

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАЗРАБОТКА СИСТЕМЫ ДЛЯ КЛАССИФИКАЦИИ ПРОЕКТОВ GITHUB ПО ПОПУЛЯРНОСТИ

КУРСОВАЯ РАБОТА
по дисциплине «Программная инженерия»
ЮУрГУ – 09.03.04.2024.308-570.КР

Нормоконтролер,
доктор физ.-мат. наук
_____ М.Л. Цымблер
“ ____ ” _____ 2024 г.

Научный руководитель
доктор физ.-мат. наук
_____ М.Л. Цымблер

Автор работы,
студент группы КЭ-303
_____ Д.И. Гольденберг

Работа защищена с оценкой

_____ “ ____ ” _____ 2024 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ

Зав. кафедрой СП

_____ Л.Б. Соколинский

06.02.2024

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра
студентке группы КЭ-303
Гольденберг Дарье Игоревне,
обучающейся по направлению 09.03.04 «Программная инженерия»

1. Тема работы

Разработка системы для классификации проектов GitHub по популярности.

2. Срок сдачи студентом законченной работы: 05.06.2024.

3. Исходные данные к работе

- 3.1. Milovidov A., 2020. Everything You Ever Wanted To Know About GitHub (But Were Afraid To Ask). [Электронный ресурс] URL: <https://ghe.clickhouse.tech/>
- 3.2. Документация по использованию библиотеки Scikit learn. [Электронный ресурс] URL: https://scikit-learn.org/stable/user_guide.html
- 3.3. Soll M., Vosgerau M. ClassifyHub: An Algorithm to Classify GitHub Repositories. [Электронный ресурс] URL: https://doi.org/10.1007/978-3-319-67190-1_34

4. Перечень подлежащих разработке вопросов

- 4.1. Выполнить анализ предметной области и провести обзор существующих решений.
- 4.2. Выбрать репрезентативный набор признаков и подготовить данные по существующим репозиториям GitHub.
- 4.3. Исследовать различные модели машинного обучения и выбрать наиболее эффективные.
- 4.4. Разработать приложение, которое будет классифицировать проекты по нескольким уровням на основе выбранной модели.

5. Дата выдачи задания: 09.02.2024.

Научный руководитель

М.Л. Цымблер

Задание принял к исполнению

Д.И. Гольденберг

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	6
1.1. Описание предметной области	6
1.2. Обзор аналогов и существующих решений	8
1.3. Теоретический базис	11
2. ПРОЕКТИРОВАНИЕ	14
2.1. Требования к системе	14
2.2. Варианты использования системы	14
2.3. Архитектура системы	16
2.4. Проектирование пользовательского интерфейса	17
3. РЕАЛИЗАЦИЯ	20
3.1. Программные средства реализации	20
3.2. Реализация компонентов приложения	20
3.3. Реализация пользовательского интерфейса	23
4. ТЕСТИРОВАНИЕ	27
4.1. Функциональное тестирование	27
4.2. Вычислительные эксперименты	27
ЗАКЛЮЧЕНИЕ	34
ЛИТЕРАТУРА	35

ВВЕДЕНИЕ

Актуальность

С развитием технологий в сфере информационных технологий и программного обеспечения объем данных, генерируемых и публикуемых в репозиториях, таких как GitHub, Bitbucket, Gogs и др, продолжает стремительно расти. GitHub является крупнейшей платформой для хостинга и совместной разработки кода, предоставляющей доступ к миллионам проектов, созданных сообществом разработчиков со всего мира. Столь огромный объем репозиторий создает необходимость в разработке эффективных методов и инструментов для их классификации и анализа.

Одним из ключевых аспектов, определяющих значимость проекта, является его популярность, которая может влиять на привлекательность проекта для потенциальных участников, уровень вовлеченности сообщества разработчиков, а также на его долгосрочную жизнеспособность. Поэтому разработка системы, способной автоматически классифицировать проекты GitHub по их популярности, имеет огромное значение для исследования и индустрии разработки программного обеспечения.

В настоящее время отсутствуют универсальные и эффективные методы оценки популярности проектов на платформе GitHub, а существующие подходы могут предоставить лишь ограниченное представление о проектах. В свете этой проблемы разработка системы для классификации проектов GitHub по популярности становится значимой задачей. Такая система должна учитывать разные факторы, влияющие на популярность проекта, и применять современные методы машинного обучения для создания более точного и всестороннего представления о популярности проектов на GitHub.

Постановка задачи

Целью данной курсовой работы является разработка системы для классификации проектов на платформе GitHub по уровню их популярности. Для достижения этой цели были сформулированы следующие задачи.

- 1) Выполнить анализ предметной области и провести обзор существующих решений.
- 2) Выбрать репрезентативный набор признаков и подготовить дан-

ные по существующим репозиториям GitHub.

3) Исследовать различные модели машинного обучения и выбрать наиболее эффективные.

4) Разработать приложение, которое будет классифицировать проекты по нескольким уровням на основе выбранной модели.

Структура и содержание работы

Курсовая работа состоит из введения, четырех разделов, заключения и списка литературы. Объем текста составляет 38 страниц, объем списка литературы – 33 наименования.

В первом разделе описывается предметная область, а также проводится анализ существующих аналогов и методов, решающих поставленные задачи курсовой работы, и теоретический базис относительно моделей обучения.

Во втором разделе описываются функциональные и нефункциональные требования к разрабатываемому приложению. Приведены варианты использования программы, архитектура приложения, его компоненты и макет пользовательского интерфейса.

В третьем разделе содержится описание программных средств, используемых в процессе реализации системы, описание реализации определения класса популярности проекта (звезд), а также пользовательского интерфейса.

В четвертом разделе представлены функциональное тестирование приложения, а также и оценка точности полученных результатов согласно метрикам машинного обучения.

В заключении представлены основные результаты выполненной работы и направления, в которых возможны дальнейшие исследования.

В приложениях содержатся спецификация диаграммы вариантов использования системы, интерфейсы и листинги основных модулей, а также результаты матрицы ошибок моделей и сравнение метрик в соответствии со временем выполнения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Сайты для хостинга открытого исходного кода, такие как GitHub, например, GitLab, Bitbucket и др, представляют собой цифровые площадки, где разработчики могут хранить, управлять и совместно работать над своими проектами с открытым доступом к исходному коду.

Одной из основных причин популярности таких платформ является идеология открытого исходного кода. Распространение проектов с открытым исходным кодом способствует коллективному развитию программного обеспечения, позволяя разработчикам из разных уголков мира вносить свой вклад, исправлять ошибки и улучшать функционал. Это приводит к созданию более качественного, надежного и инновационного программного обеспечения.

Сами сайты основаны на системах контроля версий, которые являются ключевым инструментом в разработке программного обеспечения, обеспечивая эффективное управление изменениями в исходном коде проекта. Они позволяют разработчикам отслеживать и сохранять историю изменений в коде, а также управлять конфликтами и совмещать работу множества разработчиков. Согласно рейтингу Tagline [33] самой популярной системой контроля версий считается Git, что представлено на рисунке 1.

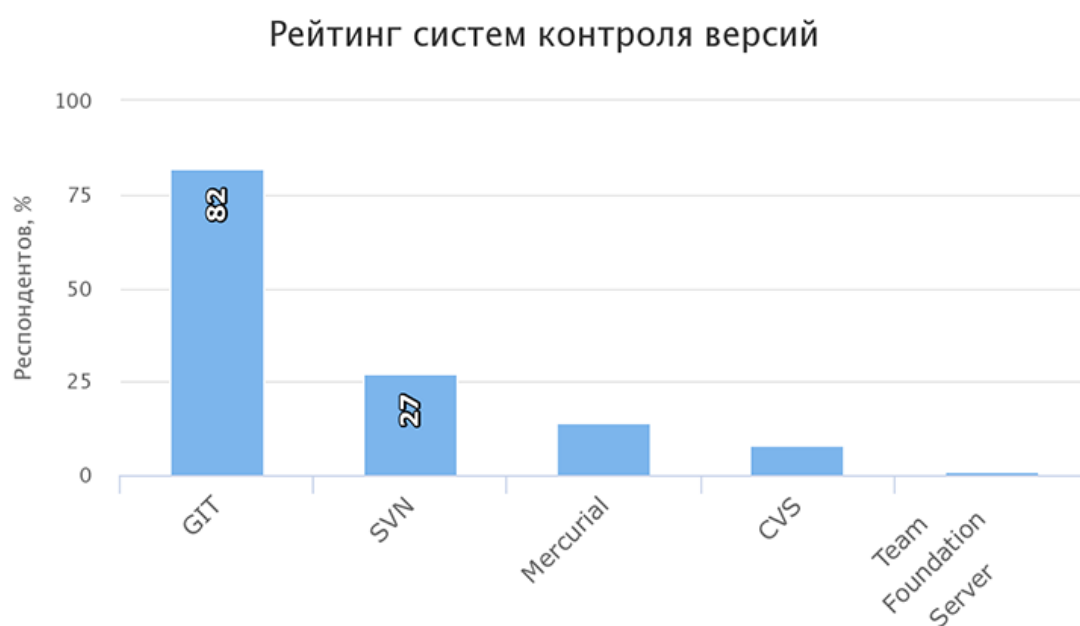


Рис. 1. Рейтинг систем контроля версий

Git, созданный Линусом Торвальдсом в 2005 году, стал стандартом для управления исходным кодом в различных проектах, от небольших локальных до масштабных проектов с открытым исходным кодом, таких как ядро Linux и сам GitHub.

