Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _БИОМЕДИЦИНСКАЯ ТЕХНИКА_____

КАФЕДРА __БИОМЕДИЦИНСКИЕ ТЕХНИЧЕСКИЕ СИСТЕМЫ_____

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КОМАНДНОМУ ПРОЕКТУ

НА ТЕМУ:

____Распознавание болезни Паркинсона на основе даннных походки_____

Студенты _БМТ1-21М, 22М, 23М
(Группа)

_____ __И. М. Башкиров__
(Подпись, дата) (И.О.Фамилия)
_____ _А. В. Гущенко___
(Подпись, дата) (И.О.Фамилия)
_____ __А. П. Гущенко_ __
(Подпись, дата) (И.О.Фамилия)
_____ __М. А. Наумов__ __
(Подпись, дата) (И.О.Фамилия)
_____ _А.И. Швецов__ __
(Подпись, дата) (И.О.Фамилия)
_____ __Д. Д. Рыбников__ __
(Подпись, дата) (И.О.Фамилия)
_____ __Д. Р. Ильенкова__ __
(Подпись, дата) (И.О.Фамилия)

Руководитель командного проекта _____ _А.А, Мошкова_____
(Подпись, дата) (И.О.Фамилия)

Москва, 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)
Научно-учебный комплекс «Радиоэлектроника, лазерная и медицинская техника»
Факультет «Биотехнические системы и технологии»
Кафедра «Биомедицинские технические системы»

УТВЕРЖДАЮ
Зав. кафедрой БМТ-1

_____ А.В. Самородов
« ___ » _____ 2025 г.

З А Д А Н И Е
на выполнение командного проекта

по дисциплине: **Разработка пайплайн**
по теме: Распознавание болезни Паркинсона на основе данных акселерометрии
(Тема командного проекта)

Студенты: Башкиров И. М. БМТ1-22М
Гущенко А. В, БМТ1-22М
Гущенко А. П. БМТ1-22М
Наумов М. А. БМТ1-22М
Швецов А. И. БМТ1-23М
Рыбников Д. Д. БМТ1-21М
Ильенкова Д. Р. БМТ1-21М
(Фамилия, инициалы, индекс группы)

График выполнения работы: __ нед. – 25 %, __ нед. – 50 %, __ нед. – 75 %, __ нед. – 100 %

Техническое задание

1. Провести анализ современных методов диагностики БП с акцентом на анализ походки.
2. Исследовать эффективность различных алгоритмов машинного обучения (Random Forest, Linear Discriminant Analysis, Logistic Regression, XGB, ClassifierSupport Vector Machine) для классификации пациентов с БП.
3. Разработать и обучить модели глубокого обучения (CNN, LSTM) для анализа временных рядов данных походки.

Оформление командного проекта

- расчётно-пояснительная записка;
- графический материал:презентация

Дата выдачи задания « ____ » _____ 2025 г.

Руководитель командного проекта _____ **А. В. Колпаков**
(Подпись, дата) (И.О. Фамилия)

Студенты **БМТ1-21М, 22М, 23М** _____ **И. М. Башкиров**
(Группа) _____ **А. В, Гущенко**
_____ **А. П. Гущенко**

_____	**М. А. Наумов**
(Подпись, дата)	(И.О. Фамилия)
_____	**А. И. Швецов**
(Подпись, дата)	(И.О. Фамилия)
_____	**Д. Д. Рыбников**
(Подпись, дата)	(И.О. Фамилия)
_____	**Д. Р. Ильенкова**
(Подпись, дата)	(И.О. Фамилия)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Болезнь Паркинсона (БП) является одним из наиболее распространенных нейродегенеративных заболеваний, характеризующимся прогрессирующей дегенерацией дофаминергических нейронов черной субстанции и развитием двигательных и немоторных симптомов. Несмотря на длительную историю изучения, ранняя диагностика БП остается актуальной проблемой в клинической практике. Современные исследования подчеркивают гетерогенность заболевания, выделяя различные клинические подтипы с отличающимся прогнозом, что требует разработки точных и объективных методов диагностики.

Одним из перспективных направлений в диагностике БП является анализ походки, поскольку двигательные нарушения, такие как брадикинезия, шаркающая походка и эпизоды "застывания", являются ключевыми маркерами заболевания. Однако традиционные методы анализа походки, основанные на сложном оборудовании и экспертной оценке, обладают ограниченной доступностью. В связи с этим актуальной задачей становится разработка автоматизированных алгоритмов машинного обучения (ML) и глубокого обучения (DL), способных анализировать данные с датчиков движения для классификации пациентов с БП и здоровых лиц.

Цель работы – разработка пайплайна обработки данных и алгоритмов ML/DL для классификации пациентов с БП на основе параметров походки, регистрируемых датчиками движения.

Задачи исследования:

1. Провести анализ современных методов диагностики БП с акцентом на анализ походки.
2. Исследовать эффективность различных алгоритмов машинного обучения (Random Forest, Linear Discriminant Analysis, Logistic Regression, XGB, ClassifierSupport Vector Machine) для классификации пациентов с БП.
3. Разработать и обучить модели глубокого обучения (CNN, LSTM) для анализа временных рядов данных походки.

1. Теоретическая часть

1.1. Болезнь Паркинсона

Болезнь Паркинсона (БП) – дегенеративное **заболевание** головного мозга, сопровождающееся симптомами нарушения двигательной функции (замедленностью движений, тремором, ригидностью и потерей равновесия) и другими осложнениями, включая снижение когнитивных функций, психические расстройства, нарушения сна, боли и расстройства чувствительности [1].

БП занимает второе место по распространенности среди нейродегенеративных заболеваний, уступая лишь болезни Альцгеймера (БА). Согласно статистике Всемирной организации здравоохранения (ВОЗ), численность пациентов с БП превышает 4 млн человек, при этом эпидемиологические прогнозы указывают на устойчивую тенденцию к росту заболеваемости. Так, если в 2005 году количество пациентов старше 50 лет составляло 4,1–4,6 млн, то к 2030 году ожидается его увеличение до 8,7–9,3 млн [1]. В Российской Федерации, по данным популяционных исследований, распространенность БП оценивается приблизительно в 210 тыс. случаев [2]. Глобальные эпидемиологические данные свидетельствуют о частоте встречаемости БП на уровне 1–2 случая на 1000 населения, с выраженной возрастной зависимостью [3].

Увеличение продолжительности жизни в сочетании с недостаточной изученностью патогенетических механизмов БП обусловили повышенный интерес научного сообщества к исследованию этиологии данного заболевания в период 2012–2017 гг.[2].

Причины БП остаются невыясненными. Согласно современным представлениям, в развитии заболевания играют роль возрастные, генетические и средовые факторы. Большинство случаев БП носят спорадический характер, однако наличие заболевания у ближайших родственников повышает риск его развития в два раза. Наследственные формы составляют лишь около 10% случаев. Предполагается, что генетическая предрасположенность может усиливать чувствительность нигростриарной системы к повреждающим факторам и возрастным изменениям. Исследуется влияние внешних факторов, таких как инфекции, интоксикации, воздействие тяжелых металлов и пестицидов, а также употребление колодезной воды в сельской местности.

1.2. Патогенез БП

Современная концепция патогенеза БП связывает дегенерацию нигростриарных нейронов с нарушениями внутриклеточного метаболизма. К ключевым механизмам относятся [4]:

- окислительный стресс;
- эксайтотоксичность глутамата;
- избыточное накопление ионов кальция в клетках;
- повышение активности внутриклеточных протеаз;
- дисфункция митохондриального дыхания, ведущая к энергетическому дефициту;
- нарушения метаболизма железа.

Эти процессы приводят к активации апоптоза, однако точные механизмы и последовательность нейродегенеративных изменений остаются не до конца изученными. Также при БП наблюдается дегенерация нейронов черной субстанции, голубого пятна, а также образование внутриклеточных включений — телец Леви, представляющих собой продукты дегенерации белков.

Нейротрансмиттерные изменения включают:

- снижение синтеза дофамина;
- избыток глутамата и ацетилхолина;
- недостаточность норадреналина и серотонина.

1.3. Клиническая картина БП

Болезнь Паркинсона вызывает моторные и немоторные симптомы (таблица 1). Моторные симптомы включают нарушения движений и физических функций: тремор, ригидность, брадикинезию (замедленность движений) и постуральную неустойчивость. Немоторные симптомы (не связанные с движением) затрагивают различные системы органов, такие как желудочно-кишечный и мочеполовой тракты, и отличаются значительной гетерогенностью. У пациентов с БП немоторные симптомы обычно развиваются постепенно в течение нескольких лет до появления двигательных нарушений, однако часто остаются неозвученными, если не заданы целенаправленные вопросы. К продромальным немоторным проявлениям относятся [5]:

- расстройство поведения во время фазы быстрого сна (RBD);
- гипосмия/аносмия (снижение или потеря обоняния);

- запоры;
- дисфункция мочевого пузыря;
- ортостатическая гипотензия;
- чрезмерная дневная сонливость;
- депрессия [6].

Хотя эти симптомы не являются специфичными исключительно для болезни Паркинсона, их сочетание повышает риск последующей постановки этого диагноза [6]. RBD, особенно подтвержденный с помощью полисомнографии, ассоциирован с высоким риском развития БП [6]. Более 90% пациентов с идиопатическим RBD в итоге заболевают синуклеинопатией (нейродегенеративным заболеванием, связанным с накоплением α-синуклеина), чаще всего БП или родственными состояниями (деменцией с тельцами Леви, мультисистемной атрофией) [7]. Среди пациентов с БП RBD встречается в 30–50% случаев [8].

Продромальные симптомы связаны с ранним поражением ствола мозга при болезни Паркинсона. Когда нейропатологический процесс приводит к гибели примерно 50% нейронов в каудальной части чёрной субстанции, появляются моторные признаки заболевания [9].

Таблица 1 – Симптомы и признаки при БП

Симптом или признак	**Определение / Ключевые характеристики**
Моторные симптомы	
Брадикинезия	Замедленность и прогрессирующее уменьшение амплитуды движений (гипокинезия) при повторении действий (например, постукивание пальцами, сжимание кулака).
Ригидность	Непроизвольное сопротивление пассивным движениям в суставе (например, в локте, запястье), не зависящее от скорости движения; может сопровождаться феноменом «зубчатого колеса».
Тремор покоя	Дрожание частотой 4–6 Гц в полностью расслабленной конечности, временно исчезающее при удержании руки вытянутой и возвращающееся в покое (реэмергентный тремор); отсутствует при движении.
Постуральная неустойчивость	Нарушение баланса, затрудняющее поддержание и смену поз (ходьба, стояние); обычно появляется на поздних стадиях БП.
Немоторные симптомы	

Нарушение обоняния	Снижение (гипосмия) или потеря (аносмия) способности различать запахи.
Нарушения сна	Симптомы расстройства поведения в фазе быстрого сна (RBD), дневная сонливость, инсомния (проблемы с поддержанием сна).
Вегетативная дисфункция	Запоры, замедленное опорожнение желудка, учащенное мочеиспускание, эректильная дисфункция, ортостатическая гипотензия, нестабильность АД.
Психические расстройства	Депрессия, тревога, апатия, психозы (чаще зрительные галлюцинации).
Когнитивные нарушения	Легкие когнитивные расстройства или деменция, чаще затрагивающие внимание, исполнительные и зрительно-пространственные функции.
Другие симптомы	Усталость, гипофония (ослабление голоса), гиперсаливация (слюнотечение), дисфагия (нарушение глотания).

1.4. Нарушение походки при БП

Анализ походки является важным инструментом в оценке двигательных нарушений при нейродегенеративных заболеваниях, в частности при БП. Походка при БП характеризуется рядом специфических паттернов, обусловленных дегенерацией нигростриарных нейронов и дисфункцией базальных ганглиев, что приводит к нарушению нервно-мышечного контроля.

Основные характеристики походки при БП

- Брадикинезия – замедленность движений, проявляющаяся в укорочении длины шага и снижении скорости ходьбы [10];
- Шаркающая походка – уменьшение высоты подъема стопы вследствие гипокинезии, что приводит к характерному шаркающему звуку [11];
- Затрудненное инициирование движения (феномен «застывания») – трудности в начале ходьбы, особенно при прохождении узких пространств или разворотах;
- Снижение амплитуды движений в суставах – наиболее выражено в голеностопном (уменьшение тыльного сгибания) и тазобедренном суставах, что способствует укорочению шага [10];
- Постуральная неустойчивость – нарушение баланса, повышающее риск падений.

Нарушения походки при БП связаны с дефицитом дофаминергической передачи в стриатуме, что приводит к дисфункции внутренней программы генерации шага. Компенсаторно пациенты увеличивают частоту шагов (каденс), однако длина шага остается уменьшенной. Важную роль играет также нарушение сенсорной интеграции, что подтверждается улучшением походки при использовании внешних стимулов (зрительных или слуховых). Исследование походки при БП включает анализ пространственно-временных (длина и частота шага, скорость ходьбы), кинематических (угловые параметры суставов) и кинетических (силовые взаимодействия) параметров.

1.4.1. Пространственно-временные параметры

Демографические факторы (возраст, рост, масса тела, пол) влияют на вариабельность походки, хотя их вклад в классификацию патологий менее значителен [12]. В исследованиях с использованием базы данных *Gait in Neurodegenerative Disease Database* [13] выделены 10 временных параметров, из которых наиболее информативными оказались длительность опоры, двойной поддержки и переноса ноги. Применение методов селекции признаков (например, анализ главных компонент) позволило достичь точности классификации 79–94%.

Аналогично, в [14] анализ 16 параметров, полученных с помощью инфракрасных камер, показал, что длина шага, скорость переноса и каденс наиболее значимы для выявления нарушений походки. Уменьшение числа признаков до 3–5 сохраняло точность классификации на уровне 93,6–98%.

1.4.2. Кинематические параметры

Кинематический анализ включает изучение угловых перемещений суставов (голеностопных, коленных, тазобедренных) в сагиттальной плоскости, а также амплитуды движений (Range of Motion, RoM) [15]. В [16] сочетание RoM с пространственно-временными параметрами (87 признаков) обеспечило точность классификации болезни Паркинсона до 96%, при этом данные только с коленных суставов превзошли по информативности пространственно-временные параметры.

Исследования в условиях двойной задачи [17] показали, что кинематические параметры (углы бедра, колена, голеностопа) более четко отражают нарушения походки при когнитивных расстройствах, чем пространственно-временные. Дополнительные параметры — симметрия походки, плавность, вариабельность шага — измеряются с помощью инерционных датчиков (IMU) и коррелируют с тяжестью заболеваний.

Таким образом, оптимальный выбор признаков зависит от задачи: пространственно-временные параметры универсальны, а кинематические более чувствительны к специфическим нарушениям. Комбинация методов повышает точность диагностики.

1.4.3. Кинетические параметры

Кинетические параметры отражают причины двигательных нарушений, такие как силы и моменты в суставах нижних конечностей (голеностопном, коленном, тазобедренном). Их измеряют с помощью платформ с датчиками силы, инструментированных дорожек или смарт-стелек, регистрирующих вертикальную составляющую реакции опоры (VGRF).

В [18] всего четыре кинетических признака на основе VGRF позволили классифицировать пациентов с БП и здоровых с точностью 96,39%. Более комплексный подход включал анализ статистических моментов (среднее, стандартное отклонение, асимметрия, эксцесс) и энтропии сигналов VGRF, что обеспечило точность 93,89–100% даже при использовании данных одной стопы. Это подтверждает эффективность кинетических параметров для диагностики нейродегенеративных заболеваний на разных стадиях

Выявление специфических маркеров, таких как брадикинезия и постуральная неустойчивость, способствует ранней диагностике и мониторингу прогрессирования заболевания [19].

1.4.4. Застывания при БП

Феномен застываний при ходьбе представляет собой специфическое двигательное нарушение, проявляющееся внезапными кратковременными эпизодами невозможности инициации или продолжения движения. В научной литературе данное состояние также обозначается как парадоксальная акинезия или моторный блок [15].

Клинически застывания могут выражаться в виде:

- парциальных застываний – совершения неэффективных мелких шагов (длиной от нескольких миллиметров до сантиметров) или топтания на месте;
- тремора нижних конечностей, возникающего при попытке начать или продолжить движение.

Эпизоды застывания чаще развиваются спонтанно в момент переключения между двигательными актами, например:

- при старте ходьбы (стартовые застывания);

- во время поворотов;
- при преодолении препятствий или прохождении через узкие пространства (дверные проемы);
- при достижении цели (например, при приближении к стулу).

Данный симптом преимущественно наблюдается на поздних стадиях БП. Однако если застывания возникают в первый год заболевания, требуется дифференциальная диагностика для исключения атипичного и вторичного паркинсонизма.

Во время эпизода застывания у пациентов отмечается фиксация стоп к поверхности, что делает невозможным продолжение движения. В ряде случаев застывания развиваются без видимых провоцирующих факторов, например, при ходьбе по прямой на открытом пространстве. Помимо этого, эмоциональное состояние и окружающая обстановка могут модулировать частоту возникновения застываний.

Провоцирующими факторами также выступают:

- выполнение двойных заданий (например, счет во время ходьбы);
- нахождение в ограниченном пространстве (в толпе);
- необходимость выполнения двигательных задач в условиях временного дефицита (переход дороги на зеленый сигнал светофора).

1.4.5. Семенящая походка

Семенящая походка — еще один характерный и уникальный паттерн ходьбы при БП. Она характеризуется появлением коротких быстрых шагов, возникающих при попытке удержать центр давления в пределах площади опоры, когда туловище непроизвольно стремится вперед. Семенящая походка может предшествовать эпизоду застываний, когда шаги становятся все короче и чаще, в итоге приводя к развитию моторного блока (застывания). Механизмы семенящей походки и ее связь с застываниями до конца не изучены. Скорее всего семенящая походка является самостоятельным эпизодическим нарушением ходьбы, встречающимся при БП.

1.4.6. Падения

Падения ассоциированы с повышенным риском летальности и травматизации, главным образом вследствие черепно-мозговых травм и переломов шейки бедра. Помимо этого, падения выступают ключевым фактором инвалидизации, приводя к:

- зависимости от посторонней помощи;
- необходимости госпитализации;
- снижению качества жизни;
- формированию страха повторных падений.

Согласно имеющимся данным, 51–68% пациентов испытывают не менее одного падения в год, при этом у 50% наблюдаются повторные падения. После первого эпизода падения прогноз выживаемости приближается к показателям при атипичных паркинсонических синдромах, для которых характерно более агрессивное течение и сокращенная продолжительность жизни [20].

В большинстве случаев падения обусловлены не внешними факторами (такими как скольжение или спотыкание), а внутренними нарушениями постурального контроля.

1.5. Датчики для регистрации походки

Ходьба человека представляет собой сложный биомеханический процесс, регулируемый взаимодействием нейромышечных, скелетно-суставных и гуморальных факторов. Нарушения походки могут возникать вследствие патологий центральной и периферической нервной системы, опорно-двигательного аппарата, эндокринных расстройств или болевого синдрома. Для объективной оценки локомоторной функции применяются различные инструментальные методы, основанные на использовании специализированных датчиков, регистрирующих кинематические, динамические и временные параметры движения.

Современные технологии анализа походки включают клинический анализ движений:

- Инерциальные датчики (акселерометры, гироскопы, магнитометры) – обеспечивают измерение ускорений, угловых скоростей и ориентации сегментов тела в трехмерном пространстве. Данные системы широко применяются благодаря портативности и возможности использования вне лабораторных условий;
- Динамометрические платформы – регистрируют вертикальные, передне-задние и медиолатеральные составляющие реакции опоры с высокой точностью. Современные платформы используют пьезоэлектрические, тензометрические или кварцевые датчики, обеспечивая цифровую передачу данных в режиме реального времени;
- Электромиографию (ЭМГ) – позволяет оценивать активность мышц в процессе ходьбы, что особенно важно при диагностике нейрогенных нарушений;

- Оптические системы захвата движения – основаны на трекинге маркеров, закрепленных на теле, и обеспечивают высокоточную регистрацию кинематики суставов;
- Подометрические системы (силовые стельки, распределительные платформы) – измеряют давление под стопой, что актуально для анализа паттернов опорной фазы шага.

В последние годы особое внимание уделяется носимым инерциальным датчикам, подометрическим система и система захвата движения (СЗД), которые позволяют анализировать походку в естественных условиях. Разрабатываются алгоритмы автоматического выделения ключевых фаз шага, таких как фаза разгрузки, что имеет важное значение для диагностики двигательных расстройств, в том числе при БП [19,21].

