

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий – РТФ
Школа профессионального и академического образования

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Директор ШПиАО
Д.В. Денисов
(подпись) (Ф.И.О.)
«_____» _____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**ИЗВЛЕЧЕНИЕ И КЛАССИФИКАЦИЯ ПРИЗНАКОВ ИЗ НАБОРА
ДАННЫХ ОКУЛОГРАФИИ МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ**

Научный руководитель: Долганов Антон Юрьевич
К.Т.Н.

подпись

Нормоконтролер: Огуренко Егор Владимирович

подпись

Студент группы: РИМ-220962 Колосов Илья Викторович

подпись

Екатеринбург
2024

РЕФЕРАТ

Выпускная квалификационная работа магистра 89 стр., 44 рис., 34 источников, 2 прил.

OCULOGRAPHY, EVENT DETECTION, EYE MOVEMENTS, MACHINE LEARNING, FIXATIONS, SACCADDES, INTERVAL THRESHOLD, ИЗВЛЕЧЕНИЕ ПРИЗНАКОВ, КЛАССИФИКАЦИЯ, АНОМАЛИИ, RFE

Целью данного исследования является анализ методов машинного обучения для извлечения и классификации признаков из наборов данных окулограмм для повышения точности диагностики дислексии.

Объектом исследования является обнаружение различных событий движения глаз из данных окулограммы, включая прогрессивные и регрессивные движения глаз, фиксации, саккады и типы полей зрения.

В исследовании изучается использование пороговых алгоритмов и методов кластеризации на основе аномалий для идентификации этих событий.

В исследовании используется стратифицированная перекрестная валидация и рекурсивный отбор признаков для выбора наиболее значимых признаков для моделей машинного обучения, а также изучается вариативность рекурсивного отбора признаков на основе точности предсказания и вероятности извлечения признаков.

СОДЕРЖАНИЕ

РЕФЕРАТ	2
СОДЕРЖАНИЕ	3
ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	5
ВВЕДЕНИЕ	6
1 Аналитический литературный обзор	8
1.1 Влияние различных патологий центральной нервной системы на окулоmotorную активность	8
1.2 Движения глаз и методы регистрации	10
1.3 Окулограмма и типы движения глаз	12
1.4 Пороговые методы извлечения типов(признаков) движения глаз из данных окулографии и их программная реализация.....	16
1.5 Обзор результатов предшествующих исследований, в которых применялись методы машинного обучения для анализа данных окулографии.....	21
1.6 Анализ моделей машинного обучения, применяемых для извлечения и классификации признаков из данных окулографии	24
1.6.1 Классификация событий с помощью классификатора Random forest	25
1.6.2 Классификация событий с помощью CNN	27
1.6.3 Классификация событий с помощью RNN (CNN с рекуррентным слоем)	29
1.7 Выявление разрыва между требованиями точной диагностики окулоmotorной активности и имеющимися методами анализа данных окулографии.....	30
1.8 Предложение изменений в методологии анализа данных окулографии с целью повышения точности диагностики окулоmotorной активности у пациентов	33
2 Материалы и методы	35
2.1 Описание набора данных окулографии	35
2.2 Описание библиотек	36
2.3 Объединение, предобработка и анализ данных. Статистики и визуализация данных	38
2.4 Извлечение признаков пороговыми методами	45
2.4.1 Метод IVT	45
2.4.2 Метод IDT	46
2.4.3 Модифицированный метод IDT	47
2.4.4 Метод идентификации транзитивных событий	50

2.5 Извлечение признаков методами машинного обучения	51
2.5.1 Метод DBSCAN	51
2.5.2 Метод HDBSCAN	54
2.5.3 Метод IOF	56
2.6 Другие подходы для извлечения признаков	57
2.6.1 Метод выявления типов полей зрения	57
2.6.2 Математические признаки	60
2.7 Результаты	62
 3 Отбор и оценка признаков. Сравнение результатов отбора. Классификация испытуемых	 65
3.1 Применение машинного обучения	66
3.2 Отбор с константным значением повторов и оценка признаков набора Quantitative	69
3.3 Отбор с константным значением повторов и оценка признаков набора Relative	73
3.4 Отбор с вычисляемым значением повторов и оценка признаков набора Quantitative	76
3.5 Классификация испытуемых на полученном наборе признаков	80
3.6 Результаты	82
 ЗАКЛЮЧЕНИЕ	 83
СПИСОК ИСПОЛЪЗУЕМЫХ ИСТОЧНИКОВ	85
ПРИЛОЖЕНИЕ А. КОД СКРИПТОВ	90
ПРИЛОЖЕНИЕ Б. ГРАФИКИ ИССЛЕДОВАНИЯ	132

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЦНС – Центральная нервная система

ЭОГ – Электроокулограммы

МО – машинное обучение

I-VT – Порог скорости

I-HMM – Скрытая марковская модель

I-DT – Порог рассеивания(дисперсии)

I-MST – Минимальное охватывающее дерево

I-AOI – Зона интереса

RF – случайный лес

CNN – сверточные нейронные сети

RNN – рекуррентные нейронные сети

PSO – постсаккадическими колебаниями

БПФ – быстрое преобразование Фурье

BLSTM – классический рекуррентный слой, сохраняющий информацию о предыдущих выборках

HR – группа в высоком риске

LR – группа с низким риском (контрольная группа)

IVT - Interval Variable Threshold

IDT - Interval of Fixations

RMS – среднеквадратичная ошибка

DBSCAN – Density-Based Spatial Clustering of Applications with Noise

HDBSCAN – Hierarchical Density-Based Spatial Clustering of Applications with Noise

LOF – Local Outlier Factor

XGBoost – (Extreme Gradient Boosting)

ВВЕДЕНИЕ

В современном мире машинное обучение и анализ данных стали неотъемлемой частью многих областей науки и техники. Одним из направлений, где применение этих методов имеет важное значение, является медицинская диагностика и исследование окуломоторной активности. Окулография, изучающая движения глаз и их характеристики, становится все более распространенным методом анализа в различных медицинских областях, включая неврологию.

Работа направлена на извлечение и классификацию признаков из данных окулографии с использованием методов машинного обучения. Основным объектом нашего анализа являются данные, собранные с помощью технологии Eye-Tracking у пациентов с неврологическими патологиями, которые могут влиять на окуломоторную систему. Предметом исследования в данной работе является задача извлечения и классификации признаков из набора данных окулографии с использованием методов машинного обучения.

Целью данной работы является повышение точности диагностики и понимания окуломоторной активности путем применения методов машинного обучения к набору данных окулографии. Это позволит более эффективно выявлять и классифицировать особенности движений глаз у пациентов с неврологическими патологиями, что в свою очередь может существенно помочь в диагностике, лечении и последующей реабилитации после данного заболевания. Для достижения указанной цели необходимо решить ряд задач:

1. Изучение различных подходов, методов ML и предметной области, связанной с глазодвигательными событиями в анализе данных окулограмм;
2. Провести анализ набора данных окулограмм, полученных в ходе экспериментов, проведенных с целью изучения взаимосвязи между определенной группой риска и дислексией;

3. Реализация и написание программного кода для пороговых алгоритмов и методов МО с целью выявления глазодвигательных событий и генерации новых признаков;

4. Реализовать модель рекурсивного отбора значимых признаков и на результатах ее работы провести процедуры классификации окулограмм испытуемых.

В ходе исследования использовались различные подходы и методы машинного обучения для анализа окуломоторной активности, которые позволяют находить различные регрессивные и прогрессивные глазодвигательные типы событий, а также выявить типы полей зрения. Это позволило провести оценку уже апробированных подходов и разработать новые методы извлечения и классификации признаков из данных окулографии, способные эффективно анализировать окуломоторную активность.

Таким образом, результаты данной работы могут оказать значительное практическое влияние на медицинскую диагностику и исследования, а также способствовать развитию методов анализа данных окулографии с применением машинного обучения. Ожидается, что исследование приведет к разработке новых методов анализа данных окулографии, что поможет улучшить точность диагностики и понимание окуломоторных нарушений у пациентов с неврологическими патологиями. Этот вклад будет иметь значительное значение для медицинской практики и способствует продвижению методов анализа данных окулографии с использованием машинного обучения.

1 Аналитический литературный обзор

Поиск актуальных источников по выбранной теме проводился с помощью баз данных научных публикаций eLibrary, Scopus, Web of Science, CyberLeninka, ResearchGate, ARxiv и поисковой системе Google. Ключевыми словами при поиске являлись: «oculography», «Event detection», «Eye movements», «Machine learning», «Fixations», «Saccades». Сортировка результатов поиска производилась по индексу цитирования и году публикации.

1.1 Влияние различных патологий центральной нервной системы на окулоmotorную активность

Центральная нервная система (ЦНС) - это сложная система, состоящая из головного и спинного мозга. Она отвечает за управление различными функциями организма, такими как движение, ощущения, мышление, память и эмоции. Мозг - это центр, отвечающий за обработку и передачу информации по всему телу, состоящий из различных частей, включая головной мозг, мозжечок и ствол мозга.

Неврология - это медицинская специальность, занимающаяся диагностикой и лечением состояний и патологий, связанных с нервной системой, включая головной и спинной мозг, а также периферические нервы. Для диагностики и лечения неврологических патологий неврологи используют такие передовые методы, как нейровизуализация, нейрофизиологические исследования и генетические тесты. К распространенным неврологическим заболеваниям в частности относятся:

Дислексия – специфическое расстройство обучения, характеризуется трудностями в точном и беглом распознавании слов, а также плохими способностями к правописанию и декодированию. Хотя дислексия не является

психическим заболеванием, она имеет неврологическую основу[15; 25; 27]. Исследования показывают, что дислексия связана с различиями в структуре и функционировании мозга, особенно в областях, связанных с языком и чтением.

Шизофрения – это тяжелое психическое расстройство, которое влияет на то, как человек думает, чувствует и ведет себя. Хотя точные причины шизофрении неизвестны, ясно, что нейробиологические факторы играют важную роль. Исследования по визуализации мозга показали, что у людей с шизофренией часто наблюдаются структурные и функциональные различия в различных областях мозга, особенно в областях, связанных с восприятием, эмоциями и познанием.

Инсульт – это серьезное заболевание, возникающее при нарушении или уменьшении притока крови к мозгу, что приводит к гибели клеток мозга. Он может быть вызван закупоркой или разрывом кровеносных сосудов в мозге.

Неврологи тесно сотрудничают с другими медицинскими специалистами для разработки комплексных планов лечения пациентов, а также могут проводить исследования в области неврологии.

В случаях упомянутых выше неврологических состояний и патологий движения глаз являются важным аспектом для понимания функционирования ЦНС. Окуломоторная активность и регистрация движений глаз предоставляют информацию о процессах регулирования движений, организации познавательных процессов, состояниях человека, его деятельности и общении, а также обнаружении патологий ЦНС. Развитие методов регистрации движений глаз имеет большое значение для нейронаучных исследований и диагностики неврологических состояний.

1.2 Движения глаз и методы регистрации

Отслеживание движения глаз - это процесс отслеживания движения глаз, чтобы точно знать, куда и как долго человек смотрит [10]. Основная цель движения глаз(рис.1.2.1) - направить взгляд на объект и удержать его в центре фовеа (центральная ямка), чтобы обеспечить четкое видение объекта. Отслеживание движения глаз используется в различных областях исследований, таких как когнитивные науки, психология, неврология, инженерия, медицина и маркетинг, и это лишь некоторые из них [14].

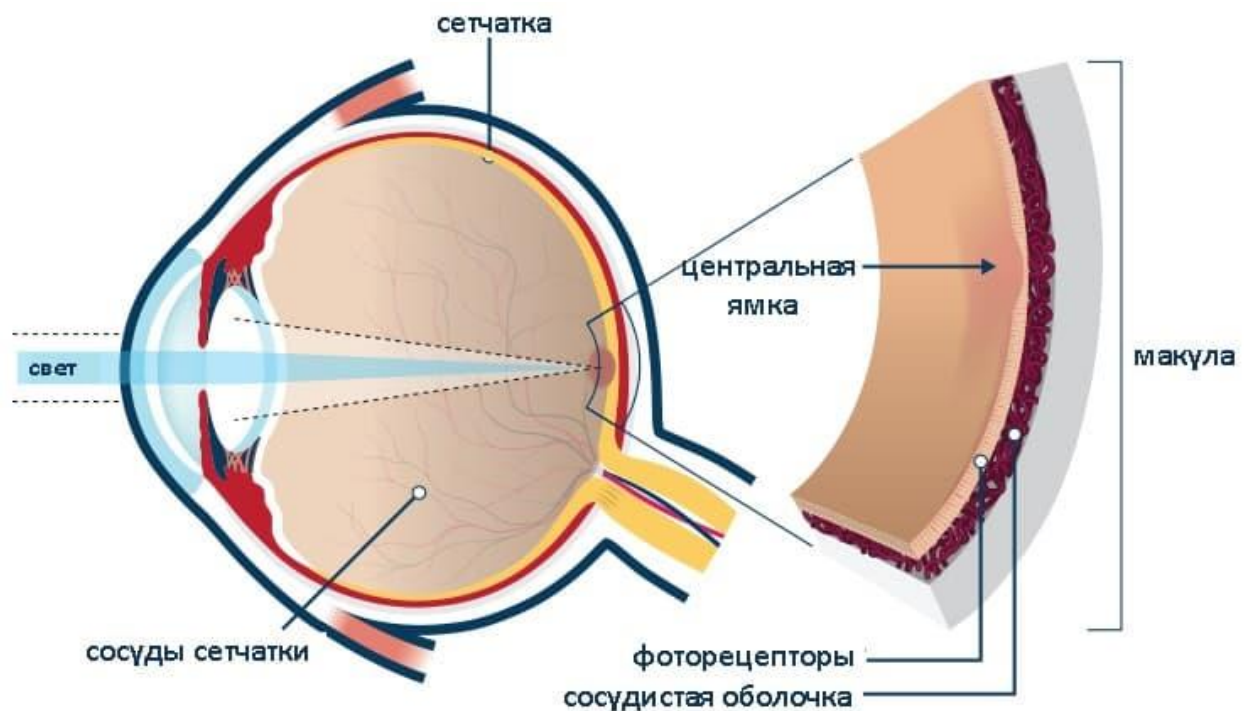


Рисунок 1.2.1 – Строение глаза

Существует несколько методов отслеживания движений глаз для оценки здоровья и функционирования ЦНС.

Окулография – процесс отслеживания движения глаз [17] для определения координат точки взгляда, где оптическая ось глазного блока пересекается с плоскостью наблюдаемого зрительного раздражителя [4].

Полученные результаты процесса обычно записываются для последующего анализа и интерпретации.

Айтрекинг – это процесс отслеживания и анализа движения глаз для получения информации о визуальном поведении и внимании. Айтрекинг используется в различных областях, включая психологию, маркетинг и дизайн пользовательского опыта[21], чтобы понять, как люди взаимодействуют с визуальными стимулами и принимают решения.

Электроокулография (ЭОГ) – процесс измеряющий существующий электрический потенциал покоя между роговицей и мембраной Бруха [29] (рис.1.2.2), а полученный сигнал называется электроокулограммой. Этот метод широко используется в медицинских и научных исследованиях, а также в клинических условиях для оценки расстройств движения глаз и неврологических состояний.

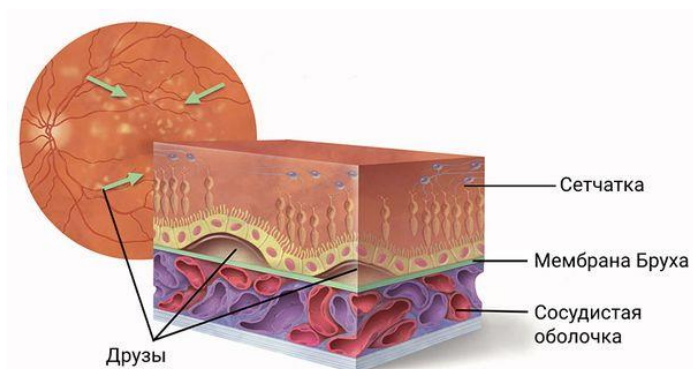


Рисунок 1.2.2 – мембрана Бруха

Описанные методы отслеживания и анализа движения глаз служат ценными инструментами для изучения ЦНС и находят широкое применение - от диагностики и лечения неврологических расстройств до оптимизации визуальных интерфейсов и пользовательского опыта.

1.3 Окулограмма и типы движения глаз

Окулограмма - это набор данных представляющих собой запись движения глаз (рис.1.3.1). Окулограмма является полезным инструментом для оценки здоровья и функции ЦНС и широко используется в диагностике и лечении различных неврологических расстройств. С помощью окулограммы неврологи и другие медицинские работники могут получить ценные сведения о здоровье и функции ЦНС, что поможет улучшить состояние пациентов и повысить качество лечения.

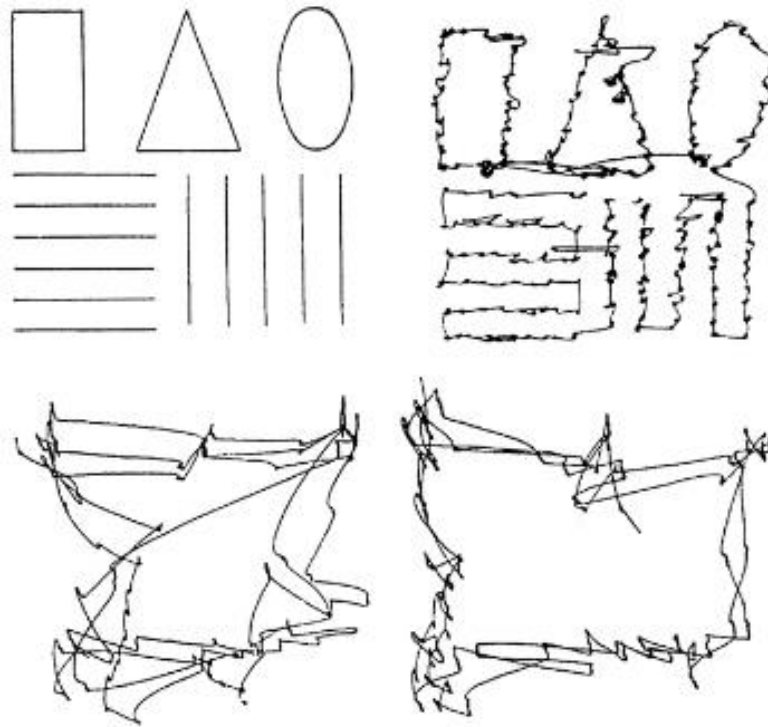


Рисунок 1.3.1 – Визуализация записи движения глаз при рассматривании геометрических фигур[28]

Обнаружение событий - это процесс анализа трекинга глаз для выявления и классификации событий движения глаз, который уменьшает сложность анализа движения глаз [26].

События движения глаз делятся на несколько типов (виды движений глаз): тремор, дрейф, микро- и макросаккады, прослеживающие, вергентные и торзионные движения, нистагм и т.д. Каждый из них обладает характерными биомеханическими свойствами (амплитудой, скоростью, частотой, траекторией и т.д.) и подчинен соответствующей системе контроля.

Приведем краткое описание движений глаз:

- Тремор – мелкие, частые колебания глаз (рис.1.3.2). Средняя амплитуда – 20–40", частота – до 250–270 Гц;

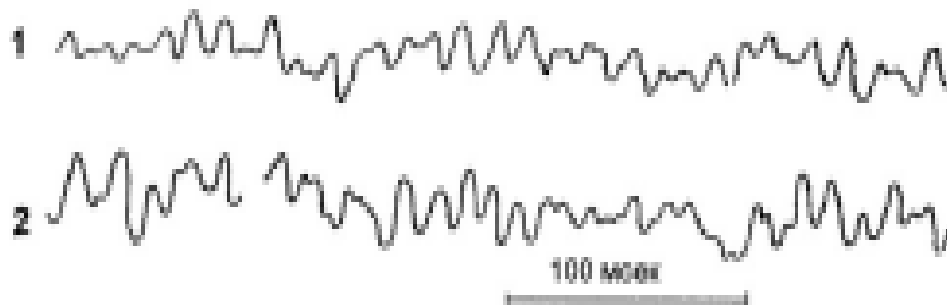


Рисунок 1.3.2 – Тремор правого (1) и левого (2) глаза в норме

- Дрейф – медленное, плавное перемещение глаза, прерываемое микросаккадами (рис.1.3.3). Скорость дрейфа меняется от 0 до 40°/с, длительность – от 30 до 5000 мс;

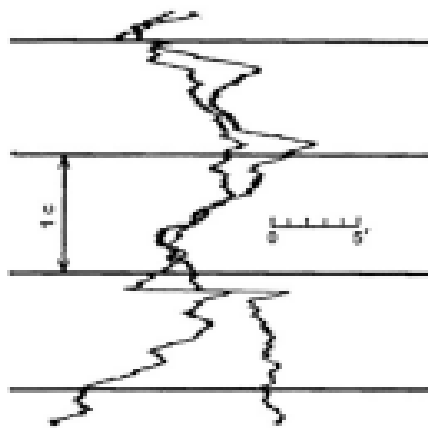


Рисунок 1.3.3 – Окулограмма движений (горизонтальная составляющая) двух глаз в процессе фиксации испытуемым неподвижной точки

- Микросаккады – быстрые резкие движения продолжительностью 10–20 мс. Диапазон амплитуд – 2–50', скорость от 3 до 12°/с;
- Макросаккады – резкие изменения позиции глаза, отличающиеся высокой скоростью и точностью (рис.1.3.4);



Рисунок 1.3.4 – Микросаккада, Запись скачков глаза между углами квадрата

- Прослеживающие (следящие) движения – плавные перемещения глаз, возникающие при движении объекта в поле зрения (рис.1.3.5);

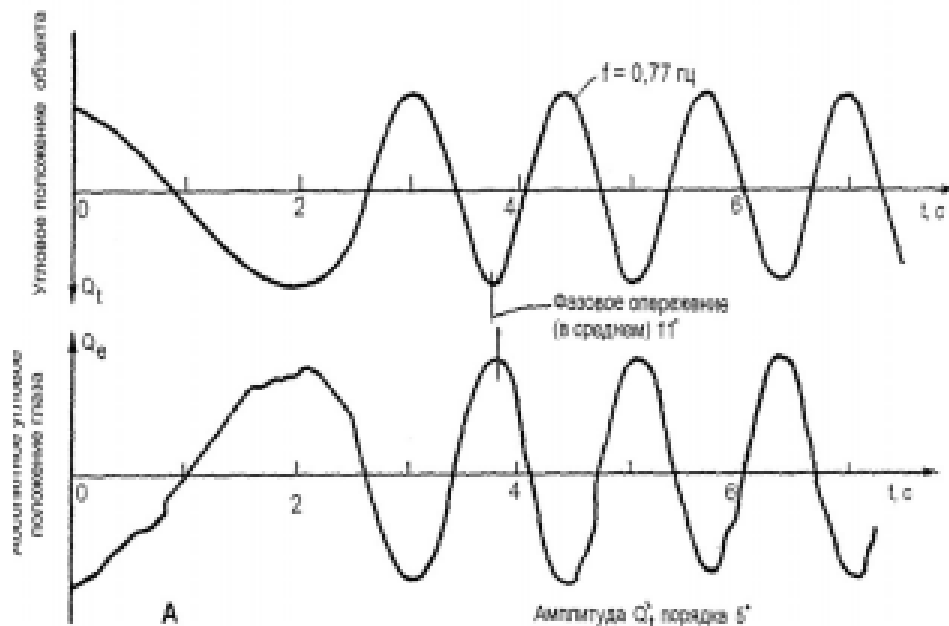


Рисунок 1.3.5 – Режим плавного слежения за перемещающейся точкой по предсказанной траектории

- Вергентные движения – сведение (конвергенция) или разведение (дивергенция) оптических осей глаз(рис.1.3.6);



Рисунок 1.3.6 – Вергентные движения глаз при смене точек фиксации

- Торзионные, или ротационные движения – вращательные перемещения глаз относительно оптической оси (рис.1.3.7);

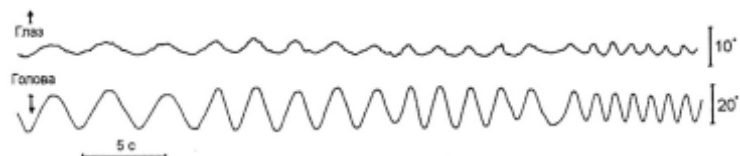


Рисунок 1.3.7 – Запись ротационных движений глаз (вверху) при наклонах головы из стороны в сторону (внизу)

- Нистагм – устойчивая окуломоторная структура, включающая чередование саккад и плавных преслеживающих движений (рис.1.3.8).

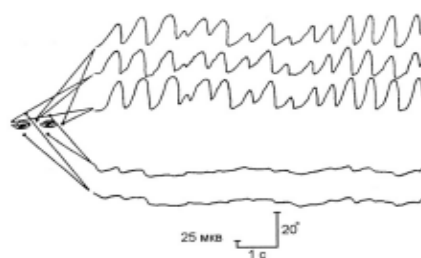


Рисунок 1.3.8 – Оптикинетический нистагм

В начальных исследованиях, проводившихся в 50-60-е годы прошлого века, ученые сталкивались с необходимостью затратить значительное количество времени на применение ручных методов [30] обработки

информации с окулограм для извлечения типов движений глаз из полученных данных. Этот процесс требовал усердной и тщательной работы, так как каждое движение глаз должно было быть вручную проанализировано и классифицировано.

1.4 Пороговые методы извлечения типов(признаков) движения глаз из данных окулографии и их программная реализация

С появлением новых цифровых возможностей в начале текущего века исследователи разработали пороговые методы обработки данных [23] и осуществили их программную реализацию (таблица 1.4.1). Это дало возможность извлечь основные признаки движений глаз: тремор, дрейф, микросаккады, макросаккады, прослеживающие (следящие) движения, вергентные движения, торзионные движения, нистагм и фиксации. Однако из-за отсутствия стандартной процедуры оценки, оценка и сравнение различных методов обнаружения в сигналах трекинга глаз является очень сложной задачей.

Таблица 1.4.1 – Методы извлечения признаков

Метод	Точность	Скорость	Устойчивость	Простота	Параметры
Порог скорости (I-VT)	v	vv	x	vv	1
Скрытая марковская модель (I-HMM)	vv	v	vv	V / x^b	$8 / 0^a$
Порог рассеивания (I-DT)	vv	v	vv	v	2
Минимальное охватывающее дерево (I-MST)	v	x	vv	x	2
Зона интереса(I-AOI)	x	v	v	v	$1 + ^b$
Ключ: vv = очень хорошо, v = хорошо, x = не очень хорошо. ^a I-HMM имеет 8 параметров в своей двухсоставной НММ, но может быть выучена путем повторной оценки. Без переоценки ИНММ имеет 8 параметров, но более проста в реализации; с переоценкой I-HMM фактически не имеет параметров, но более сложна в реализации. ^b I-AOI имеет 1 параметр, но также требует указания целевых областей.					

В итоге исследователи выяснили [23], что алгоритмы I-HMM (таблица 1.4.2) и пороговый I-DT (таблица 1.4.4) обеспечивают точную и надежную идентификацию фиксаций за счет включения последовательной информации для интерпретации. Метод I-MST (таблица 1.4.3) также надежен, но работает медленнее. Пороговый метод I-VT (таблица 1.4.2) имеет простой алгоритм и низкие вычислительные затраты, но может испытывать эффекты "всплесков" и низкую эффективность, когда сигнал зашумлен. Пороговый I-AOI (таблица 1.4.3) не рекомендуется из-за низких результатов.

Таблица 1.4.2 – Псевдокод для методов I-VT и I-HMM

I-VT (протокол. порог скорости)	I-HMM (протокол, Скрытая марковская модель)
Рассчитать скорости от точки к точке для каждой точки в протоколе; Пометить каждую точку, скорость которой ниже порога, как точку фиксации, в противном случае как точку саккады; Свернуть последовательные точки фиксации в группы фиксации, удалив точки саккады; Сопоставить каждую группу фиксации фиксации в центроиде ее точек; Вернуть фиксации.	Рассчитать скорости от точки к точке для каждой точки в протоколе; Декодировать скорости с двух состояний НММ для идентификации точек как точек фиксации или саккады Свернуть последовательные точки фиксации в группы фиксации, удалив точки саккады Сопоставить каждую группу фиксации фиксации в центроиде ее точек Вернуть фиксации

Во-первых, методы, основанные на скорости и дисперсии, обеспечивают примерно одинаковую производительность, но алгоритмы, основанные на площади, могут давать искаженные результаты.

Во-вторых, использование временной информации значительно упрощает идентификацию фиксаций. Три метода, использующие эту информацию локально адаптивным образом (I-HMM, I-DT и I-MST), обеспечивают надежные интерпретации даже при наличии шума.

В-третьих, выбор алгоритмов идентификации существенно влияет на итоговые результаты.

Таблица 1.4.3 – Псевдокод для методов I-MST и I-AOI

I-MST (протокол, соотношение рёбер, стандартное отклонение рёбер)	I-AOI (протокол, пороговая длительность, целевые области)
Построить MST из точек данных протокола с использованием алгоритма Прима; Найти максимальную глубину ветвления для каждой точки MST с помощью поиска в глубину; Идентифицировать саккады как рёбра, длина которых превышает заранее определенные критерии; Определить параметрические свойства (#, если) локальных рёбер, идентифицируя саккады, когда длина ребра превышает заданное соотношение; Идентифицировать фиксации как кластеры точек, не разделенные саккадами; Вернуть фиксации.	Пометить каждую точку как точку фиксации для целевой области, в которой она находится, или как точку саккады, если ни одна из них не соответствует Свернуть последовательные точки фиксации для одной и той же цели в группы фиксации, удалив точки саккады Удалить группы фиксации, не превышающие минимальный порог длительности Сопоставить каждую группу фиксации фиксации в центроиде ее точек Вернуть фиксации

Таблица 1.4.4 – Псевдокод для методов I-DT

I-DT (протокол, порог дисперсии, порог длительности)

Пока остаются точки

 Инициализировать окно над первыми точками для охвата порога длительности

 Если дисперсия точек окна $\leq$ порогу

 Добавить дополнительные точки в окно, пока дисперсия $>$ порога

 Зафиксировать фиксацию в центроиде точек окна

 Удалить точки окна из точек

 Иначе

 Удалить первую точку из точек

Вернуть фиксации

В дальнейшем часть программного реализованных пороговых алгоритмов была протестирована на процессе извлечения признаков в окулографии в различных включая психологическую практику [3], скрининг [24] и диагностику детской дислексии (рис.1.4.1) [32], диагностику шизофрении [33] в психологии, кинематографии [5], неврологии при изучении ЦНС [6], подготовке специалистов психолого-педагогического профиля [31], применение шлема виртуальной реальности [2] в работе интерфейса «человек-компьютер» (ИЧК) [7], а так же офтальмологии [11]. Таким образом, процедура отслеживания движений, извлечения и классификации типов движения глаз стала междисциплинарной и используется в различных областях, что также отражается в том, как аппаратное и программное обеспечение для отслеживания глаз развивалось на протяжении многих лет [13].

На рис. 1.4.1 показан пример анализа движения глаз, где отображаются горизонтальный и вертикальный сигналы движения глаз во времени.

Различные цвета обозначают различные типы движений: саккады (светло-зеленый), фиксации (светло-серый), скользящие движения (светло-синий) и транзиенты (красный). Анализ проводился с помощью алгоритма динамического порога дисперсии, который определяет четыре состояния: искажение, транзиент, фиксация и саккада. Алгоритм анализирует сигнал отслеживания выборок и переключается между состояниями в зависимости от физиологических свойств зрения.

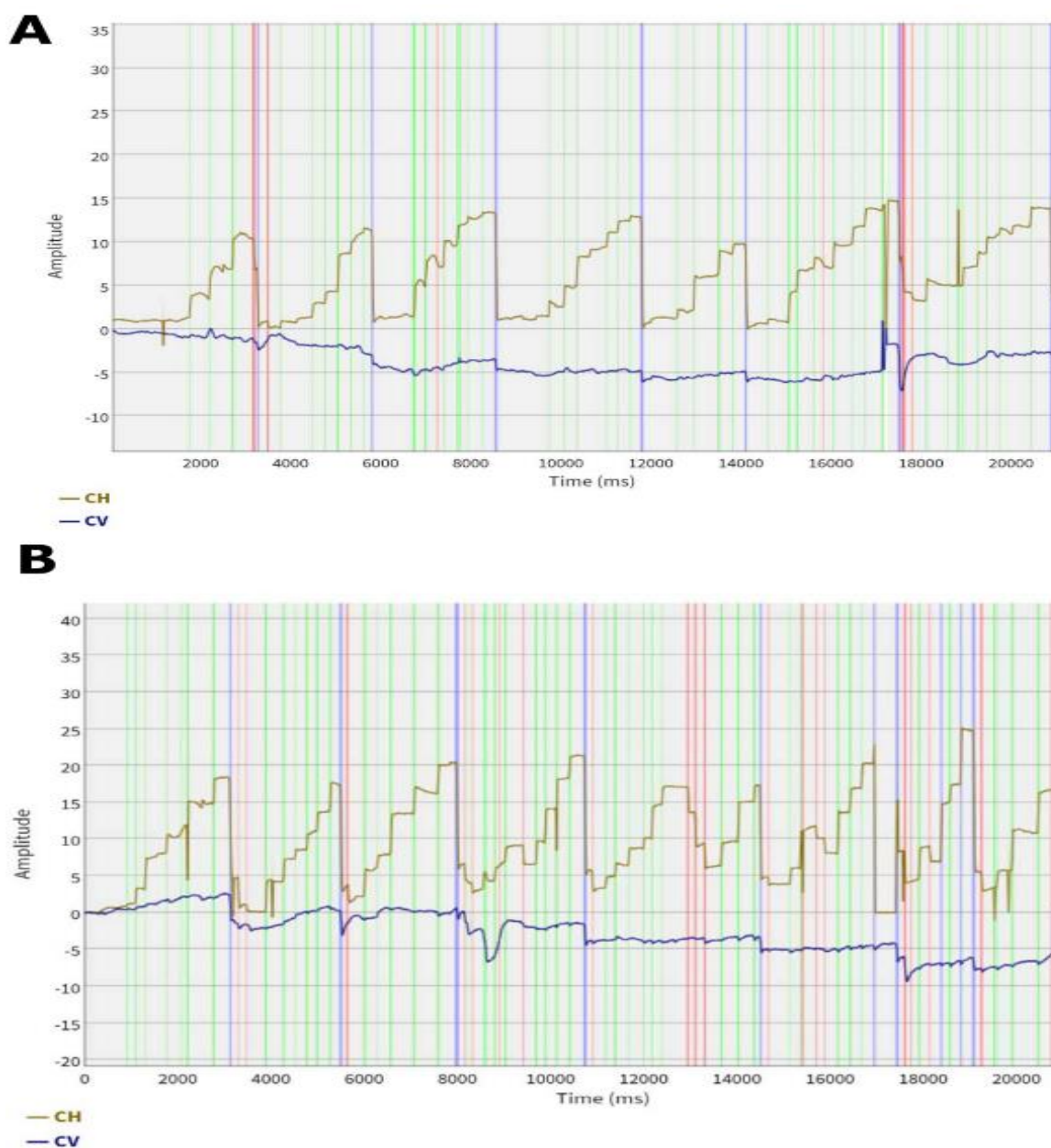


Рисунок 1.4.1 – График анализа движения глаз

Тем не менее, следует отметить, что на описанном этапе развития цифровых возможностей точность ручной обработки оставалась на более высоком уровне по сравнению с программными реализациями пороговых алгоритмов обработки данных. При ручной классификации событий один или несколько человек классифицируют необработанные данные о движении глаз на различные типы событий, основываясь на субъективных пороговых значениях. Ручная классификация по-прежнему является распространенным методом оценки алгоритмов обнаружения событий и рассматривается как "золотой стандарт". Классифицированные вручную данные часто используются в качестве обучающих данных для алгоритмов машинного обучения. Однако ручная классификация событий не является эффективным способом их классификации. Во-первых, это занимает много времени, а во-вторых, разные кодеры могут использовать разные правила субъективного отбора, которые дают разные результаты.

1.5 Обзор результатов предшествующих исследований, в которых применялись методы машинного обучения для анализа данных окулографии

Для лучшего понимания текущего состояния исследования предметной области, важно провести обзор результатов предшествующих исследований, в которых использовались методы машинного обучения для анализа данных окулографии. Этот обзор позволит выявить достижения, проблемы и тренды в данной области.

В данном разделе рассматриваются работы и исследования, проведенные другими исследователями и группами, с акцентом на применение методов машинного обучения для анализа данных окулографии. Анализируются как классические методы машинного обучения, такие как

метод опорных векторов, нейронные сети и деревья решений, так и более новые технологии, включая глубокое обучение и алгоритмы обучения с подкреплением.

Для каждой работы описывается контекст и цель исследования, используемые методы и алгоритмы машинного обучения, особенности используемых наборов данных, а также полученные результаты и их интерпретация. Кроме того, производится анализ преимуществ и недостатков каждого подхода, а также их применимости к конкретным задачам анализа окулографических данных.

Этот обзор поможет исследователю получить обширное представление о существующих методах и результатов в области анализа окулографических данных с применением машинного обучения, а также выделить перспективные направления для дальнейших исследований.

