УДК 004.912

С.Г. Букина, С.С. Харченко

Формирование набора признаков и сравнительный анализ алгоритмов классификации для выявления искусственно сгенерированного исходного кода

Представлен комплексный подход к формированию признакового пространства для решения задачи выявления искусственно сгенерированного исходного кода на языке программирования Python. Был разработан класс Algorithmic_Analyzer для извлечения 27 признаков, разделенных на четыре группы: базовые метрики кода, структурные характеристики, ключевые слова, библиотеки. Дополнительно извлекаются лексические паттерны с использованием словесных n-грамм. По результатам проведенных экспериментов с использованием классических алгоритмов машинного обучения установлено, что структурные характеристики демонстрируют более высокую значимость в сравнении с лексическими особенностями. Выявлены наиболее информативные признаки для решения представленной в работе задачи. Наилучший результат показала модель XGBClassifier (F1_macro 0,90 в среднем).

Ключевые слова: машинное обучение, исходный код, языковые модели, анализ признаков, классификация кода.
DOI: 10.21293/1818-0442-2025-28-4-121-126

Современные генеративные языковые модели активно используются для автоматической генерации исходного кода, что создает вызовы в области академической честности, безопасности программного обеспечения и защиты интеллектуальной собственности [1–4]. Наряду с этим развитие методов детекции искусственно сгенерированного исходного кода ограничено недостатком качественных наборов данных. В работе [5], посвященной вопросу наборов данных, предлагается идея развития в сторону разработки наборов под конкретную тематическую направленность и языки программирования.

Научная новизна данной работы и вклад в развитие выбранной научной тематики заключаются в разработке и валидации комбинированного подхода к формированию признакового пространства для определения искусственно сгенерированного исходного кода. В отличие от других исследований, фокусирующихся на нейросетевых структурах и способности достижения моделью более высоких значений по метрикам, в данной работе эмпирически выявлены и семантически интерпретированы паттерны, разделяющие сгенерированный и написанный людьми исходный код.

Исходные данные

В процессе работы по данной теме был сформирован авторский набор данных, включающий 4238 образцов исходного кода на языке программирования Python: 2238 решений написаны людьми, отобраны из набора APPS [6] и 2000 решений сгенерированы четырьмя языковыми моделями по 500 решений на каждую (GPT-4o-mini, Yandex GPT-4 PRO RC, Nvidia/Llama 3.1 Nemotron 70B Instruct HF, Qwen 2.5 72B Instruct). Решения написаны к 500 задачам из набора APPS, в которых охватывается три уровня сложности задач – начальный, соревнование, собеседование. Каждый образец содержит не только исходный код, но и структурированные метаданные, включая уровень сложности задачи и идентификатор источника решения. Предлагаемый набор данных предназначен для обучения и тестирования моделей машинного обучения.

Для валидации разработанного набора данных был проведен комплекс экспериментов. Выбор способа обработки данных и алгоритмов машинного обучения для решения задачи классификации исходного кода осуществлялся с учетом специфики данных и современных тенденций в данной научной тематике [7–14].

Формирование наборов признаков

Создание признакового пространства осуществлялось с использованием комбинированного подхода, учитывающего различные аспекты исходного кода.

Для обеспечения воспроизводимости и модульности реализации был разработан класс Algorithmic_Analyzer, унаследованный от BaseEstimator и TransformerMixin библиотеки scikit-learn. Класс инкапсулирует логику извлечения 27 признаков, сгруппированных в четыре категории: базовые метрики кода, структурные характеристики, анализ ключевых слов и импортируемых библиотек.

В рамках анализа ключевых слов рассматривалась частотность использования 9 распространенных элементов языка программирования Python, отражающих различные аспекты – от определения классов до обработки исключений и асинхронных операций.

Набор библиотек формировался на основе бинарных индикаторов использования 6 библиотек (sys, math, re, random, collections, itertools), выбор которых обусловлен их распространенностью в алгоритмических задачах по программированию.

В табл. 1 представлены базовые метрики исходного кода. В табл. 2 – структурные характеристики исходного кода.

Помимо класса, применяется текстовое представление кода с использованием TF-IDF векторзации с диапазоном словесных n-gram (1,2), что позволяет учитывать лексические паттерны.

