

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.ВКР

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. Провести анализ и спецификацию предметной области.
- 4.2. Сделать обзор существующих методов и алгоритмов классификации данных для определения популярности проектов.
- 4.3. Изучить доступные методы сбора данных о проектах GitHub.
- 4.4. Исследовать влияние различных факторов на популярность проектов GitHub и определить наиболее значимые из них.
- 4.5. Провести анализ и выбрать наиболее подходящие алгоритмы машинного обучения для задачи классификации популярности проектов.
- 4.6. Создать набор данных для обучения модели, включающий разнообразные характеристики проектов.
- 4.7. Обучить модель на собранном наборе данных и провести оценку её качества.
- 4.8. Разработать пользовательский интерфейс для взаимодействия с системой классификации проектов GitHub по популярности.

5. Дата выдачи задания: 09.02.2024.

Научный руководитель

к.ф.-м.н., доцент

М.Л. Цымблер

Задание принял к исполнению

Д.И. Гольденберг

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	6
1.1. Обзор работ по тематике исследования	6
2. ПРОЕКТИРОВАНИЕ	9
2.1. Требования к системе	9
2.2. Обработка данных	9
2.3. Прогнозные модели	11
2.4. Оценка моделей с помощью метрик	13
3. РЕАЛИЗАЦИЯ	17
3.1. Обработка данных для модели	17
3.2. Обучение модели	18
3.3. Графический интерфейс (веб-страница)	18
4. ТЕСТИРОВАНИЕ	21
ЗАКЛЮЧЕНИЕ	22
ЛИТЕРАТУРА	23

ВВЕДЕНИЕ

Актуальность темы

Актуальность работы заключается в том, что с развитием технологий тяжело понять, насколько может быть популярен проект, такая система может быть полезна для дальнейшей работы с репозиторием.

Цель и задачи исследования

Целью данной работы является разработка системы прогноза популярности проектов GitHub. В качестве исходных данных используются датасет ClickHouse о репозиториях на GitHub. Для достижения поставленной цели необходимо было решить следующие *задачи*:

- 1) провести анализ предметной области и обзор научной литературы по тематике исследования;
- 2) разработать алгоритм обработки данных;
- 3)

Структура и объем работы

Курсовая работа состоит из введения, пяти разделов, заключения и списка литературы. Объем работы составляет n страниц, объем библиографического списка составляет – k наименований.

Содержание работы

Первый раздел, «Обзор работ по тематике исследования», содержит обзор на работы по тематике исследования.

Второй раздел, «Анализ предметной области», описывает постановку задачи и описание данных, которые будут использоваться для анализа.

Третий раздел, «Проектирование», определяет требования к системе, описаны модели данных и структура приложения.

Четвертый раздел, «Реализация», описывает реализацию компонентов системы.

Пятый раздел, «Тестирование», описывает функциональное тестирование работы и эксперименты с разработанными моделями данных.

В заключении приведены основные итоги проделанной работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В разделе 1.1 представлен обзор других исследований, связанных с темой данного исследования.

1.1. Обзор работ по тематике исследования

Исследования, связанные с анализом данных, возникающих при использовании GitHub и разработке продуктов на этой платформе, представляют собой популярную тему 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 другой статье [11] 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 исследовании [10] 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 поставки (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. А в другом исследовании [9] 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.

Кроме того, существует статья [6], рассматривающая инструмент GiteProc, который предназначен для анализа проектов на GitHub. Этот инструмент позволяет извлекать информацию о разработке, включая исходный код и историю исправления ошибок. GiteProc 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 следующую работу [12], 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» [4], в которой рассматривается создание модели для прогнозирования популярности репозитория на GitHub, используя данные из разных репозиториях. Модель, основанная на рекуррентной нейронной сети LSTM, позволяет более точно предсказывать популярность, чем стандартные методы прогнозирования временных рядов на основе данных из одного репозитория.

