# РЕФЕРАТ

Выпускная квалификационная работа содержит 56 страницы текстового документа формата А4, включающего 35 рисунков, 12 использованных источников, 1 приложение.

ДИАГНОСТИКА АПНОЭ ЭЭГ, ВЕБ-ПРИЛОЖЕНИЕ, НЕЙРОСЕТИ, СИСТЕМА ДИАГНОСТИКИ, МЕДИЦИНСКИЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, АНАЛИЗ ЭЭГ, ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ ВРАЧЕЙ.

Объектом исследования является диагностика нарушений дыхания (апноэ сна) с использованием искусственного интеллекта.

Целью выпускной квалификационной работы является проектирование и разработка веб-приложение для диагностирования нарушений дыхания (апноэ сна) с использованием методов искусственного интеллекта.

# Содержание

[Введение 5](#_Toc199836292)

[1 Область исследования 7](#_Toc199836293)

[1.1 Методы диагностики апноэ. 11](#_Toc199836294)

[2 Используемые технологии 15](#_Toc199836295)

[3 Реализация нейронной сети 17](#_Toc199836296)

[3.1 Виды нейросетей 17](#_Toc199836297)

[3.2 Архитектура свёрточной нейросети (CNN) 18](#_Toc199836298)

[3.3 Преимущества использования свёрточной сети CNN для анализа медицинских показателей 19](#_Toc199836299)

[3.4 Архитектура нейронной сети, используемой в проекте 21](#_Toc199836300)

[3.5 Процесс обучение нейронной сети 24](#_Toc199836301)

[3.6 Тестирование нейронной сети 25](#_Toc199836302)

[4 Реализация средства диагностики апноэ 26](#_Toc199836303)

[4.1 Описание типа веб-приложения 26](#_Toc199836304)

[4.2 Структура проекта 27](#_Toc199836305)

[4.3 Общая архитектура программного средства 29](#_Toc199836306)

[4.4 Обучение нейронной сети 31](#_Toc199836307)

[4.5 Маршрутизация приложения 36](#_Toc199836308)

[4.6 Интерфейсы пользователя и их реализация 38](#_Toc199836309)

[4.7 Функциональность веб-приложения 47](#_Toc199836310)

[Заключение 53](#_Toc199836311)

[Список использованных источников 54](#_Toc199836312)

[Приложение А Отчёт о заимствовании 56](#_Toc199836313)

# Введение

В последние годы наблюдается значительный прогресс в области применения технологий искусственного интеллекта для решения различных медицинских задач. Одним из таких направлений является диагностика нарушений дыхания, включая синдром обструктивного апноэ сна (СОАС). Апноэ представляет собой состояние, при котором происходит частичное или полное прекращение дыхания во сне, что может привести к серьезным последствиям для здоровья, таким как сердечно-сосудистые заболевания, инсульт и даже внезапная смерть во сне.

Традиционные методы диагностики СОАС включают полисомнографию – комплексный анализ состояния пациента во время сна, который требует проведения многочисленных тестов и оборудования. Однако эти методы могут быть дорогостоящими и трудоемкими, а также требуют значительных временных затрат со стороны медицинского персонала.

Искусственный интеллект предлагает альтернативный подход к диагностике апноэ, основанный на анализе электроэнцефалограмм (ЭЭГ) пациентов. Скрининг ЭЭГ позволяет выявить характерные паттерны активности мозга, связанные с нарушениями дыхания, без необходимости проведения сложных и длительных исследований.

Основная мысль работы заключается в том, что современные алгоритмы глубокого обучения, в частности сверточные нейронные сети, способны выявлять характерные паттерны нарушений дыхания по данным электроэнцефалографии с точностью, приближающейся к традиционным методам диагностики. Особое внимание в работе уделяется анализу затылочных зон мозга (подробнее этот аспект будет рассмотрен далее), что позволяет повысить достоверность результатов.

Целью данной выпускной квалификационно работы является разработка веб-приложения. В качестве его технологической основы выбраны: язык программирования Python с фреймворком TensorFlow для реализации алгоритмов ИИ, веб-платформа Django для создания удобного интерфейса и монолитности приложения. В работе использованы современные методы обработки и перевода результатов исследования в подходящий для анализа вид. Такой программный продукт представит врачам следующие преимущества:

* возможность проведения в амбулаторных условиях;
* существенное сокращение времени анализа данных;
* автоматизация рутинных процессов;
* упрощение процедуры обследования пациента.

Созданный программный комплекс способен не только автоматизировать процесс диагностики, но и обеспечить врачей надежным инструментом для принятия клинических решений.

Программное средство, разработанное в данной работе, автоматизирует процесс диагностики и обеспечивает врачей надёжным инструментом для принятия клинических решений. Особое значение придается разработке масштабируемого решения, которое может быть адаптировано для использования как в крупных медицинских центрах, так и в учреждениях с ограниченными ресурсами за счёт реализации в виде веб-интерфейса.

# Область исследования

Целью данной работы является разработка программного обеспечения на базе языка программирования Python с использованием фреймворков Django и TensorFlow для создания интеллектуальной системы автоматического анализа ЭЭГ-сигналов, обеспечивающей:

* высокоточное выявление признаков синдрома обструктивного апноэ сна (СОАС) с акцентом на анализ специфических паттернов активности мозга в затылочных областях;
* автоматизацию диагностического процесса: предобработку и нормализацию исходных данных ЭЭГ; автоматическое выделение характерных маркеров апноэ; генерацию заключения для врача.
* удобный интерфейс для медицинского персонала;
* перспективную интеграцию с другими медицинскими сервисами;
* высокую диагностическую точность, сопоставимую с традиционными методами анализа при условии уменьшения времени и финансовой нагрузки для проведения исследования.

Разрабатываемая система призвана стать эффективным инструментом скрининга СОАС, сочетающим преимущества современных технологий искусственного интеллекта с требованиями практической медицины к достоверности, удобству и доступности диагностических методов.

### Применение искусственного интеллекта в диагностировании заболеваний

Искусственный интеллект (ИИ) активно внедряется в медицинскую диагностику благодаря своим возможностям анализа больших объемов данных и принятия решений на основе этих данных. Одним из наиболее перспективных направлений применения ИИ является анализ биомедицинских сигналов, таких, как электрокардиограммы (ЭКГ), электроэнцефалограммы (ЭЭГ) и другие данные, полученные от пациентов. Рассмотрим подробнее некоторые успешные примеры использования ИИ в этой сфере.

### Классификация ЭКГ-сигналов с помощью глубокого обучения

Одним из ярких примеров успешного применения ИИ в медицине является работа [1], в которой авторы предложили использовать глубокое обучение для классификации ЭКГ-сигналов на основе данных о сердечных аритмиях. Они разработали нейронную сеть, которая способна распознавать различные типы аритмических событий с высокой точностью. Исследование показало, что такая модель может стать мощным инструментом для автоматизации процесса диагностики кардиологических заболеваний.

Для разработки модели использовались большие объемы данных ЭКГ, собранные у пациентов с различными видами аритмий. Нейронная сеть была обучена на этих данных и затем протестирована на новых наборах данных. Результаты показали, что модель способна точно классифицировать различные виды аритмии, такие как фибрилляция предсердий, желудочковая тахикардия и экстрасистолы.

Это исследование демонстрирует огромный потенциал ИИ для улучшения качества медицинской диагностики. Автоматизация процессов анализа ЭКГ-сигналов может значительно сократить время, необходимое врачам для постановки диагноза, и уменьшить вероятность ошибок, связанных с человеческим фактором.

### Прогнозирование эпилептических припадков с помощью машинного обучения

Другая интересная область применения ИИ связана с прогнозированием эпилептических припадков. В работе [2] исследователи использовали машинное обучение для анализа ЭЭГ-сигналов с целью обнаружения эпилептических припадков. Они создали модель, способную предсказывать начало приступа за несколько секунд до его начала, что позволяет пациентам принимать меры предосторожности заранее.

Модель была разработана на основе алгоритмов машинного обучения, таких как случайные леса и градиентный бустинг. Эти алгоритмы позволяют анализировать сложные временные ряды данных, такие как ЭЭГ-сигналы, и находить в них закономерности, связанные с началом эпилептического припадка.

Результаты исследования показали, что предложенная модель обладает высокой чувствительностью и специфичностью в прогнозировании эпилептических припадков. Это означает, что она может правильно идентифицировать большинство случаев начинающегося приступа, минимизируя ложноположительные сигналы.

### Диагностика рака легких с использованием компьютерного зрения

Еще одно важное направление применения ИИ связано с диагностикой онкологических заболеваний. Исследователи разработали систему, которая анализирует рентгеновские снимки грудной клетки и выявляет подозрительные участки ткани [2]. Система использует компьютерное зрение (CV) для работы.

Эта технология показала высокую точность в обнаружении опухолей размером менее 10 мм. Модель была обучена на большом количестве рентгеновских снимков, помеченных экспертами-радиологами. После обучения модель смогла автоматически определять наличие опухоли на новых изображениях.

Применение ИИ в такой задаче позволило значительно ускорить процесс диагностики и снизить нагрузку на специалистов, занимающихся интерпретацией изображений. Более того, автоматический анализ снимков может помочь обнаружить рак на самых ранних стадиях, когда лечение наиболее эффективно.

### Обнаружение глаукомы с помощью глубоких сверточных нейронных сетей

Кроме того, ИИ успешно применяется для диагностики глазных заболеваний, таких как глаукома. В работе [3] авторы использовали глубокие сверточные нейронные сети для анализа изображений глазного дна и определения наличия глаукомы. Глаукома – это хроническое заболевание глаза, которое может привести к потере зрения, поэтому ранняя диагностика крайне важна.

Модель была обучена на большом наборе изображений глазного дна, полученных от здоровых людей и пациентов с глаукомой. Результаты показали, что модель способна точно отличать здоровые глаза от пораженных глаукомой, что открывает новые перспективы для автоматизированной диагностики этого заболевания.

Точность работы данной модели составляет 98,6% с чувствительностью в 95%. Также в исследовании указаны причины понижения точности – сопутствующие заболевания глаз со схожей симптоматикой при исследовании одного параметра.

### Применение ИИ в диагностике COVID-19

В условиях пандемии COVID-19 ИИ стал важным инструментом для ускорения диагностики и мониторинга распространения вируса. В работе [4] авторы представили модель, основанную на глубоком обучении, которая способна анализировать КТ-изображения легких и обнаруживать признаки пневмонии, вызванной COVID-19.

Модель была обучена на большом объеме данных КТ-изображений, собранных у пациентов с подтвержденным диагнозом COVID-19. Она продемонстрировала высокую точность в определении наличия вирусной пневмонии, что помогает врачам быстрее ставить диагноз и начинать соответствующее лечение.