1.6. Исследования аномалий в походке с использованием алгоритмов ML

Методы машинного обучения активно применяются в анализе походки, так как они дают возможность разрабатывать автоматизированные системы для дифференциации здоровых людей и пациентов с нейродегенеративными расстройствами, а также для определения стадии болезни — от начальной до запущенной.

После выделения и отбора ключевых признаков используются алгоритмы классификации, позволяющие строить прогностические модели. Эти модели затем применяются для оценки вероятности отнесения новых данных к заданным классам. Многочисленные исследования посвящены поиску оптимальных комбинаций признаков и классификаторов с целью повышения точности диагностики и отслеживания прогрессирования заболеваний. Среди распространённых методов классификации можно выделить [19]:

- k-ближайших соседей (kNN);
- наивный байесовский классификатор (NB);
- линейный дискриминантный анализ (LDA);
- деревья решений (DT);
- случайный лес (RF);
- метод опорных векторов (SVM);
- нейронные и глубокие нейронные сети.

1.6.1. SVM

Метод опорных векторов (SVM) — один из самых популярных алгоритмов машинного обучения с учителем, активно применяющийся в анализе походки. Его ключевая идея заключается в построении оптимальной разделяющей гиперплоскости между двумя классами данных. Оптимальность здесь означает максимальный зазор — наибольшее расстояние до ближайших точек каждого класса (так называемых опорных векторов).

SVM может работать как с линейной, так и с нелинейной разделяющей границей — это зависит от выбора ядерной функции. Ядра позволяют преобразовывать данные в пространство более высокой размерности, где становится возможным эффективное разделение классов.

В исследовании [22] использовался нелинейный SVM с радиально-базисным ядром (RBF) для классификации пациентов с БП и двумя другими нейродегенеративными заболеваниями. Авторы тестировали классификатор на различных наборах статистических признаков, полученных из временных параметров походки. Наилучший результат (точность 83,3%) был достигнут при использовании комбинации из 7 наиболее информативных признаков.

1.6.2. Метод K-ближайших соседей

Метод k-ближайших соседей (kNN) — один из самых простых алгоритмов классификации, работающий на основе принципа схожести: похожие объекты в данных обычно расположены близко друг к другу. Ключевым параметром здесь является число k — количество учитываемых соседей. Если k слишком мало, алгоритм может игнорировать важные закономерности, что приведёт к ненадёжной классификации. С другой стороны, слишком большое k может включить в анализ далёкие и нерелевантные объекты, ухудшая точность предсказаний.

В исследовании [23] kNN применялся для диагностики лёгких когнитивных нарушений у пациентов с БП. Анализировались пространственно-временные характеристики походки в трёх режимах: обычная ходьба, двойная двигательная задача и двойная когнитивная задача. Алгоритм показал точность 83,8% и чувствительность 88,2% при выявлении пациентов с БП во время выполнения двигательного теста. Эти результаты подтверждают существование связи между особенностями походки и когнитивными функциями.

1.6.3. Наивный байесовский классификатор

Наивный байесовский классификатор — это вероятностный метод классификации, опирающийся на теорему Байеса и формулу полной вероятности. В отличие от полного

байесовского классификатора, он работает в упрощённом режиме, предполагая условную независимость признаков. Хотя в реальности это предположение редко выполняется, алгоритм демонстрирует достаточно высокую эффективность на практике.

В задачах диагностики нейродегенеративных заболеваний по данным анализа походки наивный [24] байесовский классификатор применялся как для обработки кинематических параметров [39], так и в комбинации с пространственно-временными характеристиками и RoM [16]. Точность классификации в этих исследованиях составила 83,1–90% и 59,74–78,02% соответственно. Однако эти показатели оказались ниже, чем у таких методов, как SVM, k-NN, LDA и деревья решений, что подтверждает значительное влияние предположения о независимости признаков на качество работы наивного Байеса.

1.6.4. Дерево решений, случайный лес, градиент

Расширенное дерево (GBT) - это так называемые древовидные алгоритмы. Дерево решений (DT) - самое простое из них, которое, по сути, напоминает своего рода диаграмму принятия решений, состоящую из корневого узла, нескольких древовидных узлов и листьев. Классификация выполняется, начиная с корневого узла и следуя по дереву узлов на основе истинности условия, выраженного в каждом узле. Таким образом, можно дойти до листового узла, представляющего прогноз для текущего образца.

Случайный лес (RF) и GBT — это ансамблевые методы, которые объединяют большое количество деревьев, каждое из которых обучается на случайно выбранном подмножестве признаков. В конце процесса RF объединяет результаты путём усреднения или с помощью «правил большинства». RF строит все деревья одновременно и независимо. GBT, напротив, строит по одному дереву за раз, постепенно используя информацию о ранее построенных деревьях для повышения точности.

Метод RF с правилом большинства использовался в [25] для классификации пациентов с БП и здоровых людей. Алгоритм обеспечил очень высокую точность классификации — около 98,04% при использовании полного набора характеристик во временной и частотной областях, извлечённых из сигналов VGRF (база данных «Походка при болезни Паркинсона»). Аналогичным образом, RF и DT используются в [26] для распознавания пациентов с БП с получением относительно высокой точности классификации: 89,4% и 87,21% соответственно. В этом случае пространственно-временные и кинематические характеристики извлекаются из сигналов VGRF из подмножества базы данных «Походка при болезни Паркинсона».

1.7. Исследования аномалий в походке с использованием алгоритмов DL

Помимо машинного обучения также применяется глубокое обучения в различных областях распознавания образов, включая анализ походки. Их основное преимущество заключается в способности автоматически выявлять значимые закономерности непосредственно из исходных данных, что устраняет необходимость трудоемкого ручного извлечения признаков. Среди методов глубокого обучения особой популярностью пользуются сверточные нейронные сети (CNN), которые изначально разрабатывались для анализа изображений, но успешно адаптируются и для работы с другими типами последовательностей данных.

CNN обладают многослойной архитектурой, включающей сверточные слои с обучаемыми фильтрами, слои подвыборки, нормализации и полносвязные классифицирующие слои. Важной особенностью этих сетей является способность выявлять иерархические закономерности — от простых локальных признаков до сложных глобальных паттернов, не требуя при этом чрезмерного усложнения модели.

В исследовании [27] сравнивались различные архитектуры CNN для классификации тяжести заболевания Хантингтона на основе анализа изображений отпечатков стоп. Предложенный метод, сочетающий предварительно обученную модель VGG16 с процедурой взвешенного группирования, продемонстрировал точность классификации 89%. Для сравнения, SVM показал точность 76,9% при работе только с изображениями и 86,9% при использовании комбинации изображений и пространственно-временных признаков. Эти результаты подтверждают, что CNN эффективно работают с визуальными данными, тогда как SVM лучше справляется с признаками высокого уровня.

В работе [28] CNN применялась для классификации стадий болезни Альцгеймера по данным акселерометра, фиксирующим ускорение в трех осях (X, Y, Z). Метод показал высокую точность: 89% для ранней стадии, 93% для средней и 91% для поздней. Это подтверждает способность CNN анализировать сложные временные последовательности с вариативной длиной, выявляя в них значимые для диагностики закономерности.

Еще одно перспективное направление — использование рекуррентных архитектур, таких как сети с долгой краткосрочной памятью (LSTM). В исследовании [29] двунаправленная LSTM применялась для оценки риска падений у пациентов с БП на основе пространственно-временных параметров походки, полученных с помощью датчиков IMU. Модель учитывала как временные изменения, так и асимметрию походки, демонстрируя точность

92,1% и превосходя традиционные методы (SVM, RF, MLFNN). Это подчеркивает преимущество LSTM в работе с последовательностями, где важно учитывать долгосрочные зависимости в данных

2. Практическая часть

В рамках данного командного проекта были использованы 2 датасета:

Датасет 1: Gait in Parkinson's Disease (Jeffrey Hausdorff) ссылка: https://physionet.org/content/gaitpdb/1.0.0/

Датасет 2: PD-BioStampRC21: Parkinson's Disease Accelerometry Dataset from Five Wearable Sensor Study (Jamie L. Adams) ссылка: https://ieee-dataport.org/open-access/pd-biostamprc21-parkinsons-disease-accelerometry-dataset-five-wearable-sensor-study-0

На GitHub была создана директория: https://github.com/pdkit/pdkit

2.1. Датасет 1

База данных содержит комплексную сопутствующую информацию, включая демографические характеристики участников и клинические показатели тяжести заболевания (с использованием шкалы Хен-Яра и UPDRS)

Данные представлены в удобных форматах **txt, xsl, html**, что обеспечивает простоту их обработки и анализа. Особый научный интерес представляет подмножество данных, записанных во время когнитивно-моторного тестирования - выполнения арифметического задания (последовательное вычитание семерки) при ходьбе. Как видно из приведенных графиков:

- У пациентов с БП наблюдается значимо большая вариабельность шага (CV=2.7%) по сравнению с контрольной группой (CV=1.3%) в обычных условиях ходьбы
- При выполнении двойной задачи (ходьба+счет) разница становится еще более выраженной:
- Группа БП: CV возрастает до 6.5%
- Контрольная группа: сохраняет стабильность (CV=1.2%)

Для нашей задачи использовались только данные с походкой без дополнительного задания.

2.1.1. Структура датасета

Набор данных представлен в текстовом формате, где каждая строка содержит 19 столбцов с показателями, записанными с частотой 100 Гц.

Порядок столбцов организован следующим образом:

1: Временная метка (секунды)

2-9: Вертикальная сила реакции опоры (VGRF) для 8 датчиков левой стопы (Ньютоны)

10-17: VGRF для 8 датчиков правой стопы

18: Суммарная сила под левой стопой

19: Суммарная сила под правой стопой

Расположение датчиков:

Датчики в стельках расположены в следующих относительных координатах (условные единицы, таблица 2)

Таблица 2 – Расположение датчиков

Датчик	Левая стопа (X,Y)	Правая стопа (X,Y)
1	(-500, -800)	(500, -800)
2	(-700, -400)	(700, -400)
3	(-300, -400)	(300, -400)
4	(-700, 0)	(700, 0)
5	(-300, 0)	(300, 0)
6	(-700, 400)	(700, 400)
7	(-300, 400)	(300, 400)
8	(-500, 800)	(500, 800)

Эта система координат позволяет рассчитывать примерное положение центра давления для каждой стопы во время движения.

Система именования файлов:

Файлы данных следуют единому шаблону, например: **GaCo01_02.txt** или **JuPt03_06.txt**, где:

Первые 2 буквы обозначают исследование:

- Ga: Исследование двойной задачи при БП (Yogev et al.)
- Ju: Исследование RAS при БП (Hausdorff et al.)
- Si: Исследование ходьбы на беговой дорожке (Frenkel-Toledo et al.)

Следующие 2 буквы:

- Co: Участник контрольной группы
- Pt: Пациент с болезнью Паркинсона

Цифры после подчеркивания:

- 01: Обычная ходьба
- 10: Ходьба с выполнением задания (серийное вычитание 7, не используется в обучении)

2.1.2. Анализ датасета

Перед обучением модели был проведен небольшой анализ данного датасета для общего понимания.

На рисунке 1 представлены гистограммы распределения пациентов по полу.

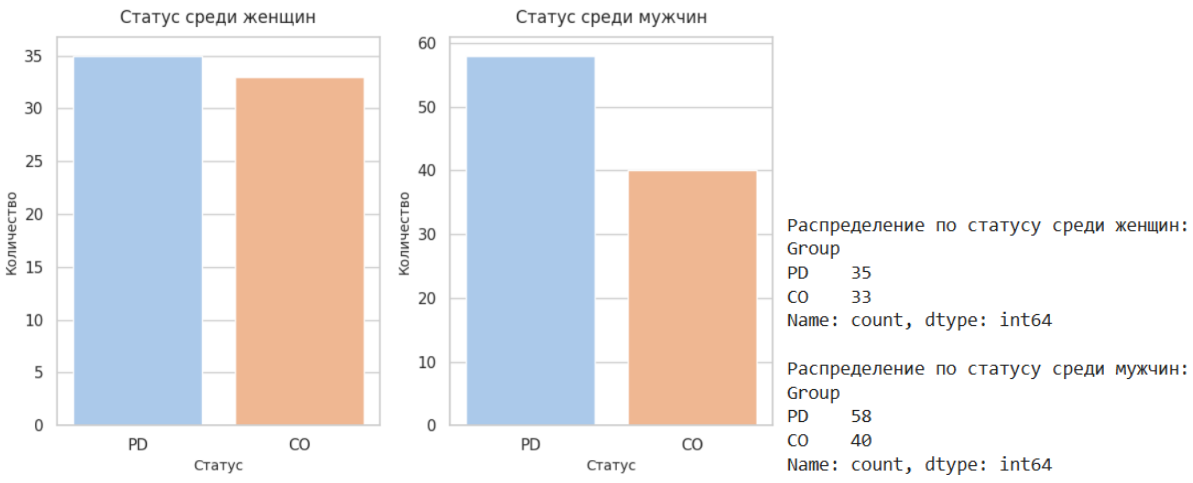


Рисунок 1 – Гистограмма распределения по полу

По ней видно, что женщин примерно одинаковое количество в обоих группах, а у мужчин больше в группе с БП.

На рисунке 2 представлена гистограмма распределения возраста по полу и статусу.

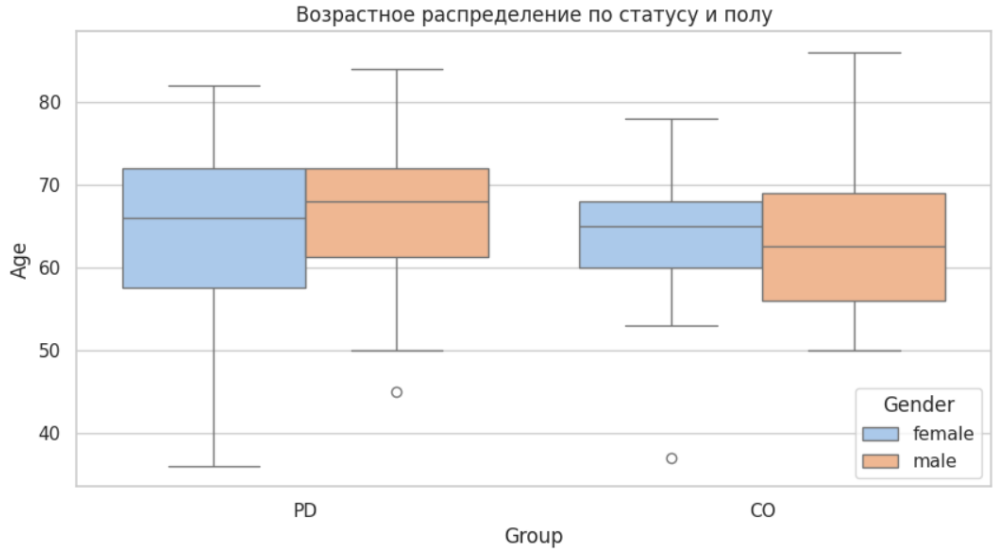


Рисунок 2 – Распределение по полу и статусу

Данные охватывают характерную для БП возрастную группу 40-70 лет с пиком представленности в 50-60 лет, что соответствует типичной эпидемиологической картине заболевания. Исследование включает как пациентов (PD), так и контрольную группу (CO) с учетом гендерного распределения, что позволяет проводить сравнительный анализ с минимизацией влияния демографических факторов.

В данном датасете также присутствует оценка пациентов по разным шкалам:

- UPDRS (Unified Parkinson's Disease Rating Scale) - комплексная шкала оценки симптомов БП. Максимальная оценка 199 баллов и соответствует максимальной (полной) инвалидности, оценка 0 инвалидности нет.
- UPDRS-M (моторная часть UPDRS) - оценивает только двигательные нарушения. Максимальная оценка 40, минимальная 0.
- TUAG (Timed Up and Go Test) - тест на подвижность: ≤10 сек - норма, 11-20 сек - умеренные нарушения (пожилые люди), ≥20 сек - выраженные проблемы с движением.
- Hoehn & Yahr - определяет стадии прогрессирования БП

Таблица 3 – Шкала Hoehn & Yahr

Этап	Шкала Хена и Яра	Модифицированная шкала Хоэна и Яр
1	Одностороннее поражение, как правило, сопровождается минимальной функциональной недостаточностью или ее отсутствием	Только одностороннее участие
1.5	-	Одностороннее и осевое вовлечение
2	Двустороннее поражение или поражение средней линии без нарушения равновесия	Двустороннее участие без нарушения баланса
2.5	-	Лёгкое двустороннее заболевание с восстановлением при проведении теста на вытягивание
3	Двустороннее поражение: легкая или умеренная инвалидность с нарушением постуральных рефлексов; физическая независимость	Двустороннее заболевание легкой или средней степени тяжести; некоторая нестабильность позы; физическая независимость
4	Тяжелое инвалидизирующее заболевание; пациент все еще может ходить или стоять без посторонней помощи	Тяжелая инвалидность; может ходить или стоять без посторонней помощи
5	Прикованы к постели или инвалидному креслу без посторонней помощи	Прикован к инвалидному креслу или постели без посторонней помощи

Эти шкалы используются для объективной оценки состояния пациентов и подбора терапии.

На рисунке 3 представлены результаты по шкале Hoehn & Yahr.

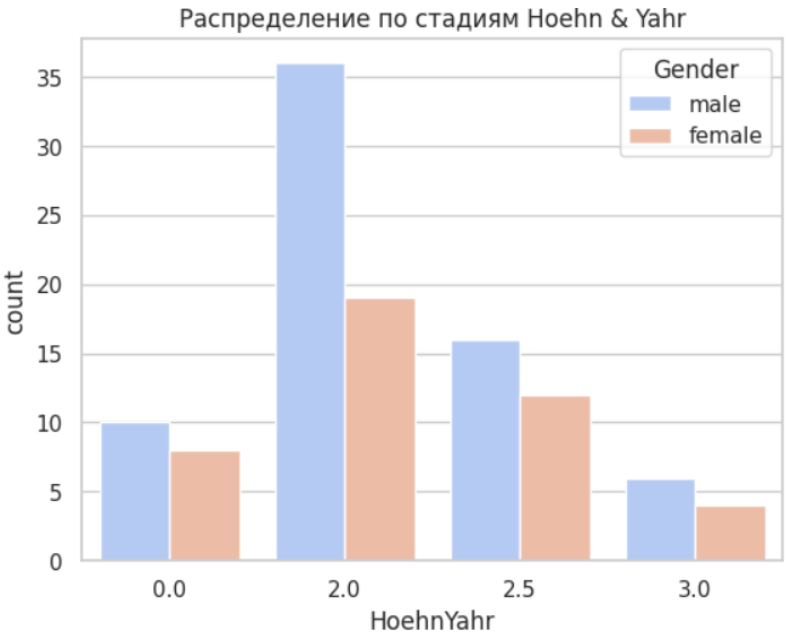


Рисунок 3 – Шкала Hoehn & Yahr для датасета 1

По гистограмме видно, что в данном датасете есть только люди с оценкой по шкале Хена и Яра 0, 2, 2.5, 3.

На рисунке 4 представлены результаты по всем шкалам.

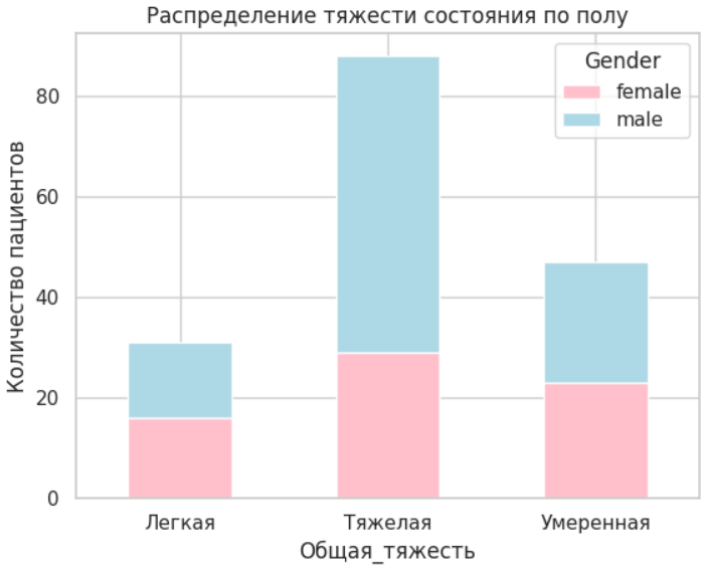


Рисунок 4 – Шкала UPDRS для датасета 1

По гистограмме видно, что в данном датасете преимущественно находятся люди с тяжелой степенью тяжести.