Ранее исследователи проводили обнаружение событий вручную, что отнимало много времени. Например, в работе [19] был разработан метод анализа движений глаз, который требовал 10 000 с (почти три часа времени анализа для 1 с записанных данных). Монти в [12] отмечает, что обычно приходится тратить дни на обработку данных, собранных всего за несколько минут.

В настоящее время существует множество работ, посвященных разработке алгоритмов обнаружения событий движения глаз. Производительность и адаптивность алгоритмов обнаружения событий зависит от различных факторов, включая тип стимула (статический или динамический), качество данных (данные могут быть зашумлены), устройство отслеживания движения глаз (частота выборки, бинокулярное или монокулярное, фиксированное или мобильное с жесткими или гибкими камерами для глаз). Эти различия затрудняют прямое сравнение методов и исследований. Уже есть несколько публикаций, посвященных сравнению алгоритмов обнаружения событий движения глаз. Одна из них - работа

Андерссона и других[22]. В этой работе авторы оценили и сравнили(рис.1.5.1) алгоритмы обнаружения событий движения глаз и порекомендовали будущим исследователям лучший метод.

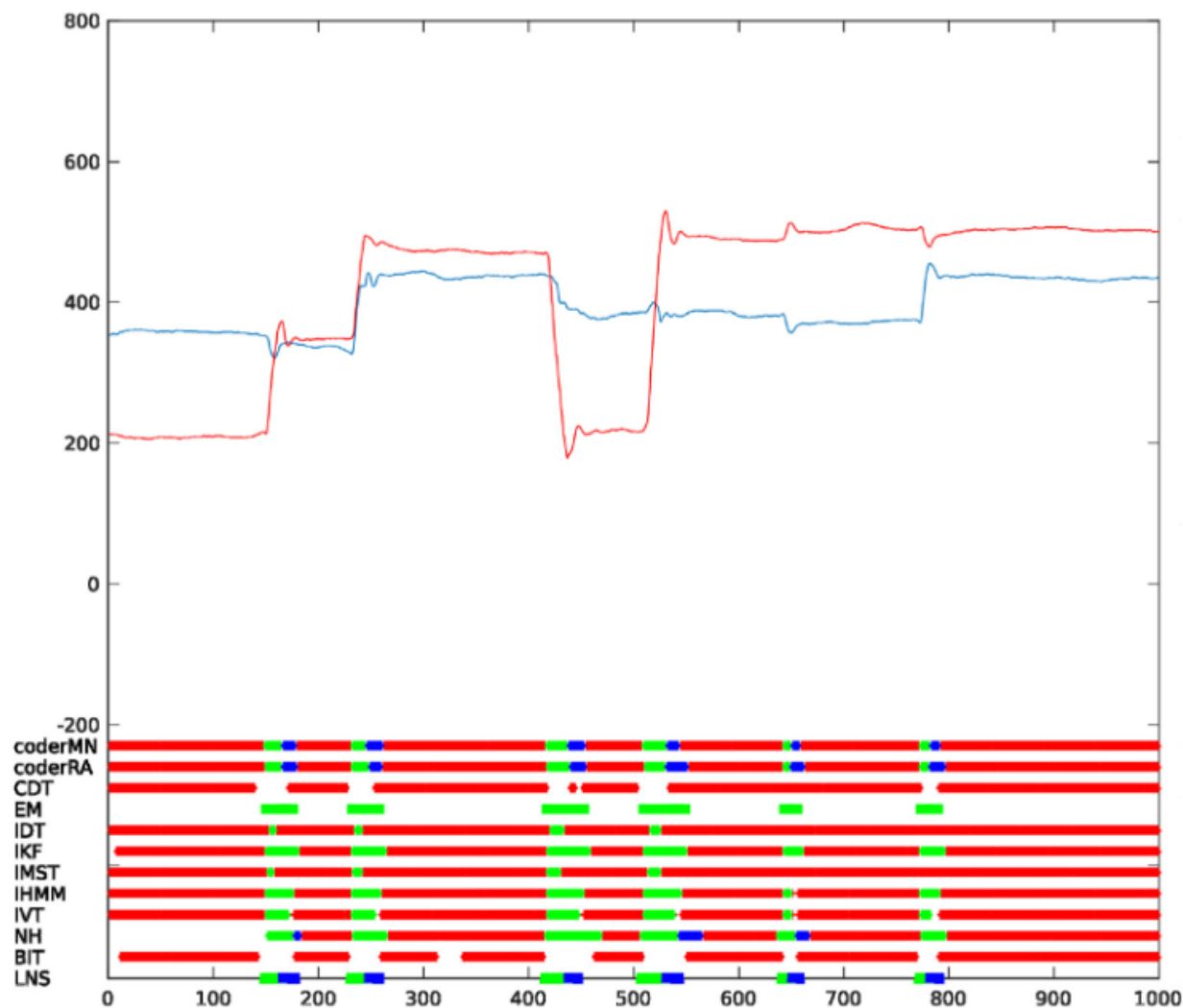


Рисунок 1.5.1 – Позиционные данные первых 1000 образцов эксперимента просмотра изображений. Координаты (x, y) отображаются как положение во времени, синим цветом обозначено положение с точки зрения испытуемых, а красным - с точки зрения алгоритмов. Ниже представлены графики-шарфы, иллюстрирующие классификацию результатов как кодировщиков, так и алгоритмов, где фиксации обозначены красным цветом, саккады - зеленым, PSO - синим. Белый цвет указывает на образцы, не классифицированные алгоритмом. Ось x подписана в образцах.

Как бы то ни было, все оцениваемые методы основаны на пороговых значениях. Разные методы обнаруживают разные типы событий. Например,

некоторые методы определяют только фиксацию и саккаду, некоторые - только фиксацию, а некоторые - фиксацию, саккаду и PSO. Из-за разницы в типах событий, которые определяют алгоритмы, сравнение алгоритмов, которые определяют одноклассовые, бинарные и мультиклассовые классификаторы событий, остается неясным, поскольку некоторые методы могут хорошо работать для классификации фиксации и саккад и плохо для других событий. Еще один обзор алгоритмов обнаружения событий провели Гонка и другие[9]. Они оценили десять алгоритмов обнаружения событий на основе пороговых значений с открытым исходным кодом.

1.6 Анализ моделей машинного обучения, применяемых для извлечения и классификации признаков из данных окулографии

В современном информационном обществе обработка и анализ данных играют ключевую роль в различных областях, включая медицину и психологию. Современные системы окулографии предоставляют огромные объемы данных, требующие эффективных методов анализа и интерпретации.

Основной недостаток всех методов обнаружения событий на основе пороговых значений заключается в том, что пользователь остается с рядом параметров, которые должны быть настроены в зависимости от качества данных о движении глаз. Поиск оптимальных пороговых значений является сложной задачей. Другой недостаток связан с тем, что пороговые методы разработаны для решения конкретной задачи одноэтапной классификации, например, фиксации и саккады. Классификация событий движения глаз с помощью машинного обучения решает эти проблемы [16]. Методы машинного обучения классифицируют необработанные данные отслеживания движения глаз по типам событий без ручной настройки параметров, вычисления и нахождения пороговых значений. Они учатся правильной классификации на основе некоторых обучающих данных.

Для достижения цели анализа окулографических данных и улучшения точности диагностики окуломоторной активности необходимо изучить существующие модели машинного обучения, используемые для обработки таких данных. В данном разделе проводится подробный обзор различных подходов к анализу окулографических данных с использованием методов машинного обучения, включая рассмотрение трех моделей: случайный лес (RF), сверточные (CNN) и рекуррентной (RNN) нейронные сети.

Каждая модель подвергается детальному анализу с учетом ее применимости к конкретным задачам извлечения и классификации признаков из окулографических данных. Рассматриваются преимущества, недостатки и области применения обеих моделей в контексте анализа окуломоторной активности. Также проводится сравнительный анализ моделей с целью выбора наиболее подходящей для решения поставленных задач.

Этот обзор моделей машинного обучения является ключевым шагом в разработке методики анализа окулографических данных и определении оптимального подхода к извлечению и классификации признаков, необходимых для точной диагностики окуломоторной активности. Данный анализ исследуемой системы позволит выявить наиболее перспективные подходы к анализу данных окулографии и способы их применения в практических задачах, что имеет важное значение для развития медицинской и научной областей, связанных с анализом глазодвигательных характеристик.

1.6.1 Классификация событий с помощью классификатора Random forest

Полностью автоматизированная классификация событий движения глаз с использованием RF-классификатора была впервые предложена в [34] для классификации фиксаций, саккад и постсаккадических колебаний. Результаты классификации сравнивались с современными алгоритмами и ручными

человеческими кодировщиками. В статье говорится, что алгоритм машинного обучения превосходит современные алгоритмы и почти достигает производительности ручных экспертов. Однако такая производительность была достигнута только для высококачественных данных (с низким уровнем шума).

Использование алгоритма классификации RF для анализа данных слежения за глазами продемонстрировано в [18]. В результате авторы показывают, что RF позволяет эффективно обнаруживать и классифицировать фиксации, саккады и PSO (псевдослучайные горизонтальные движения) с высокой точностью, что подтверждается результатами (рис.1.6.1) оценки эффективности классификации по различным метрикам.

В таблице 1.6.1 приведены результаты работы классификатора RF с точки зрения точности, прецизионности, запоминания и оценки F1 для каждого класса. Результаты показывают, что классификация событий PSO является наиболее сложной.

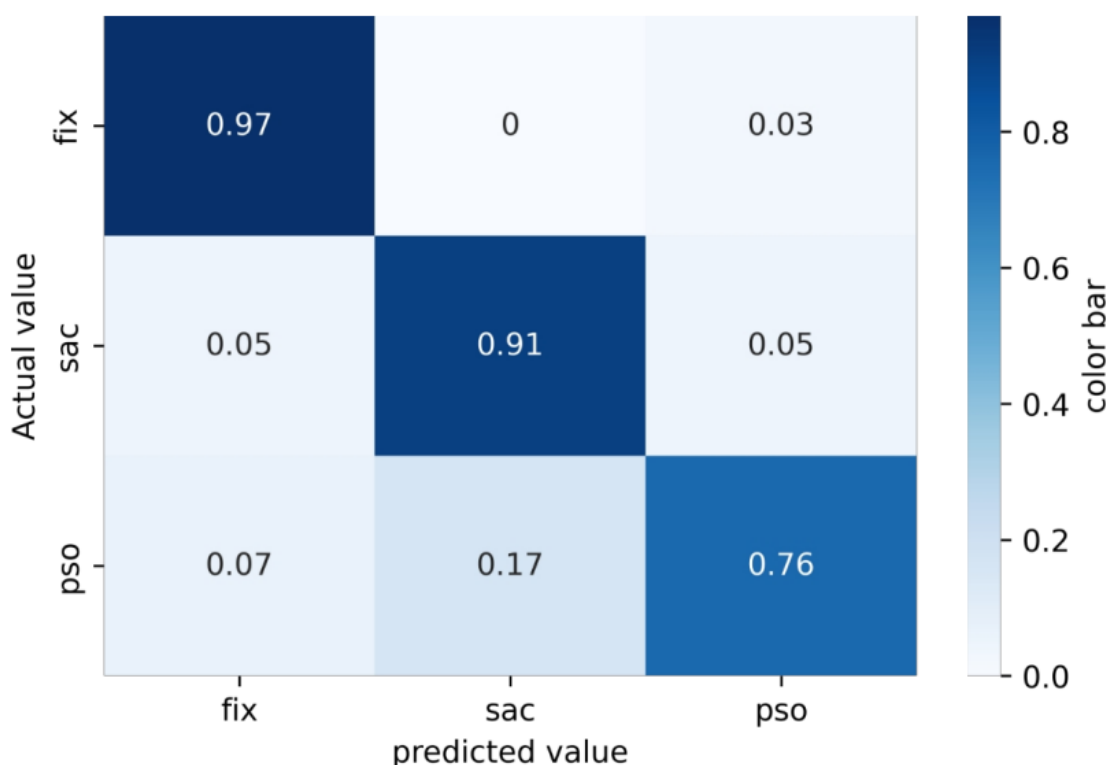


Рисунок 1.6.1 – Матрица ошибок RF

Таблица 1.6.1 - Эффективность классификации каждого класса с помощью RF-классификатора

Тип	Accuracy	Precision	Recall	F1-Score
Фиксация	97%	99%	97%	98%
Саккада	92%	87%	91%	89%
PSO	76%	64%	76%	69%

1.6.2 Классификация событий с помощью CNN

CNN хорошо справляются с поиском закономерностей в данных, поэтому их можно использовать для обнаружения событий движения глаз. Одним из примеров такого применения является метод, предложенный в [1], основанный на CNN, которая для каждого образца предсказывает последовательность вероятностей принадлежности к фиксации, саккаде или плавному преследованию из последовательности образцов взгляда. Метод пытается устранить недостаток предыдущих методов, которые используют форму и амплитуду сигнала для определения или классификации событий движения глаз, что может быть проблематично, например, для плавного преследования. Предлагаемый метод использует частоту сигнала для классификации данных на типы событий. Это означает, что сначала исходные данные о взгляде преобразуются в частотную область исходного сигнала с помощью быстрого преобразования Фурье (БПФ), а затем частотное представление сигнала передается в сеть CNN, которая, в свою очередь, выдает на выходе трехмерный сигнал активации. Каждый сигнал представляет собой вероятность каждого типа движения глаз (фиксация, саккада и SP). Наконец, метка с высокой вероятностью присваивается центральному образцу в окне.

Метод не является сквозным, поскольку на вход сети подается выходной сигнал БПФ. Он использует созданные вручную признаки - входные данные,

которые необходимо преобразовать в частоту области. Предложенный метод классифицирует фиксации, саккады и плавные преследования без учета других событий, как PSO. Метод превосходит старые алгоритмы, основанные на простой дисперсии и пороговом определении скорости.

Чтобы проверить способность сети CNN классифицировать события движения глаз, исследователи [18] создали простую сеть(рис.1.6.2). Сеть принимает на вход непрерывный поток двумерных образцов взгляда. Чтобы получить предсказание для каждого образца взгляда, окно перемещается по последовательности один за другим. Сначала мы преобразуем данные о координатах x и y в горизонтальные и вертикальные компоненты скорости, вычисляя скорость от образца к образцу. Чтобы получить соответствующие характеристики движения глаз, поток образцов взгляда анализируется в окнах по 100 образцов, что дает наилучшие результаты в наших экспериментах.

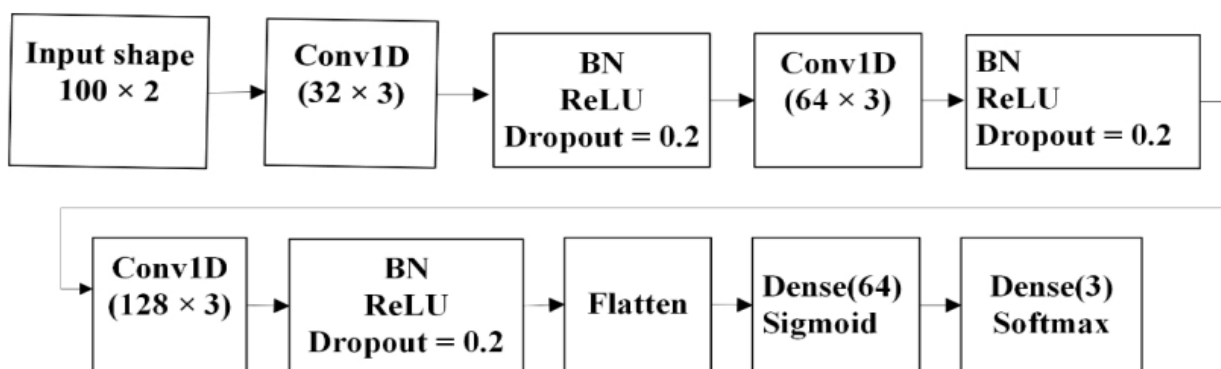


Рисунок 1.6.2 – Архитектура CNN, используемого в эксперименте

Результаты классификации в таблице 1.6.2 показывают, что CNN хорошо справляется с фиксациями и саккадами. классификация. Однако эффективность классификации для PSO далека от совершенства.

Таблица 1.6.2 – Эффективность классификации каждого класса с помощью классификатора CNN

Тип	Accuracy	Precision	Recall	F1-Score
Фиксация	99%	98%	99%	99%
Саккада	89%	93%	89%	91%
PSO	75%	83%	75%	79%

1.6.3 Классификация событий с помощью RNN (CNN с рекуррентным слоем)

Записи движений глаз представляют собой временные ряды, что позволяет использовать алгоритмы, эффективные для работы с такими данными, для классификации событий (рис. 1.6.3). Один из таких методов - применение рекуррентных нейронных сетей, как показано в работе [1]. В ней используется комбинация одномерной временной сверточной сети и слоя BLSTM для классификации фиксаций, саккад и плавного преследования. Для модели используются исходные координаты XY, скорость, направление и ускорение в качестве признаков. Однако данный метод показывает низкую производительность при определенных комбинациях параметров.

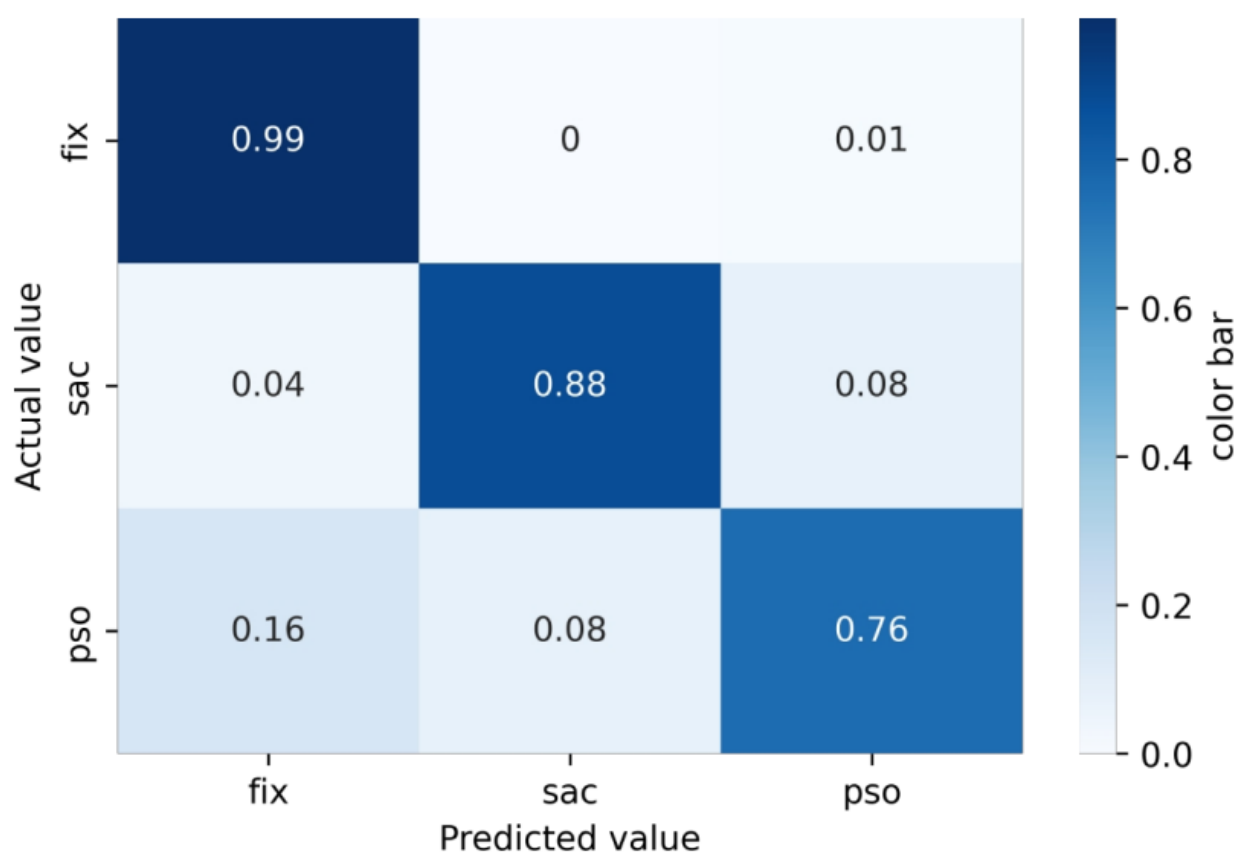


Рисунок 1.6.3 – Матрица ошибок CNN + BLSTM

В работе [18] отмечается, что алгоритм оценивался только на чистом и вручную размеченных данных, что не отражает реальные условия работы. Рекомендуется проверять алгоритм на необработанных данных движения глаз для более реалистичной оценки. Комбинирование направления и скорости значительно улучшило результаты классификации по сравнению с их отдельным использованием. Однако использование ускорения как дополнительного признака не привело к улучшению обнаружения, вероятно, из-за сложности различения плавных преследований от фиксаций.

1.7 Выявление разрыва между требованиями точной диагностики окуломоторной активности и имеющимися методами анализа данных окулографии

Проблема, выявленная в данном исследовании, состоит в разрыве между высокими требованиями к точности диагностики окуломоторной активности и ограничениями существующих методов анализа данных окулографии. Несмотря на значительные технологические достижения в области окулографии и машинного обучения, существующие методы все еще не всегда обеспечивают достаточную точность и надежность при диагностике окуломоторной активности у пациентов.

Одной из основных причин этого разрыва является сложность анализа окулографических данных, включающих в себя различные типы движений глаз, шумы и артефакты. Также существующие методы машинного обучения могут быть недостаточно адаптированы к особенностям данных окулографии, что приводит к недостаточной точности и обобщающей способности моделей.

Проводя сравнение описанных выше классических методов, основанных на пороговых значениях, и методов машинного обучения, исследователи [18] в таблице 1.7.1 привели метрики оценки производительности для каждого класса событий, а в столбцах - алгоритмы классификации. Результаты

показывают, что I-DT превосходит I-VT по всем метрикам оценки производительности. Однако алгоритмы RF и CNN превосходят пороговые алгоритмы (I-VT и I-DT) по всем показателям, за исключением показателя запоминания саккад. В случае классификационных моделей RF и CNN нет существенной разницы для классификации фиксаций и саккад. Однако CNN превосходит RF по показателям точности PSO, F1-score и каппа Козна. В таблице 1.7.2 приведены сильные и слабые стороны всех реализованных алгоритмов.

Таблица 1.7.1 – Сравнение эффективности классификации алгоритмов

Показатели метрик	IVT	IDT	RF	CNN	Coder MN[20]	Coder RA[20]
Fixation Accuracy	92%	95%	97%	99%	99%	99%
Saccade Accuracy	87%	93%	92%	89%	92%	96%
PSO Accuracy			76%	75%	88%	82%
Fixation F1-score	94%	96%	99%	99%	99%	99%
Saccade F1-score	60%	66%	87%	91%	94%	94%
PSO F1-Score			64%	79%	85%	85%
Fixation Recall	92%	95%	97%	99%	99%	99%
Saccade Recall	87%	93%	92%	89%	92%	96%
PSO Recall			76%	75%	88%	82%
Fixation Precision	96%	98%	99%	98%	99%	99%
Saccade Precision	46%	51%	87%	93%	96%	92%
PSO precision			64%	83%	82%	88%
Cochan's Kappa	0.5	0.6	0.83	0.88	1	0.90

Таблица 1.7.1 – Сравнение методов обнаружения событий

Метод	Сильные стороны	Слабые стороны
Human Coders[20]	Ручная разметка все еще является распространенным методом оценки алгоритмов обнаружения событий, а ручной классифицированные данные используются в качестве обучающих данных для алгоритмов машинного обучения.	Этот процесс требует много времени, различные кодеры могут использовать разные субъективные правила выбора, что приводит к различным результатам из-за того, что параметры и пороговые значения устанавливаются вручную кодером.
I-VT	Прост в реализации и понимании. Использует одно пороговое значение, которое является скоростью, для идентификации событий из исходных данных. Превосходно работает при идентификации фиксаций и саккад за один шаг идентификации. Требует низких вычислительных ресурсов.	Хотя I-VT прост в применении, он редко используется в реальных реализациях. Он чувствителен к шумным сигналам с большим количеством выбросов. Найти оптимальное пороговое значение сложно, так как не существует стандартного оптимального значения порога. I-VT идентифицирует только фиксации и саккады.
I-DT	Первый автоматизированный алгоритм обнаружения событий. Осуществляет идентификацию фиксаций и саккад с производительностью идентификации на уровне человека. I-DT часто доступен в коммерческом программном обеспечении.	Производительность зависит от выбора пороговых значений. Выбор метода расчета дисперсии представляет собой сложную задачу, поскольку различные методы расчета дисперсии влияют на ее значение. Разработан для идентификации только фиксаций и саккад.
RF	Для работы не требуется пороговое значение. Осуществляет многоклассовую классификацию, что позволяет использовать его для различных событий. Это полностью автоматизированный метод классификации событий. Осуществляет идентификацию фиксаций и саккад с производительностью на уровне человека.	Требует значительного объема правильно аннотированных данных для обучения. В нашей реализации для идентификации событий, таких как фиксации, саккады и PSO, использовались только признаки скорости. Результат классификации для PSO был обусловлен неправильной классификацией между саккадами и PSO из-за сходства их скоростей.
CNN	Подобно RF, CNN также решает проблемы метода обнаружения на основе порогов. Осуществляет одношаговое обнаружение без человеческого вмешательства. Демонстрирует производительность на уровне человека при идентификации фиксаций.	Требует для обучения еще больше правильно аннотированных данных, чем алгоритм RF. Мы использовали только параметры скорости для идентификации событий из входных данных. Плавное преследование не рассматривалось, потому что параметр скорости, который мы использовали, недостаточен для идентификации плавного преследования от фиксации, так как оба типа движения имеют низкую скорость. Для идентификации плавного преследования следует использовать другие параметры, такие как направление или образцы движения. CNN показал худшие результаты по сравнению с RF и I-DT при обнаружении саккад.

1.8 Предложение изменений в методологии анализа данных окулографии с целью повышения точности диагностики окуломоторной активности у пациентов

На основе обзора предшествующих исследований и выявленных проблем в существующих методах анализа окулографических данных, предлагается разработать и внедрить изменения в методологию анализа.

Целью этих изменений является повышение точности диагностики окуломоторной активности у пациентов для более эффективного лечения и управления заболеваниями, связанными с нарушениями окуломоторной функции. Предлагаемые изменения включают модификации существующих методов анализа данных окулографии с использованием новых алгоритмов машинного обучения и техник предобработки данных, а также разработку новых подходов к анализу и интерпретации окулографических данных.

В частности, возможны следующие направления изменений:

- Разработка и оптимизация алгоритмов для обработки окулографических данных с учетом их особенностей и шумов;
- Использование современных методов машинного обучения, таких как глубокое обучение, для извлечения и классификации признаков из данных;
- Интеграция информации из различных источников, таких как данные о движениях глаз и психофизиологические показатели, для улучшения точности диагностики;
- Разработка комплексных методологий анализа, учитывающих не только отдельные параметры окуломоторной активности, но и их взаимосвязь и динамику.

Эти изменения направлены на повышение точности диагностики окуломоторной активности и, следовательно, на улучшение результатов

лечения и управления заболеваниями, связанными с нарушениями окуломоторной функции.

Исследователи подчеркивают важность учета различных аспектов при использовании окулографических данных в моделях машинного обучения для качественного решения задач исследования и эффективного извлечения и классификации признаков. Среди этих аспектов — психологическое состояние испытуемого, внешние раздражители, рекурсивное исключение признаков, оценка важности признаков, сплайн-интерполяция и фильтрация, устранение шумов, уменьшение разреженности для уменьшения шума квантования.

Также важен процесс извлечения и расширения признакового пространства. Необходимо рассмотрение описательных статистик, параметров распределений, значений статистических тестов, временных параметров, спектральных параметров, значений главных компонент и коэффициентов авторегрессии.

2 Материалы и методы

2.1 Описание набора данных окулографии

В данной работе использовался набор данных окулограмм испытуемых прошедших тестирование на предмет наличия заболевания. Данный набор состоит из 185 папок, каждая из которых была именована уникальным идентификатором, соответствующим испытуемому. Данные окулограммы представлены в файлах формата txt. Каждый файл содержит записи окулограммы испытуемого читавшего текст, разбитый на 8 строк и состоящий из 10 предложений, и далее отвечавшего на серию вопросов, не связанные с текстом. Важно отметить, что в данном наборе данных окулограмм испытуемых отсутствовала разметка типов движений глаз, что потребовало процесса разметки с использованием пороговых алгоритмов или методов машинного обучения.

Выборка испытуемых представлена следующим образом: 97 человек с высоким(HR) риском (76 мужчин и 21 женщина) и 88 человек с низким(LR) риском (69 мужчин и 19 женщин).

Каждому испытуемому присвоен уникальный идентификатор, например, '112KA1', либо '111GM3'. Последняя цифра в идентификаторе может принимать значения от 1 до 4:

- 1 или 2 указывают на наличие заболевание;
- 3 или 4 обозначают контрольную группу;
- 1 или 3 представляют мужчины;
- 2 или 4 представляют женщин.

Данные в файле представлены следующими признаками:

- Т – временной признак с шагом 20 миллисекунд и диапазоном от 0 до 2000-4000(в зависимости от продолжительности испытания в конкретном случае);

- LX – горизонтальные координаты положения левого глаза;
- LY – вертикальные координаты положения левого глаза;
- RX – горизонтальные координаты положения правого глаза;
- RY – вертикальные координаты положения левого глаза.

Дополнительной информации по набору данных не было предоставляется.

2.2 Описание библиотек

В данной работе будут использоваться следующие библиотеки:

1. Pandas – Библиотека для работы с данными, предоставляющая структуры данных и функции для их анализа и манипуляций;
2. Numpy – Библиотека для выполнения вычислительных операций с массивами и матрицами, включая множество математических функций;
3. Matplotlib – Библиотека для создания статических графиков и визуализации данных в Python;
4. Seaborn – Более высокоуровневая библиотека для визуализации данных, которая упрощает создание информативных статистических графиков;
5. plotly.graph_objects – Библиотека для создания интерактивных графиков и визуализации данных, предоставляющая возможность создавать высококачественные визуализации для веб-приложений;
6. scipy.interpolate – Подмодуль библиотеки SciPy для интерполяции данных;
7. statsmodels – Библиотека для статистических моделей и тестов, предоставляющая инструменты для исследования данных и оценки моделей;

8. `scikit-learn` – Библиотека машинного обучения, предоставляющая инструменты для разработки и применения различных алгоритмов машинного обучения и анализа данных;

9. `sys` – Встроенная библиотека Python для взаимодействия с системой, включая функции для работы с интерпретатором Python и системными ресурсами;

10. `os` – Встроенная библиотека Python для взаимодействия с операционной системой, включая функции для работы с файлами и каталогами;

11. `Glob` – Встроенная библиотека Python для работы с путями к файлам и каталогам по шаблонам;

12. `Xgb` – XGBoost (Extreme Gradient Boosting) – это open-source библиотека на языке Python (также доступна на других языках, в том числе R), которая реализует алгоритм градиентного бустинга;

13. `Math` – Математическая библиотека в Python - это встроенный модуль, который предоставляет различные математические функции. Она включает в себя функции для основных математических операций, экспоненцирования, логарифмов, тригонометрических функций, гиперболических функций, специальных функций и констант.

Эти библиотеки предоставляют широкий спектр функций для анализа данных, визуализации, машинного обучения и статистического анализа, что позволяет эффективно проводить исследования и анализ данных.

2.3 Объединение, предобработка и анализ данных. Статистики и визуализация данных

Для проведения исследования был использован скрипт (Приложение A1) для объединения всех *.txt файлов в один набор данных. Использование данного скрипта также позволяет задать длину данных окулограмм, привести признаки к необходимому типу данных и расшифровать идентификаторы пациентов, а также добавит следующие признаки:

- length – длину выборки по испытуемому;
- X_mean – среднее значение по горизонтальной оси;
- Y_mean – среднее значение по вертикальной оси;
- gender – пол испытуемого;
- disabled – статус заболевания.

Далее признаки X_mean и Y_mean были использованы в процессе извлечения других признаков и визуализации данных на графиках.

Так же объединение позволило сохранить полученный набор данных в новый файл формата *.csv и использовать группировку по идентификатору пациента для последующих исследований с использованием различных алгоритмов и методов, реализованных в скриптах на языке Python.

Была введена возможность выбора набора данных с конкретной длиной и идентификатора пациента из выпадающего списка (Приложение A2) для ускорения процесса передачи индивидуальных данных испытуемых в различные скрипты это обеспечивало более удобный и быстрый переход от одного набора данных к другому, а также к индивидуальным данным испытуемых и визуализации окулограмм, что позволило более эффективно проводить анализ и интерпретацию результатов исследования.

В таблице 2.3.1 представлены статистики полученного общего набора данных из которой видно, что:

- Признаки T и Length вероятнее всего имеют значимое влияние на наборе данных;
- Признаки LX, RX, X_mean указывают, что по горизонтальной оси имеется какая-то вариативность.

