost

CatBoost – это бустинговый алгоритм, разработанный для работы с категориальными данными без необходимости их явного кодирования. CatBoost использует специальный метод кодирования категориальных признаков (Ordered Boosting) и строит последовательность деревьев, минимизируя ошибку, аналогично XGBoost.

1. LightGBM

LightGBM – это высокоэффективный алгоритм градиентного бустинга. Он предназначен для решения задач классификации, регрессии и ранжирования.

В отличие от классического бустинга, LightGBM использует две ключевые техники оптимизации:

* Gradient-based One-Side Sampling (GOSS) — отбор только наиболее важных объектов для обновления градиентов, что снижает вычислительные затраты.
* Exclusive Feature Bundling (EFB) — объединение слабо коррелированных признаков в один, что уменьшает размерность признакового пространства без потери информации.

Результаты обучения моделей без гиперпараметров представлены в таблице 2.

Таблица 2 – Результаты обучения моделей без гиперпараметров

|  |  |  |
| --- | --- | --- |
| Модель | F1-Score | ROC-AUC |
| Logistic Regression | 0.409110 | 0.579392 |
| Random Forest | 0.813414 | 0.942297 |
| XGBoost | 0.7742457 | 0.904163 |
| CatBoost | 0.724808 | 0.893604 |
| LightGBM | 0.696315 | 0.876578 |

Из данной таблицы можно сделать вывод, что логистическая регрессия не подходит для решения данной задачи. Для остальных моделей были подобраны гиперпараметры с помощью Grid Search. Gread Search – это метод подбора гиперпараметров модели машинного обучения, при котором перебираются все возможные комбинации заданных параметров для выбора наилучшего варианта [22].

Алгоритм работы:

1. Определение гиперпараметров – задается сетка параметров (например, количество деревьев в случайном лесе, глубина деревьев и т. д.).
2. Перебор всех комбинаций – метод тестирует каждую комбинацию гиперпараметров.
3. Оценка модели – для каждой комбинации вычисляется метрика качества.
4. Выбор наилучших параметров – выбирается комбинация, при которой достигается наилучший результат.

Результаты представлены в таблице 3.

Таблица 3 – результаты обучения моделей с гиперпараметрами

|  |  |  |
| --- | --- | --- |
| Модель | F1-Score | ROC-AUC |
| Random Forest | 0.8193 | 0.9445 |
| XGBoost | 0.7657 | 0.9112 |
| CatBoost | 0.7984 | 0.9374 |
| LightGBM | 0.7088 | 0.8825 |

Исходя из таблицы можно сделать вывод, что XGBoost и CatBoost улучшили свои метрики, а Random Forest ухудшил. Ухудшение модели после подбора гиперпараметров может быть связана с переобучением, так как GridSearch может найти гиперпараметры, которые слишком хорошо подгоняют модель под обучающие данные, но ухудшают обобщающую способность на тестовых данных. Лучшая модель для данной задачи – Random Forest без гиперпараметров.

1. BaggingClassifier

BaggingClassifier – это ансамблевый метод, реализующий технику бэггинга. Он объединяет несколько моделей и обучает их на различных подвыборках исходных данных с возвращением. Полученные предсказания агрегируются, что позволяет снизить дисперсию модели и повысить устойчивость к переобучению.

Результаты обучения разных сочетаний этой модели представлены в таблице 4.

Таблица 4 – Результаты бэггинга

|  |  |  |
| --- | --- | --- |
| Сочетание | F1-Score | ROC-AUC |
| Random Forest и Random Forest | 0.800069 | 0.936945 |
| CatBoost и CatBoost | 0.699878 | 0.885191 |
| XGBoost и XGBoost | 0.766578 | 0.914348 |
| LightGBM и LightGBM | 0.712533 | 0.886171 |

1. AdaBoostClassifier

AdaBoostClassifier — это классический алгоритм бустинга (адаптивного усиления), который обучает слабые модели последовательно, уделяя больше внимания ошибкам предыдущих.

Результаты обучения разных сочетаний этой модели представлены в таблице 5.

Таблица 5 – Результаты бустинга

|  |  |  |
| --- | --- | --- |
| Сочетание | F1-Score | ROC-AUC |
| Random Forest и Random Forest | 0.873055 | 0.817622 |
| CatBoost и CatBoost | 0.726837 | 0.880846 |
| LightGBM и LightGBM | 0.721823 | 0.867973 |

AutoGluon — это открытая библиотека автоматизированного машинного обучения (AutoML), разработанная Amazon Web Services (AWS). Её основная цель — предоставить удобный и производительный инструмент, который позволяет пользователям без глубоких знаний в области машинного обучения быстро обучать высококачественные модели [23].

AutoGluon автоматизирует весь жизненный цикл машинного обучения, включая:

* предварительную обработку данных;
* подбор моделей;
* оптимизацию гиперпараметров;
* ансамблирование (stacking);
* оценку качества моделей.

Одним из ключевых компонентов AutoGluon является модуль TabularPredictor, предназначенный для анализа табличных данных. Он автоматически выбирает наиболее подходящие модели и объединяет их в ансамбли, что повышает точность предсказаний.

Лучшую производительность по метрикам F1-Score (0.485126) и ROC-AUC (0.525844) показала модель WeightedEnsemble\_L2, сформированная AutoGluon в процессе обучения. Данный алгоритм представляет собой второй уровень ансамблирования (level 2 stacking), в котором объединяются предсказания нескольких базовых моделей, обученных на первом уровне (L1).

На рисунке 5.7 представлен датасет, полученный с помощью усовершенствованного API. Он содержит информацию о времени записи данных, показатели усталости, концентрации, расслабления, а также накопленной усталости.

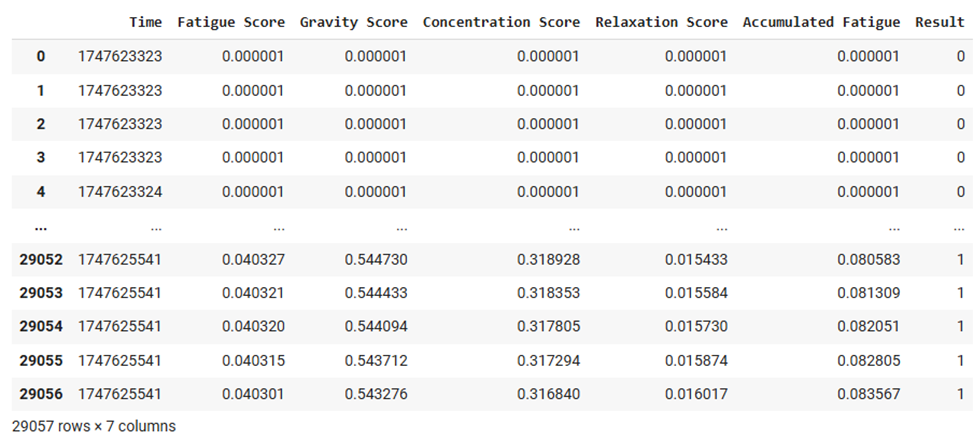


Рисунок 5.7 – Датасет, полученный с помощью усовершенствованного API

По матрице корреляции (рисунок 5.8) видна небольшая зависимость целевой переменной от всех признаков, также прослеживается сильная корреляция между всеми признаками. Можно сделать вывод, что между признаками и целевой переменной присутствует нелинейная зависимость.

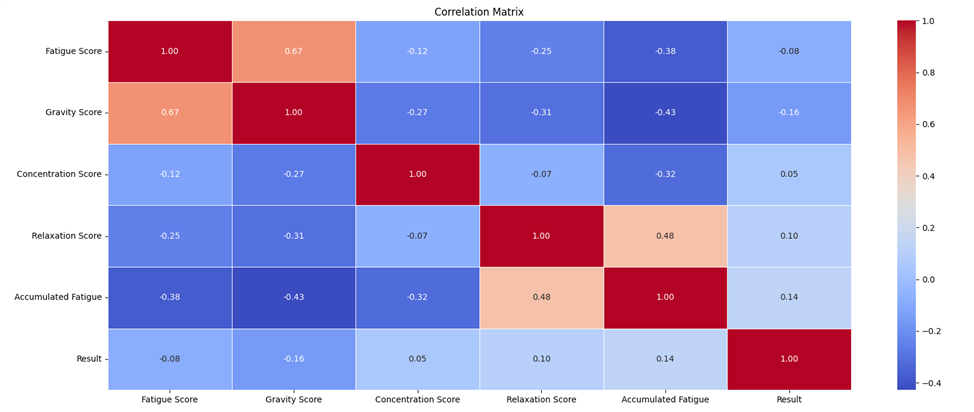


Рисунок 5.8 – Матрица корреляции

Матрица phik (рисунок 5.9) показывает умеренную и слабую зависимость целевой переменной и признаков.