Также была построена тепловая матрица корреляции. Она представлена на рисунке 5.

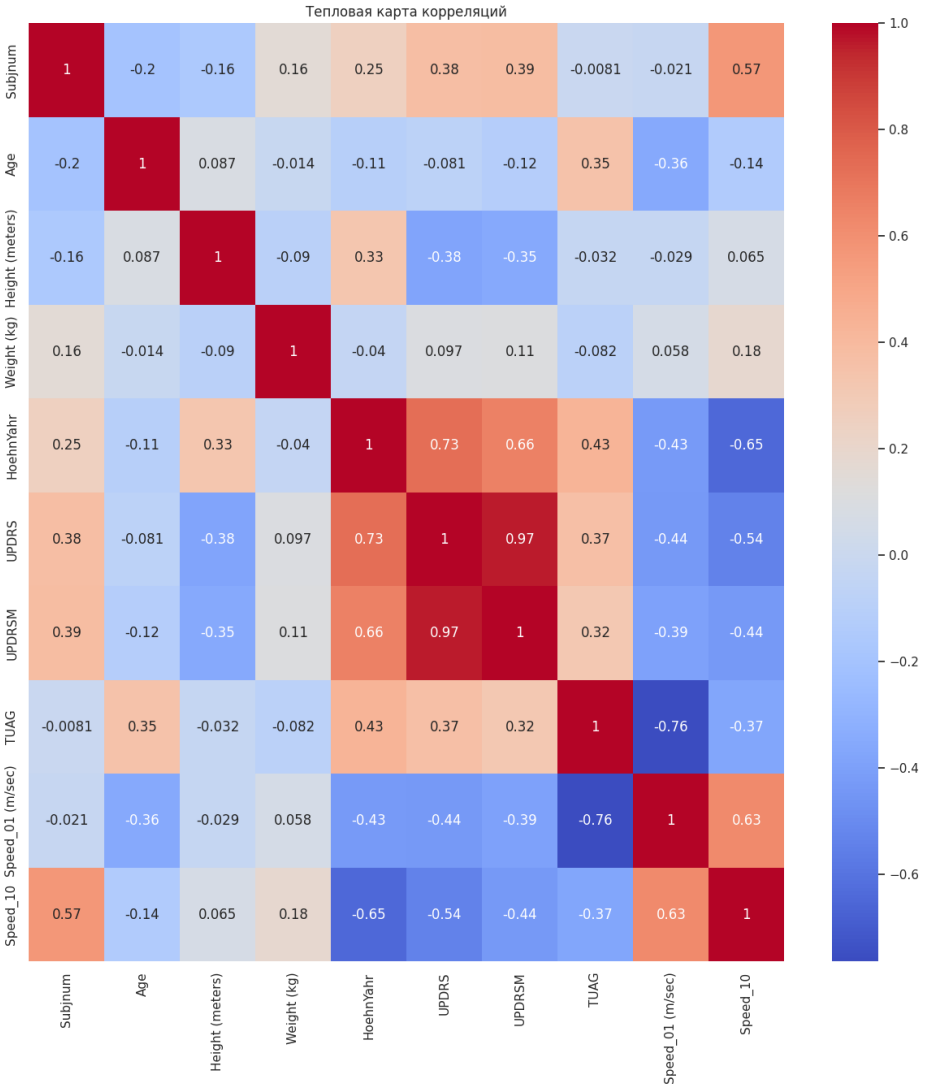


Рисунок 5 – Тепловая матрица корреляции для датасета 1

Наиболее сильная корреляция наблюдается между общим баллом UPDRS и его моторной частью UPDRSM (r=0.97), что подтверждает взаимосвязанность этих шкал в оценке симптоматики. Умеренные положительные корреляции шкалы Хена-Яра с UPDRS (r=0.73) и UPDRSM (r=0.66) свидетельствуют о логичном нарастании двигательных нарушений по мере прогрессирования стадии заболевания. Высокая отрицательная корреляция между временем выполнения теста TUAG и скоростью ходьбы (r=-0.76) демонстрирует чувствительность этих показателей к нарушениям мобильности. При этом слабые корреляции антропометрических показателей (веса и роста) с клиническими параметрами (|r|<0.2) указывают на их незначительное влияние на выраженность симптомов.

2.1.3. Используемые датчики

При болезни Паркинсона укорочение шага и шаркающая походка существенно изменяют распределение нагрузки на стопу. Это происходит из-за характерных двигательных

нарушений, связанных с дисфункцией базальных ганглиев и дефицитом дофамина. В норме при ходьбе стопа совершает плавный перекат с пятки на носок, но у пациентов с паркинсонизмом этот естественный механизм нарушается. Основная проблема заключается в том, что шаги становятся короче, а стопа перестает полноценно отрываться от поверхности. Вместо четкого последовательного движения - удар пяткой, перекат через средний отдел и толчок носком - стопа часто опускается на землю всей плоскостью или даже передним отделом. Это приводит к перераспределению нагрузки, при которой передняя часть стопы испытывает чрезмерное давление.

Система Computer Dyno Graphy (CDG)® (Infotronic, Нидерланды, http://www.infotronic.nl) использовалась для регистрации вертикальных сил реакции опоры как функции времени от обеих стоп во время ходьбы.

Эта система состоит из обуви, изготовляемой по индивидуальному размеру стопы, с 8 датчиками силы, встроенными в подошву (рисунок 6). В данном исследовании использовались два размера обуви: 35–40 и 40–45. Ботинки имеют кожаную подошву, состоящую из двух отдельных частей, соединенных эластичной вставкой, и эластичный верх, который можно затягивать для плотного прилегания к босой стопе испытуемого. Это обеспечивает плотное прилегание подошвы ботинка к стопе, даже если индивидуальный размер стопы незначительно варьируется в пределах одного размера обуви [1].

Рисунок 6 - Внешний вид системы регистрации Computer Dyno Graphy

Расположение датчиков на подошве показано на рисунке 7. Эта система позволяет записывать данные о VGRF с обеих стоп во время последовательных шагов в течение нескольких циклов походки за один сеанс ходьбы.

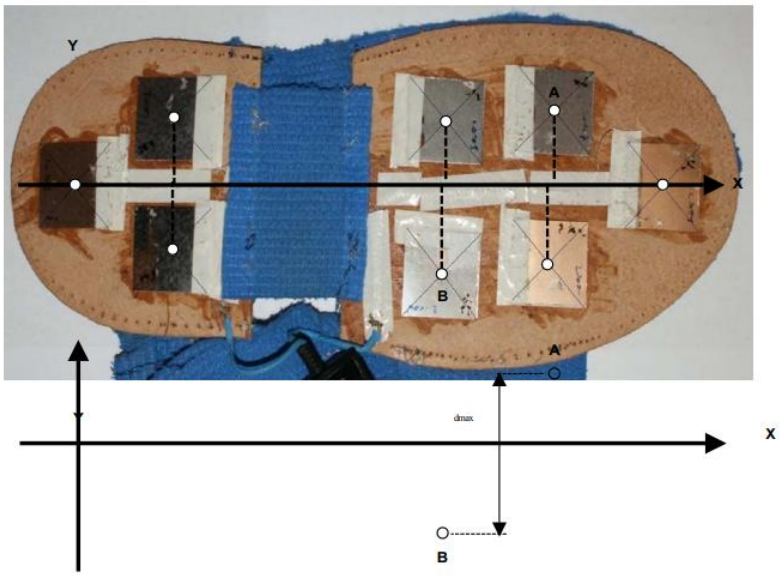


Рисунок 7 - Внутреннее строение системы регистрации Computer Dyno Graphy

Изменения походки при болезни Паркинсона приводят к тому, что фаза переката через стопу укорачивается, а отталкивание носком становится менее эффективным. В результате основная нагрузка приходится на плюсневые кости и подушечки стопы. Данным областям на стопе соответствуют датчики 4,5,6 и 12,13,14, изображённые на рисунке 8 что позволяет сделать предположение о наибольшей информативности данных датчиков.

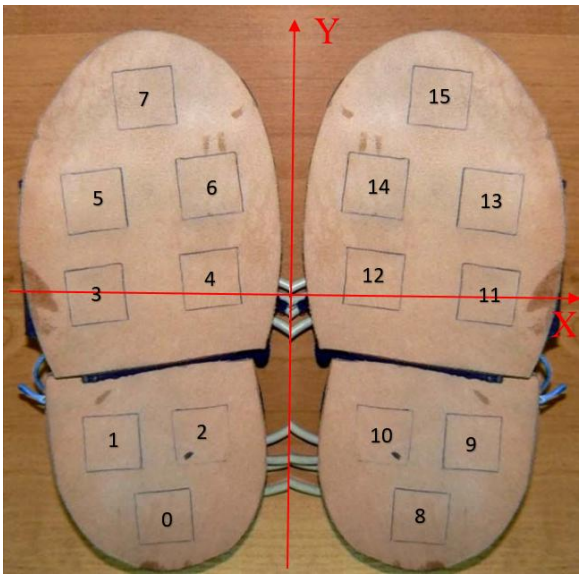


Рисунок 8 - Расположение датчиков в системе регистрации Computer Dyno Graphy

Для сбора исследуемого датасета каждый участник шел в самостоятельно выбранном «комфортном» темпе в течение 20 секунд, что соответствовало примерно 38 шагам.

Датчики силы записывали данные с частотой 50 Гц, которые сохранялись в регистраторе данных, закрепленном на поясе испытуемого.

2.1.4. DL

Целью данной работы является определение наиболее информативных датчиков для классификации болезни Паркинсона на основе временных рядов данных, полученных с сенсоров. Исходный набор данных содержит 18 признаков, сгруппированных в 9 пар, соответствующих датчикам, расположенным симметрично на теле пациента. Для решения задачи применяется метод итеративного отбора признаков, направленный на максимизацию качества классификации.

Для определения наиболее значимых пар признаков используется жадный алгоритм обратного отбора (backward feature elimination), который на каждом шаге исключает наименее информативную группу признаков. Критерием отбора служит изменение среднего значения метрики ROC-AUC на кросс-валидации. Процесс продолжается до тех пор, пока исключение следующей группы признаков не приводит к значимому ухудшению качества модели (падению ROC-AUC ниже заданного порога).

Для классификации используется ансамбль нейронных сетей следующей архитектуры:

- **Свёрточные слои (Conv1D):**

Применяются для автоматического выделения локальных паттернов в временных рядах. Используется 4 последовательных слоя с количеством фильтров 32, 64, 128 и 128 соответственно, ядром размером 3 и активацией ReLU. После каждого свёрточного слоя добавляются:

- **BatchNormalization для ускорения обучения,**
- **Dropout (p=0.25) для регуляризации,**
- **MaxPooling1D (pool_size=2) для уменьшения размерности.**
- **Слой LSTM:**

Обрабатывает последовательные зависимости в данных с размерностью 64.

- **Полносвязные слои (Dense):**

Включают скрытый слой с 64 нейронами и ReLU-активацией, а также выходной слой с сигмоидной активацией для бинарной классификации.

- Модель обучается с оптимизатором Adam (learning rate=1e-4) и функцией потерь binary_crossentropy.

Для повышения устойчивости оценки качества модели применяется 5-кратная стратифицированная кросс-валидация. На каждой итерации отбора признаков обучается ансамбль из 5 моделей, а их предсказания усредняются. Это позволяет снизить дисперсию оценки ROC-AUC и избежать переобучения.

Процесс отбора признаков прекращается при выполнении одного из условий:

- Остаётся только одна группа признаков.
- Удаление следующей группы приводит к падению ROC-AUC ниже 90% от максимального достигнутого значения (параметр min_thresh=0.9).

Финальная оценка качества модели проводится на отложенной тестовой выборке (18% данных). Используются следующие метрики:

- ROC-AUC (площадь под ROC-кривой) — основная метрика, устойчивая к дисбалансу классов.
- Accuracy, Precision, Recall, F1-score — дополнительные метрики для оценки эффективности классификации.

Для анализа процесса отбора признаков строится график зависимости ROC-AUC и Accuracy от количества оставшихся групп признаков.

Все этапы работы реализованы в среде Python с использованием библиотек:

- TensorFlow/Keras — построение и обучение нейронных сетей.
- Scikit-learn — кросс-валидация, метрики качества, разделение данных.
- Pandas/NumPy — обработка данных.
- Matplotlib — визуализация результатов.

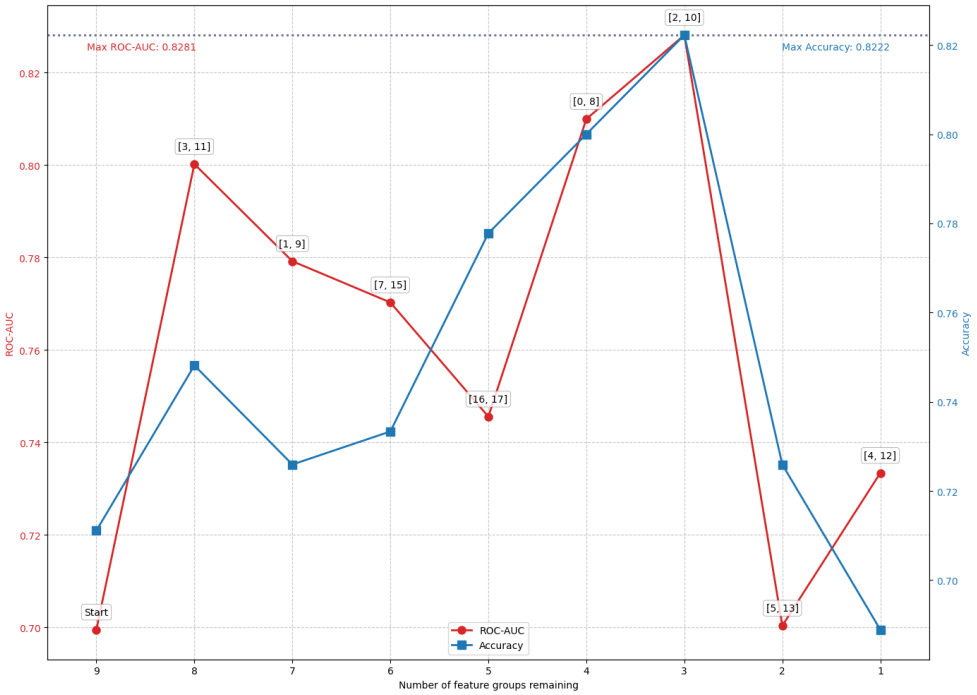


Рисунок 9 – ROC-AUC и Accuracy

Таким образом, датчики [4, 12], [5, 13], [6,14] способствуют построению ансамбля моделей с наилучшим качеством классификации.

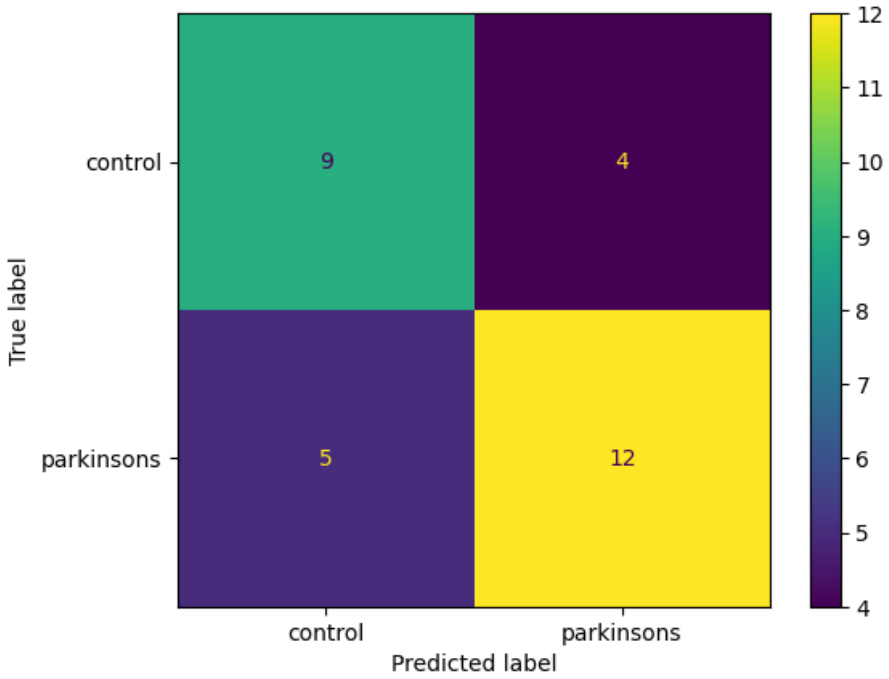


Рисунок 10 – Матрица потерь

Результаты представлены в таблице 4:

Таблица 4 – Оценка ансамбля моделей

Метрика	**Значение**
Accuracy	0.7000
Precision	0.7500
Recall	0.7059
Specificity	0.6923
O1 (False Omission Rate)	0.1333
O2 (False Negative Rate)	0.2941
ROC-AUC	0.8009

Выводы:

- ROC-AUC = 0.8009 свидетельствует о хорошей разделяющей способности модели
- Precision (75%) выше, чем Recall (~71%), что указывает на осторожность модели в постановке диагноза (меньше ложных положительных случаев)

2.2. Датасет 2

Датасет PD-BioStampRC21 содержит данные, полученные с носимых сенсоров (MC10 BioStamp RC) для изучения двигательных симптомов БП, включая активность, походку, тремор и другие проявления. В исследовании участвовали 17 пациентов с БП и 17 здоровых добровольцев (контрольная группа). Данные записывались в течение ~2 дней с помощью пяти сенсоров, закрепленных на теле:

- Грудь ("ch");
- Передняя часть левого/правого бедра ("ll"/"rl");
- Передняя часть левого/правого предплечья ("lh"/"rh").

Основные клинические оценки включали MDS-UPDRS (двигательные симптомы) и фиксацию состояния пациентов (ON/OFF медикаментозная терапия). Датасет состоит из двух частей:

- Сенсорные данные и аннотации (акселерометрия, временные метки оценок, 56 ГБ);
- Демографические и клинические данные (пол, возраст, тяжесть тремора и др.).

2.2.1. Структура датасета

Сенсорные данные и аннотации (папки по ID участников):

- Акселерометрия (CSV-файлы):
- Формат: "ch_ID018Accel.csv" (префикс = расположение сенсора).
- Столбцы: временная метка (мс), ускорение по осям X/Y/Z (в единицах g).
- Аннотации задач (CSV):
- Формат: AnnotID018.csv.
- Столбцы: тип задачи (MDS-UPDRS), временные интервалы, статус медикации (ON/OFF), локализация симптомов (например, RIGHT HAND).

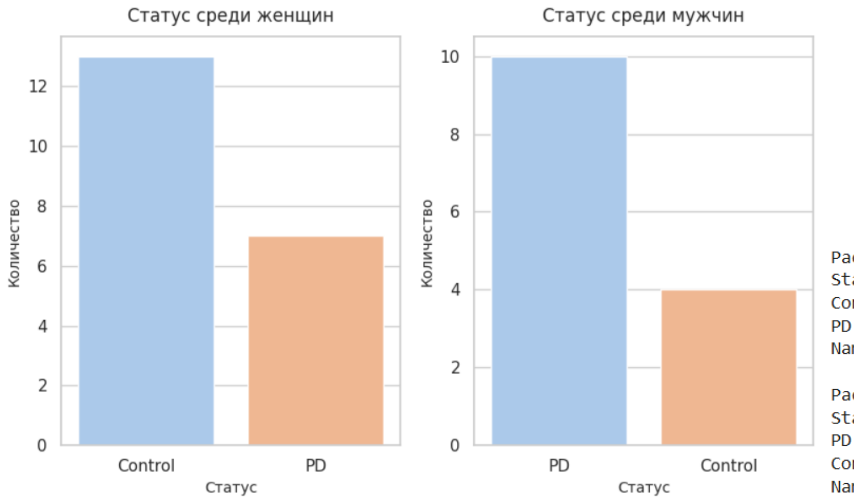
Клинические данные (CSV-файл Clinic_Data_PD-BioStampRC21.csv):

- Демография: ID, пол, статус (БП/контроль), возраст.
- Оценка тремора (MDS-UPDRS Part 3.17):
- Амплитуда тремора для 5 областей тела (RUE, LUE, RLE, LLE, Lip/Jaw).
- Данные для состояний ON и OFF (только для пациентов с БП).