GitHub – это веб-платформа для хранения и совместной разработки проектов с открытым исходным кодом. С момента своего запуска в 2008 году GitHub стал одной из самых популярных и важных платформ для разработчиков по всему миру. Его привлекательность обусловлена множеством факторов. Так, издание The New Stack [12] собрало статистику по используемым хостингам кода на основе Git за 2018-2019 года (рис. 2). Несмотря на падение популярности GitHub в 2019 году по сравнению с 2018, он всё ещё остается самым используемым сайтом для размещения своего кода в рамках репозитория.

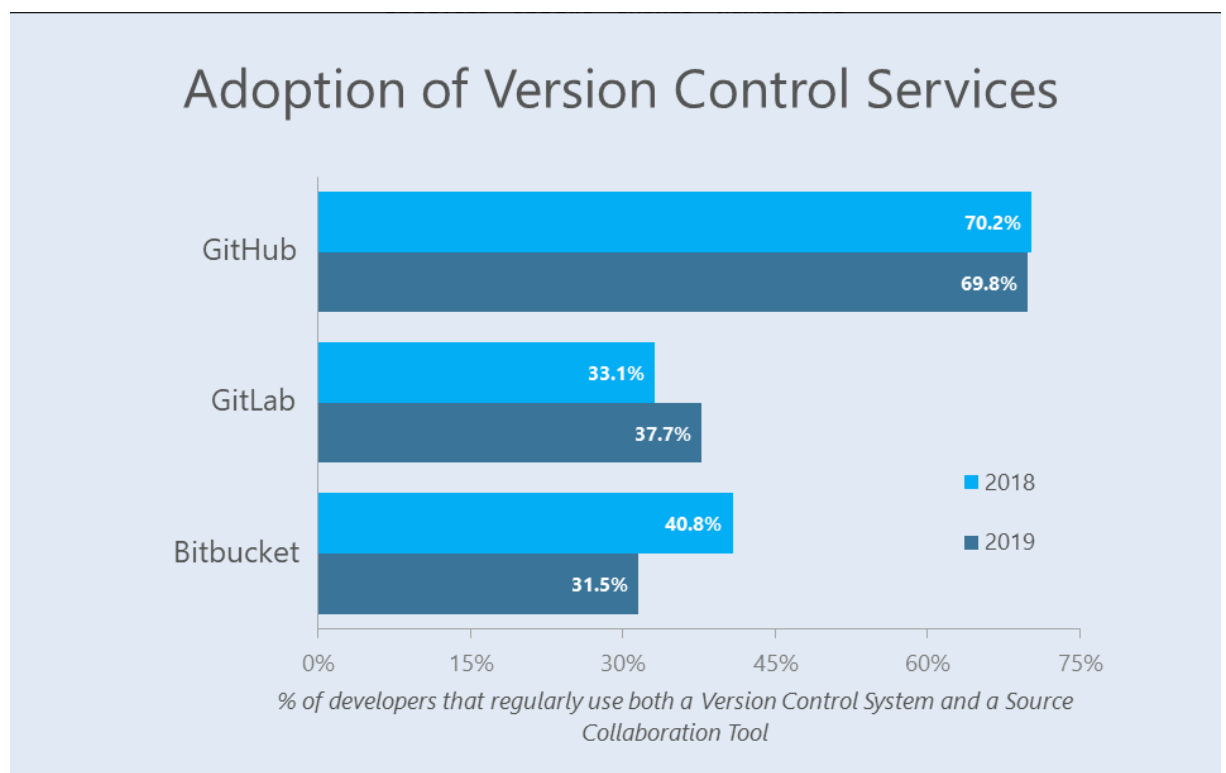


Рис. 2. Рейтинг хостингов кода на основе Git

Каждый проект на GitHub представлен в виде репозитория, где хранятся файлы проекта и история их изменений. Для отслеживания действий других пользователей с текущим репозиторием на GitHub используются различные параметры.

- Звезды (Stars): Звезды представляют собой показатель популярности проекта. Пользователи GitHub могут добавлять звезды к проектам, которые им нравятся или которые они хотят отслеживать. Чем больше звезд у проекта, тем он популярнее.
- Форки (Forks): Форки представляют собой копии репозитория проекта, созданные другими пользователями. Форк может использоваться для внесения изменений в проект без изменения оригинального кода. Чем больше форков у проекта, тем больше разработчиков заинтересовано в его доработке и участии.
- Проблемы (Issues): Проблемы представляют собой список задач, багов или идей для улучшения проекта. Количество проблем может служить показателем активности и вовлеченности сообщества в развитие проекта.
- Запросы на слияние (Pull Requests): Запросы на слияние представляют собой предложения от разработчиков о внесении изменений в код проекта. Они могут быть использованы для исправления ошибок, добавления нового функционала или внесения других улучшений.

Все эти метрики помогают разработчикам оценивать популярность и активность проектов на GitHub. Звезды и форки позволяют судить о популярности проекта среди пользователей, а проблемы и запросы на слияние – об активности и вовлеченности сообщества разработчиков.

Использование GitHub и его метрик стало стандартной практикой в сообществе разработчиков благодаря его открытости, удобству использования и широким возможностям для совместной работы и обмена знаниями.

1.2. Обзор аналогов и существующих решений

Исследования, посвященные анализу данных, связанных с использованием платформы GitHub и разработкой программного обеспечения на этой платформе, являются актуальной и распространенной темой для научных исследований.

Так, авторы статьи [1] обсуждают проблему классификации отзывов о приложениях, которые содержат важную информацию о потребностях пользователей. Исследователи предлагают подход для создания более

обобщенной модели, используя информацию из системы отслеживания задач GitHub, которая содержит ценные данные о потребностях пользователей. После проведения экспериментов, они показывают, что использование размеченных задач из GitHub может улучшить точность и полноту классификации отзывов, особенно для отчетов о багах и запросов на новые функции.

В работе [22] поднимается важность репозитория программного обеспечения для управления проектами, включая исходный код, документацию и отчеты об ошибках. Особое внимание уделяется платформе GitHub, которая для помощи разработчикам в поиске подходящих артефактов использует темы (topics), являющимися короткими текстами, присваиваемыми хранимым артефактам. Однако неправильное присвоение тем может негативно сказаться на популярности репозитория.

Закария Альшара и другие авторы в своей статье [2] рассматривают проблему управления задачами (issues) на платформе GitHub, особенно в случае быстрого роста числа создаваемых задач. Для помощи разработчикам в обработке задач существуют внешние участники, которые исправляют задачи, создавая pull-запросы (Pull Requests, они же PR). Однако часто такие PR не связываются с соответствующими задачами (issues), что затрудняет управление проектом. В статье предлагается использование моделей машинного обучения (ML) для автоматического восстановления связей между PR и задачами на GitHub. Установление связей между PR и задачами ценно, так как это помогает улучшить управление разработкой и обслуживанием проектов, что влияет на популярность и развитие проекта в дальнейшем.

В работе [21] исследует, как проводится кодирование в области науки о данных на GitHub. Авторы анализируют, как данные ученые переходят между разными этапами работы с данными. Результаты исследования показывают, что кодирование имеет определенные паттерны. Кроме того, авторы попытались обучить модели машинного обучения для предсказания этапов работы с данными и достигли точности примерно 71%.

О популярных проектах говорят в статье Джесси Айала и ее коллеги [3], а именно о важности использования непрерывной интеграции и по-

ставки (CI/CD) и политики безопасности в известных и пользующихся интересом проектах с открытым исходным кодом, особенно на GitHub. Исследование показало, что многие проекты не активно используют эти возможности, и призывает управляющих таких проектов уделить им больше внимания для предотвращения уязвимостей.

