

Волгоградский государственный университет
Институт математики и информационных технологий
Кафедра информационных систем и компьютерного моделирования

Работа допущена к защите

Заведующий кафедрой

_____ А. В. Хоперсков

«_____» _____ 2021 г.

Борисовский Егор Иванович

Нейросетевая регрессия экспериментальных данных

Отчет по учебной практике, научно-исследовательской работе

Направление: 09.04.01 – Информатика и вычислительная техника

Студент группы ИВТ_м-201:

Е. И. Борисовский

подпись

Руководитель практики:

С. С. Храпов, к.ф.-м.н., доцент каф. ИСКМ

подпись

Ответственный за организацию практики:

А. В. Хоперсков, д.ф.-м.н., профессор каф. ИСКМ

подпись

Содержание

Введение	4
Глава 1. Методы и области применения машинного обучения и нейросетей	6
1.1 Обзор литературы	6
1.2 Области применения	8
1.3 Популярные библиотеки с реализацией нейронных сетей	9
1.3.1 TensorFlow	10
1.3.2 Keras	10
1.3.3 Scikit-learn	10
1.3.4 PyTorch	11
1.3.5 Theano	11
1.3.6 Обзор существующих решений реализации графического- го интерфейса для задач машинного обучения	12
Глава 2. Разработка программного обеспечения для обработки экспериментальных данных с использованием нейросетевой регрессии	18
2.1 Разработка нейросетевой регрессионной модели	20
2.1.1 Формат входных данных для модели	20
2.1.2 Выбор технологий	22
2.1.3 Структура нейросети	22
2.1.4 Обучение нейронной сети	23
2.2 Архитектура приложения	24
2.3 Разработка frontend-части	28
2.4 Разработка REST API-сервиса	30
2.5 Сборка в pip-пакет	32

Глава 3. Тестирование разработанной библиотеки и публика-	
ция в сервис PyPI	33
3.1 Тестирование библиотеки на алгоритме машинного обучения	
SVM	33
3.2 Настройка автоматической сборки и публикации библиотеки в	
сервис PyPI	35
Заключение	36
Список литературы	37
Приложение А. Листинг разработанной программы	38

Введение

В последние несколько лет в различных сферах жизни человека очень часто используются алгоритмы машинного обучения и нейросети. Они помогают решать совершенно разные задачи - от подбора персональных рекомендаций для просмотра фильмов, до помощи в диагностировании различных заболеваний на ранней стадии. В процессе разработки программного обеспечения с подобным функционалом происходит очень много итераций по настройке модели и подбору параметров для нее. Эти рутинные операции в основном производятся с помощью замены обучающей выборки, изменения значений параметров в коде и перезапуска программы. На некоторых этапах разработки может потребоваться показать свои результаты другому человеку, возможно далекому от деталей реализации получившейся нейросетевой модели и появляется необходимость в простом универсальном пользовательском интерфейсе, который можно было бы к ней подключить и использовать.

В данной работе рассматриваются области применения нейросетей, популярные библиотеки с их реализацией и существующие в данный момент решения на рынке для подключения к ним графического интерфейса.

Так же в работе рассматривается использование результатов измерений температур по методике микроволновой радиотермометрии молочных желез на основе работы специального диагностического комплекса РТМ-01-РЭС [**problemIzmereniyaVolsu**] для моделирования температурных данных с помощью нейросетевой регрессии.

Одной из целей работы является проектирование и разработка программного обеспечения для обработки и моделирования экспериментальных температурных данных с использованием нейросетевой регрессии, а так же разработка библиотеки с реализацией пользовательского графического интерфейса для задач машинного обучения и нейросетей на языке Python. Так же необходимо оформить получившуюся программу в `pip`-пакет и

опубликовать его в сервисе РuPI.

Глава 1

Методы и области применения машинного обучения и нейросетей

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 [polyakovKhoperskov].

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 наличии злокачественной опухоли [vesninMinAntenn].

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 [**vanOngeval**].

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 диагностике 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 [**nisrenml**].

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 подход 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 [**lev-polyakov-losev-hoperskov**].

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 [**ramsundar-deep-learning**].

Статья Jian Ma, Pengchao Shang, Chen Lu, Safa Meraghni «A portable breast cancer detection system based on smartphone with infrared camera» посвя-

щена 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% [**mobile-breast-cancer-detection**].

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.

1.2 Области применения

Использование алгоритмов машинного обучения и нейросетей позволяет решать задачи в различных сферах деятельности человека, таких как недвижимость, сельское хозяйство, экономика, а так же медицина. По данным агентства Frost & Sullivan спрос на разработки, в которых используется машинное обучение в медицине, увеличивается с каждым годом примерно на 40% [**habrbigdatamedicine**]. 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 определение 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 в качестве экспертных систем.

При компьютерном моделировании алгоритмы машинного обучения мо-

гут использоваться для валидации получившихся данных, или прогнозирования течения каких-либо физических процессов.

1.3 Популярные библиотеки с реализацией нейронных сетей

На текущий момент существует множество готовых реализаций нейросетей и алгоритмов машинного обучения, что 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 1.1)

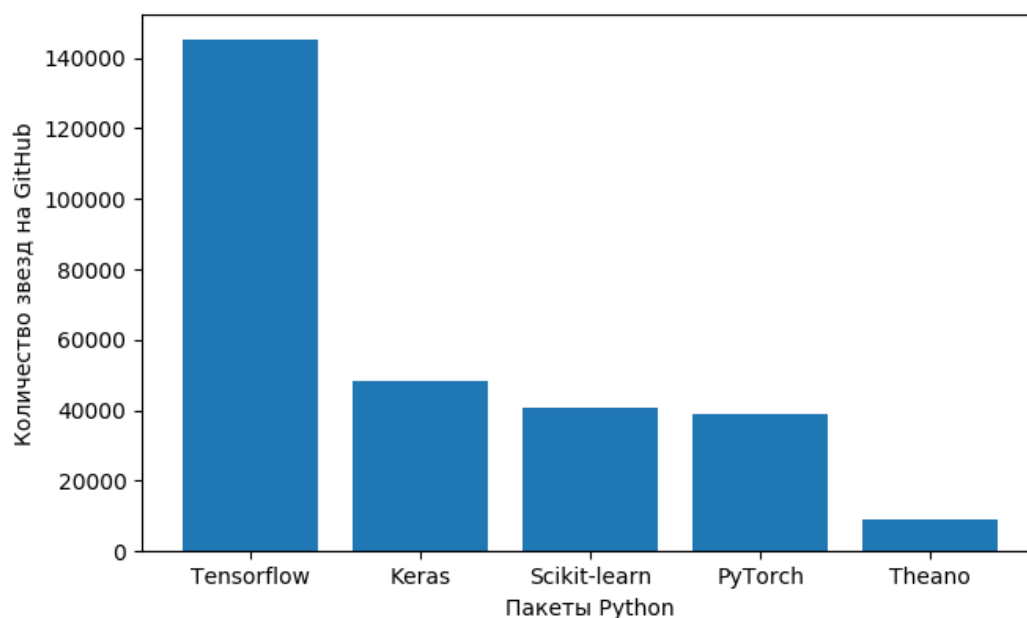


Рисунок 1.1. Популярные пакеты Python для машинного обучения по данным рейтинга на GitHub

1.3.1 TensorFlow

Самой популярной и масштабной по применению является библиотека TensorFlow, используемая для глубокого машинного обучения [gudfellow]. 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 [muller].

1.3.2 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

1.3.3 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 [rashka].

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.

1.3.4 PyTorch

PyTorch – это популярный 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 высокой скоростью работы и удобным API-интерфейсом расширения с помощью своей логики, написанной на C или C++.

1.3.5 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 [**rashka**].

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 ресурсоемких задач. Также возможно использование динамической генерации кода на языке программирования C [**douson**].

1.3.6 Обзор существующих решений реализации графического интерфейса для задач машинного обучения

Прежде чем разрабатывать приложение из данной работы была произведена попытка найти существующие готовые решения для текущей задачи. Были найдены всего лишь два решения: проект на GitHub MachineLearningGUI и программный комплекс Weka. Рассмотрим каждое из них отдельно.

MachineLearningGUI – десктопное приложение, написанное на языке программирования Python с помощью библиотеки PyQt. Работает только с библиотекой Scikit-learn и алгоритмом классификации с деревом принятия решений. Так же автор проекта указал в описании, что приложение работает только с одним набором данных, который идет вместе с проектом. Интерфейс программы состоит из четырех вкладок, каждая из которых отвечает за конкретный шаг: загрузка обучающей выборки, препроцессинг данных, запуск алгоритма и просмотр результатов. На рисунках 1.2 и 1.3 представлены первый и третий шаги.

