МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Школа компьютерных наук

ОТЧЕТ ПО ДИСЦИПЛИНЕ «ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ»

О РЕЗУЛЬТАТАХ ВЫПОЛНЕНИЯ ПРОЕКТА «РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ПРОВЕРКИ СТУДЕНЧЕСКИХ IT-РЕШЕНИЙ НА АНТИПЛАГИАТ»

ФГАОУ ВО «Тюменский государственный университет»

Кафедра программного обеспечения

| Выполнил  обучающийся 4 курса,  2021 года поступления | (подпись) | Низамов Артем Рустамович  (ФИО) |
| --- | --- | --- |
| Выполнил  обучающийся 4 курса,  2021 года поступления | (подпись) | Стрюков Владислав Николаевич  (ФИО) |
| Научный руководитель  проекта  Доцент (к.н.) | (подпись) | Плотоненко Юрий Анатольевич  (ФИО) |

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_

Результаты экзамена \_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись руководителя практики от института / школы)

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_heading=h.u85uhomb8qvo)

[ГЛАВА 1. ПОДВОДКА 7](#_heading=h.lwhyh4qakb1g)

[1.1. Актуальность проекта 7](#_heading=h.fovp3rxkfsi4)

[1.2. Постановка проблем заказчика 9](#_heading=h.mpjw1wkrfwya)

[1.3. Требования заказчика 9](#_heading=h.8pbi8y8xowv1)

[1.4. Проблемы пользователей 9](#_heading=h.y1cnc5u8a9kq)

[1.5. Требования пользователей 10](#_heading=h.v9zuyx1tq24k)

[1.6. Идея проекта 10](#_heading=h.w8jpkzl6ukya)

[1.7.Описание специфики предметной области 11](#_heading=h.bofrnn159239)

[1.7.2. Особенности работы с данными 11](#_heading=h.p7dc89nlw0cw)

[1.7.3. Интеграция в образовательный процесс 12](#_heading=h.mrxgnxeqn2pf)

[1.8. Учитываемые ограничения: 12](#_heading=h.4fwkqz89b56h)

[ГЛАВА 2. ПОСТРОЕНИЕ КЛАССИФИКАТОРА НЕЙРОСЕТЕВОГО ПЛАГИАТА 13](#_heading=h.2852d3jrs2vo)

[2.1. Постановка задачи определения нейросетевого плагиата 13](#_heading=h.z4cd6trjw6z9)

[2.2 Используемые датасеты 15](#_heading=h.g93zca4bwyoe)

[2.2.1. Требования к датасетам 15](#_heading=h.3m5e6oczoi1m)

[2.2.2. Обоснование выбора датасетов 16](#_heading=h.mhoi7sxm9wao)

[2.2.3. Дополнение датасета решениями, сгенерированными ИИ 17](#_heading=h.ty2ut4ss6iv7)

[2.3. Выбор классификатора 18](#_heading=h.mwq07nt83e6j)

[2.4. Подбор гиперпараметров 18](#_heading=h.qmqsk72csrv8)

[2.5. Результаты 19](#_heading=h.od3xg1wygynq)

[ГЛАВА 3. ПОСТРОЕНИЕ КЛАССИФИКАТОРА МЕЖСТУДЕНЧЕСКОГО ПЛАГИАТА 24](#_heading=h.mqyor65dkv0r)

[3.1. Постановка задачи межстуденческого плагиата 24](#_heading=h.lswgoixmstjm)

[3.2. Выбор эмбеддинг-моделей 24](#_heading=h.b30iq6ntyy22)

[3.3. Результаты 25](#_heading=h.vsba8tpbdejd)

[ГЛАВА 4. РЕАЛИЗАЦИЯ 26](#_heading=h.6gvh9gv39lmn)

[4.1. Архитектура приложения 26](#_heading=h.kg95pov89gb8)

[4.2. Реализация frontend-сервиса 27](#_heading=h.oeakzoif6ig1)

[4.3. Реализация backend-сервиса 31](#_heading=h.20a2pra9g6x9)

[4.4. Организация взаимодействия между компонентами 31](#_heading=h.tltnfx6lana1)

[ЗАКЛЮЧЕНИЕ 34](#_heading=h.11bsk9v3zyzw)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 37](#_heading=h.60dm9888hdyt)

## 

# ВВЕДЕНИЕ

В эпоху цифровизации, которая началась в конце двадцатого века, совершается немало значимых открытий и изобретений: персональные компьютеры, всемирная сеть интернет, мобильные телефоны, виртуальная реальность, блокчейн, а также искусственный интеллект или же нейронные сети, которые подразумевают под собой функцию. Эта функция может обладать миллионами параметров и решать различные задачи. Нас в этой функции больше всего интересует генеративная часть, то есть задача создания чего-либо по конкретному запросу. На сегодняшний день искусственный интеллект способен отвечать на очень большое количество вопросов.

На основании тех баз знаний, на которых обучался искусственный интеллект, складываются возможности использования нейронной сети для решения задач или вовсе создания чего-то нового.

Количество задач, которые решает на сегодняшний день генеративный искусственный интеллект поражают сознание:

1. Текст: написание текстовых работ (таких как - статьи, посты, сценарии, переводы, стихи, резюме, диалоги и т.д.), резюмирование текстов;
2. Изображения: реалистичные изображения, редактирование изображений, 3D-модели;
3. Видео: реалистичные видео, редактирование видео;
4. Звуки: звуковые эффекты для видеоигр, фильмов;
5. Музыка: треки разных жанров;
6. Речь: реалистичная, синтезированная речь;
7. Код: код на различных языках программирования;
8. Презентации: презентация на заданную тему в различных форматах.

Опять же, доступность и лёгкость использования искусственного интеллекта растёт и всё больше людей пользуются возможностями нейронных сетей, немалую часть которых составляют студенты. Они изучают много новой информации и выполняют большое количество заданий семестр за семестром. Отсюда и вытекает необходимость использования «умного помощника», который может по запросу выдать необходимую информацию или даже конечный продукт в виде программного кода или презентации. Возникает необходимость разобраться, а улучшают ли технологии образовательный опыт студентов?

Для ответа на данный вопрос следует провести сравнение положительных и негативных сторон использования технологий искусственного интеллекта в образовательном опыте студентов.

Положительные аргументы:

* Персонализация обучения – искусственный интеллект (Далее – ИИ) может адаптировать информацию для студента в нужный формат  
   по запросу;
* Ускорение поиска и систематизация информации – ИИ может предоставить информацию по запросу без необходимости поиска  
   её в интернете;
* Удобство – ИИ может помочь обучающимся с ограниченными возможностями здоровья выполнять рутинные задачи;
* Углублённое изучение тем – ИИ может помочь студентам глубже изучать интересующие их темы, приводя примеры и дополняя их пояснениями;
* Помощь в оценивании заданий преподавателям – ИИ может помочь в разгрузке преподавателей различных специальностей путём первичной проверки сданных студентами тестов или заданий;
* Новые задания для дополнительного обучения – ИИ способен генерировать задания для самопроверки после прохождения нового материала и закрепить полученные знания.

Негативные аргументы:

* Академическая честность – студенты могут использовать ИИ для выполнения заданий в процессе обучения, выдавая выполненное задание за своё, что может создать неверное представление у преподавателя о знаниях студента;
* Зависимость от технологии – при постоянном использовании студенты, выполняющие задания исключительно с помощью ИИ, могут не получать нужные знания в связи с чем в дальнейшем они не смогут самостоятельно выполнять учебные задания;
* Некачественные материалы – ИИ может выдавать некачественные ответы, которые могут быть связаны с узкостью запрашиваемой темы;  
   или недостаточным количеством данных, на которых обучался ИИ;
* Неправильность использования – нужно уметь пользоваться ИИ, для того, чтобы он давал по-настоящему нужную и корректную информацию. Не все пользователи ИИ умеют писать детальные запросы.

В заключении сравнения можно сделать вывод, что ИИ очевидно может быть полезен в образовательном процессе, но нужно использовать его во благо обучения, а не для его того, чтобы переложить процесс обучения на нейронную сеть.

В программе «Приоритет 2030» есть проект, который называется «Цифровые кафедры». Этот проект должен обеспечить повышение квалификации, возможность получения новых профессий в сфере информационных технологий, получение современных цифровых компетенций. Всё это несомненно необходимо для подготовки квалифицированных специалистов, которые будут востребованы на рынке труда.  
 В проекте предусмотрено 2 направления подготовки:

1. Направление подготовки для студентов, обучающихся  
    по специальностям и направлениям подготовки в ИТ-сфере;
2. Направление подготовки для студентов, обучающихся  
    по специальностям и направлениям подготовки, не отнесенным к ИТ-сфере.

