Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)

Обнаружение дефектов программного обеспечения с использованием алгоритмов машинного обучения

Студент: Климов Илья Сергеевич, ИУ7-82Б
Научный руководитель: Вишневская Татьяна Ивановна, доцент кафедры ИУ7, к.ф.-м.н.

Москва, 2023

Актуальность

- Одна из актуальных проблем разработки программного обеспечения (ПО) – наличие дефектов.
- Затраты на выявление и устранение дефектов могут составлять до 80% от общей стоимости ПО.
- Существуют различные технологии для определения дефектов. В данной работе предлагается использовать методы машинного обучения.

Цель и задачи

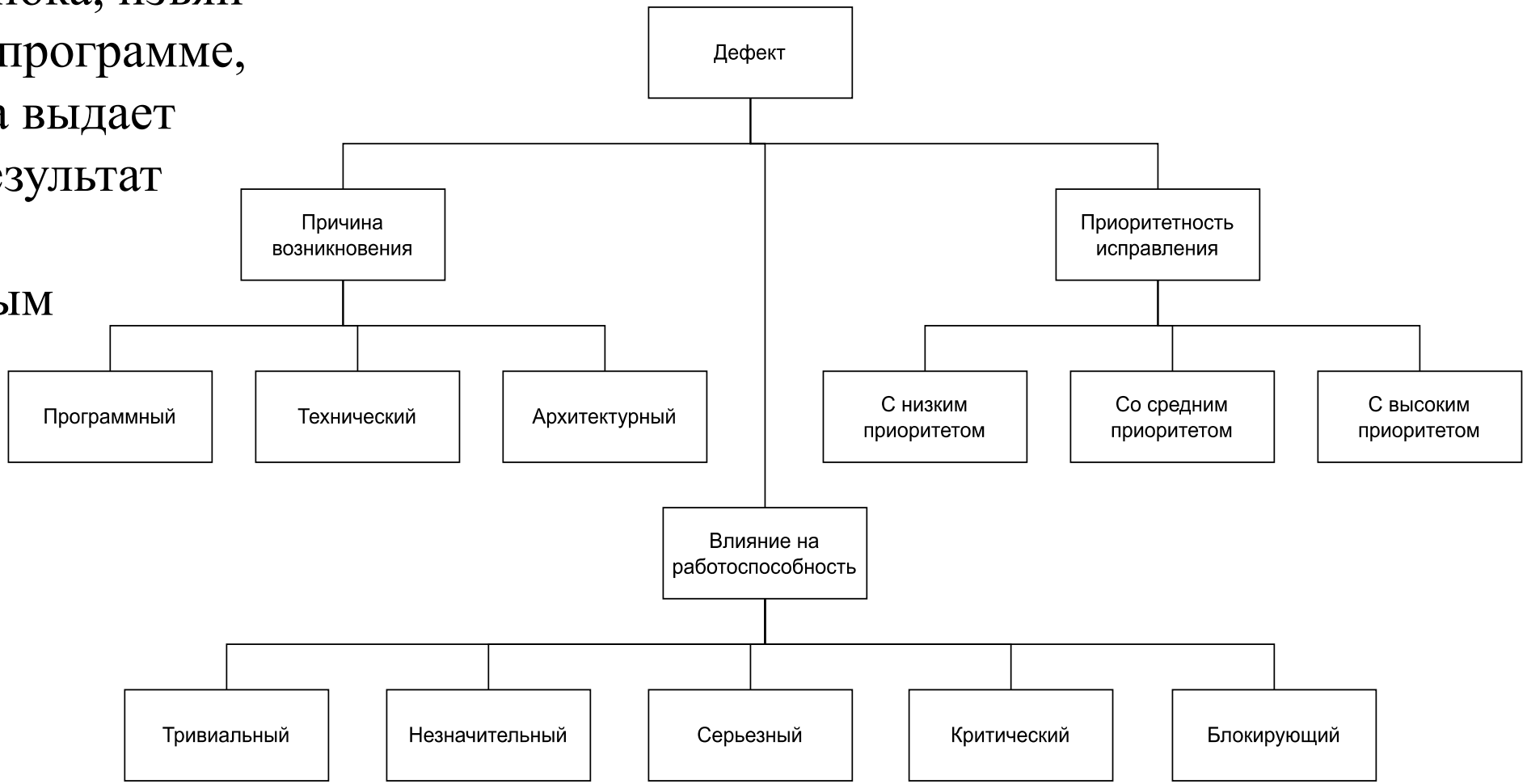
Цель: разработка и программная реализация метода обнаружения дефектов ПО с использованием алгоритмов машинного обучения.

