

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	5
1.1 Области применения	5
1.2 Популярные библиотеки с реализацией нейронных сетей	6
1.2.1 TensorFlow	6
1.2.2 Keras	7
1.2.3 Scikit-learn	7
1.2.4 PyTorch	8
1.2.5 Theano	8
1.2.6 Обзор существующих решений реализации графического интерфейса для задач машинного обучения	8
Глава 2. Разработка пользовательского интерфейса для нейронной сети	14
2.1 Архитектура приложения	16
2.2 Разработка frontend-части	19
2.3 Разработка REST API-сервиса	21
2.4 Сборка в pip-пакет	23
Глава 3. Тестирование разработанной библиотеки и публикация в сервис PyPI	24
3.1 Тестирование библиотеки на алгоритме машинного обучения SVM	24
3.2 Настройка автоматической сборки и публикации библиотеки в сервис PyPI	26

Заключение	27
Список литературы	28
Приложение А. Листинг разработанной программы	29

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.

Главной целью работы является проектирование и 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 PyPI.

Глава 1

Методы и области применения машинного обучения и нейросетей

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

1.2 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)

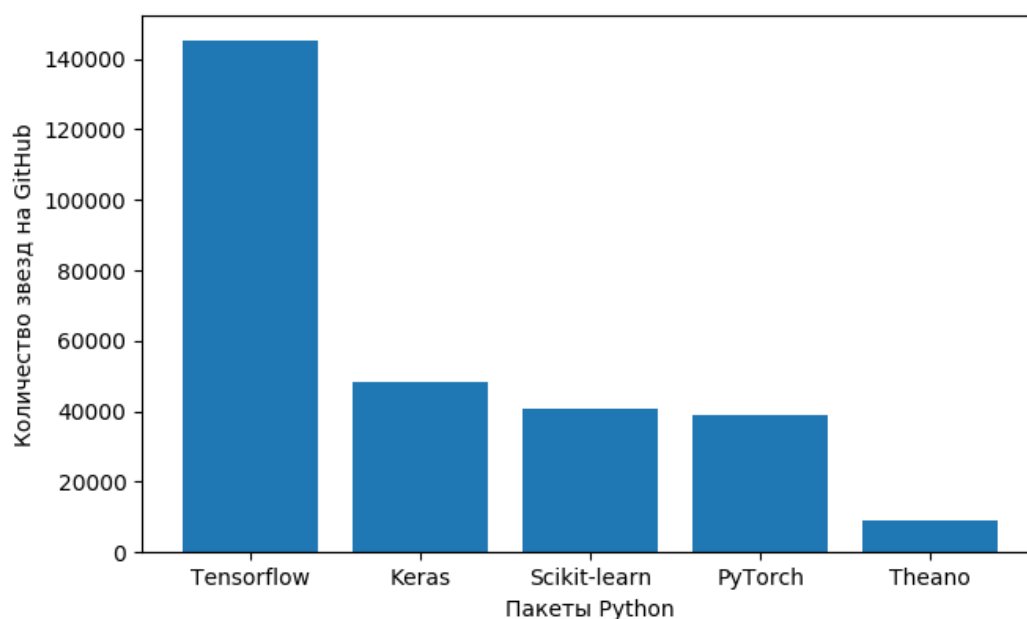


Рисунок 1.1. Популярные пакеты Python для машинного обучения по данным рейтинга на GitHub

1.2.1 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].

1.2.2 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

1.2.3 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.

1.2.4 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++.

1.2.5 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].

1.2.6 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.