### Вывод актуальности применении ИИ в диагностировании заболеваний

Примеры, приведенные выше, показывают, насколько разнообразными и эффективными могут быть применения ИИ в медицинской диагностике. От анализа ЭКГ-сигналов и прогнозирования эпилептических припадков до диагностики рака легких и глаукомы — ИИ становится незаменимым помощником для врачей и исследователей.

В контексте нашего исследования применение ИИ для анализа ЭЭГ-сигналов в целях диагностики апноэ представляется особенно перспективным направлением. Использование современных алгоритмов машинного обучения и глубокого обучения может позволить создать эффективные инструменты для автоматической диагностики этого распространенного заболевания, что приведет к улучшению качества жизни миллионов людей.

Также на сегодняшний день проводятся десятки различных исследований и разработок для создания масштабных сервисов, которые в перспективе позволят врачам приблизить точность диагностирования к 100%.

## Методы диагностики апноэ.

Диагностика синдрома обструктивного апноэ сна (СОАС) включает использование нескольких методов, каждый из которых имеет свои особенности, преимущества и ограничения. Рассмотрим основные из них подробно.

### Полисомнография

Полисомнография является золотым стандартом диагностики СОАС. Этот метод предполагает проведение комплексного обследования пациента во время сна, включающего мониторинг множества физиологических параметров, таких как электрическая активность мозга (ЭЭГ), движения глаз (ЭОГ), мышечная активность (ЭМГ), дыхание, пульсоксиметрия и другие показатели.

Преимущества полисомнографии:

* высокая точность – полисомнография предоставляет полную картину состояния пациента во время сна, позволяя точно определить наличие и степень тяжести апноэ;
* комплексность – мониторинг большого количества параметров дает возможность оценить влияние апноэ на различные системы организма, такие как сердечно-сосудистая и нервная системы;
* объективность – результаты полисомнографии объективны и не зависят от субъективных ощущений пациента.

Недостатки полисомнографии:

* дорогостоящие оборудование и персонал – проведение полисомнографии требует наличия специализированного оборудования и высококвалифицированных специалистов, что увеличивает стоимость процедуры;
* неудобства для пациента – пациенту необходимо провести ночь в клинике, что может вызывать дискомфорт и влиять на качество сна;
* ограниченная доступность – из-за высоких требований к оборудованию и персоналу полисомнография доступна далеко не всем пациентам.

### Домашний мониторинг

Домашний мониторинг представляет собой упрощенную версию полисомнографии, которую пациент может проводить самостоятельно дома. Обычно используются портативные устройства, регистрирующие ограниченное количество показателей, таких как уровень кислорода в крови, частота пульса и дыхательные усилия.

Преимущества домашнего мониторинга:

* удобство – пациент может проводить процедуру в домашних условиях, что снижает стресс и повышает комфорт;
* доступность – портативные устройства стоят дешевле, чем оборудование для полисомнографии, и доступны большему числу пациентов;
* быстрота – процедура занимает меньше времени, чем полная полисомнография, и результаты могут быть получены быстрее.

Недостатки домашнего мониторинга:

* низкая точность – ограниченный набор измеряемых параметров приводит к меньшей точности диагностики по сравнению с полисомнографией;
* риск неправильного использования – пациенты могут неправильно установить или использовать устройство, что повлияет на достоверность результатов;
* отсутствие комплексной оценки: домашний мониторинг не позволяет получить полную картину влияния апноэ на организм, как это возможно при полисомнографии.

### Анализ электроэнцефалограмм

Анализ электроэнцефалограмм (ЭЭГ) является одним из перспективных методов диагностики апноэ. ЭЭГ регистрирует электрическую активность мозга, которая изменяется в зависимости от фаз сна и наличия дыхательных пауз. Использование ИИ для анализа ЭЭГ-сигналов позволяет автоматизировать процесс выявления характерных паттернов, связанных с апноэ.

Преимущества анализа ЭЭГ:

* простота и доступность: – для проведения ЭЭГ требуется минимальное оборудование, что делает метод доступным для широкого круга пациентов;
* высокая чувствительность – ЭЭГ способна уловить мельчайшие изменения в электрической активности мозга, что позволяет выявить даже незначительные нарушения дыхания;
* автоматизация – использование ИИ для анализа ЭЭГ-сигналов позволяет автоматизировать процесс диагностики, снижая необходимость участия специалиста.

Недостатки анализа ЭЭГ:

* требует дополнительных исследований – хотя ЭЭГ может указать на наличие апноэ, для подтверждения диагноза часто требуются дополнительные тесты, такие как полисомнография;
* зависимость от качества сигнала – качество записи ЭЭГ влияет на точность диагностики, поэтому важно соблюдать правильные условия проведения процедуры;
* необходимость калибровки моделей ИИ – модели ИИ, используемые для анализа ЭЭГ, должны быть тщательно откалиброваны и проверены на больших объемах данных, чтобы гарантировать надежность результатов.

### Заключение по методам диагностики

Каждый из рассмотренных методов диагностики апноэ имеет свои сильные и слабые стороны. Выбор конкретного метода зависит от целей исследования, доступности оборудования и предпочтений врача. Современные технологии, такие как искусственный интеллект и высокоскоростной анализ данных, открывают новые горизонты в диагностике апноэ, делая этот процесс более эффективным и доступным. В процессе выполнение ВКР принято решение использовать скрининги ЭЭГ для построения системы диагностики апноэ.

# используемые технологии

В данной работе для реализации веб-приложения и обработки данных выбраны следующие технологии:

Python – основной язык программирования проекта. Python выбран благодаря своей гибкости, простоте и широкому спектру библиотек для научных вычислений и машинного обучения. Он идеально подходит для разработки приложений, работающих с данными и искусственным интеллектом.

Django – высокоуровневый веб-фреймворк на Python, который использован для создания серверной части приложения. Django обеспечивает быстрый и безопасный способ разработки веб-приложений, предлагая готовые решения для аутентификации пользователей, управления базой данных, маршрутизации запросов и безопасности.

TensorFlow – библиотека для машинного обучения, разработанная Google. Она использована для построения и тренировки нейронной сети, которая анализирует ЭЭГ-сигналы для диагностики апноэ. TensorFlow предоставляет мощные инструменты для реализации глубокого обучения и оптимизирован под выполнение на GPU, что ускоряет обработку данных.

NumPy – библиотека для научных вычислений в Python. Она используется для эффективной работы с многомерными массивами данных, необходимыми для обработки ЭЭГ-сигналов. NumPy предоставляет множество функций для математической обработки данных, таких как операции с матрицами, линейная алгебра и генерация случайных чисел.

SQLite – легкая реляционная база данных, встроенная непосредственно в Python. Она использована для хранения весов модели нейронной сети и промежуточных данных. SQLite идеально подходит для небольших проектов, так как не требует установки отдельного сервера базы данных и поддерживает транзакционную целостность данных.

Исходные данные для обучения и тестирование хранятся в файлах расширением .REC, которые размещены в специальных папках, организованных специальным образом (рисунок 1). Папка NP 1 хранит в себе две папки с анализами NR 1 и NR 2, которые в свою очередь хранят файл записи ЭЭГ (с форматом .REC) и служебную информацию об исследовании в файле с форматом .cm.

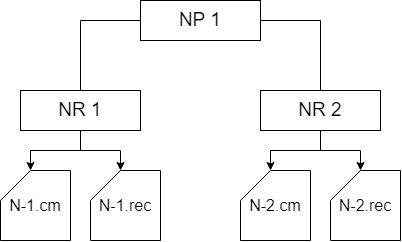


Рисунок – Файловая структура исходнх данных

Файлы с форматом .REC широко применяются в медицинской сфере за счёт компактности данных, удобства их размещения и обширной поддержки различным медицинским программным обеспечением. Сам файл хранит временные ряды данных и различные мета-данные о текущей записи. Данный формат является наиболее популярным. Однако для работы с файлом, необходимо преобразовать его в числовые ряды нужной кодировки, так как библиотеки Tensorflow и NumPy не поддерживают его напрямую. О логики преобразовании формата будет рассказано далее в соответствующей главе.

# Реализация нейронной сети

## Виды нейросетей

Как было сказано ранее, нейросетевые технологии нашли своё применение в диагностировании различных заболеваний с особой точностью и минимальной погрешностью. В связи с этим именно на искусственный интеллект делается опора в текущем проекте.

В данный главе представлен подробный разбор архитектур нейронных сетей, их принципов работы, а также описание свёрточной нейронной сети, которая используется в работе.

Перед тем, как начать подробное представление нейросети, которая используется в проекте, необходимо разобраться в основных видах архитектур нейросетей и их области применения.

В качестве первого вида рассмотрим рекуррентные нейронные сети (RNN). Данный тип представляет из себя сети с циклами, которые подходят для обработки последовательностей. Обучение сети происходит обычным образом, за исключением алгоритма обратного распространения ошибки, в котором нам нужно учитывать все предыдущие шаги (алгоритмом обратного распространения ошибки сквозь время). Данный вид используется для анализа текста, автоматического перевода, прогнозирования соседних пикселей (на основе текущего), распознавание эмоций и описание изображений.

Теперь рассмотрим глубокие нейронные сети (DNN). Нейронная сеть такого типа представляет собой множество слов искусственных нейронов и является развитием (или следствием) многослойных перцептронов и позволяют обрабатывать сложные и нелинейные зависимости в данных. Такое свойство достигается за счёт использования иерархических признаков. Если говорить конкретнее, то DNN это общий тип сетей, а RNN и CNN, о которой будет рассказано далее, это частные случаи DNN с изменённое архитектурой.

И последнем видом архитектуры, перед разбором используемой свёрточной нейросетевой архитектурой, рассмотрим гибридные архитектуры, которые включают в себя другие виды нейросетей в качестве слоёв или точек входа.

## Архитектура свёрточной нейросети (CNN)

Свёрточные нейронные сети (CNN) предназначены для эффективного извлечения признаков из данных с пространственной структурой, таких как изображения или временные ряды. Скрининг ЭЭГ в формате .REC как раз представляет собой такие пространственные данные временных рядов.

Архитектура свёрточной нейронной сети включает 5 основных слоёв, которые описаны далее.

Входной слой принимает данные в виде матрицы, одномерных рядов или тензоров и может включать нормализацию или стандартизацию.

Свёрточные слои являются основными компонентами свёрточных сетей и отвечают за выделение локальных признаков (паттернов) и создаёт карту этих самых признаков.

