Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ — Информатика и системы управления
КАФЕДРА ИУ7 — Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент

Климов Илья Сергеевич

Группа

ИУ7-82Б

Тип практики

преддипломная практика

Название предприятия

НУК ИУ МГТУ им. Н. Э. Баумана

Студент	_____	**Климов И. С.**
Руководитель практики от предприятия	_____	**Вишневская Т. И.**
Руководитель практики	_____	**Кострицкий А. С.**

Оценка_____

2023 г.

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
_____ Рудаков И. В.
«13» мая 2023 г.

З А Д А Н И Е
на прохождение производственной практики (преддипломная практика)

Студент **4** курса группы **ИУ7-82Б**

Климов Илья Сергеевич

в период с 15.05.2023 г. по 28.05.2023 г.

Предприятие: **НУК ИУ МГТУ им. Н. Э. Баумана**

Руководитель практики от предприятия (наставник):

Вишневская Т. И.

Руководитель практики от кафедры:

Кострицкий А. С.

Задание:

1. Изучить программные средства проектирования и разработки информационных систем.

2. Собрать материалы в области разработки информационных систем.

3. Получить практические навыки в области разработки информационных систем.

Дата выдачи задания «13» мая 2023 г.

Руководитель практики от кафедры _____ **Кострицкий А. С.**

Руководитель практики от предприятия _____ **Вишневская Т. И.**

Студент _____ **Климов И. С.**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во время выполнения выпускной квалификационной работы был разработан метод обнаружения дефектов программного обеспечения (ПО) с использованием алгоритма градиентного бустинга.

1 Основная часть

1.1 Формализованная постановка задачи

Перед выполнением работы необходимо формализовать задачу обнаружения дефектов ПО. Наиболее удобным способом для этого является диаграмма в нотации IDEF0. Поставленная задача представлена на рисунке 1. На вход системе подаются набор данных для обучения и исходный код на языке программирования C++, содержащий блоки с функциями. Система путем статического анализа кода с использованием алгоритма градиентного бустинга определяет и размечает блоки, подверженные дефектам, то есть места в коде, в которых вероятней всего будет возникать отклонения от первоначальных бизнес-требований.

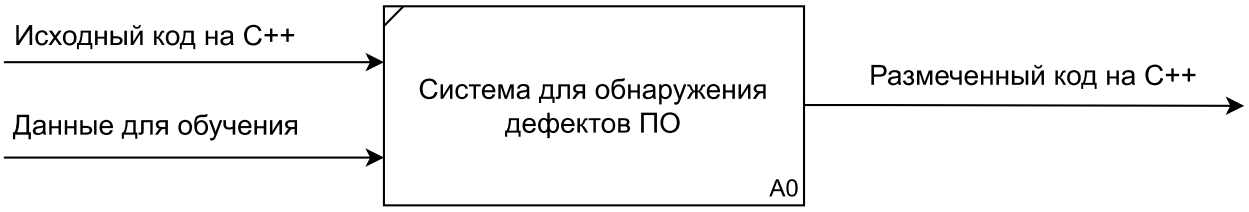

Рисунок 1 – IDEF0-диаграмма метода обнаружения дефектов ПО

1.2 Метод обнаружения дефектов ПО

Предлагаемый метод обнаружения дефектов разрабатываемого ПО представляет собой анализ текста программ, написанных на языке программирования C++. Предполагается, что на вход подается код, состоящий из функций, каждая из которых занимает не более 2000 строк, и написанный с соблюдением общих правил оформления кода.

Данный метод включает в себя несколько этапов. Перед выполнением процесса обнаружения дефектов происходит обучение модели градиентного бустинга на обучающей выборке. Код, для которого нужно определить наличие

дефектов, делится на блоки, содержащие входящие функции. Для каждого блока вычисляются метрики. Вычисленные значения поступают обученной модели, где происходит классификация каждого из блоков по наличию или отсутствию дефекта. Последним этапом метода является разметка: необходимо графически отметить места в коде, в которых вероятней всего имеются дефекты, а также вероятность этого для каждого блока.