Таблица 2.3.1 – Статистики общего набора данных испытуемых

index	T	LX	LY	RX	RY	X_mean	Y_mean	gender	disabled	length
count	323250.0	323250.0	323250.0	323250.0	323250.0	323250.0	323250.0	323250.0	323250.0	323250.0
mean	18046.84	38.26	-26.99	38.33	-26.98	38.3	-26.98	0.22	0.6	1805.68
std	10945.03	55.62	56.79	55.63	56.81	55.5	56.76	0.41	0.49	288.66
min	0.0	-67.5	-296.22	-77.33	-295.57	-72.42	-295.9	0.0	0.0	1000.0
25%	8720.0	-1.31	-49.81	-1.31	-49.15	-1.31	-49.48	0.0	0.0	1500.0
50%	17460.0	33.42	-16.38	33.42	-16.38	33.1	-16.06	0.0	1.0	2000.0
75%	26680.0	74.71	7.21	75.37	7.21	75.04	7.21	0.0	1.0	2000.0
max	39980.0	294.26	64.88	294.26	64.23	294.26	64.55	1.0	1.0	2000.0

При рассмотрении графиков окулограмм можно заметить, что имеются ярко выраженные отличия в поведении испытуемых из разных групп в процессе тестирования:

1. На рисунках 2.3.1 и 2.3.2 представлена амплитуда и частота движений глаз. На рисунке 2.3.1 видно, что испытуемый до временной отметки 20000 справился с чтением 8 строк (8 последовательных локальных максимумов по оси X) и в дальнейшем перешёл к процедуре ответов на вопросы. На рисунке 2.3.2 отсутствуют ярко выраженные переходы (отсутствие последовательных локальных максимумов по оси X) между строк, что может быть следствием, того, что испытуемый испытывает затруднения в процессе испытания;

2. На рисунке 2.3.3 видно, что испытуемый проходит взглядом по словам и строкам (горизонтальные линии), а после приступает к процедуре ответов на вопросы(вертикальные). На рисунке 2.3.4 наблюдается хаотичное поведение взгляда;

3. На рисунках 2.3.5 и 2.3.6 при наложении видно, что имеются выбросы в поведении глаз, выходящие за общий тренд;

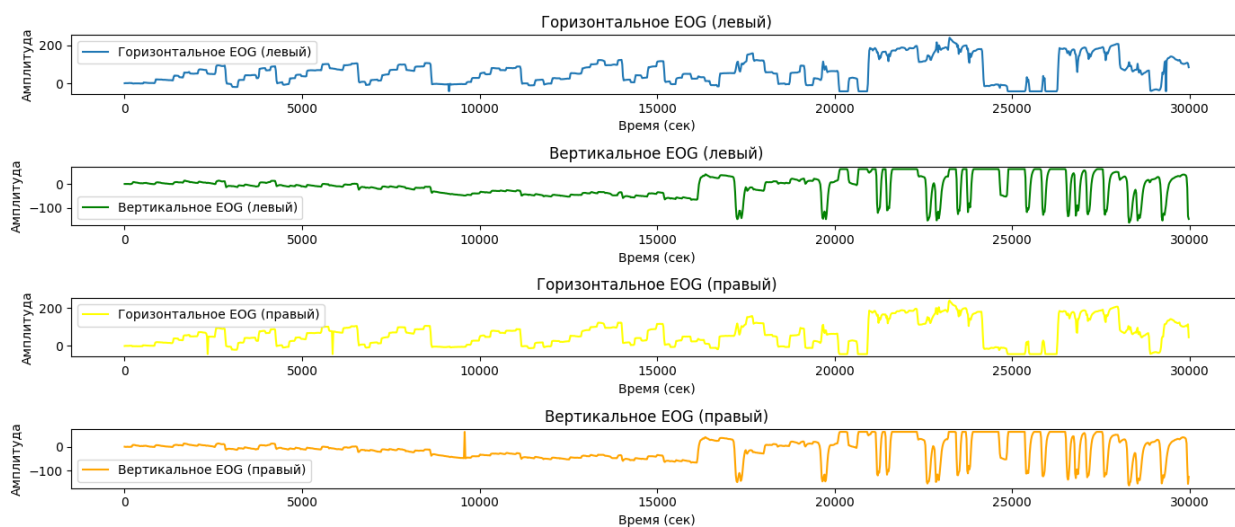


Рисунок 2.3.1 – График EOG признаков LX, LY, RX, RY в зависимости от признака T для испытуемого 111GM3

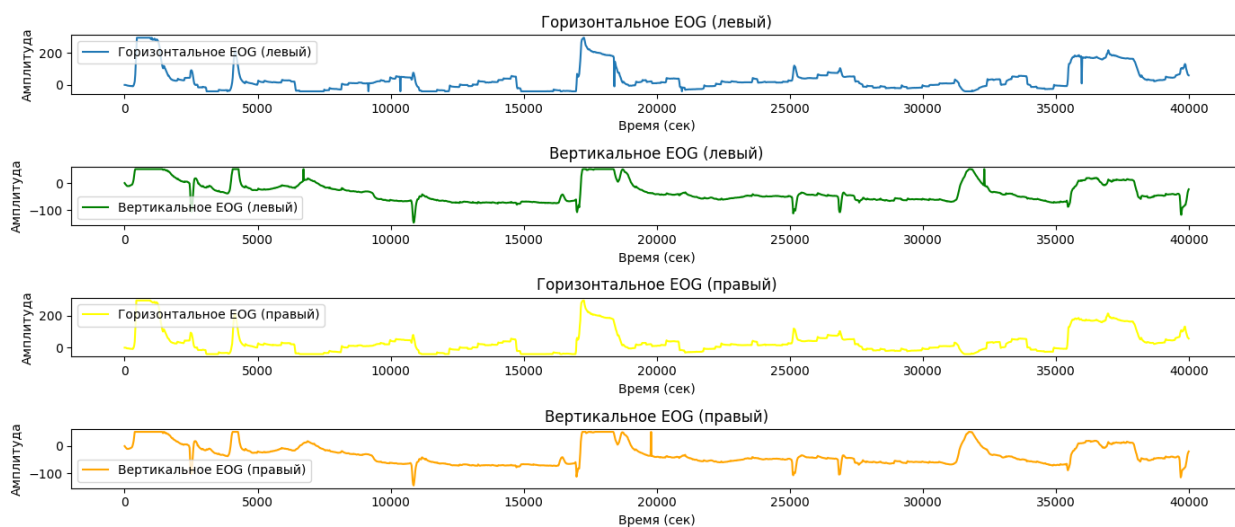


Рисунок 2.3.2 – График EOG признаков LX, LY, RX, RY в зависимости от признака T для испытуемого 141BE2

4. На рисунках 2.3.7 и 2.3.8 также заметно, что квантили для глаз не всегда совпадают, что может свидетельствовать о различиях в движении глаз у разных испытуемых. Кроме того, по обоим рисунка видно, что процедура чтения находится примерно в границах квантиля 25 и квантиля 75;

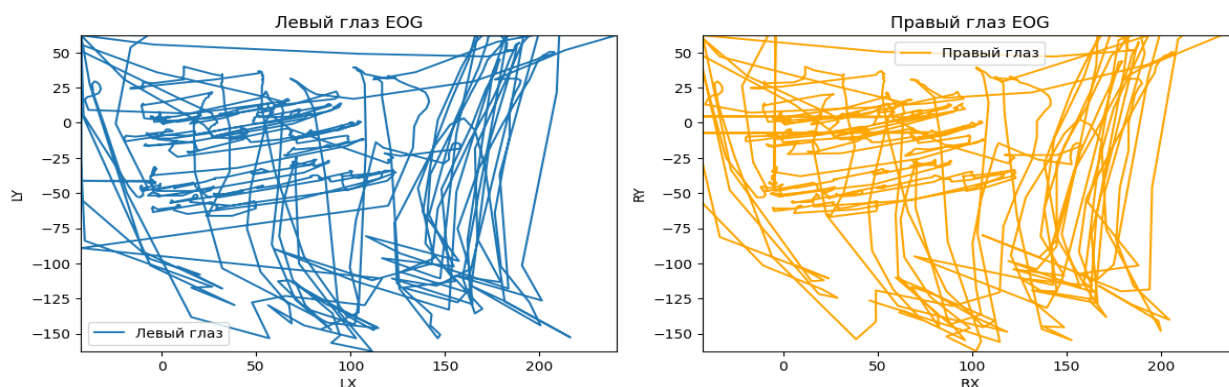


Рисунок 2.3.3 – График EOG зависимости LX от LY и зависимости RX от RY для испытуемого 111GM3

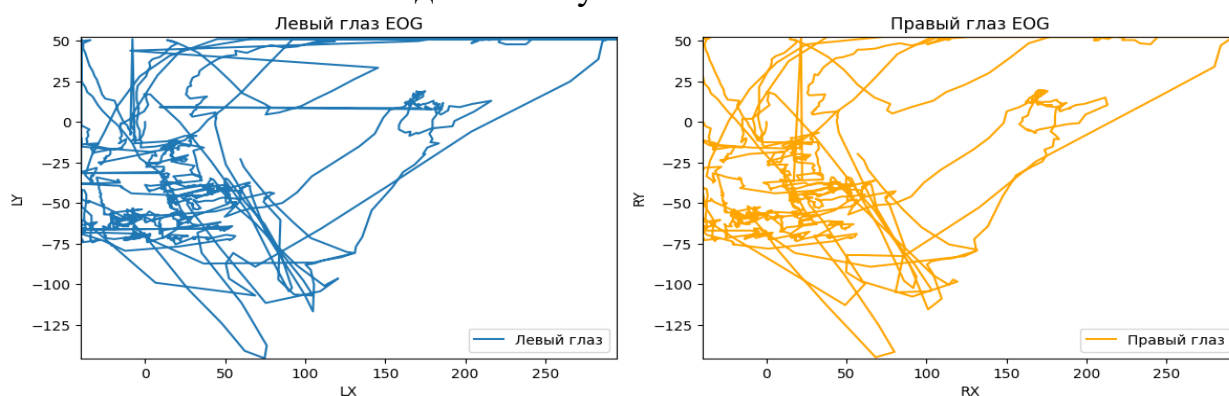


Рисунок 2.3.4 – График EOG зависимости LX от LY и зависимости RX от RY для испытуемого 141BE2

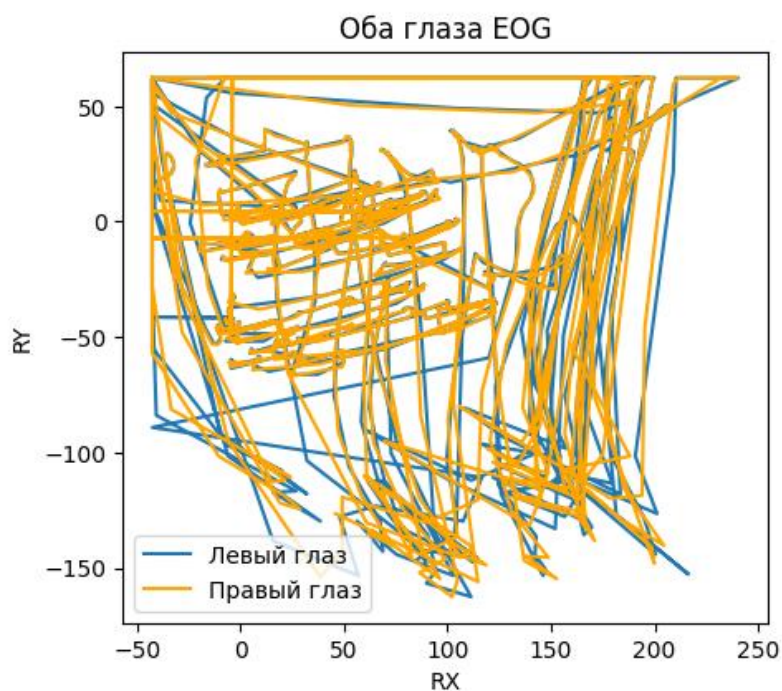


Рисунок 2.3.5 – График EOG зависимостей LX от LY и RX от RY с наложением друг на друга для испытуемого 111GM3

5. На рисунке 2.3.9 в трехмерном пространстве видно, что испытуемый фиксирует взгляд на словах и постепенно опускает взгляд на ниже лежащие строки. Из рисунка 2.3.10 не всегда ясно, что происходит с взглядом испытуемого во время тестирования.

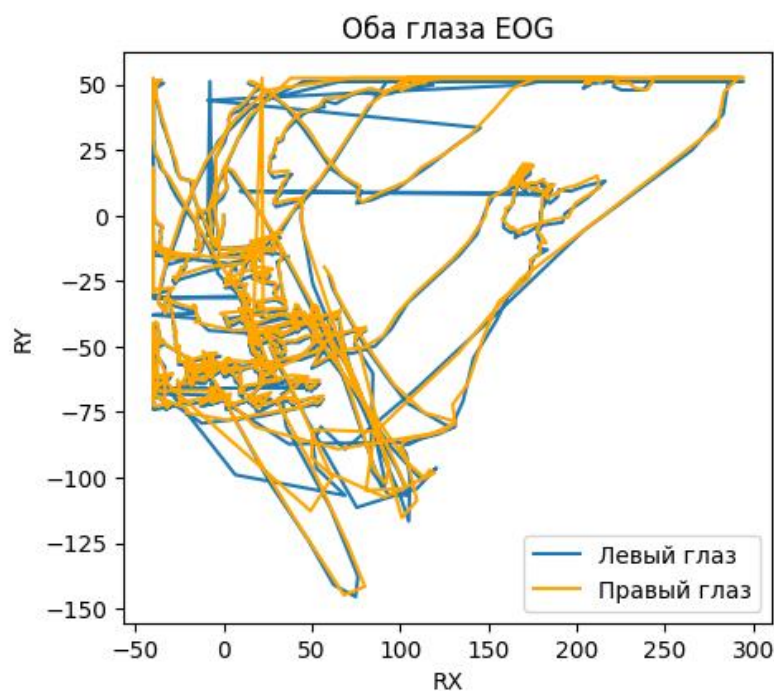


Рисунок 2.3.6 – График EOG зависимостей LX от LY и RX от RY с наложением друг на друга для испытуемого 141BE2

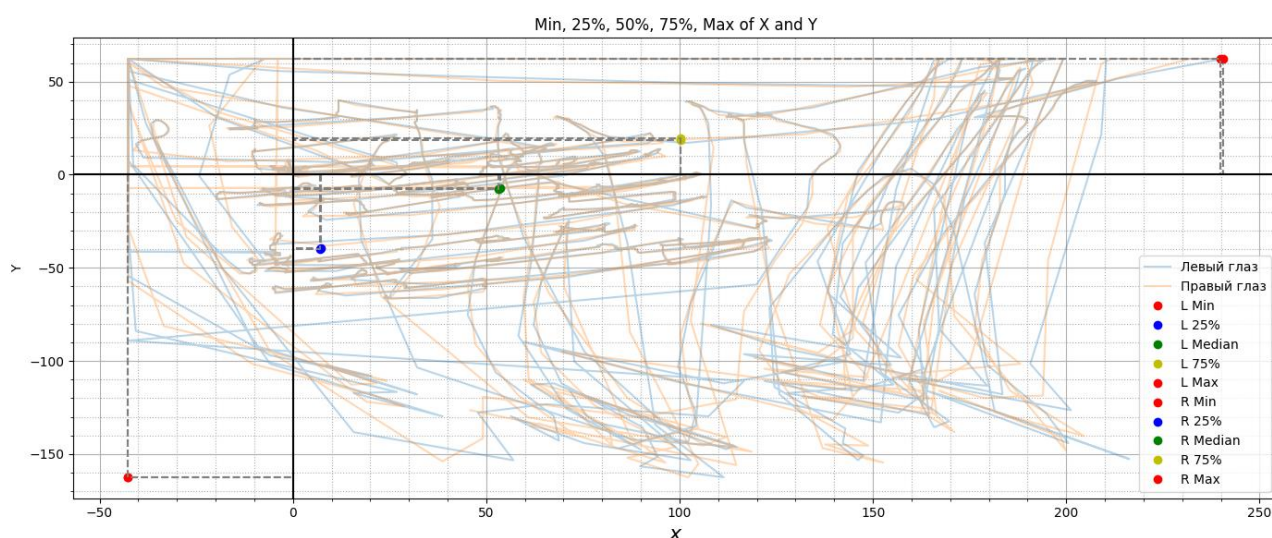


Рисунок 2.3.7 – График EOG зависимостей LX от LY и RX от RY с наложением друг на друга и отображением квантилей для испытуемого 111GM3

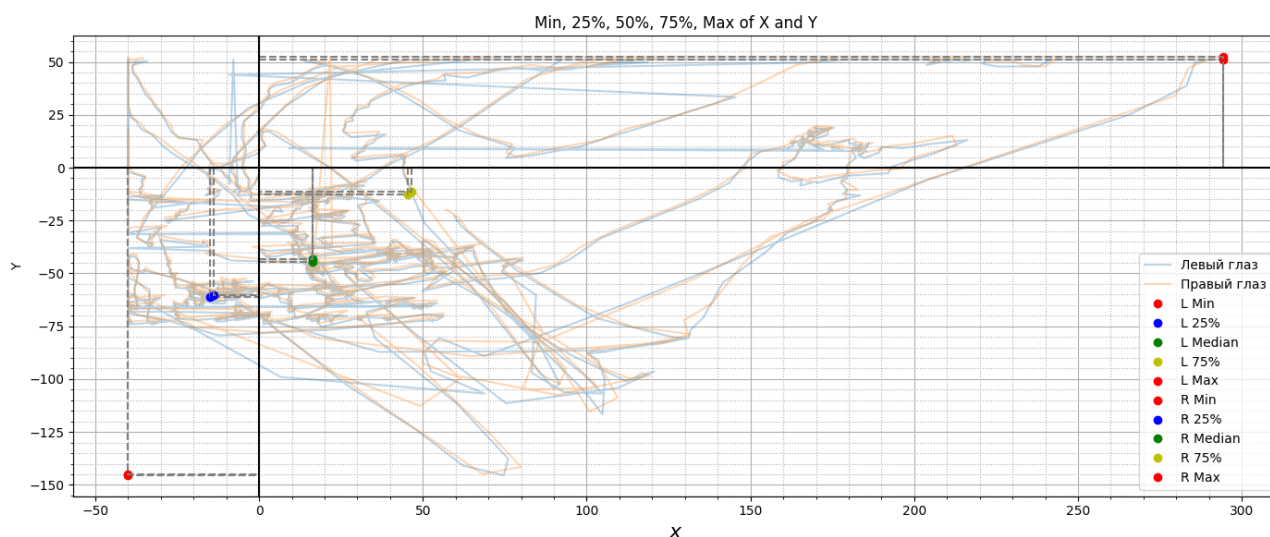


Рисунок 2.3.8 – График EOG зависимостей LX от LY и RX от RY с наложением друг на друга и отображением квантилей для испытуемого 141BE2

3D Visualization

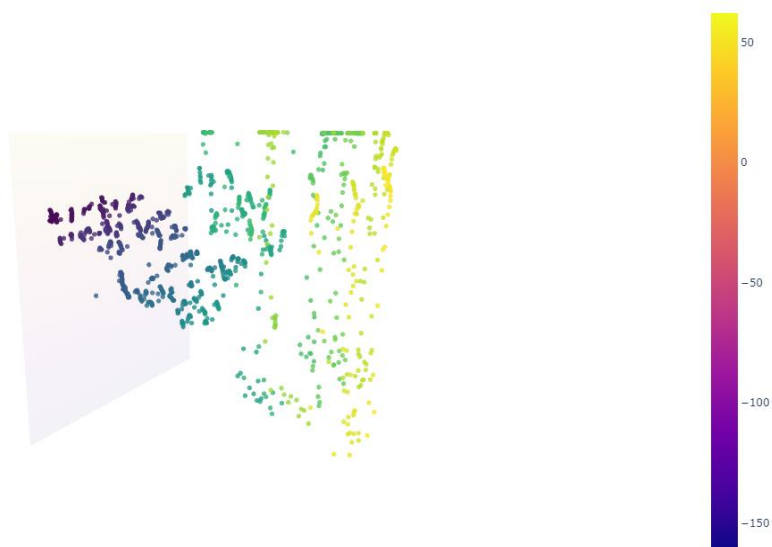


Рисунок 2.3.9 – График 3D визуализация осевых признаков в зависимости от признака T для испытуемого 111GM3

Важно отметить, что распределение (рис. 2.3.11) по времени прохождения испытания в данной работе является значимым фактором т.к. имеется явный перекос в сторону группы риска, что в дальнейшем скажется в работе классификатора.

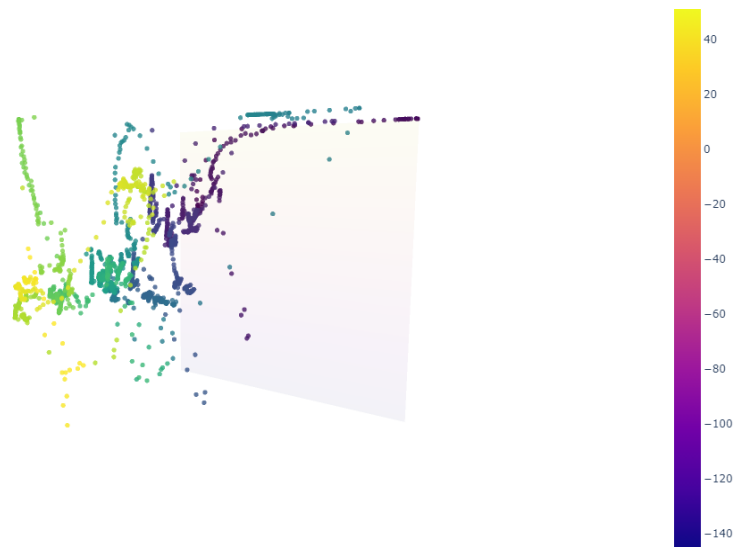


Рисунок 2.3.10 – График 3D визуализация осевых признаков в зависимости от признака T для испытуемого 141BE2

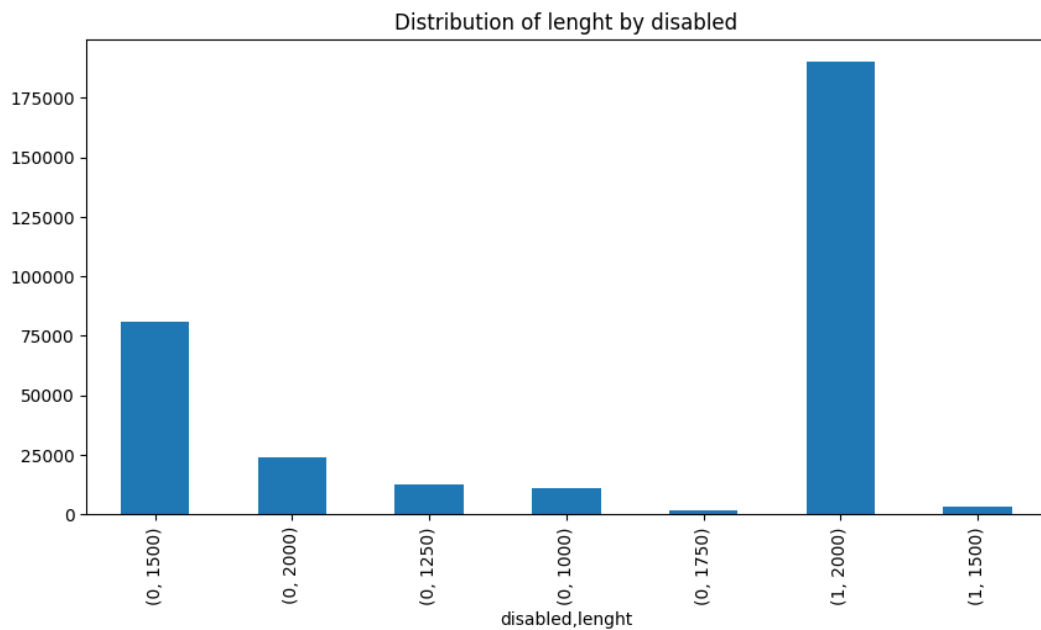


Рисунок 2.3.11 – График количественного распределение времени прохождения испытания в группах риска (LR – 0, HR - 1)

2.4 Извлечение признаков пороговыми методами

С целью выделения информативных характеристик в данном разделе рассматривается процесс извлечения признаков из окулограмм, с использованием различных пороговых методов, описанных в предыдущих разделах работы.

2.4.1 Метод IVT

В данной работе был использован метод IVT (Interval Variable Threshold) идентификации по порогу скорости. Псевдокод алгоритма представленный в таблице 1.4.2. Реализация метода описана в Приложении А8. Этот метод по формуле (2.8), рассчитывает скорость между парами точек и выявляет фиксации и саккады на основе порога скорости, определенного эмпирическим способом.

В результате работы метода IVT были получены признаки:

- Velocity(скорость) – вычисляется как евклидово расстояние между последовательными точками, разделенное на квадрат временного интервала;
- IVT – признак помечается как фиксация, если скорость находится ниже порога, и саккада в противном случае (рис. 2.4.1.1 и рис. 2.4.1.2).

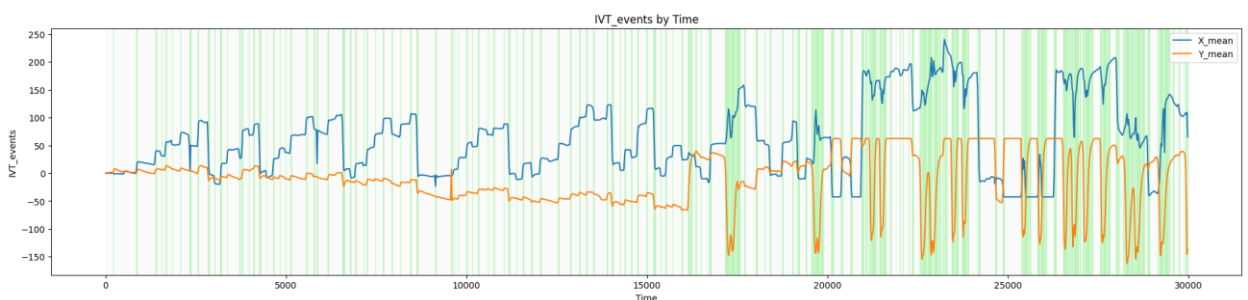


Рисунок 2.4.1.1 – График моментов глазодвигательной активности IVT для испытуемого 111GM3 (серые вертикальные линии – фиксации, зеленые вертикальные линии саккады)

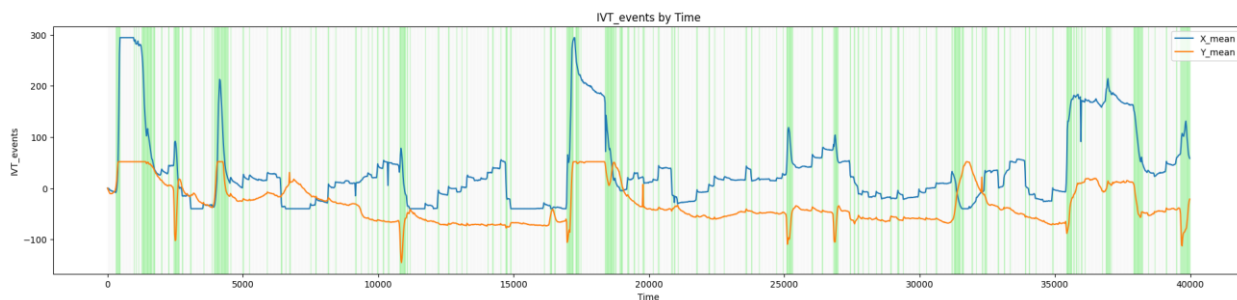


Рисунок 2.4.1.2 – График моментов глазодвигательной активности IVT для испытуемого 141BE2 (серые вертикальные линии – фиксации, зеленые вертикальные линии саккады)

Из приведенных графиков можно заключить, что в соответствии с [23] алгоритм достаточно хорошо определяет типы движения глаз, для которых он был разработан.

При анализе распределения, Рисунок Б2.4.1.3, было заметно, что имеется явный перекося в количестве фиксаций и саккад в пользу групп HR. Реализация графика распределения описана в приложении А21.

2.4.2 Метод IDT

В данной работе был использован метод IDT идентификации по порогу дисперсии. Псевдокод представлен в таблице 1.4.4. Реализация метода описана в Приложении А10. Этот метод вычисляет дисперсию для окна с заданным размером и определяет фиксации и саккады на основе порога дисперсии. Порог дисперсии и окно определяются эмпирическим способом.

По завершению работы метод был извлечен признак 'IDT'. Признак помечается как 'фиксация', если дисперсия находится ниже порога, и 'саккада' в противном случае.

Результаты работы метод IDT представлены с помощью графика моментов глазодвигательной активности (рис. 2.4.2.1 и рис. 2.4.2.2).

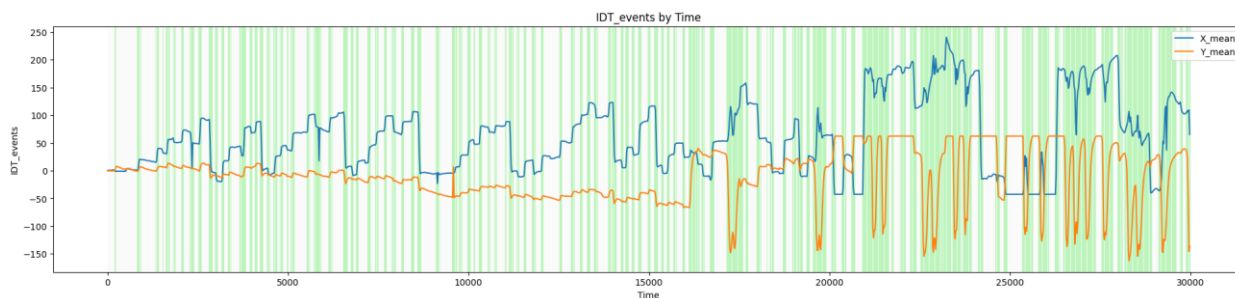


Рисунок 2.4.2.1 – График моментов глазодвигательной активности IDT для испытуемого 111GM3 (серые вертикальные линии – фиксации, зеленые вертикальные линии саккады)

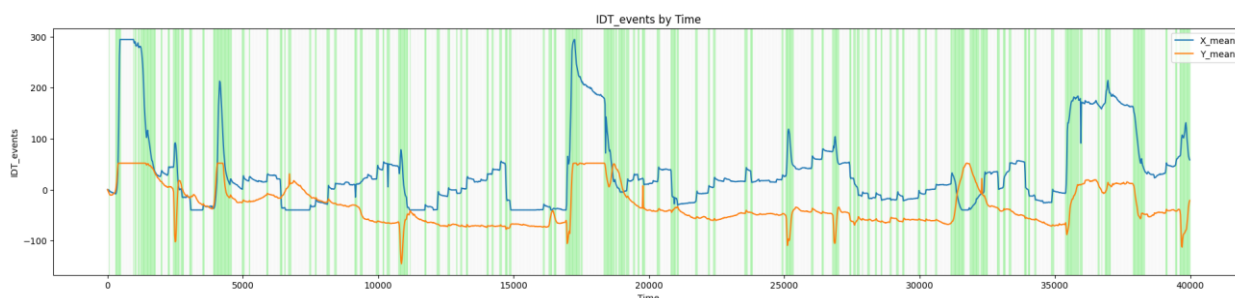


Рисунок 2.4.2.2 – График глазодвигательных моментов IDT для испытуемого из LR с идентификатором 141BE2 (серые вертикальные линии – фиксации, зеленые вертикальные линии саккады)

Из приведенных графиков можно заключить, что в соответствии с [23] алгоритм достаточно хорошо определяет типы движения глаз, для которых он был разработан.

При анализе распределения, Рисунок Б2.4.2.3, было заметно, что имеется явный перекося в количестве фиксаций и саккад в пользу групп HR.

2.4.3 Модифицированный метод IDT

В данной работе был использован метод модифицированный IDT за основу для написания которого была взята идея из работа по скринингу дислексии[24]. Реализация метода описана в Приложении А11-12. Данный метод на основании вычисляемого знака производной функции от значения оси в текущий момент времени и двух порогов дисперсии для окна с заданным размером, определяет две группы событий: прогрессивные и регрессивные

фиксации и саккады в прямом и обратном направлении. Дополнительно метод определяет и другие события, но ввиду малочисленности объединяет их под одним названием «other» (другие). В завершении работы метода производится сравнение прямой и обратной группы на наличии одинаковых событий в текущий момент времени. Порог дисперсии для прогрессивных и регрессивных событий и окно определяются эмпирическим способом.



Рисунок 2.4.3.1 – График моментов глазодвигательной активности модифицированного IDT для испытуемого 111GM3 (серые линии – фиксации прогрессивные, фиолетовые линии – фиксации регрессивные, зеленые линии – прогрессивные саккады, бордовый линии – саккады регрессивные, оранжевые линии – другие активности, белые линии – неизвестные средние активности)

В результаты работы модифицированного метода IDT для прогрессивные и регрессивные фиксации и саккад, а также других глазодвигательных активностей, были извлечены 3 признака:

- NE_event – со значениями прямого прохода;
- EN_event – со значениями обратного прохода;

- Mean_event – сравнение групп.

Полученные признаки, в соответствии с полученной последовательностью, представлены с помощью графиков моментов глазодвигательной активности (рис. 2.4.3.1 и рис. 2.4.3.2).



Рисунок 2.4.3.2 – График моментов глазодвигательной активности модифицированного IDT для испытуемого 141BE2 (серые линии – фиксации прогрессивные, фиолетовые линии – фиксации регрессивные, зеленые линии – прогрессивные саккады, бордовый линии – саккады регрессивные, оранжевые линии – другие активности, белые линии – неизвестные средние активности)

При анализе распределения, Рисунки Б2.4.3.3, Б2.4.3.4, Б2.4.3.5, было заметно, что имеется явный перекося в количестве прогрессивных и регрессивных фиксаций и саккад в пользу групп HR.

2.4.4 Метод идентификации транзитивных событий

В данной работе был использован метод выявления Транзитивных событий за основу для написания которого была взята идея из работа по скринингу дислексии[24]. Реализация метода описана в Приложении А13. Данный метод для каждой точки в данных создает окно с заданным размером рассчитывая среднеквадратичную ошибку, формула (2.6). Затем проверяется расстояние между текущим значением и средним значением окна, и сравнивается с порогом расстояния и среднеквадратичной ошибкой RMS. Результат сохраняется в новом признаке 'TRANS'. Параметрами метода:

- windows_size_rms(размер скользящего окна) = 25;
- distance_threshold(порог дистанции) = 0.5;
- rms_threshold_factor(порог среднеквадратичной ошибки) = 2.5.

Результаты работы метода выявления Транзитивных событий представлены в графиках моментов глазодвигательной активности (рис. 2.4.4.1 и рис. 2.4.4.2).

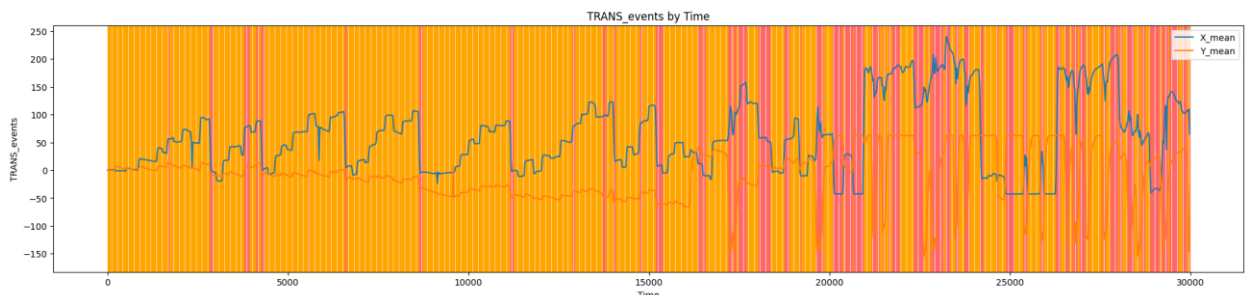


Рисунок 2.4.4.1 – График моментов глазодвигательной активности Транзитивных событий для испытуемого 111GM3 (оранжевые линии – другие активности, красные линии – транзитивные события)

При анализе распределения, Рисунки Б2.4.4.3, было заметно, что имеется явный перекося в количестве транзитивных событий в пользу HR.

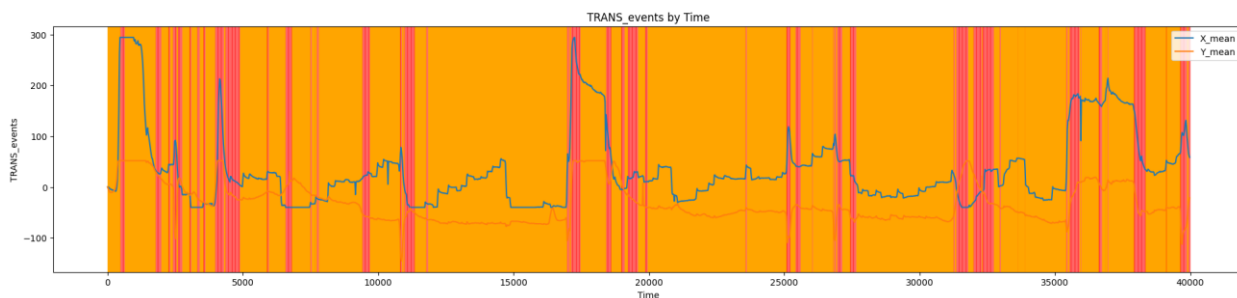


Рисунок 2.4.4.2 – График моментов глазодвигательной активности Транзитивных событий для испытуемого 141BE2 (оранжевые линии – другие активности, красные линии – транзитивные события)

2.5 Извлечение признаков методами машинного обучения

В данной разделе будет описан процесс использования методов машинного обучения, применяемых к наборам окулограмм испытуемых в качестве входных данных. Цель состояла в том, чтобы опробовать методы МО в кластеризации неразмеченных данных окулограмм для выявления новых признаков фиксации, саккад и по возможности других глазодвигательных событий. Для достижения этой цели были использованы методы выявляющие аномалии.

2.5.1 Метод DBSCAN

В качестве первичного метода машинного обучения был применен метод кластеризации DBSCAN (Density-Based Spatial Clustering of Applications with Noise), который позволяет выделить данные на основе их пространственной близости и плотности. Этот метод является оптимальным для поиска кластеров различной формы и размера, а также имеет важную возможность поиска аномалий. Такие особенности делают метод DBSCAN подходящим для анализа окулографических данных в трехмерном пространстве и интерпретации данных об аномалиях как глазодвигательных событий.

Реализация метода была осуществлена через функцию и описана в Приложении 14А. В качестве входных данных функция принимает:

- Данные окулограммы испытуемого;
- Список осей, по которым происходит кластеризация (Т, X_mean, Y_Mean);
- Префикс для типа поиска аномалий (например, взгляд в точку);
- Конкретная Лямбда-функция для обработки полученных аномалий;
- Радиус окрестности центроида;
- Минимальное количество точек для формирования кластера.

Функция преобразует данные в двумерный массив без временной метки и применяет алгоритм DBSCAN для кластеризации. Затем результаты кластеризации сохраняются в исходном наборе данных в виде нового столбца, указывающего принадлежность к заданному в префиксе типу событий. Наконец, функция применяет лямбда-функцию к результатам кластеризации для дополнительной обработки. Важно обратить внимание, что работа метода очень сильно зависит от шага временной метки, например, в данном наборе данных шаг равен 20 ед., поэтому эpsilon должен быть в интервале $[20, \max(T)]$, в одном из частных случаев для поиска событий типа «фиксирование взгляда в точку» в работе был взят эpsilon равный 20.000000001.

В результаты работы трех последовательных запусков функции с различными параметрами были извлечены три признака, со значениями other или event:

- DBSCAN_P – взгляд в точку;
- DBSCAN_FS – наличие саккад;
- DBSCAN_T – наличие транзитивных событий.

Полученные признаки, представлены с помощью графиков моментов глазодвигательной активности (рис. 2.5.1.1 и рис. 2.5.1.2).



Рисунок 2.5.1.1 – График глазодвигательных моментов DBSCAN точки для испытуемого из LR с идентификатором 111GM3 (красные линии – заданные события, оранжевые линии – другие активности)



Рисунок 2.5.1.2 – График глазодвигательных моментов IDT для испытуемого из LR с идентификатором 141BE2 (красные линии – заданные события, оранжевые линии – другие активности)

При рассмотрении графиков можно заметить, что результаты работы метода DBSCAN схожи с результатами работы пороговых алгоритмов IVT, IDT и алгоритмом поиска транзитивных событий.

Так же результат кластеризации представлен на рисунках Б2.5.1.3, Б2.5.1.4, Б2.5.1.5. для испытуемого 111GM3 и на рисунках Б2.5.1.6, Б2.5.1.7, Б2.5.1.8. для испытуемого 141BE2.

При анализе распределения, Рисунки Б2.5.1.9, Б2.5.1.10, Б2.5.1.11 было заметно, что имеется явный перекося в количестве глазодвигательных событий в пользу групп HR.

2.5.2 Метод HDBSCAN

После метода DBSCAN был опробован его расширенный метод кластеризации HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) позволяющий выявлять кластеры различной формы и размера, объединять близкие кластеры в одну группу и создавать иерархическую структуру кластеров. Метод использовался для поиска аномалий, которые можно было бы интерпретировать как глазодвигательные события.