Несомненно, ИИ может быть полезен для обоих направлений подготовки, так как в образовательной программе «2+2», для обучающихся 1 и 2 курса предусмотрены дисциплины, в которых изучается программирование.

Уже были упомянуты такие негативные аргументы к использованию ИИ в образовательном процессе студентов как: академическая честность и зависимость от технологии, которые отражаются на обучении больше всего на качестве знаний.

С развитием ИИ студенты начали пользоваться им для выполнения заданий и тестов, которые преподаватель выдает в рамках дисциплины по программированию.

# ГЛАВА 1. ПОДВОДКА

По плану наш проект должен представлять из себя веб-приложение, выполняющее проверку на антиплагиат двух видов:

1. Определение ответа, заимствованного одним студентом у другого, который студент выдаёт за оригинальное и самостоятельное выполнение задания;
2. Определение ответа, сгенерированного ИИ, который студент выдаёт  
    за оригинальное и самостоятельное выполнение задания.

К сожалению, данный проект не сможет полностью избавить преподавателя от рутинной проверки многочисленных ответов на одни и те же задания, но он сможет обрабатывать ответы студентов и обозначать сомнительные участки или даже ответы целиком, которые не прошли проверку на один из видов плагиата. Очевидно, что время преподавателя на проверку в какой-то мере сократится, и он сможет больше времени уделять индивидуальному общению со студентами, что может положительно повлиять на качество образования в университете.

## 1.1. Актуальность проекта

В рамках дисциплины «Программирование и основы алгоритмизации» (Далее – ПИОА) большое количество студентов первого и второго курса проходят данную дисциплину. За год около 3000 студентов. Нашим заказчиком выступает наш научный руководитель – Плотоненко Юрий Анатольевич, кандидат педагогических наук, доцент кафедры программного обеспечения.

Юрий Анатольевич столкнулся с тем, что в условиях растущего числа студентов в группах (до 35 человек). Преподавателям становится практически невозможным проводить индивидуальные беседы и оценивать уровень понимания материала каждым студентом, в том числе определить насколько самостоятельно студент выполнил задание.

Заказчиком данного проекта является ФГАОУ ВО «Тюменский государственный университет», но в также потенциальными заказчиками могут выступать образовательные учреждения, такие как: университеты, колледжи, техникумы, а также отдельные факультеты, кафедры или образовательные программы, на которых выполняются студентами практические работы и проекты в области IT. Заказчики могут быть заинтересованы в обеспечении высокого уровня академической честности и предотвращении плагиата среди студентов.

Целевой аудиторией нашего проекта являются преподаватели и сотрудники ТюмГУ, ответственные за проверку студенческих IT-решений.

В эпоху цифровизации студенты не только выдают чужие работы за свои, но также используют технологии генеративного искусственного интеллекта для создания решений тех или иных заданий по программированию. Сдавая задания таким образом, студенты не разбираются в предлагаемых преподавателем темах и не получают должных компетенций для дальнейшего обучения в сфере информационных технологий, что сказывается на качестве образования.

В программе «Приоритет 2030» есть проект, который называется «Цифровые кафедры». Этот проект должен обеспечить повышение квалификации, возможность получения новых профессий в сфере информационных технологий, получение современных цифровых компетенций. Всё это несомненно необходимо для подготовки квалифицированных специалистов, которые будут востребованы на рынке труда.  
 В проекте предусмотрено 2 направления подготовки:

1. Направление подготовки для студентов, обучающихся  
    по специальностям и направлениям подготовки в ИТ-сфере;
2. Направление подготовки для студентов, обучающихся  
    по специальностям и направлениям подготовки, не отнесенным к ИТ-сфере.

Несомненно, наш проект может быть полезен для обоих направлений подготовки, так как в образовательной программе «2+2», так как для обучающихся 1 и 2 курса предусмотрена дисциплина ПИОА.

Также, актуальность проекта была подтверждена выступлением Стрюкова Владислава на защите проектных идей в рамках стипендиального конкурса имени Г. Ф. Куцева, где наш проект был высоко оценен администрацией Тюменского государственного университета.

## 1.2. Постановка проблем заказчика

* Плагиат и академическая нечестность: увеличение количества случаев плагиата приводит к снижению уровня знаний и умений студентов, что влияет на престиж учебного заведения и его образовательных программ.
* Неэффективность ручных проверок: преподаватели и ассистенты, которые вручную проверяют код, тратят много времени, а выявление совпадений затруднено при высоком объеме работ.
* Отсутствие комплексного анализа кода: существующие решения часто работают с текстами, но не имеют эффективных алгоритмов для анализа кода, что мешает точно определять схожесть и факт заимствования.

## 1.3. Требования заказчика

* Точность и надежность проверки: система должна определять заимствования в студенческом коде, при этом минимизируя ложные совпадения.
* Интеграция с существующими системами управления обучением (LMS): многие образовательные учреждения уже используют платформы для управления курсами, такие как Moodle, Blackboard и другие. Возможность интеграции с такими платформами облегчит процесс загрузки и проверки работ.

Пользователями проекта являются преподаватели и ассистенты, которые проверяют студенческие работы, а также сами студенты, которые могут захотеть проверить свои работы перед финальной сдачей самостоятельно.

## 1.4. Проблемы пользователей

* Сложность анализа кода на предмет плагиата: для преподавателей и ассистентов сложно быстро и эффективно анализировать код, особенно если студенты вносят небольшие изменения в существующий код, чтобы скрыть заимствования.
* Ограниченное время на проверку: преподаватели имеют большое количество работ на проверку, что ограничивает возможности детального анализа каждой работы на предмет плагиата.
* Прозрачность процесса для студентов: для студентов важно понимать, что именно считается плагиатом и как можно улучшить свою работу, избегая заимствований.

## 1.5. Требования пользователей

* Удобный и понятный интерфейс: преподаватели и студенты должны иметь доступ к интуитивно понятному интерфейсу, где можно легко загружать работы и получать результаты.
* Пояснительные отчеты с деталями заимствований: преподаватели нуждаются в подробных отчетах, которые указывают не только уровень заимствования, но и конкретные совпадающие фрагменты кода с пояснениями.
* Быстрая проверка: пользователи ожидают, что система будет работать быстро, даже с большим объемом кода, чтобы не задерживать процесс проверки и сдачи работ.
* Настраиваемые параметры проверки: преподаватели хотят иметь возможность регулировать глубину проверки, учитывать или исключать определенные типы заимствований, что поможет избежать ложных срабатываний.
* Возможность предварительной проверки для студентов: студенты хотят проверять свои работы перед сдачей, чтобы убедиться в отсутствии плагиата и избежать дисциплинарных проблем.

## 1.6. Идея проекта

Идея проекта заключается в создании веб-приложения, которое будет выполнять анализ исходного кода, сравнивая его с работами из внутренней базы данных и с внешними источниками. За счет автоматизации и интеграции с системами управления обучением (LMS) будут решены проблемы времени и качества проверки, а также соблюдение академической честности.

## 1.7.Описание специфики предметной области

Разработка веб-приложения для проверки студенческих IT-решений на антиплагиат затрагивает несколько специфических аспектов, связанных с особенностями образовательной и технической сфер.  
  
1.7.1. Образовательная специфика

Учебные задания: В нашем образовательном учреждении студенты в рамках обучения по системе «2+2» на первом курсе все студенты проходят дисциплину «Программирование и основы алгоритмизации» (Далее – ПИОА) выполняют задания по программированию, реализации алгоритмов. Эти работы требуют самостоятельного выполнения студентами для повышения уровня знаний.

Объём и структура кода: В рамках дисциплины ПИОА студенты выполняют задания на языке Python и результатом своей работы представляют файл-отчёт формата DOCX, где они предоставляют код решения задания и скриншоты выполненной отладки кода.

Этика и академическая честность: Использование антиплагиата направлено на поощрение самостоятельной работы, развитие навыков программирования и предотвращение копирования чужих решений.

Обучающие цели: Решение должно быть не только инструментом контроля, но и образовательным ресурсом, позволяющим выявить и анализировать динамику развития студента в рамках дисциплины ПИОА.

### 1.7.2. Особенности работы с данными

Объемы данных: Система должна обрабатывать разный объем данных (от 2 файлов формата DOCX до 25).