Таблица 1

Базовые метрики исходного кода

Наименование	Описание признака
num_lines	Общее количество строк кода
Num_strings	Количество строковых литералов
Snake_case	Частота использования именований в стиле snake_case
Num_loops	Количество циклов
Num_conditionals	Количество условных конструкций
Num_lambda	Количество лямбда-функций
Avg_line_length	Средняя длина строки кода
Code_density	Плотность кода (отношение количества строк к общему объему)

Таблица 2

Структурные характеристики

Наименование	Описание признака
Recursion	Индикатор наличия рекурсивных вызовов
Has_dp	Индикатор использования динамического программирования
Num_sorting	Индикатор использования алгоритмов сортировки

Выбор диапазона осуществлялся с упором на работы по данной научной тематике, а также нацеленностью на компромисс между информативностью и разреженностью данных – униграммы представляют отдельные лексемы, такие как, например, наименование переменных, ключевые слова, в то время как биграммы учитывают пары токенов, отражающих шаблоны использования языковых конструкций. Для обработки текстовой составляющей использовались настройки токенизации, учитывающие особенности исходного кода, включая фильтрацию стоп-слов.

Группы признаков объединяются посредством FeatureUnion, что позволяет независимо извлекать структурные и текстовые характеристики с последующим формированием единого признакового пространства.

Планирование экспериментов

В исследовании была применена стратегия сравнения разнородных подходов на конкретной задаче, включающая как классические линейные, так и ансамблевые алгоритмы. Логистическая регрессия (LogisticRegression) была выбрана в качестве базового линейного классификатора, обладающего преимуществами в интерпретируемости результатов и эффективностью работы с высокоразмерными данными, характерными для текстовых и структурных признаков исходного кода.

Метод опорных векторов (SVC) с линейным ядром применялся для оценки степени линейной разделимости классов в сформированном признаковом пространстве, а также как эталонный метод со строгим теоретическим обоснованием. Со стороны ансамблевых методов были выбраны случайный лес (RandomForestClassifier) и градиентный бустинг (XGBClassifier) как наиболее представительные алгоритмы для задач классификации со сложными нелинейными зависимостями. RandomForestClassifier демонстрирует устойчивость к шуму и переобучению за счет бэггинга, в то время как XGBClassifier известен высокой предсказательной способностью благодаря последовательной оптимизации ансамбля. Таким образом, выбор моделей обусловлен потребностью в интерпретации результатов, в отличие от нейросетевых моделей, линейные позволяют оценить вклад каждого признака через весовые коэффициенты, а ансамблевые предоставляют метрики важности признаков.

Оптимизация гиперпараметров для каждого алгоритма проводилась с использованием стратегии случайного поиска (RandomizedSearchCV) с 10-кратной стратифицированной кросс-валидацией. Данный подход обеспечивает надежную оценку обобщающей способности моделей на ограниченном объеме данных за счет многократного разбиения на обучающую и тестовую подвыборки при сохранении распределения классов в каждой итерации. Для комплексной оценки результатов использовался расширенный набор метрик, позволяющий всесторонне проанализировать качество классификации.

Помимо общей точности (Accuracy), рассчитывался F1-score (macro-averaged) для сбалансированной оценки на обоих классах. Детальный анализ матрицы ошибок позволяет идентифицировать природу ошибочных классификаций и определить, к каким типам ошибок склонна каждая из моделей. Для статистического обоснования значимости различий между моделями применялся парный критерий Уилкоксона, позволяющий сравнивать распределения метрик качества на одних и тех же фолдах кросс-валидации.

Статистическая значимость различий в качестве классификации между моделями оценивалась с использованием парного критерия Уилкоксона с поправкой Бонферрони для множественных сравнений. Важной составляющей частью общей оценки стал анализ важности признаков с последующей визуализацией результатов. Для каждой модели строились гистограммы распределения значений трех наиболее значимых признаков, что позволяло наглядно продемонстрировать различия в распределении этих характеристик между классом со сгенерированным кодом и классом с кодом, написанным людьми.

Данный комплексный подход к оценке не только позволяет объективно определить качество предложенного набора данных и эффективность различных методов классификации, но и выявить наиболее информативные признаки для задачи выявления искусственно сгенерированного исходного кода, что имеет ценность для дальнейших исследований в данной области.

На рис. 1 представлена диаграмма решения задачи определения искусственно сгенерированного исходного кода. На этапе предобработки последовательно выполняются очистка, токенизация, синтаксический анализ исходного кода с построением абстрактного синтаксического дерева (AST, Abstract Syntax Tree) для унификации представления кода и выделения структурных элементов, не зависимых от форматирования. После предобработки исходного набора данных выполняется извлечение двух групп признаков.