Слой подвыборки (далее «пулинга» от англ. “Pooling” – объединение) отвечает за уменьшение размерности карт признаков, сохраняя при этом важную информацию и повышая устойчивость к сдвигам и шуму. Чаще всего применяет один из двух видов: «Max polling» – выбор максимального значения из области или «Average Pooling» – среднее значение.

Полносвязные слои отвечают за интерпретацию данных с группы свёрточных и пулинговых слоёв в итоговое предсказание и работает аналогично обычным нейросетям.

Выходной слой производит окончательное предсказание (в нашем случае указывает вероятность принадлежности к классу «нарушение дыхания»). Прогнозирование основано на следующих функция активации, которые используются в зависимости от задач: «Softmax» - используется для классификации, в то время как «Sigmoid» для бинарных задач.

Удобство свёрточной нейронной сети также заключает в том, что снижается необходимость ручной разметки и обработки данных.

## Преимущества использования свёрточной сети CNN для анализа медицинских показателей

Свёрточные нейронные сети (CNN) на сегодняшний день являются одними из наиболее эффективных методов анализа медицинских данных, особенно когда речь идет об изображениях и сигналах с чётко выраженной пространственной или временной структурой. Ниже рассмотрены основные преимущества, которые делают использование CNN предпочтительным для диагностики медицинских показателей, таких как электроэнцефалограммы (ЭЭГ).

Одним из ключевых преимуществ свёрточных нейронных сетей над другими является способность самостоятельно извлекать важные характеристики данных без участия человека. CNN автоматически обнаруживает ключевые паттерны, такие как формы волн или аномалии в сигнале ЭЭГ, избавляя исследователей от необходимости вручную выделять признаки. Это существенно ускоряет процесс анализа и уменьшает вероятность субъективных ошибок, обусловленных человеческим фактором.

CNN отлично справляются с анализом пространственно-временных структур данных, таких как ЭЭГ или медицинские изображения (МРТ, КТ, рентген). Благодаря механизму свёртки сети могут выявлять даже самые тонкие изменения в структуре сигнала, которые трудно обнаружить стандартными методами. Это позволяет значительно повысить точность диагностики. Далее рассмотрим их иерархическую структуру анализа.

Свёрточные сети анализируют данные послойно, начиная от простых признаков (линии, изгибы сигналов, пики и спады) и заканчивая более сложными паттернами (аномалии, признаки заболевания). Такая иерархическая обработка даёт нейросети возможность детально «понимать» и обобщать информацию, делая её особенно эффективной для диагностики.

Также свёрточные нейронные сети наименее подвержены переобучению за счёт использования слоёв подвыборки (пулинга) и регуляризации, что позволяет снизить количество обучаемых параметров и уменьшить риск переобучения нейросети. Это значит, что сеть хорошо обобщает знания и показывает стабильные результаты на новых, ранее неизвестных данных.

Важным преимуществом также является легкость адаптации к различным типам данных и задачам. Кроме того, современные инструменты (например, TensorFlow или PyTorch) позволяют быстро создавать, тестировать и масштабировать CNN для различных медицинских приложений. Это существенно ускоряет и упрощает процесс разработки.

Для медицинской области применения нейросетей также важно максимально повышать адаптацию к шумам (чтобы сеть не реагировала на различный шум). CNN обладают высоким уровнем устойчивости к шумам, искажениям и смещениям исходных данных. В медицинской практике сигналы часто подвержены различным внешним помехам и артефактам, поэтому способность сети правильно интерпретировать данные даже при наличии шума крайне важна.

Отдельно хотелось бы отметить большое количество предобученных моделей по типу ResNet, VGG или Inception (такие модели обучены на сотни тысяч единиц данных), что позволяет быстрее внедрить модели с систему. В медицинской сфере это также позволяет повысить устойчивость к шуму и стабилизировать модель.

Таким образом, свёрточные нейронные сети обеспечивают эффективный, точный и надёжный инструмент анализа медицинских данных, что делает их особенно востребованными в области диагностики заболеваний, в том числе синдрома обструктивного апноэ сна на основе анализа ЭЭГ. Опираясь на эти преимущества, в качестве модели была выбрана именно свёрточная нейронная сеть.

## Архитектура нейронной сети, используемой в проекте

В качестве нейронной сети, которая используется в проекте, выбрана архитектура свёрточной нейронной сети.

Архитектура нейронной сети, реализованной в данной работе, построена на основе последовательной модели (Sequential), использующей свёрточные нейронные слои (Conv2D). Рассмотрим подробно структуру сети и причины выбора каждого слоя.

Входной слой (Input Layer). Входной слой принимает предварительно обработанные сигналы ЭЭГ в виде матрицы данных. Его задача состоит в том, чтобы обеспечить корректную передачу входных данных в нейронную сеть, сохраняя исходные характеристики сигнала, что является необходимым для последующего глубокого анализа.

Сверточные слои (Conv2D). Сверточные слои являются основными компонентами архитектуры CNN. В нашей модели используются два сверточных слоя с количеством фильтров 64 и 128 соответственно:

* первый сверточный слой с 64 фильтрами извлекает базовые локальные признаки сигналов, такие как простые паттерны и колебания. Также в первом слое используется ядро 3х3 и входной размерность 512х512;
* второй сверточный слой с 64 фильтрами используется для более подробного извлечения первичный признаков;
* третий сверточный слой с 128 фильтрами предназначен для выявления более сложных и абстрактных признаков, которые характеризуют специфические паттерны нарушения дыхания.

Функция активации ReLU (Rectified Linear Unit) выбрана за её способность эффективно справляться с исчезающим градиентом и ускорять обучение сети. Использование параметра padding="same" сохраняет размерность данных после свертки, позволяя сети глубоко изучать входные сигналы.

Слои подвыборки (MaxPooling2D) или пулинга применяются для уменьшения размерности полученных карт признаков. Метод максимального пулинга (MaxPooling2D) выбирает максимальное значение из небольших участков карты признаков:

* уменьшение размерности данных снижает вычислительные затраты и делает сеть более устойчивой к шумам и незначительным вариациям входного сигнала;
* пулинг помогает выделить доминирующие признаки, улучшая способность сети к обобщению.

Слой выравнивания (Flatten) преобразует многомерные данные (карты признаков) в одномерный вектор, что необходимо для передачи данных в последующие полносвязные слои. Этот этап является обязательным при переходе от сверточной части к полносвязной.

Полносвязный слой (Dense) с 256 нейронами используется для интеграции и анализа признаков, полученных на предыдущих этапах обработки данных. Функция активации ReLU помогает сети выявлять сложные нелинейные зависимости и увеличивает эффективность обучения, выделяя более важные и релевантные признаки.

Регуляризационный слой Dropout используется с вероятностью отключения нейронов 50%. Это необходимо для снижения риска переобучения, поскольку Dropout предотвращает чрезмерное «запоминание» тренировочных данных и улучшает способность модели обобщать знания на новых данных.

Нормализующий слой BatchNormalization стабилизирует и ускоряет процесс обучения нейросети, нормализуя входные данные каждого слоя и предотвращая проблемы, связанные с внутренним сдвигом данных (internal covariate shift). Это позволяет использовать более высокие скорости обучения и улучшает итоговую точность модели.

Выходной полносвязный слой (Dense с активацией Softmax). Выходной слой использует активацию Softmax и состоит из двух нейронов, соответствующих двум классам: «Нет апноэ» и «Апноэ». Softmax позволяет интерпретировать результаты сети как вероятности принадлежности входного образца к каждому из классов, обеспечивая медицинскому персоналу ясные и интуитивно понятные результаты анализа.

На рисунке 2 показан программный код конструктора класса Network, создающий объект с описанными параметрами сети.



Рисунок – Архитектура сети

Архитектура текущей нейронной сети была получена в ходе изменения и переобучения, пока точность не превысила порога в 80% и обеспечила качественное диагностирование апноэ.

## Процесс обучение нейронной сети

Процесс обучения разработанной нейронной сети проводился на подготовленном наборе данных, состоящем из записей ЭЭГ в формате .REC, полученных от пациентов с подтвержденным диагнозом [5] синдрома обструктивного апноэ сна (СОАС) и здоровых испытуемых. Рассмотрим подробно этапы и методы, использованные для эффективного обучения сети.

На начальном этапе данные в формате .REC преобразованы в удобный для работы формат, и проведена их нормализация для обеспечения единообразия и повышения точности обучения. Все данные разделены на обучающую и тестовую выборки в пропорции 80% и 20% соответственно. Это разделение позволило адекватно оценить качество обучения и способность модели к обобщению.

Для обучения сети использована функция потерь Cross-Entropy (перекрестная энтропия), которая подходит для решения задач бинарной классификации и обеспечивает наиболее эффективное обучение путем минимизации ошибки между предсказанными и реальными значениями классов. В качестве оптимизатора выбран алгоритм Adam, характеризующийся высокой эффективностью и быстрой сходимостью при обучении глубоких нейросетей.

Обучение нейросети проводилось в несколько эпох, с применением метода ранней остановки (early stopping), чтобы избежать переобучения. В ходе обучения постоянно отслеживалась точность и потеря на обучающем и валидационном наборах данных. Методом валидации выбрана перекрёстная проверка (cross-validation), которая обеспечивает надежную оценку качества модели.

Для предотвращения переобучения применялись следующие методы:

* «Dropout» слой с коэффициентом отключения нейронов 50% для регуляризации сети;
* «BatchNormalization», который стабилизировал процесс обучения и позволял сети быстрее сходиться к оптимальному решению;
* «Early stopping», чтобы завершить обучение модели в тот момент, когда точность на валидационном наборе переставала улучшаться.

После завершения процесса обучения сеть протестирована на отдельной тестовой выборке данных. Итоговые результаты показали точность классификации выше 86%, что подтверждает эффективность выбранной архитектуры и методов обучения. Дополнительно проведён анализ результатов с использованием матрицы ошибок (confusion matrix), которая позволила визуально оценить эффективность распознавания классов и выявить типичные ошибки.

Таким образом, проведенный процесс обучения нейросети обеспечил высокую точность диагностики и подтвердил возможность использования разработанной системы в практических медицинских приложениях для автоматизированного анализа и диагностики СОАС на основе ЭЭГ-данных.

## Тестирование нейронной сети

Также после разработки архитектуры нейронной сети и её обучения, необходимо проверить её работоспособность и уровень точности. Тестирование проводилось на исходных данных в виде автоматического прохода по каждому из семплов. В ходе тестирование выявлено, что нейронная сеть достаточно точно определяет наличие апноэ. Результат представляет из себя число в диапазоне [0, 1], которое интерпретируется нами как вероятность, что обнаружено апноэ. При тестировании среднее значение точности получено 86% (или 0.86 в исходном диапазоне).