Машинное обучение и нейросети: Возможное использование моделей для автоматического выявления сложных случаев плагиата и адаптации системы к новым типам нарушений.

### 1.7.3. Интеграция в образовательный процесс

Подключение к LMS: в будущем может быть создан плагин для интеграции с системами управления обучением (Moodle и др.) для автоматизации проверки.

Форматы представления данных: Система должна поддерживать разнообразные форматы входных данных (архивы, файлы, ссылки на репозитории).

Интерактивность: Возможность обратной связи для студентов (например, подсказки для улучшения кода или комментарии).

Таким образом, проект охватывает пересечение образовательных и технических специфик, что делает разработку сложной, но перспективной задачей.

## 1.8. Учитываемые ограничения:

1. Ограничение на объем обрабатываемых данных.
2. Бюджет проекта: 0 рублей
3. Дедлайн сдачи проекта: 16 декабря 2024

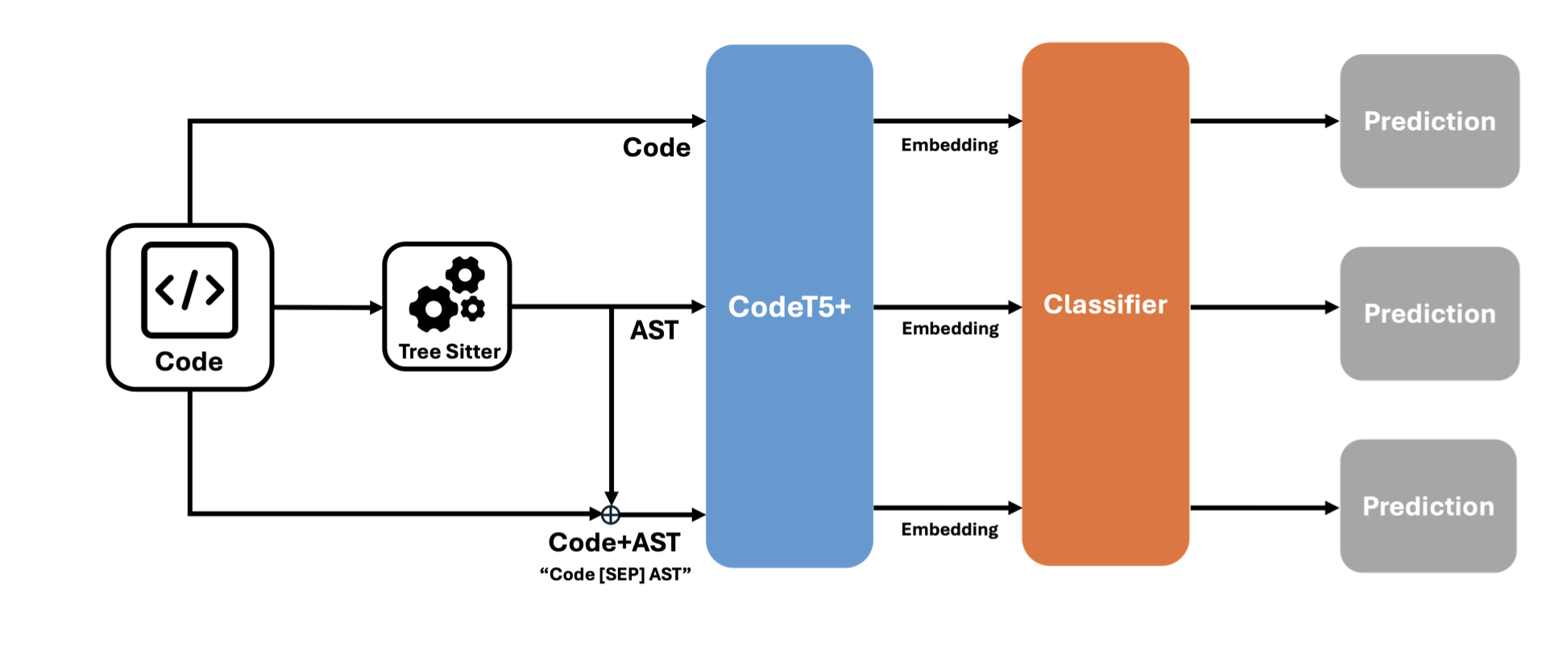
# 

# ГЛАВА 2. ПОСТРОЕНИЕ КЛАССИФИКАТОРА НЕЙРОСЕТЕВОГО ПЛАГИАТА

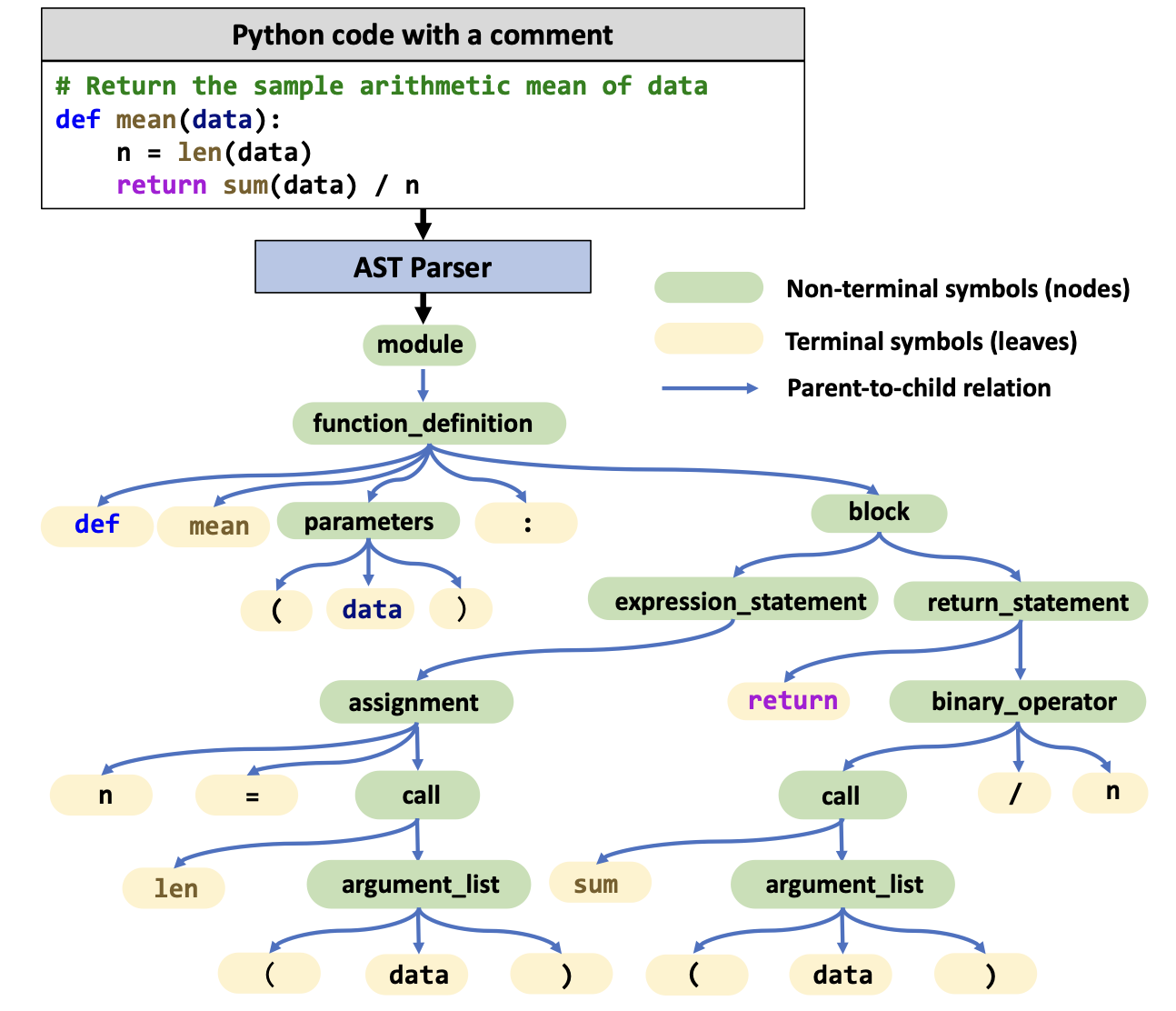
## 2.1. Постановка задачи определения нейросетевого плагиата

Дано: X - множество исходных кодов решений студентов из отчетов, Y - классы, где 0 - нейросетевой плагиат не обнаружен, 1 - обнаружен.