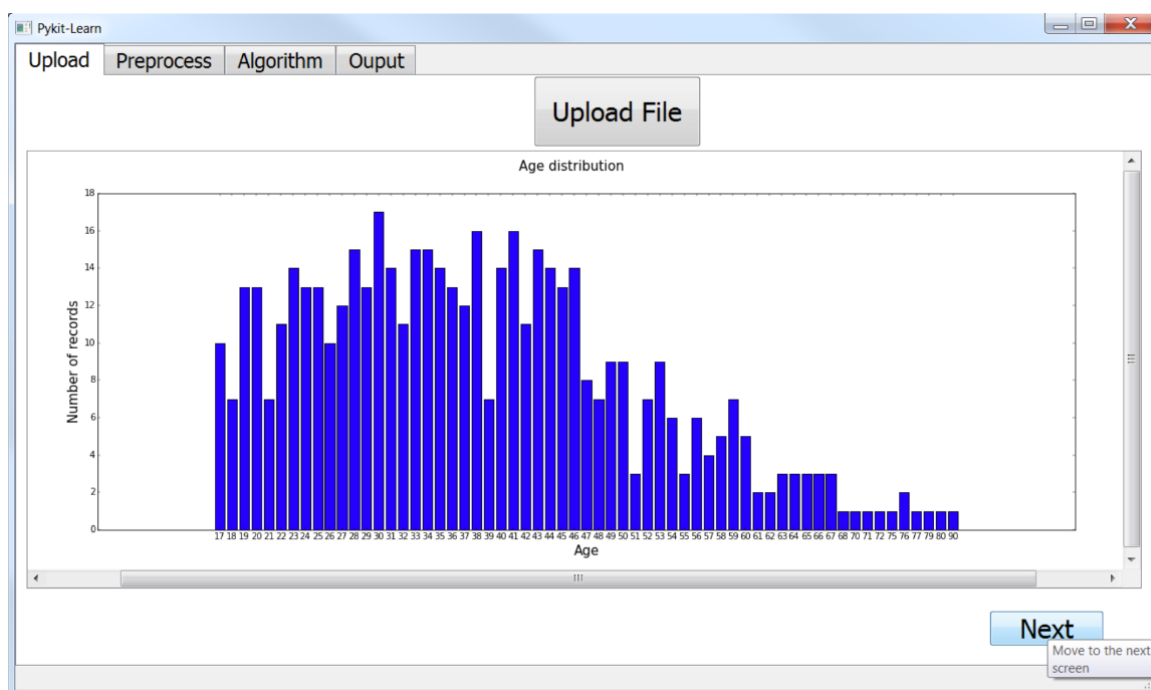


Рисунок 1.2. Интерфейс первого шага с загрузкой файла обучающей выборки

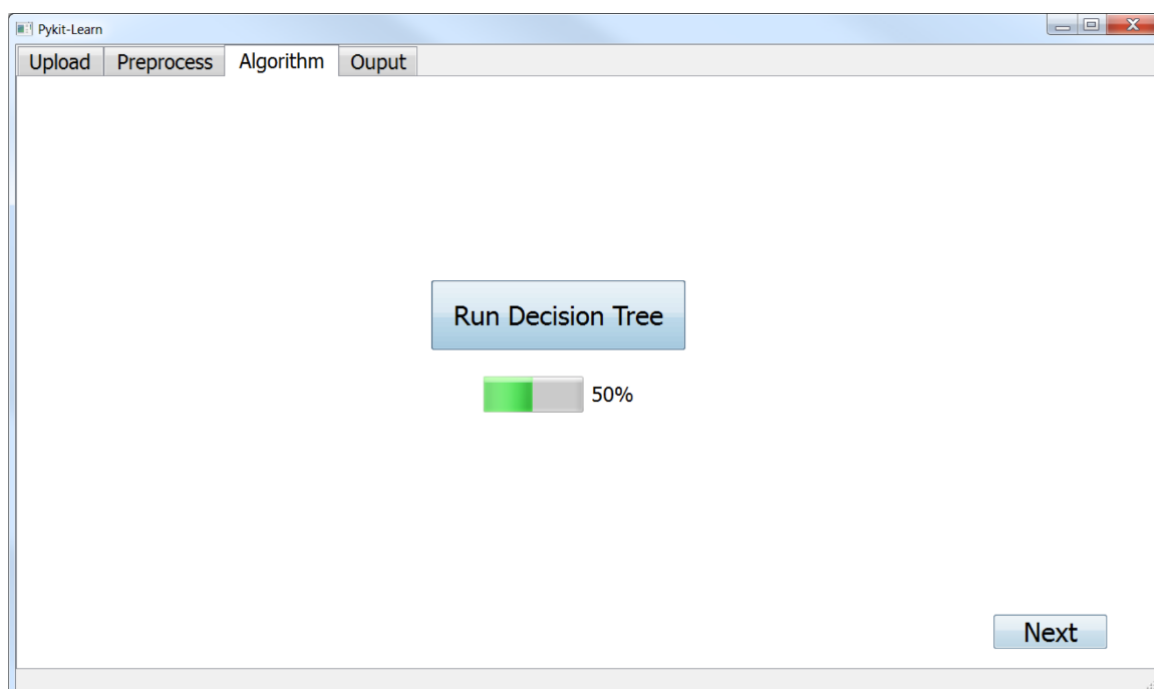


Рисунок 1.3. Интерфейс третьего шага с запуском алгоритма

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.