Реализация метода была осуществлена через функцию и описана в Приложении 16А. В качестве входных данных функция принимает:

- Данные окулограммы испытуемого;
- Список осей, по которым происходит кластеризация (Т, X_mean, Y_Mean);
- Эпсилон – максимальное расстояние для объединения точек в один кластер;
- Мин_семпел – минимальное количество точек, необходимое для формирования кластера.

Функция преобразует данные в двумерный массив без временной метки и применяет алгоритм HDBSCAN для кластеризации. Затем результаты кластеризации сохраняются в исходном наборе данных в виде нового столбца, указывающего принадлежность к глазодвигательному событию или другим событиям.

Результаты работы алгоритма HDBSCAN представлены с помощью графика моментов глазодвигательной активности (рис. 2.5.2.1 и рис. 2.5.2.2).

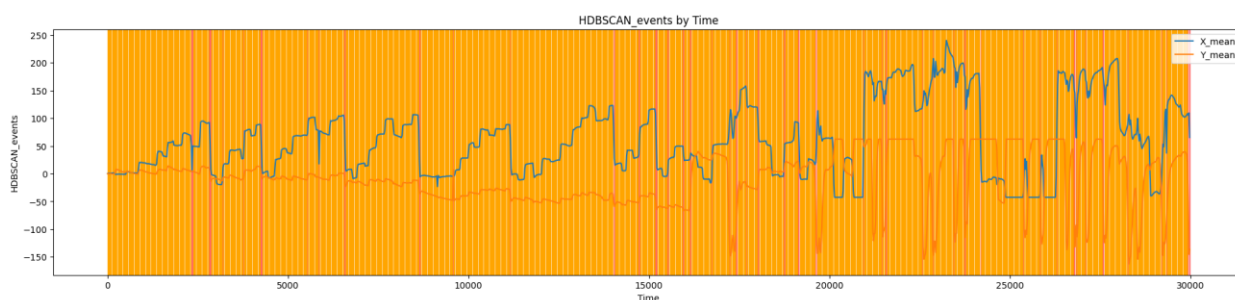


Рисунок 2.5.2.1 – График моментов глазодвигательной активности HDBSCAN для испытуемого 111GM3 (оранжевые линии – другие активности, красные линии – транзитивные события)

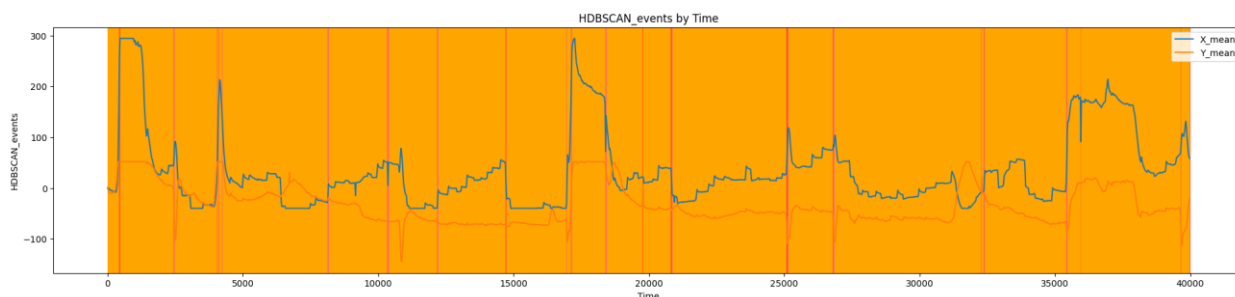


Рисунок 2.5.2.2 – График моментов глазодвигательной активности HDBSCAN для испытуемого 141BE2 (оранжевые линии – другие активности, красные линии – транзитивные события)

При рассмотрении графиков можно заметить, что результаты работы метода HDBSCAN схожи с результатами работы DBSCAN_P.

Так же результат кластеризации представлен на рисунке Б2.5.2.3, для испытуемого 111GM3 и на рисунке Б2.5.2.4, для испытуемого 141BE2.

При анализе распределения, Рисунок Б2.5.2.5 было заметно, что имеется явный перекос в количестве HDBSCAN событий в пользу групп HR.

2.5.3 Метод IOF

Далее для поиска глазодвигательных моментов был опробован метод LOF (Local Outlier Factor). Т.к. этот метод позволяет выявлять выбросы (аномалии) в данных на основе локальной плотности то этот метод был полезен для выявления выбросов, которые были интерпретированы как транзитивные события.

Реализация метода была осуществлена через функцию и описана в Приложении 17А. В качестве входных данных функция принимает:

- Данные окулограммы испытуемого;
- Список осей, по которым происходит кластеризация (Т, X_mean, Y_mean);
- contamination - долю выбросов в данных.

Функция преобразует данные в двумерный массив без временной метки и применяет алгоритм LOF для выявления выбросов. Затем результаты выявления выбросов сохраняются в виде нового столбца в исходном наборе данных.

Результаты работы алгоритма LOF представлены с помощью графика моментов глазодвигательной активности (рис. 2.5.3.1 и рис. 2.5.3.2).

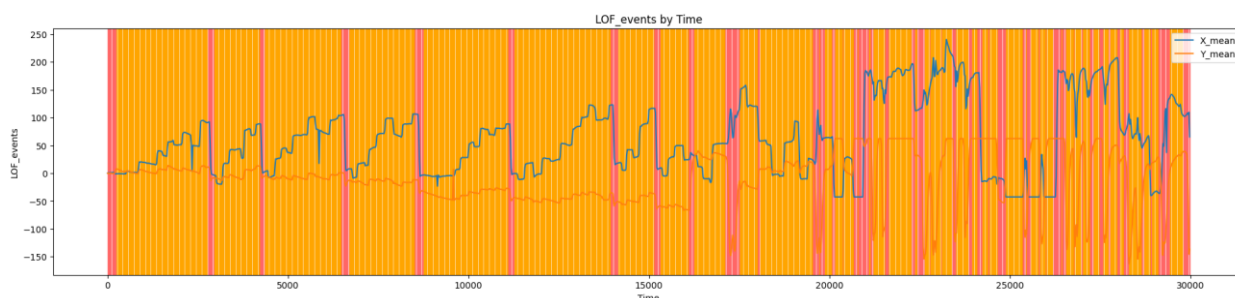


Рисунок 2.5.3.1 – График глазодвигательных моментов LOF для испытуемого из LR с идентификатором 111GM3

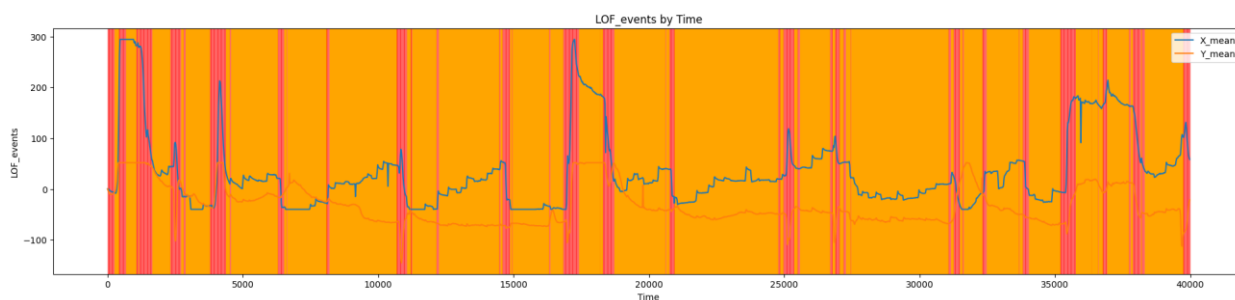


Рисунок 2.5.3.2 – График глазодвигательных моментов LOF для испытуемого из LR с идентификатором 141BE2

Так же результат кластеризации представлен на рисунке Б2.5.3.3, для испытуемого 111GM3 и на рисунке Б2.5.3.4, для испытуемого 141BE2.

При анализе распределения, Рисунок Б2.5.3.5 было заметно, что имеется явный перекося в количестве LOF событий в пользу групп HR.

2.6 Другие подходы для извлечения признаков

Кроме выше упомянутых пороговых алгоритмов и методов МО, в работе был использован и другие подходы, позволившие расширить признаковое пространство.

2.6.1 Метод выявления типов полей зрения

В работе был использован метод выявления типов полей зрения (фовеальное, парафовеальное, периферическое или иное) для каждой точки в данных окулографии. Основой идеей для написания метод является упоминание о полях зрения(рис. 2.6.1.1) в работе[8].

За идею было взято, что человек читая текст скользит глазами по нему, что очень похоже не скользящее окно. В процессе скольжения глаза у испытуемого подвержены случайным колебаниям в определенной зоне, другими словами в окружности с заданным радиус или нескольких зон, другими словами нескольких окружностей с заданными радиусам. Цепочка

данных умозаключений, привела к идее реализации метод вычисления полей зрения.

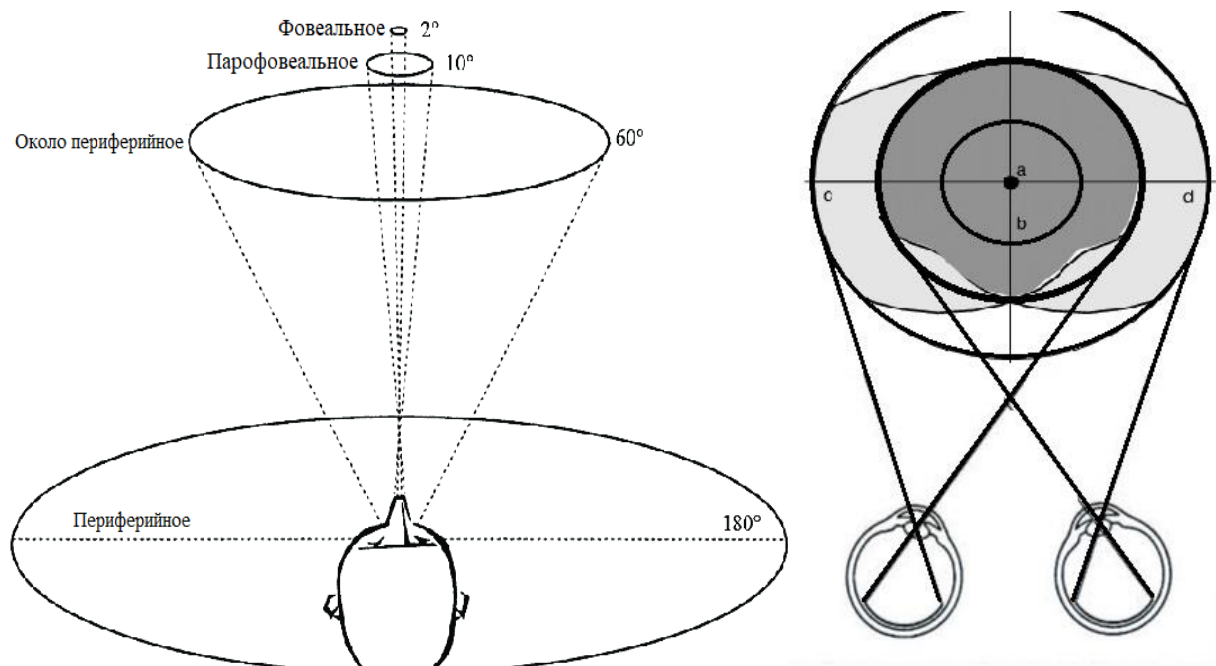


Рисунок 2.6.1.1 – Поля зрения человека

Реализация метода была осуществлена через функцию и описана в Приложении A18. В качестве входных данных функция принимает:

Параметры:

- Набор данных окулограм;
- Размер окна для вычисления средних значений координат точек (по умолчанию 1);
- Радиус фовеального поля зрения (по умолчанию 0.02);
- Радиус около парафовеального поля зрения (по умолчанию 0.1);
- Радиус около периферийного поля зрения (по умолчанию 0.6).

Возвращает набор данных с добавленными столбцами X_mean_FV , Y_mean_FV и FV , где X_mean_FV и Y_mean_FV содержат средние значения координат точек в окне, а FV содержит тип поля зрения для каждой точки.

Функция вычисляет средние значения координат точек в окне заданного размера, рассчитывает расстояние между средними координатами и

координатами текущей точки по формуле (2.1), и вычисляет тип поля зрения на основе заданных радиусов фовеального, паравовеального и периферического полей зрения.

Результаты работы алгоритма выявления типов полей взгляда представлены в графиках моментов глазодвигательной активности (рис. 2.6.1.2 и рис. 2.6.1.3). Реализация графиков в Приложении А19.

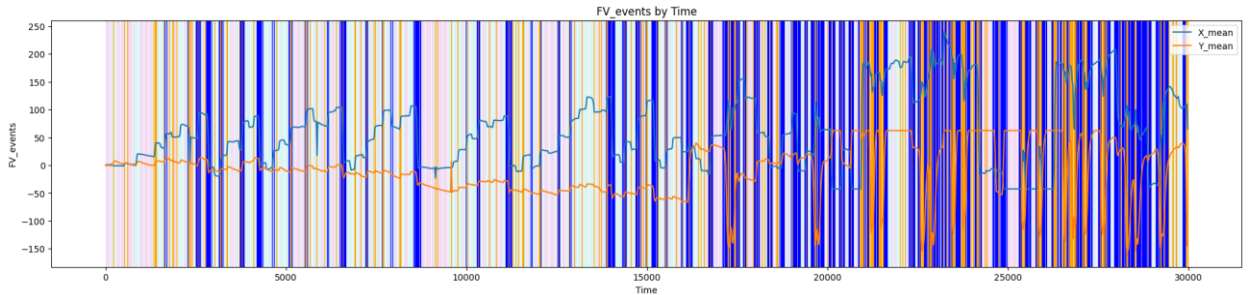


Рисунок 2.6.1.2 – График глазодвигательных моментов полей зрения испытуемого из LR с идентификатором 111GM3 (фиолетовые линии – фовеальные поля, голубые линии – паравовеальное поля, синие линии – около периферийное поля, красные линии – периферийное поля)

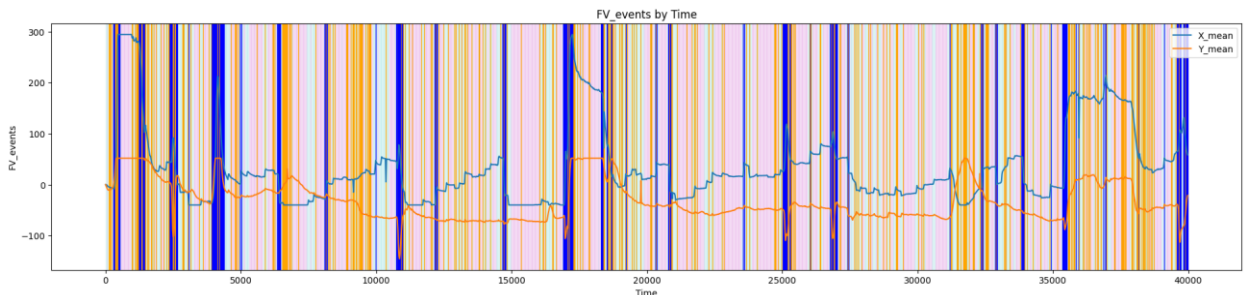


Рисунок 2.6.1.3 – График глазодвигательных моментов полей зрения для испытуемого из LR с идентификатором 141BE2 (фиолетовые линии – фовеальные поля, голубые линии – паравовеальное поля, синие линии – около периферийное поля, красные линии – периферийное поля)

$$distance = \sqrt{(x_{mean} - x)^2 + (y_{mean} - y)^2} \quad (2.1)$$

где x_{mean} – среднее по оси X заданного окна;

y_{mean} – среднее по оси Y заданного окна;

x – текущая координата левого, либо правого глаза;

y – текущая координата левого, либо правого глаза.

При проведении анализа распределения, Рисунок Б2.6.1.4, стало очевидно, что количество фовеальных и парофовеальных типов полей было значительно выше у представителей группы ЛР, что можно объяснить сильной концентрацией взгляда на словах во время чтения в связи с трудностями восприятия.

2.6.2 Математические признаки

В работе были также извлечены математические признаки к окулографическим данным при помощи реализованной функции, представленной в Приложении А20.

Список выделенных признаков:

- абсолютную разницу между координатами левого и правого глаза, формула (2.2);

$$\Delta r = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \quad (2.2)$$

где x_i – текущая координата левого, либо правого глаза;

y_i – текущая координата левого, либо правого глаза.

- скользящее среднее по каждой оси, формула (2.3);

$$\mu = \frac{(x_n + x_{n-1} + \dots + x_{n-n+1})}{n} \quad (2.3)$$

где x_n – значение оси в окне;

n – размер окна.

- стандартное отклонение координат, формула (2.4);

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n-1}} \quad (2.4)$$

где x_i – координаты;

μ – стандартное отклонение;

n – количество наблюдений.

- абсолютную разницу между двумя соседними горизонтальными координатами левого глаза и правого глаза, формула (2.5);

$$|x_2 - x_1| \quad (2.5)$$

где x_i – координаты.

- среднеквадратичную ошибку, формула (2.6);

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y}_i)^2 \quad (2.6)$$

где n – количество наблюдений;

y_i – фактическое значение;

$\bar{y}_i$ – предсказанное значение.

- скользящую дисперсию, формула (2.7);

$$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n-1} \quad (2.7)$$

где n – количество наблюдений;

x_i – координаты;

μ – стандартное отклонение.

- мгновенную скорость, формула (2.8);

$$v = \frac{s}{t} \text{ или } v = \lim_{\Delta t \rightarrow 0} \frac{\Delta s}{\Delta t} \quad (2.8)$$

где s – путь;

t – время.

- мгновенное ускорение, формула (2.9).

$$a = \lim_{\Delta t \rightarrow 0} \frac{\Delta v}{\Delta t} \quad (2.9)$$

где v – скорость;

t - время.

Функция принимает в качестве входных данных окулографические данные и список размеров окон для вычисления признаков. Функция возвращает оригинальные окулографические данные с добавленными математическими признаками.

2.7 Результаты

В результате к начальному набору данных были применены следующие методы:

1. Пороговые: IVT, IDT, IDT-mod, транзитивный методов;
2. Методы МО: DBSCAN, HDBSCAN, LOF;
3. Другие методы: Метод выявления типов полей зрения, добавление признаков на основе математических статистик с размером окна равным 10.

В результате применения методов признаковое пространство первоначального набора данных было увеличено на 180 признаков.

После использования каждого метода данные сохранялись в отдельный именованный набор данных для облегчения процесса извлечения последующих признаков.

Так же стоит отметить, что в процессе написания работы, было рассмотрено три дополнительных подхода по обработке признаков, подробного описания которых не предполагается, за исключением краткого:

1. В рамках данного исследования была проведена процедура интерполирования данных набора, которая включала в себя масштабирование данных (апскейлинг), в результате чего размер данных окулограмм увеличился в разы. Это позволило получить более детальную и точную информацию о движениях глаз. Реализация данной процедуры представлена в Приложении A24. Увеличение масштаба данных привело к тому, что:

- Пороговые алгоритмы стали работать значительно медленнее;
- Точность методов МО по поиску событий через кластеризацию аномалий дали значительную точность;
- Значительно выросло время на осуществление процесса классификации испытуемых методами МО.

По совокупности последствий от интерполирования, данный подход в дальнейшем не рассматривался, хотя при условии до стачного количества времени и ресурсов выглядит более перспективным и точным;

2. Обрезка данных с помощью сезонной декомпозиции. Главной идеей подхода был поиск глобального минимума на данных окулограмм по оси Y, которая явным образом показывала, что испытуемый закончил процесс чтения и перешел к процедуре ответов на вопросы, но подход не оправдал себя по причине, что не все записи окулограмм начинаются с момента чтения текста -верхнего левого угла пространства, отведенного для испытания. По этой причине данный подход совершал обрезку данных с первых временных отметок, что соответственно привело к потере данных окулограмм у части испытуемых, поэтому данный подход оказался не эффективным и был отвергнут. Реализация подхода представлена в Приложении A25;

3. Обрезка данных по временной отметке соответствующей минимальной продолжительности окулограмм в испытаниях. Данный подход,

предполагал, что процесс чтения, на данном отрезке, у всех испытуемых происходит в среднем одинаково и соответственные глазодвигательные события должны быть плюс-минус одинаковые, но в процессе отбора признаков и классификации пациентов было выявлена весьма ощутима потеря точности предсказания. В дальнейшем при тщательном выборочном осмотре стало ясно, что процедура испытания не была достаточно стандартизирована для испытуемых, кроме того возможно у испытуемых обеих групп были и другие патологии, оказывающие влияние на результаты испытания.

3 Отбор и оценка признаков. Сравнение результатов отбора. Классификация испытуемых

Для дальнейшего использования в процессе классификации испытуемых по группам риска было произведено два различных метода свертки расширенного набора данных в итоговый набор. Каждому набору было присвоено соответствующие название.

Описание методов свертки:

1. Реализация описана в Приложении A22. и основана на:

- Среднем и стандартном отклонении координатных и математических признаков;
- Подсчете количества глазодвигательных событий и их длительности в признаках.

2. Реализация описана в Приложении A23. и основана на:

- Среднем и стандартном отклонении координатных и математических признаков;
- Отношении количества глазодвигательных событий к общей длительности окулограммы испытуемого.

После использования методов свертки была проведена стандартизация данных для увеличения точности оценки признаков и предсказания модели классификации. Необходимости в применении OneHotEncoder не было.

В результате выполнения стандартизации были получены два итоговых набора данных: Quantitative и Relative. Наборы были использованные в последующем для отбора и оценки извлеченных признаков.

3.1 Применение машинного обучения

В работе была применена модель оценки классификатора LogisticRegression с отбором признаков на основе рекурсивного исключения признаков (RFE) и стратифицированной кросс-валидации. Реализация модели представлена в приложении A26.

Модель состоит из четырех основных функций: `svm_rfe_feature_selection`, `evaluate_classifier`, `perform_cross_validation` и `start_testing`.

Функция `svm_rfe_feature_selection` реализует отбор признаков с использованием рекурсивного исключения признаков. Она принимает в качестве параметров модель классификатора, матрицу признаков `X_train`, вектор целевой переменной `y_train` и количество признаков, которые необходимо отобрать `n_features`. Функция возвращает индексы отобранных признаков.

Функция `evaluate_classifier` оценивает качество классификатора на основе `accuracy`, `sensitivity`, `specificity` и матрицы конфузии, принимает в качестве параметров модель классификатора, матрицу признаков `X_train`, вектор целевой переменной `y_train`, матрицу признаков `X_test` и вектор целевой переменной `y_test`. Функция возвращает `accuracy`, `sensitivity`, `specificity` и матрицу неточностей.

Функция `perform_cross_validation` реализует Стратифицированная кросс-валидация (stratified k-fold), принимает в качестве параметров модель классификатора, матрицу признаков `X`, вектор целевой переменной `Y`, функцию отбора признаков, количество признаков, которое необходимо отобрать `n_features`, количество фолдов `n_splits`, фиксированного значения количества повторений `n_repeats`, список результатов и прогресс-бар. Функция добавляет результаты кросс-валидации в список результатов.

Функция **start_testing** запускает тестирование модели классификатора с использованием кросс-валидации и отбора признаков, принимает в качестве параметров модель классификатора, количество признаков, которое необходимо отобрать *num_feature*, количество фолдов *n_splits*, количество повторений *n_repeats* и путь к файлу для сохранения результатов.

Важно отметить, что реализовано два подхода к расчетам повторов:

1. Задаются константным числом и рассчитывается по формуле (3.1);

$$n_{repeats} = Const \quad (3.1)$$

где *Const* – константное значение.

2. Для каждой итерации значение повторов задает суммой чисел от 1 до *n_features*, рассчитывается по формуле (3.2).

$$n_{repeats} = step * \left(\frac{(n_{features} * (n_{features} - 1))}{2} \right) \quad (3.2)$$

где *step* – коэффициент шага;

n_features – количество признаков указанных для отбора;

step – размер шага.

Функция возвращает итоговый набор данных с результатами тестирования имеющий следующие признаки:

- *num_feature* – количество признаков;
- *selected_features* – отобранные признаки;
- *accuracy* – точность предсказания;
- *sensitivity* – чувствительность;
- *specificity* – специфичность;

- `confusion_matrices` – матрица неточностей.

Далее полученный набор данных был передан в функцию агрегирования **`get_result_total`**. Реализация функции представлена в приложении A27. В процессе выполнения функция, совершает две последовательные группировки данных:

1. Данные группируются по количеству признаков (`num_feature`) и из каждой полученной **группы** по средствам выполнения сортировки выбирается комбинация конкретных наиболее часто отбирившихся признаков (`selected_features`);

2. Данные с наиболее часто отбирившихся комбинаций признаков были сгруппированы и агрегированы по средним значениям признаков.

В результате выполнения функции **`get_result_total`** был получены сгруппированный набор данных со следующими признаками:

- `n_f` – необходимое количество признаков в группе (размер группы);
- `u_f` – общее количество отобранных уникальных признаков для группы за все количество повторов;
- `n_r` – количество повторов;
- `g_s` – количество повторов отбора лучшей уникальной комбинации признаков для данной группы;
- `g_r` – показатель отношения количество повторов отбора лучшей уникальной комбинации признаков к общему количеству повторов в группе;
- `selected_features` – названия признаков, входящих в лучшую уникальную комбинации для данной группы;
- `b_prob` – биномиальная вероятность, того что именно эта уникальная комбинация была выбрана из общего количества признаков с таким количеством положительных повторов из общего количества повторов в группе. Данный признак рассчитан по формуле (3.3);

- асс – среднее значение точности лучшей уникальной комбинации признаков;
- sens – среднее значение чувствительности лучшей уникальной комбинации признаков;
- спес – среднее значение специфичности лучшей уникальной комбинации признаков;
- с_m – матрица невязок со средними значениями лучшей уникальной комбинации признаков.

$$h_{prob} = C_n^m * p^m * q^{n-m} \quad (3.3)$$

где n – количество экспериментов;

m – количество успехов;

C_n^m - число сочетаний из m элементов по n;

p – вероятность успеха в эксперименте;

q – вероятность неудачи в эксперименте.

3.2 Отбор с константным значением повторов и оценка признаков набора Quantitative

В таблице 3.2.1 были приведены агрегированные результаты работы модели с рекурсивным отбором признаков с константным значением повторов, по итоговому набору Quantitative. Промежуточные выводы:

1. Точность предсказания по признакам, сформированным через подсчет количества глазодвигательных событий и их длительности в признаках, является довольно высокой;
2. Признаки, основанные на продолжительности испытания выделяются в общем признаковом пространстве своей частотой появления;

3. Группа 1-4 представленные пороговыми алгоритмами имеют среднюю точность около 95% и достаточно неплохо классифицируют испытуемого из HR, но гораздо хуже справляется с классификацией LR;

4. При появлении в группах признаков, выявленных методом МО DBSCAN средняя точность постепенно увеличивается, а средние показатели чувствительности и специфичности выравниваются и повышаются, но отношение повторов уникальных признаков к общему количеству повторов имеет тенденцию к значительному снижению при увеличении размера группы, поэтому в расчет можно брать только группы с размером до 6;

5. Из 1000 повторений на каждую группу признаков, в группах со значениями размера 1, 2 отношение количества уникальных повторов к общему количеству повторов больше половины, а значение вероятности стремиться к 0. Данный факт может свидетельствовать о хорошей предсказательной способности этих признаков;

6. Так же стоит отметить группу размером 3 имеющую довольно хорошие показатели;

7. Остальные группы довольно малочисленны.

Общие выводы: Вероятность повторного отбора признаков падает в зависимости от увеличения необходимого количества признаков для группы. Вероятно, подход со константным значением повторов для рекурсивного отбора признаков при условии поэтапного уменьшения размера группы, является ошибочным, т.к. на группу размером 1 отобрано 7 признаков из которых 1 лучший признак повторно выбирался 641 раз, а на группу размером 15 было отобрано 60 признаков из которых 14 лучших признаков отобраны всего 7 раз, что пропорционально 641 против 7. Данный факт означает, что нельзя сравнивать признаки из разных групп, особенно далеких друг от друга.

Таблица 3.2.1 – Агрегированные результаты

i	n_f	u_f	n_r	g_s	g_r	selected_features	b_prob	acc	sens	spec	c_m
0	1	7	1000	631	0.63	IDT_fixation_count	0	0.94236	0.97265	0.90909	[[8 1] [0 9]]
1	2	14	1000	641	0.64	IDT_fixation_count, EN_event_fixation_progressive_count	0	0.9446	0.95731	0.93077	[[8 1] [0 9]]
2	3	16	1000	355	0.36	IDT_fixation_count, IDT_fixation_mean_duration, EN_event_fixation_progressive_count	4.7352e-36	0.9594	0.96898	0.94875	[[8 0] [0 9]]
3	4	22	1000	151	0.15	FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, EN_event_fixation_progressive_count	0.0012397	0.94903	0.9638	0.93203	[[8 1] [0 9]]
4	5	27	1000	135	0.14	FV_foveal_count, IDT_fixation_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	4.4236e-06	0.96389	0.9856	0.94053	[[8 1] [0 10]]
5	6	33	1000	197	0.2	FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	0.014848	0.96824	0.98934	0.94487	[[8 0] [0 10]]
6	7	39	1000	55	0.06	FV_peripheral_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration, DBSCAN_T_other_count	1.0516e-31	0.99415	0.99253	0.99596	[[9 0] [0 10]]
7	8	48	1000	47	0.05	FV_peripheral_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	1.1956e-31	0.95415	0.93901	0.97104	[[9 0] [1 9]]
8	9	50	1000	22	0.02	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	1.4932e-56	0.96079	0.96263	0.9577	[[8 0] [0 9]]

Продолжение таблицы 3.2.1

9	10	51	1000	18	0.02	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration, DBSCAN_P_event_mean_duration, DBSCAN_T_other_count	2.0436e-68	0.99691	1.0	0.99383	[[9 0] [0 10]]
10	11	53	1000	12	0.01	FV_peripheral_count, IDT_fixation_count, IDT_fixation_mean_duration, HDBSCAN_other_count, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_T_event_mean_duration	1.9144e-81	0.93665	0.89722	0.98032	[[9 0] [1 9]]
11	12	54	1000	15	0.02	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_FS_event_count, DBSCAN_T_event_mean_duration	3.4056e-85	0.94659	0.90593	0.99259	[[9 0] [1 9]]
12	13	56	1000	15	0.02	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, HDBSCAN_other_count, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_FS_event_count, DBSCAN_T_event_mean_duration	2.1146e-90	0.9425	0.89778	0.99259	[[9 0] [1 9]]

Продолжение таблицы 3.2.1

1 3	14	59	1000	9	0.01	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, MEAN_event_fixation_progressive_mean_duration, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_FS_event_count, DBSCAN_T_event_mean_duration	1.6646e-101	0.94672	0.89877	1.0	[[9 0] [1 9]]
1 4	15	60	1000	7	0.01	FV_peripheral_count, FV_peripheral_mean_duration, FV_foveal_count, IVT_fixation_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, MEAN_event_saccade_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_FS_event_count, DBSCAN_T_other_count, gender	1.0229e-111	0.9457	1.0	0.8869	[[8 1] [0 10]]

3.3 Отбор с константным значением повторов и оценка признаков набора Relative

В таблице 3.3.1 были приведены агрегированные результаты работы модели с рекурсивным отбором признаков с константным значением повторов, по итоговому набору Relative. По результатам видно, что:

1. Точность предсказания по признакам, сформированным через отношение количества глазодвигательных событий к общей длительности окулограмм испытуемых, является не высокой;
2. Признаки, основанные продолжительности испытания не выделяются в общем признаковом пространстве;

3. Из 1000 повторений на каждую группу признаков, в группах со значениями размера 1, 2 отношение количества уникальных повторов к общему количеству повторов примерно больше половины;

4. Вероятность отбора таких уникальных комбинаций как в группах 1,2 с таким большим количеством повторов равна близкому к 0 значению;

5. Группы со значением 3-15 являются крайне малочисленными, значит их агрегированные средние значения имеют большое стандартное отклонение. Также возможно, данные признаки небыли обучены на всех фолдах.

Общие выводы: в данном наборе были выявлены проблемы аналогичные проблемам набора Quantitative. Дополнительно стоит отметить, что признаки, основанные на длительности испытаний, утратили часть предсказательной силы, что сделало использование этого набора не целесообразным.

Таблица 3.3.1 – Агрегированные результаты

i	n_f	u_f	n_r	g_s	g_r	selected_features	b_prob	acc	sens	spec	c_m
0	1	8	1000	717	0.72	IDT_fixation_mean_duration	0	0.86759	0.84973	0.88718	[[8 1] [1 8]]
1	2	13	1000	492	0.49	IDT_fixation_mean_duration, NE_event_saccade_progressive_count	0	0.88158	0.86075	0.90481	[[8 1] [1 8]]
2	3	22	1000	250	0.25	FV_foveal_count, IDT_fixation_mean_duration, NE_event_saccade_progressive_count	4.0377e-22	0.89915	0.89778	0.90094	[[8 1] [1 9]]
3	4	34	1000	149	0.15	FV_peripheral_count, FV_foveal_count, IDT_fixation_mean_duration, NE_event_saccade_progressive_count	0.00042904	0.87309	0.8569	0.89066	[[8 1] [1 8]]
4	5	39	1000	120	0.12	FV_foveal_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, DBSCAN_P_other_mean_duration	0.028544	0.90448	0.90102	0.90868	[[8 1] [1 9]]
5	6	45	1000	123	0.12	FV_peripheral_count, FV_foveal_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, DBSCAN_P_other_mean_duration	0.02393	0.90508	0.89973	0.91102	[[8 1] [1 9]]
6	7	49	1000	77	0.08	FV_peripheral_count, FV_foveal_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	4.6687e-11	0.91972	0.92049	0.91955	[[8 1] [1 9]]
7	8	54	1000	66	0.07	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	3.4477e-16	0.90191	0.92508	0.87647	[[8 1] [1 9]]

Продолжение таблицы 3.3.1

8	9	60	1000	42	0.04	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, EN_event_fixation_progressive_count, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	1.7876e-28	0.93003	0.93069	0.9289	[[8 1] [1 9]]
9	10	63	1000	38	0.04	FV_peripheral_count, FV_foveal_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, NE_event_fixation_progressive_count, EN_event_fixation_progressive_mean_duration, MEAN_event_fixation_progressive_mean_duration, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	2.4273e-34	0.90628	0.8617	0.95577	[[8 0] [1 8]]
10	11	64	1000	41	0.04	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, NE_event_fixation_progressive_count, EN_event_fixation_progressive_mean_duration, MEAN_event_fixation_progressive_mean_duration, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	1.6301e-37	0.90137	0.86965	0.93699	[[8 1] [1 8]]
11	12	65	1000	22	0.02	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, NE_event_fixation_progressive_count, EN_event_fixation_progressive_mean_duration, MEAN_event_fixation_progressive_mean_duration, TRANS_other_mean_duration, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	1.0438e-58	0.91122	0.92374	0.89583	[[8 1] [1 9]]
12	13	66	1000	25	0.02	LX, X_mean_derivative, FV_peripheral_count, FV_foveal_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, NE_event_fixation_progressive_count, EN_event_fixation_progressive_mean_duration, MEAN_event_fixation_progressive_mean_duration, MEAN_event_saccade_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration	1.4178e-61	0.95637	0.96311	0.94889	[[8 0] [0 9]]

Продолжение таблицы 3.3.1

1 3	14	67	1000	19	0.02	LX, X_mean_derivative, FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, NE_event_fixation_progressive_count, EN_event_fixation_progressive_mean_d uration, MEAN_event_fixation_progressive_mea n_duration, MEAN_event_saccade_progressive_cou nt, MEAN_event_saccade_progressive_mea n_duration, DBSCAN_P_other_mean_duration	1.1388e-73	0.9486	0.96316	0.93129	[[8 1] [0 10]]
1 4	15	67	1000	20	0.02	LX, X_mean_derivative, FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, IDT_saccade_count, NE_event_saccade_progressive_count, NE_event_fixation_progressive_count, EN_event_fixation_progressive_mean_d uration, MEAN_event_fixation_progressive_mea n_duration, MEAN_event_saccade_progressive_cou nt, MEAN_event_saccade_progressive_mea n_duration, TRANS_event_mean_duration, DBSCAN_P_other_mean_duration	4.5827e-80	0.9652	0.94944	0.98264	[[8 0] [0 9]]

3.4 Отбор с вычисляемым значением повторов и оценка признаков набора Quantitative

В таблице 3.3.1 были приведены агрегированные результаты работы модели с рекурсивным отбором признаков с вычисляемым значением повторов, по итоговому набору Quantitative.

По результатам были проведены расчеты процентного изменения, по формуле (3.4):

1. Сократилась скорость снижения вероятности у групп на 95.6 %;
2. Среднее отклонение у групп выросло на 530.86% процентов, т.к. значения вероятности не так быстро стремятся к 0.

$$p = \frac{a - b}{abs(b)} * 100 \quad (3.4)$$

где a – новое значение переменной;

b – старое значение переменной.

Данные результаты дают основания полагать, что расчет значения повтора, по средствам использования **комбинаторики**, позволит улучшить качество отбора признаков. Однако это приведет к значимому увеличению количества итераций, что повлечет за собой длительность отбора признаков, в противные случаи переход на более мощные вычислительные средства.

