МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра информационных технологий

КУРСОВАЯ РАБОТА

ВЕБ-РЕСУРС КЛАСТЕРИЗАЦИИ СИМПТОМОВ ЗАБОЛЕВАНИЯ

Работу выполнил _____ Б.А. Михаенко

(подпись)

Направление подготовки 09.03.03 Прикладная информатика

Направленность Прикладная информатика в экономике

Научный руководитель
ст. преп. _____ А.В. Уварова

(подпись)

Нормоконтролер
канд. пед. наук, доц. _____ А.В. Харченко

(подпись)

Краснодар
2022

РЕФЕРАТ

Курсовая работа 25 с., 11 рис., 4 таб., 7 источников.

КЛАСТЕРНЫЙ АНАЛИЗ, МЕТРИКИ, ЕВКЛИДОВА МЕТРИКА, БАЗЫ ДАННЫХ, POSTGRESQL, ВЕБ-ПРИЛОЖЕНИЕ.

Объектом исследования являются методы кластерного анализа и принципов создания собственного веб-приложения.

Цель работы: изучение основных метрик для решения задачи кластерного анализа и принципов создания собственного веб-приложения.

В курсовой работе рассмотрены основы кластерного анализа, метрики и методы разработки веб-приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время доступность сети Интернет привела к увеличению не только количества информации, но и ее сложности. Каждый сайт собирает огромное количество данных, чтобы потом предложить пользователю более выгодные услуги или товары. Но для выбора предложений их необходимо сравнивать между собой, чтобы найти максимально подходящее пользователю.

Сейчас задача метрик данных актуальна, как никогда. Количество информации и ее сложность растет, а вместе с тем изменяются предпочтения пользователей.

Метрика – это функция определения расстояния между двумя точками или определение меры угла в той или иной геометрической системе (евклидовой и не евклидовых – Лобачевского, Римана и др.). Полученные от пользователя данные можно представлять в виде точек, находящихся в n-мерном пространстве, где n-общее количество собранных данных.

Предложенная курсовая работа посвящена изучению разных метрик и способов сравнения данных, изучению современных технологий разработки в области Web, а также разработке собственного сайта для диагностирования возможных болезней по предварительно выбранным симптомам.

Первая глава курсовой работы содержит описание основных метрик и их отличий.

Вторая глава посвящена разработке сайта и его основным составляющим.

Третья глава содержит описание взаимодействия пользователя с сайтом.

1 Основы кластерного анализа

Кластерный анализ – это метод классификационного анализа; его основное назначение – разбиение множества исследуемых объектов и признаков на однородные в некотором смысле группы, или кластеры. Это многомерный статистический метод, поэтому предполагается, что исходные данные могут быть значительного объема, т. е. существенно большим может быть как количество объектов исследования (наблюдений), так и признаков, характеризующих эти объекты [1].

В прикладной статистике многомерными статистическими методами долго не могли пользоваться из-за отсутствия вычислительной техники для обработки больших массивов данных. Активно эти методы стали развиваться со второй половины XX в. при появлении быстродействующих компьютеров, выполняющих за доли секунды необходимые вычисления, на которые до этого уходили дни, недели, месяцы. В настоящее время препятствием к широкому использованию многомерных статистических методов, в том числе и кластерного анализа, является отсутствие у исследователей навыков и умения работать со статистическими пакетами прикладных программ.

Техника кластеризации может применяться в самых различных прикладных областях, в том числе и в медицине. Например, кластеризация заболеваний, симптомов, признаков заболеваний, методов лечения может привести к более полному и глубокому пониманию медицинских проблем, связанных с лечением больных.

Большое достоинство кластерного анализа в том, что он дает возможность производить разбиение объектов не по одному признаку, а по ряду признаков. Кроме того, кластерный анализ в отличие от большинства математико-статистических методов не накладывает никаких ограничений на вид рассматриваемых объектов и позволяет исследовать множество исходных данных практически произвольной природы.

Обычно исходные данные представляют в виде матрицы измеренных значений признаков для рассматриваемых объектов (рисунок 1.1).

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

Рисунок 1.1 – Матрица признаков рассматриваемых объектов

В то же время большинство алгоритмов кластерного анализа основывается на исследовании матрицы расстояний. Поэтому первым этапом решения задачи кластеризации является выбор способа вычисления расстояний между объектами или признаками [1].