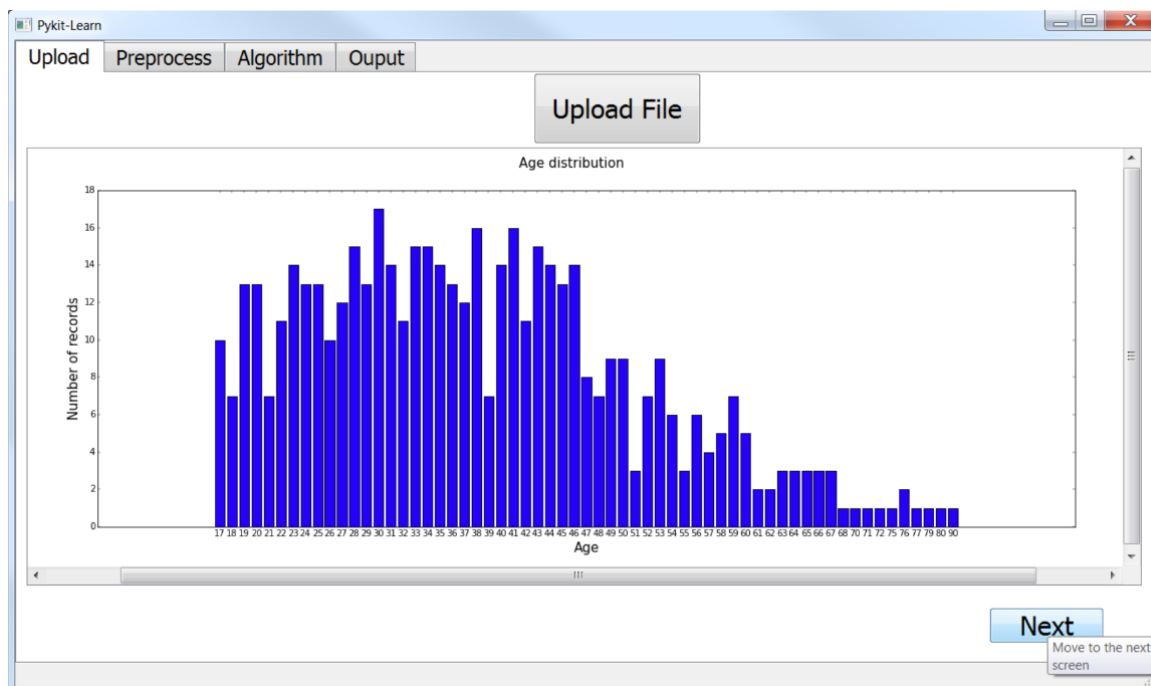


Рисунок 1.2. Интерфейс первого шага с загрузкой файла обучающей выборки

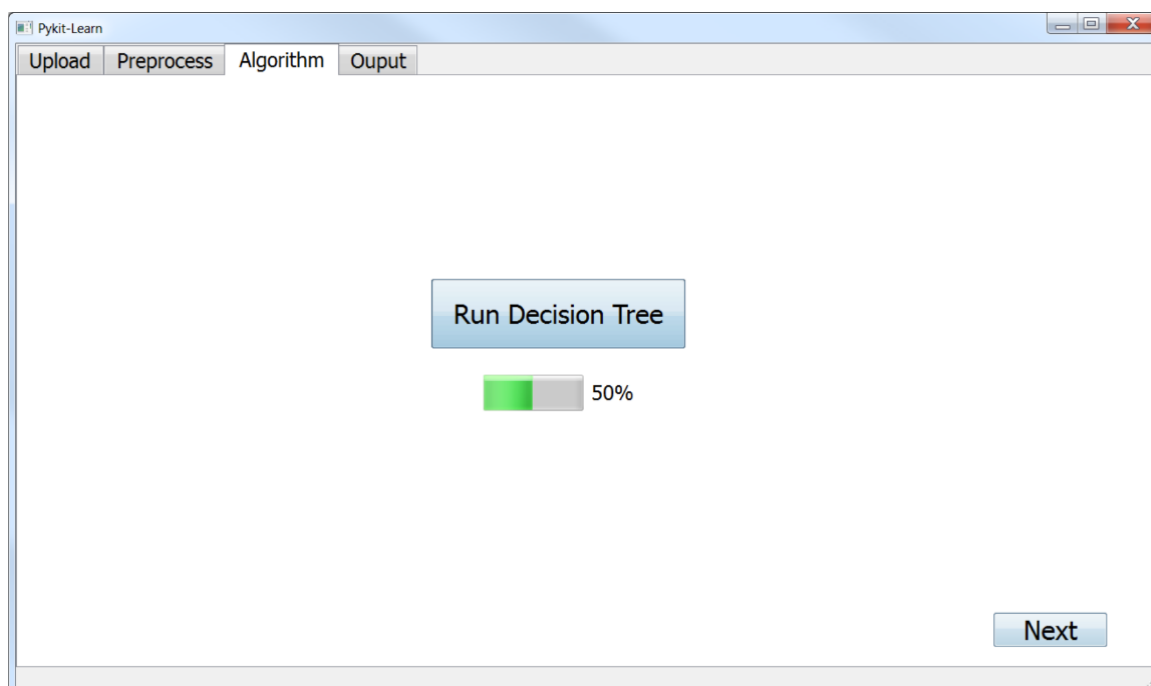


Рисунок 1.3. Интерфейс третьего шага с запуском алгоритма

В целом проект выглядит сыро и не кажется пригодным для использования, по крайней мере для задачи из данной работы.

Weka – открытый программный комплекс, содержащий в себе реализации алгоритмов машинного обучения для решения задач интеллектуального анализа. Проект разработан на языке программирования Java на базе университета Вайкато в Новой Зеландии. Целью проекта является создание современной среды для разработки и применения методов машинного обучения к реальным данным и упрощения этого процесса. Weka широко используется в учебных целях и исследователями в области машинного обучения. В состав комплекса входят средства для препроцессинга данных, классификации, регрессии, кластеризации и визуализации результатов[**weka1**].

Для работы в Weka необходимо загрузить файл с обучающей выборкой. На вкладке Preprocess (рисунок 1.4) можно увидеть статистические метрики, рассчитанные по выборке и применить один или несколько фильтров к набору данных.

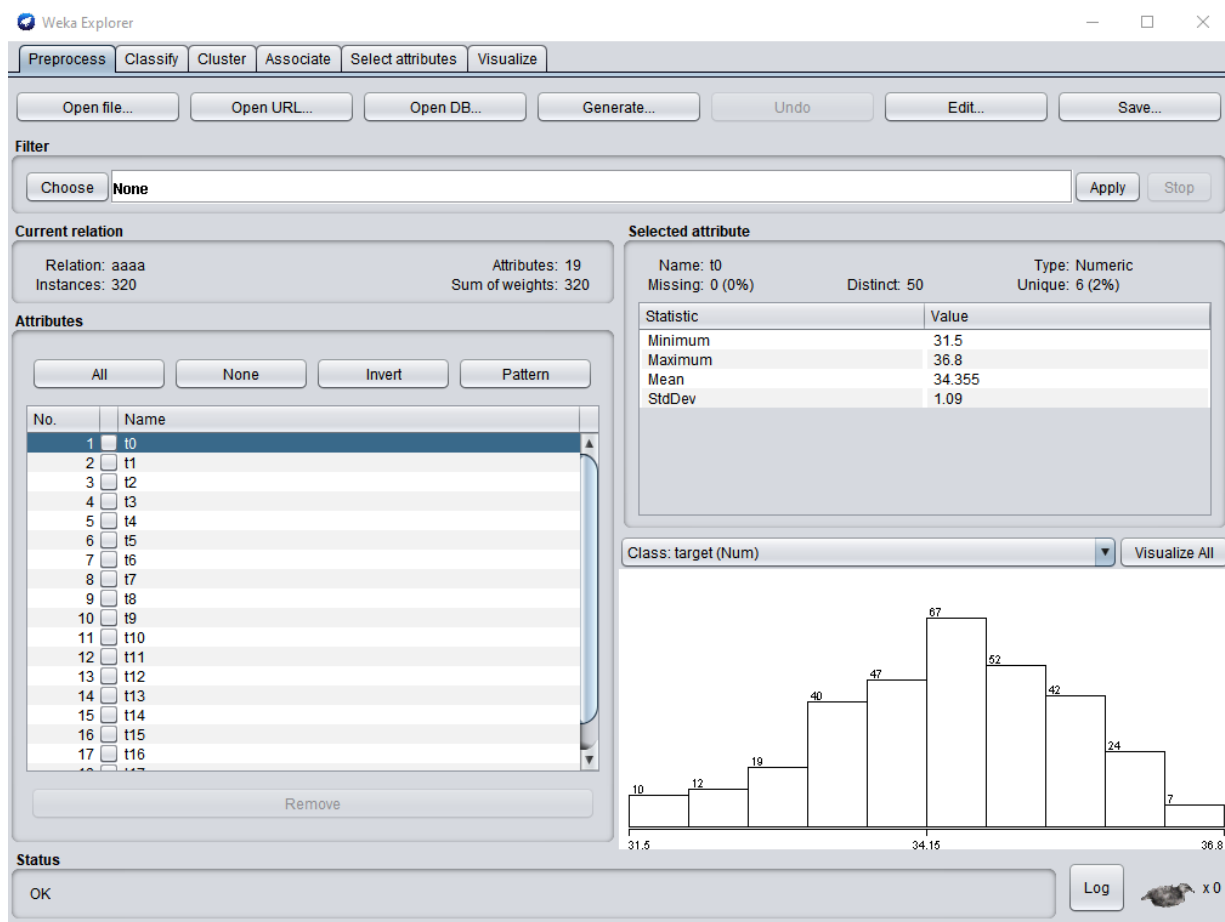


Рисунок 1.4. Интерфейс программного комплекса Weka - вкладка Preprocess

На вкладке Classify происходит выбор алгоритма классификации, выбор столбца для класса и запуск процесса классификации (рисунок 1.5).

