Содержание

Введение

На протяжении всего времени существования человечества проблема возникновения, исследования и лечения различных заболеваний у человека является важной задачей медицинской деятельности. Особенно это касается онкологических заболеваний. На сегодняшний день нет четкой причины, по которой люди заболевают раком, но существует множество способов ранней диагностики таких заболеваний [4].

В данной работе рассматривается использование результатов моделирования, соответствующих методике микроволновой радиотермометрии молочных желез на основе работы специального диагностического комплекса РТМ-01-РЭС [34]. Также рассматриваются популярные алгоритмы классификации данных и библиотеки для языка программирования Python, реализующие данные алгоритмы, и сферы применения данных алгоритмов.

Главной целью работы является разработка программного обеспечения для валидации данных компьютерного моделирования биотканей и возможности диагнорстирования рака молочной железы по температурным данным пациентов с использованием машинного обучения и различных методов классификации. Так как проведение классификации только на температурных данных может не дать хорошего результата, необходимо выявить какой из алгоритмов будет лучше работать с разными размерами опухоли и вариациями остальных параметров.

1 Методы и области применения машинного обучения

1.1 Обзор литературы

Статья Полякова М.В., Хоперскова А.В. «Математическое моделирование пространственного распределения радиационного поля в биоткани: определение яркостной температуры для диагностики», опубликованная в Вестнике Волгоградского государственного университета, посвящена проведению имитационных экспериментов по моделированию динамики температурных и радиационных полей в биотканях молочной железы. В работе вместо традиционно используемых моделей с однородными параметрами используются вычислительные модели максимально приближенные к реалистичной геометрической структуре тканей с неоднородными характеристиками [39].

В статье Веснина С.Г., Седанкина М.К. «Миниатюрные антенны-аппликаторы для микроволновых радиотермометров медицинского назначения», опубликованная в журнале «Биомедицинская радиоэлектроника», описывается анализ миниатюрных антенн-аппликаторов, предназначенных для измерения собственного излучения тканей человека с помощью микроволновых радиотермометров. Приведены простые аппроксимационные формулы для распределения температуры в молочной железе при наличиии злокачественной опухоли [25].

В работе Van Ongeval Ch. «Digital mammography for screening and diagnosis of breast cancer: an overview» обсуждается цифровая телемаммография как новая техника для диагностирования заболеваний молочных желез. Также в данной работе детально рассматривается проблема практической реализации различных систем для визуализации телемаммографической диагностики, высокой стоимости обследования и высокой квалификации специалистов-радиологов [17].

Работа Nisreen I. Yassin, Shaimaa Omran, Enas M. F. El Houby, Enas M. F. El Houby, Hemat Allam «Machine Learning Techniques for Breast Cancer Computer Aided Diagnosis Using Different Image Modalities: A Systematic Review»

посвящена опыту медиков в диагностики и обнаружении рака молочной железы с использованием алгоритмов машинного обучения на основе визуализированных данных обследования пациентов. Целью работы является исследование современного уровня техники в отношении систем компьютерной диагностики и обнаружения рака молочной железы [12].

В статье Левшинского В., Полякова М., Лосева А., Хоперскова А. «Verification and Validation of Computer Models for Diagnosing Breast Cancer Based on Machine Learning for Medical Data Analysis» рассмотрен поход проверки результатов моделирования физических процессов в биотканях с использованием глубокого анализа и машинного обучения. При обучении моделей используются данные измерений температуры пациентов согласно методу радиотермометрии. Так же в работе выделяются на основе набора данных для обучения новые признаки, похожие на те, которые используют медики при обследовании пациентов [9].

В работе Рамсундара Б., Истмана П., Уолтерса П., Панде В. «Глубокое обучение в биологии и медицине» обсуждается применение глубокого обучения в популярных направлениях современных исследований, а особенно в биологии и медицине. Работа содержит описание архитектуры алгоритмов в машинном обучении для применения в задачах данных сферах, а так же некоторые практические примеры по использованию [41].

Статья Jian Ma, Pengchao Shang, Chen Lu, Safa Meraghni «A portable breast cancer detection system based on smartphone with infrared camera» посвящена разработке системы обнаружения рака молочной железы с использованием смартфона с инфракрасной камерой. Для обследования использовался метод инфракрасной термографии и алгоритм классификации k-ближайших соседей. Авторам удалось достигнуть точности определения наличия заболевания больше 98% [6].

В части работ рассмотрен метод микроволновой радиотермометрии и его применение при обследовании рака молочных желез. Так же в некоторых работах рассмотрены способы применения машинного обучения для диагностирования различных заболеваний, в том числе онкологических. Рассмот-

рим далее подробно области применения и основные алгоритмы в машинном обучении.

1.2 Области применения

Использование алгоритмов машинного обучения позволяет решать задачи в различных сферах деятельности человека, таких как недвижимость, сельское хозяйство, экономика, а так же медицина. По данным агенства Frost & Sullivan спрос на разработки, в которых используется машинное обучение в медицине, увеличивается с каждым годом примерно на 40% [22]. Такие разработки могут использоваться как для диагностики заболеваний, так и для биохимических исследований.

Методы машинного обучения активно применяются при медицинском сканировании различных типов, таких как УЗИ или компьютерная томография. Благодаря алгоритмам распознавания образов на изображениях есть возможность анализировать результаты таких исследований и указывать на проблемные участки. Также возможно прогнозирование диагноза пациента по различным его параметрам и результатам исследования. Но программное обеспечение, использующее данные алгоритмы пока не может заменить полностью работу медиков и используется в основном при первичных исследованиях.

При компьютерном моделировании алгоритмы машинного обучения могут использоваться для валидации получившихся данных, или прогнозирования течения каких-либо физических процессов.

1.3 Методы классификации в машинном обучении

Классификация данных состоит из прогнозирования определенного результата на основе уже известных данных. Чтобы предсказать результат, ал-

горитм обрабатывает данные, содержащие набор атрибутов и соответствующий каждому набору результат, обычно называемый атрибутом прогнозирования цели или классом [5]. Формируется модель алгоритма, которая пытается обнаружить отношения между атрибутами, которые позволили бы предсказать результат [7]. Такая процедура называется обучением модели, а набор данных, используемый для этого – тестовой выборкой [10].

Следующим шагом после обучения модели является прогнозирование – процедура определения класса, при которой используется набор данных с неизвестными классами. Такой набор данных, который содержит тот же набор атрибутов, за исключением атрибута прогнозирования, часто называют тестовой выборкой [19].

Алгоритм анализирует входные данные и выдает прогноз. Точность прогноза определяет, насколько «хорош» алгоритм. Например, в медицинской базе данных обучающий набор должен иметь соответствующую информацию о пациенте, записанную ранее, где атрибутом прогноза является наличие или отсутствие у пациента проблем со здоровьем.

Для определения того, какой именно алгоритм использовать для конкретной задачи можно воспользоваться схемой, изображенной на рисунке 1. Исходя из того, что в текущей задаче используется не тестовая информация и имеется 160 примеров, то были выбраны алгоритмы SVM, k-ближайших соседей и наивный байесовский классификатор, описанные ниже.

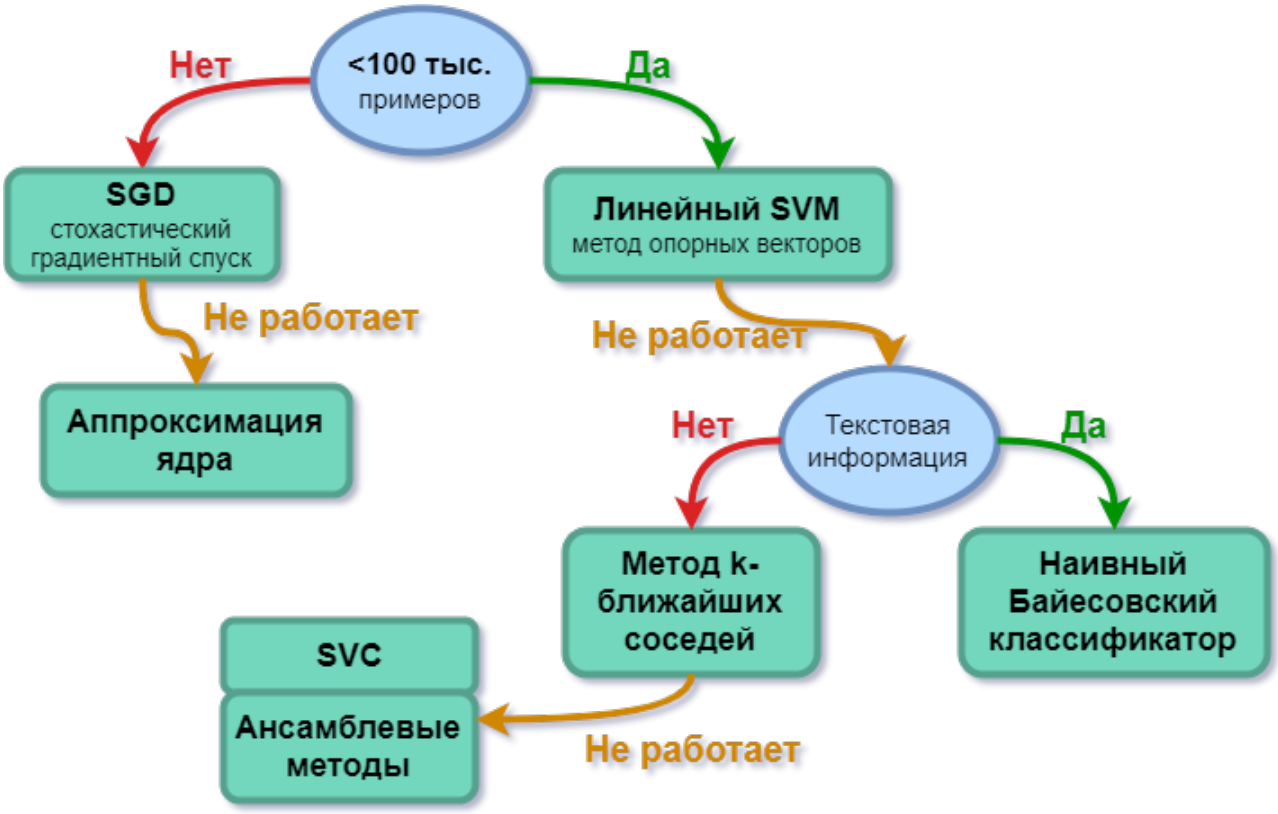

Рисунок 1 – Схема для определения алгоритма классификации для конкретной задачи

1.4 Метод опорных векторов

Метод опорных вектором или SVM – это метод статистической классификации [2]. Он широко используется для задач различного рода и хорошо себя в них показывает [3].

Основной идеей метода является представление атрибутов данных в виде векторов и переход в пространство более высокой размерности, чем получившееся на этапе представления векторами. Затем ищется гиперплоскость с максимальным зазором в пространстве между объектами разных классов [42] [15].

На рисунке 2 показан пример классификации методом SVM. Красной линией выделена как раз та самая гиперплоскость, четко разделяющая объекты разных классов друг от друга.

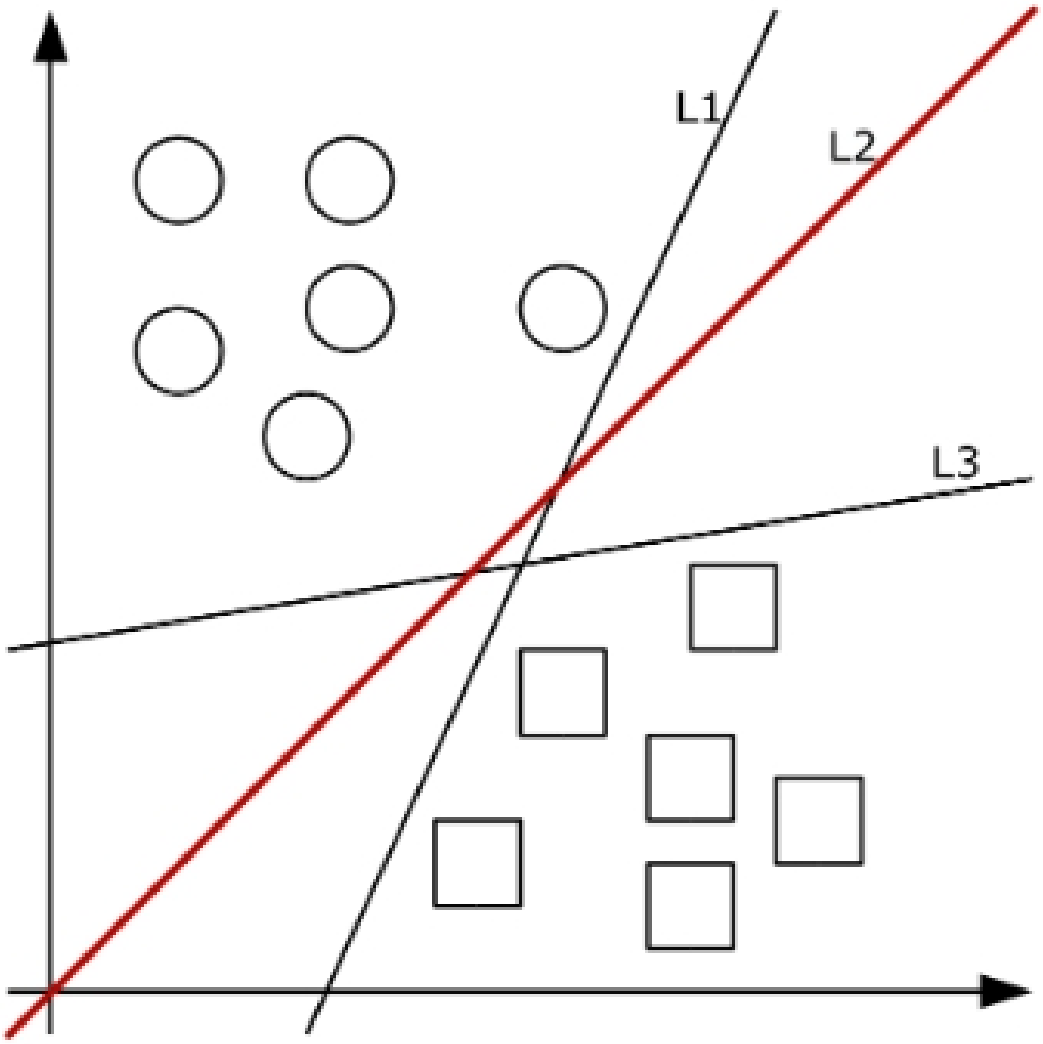

Рисунок 2 – Пример классификации методом SVM

Алгоритм может использоваться с одним из следующих видов ядер [3]:

- Полиномиальное (однородное) $k(x, x') = (x \cdot x')^d$;
- Полиномиальное (неоднородное) $k(x, x') = (x \cdot x' + 1)^d$;
- Радиальная базисная функция $k(x, x') = exp(-\gamma||x - x'||^2)$, для $\gamma > 0$;
- Радиальная базисная функция Гаусса $k(x, x') = exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right)$;
- Сигмоид $k(x, x') = tanh(kx \cdot x' + c)$.

1.5 Метод k-ближайших соседей

Алгоритм k-ближайших соседей является простым статистическим алгоритмом обучения, в котором объект классифицируется своими соседями. При классификации таким методом объект относится к классу, наиболее распространенному среди его k-ближайших соседей [40] [44]. Пример классификации приведен на рисунке 3, где в качестве классифицируемого объекта используется прямоугольник и существует несколько объектов известных классов – белые точки и черные. Замерив расстояние от объекта до его соседей с различными классами и основываясь на методе k-ближайших соседей данный объект будет отнесен к классу черный точек, а не белых.