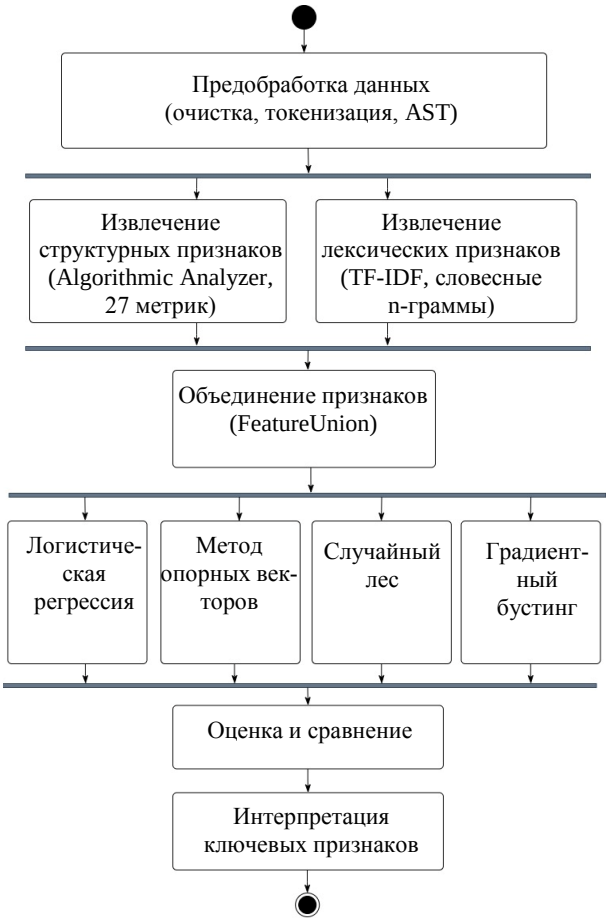

Рис. 1. Диаграмма решения

Объединенное признаковое пространство подается на вход четырем классификаторам. Для каждого классификатора выполняется 10-кратная стратифицированная кросс-валидация с оптимизацией гиперпараметров. Заключительный этап включает оценку качества по метрикам и интерпретацию наиболее значимых признаков.

Результаты экспериментов

Проведенные эксперименты с моделями демонстрируют следующее. Модель RandomForestClassifier достигла значения общей точности 92% и F1_macro 91% для обоих классов. Модель показывает лучшие результаты в определении кода, написанного людьми, в сравнении со сгенерированным кодом. Основная масса ошибок в матрице приходится на ложные срабатывания, когда написанный людьми код был ошибочно отнесен к сгенерированному, что указывает на необходимость дополнительной работы над признаками (рис. 2).

Анализ признаков выявил важность характеристик кода: snake_case имеет две выраженные группы с разными характеристиками (люди активно используют паттерн или избегают, у моделей меньше вариативности и больше шаблонности), kw_def демонстрирует более частое пользование генеративными моделями, kw_return показывает различные паттерны использования возвращаемых значений. Например, люди придерживаются или минималистичного подхода с небольшим числом возвращаемых значений, либо частое использование, в то время у моделей наблюдается шаблонность использования return (10–15 на образец), т.е. код со средними значениями в контексте данного признака, требует дополнительной проверки другими признаками. В случае текстовых признаков преобладают sys, input, list, что свидетельствует о различном использовании стандартных библиотек и шаблонных конструкций.

Модель XGBClassifier показывает значение общей точности 91% и F1_macro 91,5% для обоих классов. Матрица ошибок показывает симметричное распределение ошибок: 37 ошибок первого рода, 36 – второго (рис. 3). Иными словами, модель не имеет выраженного смещения в сторону одного из классов. Анализ признаков привел к аналогичным выводам с предыдущей моделью.

Модель SVC демонстрирует более низкие значения – общая точность 88%, F1_macro 88% для обоих классов. Большее число ошибок второго рода, когда сгенерированный код модель пропускает и принимает за написанный людьми (рис. 4).

Исходя из анализа признаков, выводы следующие: для сгенерированного кода (класс 1) более характерны высокая плотность кода, большее число строк, использование системных библиотек, а активное использование snake_case указывает на написанный людьми код (класс 0).