Выбор метрики или меры близости, является нетривиальным и одним из основных моментов исследования, от которого в значительной степени зависит окончательный вариант разбиения объектов на классы при данном алгоритме разбиения.

Рассмотрим наиболее часто используемые расстояния и меры близости в задачах кластерного анализа.

1.1 Основные метрики

Евклидова метрика - наиболее популярная метрика, является геометрическим расстоянием в многомерном евклидовом пространстве. Данная метрика, как и большинство других, чувствительна к изменению единиц измерения осей. Например, если сантиметры перевести в миллиметры, то изменится и исчисляемое расстояние. Поэтому при использовании большинства метрик кластерный анализ предполагает предварительную стандартизацию исходных данных [2].

Для точек $p = (p_1, \dots, p_n)$ и $q = (q_1, \dots, q_n)$ евклидово расстояние определяется как показано в формуле (1).

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2} \quad (1)$$

Евклидова метрика – наиболее естественная функция расстояния, возникающая в геометрии, отражающая интуитивные свойства расстояния между точками. При этом существуют и другие метрики в евклидовых пространствах, применяемые как в геометрии, так и в приложениях. Параметрическое расстояние Минковского является обобщением некоторых из этих метрик, при параметре со значением 2 оно обращается в евклидову метрику [2].

Евклидово пространство – конечномерное векторное пространство над полем вещественных чисел, на парах векторов которого задана вещественнозначная функция, обладающая следующими тремя свойствами:

1) Линейность для любых векторов u, v, w и для любых вещественных чисел a, b справедливы соотношения $(au + bv, w) = a(u, w) + b(v, w)$;

2) Симметричность: для любых векторов u, v верно равенство $(u, v) = (v, u)$

3) Положительная определенность: $(u, u) \geq 0$ для любого u, причем $(u, u) = 0 \rightarrow u = 0$

Также существует Квадрат Евклидового расстояния. Его используют, если необходимо придать большие веса более отдаленным друг от друга объектам формула (2).

$$(x, y) = \sum_{i=1}^{n}(x_i - y_i)^2 \quad (2)$$

Расстояние Чебышёва – метрика на векторном пространстве, задаваемая как максимум модуля разности компонент векторов. Расстояние Чебышева

применяют, когда желают определить два объекта как различные, если они различаются по какой-либо одной координате формула (3) [2].

$$l_\infty(\vec{x}, \vec{y}) = \max_{i=1,\dots,n} |x_i - y_i| \quad (3)$$

Расстояние Минковского (метрика) – параметрическая метрика на евклидовом пространстве, которую можно рассматривать как обобщение евклидова расстояния и расстояния городских кварталов. Расстояние Минковского используют, когда надо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Данная метрика порядка p между двумя точками $x, y \in R^n$ определяется как показано в формуле (4).

$$p(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \quad (4)$$

Манхэттенское Расстояние – метрика, введенная Германом Минковским. Данная метрика уменьшает влияние отдельных больших разностей между одноименными координатами точек, так как при вычислении расстояния эти разности не возводятся в квадрат (в отличие от евклидовой метрики).

Согласно этой метрике, расстояние между двумя точками равно сумме модулей разностей их координат. Более формально это показано в формуле (5).

$$d_1(p, q) = ||p - q||_1 = \sum_{i=1}^{n} |p_i - q_i| \quad (5)$$

где $p = (p_1, p_2, \dots, p_n)$ и $q = (q_1, q_2, \dots, q_n)$ – векторы.

2. Описание разработки сайта и его основных составляющих

В рамках курсовой работы ставится задача разработки веб-приложения для установки диагноза болезни по выбранным симптомам/ основными функциями которого являются:

– выбор категории симптомов,

– выбор подходящих симптомов пользователем из определенной категории,

– установка диагноза по предварительно выбранным симптомам,

– возможность просмотра описания и советов по лечению наиболее вероятных болезней.

2.1 Описание реализации базы данных

Разработанное приложение состоит из Базы Данных, клиентской и серверной части.

База данных была разработана с помощью СУБД PostgreSQL [3].

Она состоит из 4 таблиц (рисунок 2.1).