2.2.2. Анализ датасета

Перед обучением модели был проведен небольшой анализ данного датасета для общего понимания.

На рисунке 11 представлены гистограммы распределения пациентов по полу.




```
Распределение по статусу среди женщин:
Status
Control    13
PD          7
Name: count, dtype: int64

Распределение по статусу среди мужчин:
Status
PD         10
Control     4
Name: count, dtype: int64
```

Рисунок 11 – Гистограмма распределения по полу

По ней видно, что здоровых женщин больше в контрольной группе, а у мужчин больше в группе с БП, как и в первом датасете.

На рисунке 12 представлена гистограмма распределения возраста по полу и статусу.

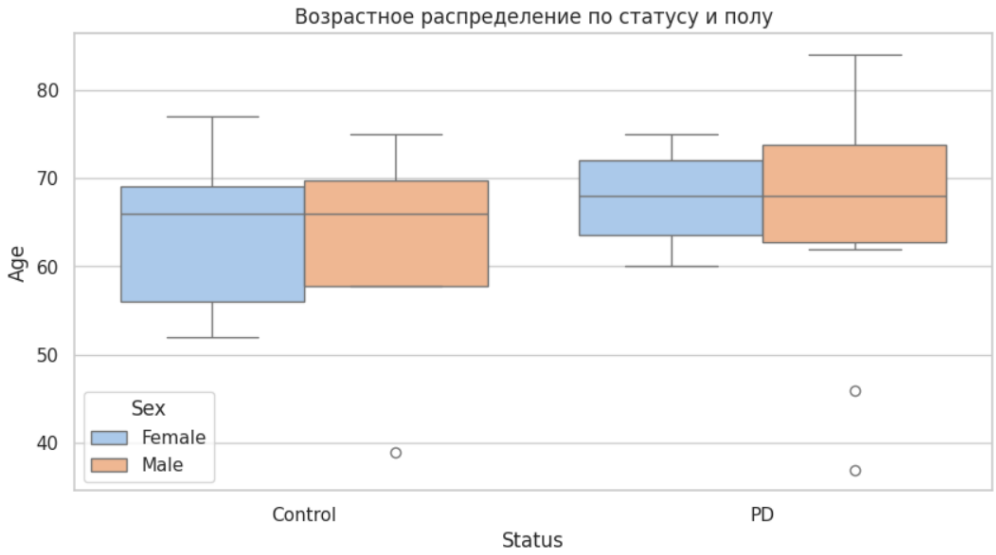


Рисунок 12 – Распределение по полу и статусу

Данные охватывают более старшею группу для людей с 65-72 с пиком представленности в 68-70 лет, что соответствует в целом типичной эпидемиологической картине заболевания. Исследование включает как пациентов (PD), так и контрольную группу (CO) с учетом гендерного распределения, что позволяет проводить сравнительный анализ с минимизацией влияния демографических факторов.

Как и для первого датасета была построена матрица корреляции. Она представлена на рисунке 13.

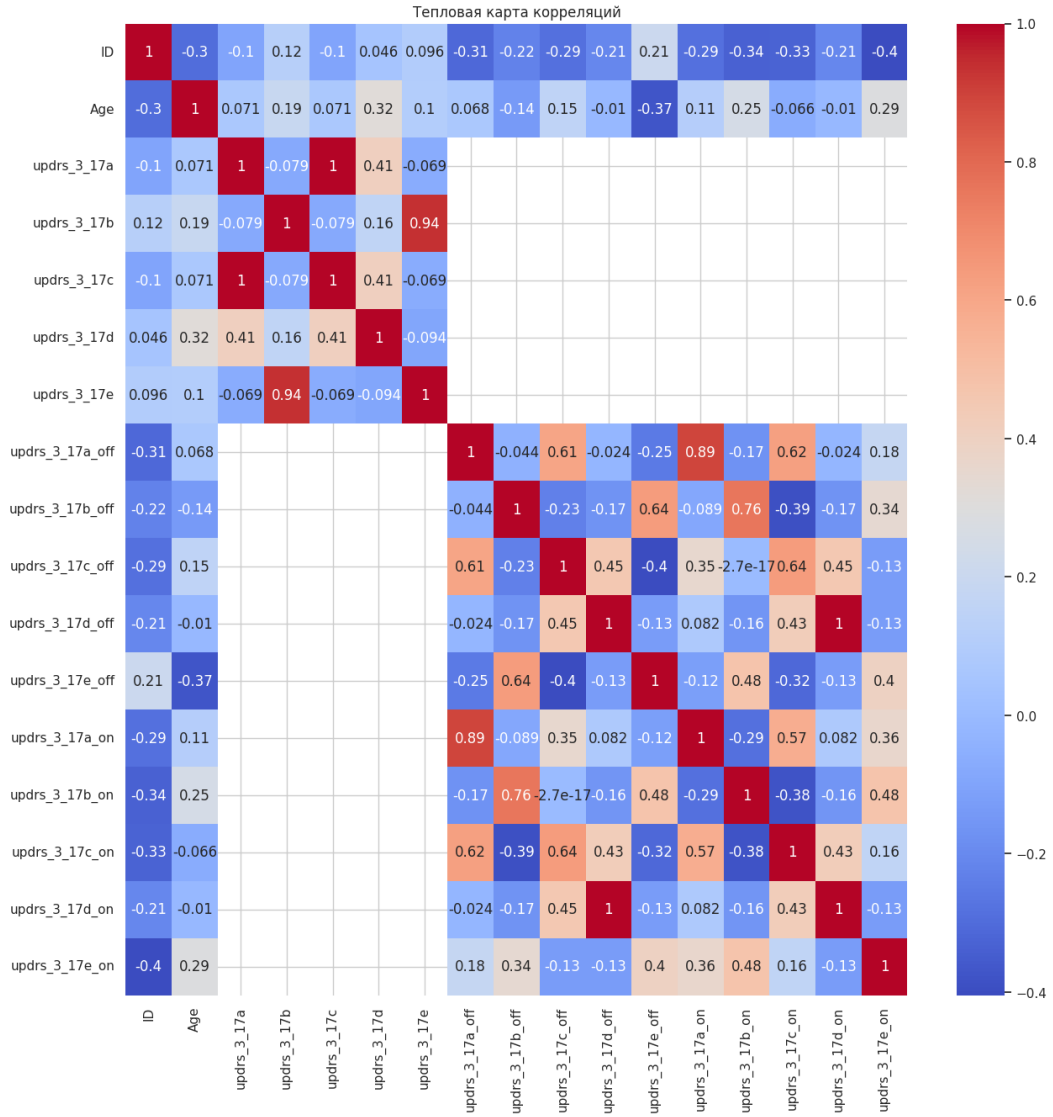


Рисунок 13 – Тепловая матрица корреляции для датасета 2

Наблюдается выраженная отрицательная корреляция между возрастом и показателями тремора в различных конечностях (updrs_3_17a-on: r=-0.4; updrs_3_17b-on: r=-0.34), что может свидетельствовать о возрастном снижении выраженности двигательных симптомов или особенностях медикаментозной терапии у пожилых пациентов. Особого внимания заслуживает сильная отрицательная корреляция между показателями тремора в верхних и нижних конечностях (updrs_3_17b и updrs_3_17e: r=-0.94), что указывает на возможную асимметричность проявления симптомов при отсутствии медикаментозной поддержки. При этом выявлены умеренные корреляции между различными локализациями тремора в состоянии "on" (значения r в диапазоне -0.25...-0.4), демонстрирующие более равномерное распределение симптоматики на фоне терапии.

2.2.3. Используемые датчики

MC10 BioStamp RC (рисунок 14) — это гибкие, носимые сенсоры, разработанные для сбора физиологических и биомеханических данных в клинических и удаленных условиях.

Рисунок 14 - Внешний вид датчика MC 10 BioStampRC

Датчик BioStamp RC предназначен для сбора определенных физиологических данных в научных исследованиях. Система предназначена для сбора исходных данных о движениях и биопотенциалах. Данные, собранные системой BioStamp RC, могут обеспечить количественный анализ физических движений и электрофизиологии.

Низкопотребляющий микроконтроллер датчика BioStamp RC обрабатывает сигналы от 3-осевого акселерометра и гироскопа, а данные датчиков обрабатываются и выбираются микроконтроллером, который передает данные во флеш-память или передает их по беспроводной связи через Bluetooth. Для настройки и управления устройством BioStamp RC пользователь мог беспроводным образом установить рабочие параметры, такие как частота выборки, тип измерения и диапазон измерения перед сбором данных, с помощью специализированного программного приложения на мобильном устройстве. Умное мобильное устройство позволяло контролировать передачу данных от датчиков BioStamp на облачный сервер для дальнейшего анализа. Датчик содержит следующие сенсорные модули:

- Высокочастотный 3-осевой гироскоп
- Акселерометр
- Аналоговый фронтальный блок для измерения sEMG и ECG

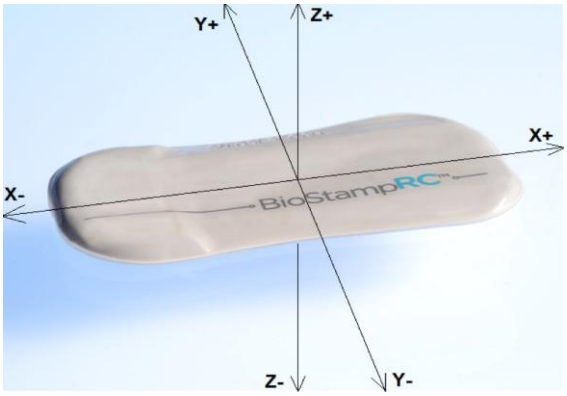


Рисунок 15 - Система координат датчика MC 10 BioStamp RC

Датчик BioStamp RC можно размещать в различных местах тела, на одном человеке может быть размещено несколько датчиков. После настройки с помощью приложения Investigator App каждый датчик работает независимо, сохраняя данные в своей локальной памяти. Затем данные могут быть переданы по беспроводной связи для отображения и последующего анализа. Датчик BioStamp RC предназначен для использования в течение 24 часов непрерывного ношения.

Датчики накладываются на тело с помощью одноразового клеевой плёнки для крепления и проводящего геля для электродов. Эти двусторонние клеевые плёнки имеют глянцевую подложку на одной стороне и матовую подложку на другой. Сторона с глянцевой поверхностью приклеивается к сенсору, а сторона с матовой - к коже. Клеевые плёнки и вкладыши имеют отверстия, чтобы электроды датчика могли соприкасаться с кожей (рисунок 16).

Рисунок 16 – Контактная часть датчика MC 10 BioStamp RC

На рисунке 17 показаны некоторые примеры рекомендуемых мест расположения датчиков BioStamp RC на теле. При расположении датчиков в других местах есть вероятность того, что они будут плохо прилегать к коже и сигнал получится неверным. Для получения качественных результатов можно использовать сразу несколько датчиков.

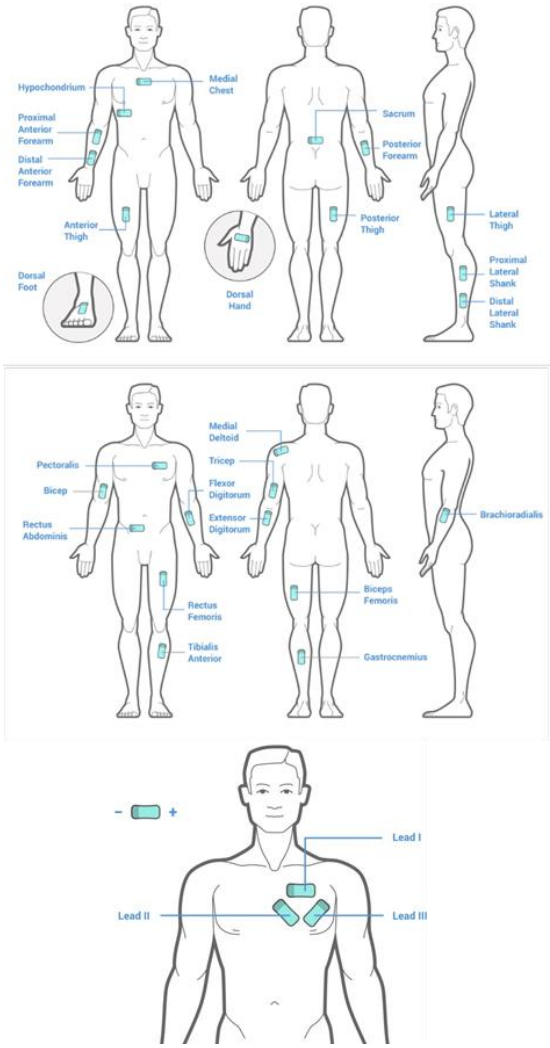


Рисунок 17 – Рекомендуемые точки крепления датчика MC 10 BioStamp RC

Датчик BioStamp RC имеет несколько настраиваемых режимов считывания. В разных режимах используются различные комбинации 3-осевого акселерометра, 3-осевого акселерометра с гироскопом и однопроводного (двухэлектродного) аналогового датчика напряжения. В таблице 5 перечислены поддерживаемые режимы записи. Время жизни записи зависит от выбранного режима записи и частоты дискретизации.

Таблица 5 - Режимы записи, поддерживаемые BioStamp RC

Режим	**Частота дискретизации (Гц)**	**Динамический диапазон**	**Максимальное время работы в (ч)**
Акселерометр	50, 100, 200	±2, ±4, ±8, ±16 G	8-35
ЭКГ	125, 250	±0.2 V	17-35

EMG	250	±0.2 V	17
Гироскоп+ Акселерометр	25, 50, 100, 250	±2, ±4, ±8, ±16 G (ускорение) ±250, ±500, ±1000, ±2000 °/сек (гироскоп)	2-4

На рисунке 18 изображено устройство датчика BioStamp RC.

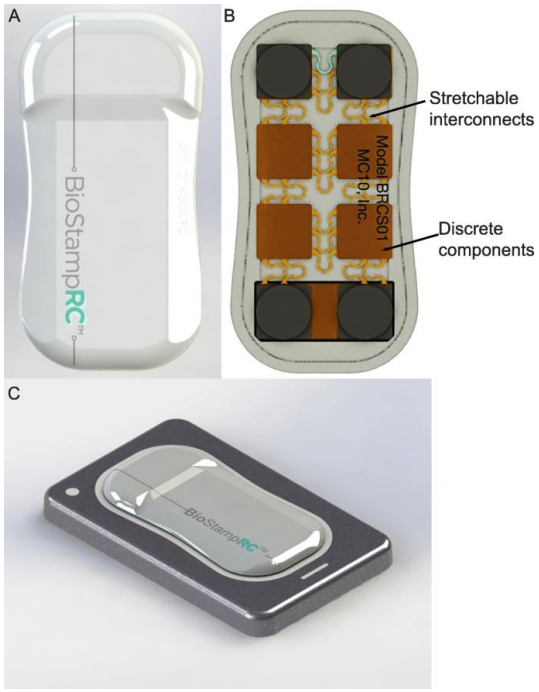


Рисунок 18 – Устройство датчика BioStamp RC

Технические характеристики датчика представлены в таблице 5.

Таблица 6 - Технические характеристики датчика BioStamp RC

Размеры	8,6 × 5,5 × 1,1 см (длина, ширина, высота)
Вес	48,5 г
Материал	Поликарбонат с прозрачным покрытием
Длина кабеля	150 ± 1,2 см
Мощность	5 В, 1 А
Частота зарядки	13,56 МГц
Класс защиты	IEC II
Сертификаты	FCC Part 18, Declaration of Conformity и UL/CSA 60950-1

Рабочая температура	20–40 °C
Водонепроницаемость	IPX7 (до 1 метра на 30 минут)
Связь	Bluetooth Smart (BLE)
Объём памяти	32 МБ
Аккумулятор	Литий-полимерный, перезаряжаемый, 15 мАч
Цена	От 1500$
Минимальная комплектация	3 датчика, сменные наклейки, гель, зарядное устройство

2.2.4. ML

Сравнение графиков, построенных по данным, полученных с использованием акселерометра, для здоровых и больных пациентов.

Пациенты 008 и 015 (рисунок 19 – 23)