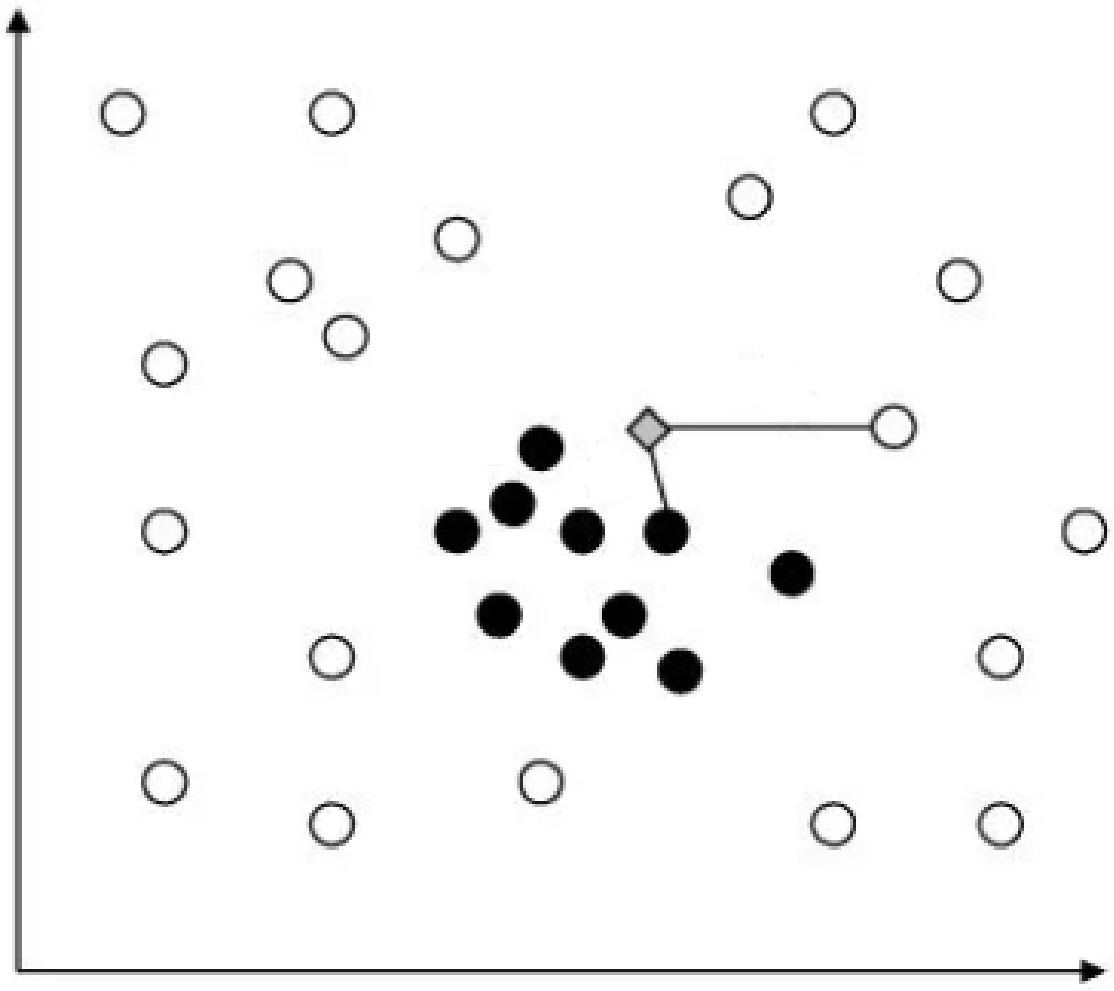

Рисунок 3 – Пример классификации методом k-ближайших соседей

При нахождении атрибутов учитывается значимость атрибутов и часто применяется прием растяжения осей, демонстрируемый в формуле (1). Использование данного приема снижает ошибку классификации.

$$D_E = \sqrt{3(x_A - y_A)^2 + (x_B - y_B)^2}, \tag{1}$$

где x_A, y_A – значения атрибута A в наборе данных, x_B, y_B – значения атрибута B.

Данный алгоритм возможно применять как для данных с маленьким количеством атрибутов, так и с достаточно большим. Важным моментом при работе с алгоритом является определение функции расстояния между значениями. Примером такой функции может быть евклидово расстояние – формула (2).

$$D_E = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}, \tag{2}$$

где x_i, y_i – значения атрибутов в наборе данных.

1.6 Наивный байесовский классификатор

Наивный байесовский классификатор является простым вероятностным классификатором и основывается на применении теоремы Байеса со строгими предположениями о независимости [40] [29]. Хотя наивный байесовский классификатор редко применим к большинству реальных задач, но зачастую в определенных задачах он демонстрирует хорошие результаты и часто конкурирует с более сложными методами, такими как SVM и классификационным деревьями [10]. Классификация данным методом очень зависит от распределения зависимостей атрибутов, а не от самих зависимостей [32].

Вероятностная модель классификатора:

$$p(C|F_1, ..., F_n) = \frac{p(C)p(F_1, ..., F_n|C)}{p(F_1, ..., F_n)}, \tag{3}$$

где C – класс модели, а F_i – классифицируемые модели [10].

Использование формулы (3) при классификации дает минимально значение среднего риска или математичского ожидания ошибки:

$$R(a) = \sum_{y \in Y} \sum_{\varsigma \in Y} \lambda_y P_y P_{x,y} \{a(x) = \varsigma | y\}, \tag{4}$$

где λ_y – цена ошибки при отнесении объекта класса Y к какому-либо другому классу.

1.7 Популярные библиотеки с реализацией методов машинного обучения

На текущий момент существует множество готовых реализаций алгоритмов машинного обучения и не имеет смысла делать то же самое с нуля, если задача не имеет каких-то особенностей, делающих невозможным использование готовых библиотек. Каждая из библиотек, рассматриваемых в работе, хороша в своей области, успешно используется в решении задач и проверена временем. Рассмотрим некоторые из популярных библиотек для языка программирования Python по данным рейтинга на GitHub (рисунок 4)

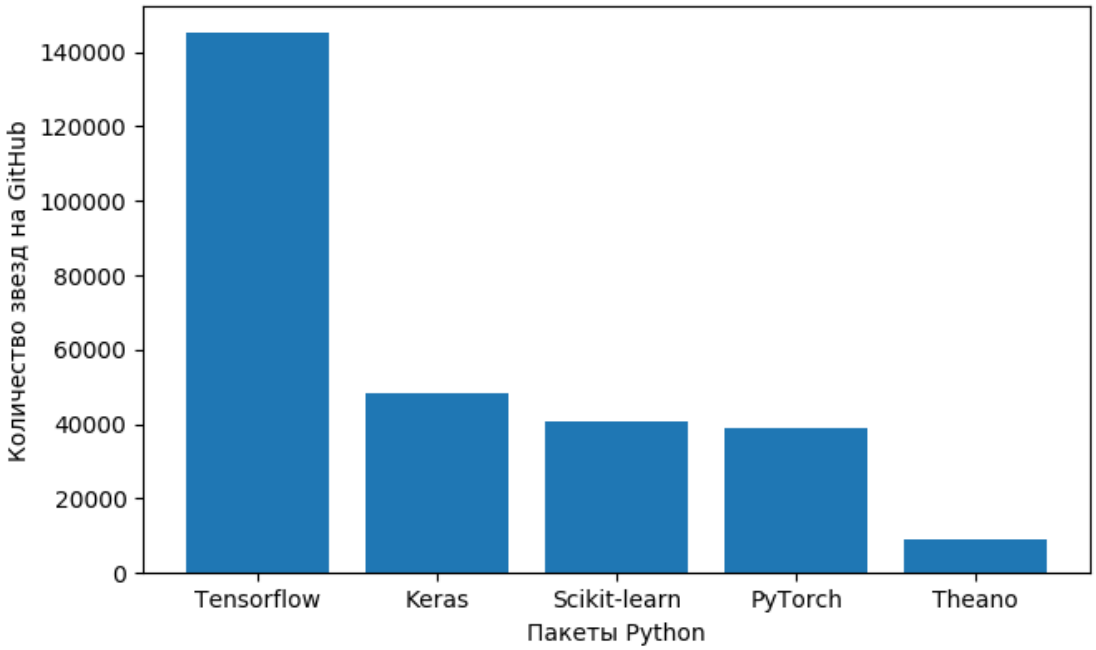

Рисунок 4 – Популярные пакеты Python для машинного обучения по данным рейтинга на GitHub

1.7.1 TensorFlow

Самой популярной и масштабной по применению является библиотека TensorFlow, используемая для глубокого машинного обучения [28]. Библиотека разрабатывается в тесном сотрудничестве с компанией Google и применяется в большинстве их проектов где используется машинное обучение. Библиотека использует систему многоуровневых узлов, которая позволяет вам быстро настраивать, обучать и развертывать искусственные нейронные сети с большими наборами данных.

Библиотека хорошо подходит для широкого семейства техник машинного обучения, а не только для глубокого машинного обучения. Программы с использованием TensorFlow можно компилировать и запускать как на CPU, так и на GPU. Также данная библиотека имеет обширный встроенный функционал логирования, собственный интерактивный визуализатор данных

и логов [36].

1.7.2 Keras

Keras используется для быстрого прототипирования систем с использованием нейронных сетей и машинного обучения. Пакет представляет из себя высокоуровневый API, который работает поверх TensorFlow или Theano. Поддерживает как вычисления на CPU, так и на GPU

1.7.3 Scikit-learn

Scikit-learn – это одна из самых популярных библиотек для языка Python, в которой реализованы основные алгоритмы машинного обучения, такие как классификация различных типов, регрессия и кластеризация данных. Библиотека распространяется свободно и является бесплатной для использования в своих проектах [42].

Данная библиотека создана на основе двух других – NumPy и SciPy, имеющих большое количество готовых реализаций часто используемых математических и статистических функций. Библиотека хорошо подходит для простых и средней сложности задач, а также для людей, которые только начинают свой путь в изучении машинного обучения.

1.7.4 PyTorch

PyTorch – это популяный пакет Python для глубокого машинного обучения, который можно использовать для расширения функционала совместно с такими пакетами как NumPy, SciPy и Cython. Главной функцией PyTorch является возможность вычислений с использованием GPU. Отличается высо-

кой скоростью работы и удобным API-интерфейсом расширения с помощью своей логики, написанной на C или C++.

1.7.5 Theano

Theano – это библиотека, в которой содержится базовый набор инструментов для машинного обучения и конфигурирования нейросетей. Так же у данной библиотеки есть встроенные методы для эффективного вычисления математических выражений, содержащих многомерные массивы [42].

Theano тесно интегрирована с библиотекой NumPy, что дает возможность просто и быстро производить вычисления. Главным преимуществом библиотеки является возможность использования GPU без изменения кода программы, что дает преимущество при выполнении ресуркоемких задач. Также возможно использование динамической генерации кода на языке программирования C [31].

2 Проектирование и разработка программы для обработки данных компьютерного моделирования биотканей

Для имеющихся результатов компьютерного моделирования биотканей необходимо было разработать программное обеспечение, которое позволяло бы определять насколько смоделированные данные соответствуют реальным, а так же для возможности проводить тесты на данных реальных пациентов в будущем.

Для решения данной задачи хорошо подходят методы машинного обучения, а в частности задача классификации, т.к. имея некоторый набор данных, можно обучить модель и настроить параметры для более точной работы в дальнейшем. При обучении модели с помощью метрики точности определения класса можно будет судить о качестве обучения.

2.1 Формат входных данных для программы

В работе использовались данные компьютерного моделирования яркостной температуры молочных желез больных и здоровых пациентов. Данные были представлены в виде девяти значений температуры на поверхности кожи и девяти значений внутренней температуры, согласно методике обследования методом радиотермометрии [4] [1]. Схема расположения точек при замере температур представлена на рисунке 5 [24]. Отдельным атрибутом является класс модели. Для здоровых моделей значения класса было равно нулю, а для больных – единице. Исходя из количества классов, классификацию в данной работе можно считать бинарной.

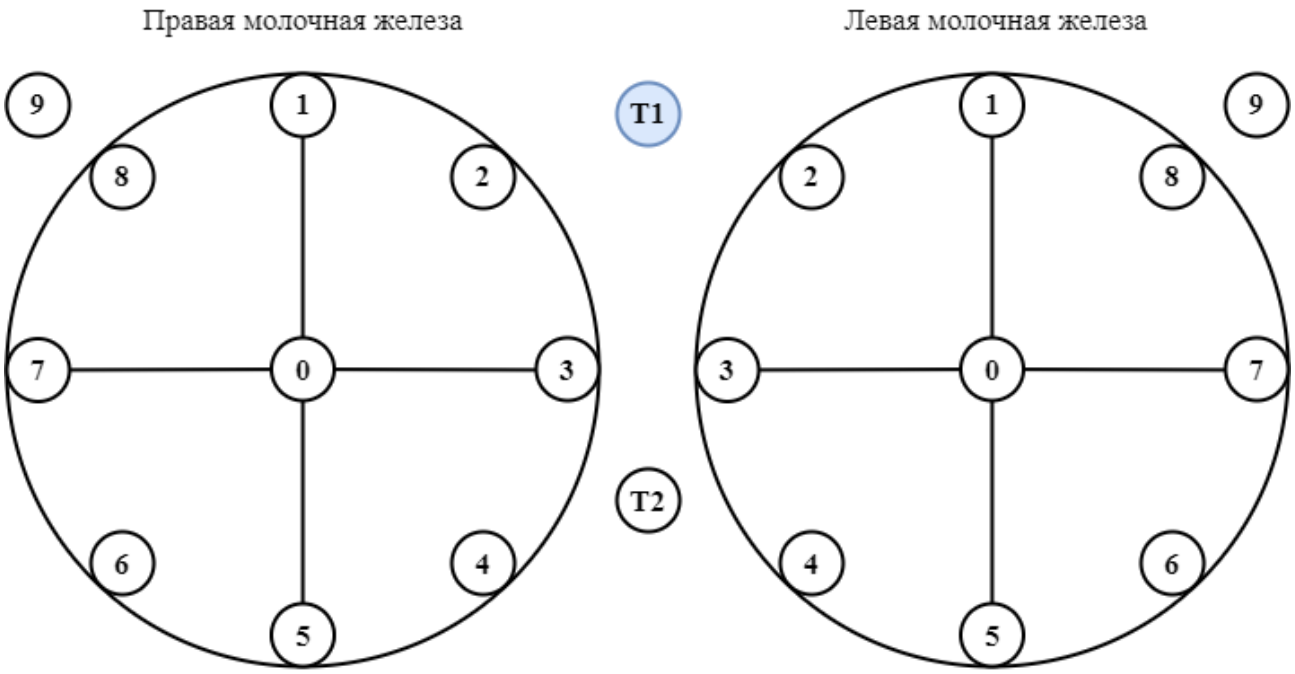

Рисунок 5 – Схема расположения точек при замере температур методом РТМ

Для исследования были взяты температурные данные моделей с радиусом опухоли 0.5 см и 0.75 см. Данные были представлены в виде CSV-файла, в котором находилось по 160 моделей для каждого размера опухоли соответственно (всего 320 моделей) (рисунок 6).

0ртм	1ртм	2ртм	3ртм	4ртм	5ртм	6ртм	7ртм	8ртм	0ик	1ик	2ик	3ик	4ик	5ик	6ик	7ик	8ик	target	point
35,5	35,5	35,6	35,4	34,9	35,2	35,1	35,3	35,6	32,3	33,9	33	33,4	33,4	33,1	32,5	33,2	34,4	0	10
34,4	33,5	33,4	33,5	34,2	34,1	34,4	34,1	34,6	32,2	32	31,9	31,4	32,6	32,8	31,9	32	32,8	0	10
35,7	34,6	34,5	34,4	34,9	34,2	34,2	33,9	34,5	33,8	33,1	32,6	33,2	33,4	32,2	31,8	31,6	32,6	1	0
33,3	32,7	32,9	33,5	34,1	33,5	32,8	33,4	33,8	31,6	32,3	31,7	32,4	32,7	31	30,6	31,4	32,6	1	3
33,6	32	33,3	32,5	32,6	33,5	33	33	32,6	31,8	32,2	33,4	30,9	31,9	31,5	31	31,4	30,8	1	5
34,4	34,4	34,4	34,6	34,3	33,5	33,3	33,5	34,2	33,1	33,2	33	33,3	32,5	32,3	31,8	31,8	32,5	1	4
34,2	33,5	33,8	33,5	33,8	34,1	33,5	33	33,3	32,1	31,5	31,8	30,6	31,1	31,3	30,9	30,3	30,4	1	0
33,1	32,9	33,3	33	33,7	33,3	32,5	32	32,4	31,4	31,7	31,7	31,5	32,1	31,5	31,1	30,5	30,1	1	4
35,5	34,8	34,7	34,7	35,1	35	34,6	34,7	34,8	33,6	33,3	33,3	33,9	33,9	33,4	33,1	32,9	33,4	1	0
33,6	32	32,2	32,7	33	32,9	31,9	32,5	33,5	31,6	30,6	30,7	31,4	31,6	31,4	30,3	31,9	31,7	1	8
36	35,4	35,6	35,7	35,7	35,4	34,9	34,8	35,1	34,3	34,3	34,7	34,4	34,5	33,8	33,1	32,5	33,2	1	2
35,5	34,1	34,5	35,2	34,6	34,9	34,4	33,7	35,4	34,2	33,3	33,6	34	33,8	33,6	32,7	32,2	34,7	1	8
31,5	31,8	32,4	33,6	33,2	33	32	31,7	31,7	29,6	29,1	29,4	30,9	31	30,4	29,7	29,3	28,8	1	3
35	33,7	34	34,4	34	33,9	34,1	33,3	33,6	33,2	32,5	32,3	32,6	32,4	32,5	31,8	31,5	32,2	1	0
31,8	31,9	34,1	32,3	33,3	32,9	32,6	32,8	31,8	29,9	29,9	32,3	30	31,6	30,5	29,9	29,4	30,5	1	2
33,5	33,1	32,8	33,6	33,5	34,2	33,1	33,3	33,3	31,2	29,9	30,4	31	31,3	31,2	30,1	30,2	29,9	1	4

Рисунок 6 – Пример данных температурных данных компьютерного моделирования, где в столбце «target» здоровые — «0», больные — «1»

Одна половина моделей состоит из здоровых пациентов, а другая из больных. В столбце point находятся данные о том, к какой точке ближе всего располагается опухоль. Эти данные можно использовать для локализации

опухоли в дальнейшем. На подготовительном этапе данные были разбиты на обучающую и тестовую выборки. По умолчанию тестовая выборка бралась как 25% от всех данных.

2.2 Проектирование структуры программы и интерфейса

В разрабатываемой программе должна быть возможность определить данные для обучения и классификации, обучить модель с заданными параметрами и методами. Для каждого метода необходимо после обучения и классификации тестовой выборки расчитать точность определения класса и показатели информативности диагностики.