Задачи:

1) проанализировать и сравнить существующие методы машинного обучения для обнаружения дефектов ПО;
2) разработать метод обнаружения дефектов ПО с применением ансамбля деревьев решений (градиентного бустинга);
3) разработать ПО, реализующее метод обнаружения дефектов ПО;
4) провести исследование эффективности разработанного метода и сравнение его с существующими реализациями.

Дефекты разрабатываемого ПО

Дефект ПО – ошибка, изъян в компьютерной программе, из-за которой она выдает неправильный результат или ведет себя непреднамеренным образом.



Сравнительная таблица результатов работы методов машинного обучения

Метрики / Методы	**Accuracy (точность)**	**Precision (точность)**	**Recall (полнота)**	**F-measure (F-мера)**
Наивный байесовский классификатор	0.795	0.845	0.803	0.849
Метод опорных векторов	0.841	0.901	0.879	0.902
Дерево решений	0.823	0.845	0.878	0.889
Случайный лес	0.845	0.859	0.863	0.890
Градиентный бустинг	**0.847**	**0.903**	**0.883**	**0.903**
Адаптивный бустинг	0.835	0.858	0.861	0.889

Метрики для оценки точности алгоритмов

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

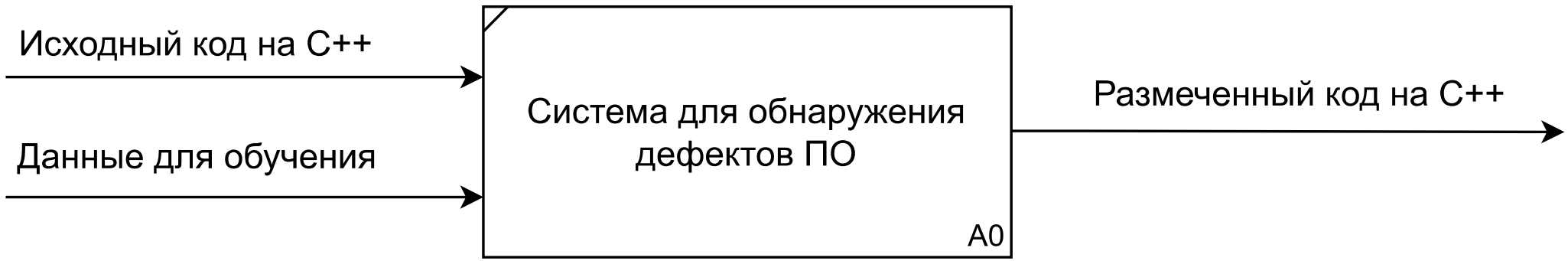
$$\text{recall} = \frac{TP}{TP + FN}$$

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

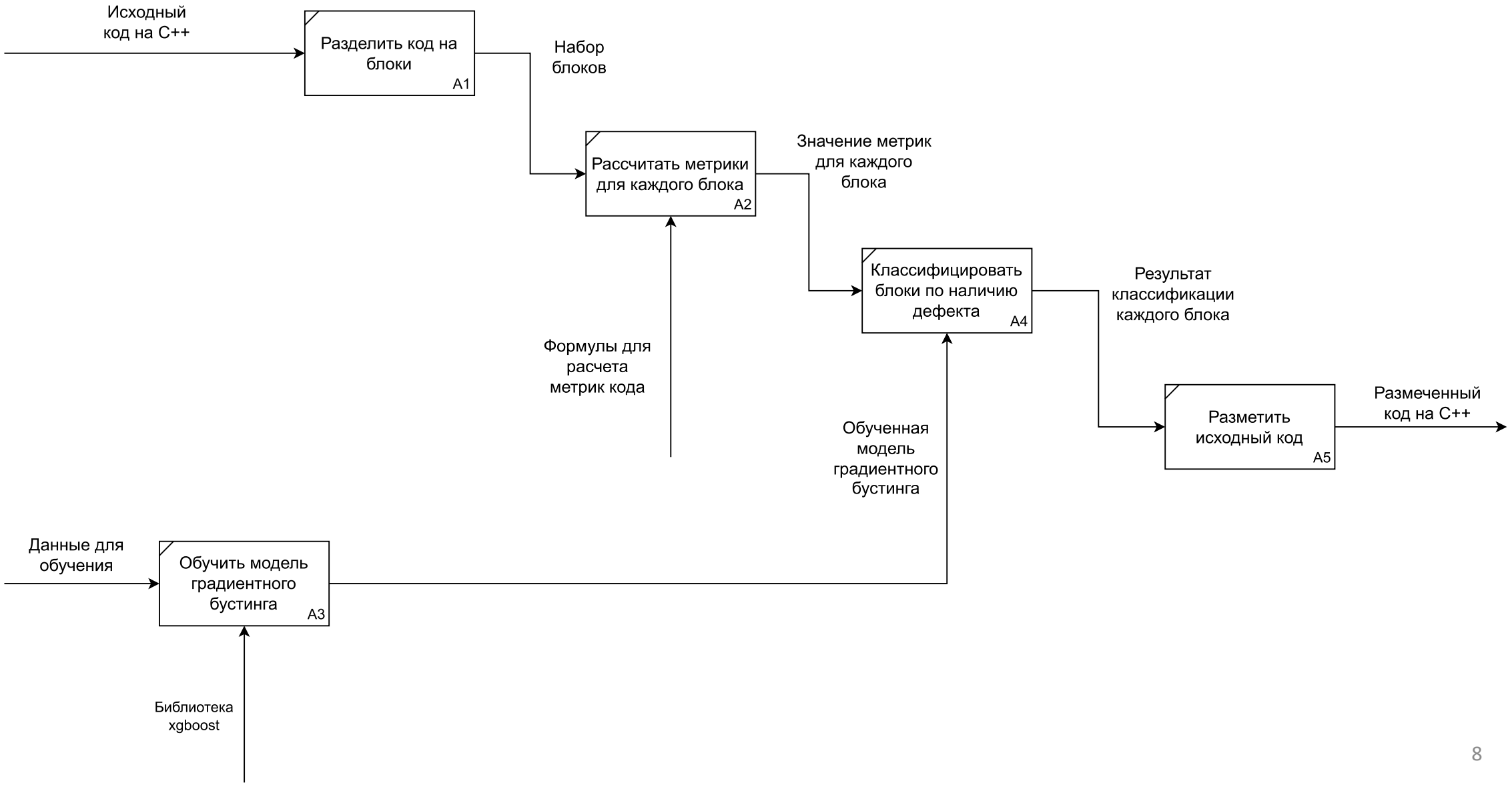
$$F - \text{measure} = \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