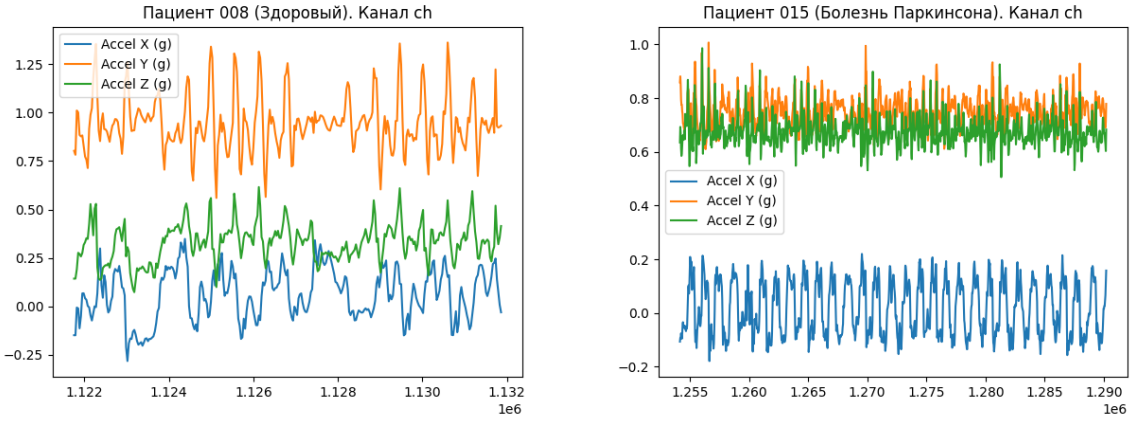


Рисунок 19 – Сигнал с грудной клетки для пациентов 008 и 015

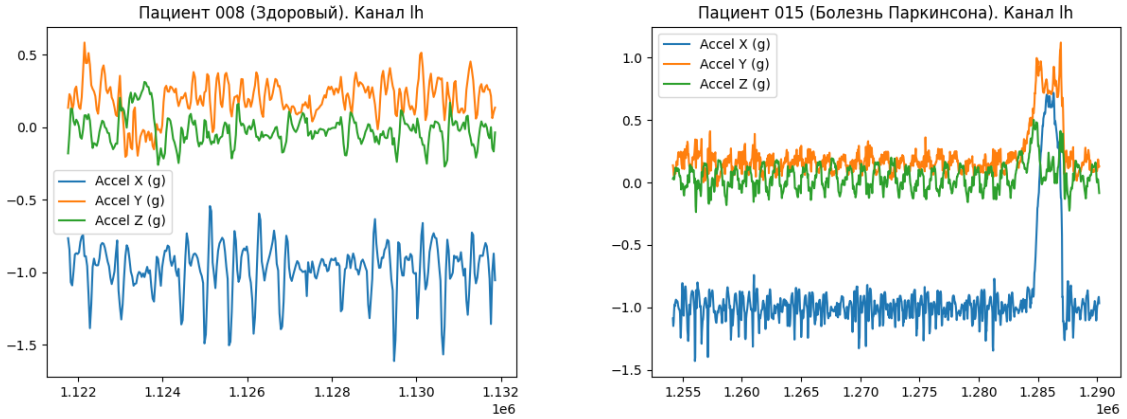


Рисунок 20 – Сигнал с левой руки для пациентов 008 и 015

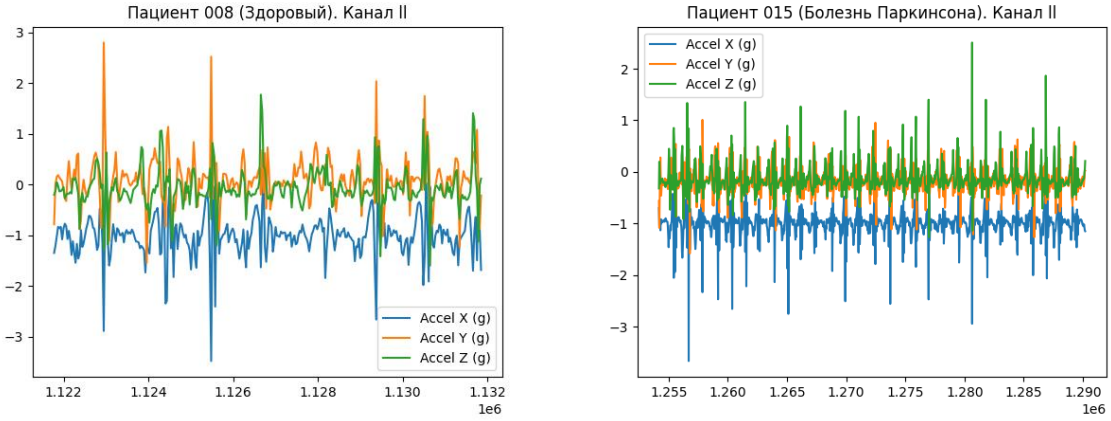


Рисунок 21 – Сигнал с левой ноги для пациентов 008 и 015

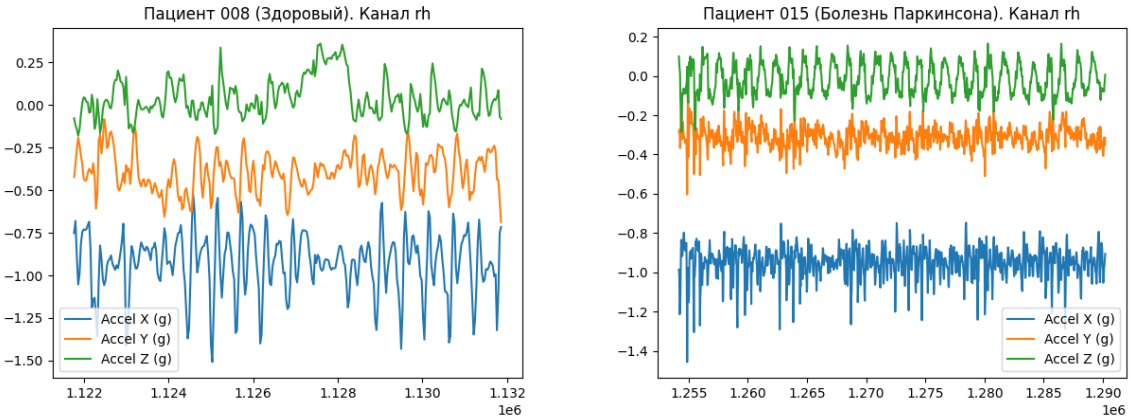


Рисунок 22 – Сигнал с правой руки для пациентов 008 и 015

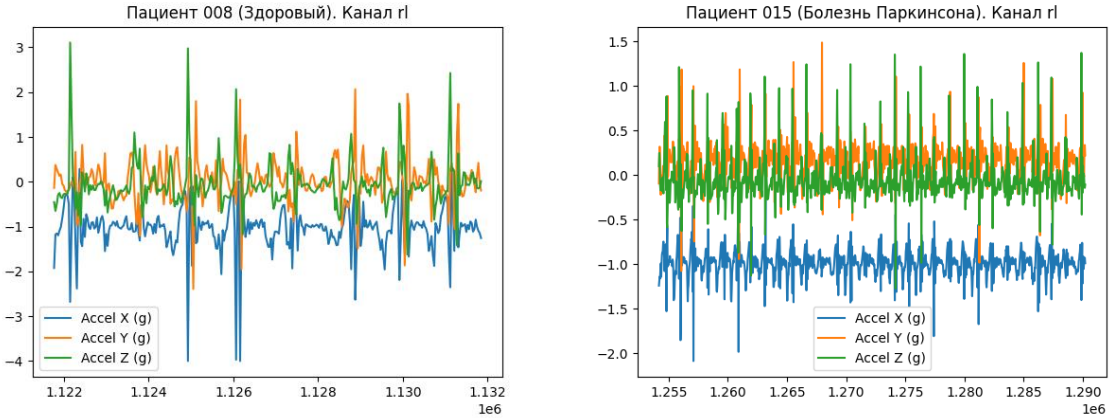


Рисунок 23 – Сигнал с правой ноги для пациентов 008 и 015

Пациенты 023 и 030 (рисунок 24 - 28)

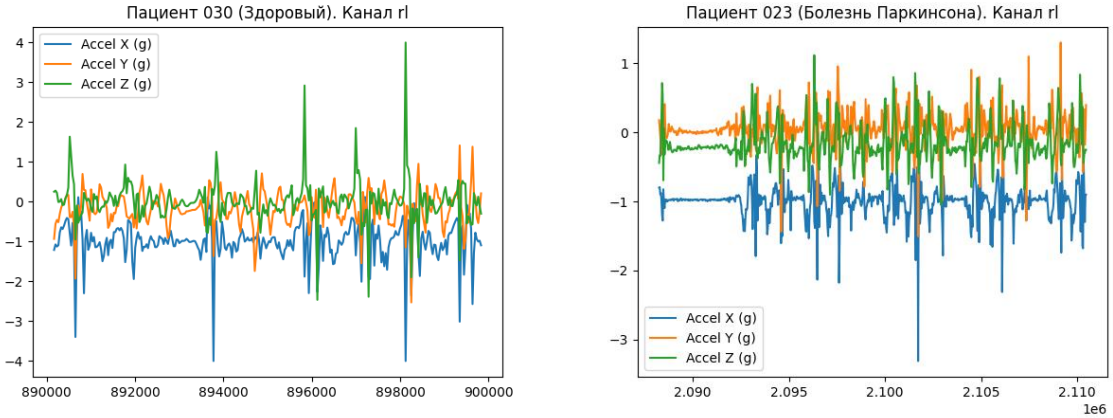


Рисунок 24 – Сигнал с правой ноги для пациентов 030 и 023

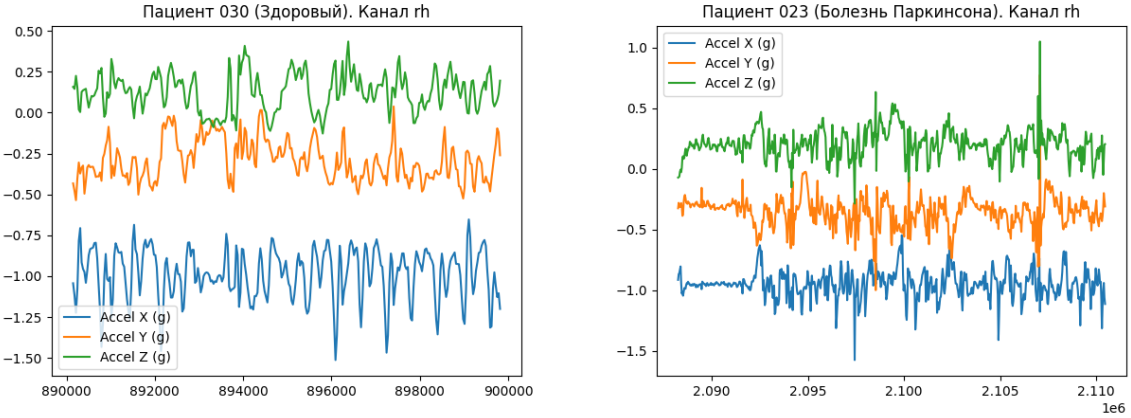


Рисунок 25 – Сигнал с правой руки для пациентов 030 и 023

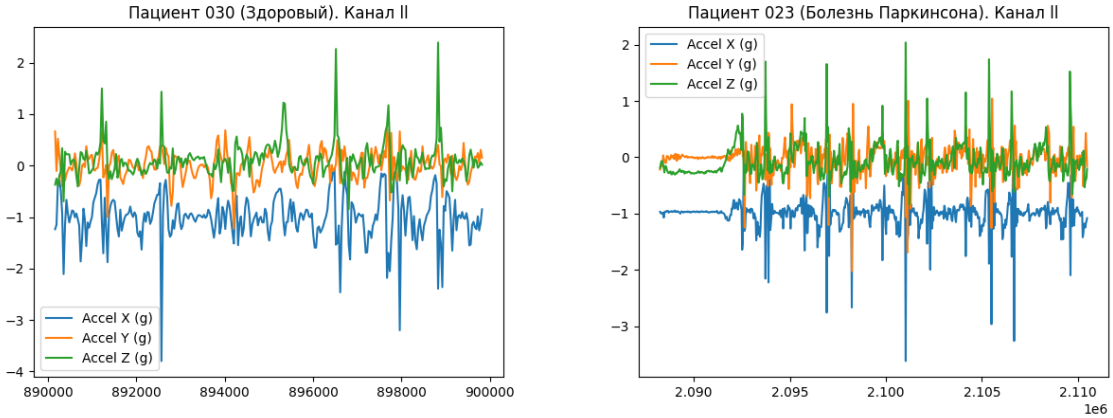


Рисунок 26 – Сигнал с левой ноги для пациентов 030 и 023

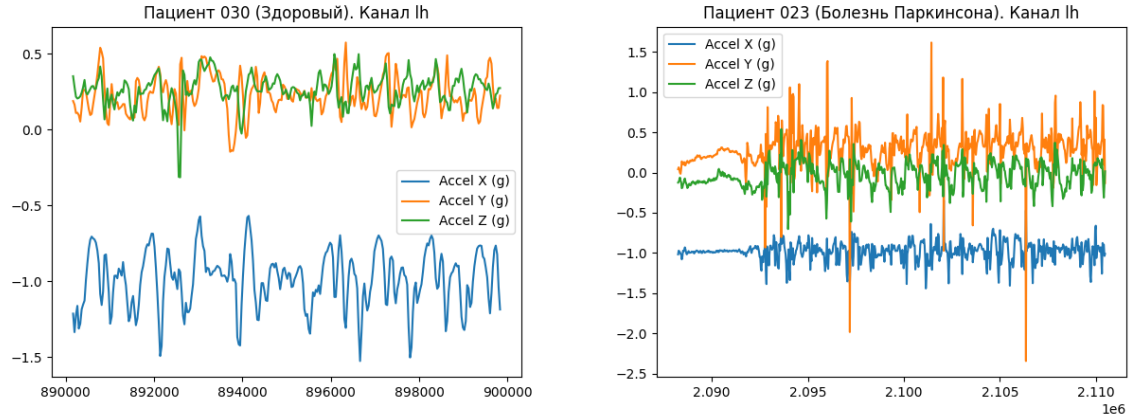


Рисунок 27 – Сигнал с левой руки для пациентов 030 и 023

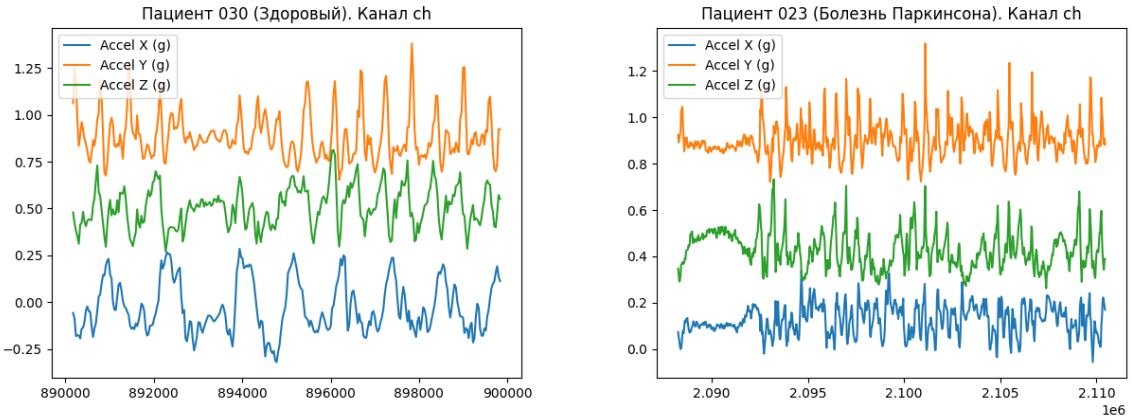


Рисунок 28 – Сигнал с грудной клетки для пациентов 030 и 023

Методы формирования вектора признаков (ВП):

1. По каждому столбцу вектора данных (ВД) рассчитывается СКО. Из полученных значений СКО формируется ВП – 15 признаков на каждого пациента.

ВП: [STD_X, STD_Y, STD_Z]*[ch, lh, ll, rh, rl]

2. В каждом столбце ВД выделяются пики выше среднего значения, таким образом формируется список индексов, соответствующих этим пикам. Далее рассчитывается разность соседних индексов, формируется список интервалов между соседними пиками сигнала, интервалы выражены в условных единицах с возможностью их перевода в секунды. Вектор признаков формируется из среднего значения первых 50 значений из списка интервалов по каждому из каналов и каждой оси. Таким образом, длина получаемого ВП – 15 значений на каждого пациента.

ВП: [MEAN_PEAKS_INTERVALS_X[:50], MEAN_PEAKS_INTERVALS_Y[:50], MEAN_PEAKS_INTERVALS_Z[:50]]*[ch, lh, ll, rh, rl]

3. В исходном ВД каждый датчик формирует три столбца, каждый из которых характеризует данные по одной из осей – X, Y, Z. Формируется новый вектор данных. Из

трёх столбцов формируется один столбец методом вычисления среднего геометрического значения. Таким образом, получается изменение радиус-вектора в трёхмерном пространстве по времени. На каждого пациента получается пять столбцов новых данных. Далее к этим столбцам применяется метод, описанный в п.2: вычисление среднего интервала между пиками сигнала. Длина ВП на каждого пациента – пять значений.

ВП: [MEAN_PEAKS_INTERVALS_XYZ]*[ch, lh, ll, rh, rl]

4. По данным нового ВД (5 столбцов данных на пациента), по каждому столбцу, берётся Фурье-преобразование. ВП формируется из СКО сигнала и СКО спектра сигнала. Длина ВП на каждого пациента – 10 значений.

ВП: [STD_XYZ, STD_SPECTRUM]*[ch, lh, ll, rh, rl]

5. ВП сформирован из СКО сигналов по каждой из осей каждого датчика, как в п.1, из СКО спектра сигнала, полученного вычислением среднего геометрического по трём осям, как в п.3, из среднего интервала между пиками сигнала, полученного вычислением среднего геометрического по трём осям. Длина ВП на каждого пациента – 25 значений.

ВП: [STD_X, STD_Y, STD_Z, STD_SPECTRUM, MEAN_PEAKS_INTERVALS_XYZ]*[ch, lh, ll, rh, rl]

6. ВП сформирован, как в п.5, добавлены средние интервалы между пиками сигнала каждой оси каждого датчика (без выделения первых 50-ти значений, в расчёте учитывались все пики выше среднего). Длина ВП на каждого пациента – 45 значений.

ВП: [STD_X, STD_Y, STD_Z, STD _XYZ, STD_SPECTRUM, MEAN_PEAKS_INTERVALS_XYZ, MEAN_PEAKS_INTERVALS_X, MEAN_PEAKS_INTERVALS_Y, MEAN_PEAKS_INTERVALS_Z]*[ch, lh, ll, rh, rl]

7. ВП признаков сформирован, как в п.6, однако средние интервалы между пиками высчитывались по первым 50-ти пикам. Длина ВП на каждого пациента – 45 значений.

ВП: [STD_X, STD_Y, STD_Z, STD _XYZ, STD_SPECTRUM, MEAN_PEAKS_INTERVALS_XYZ[:50], MEAN_PEAKS_INTERVALS_X[:50], MEAN_PEAKS_INTERVALS_Y[:50], MEAN_PEAKS_INTERVALS_Z[:50]]*[ch, lh, ll, rh, rl]

Вектор признаков разделён на тренировочную и тестовую выборки в соотношении 3:1, разделение случайное.

Применяемые алгоритмы машинного обучения:

1. Random Forest
2. Linear Discriminant Analysis
3. Logistic Regression
4. XGBClassifier

5. Support Vector Machine

В таблице 8 продемонстрированы результаты применения различных классифика торов на различных датасетах (различных ВП).

Таблица 8 – Результаты вычисления метрик при обучении разных моделей на разных ВП

Метод формирования ВП	**Классификатор**	**Accuracy**	**Balanced accuracy**	**Precision**	**Recall**	**F1**	**ROC-AUC score**
1	Random Forest	0.62	0.57	0.50	0.33	0.40	0.57
	Linear Discriminant Analysis	0.38	0.37	0.25	0.33	0.29	0.37
	Logistic Regression	0.38	0.50	0.38	1.00	0.55	0.50
	XGBClassifier	0.75	0.73	0.67	0.67	0.67	0.73
	Support Vector Machine	0.38	0.50	0.38	1.00	0.55	0.50
2	Random Forest	0.75	0.73	0.67	0.67	0.67	0.73
	Linear Discriminant Analysis	0.88	0.90	0.75	1.00	0.86	0.90
	Logistic Regression	0.50	0.53	0.40	0.67	0.50	0.53
	XGBClassifier	-	-	-	-	-	-
	Support Vector Machine	0.38	0.37	0.25	0.33	0.29	0.37
3	Random Forest	0.38	0.43	0.33	0.67	0.44	0.43
	Linear Discriminant Analysis	0.50	0.53	0.40	0.67	0.50	0.53
	Logistic Regression	0.38	0.43	0.33	0.67	0.44	0.43
	XGBClassifier	-	-	-	-	-	-
	Support Vector Machine	0.25	0.33	0.29	0.67	0.40	0.33
4	Random Forest	0.62	0.70	0.50	1.00	0.67	0.70
	Linear Discriminant Analysis	0.50	0.53	0.40	0.67	0.50	0.53
	Logistic Regression	0.38	0.50	0.38	1.00	0.55	0.50
	XGBClassifier	0.50	0.6	0.43	1.00	0.60	0.60
	Support Vector Machine	0.38	0.50	0.38	1.00	0.55	0.50
5	Random Forest	0.75	0.73	0.67	0.67	0.67	0.73
	Linear Discriminant Analysis	0.62	0.70	0.50	1.00	0.67	0.70
	Logistic Regression	0.62	0.70	0.50	1.00	0.67	0.70

	XGBClassifier	0.75	0.73	0.67	0.67	0.67	0.73
	Support Vector Machine	0.38	0.50	0.38	1.00	0.55	0.50
6	Random Forest	0.62	0.57	0.50	0.33	0.40	0.57
	Linear Discriminant Analysis	0.62	0.63	0.50	0.67	0.57	0.63
	Logistic Regression	0.75	0.80	0.60	1.00	0.75	0.80
	XGBClassifier	0.75	0.73	0.67	0.67	0.67	0.73
	Support Vector Machine	0.38	0.50	0.38	1.00	0.55	0.50
7	Random Forest	0.88	0.90	0.75	1.00	0.86	0.90
	Linear Discriminant Analysis	0.25	0.27	0.20	0.33	0.25	0.27
	Logistic Regression	0.75	0.80	0.6	1.00	0.75	0.80
	XGBClassifier	0.75	0.73	0.67	0.67	0.67	0.73
	Support Vector Machine	0.38	0.50	0.38	1.00	0.55	0.50

Таким образом, полученные результаты демонстрируют пригодность использования методов классического машинного обучения для целей классификации наличия/отсутствия болезни Паркинсона. Пригодность оценивается по метрике ROC-AUC, при её значении > 0,7 принято считать, что не только шум, но и полезный сигнал вносит вклад в результаты обучения модели.

Наилучшие значения метрик в общем по разным классификаторам дают пятый и седьмой методы выделения ВП, учитывающие СКО сигнала и средние временные интервалы.

Применение метода кросс-валидации

Метод кросс-валидации позволяет разбить исходную выборку ВП на несколько частей – фолдов. Далее каждый из фолдов становится тестовой частью выборки в каждой новой итерации подсчёта метрик, характеризующих работу модели. Таким образом, количество фолдов эквивалентно количеству итераций обучения модели. На каждой новой итерации модель обучается с нуля, рассчитываются метрики. По итогам кросс-валидации пользователь получает средние значения метрик.

Сформируем разделение на фолды методом StratifiedKFold для уравновешивания классов в тренировочной и тестовой выборках. Применим кросс-валидацию с разным количеством для седьмого метода выделения признаков и пятого, так как они показали наилучшие результаты. При применении метода кросс-валидации меняется количество

фолдов, на который разбивается исходная выборка. Результаты представлены в таблице 9, 10.