Кроме неё, есть также исследование [7], 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. Модель, основанная на случайном лесе, достигает высокой точности значительно превосходит существующие методы. Основными признаками для определения популярности оказались количество веток, количество открытых задач и количество участников проекта.

2. ПРОЕКТИРОВАНИЕ

Тут какой-то текст про то, что в каких разделах описано.

2.1. Требования к системе

Перечисление требований в зависимости от дальнейших выбранных библиотек и интеграций.

2.2. Обработка данных

Для создания классификации проектов на GitHub необходимо понять, по каким критериям мы сможем выстроить и определить «рейтинг» популярных репозиторий. Для работы с информацией о проектах на GitHub будет использоваться существующий датасет, загруженный в ClickHouse, кроме того, он содержит достаточно много данных, которые могут быть полезны и иметь достаточно большой вес в возможности классифицировать репозиторий.

Выделим основные параметры:

- Звезды (Stars). Количество звезд репозитория указывает на его популярность. Звезды представляют интерес и поддержку сообщества разработчиков и пользователей. Большое количество звезд может привлечь внимание новых разработчиков и повысить репутацию репозитория.
- Форки (Forks). Количество форков показывает, сколько раз репозиторий был скопирован другими разработчиками для дальнейшей работы. Большое количество форков может означать активное участие сообщества, что не редко приводит к активному развитию.
- Проблемы (Issues). Количество задач отражает активность сообщества в обнаружении и решении проблем и задач. Активные задачи могут привлечь новых участников и улучшить качество проекта.
- Коммиты (Commits). Количество коммитов указывает на активность разработчиков в репозитории. Активность и последовательность коммитов важны для развития и поддержания проекта.
- Пулл-реквесты (Pull Requests). Количество запросов на слияние (pull requests) отражает вклад и сотрудничество участников проекта.

Пулл-реквесты представляют собой важный инструмент совместной разработки и улучшения кода.

- Количество пользователей, делающих коммиты. Когда множество разработчиков активно участвует в проекте, это может служить своеобразным подтверждением его ценности и качества. Это может создать доверие у новых пользователей и их убеждение в том, что проект стоит внимания.

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 нас интересуют 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.

После установления ключевых параметров, необходимо определить цель использования этих данных и методы их получения. В рамках данного исследования, нацеленного на разработку системы классификации проектов на платформе GitHub в зависимости от их популярности, имеется неотъемлемая потребность в доступе к данным из базы данных. Эти данные из датасета GitHub будут использованы для обучения модели классификации, её оценки и настройки. Оценка модели позволит установить, насколько успешно она способна разделять проекты на популярные и непопулярные.

Исходными данными для этой работы служит статья о наборе данных GitHub [8], 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 со-

бой открытую систему управления базами данных, специально разработанную для эффективного анализа и хранения больших объемов информации. Одной из важных особенностей этой технологии является её акцент на аналитических задачах, что обеспечивает возможность проведения сложных анализов данных.

Текущий набор данных GH Archive представлен как в формате ClickHouse Native с объемом более 70 ГБ, так и в альтернативном формате, разделенном табуляцией, с объемом 85 ГБ. Учитывая, что данные из этого набора нужны лишь для однократного обучения модели без необходимости долгосрочного доступа ко всей базе данных, было принято решение извлекать такие объемы данных из других хранилищ данных, а не с физических устройств.

Помимо того, для демонстрации работы с общедоступными данными ClickHouse существует веб-страница, способная обрабатывать SELECT-запросы и предоставлять обширный объем информации. Все данные, полученные таким образом, являются полными, вследствие чего было определено осуществлять извлечение результатов запросов из HTML-страницы.

2.3. Прогнозные модели

Существует разнообразие моделей машинного обучения, каждая из которых ориентирована на решение конкретных типов задач:

- Регрессионные модели: Они используются для прогнозирования численных значений или характеристик объектов.
- Модели классификации: Они предсказывают принадлежность объекта к определенной категории на основе заданных параметров.