Кроме того, в исследовании [20] фокусируются на том, как документируется информация о функциональных возможностях программного обеспечения в проектах на GitHub и связана ли она с исходным кодом. Авторы провели анализ 25 популярных репозиторий на GitHub и обнаружили, что хотя документация о функциональности часто присутствует в различных текстовых файлах, она часто неструктурирована, и связь с исходным кодом редко устанавливается, что может привести к затруднениям в его поддержке на долгосрочной перспективе.

Статья [7] рассматривает инструмент GitcProc, который предназначен для анализа проектов на GitHub. Этот инструмент позволяет извлекать информацию о разработке, включая исходный код и историю исправления ошибок. GitcProc может отслеживать изменения в исходном коде и связывать их с функциями с минимальными настройками. Он успешно работает с проектами на разных языках программирования, обнаруживая исправления ошибок и контекст изменений в коде.

Помимо этого, стоит также рассмотреть работу [27], где авторы обсуждают задачу классификации репозиторий на GitHub, которая представляет собой сложную задачу. Они представляют алгоритм ClassifyHub, основанный на методах ансамблирования, разработанный для соревнования InformatiCup 2017. Этот алгоритм успешно решает задачу классификации с высокой точностью и полнотой, что может быть полезно для различных приложений, таких как рекомендательные системы.

Самой приближенной к поставленной задаче статьей является «A Cross-Repository Model for Predicting Popularity in GitHub» [5], в которой рассматривается создание модели для прогнозирования популярности репозиторий на GitHub, используя данные из разных репозиторий. Модель, основанная на рекуррентной нейронной сети LSTM, позволяет более точно предсказывать популярность, чем стандартные методы прогнозирования.

ния временных рядов на основе данных из одного репозитория.

Кроме неё, есть также исследование [9], в котором предлагают метод для прогнозирования популярности проектов на GitHub. Он использует 35 признаков, извлеченных из GitHub и Stack Overflow, чтобы классифицировать проекты как популярные или нет. Основными признаками для определения популярности оказались количество веток, количество открытых задач и количество участников проекта.

1.3. Теоретический базис

В этом разделе рассматриваются основные концепции и методы машинного обучения, которые лежат в основе разработки моделей для классификации и регрессии. Машинное обучение – это раздел искусственного интеллекта, который изучает методы анализа данных и построения предсказательных моделей без явного программирования. Одним из ключевых направлений в машинном обучении является обучение с учителем, где модели создаются на основе предварительно размеченных данных. [26] Чаше встречаются модели двух типов: модель классификации и модель регрессии.

Модели классификации являются методами обучения с учителем, цель которых заключается в том, чтобы прогнозировать принадлежность объекта к одной из заданных категорий или классов на основе набора предварительно размеченных данных. Модели классификации используют алгоритмы, которые находят закономерности в данных и создают функцию, относящую новый объект к одному из заранее определенных классов.

Рассмотрим модели, которые будут использованы для работы.

Decision Tree (Дерево решений)

Дерево решений строит структуру в виде дерева, в которой каждый узел представляет собой тест на значение определенного признака, а каждое ребро - результат этого теста. [13] Для построения дерева используются различные алгоритмы, например, алгоритм CART (классификация и регрессия на основе дерева). Процесс построения дерева продолжается, пока не будет выполнен какой-то критерий останова, например, достигнута максимальная глубина дерева или узел содержит объекты только одного класса.

Random Forest (Случайный лес)

Random Forest строит ансамбль деревьев решений, каждое из которых обучается на случайной подвыборке данных. [6] В процессе построения дерева для каждого разбиения выбирается случайный поднабор признаков. После построения всех деревьев в ансамбле, предсказание производится путем агрегации результатов всех деревьев, например, путем голосования (в случае классификации) или усреднения (в случае регрессии).

Gradient Boosting (Градиентный бустинг)

Gradient Boosting последовательно обучает слабые модели, каждая из которых предсказывает остатки (разницу между истинными значениями и предсказанными значениями) предыдущей модели. [10] На каждом шаге минимизируется функция потерь (например, среднеквадратичная ошибка для задачи регрессии) с помощью градиентного спуска. Таким образом, каждая новая модель исправляет ошибки предыдущей модели.

AdaBoost

AdaBoost обучает слабые модели на последовательно изменяющихся весах обучающих образцов. [11] На каждой итерации модель фокусируется на объектах, которые были классифицированы неправильно на предыдущих итерациях. После обучения всех моделей, их прогнозы комбинируются с помощью взвешенного голосования.

Gaussian Naive Bayes (Наивный байесовский классификатор)

Этот классификатор основан на применении теоремы Байеса с допущением о независимости между признаками. [29] Он предполагает, что каждый признак влияет на класс независимо от других признаков. При классификации модель оценивает вероятность принадлежности объекта к каждому классу и выбирает класс с наибольшей вероятностью.

Модели регрессии также являются методами обучения с учителем, но в отличие от классификации, их цель состоит в прогнозировании непрерывных значений целевой переменной на основе имеющихся данных. Модели регрессии стремятся к поиску математической функции, которая наилучшим образом описывает связь между входными признаками и выходными значениями.

Linear Regression (Линейная регрессия)

Линейная регрессия моделирует линейную зависимость между предикторами и целевой переменной. Она находит коэффициенты линейной функции таким образом, чтобы минимизировать сумму квадратов разностей между наблюдаемыми и предсказанными значениями.

Для моделей регрессии Decision Tree, Random Forest, Gradient Boosting главное отличие от их использования в классификации заключается в том, как они строят свои прогнозы и как оценивают качество модели. [18] Так, Случайный лес после обучения всех деревьев результаты агрегируются, например, путем усреднения, чтобы получить окончательный прогноз, а для регрессионных Деревьев решений обычно используется критерий ветвления, такой как среднеквадратичная ошибка.

2. ПРОЕКТИРОВАНИЕ

В ходе проектирования приложения были определены следующие функциональные и нефункциональные требования.

2.1. Требования к системе

Функциональные требования. Функциональные требования описывают, какое поведение должна предоставлять разрабатываемая система.

- Система должна проводить классификацию проектов на основе параметров репозитория.
- Пользователи должны иметь возможность просматривать результаты классификации и информацию о проекте.
- Система должна предоставлять интерфейс для взаимодействия с пользователем.

Нефункциональные требования.

К нефункциональным требованиям системы относятся свойства, которыми она должна обладать. Например, удобство использования, безопасность и т.д.

- Приложение должно иметь интуитивно понятный в использовании пользовательский интерфейс.
- Основная программная реализация осуществляется с использованием Python.
- Пользовательский интерфейс должно быть реализовано с использованием веб-сайта.

2.2. Варианты использования системы

Для проектирования приложения был использован язык графического описания для объектного моделирования UML. Была построена модель взаимодействия внешнего актера с приложением в виде диаграммы вариантов использования. В ходе анализа разрабатываемого приложения были выявлены основные варианты использования (рисунок 3).

Для данной диаграммы определен актер – пользователь, который взаимодействует с приложением. Пользователю доступны следующие варианты использования.



Рис. 3. Варианты использования приложения

1) Вариант использования "Выбрать тип модели (критерий)". Пользователь может выбрать тип модели, который будет использоваться для классификации или регрессии. Выбор типа модели определяет метод, который будет применен к входным данным для определения класса популярности или количества звезд.

2) Вариант использования "Выполнить определение". Пользователь может запустить процесс определения класса популярности или количества звезд на основе выбранной модели и входных данных. Данный прецедент включает в себя вариант "Выбрать модель".

3) Вариант использования "Выбрать модель". Пользователь может выбрать конкретную модель из представленных в системе. Представленные модели могут включать в себя такие методы, как Градиентный бустинг, Случайный лес и другие.

4) Вариант использования "Визуализировать результат". Пользователь может визуализировать результаты, представляющие собой класс популярности или количество звезд относительно перечисленных парамет-

ров на веб-странице.

Детальная спецификация вариантов использования приложения представлена в приложении А.

2.3. Архитектура системы

В данном разделе рассматривается спроектированная архитектура приложения в виде диаграммы компонентов, которая показывает разбиение системы на структурные компоненты.

Спроектированная архитектура приложения представлена на рисунке 4 в виде диаграммы компонентов.