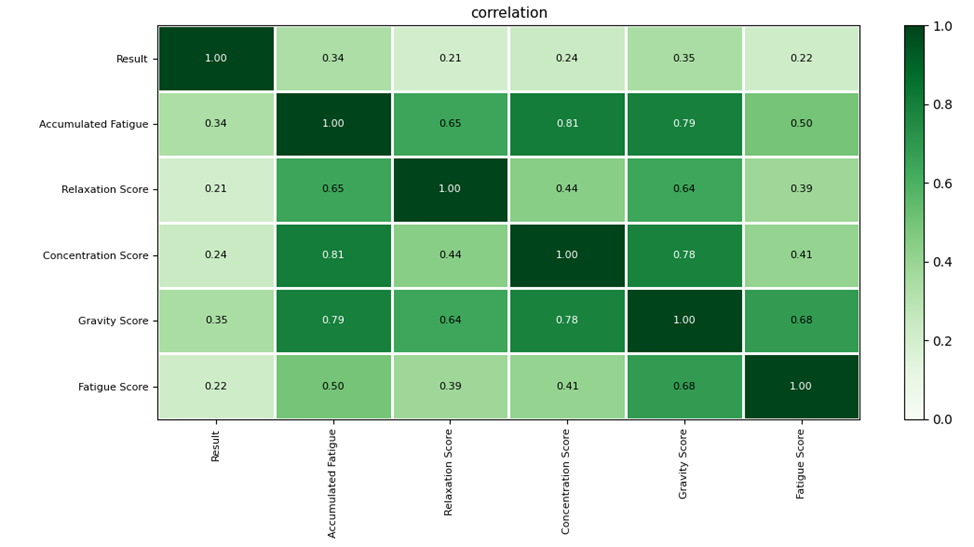


Рисунок 5.9 – Матрица phik

На этих данных были обучены: логистическая регрессия, 6 ансамблевых моделей, а также автоматизированное машинное обучение. Результаты обучения по метрикам представлены в таблице 6.

Таблица 6 – Результаты обучения моделей.

|  |  |  |
| --- | --- | --- |
|  | F1-Score | ROC-AUC |
| Модели без гиперпараметров | | |
| Logistic Regression | 0.311715 | 0.619404 |
| Random Forest | 0.990281 | 0.998892 |
| XGBoost | 0.990313 | 0.997748 |
| CatBoost | 0.984610 | 0.996675 |
| LightGBM | 1.000000 | 1.000000 |
| Модели с гиперпараметрами | | |
| Random Forest | 0.9107 | 0.9975 |
| XGBoost | 0.9681 | 0.9981 |
| CatBoost | 0.9724 | 0.9969 |
| LightGBM | 0.9953 | 1.0000 |
| Бэггинг | | |
| Random Forest и Random Forest | 0.941692 | 0.997252 |
| XGBoost и XGBoost | 0.964349 | 0.998056 |
| LightGBM и LightGBM | 0.995349 | 0.999987 |
| Бустинг | | |
| Random Forest и Random Forest | 0.956626 | 0.991726 |
| LightGBM и LightGBM | 0.992110 | 0.999014 |
| Автоматизированное машинное обучение | | |
| LightGBM | 1.00000 | 1.00000 |

## Визуализация работы полученных моделей

Дальнейшей целью после успешно обученных машинных моделей была визуализация их работы. Основной целью разработки стало создание интерактивного аналитического инструмента, позволяющего исследователям:

1. Наглядно интерпретировать результаты работы модели
2. Проводить верификацию классификатора на новых данных
3. Анализировать временную динамику маркеров лжи

Был выбран следующий стек технологий для реализации данной задумки:

1. HTML5/CSS3 для структуры и стилизации интерфейса, JavaScript с использованием сборщика Gulp для фронтенда: с помощью данных инструментов был разработан интуитивно понятный интерфейс с кнопкой загрузки CSV-файла с данными. Была добавлена визуализация результатов в виде таблицы с маркировкой: выводится вероятность, с которой была сказана правда/ложь в определенный момент времени снятия данных. Выводятся значения биоритмов.
2. Python, FastApi для бэкенда: на сервере реализован скрипт для обработки загруженных данных, в котором использована предобученная модель машинного обучения, классифицирующая данные.

Сверстанный веб-сайт (рис. 5.10) подразумевает следующий рабочий процесс:

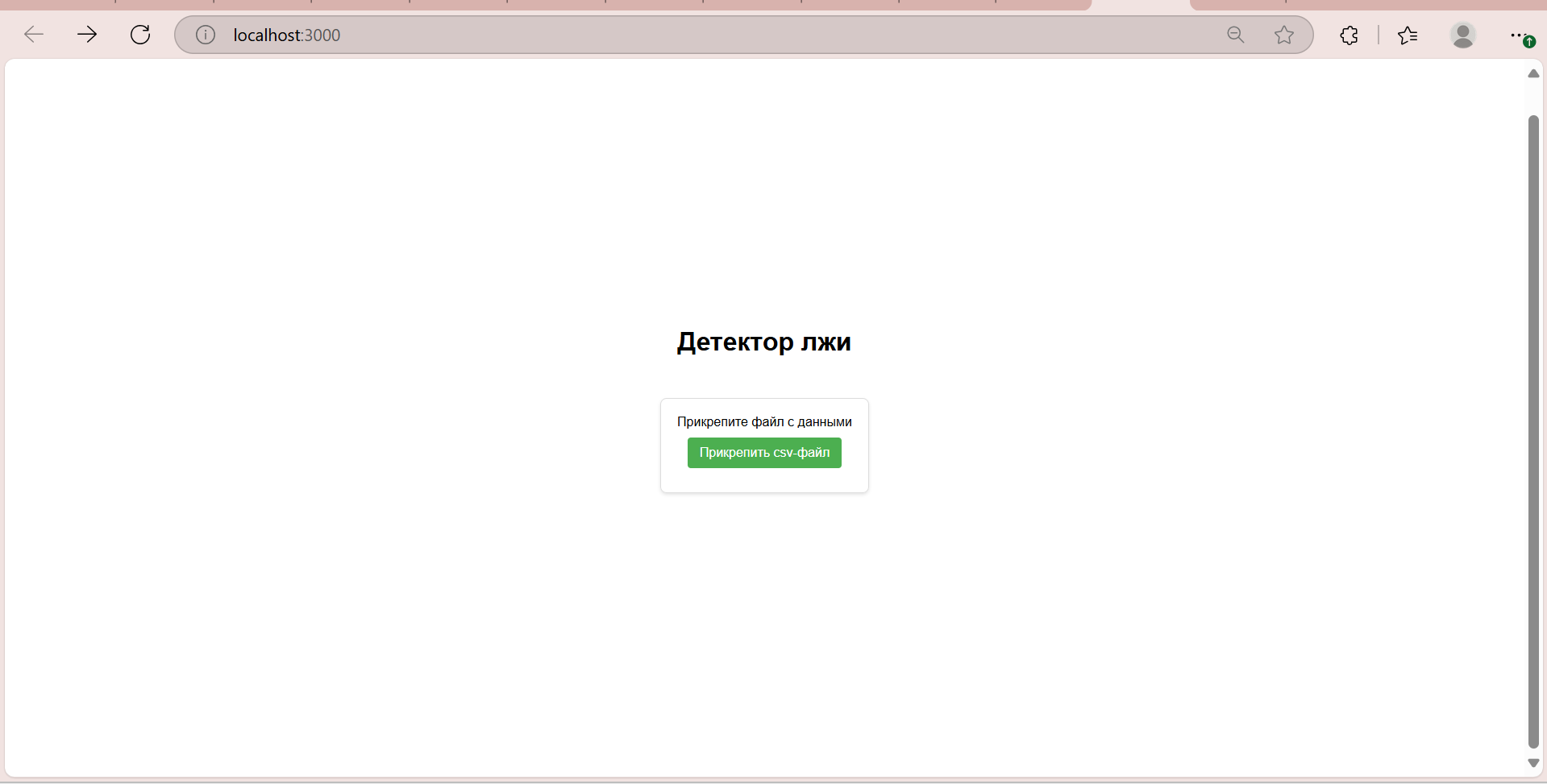


Рисунок 5.10 – Главная страница сайта

1. Пользователь загружает CSV-файл через веб-интерфейс (рис. 5.11 - 5.12).