Регрессионные модели ориентированы на прогнозирование количественных значений, в то время как модели классификации сосредоточены на определении принадлежности объекта к определенным категориям или классам на основе входных параметров.

При выполнении данной работы будет произведен анализ с использованием моделей классификации для решения задачи определения ранга популярности проекта на основе имеющихся данных. Рассмотрим модели, которые могут быть использованы для работы.

Деревья решений

Деревья решений – это графические структуры, используемые в машинном обучении для принятия решений на основе условий или правил, представленных в виде дерева. Они представляют собой модель, которая аппроксимирует входные данные с помощью последовательности решений, ведущих к конечным выводам или предсказаниям.

При обучении деревьев решений модель строит дерево, разбивая данные на более чистые подгруппы на основе признаков. Процесс деления данных происходит таким образом, чтобы максимизировать чистоту (уменьшить неопределенность) в полученных подгруппах. Цель состоит в том, чтобы создать дерево, способное эффективно делать прогнозы для новых данных.

Деревья решений могут использоваться как для задач классификации, когда необходимо отнести объект к одной из категорий, так и для задач регрессии, когда нужно предсказать численное значение. Они предоставляют простой и понятный способ моделирования данных, однако большие деревья могут быть склонны к переобучению, поэтому часто используются ансамбли деревьев, такие как Случайный лес или Градиентный бустинг, для улучшения результатов и борьбы с переобучением.

Ансамбль деревьев представляет собой подход в машинном обучении, в котором несколько моделей деревьев решений объединяются для получения более точных и устойчивых прогнозов по сравнению с отдельными деревьями. Два основных типа ансамблей деревьев – это Случайный лес и Градиентный бустинг.

Случайный лес

Случайный лес (Random Forest) – это вид ансамбля деревьев, где каждое дерево строится независимо друг от друга. В процессе обучения каждое дерево получает подмножество данных (выбирается случайным образом из общего набора данных), и на основе этого подмножества строится свое дерево решений. При прогнозировании результаты всех деревьев усредняются или объединяются для получения окончательного предсказания. Random Forest способен справляться с переобучением, более устойчив к шумам в данных и хорошо работает на больших наборах данных.

Градиентный бустинг

Градиентный бустинг (Gradient Boosting) – это метод построения ансамбля деревьев последовательно, каждое новое дерево исправляет ошибки предыдущего. На каждом шаге новое дерево строится таким образом, чтобы минимизировать остатки (разницу между предсказаниями модели и реальными значениями). При обучении каждое следующее дерево фокусируется на тех объектах, на которых предыдущие модели ошиблись больше всего. Градиентный бустинг способен давать очень точные прогнозы и часто используется в соревнованиях по машинному обучению. Однако он более склонен к переобучению, чем случайный лес, и требует тщательной настройки параметров.

2.4. Оценка моделей с помощью метрик

Существует несколько метрик, которые широко используются для оценки качества моделей машинного обучения. Перечислим некоторые из них.

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).

Аккуратность (Accuracy) – это общая метрика, показывающая долю правильно классифицированных объектов относительно общего числа объектов в наборе данных.

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

Данная метрика имеет свои ограничения и не всегда является достаточно информативной. Так, если классы в данных имеют различную долю представленности, модель может показать высокую точность, просто предсказывая наиболее часто встречающийся класс, не учитывая другие классы. В таких случаях высокий процент Accuracy может быть обманчивым и не отражать реальную способность модели различать разные классы.

Точность (Precision) – это метрика, измеряющая точность предсказания положительных классов, то есть долю правильно предсказанных положительных объектов среди всех объектов, предсказанных как положительные.

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

Полнота (Recall) – это метрика, измеряющая способность модели обнаруживать все положительные объекты и показывающая долю правильно предсказанных положительных объектов от общего числа реальных положительных объектов.

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

В реальности одновременное максимальное достижение значений Recall и Precision невозможно, поэтому требуется поиск оптимального компромисса. Именно в этом контексте возникает потребность в метрике, объединяющей информацию о точности и полноте предсказаний модели.