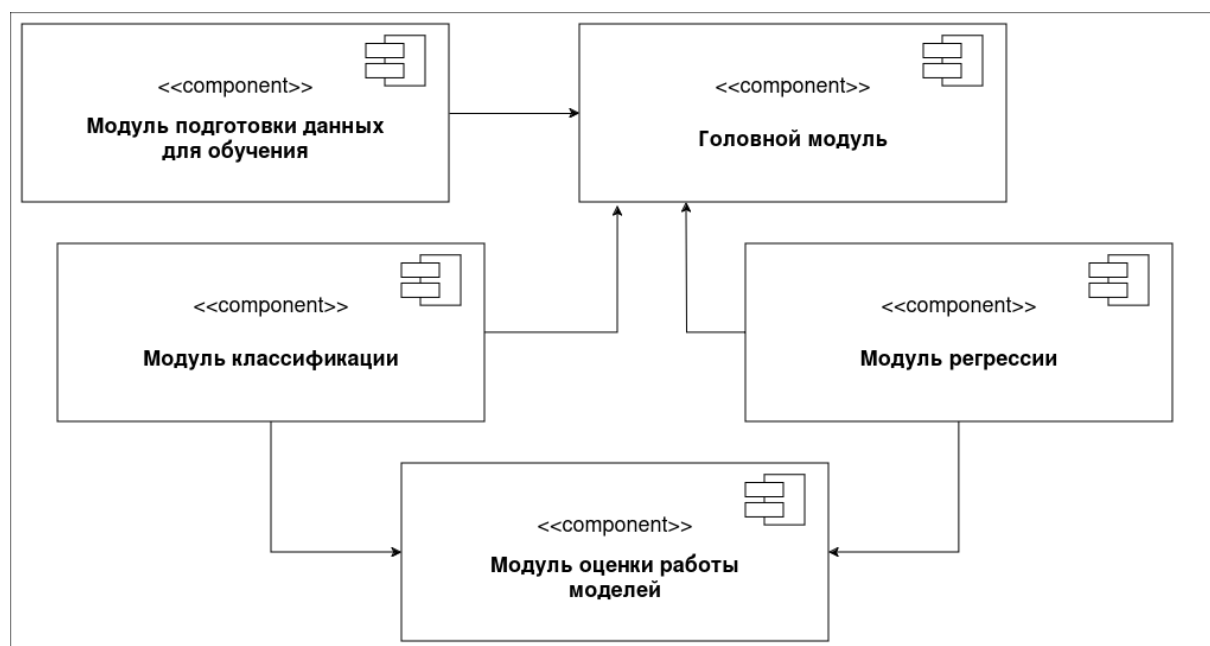


Рис. 4. Архитектура приложения

Головной модуль представляет собой главный модуль для отображения работы с работой веб-интерфейсом, с которым пользователь может взаимодействовать, и осуществляет работу модулей по работе с моделями.

Модуль подготовки данных для обучения отвечает за подготовку и предобработку данных, необходимых для обучения моделей машинного обучения. Включает в себя различные этапы предварительной обработки данных, такие как удаление выбросов, выбор и сортирование критериев репозитория.

Модуль классификации отвечает за определение класса популярно-

сти проекта. Он принимает на вход предварительно подготовленные данные и выбранную модель классификации, а затем определяет класс популярности проекта по модели. Модуль классификации также может работать с модулем оценки работы за счет выбранной модели.

Модуль регрессии отвечает за прогнозирование количества звезд относительно других параметров проекта с использованием различных моделей машинного обучения. Он принимает на вход предварительно подготовленные данные и выбранную модель регрессии, а затем выполняет прогнозирование целевых числовых значений. Модуль регрессии также может работать с модулем оценки работы за счет выбранной модели, т.е. проводить оценку качества регрессии.

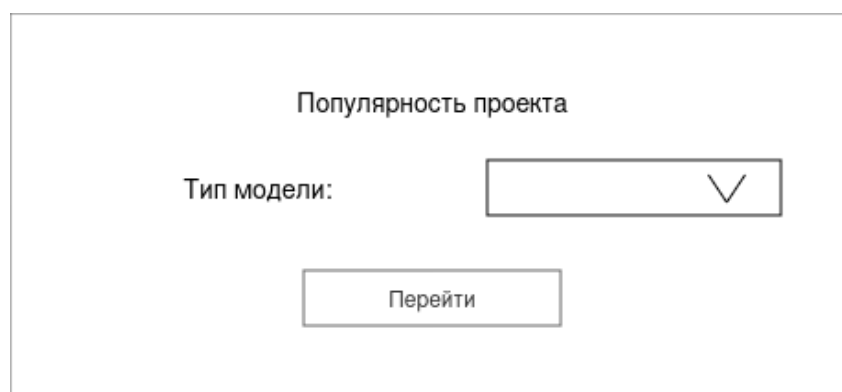
Модуль оценки работы моделей отвечает за оценку эффективности работы различных моделей машинного обучения. Включает в себя запуск алгоритмов оценки с использованием предварительно загруженных данных, разделенных на обучающий и тестовый наборы. Модуль оценки работы моделей запускает процессы обучения и тестирования моделей классификации и регрессии, сравнивая их результаты с заданными метриками, такими как точность, полнота, F1-мера, MSE и другими.

2.4. Проектирование пользовательского интерфейса

В данном разделе будут представлены спроектированные макеты пользовательского интерфейса приложения. Данные макеты являются примерным представлением итогового продукта и содержат в себе основные необходимые функции.

Для разрабатываемого приложения было решено создать простой и интуитивно понятный интерфейс для удобства пользователя.

Главная страница содержит в себе форму выбора типа модели: модели классификации, которая отвечает за определение класса популярности проекта по заданным параметрам, и модели регрессии, которая прогнозирует возможный результат значения параметра "звезды". Также на форме будет кнопка для перехода на страницу с вводом параметров и выбранным типом модели. Макет формы на главной странице представлен на рисунке 5.

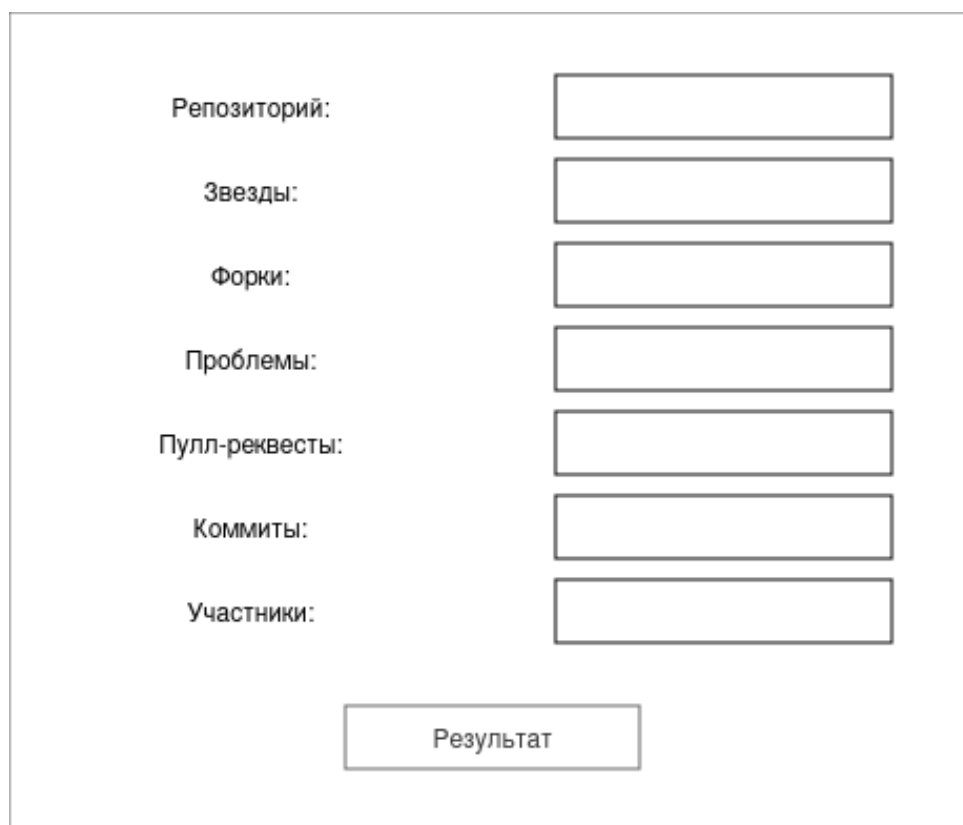


Популярность проекта

Тип модели:

Рис. 5. Макет формы выбора типа модели

Страница для моделей классификации и моделей регрессии выглядит одинаково, содержит в себе форму для отправки данных. Окно содержит поля для данных по параметрам проекта, включая имя репозитория и численных характеристики. А также выбор модели из выпадающего списка для определения класса популярности и кнопку, которая переадресует на другую страницу с результатами. На странице модели классификации также содержится поле для ввода звезд. На рисунке 6 представлен макет формы ввода.



