Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ СКРЫТОЙ ФОРМЫ АРТЕРИАЛЬНОЙ ГИПЕРТЕНЗИИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Череваткина Андрея Валерьевича

Научный руководитель
доцент кафедры ТП, к. ф.-м. н. _____ А. А. Кузнецов

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

ВВЕДЕНИЕ

Маскированная(скрытая) форма артериальной гипертензии (МАГ) является нерешенной проблемой современной кардиологии ввиду сложности ее диагностирования. Распространенность МАГ составляет 13–24%, она чаще связана с мужским полом, высоким индексом массы тела (ИМТ), курением, стрессом и сахарным диабетом [1].

Основной задачей данной работы является разработка программного решения для диагностирования маскированной артериальной гипертензии. Оно должно представлять собой форму для ввода ряда медицинских параметров, после ввода которых будет выведен процент риска МАГ.

В настоящей бакалаврской работе ставятся следующие задачи:

— реализовать автоматизированные процессы:
  – разбора исходных данных — анонимизированной базы из порядка 1000 пациентов, предоставленной НИИ кардиологии, в формате CSV;
  – обучения классификатора для диагностирования маскированной артериальной гипертензии по методу логистической регрессии;
— реализовать веб-интерфейс приложения — форму для ввода медицинских параметров пациента, выводящая процент риска МАГ.

1 Предметная область и используемые инструменты

1.1 Маскированная артериальная гипертензия

Артериальная гипертензия — одно из часто встречающихся хронических заболеваний, связанное с повышением систолического (САД $\geq$ 140 мм рт. ст.) и/или диастолического артериального давления (ДАД $\geq$ 90 мм рт. ст.). Опасность артериальной гипертензии состоит в том, что она способна вызвать сердечно-сосудистые и цереброваскулярные заболевания, которые зачастую приводят к смерти. При этом полностью вылечить данное заболевание невозможно. Единственный выход — держать артериальное давление под контролем [2].

Отдельно следует выделить маскированную форму артериальной гипертензии. Для нее характерны значения систолического артериального давления в диапазоне от 120 до 139 мм рт. ст., а диастолическое артериальное давление находится в интервале $80-89$. Такие значения могут быть зафиксированы на приеме у врача (офисные САД и ДАД), но при суточном (СМАД) и домашнем мониторинге (ДМАД) показатели давления могут быть выше.

Данная форма артериальной гипертензии трудно выявляемая. Без своевременного лечения она может перерасти в устойчивую артериальную гипертензию. Этим можно объяснить тот факт, что риск развития сердечно-сосудистых заболеваний при скрытой артериальной гипертензии такой же, как и при манифестной форме [3].

При диагностике артериальной гипертензии любой формы необходимо учитывать ряд факторов: значение офисного АД, частоту сердечных сокращений, пол, наличие ожирения и сахарного диабета, физические нагрузки, особенности питания, наличие депрессии, тревоги и стрессов, вредные привычки (курение, употребление алкоголя), различные биохимические показатели (общий холестерин, холестерин липопротеинов низкой и высокой плотности, уровень креатинина), наследственность, условия и качество жизни. При подозрении наличия скрытой артериальной гипертензии необходимо также использовать методы СМАД и ДМАД.

1.2 Задача классификации

Задача классификации — это задача интеллектуального анализа данных (Data Mining), заключающаяся в следующем: есть множество объек-

тов, которые могут быть распределены по нескольким классам, также выделено конечное множество объектов, для которых заданы соответствующие им классы (обучающая выборка). Требуется разработать алгоритм, который сможет классифицировать любой объект исходного множества.

Каждый объект задаётся определённым образом. Типичный вид входных данных — признаковое описание.

В общем случае классификация может быть многоклассовой, но чаще всего решается задача бинарной классификации. Многоклассовую задачу всегда можно свести к бинарной.

Рассматриваемая в данной работе задача классификации предполагает разбиение данных всех пациентов на 2 класса: класс здоровых пациентов и класс пациентов с маскированной формой артериальной гипертензии.

1.3 Подготовка и очистка данных

Анализировать можно как качественные, так и некачественные данные. Результат будет достигнут и в том, и в другом случае. Для обеспечения качественного анализа необходимо проведение предварительной обработки данных, которая является необходимым этапом процесса Data Mining.