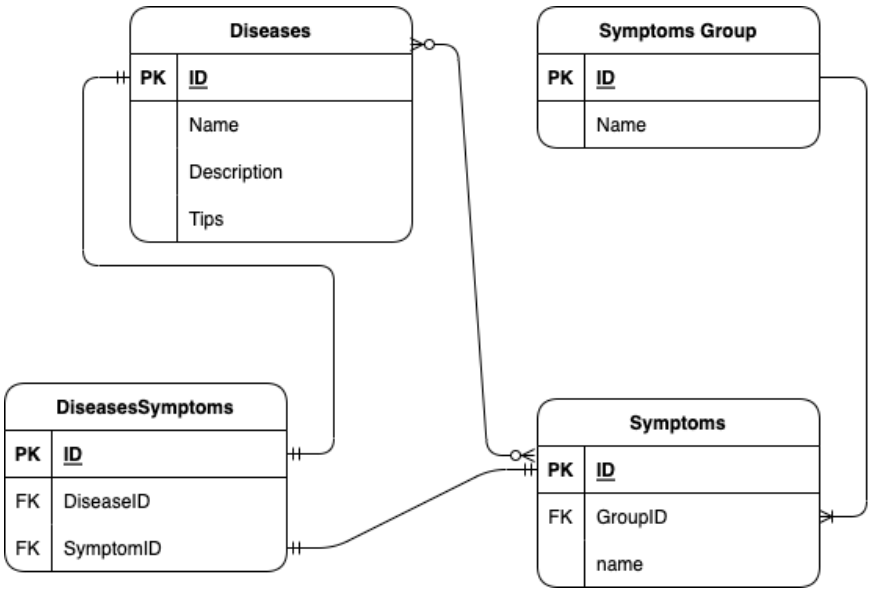

Рисунок 2.1 – Схема структуры базы данных

Таблица Diseases предназначена для хранения информации о существующих заболеваниях их наименование, описание, симптоматику и некоторые советы по лечению. Её структура приведена в таблице 1. Данная таблица имеет связь Many-to-Many с таблицей Symptoms через промежуточную таблицу DiseasesSymptoms.

Таблица 1 - Структура таблицы Diseases

Наименование	Тип	Описание
ID	Primary Key	Уникальный идентификатор заболевания
Name	Unique String	Уникальное наименование заболевания
Description	Text	Описание заболевания
Tips	Text	Строка советов по лечению заболевания

Таблица Symptoms предназначена для хранения информации о симптомах заболеваний их наименование, а также уникальный номер группы, к которой они принадлежат. Её структура приведена в таблице 2. Данная таблица имеет связь Many-to-Many с таблицей Diseases через промежуточную таблицу DiseasesSymptoms, а также связь Many-to-One с таблицей SymptomsGroup, через внешний ключ GroupID

Таблица 2 - Структура таблицы Symptoms

Наименование	Тип	Описание
ID	Primary Key	Уникальный идентификатор симптома
GroupID	Foreign Key	Уникальный идентификатор группы симптомов
Name	Unique String	Наименование симптома

Таблица SymptomsGroup предназначена для хранения информации о группах симптомов. Её структура представлена в таблице 3. Данная таблица имеет связь One-to-Many с таблицей Symptoms устанавливая свой уникальный идентификатор группы, как внешний ключ для таблицы Symptoms.

Таблица 3 - Структура таблицы SymptomsGroup

Наименование	Тип	Описание
ID	Primary Key	Уникальный идентификатор группы
Name	Unique String	Уникальное наименование группы

Таблица DiseaseSymptoms представляет собой промежуточную таблицу, предназначенную для хранения пары ключей вида: уникальный идентификатор заболевания - уникальный идентификатор симптома. Её структура представлена в таблице 4.

Таблица 4 - Структура таблицы DiseasesSymtptoms

Наименование	Тип	Описание
ID	Primary Key	Уникальный идентификатор пары
DiseaseID	Foreign Key	Уникальный внешний идентификатор заболевания
SymptomID	Foreign Key	Уникальный внешний идентификатор симптома

2.2 Описание реализации серверной части веб-приложения

Для возможности взаимодействовать с базой данных необходим сервер, который будет предоставлять доступ к возможностям добавления, изменения,

удаления и поиска данных в созданной базе. Схема взаимодействия между сервером и базой данных (рисунок 2.2).