Репозиторий:

Звезды:

Форки:

Проблемы:

Пулл-реквесты:

Коммиты:

Участники:

Рис. 6. Макет формы ввода параметров проекта

Страница с результатом определения класса популярности проекта содержит в себе форму с определенным классом в виде числа от 0 до 10. Так на странице выведена та же информация, что была представлена для классификации, в качестве указания на то, что система верно интерпретировала данные. На рисунке 7 представлен макет результата определения класса. Страница с определением количеством звезд проекта представлена аналогично.

Результат

Класс популярности вашего проекта: 0, где

0 -- не популярен
10 -- очень популярен

Репозиторий: 'name'

1 Форк

1 Проблема

1 Пулл-реквест

1 Коммит

1 Участник

Рис. 7. Макет формы результата класса популярности

3. РЕАЛИЗАЦИЯ

3.1. Программные средства реализации

Для разработки программной части приложения использовался высокоуровневый язык программирования Python версии 3.9.5. Программирование осуществлялось в редактор исходного кода Visual Studio Code [28].

3.2. Реализация компонентов приложения

В данном разделе приводится описание реализации отдельных компонентов приложения. Код программной части представлен в репозитории на GitHub [32].

Реализация модуля подготовки данных

Исходными данными для этой работы служит статья о наборе данных GitHub [17], содержащем информацию о всех событиях на этой платформе с 2011 года и насчитывающем более трех миллиардов записей. Данный набор данных был загружен в ClickHouse, который представляет собой открытую систему управления базами данных, специально разработанную для эффективного анализа и хранения больших объемов информации.

Текущий датасет содержит множество полей, где каждая запись соответствует определенной действия, выполненной пользователем в одном из репозиториях. В частности, существует столбец «event_type», который описывает различные операции. Для данной работы имеют вес следующие операции: «CommitCommentEvent» (добавление коммита с комментарием), «ForkEvent» (создание форка репозитория), «IssuesEvent» (добавление обсуждения), «PullRequestEvent» (создание запроса на включение изменений), и «WatchEvent» (добавление звезды к проекту). В контексте работы с проектами, необходимо сгруппировать выполненные операции для каждого проекта. Следовательно, с использованием запросов можно получить информацию о названии проекта (репозитория) и количестве звезд, форков и других характеристик, связанных с данным репозиторием.

Помимо того, для демонстрации работы с общедоступными данными ClickHouse существует веб-страница, способная обрабатывать SELECT-запросы и предоставлять обширный объем информации. В силу того, что

объем датасета более 80 ГБ, данные из него было решено получать с помощью веб-приложения, поэтому необходимо разработать способ, которым можно корректно их преобразовать.

Для реализации извлечения данных из датасета используется библиотека Selenium [25], которая запускает имитацию использования браузера со вкладкой сайта в течение некоторого времени для выполнения запроса. Ждет время выполнения SQL-запроса в течение 15 секунд с помощью встроенной библиотеки time [31].

Парсинг веб-страницы и получение данных с сайта – поиск необходимых элементов на веб-странице – производится с помощью библиотеки BeautifulSoup [4].

В результате подготовки данных становятся доступны следующие данные о репозиториях: название, количество коммитов, пулл-реквестов, проблем, звезд, форков и участников. Для формирования и записи полученных результатов используется встроенная библиотека csv [30], позволяющая записывать результаты в CSV-файл.

Реализация модуля классификации

Модуль классификации включает в себя определение класса популярности проекта, исходя из параметров репозитория, и обучение моделей на основе данных csv-файла.

На вход приходит список, содержащий названия различных метрик (forks, commits, issues и т. д.), которые будут использоваться для вычисления общей метрики популярности проекта. Для вычисления общей метрики популярности берутся значения каждой метрики из списка metrics для каждого проекта в наборе данных. Вычисляются квантили для равномерного распределения по децилям – на 10 классов популярностей. Вычисление квантилей применяется ко всем значениям общей метрики популярности и сохраняет результатов в наборе данных data. Таким образом, каждый проект получает свой собственный класс популярности на основе общей метрики популярности и квантилей.

С помощью библиотеки pandas [19] выполняется чтение данных из файла форма csv для дальнейшей обработки классов по значениям.

Для обучения моделей использовались методы, функции и классы из

библиотеки `scikit-learn` [23], которая предоставляет широкий набор инструментов для машинного обучения и анализа данных с установкой необходимых параметров.

Реализация модуля регрессии

Модуль регрессии выполняет прогнозирование значения звезд у репозиториев на основе остальных параметров.

На вход аналогично модулю классификации приходит набор данных со списком различных метрик. Основное отличие, что в данном случае целевой переменной является уже имеющееся значение звезд в наборе данных. На основе этих данных модель выделяет 20% тестовых данных и 80% данных для обучения. Реализован модуль с помощью класса `GNregressor`, который наследуется от `IGNModel`, имеющий абстрактные методы для предсказания результата, обучения модели и сохранения модели в файл или загрузки – из файла.

Метод загрузки и сохранения файла выполнен с помощью библиотеки `joblib` [14], которая предоставляет инструменты для эффективного сохранения и загрузки объектов моделей.

Реализация модуля оценки работы моделей

В рамках модуля оценки работы моделей реализован подсчет метрик машинного обучения и отображения результатов.

На вход приходит обученная ранее модель, способная предоставить информацию. Для моделей классификации ведется подсчет метрик `F1`, `precision`, `recall` и `accuracy`, для моделей регрессий – `MSE`, `MAE`, R^2 . Кроме того, имеется построение матрицы ошибок относительно данных в рамках моделей классификации, которые отображают верно выбранные результаты классов, а также сохранение матрицы ошибок в файл.

С помощью библиотеки `scikit-learn` [23] получают результаты матрицы ошибок по обучению моделей и определению класса популярности данных.

Визуализация матриц ошибок выполняется с помощью библиотеки `matplotlib` [16], а его модуль `pyplot` помогает создавать диаграмму, позволяя при этом настроить его внешний вид. А получение и сортирование по критериям в рамках графика данных выполняется с помощью библиотеки

seaborn [24].

Данную реализацию перечисленных модулей можно представить в виде диаграммы классов UML, которая представлена на рисунке 8.