Оценивание качества данных. Данные, полученные в результате сбора, должны соответствовать определенным критериям качества. Таким образом, можно выделить важный подэтап процесса Data Mining — оценивание качества данных.

Качество данных (Data quality) — это критерий, определяющий полноту, точность, своевременность и возможность интерпретации данных.

Данные могут быть высокого качества и низкого качества, последние - это так называемые грязные или "плохие"данные.

Данные высокого качества — это полные, точные, своевременные данные, которые поддаются интерпретации.

Данные низкого качества, или грязные данные — это отсутствующие, неточные или бесполезные данные с точки зрения практического применения (например, представленные в неверном формате, не соответствующем стандарту). Грязные данные появились не сегодня, они возникли одновременно с системами ввода данных.

Наиболее распространенные виды грязных данных:

— пропущенные значения;

— дубликаты данных;
— шумы и выбросы [4].

1.4 Регулярные выражения

Регулярные выражения — это небольшой язык, который вы можете использовать внутри Python и многих других языках программирования. Зачастую регулярные выражения упоминаются как «regex», «regexp» или просто «RE», от regular expressions. Такие языки как Perl и Ruby фактически поддерживают синтаксис регулярных выражений прямо в собственном языке. Python же поддерживает благодаря библиотеке, которую вам нужно импортировать. Основное использование регулярных выражений — это сопоставление строк. Вы создаете правила сопоставления строк, используя регулярные выражения, после чего вы применяете их в строке, чтобы увидеть, присутствуют ли какие-либо сопоставления [5].

1.5 Структура данных «куча»

Структура данных «куча» используется в проекте для проведения сортировки слиянием в одном из этапов подготовки данных.

Структура данных «куча» поддерживает следующие операции:
— Добавить элемент в структуру данных.
— Извлечь из структуры данных наименьший (вариант — наибольший) элемент. Извлеченный элемент удаляется из структуры

При этом в структуре могут храниться одинаковые элементы.

В куче элементы хранятся в виде двоичного дерева, то есть у элементов есть два потомка — левый и правый. В вершине кучи находится один элемент, у него — два потомка на следующем уровне, у них, в свою очередь, по два потомка на третьем уровне (итого — 4 элемента на третьем уровне) и т. д. Уровни заполняются в порядке увеличения номера уровня, а сам уровень заполняется слева направо. У элементов последнего уровня нет ни одного потомка, возможно, что и у некоторых элементов предпоследнего уровня нет потомков. Также в куче может быть один элемент, у которого только один потомок (левый).

При этом для элементов кучи верно следующее свойство — каждый из элементов кучи большее или равен всех своих потомков. В частности это означает, что в вершине кучи хранится наибольший элемент.

Операции добавления и удаления элемента из кучи имеют временную сложность $O(\log n)$, где n — число элементов в куче.

1.6 Формат JSON

В продукте используется, в числе прочих, формат JSON для хранения данных. Это один из наиболее удобных форматов данных при взаимодействии с JavaScript. Если нужно с сервера взять объект с данными и передать его клиенту, то в качестве промежуточного формата – для передачи по сети, почти всегда используют именно его.

Данные в формате JSON (RFC 4627) могут представлять собой:

- JavaScript-объекты ... ;
- Массивы [...];
- Значения одного из типов:
  - строки в двойных кавычках;
  - число;
  - логическое значение true/false;
  - null.

Почти все языки программирования имеют библиотеки для преобразования объектов в формат JSON [6].

1.7 Технология AJAX

AJAX (аббревиатура от «Asynchronous Javascript And Xml») — технология обращения к серверу без перезагрузки страницы.

За счет этого уменьшается время отклика и веб-приложение по интерактивности больше напоминает десктоп.

Несмотря на то, что в названии технологии присутствует буква X (от слова XML), использовать XML вовсе не обязательно. Под AJAX подразумевают любое общение с сервером без перезагрузки страницы, организованное при помощи JavaScript.

В первую очередь AJAX полезен для форм и кнопок, связанных с элементарными действиями, например, добавить в корзину или подписаться.

2 Реализация продукта

2.1 Исходные данные

Результаты обследования пациентов были предоставлены саратовским НИИ кардиологии. В исследовании участвовал 901 человек. По результатам обследования было выявлено 542 здоровых пациента, 290 пациентов с явной формой артериальной гипертензии и 69 пациентов со скрытой формой.