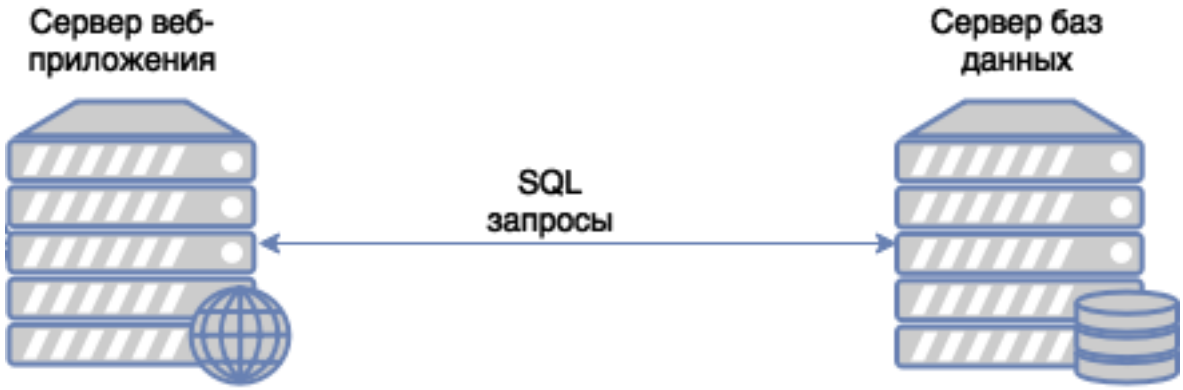

Рисунок 2.2 – Схема взаимодействия веб-приложения, сервера и БД

Для реализации серверной части веб-приложения использовались следующие технологии:

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript.

Обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управления памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Node.js – программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера [4].

Express.js – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений [6].

Имея в своем распоряжении множество служебных методов HTTP и промежуточных обработчиков, создать надежный API можно быстро и легко. Express предоставляет тонкий слой фундаментальных функций веб-приложений, которые не мешают вам работать с давно знакомыми и любимыми вами функциями Node.js.

Axios – это HTTP-клиент, основанный на Promise для node.js и браузера. Он *изоморфный* (он может работать в браузере и node.js с той же базой кодов). На стороне сервера он использует нативный node.js http-модуль, тогда как на стороне клиента (браузер) он использует XMLHttpRequests.

Sequelize – это ORM-библиотека для приложений на Node.js, которая осуществляет сопоставление таблиц в БД и отношений между ними с классами. При использовании. Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами. Причем Sequelize может работать с рядом СУБД - MySQL, Postgres, MariaDB, SQLite, MS SQL Server [5].

PostgreSQL – это мощная объектно-реляционная система баз данных с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании со многими функциями, позволяющими безопасно хранить и масштабировать самые сложные рабочие нагрузки с данными.

Истоки PostgreSQL восходят к 1986 году в рамках проекта POSTGRES в Калифорнийском университете в Беркли и насчитывает более 35 лет активной разработки на базовой платформе.

Далее представлен разбор основного скрипта для инициализации веб-сервера и подключения к базе данных.

```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const sequelize = require('./db')
const models = require('./models/models')
const router = require('./routes/route');
const ErrorHandlingMiddleware =
require('./middlewares/ErrorHandlingMiddleware');

const app = express()
app.use(cors())
app.use(express.json())
app.use('/api', router)
app.use(ErrorHandlingMiddleware)


const PORT = process.env.PORT || 5000;

const start = async () => {
     try {
          await sequelize.authenticate()
          await sequelize.sync()
          app.listen(PORT, () => console.log(`Успешно подклюено,
          порт:${PORT}`))
     } catch (e) {
          console.log('Не удалось подключиться к серверу')
     }
}
start()
```

Рассмотрим основные функции:

`require('dotenv').config()` – импортирование и настройка библиотеки для работы с переменными окружения,

`const express = require('express')` – импорт библиотеки express для работы с сервером и ее инициализация.

`const cors = require('cors')` – импорт библиотеки cors для отправки cross-origin запросов и ее инициализации.