Построить: классификатор исходных кодов .

  
Рис. 1. Методология определения кода, сгенерированного AI

Источник: Suh H., Tafreshipour M., Li J., Bhattiprolu A., Ahmed I.. An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We?

Для того, чтобы решить задачу определения нейросетевого плагиата мы использовали классификатор (логистическую регрессию) [Suh H., Tafreshipour M., Li J., Bhattiprolu A., Ahmed I., URL], которая по эмбеддингу, полученному с помощью модели CodeT5+ из представления исходного кода в виде абстрактного синтаксического дерева (AST) [Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J., URL], определяет кем был написан код (человеком или LLM). Модель CodeT5+ является одной из современных моделей, способных генерировать код и анализировать его на основе контекста, что делает её подходящей для различения кодов, написанных человеком, и сгенерированным ИИ. Модель CodeT5+ использует механизмы внимания и трансформеры, что позволяет ей эффективно захватывать семантические и синтаксические особенности кода.   
  
Рис. 2. Представление исходного кода в виде абстрактного синтаксического дерева (AST).

Источник: Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J.. UniXcoder: Unified Cross-Modal Pre-training for Code Representation.

Эмбеддинг мы получаем из представления исходного кода в виде абстрактного синтаксического дерева (AST) [Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J., URL], т. е. учитываем только логику работы программы, игнорируя комментарии и стилеметрические различия в написании исходного кода генеративной моделью и человеком.

Псевдокод генерации AST представления [Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J., URL]:

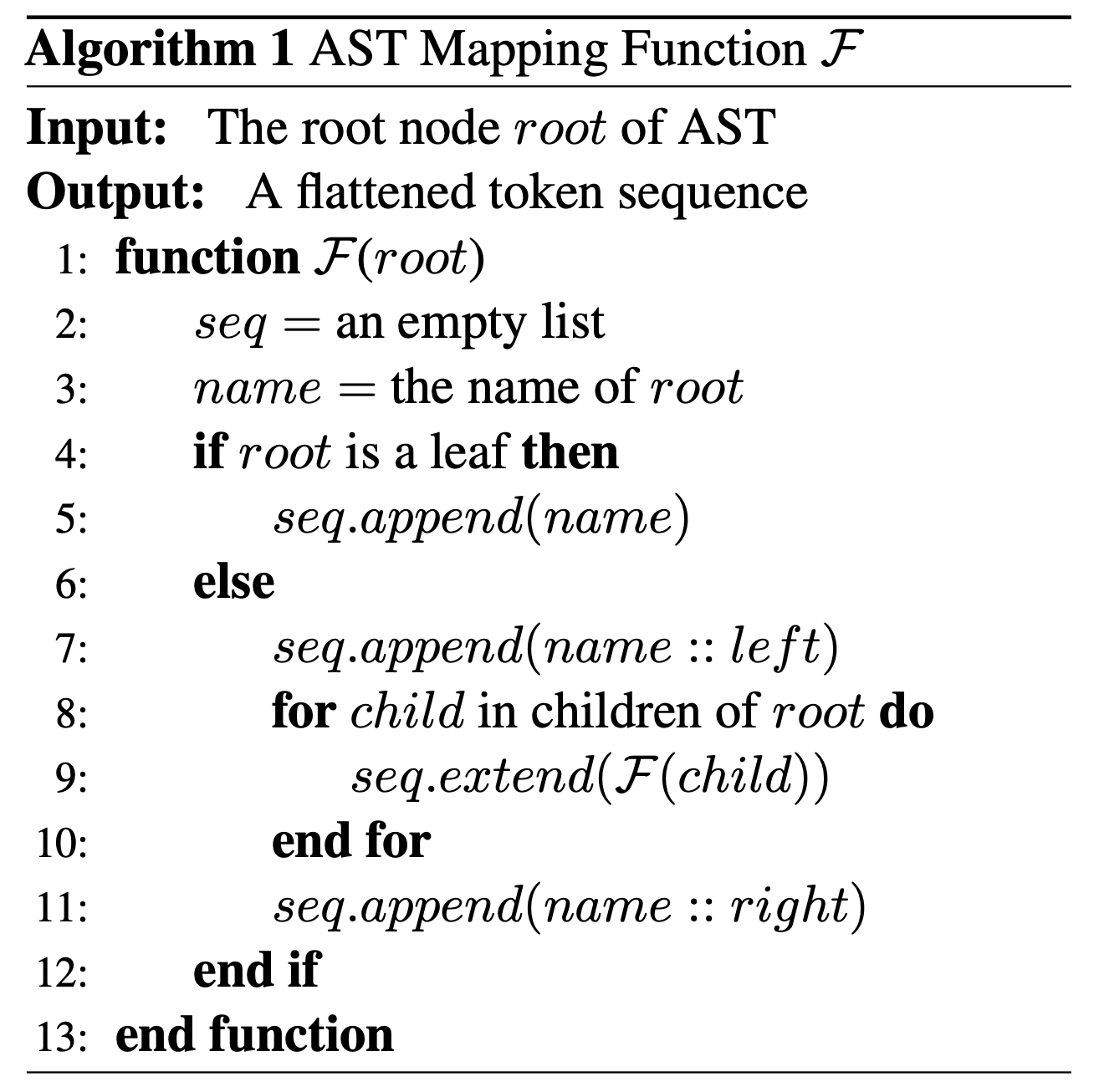


Рис. 3. Псевдокод формирования строкового представления AST-дерева

Источник: Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J.. UniXcoder: Unified Cross-Modal Pre-training for Code Representation.

## 2.2 Используемые датасеты

### 2.2.1. Требования к датасетам

Чтобы построить классификатор, мы сформулировали некоторые требования к датасетам:

1. Датасет должен содержать описания задач и их решения людьми.
2. Задачи должны быть доступны для решения разработчикам начального уровня.
3. Датасет должен содержать не менее 400 задач с решением на языке программирования Python.

### 2.2.2. Обоснование выбора датасетов

Нами было найдено несколько датасетов, удовлетворяющих нашим требованиям: MBPP [Hugging Face. Dataset «MBPP: A Python…, URL], HumanEval-X [Hugging Face. Dataset «HumanEval-X: Multilingual Evaluation…», URL]. CodeSearchNet [Hugging Face. Dataset «CodeSearchNet: A Benchmark…, URL] содержит код решения задач из разных предметных областей, но большая часть задач недоступна для решения начинающим разработчикам.

  
Рис. 4. Пример записи из датасета HumanEval-X

Источник: Hugging Face. Dataset «HumanEval-X: Multilingual Evaluation of Code Generation Models»

HumanEval-X [Hugging Face. Dataset «HumanEval-X: Multilingual Evaluation…», URL] содержит мало записей (164 строки для выборки “Python”), а также не содержит четких постановок задач, он больше подойдет для задач дополнения исходного кода (code completion problem). Наш выбор остановился на открытом датасете MBPP [Hugging Face. Dataset «MBPP: A Python…, URL] от Google. Пример строки из выбранного датасета:

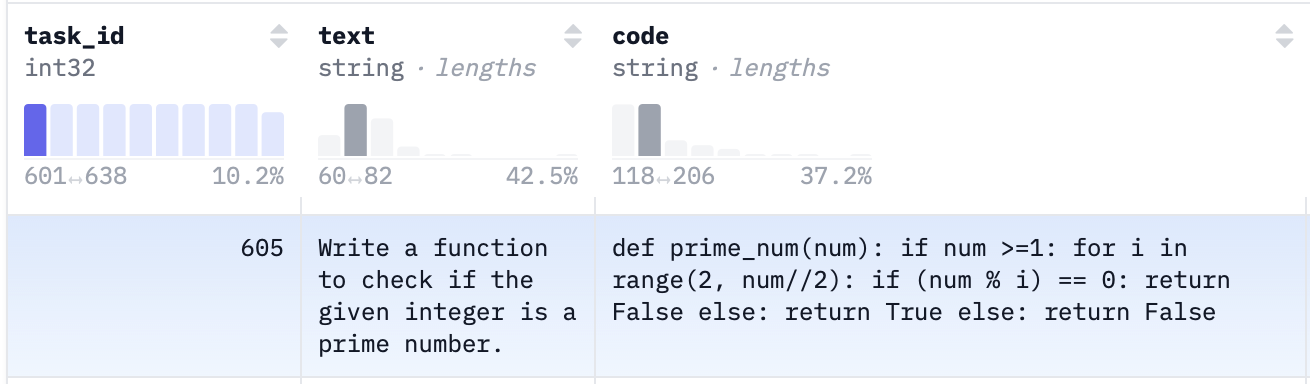


Рис. 5. Пример записи из датасета MBPP

Источник: Hugging Face. Dataset «MBPP: A Python Programming Benchmark» // Hugging Face.