F-мера (F-score) – эта метрика, представляющая собой гармоническое среднее между точностью и полнотой. Она особенно полезна, когда классы несбалансированы. F-мера выполняет данную задачу, обеспечивая более глубокий анализ и упрощая процесс выбора наилучшей реализации алгоритма для запуска в производство.

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Для моделей случайного леса и градиентного бустинга была составлена матрица ошибок, представленная на рис. 1, отражающая, какое количество элементов модель смогла верно определить относительно принадлежащего класса популярности проекта.

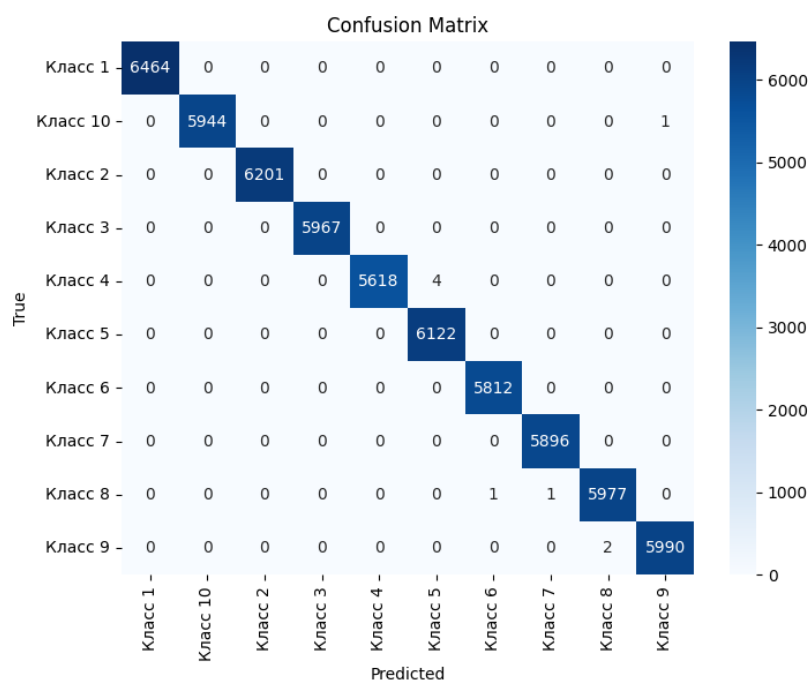


Рис. 1. Матрица ошибок

Результаты рассчитанных метрик для реализованных моделей представлены в таблице 1.

Табл. 1. Метрики алгоритмов

	precision	recall	F-score
Random Forest	0.999850046	0.999850000	0.999849998
Gradient Boosting	0.99998995	0.99959719	0.99979353

3. РЕАЛИЗАЦИЯ

Тут какой-то текст про то, что в каких разделах реализовано.

3.1. Обработка данных для модели

В силу того, что данные из датасета ClickHouse было решено получать из веб-приложения, необходимо разработать способ, которым можно корректно их преобразовать.

Для этой цели был разработан класс, содержащий несколько методов. Самый важный метод позволяет извлекать данные из разметки страны формата html. Этот метод принимает параметр, который определяет желаемый объем данных для извлечения.

Данная функция имитирует взаимодействие с веб-страницей, используя библиотеку Selenium WebDriver для управления браузером. Он создает виртуальное окружение браузера, отправляет запрос. После ожидания получения результата в течение 10 секунд данные извлекаются из таблицы на веб-странице с использованием библиотеки BeautifulSoup, предназначенной для анализа HTML-файлов.

Исследователи из Федерального университета Минас-Жерайс, Бразилия представили результаты своей работы, в которой они анализировали прогноз популярности проектов на GitHub [5]. Вследствие того, что GitHub является динамичной платформой, на которой постоянно появляются новые проекты, технологии и тренды, анализ данных за последние 5-8 лет позволит учесть актуальные тенденции в мире разработки программного обеспечения. Использование данных за это время позволит увидеть динамику изменений в популярности проектов и научиться предсказывать будущие тренды.