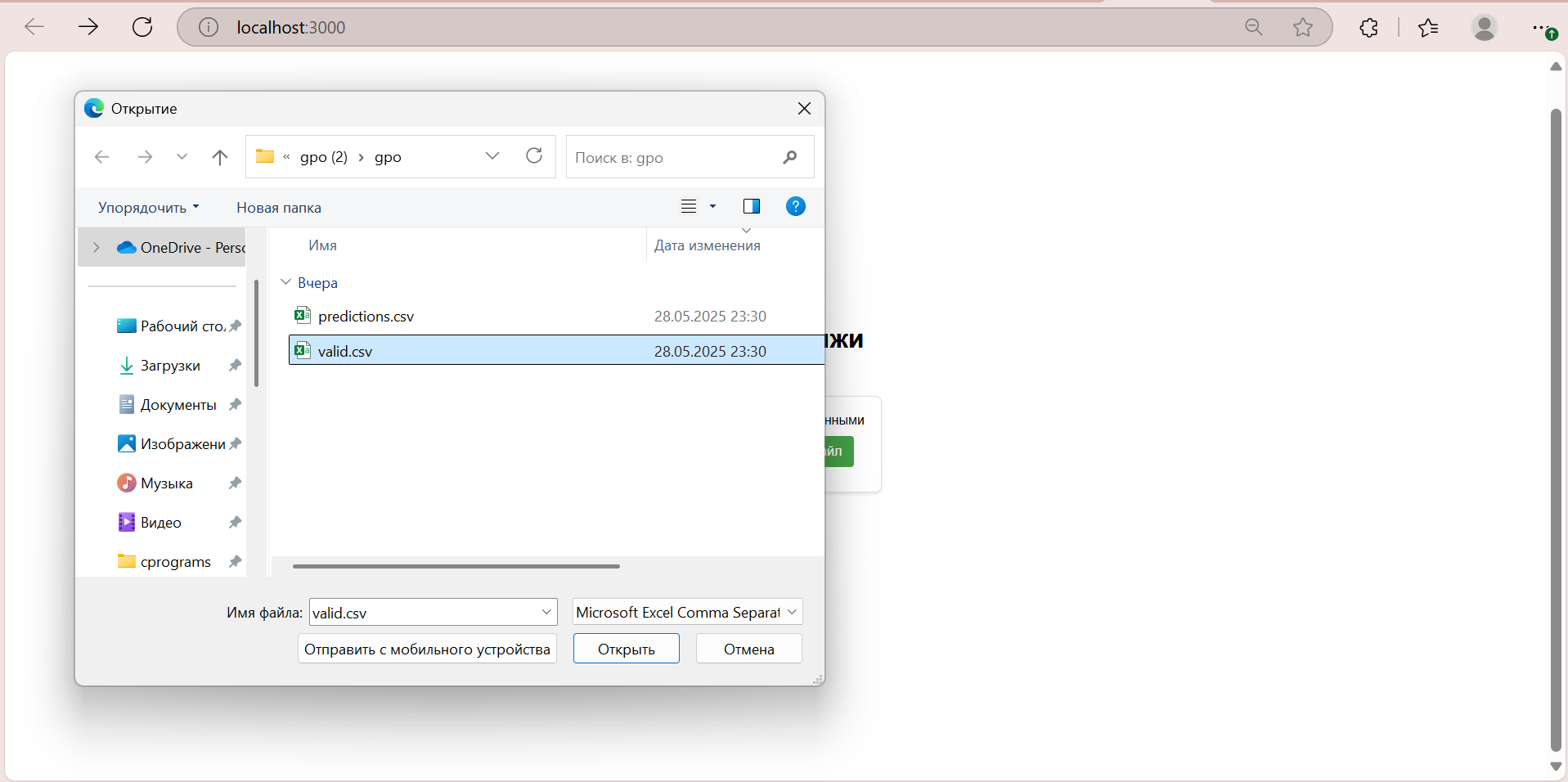


Рисунок 5.11 – Прикрепление csv-файла

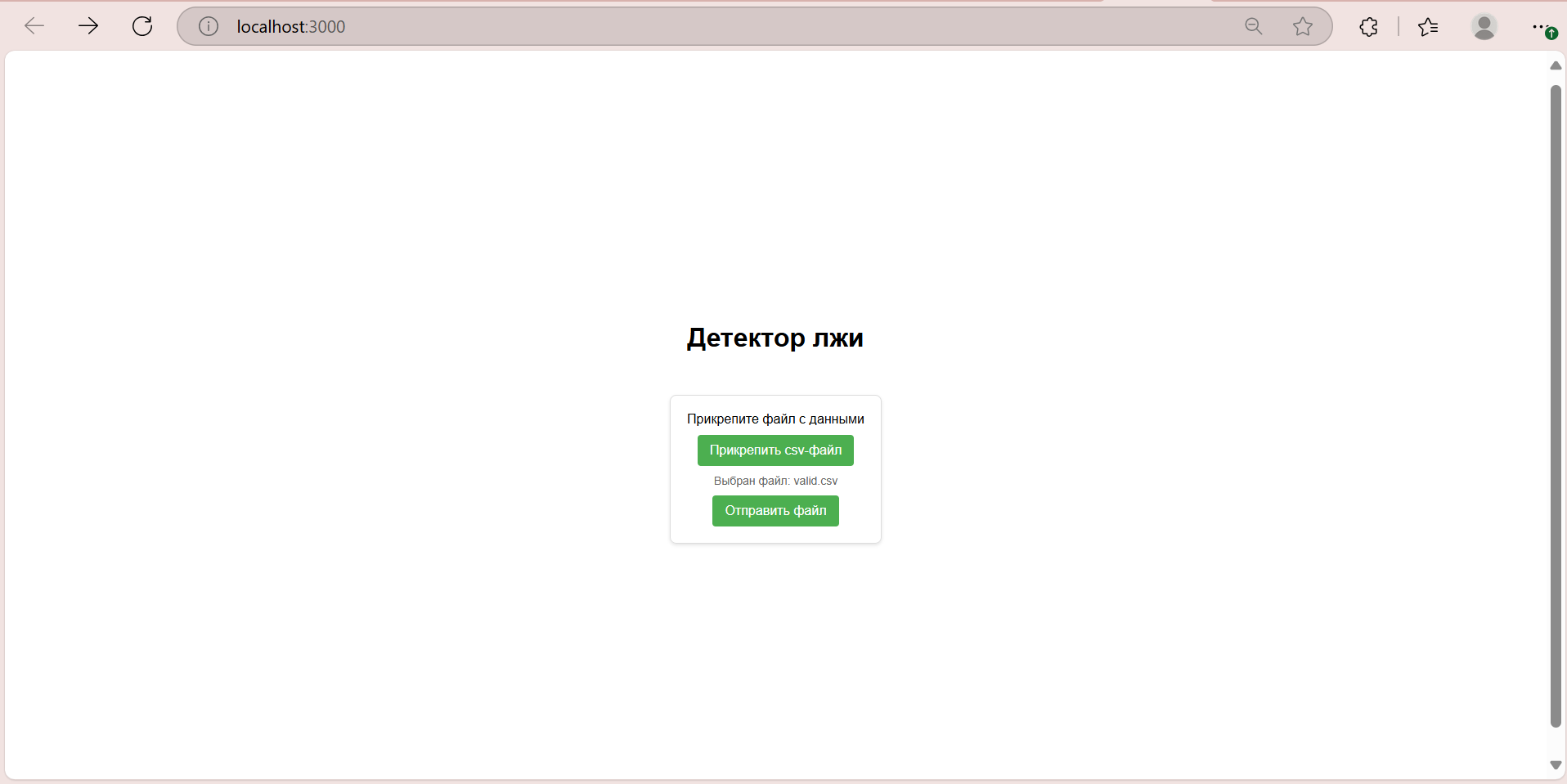


Рисунок 5.12 – Отображение выбранного файла

1. После отправки пользователем файла (рис. 5.13) сервер обрабатывает данные, применяет модель и возвращает таблицу с результатами (рис. 5.14). Пользователь имеет возможность ознакомиться с ней, нажав кнопку “Посмотреть результаты”.