После обучения модели необходимо дать пользователю возможность протестировать ее на данных пациента и показать результат в виде класса и точки с опухолью. Для контроля хода обучения и последующей корректировки параметров обучения будет полезным отображение статистических данных о выборках и графики с точностью классификации.

После определения того, что может сделать пользователь в программе и что он увидит в результате, была разработана диаграмма деятельности (рисунок 7). Данная диаграмма будет полезной как при разработке, так и при тестировании, т.к. содержит последовательную схему действий пользователя.

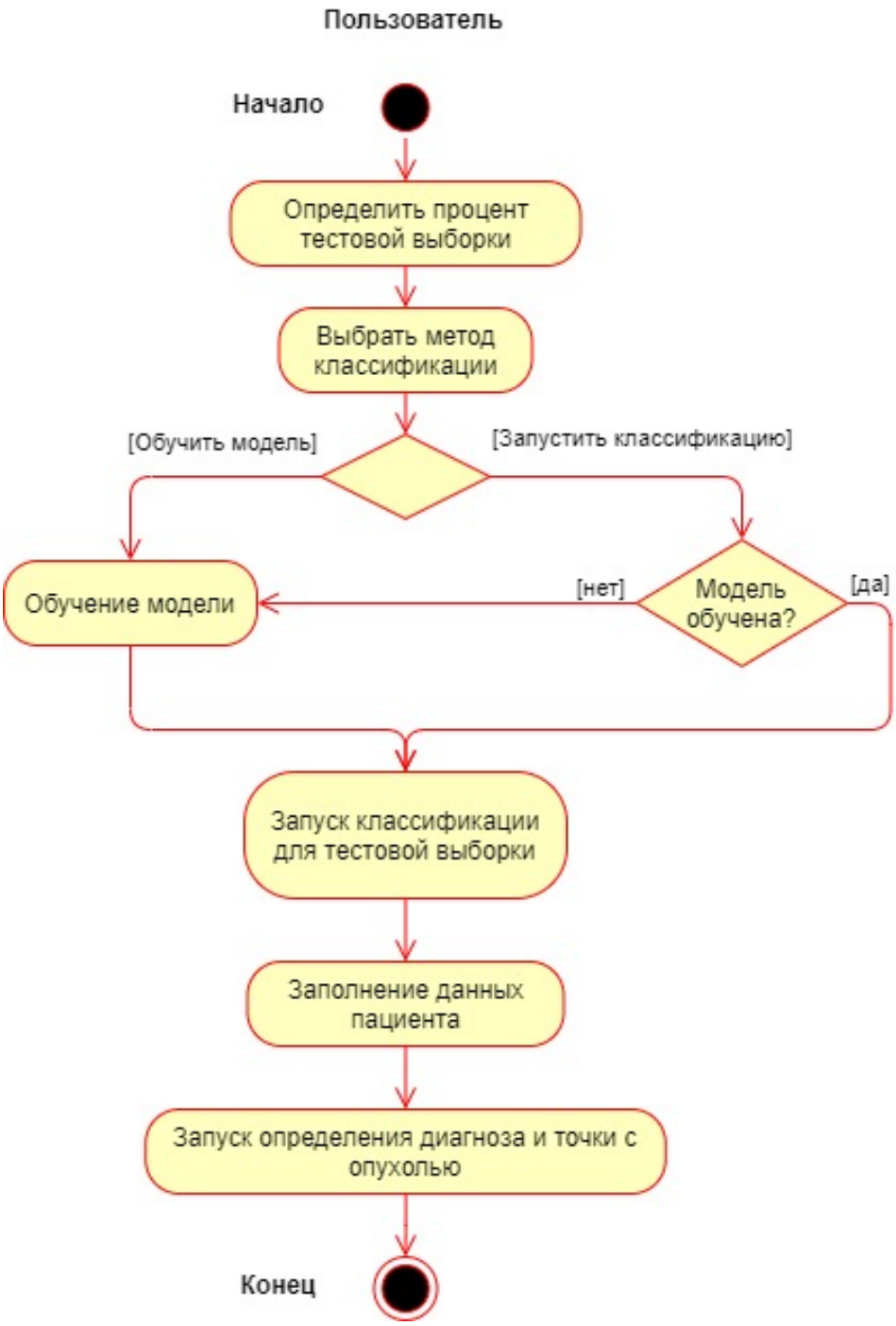

Рисунок 7 – Диаграмма деятельности для программы

После определения возможных действий пользователя был разработан макет интерфейса программы (рисунок 7). При разработке макеты были учтены требования по возможностям настройки и контроля процесса обучения моделей.

Рисунок 8 – Макет интерфейса программы

Интерфейс программы разделен на две части: слева находятся элементы управления для настройки параметров обучения, справа – результаты обучения и статистические данные по выборкам. Для каждого метода классификации будет использован одинаковый набор полей с настройками и одинаковый набор графиков. В блоке с графиками результаты обучения представлены как круговые диаграммы с точностью. Так же имеются графики со статистическими данными – частотным распределением опухолей по точкам и распределением температуры по точкам.

2.3 Выбор технологий и архитектуры

Перед началом разработки программы встал вопрос о том, с помощью каких технологий она будет реализована. Если в самом начале выбрать неправильные инструменты для разработки, то в дальнейшем это может сильно

усложнить поддержку программного обеспечения.

В качестве языка программирования был выбран Python и библиотека Scikit-learn, т.к. для них есть множество примеров использования под текущую задачу и обучающих материалов.

Для пользователей программы было бы удобно не иметь копию данных с результатами моделирования, т.к. файл с этими данными может быть достаточно большого размера. Если данных будет слишком много, то модель будет гораздо дольше на таком наборе данных. Исходя из этого было принято решение использовать клиент-серверную архитектуру при разработке. Взаимодействие клиента и сервера можно условно разделить на две части:

- Загрузка данных, обучение и классификация тестовой выборки;
- Определение диагноза пациента и локализация опухоли.

На рисунках 9 и 10 показаны диаграммы последовательности, описывающие взаимодействие клиента и сервера для каждой из частей. Для общения клиента и сервера был выбран протокол HTTP из-за большой поддержки во многих языках программирования, библиотеках и фреймворках.

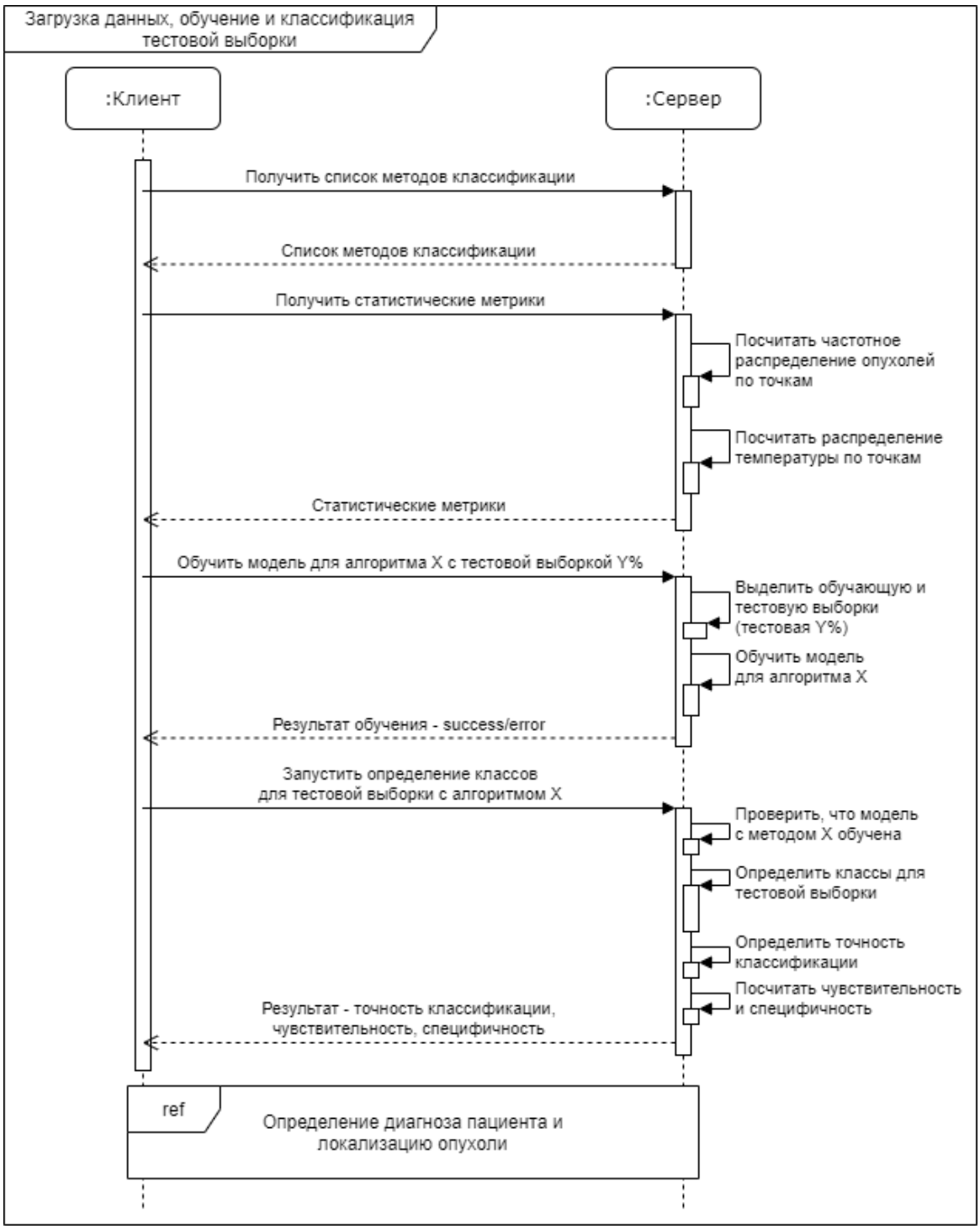

Рисунок 9 – Диаграмма последовательности для этапа загрузки данных, обучения и классификации тестовой выборки

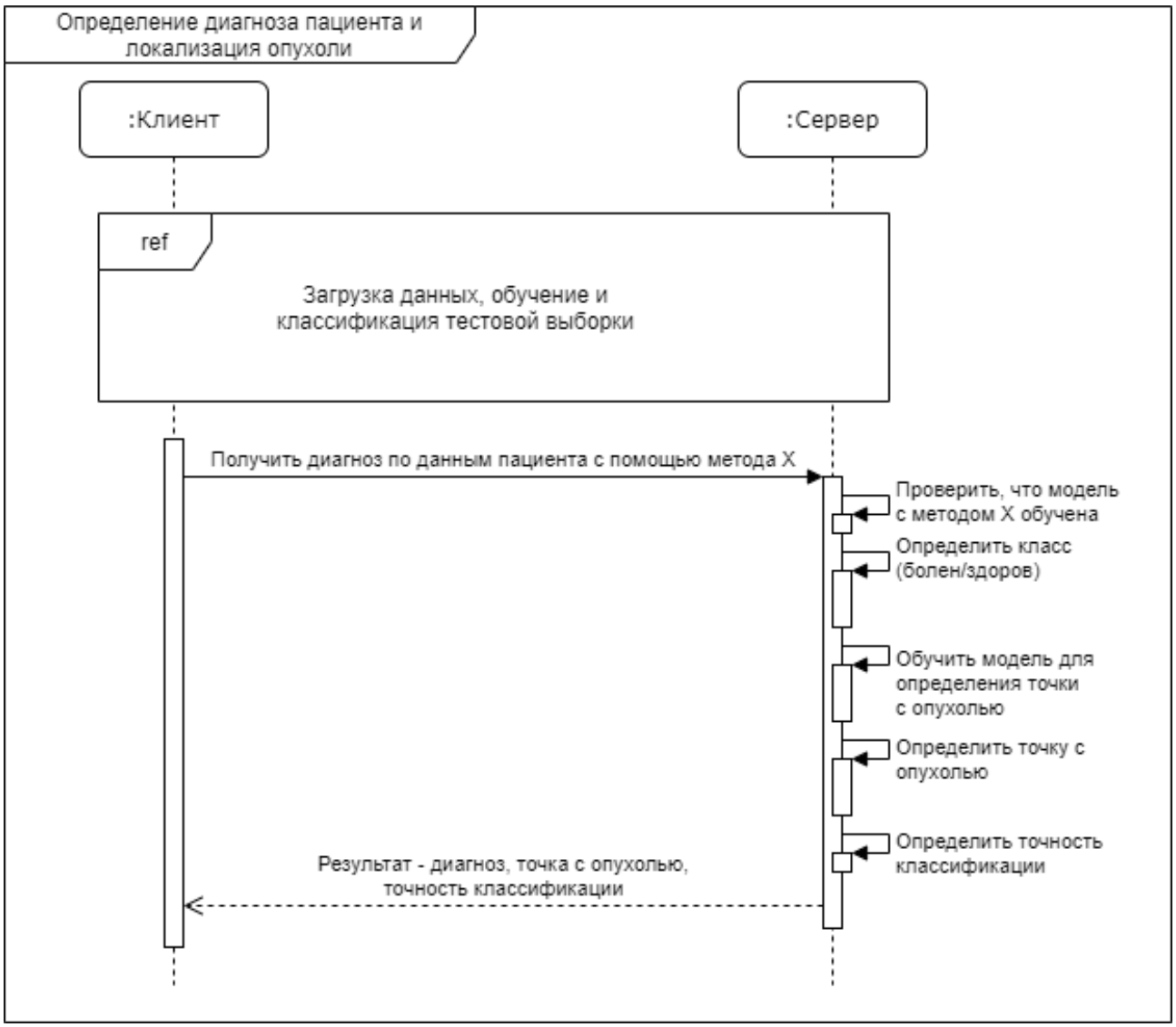

Рисунок 10 – Диаграмма последовательности для этапа определения диагноза пациента и локализации опухоли

В силу простоты реализации и возможности быстрого прототипирования API-интерфейса для бэкенд-части был выбран веб-фреймворк Flask. Flask имеет множество дополнительных библиотек для расширения функционала, а так же подробную документацию.

Для разработки интерфейса рассматривались такие библиотеки для языка Python как Kivy, Tkinter и PyQt. Каждая из них имеет большие возможности для визуализации данных и реализации различных элементов интерфейса. Так же был рассмотрен вариант реализации веб-интерфейса, который и был выбран в дальнейшем из-за возможности его использования на различных типах устройств. Вторым плюсом веб-интерфейса является простота

обновления программного обеспечения в будущем, т.к. такой вариант реализации не требует от пользователя загрузки и установки программы к себе на устройство. Для разработки современного и быстро работающего без перезагрузки страницы интерфейса был выбран язык программирования JavaScript и фреймворк VueJS. Приложение на VueJS состоит из отдельных компонентов, каждый из которых имеет свое состояние и свойства. Такой подход позволяет переиспользовать компоненты и удобно настраивать взаимодействие между ними.

2.4 Разработка программы

На рисунке 11 изображена структура проекта. В отдельных директориях хранятся стили, JS-файлы, шаблоны и тесты.

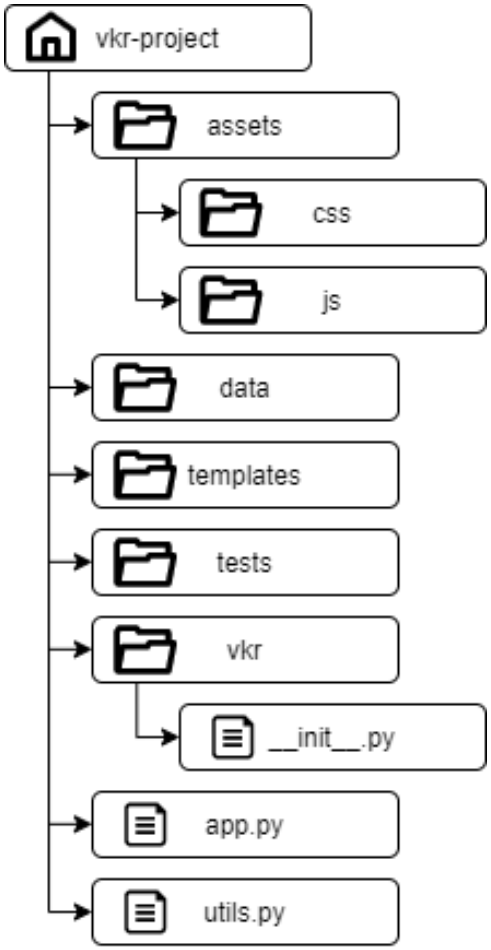

Рисунок 11 – Схема структуры проекта

Разработка программы в рамках данной работы была разделена на этапы, каждый из которых детально описан далее.

2.4.1 Реализация API-интерфейса

Бэкенд-часть представляет собой веб-приложение на Flask. Приложение состоит из одного файла app.py, в котором инициализируется объект приложения с определенными настройками. При запуске файла app.py приложение запускается и становится доступным по адресу http://127.0.0.1:5000/. Также в этом файле описываются все маршруты приложения. В рамках данной работы были реализованы следующие маршруты:

- / – главная страница приложения, с которой и работает пользователь;
- /methods/ – API-метод для получения доступных методов классификации;
- /static_metrics/ – API-метод для получения статистических метрик, рассчитанных для текущей выборки данных;
- /upload_data/ – API-метод для загрузки CSV-файла с температурными данными;
- /train/ – API-метод для обучения модели с выбранным алгоритмом классификации;
- /predict/ – API-метод для запуска классификации для тестовой выборки;
- /diagnose/ – API-метод для получения диагноза пациента по его температурным данным.