Кроме того, важно учитывать только уникальные вклады от каждого пользователя потому, что это позволит избежать искажений в данных, вызванных многократным участием одного пользователя. Также уникальные авторы коммитов обеспечат разнообразие данных, позволяя модели учитывать вклад различных участников проекта при определении его популярности. В результате было решено использовать данные за последние 8 лет и учитывать данные об уникальных авторах коммитов.

Полученные данные могут быть представлены в различных форма-

тах, таких как JSON, CSV и другие. Для работы в дальнейшем будет использоваться формат CSV для обработки. Так, было получено около трехсот тысяч данных.

3.2. Обучение модели

Для решения задачи классификации на основе данных платформы GitHub был разработан класс `GitHubClassifier`. Этот класс содержит несколько методов, предназначенных для обработки данных, обучения моделей и оценки их производительности.

Класс инициализируется путем загрузки данных из CSV-файла и предварительной обработки. Данные подвергаются масштабированию с использованием метода `MinMaxScaler` из библиотеки `sklearn.preprocessing`, что позволяет привести значения признаков к одному масштабу.

Для обучения моделей классификации, таких как `RandomForestClassifier` и `GradientBoostingClassifier` из библиотеки `sklearn.ensemble`, используются подготовленные данные. Обученные модели позволяют прогнозировать классы новых данных и оценивать их точность с помощью методов `count_f1`, `count_precision` и `count_recall`, возвращающих значения F1-меры, точности и полноты соответственно.

Также предоставлены методы для вывода матрицы ошибок (`get_confusion_matrix`) в форме графика (`plot_confusion_matrix`), сохранения и загрузки обученных моделей (`save_model` и `load_model`). Эти методы обеспечивают анализ результатов классификации и позволяют использовать обученные модели для прогнозирования классов новых данных.

3.3. Графический интерфейс (веб-страница)

В качестве первичного графического интерфейса и возможности отображения работы с моделью было выбрано веб-приложение, которое написано на популярном фреймворке Flask.

На главной странице отображается форма (рис. 2) для добавления параметров репозитория, что необходимо для определения модели ранга популярности.

Популярность проекта

Название репозитория:

Далее введите количество следующих параметров:

форков (forks):

звезд (stars):

проблем (issues):

коммитов (commits):

пулл-реквестов (pull-requests):

участников (contributors):

Выберите модель: Градиентный бустинг ▾

[Узнать класс](#)

Рис. 2. Форма для ввода данных

Кроме того, можно выбрать модель, на базе которой будет определен класс популярности проекта. После получения данные проходят обработку, и открывается страницу, на которой отображен результат определения полученного ранга с перечислением того, какие данные были отправлены (рис. 3)

Класс популярности проекта

По вашим данным ранг популярности проекта: Класс 2, где

1 - мало популярен

10 - очень популярен

Выбранная модель: Градиентный бустинг

Название: 'goldria/popularity-github'

10 forks

3 stars

0 issues

20 commits

3 pull-requests

1 contributors

Рис. 3. Форма для ввода данных

4. ТЕСТИРОВАНИЕ

Тестирование разработанных методов и пр, а также “эксперименты” для проверки прогнозирования.

ЗАКЛЮЧЕНИЕ

Перечисление выполненных задач и дальнейших работ в качестве улучшений.

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. 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.
5. Borges H., Hora A.C., Valente M.T. Predicting the Popularity of GitHub Repositories. // Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2016, Ciudad Real, Spain, September 9, 2016. – ACM, 2016. – P. 9:1–9:10. – URL: <https://doi.org/10.1145/2972958.2972966>.
6. 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>.
7. 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>.

8. Milovidov A. Everything You Ever Wanted To Know About GitHub (But Were Afraid To Ask). – 2020. – URL: <https://ghe.clickhouse.tech/>.
9. 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>.
10. 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>.
11. 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>.
12. 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.