TP – объекты, которые были верно классифицированы как положительные;
TN – объекты, которые были верно классифицированы как отрицательные;
FP – объекты, которые были ложно классифицированы как положительные;
FN – объекты, которые были ложно классифицированы как отрицательные.

Формализованная постановка задачи



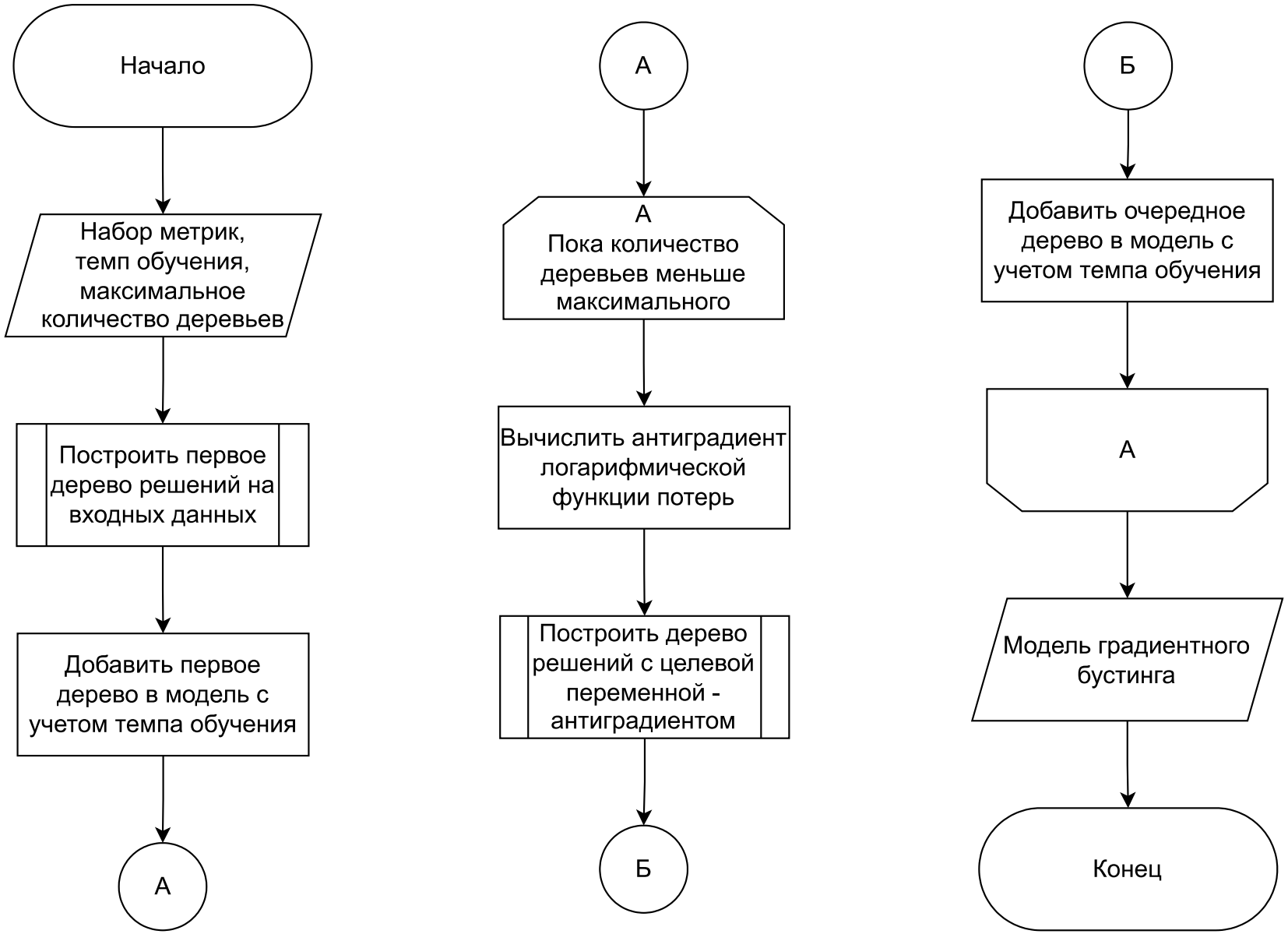
Метод обнаружения дефектов ПО



Метрики для оценки кода

- Количество строк кода Маккейба
- Цикломатическая сложность Маккейба
- Количество операторов и операндов Холстеда
- Объем Холстеда
- Сложность Холстеда
- Интеллект Холстеда
- Усилия Холстеда
- Количество предполагаемых ошибок Холстеда
- Оценка времени Холстеда
- Количество строк кода Холстеда
- Количество строк комментариев Холстеда
- Количество пустых строк Холстеда
- Количество смешанных строк Холстеда
- Количество уникальных операторов
- Количество уникальных операндов
- Общее количество операторов
- Общее количество операндов

Обучение модели градиентного бустинга



Обучение модели градиентного бустинга

Логарифмическая функция потерь

$$\mathcal{L}(y, p) = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p))$$

Антиградиент

$$g_i^k = -\frac{a_k(x_i) - y_i}{a_k(x_i)(a_k(x_i) - 1)}$$

y – действительное значение целевой переменной;

p – предсказанное значение целевой переменной;

y_i – действительное значение целевой переменной для i-го объекта;

$a_k(x_i)$ – предсказанное значение целевой переменной для i-го объекта;

x_i – значения метрик для i-го объекта.

Объект – набор из 17 значений метрик, посчитанных для определенного кода.
Целевая переменная – переменная, показывающая вероятность наличия дефекта для объекта.

Результат обучения модели градиентного бустинга

$$a_k(x) = \eta b_1(x, y_1) + \eta b_2(x, y_2) + \cdots + \eta b_k(x, y_k)$$

k – количество деревьев, участвующих при построении модели;