Для удобства разработки был создан модуль vkr, где был размещен весь код, связанный с классификацией и обработкой данных.

Чтение файла с данными и инициализация методов для обучения происходит при старте приложения, а так же после загрузки пользователем нового файла.

Для разделения данных на обучающую и тестовую выборки с опре-

деленным соотношением была использована функция train_test_split() из библиотеки Scikit-learn. Во время ее вызова ей необходимо передать массив данных, размер тестовой выборки в процентном соотношении и нужно ли перемешивать данные при разбиении.

Чтобы не обучать модель каждый раз, при первом обучении она сохраняется в файл в бинарном виде. При последующих обращениях к объекту модели, сначала вызывается функция, которая проверяет наличие такого файла. Если файла нет – то модель обучается и он создается. Если файл есть – то данные берутся из него. Это сделано с помощью пакета Pickle. Он позволяет экспортировать в файл и импортировать из файла переменные любых типов.

При определении диагноза пациента происходит еще и определение точки, в которой находится опухоль. При определении точки создается новая модель с алгоритмом многослойной классификации с Перспептроном и обучается на данных только больных пациентов. В качестве классов используются данные из столбца point. Результатом классификации является номер точки. После определения точки расчитывается точность классификации.

Список доступных методов хранится на стороне бэкенда, что позволяет создать универсальный интерфейс на фронтенде для работы с ними.

2.4.2 Разработка фронтенд-части и связь с API

Для разработки фронтенд-части был использован подход работы с VueJS как с библиотекой, подключаемой на странице, т.е. не использовалась сборка Webpack или Vue CLI. Этот выбор связан с небольшим количеством компонентов. Если в будущем при какой-либо масштабной доработке количество компонентов начнет стремительно увеличиваться, то переход на вариант со сборкой не будет большой проблемой, ведь все Vue-компоненты были вынесены в отдельный JS-файл и имеют схожую структуру. В зависимости от того, в каком режиме запущен проект (для разработки или как «боевой») подклю-

чается либо версия VueJS «dev-версия» для разработки, либо «production-версия». Главное отличие этих версий в том, что в «production-версии» отсутствуют инструменты для отладки и минифицирован код, чтобы файл с библиотекой занимал меньше места.

На начальном этапе были выделены Vue-компоненты (рисунок 12). Данные о методах классификации, текущем выбранном методе и статистические данные будут храниться в главном компоненте с названием App.

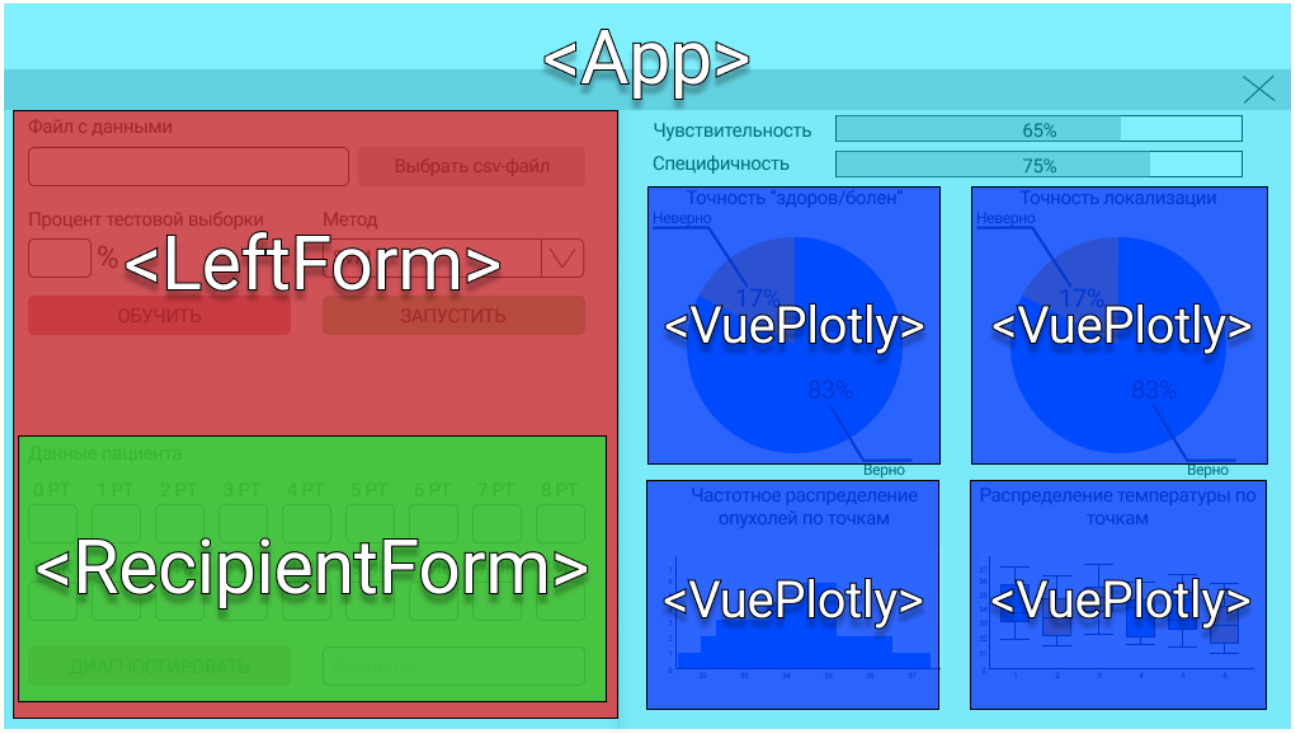

Рисунок 12 – Структура Vue-компонентов

Для построения графиков была использована библиотека Plotly. Данная библиотека позволяет строить различные графики и имеет встроенный функционал для масштабирования и сохранения графиков на компьютер. Но она не имеет встроенную поддержку фреймворка VueJS. Поэтому был разработан компонент для отображения графиков с помощью Plotly.

Фронтенд получает данные отправляя AJAX-запросы к API-методам сервера, т.е. без перезагрузки страницы. Для каждого такого запроса сервер возвращает статус ответа и данные. При любом ответе пользователь увидит либо сообщение об ошибке (рисунок 13), либо с успешным статусом(рисунок 14).

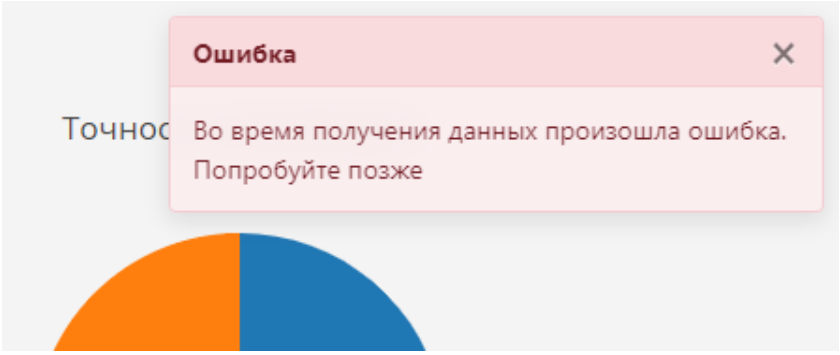

Рисунок 13 – Пример сообщения об операции с ошибкой

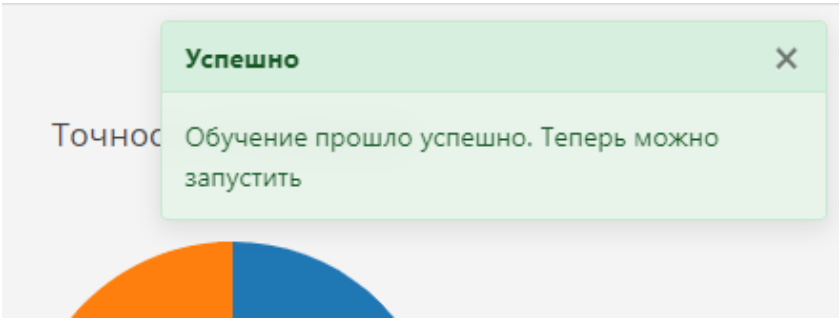

Рисунок 14 – Пример сообщения об успешной операции

Если пользователь нажмет на кнопку "Запустить но при этом модель для выбранного метода не будет обучена, то он получит уведомление как на рисунке 15).

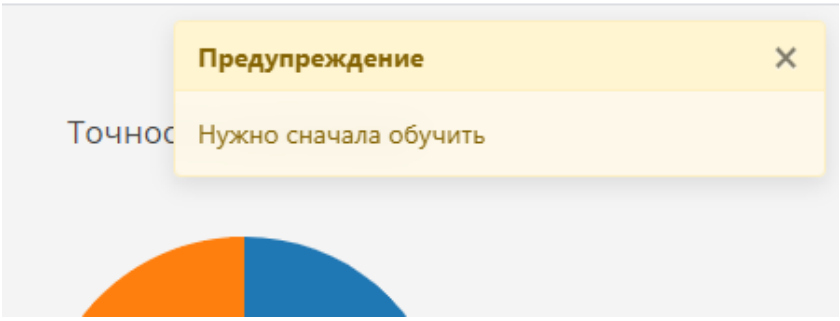

Рисунок 15 – Сообщение, если в момент запуска классификации не была найдена обученная модель

Итоговый вариант интерфейса получившейся программы представлен на рисунке 16.

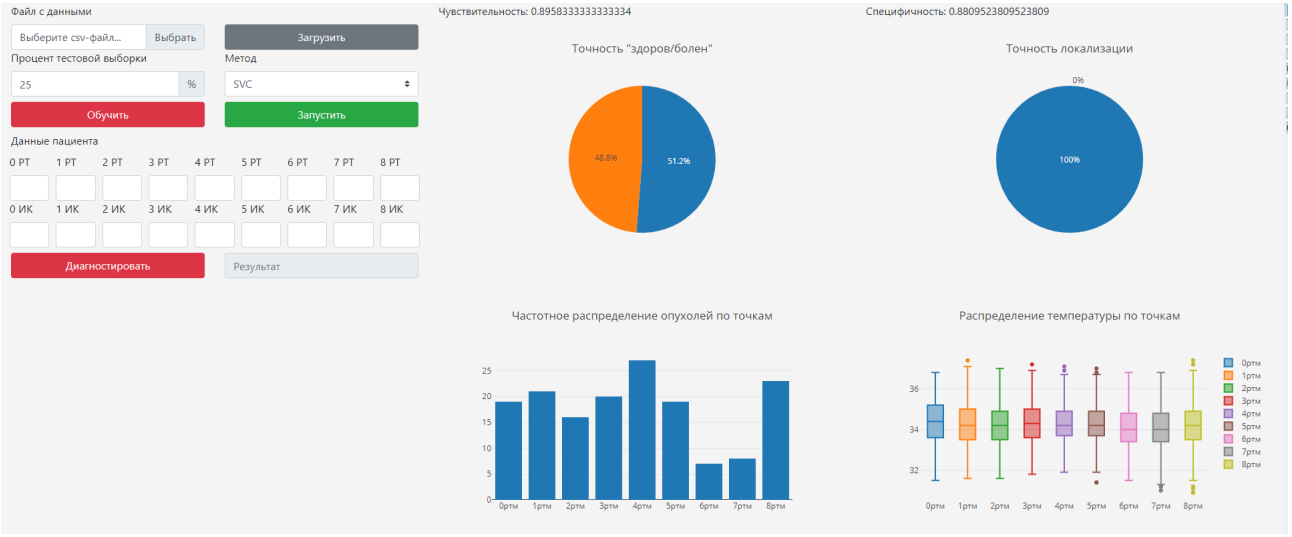

Рисунок 16 – Скриншот получившейся программы

2.4.3 Разработка Unit-тестов

Для контроля правильности работы уже существующих функций и методов приложения во время добавления нового функционала были разработаны Unit-тесты.

Unit-тесты – это набор скриптов, которые в автоматическом режиме проверяют результат работы наиболее частых вариантов вызовов функций с заранее известным результатом. Такой подход широко используется при разработке программного обеспечения. Тесты пишутся разработчиком и запускаются перед переносом нового функционала в «боевое» окружение.

В рамках данной работы были реализованы Unit-тесты для методов расчета чувствительности и специфичности. Так же был разработан набор тестов для методов API, где проверяются статусы и тело ответов при отправке запросов.

Запуск тестов производится с помощью пакета PyTest и запускается командой pytest в командной строке, либо это можно настроить в IDE, в которой происходит разработка. На рисунке 17 показан пример с результатом запуска тестов в IDE PyCharm от компании JetBrains.

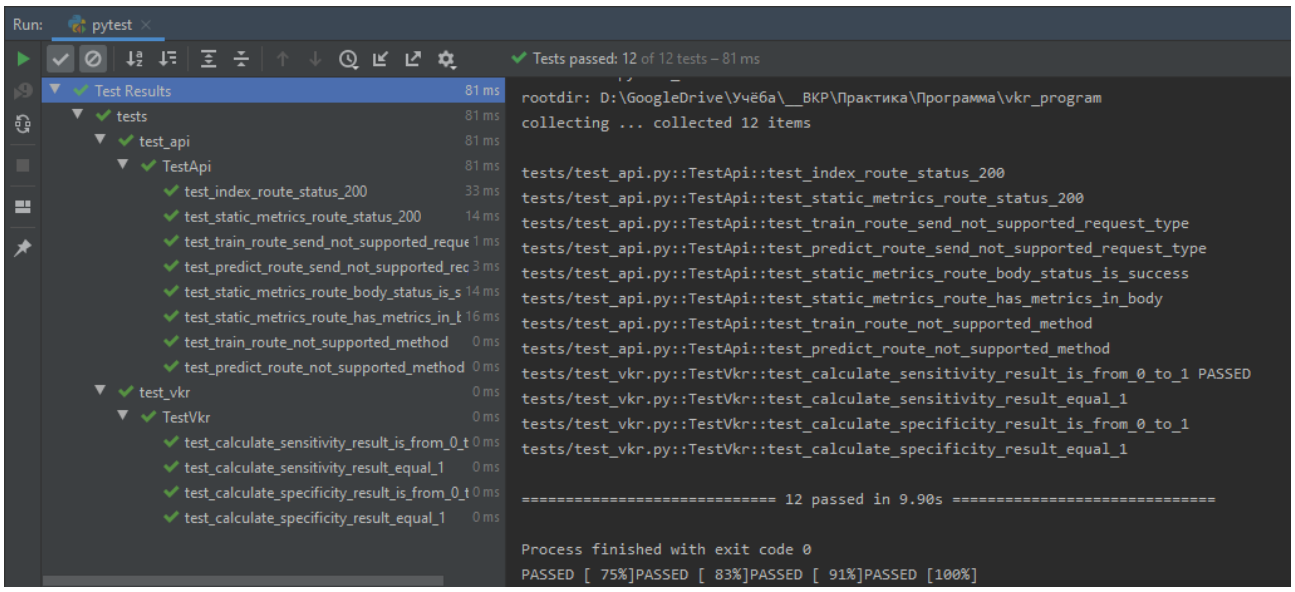

Рисунок 17 – Результат запуска Unit-тестов в IDE PyCharm

3 Применение методов машинного обучения для классификации данных компьютерного моделирования

Работа получившейся программы была протестирована на тестовом наборе данных из общей выборки(рисунок 6). В данной главе расмотрен анализ данных для классификации и этапы работы с разработанной программой.

3.1 Показатели информативности диагностики

Для правильной оценки результатов диагностики существует несколько показателей информативности, которые расчитываются после каждого эксперимента. Одними из интересных для текущей задачи показателями являются чувствительность (5) и специфичность (6).

$$Se = \frac{TP}{TP + FN}, \tag{5}$$

где TP – количество истинно положительных результатов, FN – количество ложноотрицательных результатов.

$$Sp = \frac{TN}{TN + FP}, \tag{6}$$

где TN – количество истинно отрицательных результатов, FP – количество ложноположительных результатов.

Другим важным показателем является точность определения классов, которая расчитывается по формуле (7) как доля правильных ответов алгоритма классификации.

$$Sp = \frac{TP + TN}{TP + TN + FP + FN}, \tag{7}$$

где TP – количество истинно положительных результатов, TN – количество истинно отрицательных результатов, FP – количество ложноположительных результатов, FN – количество ложноотрицательных результатов.

3.2 Частотный анализ классифицируемых данных

Перед классификацией температурных данных был проведен частотный анализ для исследования распределения температуры в зависимости от того, в какой точке она была замерена и есть ли у данной модели опухоль. Частотный анализ был произведен с помощью библиотеки Pandas, а графическое отображение результатов с помощью Matplotlib. Точки, которые были проанализрованы соответствуют точкам на схеме измерений согласно методу РТМ (рисунок 5).

Сначала для исследования была взяла точка 0ртм. Результаты исследования представлены в виде диаграммы частот (рисунок 18), где верхняя диаграмма – это здоровые пациенты, а нижняя – больные.