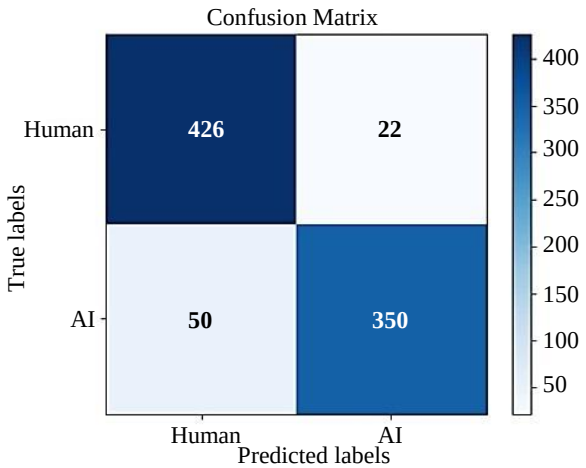

Рис. 2. Матрица ошибок, RandomForestClassifier

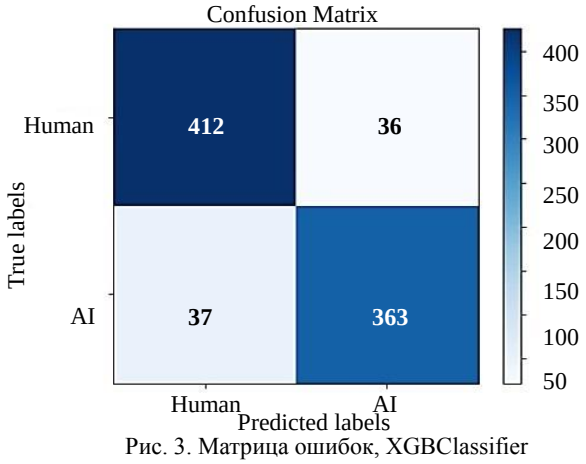

Рис. 3. Матрица ошибок, XGBClassifier

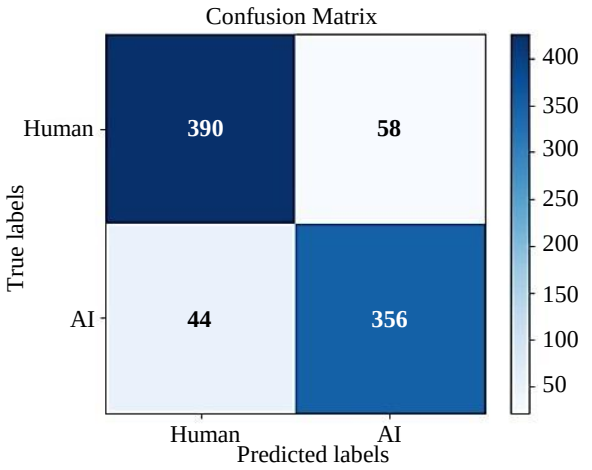

Рис. 4. Матрица ошибок, SVC

Модели придерживаются определенной шаблонности на примере числа строк, у образцов класса 0 широкий диапазон (25–150 строк), в то время как значение для образцов класса 1 сконцентрировано в области 50–100.

Логистическая регрессия достигла 89% общей точности и 89% F1_macro. Из матрицы ошибок можно сделать вывод о сбалансированном соотношении ошибок первого и второго рода (рис. 5). Анализ признаков показывает результаты, сходные с SVC.

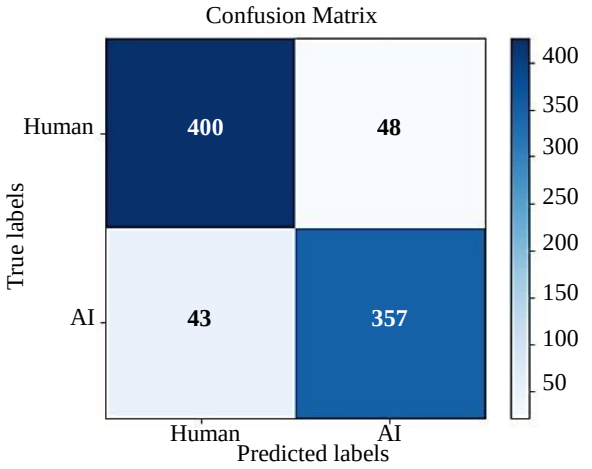

Рис. 5. Матрица ошибок, LogisticRegression

Обсуждение результатов

Проведенный анализ важности признаков потребовал не только количественной оценки вклада каждого признака в модель, но и их семантической интерпретации – перехода от формальных метрик к содержательному объяснению выявленных закономерностей. Под семантической интерпретацией понимается установление связи между математической значимостью признака и конкретными стилевыми паттернами программирования, которые этот признак кодирует. Например, признак snake_case формально отражает частоту использования имен переменных и функций в стиле snake_case. Однако его семантическая интерпретация раскрывает более глубокую закономерность: наблюдается бимодальное распределение для кода, написанного людьми, с пиками при минимальном и максимальном использовании, в то время как для сгенерированного кода распределение унимодально с концентрацией вокруг средних значений. Это отражает два характерных стиля программирования у людей: минималистичный и насыщенный функциями, тогда как модели придерживаются усредненного шаблона.