Данные хранятся в файле формата CSV (Comma-Separated Values — значения, разделённые запятыми).

Задача классификации, рассматриваемая в данной работе, состоит в распределении пациентов по 3 классам:

— ОК — нормотоники;
— МАГ — пациенты с маскированной артериальной гипертензией;
— Манифестная АГ — пациенты с устойчивой формой артериальной гипертензии.

Манифестная АГ сравнительно проста в диагностировании — если артериальное давление пациента (систолическое и диастолическое) превышает определенные пороги, либо же пациент принимает препараты для снижения артериального давления, то можно с уверенностью сказать, что данный пациент болен манифестной формой АГ. Перед разрабатываемым классификатором не стоит задача диагностирования манифестной АГ, поэтому все записи о пациентах с данной формой АГ можно удалить из обучающей выборки. Классификатор должен лишь отличать здоровых людей от больных маскированной формой АГ.

Продукт создан в ходе командной работы. Состав команды: Неверова Елена Андреевна, Череваткин Андрей Валерьевич, Кузнецов Александр Александрович (научный руководитель).

2.2 Используемые инструменты

Все программы написаны на языке программирования `Python 3.5`. В настоящее время эта наиболее популярные инструмент решения задач `Data Mining`.

Для работы с данными и параметрами были использованы следующие библиотеки:

— `pandas` — пакет средств, предназначенный для первичной обработки

и представления данных в программе (`read_csv` — функция чтения данных из файла `CSV`, работа с классом `DataFrame`);
— `numpy` — библиотека, обеспечивающая поддержку больших многомерных массивов и матриц, а также обширный набор высокоуровневых математических функций для выполнения операций с этими массивами;
— `scipy` — библиотека, предоставляющая набор функция для научных и инженерных расчётов (была использована функция `interp`);
— `csv` — библиотека, предназначенная для работы с файлами формата `CSV`;
— `json` — библиотека для работы с файлами формата `JSON`, в которых сохраняются списки параметров;
— `heapq` — библиотека для работы со очередью в виде структуры данных куча (heap);
— `argparse` — библиотека для работы с аргументами командной строки;
— `subprocess` — библиотека для запуска сторонних процессов из среды Python.

2.3 Процесс подготовки данных

Была сформирована последовательность этапов подготовки данных:
— Сгенерировать условия для параметров, которые будут использоваться в классификаторе. Условие заключается в том, что данный числовой параметр больше или меньше некоторого значения;
— Сгенерировать для каждого пациента попарные списки условий, которые выполняются для данного пациента;
— Подсчитать количество пациентов с МАГ и здоровых для каждой сгенерированной пары условий;
— Отфильтровать пары условий по кси-статистике;
— Переформатировать пары условий;
— Сгенерировать таблицу CSV, в которой для каждой пары условий и для каждого пациента хранится значение 0 или 1, которое показывает, выполняется ли пара условий для данного пациента. Данный формат данных наиболее удобен для обучения классификатора;
— Применить метод логистической регрессии, обучить классификатор и сохранить его модель в файлы, которые будут использоваться в веб-

интерфейсе.

2.3.1 Программа генерации условий для параметров

Программа считывает параметры командной строки, загружает базу пациентов из файла `AGDatabase.csv` в объект типа `DataFrame`, затем фильтрует ее, после чего генерирует список условий для параметров и сохраняет их в файл JSON.

Программа написана Неверовой Е. А. В ходе выполнения данной работы в программу были добавлены обработка параметров командной строки и фиксация факта завершения выполнения программы.

2.3.2 Программа генерации списков пар условий для каждого пациента

Программа считывает параметры командной строки, загружает базу пациентов из файла `AGDatabase.csv` в объект типа `DataFrame`, затем фильтрует ее с помощью функции `define_df` из предыдущей программы, после чего считывает условия для параметров, сгенерированные предыдущей программой, с помощью функции `load_parameters`. После этого запускается функция `create_pairs`, выполняющая основную задачу программы.

Программа написана Неверовой Е. А. В ходе выполнения данной работы в программу были добавлены обработка параметров командной строки и фиксация факта завершения выполнения программы.