`const sequelize = require('sequelize')` – импорт библиотеки sequelize для работы с моделями базы данных и ее инициализация,

`const models = require('./models/models')` – импорт моделей таблиц базы данных и их инициализация,

`const router = require('./routes/route')` – импорт ключевого элемента маршрутизации,

`constErrorHandlingMiddleware = require('./middlewares/ErrorHandlingMiddleware)` – импорт middleware перехватывающего ошибку во время ответа сервера,

`const app = express()` – создание нового приложения express,

`app.use(cors())` – установка метода cors() как middleware для создания cross-origin запросов,

`app.use(express.json())` – установка метода json() как middleware для представления полученных данных в виде JSON объекта

`app.use('/api', router)` – установка ключевого элемента маршрутизации как middleware для перехвата отдельных маршрутов.

`app.use(ErrorHandlingMiddleware)` – установка middleware перехватывающего ошибки при ответе сервера.

Middleware (или функции промежуточной обработки) – это функции, имеющие доступ к объекту запроса, объекту ответа и к следующей функции промежуточной обработки в цикле "запрос-ответ" приложения.

`const PORT = process.env.PORT || 5000;` – объявление и инициализация переменной PORT с помощью переменной окружения `process.env.PORT` или, в случае отсутствия переменной окружения, портом 5000 по умолчанию.

```
const start = async () => {
    try {
        await sequelize.authenticate()
        await sequelize.sync()
        app.listen(PORT, () => console.log(`Успешно подклюено,
порт:${PORT}`))
    } catch (e) {
        console.log('Не удалось подключиться к серверу')
    }
}
```

`start()` – объявление асинхронной функции запуска сервера.

`await sequelize.authenticate()` – вызов метода аутентификации БД из библиотеки Sequelize.

`await sequelize.sync()` – вызов метода синхронизации с БД из библиотеки Sequelize.

`app.listen(PORT, () => console.log(`Успешно подклюено, порт:${PORT}`))` – привязка и прослушивание соединения на указанном хосте и порту и вывод сообщения о успешном подключении.

```
catch (e) {
        console.log('Не удалось подключиться к серверу')
    }
```
– перехват ошибок во время запуска и работы сервера и вывод сообщения о конкретной ошибке.

`start()` – вызов асинхронной функции запуска сервера.

2.3 Описание реализации клиентской части веб-приложения

Для реализации клиентской части веб-приложения использовались следующие технологии:

React – это библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через

инкапсулированные единицы (называются компонентами), которые сохраняют состояние и генерируют элементы пользовательского интерфейса.

React-Router-DOM – это модуль узла, который предназначен для маршрутизации в веб-приложениях. Он позволяет инженерам создавать маршруты для одностраничного приложения React.

Redux – библиотека для JavaScript с открытым исходным кодом, предназначенная для управления состоянием приложения. Чаще всего используется в связке с React или Angular для разработки клиентской части. Содержит ряд инструментов, позволяющих значительно упростить передачу данных хранилища через контекст.

RTK Query (или Redux Toolkit Query) – это мощный инструмент выборки и кэширования данных. Он предназначен для упрощения распространенных случаев загрузки данных в веб-приложение, устраняя необходимость самостоятельного написания логики выборки и кэширования данных [7].

TypeScript – язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript является обратно совместимым с JavaScript и компилируется в последний. Фактически, после компиляции программу на TypeScript можно выполнять в любом современном браузере или использовать совместно с серверной платформой Node.js. Код экспериментального компилятора, транслирующего TypeScript в JavaScript, распространяется под лицензией Apache. Его разработка ведётся в публичном репозитории через сервис GitHub

Sass (или Syntactically Awesome Stylesheets) – это скриптовый метаязык (т.е. язык, описывающий другой язык), разработанный для упрощения файлов CSS. Этот модуль входи в Haml (HTML abstraction markup language), который используется для упрощения HTML.

Схема взаимодействия между веб-приложением, сервером и БД (рисунок 2.3).