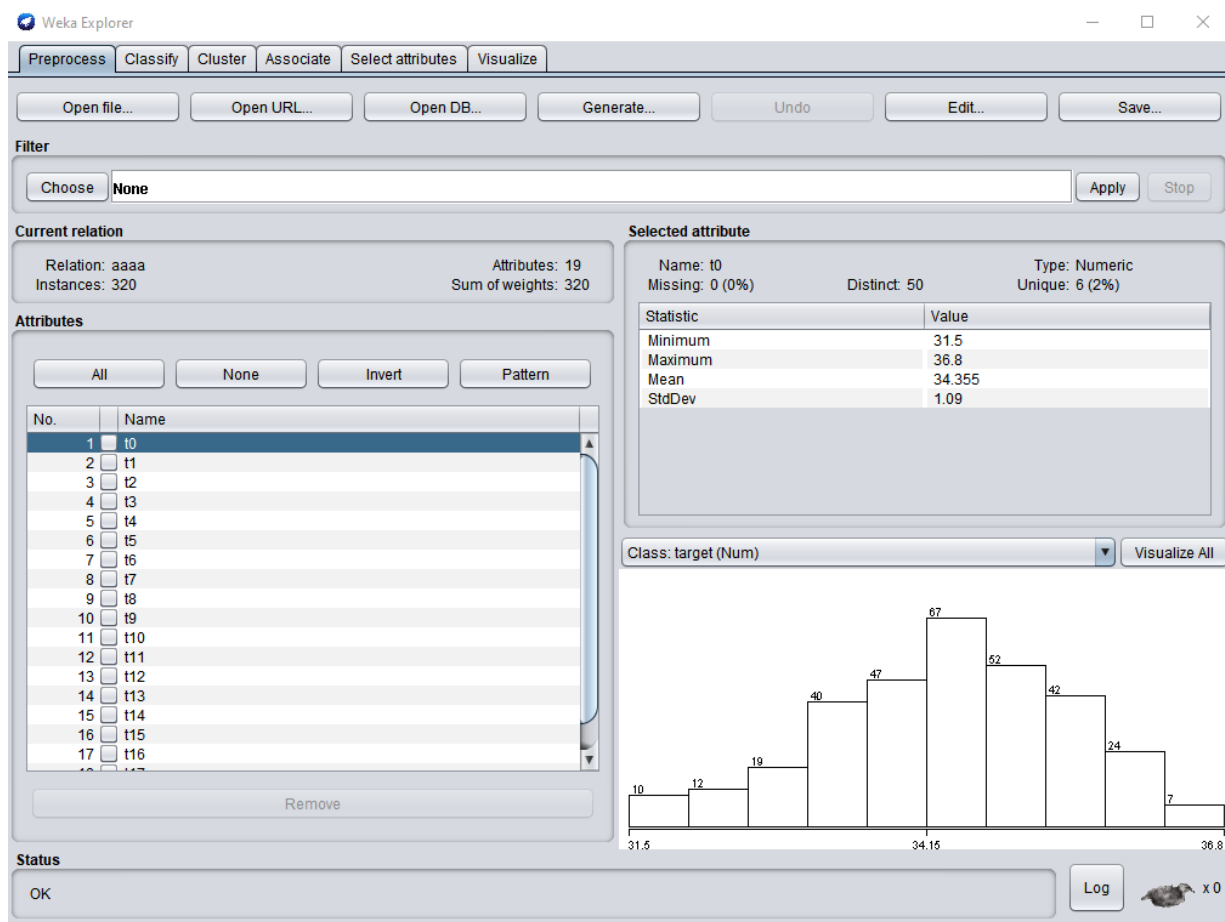


Рисунок 1.4. Интерфейс программного комплекса Weka - вкладка Preprocess

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

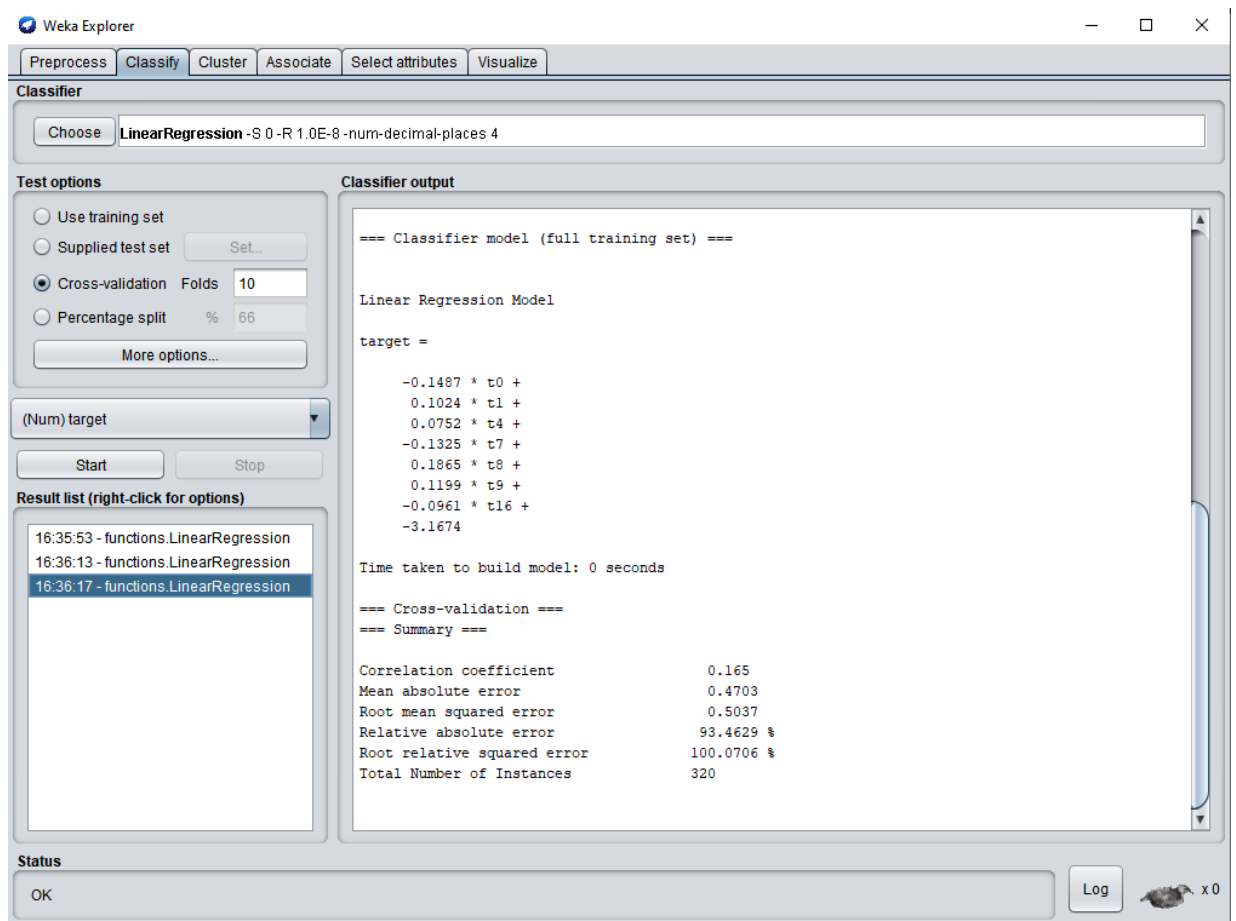


Рисунок 1.5. Интерфейс программного комплекса Weka - вкладка Classify