На рисунке 2 представлена детализированная IDEF0-диаграмма разработанного метода.

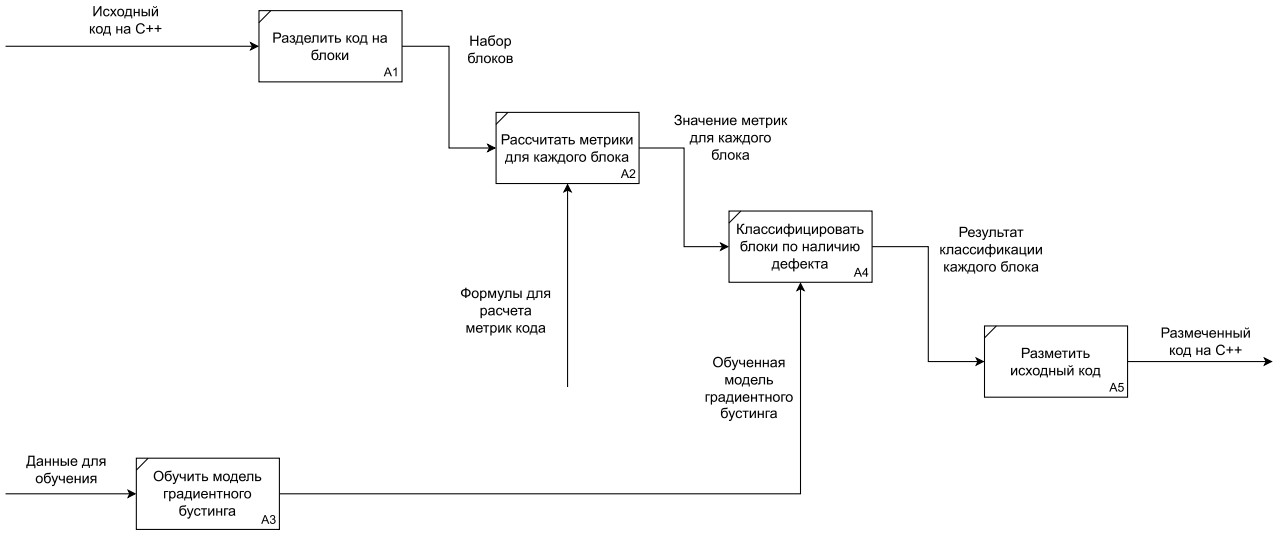

Рисунок 2 – Детализированная IDEF0-диаграмма разработанного метода обнаружения дефектов ПО

1.3 Программная реализация разработанного метода

Реализованное программное обеспечение имеет многофайловую структуру, состоит из 5 классов, каждый из которых отвечает за определенный этап разработанного метода.

1. **PromiseDataset** – класс, отвечающий за загрузку и обработку набора данных для обучения. Единственный метод prepare производит обработку данных: анализ, удаление выбросов, дубликатов, масштабирование. Происходит разделение выборки на тренировочную и тестовую,

выделенные признаки и целевая переменная является результатом возврата.

2. **GBDDModel** – класс, в котором реализуется обучение методом градиентного бустинга. Атрибутами является классификатор XGBClassifier из библиотеки xgboost, использующий соответствующий алгоритм, и название файла, в котором сохраняется модель. Для подбора параметров используется метод grid_search, который в автоматическом режиме сопоставляет разные параметры и выбирает те, которые в совокупности показывают наибольшую точность. Исследования показали, что для данной задачи лучшим образом подходят следующие: learning_rate = 0.01 (темп обучения), n_estimators = 1000 (количество деревьев), max_depth = 7 (максимальная глубина обучаемых деревьев). Для контроля обучения применяется метод debug_fit, позволяющий по окончании отобразить на графике значения функции потерь на каждой итерации. Это дает возможность более точно изменять параметры и методы для обработки данных.