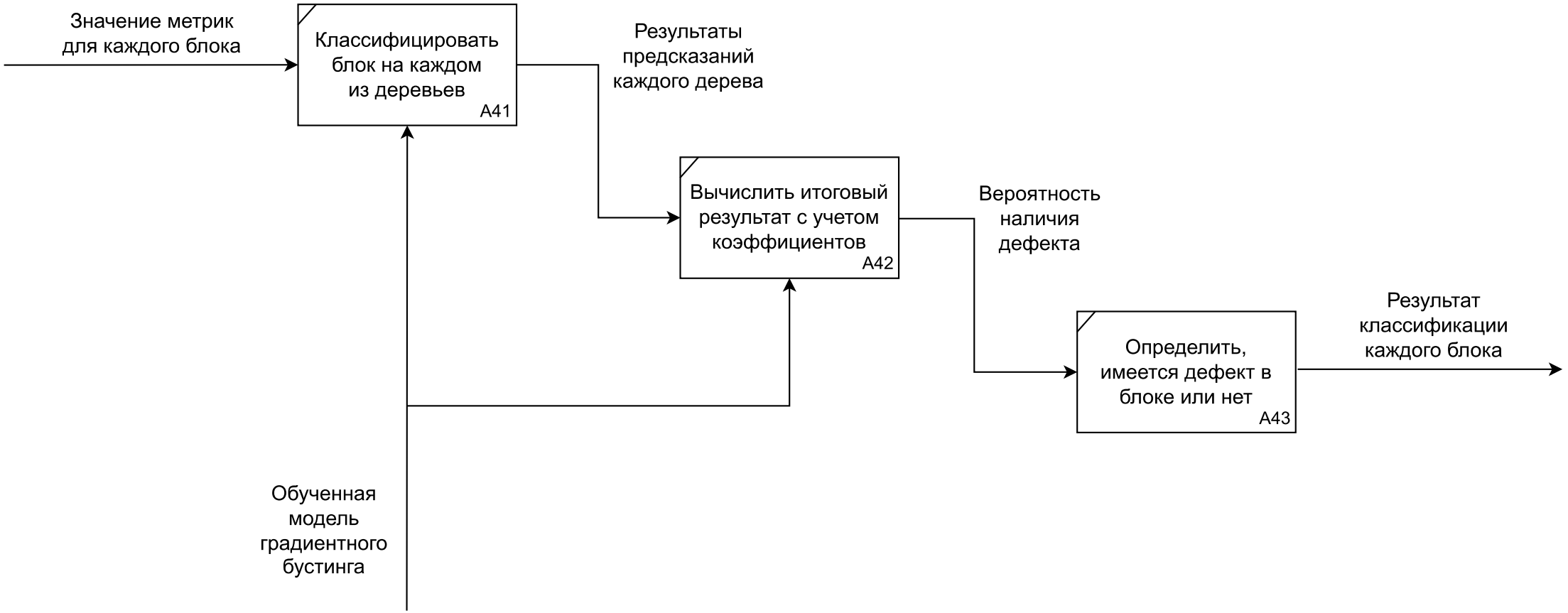
η – темп обучения, $\eta \in (0, 1]$;

$b_j\left(x, y_j\right)$ – результат классификации j-го дерева, $j \in \overline{1; k}$;