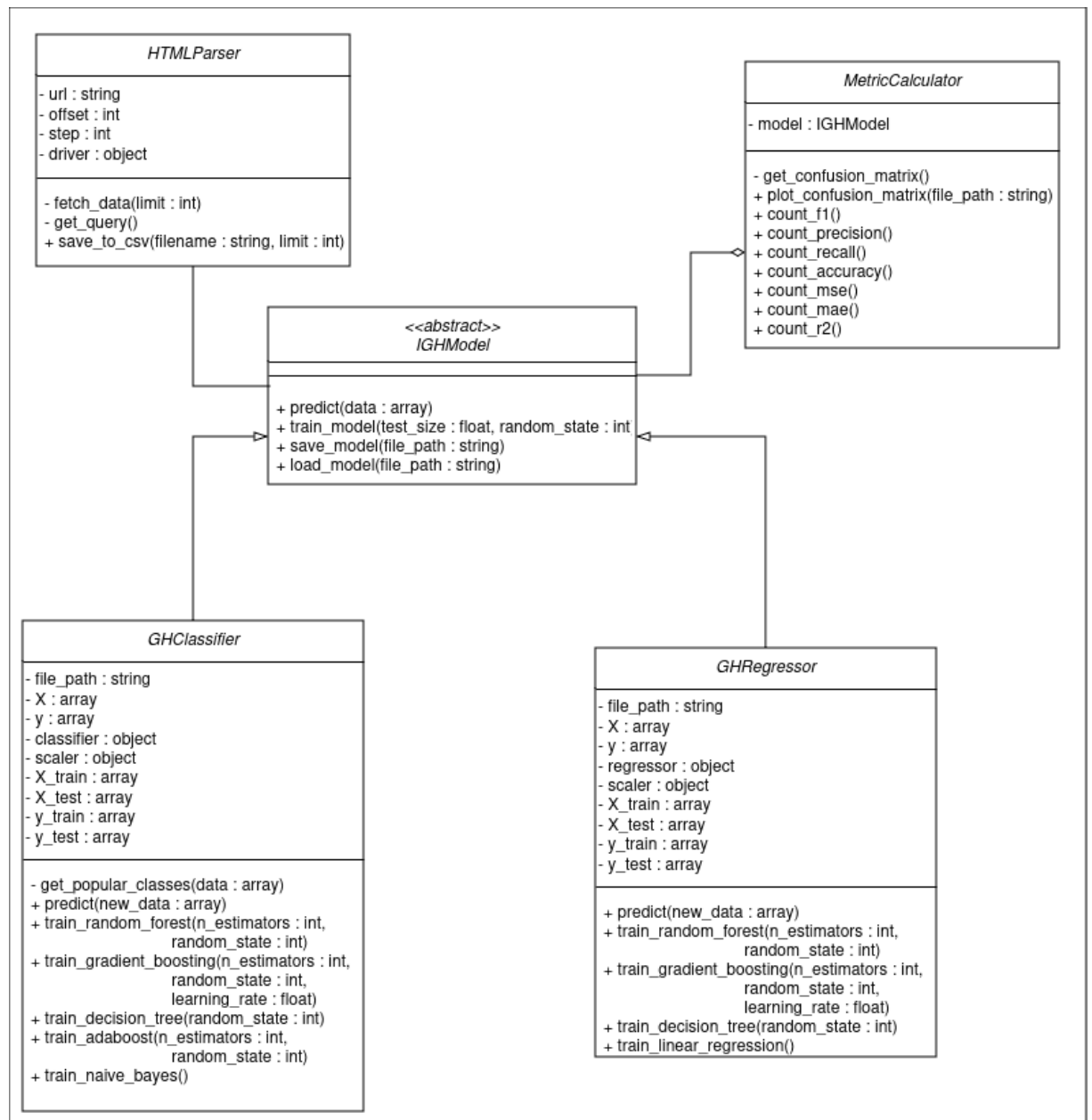


Рис. 8. Диаграмма классов UML

3.3. Реализация пользовательского интерфейса

Реализация пользовательского интерфейса осуществлялась на основе разработанных макетов. Реализованы главной страницы, страницы с вводом параметров и страницей результатов. В качестве основы разработки интерфейса использовалась библиотека Flask [8].

Главная страница (рис. 9) содержит приветственную форму с информацией о типах моделей и о том, что в результате можно получить. В данной форме содержится выпадающий список с выбором типа модели, где можно выбрать только один вариант для работы в дальнейшем, по умолчанию в данном списке всегда выбран вариант "Модель классификации". При нажатии на кнопку "Перейти" происходит переадресация на страницу в зависимости от выбора в списка.

The screenshot shows a light gray rounded rectangle containing the following elements:

- Header:** "Популярность проекта" in bold black font.
- Text 1:** "В этом проекте реализовано обучение модели для определения класса популярности проекта на GitHub."
- Text 2:** "Если требуется определить ранг популярности вашего проекта по всем параметрам репозитория, выберите 'Модель классификации'."
- Text 3:** "Если хотите определить возможное количество звезд вашего репозитория по другим параметрам, выберите 'Модель регрессии'."
- Form:** A label "Выберите тип модели:" followed by a dropdown menu with "Модель классификации" selected and a downward arrow.
- Button:** A dark gray button with the text "Перейти" in white.

Рис. 9. Форма на главной странице

Страница для определения класса популярности и страница для прогнозирования значения звезд репозитория имеют минимальные отличия. На рисунке 10 представлена страница с вводом параметров для определения класса популярности проекта.

Для каждого параметра имеется свое поле для ввода, где название может содержать различные символы, а значение форков, звезд, проблем, коммитов, пулл-реквестов, участников – только целочисленное значение. Все поля в данной форме обязательны для заполнения, при этом в случае отсутствия заполнения нет возможности узнать класс популярности. Вни-

зу формы содержится выбор моделей классификации из тех, что показали самые лучшие результаты в метриках, по умолчанию в данном выпадающем списке стоит значение "Градиентный бустинг". При нажатии на кнопку "Узнать класс" выполняется получение значения с помощью модуля классификации, после чего открывается страница с результатом. Страница для прогнозирования значения звезд отличается от данной отсутствием поля "звезды".

Рис. 10. Форма на странице определения популярности

Страница с результатами для определения класса популярности или звезд также имеют минимальные отличия. На рисунке 11 представлен результат выполнения определения репозитория к классу.

На страницу приходит GET-запрос в котором содержатся данные для вывода результатов. Сама форма содержит строку с текстом, в котором содержится значение класса от 0 до 10 с пояснением о границах интервалов популярности. Поскольку результат может сильно отличаться о разных мо-

делях также идет информация о ней. Кроме того, ниже перечислена введенная ранее пользователем информация для подтверждения того, что текущий результат получен именно от этих данных.

Класс популярности проекта

По вашим данным ранг популярности проекта: *10*, где
0 - мало популярен
10 - очень популярен

Выбранная модель: Случайный лес

Название: 'javascript-tutorial / ru.javascript.info'

29541 forks

368406 stars

28836 issues

2761 commits

3967 pull-requests

210 contributors

Рис. 11. Форма на странице результатов

4. ТЕСТИРОВАНИЕ

В данном разделе представлены результаты функционального тестирования и экспериментов по исследованию эффективности разработанной программы.

4.1. Функциональное тестирование

В ходе данного тестирования проверялось соответствие приложения предъявленным функциональным требованиям. Тестирование приложения проводилось вручную. В таблице 1 приведен протокол ручного тестирования основных аспектов работы системы.

Табл. 1. Функциональное тестирование приложения

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
1	Проверка определения класса популярности.	На странице с вводом отправить данные и получить значение популярности.	Ранг популярности определен.	Да
2	Проверка прогнозирования звезд.	В форме на странице в вводом нажать на кнопку "Определить" и получить значение звезд.	Значение возможного количества звезд отображено.	Да
3	Проверка ввода только численных значений.	На странице с вводом данных ввести значения, отличные от чисел.	В поля не вводятся нечисленные значения.	Да
4	Проверка отправки формы данных.	При незаполненных данных на странице ввода отправить форму.	Получено всплывающее уведомление, что поле не заполнено.	Да
5	Проверка выбора модели.	На главной странице выбрать "Модель регрессии" и отправить форму с выбранной моделью "Случайный лес".	На странице результатов отображены результаты после прогнозирования с помощью модели "Случайный лес".	Да

4.2. Вычислительные эксперименты

Оценка моделей классификации с помощью метрик

Метрики представляют собой инструменты оценки эффективности

алгоритмов классификации, позволяющие оценить качество предсказаний модели на основе различных аспектов ее работы [15]. В данном разделе рассматриваются основные метрики, широко применяемые в задачах классификации.

Матрица ошибок (Confusion Matrix) – это таблица, используемая для оценки производительности модели классификации, позволяющая визуализировать результаты предсказаний по каждому классу. Она представляет собой матрицу, когда реальные значения классов из тестового набора данных сравниваются с предсказанными моделью.

	positive	negative
positive	True Positive (TP)	False Positive (FP)
negative	False Negative (FN)	True Negative (TN)

В матрице ошибок присутствуют четыре основных термина:

- True Positive (TP): Это количество объектов, которые модель верно предсказала как положительные (верно идентифицированные положительные классы).
- True Negative (TN): Количество объектов, которые модель верно предсказала как отрицательные (верно идентифицированные отрицательные классы).
- False Positive (FP): Также называется ошибкой первого рода. Это количество объектов, которые модель неверно предсказала как положительные, хотя они на самом деле относятся к отрицательному классу (ложноположительные результаты).
- False Negative (FN): Также называется ошибкой второго рода. Это количество объектов, которые модель неверно предсказала как отрицательные, хотя они на самом деле относятся к положительному классу (ложноотрицательные результаты).

Точность (Accuracy) представляет собой долю правильных предсказаний модели относительно общего числа предсказаний. Она вычисляется по формуле:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

где TP - количество истинно положительных предсказаний, TN - количество истинно отрицательных предсказаний, FP - количество ложно положительных предсказаний и FN - количество ложно отрицательных предсказаний.

Точность класса (Precision) измеряет долю истинно положительных предсказаний среди всех предсказанных положительных случаев, в то время как *полнота класса* (Recall) измеряет долю истинно положительных предсказаний среди всех истинно положительных случаев. Они вычисляются по формулам:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

F-мера (F1-score) представляет собой гармоническое среднее между точностью и полнотой класса и является компромиссом между ними. Она вычисляется по формуле:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Для каждой из моделей была составлена матрица ошибок, отражающая, какое количество элементов модель смогла верно определить относительно того класса популярности проекта, к которому она была определена ранее. На рисунке 12 представлена матрица ошибок Градиентного бустинга, который показал хороший результат в сравнении с другими моделями. Для остальных рассмотренных в работе моделей были составлены аналогичные матрицы ошибок. Результаты представлены в приложении Б.