MBPP [Hugging Face. Dataset «MBPP: A Python…, URL] содержит формулировки заданий для программистов начального уровня и их решения реальными людьми. Датасет имеет 2 вариации: очищенный (427 строк) и полный (974 строки). Отличие в том, что в очищенном удалены символы (разделители, например, ‘\n’, ‘\t’, ‘\r’ и др.).

Датасет состоит из четырех выборок: ‘train’ (374 записи), ‘test’ (500 записей), ‘valid’ (90 записей) и ‘prompt’ (10 записей). Набор полей датасета: 'task\_id', 'text', 'code', 'test\_list', 'test\_setup\_code', 'challenge\_test\_list'.

### 2.2.3. Дополнение датасета решениями, сгенерированными ИИ

Чтобы дополнить датасет решениями, сгенерированными LLM, было принято решение использовать технику Awesome Prompting, заключающуюся в работе с LLM в потоковом режиме и написании первого сообщения по шаблону “I want you to act as an entry level programmer. I will give you natural language specification problems. Your goal is to write code solutions for given problems on Python without commentaries and doc-strings. My first request is {code}”. [GitHub**.** Awesome ChatGPT Prompts., URL]. Использование данной техники повышает вероятность того, что LLM поймет контекст и в ответе выведет именно исходный код на Python без комментариев и док-строк.

2.2.4. Описание датасета реальных данных

Нам был предоставлен доступ к датасету реальных студенческих работ посредством LMS. Мы взяли выгрузки по каждой работе в формате zip-архивов. Каждый zip-архив содержал внутри себя набор папок с ФИО, а в каждой папке - word-файлы отчетов студентов. Большая часть word-файлов (~99.7% работ) была написана по соответствующей структуре, исходный код для каждого задания находился между блоками: “Текст программы на языке Python” и “Результат тестирования программы на языке Python”, поэтому нам не составило труда извлечь его с помощью соответствующего регулярного выражения. Однако, нам также встречались пустые работы, работы без исходного кода внутри, работы с скриншотами кода. Такие работы мы на данный момент отбрасываем, они составляют ~0.3%.

## 2.3. Выбор классификатора

В исследовании [Suh H., Tafreshipour M., Li J., Bhattiprolu A., Ahmed I., URL] был проведен сравнительный анализ различных классификаторов на открытых датасетах, для случая с температурой по умолчанию для LLM, наилучшим образом показала себя логистическая регрессия. Для обучения классификатора мы разделим достроенный датасет из п. 2.2.3 в соотношении 80:10:10, где 80% - обучающий набор, 10% - валидационный и 10% - тестовый.

## 2.4. Подбор гиперпараметров

Для подбора гиперпараметров мы воспользовались методом GridSearchCV из библиотеки Scikit-Learn [Scikit-learn. GridSearchCV., URL], составив сетку следующим образом:

Таблица 1

Сетка гиперпараметров

| Название параметра | Описание | Диапазон значений сетки | Оптимальное значение |
| --- | --- | --- | --- |
| C | Величина коэффициента регуляризации | 0.001, 0.01, 0.1, 1, 10, 100 | 100 |
| solver | Алгоритм для решения задачи на оптимизацию при подборе параметров GridSearch. | 'liblinear', 'saga' | liblinear |

## 2.5. Результаты

В данной работе метки классов “human” = 0, “ai” = 1.

Для оценки качества полученной модели воспользуемся следующей метрикой:

,

где Accuracy – точность,

TP – количество кодов, сгенерированных LLM и которые модель посчитала сгенерированными LLM,

TN – число кодов, которые были написаны людьми и которые модель посчитала написанными людьми,

FP – количество кодов, которые являются написанными людьми и которые модель ошибочно посчитала сгенерированными LLM,

FN – количество кодов, которые являются сгенерированными LLM и которые модель ошибочно классифицировала как написанными людьми.

1. True Positive Rate (Recall)

,

где TP – количество кодов, сгенерированных LLM и которые модель посчитала сгенерированными LLM,

TN – число кодов, которые были написаны людьми и которые модель посчитала написанными людьми,

FP – количество кодов, которые являются написанными людьми и которые модель ошибочно посчитала сгенерированными LLM.

1. True Negative Rate

,

где TP - количество кодов, сгенерированных LLM и которые модель посчитала сгенерированными LLM,

TN - число кодов, которые были написаны людьми и которые модель посчитала написанными людьми,

FP - количество кодов, которые являются написанными людьми и которые модель ошибочно посчитала сгенерированными LLM.

1. Средняя F1-мера

F1 - среднее гармоническое между precision и recall, где precision вычисляется по формуле ,

где Precision – точность,

TP - количество кодов, сгенерированных LLM и которые модель посчитала сгенерированными LLM,

TN - число кодов, которые были написаны людьми и которые модель посчитала написанными людьми,

FP - количество кодов, которые являются написанными людьми и которые модель ошибочно посчитала сгенерированными LLM.

,

где Avg F1 – среднее арифметическое между Human F1 и AI F1 ,

Human F1 – среднее гармоническое между precision и recall для класса ‘human’,

AI F1 – среднее гармоническое между precision и recall для класса ai.

Использование метрики Average F1 вместо Human F1 или AI F1 было обусловлено тем, что оно позволяет получить более сбалансированное представление о производительности модели, учитывая оба класса. Это особенно важно в задачах, где классы могут быть несбалансированными, и одна из метрик (Human F1 или AI F1) может не полностью отражать качество модели.

Таблица 2

Результаты на обучающей выборке

| Метрика | Значение |
| --- | --- |
| Точность (Accuracy) | 0.779487 |
| True Positive Rate | 0.759259 |
| True Negative Rate | 0.804598 |
| Human F1 Score | 0.792271 |
| AI F1 Score | 0.765027 |
| Average F1 Score | 0.778649 |

Источник: составлено авторами.

Таблица 3

Результаты обучения на валидационной выборке

| Метрика | Значение |
| --- | --- |
| Точность (Accuracy) | 0.841026 |
| True Positive Rate | 0.836655 |
| True Negative Rate | 0.786517 |
| Human F1 Score | 0.818713 |
| AI F1 Score | 0.858447 |
| Average F1 Score | 0.838581 |

Источник: составлено авторами.

Таблица 4

Результаты обучения на тестовой выборке

| Метрика | Значение |
| --- | --- |
| Точность (Accuracy) | 0.779487 |
| True Positive Rate | 0.781928 |
| True Negative Rate | 0.759259 |
| Human F1 Score | 0.792271 |
| AI F1 Score | 0.765027 |
| Average F1 Score | 0.778649 |

Источник: составлено авторами.

Также решено было собрать статистику на реальных работах студентов, но так как мы были ограничены во времени, мы собрали статистику по двадцати случайным работам по каждой из трех тем: “Сортировки списков”, “Использование функций для работы с одномерными массивами” и “Написание пользовательских функций”. Тема “Сортировки списков” была выбрана на основе нашего предположения, что малое число студентов самостоятельно реализует алгоритм сортировки (зачастую люди переписывают код с книг по алгоритмам или сгенерированный ChatGPT). Тема “Написание пользовательских функций” была выбрана, чтобы посмотреть как алгоритм отреагирует на AST-представление кодов с множеством функций, а тема “Использование функций для работы с одномерными массивами”, как нам кажется, лучше всего отражает “средний” код первокурсника, там используются функции, но также и не только они.

Таблица 5

Результаты работы нейросетевого классификатора на датасете студенческих работ

| Тема практической работы | AI | Human |
| --- | --- | --- |
| Сортировки списков | 20 | 0 |
| Использование функций для работы с одномерными массивами | 18 | 2 |
| Написание пользовательских функций | 15 | 5 |

Из таблицы 5 видно, что модель чаще всего классифицирует код как сплагиаченный у нейросети. Исходные коды действительно могут оказаться сплагиаченными, однако нам кажется, что проблема присутствует и в самой модели. Отличить код человека от нейросети - отнюдь не тривиальная задача. Поэтому в качестве точек роста для этого проекта мы выделяем следующие: рассмотрение различных представлений исходного кода (AST Only, Code Only, Code+AST) [Suh H., Tafreshipour M., Li J., Bhattiprolu A., Ahmed I., URL], рассмотрение моделей, захватывающих разные признаки исходного кода (например, смотреть не только на логическое содержание программ, но и на стилеметрию), использование другого набора классификаторов в сочетании с приведенными представлениями и моделями, разных способов подбора гиперпараметров. Также возможна полная смена методологии проверки решений на нейросетевой антиплагиат.