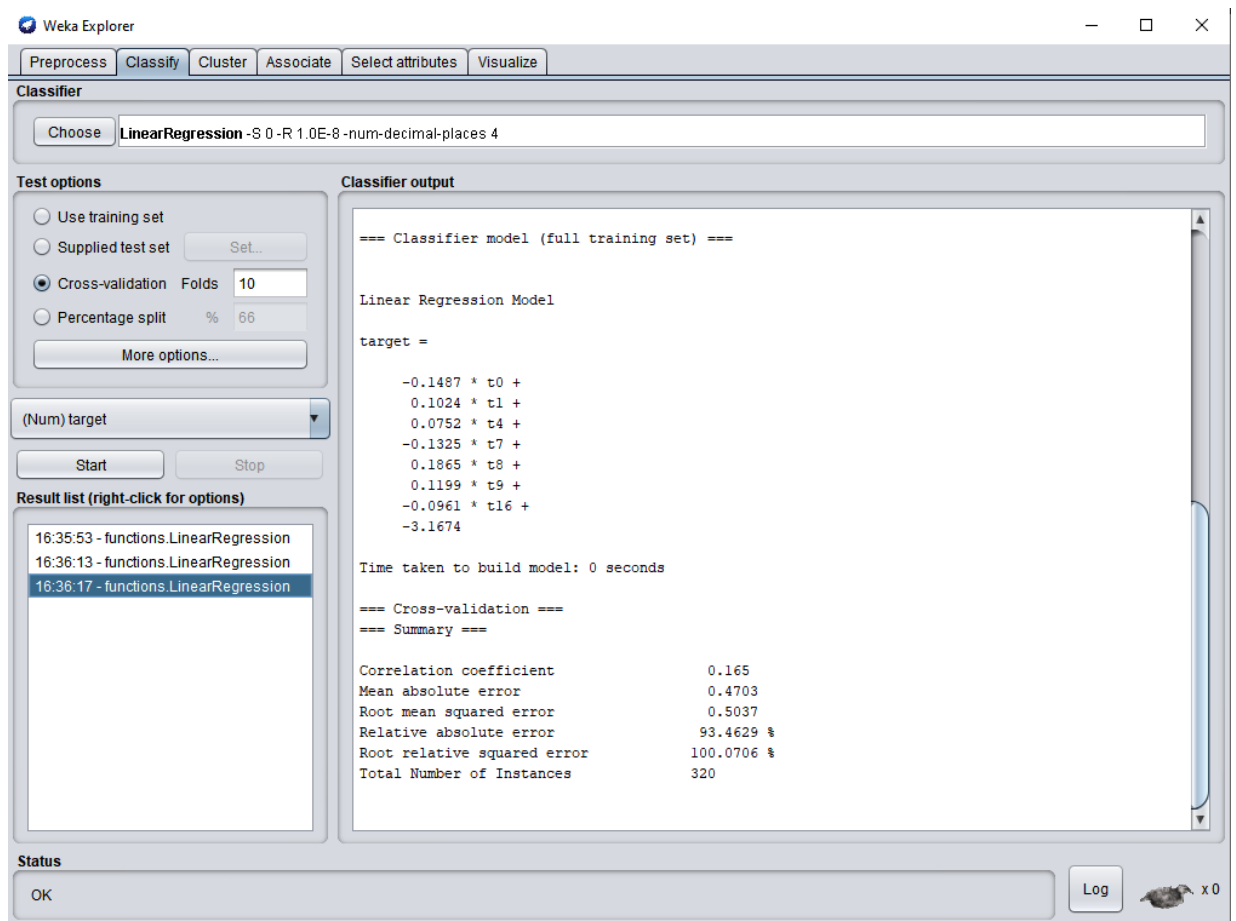


Рисунок 1.5. Интерфейс программного комплекса Weka - вкладка Classify

На вкладке Visualize имеется возможность построить графики с распределением выборки (рисунок 1.6).

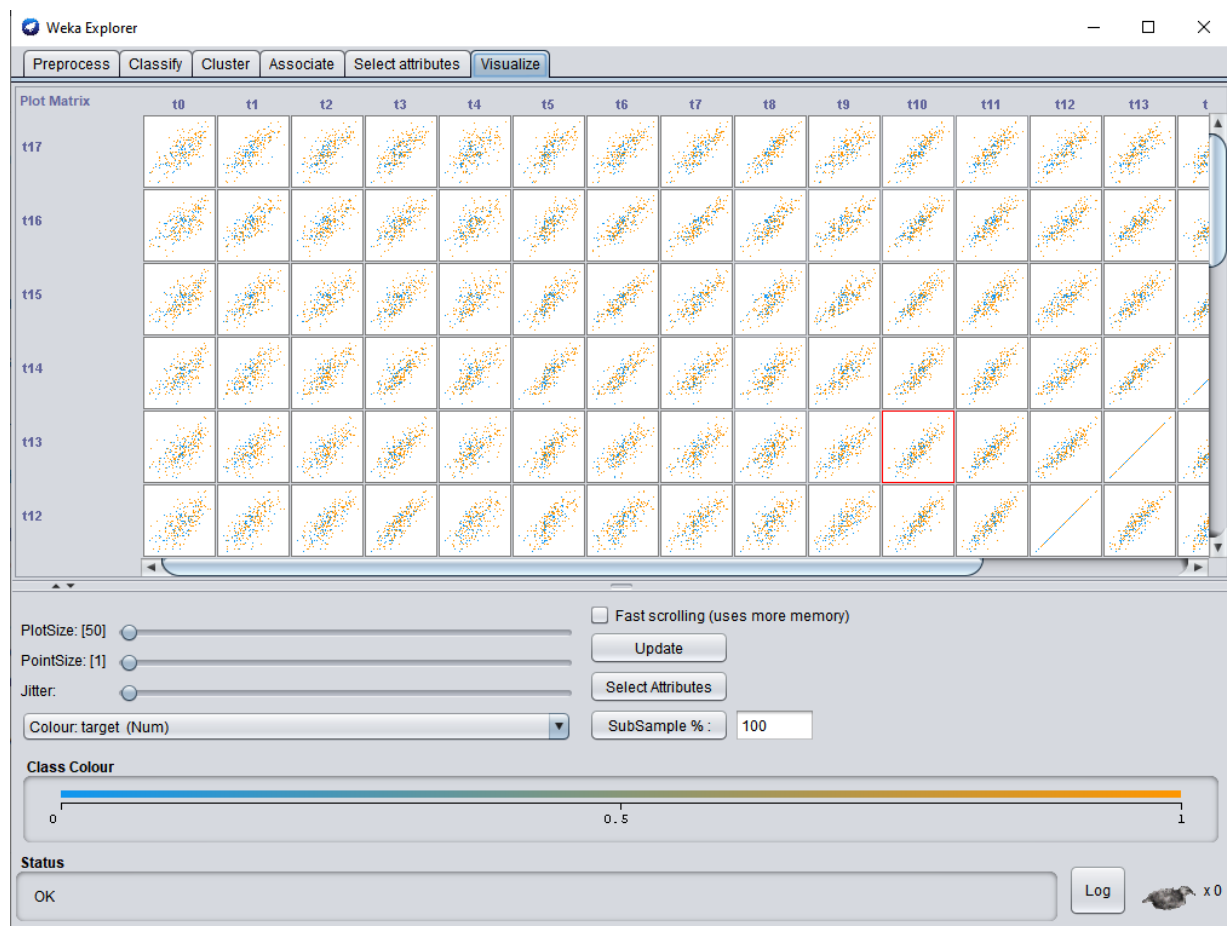


Рисунок 1.6. Интерфейс программного комплекса Weka - вкладка Visualize

В целом, Weka может помочь решить большинство задач машинного обучения за счет обширного функционала, но данная программа кажется слишком перегруженной для задачи из текущей работы. Интерфейс кажется перегруженным и нужно сидеть разбираться в нем.

Глава 2

Разработка программного обеспечения для обработки экспериментальных данных с использованием нейросетевой регрессии

В рамках данной работы нужно было разработать приложение, предоставляющее пользовательский интерфейс для работы с моделью нейросетевой регрессии. Необходимо, чтобы была возможность загрузить файл с выборкой, настроить параметры модели, запустить процесс обучения и отобразить результаты в виде статистических метрик.

Регрессионная модель использовалась для моделирования температурных данных левой или правой молочных желез после обучения на температурных данных реальных пациентов.

Для определения того, что может сделать пользователь и что он увидит в результате, была разработана диаграмма деятельности (рисунок 2.1). Данная диаграмма будет полезной как при разработке, так и при тестировании, т.к. содержит последовательную схему действий пользователя.