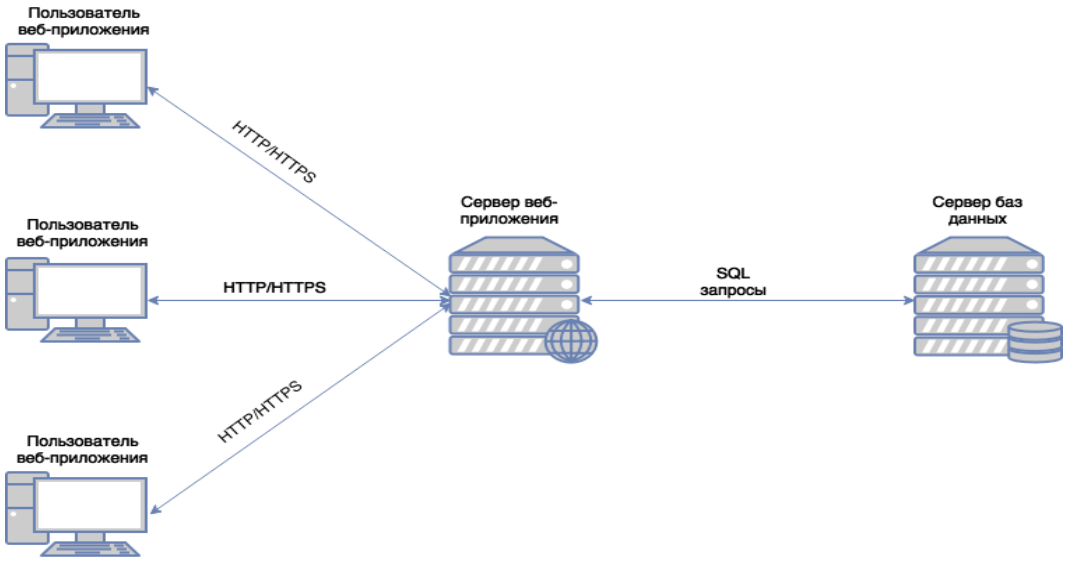

Рисунок 2.3 – Взаимодействие между веб-приложением, сервером и БД

Для отправки запросов с клиентской части использовался RTK Query [7].

Рассмотрим пример некоторых запросов на сервер:

```
export const symptomsApi = createApi({
    reducerPath: 'symptoms',
    baseQuery: fetchBaseQuery({ baseUrl:
`${process.env.REACT_APP_HOST_LINK}` }),
    tagTypes: ['SYMPTOM', 'GROUP', 'DISEASE'],
    endpoints: (builder) => ({
        getSymptoms: builder.query<Symptom[], void>({
            query: () => '/symptoms',
            providesTags: ['SYMPTOM']
        }),
        addSymptom: builder.mutation<Symptom, SymptomPreview>({
            query: (symptom) => ({
                url: '/symptoms',
                method: 'POST',
                body: symptom,
            }),
            invalidatesTags: ['SYMPTOM']})
```

`export const symptomsApi = createApi` – инициализация и экспорт переменной

`reducerPath: 'symptoms'` – название редуктора

`baseQuery: fetchBaseQuery({ baseUrl: `${process.env.REACT_APP_HOST_LINK}` })` – инициализация основного пути для создания запроса

`tagTypes: ['SYMPTOM', 'GROUP', 'DISEASE']` – тэги для отслеживания изменений

```
endpoints: (builder) => ({
        getSymptoms: builder.query<Symptom[], void>({
            query: () => '/symptoms',
            providesTags: ['SYMPTOM']
        }),
        addSymptom: builder.mutation<Symptom, SymptomPreview>({
            query: (symptom) => ({
                url: '/symptoms',
                method: 'POST',
                body: symptom,
            }),
            invalidatesTags: ['SYMPTOM']}) – создание конкретных
```

типов запросов используя объект builder и его встроенную функцию query.

3. Описание внешнего вида приложения

При запуске приложения, пользователя встречает страница запуска диагностирования (рисунок 3.1).

Рисунок 3.1 – Стартовая страница приложения

При нажатии на кнопку старта пользователя перенаправит на страницу выбора симптомов (рисунок 3.2).

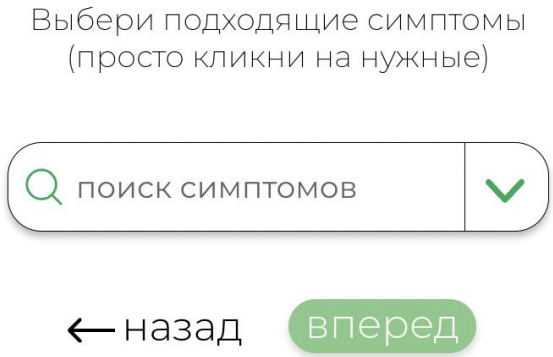

Рисунок 3.2 – Страница выбора симптомов

При нажатии на зеленую галочку откроется выпадающий список с существующими в базе данных симптомами, разделенными на конкретные группы (рисунок 3.3).