# ГЛАВА 3. ПОСТРОЕНИЕ КЛАССИФИКАТОРА МЕЖСТУДЕНЧЕСКОГО ПЛАГИАТА

## 3.1. Постановка задачи межстуденческого плагиата

Дано: X - пара исходных кодов решений студентов из отчетов, Y - класс, где 0 - плагиат не обнаружен, 1 - обнаружен.

Построить: классификатор исходных кодов .

Для того, чтобы решить проблему межстуденческого плагиата нами было найдено решение:

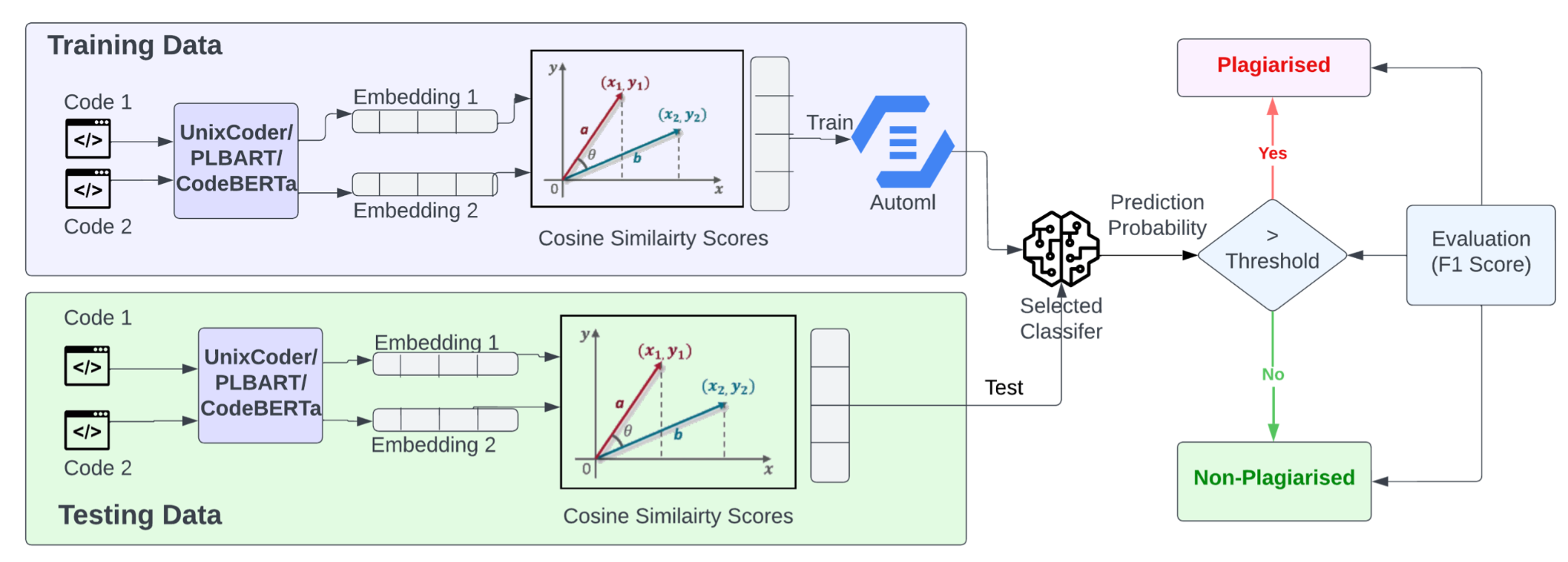


Рис. 6. Методология поиска схожести кода с помощью косинусного схосдтва

Источник: Ebrahim F., Joy M.. Source Code Plagiarism Detection with Pre-Trained Model Embeddings and Automated Machine Learning.

Для решения задачи межстуденческого плагиата используется собственная теоретическая разработка: рассчитывать среднее косинусное расстояние между парами эмбеддингов, полученных из исходных кодов, и определим порог, по достижении которого пара работ будет считаться сплагиаченной.

## 3.2. Выбор эмбеддинг-моделей

В качестве моделей для получения эмбеддингов были выбраны PLBART и CodeT5+. Каждая из них учитывает различные аспекты исходного кода. В то время как PLBART учитывает стилеметрические особенности, комментарии, а также поток данных, CodeT5+ в сочетании с представлением исходного кода в виде абстрактного синтаксического дерева (AST) [Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J., URL], учитывает исключительно логику работы программы: последовательность выполнения инструкций, ветвления, циклы, функции и т. п.

## 3.3. Результаты

Мы решили проверить работу метода на датасете реальных студенческих работ из п 2.2.4, варьируя порог схожести, получили

Таблица 5

Результат работы межстуденческого классификатора на датасете студенческих работ

| Тема практической работы | Порог | Количество пар схожих работ |
| --- | --- | --- |
| Сортировки списков | 20% | 10 |
| 50% | 8 |
| 70% | 8 |
| Использование функций для работы с одномерными массивами | 20% | 10 |
| 50% | 9 |
| 70% | 7 |
| Написание пользовательских функций | 20% | 10 |
| 50% | 9 |
| 70% | 6 |

Получили, что схожесть работ довольно высокая, даже для относительно несхожих работ, схожесть которых мы оценили самостоятельно. Ясно, что модель нуждается в доработке, стоит меньше обращать внимания на стилеметрические особенности и больше на логику кода.

# ГЛАВА 4. РЕАЛИЗАЦИЯ

## 4.1. Архитектура приложения

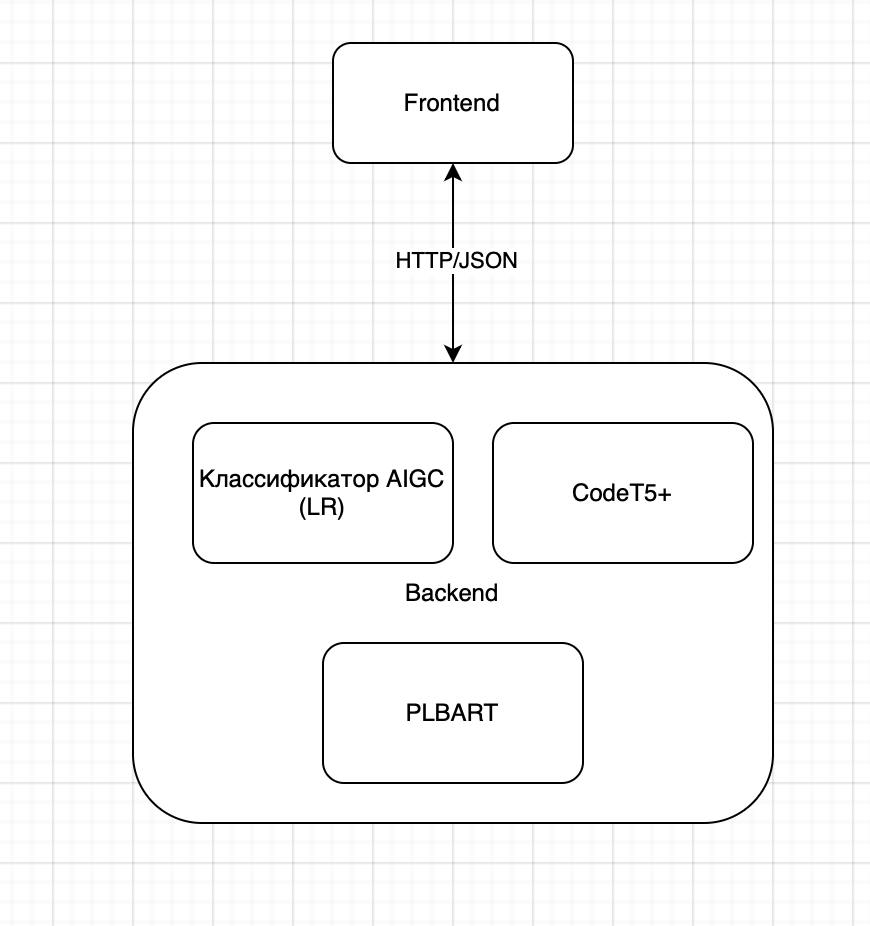


Рис. 7. Архитектура веб-приложения PlagiCode

Источник: составлено авторами.