   Метод fit позволяет обучить модель и сохранить ее в соответствующий файл. Модель из файла используется в дальнейшем в методах predict и predict_proba, рассчитывающие на ее основе результат классификации и вероятности принадлежности каждому классу соответственно. Для итоговой оценки точности модели используются метрики, описанные в аналитической части работы (accuracy, precision, recall, f-measure).
3. **MetricsCppCode** – класс, в котором происходит расчет метрик кода на основе описанных формул. Метод count возвращает словарь, содержащий названия метрик и их значения. Также написаны вспомогательные функции для подсчета каждой из них в отдельности.
4. **Window –** класс, отвечающий за пользовательский интерфейс. Реализованы методы для обработки соответствующих кнопок: open_file – открытие файла и загрузка в текстовый редактор, clean_editor – очистка поля с текстом, run_searching – запуск работы алгоритма и разметка кода

на основе посчитанных вероятностей. Для разделения исходного кода на функции применяется split_code_by_lines, которая на основе регулярного выражения делит текст на блоки.

5. **SyntaxHighlighter** – класс, позволяющий отображать подсветку строк кода. В метод hightlight_line передается номер строки и формат, который применяется для данной строки.

На рисунке 3 представлена диаграмма классов разработанной программы.

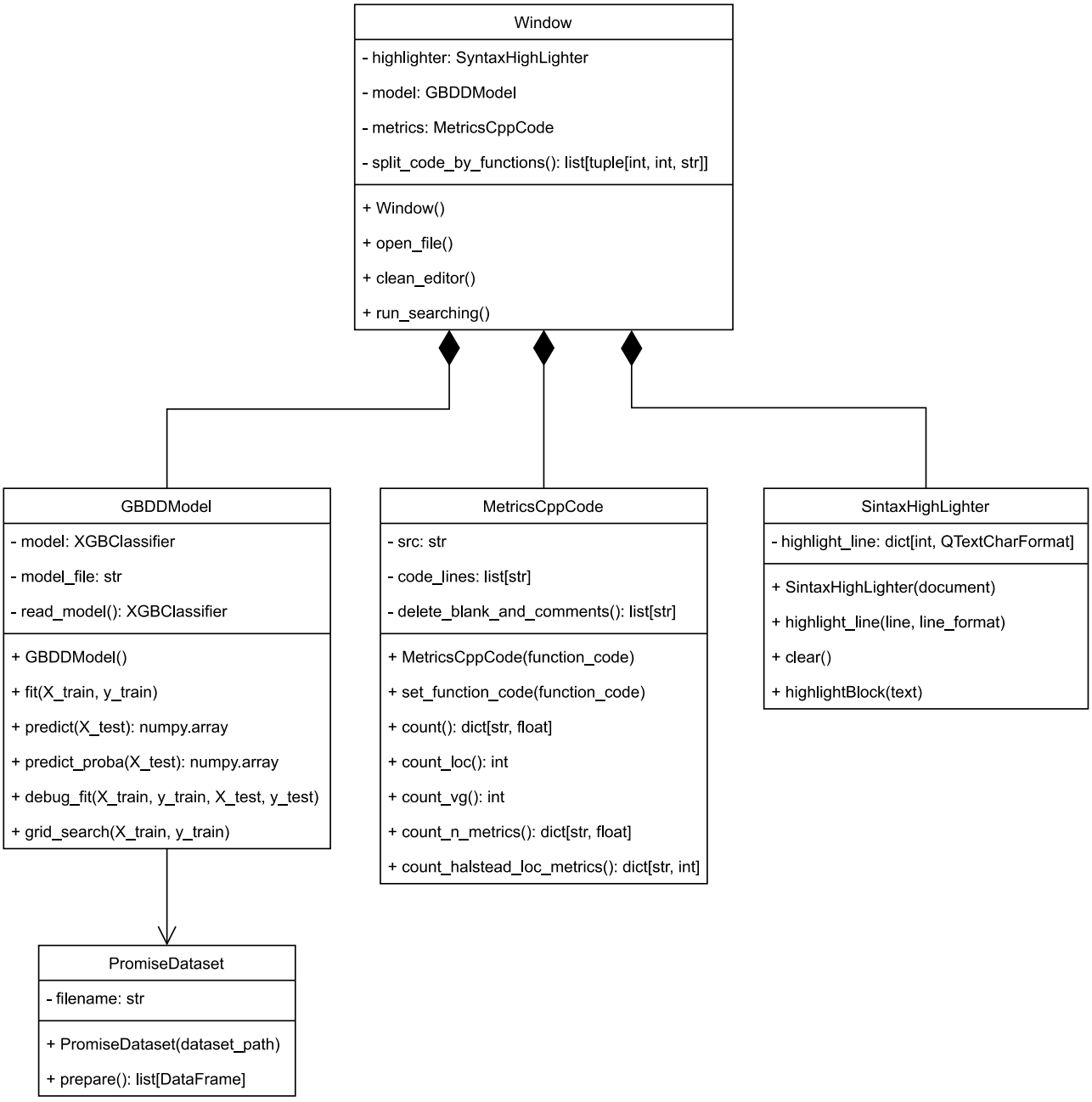

Рисунок 3 – Диаграмма классов разработанной программы

1.4 Примеры работы программы

Реализованная система обнаружения дефектов ПО представляет собой приложение, состоящее из текстового поля для ввода и загрузки текста программы и кнопок (загрузка файла, очистка окна и запуск алгоритма поиска дефектов). Пользовательский интерфейс представлен на рисунке 4.