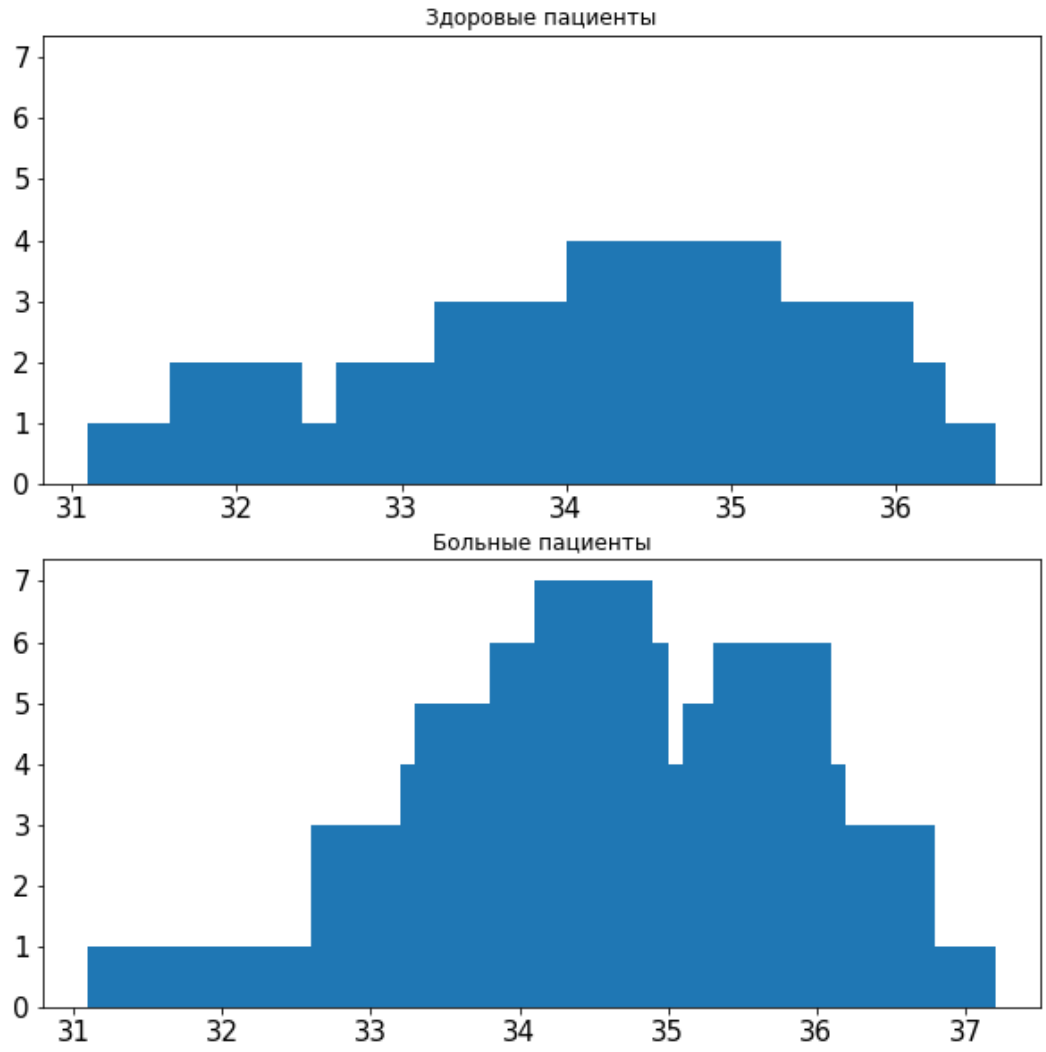

Рисунок 18 – Диаграмма частот температур в точке 0ртм. Верхняя диаграмма — здоровые пациенты, нижняя — больные

Изучив диаграммы, можно заметить, что в данной точке у пациентов чаще встречаются более высокие температуры, нежели у здоровых. Точка 0ртм больше всего подверждена воздействию от опухоли в температурном смысле, так как находится ровно в центре молочной железы.

Следующей для исследования была рассмотрена точка 3ртм, диаграмма частот температур которой представлена на рисунке 19. Для данной точки можно наблюдать схожую картину с точкой 0ртм – тут так же больше значе-

ний со средней температурой у больных пациентов, несмотря на то, что эта точка не является центральной.

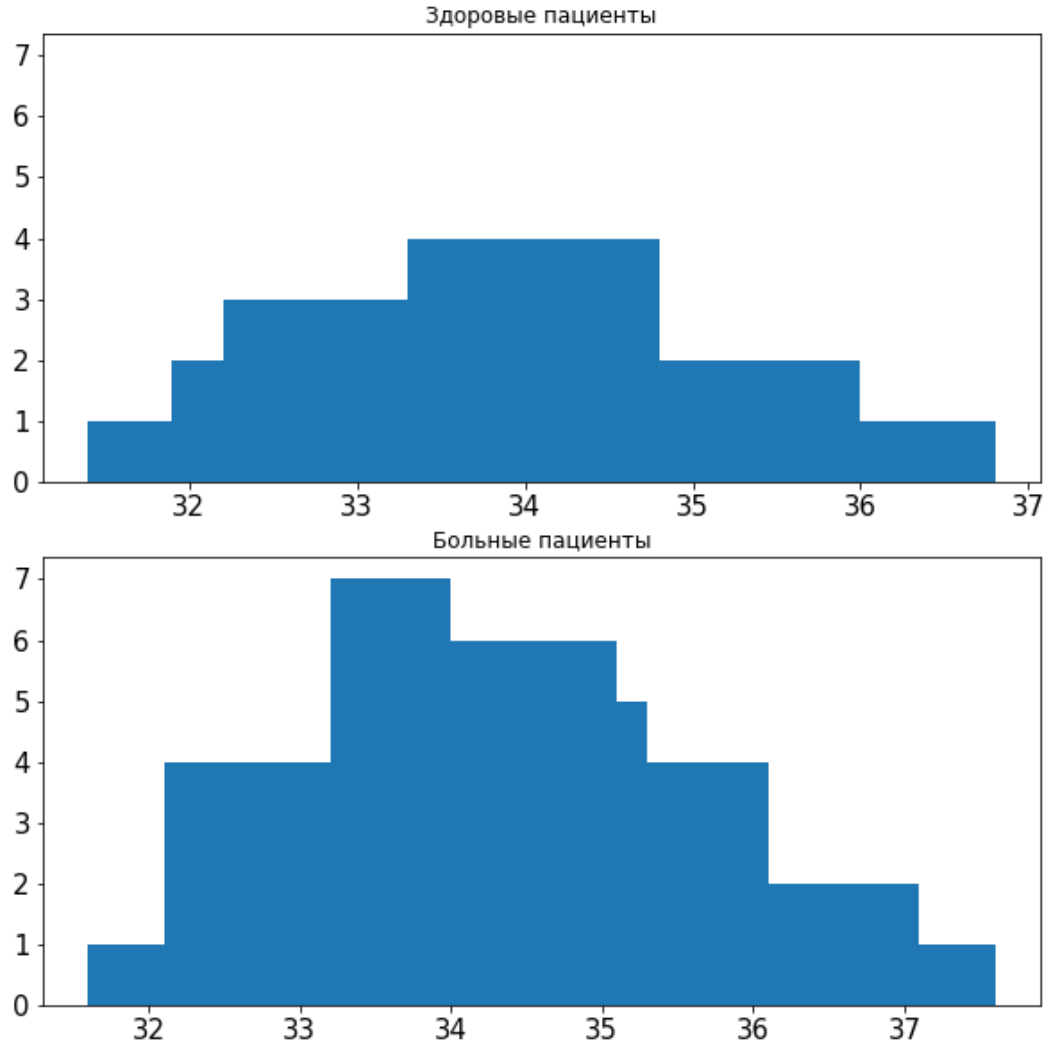

Рисунок 19 – Диаграмма частот температур в точке 3ртм. Верхняя диаграмма — здоровые пациенты, нижняя — больные

Также была рассмотрена точка 7ртм, для которой как видно из диаграммы частот температур (рисунок 20) ситуация аналогична с точками 0ртм и 3ртм.

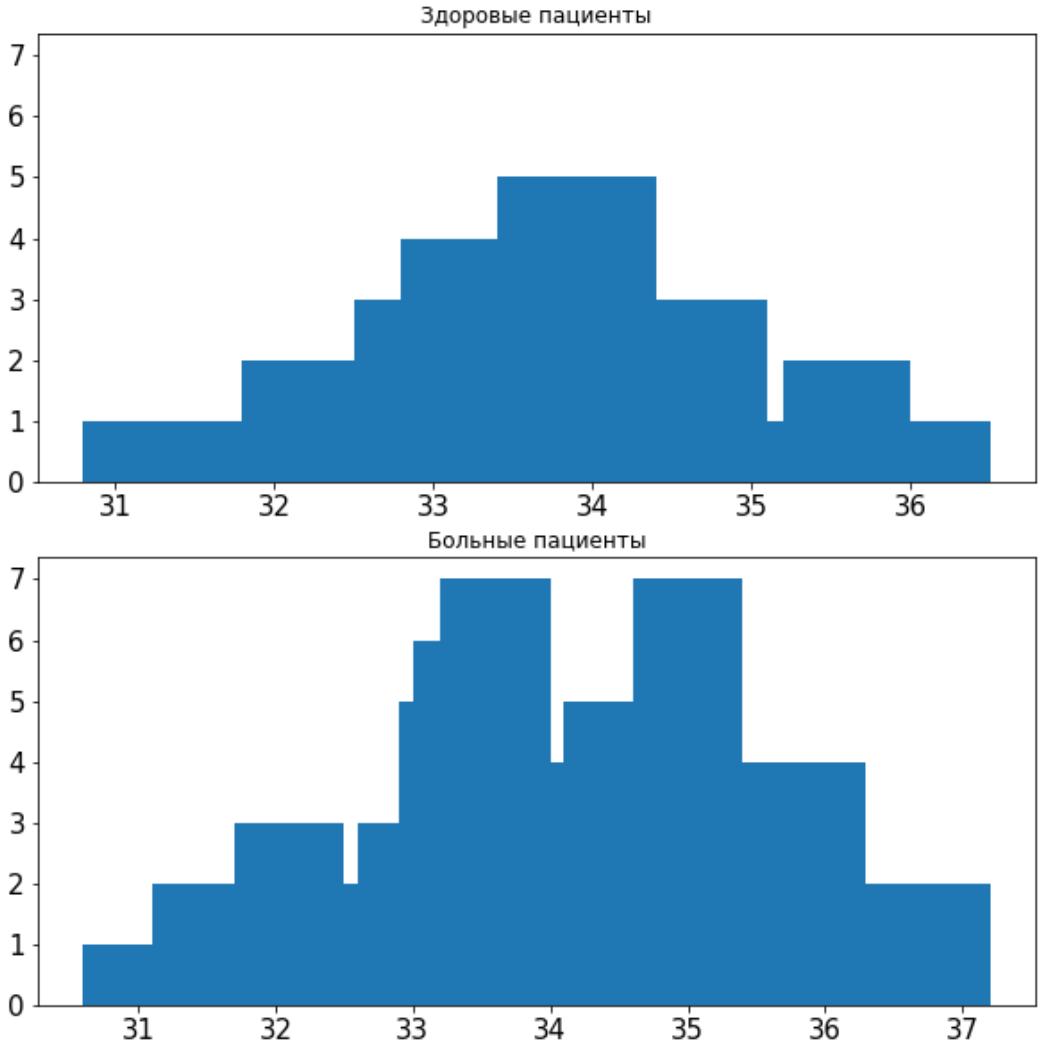

Рисунок 20 – Диаграмма частот температур в точке 7ртм. Верхняя диаграмма — здоровые пациенты, нижняя — больные

Так же в качестве эксперимента к модельным данным были добавлены данные реальных пациентов. Данные были получены путем произведения замеров согласно методу РТМ у реальных людей. В выборке реальных данных представлено 168 пациентов, каждый из которых считается изначально здоровым.

Для определения того, насколько модельные данные близки к реальным они были объединены вместе и перемешаны. Затем получившийся на-

бор данных был разбит на два класса – соответственно модельные и реальные данные. Далее была проведена кластеризация на эти два класса методом k-средних. На рисунке 21 представлены графики распределения температурных точек.

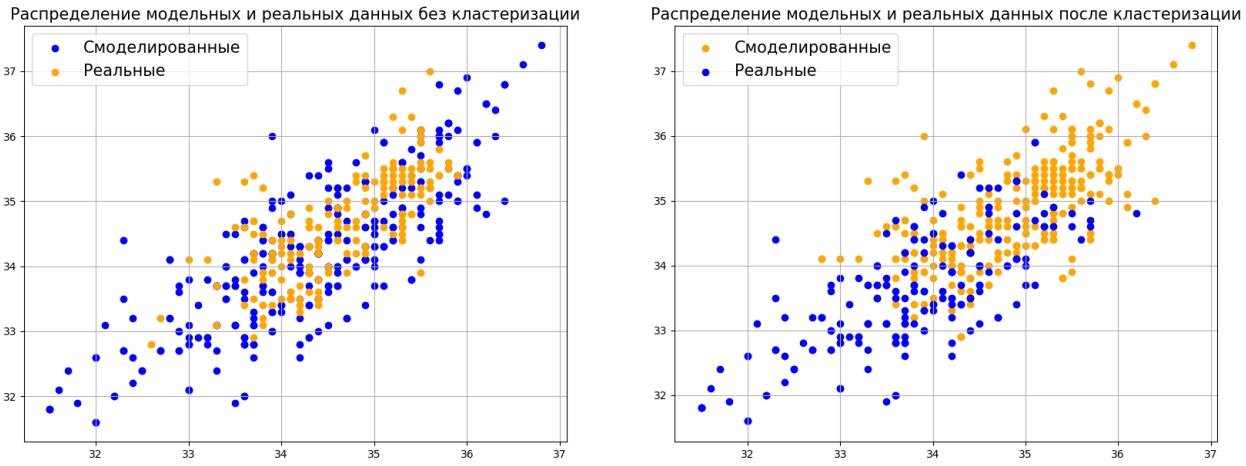

Рисунок 21 – Графики распределения модельных и реальных данных. График слева – распределение до кластеризации, справа – распределение после кластеризации

Так как заранее было известно к какому из двух классов принадлежат данные, возможным было расчитать точность кластеризации по формуле (7). В результате точность кластеризации оказалась равной 68%, что является неплохим результатом для текущего набора данных.

3.3 Обучение модели и классификация тестовой выборки

Первым этапом при работе с программой является загрузка CSV-файла с результатами компьютерного моделирования биотканей. Для этого нужно нажать на кнопку «Выбрать» возле специального поля и выбрать файл на компьютере(рисунок 22).

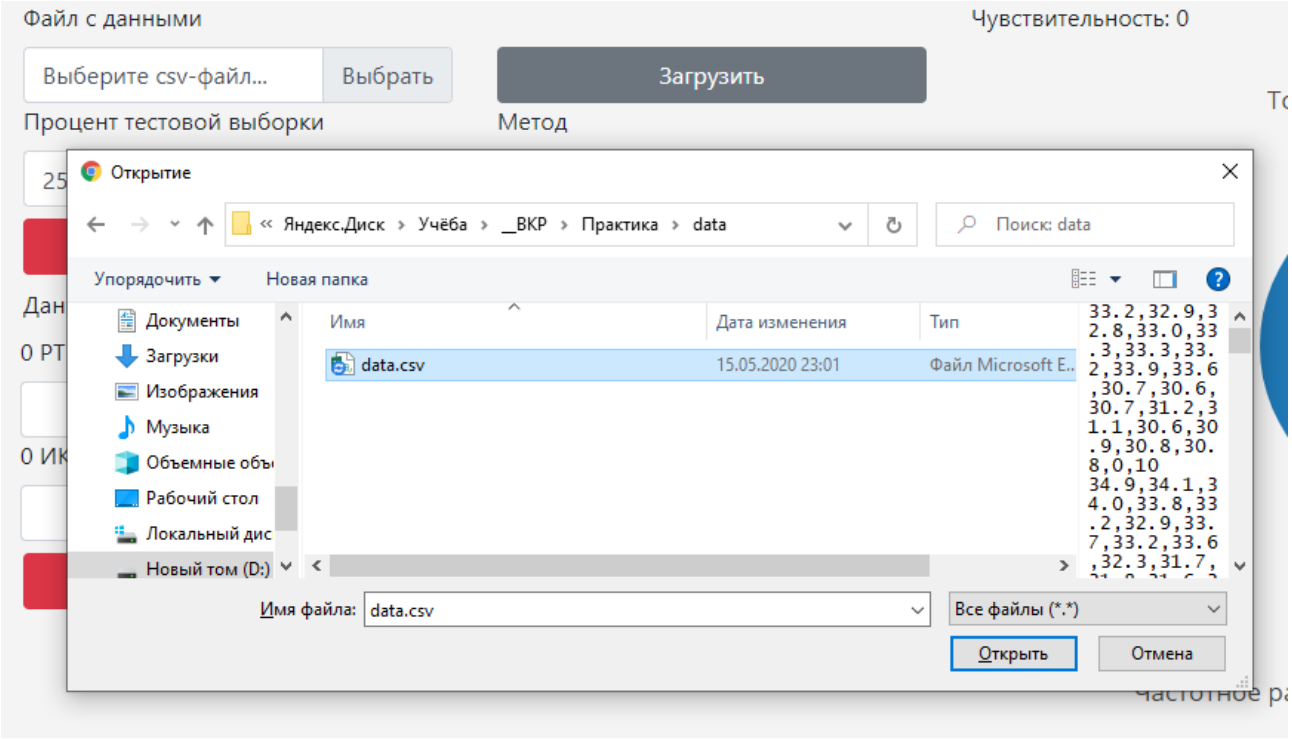

Рисунок 22 – Скриншот интерфейса программы при выборе файла с данными для загрузки

После выбора файла и нажатия на кнопку «Загрузить» файл будет загружен на сервер. Если файл загрузился успешно, то пользователь увидит сообщение как на рисунке 23. После загрузки произойдет переинициализация всех используемых в приложении методов библиотеки Scikit-learn и будут очищены файлы с сохраненными моделями.