Архитектура PlagiCode содержит 2 основных сервиса: Frontend и Backend, взаимодействующих посредством REST API.

Backend-сервис содержит следующие взаимодействующие компоненты:

1. Слой для работы с классификаторами, получающими эмбеддинги на вход и возвращающими масиив ‘label’(значений ‘ai’или ‘human’). На момент написания отчета используется логистическая регрессия.
2. Слой для работы с эмбеддинг-моделями, получающими на вход исходные коды в виде строки и возвращающими эмбеддинги. На момент написания отчета используются 2 слоя, работающие с моделями CodeT5+ и PLBART. Отличие состоит в том, что для CodeT5+ мы генерируем AST [Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J., URL], а в PLBART отдаем неизменный исходный код, даже без удаления комментариев.

Архитектура была выстроена таким образом, чтобы легко можно было подменять модели (предполагается, что далее в проекте будут использоваться иные модели), для изменения модели достаточно реализовать необходимый интерфейс (т. е. благодаря “утиной” типизации языка Python, реализовать все методы интерфейса).

## 4.2. Реализация frontend-сервиса

React:

Компоненты: Приложение было разбито на отдельные функциональные компоненты (например, компонент загрузки файлов, компонент отображения результатов, компонент навигации). Это позволило создавать переиспользуемые блоки кода и улучшило структуру приложения.

JSX: Разметка приложения создана с помощью JSX, что объединяет HTML и JavaScript, повышая читаемость и удобство разработки.

State и Props: Компоненты будут использовать state для хранения локальных данных и props для передачи данных между компонентами. Это позволит управлять состоянием приложения и обеспечивать связь между разными частями кода.

Функциональные и Классовые компоненты: В зависимости от сложности и потребностей, компоненты могут быть реализованы как функциональные или классовые. Функциональные компоненты, как правило, более предпочтительны для небольших задач, в то время как классовые компоненты могут быть полезны для более сложных задач с жизненными циклами и управлением состояниями.

JavaScript (логика):

API интеграция: JavaScript код будет отвечать за взаимодействие с серверной частью приложения (например, через API) для загрузки файлов, отправки данных на проверку и получения результатов. Важной частью является организация запросов с обработкой успешных и ошибочных ответов.

Обработка данных: JavaScript код будет обрабатывать полученные от сервера данные, преобразовывать их в подходящий формат для отображения и отображать на веб-странице с помощью React.

Логика проверки: JavaScript код может содержать (или вызывать) функциональность проверки на плагиат, например, сравнение загруженных файлов, использование внешних API антиплагиата и обработку результатов.

Функции для управления данными: Функции для управления состоянием приложения, например, добавление, обновление и удаление данных.

CSS:

Внешний вид приложения: CSS будет отвечать за внешний вид приложения, стилизацию элементов (цвет, шрифт, размер, расположение). Это позволит создать привлекательный и удобный интерфейс для пользователя.

Стилизация компонентов: Стилизация отдельных компонентов будет реализована в отдельном CSS файле или с помощью styled-components для повышения читаемости и поддерживаемости.

Адаптивность:CSS будет обеспечивать адаптивный дизайн, чтобы приложение выглядело одинаково хорошо на различных устройствах и разрешениях экранов.

React Router DOM:

Маршруты: С помощью React Router DOM будут определены различные маршруты для разных страниц (страница загрузки файлов, страница результатов нейросетевого антиплагиата, страница межстуденческого антиплагиата).

Навигация: Вся навигация между страницами (по кнопкам) осуществляется через React Router DOM.

Компоненты маршрутов: Компоненты (определённые для каждого маршрута) отрисовывают содержимое страниц приложения.

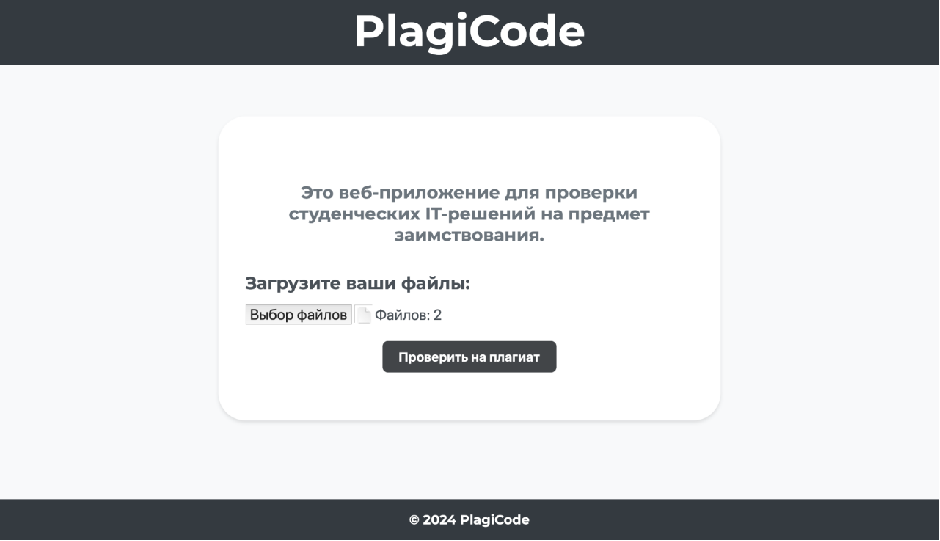


Рис. 8. Страница загрузки файлов для проверки

Источник: составлено авторами.

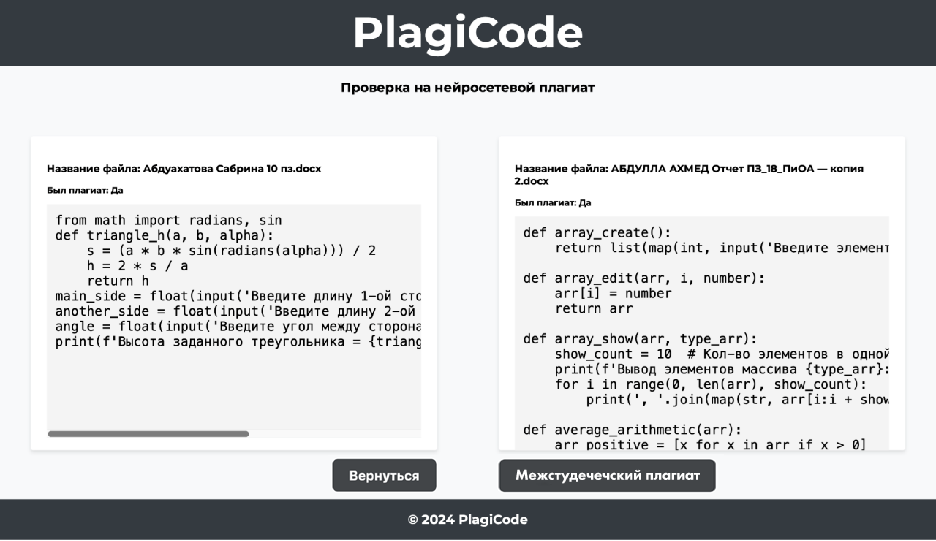


Рис.9. Страница результата проверки отчетов на нейросетевой плагиат

Источник: составлено авторами.

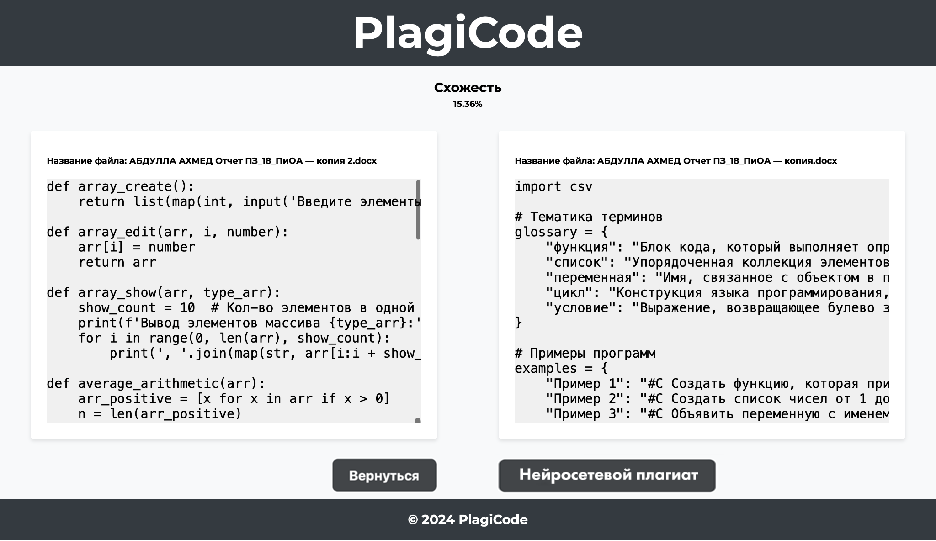


Рис. 10. Страница результата проверки отчетов на межстуденческий плагиат

Источник: составлено авторами.