# Реализация средства диагностики апноэ

## Описание типа веб-приложения

Набор технологий, выбранных для реализации проекта, позволяет спроектировать и создать полноценное fullstack веб-приложение.

Полноценное fullstack-приложение на Django может включать как серверную логику на Python, так и клиентскую часть, представленную HTML-шаблонами и JavaScript, что позволяет разрабатывать веб-приложения без использования стороннего фронтенда.

Разработка веб-приложения разделена на четыре основных этапа:

Проектирование архитектуры – формулировка функциональных требований, выбор формата данных, разработка UI-интерфейса и модульного разбиения приложения.

Обучение нейросети – скрининг ЭЭГ датасета (50 размеченных сессий), раздельное обучение двух CNN моделей для правого и левого затылочных каналов.

Интеграция модели в веб приложение – разработка эндпоинта, автоматическое преобразование формата REC в NumPy и единая обвязка вывода вероятностей (точность классификации после финального тюнинга превысила 86 %).

И тестирование всего приложения на работоспособность.

Данная глава описывает, как реализовано средство для диагностики в виде веб-приложения: загрузка и нормализация данных, блок предобработки сигналов, архитектура и обучение CNN ядра, сервис предсказаний, веб интерфейс для врача, подсистема тестирования. Такой «сквозной» разбор призван показать шаг за шагом как академические алгоритмы трансформируются в надёжное программное средство, готовое к реальному применению в клинике.

## Структура проекта

В этом разделе разберём структуру проекта программного средства, она представлена на рисунке 3.

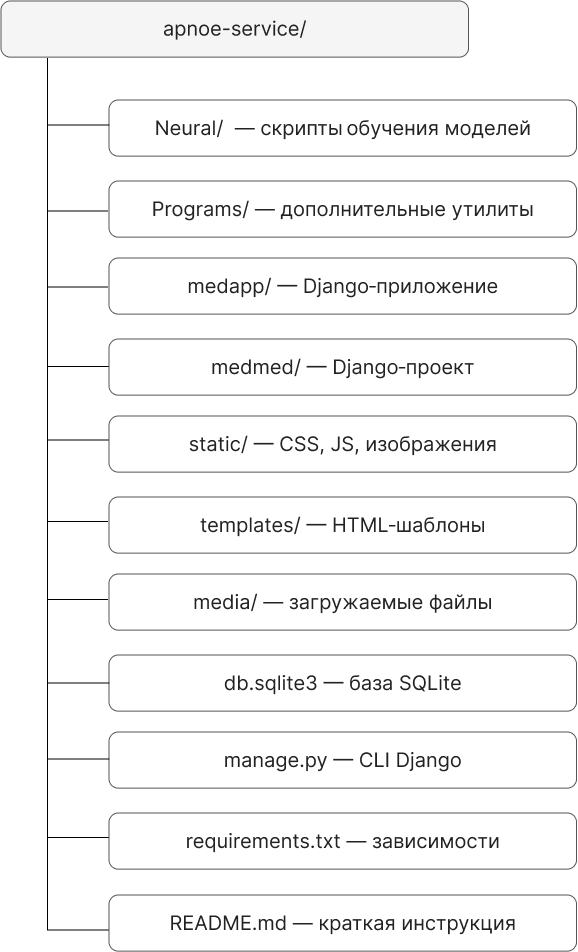


Рисунок – Структура проекта.

Главным (корневым) модулем является medmed, он содержит все настройки приложения, отвечающие за запуск модулей, работу с базой данных, инициализацию путей до файлов и множество других настроек, например, переменная ROOT\_URLCONF = 'medmed.urls', указывающая путь к корню маршрутизации (корень может быть определён в любом модуле, но принято брать за корень маршрутизации маршрутизацию корневого модуля); массив объектов DATABASES для определения баз данных и путей для подключения к ним (поддерживается одновременное подключение к нескольким базам, их работа регулируется ORM Django); а также временные и языковые настройки.

Модуль medapp содержит в себе весь функционал разрабатываемого приложения. Здесь расположены модели, views и другие файлы.

Модуль Programs содержит в себе утилиты и программы, необходимые для декодирования ASCII кода в файлах с расширением REC.

Модуль Neural отвечает за архитектуру и ядро искусственного интеллекта.

Папка templates хранит в себе HMTL разметку с использованием шаблонизатора Django Template Language (далее DTL) – это собственная система шаблонов Django, позволяющая создавать текстовый документ или строку Python, размеченную с использованием языка шаблонов Django [6]. Примеры использования шаблонизатора будут представлены далее в ходе подробного описания модуля medapp.

Папка static отвечает за хранение статических файлов, таких как стилевые файлы CSS, скрипты JavaScript, а также изображения, файлы и другое. Путь до папки static определяется в medmed/settings.py и выглядит следующим образом (рисунок 4): переменная STATIC\_URL отвечает за путь до папки на сервере после команды сборки статики (python manage.py collectstatic – команда собирает все статические файлы в папку STATIC\_URL); переменная STATICFILES\_DIRS содержит массив с путями до папок со статикой – их может быть несколько, допустим, в каждом из модулей присутствуют свои статические файлы.

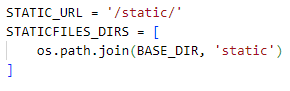


Рисунок – Инициализация папки static.

Текстовый файл requirements.txt хранит список библиотек, подключенных к проекту. Файл генерируется автоматически с помощью команды pip freeze > requirements.txt. Файл содержит список библиотек и их версий, используемых в проекте (рисунок 5).

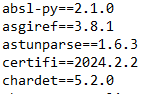


Рисунок – Список используемых библиотек.

Файл manage.py – универсальный CLI (инструмент командной строки) проекта Django. Его главная задача – упростить ежедневную работу разработчика и администратора, предоставив единую точку доступа к наиболее частым операциям. Файл отвечает за такие операции, как запуск локального сервера разработки, создание/применение миграций БД, управление пользователями, сбор статических файлов и многое другое.

Папка media отвечает за хранение данных, которые загружаются на сервер (если быть точнее, то в приложение). Путь до этой папки указывается в настройках medmed/settings.py следующим образом MEDIA\_URL = '/media/'  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media') и гарантирует доступ к папке из любого места программы.

## Общая архитектура программного средства

В основу системы положена многослойная архитектура, разделяющая ответственность между веб‑частью под руководством Django и ядром искусственного интеллекта TensorFlow (рисунок 6). Такое деление позволяет независимо развивать пользовательский интерфейс, бизнес‑логику и алгоритмическую часть, а при необходимости масштабировать только «узкие» места.

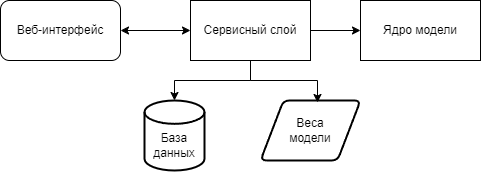


Рисунок – Многослойная архитектура проекта.

Разберём ключевые слои и их зоны ответственности:

Слой представления (он же User Interface) включает в себя такие компоненты как Django templates, статические файлы, JavaScript‑виджеты и отвечает за приём файлов ЭЭГ, вывод прогресса, визуализация результата врачу. Данный слой будет подробно описан в главе о проектировании интерфейса.

Сервисный слой отвечает за обработку файлов, предсказание и подведение итогов тестирования. Он включает в себя такие компоненты, как Django Views, обработчики и различные энпоинты.

Ядро искусственного интеллекта содержит модуль Neural и Wights, в которых содержатся файлы-описания архитектуры сети и веса соответственно. Слой отвечает за строение нейронной сети, её работоспособность и веса, полученные в ходе обучения сети.

Слой хранения данных представлен реляционной базой данных SQLite – это библиотека на языке C, реализующая небольшой, быстрый, автономный, высоконадежный, полнофункциональный движок базы данных SQL. SQLite – наиболее используемый движок базы данных в мире. SQLite встроен во все мобильные телефоны и большинство компьютеров, а также поставляется в комплекте с бесчисленным множеством других приложений, которые люди используют каждый день [7]. Именно такие характеристики подходят для широкого распространения её вместе с разрабатываемым продуктом.

Данная архитектура программного средства позволяет с лёгкостью производить обновления отдельных компонентов системы, не отключая при этом всю систему в целом – тем самым обеспечивая бесшовную работу всего продукта.

## Обучение нейронной сети

Для обучения и тестирования нейронной сети использовался язык программирования Python с библиотеками Keras, TensorFlow и sklearn. Обучение нейронной сети можно разделить на стадии подготовки данных и их загрузки, инициализация двух нейросетей, а также стадия обучения и сохранения весов.

### Стадия подготовки и загрузки данных.

Стадия подготовки и загрузки данных начинает процесс обучения нейронной сети (рисунок 7).

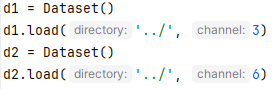


Рисунок – Загрузка данных в процесс обучения нейронной сети

Для загрузки данных используется класс Dataset c методом load, который принимает в качестве аргументов путь до директории, где хранятся файлы для обучения, а также какой из 21 каналов в скрининге нужно выделить для обучения. Здесь важно отметить, что мы обучаем по одной нейронной сети (одной архитектуры) для каждого канала затылочной зоны. В нашем случае мы используем 3 и 6 каналы, так как они наиболее полно отображают изменения в дыхании человека.

После инициализации класса Dataset мы используем метод load. Внутри метода мы проходим по структуре данных для обучения, которая была описана ранее. В методе мы размечаем данные на «болеет апноэ» и «здоров», используя массив y со значениями 0 и 1. Далее, так как данные для обучения хранят два сегмента, мы вынуждены дважды рассматривать один и тот же файл, используя код, представленный на рисунке 8.

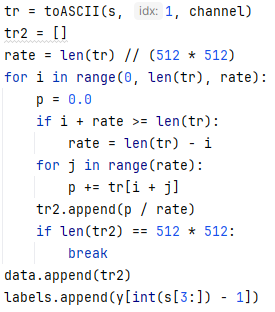


Рисунок – Расшифровка файла скрининга.

Расшифровка файла выполняется с помощью авторской функции toASCII (рисунок 9), которая принимает в качестве значения название папки, которую мы рассматриваем (допустим, оно равно Nr 1 и содержит N-1.REC), индекс сегмента и номер канала.

