# **Описание алгоритмов**

## **Методы обработки и анализа физиологических процессов**

### **Калмановская фильтрация с моделью постоянной скорости**

Функция kalman\_cv() реализует дискретный Калмановский фильтр для одномерного временного ряда (например, сигнал ЧСС).

Особенности:

* Используется модель постоянной скорости: состояние описывается положением (значение сигнала) и скоростью изменения.
* Матрицы перехода F, наблюдения H, ковариации шума Q, R формируются стандартно.
* Алгоритм последовательно оценивает истинное значение сигнала, сглаживая шум и артефакты.

### **Фильтрация сигнала частоты сердечных сокращений плода**

Функция filter\_fhr\_df() предназначена для обработки данных ЧСС.

Этапы:

* Ограничение диапазона: остаются значения в физиологических пределах 70–200 уд/мин.
* Интерполяция: данные приводятся к равномерной сетке по времени (1 сек).
* Медианная фильтрация (medfilt): удаляются резкие выбросы и шум.
* Анализ градиента: вычисляется производная, чтобы выявить «скачки» > 10 уд/мин/с.
  + Такие точки заменяются линейной интерполяцией между «хорошими» соседями.
* Калмановский фильтр: окончательное сглаживание и оценка истинного сигнала.

**Фильтрация сигнала маточного тонуса**

Функция filter\_uc\_df() обрабатывает данные uterine contractions (UC).  
Этапы:

* Ограничение диапазона: значения 0–300 единиц.
* Интерполяция: выравнивание по временной оси (1 сек).
* Медианная фильтрация (окно 31): подавление одиночных артефактов.
* Цифровая фильтрация Баттерворта (low-pass): частота среза 0.05 Гц (~периоды > 20 секунд), что позволяет отфильтровать быстрые колебания и оставить только медленные изменения тонуса.
* Если фильтр неустойчив (ошибка), оставляется медианный результат.

## **Алгоритмы выявления аномалий и построения прогнозов**

**Что делает система**

Модуль ИИ обрабатывает сигналы КТГ (ЧСС плода и тонус матки) и решает две задачи:

* **Выявление аномалий** — фиксирует подозрительные участки записи (например, тахикардия, брадикардия, слабая вариабельность, патологические деселерации).
* **Построение прогноза** — оценивает вероятность неблагополучного состояния (например, риск гипоксии) и относит запись к категории: *норма / сомнительно / патологично*.

**Как это работает**

* **Извлечение признаков**

Из сигнала считаются простые показатели: средняя ЧСС, разброс значений, наличие ускорений/замедлений, частота и сила схваток.

* **Пороговые проверки**
  + Если значения выходят за границы (например, ЧСС > 160 или < 110), система отмечает это как аномалию.
* **Нейросетевая модель**
  + Обученная нейросеть анализирует весь набор признаков сразу и выдаёт вероятности:
    1. всё в норме,
    2. ситуация сомнительная,
    3. высокая вероятность патологии.
* **Обновление в реальном времени**
  + Каждые несколько секунд сигнал пересчитывается и обновляется прогноз. В интерфейсе отображаются вероятности («Hypoxia 60%», «Emergency 30%» и т.п.).

**Результат для врача**

* На экране видно: графики ЧСС и тонуса + блок вероятностей.
* Система показывает:
  + **Норму** → продолжаем обычное наблюдение,
  + **Сомнительное состояние** → нужно внимательнее наблюдать,
  + **Патологию** → требуется срочная оценка врача.
* В отчёте сохраняются все события и итоговый прогноз.

## **Обоснование выбора ML-модели и их характеристики**

Для анализа кардиотокографических сигналов мы остановились на многослойном перцептроне. Он не такой тяжёлый, как рекуррентные или сверточные сети, но при этом хорошо справляется с табличными признаками, которые мы формируем из сигналов. По сути, у нас после фильтрации и агрегации остаётся набор чисел (вариабельность, количество децелераций и т.п.), и для таких данных MLP подходит оптимально.

Модель имеет два скрытых слоя по 128 нейронов с функцией активации ReLU и dropout для защиты от переобучения. Выходы сделаны раздельно: один отвечает за краткосрочный прогноз (норма/патология), а долгосрочные риски сначала пытались учить через отдельные выходы, но позже заменили на правила, так как данные для них слишком шумные.

## **Метрики качества работы системы**

Основной акцент сделан на краткосрочном прогнозе, так как именно здесь у нас есть адекватная разметка и достаточный объём данных. На тестовых данных сеть показала F1 около 0.994 при AUC=1.0 — то есть практически безошибочное различение нормы и патологии. Для долгосрочных задач (hypoxia и emergency) метки строились по правилам, поэтому качество по нейросети было низким. В итоге мы перенесли их в rule-based блок поверх short-term прогноза: если подряд несколько патологических окон — фиксируем гипоксию, если гипоксия держится дольше — фиксируем экстренную ситуацию. Это дало более логичное и стабильное поведение системы.

# **Обоснование выбора технологического стека**

## **Операционная система**

Нами была выбрана ОС Ubuntu Desktop 24.04 версии. Это последняя версия популярной операционной системы, которая имеет дружелюбный интерфейс для пользователя, ранее не пользовавшимся Linux. Ubuntu активно поддерживается сообществом Orange Pi, хорошо интегрируется с одноплатным компьютером, а также имеет свежие обновления пакетов и долговременную поддержку, что упрощает обслуживание ПО и избавляет от необходимости перехода в новую среду.

## **Язык программирования**

Основной язык программирования - Python. Он прост не только в изучении, но и в самом программировании. С его помощью можно быстро и легко написать проект небольших размеров, имеет большую базу различных готовых библиотек, а также удобен в разработке web-сервисов.

Backend написан с использованием FAST API, он имеет следующие преимущества: низкие трудозатраты на написание минимальной конфигурации рабочего сервиса, встроенный swager для документирования API, асинхронный по умолчанию и встроенная валидация данных.

Frontend написан на TypeScript с использованием библиотеки Next.js.

## **База данных**

Основной базой данных выбрана PostgreSQL, известная своей надежностью и расширяемостью, с расширением TimescaleDB, отлично подходящим для хранения временных рядов, что является основными данными в нашей задаче. TimescaleDB позволяет на высокой скорости обрабатывать миллиарды строк, а PostgreSQL позволяет использовать привычный SQL. Преимущества очевидны: хранения обычных данных и временных рядов в одной БД, высокая скорость обработки данных, высокая совместимость.

## **Инфраструктура**

API-документация реализована автоматически swager'ом FAST API, что поддерживает её в актуальном состоянии даже при изменении кода.

WEB-интерфейс полностью «дружелюбен» для пользователя, достаточно минималистичен и дает только необходимую информацию

# **Описание оптимизаций для работы на edge-устройствах**

Мы сознательно избегали тяжёлых архитектур, чтобы система могла работать на любом ноутбуке или даже планшете, вот что вышло:

* Сам MLP лёгкий, занимает считанные мегабайты.
* Признаки считаются заранее, инференс работает очень быстро.
* Если потребуется, модель можно перевести в ONNX или TorchScript и запускать на ARM-процессорах.
* В реальности достаточно даже простого двухъядерного процессора и 1 ГБ памяти.

# **Требования к аппаратному обеспечению**

* 4 CPU, 2 Ггц или больше
* 8 GB ОЗУ
* 2 USB порта или больше
* 30 Гб свободного места на жестком диске
* 1 Ethernet порт (если необходима работа в сети)