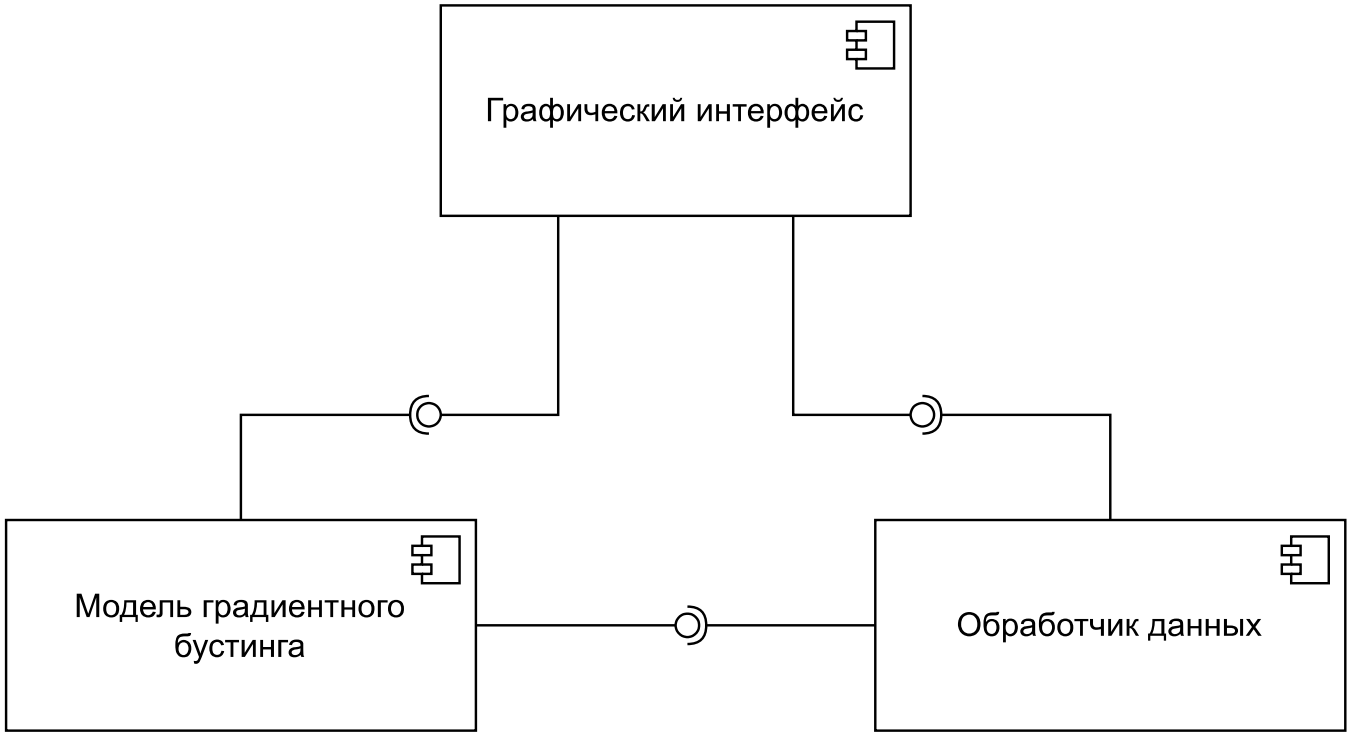
x – значения метрик для каждого объекта;

y_j – значения целевой переменной для каждого объекта.