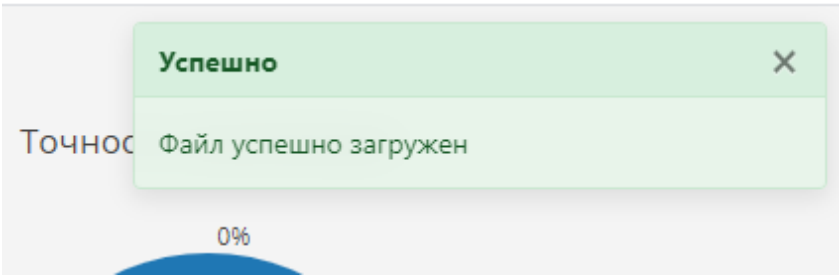

Рисунок 23 – Сообщение об успешной загрузке файла на сервер

Также в результате обновления файла были обновлены данные статистических метрик и отрисованы графики (рисунок 24). Исходя из этих данных можно сделать первые выводе об используемых при обучении данных.

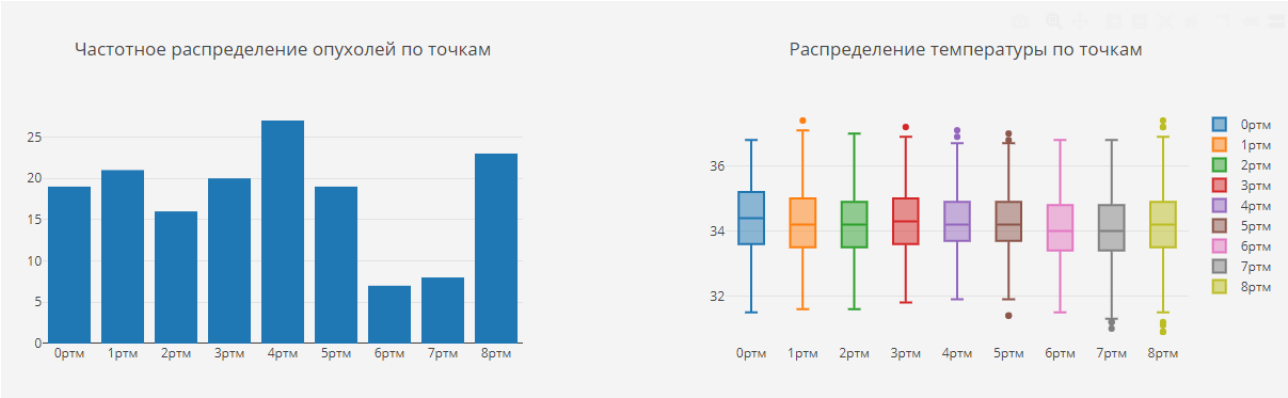

Рисунок 24 – Скриншот графиков со статистическими метриками, вычисленных на основе загруженных данных

На примере графика с частотным распределением опухолей по точкам можно заметить, что больше всего опухолей расположено в крайних точках с номерами 4 и 8, а меньше всего моделей с опухолями в соседних точках с номерами 6 и 7.

Следующим этапом после загрузки файла следует выбор метода классификации из списка, обучение модели и запуск классификации для тестовой выборки. Для всех перечисленных действий есть отдельные элементы управления (рисунок 25).

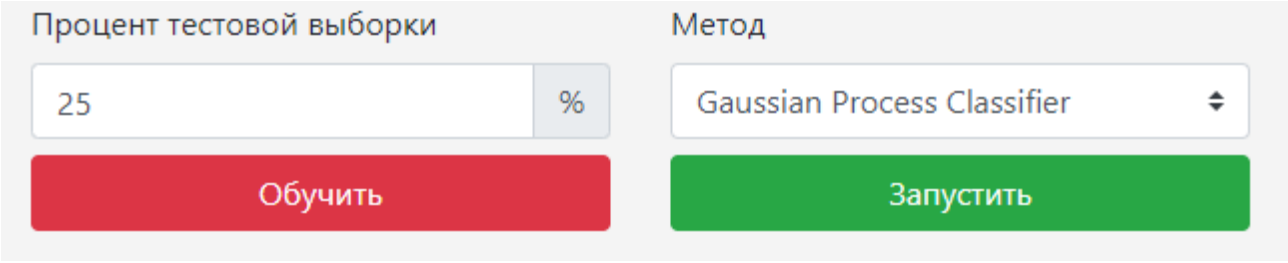

Рисунок 25 – Элементы управления для выбора размера тестовой выборки, обучения модели и классификации тестовой выборки

3.4 Определение класса «Болен»/«Здоров» и точки с опухолью

Результатом обучения и классификации является круговая диаграмма с точностью определения класса «Болен»/«Здоров». На рисунке 26 изображен пример круговой диаграммы с точностью для наивного байесовского классификатора.

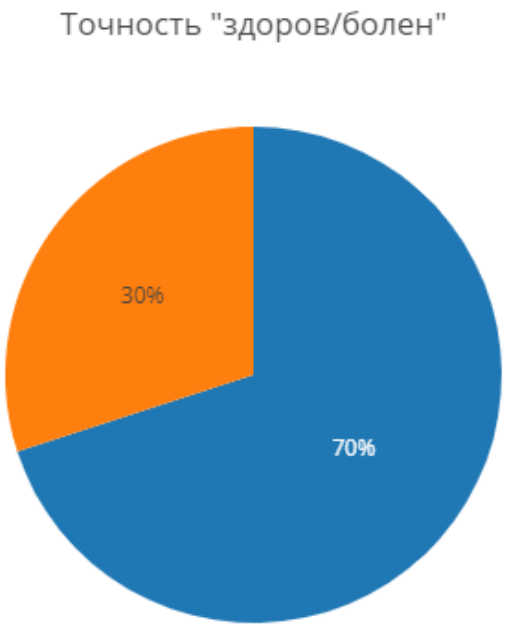

Рисунок 26 – Круговая диаграмма с точностью классификации для наивного байесовского классификатора

3.4.1 Зависимость результата классификации от соотношения обучающей и тестовой выборок

Точность классификации зависит от многих параметров, в частности от размера обучающей и тестовой выборок. Были проведены эксперименты для

разных алгоритмов с различиным процентом тестовой выборки. Результаты данных экспериментов приведены в таблице 1.

Таблица 1 – Результаты классификации с разным соотношением обучающей и тестовой выборок

Алгоритм	Тестовая выборка, %	Точность классификации, %
SVM	5	62.5
	10	56.3
	15	56.3
	20	57.8
k-ближайших соседей	5	87.5
	10	65.5
	15	68.8
	20	67.2
Bagging+SVM	5	62.5
	10	65.6
	15	56.3
	20	59.4
Дерево решений	5	75
	10	68.8
	15	64.6
	20	70.3

Для тестовой выборки равной 5% лучше всего сработал алгоритм k-ближайших соседей с точностью классификации 87.5%. В экспериментах с тестовой выборкой 10% и 15% у всех алгоритмов точность классификации снилизась в среднем на 5-10%. Исходя из этого можно сделать вывод, что чем меньше процент тестовой выборки, тем хуже точность классификации. Для тестовой выборки 20% лучше всего показали себя алгоритмы дерева решений 70.3% и k-ближайших соседей 67.2%. Оптимальным вариантом для дальнейшего использования было выбрано значение 20% для тестовой выборки, так как эксперимент с таким процентом выборки показал себя хуже

чем с 5%, но лучше чем с 10% и 15%.

3.4.2 Зависимость результата классификации от соотношения количества больных и здоровых пациентов в выборке

На точность классификации значительно может влиять соотношение количества больных и здоровых температурных данных пациентов в выборке. Представленные в предыдущем параграфе результаты были получены для выборки с равным количеством больных и здоровых пациентов. Так же были проведены эксперименты для разного соотношения количества больных и здоровых пациентов, которые представлены в таблице 2.

Таблица 2 – Результаты классификации с разным соотношением больных и здоровых пациентов в выборках

Алгоритм	Количество больных, %	Точность классификации, %
SVM	30	62
	50	57.8
	60	56.3
	65	56
k-ближайших соседей	30	73.5
	50	67.2
	60	71.2
	65	54
Bagging+SVM	30	69.4
	50	59.4
	60	55.8
	65	56
Дерево решений	30	74.5
	50	70.3
	60	59.6
	65	52

Исходя из результатов можно заметить, что все алгоритмы показывают лучший результат по точности в выборках с меньшим количеством больных пациентов. Исключением являются эксперименты, где использовался алгоритм k-ближайших соседей, где в трех из четырех случаях точность классификации получилась около 70%.

3.4.3 Определение точки с опухолью

Так же в разработанной программе есть возможность определить класс (диагноз) по данным пациента. После заполнение всех нужных полей и нажа-

тия на кнопку «Диагностировать» буду получены данные о классе и точки, к которой ближе всего расположена опухоль.

Для определения точки с опухолью используется алгоритм классификации с помощью многослойного персептрона. В обучающую выборку входят только данные больных моделей. В качестве класса используется номер точки, в которой расположена опухоль.

На рисунке 27 приведен пример результата дианостирования на температурных данных одной из моделей, не попавшей в обучающую выборку. После определения точки с опухолью строится круговая диаграмма с точностью.

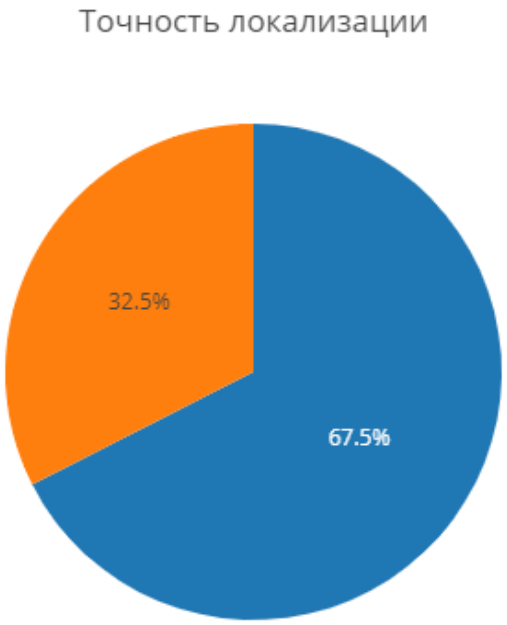

Рисунок 27 – Круговая диаграмма с точностью классификации при определении локализации опухоли

Как видно на диаграмме – получилось добиться точности определения класса равной 67.5%. Возможно этот результат получится улучшить с помощью большего объема обучающей выборки или использования другого алгоритма классификации.

Заключение

В данной работе были рассмотрены сферы деятельности и основные задачи, где используются методы машинного обучения, а также некоторые из популярных библиотек языка программирования Python для решения таких задач. Был описан принцип работы таких алгоритмов классификации как метод опорных векторов (SVM), k-ближайших соседей и наивный байесовский классификатор.

Было реализовано клиент-серверное приложение для для классификации данных компьютерного моделирования яркостной температуры. При разработке использовались такие библиотеки языка Python как Flask и Scikit-learn. При разработке клиентской части использовался фреймворк VueJS. Были разработы Unit-тесты для упрощения доработок программы в будущем. Температурные данные были разбиты на обучающую и тестовую выборки и классифицированы с помощью получившейся программы.

Исходя из результатов классификации моделей был сделан вывод, что точность классификации данных сильно зависит от используемого алгоритма. Лучше всего в проведенных экспериментах себя показал метод опорных векторов (SVM) и k-ближайших соседей.

Список литературы

1. Bardati, F. Modeling the Visibility of Breast Malignancy by a MicrowaveRadiometer / F. Bardati, S. Iudicello. – Текст : непосредственный // Biomed. Engineering. – 2008. – Vol.55 (6). – С. 214-221.

2. Cristianini, T. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods / Nello Cristianini, John Shawe-Taylor. – Текст : непосредственный // Cambridge University Press. – 2000. – 204 с.

3. Crammer, K. On the algorithmic implementation of multiclass kernel-based vector machines / Koby Crammer, Yoram Singe. – Текст : непосредственный // Journal of Machine Learning Research. – 2002. – № 2. – С. 265–292.

4. Fear, K.E. Microwave detection of breast cancer / K.E. Fear, M. Ctuchly. – Текст : непосредственный // IEEE Trans. Microwave Theory Tech. – 2000. – Vol.48 (11). – С. 1854-1863.

5. Hetal, B. An Empirical Evaluation of Data MiningClassification Algorithms / He-tal Bhavsar, Amit Ganatra. — Текст : непосредственный // International Journal of Computer Scienceand Information Security (IJCSIS). – 2016 – № 5. – С. 142–150.

6. Jian, M. A portable breast cancer detection system based on smartphone with infrared camera / Jian Ma, Pengchao Shang, Chen Lu, Safa Meraghni. — Текст : непосредственный // PROCEDIA : Vibroengineering. – 2019 – Vol. 26. – С. 57-63.

7. Kumbhar, S. Comparative Analysis of Classification Algorithms / Vijaykumar S. Kumbhar. – Текст : электронный // NCORTIT. – 2017. – 5 С. – URL: https://www.researchgate.net/publication/313440536, свободный. – Загл. с экрана.

8. Leroy, Y. Non-invasive microwave radiometry thermometry / Y. Leroy, B. Bocquet, A. Mammouni. – Текст : непосредственный // Physiol. Means. – 1998. – Vol.19. – С. 127-148.

9. Levshinskii, V. Verification and Validation of Computer Models for Diagnosing Breast Cancer Based on Machine Learning for Medical Data Analysis / V. Levshinskii, M. Polyakov, A. Losev, A. Khoperskov. – Текст : электронный // Communications in Computer and Information Science. – 2019. – Vol. 1084. – С. 447-460. – URL: https://link.springer.com/chapter/10.1007%2F978-3-030-29750-3_35, свободный. – Загл. с экрана.

10. Mirmozaffari, M. Data Mining Classification Algorithms for Heart Disease Prediction / Mirpouya Mirmozaffari, Alireza Alinezhad, Azadeh Gilanpour. – Текст : непосредственный // n-ternational Journal of Computing Communications & InstrumentationEngg (IJCCIE). – 2017. – 4, № 1 – С. 11-15.

11. Mossina, L. Naive Bayes Classification for Subset Selection / Luca Mossina, Emmanuel Rachelson. – Текст : электронный // Physiol. Means. – 1998. – Vol. 19. – С. 127-148. – URL: https://www.researchgate.net/publication/318560282, свободный. – Загл. с экрана.

12. Nisreen, I. Machine Learning Techniques for Breast Cancer Computer Aided Diagnosis Using Different Image Modalities: A Systematic Review / Nisreen I. Yassin, Shaimaa Omran, Enas M. F. El Houby, Hemat Allam. – Текст : электронный // Computer Methods and Programs in Biomedicine. – 2018. – Vol. 156. – С. 25-45. – URL: https://www.sciencedirect.com/science/article/abs/pii/S0169260717306405?via%3I свободный. – Загл. с экрана.

13. Ongeval, V. Ch. Digital mammography for screening and diagnosis of breast cancer: an overview / Ch. Van Ongeval. – Тест : непосредственный // PubMed PMID. – 2007. – Vol. 90 (3). – С. 163–166.

14. Sherwood, L. Fundamentals of Human Physiolog / L. Sherwood. – Текст : непосредственный // Belmon : Brooks/Cole – 2012. – 720 с.

15. Statnikov, A. A Gentle Introduction to Support Vector Machines in Biomedicine: Theory and methods / Alexander Statnikov, Constantin F.

Aliferis, Douglas P. Hardin. – Текст : непосредственный // World Scientific. – 2011. – 183 с.

16. Stauffer, P.R. Utility of Microwave Radiometry for Diagnostic and Therapeutic Applications of Non-Invasive Temperature Monitoring / P.R. Stauffer, D.R. Rodrigues. – Текст : непосредственный // IEEE BenMAS (Benjamin Franklin Symposium on Microwave and Antenna Sub-systems). – 2014. – С. 1-3.

17. Van Ongeval, Ch. Digital mammography for screening and diagnosis of breast cancer: an overview / Ch. Van Ongeval. – Текст : непосредственный // PubMed PMID. – 2007. – Vol. 90 (3). – С. 163–166.

18. Айвазян , С. Прикладная статистика: классификация и снижение размерности / Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. – Текст : непосредственный // Москва : Финансы и статистика, 1989. – 487 с.

19. Алгоритмы интеллектуального анализа данных. / Текст : электронный // 2015. – URL: https://tproger.ru/translations/top-10-data-mining-algorithms/, свободный. — Загл. с экрана.

20. Барсегян, А. Методы и модели анализа данных: OLAP и Data Mining / А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод. – Текст : непосредственный // Санкт-Петербург : БХВ-Петербург, 2004. – 336 с.