Таблица 3.4.1 – Агрегированные результаты

i	n_f	u_f	n_r	g_s	g_r	selected_features	b_prob	acc	sens	spec	c_m
0	1	2	10	8	0.8	IDT_fixation_count	0.043945	0.93165	0.96111	0.90104	[[8 1] [0 9]]
1	2	8	20	13	0.65	IDT_fixation_count, EN_event_fixation_progressive_count	0.00015419	0.94062	0.95214	0.92842	[[8 1] [0 9]]
2	3	11	30	11	0.37	IDT_fixation_count, IDT_fixation_mean_duration, EN_event_fixation_progressive_count	0.079923	0.93567	0.93434	0.93687	[[8 1] [1 9]]
3	4	14	40	7	0.18	FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, EN_event_fixation_progressive_count	0.043634	0.91395	0.92698	0.89683	[[8 1] [1 9]]
4	5	16	50	8	0.16	IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	0.0071471	0.98611	0.97361	1.0	[[9 0] [0 9]]
5	6	18	60	11	0.18	FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	0.004551	0.9806	1.0	0.95833	[[8 0] [0 10]]
6	7	26	70	6	0.09	FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration, DBSCAN_P_event_mean_duration	9.5574e-05	0.99074	1.0	0.97917	[[8 0] [0 10]]
7	8	31	80	4	0.05	FV_peripheral_count, IVT_fixation_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	9.859e-07	0.89035	0.94722	0.83333	[[8 2] [0 9]]
8	9	35	90	3	0.03	FV_peripheral_count, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, LOF_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	1.1729e-08	0.96296	0.96296	0.96296	[[9 0] [0 9]]

Продолжение таблицы 3.4.1

9	10	36	100	3	0.03	FV_peripheral_count, FV_peripheral_mean_duration, IDT_fixation_count, IDT_fixation_mean_duration, HDBSCAN_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration	6.7745e-11	1.0	1.0	1.0	[[9 0] [0 10]]
10	11	38	110	3	0.03	FV_peripheral_count, FV_peripheral_mean_duration, IDT_fixation_count, IDT_fixation_mean_duration, HDBSCAN_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, DBSCAN_P_other_mean_duration, DBSCAN_T_event_mean_duration	6.8865e-13	0.94542	0.8963	1.0	[[9 0] [1 9]]
11	12	40	120	3	0.02	FV_peripheral_count, FV_peripheral_mean_duration, IDT_fixation_count, IDT_fixation_mean_duration, HDBSCAN_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_T_event_mean_duration	5.7055e-15	0.94542	0.8963	1.0	[[9 0] [1 9]]
12	13	44	130	3	0.02	FV_peripheral_count, FV_peripheral_mean_duration, FV_foveal_count, IDT_fixation_count, IDT_fixation_mean_duration, HDBSCAN_other_count, LOF_event_count, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_T_event_mean_duration	4.4618e-16	0.94542	0.8963	1.0	[[9 0] [1 9]]
13	14	46	140	3	0.02	FV_peripheral_count, FV_peripheral_mean_duration, FV_foveal_count, IVT_fixation_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_FS_event_count, DBSCAN_T_other_count, gender	3.2263e-18	0.94542	1.0	0.88426	[[8 1] [0 10]]

Продолжение таблицы 3.4.1

14	15	48	150	3	0.02	FV_peripheral_count, FV_peripheral_mean_duration, FV_foveal_count, IVT_fixation_count, IDT_fixation_count, IDT_fixation_mean_duration, NE_event_fixation_regressive_count, EN_event_fixation_progressive_count, MEAN_event_fixation_progressive_count, MEAN_event_saccade_progressive_count, MEAN_event_saccade_progressive_mean_duration, DBSCAN_P_other_mean_duration, DBSCAN_FS_event_count, DBSCAN_T_other_count, gender	2.0185e-20	0.94542	1.0	0.88426	[[8 1] [0 10]]
----	----	----	-----	---	------	--	------------	---------	-----	---------	---------------------

3.5 Классификация испытуемых на полученном наборе признаков

В завершении работы два подхода к расчету повторов были протестированы на классификаторе XGBClassifier из библиотеки xgboost, при помощи поиска по сетке. Модель представлена в Приложении A28. Были получены следующие результаты двух групп из итогового набора Quantitative:

1. Группа 6 размером была взята с фиксированным значением повторов. Были получены следующие параметры модели:

- colsample_bytree: 0.1;
- learning_rate: 0.01;
- max_depth: 3;
- n_estimators: 100;
- subsample: 0.7.

Точность 0.95310, Матрица невязок $\begin{bmatrix} 18 & 2 \\ 0 & 17 \end{bmatrix}$.

2. Группа 6 размером с вычисляемым значением повторов. Были получены следующие параметры модели:

- colsample_bytree: 0.7;
- learning_rate: 0.01;
- max_depth: 1;
- n_estimators: 50;
- subsample: 0.5.

Точность 0.95970, Матрица невязок $\begin{bmatrix} 18 & 2 \\ 0 & 17 \end{bmatrix}$.

3. Группа, присутствующая в обоих подходах, из одного признака IDT_fixation_count:

- colsample_bytree: 0.1;
- learning_rate: 0.1;
- 'max_depth': 1;
- 'n_estimators': 50;

- 'subsample': 0.5;
- Точность 0.94643, Матрица невязок $\begin{bmatrix} 18 & 2 \\ 0 & 17 \end{bmatrix}$.

Возникло подозрение часть испытаний были проведены с некими значимыми отклонениями, что приводит к потере точности именно по показателю специфичности, то есть для группы LR. При детальном ручном анализе было выявлено, что имеется, часть испытуемых, таких как испытуемый под идентификационным номером 761NMS (рис. 3.5.1). На окулограмме данного испытуемого, отчетливо видно, что быстро прочитав текст (до отметки в 20000, что соответствует диапазону LR) испытуемый половину оставшегося времени отвечал на вопросы. Длительность его испытания стала максимальной среди LR, что является аномалией, но также его длительность максимальна и для HR. Поэтому встает вопрос, о соблюдении стандарта проведения испытаний в этом наборе данных.

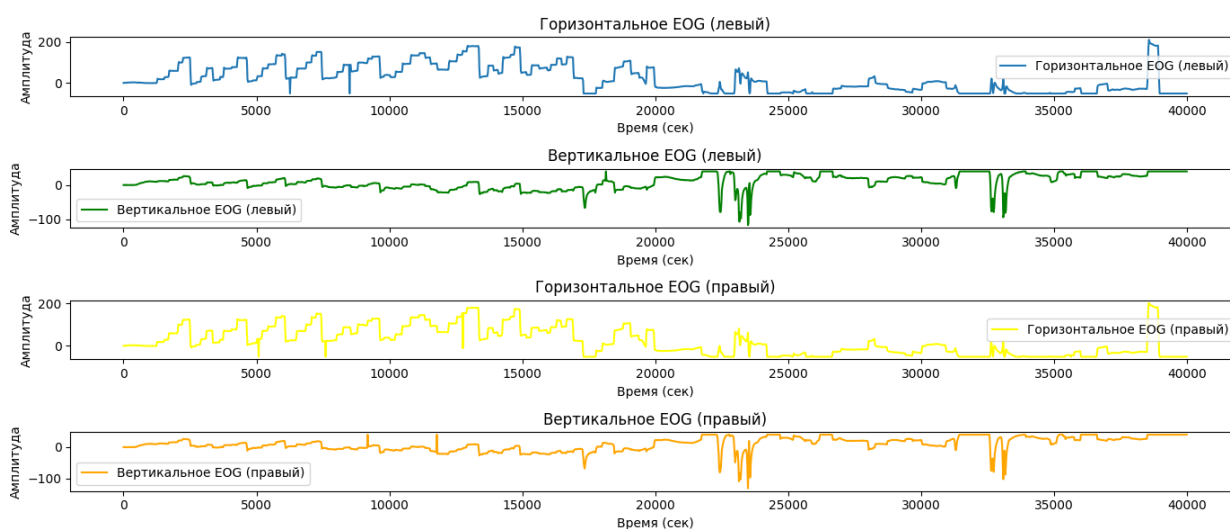


Рисунок 3.5.1 – График EOG признаков LX, LY, RX, RY в зависимости от признака T для испытуемого 761NMS

3.6 Результаты

Признаки, отобранные рекурсивным методом при условии, что они берутся из групп с большим количеством повторов, но малого размера, могут дать достаточную точность;

Признаки, основанные на математических статистиках, не вошли в первые 15 наиболее значимых групп, однако при увеличении числа повторов и размера выборки признаков они могут оказаться более значимыми. Это указывает на необходимость дальнейших исследований с более обширными данными для оценки их потенциала.

В ходе исследования было установлено, что пороговые методы оказывают значительное влияние на идентификацию типов движения глаз. Методы, основанные на выявлении полей зрения, также показали свою эффективность. Признаки, полученные с использованием этих методов, продемонстрировали хорошие результаты в процессе классификации.

Рекурсивный метод отбора признаков, включающий вычисляемые повторы, показал перспективность в улучшении диагностической точности. Предполагается, что использование комбинаторного подхода в определении повторов для уникальных групп признаков может значительно улучшить результаты отбора, хотя это и потребует дополнительных ресурсов.

Кроме того, было выявлено, что методы кластеризации и поиска аномалий являются перспективными для анализа окулографических данных. В будущих экспериментах, при наличии соответствующих материально-технических средств, применение этих методов в сочетании с интерполяцией данных, что может существенно улучшить процесс выявления аномалий.

ЗАКЛЮЧЕНИЕ

В данной исследовательской работе была поставлена цель повышения точности диагностики и понимания окуломоторной активности путем применения методов машинного обучения к набору данных окулографии. Цель была достигнута за счет решения ряда задач.

В первой задаче проводилось изучение различных подходов, методов машинного обучения и предметной области, связанной с глазодвигательными событиями в анализе данных окулограмм. Были изучены пороговые методы и методы машинного обучения, которые могут быть использованы для анализа данных окулограмм и выявления особенностей движений глаз у пациентов с неврологическими патологиями.

Во второй задаче был проведен анализ набора данных окулограмм, полученных в ходе экспериментов, проведенных с целью изучения взаимосвязи между определенной группой риска и дислексией. В ходе анализа были выявлены определенные паттерны поведения, свойственные своим группам, из чего следовало значительное количественное распределение данных в наборах окулограмм для группы с высоким риском.

В третьей задаче для выявления глазодвигательных событий и генерации новых признаков была осуществлена программная реализация пороговых методов IVT, IDT, IDT-mod и Транзитивного метода, и методов машинного обучения DBSCAN, HDBSCAN и LOF, также в работе была осуществлена программная реализация метода поиска полей зрения. Дополнительно были использованы различные математические статистики. В результате выполнения этой задачи было извлечено 180 признаков.

В четвертой задаче была реализована модель рекурсивного отбора значимых признаков в совокупности с двумя различными подходами к заданию значения количества повторов основанными на константном и

вычисляемом значении. Данная реализация на основании метрик точности и дополнительно вычисленных метрик отношений и биномиальной вероятности повторного появления группы признаков, позволила выявить значимые признаки для всех групп с заданным размером. В заключении было осуществлено несколько независимых процедур обучения модели классификации на отобранных значимых признаках из разных групп, в результате чего была достигнута точность классификации в 96%. Данную точность можно считать достаточно хорошим показателем, т.к. в процессе исследования были обнаружены окулограммы которые можно отнести к аномальным.

На основе полученных результатов можно сделать вывод о том, что цель этой работы была достигнута и значимость применения методов машинного обучения для анализа окулографических данных в диагностических целях дислексии, невозможно переоценить. Также стоит упомянуть, что рекурсивный отбор признаков с реализацией через комбинаторный анализ повторов представляются многообещающими направлениями для дальнейших исследований. Методы кластеризации и поиска аномалий, дополненные интерполяцией данных, также обладают потенциалом для улучшения качества диагностики.

Результаты данной работы могут быть использованы для более эффективной диагностики, лечения и последующей реабилитации после неврологических заболеваний.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. 1D CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits | Behavior Research Methods. – URL: <https://link.springer.com/article/10.3758/s13428-018-1144-2> (дата обращения: 22.03.2024). – Текст : электронный.
2. А. А. Делов, К. Ю. Цыганкова. Возможности применения в изучении цветовой когнции методов интеллектуального анализа данных, полученных с помощью шлема виртуальной реальности с функцией айтрекинга / А. А. Делов, К. Ю. Цыганкова // Социальные трансформации. – 2022. – Т. 33. – С. 55-64.
3. В. А. Барабанщиков, А. В. Жегалло. Айтрекинг: Методы регистрации движений глаз в психологических исследованиях и практике. Айтрекинг / В. А. Барабанщиков, А. В. Жегалло. – URL: <https://elibrary.ru/item.asp?id=23345071> (дата обращения: 08.02.2024). – Текст : электронный.
4. Википедия: Свободная энциклопедия. Окулография [Электронный ресурс] / Википедия: Свободная энциклопедия. – URL: (дата обращения: 08.02.2024). – Текст : электронный.
5. Восприятие Мультсериала «Смешарики» Московской Молодежью: Подход К Исследованию / Текст : электронный. – Восприятие Мультсериала «Смешарики» Московской Молодежью. – URL: <https://elibrary.ru/item.asp?id=50748361> (дата обращения: 22.12.2023).
6. Е. А. Новиков, И. А. Ваколюк, Р. Д. Ахапкин [и др.]. Автоматизация метода компьютерной окулографии для исследований центральной нервной системы на основе пассивного анализа видеоизображения / Е. А. Новиков, И. А. Ваколюк, Р. Д. Ахапкин [и др.] // Машинное обучение и анализ данных. – 2015. – Т. 1. – № 12. – С. 1752-1761.

7. Исследование методов машинного обучения для решения задачи прогноза успешности работы оператора по дополнительной информации / Текст : электронный. – URL: <https://elibrary.ru/item.asp?id=46481642> (дата обращения: 08.02.2024).
8. Ощущение и восприятие. – URL: <https://www.elibrary.ru/item.asp?id=22037753> (дата обращения: 14.05.2024). – Текст : электронный.
9. Dalveren G. G. M. Evaluation of Ten Open-Source Eye-Movement Classification Algorithms in Simulated Surgical Scenarios / G. G. M. Dalveren, N. E. Cagiltay // IEEE Access. – 2019. – Т. 7. – С. 161794-161804.
10. Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and Internet of Things technologies / A. F. Klaib, N. O. Alsrehin, W. Y. Melhem [и др.] // Expert Systems with Applications. – 2021. – Т. 166. – С. 114037.
11. Kovalskaya A. Eye tracing in ophthalmolog / A. Kovalskaya, S. Koskin. – 2019. – Т. 38. – № 1. – С. 30-32.
12. Monty R. A. An advanced eye-movement measuring and recording system / R. A. Monty // American Psychologist. – 1975. – Т. 30. – № 3. – С. 331-335.
13. On the necessity of adaptive eye movement classification in conditionally automated driving scenarios / C. Braunagel, D. Geisler, W. Stolzmann [и др.]. – Текст : электронный // Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications : ETRA '16. – New York, NY, USA : Association for Computing Machinery, 2016. – С. 19-26. – URL: <https://doi.org/10.1145/2857491.2857529> (дата обращения: 21.03.2024).
14. Punde P. A study of eye tracking technology and its applications / P. Punde, M. Jadhav, R. Manza. – 2017. – С. 86-90.
15. The Neurobiological Basis of Skilled and Impaired Reading: Recent Findings and New Directions: Scientific Studies of Reading: Vol 8, No 3. – URL:

https://www.tandfonline.com/doi/abs/10.1207/s1532799xssr0803_6?casa_token=l7LGWybXdtEAAAAA:C5sKs8uIdnfig-vYANFr18HM0chKdLse6_UCdSYsGXXK_8vfumISYNBQII9-WYne-t7aInISaoQUAnw (дата обращения: 22.03.2024). – Текст : электронный.

16. Zemblys R. Eye-movement event detection meets machine learning / R. Zemblys. – Siauliai University, 2016.

17. Alpern M. Eye Movements / M. Alpern. – Text : electronic // Visual Psychophysics / eds. M. Alpern [et al.]. – Berlin, Heidelberg : Springer, 1972. – P. 303-330. – URL: https://doi.org/10.1007/978-3-642-88658-4_12 (date accessed: 16.03.2024).

18. Birawo B. Review and Evaluation of Eye Movement Event Detection Algorithms / B. Birawo, P. Kasprowski // Sensors. – 2022. – Vol. 22. – № 22. – P. 8810.

19. Hartridge H. Methods of Investigating Eye Movements / H. Hartridge, L. C. Thomson // British Journal of Ophthalmology. – 1948. – Vol. 32. – № 9. – P. 581-591.

20. Is human classification by experienced untrained observers a gold standard in fixation detection? / I. T. C. Hooge, D. C. Niehorster, M. Nyström [et al.] // Behavior Research Methods. – 2018. – Vol. 50. – № 5. – P. 1864-1881.

21. Naqvi R. Fuzzy System-Based Target Selection for a NIR Camera-Based Gaze Tracker / R. Naqvi, M. Arsalan, K. Park // Sensors (Basel, Switzerland). – 2017. – Vol. 17. – № 4. – P. 862.

22. One algorithm to rule them all? An evaluation and discussion of ten eye movement event-detection algorithms / R. Andersson, L. Larsson, K. Holmqvist [et al.] // Behavior Research Methods. – 2017. – Vol. 49. – One algorithm to rule them all? – № 2. – P. 616-637.

23. Salvucci D. D. Identifying fixations and saccades in eye-tracking protocols / D. D. Salvucci, J. H. Goldberg. – Text : electronic // Proceedings of the symposium on Eye tracking research & applications - ETRA '00 the symposium. –

Palm Beach Gardens, Florida, United States : ACM Press, 2000. – P. 71-78. – URL: <http://portal.acm.org/citation.cfm?doid=355017.355028> (date accessed: 15.03.2024).

24. Screening for Dyslexia Using Eye Tracking during Reading / M. N. Benfatto, G. Ö. Seimyr, J. Ygge [et al.] // PLOS ONE. – 2016. – Vol. 11. – № 12. – P. e0165508.

25. Habib M. The neurological basis of developmental dyslexia: an overview and working hypothesis / M. Habib // Brain: A Journal of Neurology. – 2000. – T. 123 Pt 12. – The neurological basis of developmental dyslexia. – С. 2373-2399.

26. Larsson L. Event Detection in Eye-Tracking Data for Use in Applications with Dynamic Stimuli : thesis/doccomp / L. Larsson. – Lund University, 2016. – URL: <http://lup.lub.lu.se/record/8600501> (дата обращения: 22.03.2024). – Текст : электронный.

27. Shaywitz S. E. Dyslexia (specific reading disability) / S. E. Shaywitz, B. A. Shaywitz // Biological Psychiatry. – 2005. – Т. 57. – № 11. – С. 1301-1309.

28. Альфред Ярбус. Роль движений глаз в процессе зрения / Альфред Ярбус Google-Books-ID: K1bQDAAAQBAJ. – Directmedia, 2016. – 173 с.

29. Дегенерация сетчатки: причины, симптомы и лечение в статье детского офтальмолога Бекетова Е. Ю. – URL: <https://probolezny.ru/degeneraciya-setchatki/> (дата обращения: 20.03.2024). – Текст : электронный.

30. Жегалло А. Регистрация и анализ направленности взора человека : Методы психологии / А. Жегалло, В. Барабанщиков. – ФГБУН Институт психологии РАН, 2022. – 323 с.

31. Использование кардиометрических и окулографических методов в подготовке специалистов психолого-педагогического профиля (на примере песочного моделирования) / В. В. Батколина, В. А. Зернов, Э. В. Лихачева [и

др.] // Высшее образование сегодня. – 2021. – Т. Выпуск 5 2021. – С. Pages 071080.

32. Gracheva M. A. Dyslexia Diagnostics Based on Eye Movements and Artificial Intelligence Methods: A Review / M. A. Gracheva, S. Shalileh // Clinical Psychology and Special Education. – 2023. – Т. 12. – Dyslexia Diagnostics Based on Eye Movements and Artificial Intelligence Methods. – № 3. – С. 1-29.

33. Network analysis of cognitive, oculomotor and speech parameters in schizophrenia / A. B. Shmukler, G. P. Kosytuk, A. V. Latanov [и др.] // Zhurnal nevrologii i psikhiatrii im. S.S. Korsakova. – 2020. – Т. 120. – № 6. – С. 54.

34. Using machine learning to detect events in eye-tracking data / R. Zemblys, D. C. Niehorster, O. Komogortsev, K. Holmqvist // Behavior Research Methods. – 2018. – Т. 50. – № 1. – С. 160-181.

ПРИЛОЖЕНИЕ А. КОД СКРИПТОВ

А1. ОБЪЕДИНЕНИЯ ОКУЛЛОГРАМ ПАЦИЕНТОВ В ОБЩИЙ НАБОР ДАННЫХ

```
def preprocessing(data):
    """
    Функция для предварительной обработки данных окулограмм.

    Args:
        data (pandas.DataFrame): Данные окулограмм, требующие предварительной
        обработки.

    Returns:
        pandas.DataFrame: Обработанные данные окулограмм.
    """
    data = data.replace({' ': '.'}, regex=True)
    data[['LX', 'LY', 'RX', 'RY']] = data[['LX', 'LY', 'RX',
'RY']].apply(pd.to_numeric, errors='coerce')
    return data


def patient_id_decoding(data):
    """
    Функция для декодирования идентификатора пациента и добавления
    информации о поле и наличии инвалидности.

    Args:
        data (pandas.DataFrame): Данные, включающие идентификаторы пациентов.

    Returns:
        pandas.DataFrame: Данные с добавленными столбцами 'gender' и
        'disabled', содержащими информацию о поле и наличии инвалидности пациентов.
    """
    folder_map = {
        '1': (0, 1),
        '2': (1, 1),
        '3': (0, 0),
        '4': (1, 0)
```

```

    }

    data['gender'] = data['id_patient'].str[-1].map(lambda x:
folder_map.get(x, (None, None))[0])
    data['disabled'] = data['id_patient'].str[-1].map(lambda x:
folder_map.get(x, (None, None))[1])

    return data

def get_data_all(length=None):
    """
    Args:
    length (int): Длина для обрезки данных.

    Функция для объединения данных из нескольких файлов окулограмм в общий
    набор данных.

    Returns:
    pandas.DataFrame: Общий набор данных, содержащий информацию из всех
    файлов окулограмм.
    """
    df_all = pd.DataFrame()

    root_dir =
'/content/drive/MyDrive/lectures/diplom/RecordingData/Recording Data/'
    data_files = glob.glob(os.path.join(root_dir, '**', '*.txt'),
recursive=True)
    dfs = []

    for data_file in data_files:
        folder_id = os.path.basename(os.path.dirname(data_file))
        file_df = pd.read_csv(data_file, header=0, delimiter='\t')

        # Ограничение длины датасета
        if length is not None:
            file_df = file_df[:length]

        # Обработка колонок
        file_df = preprocessing(file_df)
        # Центрирование взгляда
        file_df['X_mean'] = file_df[['LX', 'RX']].mean(axis=1)
        file_df['Y_mean'] = file_df[['LY', 'RY']].mean(axis=1)

```

```

# Добавление идентификатора пациента
file_df['id_patient'] = folder_id
file_df = patient_id_decoding(file_df.copy())

dfs.append(file_df)

df_all = pd.concat(dfs, ignore_index=True)

return df_all

```

ПРИЛОЖЕНИЕ А2. ВЫПАДАЮЩИЙ СПИСОК ИСПЫТУЕМЫХ

```

#@TITLE **ВЫБОР ПАРАМЕТРОВ** { RUN: "AUTO" }
DATA_SET = DF_ALL #@PARAM ['DF_500','DF_ALL'] {TYPE:"RAW"}
FOR NAME, VALUE IN GLOBALS().ITEMS():
    IF ISINSTANCE(VALUE, PD.DATAFRAME) AND VALUE.EQUALS(DATA_SET):
        DF_NAME = NAME
        BREAK
ID_PATIENT = '761NM3' #@PARAM ['112KA1'", "'111RP1'", "'112JU3'",
"'131CV4'", "'111GM3'", "'132AD3'", "'111JA2'", "'125KM1'", "'132FJ1'",
"'131SA2'", "'141BE2'", "'142EJ1'", "'142MM1'", "'141MS4'", "'132IM3'",
"'133AM4'", "'133GJ3'", "'132SJ1'", "'141NK2'", "'141NH4'", "'224BE4'",
"'143JM3'", "'224CM2'", "'211LJ1'", "'211ND1'", "'142OJ3'", "'143SM3'",
"'211CF3'", "'311HL2'", "'142SJ3'", "'335HJ3'", "'322BJ3'", "'312BE1'",
"'312BM4'", "'332PM1'", "'322RE1'", "'312HJ3'", "'332JH3'", "'322EM1'",
"'322PS3'", "'342HM1'", "'343SJ1'", "'342GD1'", "'343BJ3'", "'343BL1'",
"'343SF1'", "'335NJ1'", "'346KU1'", "'344LL1'", "'351LR1'", "'421LE2'",
"'414JM3'", "'352KM1'", "'422JN3'", "'414FJ1'", "'422BC1'", "'351SS3'",
"'421LS4'", "'421SP4'", "'421AT2'", "'512LM1'", "'511EE1'", "'512ED1'",
"'511YD3'", "'512EO3'", "'423MM3'", "'423MD1'", "'511RD3'", "'512BM2'",
"'512PM4'", "'533KM1'", "'522HD3'", "'523GM1'", "'533OA3'", "'522LA3'",
"'523FM3'", "'533HM2'", "'532EJ2'", "'522NC1'", "'522BK1'", "'613KF1'",
"'612SA1'", "'612LJ3'", "'622FD1'", "'534FT1'", "'621NL4'", "'613SP3'",
"'621NP2'", "'622LB3'", "'534SR1'", "'622RM1'", "'623JA1'", "'623WL2'",
"'711TA3'", "'623BM4'", "'701AA4'", "'712FJ1'", "'623GH1'", "'622SA3'",
"'701LV2'", "'713VC3'", "'714GC4'", "'712WA1'", "'713PE4'", "'712KO3'",
"'713PE2'", "'712HL3'", "'721AE4'", "'721AF1'", "'712SJ1'", "'724OP3'",
"'724VS1'", "'721KJ2'", "'721HL2'", "'723DT1'", "'725MF1'", "'724SM3'",
"'721SM3'", "'721JL4'", "'723SJ1'", "'726PT3'", "'745DJ1'", "'726HD1'",

```

```

''742EO3'', ''742AJ3'', ''726OG1'', ''725OM3'', ''732KD1'', ''741US2'',
''745CI4'', ''751GJ3'', ''745RK3'', ''753EM1'', ''752PJ3'', ''752LM3'',
''752AK1'', ''753HP1'', ''752NA1'', ''751AH1'', ''751GS1'', ''754WF3'',
''756FJ3'', ''761SJ1'', ''755TD3'', ''761SJ3'', ''761EL1'', ''761NM3'',
''754TD3'', ''753PJ3'', ''762ET3'', ''775SM1'', ''762VH1'', ''774HJ2'',
''772AK3'', ''771TJ1'', ''762LB3'', ''781PT1'', ''762SL1'', ''772CC1'',
''775DA3'', ''794HH3'', ''782SC3'', ''793MD1'', ''794EM1'', ''781SD3'',
''791LK3'', ''782PR1'', ''794GO3'', ''793LL3'', ''781SE4'', ''794SD1'',
''796PA3'', ''794LA3'', ''795MM1'', ''795SJ1'', ''796GA3'', ''796NT3'',
''826AL2'', ''795JM1'', ''825PA3'', ''826ES4'', ''831OK2'', ''831PA3'',
''826SJ1'', ''826PP1''] {TYPE:"RAW"}

```

ПРИЛОЖЕНИЕ А3. ГРАФИК ЕОГ ПРИЗНАКОВ LX, LY, RX, RY В ЗАВИСИМОСТИ ОТ ПРИЗНАКА T

```

def plot_eog(data):
    """
    Функция для построения графиков окулограмм.

    Args:
        data (pandas.DataFrame): Данные окулограмм для построения графиков.

    Returns:
        None
    """
    plt.figure(figsize=(14, 6))

    plt.subplot(4, 1, 1)
    plt.plot(data['T'], data['LX'], label='Горизонтальное ЕОГ (левый)')
    plt.title('Горизонтальное ЕОГ (левый)')
    plt.xlabel('Время (сек)')
    plt.ylabel('Амплитуда')
    plt.legend()

    plt.subplot(4, 1, 2)
    plt.plot(data['T'], data['LY'], label='Вертикальное ЕОГ (левый)',
color='green')
    plt.title('Вертикальное ЕОГ (левый)')

```

```

plt.xlabel('Время (сек)')
plt.ylabel('Амплитуда')
plt.legend()

plt.subplot(4, 1, 3)
plt.plot(data['T'], data['RX'], label='Горизонтальное EOG (правый)',
color='yellow')
plt.title('Горизонтальное EOG (правый)')
plt.xlabel('Время (сек)')
plt.ylabel('Амплитуда')
plt.legend()

plt.subplot(4, 1, 4)
plt.plot(data['T'], data['RY'], label='Вертикальное EOG (правый)',
color='orange')
plt.title('Вертикальное EOG (правый)')
plt.xlabel('Время (сек)')
plt.ylabel('Амплитуда')
plt.legend()

plt.tight_layout()
plt.show()

```

ПРИЛОЖЕНИЕ А4. ГРАФИК EOG ЗАВИСИМОСТИ LX ОТ LY И ЗАВИСИМОСТИ RX ОТ RY

```

def plot_eog_lr(data):
    """
        Функция для построения графиков горизонтального и вертикального
        движения глаз для обоих глаз.

        Args:
            data (pandas.DataFrame): Данные окулограмм для построения графиков.

        Returns:
            None
    """
    plt.figure(figsize=(12, 8))

```

```

plt.subplot(2, 2, 1)
plt.plot(data['LX'], data['LY'], label='Левый глаз')
plt.title('Левый глаз EOG')
plt.xlabel('LX')
plt.ylabel('LY')
plt.legend()

plt.xlim([min(data['LX'].min(), data['RX'].min()),
max(data['LX'].max(), data['RX'].max())])
plt.ylim([min(data['LY'].min(), data['RY'].min()),
max(data['LY'].max(), data['RY'].max())])

plt.subplot(2, 2, 2)
plt.plot(data['RX'], data['RY'], label='Правый глаз', color='orange')
plt.title('Правый глаз EOG')
plt.xlabel('RX')
plt.ylabel('RY')
plt.legend()

plt.xlim([min(data['LX'].min(), data['RX'].min()),
max(data['LX'].max(), data['RX'].max())])
plt.ylim([min(data['LY'].min(), data['RY'].min()),
max(data['LY'].max(), data['RY'].max())])

plt.tight_layout()
plt.show()

```

ПРИЛОЖЕНИЕ А5. ГРАФИК EOG ЗАВИСИМОСТЕЙ LX ОТ LY И RX ОТ RY С НАЛОЖЕНИЕМ ДРУГ НА ДРУГА

```

def plot_eog_lr_union(data):
    """
    Функция для построения графиков горизонтального и вертикального
    движения глаз для обоих глаз на одном графике.

    Args:
        data (pandas.DataFrame): Данные окулограмм для построения графиков.

    Returns:

```

```

None
"""
plt.figure(figsize=(9, 8))

plt.subplot(2, 2, 1)
plt.plot(data['LX'], data['LY'], label='Левый глаз')
plt.title('Левый глаз EOG')
plt.xlabel('LX')
plt.ylabel('LY')
plt.legend()

plt.subplot(2, 2, 1)
plt.plot(data['RX'], data['RY'], label='Правый глаз', color='orange')
plt.title('Оба глаза EOG')
plt.xlabel('RX')
plt.ylabel('RY')
plt.legend()

plt.tight_layout()
plt.show()

```

ПРИЛОЖЕНИЕ А6. ГРАФИК EOG ЗАВИСИМОСТЕЙ LX ОТ LY И RX ОТ RY С НАЛОЖЕНИЕМ ДРУГ НА ДРУГА И ОТОБРАЖЕНИЕМ КВАНТИЛЕЙ

```

# @TITLE ВИЗУАЛИЗАЦИЯ СТАТИСТИК ОБОИХ ГЛАЗ - PLOT_STATS_XY
DEF PLOT_STATS_XY(CHANGE_DATA, INITIAL_DATA, SLICE_LEN=0):
    """
    ФУНКЦИЯ ДЛЯ ПОСТРОЕНИЯ ГРАФИКОВ СТАТИСТИЧЕСКИХ ЗНАЧЕНИЙ X И Y ДЛЯ ОБОИХ
    ГЛАЗ.

    ARGS:
    CHANGE_DATA (PANDAS.DATAFRAME): ИЗМЕНЕННЫЕ ДАННЫЕ ОКУЛОГРАММ.
    INITIAL_DATA (PANDAS.DATAFRAME): ИСХОДНЫЕ ДАННЫЕ ОКУЛОГРАММ.
    SLICE_LEN (INT, OPTIONAL): ДЛИНА СРЕЗА ДАННЫХ ДЛЯ ОТОБРАЖЕНИЯ. ПО
    УМОЛЧАНИЮ 0.

    RETURNS:
    NONE

```

```

"""
DEF PLOT_STATS(STATS, LABEL_PREFIX):
    PLT.PLOT(STATS.LOC['MIN'][0], STATS.LOC['MIN'][1], 'RO',
LABEL=LABEL_PREFIX + ' MIN')
    PLT.PLOT(STATS.LOC['25%'][0], STATS.LOC['25%'][1], 'BO',
LABEL=LABEL_PREFIX + ' 25%')
    PLT.PLOT(STATS.LOC['50%'][0], STATS.LOC['50%'][1], 'GO',
LABEL=LABEL_PREFIX + ' MEDIAN')
    PLT.PLOT(STATS.LOC['75%'][0], STATS.LOC['75%'][1], 'YO',
LABEL=LABEL_PREFIX + ' 75%')
    PLT.PLOT(STATS.LOC['MAX'][0], STATS.LOC['MAX'][1], 'RO',
LABEL=LABEL_PREFIX + ' MAX')

    PLT.PLOT([STATS.LOC['MIN'][0], STATS.LOC['MIN'][0], 0],
[0, STATS.LOC['MIN'][1], STATS.LOC['MIN'][1]], LINESTYLE='--',
COLOR='GREY')

    PLT.PLOT([STATS.LOC['25%'][0], STATS.LOC['25%'][0], 0],
[0, STATS.LOC['25%'][1], STATS.LOC['25%'][1]], LINESTYLE='--',
COLOR='GREY')

    PLT.PLOT([0, STATS.LOC['50%'][0], STATS.LOC['50%'][0]],
[STATS.LOC['50%'][1], STATS.LOC['50%'][1], 0], LINESTYLE='--',
COLOR='GREY')

    PLT.PLOT([STATS.LOC['75%'][0], STATS.LOC['75%'][0], 0],
[0, STATS.LOC['75%'][1], STATS.LOC['75%'][1]], LINESTYLE='--',
COLOR='GREY')

    PLT.PLOT([0, STATS.LOC['MAX'][0], STATS.LOC['MAX'][0]],
[STATS.LOC['MAX'][1], STATS.LOC['MAX'][1], 0], LINESTYLE='--',
COLOR='GREY')

    PLT.FIGURE(FIGSIZE=(14, 6))
    CHANGE_DATA_LEN = SLICE_LEN IF SLICE_LEN ELSE LEN(CHANGE_DATA)

    PLT.PLOT(CHANGE_DATA['LX'][:CHANGE_DATA_LEN],
CHANGE_DATA['LY'][:CHANGE_DATA_LEN], LABEL='ЛЕВЫЙ ГЛАЗ', ALPHA=0.3)
    PLT.PLOT(CHANGE_DATA['RX'][:CHANGE_DATA_LEN],
CHANGE_DATA['RY'][:CHANGE_DATA_LEN], LABEL='ПРАВЫЙ ГЛАЗ', ALPHA=0.3)

```

```

STATS = INITIAL_DATA[['LX', 'LY']].DESCRIBE().LOC['MIN': 'MAX']
PLOT_STATS(STATS, 'L')

STATS = INITIAL_DATA[['RX', 'RY']].DESCRIBE().LOC['MIN': 'MAX']
PLOT_STATS(STATS, 'R')

PLT.XLABEL('STATISTICAL VALUES')
PLT.YLABEL('Y')
PLT.TITLE('MIN, 25%, 50%, 75%, MAX OF X AND Y')
PLT.LEGEND()

PLT.AXHLINE(0, COLOR='BLACK', LINEWIDTH=1.5)
PLT.AXVLINE(0, COLOR='BLACK', LINEWIDTH=1.5)

PLT.MINORTICKS_ON()
PLT.XLABEL(R'$X$', FONTSIZE=16)

PLT.GRID(WHICH='MAJOR')
PLT.GRID(WHICH='MINOR', LINESTYLE=':')
PLT.TIGHT_LAYOUT()

PLT.SHOW()

```

ПРИЛОЖЕНИЕ А7. ГРАФИК 3D ВИЗУАЛИЗАЦИЯ ОСЕВЫХ ПРИЗНАКОВ В ЗАВИСИМОСТИ ОТ ПРИЗНАКА Т

```

def plotly_3d(data, axes=['T', 'LX', 'LY'], color='T'):
    """
    Функция для построения интерактивного трехмерного графика.

    Args:
    data (pandas.DataFrame): Данные для построения графика.
    axes (list, optional): Список осей. По умолчанию ['T', 'LX', 'LY'].
    color (str, optional): Параметр цвета. По умолчанию 'T'.

    Returns:
    None
    """
    axis_x, axis_y, axis_z = axes
    color_map = {'event': 'red', 'other': 'orange', 'transition': 'blue'}