Классификация кода по наличию дефекта



Программная реализация разработанного метода



Язык программирования – Python.

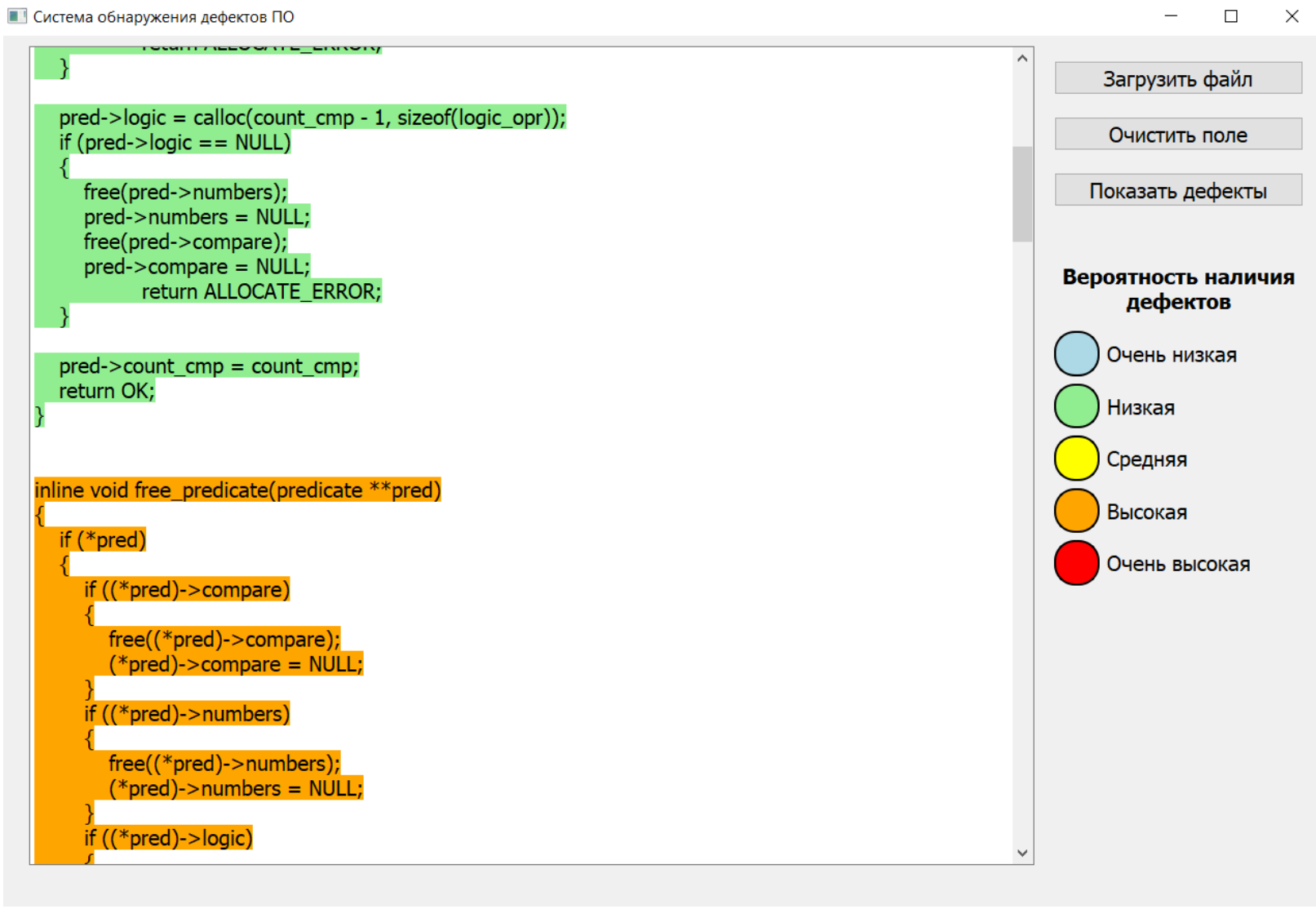
Библиотека для предобработки данных – scikit-learn.

Библиотека для обучения модели методом градиетного бустинга – XGBoost.