Таблица 9 – Результаты применения кросс-валидации к ВП, сформированному по пятому методу

Классификатор	**Кол-во фол-дов**	**Accuracy**	**Balanced accuracy**	**Precision**	**Recall**	**F1**	**ROC-AUC score**
RandomForestClassifier	2	0,56	0,56	0,54	0,44	0,47	0,56
LinearDiscriminantAnalysis		0,66	0,67	0,79	0,5	0,61	0,67
LogisticRegression		0,49	0,5	0,5	0,31	0,38	0,5
XGBClassifier		0,69	0,68	0,7	0,75	0,72	0,68
SVC		0,52	0,49	0,54	0,69	0,58	0,49
RandomForestClassifier	3	0,69	0,69	0,71	0,74	0,73	0,69
LinearDiscriminantAnalysis		0,55	0,54	0,57	0,68	0,61	0,54
LogisticRegression		0,59	0,57	0,6	0,62	0,57	0,57
XGBClassifier		0,56	0,56	0,59	0,58	0,58	0,56
SVC		0,55	0,5	0,55	1	0,71	0,5
RandomForestClassifier	4	0,56	0,55	0,62	0,62	0,62	0,55
LinearDiscriminantAnalysis		0,49	0,5	0,46	0,5	0,46	0,5
LogisticRegression		0,42	0,41	0,38	0,5	0,43	0,41
XGBClassifier		0,59	0,59	0,67	0,62	0,6	0,59
SVC		0,48	0,45	0,49	0,81	0,6	0,45
RandomForestClassifier	5	0,56	0,57	0,62	0,57	0,57	0,57
LinearDiscriminantAnalysis		0,37	0,36	0,36	0,42	0,38	0,36
LogisticRegression		0,46	0,43	0,43	0,53	0,47	0,43
XGBClassifier		0,55	0,53	0,56	0,67	0,6	0,53
SVC		0,52	0,47	0,53	0,93	0,68	0,47
RandomForestClassifier	6	0,62	0,64	0,67	0,64	0,59	0,64
LinearDiscriminantAnalysis		0,43	0,46	0,48	0,42	0,4	0,46
LogisticRegression		0,39	0,4	0,36	0,47	0,4	0,4
XGBClassifier		0,47	0,49	0,4	0,5	0,43	0,49
SVC		0,45	0,42	0,49	0,83	0,61	0,42

Таблица 10 – Результаты применения кросс-валидации к ВП, сформированному по седьмому методу

Классификатор	Кол-во фол-дов	Accuracy	Balanced accuracy	Precision	Recall	F1	ROC-AUC score
RandomForestClassifier	2	0,63	0,63	0,64	0,5	0,51	0,63
LinearDiscriminantAnalysis		0,59	0,6	0,66	0,5	0,56	0,6
LogisticRegression		0,52	0,52	0,56	0,44	0,48	0,52
XGBClassifier		0,62	0,62	0,65	0,69	0,67	0,62
SVC		0,49	0,45	0,5	0,69	0,56	0,45
RandomForestClassifier	3	0,73	0,73	0,74	0,76	0,75	0,73
LinearDiscriminantAnalysis		0,49	0,5	0,58	0,38	0,45	0,5
LogisticRegression		0,39	0,38	0,38	0,46	0,4	0,38
XGBClassifier		0,63	0,63	0,68	0,63	0,65	0,63
SVC		0,55	0,5	0,55	1	0,71	0,5
RandomForestClassifier	4	0,69	0,7	0,82	0,69	0,68	0,7
LinearDiscriminantAnalysis		0,49	0,5	0,46	0,38	0,41	0,5
LogisticRegression		0,48	0,47	0,45	0,62	0,52	0,47
XGBClassifier		0,62	0,64	0,79	0,62	0,62	0,64
SVC		0,48	0,45	0,49	0,81	0,6	0,45
RandomForestClassifier	5	0,73	0,72	0,76	0,73	0,74	0,72
LinearDiscriminantAnalysis		0,53	0,52	0,5	0,48	0,48	0,52
LogisticRegression		0,56	0,53	0,41	0,6	0,48	0,53
XGBClassifier		0,62	0,62	0,69	0,67	0,62	0,62
SVC		0,52	0,47	0,53	0,93	0,68	0,47
RandomForestClassifier	6	0,69	0,69	0,76	0,69	0,7	0,69
LinearDiscriminantAnalysis		0,52	0,53	0,61	0,53	0,54	0,53
LogisticRegression		0,46	0,5	0,3	0,5	0,37	0,5
XGBClassifier		0,52	0,54	0,47	0,5	0,43	0,54
SVC		0,45	0,42	0,49	0,83	0,61	0,42

Применение кросс-валидации понижает метрики, получаемые раньше.

2.2.5. DL

Попытка обучить этот датасет на модели из предыдущего датасета представлена на рисунке X.

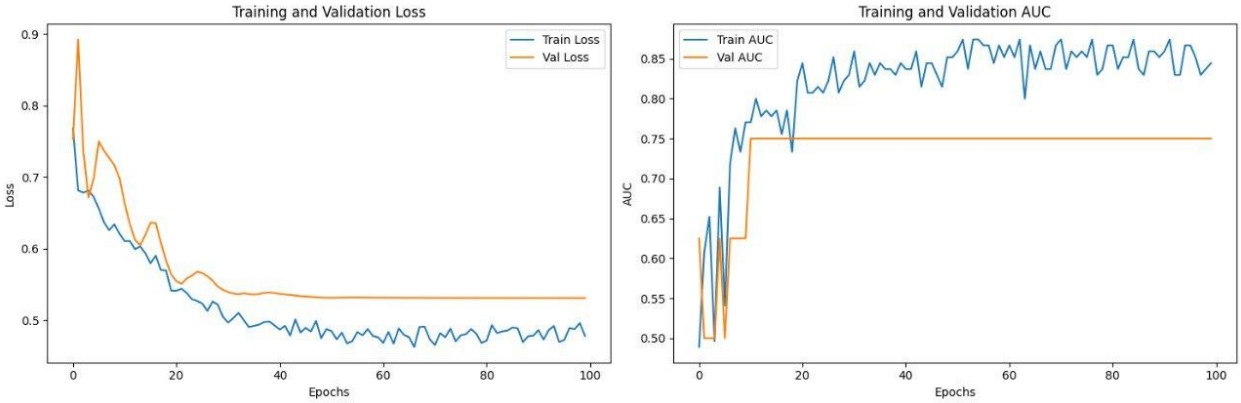


Рисунок 29 – Попытка обучить датасет 2

ЗАКЛЮЧЕНИЕ

В ходе исследования были проанализированы современные методы диагностики БП с акцентом на анализ походки, а также оценена эффективность различных алгоритмов ML и DL для классификации пациентов с БП.

Глубокое обучение показало точность (Accuracy = 0.7000) и высокую AUC-ROC (0.8009), что свидетельствует о хорошей способности моделей DL анализировать временные ряды данных походки.

Сравнение алгоритмов ML на разных вариантах предобработки данных (ВП) выявило, что наилучшие результаты достигаются при использовании XGBClassifier (Accuracy = 0.75, F1 = 0.67, ROC-AUC = 0.73) и Linear Discriminant Analysis (Accuracy = 0.88, F1 = 0.86, ROC-AUC = 0.90).

Logistic Regression и SVM в ряде случаев демонстрируют высокий Recall (1.00), но низкую точность (Precision ≈ 0.25–0.60), что указывает на склонность к ложноположительным прогнозам.

Таким образом, применение методов машинного и глубокого обучения для анализа походки позволяет эффективно классифицировать пациентов с БП. Наилучшие результаты показали XGBClassifier и LDA, а также нейросетевая модель.

СПИСОК ИСТОЧНИКОВ

1. ВОЗ. Болезнь Паркинсона [Electronic resource]. 2023.
2. Таппахов А.А., Николаева Т.Я. Современнные Представления Об Этиологии И Патогенезе Болезни Паркинсона (Обзор) // Вестник Северо-Восточного Федерального Университета Имени М.К. Аммосова. 2016. Vol. 2, N. 03. P. 19–27.
3. Lees A.J., Hardy J., Revesz T. Parkinson’s disease. // Lancet (London, England). England, 2009. Vol. 373, N. 9680. P. 2055–2066.
4. Фёдорова Н.В. БОЛЕЗНЬ ПАРКИНСОНА : ДИАГНОСТИКА И ЛЕЧЕНИЕ. 1817.
5. Chaudhuri K.R., Prieto-Jurcynska C., Naidu Y., Mitra T., Frades-Payo B., Tluk S., Ruessmann A., Odin P., Macphee G., Stocchi F., Ondo W., Sethi K., Schapira A.H. V, Castrillo J.C.M., Martinez-Martin P. The nondeclaration of nonmotor symptoms of Parkinson’s disease to health care professionals: An international study using the nonmotor symptoms questionnaire // Movement Disorders. 2010. Vol. 25, N. 6. P. 704–709.
6. Berg D., Postuma R.B., Adler C.H., Bloem B.R., Chan P., Dubois B., Gasser T., Goetz C.G., Halliday G., Joseph L., Lang A.E., Liepelt-Scarfone I., Litvan I., Marek K., Obeso J., Oertel W., Olanow C.W., Poewe W., Stern M., et al. MDS research criteria for prodromal Parkinson’s disease // Movement Disorders. 2015. Vol. 30, N. 12. P. 1600–1611.
7. Galbiati A., Verga L., Giora E., Zucconi M., Ferini-Strambi L. The risk of neurodegeneration in REM sleep behavior disorder: A systematic review and meta-analysis of longitudinal studies // Sleep Medicine Reviews. 2019. Vol. 43. P. 37–46.
8. Howell M.J., Schenck C.H. Rapid Eye Movement Sleep Behavior Disorder and Neurodegenerative Disease // JAMA Neurology. 2015. Vol. 72, N. 6. P. 707–712.
9. FEARNLEY J.M., LEES A.J. AGEING AND PARKINSON’S DISEASE: SUBSTANTIA NIGRA REGIONAL SELECTIVITY // Brain. 1991. Vol. 114, N. 5. P. 2283–2301.
10. Mc Ardle R., Morris R., Wilson J., Galna B., Thomas A.J., Rochester L. What Can Quantitative Gait Analysis Tell Us about Dementia and Its Subtypes? A Structured Review. // Journal of Alzheimer’s disease : JAD. United States, 2017. Vol. 60, N. 4. P. 1295–1312.
11. Moritz D.J., Fox P.J., Luscombe P.A., Kraemer H.C. Neurological and psychiatric predictors of mortality in patients with Alzheimer disease in California // Archives of Neurology. 1997. Vol. 54, N. 7. P. 878–885.

12. Yang M., Zheng H., Wang H., McClean S. Feature selection and construction for the discrimination of neurodegenerative diseases based on gait analysis // 2009 3rd International Conference on Pervasive Computing Technologies for Healthcare - Pervasive Health 2009, PCTHealth 2009. 2009.

13. Goldberger A.L., Amaral L.A., Glass L., Hausdorff J.M., Ivanov P.C., Mark R.G., Mietus J.E., Moody G.B., Peng C.K., Stanley H.E. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. // Circulation. United States, 2000. Vol. 101, N. 23. P. E215-20.

14. Altilio R., Paoloni M., Panella M. Selection of clinical features for pattern recognition applied to gait analysis. // Medical & biological engineering & computing. United States, 2017. Vol. 55, N. 4. P. 685–695.

15. Tien I., Glaser S.D., Aminoff M.J. Characterization of gait abnormalities in Parkinson's disease using a wireless inertial sensor system. // Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference. United States, 2010. Vol. 2010. P. 3353–3356.

16. Caramia C., Torricelli D., Schmid M., Munoz-Gonzalez A., Gonzalez-Vargas J., Grandas F., Pons J.L. IMU-Based Classification of Parkinson's Disease From Gait: A Sensitivity Analysis on Sensor Location and Feature Selection. // IEEE journal of biomedical and health informatics. United States, 2018. Vol. 22, N. 6. P. 1765–1774.

17. Rucco R., Agosti V., Jacini F., Sorrentino P., Varriale P., De Stefano M., Milan G., Montella P., Sorrentino G. Spatio-temporal and kinematic gait analysis in patients with Frontotemporal dementia and Alzheimer's disease through 3D motion capture. // Gait & posture. England, 2017. Vol. 52. P. 312–317.

18. Zeng W., Liu F., Wang Q., Wang Y., Ma L., Zhang Y. Parkinson's disease classification using gait analysis via deterministic learning. // Neuroscience letters. Ireland, 2016. Vol. 633. P. 268–278.

19. Cicirelli G., Impedovo D., Dentamaro V., Marani R., Pirlo G., D'Orazio T.R. Human Gait Analysis in Neurodegenerative Diseases: A Review // IEEE Journal of Biomedical and Health Informatics. IEEE, 2022. Vol. 26, N. 1. P. 229–242.

20. Bloem B.R., Hausdorff J.M., Visser J.E., Giladi N. Falls and freezing of gait in Parkinson's disease: a review of two interconnected, episodic phenomena. // Movement disorders :

official journal of the Movement Disorder Society. United States, 2004. Vol. 19, N. 8. P. 871–884.

21. Bonora G., Carpinella I., Cattaneo D., Chiari L., Ferrarin M. A new instrumented method for the evaluation of gait initiation and step climbing based on inertial sensors: a pilot application in Parkinson's disease. // Journal of neuroengineering and rehabilitation. England, 2015. Vol. 12. P. 45.

22. Shetty S., Rao Y. SVM based machine learning approach to identify Parkinson's disease using gait analysis. 2016. 1–5 p.

23. Weiss A., Brozgol M., Dorfman M., Herman T., Shema S., Giladi N., Hausdorff J. Does the Evaluation of Gait Quality During Daily Life Provide Insight Into Fall Risk? A Novel Approach Using 3-Day Accelerometer Recordings // Neurorehabilitation and neural repair. 2013. Vol. 27.

24. Rastegari E., Azizian S., Ali H. Machine learning and similarity network approaches to support automatic classification of Parkinson's diseases using accelerometer-based gait analysis // Proceedings of the Annual Hawaii International Conference on System Sciences. 2019. Vol. 2019-January, N. January. P. 4231–4242.

25. Computer Vision and Pattern Recognition Group. ID Gait Challenge Problem [Electronic resource]. URL: https://www.gaitchallenge.org/.

26. Alam M.N., Garg A., Munia T.T.K., Fazel-Rezai R., Tavakolian K. Vertical ground reaction force marker for Parkinson's disease // PLoS ONE. 2017. Vol. 12, N. 5. P. 1–13.

27. Ricciardi C., Amboni M., De Santis C., Improta G., Volpe G., Iuppariello L., Ricciardelli G., D'Addio G., Vitale C., Barone P., Cesarelli M. Using gait analysis' parameters to classify Parkinsonism: A data mining approach. // Computer methods and programs in biomedicine. Ireland, 2019. Vol. 180. P. 105033.

28. Bringas Tejero S., Salomón García S., Duque R., Montaña J., Lage C. A Convolutional Neural Network-Based Method for Human Movement Patterns Classification in Alzheimer's Disease // Proceedings. 2019. Vol. 31. P. 72.

29. Tunca C., Salur G., Ersoy C. Deep Learning for Fall Risk Assessment With Inertial Sensors: Utilizing Domain Knowledge in Spatio-Temporal Gait Parameters. // IEEE journal of biomedical and health informatics. United States, 2020. Vol. 24, N. 7. P. 1994–2005.

ПРИЛОЖЕНИЯ

Код для DL:

Импорт библиотек

In [1]:

```
import tensorflow as tf

gpus = tf.config.list_physical_devices('GPU')
strategy = tf.distribute.MirroredStrategy()
```

```
2025-06-13 12:11:39.173215: E external/local_xla/xla/stream_executor/cuda/cud
a_fft.cc:477] Unable to register cuFFT factory: Attempting to register factor
y for plugin cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are written
to STDERR
E0000 00:00:1749816699.354900      35 cuda_dnn.cc:8310] Unable to register cu
DNN factory: Attempting to register factory for plugin cuDNN when one has alr
eady been registered
E0000 00:00:1749816699.405414      35 cuda_blas.cc:1418] Unable to register c
uBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
I0000 00:00:1749816712.347080      35 gpu_device.cc:2022] Created device /job
:localhost/replica:0/task:0/device:GPU:0 with 15513 MB memory:  -> device: 0,
name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
```

In [2]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import uuid
import json
from IPython.display import clear_output
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.metrics import roc_auc_score, accuracy_score, precision_score,
recall_score, f1_score
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Masking, Conv1D, BatchNormaliza-
tion, Dropout, MaxPooling1D, LSTM, Dense
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.optimizers import Adam
```

Определение функций

In [3]:

```
def load_data(folder, num_steps=8192, num_features=18):
    target = pd.read_csv(f'{folder}target.csv', header=None, sep='-')
    target.columns=['filename','target']
    X = np.zeros([len(target), num_steps, num_features])
    Y = np.array(target.target)
    for i, filename in enumerate(target.filename):
        df = pd.read_csv(f'{folder}{filename}_01.csv',header=None)
        min_len = min(num_steps,len(df))
        X[i, :min_len, :] = df.iloc[:num_steps,1:]
```

```
    return X, Y
```

In [4]:

```
def build_model(max_timesteps, n_features, use_masking=False):
    inputs = Input(shape=(max_timesteps, n_features))
    x = inputs
    for filters in [32, 64, 128, 128]:
        x = Conv1D(filters, 3, activation='relu', padding='same')(x)
        x = BatchNormalization()(x)
        x = Dropout(0.25)(x)
        x = MaxPooling1D(2)(x)
    if use_masking:
        x = Masking(mask_value=0.0)(x)
    x = LSTM(64)(x)
    x = Dense(64, activation='relu')(x)
    outputs = Dense(1, activation='sigmoid')(x)
    model = Model(inputs, outputs)
    model.compile(optimizer=Adam(1e-4), loss='binary_crossentropy',met-
rics=['acc'])
    return model

def train_ensemble(X, y, n_fold=5, epochs=150, batch_size=12):
    skf = StratifiedKFold(n_splits=n_fold, shuffle=True, random_state=0)
    models, aucs, accs = [], [], []
    if not os.path.exists('models/'):
        os.mkdir('models/')
    for train_idx, val_idx in skf.split(X, y):
        K.clear_session()
        X_train, X_val = X[train_idx], X[val_idx]
        y_train, y_val = y[train_idx], y[val_idx]
        model = build_model(X.shape[1], X.shape[2], use_masking=True)
        model.fit(
            X_train, y_train,
            validation_data=(X_val, y_val),
            epochs=epochs,
            batch_size=batch_size,
            callbacks=[EarlyStopping('val_acc', mode='max', patience=75, re-
store_best_weights=True)],
            verbose=False)
        save_path = f"models/model_{uuid.uuid4().hex}.keras"
        model.save(save_path)
        y_pred = model.predict(X_val).flatten()
        models.append(save_path)
        aucs.append(roc_auc_score(y_val, y_pred))
        accs.append(accuracy_score(y_val, y_pred > 0.5))
    return np.mean(aucs), models, aucs, accs

def get_models_and_indices(removal_log, feature_groups):
    groups = feature_groups.copy()
    best_log = max(removal_log, key=lambda x: x[2])
    for log in removal_log:
        group, *_ = log
        if group:
            groups.remove(group)
        if log == best_log:
            break
    return best_log[5], sorted(np.array(groups).flatten())

def load_ensemble(save_paths):
```

```
    models = []
    for save_path in save_paths:
        models.append(load_model(save_path))
    return models

def predict(X, models):
    predictions = []
    for model in models:
        predictions.append(model.predict(X))
    return np.mean(predictions, axis=0)
```

In [6]:

```
def feature_removal(X, y, feature_groups, n_fold=5, epochs=150,
batch_size=12, min_thresh=0.9):
    orig_groups = [list(g) for g in feature_groups]
    orig_dim = X.shape[2]
    X_full = X.copy()
    active_indices = list(range(orig_dim))
    initial_score, models, aucs, accs = train_ensemble(X, y, n_fold, epochs,
batch_size)
    threshold = initial_score * min_thresh
    print(f"Initial AUC: {initial_score:.4f}, stopping threshold: {thresh-
old:.4f}")
    best_score = initial_score
    removal_log = [(None, 0.0, initial_score, aucs, accs, models,
orig_groups, len(orig_groups))]
    remaining_groups = orig_groups.copy()
    while len(remaining_groups) > 1:
        drops = []
        for group in remaining_groups:
            pos_to_remove = [active_indices.index(idx) for idx in group if
idx in active_indices]
            if not pos_to_remove:
                continue
            mask = np.ones(len(active_indices), dtype=bool)
            mask[pos_to_remove] = False
            X_reduced = X[:, :, mask]
            score, models, aucs, accs = train_ensemble(X_reduced, y, n_fold,
epochs, batch_size)
            drops.append((group, best_score - score, score, models, aucs,
accs))
        group_to_remove, drop, new_score, models, aucs, accs = min(drops,
key=lambda x: x[1])
        if new_score < threshold:
            print(f"Stop removal: removing {group_to_remove} lowers AUC below
threshold ({new_score:.4f} < {threshold:.4f})")
            break
        print(f"Removing group {group_to_remove}: drop {drop:.4f}, AUC
{new_score:.4f}, ACC {np.mean(accs):.4f}")
        for idx in group_to_remove:
            if idx in active_indices:
                active_indices.remove(idx)
        X = X_full[:, :, active_indices]
        best_score = max(best_score, new_score)
        threshold = best_score * min_thresh
        remaining_groups.remove(group_to_remove)
        removal_log.append((group_to_remove, drop, new_score, aucs, accs,
models, remaining_groups, len(remaining_groups)))
    return removal_log
```

Отбор признаков

In [7]:

```
X, y = load_data('/kaggle/input/parkinson-1/csv/csv/', num_steps=8192,
num_features=18)

X_train_full, X_test, y_train_full, y_test = train_test_split(X, y,
test_size=0.18, stratify=y, random_state=42)
```

In [8]:

```
feature_groups = [[0,8], [1,9], [2,10], [3,11], [4,12], [5,13], [6,14],
[7,15], [16,17]]

removal_log = feature_removal(X_train_full, y_train_full, feature_groups,
n_fold=5, min_thresh=0.8)

clear_output()
print("Feature removal sequence:")
for group, drop, score, *_ in removal_log:
    print(f"Group {group} removed: AUC drop {drop:.4f}, new AUC {score:.4f}")
Feature removal sequence:
Group None removed: AUC drop 0.0000, new AUC 0.6995
Group [3, 11] removed: AUC drop -0.1007, new AUC 0.8002
Group [1, 9] removed: AUC drop 0.0210, new AUC 0.7792
Group [7, 15] removed: AUC drop 0.0299, new AUC 0.7703
Group [16, 17] removed: AUC drop 0.0546, new AUC 0.7456
Group [0, 8] removed: AUC drop -0.0097, new AUC 0.8099
Group [2, 10] removed: AUC drop -0.0182, new AUC 0.8281
Group [5, 13] removed: AUC drop 0.1278, new AUC 0.7004
Group [4, 12] removed: AUC drop 0.0947, new AUC 0.7334
```

In [84]:

```
with open('removal_log.json','w') as f:
    json.dump(removal_log, f)
```

Визуализация результатов

In [129]:

```
def plot_removal_performance(removal_log):

    groups_remaining = [log[7] for log in removal_log]
    auc_values = [log[2] for log in removal_log]
    acc_values = [np.mean(log[4]) for log in removal_log]
    removed_groups = [str(log[0]) if log[0] is not None else "Start" for log
in removal_log]
    max_auc = max(auc_values)
    max_acc = max(acc_values)

    fig, ax1 = plt.subplots(figsize=(14, 10))
    color_auc = 'tab:red'
    ax1.set_xlabel('Number of feature groups remaining', fontsize=10)
    ax1.set_ylabel('ROC-AUC', color=color_auc, fontsize=10)
    auc_line = ax1.plot(groups_remaining, auc_values, 'o-', color=color_auc,
        linewidth=2, markersize=8, label='ROC-AUC')
    ax1.tick_params(axis='y', labelcolor=color_auc)
    ax1.grid(True, linestyle='--', alpha=0.7)
```

```
    ax1.axhline(y=max_auc, color=color_auc, linestyle=':', linewidth=2, al-
pha=0.7)
    ax1.annotate(f'Max ROC-AUC: {max_auc:.4f}', xy=(groups_remaining[0],
max_auc),
        xytext=(-10, -15), textcoords='offset points', color=color_auc, font-
size=10)

    ax2 = ax1.twinx()
    color_acc = 'tab:blue'
    ax2.set_ylabel('Accuracy', color=color_acc, fontsize=10)
    acc_line = ax2.plot(groups_remaining, acc_values, 's-',
        color=color_acc, linewidth=2, markersize=8, label='Accuracy')
    ax2.tick_params(axis='y', labelcolor=color_acc)

    ax2.axhline(y=max_acc, color=color_acc, linestyle=':', linewidth=2, al-
pha=0.7)
    ax2.annotate(f'Max Accuracy: {max_acc:.4f}', xy=(groups_remaining[-1],
max_acc),
        xytext=(10, -15), textcoords='offset points', color=color_acc, font-
size=10, ha='right')

    for i, (x, y_auc, y_acc, group) in enumerate(zip(groups_remaining,
auc_values, acc_values, removed_groups)):
        ax1.annotate(group, xy=(x, y_auc), xytext=(0, 15), textcoords='offset
points', ha='center', fontsize=10,
            bbox=dict(boxstyle="round,pad=0.3", fc="white", ec="grey",
lw=0.5, alpha=0.8))

    lines = auc_line + acc_line
    labels = [l.get_label() for l in lines]
    ax1.legend(lines, labels, loc='lower center', fontsize=10)
    ax1.set_xticks(groups_remaining)
    ax1.set_xticklabels(groups_remaining, fontsize=10)
    ax1.set_xlim(groups_remaining[-1] - 0.5, groups_remaining[0] + 0.5)
    ax1.invert_xaxis()
    fig.tight_layout()
    plt.show()
```

In [130]:

```
plot_removal_performance(removal_log)
```

Таким образом, датчики [4, 12], [5, 13], [6,14] способствуют построению ансамбля моделей с наилучшим качеством классификации.

Тестирование на отложенной выборке

In [202]:

```
model_paths, features = get_models_and_indices(removal_log, feature_groups)
```

In [203]:

```
models = load_ensemble(model_paths)
y_pred = predict(X_test[:,:,features],models)
print(f'Accuracy on test data: {accuracy_score(y_test, y_pred>0.5):.4f}')
print(f'Precision on test data: {precision_score(y_test, y_pred>0.5):.4f}')
print(f'Recall on test data: {recall_score(y_test, y_pred>0.5):.4f}')
print(f'F1 on test data: {f1_score(y_test, y_pred>0.5):.4f}')
print(f'ROC-AUC on test data: {roc_auc_score(y_test, y_pred):.4f}')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 367ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 369ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 369ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 362ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 376ms/step
Accuracy on test data: 0.7000
Precision on test data: 0.7500
Recall on test data: 0.7059
F1 on test data: 0.7273
ROC-AUC on test data: 0.8009
```

Пересохранение лучшего ансамбля и индексов признаков

In [204]:

```
os.mkdir('best_models/')
for i, model in enumerate(models):
    save_path = f"best_models/model_{i}.keras"
    model.save(save_path)
```

In [217]:

```
with open('features_indices.txt','w') as f:
    f.write(' '.join([str(i) for i in features]))
```

In [218]:

```
np.save('X_train.npy', X_train_full)
np.save('X_test.npy', X_test)
np.save('y_train.npy', y_train_full)
np.save('y_test.npy', y_test)
```

Код для ML:

```
from scipy.signal import find_peaks
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import glob

df = {}
col_names = []
channels = ['ch', 'lh', 'll', 'rh', 'rl']
for c in channels:
    col_names.append('STD_X_' + c)
    col_names.append('STD_Y_' + c)
    col_names.append('STD_Z_' + c)
    col_names.append('STD_XYZ_' + c)
    col_names.append('STD_SPECTRUM_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_X_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_Y_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_Z_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_XYZ_' + c)
target = [0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 1, 0, 1, 0, 0]
j = 0
```

```
for file in sorted(glob.glob('/kaggle/input/parkinson/parkinson/**/*.csv',
recursive=True)):
    if not ('Annot' in str(file) or '007' in str(file) or '014' in
str(file) or '060' in str(file) or '062' in str(file) or '063' in
str(file)):
        if target[j//5] == 0:
            s = 'Здоровый'
        else:
            s = 'Больной'
        j = j+1
        ID = file.split("ID",1)[1][:3]
        CH = file.split("ID",1)[0][-3:-1]
        data = pd.read_csv(file)
        x_data = data['Accel X (g)']
        y_data = data['Accel Y (g)']
        z_data = data['Accel Z (g)']
        xyz_data = []
        for i in range(len(x_data)):
            elem = (x_data[i]**2 + y_data[i]**2 + z_data[i]**2)**0.5
            xyz_data.append(elem)
        fs = 100
        fft_result = np.fft.fft(xyz_data)
        freq = np.fft.fftfreq(len(xyz_data), 1/fs)
        peaks_X, heights_X = find_peaks(x_data, height =
np.sum(x_data)/len(x_data))
        peaks_Y, heights_Y = find_peaks(y_data, height =
np.sum(y_data)/len(y_data))
        peaks_Z, heights_Z = find_peaks(z_data, height =
np.sum(z_data)/len(z_data))
        peaks_XYZ, heights_XYZ = find_peaks(xyz_data, height =
np.sum(xyz_data)/len(xyz_data))
        a = []
        b = []
        c = []
        d = []
        for i in range(len(peaks_X[:50])-1):
            a.append(peaks_X[i+1]-peaks_X[i])
        for i in range(len(peaks_Y[:50])-1):
            b.append(peaks_Y[i+1]-peaks_Y[i])
        for i in range(len(peaks_Z[:50])-1):
            c.append(peaks_Z[i+1]-peaks_Z[i])
        for i in range(len(peaks_XYZ[:50])-1):
            d.append(peaks_XYZ[i+1]-peaks_XYZ[i])
        STD_X = np.std((list(data['Accel X (g)'])))
        STD_Y = np.std((list(data['Accel Y (g)'])))
        STD_Z = np.std((list(data['Accel Z (g)'])))
        STD_XYZ = np.std(xyz_data)
        STD_SPECTRUM = np.std(np.abs(fft_result))
        MEAN_PEAKS_INTERVALS_X = np.sum(a)/len(a)
```

```
        MEAN_PEAKS_INTERVALS_Y = np.sum(b)/len(b)
        MEAN_PEAKS_INTERVALS_Z = np.sum(c)/len(c)
        MEAN_PEAKS_INTERVALS_XYZ = np.sum(d)/len(d)
        if df.get(ID, False) is False:
            df[ID] = [STD_X, STD_Y, STD_Z, STD_XYZ, STD_SPECTRUM,
MEAN_PEAKS_INTERVALS_X, MEAN_PEAKS_INTERVALS_Y, MEAN_PEAKS_INTERVALS_Z,
MEAN_PEAKS_INTERVALS_XYZ]
        else:
            df[ID].append(STD_X)
            df[ID].append(STD_Y)
            df[ID].append(STD_Z)
            df[ID].append(STD_XYZ)
            df[ID].append(STD_SPECTRUM)
            df[ID].append(MEAN_PEAKS_INTERVALS_X)
            df[ID].append(MEAN_PEAKS_INTERVALS_Y)
            df[ID].append(MEAN_PEAKS_INTERVALS_Z)
            df[ID].append(MEAN_PEAKS_INTERVALS_XYZ)
df = pd.DataFrame(df).transpose()
df.columns = col_names
df.to_csv('/kaggle/working/dataset2.csv')
df_target = pd.DataFrame(target, columns=['Class'])
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df, df_target,
test_size=0.25, random_state=0)

from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, balanced_accuracy_score, roc_curve, auc, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt

model = RandomForestClassifier(n_estimators=100, random_state=42)
cross_val_score(model, X_train, y_train, cv=5)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df, target,
test_size=0.25, random_state=0)
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, balanced_accuracy_score, roc_curve, auc, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt

def metrics(y_test, y_pred, model):
    print("CV1: %0.2f accuracy with a standard deviation of %0.2f" %
(cross_val_score(model, X_train, y_train, cv=5).mean(),
cross_val_score(model, X_train, y_train, cv=5).std()))
```

```
    print("CV2: %0.2f accuracy with a standard deviation of %0.2f" %
(cross_val_score(model, X_test, y_test, cv=5).mean(),
cross_val_score(model, X_test, y_test, cv=5).std()))
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print(f"Balanced accuracy: {balanced_accuracy_score(y_test,
y_pred):.2f}")
    print(f"Precision: {precision_score(y_test, y_pred):.2f}")
    print(f"Recall: {recall_score(y_test, y_pred):.2f}")
    print(f"F1: {f1_score(y_test, y_pred):.2f}")
    print(f"ROC AUC score: {roc_auc_score(y_test, y_pred):.2f}")
    fpr, tpr, thresholds = roc_curve(y_test, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(10, 8))
    plt.plot(fpr, tpr, color='darkorange', lw=2,
    label=f'ROC curve (AUC = {roc_auc:.3f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0, 1.0])
    plt.ylim([0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('====================\nRandom Forest Met-
rics\n====================')
metrics(y_test, y_pred, model)