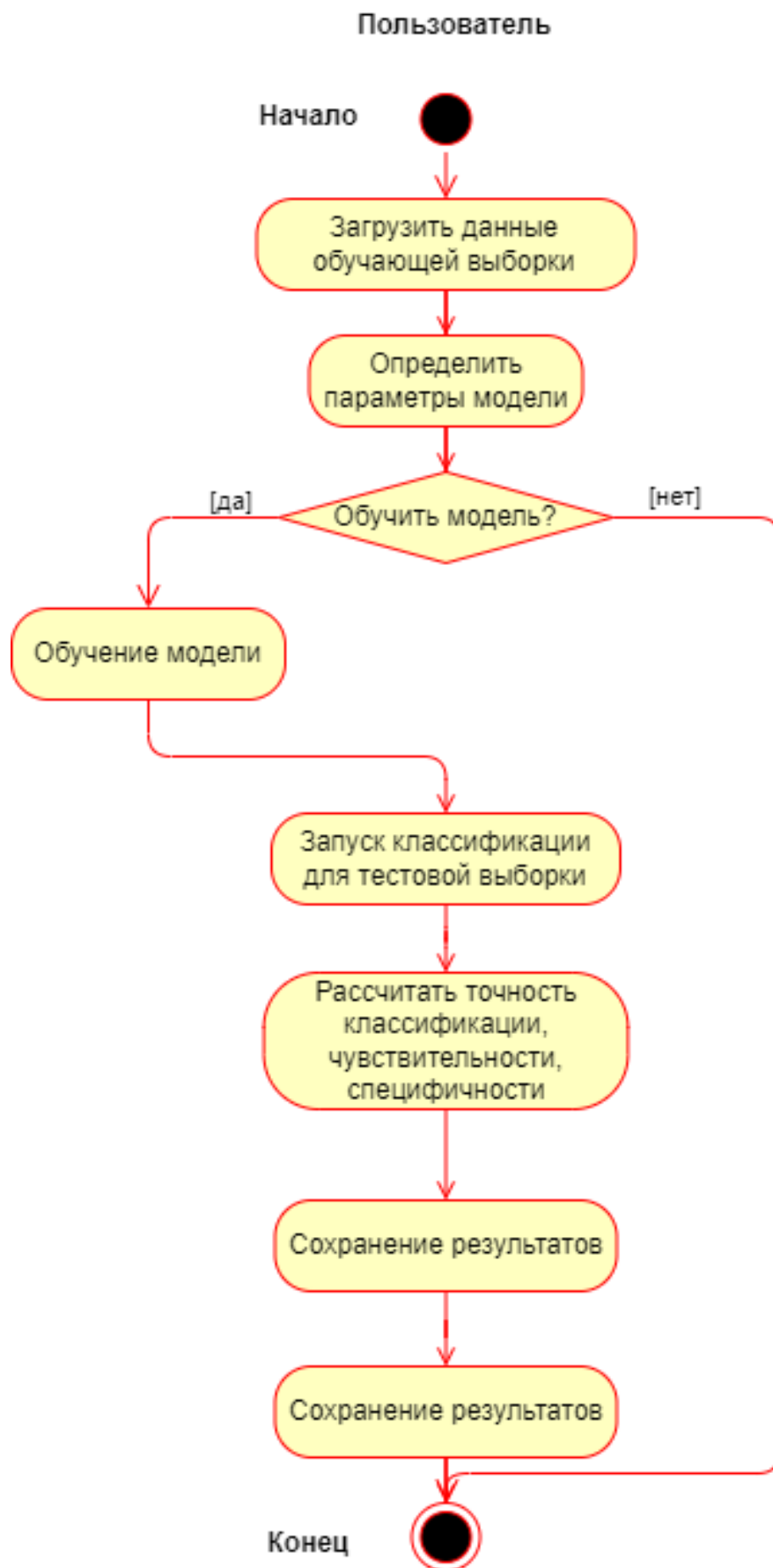


Рисунок 2.1. Диаграмма деятельности для программы

В большинстве популярных библиотек с реализацией алгоритмов машинного обучения и нейросетей описанных ранее для обучения модели используется метод с названием `fit`, принимающий данные тренировочной выборки, а для запуска алгоритма для тестовой выборки метод с названием `predict`, принимающий данные тестовой выборки. Исходя из этого, было принято решение разрабатывать приложение в рамках данной работы в виде библиотеки, которую можно будет подключить к любой модели машинного обучения, имеющей перечисленные выше методы.

2.1 Разработка нейросетевой регрессионной модели

2.1.1 Формат входных данных для модели

При выполнении работы были использованы базы с температурными данными молочных желез реальных пациентов, полученными с помощью измерений температур согласно методике РТМ. Базы были предоставлены старшим преподавателем кафедры ИСКМ Поляковым М.В. Было использовано две базы: «локальная», содержащая данные о 119 пациентах и «расширенная», содержащая данные о 7549 пациентах.

Данные были представлены в виде девяти значений температуры на поверхности кожи и девяти значений внутренней температуры для левой и правой молочных желез, согласно методике обследования методом радиотермометрии [bardati] [fear2000]. Схема расположения точек при замере температур представлена на рисунке 2.2 [vesninSovMicrowave].

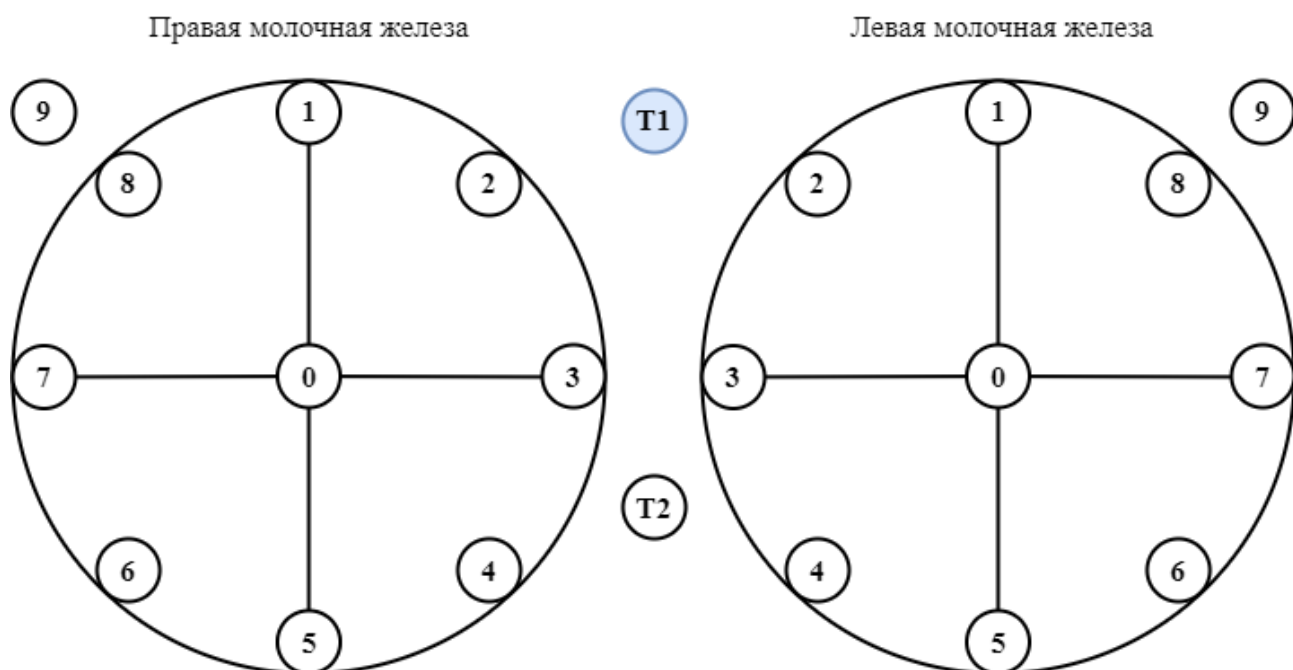


Рисунок 2.2. Схема расположения точек при замере температур методом PTM [pavlovrtm]

Одна часть данных с пациентами размечена как больные, а другая как здоровые. В рамках данной работы в процессе обучения нейросети использовались и больные и здоровые пациенты.

На подготовительном этапе данные были разбиты на обучающую и тестовую выборки. Размер тестовой выборки был определен как примерно 10% от всех данных. И обучающая и тестовая выборки перед использованием в нейросети были перемешаны в случайном порядке.

Т.к. в температурных данных все значения имеют маленькую точность и сосредоточены возле одного числа (примерно возле 34), то их нужно нормализовать. Перед использованием данных в нейронной сети они были нормализованы с помощью метода `StandardScaler` из библиотеки `Scikit-learn`. Основная идея этого метода заключается в том, что он преобразует датасет так, что его распределение будет иметь среднее значение равно нулю и стандартное отклонение равно единице. Учитывая распределение данных каждое значение в получившемся на выходе наборе данных будет высчитываться на основе

среднего значения выборки, а затем делиться на стандартное отклонение всего набора данных. В таблице 2.1 приведен пример части набора данных до нормализации и после.

Таблица 2.1. Пример значений до и после нормализации с помощью StandardScaler

	mw0	mw1	mw2	mw3
До нормализации	35.6	35.4	35.9	35.8
После нормализации	0.54550569	0.33900783	0.77272125	0.77104202

2.1.2 Выбор технологий

В качестве языка программирования был выбран Python и связка библиотек Tensorflow и Keras, т.к. для них есть множество примеров использования под текущую задачу и обучающих материалов.

Так же использование Tensorflow и Keras позволит облегчить процесс перевода вычислений на графическом ускорителе. Для выполнения расчетов при обучении на GPU необходимо только установить особый набор драйверов и библиотек для видеокарты. Если все будет успешно установлено, то tensorflow по умолчанию будет выполнять все вычисления на GPU.

2.1.3 Структура нейросети

Нейросеть, используемая в рамках данной работы состоит из пяти слоев (рисунок 2.3).

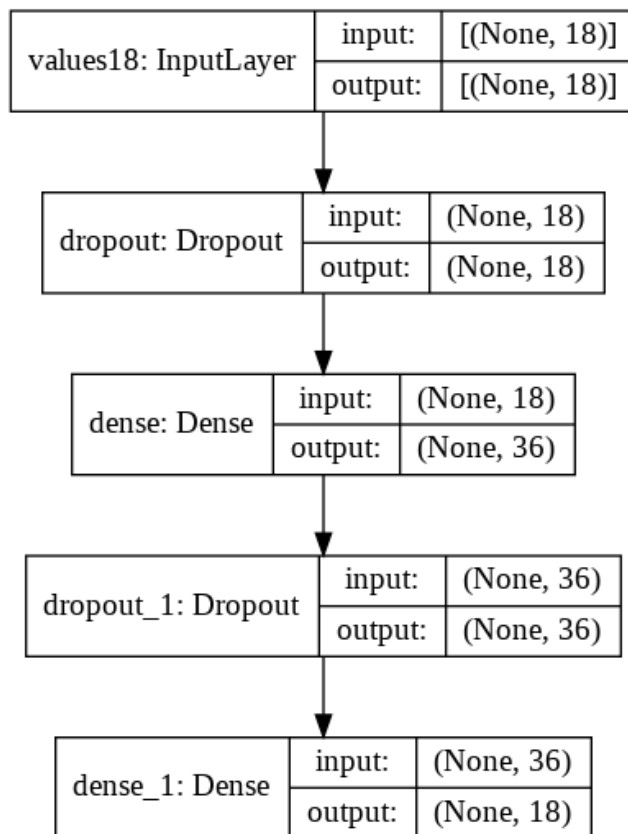


Рисунок 2.3. Структура нейронной сети

Первый слой - это входной слой, который принимает 18 точек. Остальные слои - это вариации слоев Dropout и Dense с различными параметрами. Количество слоев и их типы были определены экспериментальным способом.

Dropout слой – это слой, в котором случайные элементы датасета с определенным количеством зануляются для предотвращения переобучения. Количество точек, которое будет обнулено задается параметром слоя в процентах.

Dense – это полносвязный слой, где каждый его выход связан с каждым входом следующего слоя.

2.1.4 Обучение нейронной сети

При создании модели нейросети был использован оптимизатор градиентного спуска из библиотеки Keras. Для валидации результатов обучения модели использовалась среднеквадратичная ошибка. Параметр с количеством эпох

обучения был определен числом 1000 и был не изменен в процессе настройки остальных параметров модели. Менялось только значение параметра, отвечающего за скорость обучения.