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

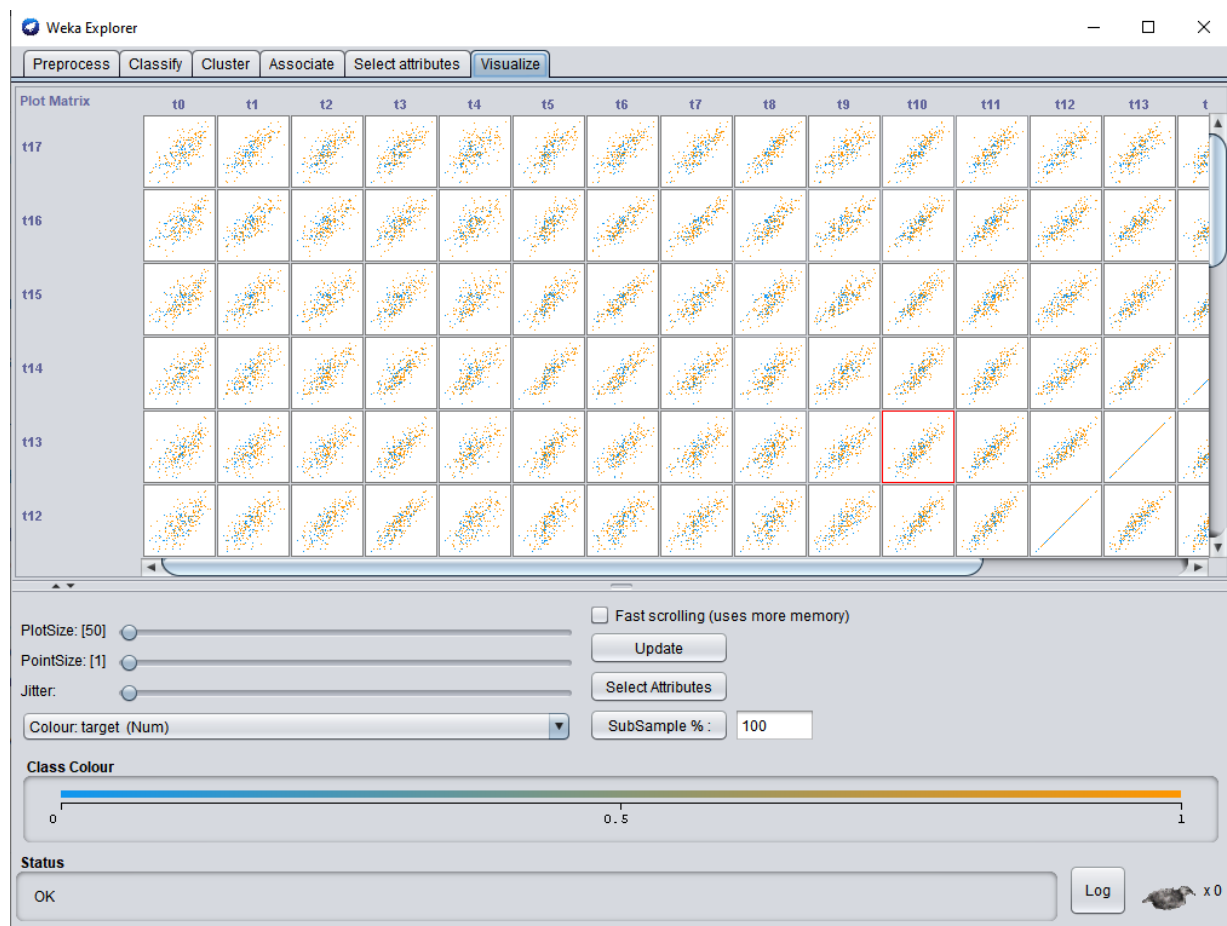


Рисунок 1.6. Интерфейс программного комплекса Weka - вкладка Visualize

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 пользовательского интерфейса для нейронной сети