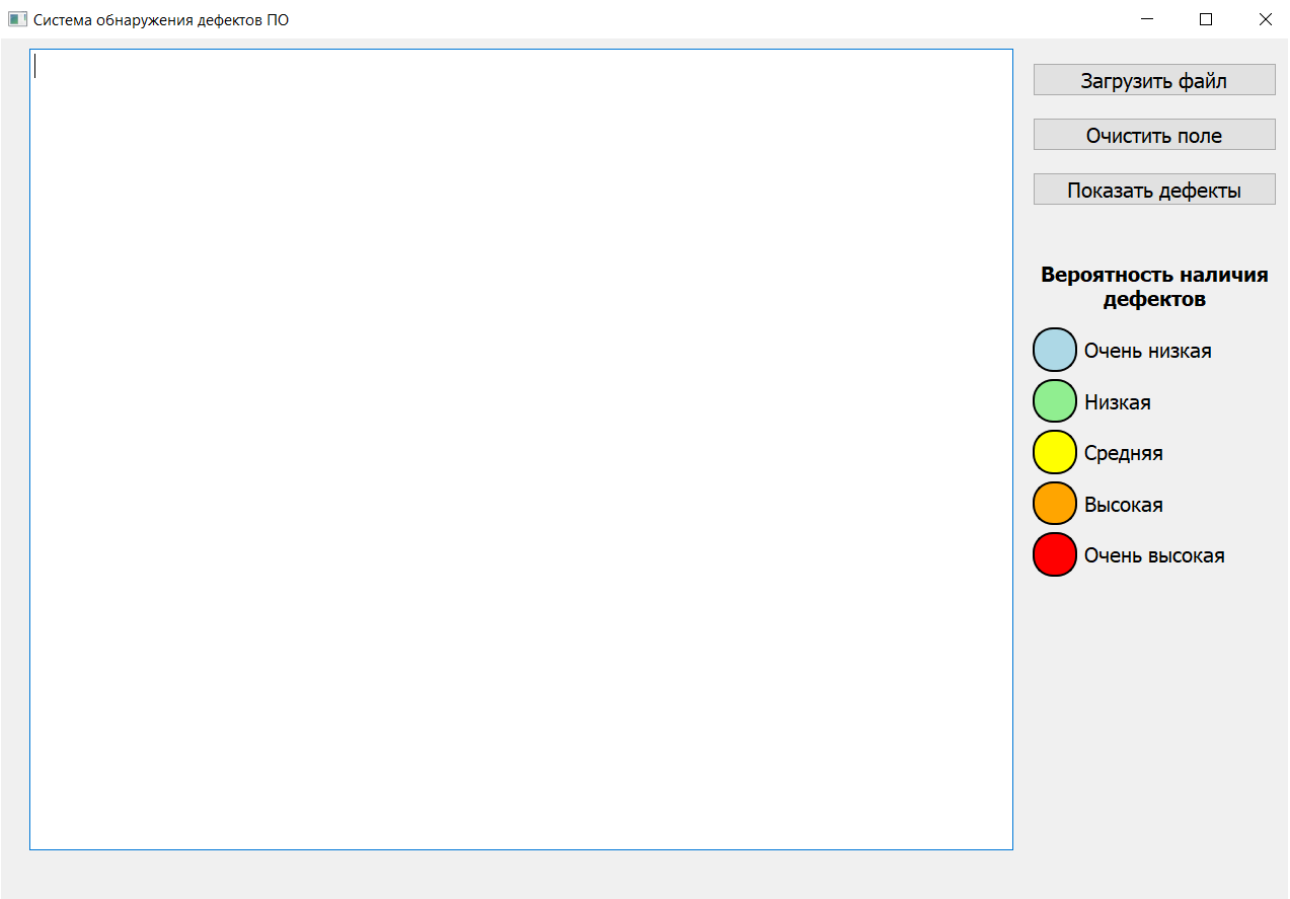

Рисунок 4 –Пользовательский интерфейс системы обнаружения дефектов ПО

Пользователь имеет возможность загрузить файл с кодом на C++ либо ввести его вручную. Выводится предупреждение об ограничения на размер исходных данных. После нажатия на кнопку «Показать дефекты» код размечается разными цветами в зависимости от вероятности нахождения в функции дефектов (голубой – очень низкая, 0-20%; зеленый – низкая, 20-40%; желтый – средняя, 40-60%; оранжевый – высокая, 60-80%; красный – 80-100%).

На основе результатов можно сделать вывод, какие функции необходимо тестировать в первую очередь.

На рисунках 5-6 представлены примеры работы программы.