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

model = LinearDiscriminantAnalysis()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('============================\nLinear Discriminant Analy-
sis\n============================')
metrics(y_test, y_pred, model)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

```
print('==================\nLogistic Regression\n==================')
metrics(y_test, y_pred, model)
from xgboost import XGBClassifier

model = XGBClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('=============\nXGBClassifier\n=============')
metrics(y_test, y_pred, model)

from sklearn.svm import SVC

model = SVC()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('======================\nSupport Vector Ma-
chine\n======================')
metrics(y_test, y_pred, model)
from scipy.signal import find_peaks
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import glob

df = {}
col_names = []
channels = ['ch', 'lh', 'll', 'rh', 'rl']
for c in channels:
    col_names.append('MEAN_PEAKS_INTERVALS_X_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_Y_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_Z_' + c)
target = [0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 1, 0, 1, 0, 0]
j = 0

for file in sorted(glob.glob('/kaggle/input/parkinson/**/*.csv', recur-
sive=True)):
    if not ('Annot' in str(file) or '007' in str(file) or '014' in
str(file) or '060' in str(file) or '062' in str(file) or '063' in
str(file)):
        if target[j//5] == 0:
            s = 'Здоровый'
        else:
            s = 'Больной'
        j = j+1
```

```
        ID = file.split("ID",1)[1][:3]
        CH = file.split("ID",1)[0][-3:-1]
        data = pd.read_csv(file)
        x_data = data['Accel X (g)']
        y_data = data['Accel Y (g)']
        z_data = data['Accel Z (g)']
        xyz_data = []
        for i in range(len(x_data)):
            elem = (x_data[i]**2 + y_data[i]**2 + z_data[i]**2)**0.5
            xyz_data.append(elem)
        fs = 100
        fft_result = np.fft.fft(xyz_data)
        freq = np.fft.fftfreq(len(xyz_data), 1/fs)
        peaks_X, heights_X = find_peaks(x_data, height =
np.sum(x_data)/len(x_data))
        peaks_Y, heights_Y = find_peaks(y_data, height =
np.sum(y_data)/len(y_data))
        peaks_Z, heights_Z = find_peaks(z_data, height =
np.sum(z_data)/len(z_data))
        peaks_XYZ, heights_XYZ = find_peaks(xyz_data, height =
np.sum(xyz_data)/len(xyz_data))
        a = []
        b = []
        c = []
        d = []
        for i in range(len(peaks_X[:50])-1):
            a.append(peaks_X[i+1]-peaks_X[i])
        for i in range(len(peaks_Y[:50])-1):
            b.append(peaks_Y[i+1]-peaks_Y[i])
        for i in range(len(peaks_Z[:50])-1):
            c.append(peaks_Z[i+1]-peaks_Z[i])
        for i in range(len(peaks_XYZ[:50])-1):
            d.append(peaks_XYZ[i+1]-peaks_XYZ[i])
        STD_X = np.std((list(data['Accel X (g)'])))
        STD_Y = np.std((list(data['Accel Y (g)'])))
        STD_Z = np.std((list(data['Accel Z (g)'])))
        STD_XYZ = np.std(xyz_data)
        STD_SPECTRUM = np.std(np.abs(fft_result))
        MEAN_PEAKS_INTERVALS_X = np.sum(a)/len(a)
        MEAN_PEAKS_INTERVALS_Y = np.sum(b)/len(b)
        MEAN_PEAKS_INTERVALS_Z = np.sum(c)/len(c)
        MEAN_PEAKS_INTERVALS_XYZ = np.sum(d)/len(d)
        if df.get(ID, False) is False:
            df[ID] = [MEAN_PEAKS_INTERVALS_X, MEAN_PEAKS_INTERVALS_Y,
MEAN_PEAKS_INTERVALS_Z]
        else:
            df[ID].append(MEAN_PEAKS_INTERVALS_X)
            df[ID].append(MEAN_PEAKS_INTERVALS_Y)
            df[ID].append(MEAN_PEAKS_INTERVALS_Z)
df = pd.DataFrame(df).transpose()
```

```
df.columns = col_names
df.to_csv('/kaggle/working/dataset2.csv')
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df, target,
test_size=0.25, random_state=0)
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, balanced_accuracy_score, roc_curve, auc, roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt

def metrics(y_test, y_pred, model):
    print("CV1: %0.2f accuracy with a standard deviation of %0.2f" %
(cross_val_score(model, X_train, y_train, cv=5).mean(),
cross_val_score(model, X_train, y_train, cv=5).std()))
    print("CV2: %0.2f accuracy with a standard deviation of %0.2f" %
(cross_val_score(model, X_test, y_test, cv=5).mean(),
cross_val_score(model, X_test, y_test, cv=5).std()))
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
    print(f"Balanced accuracy: {balanced_accuracy_score(y_test,
y_pred):.2f}")
    print(f"Precision: {precision_score(y_test, y_pred):.2f}")
    print(f"Recall: {recall_score(y_test, y_pred):.2f}")
    print(f"F1: {f1_score(y_test, y_pred):.2f}")
    print(f"ROC AUC score: {roc_auc_score(y_test, y_pred):.2f}")
    fpr, tpr, thresholds = roc_curve(y_test, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(10, 8))
    plt.plot(fpr, tpr, color='darkorange', lw=2,
    label=f'ROC curve (AUC = {roc_auc:.3f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0, 1.0])
    plt.ylim([0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('====================\nRandom Forest Met-
rics\n====================')
metrics(y_test, y_pred, model)
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
model = LinearDiscriminantAnalysis()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('============================\nLinear Discriminant Analy-
sis\n============================')
metrics(y_test, y_pred, model)
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('===================\nLogistic Regression\n===================')
metrics(y_test, y_pred, model)

from xgboost import XGBClassifier

model = XGBClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('=============\nXGBClassifier\n=============')
metrics(y_test, y_pred, model)


from sklearn.svm import SVC

model = SVC()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('======================\nSupport Vector Ma-
chine\n======================')
metrics(y_test, y_pred, model)
```

```
from scipy.signal import find_peaks
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import glob

df = {}
col_names = []
channels = ['ch', 'lh', 'll', 'rh', 'rl']
```

```
for c in channels:
    col_names.append('STD_X_' + c)
    col_names.append('STD_Y_' + c)
    col_names.append('STD_Z_' + c)
    col_names.append('STD_XYZ_' + c)
    col_names.append('STD_SPECTRUM_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_X_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_Y_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_Z_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_XYZ_' + c)
target = [0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 1, 0, 1, 0, 0]
j = 0

for file in sorted(glob.glob('/kaggle/input/parkinson/parkinson/**/*.csv',
recursive=True)):
    if not ('Annot' in str(file) or '007' in str(file) or '014' in
str(file) or '060' in str(file) or '062' in str(file) or '063' in
str(file)):
        if target[j//5] == 0:
            s = 'Здоровый'
        else:
            s = 'Больной'
        j = j+1
        ID = file.split("ID",1)[1][:3]
        CH = file.split("ID",1)[0][-3:-1]
        data = pd.read_csv(file)
        x_data = data['Accel X (g)']
        y_data = data['Accel Y (g)']
        z_data = data['Accel Z (g)']
        xyz_data = []
        for i in range(len(x_data)):
            elem = (x_data[i]**2 + y_data[i]**2 + z_data[i]**2)**0.5
            xyz_data.append(elem)
        fs = 100
        fft_result = np.fft.fft(xyz_data)
        freq = np.fft.fftfreq(len(xyz_data), 1/fs)
        peaks_X, heights_X = find_peaks(x_data, height =
np.sum(x_data)/len(x_data))
        peaks_Y, heights_Y = find_peaks(y_data, height =
np.sum(y_data)/len(y_data))
        peaks_Z, heights_Z = find_peaks(z_data, height =
np.sum(z_data)/len(z_data))
        peaks_XYZ, heights_XYZ = find_peaks(xyz_data, height =
np.sum(xyz_data)/len(xyz_data))
        a = []
        b = []
        c = []
        d = []
        for i in range(len(peaks_X[:50])-1):
```

```
            a.append(peaks_X[i+1]-peaks_X[i])
        for i in range(len(peaks_Y[:50])-1):
            b.append(peaks_Y[i+1]-peaks_Y[i])
        for i in range(len(peaks_Z[:50])-1):
            c.append(peaks_Z[i+1]-peaks_Z[i])
        for i in range(len(peaks_XYZ[:50])-1):
            d.append(peaks_XYZ[i+1]-peaks_XYZ[i])
        STD_X = np.std((list(data['Accel X (g)'])))
        STD_Y = np.std((list(data['Accel Y (g)'])))
        STD_Z = np.std((list(data['Accel Z (g)'])))
        STD_XYZ = np.std(xyz_data)
        STD_SPECTRUM = np.std(np.abs(fft_result))
        MEAN_PEAKS_INTERVALS_X = np.sum(a)/len(a)
        MEAN_PEAKS_INTERVALS_Y = np.sum(b)/len(b)
        MEAN_PEAKS_INTERVALS_Z = np.sum(c)/len(c)
        MEAN_PEAKS_INTERVALS_XYZ = np.sum(d)/len(d)
        if df.get(ID, False) is False:
            df[ID] = [STD_X, STD_Y, STD_Z, STD_XYZ, STD_SPECTRUM,
MEAN_PEAKS_INTERVALS_X, MEAN_PEAKS_INTERVALS_Y, MEAN_PEAKS_INTERVALS_Z,
MEAN_PEAKS_INTERVALS_XYZ]
        else:
            df[ID].append(STD_X)
            df[ID].append(STD_Y)
            df[ID].append(STD_Z)
            df[ID].append(STD_XYZ)
            df[ID].append(STD_SPECTRUM)
            df[ID].append(MEAN_PEAKS_INTERVALS_X)
            df[ID].append(MEAN_PEAKS_INTERVALS_Y)
            df[ID].append(MEAN_PEAKS_INTERVALS_Z)
            df[ID].append(MEAN_PEAKS_INTERVALS_XYZ)
df = pd.DataFrame(df).transpose()
df.columns = col_names
df.to_csv('/kaggle/working/dataset2.csv')

df_target = pd.DataFrame(target, columns=['Class'])
from sklearn.model_selection import StratifiedKFold
df_metrics = pd.DataFrame()
def metriks_cross(model, n, df, df_target, name_model, df_metrics):
    skf = StratifiedKFold(n_splits=n)
    i = 1
    Accuracy = []
    B_accuracy = []
    Precision = []
    Recall = []
    F1 = []
    ROC_AUC = []
    for train_index, test_index in skf.split(df, df_target):
        # print('Итерация %d' %i)
        model.fit(df.iloc[train_index], df_target.iloc[train_index])
        y_pred = model.predict(df.iloc[test_index])
```

```
        Accuracy.append(accuracy_score(df_target.iloc[test_index],
y_pred))
        B_accuracy.append(balanced_accuracy_score(df_target.iloc[test_in-
dex], y_pred))
        Precision.append(precision_score(df_target.iloc[test_index],
y_pred))
        Recall.append(recall_score(df_target.iloc[test_index], y_pred))
        F1.append(f1_score(df_target.iloc[test_index], y_pred))
        ROC_AUC.append(roc_auc_score(df_target.iloc[test_index], y_pred))

        # print('\nМетрики по данному разбиению фолдов')
        # print(f"Accuracy: {accuracy_score(df_target.iloc[test_index],
y_pred):.2f}")
        # print(f"Balanced accuracy: {balanced_accuracy_score(df_tar-
get.iloc[test_index], y_pred):.2f}")
        # print(f"Precision: {precision_score(df_target.iloc[test_index],
y_pred):.2f}")
        # print(f"Recall: {recall_score(df_target.iloc[test_index],
y_pred):.2f}")
        # print(f"F1: {f1_score(df_target.iloc[test_index], y_pred):.2f}")
        # print(f"ROC AUC score: {roc_auc_score(df_target.iloc[test_in-
dex], y_pred):.2f}")
        # print('\n\n')
        i = i+1
    # print('\nСредние значения метрик:')
    # print(f"Accuracy: {np.sum(Accuracy)/len(Accuracy):.2f}")
    # print(f"Balanced accuracy: {np.sum(B_accuracy)/len(B_accu-
racy):.2f}")
    # print(f"Precision: {np.sum(Precision)/len(Precision):.2f}")
    # print(f"Recall: {np.sum(Recall)/len(Recall):.2f}")
    # print(f"F1: {np.sum(F1)/len(F1):.2f}")
    # print(f"ROC AUC score: {np.sum(ROC_AUC)/len(ROC_AUC):.2f}")

    # df_metrics.at[name_model, 'n_folds'] = n
    # df_metrics.at[name_model, 'Accuracy'] = round(np.sum(Accu-
racy)/len(Accuracy), 2)
    # df_metrics.at[name_model, 'B_accuracy'] = round(np.sum(B_accu-
racy)/len(B_accuracy), 2)
    # df_metrics.at[name_model, 'Precision'] = round(np.sum(Preci-
sion)/len(Precision), 2)
    # df_metrics.at[name_model, 'Recall'] = round(np.sum(Recall)/len(Re-
call),2)
    # df_metrics.at[name_model, 'F1'] = round(np.sum(F1)/len(F1), 2)
    # df_metrics.at[name_model, 'ROC_AUC'] =
round(np.sum(ROC_AUC)/len(ROC_AUC), 2)
```

```
    new_row = pd.DataFrame({"n_folds": n, "Accuracy": round(np.sum(Accu-
racy)/len(Accuracy), 2), 'B_accuracy': round(np.sum(B_accuracy)/len(B_ac-
curacy), 2), 'Precision': round(np.sum(Precision)/len(Precision), 2), 'Re-
call': round(np.sum(Recall)/len(Recall),2), 'F1':round(np.sum(F1)/len(F1),
2), 'ROC_AUC':round(np.sum(ROC_AUC)/len(ROC_AUC), 2)}, index =
[name_model])
    # print(new_row)
    df_metrics = pd.concat([df_metrics, new_row])
    # print(df_metrics)
    return df_metrics
    print('\n\n')
for n in range(2, 7):
    from sklearn.ensemble import RandomForestClassifier
    rf_classifier = RandomForestClassifier(n_estimators=100, ran-
dom_state=42)
    # print('====================\nRandom Forest Met-
rics\n====================')
    df_metrics = metriks_cross(rf_classifier, n, df, df_target, 'Random-
ForestClassifier', df_metrics)




    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    lda = LinearDiscriminantAnalysis()
    # print('============================\nLinear Discriminant Analy-
sis\n============================')
    df_metrics = metriks_cross(lda, n, df, df_target, 'LinearDiscriminan-
tAnalysis', df_metrics)



    from sklearn.linear_model import LogisticRegression
    logr = LogisticRegression()
    # print('==================\nLogistic Regres-
sion\n==================')
    df_metrics = metriks_cross(logr, n, df, df_target, 'LogisticRegres-
sion', df_metrics)



    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    # print('=============\nXGBClassifier\n=============')
    df_metrics = metriks_cross(XGBC, n, df, df_target, 'XGBClassifier',
df_metrics)



    from sklearn.svm import SVC
```

```
    svc = SVC()
    # print('======================\nSupport Vector Ma-
chine\n======================')
    df_metrics = metriks_cross(svc, n, df, df_target, 'SVC', df_metrics)

# print(df_metrics)
df_metrics.to_excel('Кросс-валидация_7.xlsx')
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
scaled_df = scaler.transform(df)
print(df.head())
# print(scaled_df)
scaled_df = pd.DataFrame(scaled_df, columns=df.columns, index = df.index)
# print(scaled_df)
from sklearn.decomposition import PCA
pca = PCA(n_components = 6)
df_pca = pca.fit_transform(df)
# print(pca.feature_names_in_)
# print(pca.components_)
# print('\n')
# print(pca.get_feature_names_out(input_features=pca.feature_names_in_))
# print(df_pca.shape)
# print(df_pca)
print(pca.explained_variance_ratio_)

component_weights = pca.components_
# print(type(df_pca))
df_pca = pd.DataFrame(df_pca)
# print(type(df_pca))
feature_weights_mapping = {}
for i, component in enumerate(component_weights):
  component_feature_weights = zip(pca.feature_names_in_, component)
  sorted_feature_weight = sorted(
      component_feature_weights, key=lambda x: abs(x[1]), reverse=True)
  feature_weights_mapping[f"Component {i+1}"] = sorted_feature_weight

# Accessing feature names contributing to Principal Component
print("Feature names contributing to Principal Components")
for feature, weight in feature_weights_mapping.items():
  print(f"{feature}: {weight}")
df_metrics = pd.DataFrame()
for n in range(2, 7):
    from sklearn.ensemble import RandomForestClassifier
    rf_classifier = RandomForestClassifier(n_estimators=100, ran-
dom_state=42)
    # print('====================\nRandom Forest Met-
rics\n====================')
    df_metrics = metriks_cross(rf_classifier, n, df_pca, df_target, 'Ran-
domForestClassifier', df_metrics)
```

```
    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    lda = LinearDiscriminantAnalysis()
    # print('============================\nLinear Discriminant Analy-
sis\n============================')
    df_metrics = metriks_cross(lda, n, df_pca, df_target, 'LinearDiscrimi-
nantAnalysis', df_metrics)



    from sklearn.linear_model import LogisticRegression
    logr = LogisticRegression()
    # print('====================\nLogistic Regres-
sion\n====================')
    df_metrics = metriks_cross(logr, n, df_pca, df_target,
'LogisticRegression', df_metrics)



    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    # print('=============\nXGBClassifier\n=============')
    df_metrics = metriks_cross(XGBC, n, df_pca, df_target, 'XGBClassifi-
er', df_metrics)



    from sklearn.svm import SVC
    svc = SVC()
    # print('======================\nSupport Vector Ma-
chine\n======================')
    df_metrics = metriks_cross(svc, n, df_pca, df_target, 'SVC', df_met-
rics)


df_metrics.to_excel('Кросс-валидация_PCA_6.xlsx')
pca = PCA(n_components = 15)
df_pca = pca.fit_transform(df)
# print(pca.feature_names_in_)
# print(pca.components_)
# print('\n')
# print(pca.get_feature_names_out(input_features=pca.feature_names_in_))
# print(df_pca.shape)
# print(df_pca)
print(pca.explained_variance_ratio_)

component_weights = pca.components_
```

```
# print(type(df_pca))
df_pca = pd.DataFrame(df_pca)
# print(type(df_pca))
df_metrics = pd.DataFrame()
for n in range(2, 7):
    from sklearn.ensemble import RandomForestClassifier
    rf_classifier = RandomForestClassifier(n_estimators=100, ran-
dom_state=42)
    # print('====================\nRandom Forest Met-
rics\n====================')
    df_metrics = metriks_cross(rf_classifier, n, df_pca, df_target, 'Ran-
domForestClassifier', df_metrics)




    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    lda = LinearDiscriminantAnalysis()
    # print('============================\nLinear Discriminant Analy-
sis\n============================')
    df_metrics = metriks_cross(lda, n, df_pca, df_target, 'LinearDiscrimi-
nantAnalysis', df_metrics)



    from sklearn.linear_model import LogisticRegression
    logr = LogisticRegression()
    # print('==================\nLogistic Regres-
sion\n==================')
    df_metrics = metriks_cross(logr, n, df_pca, df_target,
'LogisticRegression', df_metrics)



    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    # print('=============\nXGBClassifier\n=============')
    df_metrics = metriks_cross(XGBC, n, df_pca, df_target, 'XGBClassifi-
er', df_metrics)



    from sklearn.svm import SVC
    svc = SVC()
    # print('======================\nSupport Vector Ma-
chine\n======================')
    df_metrics = metriks_cross(svc, n, df_pca, df_target, 'SVC', df_met-
rics)
```

```
df_metrics.to_excel('Кросс-валидация_PCA_15.xlsx')
pca = PCA(n_components = 25)
df_pca = pca.fit_transform(df)
# print(pca.feature_names_in_)
# print(pca.components_)
# print('\n')
# print(pca.get_feature_names_out(input_features=pca.feature_names_in_))
# print(df_pca.shape)
# print(df_pca)
print(pca.explained_variance_ratio_)

component_weights = pca.components_
# print(type(df_pca))
df_pca = pd.DataFrame(df_pca)
# print(type(df_pca))
df_metrics = pd.DataFrame()
for n in range(2, 7):
    from sklearn.ensemble import RandomForestClassifier
    rf_classifier = RandomForestClassifier(n_estimators=100, ran-
dom_state=42)
    # print('====================\nRandom Forest Met-
rics\n====================')
    df_metrics = metriks_cross(rf_classifier, n, df_pca, df_target, 'Ran-
domForestClassifier', df_metrics)




    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    lda = LinearDiscriminantAnalysis()
    # print('============================\nLinear Discriminant Analy-
sis\n============================')
    df_metrics = metriks_cross(lda, n, df_pca, df_target, 'LinearDiscrimi-
nantAnalysis', df_metrics)



    from sklearn.linear_model import LogisticRegression
    logr = LogisticRegression()
    # print('==================\nLogistic Regres-
sion\n==================')
    df_metrics = metriks_cross(logr, n, df_pca, df_target,
'LogisticRegression', df_metrics)



    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    # print('============\nXGBClassifier\n============')
```

```
    df_metrics = metriks_cross(XGBC, n, df_pca, df_target, 'XGBClassifi-
er', df_metrics)



    from sklearn.svm import SVC
    svc = SVC()
    # print('======================\nSupport Vector Ma-
chine\n======================')
    df_metrics = metriks_cross(svc, n, df_pca, df_target, 'SVC', df_met-
rics)


df_metrics.to_excel('Кросс-валидация_PCA_25.xlsx')
pca = PCA(n_components = 29)
df_pca = pca.fit_transform(df)
# print(pca.feature_names_in_)
# print(pca.components_)
# print('\n')
# print(pca.get_feature_names_out(input_features=pca.feature_names_in_))
# print(df_pca.shape)
# print(df_pca)
print(pca.explained_variance_ratio_)

component_weights = pca.components_
# print(type(df_pca))
df_pca = pd.DataFrame(df_pca)
# print(type(df_pca))
df_metrics = pd.DataFrame()
for n in range(2, 7):
    from sklearn.ensemble import RandomForestClassifier
    rf_classifier = RandomForestClassifier(n_estimators=100, ran-
dom_state=42)
    # print('======================\nRandom Forest Met-
rics\n======================')
    df_metrics = metriks_cross(rf_classifier, n, df_pca, df_target, 'Ran-
domForestClassifier', df_metrics)



    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    lda = LinearDiscriminantAnalysis()
    # print('============================\nLinear Discriminant Analy-
sis\n============================')
    df_metrics = metriks_cross(lda, n, df_pca, df_target, 'LinearDiscrimi-
nantAnalysis', df_metrics)
```

```
    from sklearn.linear_model import LogisticRegression
    logr = LogisticRegression()
    # print('==================\nLogistic Regres-
sion\n==================')
    df_metrics = metriks_cross(logr, n, df_pca, df_target,
'LogisticRegression', df_metrics)



    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    # print('=============\nXGBClassifier\n=============')
    df_metrics = metriks_cross(XGBC, n, df_pca, df_target, 'XGBClassifi-
er', df_metrics)



    from sklearn.svm import SVC
    svc = SVC()
    # print('======================\nSupport Vector Ma-
chine\n======================')
    df_metrics = metriks_cross(svc, n, df_pca, df_target, 'SVC', df_met-
rics)


df_metrics.to_excel('Кросс-валидация_PCA_29.xlsx')
df = {}
col_names = []
channels = ['ch', 'lh', 'll', 'rh', 'rl']
for c in channels:
    col_names.append('STD_X_' + c)
    col_names.append('STD_Y_' + c)
    col_names.append('STD_Z_' + c)
    #col_names.append('STD_XYZ_' + c)
    col_names.append('STD_SPECTRUM_' + c)
    col_names.append('MEAN_PEAKS_INTERVALS_XYZ_' + c)
target = [0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
0, 1, 0, 1, 0, 1, 0, 0]
j = 0

for file in sorted(glob.glob('/kaggle/input/parkinson/parkinson/**/*.csv',
recursive=True)):
    if not ('Annot' in str(file) or '007' in str(file) or '014' in
str(file) or '060' in str(file) or '062' in str(file) or '063' in
str(file)):
        if target[j//5] == 0:
            s = 'Здоровый'
        else:
            s = 'Больной'
        j = j+1
```

```
        ID = file.split("ID",1)[1][:3]
        CH = file.split("ID",1)[0][-3:-1]
        data = pd.read_csv(file)
        x_data = data['Accel X (g)']
        y_data = data['Accel Y (g)']
        z_data = data['Accel Z (g)']
        xyz_data = []
        for i in range(len(x_data)):
            elem = (x_data[i]**2 + y_data[i]**2 + z_data[i]**2)**0.5
            xyz_data.append(elem)
        fs = 100
        fft_result = np.fft.fft(xyz_data)
        freq = np.fft.fftfreq(len(xyz_data), 1/fs)
        peaks_XYZ, heights_XYZ = find_peaks(xyz_data, height =
np.sum(xyz_data)/len(xyz_data))
        b = []
        for i in range(len(peaks_XYZ)-1):
            b.append(peaks_XYZ[i+1]-peaks_XYZ[i])
        STD_X = np.std((list(data['Accel X (g)'])))
        STD_Y = np.std((list(data['Accel Y (g)'])))
        STD_Z = np.std((list(data['Accel Z (g)'])))
        STD_XYZ = np.std(xyz_data)
        STD_SPECTRUM = np.std(np.abs(fft_result))
        MEAN_PEAKS_INTERVALS_XYZ = np.sum(b)/len(b)
        if df.get(ID, False) is False:
            df[ID] = [STD_X, STD_Y, STD_Z, STD_SPECTRUM, MEAN_PEAKS_INTER-
VALS_XYZ]
        else:
            df[ID].append(STD_X)
            df[ID].append(STD_Y)
            df[ID].append(STD_Z)
            #df[ID].append(STD_XYZ)
            df[ID].append(STD_SPECTRUM)
            df[ID].append(MEAN_PEAKS_INTERVALS_XYZ)
df = pd.DataFrame(df).transpose()
df.columns = col_names
# df.to_csv('/kaggle/working/dataset2.csv')
df_target = pd.DataFrame(target, columns=['Class'])
df_metrics = pd.DataFrame()
for n in range(2, 7):
    from sklearn.ensemble import RandomForestClassifier
    rf_classifier = RandomForestClassifier(n_estimators=100, ran-
dom_state=42)
    # print('====================\nRandom Forest Met-
rics\n====================')
    df_metrics = metriks_cross(rf_classifier, n, df, df_target, 'Random-
ForestClassifier', df_metrics)
```

```
    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    lda = LinearDiscriminantAnalysis()
    # print('============================\nLinear Discriminant Analy-
sis\n============================')
    df_metrics = metriks_cross(lda, n, df, df_target, 'LinearDiscriminan-
tAnalysis', df_metrics)



    from sklearn.linear_model import LogisticRegression
    logr = LogisticRegression()
    # print('==================\nLogistic Regres-
sion\n==================')
    df_metrics = metriks_cross(logr, n, df, df_target, 'LogisticRegres-
sion', df_metrics)



    from xgboost import XGBClassifier
    XGBC = XGBClassifier()
    # print('=============\nXGBClassifier\n=============')
    df_metrics = metriks_cross(XGBC, n, df, df_target, 'XGBClassifier',
df_metrics)



    from sklearn.svm import SVC
    svc = SVC()
    # print('======================\nSupport Vector Ma-
chine\n======================')
    df_metrics = metriks_cross(svc, n, df, df_target, 'SVC', df_metrics)

# print(df_metrics)
df_metrics.to_excel('Кросс-валидация_5.xlsx')
```