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

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.

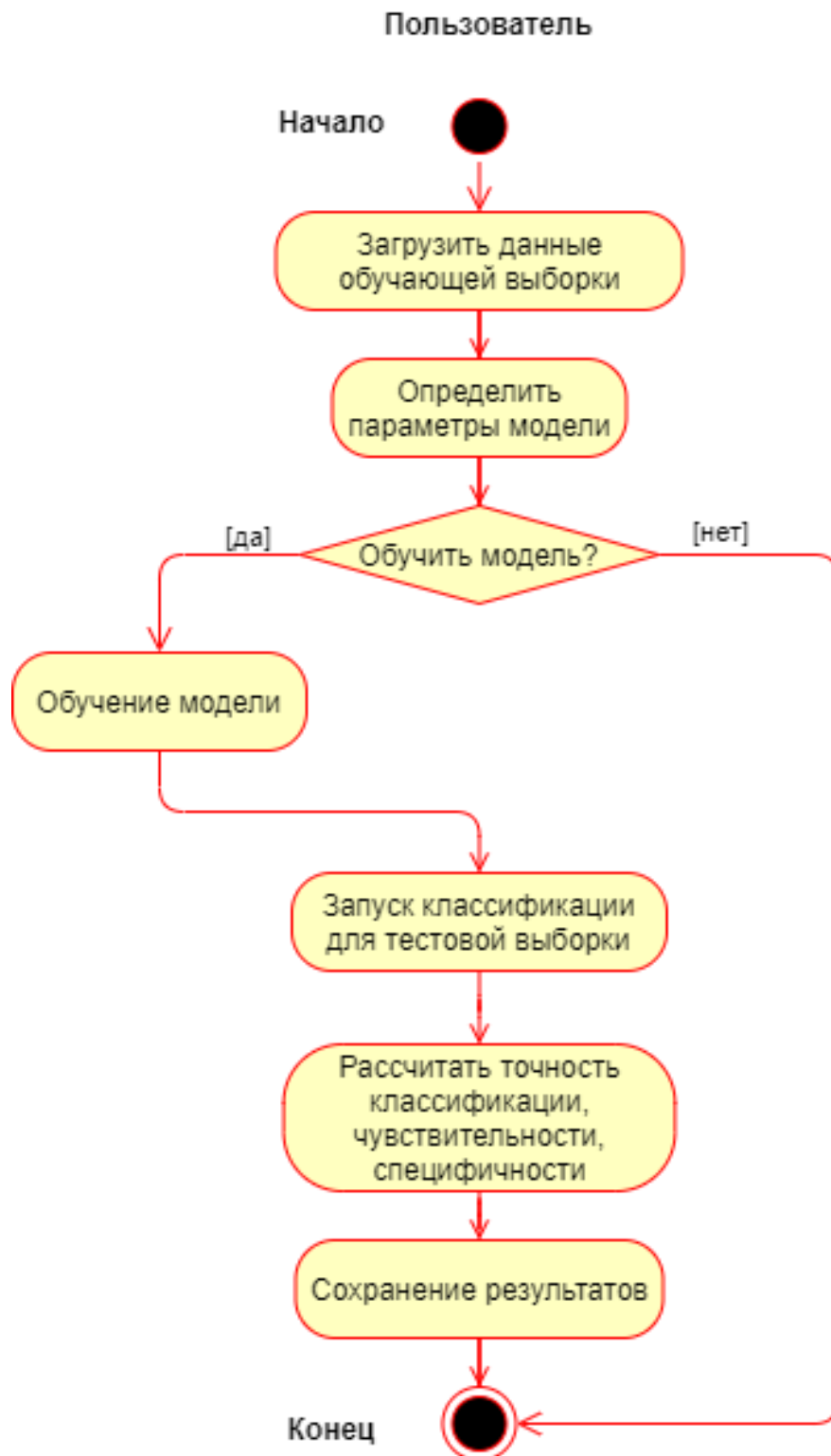


Рисунок 2.1. Диаграмма деятельности для программы

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 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 2.2).