```

```

fig = go.Figure(data=[go.Scatter3d(
    x=data[axis_x],
    y=data[axis_y],
    z=data[axis_z],
    mode='markers',
    marker=dict(
        size=3,
        color=[color_map[val] for val in data[color]] if color != 'T'
else data[color],
        colorscale='Viridis',
        opacity=0.8
    ),
    text=[f'X: {lx}, Y: {ly}, T: {t}' for lx, ly, t in
zip(data[axes[1]], data[axes[2]], data[axes[0]])],
    hoverinfo='text'
)])

# Создаем данные для плоскости
xx, yy = np.meshgrid(np.linspace(data[axes[1]].min(),
data[axes[1]].max(), 10),
                    np.linspace(data[axes[2]].min(),
data[axes[2]].max(), 10))
zz = np.zeros_like(xx)

offset = -0.01

# Добавляем плоскость
fig.add_trace(go.Surface(x=zz + offset, y=xx, z=yy, opacity=0.05,
hoverinfo='none'))

fig.update_layout(scene=dict(
    xaxis_title=axes[1],
    yaxis_title=axes[0],
    zaxis_title=axes[2],
    xaxis_visible=False, # убираем ось
    yaxis_visible=False,
    zaxis_visible=False,
    # xaxis=dict(range=[offset, 1], autorange=False) # Отключение
автоматического масштабирования для оси z)
),

```

```

        title='3D Visualization',
        width=1200,
        height=800)

fig.show()

```

ПРИЛОЖЕНИЕ А8. РЕАЛИЗАЦИЯ АЛГОРИТМА IVT

```

def add_event_features_ivt(data, velocity_threshold):
    """
        Рассчитывает скорость между парами точек и определяет фиксации и
        саккады на основе порога скорости.

        Параметры:
            data (pandas.DataFrame): Входные данные, содержащие столбцы 'X_mean',
            'Y_mean' и 'T'.
            velocity_threshold (float): Порог скорости для различения фиксаций и
            саккад.

        Возвращает:
            pandas.DataFrame: Входные данные с добавленными столбцами 'velocity'
            и 'IVT'.

        Примечания:
            - Столбец 'velocity' вычисляется как евклидово расстояние между
            последовательными точками, разделенное на квадрат временного интервала.
            - Столбец 'IVT' помечается как 'фиксация', если скорость находится
            ниже порога, и 'саккада' в противном случае.
    """
    # Рассчитываем скорость между каждой парой точек
    data['velocity'] = np.sqrt((data['X_mean'].diff() ** 2 +
    data['Y_mean'].diff() ** 2) / (data['T'].diff() ** 2)).fillna(0)

    # Метки фиксации и саккады на основе порога скорости
    data['IVT'] = np.where(data['velocity'] <= velocity_threshold,
    'fixation', 'saccade')

    return data

```

ПРИЛОЖЕНИЕ А9. ГРАФИК МОМЕНТОВ ГЛАЗОДВИГАТЕЛЬНОЙ АКТИВНОСТИ

```
def plot_events(data, methods):  
    """  
    Рисует диаграммы событий для каждого метода.  
  
    Параметры:  
    data (pandas.DataFrame): Входные данные, содержащие столбцы 'T',  
    'X_mean', 'Y_mean', и дополнительные столбцы для каждого метода.  
    methods (list): Список методов, для которых необходимо нарисовать  
    диаграммы событий.  
  
    Возвращает:  
    None  
  
    Примечания:  
    - Для каждого метода в methods создается отдельная диаграмма,  
    содержащая временную шкалу и события 'X_mean' и 'Y_mean'.  
    - Для каждого события в данных заполняется вертикальная область на  
    диаграмме с соответствующим цветом и прозрачностью.  
    """  
    colors = {  
        'fixation': 'lightgray',  
        'saccade': 'lightgreen',  
        'peripheral': 'orange',  
        'foveal': 'm',  
        'parafoveal': 'c',  
        'other': 'b',  
        'distortions': 'black',  
        'fixation_progressive': 'lightgray',  
        'fixation_regressive': 'lightgray',  
        'saccade_progressive': 'lightgreen',  
        'saccade_regressive': 'maroon',  
        'transition': 'blue',  
        'event': 'red',  
    }  
  
    alphas = {
```

```

        'fixation': 0.05,
        'saccade': 0.5,
        'peripheral': 1,
        'foveal': 1,
        'parafoveal': 1,
        'other': 1,
        'distortions': 0.8,
        'fixation_progressive': 0.05,
        'fixation_regressive': 0.1,
        'saccade_progressive': 0.5,
        'saccade_regressive': 0.05,
        'transition': 0.5,
        'event': 0.5
    }

    fig, axes = plt.subplots(len(methods), 1, figsize=(24, 5 *
len(methods)))

    for i, method in enumerate(methods):
        ax = axes[i] if len(methods) > 1 else axes
        ax.plot(data['T'], data['X_mean'], label='X_mean', alpha=1)
        ax.plot(data['T'], data['Y_mean'], label='Y_mean', alpha=1)

        ax.set_xlabel('Time')
        ax.set_ylabel(f'{method}_events')
        ax.set_title(f'{method}_events by Time')

        for j, row in data.iterrows():
            time = row['T']
            event_type = row[method]
            color = colors.get(event_type, 'none')
            alpha = alphas.get(event_type, 0.5)

            ax.axvspan(time, time, color=color, alpha=alpha)

        ax.legend(loc="upper right")

    plt.show()

```

ПРИЛОЖЕНИЕ А10. РЕАЛИЗАЦИЯ АЛГОРИТМА IDT

```
def add_event_features_idt(data, window_size=50,
dispersion_threshold=0.5):
    """
        Рассчитывает дисперсию для окна размером window_size и определяет
        фиксации и саккады на основе порога дисперсии.

        Параметры:
        data (pandas.DataFrame): Входные данные, содержащие столбцы 'X_mean'
        и 'Y_mean'.
        window_size (int): Размер окна для расчета дисперсии. Значение по
        умолчанию - 50.
        dispersion_threshold (float): Порог дисперсии для различения фиксаций
        и саккад. Значение по умолчанию - 0.5.

        Возвращает:
        pandas.DataFrame: Входные данные с добавленным столбцом 'IDT'.

        Примечания:
        - Для каждой точки в данных создается окно размером window_size, и
        рассчитывается дисперсия для каждого измерения.
        - Если дисперсия для какого-либо измерения превышает порог дисперсии,
        точка помечается как 'саккада'.
        - Если дисперсия для обоих измерений не превышает порог дисперсии,
        точка помечается как 'фиксация'.
    """
    for i, row in data.iterrows():
        window_start = max(data.index.min(), i - int(window_size / 2))
        window_end = min(data.index.max(), i + int(window_size / 2))

        x_window_values = data.loc[window_start:window_end,
        'X_mean'].values
        y_window_values = data.loc[window_start:window_end,
        'Y_mean'].values

        x_dispersion = np.var(x_window_values)
        y_dispersion = np.var(y_window_values)
```

```

        if dispersion_threshold < x_dispersion or dispersion_threshold <
y_dispersion:
            data.loc[i, 'IDT'] = 'saccade'

        elif x_dispersion < dispersion_threshold and y_dispersion <
dispersion_threshold:
            data.loc[i, 'IDT'] = 'fixation'

    return data

```

ПРИЛОЖЕНИЕ А11. ИЗВЛЕЧЕНИЕ ЗНАКА ПРОИЗВОДНОЙ

```

def get_sign(data, columns):
    """
    Рассчитывает производную и знак для указанных столбцов данных.

    Параметры:
    data (pandas.DataFrame): Входные данные, содержащие столбцы для
расчета производной и знака.
    columns (list): Список имен столбцов для расчета производной и знака.

    Возвращает:
    pandas.DataFrame: Входные данные с добавленными столбцами
'_derivative' и '_sign' для каждого столбца в columns.

    Примечания:
    - Для каждого столбца в columns рассчитывается производная по времени
'T' с помощью функции numpy.gradient.
    - Затем для каждого столбца в columns рассчитывается знак с помощью
функции numpy.sign.
    - Результат сохраняется в новых столбцах '_derivative' и '_sign'
соответственно.
    """
    for column in columns:
        data[column + '_derivative'] = np.gradient(data[column],
data['T'])
        data[column + '_sign'] = np.sign(data[column + '_derivative'])

```

```
return data
```

ПРИЛОЖЕНИЕ А12. МОДИФИЦИРОВАННЫЙ АЛГОРИТМ IDT

```
def process_window(data, window_size, dispersion_threshold_prog,
dispersion_threshold_reg, prefix):
    """
    Определяет тип события для окна размером window_size на основе порогов
    дисперсии и знака.

    Параметры:
    data (pandas.DataFrame): Входные данные, содержащие столбцы 'X_mean',
    'Y_mean', 'X_mean_sign', 'Y_mean_sign', 'LX', 'LY', 'RX', 'RY' и 'T'.
    window_size (int): Размер окна для расчета дисперсии.
    dispersion_threshold_prog (float): Порог дисперсии для прогрессивных
    событий.
    dispersion_threshold_reg (float): Порог дисперсии для регрессивных
    событий.
    prefix (str): Префикс для имени столбца события.

    Возвращает:
    pandas.DataFrame: Входные данные с добавленным столбцом события с
    префиксом prefix.

    Примечания:
    - Для каждой точки в данных создается окно размером window_size, и
    рассчитывается дисперсия для каждого измерения.
    - Затем проверяется знак производной в конце окна и сравнивается с
    порогом дисперсии для определения типа события.
    - Результат сохраняется в новом столбце события с префиксом prefix.
    """
    for i, row in (data if prefix == 'NE' else data[:::-1]).iterrows():
        window_start = max(data.index.min(), i - int(window_size / 2))
        window_end = min(data.index.max(), i + int(window_size / 2))

        x_window_values = data.loc[window_start:window_end,
        'X_mean'].values
        y_window_values = data.loc[window_start:window_end,
        'Y_mean'].values
```

```

x_dispersion = np.var(x_window_values)
y_dispersion = np.var(y_window_values)

invert_sign = -1 if prefix == 'EN' else 1

if row['LX'] == 0 and row['RX'] == 0 or row['LY'] == 0 and
row['RY'] == 0:
    data.loc[i, f'{prefix}_event'] = 'other' # distortions

    elif dispersion_threshold_prog < x_dispersion and
data.loc>window_end, 'X_mean_sign'] * invert_sign > 0 or \
        dispersion_threshold_prog < y_dispersion and
data.loc>window_end, 'Y_mean_sign'] * invert_sign > 0:
        data.loc[i, f'{prefix}_event'] = 'saccade_progressive' # i

    elif dispersion_threshold_reg < x_dispersion and
data.loc>window_end, 'X_mean_sign'] * invert_sign < 0 or \
        dispersion_threshold_reg < y_dispersion and
data.loc>window_end, 'Y_mean_sign'] * invert_sign < 0:
        data.loc[i, f'{prefix}_event'] = 'saccade_regressive' # i

    elif x_dispersion < dispersion_threshold_prog and
data.loc>window_end, 'X_mean_sign'] * invert_sign > 0 or \
        y_dispersion < dispersion_threshold_prog and
data.loc>window_end, 'Y_mean_sign'] * invert_sign > 0:
        data.loc[i, f'{prefix}_event'] = 'fixation_progressive'

    elif x_dispersion < dispersion_threshold_reg and
data.loc>window_end, 'X_mean_sign'] * invert_sign < 0 or \
        y_dispersion < dispersion_threshold_reg and
data.loc>window_end, 'Y_mean_sign'] * invert_sign < 0:
        data.loc[i, f'{prefix}_event'] = 'fixation_regressive'
    else:
        data.loc[i, f'{prefix}_event'] = 'other'

return data

def add_event_features_idt_mod(data, window_size = 50,
dispersion_threshold_prog = 0.5, dispersion_threshold_reg = 0.5):

```

```

        data = process_window(data, window_size, dispersion_threshold_prog,
dispersion_threshold_reg, 'NE')
        data = process_window(data, window_size, dispersion_threshold_prog,
dispersion_threshold_reg, 'EN')

        data['MEAN_event'] = data.apply(lambda row: row['EN_event'] if
row['NE_event'] == row['EN_event'] else 'unknown', axis=1)

    return data

```

ПРИЛОЖЕНИЕ A13. АЛГОРИТМ ТРАНЗИТИВНЫХ СОБЫТИЙ

```

def add_event_features_transition(data, windows_size_rms,
distance_threshold, rms_threshold_factor):
    """
    Определяет тип события для окна размером windows_size_rms на основе
    среднеквадратичной ошибки RMS и расстояния.

    Параметры:
    data (pandas.DataFrame): Входные данные, содержащие столбцы 'X_mean'
и 'Y_mean'.
    windows_size_rms (int): Размер окна для расчета среднеквадратичной
ошибки RMS.
    distance_threshold (float): Порог расстояния для определения события.
    rms_threshold_factor (float): Коэффициент для умножения на
среднеквадратичную ошибку RMS.

    Возвращает:
    pandas.DataFrame: Входные данные с добавленным столбцом события
'TRANS'.

    Примечания:
    - Для каждой точки в данных создается окно размером windows_size_rms,
и рассчитывается среднеквадратичная ошибка RMS.
    - Затем проверяется расстояние между текущим значением и средним
значением окна, и сравнивается с порогом расстояния и среднеквадратичной ошибкой
RMS.
    - Результат сохраняется в новом столбце события 'TRANS'.
    """

```

```

data['TRANS'] = 'other'

# Вычисление среднеквадратичной ошибки RMS
rms_errors = data[['X_mean',
'Y_mean']].rolling(window=windows_size_rms).std()
rms_error = np.sqrt(rms_errors.mean(axis=1).iloc[windows_size_rms-
1:])

# Инициализация переменных состояния
current_state_rx = []
current_state_ry = []

for i, row in data.iterrows():
    # Вычисление средних значений текущего состояния
    current_state_rx.append(row['X_mean'])
    current_state_ry.append(row['Y_mean'])

    if len(current_state_rx) > windows_size_rms:
        current_state_rx.pop(0)
        current_state_ry.pop(0)

    current_state_rx_avg = np.mean(current_state_rx)
    current_state_ry_avg = np.mean(current_state_ry)

    # Проверка, является ли текущий образец временным состоянием
    if current_state_rx_avg is not None and current_state_ry_avg is
not None:
        if i >= min(rms_error.index): # Проверка, что индекс
существует в массиве rms_error
            if (np.abs(row['X_mean'] - current_state_rx_avg) >
distance_threshold + rms_threshold_factor * rms_error[i]) and \
                (np.abs(row['Y_mean'] - current_state_ry_avg) >
distance_threshold + rms_threshold_factor * rms_error[i]):
                data.at[i, 'TRANS'] = 'event'

return data

```

ПРИЛОЖЕНИЕ A14. МЕТОД DBSCAN

```
def find_fixations_DBSCAN(data, axis, prefix, lambda_func, eps=20,
min_samples=5):
    """
    Метод для поиска глазодвигательных моментов с использованием DBSCAN.

    Параметры:
    - data: DataFrame, содержащий данные окулографии.
    - axis: Список с осями по которым происходит кластеризация
    - prefix: Тип поиска аномалий (Взгляд в точку и другие события,
    саккады и другие события, торзионные и другие события)
    - lambda_func: Лямбда функция по которой обрабатываются полученные
    аномалии из метода DBSCAN
    - eps: радиус окрестности для определения соседей в DBSCAN.
    - min_samples: минимальное количество точек в окрестности, необходимое
    для формирования кластера в DBSCAN.

    Возвращает:
    - data: DataFrame с добавленным столбцом 'DBSCAN_{prefix}',
    указывающим принадлежность к заданному типу событий
    """

    # Преобразование данных в двумерный массив (без временной метки)
    data_xy = data[axis].values
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    data[f'DBSCAN_{prefix}'] = dbscan.fit_predict(data_xy)
    data[f'DBSCAN_{prefix}'] = data[f'DBSCAN_{prefix}'].apply(lambda_func)
    return data
```

ПРИЛОЖЕНИЕ A15. ГРАФИК РАСПРЕДЕЛЕНИЯ КЛАСТЕРОВ

```
def plot_clusters(data, method, axis):
    """
    Метод для визуализации результатов кластеризации методом DBSCAN.

    Параметры:
    - data: DataFrame, содержащий данные окулографии.
```

```

- labels: массив меток кластеров для каждой точки данных.
"""
cmap = {'event': 'red', 'other': 'orange', 'transition': 'blue'}

# Отображение кластеров
plt.figure(figsize=(12, 6))
plt.scatter(data[axis[0]], data[axis[1]], c=[cmap[label] for label in
data[method]], cmap='viridis')
plt.title(f'{method} Clustering')
plt.xlabel(axis[0])
plt.ylabel(axis[1])
plt.colorbar(label='Cluster Label')
plt.show()

```

ПРИЛОЖЕНИЕ A16. МЕТОД HDBSCAN

```

def find_fixations_hdbscan(data, axis, min_cluster_size, min_samples):
    """
        Метод для поиска аномалий, которые можно интерпретировать как
        глазодвигательные события с использованием OPTHDBSCANICS.

        Параметры:
        - data: DataFrame, содержащий данные окулографии.
        - eps: максимальное расстояние для объединения точек в один кластер.
        - min_samples: минимальное количество точек, необходимое для
        формирования кластера.

        Возвращает:
        - data: DataFrame с добавленным столбцом 'HDBSCAN', указывающим
        принадлежность к евенту или другим событиям.
    """
    # Преобразование данных в двумерный массив (без временной метки)
    data_xy = data[axis].values
    clusterer = hdbscan.HDBSCAN(min_cluster_size=min_cluster_size,
min_samples=min_samples) # , metric='euclidean'
    clusterer.fit(data_xy)
    data['HDBSCAN'] = clusterer.labels_
    return data

```

ПРИЛОЖЕНИЕ A17. МЕТОД LOF

```
def find_fixations_lof(data, axis, contamination=0.05):  
    """  
        Метод для поиска глазодвигательных моментов с использованием Local  
        Outlier Factor (LOF).  
  
        Параметры:  
        - data: DataFrame, содержащий данные окулографии.  
        - contamination: доля выбросов в данных.  
  
        Возвращает:  
        - data: DataFrame с добавленным столбцом 'LOF', указывающим  
        принадлежность к фиксации или саккаду.  
    """  
    # Преобразование данных в двумерный массив (без временной метки)  
    data_xy = data[axis].values  
    # Выполнение Local Outlier Factor  
    lof = LocalOutlierFactor(contamination=contamination)  
    lof.fit_predict(data_xy)  
    # Добавление меток выбросов в DataFrame  
    data['LOF'] = lof.fit_predict(data_xy)  
    return data
```

ПРИЛОЖЕНИЕ A18. МЕТОД ИЗВЛЕЧЕНИЯ ПОЛЕЙ ЗРЕНИЯ (ФОВЕАЛЬНОЕ, ПАРОФОВЕАЛЬНОЕ И ДРУГИЕ)

```
def get_foveal_parafoveal(data,  
                           window_size=1,  
                           radius_foveal=0.02,  
                           radius_parafoveal=0.1,  
                           radius_peripheral=0.6):  
    """  
        Функция для классификации каждой точки в окулографических данных как  
        foveal, parafoveal, peripheral или other.  
  
        Параметры:
```

```

- data (pd.DataFrame): Окулографические данные с колонками 'LX', 'LY',
'RX', 'RY'.
- window_size (int): Размер окна для вычисления средних значений
координат точек. По умолчанию 1.
- radius_foveal (float): Радиус фовеального поля зрения. По умолчанию
0.02.
- radius_parafoveal (float): Радиус парафовеального поля зрения. По
умолчанию 0.1.
- radius_peripheral (float): Радиус периферического поля зрения. По
умолчанию 0.6.

Возвращает:
- pd.DataFrame: Оригинальные окулографические данные с добавленными
колонками 'X_mean_FV', 'Y_mean_FV' и 'FV'.
"""

# Функция для вычисления расстояния между двумя точками
def distance(x1, y1, x2, y2):
    return np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

# Функция для вычисления типа поля зрения
def get_field_vision(L_point_distance, R_point_distance):
    if L_point_distance <= radius_foveal and R_point_distance <=
radius_foveal:
        return 'foveal' # Фовеальное

    elif radius_foveal < L_point_distance <= radius_parafoveal and
radius_foveal < R_point_distance <= radius_parafoveal:
        return 'parafoveal' # Парафовеальное

    elif radius_parafoveal < L_point_distance <= radius_peripheral
and radius_parafoveal < R_point_distance <= radius_peripheral:
        return 'peripheral' # Периферическое

    else:
        return 'other' # Иное

window_size_str = str(window_size)

for i, row in data.iterrows():

```

```

        # Если окно присутствует, вычисляем среднее значение координат
точек в окне
        window_start = max(data.index.min(), i - int(window_size / 2))
        window_end = min(data.index.max() + 1, i + int(window_size / 2))

        LX_mean = data.loc[window_start:window_end, 'LX'].mean() # loc по
5 строк
        LY_mean = data.loc[window_start:window_end, 'LY'].mean()
        RX_mean = data.loc[window_start:window_end, 'RX'].mean()
        RY_mean = data.loc[window_start:window_end, 'RY'].mean()

        X_mean_window = (LX_mean + RX_mean) / 2
        Y_mean_window = (LY_mean + RY_mean) / 2

        data.loc[i, 'X_mean_FV'] = X_mean_window
        data.loc[i, 'Y_mean_FV'] = Y_mean_window

        # Расстояние между средними точками и текущей точкой
        L_point_distance = distance(X_mean_window, Y_mean_window,
row['LX'], row['LY'])
        R_point_distance = distance(X_mean_window, Y_mean_window,
row['RX'], row['RY'])

        # Вычисление поля зрения
        data.loc[i, 'FV'] = get_field_vision(L_point_distance,
R_point_distance)

    return data

```

ПРИЛОЖЕНИЕ А19. ГРАФИК ТИПОВ ЗРЕНИЯ

```

# @title Визуализация типов зрения - plot_field_view
def plot_field_view(data):
    """
    Рисует график поля зрения окулографических данных.

    Параметры:
    - data (pd.DataFrame): Окулографические данные с колонками 'FV', 'LX',
'LY'.

```

Возвращает:

- None: Отображает график поля зрения.

"""

```
plt.figure(figsize=(14, 6))
```

```
#####
```

```
categories = {'other': {'color': 'b', 'alpha': 0.1, 'size': 10},  
              'peripheral': {'color': 'orange', 'alpha': 0.1, 'size':
```

10},

```
              'parafoveal': {'color': 'c', 'alpha': 1, 'size': 10},
```

```
              'foveal': {'color': 'm', 'alpha': 1, 'size': 0.2}}
```

```
for category, props in categories.items():
```

```
    for side in ['L', 'R']:
```

```
        plt.scatter(data[data['FV'] == category][f'LX'],
```

```
                    data[data['FV'] == category][f'LY'],
```

```
                    s=props['size'],
```

```
                    alpha=props['alpha'],
```

```
                    color=props['color'],
```

```
                    label=f'FV_{side}_{category}')
```

```
#####
```

```
plt.xlabel('Statistical Values')
```

```
plt.ylabel('Y')
```

```
plt.title('Field of vision')
```

```
plt.legend()
```

```
plt.axhline(0, color='black', linewidth=1.5)
```

```
plt.axvline(0, color='black', linewidth=1.5)
```

```
plt.minorticks_on()
```

```
plt.xlabel(r'$x$', fontsize=16)
```

```
plt.grid(which='major')
```

```
plt.grid(which='minor', linestyle=':')
```

```
plt.tight_layout()
```

```
plt.show()
```

ПРИЛОЖЕНИЕ A20. ПОИСК МАТЕМАТИЧЕСКИХ ПРИЗНАКОВ

```
def add_math_features(data, window_sizes=[5, 10, 20]):  
    """  
    Добавляет математические признаки к окулографическим данным.  
  
    Параметры:  
    - data (pd.DataFrame): Окулографические данные с колонками 'LX', 'LY',  
    'RX', 'RY', 'T'.  
    - window_sizes (list): Размеры окон для вычисления признаков. По  
    умолчанию [5, 10, 20].  
  
    Возвращает:  
    - pd.DataFrame: Оригинальные окулографические данные с добавленными  
    математическими признаками.  
    """  
    eye_prefixes = ['L', 'R']  
    axes_prefixes = ['X', 'Y']  
  
    window_sizes_str = str(window_sizes)  
  
    for axes_prefixe in axes_prefixes:  
        # Абсолютная разница  
        data[axes_prefixe + '_abs'] = abs(data[eye_prefixes[0] +  
axes_prefixe] - data[eye_prefixes[1] + axes_prefixe])  
  
        for window_size in window_sizes:  
            # скользящая среднее текущей позиции (Среднее с окном в 5 для  
того что бы было 50мс)  
            data[axes_prefixe + '_mean_std_' + window_sizes_str] =  
data[axes_prefixe + '_mean'].rolling(window=window_size).std().fillna(0)  
            # скользящая дисперсия текущей позиции  
            data[axes_prefixe + '_mean_var_' + window_sizes_str] =  
data[axes_prefixe + '_mean'].rolling(window=window_size).var().fillna(0)  
  
        for eye_prefixe in eye_prefixes:  
            # Абсолютная разница между двумя соседними горизонтальными  
координатами левого глаза  
            data[eye_prefixe + axes_prefixe + '_ch_abs'] = data[eye_prefixe  
+ axes_prefixe].diff().abs().fillna(0)
```

```

# Признаки со скользящими окнами
for window_size in window_sizes:
    #RMS для обоих глаз с окном
    data[eye_prefixe + axes_prefixe + '_RMS_' +
window_sizes_str] = data[eye_prefixe +
axes_prefixe].rolling(window=window_size).apply(lambda x:
np.sqrt(np.mean(np.square(x)))) .fillna(0)

    # Разница между среднеквадратичными ошибками, рассчитанными
на окне до и после текущей точки
    data[eye_prefixe + axes_prefixe + '_MSE_before_' +
window_sizes_str] = (data[eye_prefixe +
axes_prefixe].rolling(window=window_size).mean()*2).fillna(0)
    data[eye_prefixe + axes_prefixe + '_MSE_after_' +
window_sizes_str] = (data[eye_prefixe +
axes_prefixe].rolling(window=window_size).mean().shift(-
window_size)*2).fillna(0)
    data[eye_prefixe + axes_prefixe + '_MSE_diff_' +
window_sizes_str] = data[eye_prefixe + axes_prefixe + '_MSE_after_' +
window_sizes_str] - data[eye_prefixe + axes_prefixe + '_MSE_before_' +
window_sizes_str]

    # скользящие среднее
    data[eye_prefixe + axes_prefixe + '_std_' +
window_sizes_str] = data[eye_prefixe +
axes_prefixe].rolling(window=window_size).std().fillna(0)

    # скользящая дисперсия
    data[eye_prefixe + axes_prefixe + '_var_' +
window_sizes_str] = data[eye_prefixe +
axes_prefixe].rolling(window=window_size).var().fillna(0)

for eye_prefixe in eye_prefixes:
    # Вычисляем мгновенную скорость
    data[eye_prefixe + '_speed'] = (np.sqrt(np.square(data[eye_prefixe
+ axes_prefixes[0]].diff()) + np.square(data[eye_prefixe +
axes_prefixes[1]].diff())) / data['T'].diff()).fillna(0)

    # Вычисляем мгновенное ускорение
    data[eye_prefixe + '_acceleration'] = ((data[eye_prefixe +
'_speed'] - data[eye_prefixe + '_speed'].shift(1)) /
data['T'].diff()).fillna(0)

```

```
return data
```

ПРИЛОЖЕНИЕ А21. ГРАФИК КОЛИЧЕСТВЕННОГО РАСПРЕДЕЛЕНИЕ

```
def get_grouped_data(data, column):  
    """  
    Группирует данные по столбцу 'disabled' и считает значения указанного  
    столбца.  
  
    Параметры:  
    data (pandas.DataFrame): Входные данные  
    column (str): Столбец для группировки и подсчета  
  
    Возвращает:  
    None  
  
    Строит диаграмму с распределением указанного столбца по статусу  
    'disabled'.  
  
    Примечания:  
    Эта функция использует библиотеки pandas и matplotlib.  
    """  
    grouped_data = data.groupby('disabled')[column].value_counts()  
  
    # Отрисовываем распределение признака DB  
    grouped_data.plot(kind='bar', x='disabled', y=column,  
title=f'Распределение {column} по disabled', figsize=(10, 5))  
    plt.show()
```

ПРИЛОЖЕНИЕ A22. СВЕРТКА ОБЩЕГО НАБОРА ДАННЫХ В ИТоговый на основе количественных показателей

```
def get_duration(column: pd.Series, unique_value: any) -> float:
    """
    Рассчитывает среднюю длительность уникального значения в столбце.

    Параметры:
        column (pandas.Series): Столбец, для которого необходимо рассчитать
        длительность.
        unique_value (any): Уникальное значение, для которого необходимо
        рассчитать длительность.

    Возвращает:
        float: Средняя длительность уникального значения.

    Примечания:
        Эта функция использует библиотеку pandas.
    """
    # Создаем серию, в которой 1 для каждого нового значения в столбце
    'LOF' и 0 в противном случае
    is_new_group = column != column.shift(1)

    # Используем cumsum для накопления значений, чтобы получить уникальный
    идентификатор группы
    group_ids = is_new_group.cumsum()

    # Группируем по типу и по идентификаторам группы, затем считаем
    количество в каждой группе
    counts = group_ids.groupby([column, group_ids]).size()

    return counts[unique_value].mean()

def average_by_characteristics(data, event_columns):
    """
    Рассчитывает средние значения для указанных столбцов по группам
    уникальных значений других столбцов.

    Параметры:
```

data (pandas.DataFrame): Входные данные

event_columns (list): Список столбцов для расчета средних значений

Возвращает:

pandas.DataFrame: Новые данные с рассчитанными средними значениями

Примечания:

Эта функция использует библиотеку pandas.

```
"""
```

```
new_data = data.drop(columns=event_columns).mean()
```

```
for column in event_columns:
```

```
    value_counts = data[column].value_counts()
```

```
    for unique_value, count in value_counts.items():
```

```
        new_data[column + '_' + str(unique_value) + '_count'] = count
```

```
        new_data[column + '_' + str(unique_value) + '_mean_duration']
```

```
= get_duration(data[column], unique_value)
```

```
# Добавляем ключ группировки к новым данным
```

```
new_data['id_patient'] = data.name
```

```
return pd.DataFrame(new_data).T
```

```
prefixs = [
```

```
    'FV',
```

```
    'IVT',
```

```
    'IDT',
```

```
    'HDBSCAN',
```

```
    "LOF",
```

```
    'NE_event',
```

```
    'EN_event',
```

```
    'MEAN_event',
```

```
    'TRANS',
```

```
    'DBSCAN_P',
```

```
    'DBSCAN_FS',
```

```
    'DBSCAN_T',
```

```
    'HDBSCAN',
```

```
    'LOF'
```

```
]
```

```

df_grouped = DF_TOTAL.groupby('id_patient').apply(lambda group:
average_by_characteristics(group, prefixes))
df_scaler = standardize(df_grouped)

df_scaler.to_csv(f'/content/drive/MyDrive/lectures/diplom/df_general/{df
_name}_scaler.csv', index=False)

```

ПРИЛОЖЕНИЕ A23. СВЕРТКА ОБЩЕГО НАБОРА ДАННЫХ В ИТОГОВЫЙ НА ОСНОВЕ ОТНОШЕНИЯ КОЛИЧЕСТВА СОБЫТИЙ К ДЛИТЕЛЬНОСТИ ИСПЫТАНИЯ

```

def average_by_characteristics(data, event_columns):
    """
    Рассчитывает средние значения для указанных столбцов по группам
    уникальных значений других столбцов.

    Параметры:
    data (pandas.DataFrame): Входные данные
    event_columns (list): Список столбцов для расчета средних значений

    Возвращает:
    pandas.DataFrame: Новые данные с рассчитанными средними значениями

    Примечания:
    Эта функция использует библиотеку pandas.
    """
    new_data = data.drop(columns=event_columns).mean()
    new_data['LX_std'] = data['LX'].std()
    new_data['LY_std'] = data['LY'].std()
    new_data['RX_std'] = data['RX'].std()
    new_data['RY_std'] = data['RY'].std()
    new_data['X_mean_std'] = data['X_mean'].std()
    new_data['Y_mean_std'] = data['Y_mean'].std()

    for column in event_columns:
        value_counts = data[column].value_counts()
        for unique_value, count in value_counts.items():
            new_data[column + '_' + str(unique_value) + '_count'] = count
    / len(data)

```

```

        #new_data[column + '_' + str(unique_value) + '_count'] = 0
    if np.isnan(count) else (count / len(data))
        new_data[column + '_' + str(unique_value) + '_mean_duration']
= get_duration(data[column], unique_value)

# Добавляем ключ группировки к новым данным
new_data['id_patient'] = data.name

return pd.DataFrame(new_data).T

```

ПРИЛОЖЕНИЕ А24. ИНТЕРПОЛЯЦИЯ ДАННЫХ

```

def interpolate(data):
    """
    Интерполирует значения данных для каждого признака с помощью функции
    interp1d.

    Параметры:
        data (pd.DataFrame): Входной датафрейм с колонками 'T', 'LX',
        'LY', 'RX', 'RY'.

    Возвращает:
        pd.DataFrame: Выходной датафрейм с интерполированными значениями
    для каждого признака на новом временном диапазоне.
    """
    new_time_range = np.arange(0, len(data) * 20 - 19, 1)

    # Создаем функции интерполяции для каждого признака
    f_LX = interp1d(data['T'], data['LX'])
    f_LY = interp1d(data['T'], data['LY'])
    f_RX = interp1d(data['T'], data['RX'])
    f_RY = interp1d(data['T'], data['RY'])

    # Выполняем интерполяцию
    new_LX = f_LX(new_time_range)
    new_LY = f_LY(new_time_range)
    new_RX = f_RX(new_time_range)

```

```

new_RY = f_RY(new_time_range)

# Создаем новый фрейм данных с интерполированными значениями
return pd.DataFrame({'T': new_time_range, 'LX': new_LX, 'LY': new_LY,
'RX': new_RX, 'RY': new_RY})

```

ПРИЛОЖЕНИЕ A25. ОБРЕЗКА ДАННЫХ СКОЛЬЗЯЩИМ СРЕДНИМ

```

def extract_trend(data):
    """
    Извлекает трендовую компоненту из данных с помощью функции
    seasonal_decompose.

    Параметры:
        data (pd.Series): Входной ряд данных.

    Возвращает:
        pd.Series: Выходной ряд данных с извлеченной трендовой
    компонентой.
    """
    trim_first = seasonal_decompose(data, model='additive',
period=100).trend
    trim_second = seasonal_decompose(trim_first.fillna(0),
model='additive', period=100).trend
    return trim_second.fillna(0)

def trim_data(data):
    """
    Обрезает данные на основе минимального значения трендовой компоненты
    для столбцов 'LY' и 'RY'.

    Параметры:
        data (pd.DataFrame): Входной датафрейм с колонками 'LY' и 'RY'.

    Возвращает:
        pd.DataFrame: Выходной датафрейм с обрезанными данными.
    """

```

ПРИЛОЖЕНИЕ A26. КЛАССИФИКАЦИЯ ИСПЫТУЕМЫХ

```

# @title Классификация
def svm_rfe_feature_selection(model, X_train, y_train, n_features):
    """
    Выполнить отбор признаков с помощью рекурсивного исключения признаков
    (RFE) с SVM.

    Параметры:
    - model: Модель SVM для отбора признаков.
    - X_train: Тренировочные данные.
    - y_train: Метки тренировочных данных.
    - n_features: Количество признаков для отбора.

    Возвращает:
    - selected_features: Булевый массив, указывающий, какие признаки были
    отобраны.
    """

    rfe = RFE(estimator=model, n_features_to_select=n_features)
    rfe.fit(X_train, y_train)
    selected_features = rfe.support_
    return selected_features

def evaluate_classifier(model, X_train, y_train, X_test, y_test):
    """
    Оценить производительность классификатора.

    Параметры:
    - model: Модель классификатора для оценки.
    - X_train: Тренировочные данные.
    - y_train: Метки тренировочных данных.
    - X_test: Тестовые данные.
    - y_test: Метки тестовых данных.

    Возвращает:
    - accuracy: Точность классификатора.
    - sensitivity: Чувствительность классификатора.
    - specificity: Специфичность классификатора.
    - confusion_matrix: Матрица ошибок классификатора.
    """

    model.fit(X_train, y_train)

```

```

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
return accuracy, sensitivity, specificity, confusion_matrix(y_test,
y_pred)

def perform_cross_validation(model, X, y, feature_selection_method,
n_features, n_splits, n_repeats, results, progress_bar):
    """
    Выполнить кросс-валидацию с отбором признаков.

    Параметры:
    - model: Модель классификатора для использования.
    - X: Данные.
    - y: Метки.
    - feature_selection_method: Метод отбора признаков для использования.
    - n_features: Количество признаков для отбора.
    - n_splits: Количество разок для кросс-валидации.
    - n_repeats: Количество повторений кросс-валидации.
    - results: Список для хранения результатов.
    - progress_bar: Индикатор прогресса.

    Возвращает:
    - None
    """

    selected_features = []
    accuracies = []
    sensitivities = []
    specificities = []
    confusion_matrices = []

    for i in range(n_repeats):
        skf = StratifiedKFold(n_splits=n_splits, shuffle=True,
random_state=i)

        for train_index, test_index in skf.split(X, y):
            X_train, X_test = X.iloc[train_index], X.iloc[test_index]
            y_train, y_test = y.iloc[train_index], y.iloc[test_index]

```

```

        selected_feature_indices = feature_selection_method(model,
X_train, y_train, n_features)
        selected_features_names = [feature_names[i] for i in
range(len(feature_names)) if selected_feature_indices[i]]

X_train_selected = X_train.iloc[:, selected_feature_indices]
X_test_selected = X_test.iloc[:, selected_feature_indices]

        (accuracy, sensitivity, specificity, confusion_matrix) =
evaluate_classifier(model, X_train_selected, y_train, X_test_selected, y_test)

        result = {
            "n_feature": n_features,
            "selected_features": ', '.join(selected_features_names),
            "accuracy": accuracy,
            "sensitivity": sensitivity,
            "specificity": specificity,
            "confusion_matrices": confusion_matrix,
        }
        results.append(result)

        progress_bar.update(1)

def start_testing(model, num_feature, n_splits, n_repeats, output_path):
    """
    Начать тестирование с кросс-валидацией и отбором признаков.

    Параметры:
    - model: Модель классификатора для использования.
    - num_feature: Максимальное количество признаков для отбора.
    - n_splits: Количество фолдов
    - n_repeats: количество повторов
    - output_path: путь для сохранения набора данных в формате .csv

    Возвращает:
    - results_df: набор данных с результатами
    """
    total_iterations = num_feature * n_splits * n_repeats
    progress_bar = tqdm(total=total_iterations, desc="Progress",
position=0, leave=False, file=sys.stdout)

```

```

results = []

for n_feature in range(num_feature, 0, -1):
    perform_cross_validation(model, X, y, svm_rfe_feature_selection,
n_feature, n_splits, n_repeats, results, progress_bar)

progress_bar.close()

results_df = pd.DataFrame(results)

if output_path:
    results_df.to_csv(output_path, index=False)
    print("Results saved to:", output_path)

return results_df

X = df_scaler.drop(columns=drop_columns)
y = df_scaler['disabled']
feature_names = df_scaler.drop(columns=drop_columns).columns

model = LogisticRegression()

num_feature = 15
n_splits = 10
n_repeats = 100

output_path =
"/content/drive/MyDrive/lectures/diplom/df_general/results_LG_relative.csv"
results_df = start_testing(model, num_feature, n_splits, n_repeats,
output_path)

```

ПРИЛОЖЕНИЕ А111. ФОРМИРОВАНИЕ ИТОГОВОГО ОТЧЕТА

```

def get_result_total(data):
    """
    Рассчитывает и суммирует результаты экспериментов по отбору признаков.