2.3.3 Программа для подсчета количества пациентов по классам

Смысл программы в том, чтобы подсчитать для каждой пары условий для параметров, сколько пациентов (здоровых и с МАГ отдельно) удовлетворяют обоим условиям в этой паре. Поскольку пары условий в каждом файле отсортированы лексикографическом, можно произвести подобие сортировки слиянием, что и происходит в программе. По мере чтения пар условий поддерживается структура данных «куча»(«heap») из пар условий для параметров, которая позволяет получать пары условий из всех файлов пациентов в лексикографическом порядке. По окончании подсчета результаты сохраняются в текстовый файл.

2.3.4 Программа для фильтрации пар условий по кси-статистике

Программа считывает параметры командной строки, загружает базу пациентов из файла `AGDatabase.csv` в объект типа `DataFrame`, открывает

текстовый файл, созданный на предыдущем этапе, на чтение и открывает текстовый файл для результатов на запись. Пары условий, у которых число пациентов с МАГ меньше 5, а также те, у которых число здоровых пациентов меньше 10, отбрасываются. Затем для пары условий вычисляется значение кси-статистики, и если оно меньше определенного порога (7.87944), пара условий также отбрасывается. Оставшиеся пары записываются в результирующий файл вместе с вычисленным значением кси-статистики.

Данная программа работает таким образом, чтобы поддерживать порционную передачу результатов следующей программе. Каждая пара условий, которая записывается в выходной файл, размещается на новой строке, а по окончании фильтрации в файл выводится специальный символ, который сигнализирует следующей программе, считывающей пары из этого файла, об окончании вывода.

Программа написана Неверовой Е. А. В ходе выполнения данной работы в программу были добавлены обработка параметров командной строки, поддержка порционного вывода данных и фиксация факта завершения выполнения программы.

Для обеспечения порционного вывода данных запись очередной строки в файл происходит в цикле вместе со сбросом (flush), а по окончании цикла в файл записывается последняя строка со специальным символом, который означает конец данных.

2.3.5 Программа переформатирования пар условий

Программа считывает параметры командной строки, открывает файл, созданный на предыдущем этапе, на чтение, открывает выходной файл CSV на запись и в цикле считывает строки из входного файла и записывает их в выходной файл в формате CSV. Цикл повторяется до тех пор, пока во входном файле не встретится строка с символом окончания вывода. После окончания цикла программа сохраняет список пар условий для параметров в файл JSON.

Программа написана Неверовой Е. А. В ходе выполнения данной работы в программу были добавлены обработка параметров командной строки, поддержка порционного приема входных данных и фиксация факта завершения выполнения программы.

Для обеспечения поддержки порционного приема данных при чтении

очередной строки из файла происходит проверка, не содержит ли она специальный символ конца данных. Если он найден, цикл обработки ввода прерывается.

2.3.6 Программа генерации базы пациентов с парами условий

Программа считывает параметры командной строки, загружает базу пациентов из файла `AGDatabase.csv`, загружает список пар условий для параметров из файла JSON, созданного на предыдущем этапе, открывает на запись новый файл CSV и перебирает в цикле все строки исходной базы пациентов. Внутри цикла, во внутреннем цикле, перебираются все пары условий для параметров из файла. Если оба условия для данного пациента выполняются, в текущую ячейку новой базы пациентов сохраняется 1, в противном случае — 0. Также для каждого пациента сохраняется класс, к которому он принадлежит (ОК или МАГ).

Программа написана Неверовой Е. А. В ходе выполнения данной работы в программу были добавлены обработка параметров командной строки и фиксация факта завершения выполнения программы.

2.3.7 Программа обучения классификатора

Программа считывает параметры командной строки, загружает базу пациентов с парами условий, созданную на предыдущем этапе, обучает классификатор с помощью метода логистической регрессии и сохраняет его модель в файл.

Программа написана Неверовой Е. А. В ходе выполнения данной работы в программу были добавлены обработка параметров командной строки и фиксация факта завершения выполнения программы.

2.3.8 Вспомогательная программа фильтрации условий для параметров

Данная программа не входит в последовательность подготовки данных, однако в ходе разработки продукта она была полезной для отбора условий для параметров пациентов.