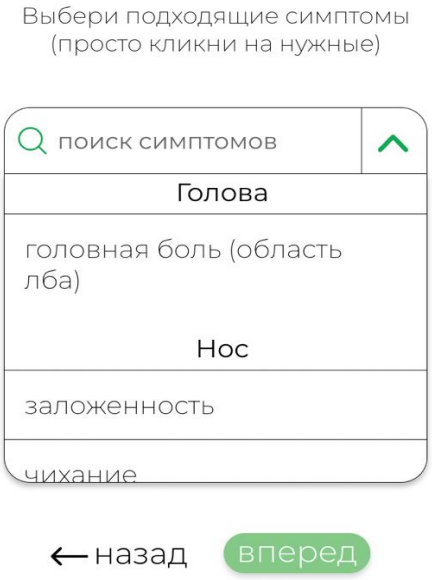

Рисунок 3.3 – Список существующих в БД симптомов

Для того чтобы выбрать нужный симптом необходимо нажать на него, тогда его цвет изменится на зеленый, а рядом появится индикатор того, что он выбран. Для получения диагноза необходимо выбрать не менее 3-х симптомов (рисунок 3.4).

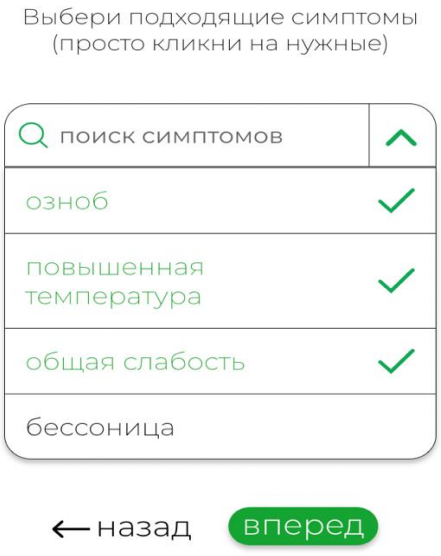

Рисунок 3.4 – Выбранные симптомы

Также для пользователя доступна возможность поиска конкретных симптомов, используя строку ввода данных (рисунок 3.5).

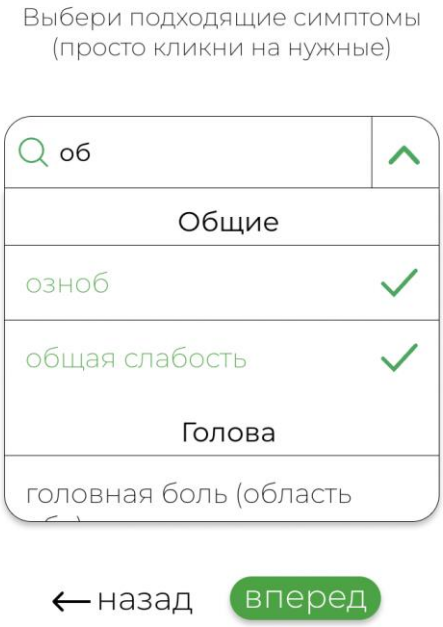

Рисунок 3.5 – Поиск симптомов

После выбора симптомов пользователь может узнать вероятный диагноз нажав на кнопку вперед. Страница с диагнозом представляет из себя список из наиболее вероятных болезней (рисунок 3.6).

Рисунок 3.6 – Страница вероятностных диагнозов

Для каждого из представленных диагнозов есть возможность узнать его название, описание, а также некоторые советы которых стоит придерживаться во время болезни (рисунок 3.7).