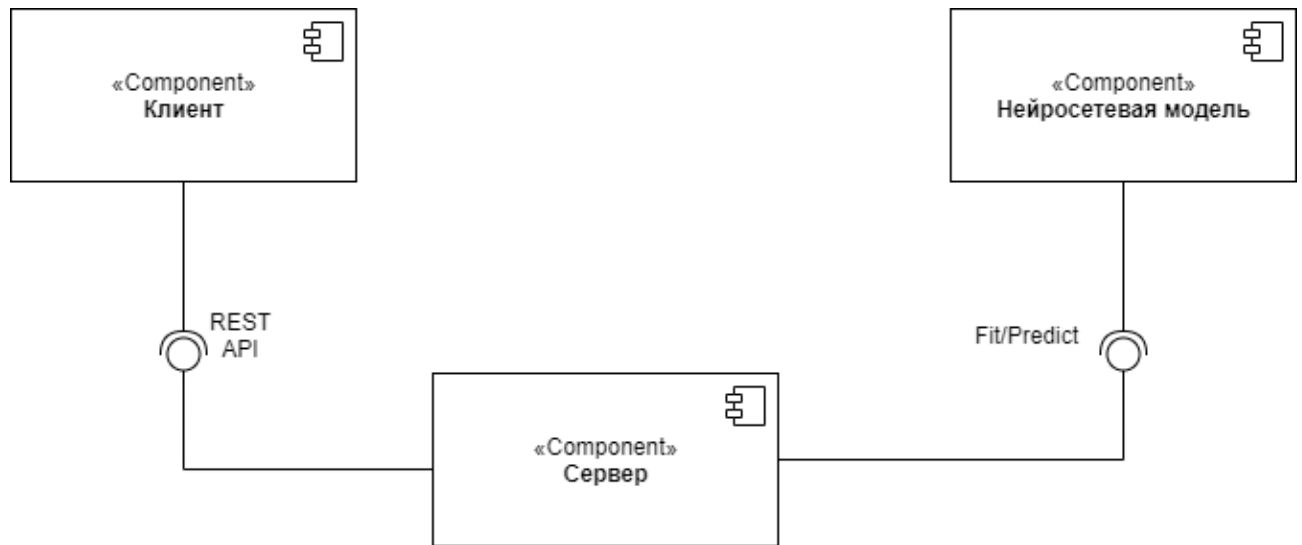


Рисунок 2.2. Диаграмма компонентов разрабатываемой системы

Схема взаимодействия клиента, сервера и нейросетевой модели представлено на диаграмме последовательности (рисунок 2.3). Для взаимодействия клиента и сервера был выбран протокол HTTP из-за большой поддержки во многих языках программирования, библиотеках и фреймворках.