Программа считывает параметры командной строки, открывает файл, созданный на этапе 4, на чтение, и в цикле считывает строки из входного файла и выбирает из всех пар условий для одноименных параметров такую, для которой доля больных МАГ, для которых эта пара условий выполняется,

максимальна. Затем пары условий сортируются в порядке убывания этой доли и записываются в выходной файл CSV. Сортировка условий помогла определить параметры, которые сильнее всего влияют на риск МАГ.

2.3.9 Модуль вспомогательных функций

В данном модуле содержатся различные вспомогательные функции, используемые в других программах.

2.4 Автоматизация процесса подготовки данных

Необходимость автоматизации процесса подготовки данных обусловлена тем, что он регулярно повторялся в процессе исследования при изменениях алгоритма или набора параметров.

Все этапы должны быть проведены дважды: с биохимическими параметрами и без них, таким образом будут созданы два классификатора. Затем, в веб-интерфейсе введенные пользователем данные поступают на вход обоих классификаторов, оба она выдают процент риска МАГ, который затем усредняется. Это обусловлено тем, что биохимические параметры были получены лишь от части пациентов, поэтому один классификатор с биохимическими параметрами недостаточно точен, а один классификатор без них не использует очевидного преимущества.

Исполняемый файл assemble.py последовательно запускает все этапы подготовки данных, причем делает это параллельно с биохимическими параметрами и без них.

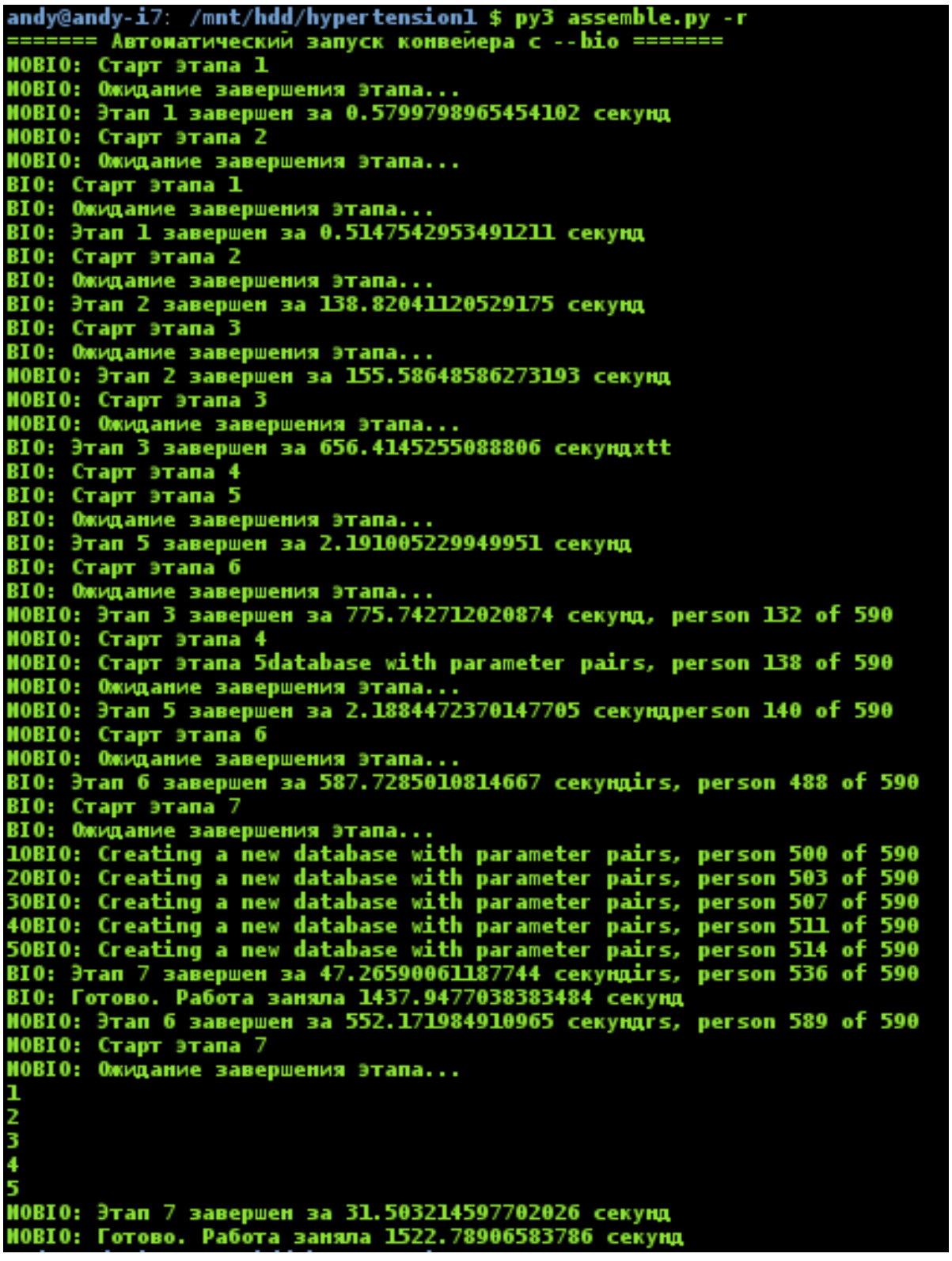
Каждая программа для подготовки данных имеет опцию командной строки `-b` или `--bio`, которая указывает ей работать с биохимическими параметрами, а ее отсутствие — не работать с ними.