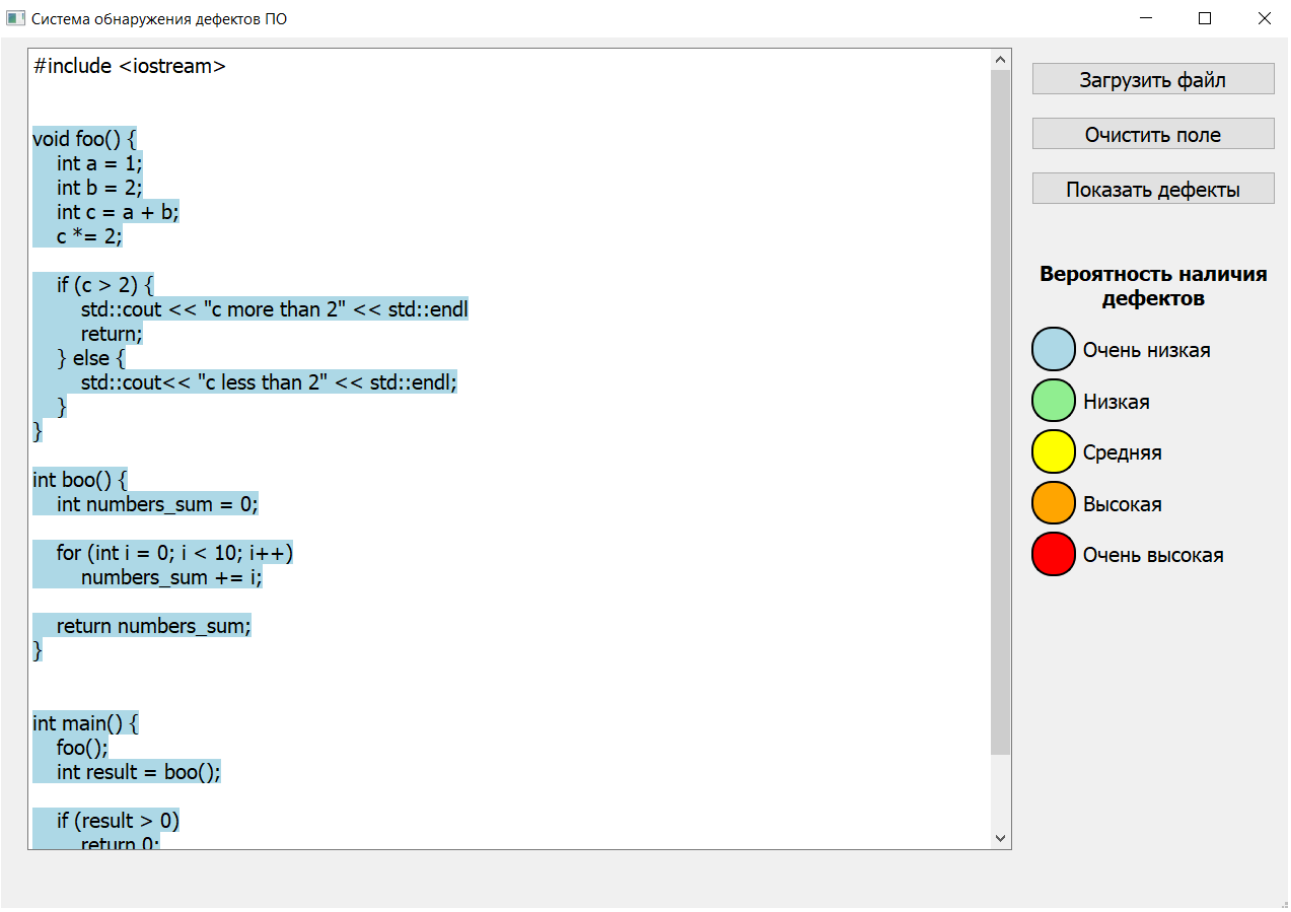

Рисунок 5 – Пример работы программы (низкая вероятность наличия дефектов)