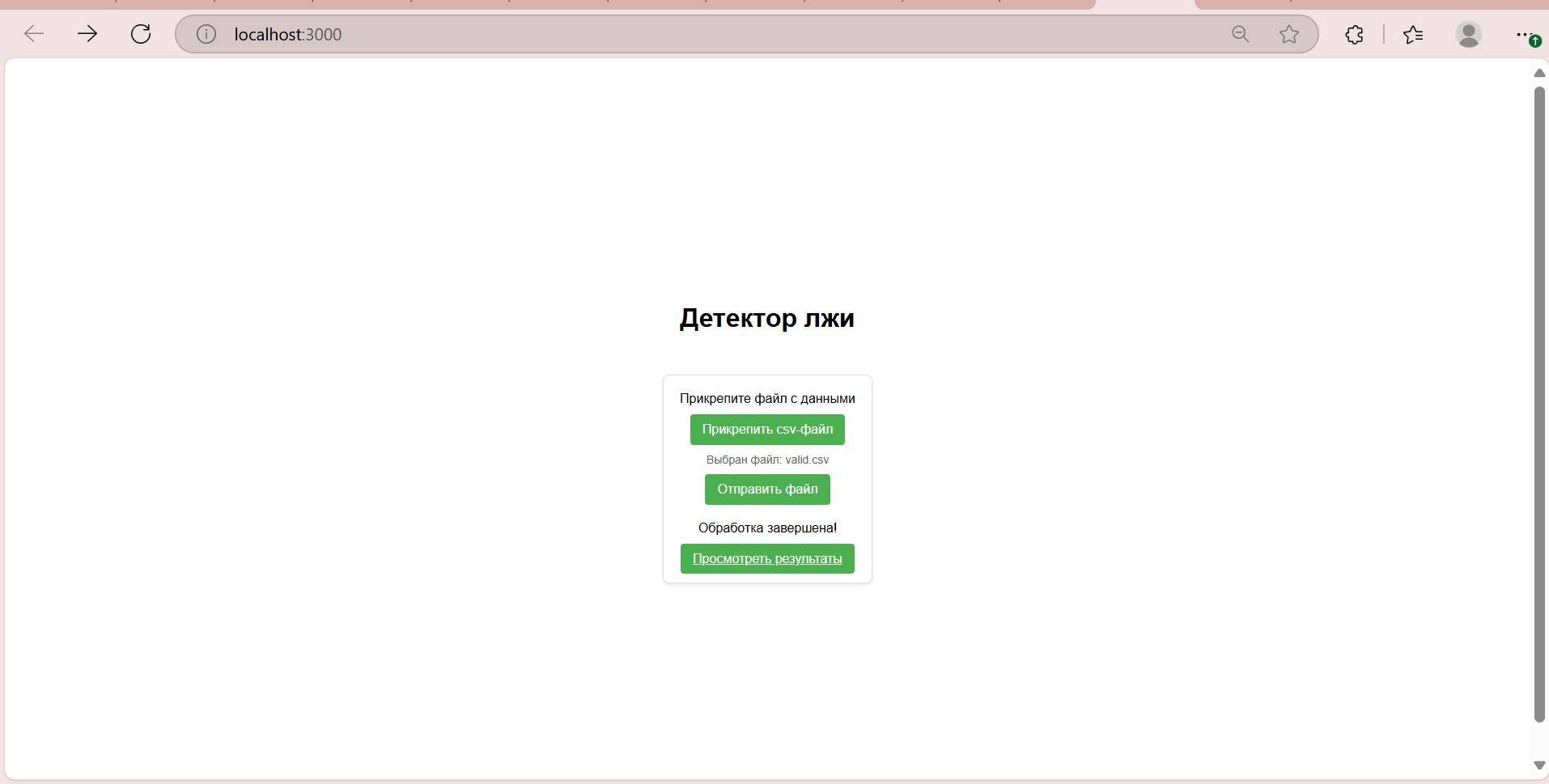


Рисунок 5.13 – Вид сайта после отправления пользователем файла

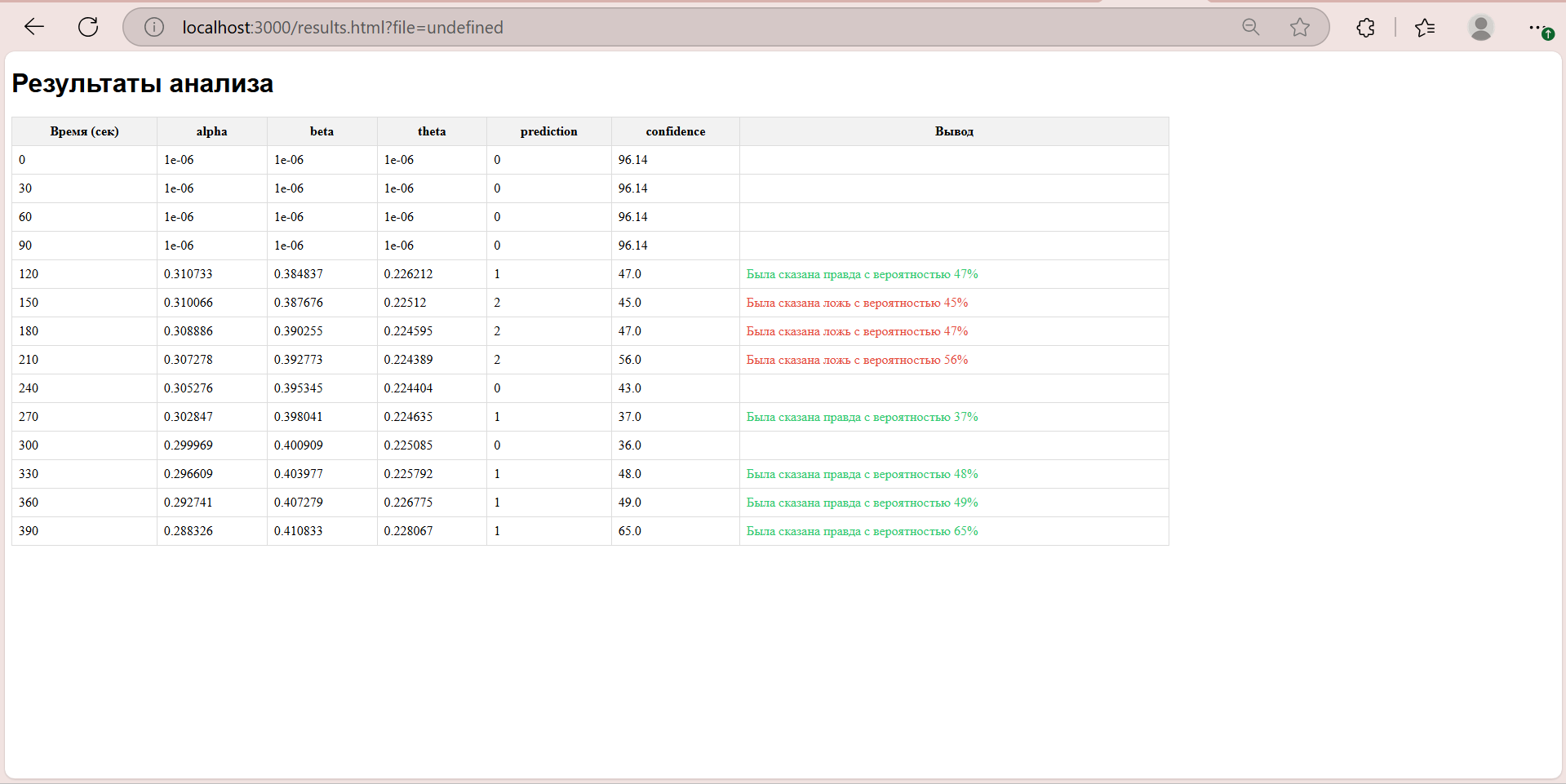


Рисунок 5.14 – Страница сайта, отображающая результаты анализа

Разработанное решение существенно упрощает процесс анализа экспериментальных данных и служит основой для создания комплексных систем тестирования с применением нейроинтерфейсов.

## Выводы

В ходе работы было установлено, что использование нейроинтерфейсов действительно имеет потенциал для развития в области детекции лжи через анализ мозговой активности. Наиболее перспективными метриками для выявления лжи оказались альфа-ритм и бета-ритм мозговой активности. Альфа-ритм демонстрирует тенденцию к снижению при лжи, что связано с увеличением когнитивной нагрузки и эмоционального напряжения. Бета-ритм, напротив, показывает повышение при лжи, что связано с активным мышлением и тревожностью.