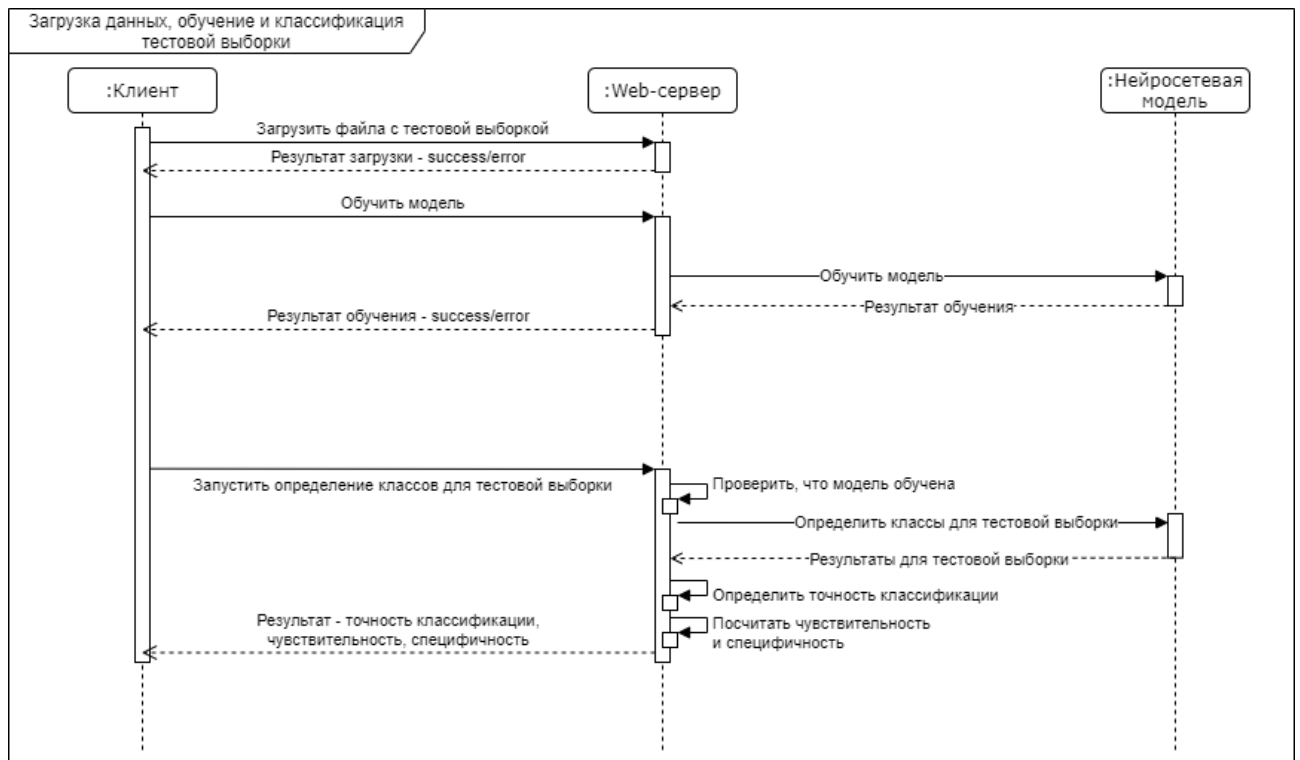


Рисунок 2.3. Диаграмма последовательности для разрабатываемого приложения

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.

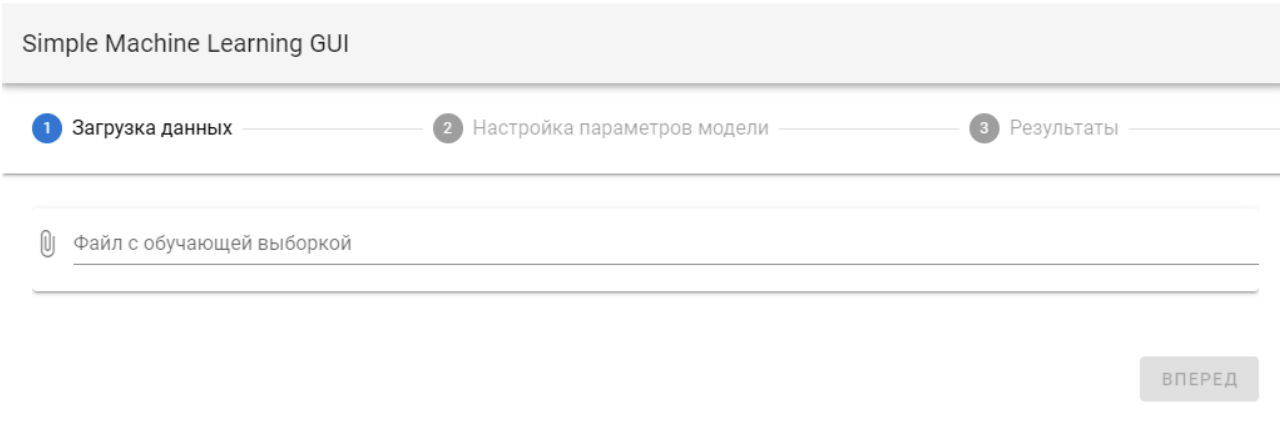
Для разработки современного и быстро работающего без перезагрузки страницы интерфейса был выбран язык программирования JavaScript и фреймворк VueJS. Приложение на VueJS состоит из отдельных компонентов, каждый из которых имеет свое состояние и свойства. Такой подход позволяет переиспользовать компоненты и удобно настраивать взаимодействие между ними, сохраняя возможность масштабирования при командной разра-

ботке.

2.2 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 2.4). Пока что поддерживаются только CSV-файлы.



The screenshot displays the 'Simple Machine Learning GUI' interface. At the top, the title 'Simple Machine Learning GUI' is shown. Below it, a progress bar indicates three steps: 1. 'Загрузка данных' (Data Upload), 2. 'Настройка параметров модели' (Model Parameter Configuration), and 3. 'Результаты' (Results). The first step is currently active. The main area features a file upload section with a document icon and the text 'Файл с обучающей выборкой' (File with training data). Below this is a large, empty rectangular box for the file. In the bottom right corner, there is a button labeled 'ВПЕРЕД' (Next).

Рисунок 2.4. 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 2.5.

Simple Machine Learning GUI

1 Загрузка данных 2 Настройка параметров модели 3 Результаты

Процент тестовой выборки 25 gamma scale 1.0

НАЗАД ВПЕРЕД

Рисунок 2.5. 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 2.6). Если пользователя не устроили получившиеся результаты, то он может вернуться на предыдущий шаг и изменить параметры модели.