## 4.3. Реализация backend-сервиса

Для загрузки датасета MBPP с HuggingFace мы использовали библиотеку datasets для языка программирования Python. При построении абстрактного синтаксического дерева (AST [Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J., URL]) была использована библиотека tree-sitter. Для загрузки модели CodeT5+ с HuggingFace мы использовали библиотеку transformers. Чтобы генерировать эмбеддинги при помощи модели была использована библиотека torch. Чтобы сохранить эмбеддинги и соответствующие им метки (‘ai’, ‘human’), мы использовали библиотеку numpy. В качестве классификатора мы использовали логистическую регрессию с подбором гиперпараметров grid search из пакета scikit-learn. Для обработки файлов отчетов мы использовали библиотеку python-docx, для обеспечения возможности загрузки и обработки zip-архивов был использован пакет zipfile из стандартной библиотеки Python. Для быстрой разработки организации клиент-серверного взаимодействия был использован фреймворк FastAPI. Чтобы программный продукт можно было задеплоить на любой станции и сделать независимым от ОС хоста, для облегчения горизонтального масштабирования, мы использовали Docker.

## 4.4. Организация взаимодействия между компонентами

Описание REST API клиент-серверного взаимодействия:

POST /check-plagiarism

Заголовки запроса:

Content-Type: multipart/form-data

Тело запроса должно быть в формате multipart/form-data и включать поле files с файлами с расширениями .zip или .docx. На данный момент предполагается, что пользователь будет делать выгрузку из lms в формате zip, но также можно отправлять одиночные работы в формате docx.

Ответ backend-сервиса формируется в формате JSON. Если запрос успешно обработан, то получаем HTTP статус 200 OK и тело ответа:

{

"files": {

"Отчет1.docx": {

"codes": [

“code1”,

“code2”

],

"is\_aigc": [

true,

false

]

},

"Отчет2.docx": {

"codes": [

“code1”,

“code3”

],

"is\_aigc": [

true,

false

]

}

},

"similarity\_results": [

{

"file1": "Отчет1.docx”,

"file2": "Отчет2.docx",

"code1": "code1",

"code2": "code1",

"similarity": 1

},

{

"file1": "Отчет1.docx”,

"file2": "Отчет2.docx",

"code1": "code1",

"code2": "code3",

"similarity": 0.4691471219062805

},

{

"file1": "Отчет1.docx”,

"file2": "Отчет2.docx",

"code1": "code2",

"code2": "code1",

"similarity": 0.4691471219062805

},

{

"file1": "Отчет1.docx”,

"file2": "Отчет2.docx",

"code1": "code2",

"code2": "code3",

"similarity": 0.4691471219062805

},

]

}.

В противном случае отправляется код ошибки и ее пояснение на русском языке в теле ответа в формате JSON {“error”: “Пояснение ошибки”}. Возможные коды ответа и описания ошибок:

Таблица 6

| Код ответа | Описание ошибки |
| --- | --- |
| 400 | {file.filename} должен быть в формате .docx или .zip |
| 400 | {filename} не содержит текста |
| 400 | {filename} не содержит исходный код или его невозможно извлечь |
| 500 | Внутренняя ошибка сервера |

Каждый сервис был упакован в Docker-контейнер, а для организации взаимодействия контейнеров с ОС хоста (сетевого моста и общего тома) и друг с другом (сетевого моста) мы использовали Docker-compose.

# ЗАКЛЮЧЕНИЕ

Функциональные возможности проекта:

* Загрузка файлов: Пользователь может загружать файлы отчётов выполненных работ в формате ‘.DOCX’;
* Проверка на плагиат: Алгоритм сравнивает загруженные файлы с API антиплагиата;
* Настройка проверки: Пользователь может проверить отчёт на нейросетевой, а также на межстуденческий плагиат.
* Множественная проверка:Пользователь может загрузить несколько файлов для одновременной обработки.
* Просмотр результатов: Программа предоставляет удобный интерфейс для отображения результатов проверки.
* Отображение результатов по уровням совпадений: Указание плагиата в файле в виде “Да/Нет” при нейросетевом плагиате и указание совпадения в процентном соотношении в межстуденческом плагиате.

Пользовательский интерфейс:

* Интуитивно понятный интерфейс:Простота навигации и понимания функций;
* Быстрая загрузка: Веб-приложение загружается и работает быстро, чтобы минимизировать время ожидания;
* Адаптивный дизайн: Поддержка различных разрешений экранов для комфортной работы в любом месте;
* Удобный вид результатов: Отчеты, таблицы и графики должны быть понятными и наглядными для анализа;
* Загрузка/отправка: Удобный и интуитивно понятный процесс загрузки и отправки файлов;
* Визуализация результатов: Присутствуют визуализация обработанного кода, вероятность плагиата.

Организация тестирования:

* Тестирование функциональности: Проверка каждой функции, чтобы убедиться, что она работает корректно;
* Тестирование на разных типах загрузки файлов: Загрузка нескольких файлов формата ‘.DOCX’, загрузка архива формата ‘.zip’;
* Тестирование на разных объёмах данных: Проверка производительности при работе с большим количеством файлов и результатами;
* Тестирование на разных браузерах и устройствах: Проверка работы на различных браузерах (Chrome, Yandex, Safari);

Подтверждение решения задач:

* Отчёт о проверке: Генерация отчёта с результатами проверки, показывающего наличие (или отсутствие) плагиата;
* Сравнение результатов: Возможность сравнить собственные результаты с результатами, полученными другими средствами;
* Обратная связь с преподавателем: Возможность передачи результатов проверки преподавателю для получения обратной связи;
* Повышение качества обучения: Уменьшение случаев списывания и стимулирование самостоятельного освоения материала;
* Уменьшение количества плагиата: В рамках возможностей, система позволит преподавателю выявлять плагиат и работать над предотвращением его появления;
* Статистика плагиата: Возможность отслеживать статистику использования и проводить анализ на предмет выявления источников плагиата, с последующим внесением улучшений в систему.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Suh H., Tafreshipour M., Li J., Bhattiprolu A., Ahmed I.. An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We? // arXiv preprint. 2024. URL:<http://www.arxiv.org/pdf/2411.04299> (дата обращения: 16.12.2024);
2. Guo D., Lu S., Duan N., Wang Y., Zhou M., Yin J.. UniXcoder: Unified Cross-Modal Pre-training for Code Representation // arXiv preprint. 2022. URL:<https://arxiv.org/pdf/2203.03850> (дата обращения: 16.12.2024).
3. Hugging Face. Dataset «MBPP: A Python Programming Benchmark» // Hugging Face. URL:<https://huggingface.co/datasets/google-research-datasets/mbpp?row=4> (дата обращения: 16.12.2024).
4. Hugging Face. Dataset «HumanEval-X: Multilingual Evaluation of Code Generation Models» // Hugging Face. URL:<https://huggingface.co/datasets/THUDM/humaneval-x> (дата обращения: 16.12.2024).
5. Hugging Face. Dataset «CodeSearchNet: A Benchmark for Code Understanding» // Hugging Face. URL:<https://huggingface.co/datasets/code-search-net/code_search_net> (дата обращения: 16.12.2024).
6. Ebrahim F., Joy M.. Source Code Plagiarism Detection with Pre-Trained Model Embeddings and Automated Machine Learning // ACL Anthology. 2023. URL:<https://aclanthology.org/2023.ranlp-1.34.pdf> (дата обращения: 16.12.2024).
7. GitHub**.** Awesome ChatGPT Prompts. // GitHub. URL:<https://github.com/f/awesome-chatgpt-prompts> (дата обращения: 20.12.2024).
8. Scikit-learn. GridSearchCV. // Scikit-learn. URL:<https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.GridSearchCV.html> (дата обращения: 20.12.2024)