```

Параметры:

`data (pandas.DataFrame):` DataFrame, содержащий результаты экспериментов по отбору признаков.

Возвращает:

`pandas.DataFrame:` DataFrame, суммирующий результаты, включая количество признаков, уникальные признаки, выбранные признаки, относительный размер группы, биномиальную вероятность, точность, чувствительность, специфичность и матрицу конфузии.

Примечания:

Функция предполагает, что входной DataFrame имеет следующие столбцы: "n_feature", "selected_features", "accuracy", "sensitivity", "specificity" и "confusion_matrices".

"""

```
def binomial_probability(n, k, f, g):
```

"""

Вычисляет вероятность биномиального распределения.

Параметры

`n : int`

Общее количество испытаний.

`k : int`

Количество успешных испытаний.

`f : int`

Количество успешных испытаний в аналогичном наборе данных.

`g : int`

Общее количество испытаний в аналогичном наборе данных.

Возвращает

`float`

Вероятность биномиального распределения.

"""

```
def binomial_coefficient(n, k):
```

"""

Вычисляет биномиальный коэффициент.

Параметры

```

n : int
    Общее количество испытаний.
k : int
    Количество успешных испытаний.

Возвращает
-----
float
    Биномиальный коэффициент.
"""
    return math.factorial(n) / (math.factorial(k) *
math.factorial(n-k))

#print(n, k, f, g)
p = f /g
probability = binomial_coefficient(n, k) * (p**k) * ((1-p)**(n-
k))

return probability

def average_matrices(matrix_list):
    """
    Вычисляет среднюю матрицу ошибок для списка матриц ошибок.

    Параметры
    -----
    matrix_list : list
        Список матриц ошибок для усреднения.

    Возвращает
    -----
    list
        Список, содержащий среднюю матрицу ошибок.

    Возбуждает
    -----
    ValueError
        Если входной список пуст или содержит не 2x2 матрицы.
    """
    result_matrix = [[0, 0], [0, 0]]

    for matrix in matrix_list:

```

```

        for i in range(2):
            for j in range(2):
                result_matrix[i][j] += matrix[i][j]

    for i in range(2):
        for j in range(2):
            result_matrix[i][j] /= len(matrix_list)

    return result_matrix

results_summary = []
data["confusion_matrices"] = data["confusion_matrices"].apply(lambda
x: [list(map(int, line.strip('[] ').split())) for line in x.split('\n')])

group_features = data.groupby("n_feature")
                                top_groups_features =
group_features.size().sort_values(ascending=False)

    for index, size_group_feature in top_groups_features.items():
#####
        group_df_f = data[(data["n_feature"] == index)]
#####
                                grouped =
group_features.get_group(index).groupby("selected_features")
        top_groups = grouped.size().sort_values(ascending=False).head(1)

    for group, size in top_groups.items():
        group_df = data[(data["n_feature"] == index) &
(data["selected_features"] == group)]
        n_feature = group_df["n_feature"].mean().astype(int)
        relative = size / size_group_feature

        accuracy = np.round(group_df["accuracy"].mean(), 5)
        sensitivity = np.round(group_df["sensitivity"].mean(), 5)
        specificity = np.round(group_df["specificity"].mean(), 5)
        confusion_matrices = group_df["confusion_matrices"].tolist()
#####
                                unique_words =
group_df_f['selected_features'].str.split(',').explode().str.strip().unique()
        .tolist()
#####

```

```

        results_summary.extend([{'n_f': n_feature,
                                "u_f": len(unique_words),
                                "selected_features": group,
                                # "size": size,
                                'g_r': relative,
                                'h_prob':
binomial_probability(size_group_feature, size, n_feature, len(unique_words)),
                                "accuracy": accuracy,
                                "sensitivity": sensitivity,
                                "specificity": specificity,
                                "confusion_matrix":
np.round(average_matrices(confusion_matrices)).astype(int)
                                ]])

        return pd.DataFrame(results_summary)

```

ПРИЛОЖЕНИЕ А233. МОДЕЛЬ КЛАССИФИКАТОРА.

```

param_grid = {
    'max_depth': [1, 3, 5, 7],
    'learning_rate': [1, 0.1, 0.01, 0.001],
    'subsample': [0.1, 0.5, 0.7, 1],
    'colsample_bytree': [0.1, 0.5, 0.7, 1],
    'n_estimators': [50, 100, 200, 300]
}

xgb_model = xgb.XGBClassifier()

grid_search = GridSearchCV(xgb_model, param_grid, cv=5,
scoring='accuracy')

grid_search.fit(X_train, y_train)

print("Best set of hyperparameters: ", grid_search.best_params_)
print("Best score: ", grid_search.best_score_)

```

```
y_pred = grid_search.best_estimator_.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion matrix:")
print(conf_matrix)
```

ПРИЛОЖЕНИЕ Б. ГРАФИКИ ИССЛЕДОВАНИЯ

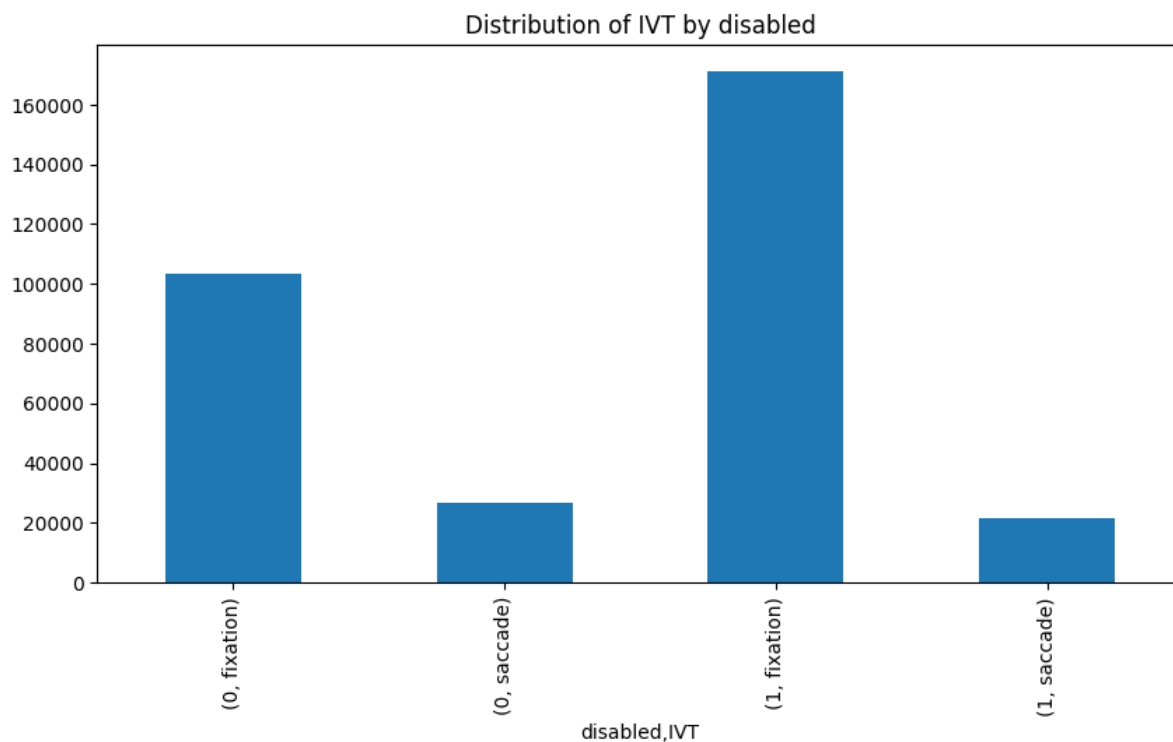


Рисунок Б2.4.1.3 – График количественное распределение выявленных событий признака IVT в группах риска (LR – 0, HR - 1)

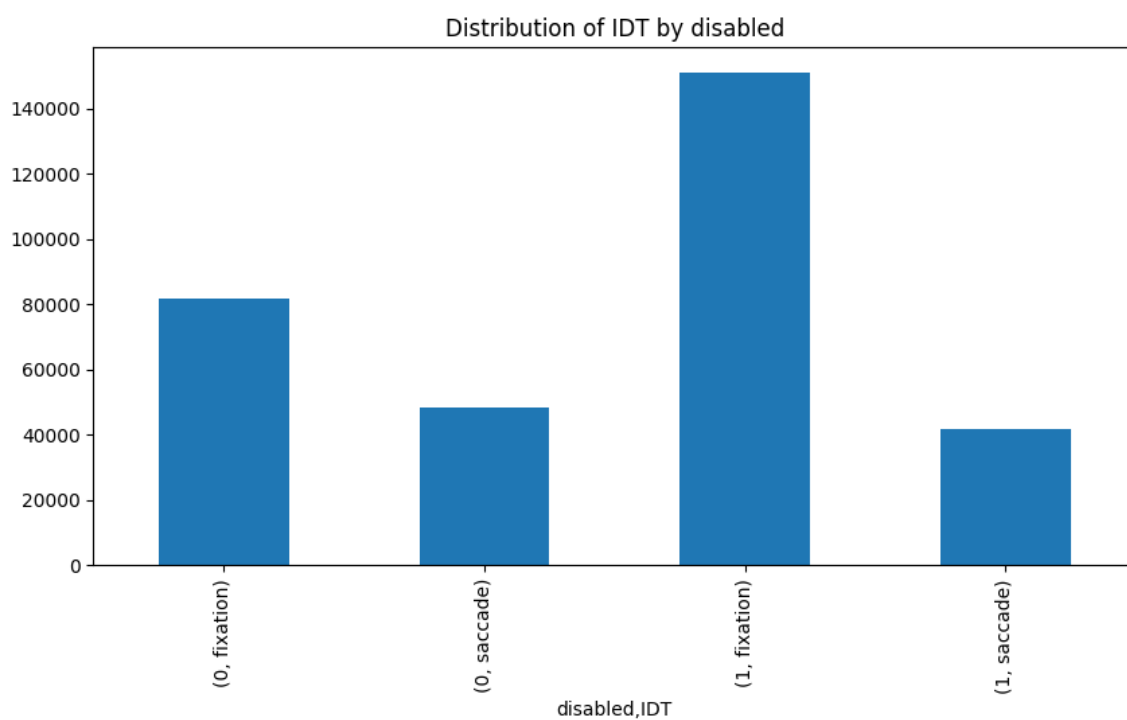


Рисунок Б2.4.2.3 – График количественное распределение выявленных событий признака IDT в группах риска (LR – 0, HR - 1)

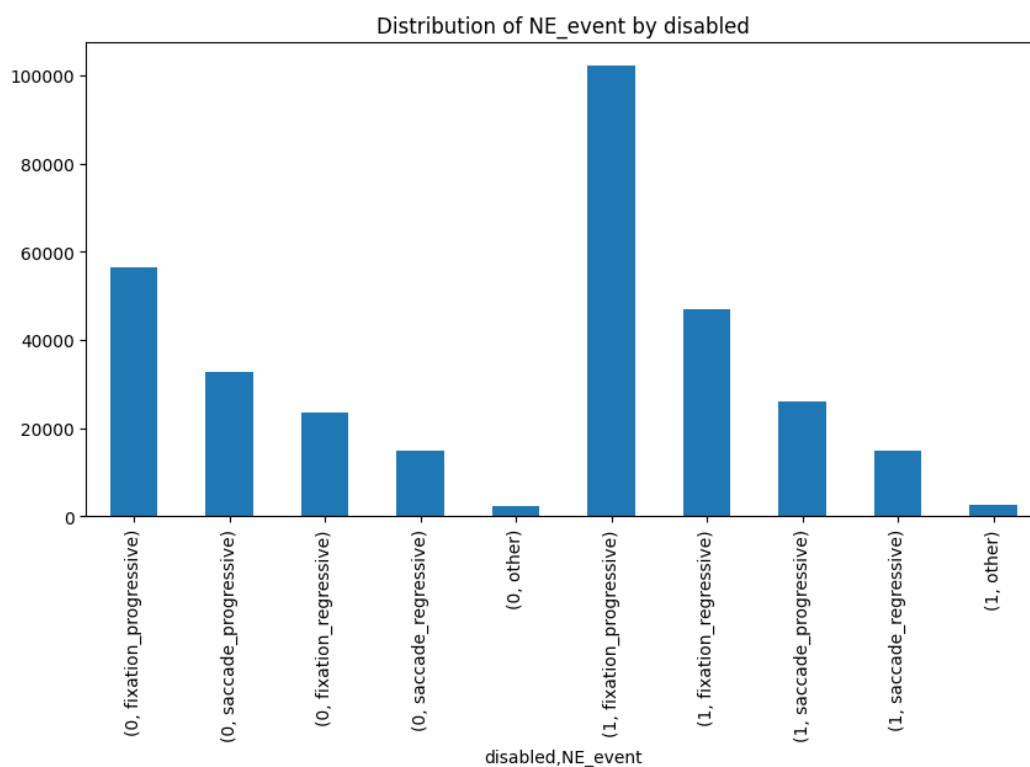


Рисунок Б2.4.3.3– График количественное распределение выявленных событий признака IDT mod в группах риска (LR – 0, HR - 1)

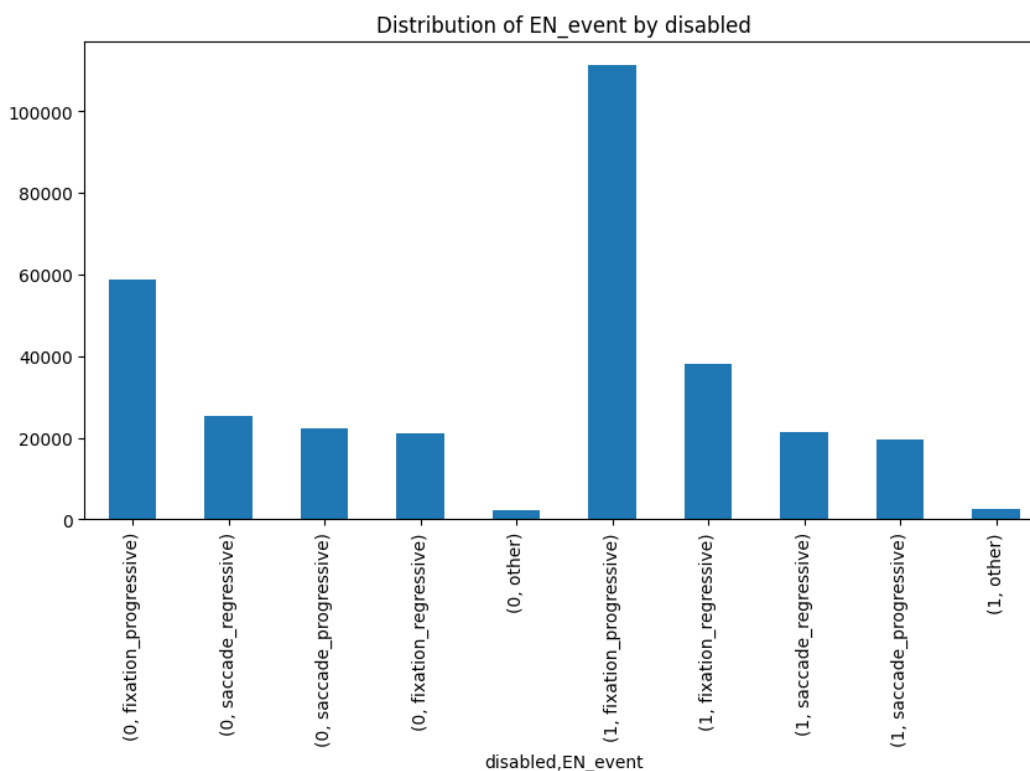


Рисунок Б2.4.3.4 – График количественное распределение выявленных событий признака IDT mod в группах риска (LR – 0, HR - 1)

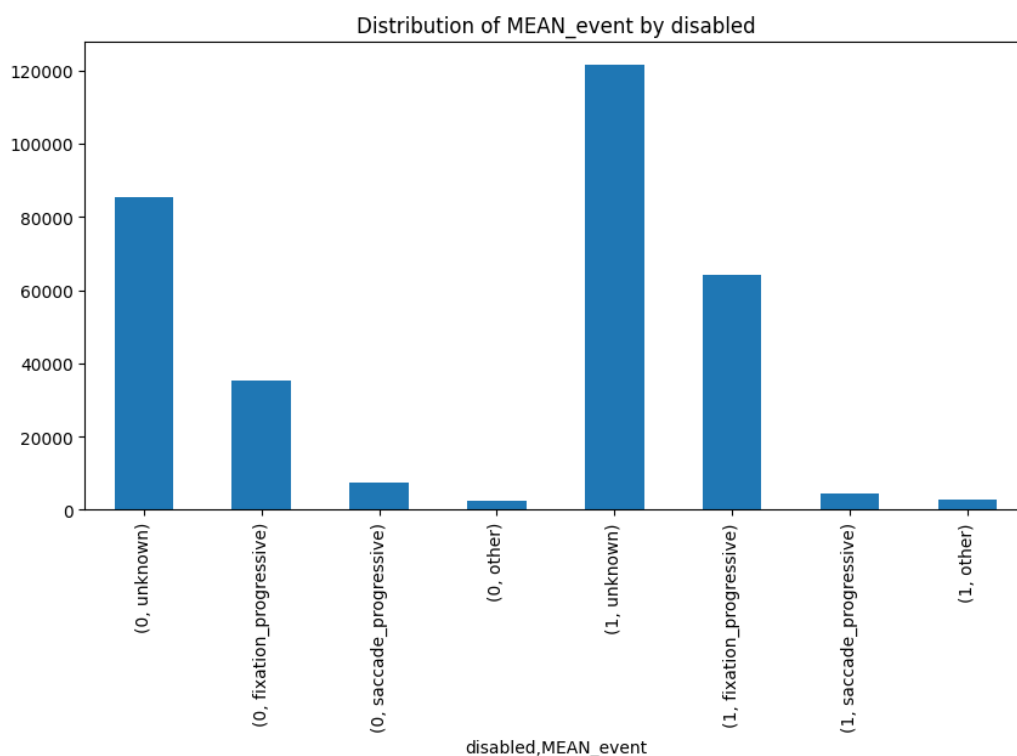


Рисунок Б2.4.3.5 – График количественное распределение выявленных событий признака IDT mod в группах риска (LR – 0, HR - 1)

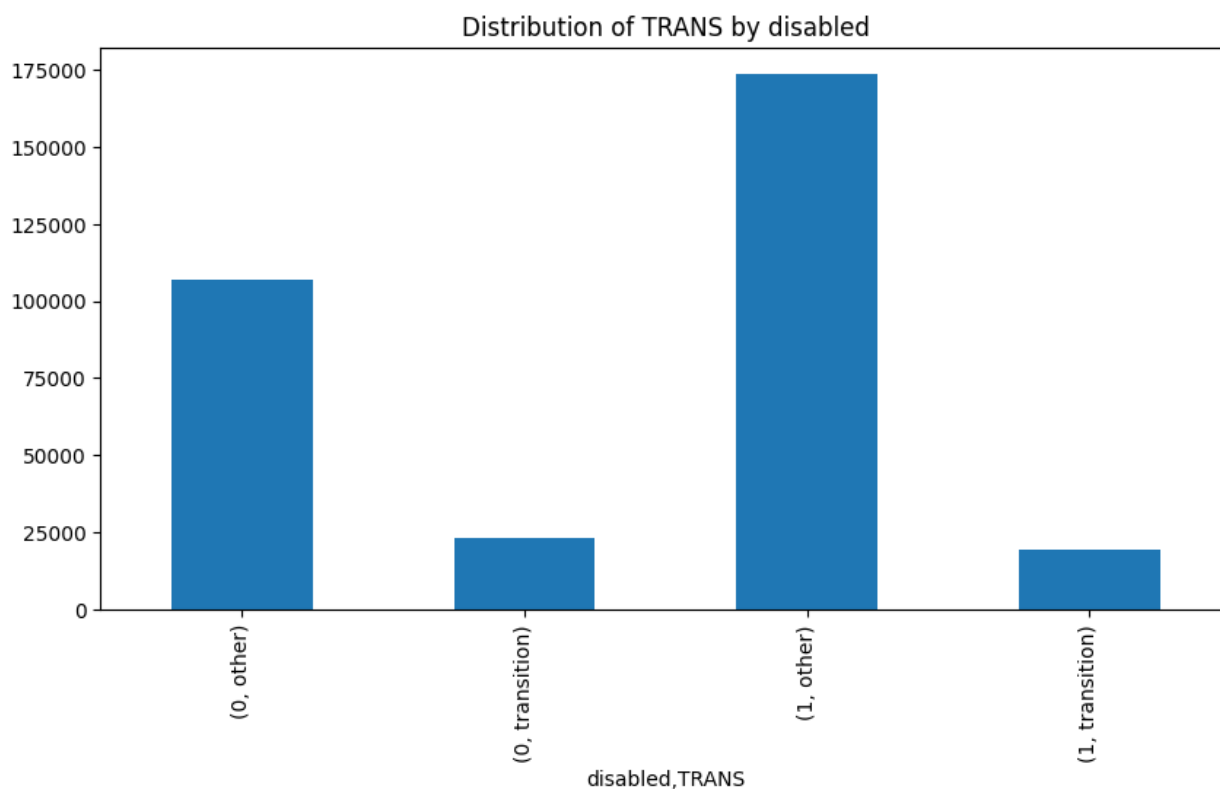


Рисунок Б2.4.4.3– График количественное распределение выявленных транзитивных событий в группах риска (LR – 0, HR - 1)