Для отслеживания изменения метрик в процессе обучения нейросети был использован инструмент TensorBoard. Он используется для сбора логов по метрикам и построения графиков их изменения в различных проекциях. На рисунке 2.4 представлен график изменения среднеквадратичной ошибки в зависимости от времени (эпохи) обучения. Исходя из графика, можно увидеть, что среднеквадратичная ошибка уменьшается и начинает сходиться в конце обучения, что является хорошим результатом, но не гарантирует высокую точность результатов на выходе из модели.

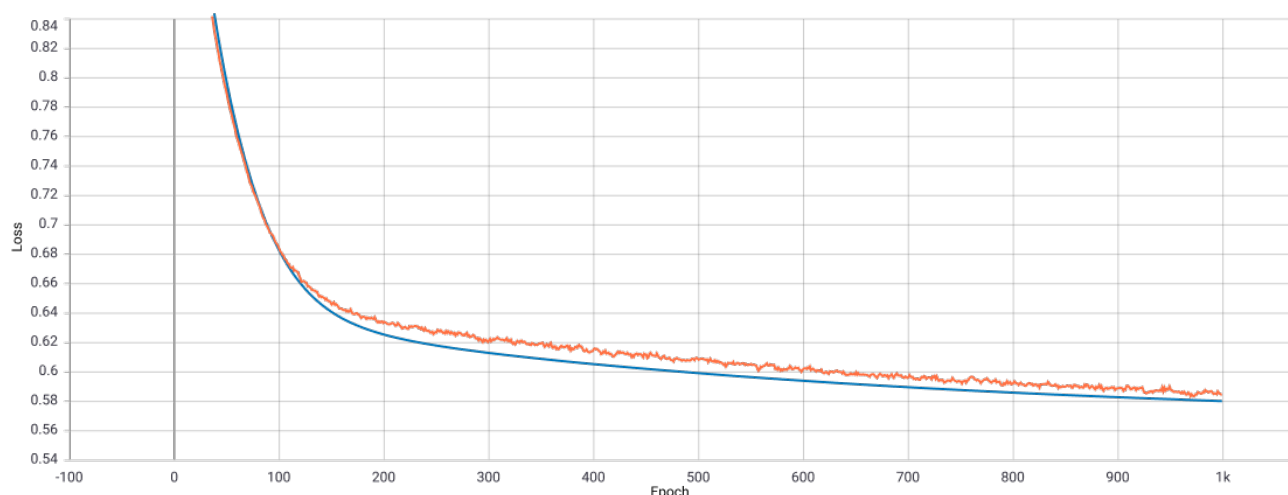


Рисунок 2.4. График изменения среднеквадратичной ошибки в зависимости от эпохи

2.2 Архитектура приложения

Перед началом разработки приложения необходимо определить, с помощью каких технологий оно будет реализовано. Если в самом начале выбрать неправильную архитектуру и инструменты для разработки, то в дальнейшем это может сильно усложнить поддержку программного обеспечения.

Описанные в предыдущей главе существующие решения MachineLearningGUI и Weka представлены в виде десктопных приложений. Когда данных для анализа много, такой подход может сильно усложнить работу, т.к. для обучения модели зачастую может потребоваться очень много времени и ресурсов. Поэтому для разрабатываемого приложения была выбрана клиент-серверная архитектура. В качестве клиента будет web-приложение, общающееся с сервером посредством REST-API. За счет использования такого подхода серверную часть с моделью можно будет вынести на отдельный сервер и производить расчеты именно там, не используя вычислительные ресурсы клиента. Также можно будет масштабировать данную систему, разместив ее в кластере. Еще одним плюсом выбора такой архитектуры является простота обновления программного обеспечения в будущем (при условии, что оно будет располагаться не локально, а с доступом через интернет).

Исходя из информации про общий принцип работы моделей в большинстве библиотек и выбранной архитектуры для лучшего понимания устройства всей схемы работы приложения были выделены компоненты и интерфейсы разрабатываемой системы, а также построена диаграмма компонентов (рисунок 2.5).

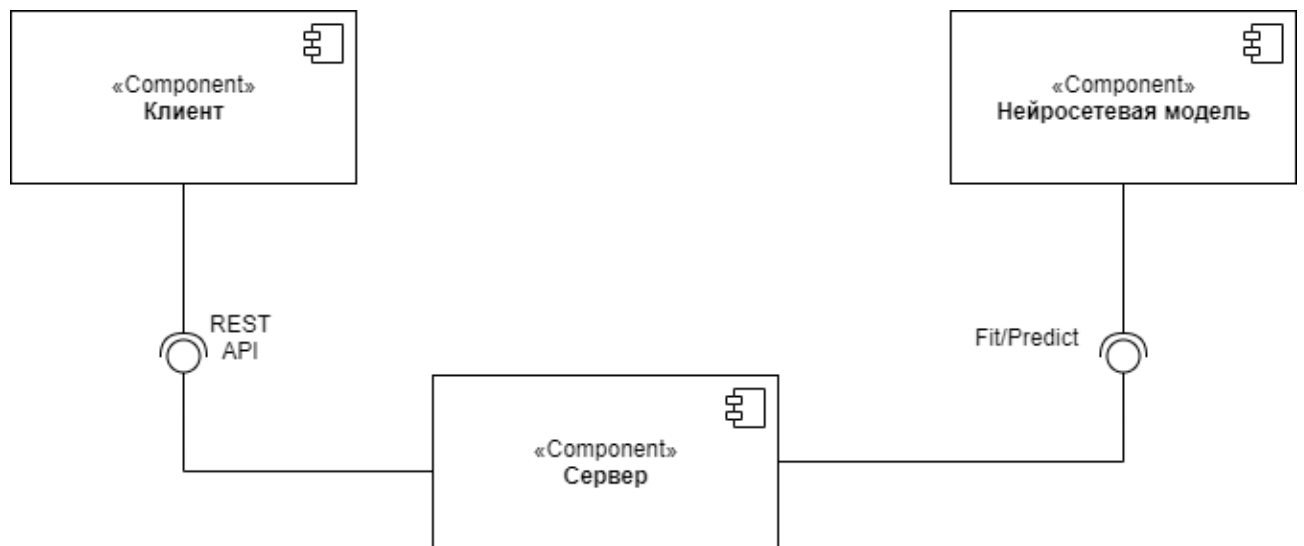


Рисунок 2.5. Диаграмма компонентов разрабатываемой системы

Схема взаимодействия клиента, сервера и нейросетевой модели представлено на диаграмме последовательности (рисунок 2.6). Для взаимодействия клиента и сервера был выбран протокол HTTP из-за большой поддержки во многих языках программирования, библиотеках и фреймворках.

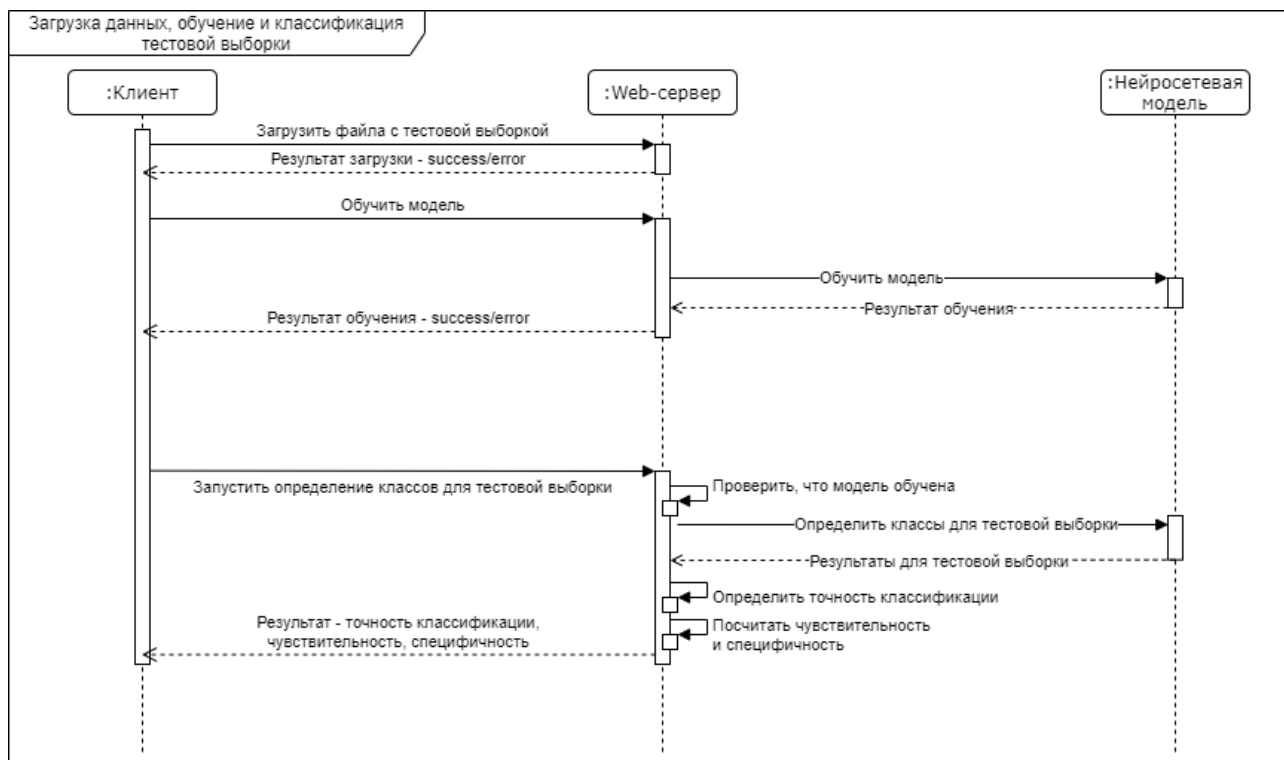


Рисунок 2.6. Диаграмма последовательности для разрабатываемого приложения

В качестве языка программирования был выбран Python, т.к. большинство популярных библиотек для машинного обучения написаны именно на нем и нужно будет с ними взаимодействовать. В силу простоты реализации и возможности быстрого прототипирования RESP-API интерфейса для backend-части был выбран веб-фреймворк Flask. Flask имеет множество дополнительных библиотек для расширения функционала, а также подробную документацию.