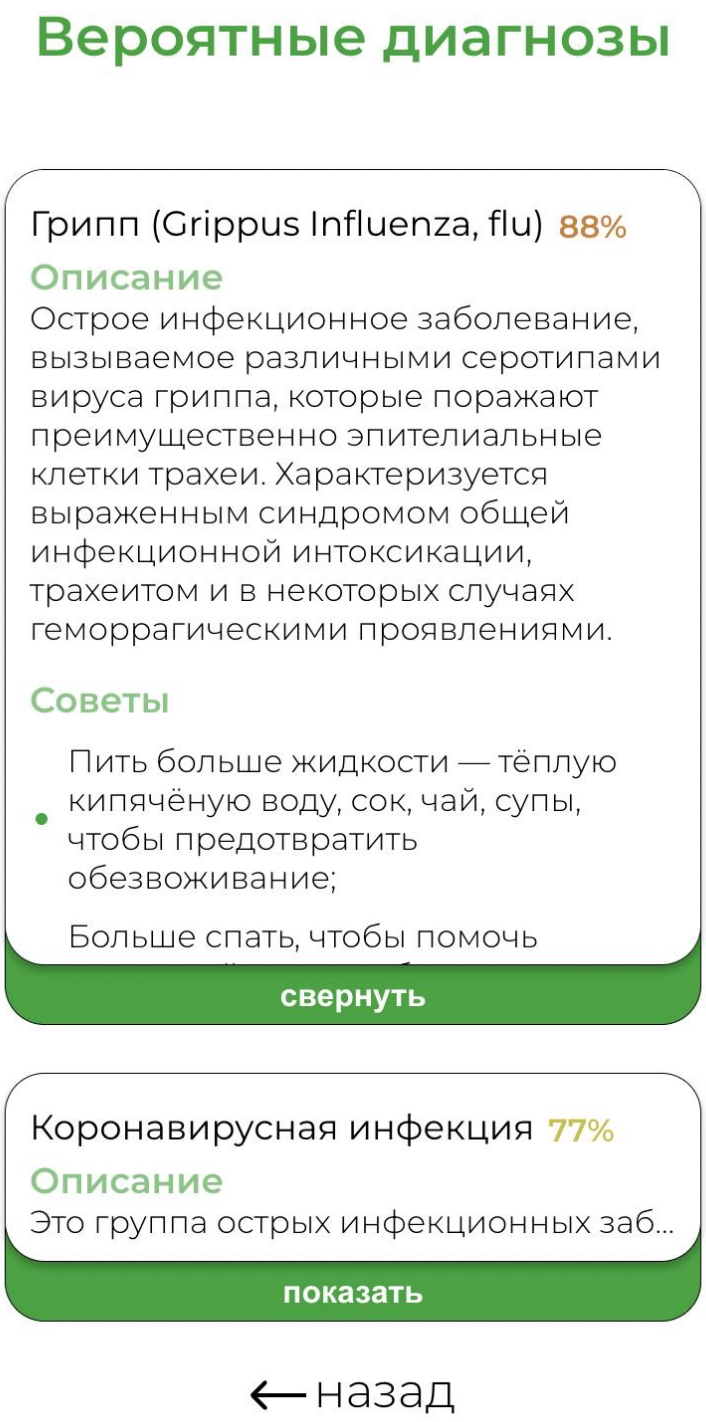

Рисунок 3.7 – Информация о конкретном диагнозе

ЗАКЛЮЧЕНИЕ

В рамках выполнения курсовой работы были изучены основные метрики, использующиеся в решении задач кластерного анализа.

Было реализовано веб-приложение по установке диагноза болезней, основными функциями которого являются: выбор и поиск необходимых симптомов, получения вероятных диагнозов заболеваний и основной информации о них.

В дальнейшем планируется расширение функциональных возможностей веб-приложения, например: добавление нейросетевых технологий для установки вероятного диагноза.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы кластерного анализа // Учебные материалы онлайн: [сайт] - URL: https://studwood.net/870242/marketing/znachenie_tseli_zadachi_klasternogo_analiza (дата обращения: 05.11.2022).

2. Основные метрики, сходства между объектами // Файловый архив: [сайт] - URL: https://studfile.net/preview/1582407/page:3/ (дата обращения: 10.11.2022).

3. Что такое PostgreSQL // Википедия – свободная энциклопедия: [сайт] - URL: https://ru.wikipedia.org/wiki/PostgreSQL (дата обращения: 17.11.2022).

4. Что такое Node.js // Node.js: сайт // URL: https://nodejs.org/ru/ – (дата обращения: 21.11.2022).

5. Что такое Sequilize // Блог SkillFactory: [сайт] - URL: https://blog.skillfactory.ru/glossary/sequelize/ (дата обращения: 28.12.2022).

6. Что такое Express.js // Express – Node.js web application framework: сайт // URL: https://expressjs.com/ru/ – (дата обращения: 03.12.2022).

7. Что такое Redux Toolkit Query (RTK Query) // Redux Toolkit: [сайт] - URL: https://redux-toolkit.js.org/rtk-query/overview (дата обращения: 06.11.2022).