21. Бирюлин, Г. Теплофизический расчет в конечно-элементном пакете COMSOL/FEMLAB : методическое пособие / Г.В. Бирюлин. – Текст : непосредственный // Санкт-Петербург : СПбГУИТМО, 2006. – 75 с.

22. Большие данные и машинное обучение: новые возможности для медицины. / Текст : электронный // 2017. – URL: https://habr.com/ru/company/spbifmo/blog/340668/, свободный. — Загл. с экрана.

23. Вандер Плас, Д. Python для сложных задач. Наука о данных и машинное обучение / Дж. Вандер Плас. – Текст : непосредственный // Санкт-Петербург : Питер, 2017. – 576 с.

24. Веснин, С. Современная микроволновая радиотермометрия молочных желез / С.Г. Веснин, М.А. Каплан, Р.С. Авакян. – Текст : электронный // Маммология/Онкогинекология. – 2008. – №3 – 8 с. – URL: https://elibrary.ru/item.asp?id=11610722, свободный. – Загл. с экрана.

25. Веснин, С. Миниатюрные антенны-аппликаторы для микроволновых радиотермометров медицинского назначения / С.Г. Веснин, М.К. Седанкин. – Текст : электронный // Биомедицинская радиоэлектроника. – 2011. – №10 – С. 51-56. – URL: https://elibrary.ru/item.asp?id=17090823, свободный. – Загл. с экрана.

26. Веснин, С. Разработка серии антенн-аппликаторов для неинвазивного измерения температуры тканей организма человека при различных патологиях / С.Г. Веснин, М.К. Седанкин. – Текст : электронный // Вестник МГТУ им. Н.Э. Баумана. Сер. «Естественные науки». – 2012. – №11 – С. 43-61. – URL: https://elibrary.ru/item.asp?id=20179995, свободный. — Загл. с экрана.

27. Вьюгин, В. Математические основы теории машинного обучения и прогнозирования / Владимир Вьюгин. – Текст : электронный // МЦМНО. – 2013. – 390 с.

28. Гудфеллоу, Я. Глубокое обучение / Гудфеллоу Я., Бенджио И., Курвилль А. – Текст : электронный // Москва : ДМК Пресс. – 2017. – 652 с.

29. Данилов, С. Интеллектуальный анализ данных с использованием системы Rapid Miner / С.В. Данилов. – Текст : электронный // Казанский (Приволжский) федеральный университет. – 2014. – 43 с.

30. Дауни, А. Байесовские модели / Дауни А.Б., пер. с анг. В. А. Яроцкого – Текст : непосредственный // Москва : ДМК Пресс. – 2018. – 182 с.

31. Доусон, М. Программируем на Python / Доусон М. – Текст : непосредственный // Санкт-Петербург : Питер. – 2019. – 416 с.

32. Журавлев, Ю. «Распознавание». Математические методы. Программная система. Практические применения / Журавлев Ю. И., Рязанов В. В., Сенько О. В. – Текст : непосредственный // Москва : Фазис, 2006. – 176 с.

33. Левитин, А. Алгоритмы. Введение в разработку и анализ / Левитин А. В. – Текст : непосредственный // Москва : Вильямс. – 2006. – 576 с.

34. Лосев, А. Проблемы измерения и моделирования тепловых и радиационных полей в биотканях: анализ данных микроволновой термометрии / А.Г. Лосев, А.В. Хоперсков, А.С. Астахов, Х.М. Сулейманова. – Текст : непосредственный // Вестн. Волгогр. гос. ун-та. Сер. 1, Мат. Физ. – 2015. – No 6 – 41 с.

35. МакГрат, М. Алгоритмы. Python. Программирование для начинающих / Майк МакГрат. – Текст : непосредственный // Эксмо. – 2013. – 194 с.

36. Мюллер, А. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / Андреас Мюллер, Сара Гвидо. – Текст : непосредственный // Вильямс. – 2017. – 480 с.

37. Николенко, С. Алгоритмы. Глубокое обучение / Николенко С., Кадурин А., Архангельская Е. – Текст : непосредственный // Санкт-Петербург : Питер. – 2018. – 480 с.

38. Паклин, Н. Бизнес-аналитика: от данных к знаниям : Учебное пособие / Паклин Н.Б., Орешков В.И. – Текст : непосредственный // Санкт-Петербург : Питер, 2013. – 2-е изд. – 704 с.

39. Поляков, М. Математическое моделирование пространственного распределения радиационного поля в биоткани: определениеяркостной температуры для диагностики / М.В. Поляков, А.В. Хоперсков. – Текст : непосредственный // Вестн. Волгогр. гос. ун-та. Сер. 1, Мат. Физ. – 2016. – No 5 – С. 73-84.

40. Потапов, М. Анализ эффективности алгоритмов интеллектуального анализа данных для решения задачи распознавания изображений со спутников / Потапов М. П. – Текст : электронный // Федеральное государственное бюджетное образовательное учреждение высшего образования "Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева": Актуальные проблемы авиации и космонавтики, 2016. – 1. – № 12 – С. 563-565.

41. Рамсундар, Б. Глубокое обучение в биологии и медицине / Рамсундар Б., Истман П., Уолтерс П., Панде В. – Текст : непосредственный // Москва : O'Reilly, 2019. – 202 с.

42. Рашка, С. Python и машинное обучение / Рашка С., пер. с англ. А. В. Логунова. – Текст : непосредственный // Москва : ДМК Пресс, 2017. – 418 с.

43. Розенблатт, Ф. Принципы нейродинамики: Перцептроны и теория механизмов мозга / Розенблатт Ф. – Текст : непосредственный // Москва : Мир, 1965. – 480 с.

44. Флах, П. Машинное обучение / Флах П. – Текст : непосредственный // Москва : ДМК Пресс, 2015. – 400 с.

45. Шлезингер, М. Десять лекций по статистическому и структурному распознаванию / Шлезингер М., Главач В. – Текст : непосредственный // Киев : Наукова думка, 2004. – 546 с.

Приложение А
(обязательное)
Программный код

Листинг А.1 – Код Flask-приложения с API-методами

```
from flask import Flask, jsonify, render_template, request
from flask_cors import CORS
import json
import os
import shutil
from sklearn.metrics import accuracy_score

from vkr import Vkr
from utils import CustomFlask, allowed_file, clear_pickle_fi
les

# instantiate the app
app = CustomFlask(__name__, static_folder="assets")
app.config.from_object(__name__)
app.config['DEBUG'] = True
app.config['DATA_DIR'] = 'data/'

# enable CORS
CORS(app)

# Vkr instance
VkrInstance = Vkr()
VkrInstance.set_data_dir(dir=app.config['DATA_DIR'])
VkrInstance.data_file = 'data.csv'
VkrInstance.check_or_copy_data_file()
VkrInstance.init()


@app.route('/')
def index():
    return render_template('index.html', debug=app.debug, ho
st=request.host_url,
                           filepath=app.config['DATA_DIR'] +
```

```
 VkrInstance.data_file)


@app.route('/train/', methods=['POST'])
def train():
    post_data = json.loads(request.get_data())
    response = {
        'status': 'error',
    }
    if post_data.get('method') in VkrInstance.methods:
        clf = VkrInstance.get_fitted_model(post_data.get('me
thod'), post_data.get('testPercent'), True)
        response['status'] = 'success'
        response['method'] = {
            'code': post_data.get('method'),
            'id': post_data.get('methodId')
        }
    return jsonify(response)


@app.route('/predict/', methods=['POST'])
def predict():
    post_data = json.loads(request.get_data())
    response = {
        'status': 'error',
    }

    if post_data.get('method') in VkrInstance.methods:
        if VkrInstance.need_fit_model(post_data.get('method'
), post_data.get('testPercent')):
            response['status'] = 'warning'
            response['message'] = 'Нужно сначала
 обучить'
        else:
            clf = VkrInstance.get_fitted_model(post_data.get
('method'), post_data.get('testPercent'))
            yPred = clf.predict(VkrInstance.xTest)
            accuracy = accuracy_score(VkrInstance.yTest, yPr
ed)
            response['status'] = 'success'
```

```
            response['metrics'] = {
                'accuracy': accuracy,
                'sensitivity': VkrInstance.calculate_sensiti
vity(yPred) * 100,
                'specificity': VkrInstance.calculate_specifi
city(yPred) * 100
            }
    return jsonify(response)


@app.route('/static_metrics/', methods=['POST'])
def static_metrics():
    try:
        post_data = json.loads(request.get_data())
        response = {
            'status': 'success',
            'metrics': {
                'frequencyTemperature': VkrInstance.get_temp
_freq(),
                'frequencyTumor': VkrInstance.get_tumor_freq
()
            }
        }
    except Exception as e:
        response = {
            'status': 'error',
            'message': e.message
        }

    return jsonify(response)


@app.route('/methods/', methods=['POST'])
def methods():
    formatted_methods = {}
    i = 0
    for key in VkrInstance.methods:
        formatted_methods[i] = {
            'name': VkrInstance.methods[key]['name'],
            'code': key,
```

```
            'canPredict': True,
            'metrics': {
                'sensitivity': 0,
                'specificity': 0,
                'accuracy': []
            }
        }
        i += 1
    response = {
        'status': 'success',
        'methods': formatted_methods
    }

    return jsonify(response)


@app.route('/diagnose/', methods=['POST'])
def diagnose():
    post_data = json.loads(request.get_data())
    response = {
        'status': 'error',
    }

    if post_data.get('method') in VkrInstance.methods:
        if VkrInstance.need_fit_model(post_data.get('method'
)):
            response['status'] = 'warning'
            response['message'] = 'Нужно сначала
 обучить'
        else:
            diagnose_class, predicted_point, accuracy = VkrI
nstance.get_diagnose(post_data.get('method'),

           post_data.get('testPercent'),

           post_data.get('patientData'))

            response['status'] = 'success'
            response['result'] = {
                'class': str(diagnose_class),
```

```
                'point': str(predicted_point),
                'accuracy': accuracy
            }

    return jsonify(response)


@app.route('/upload_data/', methods=['POST'])
def upload_data():
    file = request.files['file']
    if file and allowed_file(file.filename):
        file_path = os.path.join(app.config['DATA_DIR'], Vkr
Instance.data_file)
        # бэкап старого файла
        shutil.copyfile(os.path.join(app.config['DATA_DIR'],
 VkrInstance.data_file),
                  os.path.join(app.config['DATA_DIR'], 'old_
' + VkrInstance.data_file))
        file.save(file_path)

        # Удаляем старые pickle-файлы
        clear_pickle_files(app.config['DATA_DIR'])
        # Перечитаем файл с данными
        # VkrInstance.init()
        print("lol")
        return jsonify({
            'status': 'success',
            'result': {
                'file_path': file_path
            }
        })
    else:
        return jsonify({
            'status': 'error',
            'result': {
                'message': 'Неподходящий тип
файла'
            }
        })
```

```
if __name__ == '__main__':
    app.run(debug=app.debug)
```

Листинг А.2 – Код класса Vkr из бекэнд-части приложения

```
import shutil

from sklearn.model_selection import train_test_split
import pandas as pd
import pickle
import os.path
from sklearn.svm import SVC
from sklearn import neighbors
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.gaussian_process import GaussianProcessClassifi
er
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoos
tClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscrimin
antAnalysis
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from io import StringIO


class Vkr:
    data_dir = 'data/'
    data_file = 'data.cv'
    data = pd.DataFrame()
    temp_columns = ['0ртм', '1ртм', '2ртм', '3рт', '4ртм', '5ртм', '6ртм', '7ртм',
                    '0ик', '1ик', '2ик', '3ик', '4и
к', '5ик', '6ик', '7ик', '8ик']
```

```
    methods = {}

    xTrain, yTrain, xTest, yTest = [], [], [], []

    def init(self):
        self.xTrain, self.xTest, self.yTrain, self.yTest = s
elf.get_train_test_data(
            0.25)
        self.set_methods({
            'svm': {
                'name': 'SVC',
                'clf': SVC(gamma='scale')
            },
            'knn': {
                'name': 'k-ближайших соседей
',
                'clf': neighbors.KNeighborsClassifier(5, wei
ghts='uniform')
            },
            'bagging': {
                'name': 'Bagging meta-estimator + SVM',
                'clf': BaggingClassifier(SVC(gamma='scale'),
 max_samples=0.5, max_features=0.5)
            },
            'sgd': {
                'name': 'Stochastic Gradient Descent',
                'clf': SGDClassifier()
            },
            'gaussianprocessclassifier': {
                'name': 'Gaussian Process Classifier',
                'clf': GaussianProcessClassifier(1.0 * RBF(1
.0))
            },
            'decisiontreeclassifier': {
                'name': 'Decision Tree Classifier',
                'clf': DecisionTreeClassifier(max_depth=5)
            },
            'randomforestclassifier': {
                'name': 'Random Forest Classifier',
```

```
                'clf': RandomForestClassifier(max_depth=5, n
_estimators=10, max_features=1)
            },
            'mlpclassifier': {
                'name': 'MLP Classifier',
                'clf': MLPClassifier(alpha=1, max_iter=1000)
            },
            'adaboostclassifier': {
                'name': 'Ada Boost Classifier',
                'clf': AdaBoostClassifier()
            },
            'quadraticdiscriminantanalysis': {
                'name': 'Quadratic Discriminant Analysis',
                'clf': QuadraticDiscriminantAnalysis()
            }
        })

    def set_data_dir(self, dir):
        self.data_dir = dir

    def set_methods(self, methods):
        self.methods = methods

    def get_train_test_data(self, test_size):
        with open(self.data_dir + self.data_file, "r") as f:
            self.data = pd.read_csv(StringIO(f.read()),
                                    delimiter=',',
                                    names=['0ртм', '1рт
м', '2ртм', '3ртм', '4ртм', '5ртм', '6ртм',
'7ртм', '8ртм',
                                           '0ик', '1ик',
 '2ик', '3ик', '4ик', '5ик', '6ик', '7ик', '8и??', 'target',
                                           'position'])
            f.close()

        return train_test_split(
            self.data[self.temp_columns],
            self.data.target,
            test_size=test_size,
            random_state=0
```

```
        )

    def get_pickled_file_name(self, model_name):
        return self.data_dir + 'model_' + model_name + '_fit
ted.pickle'

    def need_fit_model(self, name, test_percent=25):
        return os.path.isfile(self.get_pickled_file_name(nam
e + '_' + str(test_percent))) == False

    def get_fitted_model(self, name, test_percent=25, need_f
it=False):
        if self.need_fit_model(name, test_percent) or need_f
it:
            self.xTrain, self.xTest, self.yTrain, self.yTest
 = self.get_train_test_data(test_percent / 100)
            self.methods[name]['clf'] = self.methods[name]['
clf'].fit(self.xTrain, self.yTrain)
            with open(self.get_pickled_file_name(name + '_'
+ str(test_percent)), 'wb') as f:
                pickle.dump(self.methods[name]['clf'], f)
        else:
            with open(self.get_pickled_file_name(name + '_'
+ str(test_percent)), 'rb') as f:
                self.methods[name]['clf'] = pickle.load(f)

        return self.methods[name]['clf']

    def calculate_sensitivity(self, yPred):
        sick_test_cnt = len(self.yTest[self.yTest == 1])
        sick_pred_cnt = len(yPred[yPred == 1])
        return sick_pred_cnt / (sick_pred_cnt + abs(sick_tes
t_cnt - sick_pred_cnt))

    def calculate_specificity(self, yPred):
        healthy_test_cnt = len(self.yTest[self.yTest == 0])
        healthy_pred_cnt = len(yPred[yPred == 0])
        return healthy_pred_cnt / (healthy_pred_cnt + abs(he
althy_test_cnt - healthy_pred_cnt))
```

```
    def get_temp_freq(self):
        return self.data[['0ртм', '1ртм', '2ртм', '
3ртм', '4ртм', '5ртм', '6ртм', '7ртм', '8рт
м']].to_dict()

    def get_tumor_freq(self):
        withoutLast = self.data[self.data['position'] != 10]
        return {
            'x': ['0ртм', '1ртм', '2ртм', '3ртм'
, '4ртм', '5ртм', '6ртм', '7ртм', '8ртм'],
            'y': withoutLast[withoutLast['target'] == 1]['po
sition'].value_counts().to_dict()
        }

    def get_diagnose(self, method, test_percent, patient_dat
a):
        clf = self.get_fitted_model(method, test_percent)
        xPredict = []
        for i in patient_data['rt']:
            xPredict.append(i.replace(',', '.'))
        for i in patient_data['rt']:
            xPredict.append(i.replace(',', '.'))
        yPred = clf.predict([xPredict])
        from sklearn.model_selection import train_test_split
        data = self.data[self.data.position != 10]
        xTrain, xTest, yTrain, yTest = train_test_split(
            data[self.temp_columns],
            data.position,
            test_size=0.25,
            random_state=0
        )

        diagnose_class = yPred[0]

        predicted_points = MLPClassifier(alpha=1, max_iter=1
000).fit(xTrain, yTrain).predict(xTest)

        return diagnose_class, predicted_points[0], accuracy
_score(yTest, predicted_points)
```

```
    def check_or_copy_data_file(self):
        if not os.path.isfile(self.data_dir + self.data_file
):
            shutil.copyfile(os.path.join('assets/data.csv'),
                            os.path.join(self.data_dir, self
.data_file))
```