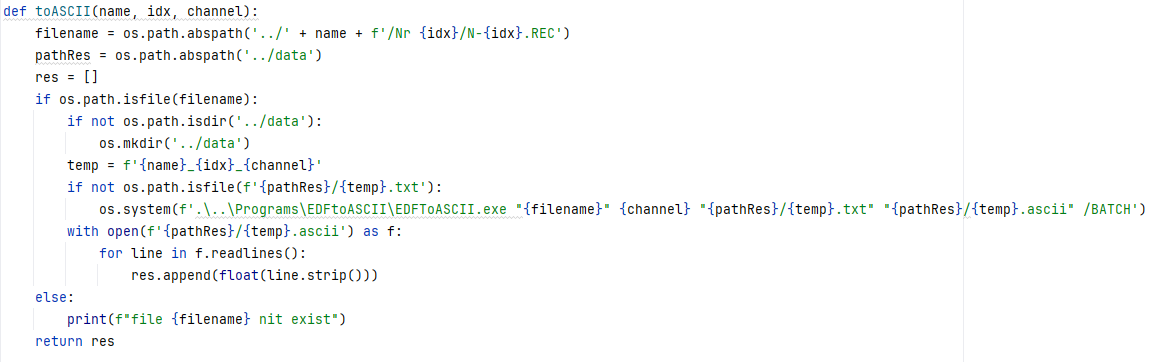


Рисунок – Функция-расшифровщик файла исследования.

Функция формирует путь до файла filename, а далее с помощью встроенной библиотеки os (для взаимодействия с системой) выполняет преобразование значений файла к типу ASCII. Изначально данные хранятся в формате EDF и не представляют возможности для явного чтения. EDF – это стандартный формат файла, который предназначен для хранения и обмена медицинскими данными с временными рядами. Для расшифровки используется программа EDFtoASCII, которая преобразует данные EDF в ASCII.

После преобразования в нужный формат данные переводятся в числа с плавающей точкой и сохраняются в результирующем массиве, который после возвращается в качестве результата работы функции.

Следующим этапом расшифрованные и размеченные данные преобразуются в матрицу формата 512 на 512, так как именно в таком формате они должны попадать в нейросеть в качестве обучающих семплов.

### Инициализация нейронных сетей

Как было сказано ранее, происходило обучение двух нейронных сетей, поэтому инициализируем обе сети (рисунок 10).



Рисунок – Инициализация нейронных сетей

Класс Network включает в себя три метода: train – отвечает за обучение нейронной сети; loadWeights – отвечает за загрузку весов при работе нейронной сети (в качестве аргумента указывается путь до весов) и saveWeights – отвечает за сохранение весов модели в файл. Метод \_\_init\_\_ содержит в себе архитектуру нейронной сети (рисунок 11).



Рисунок – Архитектура нейронной сети

Также на рисунке 12 представлены вышеописанные функции, отвечающие за загрузку и сохранение весов. Данные функции используют встроенные методы из библиотеки Keras. Однако функция train, пусть и использует встроенные метод из библиотеки, требует подробной настройки.

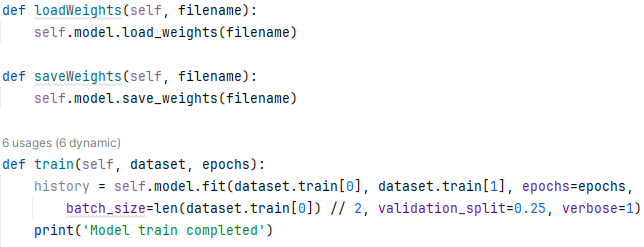


Рисунок – Функции модели Network

В функции fit первым аргументом указываются значения x – то есть данные для обучения. Вторым аргументом указываются значения y – то есть метки, задающие, есть у пациента апноэ или нет. Следом указывается количество эпох для обучения, размер batch\_size – то есть количество примеров, которая нейросеть обрабатывает за один шаг обновления весов, а также указывается сколько данных из представленных будет использовано для обучения, в нашем случае мы всегда берём 25% от исходных данных; аргумент verbose указывает, что нужно выводить в консоль логи обучения нейронной сети.

### Тренировка и сохранение весов нейронной сети

Тренировка нейронной сети происходит последовательно (рисунок 13): сначала обучается нейронная сеть, отвечающая за третий канал, после отвечающая за шестой.

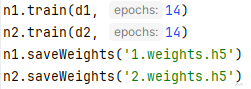


Рисунок – Тренировка и сохранение нейронной сети.

Функция train была описана ранее, однако стоит отметить выходные данные на этапе обучения. В ходе нескольких попыток обучения и исходя из графиков точности, было принято решение использовать 14 эпох обучения, так как это приводит к наиболее точной работе нейронной сети.

После обучение происходит сохранение файлов, используя методы класса Network.

## Маршрутизация приложения

Маршрутизация в проекте разделена на две части: первая часть находится в головом модуле medmed, который отвечает за всю настройку приложения; вторая часть находится в medapp и отвечает за настройку конкретно рабочей части проекта.

Перед тем, как разбираться в маршрутизации приложения, рассмотрим, как она работает в ядре Django. Официальная документация гласит следующее [8]: «Чтобы разработать URL-адреса для приложения, вы создаете модуль Python, неофициально называемый URLconf (конфигурация URL-адреса). Этот модуль представляет собой чистый код Python и является отображением между выражениями пути URL и функциями Python (вашими представлениями).». Фактически, мы создаём такой модуль Python, который хранит связь «адрес в сети» - «функция рендера» - «глобальное название пути в приложении». Адрес в сети – эта та часть, которая отображается после доменного имени сайта. Функция рендера – функция или класс (с обязательным методом as\_view), позволяет сформировать html код страницы по текущему адресу. Глобальное название пути в приложении – это уникальное имя для этого адреса. Так как в одном проекте находятся множество приложений с одинаковыми адресами (относительно корня этого приложения), то необходимо задавать уникальные имена, чтобы к ним можно было обратиться из любой точки проекта. На рисунке 14 показана маршрутизация внутри приложения диагностирования. Здесь мы определяем 4 адреса: главная страница (не имеет отличия от домена), страница загрузки скрининга ЭЭГ, страница сохранения результата (то есть скачивания данных в pdf) и страница с инструкцией по работе с веб-приложением. Каждая из страниц будет описана отдельно с её графическим интерфейсом, подробным описанием как страницы, так и функции её рендера.

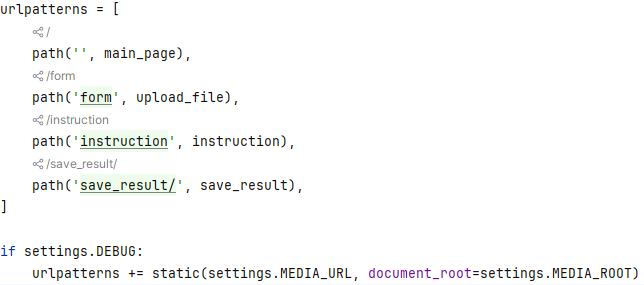


Рисунок – Настройка маршрутизации приложения.

Однако, чтобы приложение диагностики было доступно в карте маршрутов, его необходимо подключить в настройках в переменной INSTALLED\_APPS, которая представлена массивом и включает в себя названия всех модулей, подключенных к корню проекта (или требующих подключения к нему). Также в корневой папке проекта medmed в модуле urls.py необходимо подключить маршруты приложений (рисунок 15).

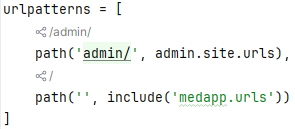


Рисунок – Подключение маршрутов приложения к проекту

Для подключения в методе path нужно указать маршрут, по которому будет доступно приложение. В нашем случае указана пустая строка. Это означает, что маршруты приложения medapp (приложения диагностики) будут считаться от доменного имени веб-приложения. Вторым аргументом была передана функция include, которая включает в себя путь к маршрутам приложения (без указания расширения модуля маршрутизации приложения). Хотелось бы отдельно отметить, что существует два варианта создания маршрутов – используя функцию path или её аналог re\_path, в котором адрес представляет из себя регулярное выражение. На рисунке 16 представлен пример комбинированного использования path и re\_path в рамках одного приложения (пример взят из документации [8]).

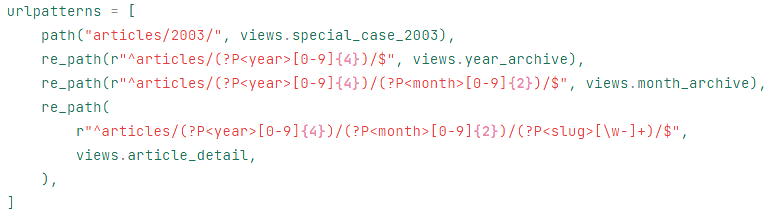


Рисунок – Пример комбинированного использования маршрутов.

## Интерфейсы пользователя и их реализация

Пользовательский интерфейс (или UI – user interface) – это визуальное представление интерфейса пользователя, включающее в себя кнопки, их расположение, разметку и цветовую гамму решений и многое другое. Именно через пользовательский интерфейс врач будет взаимодействовать с программным средством диагностики.

### Инструменты при разработке интерфейса.

Основным инструментом при проектировании пользовательского интерфейса стала библиотека Bootstrap.

Bootstrap – элегантный, интуитивно понятный и мощный front-end фреймворк для более быстрой и легкой веб-разработки [9].

Библиотека Bootstrap позволяет использовать относительно готовые компоненты, соответствующие всем правилам разработки пользовательских интерфейсов. Интерфейс, написанный с использованием Bootstrap, не будет сильно отличаться от интерфейса привычных сайтов (к примеру сайта «Госуслуги» или других государственных проектов), что позволит минимизировать ошибки при работе с интерфейсом за счёт его привычности и понятности пользователям.

Вторым важным инструментом при разработке интерфейса стало приложение для дизайнеров Figma – это кроссплатформенный графический онлайн-редактор для совместной работы. Программа позволяет создавать wireframe, UI, прототипы, презентации и с лёгкостью передавать материалы в разработку. В онлайн-режиме можно наблюдать рабочий процесс, оставлять комментарии и обсуждать макет.

В Figma был проработан «скелет» приложения – примерные расположения кнопок и объектов; общие концепты страниц и так далее.

### Цветовая гамма

К разработке цветовой гаммы приложения были особенные требования: цвета, используемые в проекте, должны были однозначно восприниматься пользователем; цветовая гамма не должна была «давить на глаза»; необходимы акцентные цвета для выделения важной информации.

С учётом требований, в качестве цветовой гаммы взяты следующие цвета (рисунок 17): главный цвет интерфейса белый (HEX: #ffffff) – классический цвет интерфейса, с которым прекрасно сочетаются все остальные цвета; синий (HEX: #0d6efd) – насыщенный и яркий оттенок синего, близкий к цвету "ярко-синий" или "электрик"; зелёный (HEX: #198754) – тёмно-зелёный с лёгким синеватым оттенком, близкий к изумрудному; чёрный (HEX: #212529) – светлый чёрный цвет, который аккуратно вписывается в цветовую гамму.