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, сохраняя возможность масштабирования при командной разра-

ботке.

2.3 Разработка frontend-части

Для разработки frontend-части приложения была взята библиотека VueJS-компонентов Vuetify, содержащая в себе большое количество готовых компонентов в Material-дизайне. Т.к. процесс работы с приложением можно разбить на шаги, то были разработаны отдельные компоненты для каждого шага. Пользователь может перемещаться между шагами с помощью кнопок “Вперед” и “Назад”, которые будут активны, если на текущем шаге все поля были заполнены верно (при их наличии).

На первом шаге пользователь может загрузить файл с обучающей выборкой (рисунок 2.7). Пока что поддерживаются только CSV-файлы.

The screenshot shows a web application titled "Simple Machine Learning GUI". It features a progress bar with three steps: 1. "Загрузка данных" (Data Upload), which is the active step and highlighted with a blue circle; 2. "Настройка параметров модели" (Model parameter configuration), highlighted with a grey circle; and 3. "Результаты" (Results), highlighted with a grey circle. Below the progress bar, there is a file upload section with a document icon and the text "Файл с обучающей выборкой" (File with training data). A "ВПЕРЕД" (Next) button is located at the bottom right of the form.

Рисунок 2.7. Скриншот разрабатываемого приложения на шаге “Загрузка данных”

На втором шаге происходит выбор значений параметров модели. По умолчанию всегда выводится параметр “Процент тестовой выборки”, который

отвечает за то, сколько процентов от общей выборки будет использоваться для тестирования. Набор параметров, которые выводятся на данном шаге определяется пользователем, при подключении данного приложения к своей модели. Интерфейс шага “Настройка параметров модели” представлен на рисунке 2.8.

Simple Machine Learning GUI

1 Загрузка данных 2 Настройка параметров модели 3 Результаты

Процент тестовой выборки
25

gamma
scale
1.0

НАЗАД ВПЕРЕД

Рисунок 2.8. Скриншот разрабатываемого приложения на шаге “Настройка параметров модели”

После нажатия пользователем кнопки “Вперед” происходит отправка запроса на сервер со значениями параметров и выбранным файлом и запускается процесс обучения и определения классов для тестовой выборки.

На шаге “Результаты” после успешного выполнения пользователь увидит рассчитанные значения точности, чувствительности и специфичности (рисунок 2.9). Если пользователя не устроили получившиеся результаты, то он может вернуться на предыдущий шаг и изменить параметры модели.

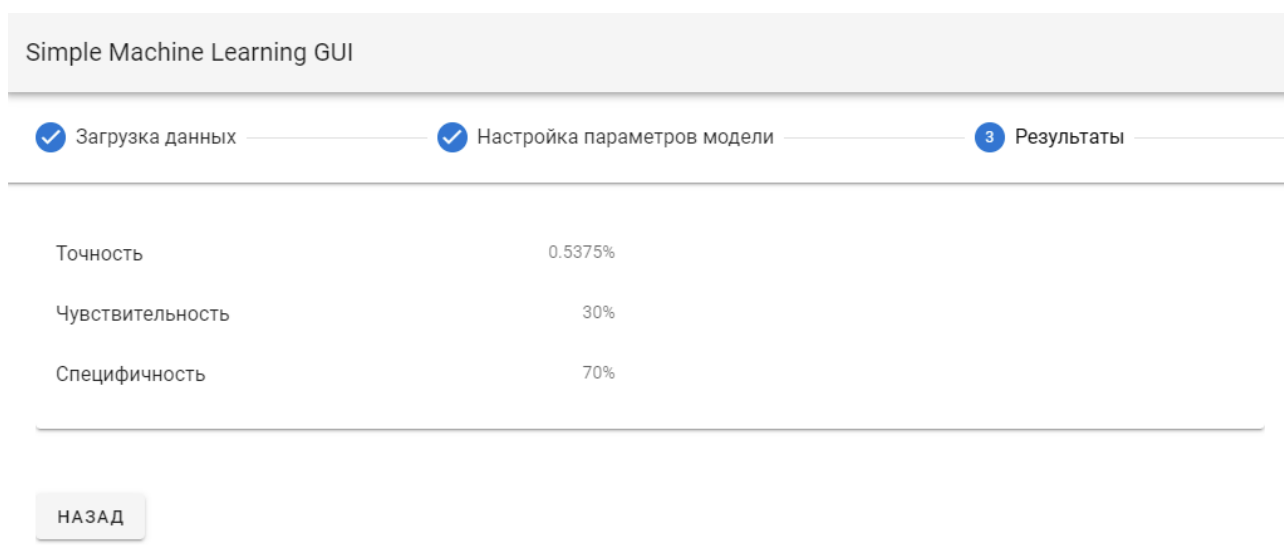


Рисунок 2.9. Скриншот разрабатываемого приложения на шаге “Результаты”

Frontend-приложение получает и отправляет данные серверу посылая AJAX-запросы к API-методам сервера, т.е. без перезагрузки страницы. Для каждого запроса сервер возвращает статус ответа и данные. При возникновении ошибок выводятся сообщения в консоль браузера.

2.4 Разработка REST API-сервиса

Backend-часть представляет собой веб-приложение на Flask. Приложение состоит из одного файла, в котором находится класс Application, который принимает на вход нейросетевую модель, путь к папке для загрузки файлов выборки, разделитель, который используется в CSV-файле, список с описанием параметров модели и порт, на котором будет запущено приложение. В одном из методов класса содержится описание и код для всех маршрутов приложения. В рамках данной работы были реализованы следующие маршруты:

- / - главная страница приложения, с которой работает пользователь;
- /model_params - API-метод для получения списка параметров модели;

- /upload_data - API-метод для загрузки CSV-файла с обучающей выборкой;
- /fit_predict - API-метод для запуска процесса обучения и классификации тестовой выборки.

Все API-методы возвращают ответ в формате JSON. Пример ответа метода /fit_predict представлен на рисунке 2.10.

```
1 {  
2   "result": {  
3     "accuracy": 0.5625,  
4     "sensitivity": 26.25,  
5     "specificity": 73.75  
6   },  
7   "status": "success"  
8 }
```

Рисунок 2.10. Пример ответа метода /fit_predict в формате JSON

При обработке запроса метода /fit_predict происходит проверка наличия у объекта модели необходимых для работы методов с названиями fit и predict. Если они отсутствуют, то вернется ответ с ошибкой. После этого происходит проверка на существование файла с данными, т.к. за время заполнения пользователем параметров модели на втором шаге с файлом могло что-то произойти.

Затем данные из файла разделяются на два набора данных - обучающую и тестовую выборки. Для этой операции 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 флаг необходимости перемешивания данных.

Значения параметров модели, указанные пользователем на втором шаге передаются внутрь модели и после этого последовательно вызываются методы fit и predict. Далее по полученным данным происходит расчет точности

классификации, чувствительности и специфичности в процентах.

Для запуска самого приложения нужно создать объект класса `Application`, передав в него все необходимые параметры, а затем вызвать метод `run`. По умолчанию приложение будет доступно по адресу `http://127.0.0.1:5000`.

2.5 Сборка в `pip`-пакет

Для возможности использования разработанного приложения в других проектах было принято решение оформить его в виде `pip`-пакета - популярного формата модулей в языке Python.

Для сборки такого пакета необходимо создать файл `setup.py` и использовать в нем функцию `setup` из библиотеки `setuptools`. Данная функция принимает следующие параметры:

- Название пакета;
- Версию пакета;
- Путь до модуля, который будет использоваться в качестве пакета;
- Описание;
- Файлы, которые должны попасть в сборку.

Так как разработанное приложение включает в себя отдельное frontend-приложение на VueJS, то последним параметром необходимо передать путь до папки с ним.

После этого с помощью команды `python setup.py sdist` будет создан `tar.gz`-архив с пакетом.

Глава 3

Тестирование разработанной библиотеки и публикация в сервис PyPI

3.1 Тестирование библиотеки на алгоритме машинного обучения SVM

Для тестирования получившегося решения была разработана программа, использующая в качестве модели алгоритм классификации SVM. В качестве файла с обучающей выборки были использованы данные компьютерного моделирования яркостной температуры молочных желез больных и здоровых пациентов, предоставленных старшим преподавателем кафедры ИСКМ Поляковым М.В.

В качестве параметров модели, отображаемых на втором шаге были выбраны следующие:

- C - параметр регуляризации;
- kernel - тип ядра, используемый в алгоритме;
- gamma - коэффициент ядра.

Код получившейся программы находится в приложении А.2.

После запуска программы и загрузки файла с обучающей выборкой на втором шаге были отображены все те параметры, которые были указаны при инициализации приложения (рисунок 3.1).

Simple Machine Learning GUI

✓ Загрузка данных

2 Настройка параметров модели

3 Результаты

Процент тестовой выборки	gamma	c	kernel
25	scale	1.0	sigmoid

НАЗАД

ВПЕРЕД

Рисунок 3.1. Скриншот тестовой программы с параметрами алгоритма SVM

После нажатия кнопки “Вперед” приложение отработало штатно и были выведены результаты обучения (рисунок 3.2).

Simple Machine Learning GUI

✓ Загрузка данных

✓ Настройка параметров модели

3 Результаты

Точность	0.4125%
Чувствительность	0%
Специфичность	100%

НАЗАД

Рисунок 3.2. Скриншот тестовой программы с результатами классификации методом SVM

3.2 Настройка автоматической сборки и публикации библиотеки в сервис PyPI

Чтобы разработанное приложение в формате `pip`-пакета можно было установить с помощью менеджера пакетов `pip` и использовать в своих проектах его нужно опубликовать в сервисе PyPI, который является бесплатным хранилищем пакетов. Для этого сначала нужно зарегистрироваться в сервисе, а после этого с помощью библиотеки `twine`, выполнив команду `twine upload dist/*`.