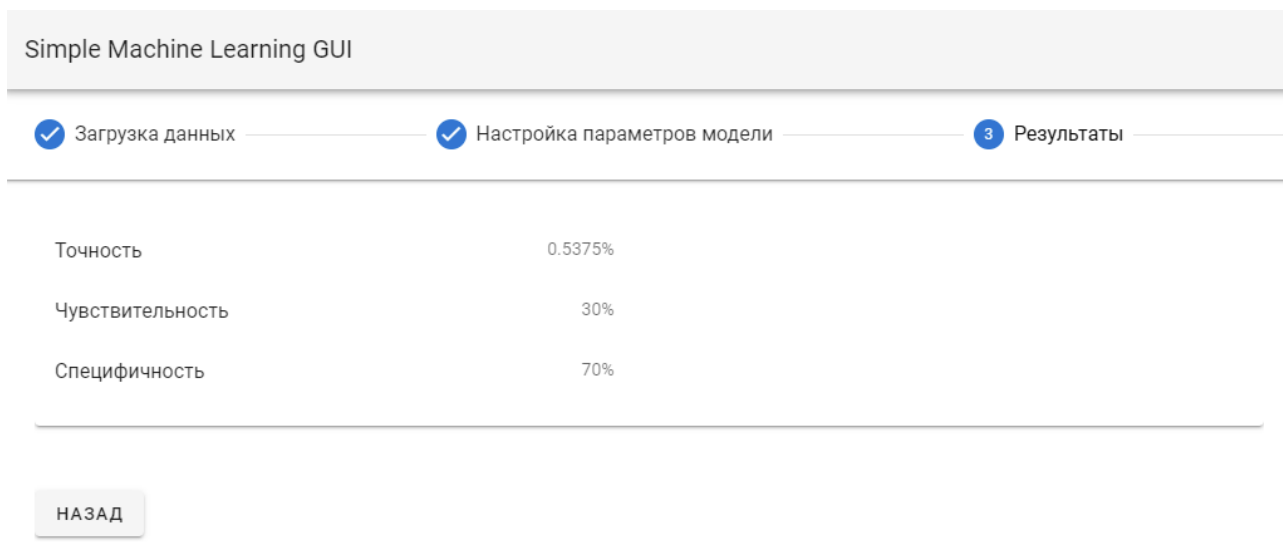


Рисунок 2.6. Скриншот разрабатываемого приложения на шаге “Результаты”

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.

2.3 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 2.7.

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

Рисунок 2.7. 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`.

2.4 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.1.

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. Пример листинга в приложении

```
1 /**
2  * Program main
3  */
4 int main(int argc, char **argv) {
5     printf("[Matrix Multiply Using CUDA] - Starting...\n");
6
7     if (checkCmdLineFlag(argc, (const char **)argv, "help") ||
8         checkCmdLineFlag(argc, (const char **)argv, "?")) {
9         printf("Usage -device=n (n >= 0 for deviceID)\n");
10        printf("-wA=WidthA -hA=HeightA (Width x Height of Matrix A)\n
11               ");
12        printf("-wB=WidthB -hB=HeightB (Width x Height of Matrix B)\n
13               ");
14        printf("Note: Outer matrix dimensions of A & B matrices" \
15               "must be equal.\n");
16    }
17
18    // This will pick the best possible CUDA capable device,
19    // otherwise
20    // override the device ID based on input provided
21    // at the command line
22    int dev = findCudaDevice(argc, (const char **)argv);
23
24    int block_size = 32;
25
26    dim3 dimsA(5 * 2 * block_size, 5 * 2 * block_size, 1);
27    dim3 dimsB(5 * 4 * block_size, 5 * 2 * block_size, 1);
```

```

27
28 // width of Matrix A
29 if (checkCmdLineFlag(argc, (const char **)argv, "wA")) {
30     dimsA.x = getCmdLineArgumentInt(argc, (const char **)argv, "
        wA");
31 }
32
33 // height of Matrix A
34 if (checkCmdLineFlag(argc, (const char **)argv, "hA")) {
35     dimsA.y = getCmdLineArgumentInt(argc, (const char **)argv, "
        hA");
36 }
37
38 // width of Matrix B
39 if (checkCmdLineFlag(argc, (const char **)argv, "wB")) {
40     dimsB.x = getCmdLineArgumentInt(argc, (const char **)argv, "
        wB");
41 }
42
43 // height of Matrix B
44 if (checkCmdLineFlag(argc, (const char **)argv, "hB")) {
45     dimsB.y = getCmdLineArgumentInt(argc, (const char **)argv, "
        hB");
46 }
47
48 if (dimsA.x != dimsB.y) {
49     printf("Error: outer matrix dimensions must be equal.
50         (%d != %d)\n", dimsA.x, dimsB.y);
51     exit(EXIT_FAILURE);
52 }
53
54 printf("MatrixA(%d,%d), MatrixB(%d,%d)\n", dimsA.x, dimsA.y,
55     dimsB.x, dimsB.y);
56
57 int matrix_result = MatrixMultiply(argc,

```

```
58         argv ,
59         block_size ,
60         dimsA ,
61         dimsB);
62
63     exit(matrix_result);
64 }
```