Листинг А.3 – Код шаблонов VueJS-компонентов

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initi
al-scale=1, shrink-to-fit=no">
    <title>BKP</title>
    <link rel="stylesheet" href="%% url_for('static', filena
me='css/bootstrap.min.css') %%">
    <link rel="stylesheet" href="%% url_for('static', filena
me='css/bootstrap-vue.min.css') %%">
    <style>
        body{
            background-color: #F4F4F4;
        }
        .plot{
            max-height: 400px;
        }
    </style>
</head>
<body>
<div id="app">
    <b-container fluid>
        <b-row>
            <b-col cols="4">
                <left-form :methods="methods" :patient-resul
t="patientResultFormatted"></left-form>
            </b-col>
            <b-col cols="8">
```

```
                <b-row>
                    <b-col cols="6">Чувствительн
ость: {{ sensitivity | toFixed }}%</b-col>
                    <b-col cols="6">Специфичност
ь: {{ specificity | toFixed }}%</b-col>
                </b-row>
                <b-row>
                    <b-col cols="6">
                        <vue-plotly :data="mainAccuracyData"
 :layout="mainAccuracy.layout" uid="mainAccuracyPie" class="
plot"/>
                    </b-col>
                    <b-col cols="6">
                        <vue-plotly :data="diagnoseAccuracyD
ata" :layout="diagnoseAccuracy.layout" uid="diagnoseAccuracy
" class="plot"/>
                    </b-col>
                </b-row>
                <b-row>
                    <b-col cols="6">
                        <vue-plotly :data="frequencyTumor.da
ta" :layout="frequencyTumor.layout" uid="frequencyTumorHisto
gram" class="plot"/>
                    </b-col>
                    <b-col cols="6">
                        <vue-plotly :data="frequencyTemperat
ure.data" :layout="frequencyTemperature.layout" uid="frequen
cyTemperatureBox" class="plot"/>
                    </b-col>
                </b-row>
            </b-col>
        </b-row>
    </b-container>
</div>

<template id="template-left-form">
    <div>
        <b-row>
            <b-col cols="12"><label>Файл с данным
и</label></b-col>
```

```
            <b-col>
                <b-form-file
                        v-model="datafile"
                        placeholder="Выберите csv-ф
айл..."
                        drop-placeholder="Перетащит
е csv-файл сюда..."
                        browse-text="Выбрать"
                        ref="datafile"
                ></b-form-file>
            </b-col>
            <b-col>
                <b-button block @click="submitFile">Загр
узить</b-button>
            </b-col>
        </b-row>
        <b-row>
            <b-col cols="6"><label>Процент тесто
вой выборки</label></b-col>
            <b-col cols="6"><label>Метод</label></b-col
>
            <b-col>
                <b-input-group append="%">
                    <b-form-input v-model="testPercent"></b-
form-input>
                </b-input-group>
            </b-col>
            <b-col>
                <b-form-select v-model="selectedMethod" :opt
ions="methodsForSelect"></b-form-select>
            </b-col>
        </b-row>
        <b-row>
            <b-col class="py-2">
                <b-button variant="danger" @click="onTrainHa
ndler" block>Обучить</b-button>
            </b-col>
            <b-col class="py-2">
                <b-button variant="success" @click="onPredic
tHandler" block>Запустить</b-button>
```

```
            </b-col>
        </b-row>
        <recipient-form :patient-result="patientResult"/>
    </div>
</template>

<template id="template-recipient-form">
    <div>
        <b-row>
            <b-col>
                <label>Данные пациента</label>
            </b-col>
        </b-row>
        <b-row class="px-2">
            <b-col v-for="(item, key) in rt" :key="key" styl
e="padding-left: 5px; padding-right: 5px;">
                <div>
                    <label>{{ key }} PT</label>
                </div>
                <b-input v-model="rt[key]"></b-input>
            </b-col>
        </b-row>
        <b-row class="px-2">
            <b-col v-for="(item, key) in ik" :key="key" styl
e="padding-left: 5px; padding-right: 5px;">
                <div>
                    <label>{{ key }} ИК</label>
                </div>
                <b-input v-model="ik[key]"></b-input>
            </b-col>
        </b-row>
        <b-row class="pt-2">
            <b-col>
                <b-button variant="danger" @click="onDiagnos
eHandler" block>Диагностировать</b-button>
            </b-col>
            <b-col>
                <b-input placeholder="Результат" :v
alue="patientResult" readonly></b-input>
            </b-col>
```

```
        </b-row>
    </div>
</template>

{% if debug %}
    <script src="%% url_for('static', filename='js/vue.js')
%%"></script>
{% else %}
    <script src="%% url_for('static', filename='js/vue.min.j
s') %%"></script>
{% endif %}
<script src="%% url_for('static', filename='js/axios.min.js'
) %%"></script>
<script src="%% url_for('static', filename='js/bootstrap-vue
.min.js') %%"></script>
<script src="%% url_for('static', filename='js/bootstrap-vue
-icons.min.js') %%"></script>
<script src="%% url_for('static', filename='js/plotly-latest
.min.js') %%"></script>

<script>
    window.appHost = '%% host %%';
</script>
<script src="%% url_for('static', filename='js/components.js
') %%"></script>
</body>
</html>
```

Листинг А.3 – Код VueJS-компонентов

```
Vue.component("vue-plotly", {
    props: ["data", "layout", "uid"],
    template: '<div :ref="uid"></div>',
    mounted() {
        Plotly.plot(this.$refs[this.uid], this.data, this.la
yout, {displaylogo: false, responsive: true});
    },
    watch: {
        data: {
```

```
            handler: function () {
                Plotly.react(
                    this.$refs[this.uid],
                    this.data,
                    this.layout,
                    {displaylogo: false}
                );
            },
            deep: true
        }
    }
});

Vue.component('left-form', {
    props: ['methods', 'patientResult'],
    data: function () {
        return {
            importFilePath: '',
            testPercent: 25,
            selectedMethod: 0,
            datafile: null
        }
    },
    computed: {
        methodsForSelect: function () {
            const methods = this.methods;
            return Object.keys(this.methods).map(key => ({va
lue: key, text: methods[key].name}));
        }
    },
    watch: {
        importFilePath: function (newVal) {
            this.$root.$emit('import_file_path_changed', new
Val);
        },
        testPercent: function (newVal) {
            this.$root.$emit('test_percent_changed', newVal)
;
        },
        selectedMethod: function (newVal) {
```

```
            this.$root.$emit('selected_method_changed', newV
al);
        }
    },
    template: '#template-left-form',
    methods: {
        onTrainHandler: function() {
            this.$root.$emit('do_train');
        },
        onPredictHandler: function() {
            this.$root.$emit('do_predict');
        },
        submitFile: function () {
            const formData = new FormData();
            formData.append('file', this.datafile);

            const self = this;
            axios.post('/upload_data/',
                formData,
                {
                    headers: {
                        'Content-Type': 'multipart/form-data
'
                    }
                }
            )
                .then(function () {
                    self.$root.$emit('show_toast', 'Файл
 успешно загружен', 'success');
                    self.$root.$emit('get_metrics');
                })
                .catch(function () {
                    self.$root.$emit('show_toast',
                    'Не удалось загрузить файл', 'error');
                });
        }
    }
});

Vue.component('recipient-form', {
```

```
    props: ['patientResult'],
    data: function () {
        return {
            rt: [],
            ik: []
        }
    },
    computed: {
        patientResultFull: function() {
            return this.patientResult ? this.patientResult :
 '';
        }
    },
    watch: {
        rt: function (newVal) {
            this.$root.$emit('rt_changed', newVal);
        },
        ik: function (newVal) {
            this.$root.$emit('ik_changed', newVal);
        },
    },
    template: '#template-recipient-form',
    created: function() {
        for (let i = 0; i < 9; i++) {
            this.rt.push('');
            this.ik.push('');
        }
    },
    methods: {
        onDiagnoseHandler: function() {
            const empty_rt = this.rt.filter(item => item.len
gth == 0);
            const empty_ik = this.ik.filter(item => item.len
gth == 0);
            if (empty_ik.length || empty_rt.length) {
                this.$root.$emit('show_toast', 'Не все
данные пациента заполнены', 'error');
            } else {
                this.$root.$emit('do_diagnose', {
                    rt: this.rt,
```

```
                    ik: this.ik
                });
            }
        }
    }
});

Vue.filter('toFixed', function (value) {
  return value.toFixed(1);
});

var app = new Vue({
    el: '#app',
    data: function() {
        return {
            importFilePath: '',
            testPercent: 25,
            methods: {
                0: {
                    name: 'svm',
                    code: 'svm',
                    canPredict: true,
                    metrics: {
                        sensitivity: 0,
                        specificity: 0,
                        accuracy: []
                    }
                },
                1: {
                    name: 'k-ближайших соседе
й',
                    code: 'knn',
                    canPredict: true,
                    metrics: {
                        sensitivity: 0,
                        specificity: 0,
                        accuracy: []
                    }
                },
                2: {
```

```
                name: 'Bagging meta-estimator + SVM',
                code: 'bagging',
                canPredict: true,
                metrics: {
                    sensitivity: 0,
                    specificity: 0,
                    accuracy: []
                }
            },
            3: {
                name: 'Stochastic Gradient Descent',
                code: 'sgd',
                canPredict: true,
                metrics: {
                    sensitivity: 0,
                    specificity: 0,
                    accuracy: []
                }
            }
        },
        selectedMethod: 0,
        apiBase: window.appHost ? window.appHost : 'http
://127.0.0.1:5000/',
        apiRoutes: {
            trainData: 'train/',
            predictData: 'predict/',
            staticMetrics: 'static_metrics/',
            diagnose: 'diagnose/',
            methods: 'methods/',
        },
        frequencyTemperature: {
            data: [],
            layout: {
                title: 'Распределение температуры по точкам',
                plot_bgcolor: '#F4F4F4',
                paper_bgcolor: '#F4F4F4'
            }
        },
        frequencyTumor: {
            data: [],
```

```
            layout: {
                title: 'Частотное распрелеление опухолей по точкам',
                plot_bgcolor: '#F4F4F4',
                paper_bgcolor: '#F4F4F4'
            }
        },
        mainAccuracy: {
            data: [
                {
                    values: [],
                    type: 'pie',
                    labels: ['Верно', 'Неверн
o'],
                    showlegend: false,
                    automargin: true
                }
            ],
            layout: {
                title: 'Точность "здоров/болен"',
                plot_bgcolor: '#F4F4F4',
                paper_bgcolor: '#F4F4F4'
            }
        },
        diagnoseAccuracy: {
            data: [
                {
                    values: [],
                    type: 'pie',
                    labels: ['Верно', 'Неверн
o'],
                    showlegend: false,
                    automargin: true
                }
            ],
            layout: {
                title: 'Точность локализации',
                plot_bgcolor: '#F4F4F4',
                paper_bgcolor: '#F4F4F4'
            }
        },
```

```
            patientResult: {
                class: null,
                point: null,
                accuracy: []
            }
        }
    },
    computed: {
        sensitivity: function () {
            return typeof this.methods[this.selectedMethod].
metrics.sensitivity != 'undefined' ? this.methods[this.selec
tedMethod].metrics.sensitivity : 0;
        },
        specificity: function () {
            return typeof this.methods[this.selectedMethod].
metrics.specificity != 'undefined' ? this.methods[this.selec
tedMethod].metrics.specificity : 0;
        },
        mainAccuracyData: function () {
            const values = this.methods[this.selectedMethod]
.metrics.accuracy.length ? this.methods[this.selectedMethod]
.metrics.accuracy : [1, 0];
            const data = [...this.mainAccuracy.data];
            data[0].values = values;
            return data;
        },
        diagnoseAccuracyData: function () {
            const values = this.patientResult.accuracy.lengt
h ? this.patientResult.accuracy : [1, 0];
            const data = [...this.diagnoseAccuracy.data];
            data[0].values = values;
            return data;
        },
        patientResultFormatted: function () {
            return this.patientResult.class ? this.patientRe
sult.class + ', в точке ' + this.patientResult.point :
 '';
        }
    },
    created: function () {
```

```
        this.$on('import_file_path_changed', function (data)
 {
            this.importFilePath = data;
        });
        this.$on('test_percent_changed', function (data) {
            this.testPercent = parseInt(data);
        });
        this.$on('selected_method_changed', function (data)
{
            this.selectedMethod = parseInt(data);
        });
        this.$on('do_train', function () {
            this.doTrain();
        });
        this.$on('do_predict', function () {
            this.doPredict();
        });
        this.$on('do_diagnose', function (data) {
            this.doDiagnose(data);
        });
        this.$on('show_toast', function (message, type) {
            this.showToast(message, type);
        });
        this.$on('get_metrics', function () {
            this.getStaticMetrics();
        });
    },
    mounted: function () {
        this.getMethods();
        this.getStaticMetrics();
    },
    methods: {
        sendRequest: function(endPoint, payload, callback) {
            axios.post(this.apiBase + endPoint, payload)
                .then(callback)
                .catch((error) => {
                    this.showToast(error, 'error');
                });
        },
        doTrain: function () {
```

```
            this.sendRequest(
                this.apiRoutes.trainData,
                {
                    method: this.methods[this.selectedMethod
].code,
                    methodId: this.selectedMethod,
                    testPercent: this.testPercent
                },
                (res) => {
                    if (res.data.status == 'success') {
                        this.methods[res.data.method.id].can
Predict = true;
                        this.showToast('Обучение
                        прошло успешно. Теперь можно запус
тить', 'success');
                    } else {
                        this.showToast('Попробуйте
 позже', 'error');
                    }
                }
            );
        },
        doPredict: function () {
            if (this.methods[this.selectedMethod].canPredict
) {
                this.sendRequest(
                    this.apiRoutes.predictData,
                    {
                        method: this.methods[this.selectedMe
thod].code,
                        testPercent: this.testPercent
                    },
                    (res) => {
                        if (res.data.status == 'success') {
                            this.methods[this.selectedMethod
].metrics['accuracy'] = [res.data.metrics.accuracy, 1 - res.
data.metrics.accuracy];
                            if (typeof res.data.metrics.sens
itivity != undefined) {
                                this.methods[this.selectedMe
```

```
thod].metrics['sensitivity'] = res.data.metrics.sensitivity;
                            }
                            if (typeof res.data.metrics.spec
ificity != undefined) {
                                this.methods[this.selectedMe
thod].metrics['specificity'] = res.data.metrics.specificity;
                            }
                            this.showToast('Данные
                            успешно получены', 'success');
                        } else if (res.data.status == 'warni
ng') {
                            this.showToast(res.data.message,
 'warning');
                        } else {
                            this.showToast('Попробуй
те позже', 'error');
                        }
                    }
                );
            } else {
                this.showToast('Нужно сначала о
бучить', 'warning');
            }
        },
        doDiagnose: function (data) {
            this.sendRequest(
                this.apiRoutes.diagnose,
                {
                    method: this.methods[this.selectedMethod
].code,
                    patientData: data,
                    testPercent: this.testPercent
                },
                (res) => {
                    if (res.data.status == 'success') {
                        this.patientResult.class = res.data.
result.class == 1 ? 'Болен' : 'Здоров';
                        this.patientResult.point = res.data.
result.point;
                        this.patientResult.accuracy = [res.d
```

```
ata.result.accuracy, 1 - res.data.result.accuracy];
                        this.showToast('Данные получены',
                        'success');
                    } else if (res.data.status == 'warning')
 {
                        this.showToast(res.data.message, 'wa
rning');
                    } else {
                        this.showToast('Попробуйте
 позже', 'error');
                    }
                }
            );
        },
        getStaticMetrics: function() {
            this.sendRequest(
                this.apiRoutes.staticMetrics,
                {},
                (res) => {
                    if (res.data.status == 'success') {
                        if (typeof res.data.metrics.frequenc
yTemperature != undefined) {
                            const freqData = [];
                            for (let i in res.data.metrics.f
requencyTemperature) {
                                freqData.push({
                                    y: Object.values(res.dat
a.metrics.frequencyTemperature[i]),
                                    type: 'box',
                                    name: i,
                                    automargin: true
                                });
                            }
                            this.frequencyTemperature.data =
 freqData;
                        }
                        if (typeof res.data.metrics.frequenc
yTumor != undefined) {
                            this.frequencyTumor.data = [{
                                x: Object.values(res.data.me
```

```
trics.frequencyTumor.x),
                                y: Object.values(res.data.me
trics.frequencyTumor.y),
                                type: 'bar',
                                automargin: true
                            }];
                        }
                    } else {
                        this.showToast('Произошла ошибка
                         во время получения статистических
                          данных', 'error');
                    }
                }
            );
        },
        getMethods: function() {
            this.sendRequest(
                this.apiRoutes.methods,
                {},
                (res) => {
                    if (res.data.status == 'success') {
                        if (typeof res.data.methods != undef
ined) {
                            this.methods = res.data.methods;
                        }
                    } else {
                        this.showToast('Произошла ошибка
                         во время получения
                         статистических данных', 'error');
                    }
                }
            );
        },
        showToast: function (message, type) {
            const types = {
                error: {
                    title: 'Ошибка',
                    variant: 'danger'
                },
                warning: {
```

```
                    title: 'Предупреждение',
                    variant: 'warning'
                },
                success: {
                    title: 'Успешно',
                    variant: 'success'
                }
            };
            if (typeof types[type] != 'undefined') {
                this.$bvToast.toast(message, {
                    title: types[type].title,
                    autoHideDelay: 5000,
                    variant: types[type].variant
                });
            }
        }
    }
});
```