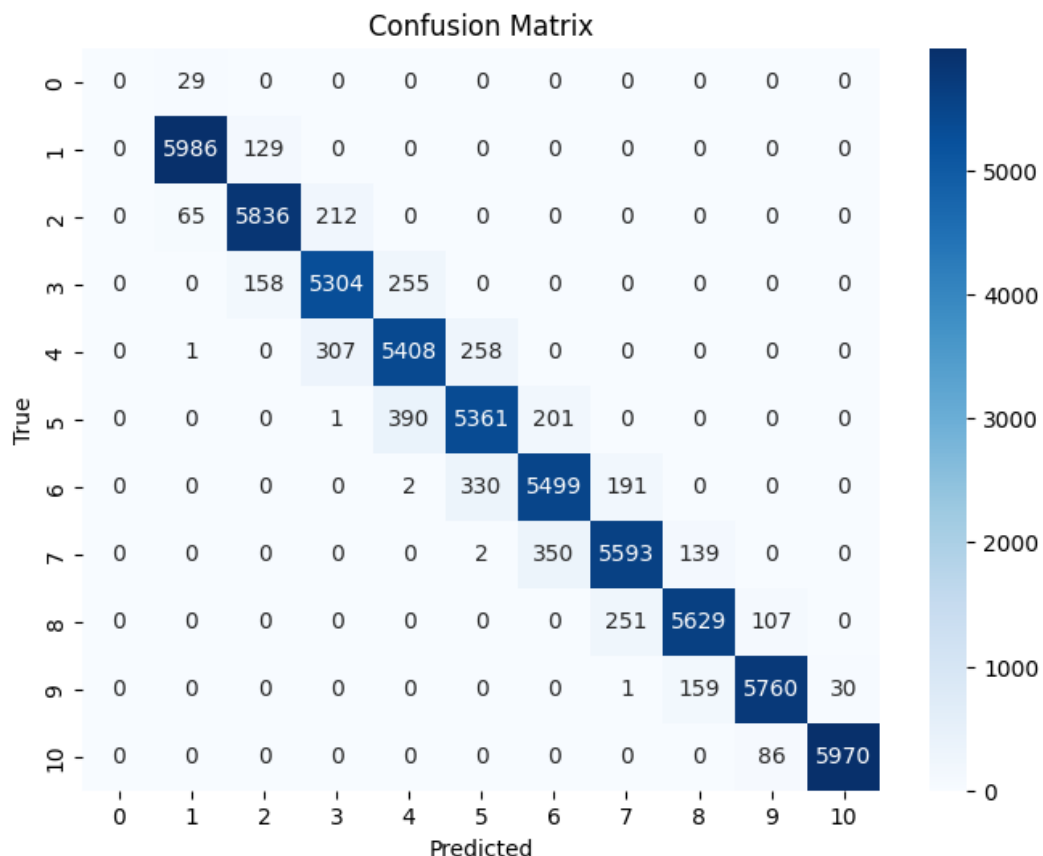


Рис. 12. Матрица ошибок Градиентного бустинга

В рамках каждой модели есть свои параметры, которые влияют на обучение моделей, на точность определения проекта к классу популярности в зависимости от исходных параметров. Так, например, параметр `n_estimators` указывает количество базовых моделей, которые будут объединены. Большее количество шагов может улучшить качество модели, но может увеличить время обучения. Значение `random_state` управляет случайностью в модели. Задавая одно и то же значение, можно получить одинаковые результаты при каждом запуске обучения модели. В модели Градиентного бустинга есть параметр `learning_rate` (float), который контролирует величину, на которую каждый базовый классификатор "учится" на ошибках предыдущих классификаторов.

Параметры помогают настроить модели таким образом, чтобы они давали оптимальные результаты для конкретного набора данных и задачи классификации. При этом изменение большинства параметров с целью улучшения точности ведет к значительному увеличению времени обуче-

ния модели. Время обучения тоже является важным критерием при выборе модели и параметров.

В результате необходимо найти подходящее соотношение времени обучения и результатов метрик, для этого было проведено сравнение скорости обучения модели и результатов значений метрик на разных значениях параметра `n_estimators`. На рисунке 13 представлен график зависимости времени обучения модели и результат значений метрик, где видно, что увеличение параметра начало всё меньше влиять на значения метрик, при этом время выполнения также растет, поэтому оптимально искать среднее значение. Для других моделей с хорошими показателями метрик были составлены аналогичные графики зависимостей. Результаты представлены в приложении В.

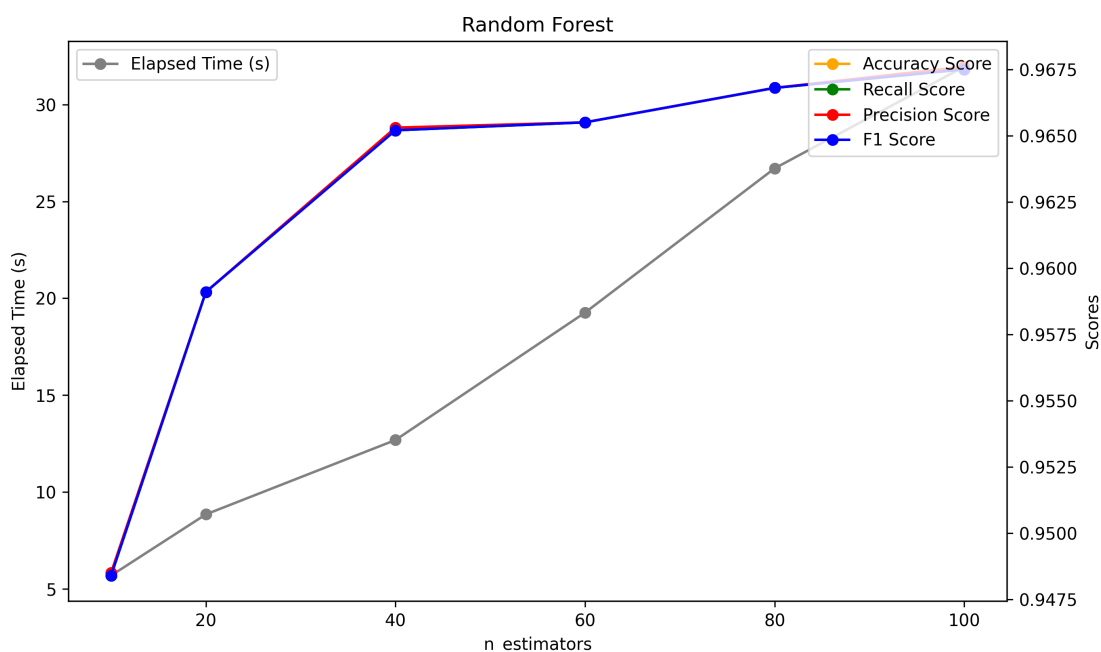


Рис. 13. График зависимости времени и метрик Случайного леса

После определения оптимальных значений в параметрах и оценке их времени выполнения в каждой модели было выполнено обучение и вычисление значений метрик. Результаты рассчитанных метрик для реализованных моделей классификации представлены в таблице 2. Чем ближе значение к 1, тем более точно модель способна определить класс.

Табл. 2. Значения метрик моделей классификации

	precision	recall	F-score
Decision Tree	0.9405	0.9406	0.9405
Random Forest	0.9675	0.9675	0.9675
Gradient Boosting	0.7463	0.7511	0.7689
AdaBoosting	0.3162	0.6698	0.2596
Naive Bayes	0.6258	0.6387	0.6273

Оценка моделей регрессии с помощью метрик

Метрики в задачах регрессии играют важную роль в оценке точности и эффективности модели. Они позволяют оценить, насколько хорошо модель соответствует реальным данным и насколько точны ее прогнозы. В данном разделе рассматриваются основные метрики, применяемые для оценки моделей регрессии.

Средняя абсолютная ошибка (Mean Absolute Error, MAE) представляет собой среднее абсолютное значение разницы между фактическими значениями и прогнозируемыми значениями модели. Она вычисляется по формуле:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

где y_i - фактическое значение, \hat{y}_i - прогнозное значение, а n - количество наблюдений.

Среднеквадратичная ошибка (Mean Squared Error, MSE) измеряет среднеквадратичную разницу между фактическими значениями и прогнозируемыми значениями модели. Она вычисляется по формуле:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

где y_i - фактическое значение, \hat{y}_i - прогнозное значение, а n - количество наблюдений.

Коэффициент детерминации (Coefficient of Determination, R^2) измеряет пропорцию вариации зависимой переменной, объясненной моделью, к общей вариации зависимой переменной. Он принимает значения от 0 до 1 и чем ближе значение R^2 к 1, тем лучше модель соответствует данным. R^2 вычисляется по формуле:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

где \bar{y} - среднее значение зависимой переменной.

Эти метрики предоставляют информацию о точности и качестве модели регрессии, помогая исследователям и практикам принимать информированные решения на основе результатов их работы.

Аналогично моделям классификации были определены оптимальные значения в параметрах. Результаты рассчитанных метрик для реализованных моделей регрессии представлены в таблице 3.