Особое внимание уделялось изучению P300-MERMER – комплексного электромагнитного ответа мозга на знакомую информацию. Этот показатель должен был быть информативным при определении наличия конкретной информации в памяти испытуемого, однако с его выявлением возникли трудности и дальнейшую работу с данным параметром пришлось отложить.

Среди протестированных моделей машинного обучения наиболее высокие результаты показали Random Forest (F1-Score: 0.8193, ROC-AUC: 0.9445) и CatBoost (F1-Score: 0.7984, ROC-AUC: 0.9374). Эти модели продемонстрировали высокую точность в классификации правдивых и ложных ответов на основе данных мозговой активности.

Разработанное решение имеет большой потенциал для дальнейшего развития и может служить основой для создания более совершенных систем тестирования на полиграфе. Однако стоит отметить, что для практического применения необходимы дополнительные исследования и улучшение точности распознавания. Тем не менее, полученные результаты демонстрируют высокий потенциал и открывают новые перспективы в этой области исследований.

# 

# НАПИСАННЫЕ СТАТЬИ

## Использование эмоционального отклика для оценки эффективности рекламы

В этой статье рассматривается инновационный подход к оценке эффективности рекламных материалов на основе анализа эмоционального отклика потребителей. В качестве ключевого инструмента измерения используется технология снятия энцефалограммы (ЭЭГ) с помощью нейроинтерфейсов, что позволяет получать объективные данные о неврологических реакциях испытуемых в режиме реального времени. Исследование направлено на выявление корреляции между эмоциональными реакциями, регистрируемыми на уровне активности головного мозга, и восприятием рекламных сообщений. Особое внимание уделяется дифференциации положительных и отрицательных эмоций, а также уровня вовлеченности аудитории. Предложенный метод оценки предоставляет возможность минимизировать влияние субъективных факторов, присущих традиционным маркетинговым исследованиям, и формирует более точное представление об эффективности рекламных кампаний. Результаты работы могут быть использованы для оптимизации рекламных стратегий, повышения их эмоциональной резонансности и, как следствие, увеличения конверсии.

Результаты исследования показали, что использование мультимедийных форматов рекламы приводит к повышению уровня вовлеченности аудитории по сравнению с традиционными форматами. Однако важно отметить, что некоторые элементы рекламы вызывали негативные реакции, что может быть связано с избыточной информационной нагрузкой или несоответствием ожиданиям потребителей. Эти данные подтверждают необходимость тщательного тестирования рекламных материалов перед запуском кампании.

На основе полученных данных можно сделать следующие рекомендации для создания эффективной рекламы. Использование элементов, вызывающих положительные эмоции (например, юмор или вдохновляющие истории), значительно повышает уровень вовлеченности аудитории, а рекламные ролики продолжительностью менее 30 секунд часто вызывают снижение интереса, что подтверждается данными ЭЭГ. Также не стоит забывать о том, что учет особенностей целевой аудитории позволяет создавать рекламу, которая вызывает максимальный резонанс.

Таким образом, современная реклама требует новых подходов к оценке своей эффективности, учитывающих как сложность человеческой психики, так и динамичность цифровой среды. Применение нейротехнологий, таких как ЭЭГ, открывает новые горизонты для понимания того, как реклама воздействует на эмоциональное состояние потребителей, что, в свою очередь, может стать основой для создания более релевантных и убедительных рекламных сообщений.

Данная статья стала заключительным этапом работы с исследованием эмоций и является итогом работы команды в прошлом семестре.

# ЗАКЛЮЧЕНИЕ

При выполнения семестровой работы были успешно достигнуты все поставленные задачи, что позволило не только углубить понимание темы нейродетектора лжи, но и разработать практическое решение для анализа данных мозговой активности. Работа охватила полный цикл исследовательского процесса: от выбора темы и изучения теоретической базы до проведения экспериментов, анализа результатов и их визуализации.

Одним из ключевых этапов стал выбор направления исследования – использование неинвазивных нейроинтерфейсов для обнаружения лжи. Этот выбор был продиктован актуальностью темы, поскольку современные технологии позволяют создавать компактные и точные системы для выявления правдивых и ложных высказываний. В рамках работы были рассмотрены различные методики детекции лжи, такие как анализ P300-MERMER, использование биоритмов мозга (альфа-, бета- и тета-ритмы) и другие метрики электроэнцефалограммы. Особое внимание уделялось интерпретации данных, полученных с помощью нейроинтерфейса Neiry Headband, а также их предварительной обработке и очистке от артефактов.

Для анализа данных были применены современные методы машинного обучения, включая логистическую регрессию, ансамблевые модели (Random Forest, XGBoost, CatBoost, LightGBM) и автоматизированное машинное обучение (AutoGluon). Результаты показали, что наиболее эффективными моделями оказались Random Forest и CatBoost, которые продемонстрировали высокие значения метрик F1-Score и ROC-AUC. Это подтвердило гипотезу о возможности использования нейроинтерфейсов для детекции лжи на основе анализа мозговой активности.

Особое значение имеет разработка удобного веб-интерфейса для загрузки и анализа данных. Созданный инструмент позволяет пользователям загружать CSV-файлы с данными ЭЭГ и получать детальный анализ с указанием вероятности правдивости или ложности каждого ответа, а также соответствующих значений биоритмов. Интерфейс предоставляет наглядную визуализацию результатов, что значительно упрощает процесс интерпретации данных и верификации классификатора на новых данных. Это решение может стать основой для создания комплексных систем тестирования, применимых в судебной практике, безопасности и других сферах.

Помимо технической реализации, работа включала глубокий анализ научной литературы и существующих исследований в области нейродетекторов лжи. Были изучены как традиционные методы, такие как полиграф, так и современные подходы, основанные на использовании ЭЭГ. Это позволило лучше понять преимущества и ограничение используемой технологии неинвазивного нейроинтерфейса, а также определить перспективы развития данного направления.

Важным результатом работы стало также создание научной статьи, посвященной использованию эмоционального отклика для оценки эффективности рекламы. Этот материал стал итогом исследований, проведенных командой в предыдущем семестре, и подчеркнул универсальность применения нейроинтерфейсов в различных областях.

Полученные результаты работы данного семестра открывают широкие перспективы для дальнейшего развития проекта. В планах – оформление результатов в виде научной статьи, публикация её в рецензируемом журнале и представление на конференциях. Также планируется продолжить работу над улучшением точности модели, расширением набора исследуемых метрик и проведением дополнительных экспериментов для подтверждения релевантности данных.

Таким образом, выполненная работа демонстрирует использование нейроинтерфейсов для детекции лжи и анализа когнитивных процессов. Разработанные решения могут быть адаптированы для применения в различных областях. Проект подтверждает, что современные технологии способны предоставлять объективные данные о работе мозга, что делает их незаменимыми инструментами для исследований и практических задач.

Несмотря на результаты исследований и прогресс в разработке технологий детекции лжи, данное направление остается слабо изученным и требует дальнейших исследований для подтверждения надежности. Данные методы пока не могут служить абсолютным доказательством достоверности или ложности высказывания. Это связано с тем, что интерпретация данных ЭЭГ зависит от множества факторов, таких как индивидуальные особенности испытуемого, уровень стресса, внешние условия эксперимента и возможные артефакты записи. Кроме того, механизмы когнитивной обработки информации при правдивых и ложных ответах все еще недостаточно изучены, что затрудняет создание универсальных алгоритмов, применимых в любой ситуации. Таким образом, технологии нейродетекции лжи пока остаются лишь дополнительным инструментом, который может указывать на вероятность обмана, но не является вещественным доказательством, приемлемым в суде или других формальных процедурах.

# СПИСОК ЛИТЕРАТУРЫ

1. Нейроинтерфейс: управлять силой мысли [Электронный ресурс] // URL: https://rostec.ru/media/news/neyrointerfeys-upravlyat-siloy-mysli/ (дата обращения 01.12.2024).

2. Как используют нейроинтерфейсы в образовании [Электронный ресурс] // URL: https://skillbox.ru/media/education/kak-ispolzuyut-neyrointerfeysy-v-obrazovanii-mozhno-opredelit-chto-pokazalos-skuchnym/?clckid=0ab2c07d (дата обращения 01.12.2024).

3. Напрямую в мозг: что такое нейроинтерфейсы и как они могут изменить человека [Электронный ресурс] // URL:https://habr.com/ru/companies/gazprombank/articles/773846/ (дата обращения 01.12.2024).