Для упрощения процесса поставки обновлений пакета в PyPI была настроена автоматическая сборка проекта с помощью CI/CD сервиса GitHub Actions. Сервис бесплатно предоставляет 2000 минут в месяц на выполнение сборок. Каждая сборка запускается в Docker-контейнере, что позволяет организовать изолированную среду. Конфигурационный файл сборки хранится в репозитории в формате `yaml`. Сборка запускается, если появляется новый тэг с версией в репозитории. Были определены и описаны в конфигурационном файле следующие шаги сборки:

- Установка интерпретатора NodeJS;
- Установка интерпретатора Python;
- Сборка frontend-части;
- Сборка `pip`-пакета;
- Публикация пакета в PyPI;

Заключение

В данной работы были рассмотрены сферы деятельности и основные задачи, в которых используются нейросети и методы машинного обучения, а также некоторые из популярных библиотек языка программирования Python для решения таких задач.

Было реализовано приложение, предоставляющее пользовательский графический интерфейс для нейросетевой модели. При разработке использовались такие библиотеки языка Python как Flask и Scikit-learn. При разработке клиентской части использовался фреймворк VueJS и библиотека компонентов Vuetify. Также получившееся приложение было упаковано в `pip`-пакет и опубликовано в сервисе PyPI для возможности использования в других проектах. Была настроена автоматическая сборка и публикация `pip`-пакета в PyPI с помощью CI/CD сервиса GitHub Actions.

Разработанное приложение было протестировано на модели алгоритма классификации SVM с различными параметрами модели.

Приложение доступно в публичном репозитории GitHub по ссылке https://github.com/warete/ml_simple_gui

Список литературы

1. The origin of low-surface-brightness galaxies in the dwarf regime / R. A. Jackson, G. Martin, S. Kaviraj, M. Ramsay, J. Devriendt, T. M. Sedgwick, C. Laigle, H. Choi, R. S. Beckmann, M. Volonteri, Y. Dubois, C. Pichon, S. K. Yi, A. D. Slyz, K. Kraljic, T. Kimm, S. Peirani, I. K. Baldry // Monthly Notices of the Royal Astronomical Society. — 2021. — Т. 502. — С. 4262—4276.

Приложение А

Листинг разработанной программы

Листинг А.1. Код backend-части приложения

```
1 from flask import Flask, jsonify, render_template, request,
2 send_file
3 from flask_cors import CORS
4 from sklearn.metrics import accuracy_score
5 from sklearn.model_selection import train_test_split
6 from .utils import check_model, calculate_sensitivity, calcu
7 late_specificity, get_data_from_csv
8 import os
9 import json
10 import glob
11
12 import pandas as pd
13 import time
14 import urllib.parse
15
16 class Application:
17     model = None
18     model_constructor = None
19     upload_path = '/upload'
20     server_port = 5000
21     csv_delimiter = ','
22     model_params = [
23         {
24             'code': 'testPercent',
25             'name': 'Процент' тестовойвыборки ',
26             'defaultValue': 25,
27         }
28     ]
29     metrics = []
```

```

30
31     def __init__(self, model, upload_path, csv_delimiter, mo
32 del_params=None, server_port=5000, metrics=None):
33         if model_params is None:
34             model_params = []
35         if callable(model):
36             self.model_constructor = model
37         else:
38             self.model = model
39         self.upload_path = upload_path
40         self.csv_delimiter = csv_delimiter
41         self.server_port = server_port
42         self.model_params += model_params
43         self.process_model_params()
44         if metrics:
45             self.metrics = metrics
46
47     def set_model(self, model):
48         self.model = model
49         return self
50
51     def set_upload_path(self, upload_path):
52         self.upload_path = upload_path
53         return self
54
55     def set_server_port(self, port):
56         self.server_port = port
57         return self
58
59     def set_csv_delimiter(self, csv_delimiter):
60         self.csv_delimiter = csv_delimiter
61         return self
62
63     def run(self):
64         flask_app = Flask(

```

```

65         __name__ ,
66         static_url_path='',
67         static_folder='./frontend/dist',
68         template_folder='./frontend/dist'
69     )
70     flask_app.debug = True
71     flask_app.config['UPLOAD_FOLDER'] = os.path.abspath(
72 self.upload_path)
73     CORS(flask_app)
74
75     self.set_routes(flask_app)
76
77     flask_app.run(port=self.server_port)
78
79     def process_model_params(self):
80         for k in range(len(self.model_params)):
81             if 'value' not in self.model_params[k]:
82                 self.model_params[k]['value'] = self.model_p
83 arams[k]['defaultValue']
84
85     def set_routes(self, app):
86         @app.route('/')
87         def index():
88             return render_template('index.html')
89
90         @app.route('/') + self.upload_path + '/<path:path>')
91         def send_upload(path):
92             return send_file(os.path.join(app.config['UPLOAD
93 _FOLDER'], path))
94
95         @app.route('/model_params')
96         def model_params():
97             return jsonify({
98                 'status': 'success',
99                 'result': {

```



```

100             'params': self.model_params
101         }
102     })
103
104     @app.route('/get_files/', methods=['GET'])
105     def get_files():
106         return jsonify({
107             'status': 'success',
108             'result': [os.path.basename(file) for file in
109 glob.glob(os.path.join(app.config
110 ['UPLOAD_FOLDER'], '*.csv'))]
111         })
112
113
114     @app.route('/upload_data/', methods=['POST'])
115     def upload_data():
116         file = request.files['file']
117         if file:
118             file_path = os.path.join(
119                 app.config['UPLOAD_FOLDER'], file.filename
120 me)
121             file.save(file_path)
122             return jsonify({
123                 'status': 'success',
124                 'result': {
125                     'file_path': file.filename
126                 }
127             })
128         else:
129             return jsonify({
130                 'status': 'error',
131                 'result': {
132                     'message': 'Неподходящий тип файла '
133                 }
134             })

```

```

135
136         @app.route('/fit_predict/', methods=['POST'])
137         def fit_predict_handler():
138             # {'file': {'file_path': 'test.csv'}, 'modelPara
139 ms': {}}
140             req_params = json.loads(request.get_data())
141             if len(req_params['file']['file_path']) == 0 or
142 not os.path.isfile(
143                 os.path.join(self.upload_path, req_param
144 s['file']['file_path'])):
145                 return jsonify({
146                     'status': 'error',
147                     'result': {
148                         'message': 'Файл' не существует '
149                     }
150                 })
151             if self.model_constructor:
152                 params = {}
153                 for paramCode in req_params['modelParams']:
154                     if paramCode == 'testPercent':
155                         continue
156                     params[paramCode] = req_params['modelPar
157 ams'][paramCode]['value']
158                     self.model = self.model_constructor(**params
159 )
160             if not check_model(self.model):
161                 return jsonify({
162                     'status': 'error',
163                     'result': {
164                         'message': 'Ошибка' прииспользованиимодели
165 . Проверьтеналичиеи необходимыхметодов '
166                     }
167                 })
168             data_from_file = get_data_from_csv(
169                 os.path.join(self.upload_path, req_params['f

```

```

170 ile'] ['file_path']), self.csv_delimiter)
171         x = data_from_file.iloc[:, :-1]
172         y = data_from_file.iloc[:, -1:]
173
174         on_before_split_method = getattr(
175             self.model, 'on_before_split', None)
176         if callable(on_before_split_method):
177             x, y = self.model.on_before_split(data_from_
178 file)
179
180         x_train, x_test, y_train, y_test = train_test_sp
181 lit(x, y,
182
183         test_size=int(req_params['modelParams']['testPercent'] [
184
185             'value']) / 100
186
187     )
188
189         on_after_split_method = getattr(self.model, 'on_
190 after_split', None)
191         if callable(on_after_split_method):
192             x_train, x_test, y_train, y_test = self.mode
193 l.on_after_split(
194             x_train, x_test, y_train, y_test)
195
196         self.model.fit(x_train, y_train)
197         print(x_test.shape)
198         y_pred = self.model.predict(x_test)
199
200         metrics_result = {}
201         for metric in self.metrics:
202             f_metric_func = metric['func']
203             metric_result = None
204             if callable(f_metric_func):

```

```

205             metric_result = f_metric_func(
206                 y_test, y_pred, y_train, x_train, x_
207 test)
208         else:
209             metric_method = getattr(self.model, f_me
210 tric_func, None)
211             if metric_method:
212                 metric_result = metric_method(
213                     y_test, y_pred, y_train, x_train
214 , x_test)
215
216             if metric_result:
217                 metrics_result[metric['code']] = {
218                     'code': metric['code'],
219                     'name': metric['name'],
220                     'result': metric_result,
221                     'result_type': metric['result_type']
222                 }
223
224             # accuracy = accuracy_score(y_test, y_pred)
225             # sensitivity = calculate_sensitivity(y_test, y_
226 pred) * 100
227             # specificity = calculate_specificity(y_test, y_
228 pred) * 100
229
230             return jsonify({
231                 'status': 'success',
232                 'result': metrics_result
233             })
234
235     @app.route('/predict/', methods=['POST'])
236     def predict_handler():
237         req_params = json.loads(request.get_data())
238         if len(req_params['file']['file_path']) == 0 or
239 not os.path.isfile(

```

```

240             os.path.join(self.upload_path, req_param
241 s['file']['file_path']))):
242         return jsonify({
243             'status': 'error',
244             'result': {
245                 'message': 'Файл' не существует '
246             }
247         })
248     if not check_model(self.model):
249         return jsonify({
250             'status': 'error',
251             'result': {
252                 'message': 'Ошибка' при использовании модели
253 . Проверьте наличие необходимых методов '
254             }
255         })
256     data_from_file = get_data_from_csv(
257         os.path.join(self.upload_path, req_params['f
258 ile']['file_path']), self.csv_delimiter)
259     x = data_from_file.iloc[:, :-1]
260     y = data_from_file.iloc[:, -1:]
261
262     on_before_split_method = getattr(
263         self.model, 'on_before_split_for_predict', N
264 one)
265     if callable(on_before_split_method):
266         x_test = on_before_split_method(data_from_fi
267 le)
268
269     print(x_test.shape)
270     y_pred = self.model.predict(x_test)
271     print(y_pred.shape)
272     on_after_real_predict_method = getattr(
273         self.model, 'on_after_real_predict', None)
274     if callable(on_after_real_predict_method):