Активное использование библиотеки sys генеративными моделями связано с шаблонными решениями задач ввода-вывода, тогда как люди чаще используют input() для простых случаев. Более узкий диапазон значений количества строк (num_lines) у моделей (50–100 строк) против кода, написанного людьми (25–150 строк), свидетельствует о тенденции к генерации решений средней сложности, избегая как излишне кратких, так и переусложненных вариантов. Полученные результаты указывают на фундаментальное различие в подходе к композиции кода между человеком и генеративной языковой моделью, что открывает перспективы для разработки более точных методов определения сгенерированного кода.

Заключение

Проведенное исследование демонстрирует возможность решения задачи выявления искусственно сгенерированного кода методами машинного обучения. Полученные результаты позволяют провести детальный анализ ошибок классификации. Как представлено на матрицах ошибок (см. рис. 1–4), модели показывают различную склонность к ошибкам первого и второго рода. Для практического применения важно минимизировать ложно положительные срабатывания, когда код, написанный людьми, помечается как сгенерированный, поскольку это может привести к необоснованным обвинениям.

Напротив, ложно отрицательные срабатывания представляют меньший интерес в сценариях мониторинга, но критичны в условиях академической честности. Модель XGBClassifier с ее сбалансированностью ошибок представляет наибольший интерес относительно других рассмотренных моделей. Экспериментально подтверждена высокая значимость синтаксических и стилистических признаков.

Статистический анализ результатов показал, что модель XGBClassifier демонстрирует наилучшее среднее значение F1-macro, статистически значимо превосходя линейные модели (p_value = 0,0101). Несмотря на то, что различия с RandomForestClassifier не достигли статистической значимости (p_value = 0,3847), что свидетельствует о сопоставимой эффективности ансамблевых методов, XGBClassifier показывает стабильно высокие результаты по всем метрикам.

Перспективным направлением развития исследования является углубленный анализ семантических и структурных особенностей кода для выявления более сложных паттернов генерации кода. В работах [15, 16] авторы предлагают свои авторские наборы, также включающие фрагменты исходного кода на языке программирования Python, но с иными языковыми моделями и задачами, таким образом, можно проанализировать устойчивость выбранной

лучшей модели в данном исследовании к изменению источника данных, т.е. проводить эксперименты с другими наборами данных.

Литература

1. Illia L. Ethical implications of text generation in the age of artificial intelligence / L. Illia, E. Colleoni, S. Zyglidopoulos // Business Ethics, the Environment & Responsibility. – 2023. – Vol. 32, No. 1. – P. 201–210.

2. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions / H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, R. Karri // 2022 IEEE Symposium on Security and Privacy (SP). – San Francisco, USA: IEEE, 2022. – P. 754–768.

3. The «Code» of Ethics: A Holistic Audit of AI Code Generators / W. Ma, Y. Song, M. Xue, S. Wen, Y. Xiang // IEEE Transactions on Dependable and Secure Computing – 2024. – Vol. 21, No. 5. – P. 4997–5013.

4. Yoo S.H. Security Analysis of Automated Code Generation: Structural Vulnerabilities in AI-Generated Code / S.H. Yoo, H.J. Kim // Technical Journal. – 2025. – Vol. 19, No. 4. – P. 560–574.

5. Букина С.Г. Набор данных для выявления искусственно сгенерированного исходного кода / С.Г. Букина, С.С. Харченко // Доклады ТУСУР. – 2025. – Т. 28, № 2. – С. 106–110.

6. Набор данных APPS на платформе Hugging Face [Электронный ресурс]. – URL: https://huggingface.co/datasets/codeparrot/apps (дата обращения: 03.08.2025).

7. Idialu O.J. Whodunit: Classifying Code as Human Authored or GPT-4 Generated – A case study on CodeChef problems / O.J. Idialu, N.S. Mathews, R. Maipradit, J.M. Atlee // Proceedings of the 21st International Conference on Mining Software Repositories. – Lisbon, Portugal: ACM, 2024. – P. 394–406.