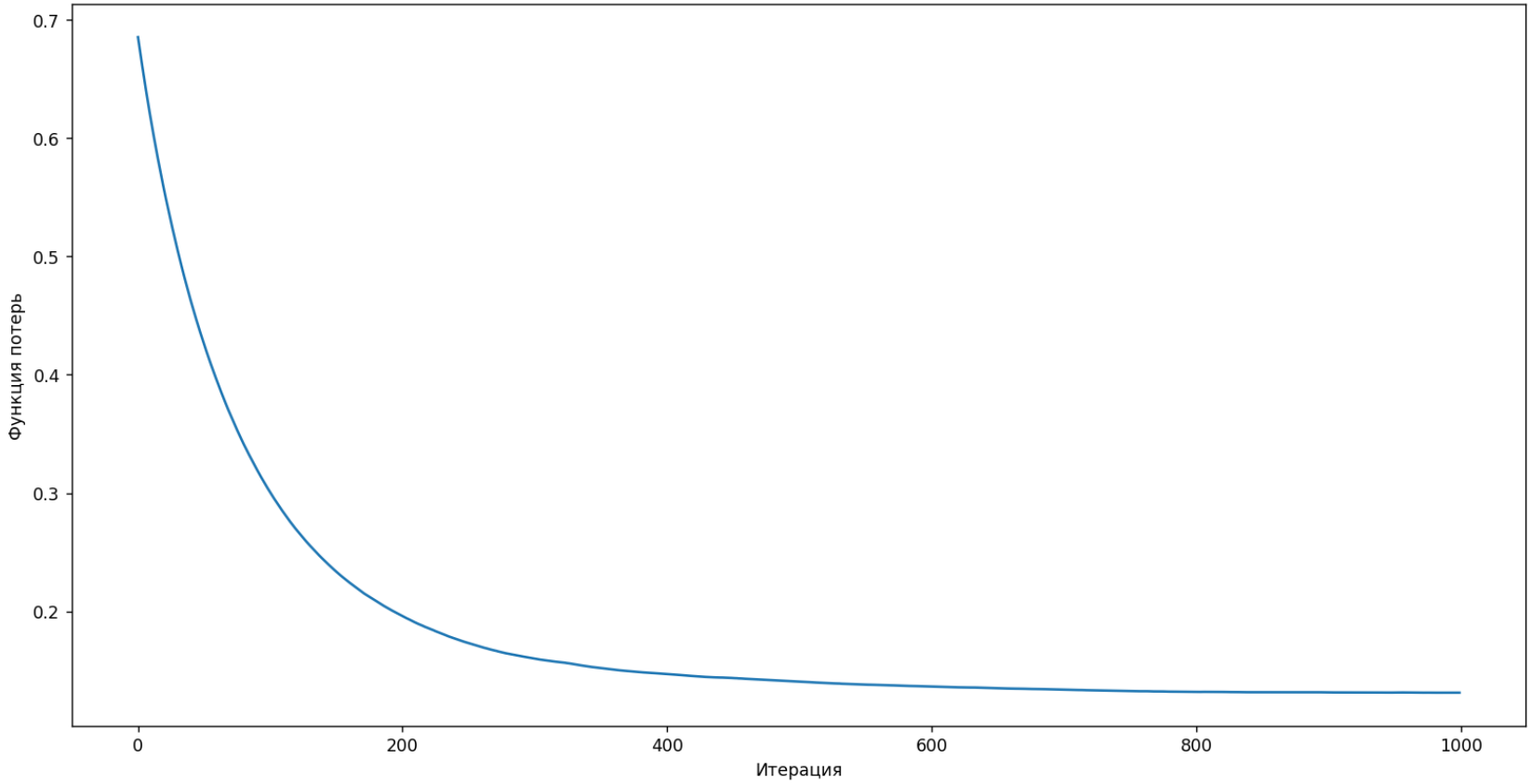
Фреймворк для разработки приложения – PyQt5.

Среда разработки – PyCharm.