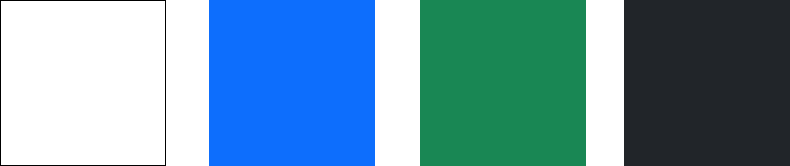


Рисунок – Цветовая гама приложения.

Каждый из цветов имеет своё место в интерфейсе: белый – ясный, понятный и визуально лёгкий цвет, который будет использоваться в качестве фона; синий – нейтральный цвет, используется в качестве основного цвета навигационных элементов; зелёный – психологически характеризуется как цвет «хорошо», в интерфейсах обозначает действия «вперёд», «отправить» и тому подобные.

### Базовый DTL-шаблон приложения

Базовый DTL-шаблон приложения отвечает за «обёртку» всех элементов вёрстки, которые будут описаны в следующих разделах. Сам шаблон представлен на рисунке 18 и использует Django template language синтаксис для превращения обычной HTML вёрстки в шаблон, который понимает Django.



Рисунок – DTL-шаблон приложения

Он включает в себя тег head, отвечающий за настройку сайта и подключение файлов, стилей и библиотек. В рамках проекта в этом шаблоне происходит подключение библиотеки Bootstrap, стилевого файла styles.css (отвечает за стилизацию проекта), а также иконки сайта, которая отображается в браузере. Синтаксис DTL выделяется на счёт уникального открытия и закрытия скобок {% контент %}. В шаблоне мы интегрируем вёрстку шапки с помощью тега include, а также размечаем место под весь контент, который будет вставлен через блочную структуру {% block название\_блока\_для\_вставки %}, в нашем случае блок имеет название content.

### Интерфейс главной страницы

Интерфейс главной страницы приложения (рисунок 19) представляет из себя страницу с описанием проекта.

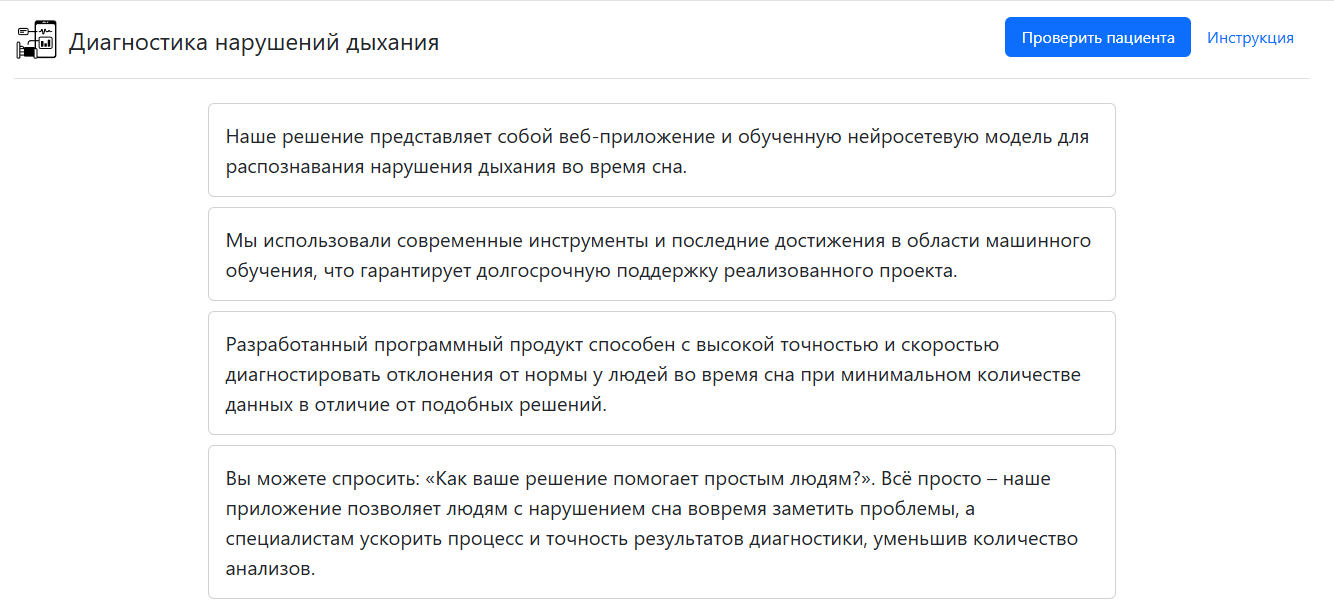


Рисунок . Интерфейс главной страницы.

Ключевым элементом этой (и других страниц) является шапка сайта. На ней представлен логотип программы и навигационные клавиши «Проверить пациента» и «Инструкция». Каждая из клавиш ведёт на отдельную страницу (они будут описаны в следующих разделах). Важно, что шапка сайта будет одинаковая на всех страницах, тем самым не запутывая пользователей.

Вёрстка главной страницы выполнена с использованием UI-компонентов библиотеки Bootstrap. Код карточек с информацией о проекте представлен на рисунке 20. Карточка представляет из себя два тега div, вложенных в друг друга. Родительский контейнер имеет класс card, который задаёт все основные стили для карточки, дочерний тег имеет класс card-body и обозначает тело контента этой карточки.

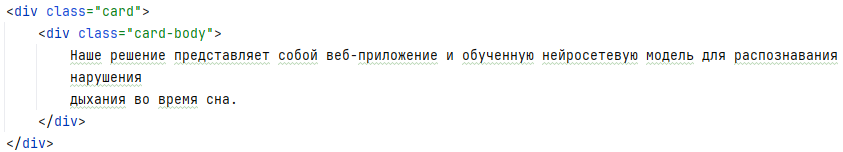


Рисунок – Вёрстка карточки

Также опишем вёрстку шапки сайта (рисунок 21). Она представляет из себя семантически верный тег header (он индексируется браузерами как шапка сайта с основной навигацией [10]) и сами навигационные ссылки, представленные тегами li внутри головного тега списка ul – ненумерованный список, в теге header индексируется как список навигационных ссылок.

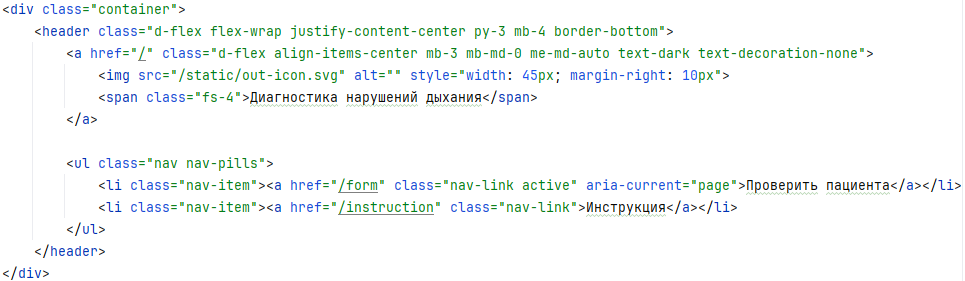


Рисунок – Вёрстка шапки

### Страница инструкций

Страница инструкций (рисунок 22) подразумевает размещение на ней инструкций по взаимодействую с веб-приложением. На странице представлена интерактивная инструкция по работе с диагностированием болезни у пациента.

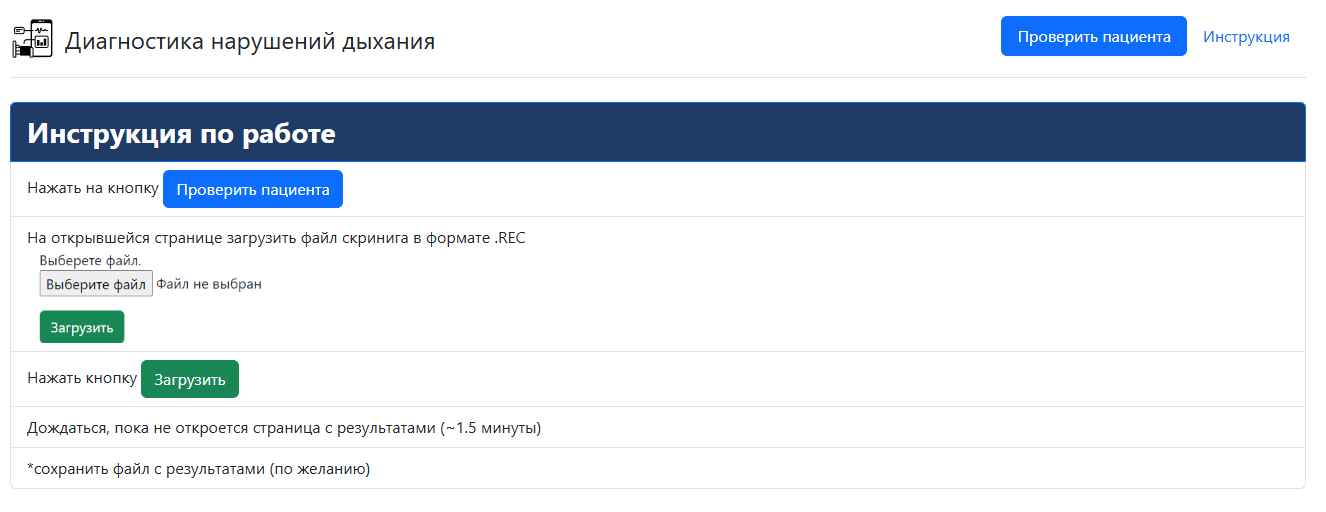


Рисунок – Интерфейс страницы инструкций

Также на рисунке 23 представлен HTML-код вёрстки пользовательского интерфейса.

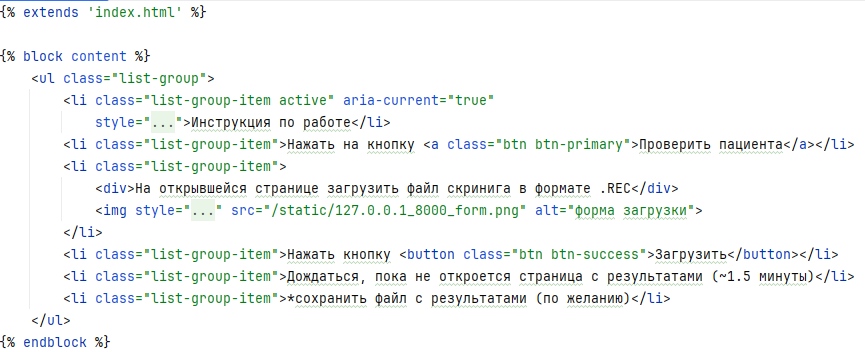


Рисунок – HTML-код пользовательского интерфейса

Стоит отметить использование DTL синтаксиса в этой вёрстке. Мы наследуем шаблон, который был описан ранее, с помощью команды {% extends путь\_к\_шаблону %}. После наследования шаблона мы указываем в каком блоке этого шаблона мы размещаем вёрстку (синтаксис совпадает с тем, как мы объявляли этот блок в шаблоне).