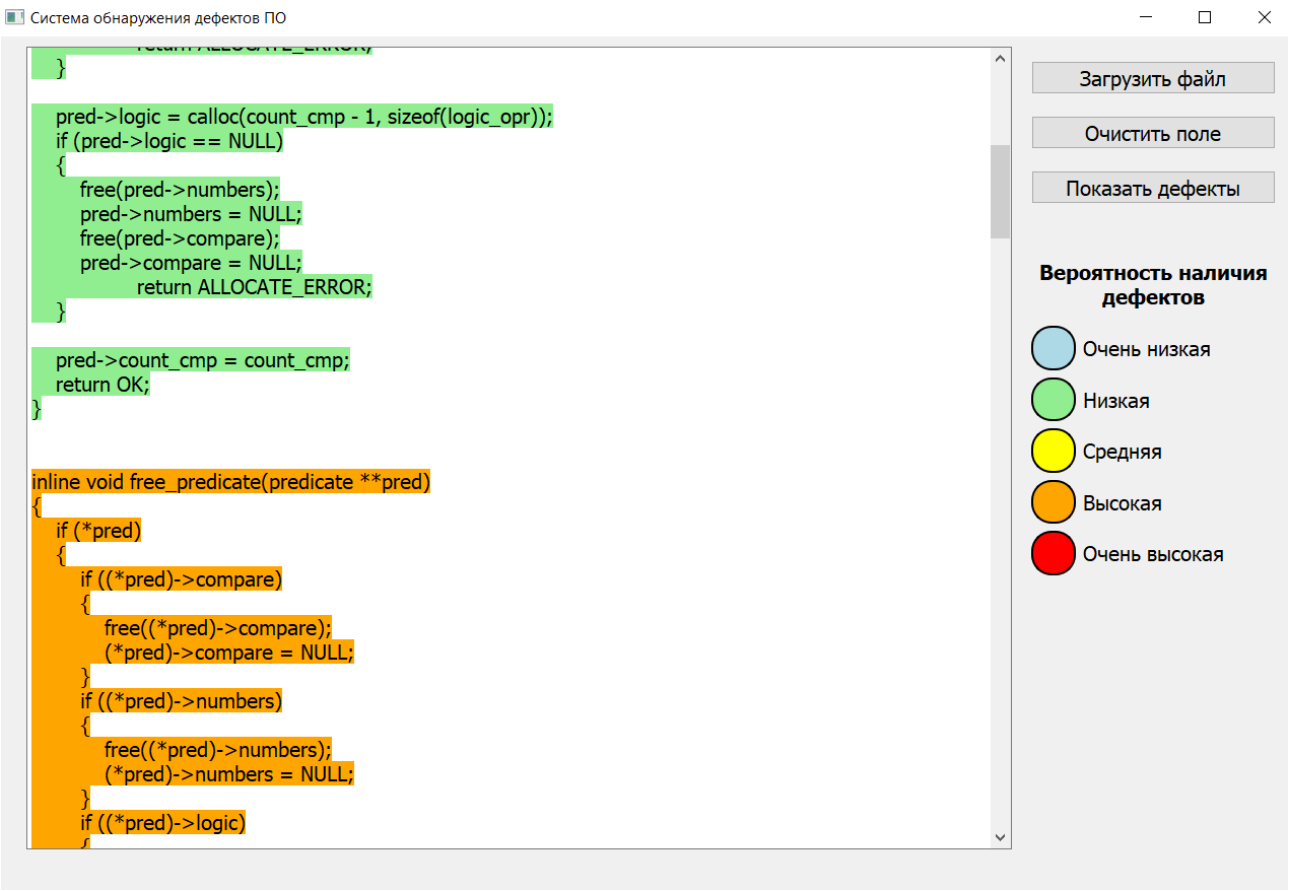

Рисунок 6 – Пример работы программы (низкая и высокая вероятности наличия дефектов)

ЗАКЛЮЧЕНИЕ

В результате выполнения работы было разработано программное обеспечение, демонстрирующее практическую осуществимость спроектированного в ходе выполнения выпускной квалификационной работы метода обнаружения дефектов ПО с использованием алгоритма градиентного бустинга.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Викторов, Д.С. Методика статического анализа для поиска дефектов естественной семантики программных объектов и ее программная реализация на базе инфраструктуры компилятора LLVM и фронтенда Clang [Текст] / Д.С. Викторов, Е.Н. Жидков, Р.Е. Жидков // Журнал Сибирского федерального университета. – 2018. – С. 801-810.
2. International Software Testing Qualifications Board. Стандартный глоссарий терминов, используемых в тестировании программного обеспечения. Версия 2.3 (от 9 июля 2014 года) [Текст] / ред. пер. Александр Александров. – С. 17.
3. ГОСТ Р 56920 – 2016. Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения. – М.: Стандартинформ, 2016. – С. 9.
4. Юхименко, Н.В. Обзор методов прогнозирования дефектов программного обеспечения [Текст] / Н.В. Юхименко, Ю.С. Белов // Программные продукты, системы и алгоритмы. – 2019. – №1 – С. 2.
5. QR Solutions. List of Defects in Software Testing | Severity & Priority in Testing [Электронный ресурс]: Режим доступа URL: https://qrsolutions.com.au/defects-in-software-testing/ (Дата обращения: 10.11.2022).
6. GeeksforGeeks. Techniques to Identify Defects [Электронный ресурс]: Режим доступа URL: https://www.geeksforgeeks.org/techniques-to-identify-defects/ (Дата обращения: 10.11.2022).
7. Logrocon Software Engineering. Что такое тестирование программного обеспечения? Определение, основы и типы [Электронный ресурс]: Режим доступа URL: https://logrocon.ru/news/testing_is (Дата обращения: 11.11.2022).
8. О развитии искусственного интеллекта в Российской Федерации [Текст]: указ Президента РФ от 10 октября 2019 г. №490 // Собр. законодательства РФ. – 2019. – 14 окт. – Ст. 5700.