Разработанная система обнаружения дефектов ПО



Анализ функции потерь

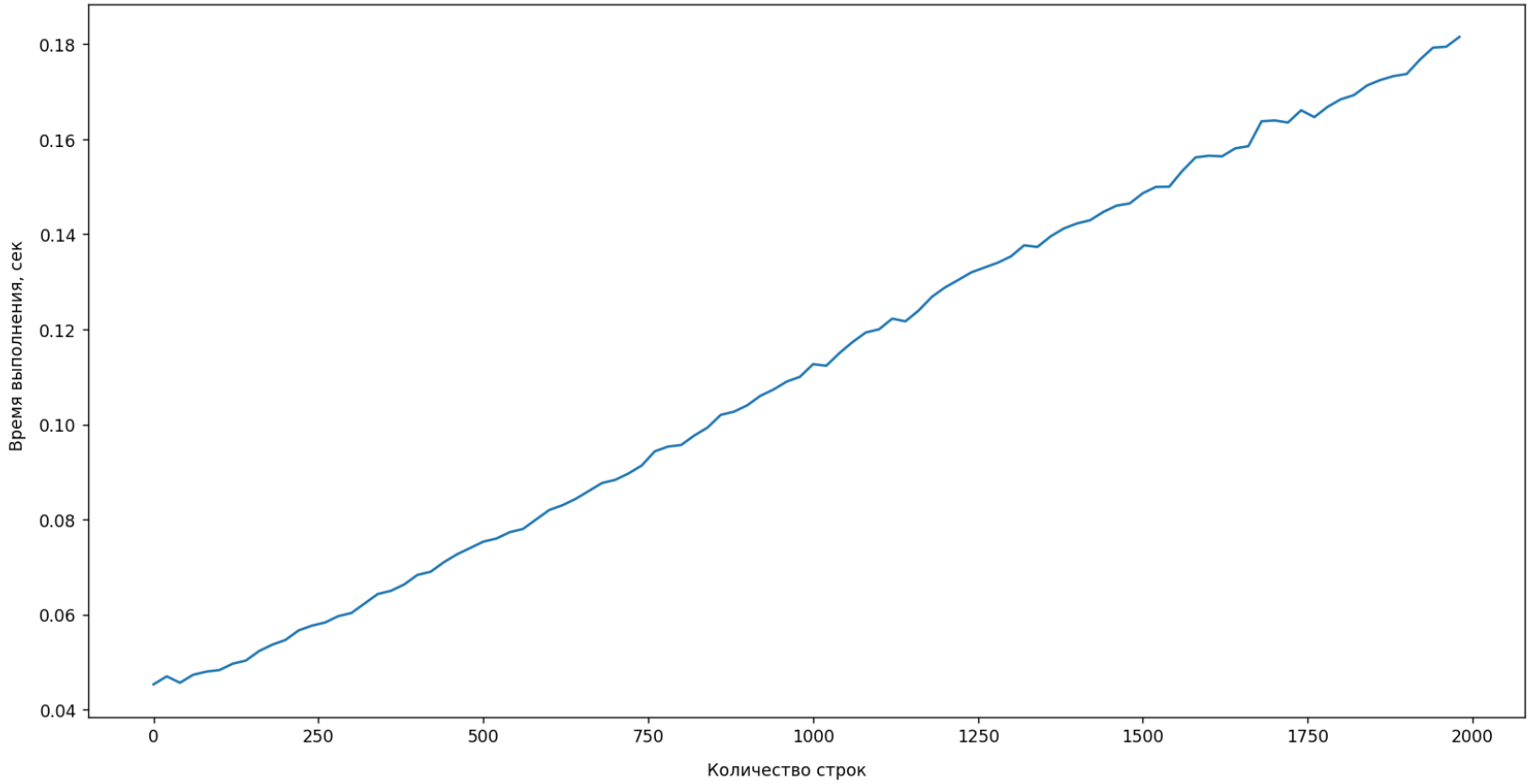


Обучение проходит верно. Функция потерь имеет гиперболическую форму, к концу обучения изменения становятся незначительными.

Сравнение точности разработанного метода

Метрики / Методы	**Accuracy (точность)**	**Precision (точность)**	**Recall (полнота)**	**F-measure (F-мера)**
Наивный байесовский классификатор	0.795	0.845	0.803	0.849
Метод опорных векторов	0.841	0.901	0.879	0.902
Дерево решений	0.823	0.845	0.878	0.889
Случайный лес	0.845	0.859	0.863	0.890
Градиентный бустинг	0.847	**0.903**	0.883	**0.903**
Адаптивный бустинг	0.835	0.858	0.861	0.889
Разработанный метод	**0.941**	0.818	**0.936**	0.873

Оценка времени выполнения программы



Зависимость между объемом входных данных и временем выполнения программы прямая. Обработка максимального количества строк занимает меньше 200 мс.

Заключение

В результате выполнения работы достигнута цель и решены все поставленные задачи:

1) произведены анализ и сравнение существующих методов машинного обучения для обнаружения дефектов ПО;
2) разработан метод обнаружения дефектов ПО с применением ансамбля деревьев решений (градиентного бустинга);
3) разработано ПО, реализующее метод обнаружения дефектов ПО;
4) проведено исследование эффективности разработанного метода и сравнение его с существующими реализациями.