```

```

275         y_pred = on_after_real_predict_method(y_pred
276 )
277
278         new_data_frame = pd.DataFrame(y_pred)
279         predict_data_file_name, _ = os.path.splitext(
280             req_params['file']['file_path'])
281         result_file_path = os.path.join(
282             self.upload_path, predict_data_file_name + '
283 _result_' + time.strftime("%Y_%m_%d_%H_%M_%S") + '.csv')
284         new_data_frame.to_csv(result_file_path, sep=',',
285                               encoding='utf-8', index=False)
286
287
288         return jsonify({
289             'status': 'success',
290             'result': {
291                 'filePath': self.upload_path + '/' + pre
292 dict_data_file_name + '_result_' + time.strftime("%Y_%m_%d_%
293 H_%M_%S") + '.csv'
294             }
295         })

```

Листинг А.2. Код программы для тестирования приложения с подключенной моделью регрессии экспериментальных данных

```

1  from ml_simple_gui import Application
2  import os
3  from sklearn.svm import SVC
4  from sklearn.metrics import accuracy_score
5
6  import numpy as np
7  import pandas as pd
8  import tensorflow as tf
9  from sklearn.preprocessing import StandardScaler
10 from sklearn.model_selection import train_test_split

```

```

11 import time
12
13
14 class CustomModel:
15     model = None
16     train_dataset = None
17     test_dataset = None
18     epochs = 1000
19     main_scaler = None
20     columns = []
21     last_fit_history = {}
22
23     def __init__(self, optimizer_learning_speed=0.01, epochs
24 =1000):
25         inputs = tf.keras.layers.Input(
26             name='values18', shape=(19,), dtype='float32')
27         outputs = tf.keras.layers.Dropout(0.1)(inputs)
28
29         outputs = tf.keras.layers.Dense(37, activation='sigm
30 oid')(outputs)
31         outputs = tf.keras.layers.Dropout(0.1)(outputs)
32
33         outputs = tf.keras.layers.Dense(18)(outputs)
34         model = tf.keras.Model(inputs=inputs, outputs=output
35 s)
36         model.compile(optimizer=tf.keras.optimizers.SGD(
37             float(optimizer_learning_speed)), loss='MSE')
38
39         self.epochs = int(epochs)
40         self.model = model
41
42     def fit(self, x_train, y_train):
43         start_t = time.time()
44         history = self.model.fit(self.train_dataset, validat
45 ion_data=self.test_dataset, epochs=self.epochs,

```

```

46                                     callbacks=[tf.keras.callbacks
47 ks.TensorBoard(log_dir='logs')])
48
49         self.last_fit_history['loss'] = history.history['los
50 s']
51         self.last_fit_history['val_loss'] = history.history[
52 'val_loss']
53
54         end_t = time.time()
55         print('Finish time: ', end_t - start_t)
56
57     def predict(self, x_test):
58         pred_test_X = self.model.predict(x_test)
59         return pred_test_X
60
61     def on_before_split(self, data_from_file):
62         df = data_from_file
63         cols = df.columns.tolist()
64         diameter_data = df['diameter'].values
65         df = df[cols[:-1]]
66         self.columns = df.columns
67
68         source_values = df.values
69
70         # data preprocessing
71         scaler = StandardScaler()
72         scaler.fit(source_values)
73
74         source_values = scaler.transform(source_values)
75
76         d_scaler = StandardScaler()
77         d_scaled = d_scaler.fit_transform([[x] for x in diam
78 eter_data])
79         diameter_data = [x[0] for x in d_scaled]
80

```



```

81         a = []
82         for i in range(len(source_values)):
83             a.append(np.insert(source_values[i], 0, diameter
84 _data[i]))
85         source_values = np.array(a)
86
87         np.random.shuffle(source_values)
88
89         self.main_scaler = scaler
90
91         return source_values[:, :19], source_values[:, 19:]
92
93     def on_before_split_for_predict(self, data_from_file):
94         df = data_from_file
95         cols = df.columns.tolist()
96         diameter_data = df['diameter'].values
97         df = df[cols[:-1]]
98
99         source_values = df.values
100
101         # data preprocessing
102         scaler = StandardScaler()
103         scaler.fit(source_values)
104
105         source_values = scaler.transform(source_values)
106
107         d_scaler = StandardScaler()
108         d_scaled = d_scaler.fit_transform([[x] for x in diam
109 eter_data])
110         diameter_data = [x[0] for x in d_scaled]
111
112         a = []
113         for i in range(len(source_values)):
114             a.append(np.insert(source_values[i], 0, diameter
115 _data[i]))

```

```

116         source_values = np.array(a)
117
118         np.random.shuffle(source_values)
119
120         self.main_scaler = scaler
121
122         return source_values
123
124     def on_after_split(self, x_train, x_test, y_train, y_test):
125
126         self.train_dataset = tf.data.Dataset.from_tensor_slices(
127             (x_train, y_train)).batch(1000)
128
129         self.test_dataset = tf.data.Dataset.from_tensor_slices(
130             (x_test, y_test)).batch(1000)
131
132         return x_train, x_test, y_train, y_test
133
134     def on_after_real_predict(self, y_pred):
135
136         return self.main_scaler.inverse_transform(y_pred).tolist()
137
138     def prepare_data_for_error_metrics(self, y_test, y_pred,
139         x_test):
140
141         test_x_without_d = x_test[:, 1:]
142
143         real_data = pd.DataFrame(self.main_scaler.inverse_transform(
144             np.hstack((test_x_without_d, y_test))),
145             columns=self.columns)
146
147         predict_data = pd.DataFrame(self.main_scaler.inverse_transform(
148             np.hstack((test_x_without_d, y_pred))),
149             columns=self.columns)
150
151         left_columns = ['t' + str(i) for i in range(18, 36)]
152
153         left_real = real_data[left_columns]
154
155         left_predict = predict_data[left_columns]
156
157         left_real.columns = [

```

```

151         'mw' + str(i) for i in range(9)] + ['ir' + str(i
152     ) for i in range(9)]
153         left_predict.columns = [
154             'mw' + str(i) for i in range(9)] + ['ir' + str(i
155     ) for i in range(9)]
156
157         return left_predict, left_real
158
159     def calc_rel_error(self, y_test, y_pred, y_train, x_train, x_test):
160
161         '''Метод для расчета относительной ошибки
162
163         '''
164         left_predict, left_real = self.prepare_data_for_error_metrics(
165     y_test, y_pred, x_test)
166         rel_error = left_real.sub(left_predict).div(
167             left_real).abs().sum().mul(100).div(len(left_real)).to_frame()
168
169         result = {}
170         for item_k in rel_error.to_dict()[0].keys():
171             result[item_k] = float('{:.3f}'.format(
172                 rel_error.to_dict()[0][item_k]))
173         return result
174
175     def calc_mae_error(self, y_test, y_pred, y_train, x_train, x_test):
176
177         '''Метод для расчета средней абсолютной ошибки
178
179         '''
180         left_predict, left_real = self.prepare_data_for_error_metrics(
181     y_test, y_pred, x_test)
182         mae_error = left_real.sub(left_predict).abs(

```

```

186         ).sum().div(len(left_real)).to_frame()
187
188         result = {}
189         for item_k in mae_error.to_dict()[0].keys():
190             result[item_k] = float('{:.3f}'.format(
191                 mae_error.to_dict()[0][item_k]))
192         return result
193
194     def get_fit_mse(self, y_test, y_pred, y_train, x_train,
195 x_test):
196         return float('{:.3f}'.format(self.last_fit_history['
197 val_loss'][-1]))
198
199     def get_loss_graph(self, y_test, y_pred, y_train, x_train,
200 x_tests):
201         import matplotlib.pyplot as plt
202         import io
203         import base64
204
205         plt.xlabel('epochs')
206         plt.ylabel('loss')
207         plt.plot(list(range(self.epochs)),
208                 self.last_fit_history['loss'], label='loss'
209 )
210         plt.plot(list(range(self.epochs)),
211                 self.last_fit_history['val_loss'], label='v
212 al_loss')
213         plt.legend()
214         plt.grid(True)
215
216         s = io.BytesIO()
217         plt.savefig(s, format='png', bbox_inches="tight")
218         plt.close()
219         s = base64.b64encode(s.getvalue()).decode("utf-8").r
220 eplace("\n", "")

```

```

221         return 'data:image/png;base64, ' + s
222
223
224     def model(**args):
225         return CustomModel(**args)
226
227
228     def accuracy_score_custom(y_test, y_pred, y_train, x_train,
229     x_test):
230         return accuracy_score(y_test, y_pred)
231
232
233     app = Application(
234         model=model,
235         upload_path=os.path.join('upload'),
236         csv_delimiter=',',
237         model_params=[
238             {
239                 'code': 'optimizer_learning_speed',
240                 'name': 'Скорость' обучения',
241                 'defaultValue': '0.01',
242             },
243             {
244                 'code': 'epochs',
245                 'name': 'Количество' эпох',
246                 'defaultValue': '50',
247             }
248         ],
249         server_port=5000,
250         metrics=[
251             {
252                 'code': 'rel_error',
253                 'name': 'Относительная' ошибка',
254                 'func': 'calc_rel_error',
255                 'result_type': 'table'

```

```

256         },
257         {
258             'code': 'mae_error',
259             'name': 'Средняя' абсолютнаяошибка ',
260             'func': 'calc_mae_error',
261             'result_type': 'table'
262         },
263         {
264             'code': 'mse',
265             'name': 'Среднеквадратичная' ошибка',
266             'func': 'get_fit_mse',
267             'result_type': 'scalar'
268         },
269         {
270             'code': 'loss',
271             'name': 'График'
272             'изменениясреднеквадратичнойошибкивовремяобучения',
273             'func': 'get_loss_graph',
274             'result_type': 'image'
275         }
276     ]
277 )
278
279
280 app.run()

```