4. Rosenfeld J.P., Hu X., Labkovsky E. et al. Review of Recent Studies and Issues Regarding the P300-Based Complex Trial Protocol for Detection of Concealed Information // International Journal of Psychophysiology. 2013. Vol. 90. Iss. 2. P. 118–134.

5. Документация по Microsoft C/C++. [Электронный ресурс] // URL: https://learn.microsoft.com/ru-ru/cpp/?view=msvc-170 (дата обращения 1.06.2025)

6. 3.13.3 Documentation - Python. [Электронный ресурс] // URL: https://docs.python.org/3/ (дата обращения 1.06.2025)

7. Что такое JavaScript и зачем он нужен. [Электронный ресурс] // URL: https://skillbox.ru/media/code/chto-takoe-javascript-i-zachem-on-nuzhen/ (дата обращения 1.06.2025)

8. LogisticRegression — scikit-learn 1.6.1 documentation. [Электронный ресурс] // URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html (дата обращения 1.06.2025)

9. RandomForestClassifier — scikit-learn 1.6.1 documentation. [Электронный ресурс] // URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата обращения 1.06.2025)

10. XGBoost Documentation — xgboost 3.0.2 documentation. [Электронный ресурс] // URL: https://xgboost.readthedocs.io/en/stable/ (дата обращения 1.06.2025)

11. CatBoost - open-source gradient boosting library. [Электронный ресурс] // URL: https://catboost.ai (дата обращения 1.06.2025)

12. Welcome to LightGBM’s documentation! [Электронный ресурс] // URL: https://lightgbm.readthedocs.io/en/stable/ (дата обращения 1.06.2025)

13. Алгоритм AdaBoost. [Электронный ресурс] // URL: https://habr.com/ru/companies/otus/articles/503888/ (дата обращения 1.06.2025)

14. BaggingClassifier. [Электронный ресурс] // URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html (дата обращения 1.06.2025)

15. Amazon представила AutoGluon для создания ИИ-моделей из трех строчек кода. [Электронный ресурс] // URL: https://habr.com/ru/news/483432/ (дата обращения 1.06.2025)

16. Farwell L.A. Brain ﬁngerprinting: a comprehensive tutorial review of detection of concealed information with event-related brain potentials // Cognitive Neurodynamics. 2012. Vol. 6. Iss. 2. P. 115–154.

17. Kozel F.A., Johnson K.A., Mu Q. et al. Detecting Deception Using Functional Magnetic Resonance Imaging // Biological Psychiatry. 2005. Vol. 58. Iss. 8. P. 605–613

18. Кремнёва Е.И., Коновалов Р.Н., Кротенкова М.В. Функциональная магнитно-резонансная томография, том 5. [Электронный ресурс]: сайт Научного центра неврологии РАМН. URL: https://annaly-nevrologii.com/journal/files/journals/1/articles/316/public/316-1213-1-PB.pdf (Дата обращения: 25.11.2024).

19. Сорокин Г.А. Фильтры нижних частот. [Электронный ресурс] // URL: https://eneff.susu.ru/m/o/1655/text.pdf (дата обращения 01.06.2025).

20. Основные метрики задач классификации в машинном обучении [Электронный ресурс] // URL: <https://webiomed.ru/blog/osnovnye-metriki-zadach-klassifikatsii-v-mashinnom-obuchenii/?ysclid=mbdgnx0mj2456173819> (дата обращения 01.06.2025).

21. Ансамблевые методы машинного обучения [Электронный ресурс] // URL: https://habr.com/ru/articles/571296/ (дата обрщения 01.06.2025)

22. GridSearchCV. [Электронный ресурс] // URL: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html (дата обращения: 01.06.2025)

23. AutoML: гид по автоматизации машинного обучения для начинающих [Электронный ресурс] // URL: https://habr.com/ru/companies/skillfactory/articles/886770/ (дата обращения: 01.06.2025)

# ПРИЛОЖЕНИЕ А

# Листинг кода приложения для считывания данных

#include <atomic>

#include <chrono>

#include <csignal>

#include <iostream>

#include <thread>

#include <fstream>

#include <future>

#include "CClientAPI.h"

#include "ExampleUtils.hpp"

#include "Capsule/CClient.h"

#include "Capsule/CDevice.h"

#include "Capsule/CSession.h"

using namespace std::chrono\_literals;

// objects for establishing a connection with the device and capsule

clCClient client = nullptr;

clCSession session = nullptr;

clCNFBCalibrator calibrator = nullptr;

clCDeviceLocator locator = nullptr;

clCDevice device = nullptr;

// object for processing raw EEG signal and receiving NFB rhythms

clCNFB nfb = nullptr;

// object for processing the user's EEG rhythms and obtaining productivity score and other data

clCNFBMetricProductivity productivity = nullptr;

// object for processing the user's ECG rhythms

clCCardio cardio = nullptr;

// object for processing the user's MEMS

clCMEMS mems = nullptr;

uint32\_t s\_time = 0;

uint32\_t deviceConnectionTime = 0;

std::atomic<bool> clientStopRequested = false;

std::atomic<bool> clientDisconnecting = false;

std::ofstream sessionEegStreamNFB;

std::ofstream sessionEegStreamProd;

std::ofstream sessionEegStreamCardio;

void onResistances([[maybe\_unused]] clCDevice device, clCResistances resistances) {

// Get total number of resistance channels.

const int32\_t count = clCResistances\_GetCount(resistances);

std::cout << "Resistances: " << count << std::endl;

for (int32\_t i = 0; i < count; ++i) {

const char\* channelName = clCString\_CStr(clCResistances\_GetChannelName(resistances, i));

const float value = clCResistances\_GetValue(resistances, i);

std::cout << "\t " << channelName << " = " << value << std::endl;

}

}

void onUpdateUserState([[maybe\_unused]] clCNFB nfb, const clCNFBUserState\* userState) {

// Getting NFB user data

// if artifacts or weak resistance on the electrodes are observed,

// the data will not be changed

std::cout << "NFB update state: alpha = " << userState->feedbackData[2] << " , beta = " << userState->feedbackData[4] << " , theta = " << userState->feedbackData[1] << std::endl;

sessionEegStreamNFB << time(NULL) << ',';

for (int i = 0; i < userState->feedbackSize; i++) {

sessionEegStreamNFB << userState->feedbackData[i] << ",";

}

sessionEegStreamNFB << std::endl;

//sessionEegStream << time(NULL) << ',' << userState->feedbackData[2] << ',' << userState->feedbackData[4] << ',' << userState->feedbackData[1] << std::endl;

}

void onNFBErrorEvent([[maybe\_unused]] clCNFB nfb, const char\* error) {

std::cerr << "NFB error: " << error << std::endl;

}

void onProductivityArtifacts([[maybe\_unused]] clCNFBMetricProductivity productivity, [[maybe\_unused]] const clCNFBUserArtifacts\*) {

sessionEegStreamProd << "ERRORonProductivityArtifacts";

std::cout << "Productivity received information about artifacts" << std::endl;

}

void onProductivityIndividualInfo([[maybe\_unused]] clCNFBMetricProductivity productivity,

[[maybe\_unused]] const clCNFBMetricsProductivityIndividualIndexes\*) {

sessionEegStreamProd << "ERRORonProductivityIndividualInfo";

std::cout << "Productivity received individual indexes" << std::endl;

}

void onProductivtyScoreUpdate([[maybe\_unused]] clCNFBMetricProductivity productivity, float productivityScore) {

sessionEegStreamProd << "ERRORonProductivtyScoreUpdate";

std::cout << "Productivity received productivity score: " << productivityScore << std::endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*

void onProductivityValuesUpdate(clCNFBMetricProductivity, const clCNFBMetricsProductivityValues\* values) {

//Запись самих метрик

sessionEegStreamProd << time(NULL)

<< ',' << values->fatigueScore

<< ',' << values->gravityScore

<< ',' << values->concentrationScore

<< ',' << values->relaxationScore

<< ',' << values->accumulatedFatigue

<< ',' << values->fatigueGrowthRate << std::endl;

std::cout << "Productivity values update:\n"

<< "\tFatigue Score: " << values->fatigueScore << '\n'

<< "\tGravity Score: " << values->gravityScore << '\n'

<< "\tConcentration Score: " << values->concentrationScore << '\n'

<< "\tRelaxation Score: " << values->relaxationScore << '\n'

<< "\tAccumulated Fatigue: " << values->accumulatedFatigue << '\n'