Программа assemble.py также имеет опцию `-b` (`--bio`), она указывает ей запускать все этапы подготовки с опцией `-b`, то есть с биохимическими параметрами. Отсутствие ее, соответственно, указывает программе запускать подготовку данных без биохимических параметров. Также, для того чтобы обеспечить параллельное выполнение подготовки с биохимическими параметрами и без них, программа assemble.py без опции `-b` запускает саму себя с опцией `-b` прежде чем приступить к запуску собственно подготовки данных.

Также каждая программа для подготовки данных фиксирует факт своего завершения в файлах complete.txt и complete.txt соответственно (послед-

ние 2 строки каждой программы). Программа assemble.py проверяет содержимое этих файлов перед запуском процесса подготовки данных, чтобы можно продолжить его с последнего завершенного этапа, не повторяя уже пройденные (строки 61–70). Опция командной строки `-r` (`--reset`) позволяет проигнорировать содержимое файлов complete.txt (completeBio.txt) и начать процесс с первого этапа (строки 29–42).

Скриншот запуска программы assemble.py:



2.5 Добавление шума к исходной базе пациентов

Количество больных МАГ в исходной базе достаточно мало для обучения классификатора, поэтому для повышения его эффективности полезно увеличить их количество, добавив к каждому пациенту с МАГ несколько похожих, с параметрами, колеблющимися в некотором диапазоне око-

ло исходного значения в соответствии с распределением Гаусса. Программа noise.py выполняет эту задачу. Она принимает в качестве входного файла AGDatabaseOriginal.csv, который должен являться копией исходной базы пациентов, и выводит новую, увеличенную в объеме базу в файл AGDatabase.csv.

У программы есть обязательный параметр — целое число записей, которые нужно добавить к каждой записи пациентов с МАГ. Также есть необязательный параметр `-a` (`--all`), указывающий программе генерировать новые записи для всех записей в исходной базе, а не только к больным МАГ.