### Страница диагностики

Пользовательский интерфейс страницы отправки скрининга на диагностику представлен на рисунке 24 в состоянии до отправки скрининга. Интерфейс также располагает небольшой информационной справкой в верхней его части. Под информационной частью находится форма загрузки и отправки файла на проверку. Форма включает в себя поле для загрузки файла и кнопку «Проверить», при нажатии на которую происходит отправка данных на сервер, их обработка и выдача результата.

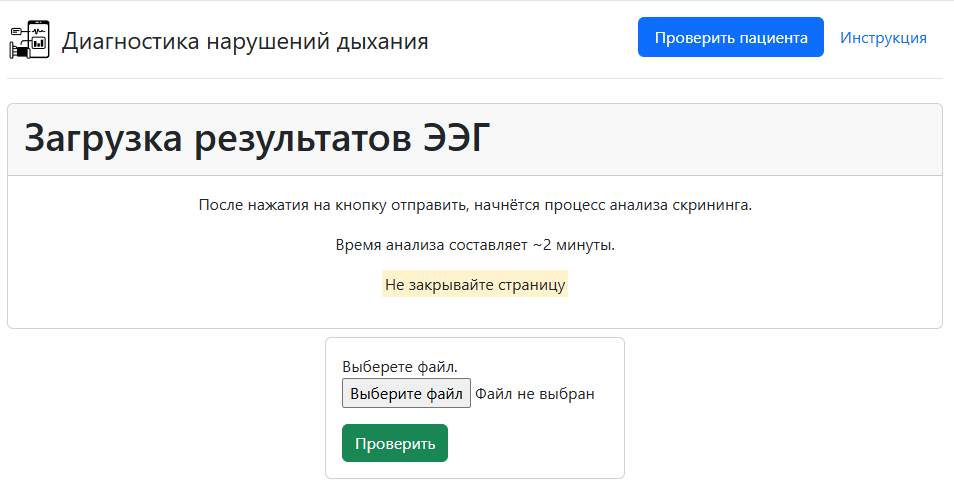


Рисунок – Страница загрузки скрининга. Состояние до отправки

Также на рисунке 25 представлено второе состояние формы – состояние после отправки. Изменения состояние формы – это общепринятое правило работы с различными формами. Оно показывает, что на сервере идёт работа и что приложение не зависло. Такое информационное сопровождение о смене состояния гарантирует спокойство пользователя.

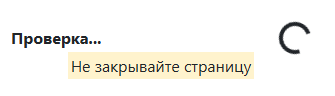


Рисунок – состояние «Проверки» формы

JavaScript-код переключения состояния изображен на рисунке 26 – в рамках кода происходит опрос страницы на отправку формы и как только форма отправлена, происходит переключения состояния формы в none (то есть форма исчезает со страницы), а у информационной панели состояния происходит включение за счёт изменения свойства display (перевод «отображение») на block.

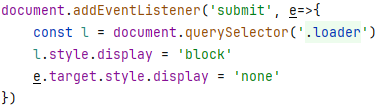


Рисунок – JavaScript-код

Отдельное внимание хотелось бы уделить вёрстке формы с DTL (рисунке 27). Внутри формы необходимо расположить DTL-тег {% csrf\_token %}, он защищает наше приложение от CSRF-атаки, добавляя в код HTML-тег <input> с токеном нашего сайта.

CSRF-атака – это атака, которая заставляет конечного пользователя выполнять нежелательные действия в веб-приложении, в котором он в данный момент аутентифицирован [11].

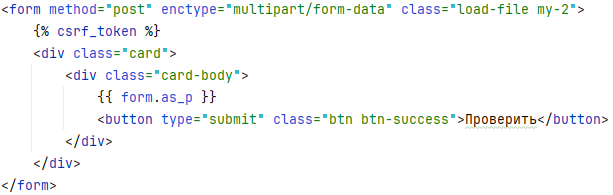


Рисунок – Вёрстка формы

Также внутри формы присутствует тег {{ form.as\_p }}. Он означает, что форму, которую мы передаём на рендер, нужно отобразить как список из HTML-тегов <p> с тегами <input> и <label> внутри.

### Страница результатов диагностики

Страница результатов выполнена в лаконичном стиле (рисунок 28 и рисунок 29). На странице расположена таблица с результатами исследований: название файла, итоговый диагноз, степень обнаружения апноэ (требуется в том случае, если врач сомневается в достоверности результатов исследований нейросети и хочет убедиться в степени и точности исследования). Также в исследованиях, где апноэ обнаружено (или «возможно обнаружено»), указывается степень заболевания.

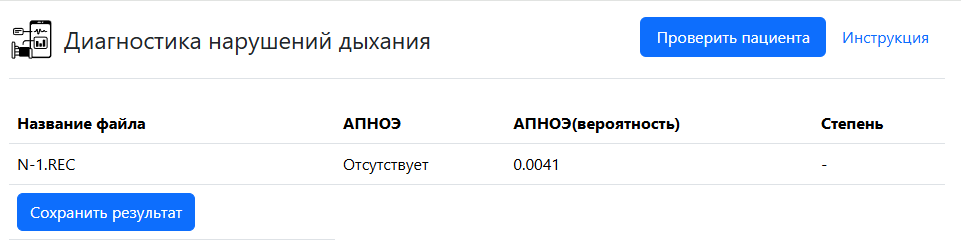


Рисунок – Пользовательский интерфейс страницы результатов. Результат исследования отрицательный

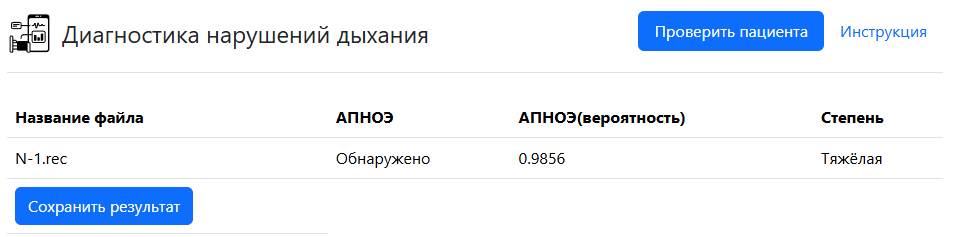


Рисунок – Интерфейс страницы результатов. Результат исследования «обнаружено»

## Функциональность веб-приложения

К функциональной составляющей веб-приложения для диагностики были предъявлены особенные критерии, такие как вывод не только обнаружено заболевание апноэ или нет, но также его степень и вероятность (точность), что выявленный диагноз апноэ.

Обработка результатов загруженного файла происходит в несколько этапов:

1. получение данных из POST-запроса на сервер;
2. обработка данных формы и проверка на их валидность;
3. получение результатов оценки нейросети;
4. интерпретация результатов в вид, понятный для врача;
5. рендер результатов.

Теперь рассмотрим каждый из этапов подробнее. Получение данных из формы происходит с помощью команды UploadForm(request.POST, request.FILES) – это класса, наследуемый от встроенного в Django класса для обработки форм Form. Внутри класса UploadForm определяются поля, которые нужно отслеживать в форме. После получения экземпляра класса мы проверяем, какой тип запроса поступил на сервер: если это POST-запрос, то производим обработку файла; если GET-запрос, то просто рендерим форму для загрузки.

Валидация данных происходит с помощью метода is\_valid() в классе UpdateForm. Метод проверяет соответствие отправленных в форме данных тем, которые сама форма ожидает. Возвращает булевый тип данных.

В случае прохождения успешной валидации происходит третья стадия – получение результата работы нейросети. Третья стадия выполняется функцией do\_result, код функции представлен на рисунке 30.



Рисунок – Код функции do\_result

В функции мы объявляем два экземпляра класса Network (он был описан в главе 0) для правой и левой затылочной зоны ЭЭГ. Далее происходит преобразование файла из EDF в ASCII (функция также была описана ранее), после преобразования данный к массиву 512 на 512, мы передаём данные в функцию predict экземпляра нейронной сети. Функция predict отвечает за получение предсказания (или же результатов диагностирования) нейронной сети по текущему ЭЭГ.

В качестве результата функции возвращается массив следующего вида [net1\_response, net2\_response, [apnoe\_index1, apnoe\_index2]], где net1\_response и net2\_response – это результат работы сети. Индексы apnoe\_index отвечают за подсчёт эпизодов апноэ (остановок дыхания) у пациента, в дальнейшем эта клиническая метрика используется для получения тяжести апноэ.

После получения результатов работы нейронной сети происходит процесс интерпретация полученных данных в привычный для врача вид (рисунок 31).

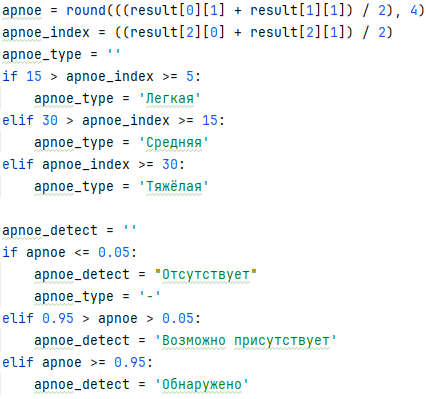


Рисунок – Интерпретация результатов

В результате интерпретации получается 3 значения:

1. статус обнаружения апноэ;
2. вероятность обнаружения;
3. степень апноэ (при обнаружении).

Расчёт вероятности обнаружения апноэ берётся как среднее значение от двух каналов – такой способ обеспечивает высокую точность получения результатов, уменьшая погрешность преобладания результатов одной затылочной зоны над другой, так как в случае заболевания апноэ оно примерно в равной степени отображается в двух зонах.

Также отмечу, что формула степени апноэ взята из стандартов по обследованию апноэ [12] и считается как количество остановок дыхания.

После проведения интерпретации, данные передаются в контекст функции рендера (рисунок 32). Контекст функции рендера – это объект, который хранит в себе переменные и данные, доступные для обработки DTL в момент формирования страницы HTML.