<< "\tFatigue Growth Rate: " << values->fatigueGrowthRate << std::endl;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*

void onCardioIndexesUpdate([[maybe\_unused]] clCCardio cardio, clCCardioData data) {

sessionEegStreamCardio << time(NULL) << ',' << data.kaplanIndex << ',' << data.heartRate << ',' << data.stressIndex << std::endl;

std::cout << "Cardio indexes update: (artifacted " << data.artifacted

<< "), Kaplan's index " << data.kaplanIndex << ", HR " << data.heartRate

<< ", stress index " << data.stressIndex << ", skin contact " << data.skinContact

<< ", motion artifacts " << data.motionArtifacts << ", metrics available" << data.metricsAvailable << std::endl;

}

void onMEMSUpdate([[maybe\_unused]] clCMEMS mems, clCMEMSTimedData data) {

const int32\_t count = clCMEMSTimedData\_GetCount(data);

std::cout << "MEMS update: showing 1 of " << count << " values" << std::endl;

const clCPoint3d accelerometer = clCMEMSTimedData\_GetAccelerometer(data, 0);

const clCPoint3d gyroscope = clCMEMSTimedData\_GetGyroscope(data, 0);

const auto timepoint = clCMEMSTimedData\_GetTimepoint(data, 0);

std::cout << "\taccelerometer: " << accelerometer.x << ", "

<< accelerometer.y << ", " << accelerometer.z << std::endl;

std::cout << "\tgyroscope: " << gyroscope.x << ", "

<< gyroscope.y << ", " << gyroscope.z << std::endl;

std::cout << "\ttime: " << std::to\_string(timepoint) << std::endl;

}

void onCalibrated(clCNFBCalibrator, const clCIndividualNFBData\* data, clCIndividualNFBCalibrationFailReason failReason) {

if (data == nullptr || failReason != clC\_IndividualNFBCalibrationFailReason\_None) {

std::cerr << "Calibration failed";

sessionEegStreamProd << "ERRORCalibrationfailed";

switch (failReason) {

case clC\_IndividualNFBCalibrationFailReason\_TooManyArtifacts:

std::cerr << ": Too many artifacts";

break;

case clC\_IndividualNFBCalibrationFailReason\_PeakIsABorder:

std::cerr << ": Alpha peak matches one of the alpha range borders";

break;

default:

std::cerr << ": Reason unknown";

}

}

std::cout << "IAF:" << data->individualFrequency << std::endl;

// Productivity metrics is an algorithm that uses custom rhythms

// and an IAPF and other indicators to calculate the productivity score

// Create Productivity metric - math constants are better not to change yet

clCError error = clC\_Error\_OK;

productivity = clCNFBMetricsProductivity\_CreateCalibrated(session, data, &error);

if (error != clC\_Error\_OK) {

std::cerr << "Failed to create Productivity classifier" << std::endl;

sessionEegStreamProd << "ErrorFailed to create Productivity classifier" << std::endl;

return;

}

// Get Productivity events

sessionEegStreamProd << "NotError";

clCNFBMetricsProductivityIndividualDelegate delegateMeasured = clCNFBMetricsProductivity\_GetOnIndividualMeasuredEvent(productivity);

clCNFBMetricsProductivityArtifactsDelegate delegateArtifacts = clCNFBMetricsProductivity\_GetOnArtifactsEvent(productivity);

clCNFBMetricsProductivityEventDelegate delegateUserUpdate = clCNFBMetricsProductivity\_GetOnUpdateEvent\_1min(productivity);

clCNFBMetricsProductivityValuesDelegate delegateValuesUpdate = clCNFBMetricsProductivity\_GetOnProductivityValuesEvent(productivity);

// Initialize Productivity events

clCNFBMetricsProductivity\_IndividualMeasuredEvent\_Set(delegateMeasured, onProductivityIndividualInfo);

clCNFBMetricsProductivity\_ArtifactsEvent\_Set(delegateArtifacts, onProductivityArtifacts);

clCNFBMetricsProductivity\_UpdateEvent\_Set(delegateUserUpdate, onProductivtyScoreUpdate);

clCNFBMetricsProductivity\_ValuesEvent\_Set(delegateValuesUpdate, onProductivityValuesUpdate);

//clCNFBMetricsProductivity\_GetOnUpdateEvent\_1min()

// Initialize Productivity

}

void onCalibratorReady(clCNFBCalibrator calibrator) {

std::cout << "Calibrator is ready to calibrate NFB\n"

<< "Close your eyes for 30 seconds" << std::endl;

clCError error = clC\_Error\_OK;

clCNFBCalibrator\_CalibrateIndividualNFBQuick(calibrator, &error);

}

void onSessionStarted(clCSession session) {

std::cout << "Session started" << std::endl;

const char\* sessionUUID = clCString\_CStr(clCSession\_GetSessionUUID(session));

std::cout << "Session UUID: " << sessionUUID << std::endl;

calibrator = clCNFBCalibrator\_CreateOrGet(session);

clCNFBCalibratorDelegateIndividualNFBCalibrated onCalibratedEvent = clCNFBCalibrator\_GetOnIndividualNFBCalibratedEvent(calibrator);

clCNFBCalibratorDelegateIndividualNFBCalibrated\_Set(onCalibratedEvent, onCalibrated);

clCNFBCalibratorDelegateReadyToCalibrate onCalibratorReadyEvent = clCNFBCalibrator\_GetOnReadyToCalibrate(calibrator);

clCNFBCalibratorDelegateReadyToCalibrate\_Set(onCalibratorReadyEvent, onCalibratorReady);

// Create NFB - NeiryFeedBack

nfb = clCNFB\_Create(session);

// get NFB events

clCNFBDelegateNFBUserState delegateState = clCNFB\_GetOnUserStateChangedEvent(nfb);

clCNFBDelegateString delegateError = clCNFB\_GetOnErrorEvent(nfb);

// Initialize NFB events

clCNFBDelegateNFBUserState\_Set(delegateState, onUpdateUserState);

clCNFBDelegateString\_Set(delegateError, onNFBErrorEvent);

// Initialize NFB

clCNFB\_Initialize(nfb);

if (true) {

sessionEegStreamNFB.open("session\_NFB.csv");

sessionEegStreamProd.open("session\_Prod.csv");

sessionEegStreamProd << "Time"

<< ',' << "Fatigue Score"

<< ',' << "Gravity Score"

<< ',' << "Concentration Score"

<< ',' << "Relaxation Score"

<< ',' << "Accumulated Fatigue"

<< ',' << "Fatigue Growth Rate" << std::endl;

sessionEegStreamCardio.open("session\_Cardio.csv");

sessionEegStreamCardio << "Time"

<< ',' << "Kaplan"

<< ',' << "Heart rate"

<< ',' << "Stress index" << std::endl;

}

clCDevice\_SwitchMode(device, clC\_DM\_Signal);

cardio = clCCardio\_Create(session);

if (cardio == nullptr) {

std::cerr << "Failed to create cardio classifier" << std::endl;

return;

}

clCCardioIndexesDelegate delegateCardio = clCCardio\_GetOnIndexesUpdateEvent(cardio);

clCCardioDelegateIndexesUpdate\_Set(delegateCardio, onCardioIndexesUpdate);

clCCardio\_Initialize(cardio);

clCDevice\_SwitchMode(device, clC\_DM\_StartPPG);

mems = clCMEMS\_Create(session);

if (mems == nullptr) {

std::cerr << "Failed to create MEMS classifier" << std::endl;

return;

}

clCMEMSTimedDataDelegate delegateMEMS = clCMEMS\_GetOnMEMSTimedDataUpdateEvent(mems);

clCMEMSDelegateMEMSTimedDataUpdate\_Set(delegateMEMS, onMEMSUpdate);

clCMEMS\_Initialize(mems);

clCDevice\_SwitchMode(device, clC\_DM\_StartMEMS);

}

void onSessionError([[maybe\_unused]] clCSession session, clCSessionError error) {

std::cerr << "Session error: " << static\_cast<int>(error) << std::endl;

}

void onSessionStopped([[maybe\_unused]] clCSession session) {

std::cout << "Session stopped" << std::endl;

}

void onConnectionStateChanged([[maybe\_unused]] clCDevice device, clCDeviceConnectionState state) {

// status of the device changed

if (state != clC\_SE\_Connected) {

std::cout << "Device disconnected" << std::endl;

return;

}

std::cout << "Device connected" << std::endl;

deviceConnectionTime = s\_time;