Табл. 3. Метрики задач регрессии

	MSE	MAE	R^2
Decision Tree	97601.26	466.59	0.2150
Random Forest	54962.27	355.25	0.5580
Gradient Boosting	55149.55	367.19	0.5563
Linear Regression	70683.04	530.50	0.4317

Для оценки качества определения классов были сформированы отдельные тестовые наборы, наиболее наглядно отражающие принадлежность к рангам. Результаты представлены в приложении Г.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана система, осуществляющая подготовку данных, обучение моделей и определение класса популярности проектов в рамках GitHub.

При этом были решены следующие задачи.

- 1) Выполнен анализ предметной области и проведен обзор существующих решений.
- 2) Выбран репрезентативный набор признаков и подготовлены данные по существующим репозиториям GitHub.
- 3) Исследованы различные модели машинного обучения и выбраны наиболее эффективные.
- 4) Разработано приложение, которое будет классифицировать проекты по нескольким уровням на основе выбранной модели.

Дальнейшая работа будет направлена на рассмотрение разных критериев определения класса популярности и других параметров в рамках выборки набора данных для обучения.

ЛИТЕРАТУРА

1. Abedini Y., Heydarnoori A. Can GitHub Issues Help in the App Review Classifications?. // CoRR. – 2023. – Vol. abs/2308.14211. – arXiv : 2308.14211.
2. Alshara Z. ML-Augmented Automation for Recovering Links Between Pull-Requests and Issues on GitHub. / Z. Alshara, H.E. Salman, A. Shatnawi, A. Seriai. // IEEE Access. – 2023. – Vol. 11. – P. 5596–5608. – URL: <https://doi.org/10.1109/ACCESS.2023.3236392>.
3. Ayala J., Garcia J. An Empirical Study on Workflows and Security Policies in Popular GitHub Repositories. // 1st IEEE/ACM International Workshop on Software Vulnerability, SVM@ICSE 2023, Melbourne, Australia, May 20, 2023. – IEEE, 2023. – P. 6–9. – URL: <https://doi.org/10.1109/SVM59160.2023.00006>.
4. Beautiful Soup. [Электронный ресурс]. – URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> (дата обращения: 02.05.2024).
5. Bidoki N.H. A Cross-Repository Model for Predicting Popularity in GitHub. / N.H. Bidoki, G. Sukthankar, H. Keathley, I. Garibay. // CoRR. – 2019. – Vol. abs/1902.05216. – arXiv : 1902.05216.
6. Breiman L. Random Forests. – Vol. 45. – Springer. – P. 5–32.
7. Casalnuovo C. GitcProc: a tool for processing and classifying GitHub commits. / C. Casalnuovo, Y. Suchak, B. Ray, C. Rubio-González. // Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, July 10 - 14, 2017 / Ed. by T. Bultan, K. Sen. – ACM, 2017. – P. 396–399. – URL: <https://doi.org/10.1145/3092703.3098230>.
8. Flask Documentation. [Электронный ресурс]. – URL: <https://flask.palletsprojects.com/> (дата обращения: 02.05.2024).
9. Han J. Characterization and Prediction of Popular Projects on GitHub. / J. Han, S. Deng, X. Xia, D. Wang, J. Yin. // 43rd IEEE Annual Computer Software and Applications Conference, COMPSAC 2019, Milwaukee, WI, USA, July 15-19, 2019, Volume 1 / Ed. by V. Getov, J. Gaudiot, N. Yamai, et al. – IEEE, 2019. – P. 21–26. – URL:

<https://doi.org/10.1109/COMPSAC.2019.00013>.

10. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. – Springer series in statistics. – Springer, 2001. – URL:

<https://books.google.ru/books?id=VRzITwgNV2UC>.

11. Hastie T.J. Multi-class AdaBoost. / T.J. Hastie, S. Rosset, J. Zhu, H. Zou. – Vol. 2. – 2009. – P. 349–360. – URL:

<https://api.semanticscholar.org/CorpusID:11803458>.

12. I Don't Git It: Tracking the Source Collaboration Market . [Электронный ресурс]. – URL: <https://thenewstack.io/i-dont-git-it-tracking-the-source-collaboration-market/> (дата обращения: 10.04.2024).

13. Izza Y., Ignatiev A., Marques-Silva J. On Explaining Decision Trees. – 2020. – 2010.11034.

14. Joblib: running Python functions as pipeline jobs. [Электронный ресурс]. – URL: <https://joblib.readthedocs.io/en/stable/> (дата обращения: 02.05.2024).

15. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. – Cambridge, UK : Cambridge University Press, 2008. – URL: <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>.

16. Matplotlib — Visualization with Python. [Электронный ресурс]. – URL: <https://matplotlib.org/> (дата обращения: 02.05.2024).

17. Milovidov A. Everything You Ever Wanted To Know About GitHub (But Were Afraid To Ask). – 2020. – URL: <https://ghe.clickhouse.tech/>.

18. Núñez E., Steyerberg E.W., Núñez J. [Regression modeling strategies].. – Vol. 64. – 2011. – 6. – P. 501–7. – 6209<m:linebreak></m:linebreak>CI: Copyright (c) 2011; JID: 0404277; 2011/01/27 [received]; 2011/01/29 [accepted]; 2011/04/29 [aheadofprint]; ppublish; URL: <http://www.ncbi.nlm.nih.gov/pubmed/21531065>.

19. Pandas - Python Data Analysis Library. [Электронный ресурс]. – URL: <https://pandas.pydata.org/> (дата обращения: 02.05.2024).

20. Puhlfürß T., Montgomery L., Maalej W. An Exploratory Study of Documentation Strategies for Product Features in Popular GitHub Projects. // IEEE International Conference on Software Maintenance and Evolution,

ICSME 2022, Limassol, Cyprus, October 3-7, 2022. – IEEE, 2022. – P. 379–383. – URL: <https://doi.org/10.1109/ICSME55016.2022.00043>.

21. Ramasamy D. Workflow analysis of data science code in public GitHub repositories. / D. Ramasamy, C. Sarasua, A. Bacchelli, A. Bernstein. // Empir. Softw. Eng. – 2023. – Vol. 28. – No. 1. – P. 7. – URL: <https://doi.org/10.1007/s10664-022-10229-z>.

22. Rocco J.D. HybridRec: A recommender system for tagging GitHub repositories. / J.D. Rocco, D.D. Ruscio, C.D. Sipio, P.T. Nguyen, R. Rubei. // Appl. Intell. – 2023. – Vol. 53. – No. 8. – P. 9708–9730. – URL: <https://doi.org/10.1007/s10489-022-03864-y>.

23. Scikit-learn: machine learning in Python. [Электронный ресурс]. – URL: <https://scikit-learn.org/> (дата обращения: 02.05.2024).

24. Seaborn: statistical data visualization. [Электронный ресурс]. – URL: <https://seaborn.pydata.org/> (дата обращения: 02.05.2024).

25. Selenium. [Электронный ресурс]. – URL: <https://www.selenium.dev/> (дата обращения: 02.05.2024).

26. Shalev-Shwartz S., Ben-David S. Understanding Machine Learning - From Theory to Algorithms.. – Cambridge University Press, 2014. – P. I–XVI, 1–397.

27. Soll M., Vosgerau M. ClassifyHub: An Algorithm to Classify GitHub Repositories. // KI 2017: Advances in Artificial Intelligence - 40th Annual German Conference on AI, Dortmund, Germany, September 25-29, 2017, Proceedings / Ed. by G. Kern-Isberner, J. Fürnkranz, M. Thimm. – Vol. 10505 of Lecture Notes in Computer Science. – Springer, 2017. – P. 373–379. – URL: https://doi.org/10.1007/978-3-319-67190-1_34.

28. Visual Studio Code. [Электронный ресурс]. – URL: <https://code.visualstudio.com/> (дата обращения: 12.04.2024).

29. Zhang H. The Optimality of Naive Bayes. // Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004) / Ed. by V. Barr, Z. Markov. – AAAI Press, 2004.

30. Библиотека csv. [Электронный ресурс]. – URL: <https://docs.python.org/3/library/csv.html> (дата обращения: 02.05.2024).

31. Библиотека time. [Электронный ресурс]. – URL:

<https://docs.python.org/3/library/time.html> (дата обращения: 02.05.2024).

32. Исходный код репозитория работы. [Электронный ресурс]. – URL: <https://github.com/Goldria/popularity-github/> (дата обращения: 02.05.2024).

33. Рейтинг систем контроля версий 2016. [Электронный ресурс]. – URL: <https://tagline.ru/version-control-systems-rating/> (дата обращения: 10.04.2024).