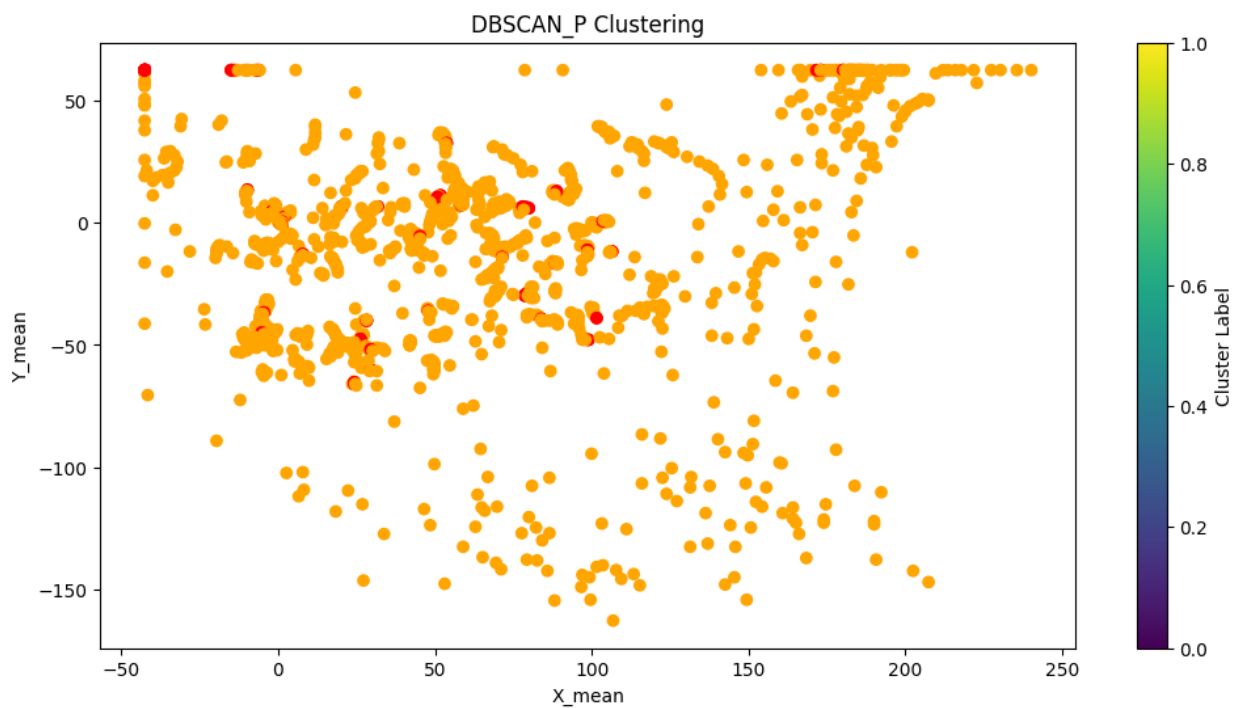


Рисунок Б2.5.1.3 – График кластеризации DBSCAN_P для испытуемого 111GM3

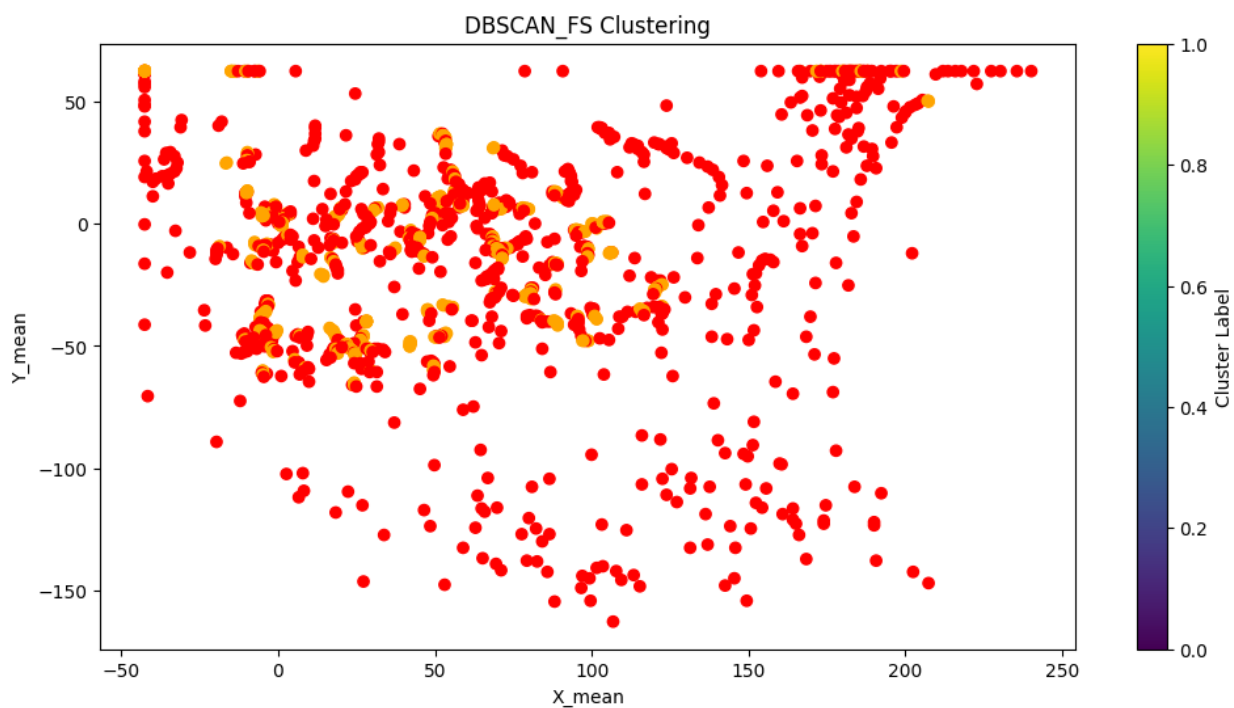


Рисунок Б2.5.1.4 – График кластеризации DBSCAN_FT для испытуемого 111GM3

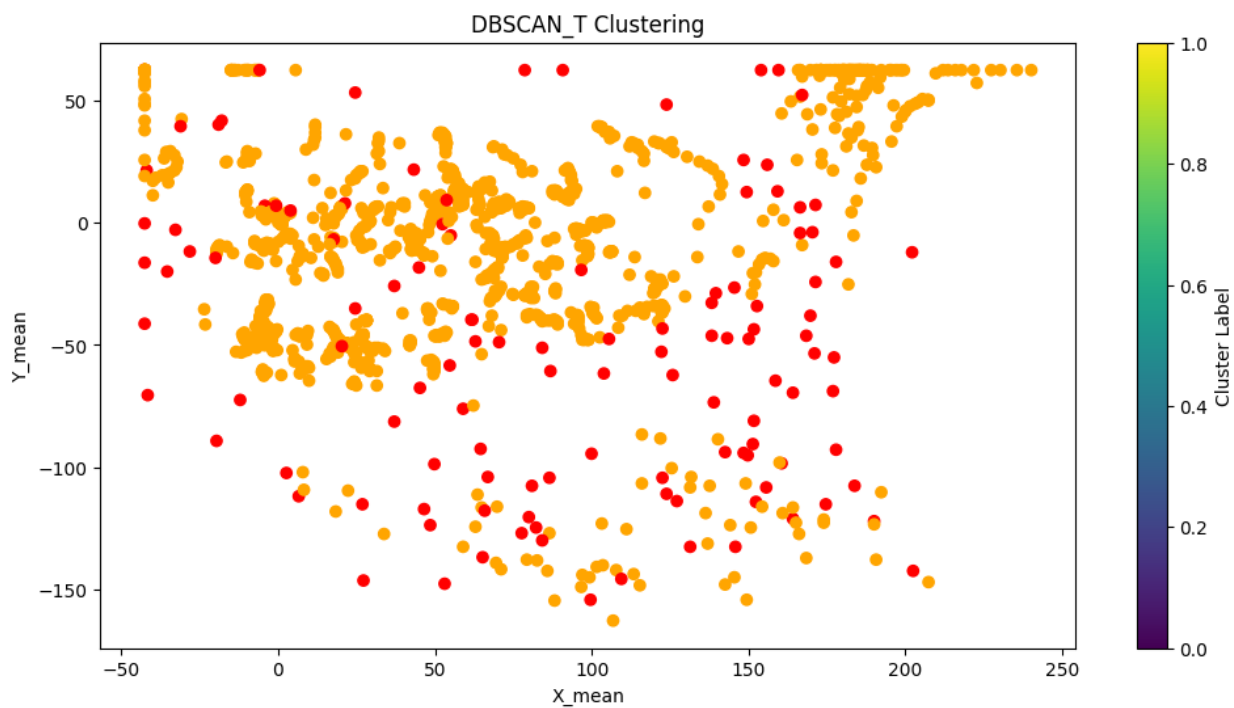


Рисунок Б2.5.1.5 – График кластеризации DBSCAN_T для испытуемого 111GM3

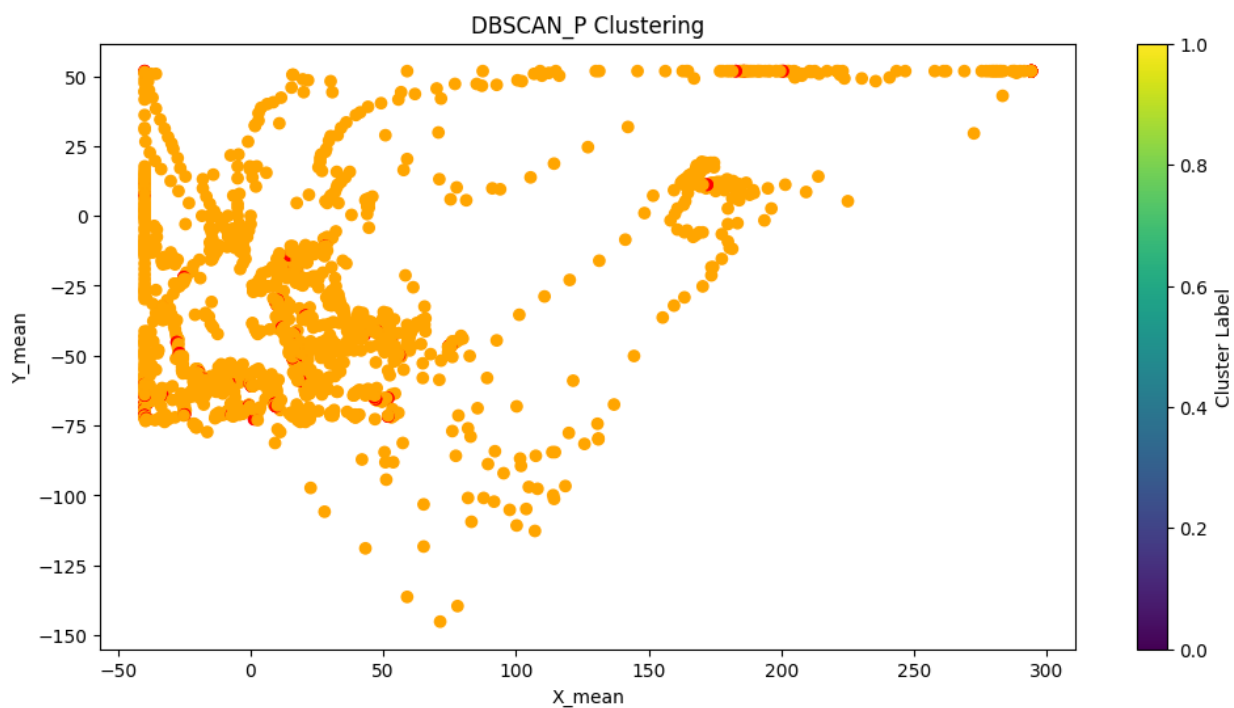


Рисунок Б2.5.1.6 – График кластеризации DBSCAN_P для испытуемого 141BE2

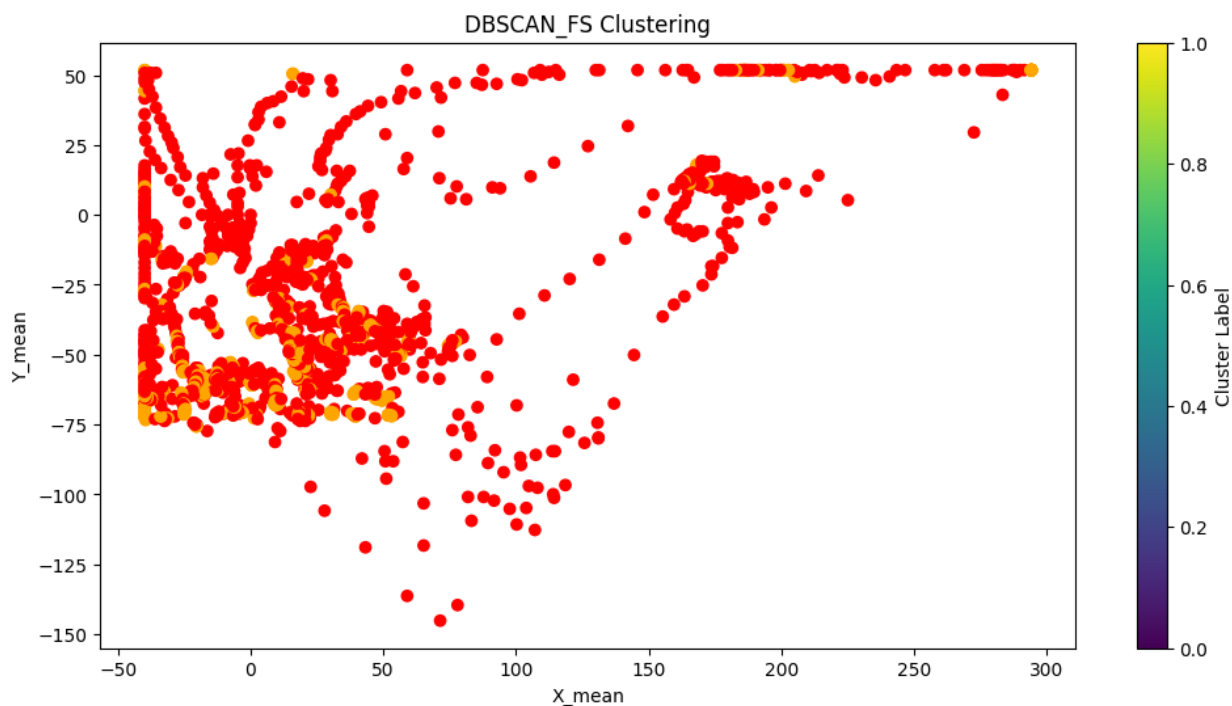


Рисунок Б2.5.1.7 – График кластеризации DBSCAN_FS для испытуемого 141BE2

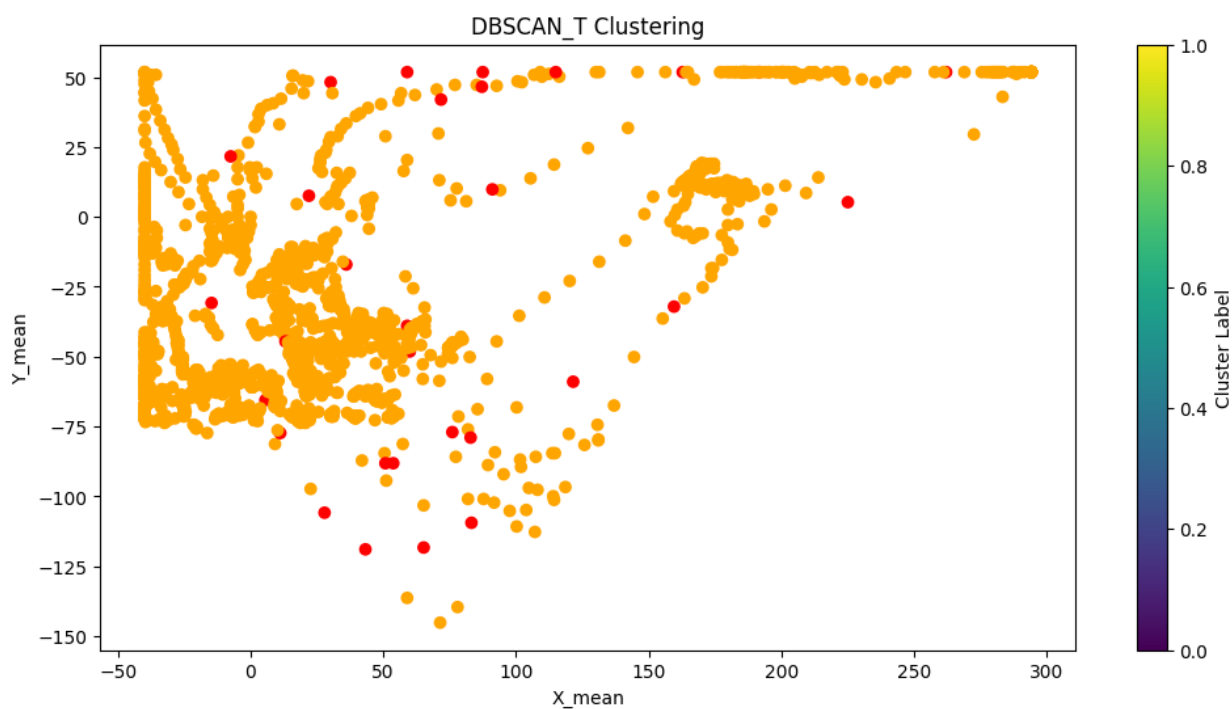


Рисунок Б2.5.1.8 – График кластеризации DBSCAN_T для испытуемого 141BE2

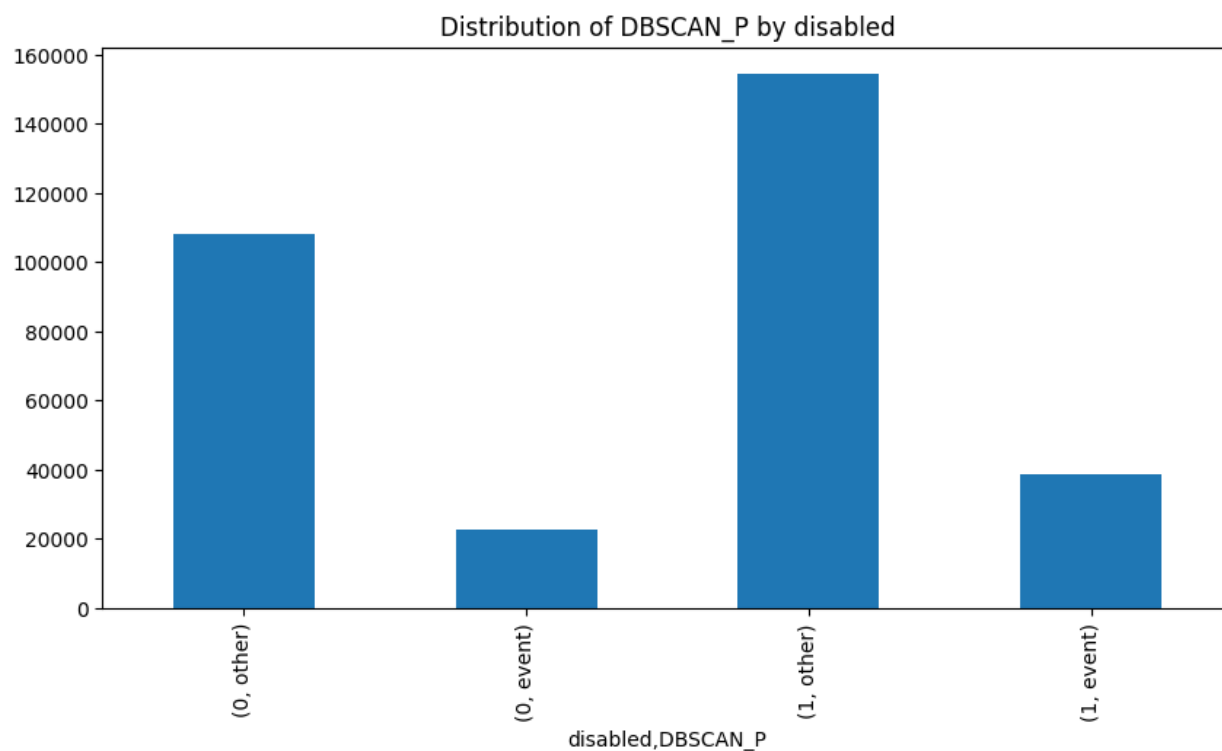


Рисунок Б2.5.1.9– График количественное распределение DBSCAN_P выявленных событий в группах риска (LR – 0, HR - 1)

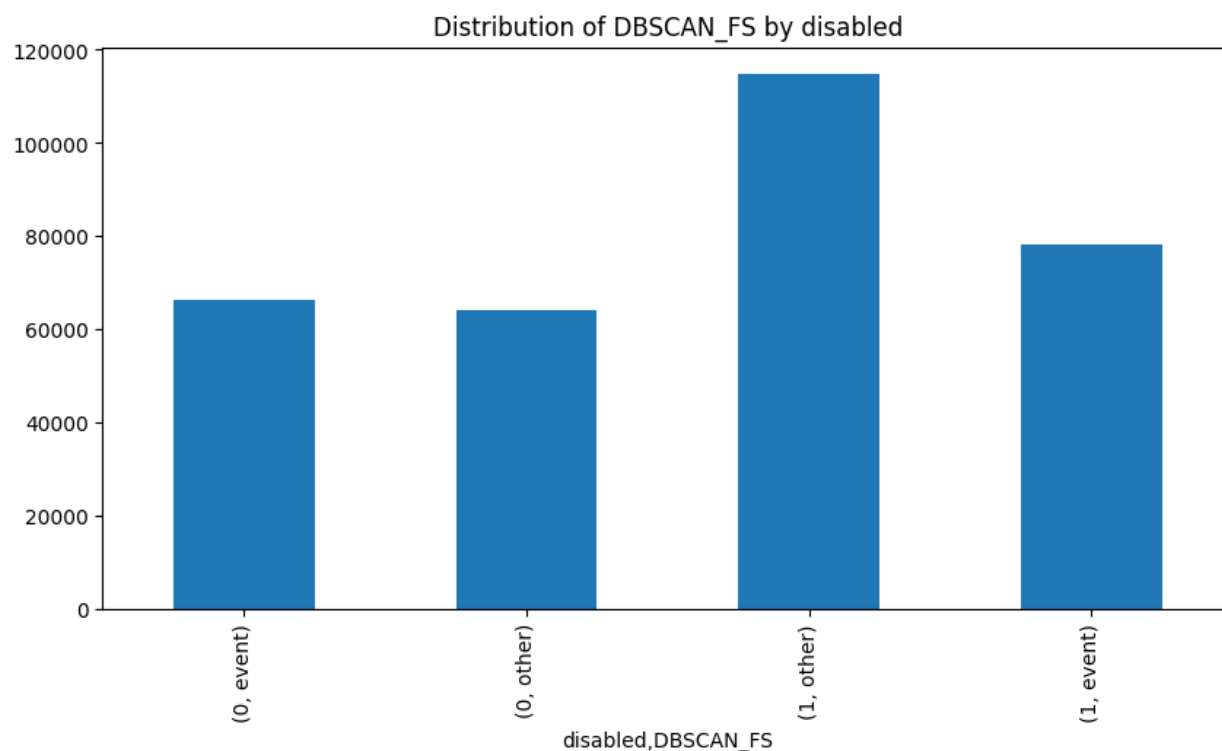


Рисунок Б2.5.1.10– График количественное распределение DBSCAN_FS выявленных событий в группах риска (LR – 0, HR - 1)

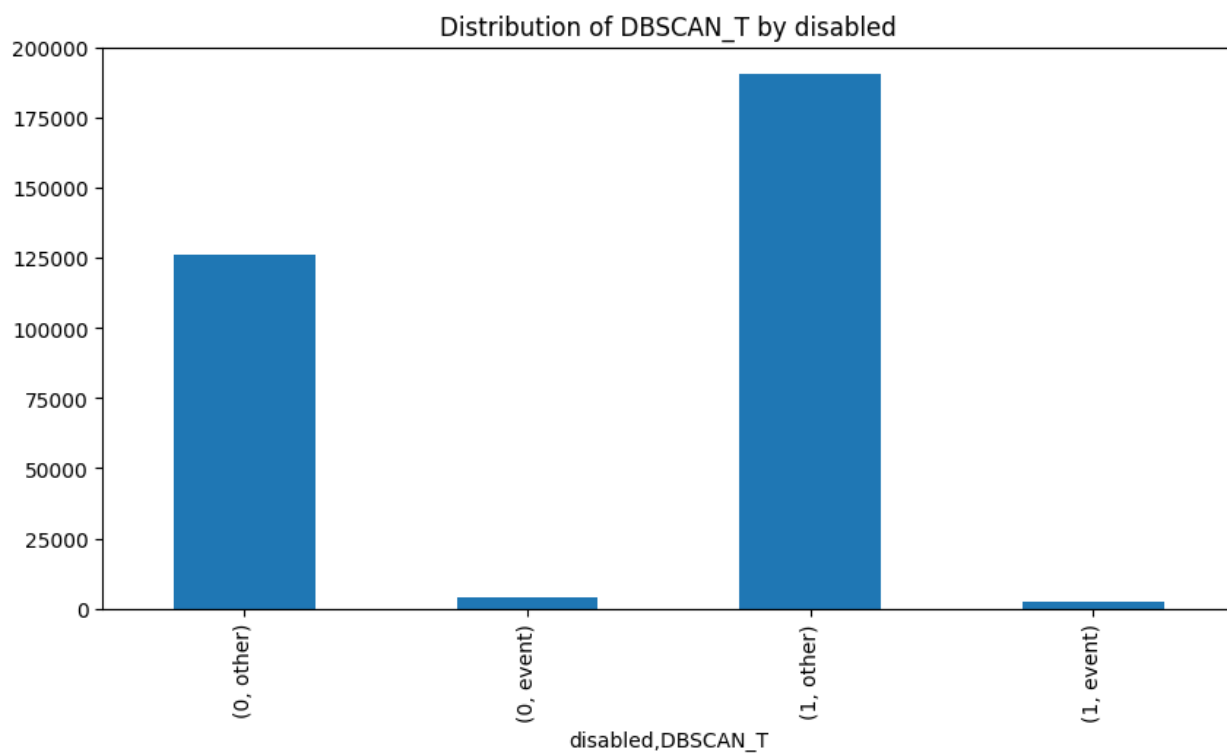


Рисунок Б2.5.1.11– График количественное распределение DBSCAN_T выявленных событий в группах риска (LR – 0, HR - 1)

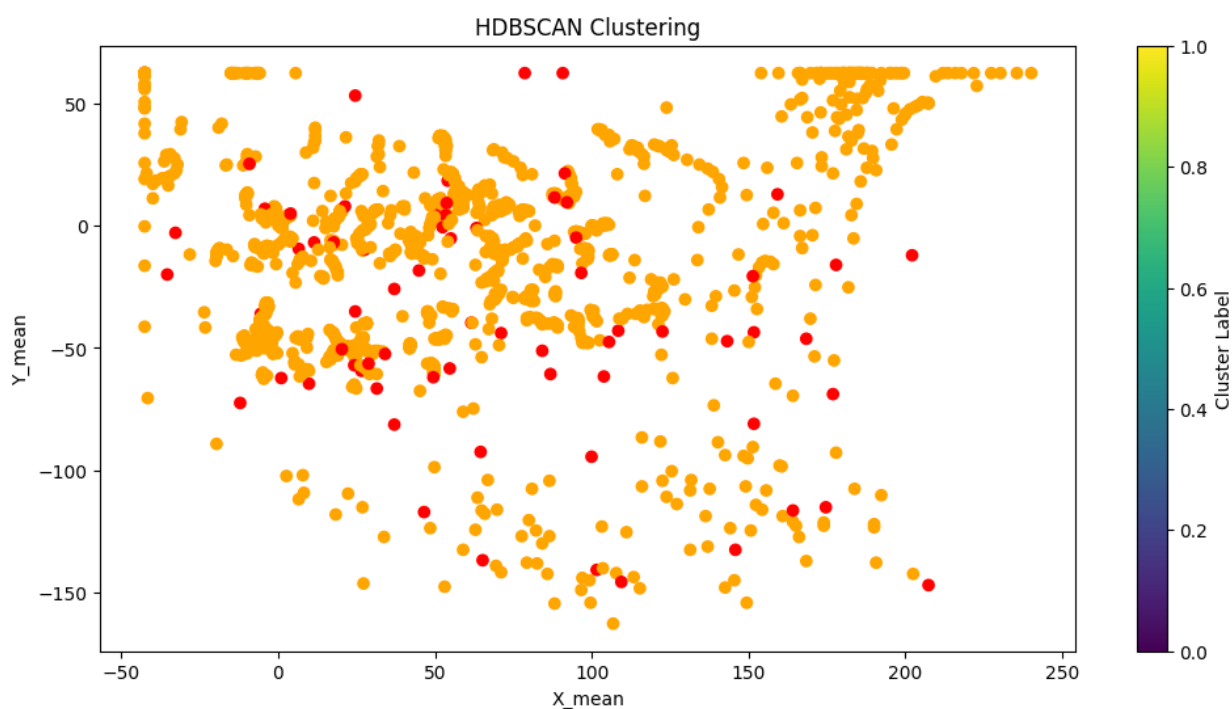


Рисунок Б2.5.2.3 – График кластеризации HDBSCAN_T для испытуемого 111GM3

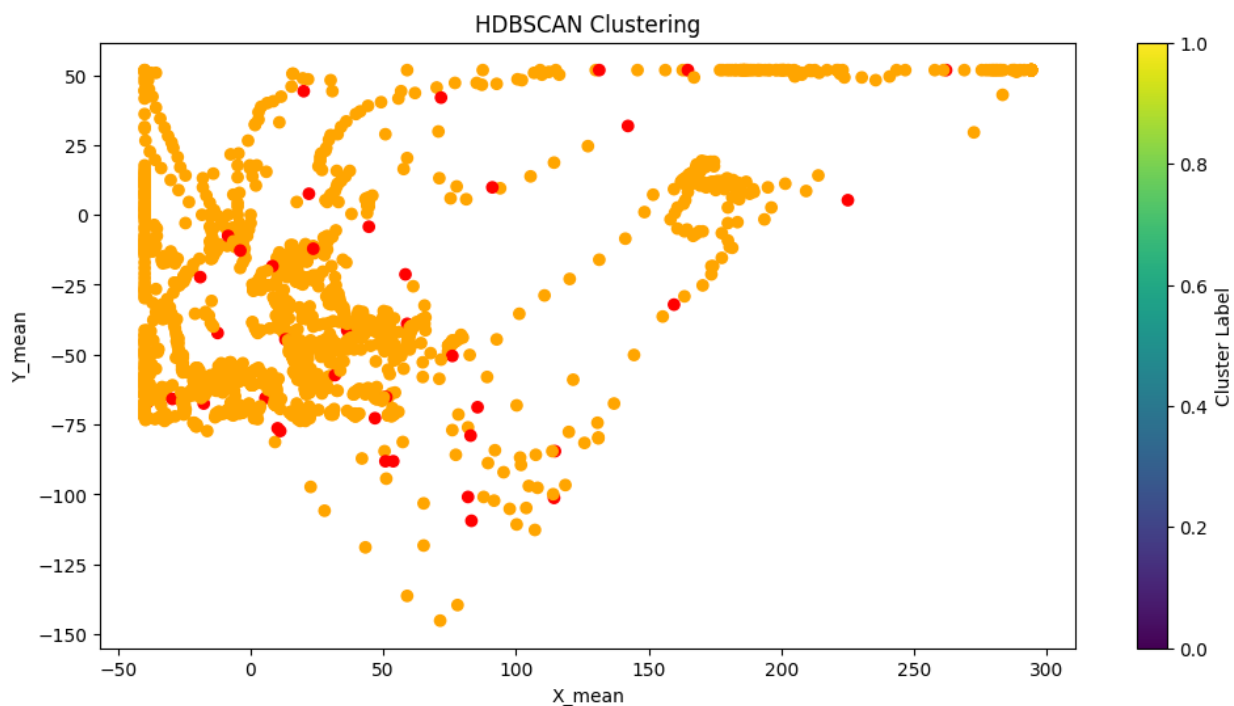


Рисунок Б2.5.2.4 – График кластеризации HDBSCAN_T для испытуемого 141BE2

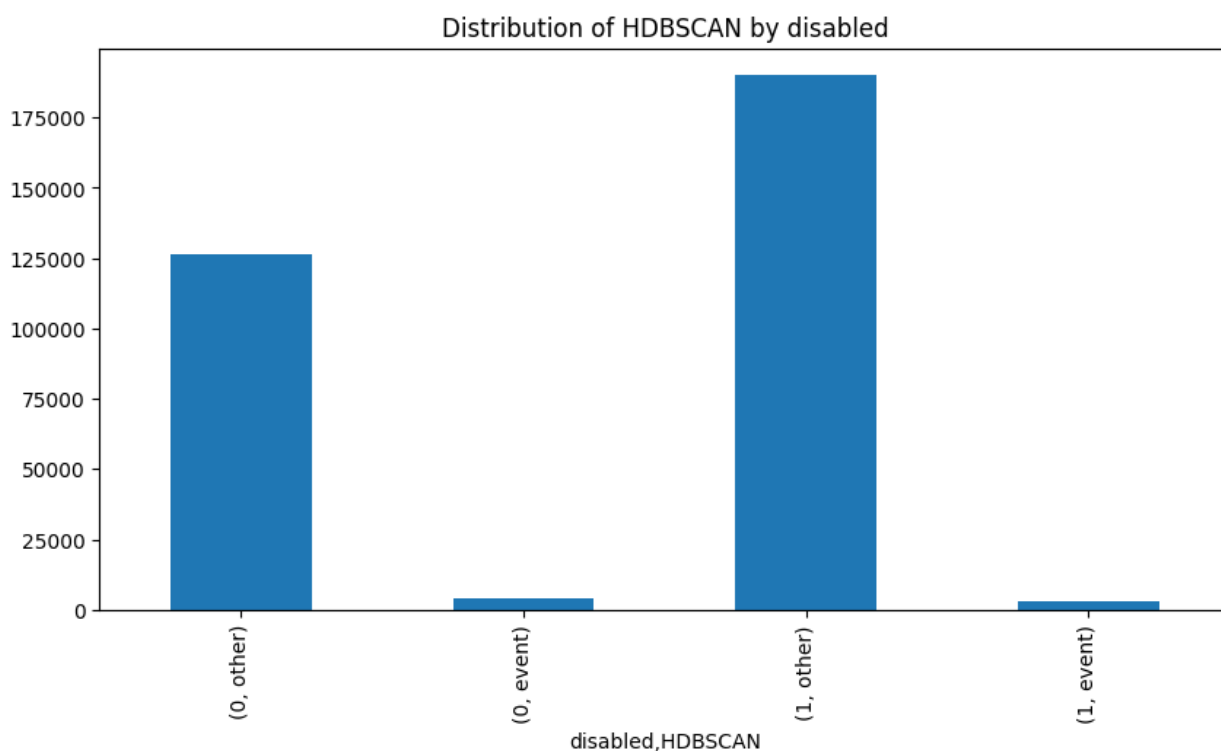


Рисунок Б2.5.2.5– График количественное распределение HDBSCAN выявленных транзитивных событий в группах риска (LR – 0, HR - 1)

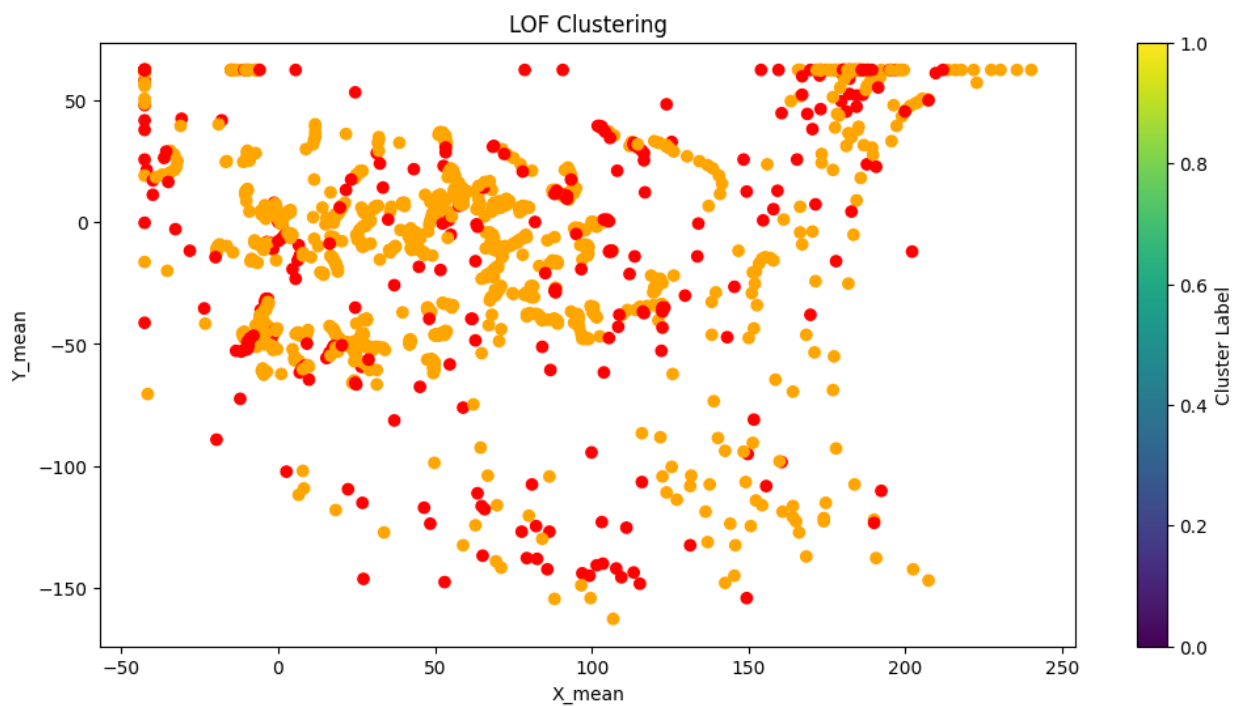


Рисунок Б2.5.3.3 – График кластеризации LOF для испытуемого 111GM3

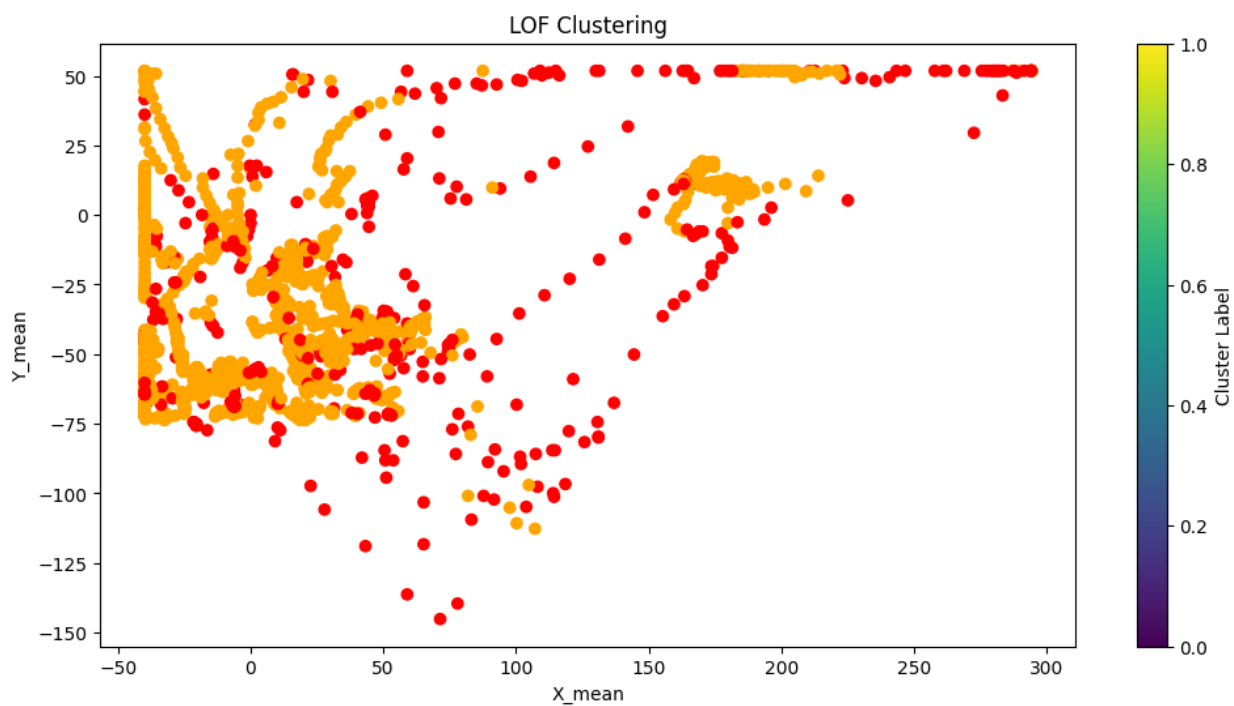


Рисунок Б2.5.3.4 – График кластеризации LOF для испытуемого 141BE2

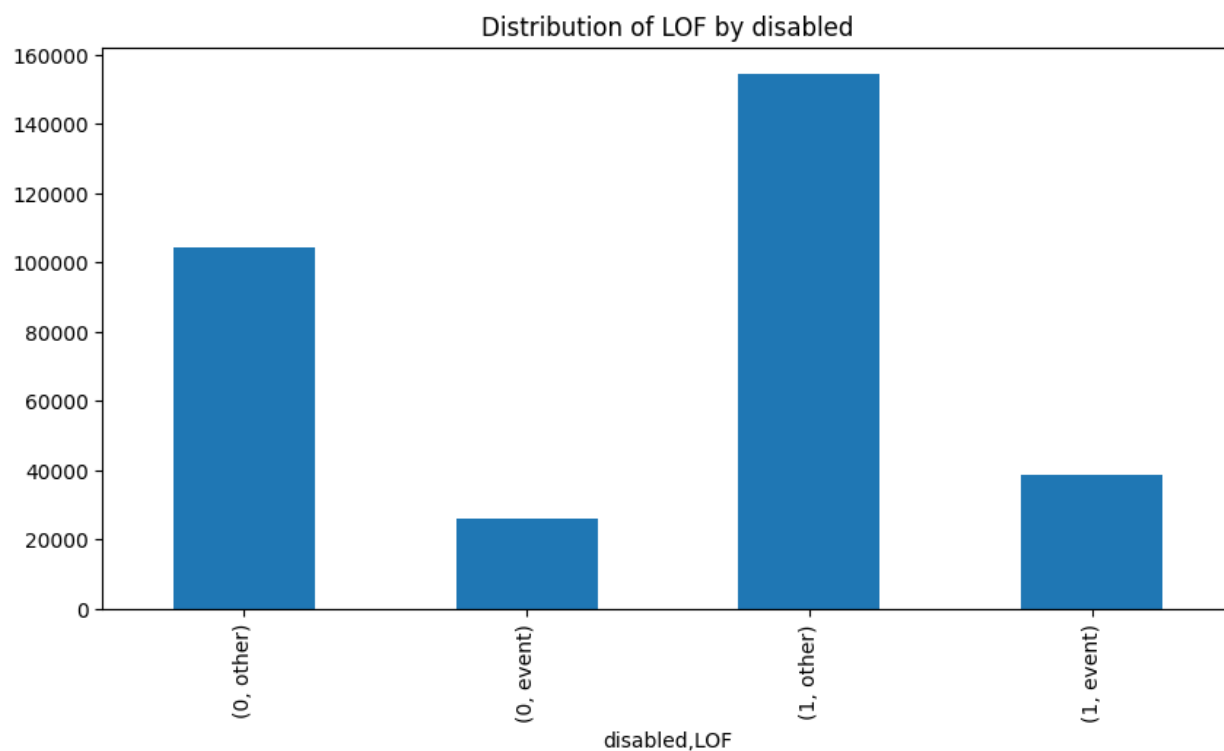


Рисунок Б2.5.3.5 – График количественное распределение выявленных событий признака LOF в группах риска (LR – 0, HR - 1)

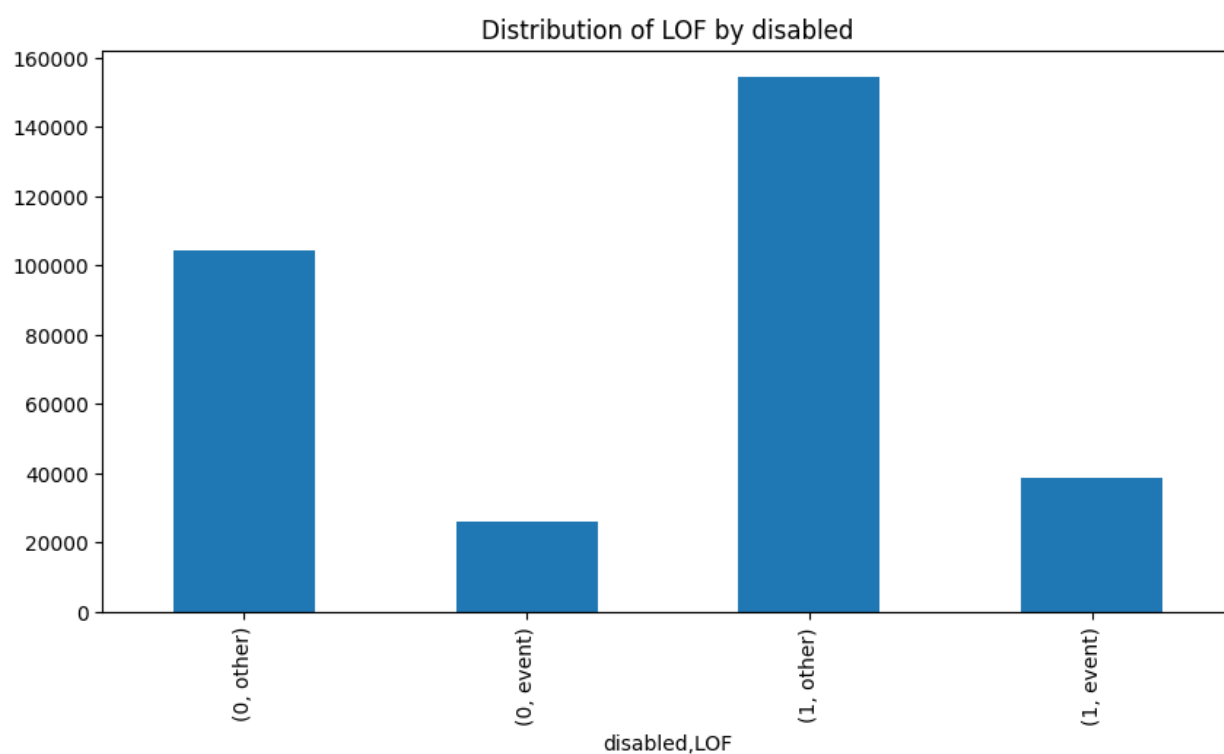


Рисунок Б2.5.3.5 – График количественное распределение выявленных событий в группах риска (LR – 0, HR - 1)

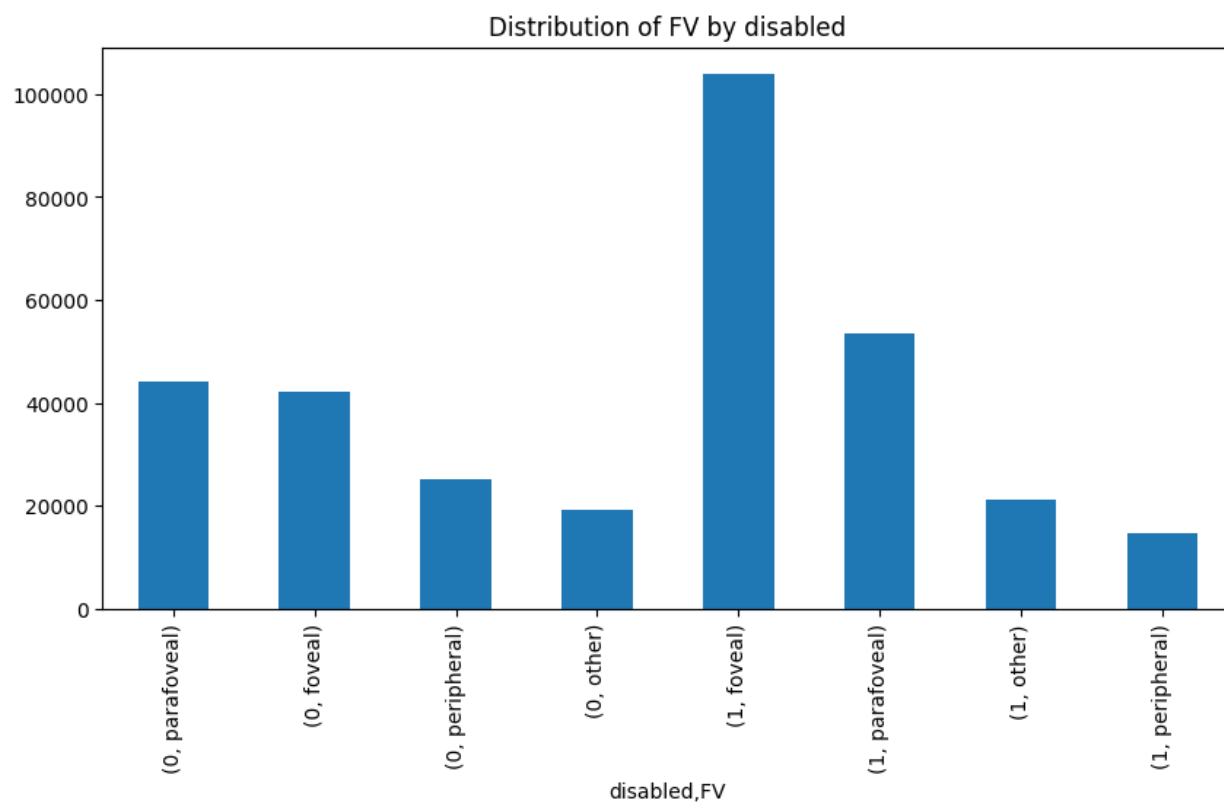


Рисунок Б2.6.1.4 – График количественное распределение выявленных событий в группах риска (LR – 0, HR - 1)