// get channel names

clCDeviceChannelNames channelNames = clCDevice\_GetChannelNames(device);

const auto channelsCount = clCDevice\_GetChannelsCount(channelNames);

std::cout << "Device has " << channelsCount << " channels: [";

for (int32\_t i = 0; i < channelsCount; ++i) {

std::cout << clCString\_CStr(clCDevice\_GetChannelNameByIndex(channelNames, i)) << ", ";

}

std::cout << "\b\b]" << std::endl;

for (int32\_t i = 0; i < channelsCount; ++i) {

const char\* channel = clCString\_CStr(clCDevice\_GetChannelNameByIndex(channelNames, i));

std::cout << "\tChannel " << channel << " has index " << clCDevice\_GetChannelIndexByName(channelNames, channel) << std::endl;

}

}

void onDeviceList(clCDeviceLocator locator, clCDeviceInfoList devices, clCDeviceLocatorFailReason error) {

// device connected

if (device != nullptr) {

return;

}

if (error != clC\_DeviceLocatorFailReason\_OK) {

switch (error) {

case clC\_DeviceLocatorFailReason\_BluetoothDisabled:

std::cerr << "Bluetooth adapter not found or disabled";

break;

default:

std::cerr << "Unknown error occurred";

}

std::cerr << ". Exiting..." << std::endl;

clientStopRequested = true;

return;

}

if (clCDeviceInfoList\_GetCount(devices) == 0) {

std::cerr << "Empty device list. Exiting..." << std::endl;

clientStopRequested = true;

return;

}

// print information about all found devices

const int32\_t count = clCDeviceInfoList\_GetCount(devices);

std::cout << "Devices: " << count << std::endl;

for (int i = 0; i < count; i++) {

clCDeviceInfo deviceDescriptor = clCDeviceInfoList\_GetDeviceInfo(devices, i);

const char\* deviceDescription = clCString\_CStr(clCDeviceInfo\_GetDescription(deviceDescriptor));

std::cout << "\t " << deviceDescription << std::endl;

}

// select device and connect

clCDeviceInfo deviceDescriptor = clCDeviceInfoList\_GetDeviceInfo(devices, 0);

const char\* deviceID = clCString\_CStr(clCDeviceInfo\_GetID(deviceDescriptor));

device = clCDeviceLocator\_CreateDevice(locator, deviceID);

if (device == nullptr) {

std::cerr << "Failed to create device. Exiting..." << std::endl;

clientStopRequested = true;

return;

}

// Get device events

clCDeviceDelegateResistances onResistancesEvent = clCDevice\_GetOnResistancesEvent(device);

clCDeviceDelegateDeviceConnectionState onConnectionStateChangedEvent = clCDevice\_GetOnConnectionStateChangedEvent(device);

// Initialize device events

clCDeviceDelegateResistances\_Set(onResistancesEvent, onResistances);

clCDeviceDelegateConnectionState\_Set(onConnectionStateChangedEvent, onConnectionStateChanged);

// Сonnect to the device

clCDevice\_Connect(device);

}

void onConnected(clCClient client) {

// As soon as the capsule has confirmed the connection,

// launch the locator to search for the selected device.

// when receiving a list of available devices, control is transferred to onDeviceList

std::cout << "Connected" << std::endl;

locator = clCClient\_ChooseDeviceType(client, clC\_DT\_NeiryBand);

clCDeviceLocatorDelegateDeviceInfoList onDevicesEvent = clCDeviceLocator\_GetOnDevicesEvent(locator);

clCDeviceLocatorDelegateDeviceInfoList\_Set(onDevicesEvent, onDeviceList);

clCDeviceLocator\_RequestDevices(locator, 15);

}

void onError([[maybe\_unused]] clCClient client, clCError error) {

std::cerr << "Error: " << static\_cast<int>(error) << std::endl;

}

void onDisconnected(clCClient client, clCDisconnectReason reason) {

// destroy all objects

std::cout << "Disconnected: " << static\_cast<int>(reason) << std::endl;

if (mems) {

clCMEMS\_Destroy(mems);

mems = nullptr;

}

if (cardio) {

clCCardio\_Destroy(cardio);

cardio = nullptr;

}

if (nfb) {

clCNFB\_Destroy(nfb);

nfb = nullptr;

}

if (productivity) {

clCNFBMetricsProductivity\_Destroy(productivity);

productivity = nullptr;

}

calibrator = nullptr;

if (session) {

clCSession\_Destroy(session);

session = nullptr;

}

if (device) {

clCDevice\_Release(device);

device = nullptr;

}

clCDeviceLocator\_Destroy(locator);

locator = nullptr;

clCClient\_Destroy(client);

::client = nullptr;

std::cout << "End work" << std::endl;

}

void ClientLoop() {

static constexpr uint32\_t kMsSec = 1000U;

s\_time = 0;

while (client) {

// Update does all the work and must be called regularly to process events.

clCClient\_Update(client);

std::this\_thread::sleep\_for(40ms);

s\_time += 40;

// Create a session object. The session is necessary for

// the client to work with the device - obtaining user state data

// or information about the device

if (deviceConnectionTime && s\_time == deviceConnectionTime + 2 \* kMsSec && !session) {

// Create session

auto error = clC\_Error\_OK;

session = clCClient\_CreateSessionWithError(client, device, &error);

if (error != clC\_Error\_OK) {

std::cerr << "Failed to create session: " << error << std::endl;

clientStopRequested = true;

continue;

}

// Get session events

clCSessionDelegate onSessionStartedEvent = clCSession\_GetOnSessionStartedEvent(session);

clCSessionDelegateSessionError onSessionErrorEvent = clCSession\_GetOnErrorEvent(session);

clCSessionDelegate onSessionStoppedEvent = clCSession\_GetOnSessionStoppedEvent(session);

// Initialize session events

clCSessionDelegate\_Set(onSessionStartedEvent, onSessionStarted);

clCSessionDelegateSessionError\_Set(onSessionErrorEvent, onSessionError);

clCSessionDelegate\_Set(onSessionStoppedEvent, onSessionStopped);

// Start session

const clCSessionState state = clCSession\_GetSessionState(session);

clCSession\_Start(session);

std::cout << "Session state: " << static\_cast<int>(state) << std::endl;

}

if (!session && s\_time == 100 \* kMsSec) {

clientStopRequested = true;

}

if (clientStopRequested && !clientDisconnecting) {

// Destroy all objects that were created at the end of the program and

// break the connection in callback onDisconnected

clCClient\_Disconnect(client);

clientDisconnecting = true;

}

if (static bool printFirmware = true; printFirmware && device && clCDevice\_FirmwareVersionReceived(device)) {

clCError error = clCError::clC\_Error\_OK;

const auto firmware = clCDevice\_GetFirmwareVersion(device, &error);

std::cout << "Device firmware: " << clCString\_CStr(firmware) << std::endl;

clCString\_Free(firmware);

printFirmware = false;

}

}

}

void SignalHandler(int signal) {

if (signal == SIGINT) {

std::cout << "\nCtrl + C pressed. Exiting gracefully...\n";

clientStopRequested = true;

}

}

int main() {

std::signal(SIGINT, SignalHandler);

std::cout << "To quit the example press Ctrl + C" << std::endl;

// Getting the version of the library

// and an example of working with a clCString

{

clCString strPtr = clCClient\_GetVersionString();

std::cout << "Version of the library: " << clCString\_CStr(strPtr) << std::endl;

clCString\_Free(strPtr);

}

// Create client

client = clCClient\_CreateWithName("CapsuleClientExample");

// Get client events

clCClientDelegate onConnectedEvent = clCClient\_GetOnConnectedEvent(client);

clCClientDelegateError onErrorEvent = clCClient\_GetOnErrorEvent(client);

clCClientDelegateDisconnectReason onDisconnectedEvent = clCClient\_GetOnDisconnectedEvent(client);

// Initialize client events

clCClientDelegate\_Set(onConnectedEvent, onConnected);

clCClientDelegateError\_Set(onErrorEvent, onError);

clCClientDelegateDisconnectReason\_Set(onDisconnectedEvent, onDisconnected);

clCClient\_Connect(client, "inproc://capsule");

// Getting the client name of the library

// and an example of working with a clCString

{

clCString strPtr = clCClient\_GetClientName(client);

std::cout << "Client name: " << clCString\_CStr(strPtr) << std::endl;

clCString\_Free(strPtr);

}

ClientLoop();

return 0;

}