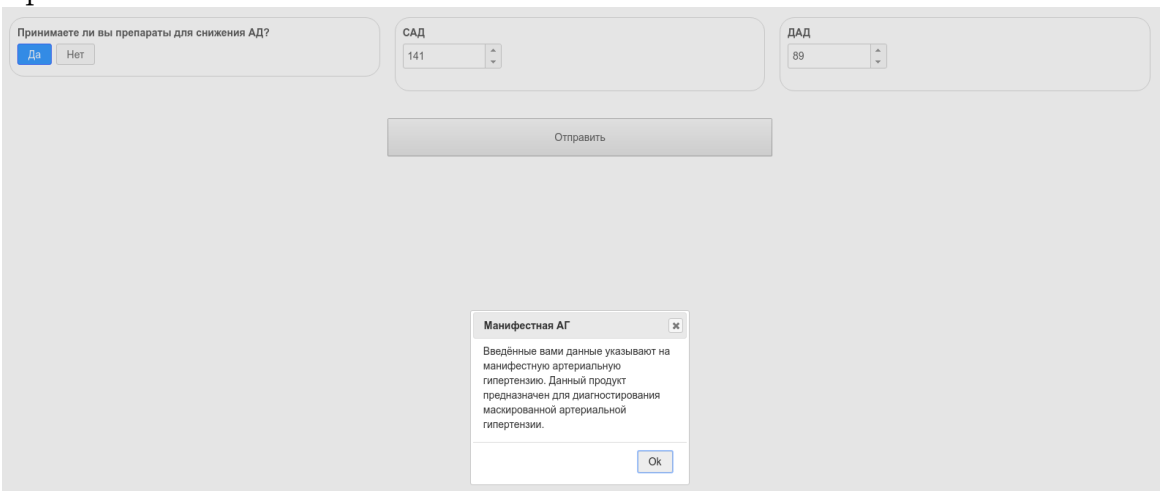
2.6 Веб-интерфейс

Пользовательским интерфейсом классификатора является веб-интерфейс.

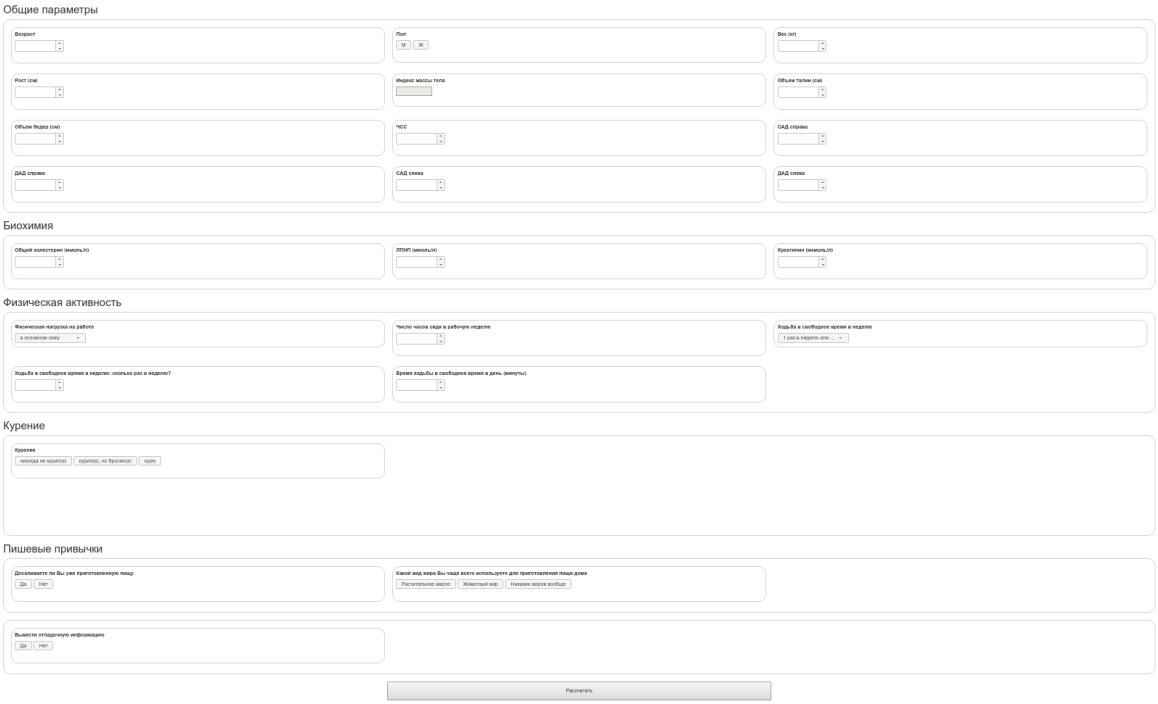
Данный классификатор предназначен для диагностирования именно маскированной артериальной гипертензии. Поэтому веб-интерфейс открывает доступ к форме классификатора только после того как убедиться, что у пациента отсутствует манифестная форма артериальной гипертензии. Эта проверка проводится путем вспомогательной формы с 3 вопросами:

1. Принимаете ли вы препараты для снижения артериального давления?
2. Систолическое артериальное давление (САД)
3. Диастолическое артериальное давление (ДАД)

В случае, если пациент принимает препараты для снижения АД, либо его САД превышает 140 мм рт. ст., либо ДАД превышает 90 мм рт. ст., веб-интерфейс выводит сообщение о том, что у пациента присутствует манифестная форма АГ.



Если же ни одно из этих условий не выполняется, веб-интерфейс переходит к форме классификатора (путь `/main`).