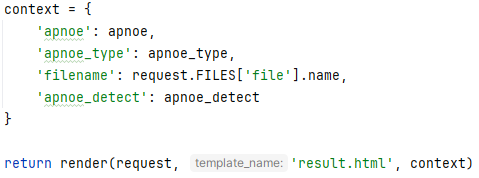


Рисунок – Контекст вызова функции рендера

Сама функция рендера представляет из себя обёртку над встроенными функциями Django, позволяющая без лишних настроек отдавать HTTP ответ в виде HTML страницы. В аргументах функции обязательно передаётся request текущего вызова, а также путь до шаблона HTML и контекст для рендера.

### Формирования отчёта исследования в PDF.

Для обеспечения архивации результатов диагностики была реализована вспомогательная функция save\_result, предназначенная для формирования отчёта исследования в формате PDF.

PDF – это электронный формат документов, разработанный компаний Adobe System и предназначенный для надёжного представления данных вне зависимости от системы, на которой они просматриваются.

Функция вызывается с помощью HTTP-запроса с методом GET и параметрами исследования, полученного ранее. Она возвращает ответ в виде файл PDF с флагом as\_attachment=True, что инициирует загрузку на компьютер врача.

Для построения отчёта (рисунок 33) используется библиотека ReportLab – это одна из самых популярных и надёжных библиотек для создания PDF-документов в Python. Библиотека позволяет не только вставлять текст, таблицы и картинки, но и изменять стили.

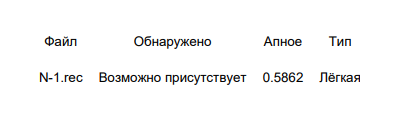


Рисунок – PDF-отчёт

Предлагаю разобрать стадии формирования PDF-документа.

Вначале мы производим создание и первичную настройку будущего PDF документа (рисунок 34). В ходе настройки мы задаём имя файла, формируем документ с помощью класса SimpleDocTemplate, в который передаём временный буфер для работы с данными и формат страницы. Также здесь подключаются основные стили с помощью функции getSampleStyleSheet().

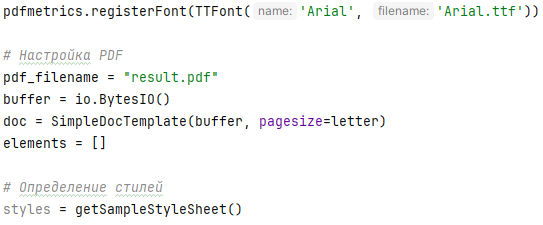


Рисунок – Настройка документа

Далее данные приводятся к виду, необходимому для создания таблицы с результатом исследования (рисунок 35)

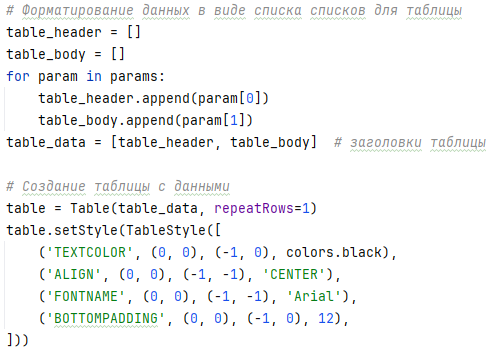


Рисунок – Создание таблицы с результатом исследования

После того, как таблица будет сформирована, она передаётся в массив elements, объявленный нами в момент настройки, далее этот массив передаётся как аргумент в функции build для созданного нами документа. После формирования документа временный буфер обнуляется, чтобы не засорять память сервера, и сформированный файл отдаётся как результат работы функции.

# Заключение

В выпускной квалификационной работе разработана система диагностики синдрома обструктивного апноэ сна (СОАС) на основе анализа электроэнцефалограмм (ЭЭГ) с использованием методов искусственного интеллекта. Основные результаты исследования включают:

* применение современных технологий – Система реализована с использованием языка программирования Python, фреймворков Django и TensorFlow, что позволило создать удобное и эффективное приложение для анализа медицинских данных;
* архитектура нейронной сети – Использована сверточная нейронная сеть, которая продемонстрировала точность более 85% при диагностике апноэ на основании двух затылочных каналов ЭЭГ. Это подтверждает потенциал глубокого обучения в медицинской диагностике;
* доступность и масштабируемость – Приложение спроектировано таким образом, чтобы его можно было использовать как в облачных средах, так и на локальных серверах медицинских учреждений. Это обеспечивает гибкость и удобство внедрения;
* тестирование и валидация – Проведено тестирование всех модулей системы, включая ручное и автоматическое, что подтверждает надежность приложения в различных сценариях использования.

Таким образом, разработанная система обладает высоким потенциалом для упрощения и автоматизации процесса диагностики СОАС, что может значительно снизить затраты времени и ресурсов медицинских учреждений, а также улучшить качество жизни пациентов.

# Список использованных источников

1. Acharya U. R., Oh S. L., Hagiwara Y., Tan J. H., Adam M., Gertych A. и др. A deep convolutional neural network model to classify heartbeats from single lead electrocardiograms [Электронный ресурс]. – Режим доступа: https://pubmed.ncbi.nlm.nih.gov/28869899/ (дата обращения: 10.11.2024)
2. Mirowski P. W., LeCun Y., Madhavan D., Kuzniecky R., DeVivo D. C. Comparing SVM and Convolutional Networks for Epileptic Seizure Prediction from Intracranial EEG [Электронный ресурс]. – Режим доступа: https://ieeexplore.ieee.org/document/4685487 (дата обращения: 05.03.2023)
3. Zhixi Li, Yifan He, Stuart Keel, Wei Meng, Robert T. Chang, Mingguang He. Efficacy of a Deep Learning System for Detecting Glaucomatous Optic Neuropathy Based on Color Fundus Photographs [Электронный ресурс]. – Режим доступа: https://pubmed.ncbi.nlm.nih.gov/29506863/ (дата обращения: 16.03.2025)
4. Jun Chen, Lianlian Wu, Jun Zhang, Liang Zhang, Dexin Gong, Yilin Zhao. Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography: a prospective study [Электронный ресурс]. Режим доступа: https://www.medrxiv.org/content/10.1101/2020.02.25.20021568v2 (дата обращения: 16.03.2025)
5. Хакатон «Цифровой прорыв: сезон ИИ» – кейс от «Саратовский государственный медицинский университет имени В. И. Разумовского» [Электронный ресурс]. Режим доступа: https://lk.hacks-ai.ru/event/1077376/case (дата обращения 17.03.2025)
6. Django Project – Django documentation [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/fr/4.2/topics/templates/ (дата обращения: 20.04.2025)
7. SQLite – документация SQLite [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/ (дата обращения: 20.04.2025)
8. Django Fun – Django документация [Электронный ресурс]. – Режим доступа: https://django.fun/docs/django/5.2/topics/http/urls/ (дата обращения: 21.04.2025)
9. Github. Bootstrap – официальный репозиторий [Электронный ресурс]. – Режим доступа: https://github.com/twbs/bootstrap (дата обращения: 21.04.2025)
10. MDN – документация HTML [Электронный ресурс]. Режим доступа: https://developer.mozilla.org/enUS/docs/Web/HTML/Reference/Elements/header (дата обращения: 21.04.2025)
11. OWASP – описание CSRF-атаки [Электронный ресурс]. Режим доступа: https://owasp.org/www-community/attacks/csrf (дата обращения: 21.04.2025)
12. AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications. Версия 2.6. – Дарьен (Иллинойс): American Academy of Sleep Medicine, 2020. – 301 с. – С. 30.

# Приложение А

(обязательное)

Отчёт о заимствовании.

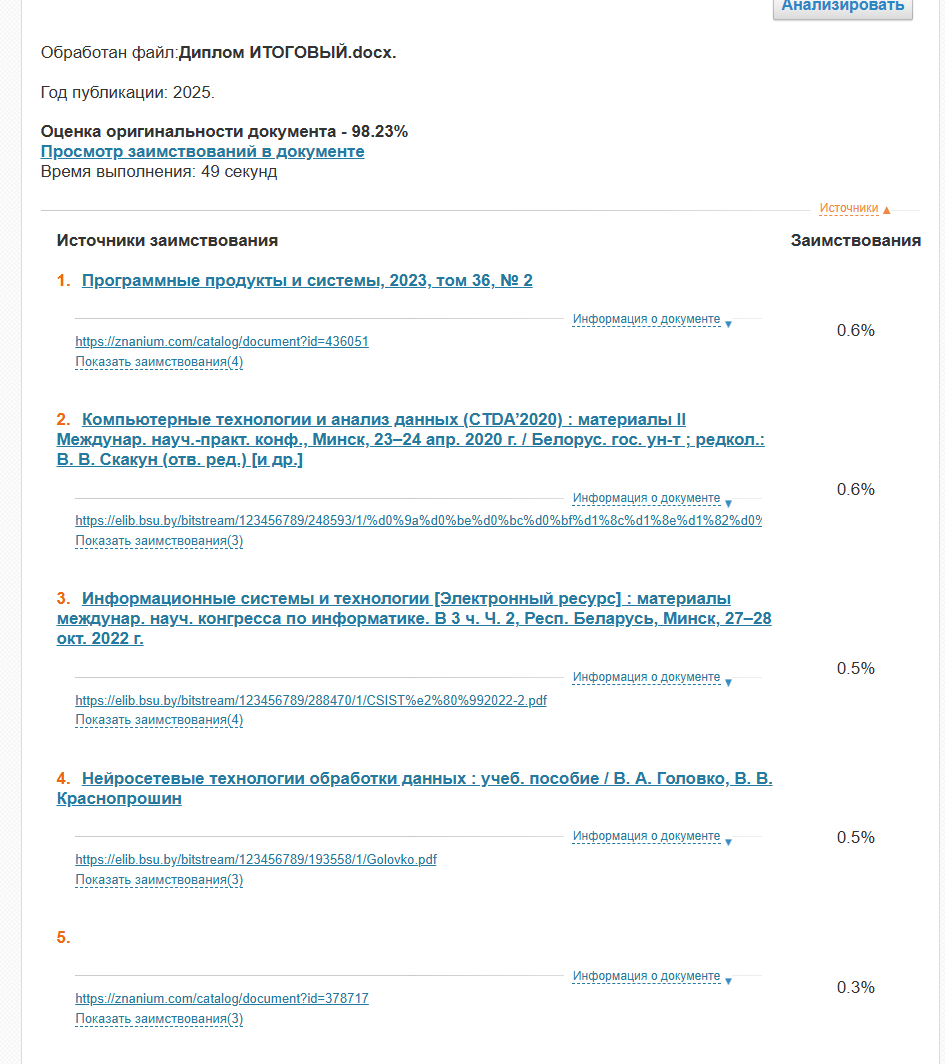


Рисунок А.1 – Отчёт о заимствовании