9. Платонов, А.В. Машинное обучение: учебное пособие для вузов [Текст] / А.В. Платонов. – Москва: Издательство Юрайт, 2022. – 85 с. – ISBN 978-5-534-15561-7.
10. Microsoft Learn. Глубокое обучение и машинное обучение в Машинном обучении Azure [Электронный ресурс]: Режим доступа URL: https://learn.microsoft.com/ru-ru/azure/machine-learning/concept-deep-learning-vs-machine-learning (Дата обращения 14.11.2022).
11. Рашка, Р. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2. 3-е издание [Текст] / Рашка Р., Мирджалили В. // Москва: ООО «Диалектика», 2020 – 848 с.
12. Sharma, T. A Survey on Machine Learning Techniques for Source Code Analysis [Текст] / T. Sharma, M. Kechagia, S. Georgiou, R. Tiwari, I. Vats, H. Moazen, F. Sarro. – 2022. – С. 11-13.
13. Assim, A. Software Defects Prediction using Machine Learning Algorithms [Текст] / A. Assim, Q. Obeidat, M. Hammad // 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) – 2020. – С. 1-6.
14. Shah, M. Software Defects Prediction Using Machine Learning [Текст] / M. Shah, N. Pujara – 2020. – С. 1-5.
15. Iqbal, A. Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets [Текст] / A. Iqbal, S. Aftab, U. Ali, A. Husen // International Journal of Advanced Computer Science and Applications – 2019. – С. 1-6.
16. Aleem, S. Comparative performance analysis of machine learning techniques for software bug detection / S.Aleem, L.F. Capretz, F. Ahmed – 2015. – С. 1-9.
17. Cetiner, M. A Comparative Analysis for Machine Learning based Software Defect Prediction Systems [Текст] / M. Cetiner, O.K. Sahingoz // 11th International Conference on Computing, Communication and Networking Technologies – 2020. – С. 1-7.

18. Bhandari, G.P. Machine learning based software fault prediction utilizing source code metrics [Текст] / G.P. Bhandari, R. Gupta // IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), Kathmandu (Nepal) – 2018 – С. 1-6.
19. Шитиков, В.К. Классификация, регрессия, алгоритмы Data Mining с использованием R [Текст] / В.К. Шитиков, С.Э. Мастицкий // Тольятти, 2017.
20. Федотов, Д.В. О решении задачи классификации методом опорных векторов [Текст] / Д.В. Федотов // Математические методы моделирования, управления и анализа данных – 2013 – С. 1-3.
21. Кафтанников, И.Л. Особенности применения деревьев решений в задачах классификации [Текст] / И.Л. Кафтанников, А.В. Парасич // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника» – 2015. – С. 26-32.
22. Amazon Web Services. Что такое бустинг? [Электронный ресурс]: Режим доступа URL: https://aws.amazon.com/ru/what-is/boosting/ (Дата обращения 23.11.2022).
23. Дудченко, П.В. Метрики оценки классификаторов в задачах медицинской диагностики [Текст] / П.В. Дудченко // Молодежь и современные информационные технологии: сборник трудов XVI Международной научно-практической конференции студентов, аспирантов и молодых ученых – 2019. – С. 164-165.
24. NASA. Promise software engineering repository [Электронный ресурс]: Режим доступа URL: http://promise.site.uottawa.ca/SERepository/datasets-page.html (Дата обращения: 29.11.2022).
25. Python 3.11.3 documentation [Электронный ресурс]. Режим доступа URL: https://docs.python.org/3/ (Дата обращения 20.05.2023).
26. Scikit-learn. Machine Learning in Python [Электронный ресурс]. Режим доступа URL: https://scikit-learn.org/stable/ (Дата обращения 20.05.2023).
27. XGBoost [Электронный ресурс]. Режим доступа URL: https://xgboost.readthedocs.io/en/stable/ (Дата обращения 20.05.2023).

28. Qt for Python [Электронный ресурс]. Режим доступа URL: https://doc.qt.io/qtforpython/ (Дата обращения 20.05.2023).
29. Qt Designer Manual [Электронный ресурс]. Режим доступа URL: https://doc.qt.io/qt-5/qtdesigner-manual.html (Дата обращения: 20.05.2023).