После заполнения всех полей формы пользователь должен нажать кнопку «Рассчитать», после чего будет выведено значение риска, что данный пациент болеет маскированной формой АГ.

Веб-интерфейс реализован с использованием микрофреймворка `Flask`. Значения параметров приходят от пользователя в виде списка, который

преобразуется к словарю `named_args`, в котором ключами являются имена параметров в исходной базе пациентов, а значениями — собственно значения параметров. Затем загружаются списки пар параметров с биохимическими параметрами и без них, из файлов, сгенерированных при подготовке данных. После этого для каждой пары определяем, выполняются ли условия в паре для пациента, параметры которого были введены в форму. Полученный список передается фунции `predict` модуля `specificity`, которая возвращает результат работы классификатора — вероятность того, что данный пациент болен МАГ в процентах. Результат работы этой функции возвращается пользователю для отображения на веб-странице.

2.6.1 Обработка ошибок

К функции `calculate`, которая обращается к классификатору, применен декоратор `errorHandler`, который перехватывает все возникающие в функции ошибки и использует функцию `notify`, чтобы выслать сообщение об этом в групповой чат разработчиков проекта в мессенджере Telegram, используя Telegram Bot API. Перехват ошибок происходит с помощью конструкции языка Python `try...except`. Обращение к Telegram Bot API происходит путем отправки запроса по протоколу HTTPS, сформированного в соответствии со стандартом Telegram Bot API.

2.6.2 Привязка доменного имени

Веб-интерфейсу присвоено доменное имя

`hypertension2.andychweb.com`

которое указывает на IP-адрес сервера, работающего на операционной системе Ubuntu 16.04. Сервер работает на операционной системе Поскольку на одном и том же сервере может работать множество сайтов одновременно, возникает проблема — сервер должен опознать, к какому из его сайтов адресован данный запрос HTTP(S). Данная проблема может быть решена с помощью веб-сервера `nginx`, при использовании его в качестве прокси-сервера.

Все HTTP(S) запросы на сервер будут поступать на стандартный порт 80. Веб-интерфейс `web_interface.py` работает локально в порту 6445. Таким образом, необходимо указать `nginx`, что все запросы по доменному имени `hypertension2.andychweb.com` необходимо перенаправлять локально на порт 6445. Сделать это можно, настроив конфигурационный файл и поместив

его в папку `/etc/nginx/sites-enabled`.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Никитина, Н.* МАСКИРОВАННАЯ АРТЕРИАЛЬНАЯ ГИПЕРТЕНЗИЯ: АКТУАЛЬНА ЛИ ПРОБЛЕМА ДЛЯ БОЛЬНЫХ РЕВМАТОИДНЫМ АРТРИТОМ? | Никитина | Артериальная гипертензия [Электронный ресурс] / Н. Никитина, Т. Романова, А. Ребров. — URL: http://htn.almazovcentre.ru/jour/article/view/467 (Дата обращения 20.05.2018).

2 *Чазова, И. Е.* Клинические рекомендации. Диагностика и лечение артериальной гипертонии / И. Е. Чазова, Е. В. Ощепкова, Ю. В. Жернакова // *Кардиологический вестник.* — 2015. — № 1. — С. 5, 7–8, 19.

3 *Рабочая группа по лечению артериальной гипертонии европейского Общества Гипертонии (European society of hypertension Esh) и европейского Общества кардиологов (European society of Cardiology, E.* Рекомендации по лечению артериальной гипертонии. esh/esc 2013 / Е. Рабочая группа по лечению артериальной гипертонии европейского Общества Гипертонии (European society of hypertension, Esh) и европейского Общества кардиологов (European society of Cardiology // *Российский кардиологический журнал.* — 2014. — № 1. — С. 15, 17, 21–22.

4 Очистка данных: проблемы и актуальные подходы | Журнал BPM World [Электронный ресурс]. — URL: http://iso.ru/ru/press-center/journal/1789.phtml Дата обращения 21.05.2018.

5 Регулярные выражения в Python [Электронный ресурс]. — URL: https://python-scripts.com/import-re-regular-expression Дата обращения 21.05.2018.

6 Формат JSON, метод toJSON [Электронный ресурс]. — URL: http://learn.javascript.ru/json Дата обращения 24.05.2018.