8. Li K. Discriminating Human-authored from ChatGPT-Generated Code Via Discernable Feature Analysis / K. Li, S. Hong, C. Fu, Y. Zhang, M. Liu // IEEE 34th International Symposium on Software Reliability Engineering Workshops. – Florence, Italy: IEEE, 2023. – P. 120–127.

9. Hoq M. Detecting ChatGPT-Generated Code Submissions in a CS1 Course Using Machine Learning Models / M. Hoq, Y. Shi, J. Leinonen, D. Babalola // Proceedings of the 55th ACM Technical Symposium on Computer Science Education. – New York, United States: ACM, 2024. – P. 526–532.

10. Bukhari S.A. Issues in Detection of AI-Generated Source Code: Master's thesis. – University of Calgary, Calgary, Canada, 2024. – 102 p.

11. Sjoerd S. The Detection of AI Generated Coding Content: Information and Computing Science: Master's thesis. – Utrecht University, Utrecht, Netherlands, 2024. – 75 p.

12. Li J. Dynamic analysis of generated code for security vetting / J. Li, X. Wang, Y. Lin // Proceedings of the 2023 IEEE Symposium on Security and Privacy (S&P). – San Francisco, USA: IEEE, 2023. – P. 1234–1249.

13. An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We? / H. Suh, M. Tafreshipour, J. Li, A. Bhattiprolu, I. Ahmed // The 47th IEEE/ACM International Conference on Software Engineering (ICSE 2025). – Ottawa, ON, Canada: ACM, 2025. – P. 859– 871.

14. Gurioli A. Is This You, LLM? Recognizing AI-written Programs with Multilingual Code Stylometry / A. Gurioli, M. Gabbrielli, S. Zacchiroli // 2025 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). – Montréal, Québec, Canada: IEEE, 2025. – P. 394–405.

15. Demirok B. AIGCodeSet: A New Annotated Dataset for AI Generated Code Detection / B. Demirok, M. Kutlu // arXiv. – 2025. DOI: 10.48550/arXiv.2412.16594.

16. Demirok B. MultiAIGCD: A Comprehensive dataset for AI Generated Code Detection Covering Multiple Languages, Models,Prompts, and Scenarios / B. Demirok, M. Kutlu, S. Mergen // arXiv. – 2025. DOI: 10.48550/arXiv.2507.21693.

Букина София Германовна
Студентка каф. безопасности информационных систем (БИС) Томского государственного ун-та систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, д. 40, г. Томск, Россия, 634050
Тел.: +7-952-183-45-99
Эл. почта: sofiabukina1714@gmail.com

Харченко Сергей Сергеевич
Канд. техн. наук, доцент каф. БИС ТУСУРа
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7 (382-2) 70-15-29
Эл. почта: kss@fb.tusur.ru

Поступила в редакцию: 22.10.2025.
Принята к публикации: 22.03.2026.

Bukina S.G., Kharchenko S.S.

Feature Set Formation and Comparative Analysis of Classification Algorithms for AI Generated Code Detection

The paper presents a comprehensive approach to to constructing a feature space for detecting artificially generated Python source code. We developed the Algorithmic_Analyzer class to extract 27 features categorized into four groups: basic code metrics, structural patterns, keywords, and libraries. Additionally, lexical patterns are captured using word n-grams. Experiments using classical machine learning algorithms demonstrate that structural characteristics exhibit significantly higher significance than lexical features. The study identifies the most informative features for artificial code detection and establishes that the XGBClassifier model achieves the best performance, with an average F1_macro score of 0.90.

Keywords: machine learning, source code, language models, feature analysis, code classification.
DOI: 10.21293/1818-0442-2025-28-4-121-126

References

1. Illia Laura, Colleoni Elanor, Zyglidopoulos Stelios Ethical implications of text generation in the age of artificial intelligence. Business Ethics, the Environment & Responsibility, 2023, vol. 32, no. 1, pp. 201–210.

2. Pearce H., Ahmad B., Tan B., Dolan-Gavitt B., Karri R. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. 2022 IEEE Symposium on Security and Privacy (SP), 2022, pp. 754–768.

3. Ma W., Song Y., Xue M., Wen S., Xiang Y. The «Code» of Ethics: A Holistic Audit of AI Code Generators. IEEE Transactions on Dependable and Secure Computing, 2024, Vol. 21, No. 5, pp. 4997–5013.

4. Yoo S.H., Kim H.J. Security Analysis of Automated Code Generation: Structural Vulnerabilities in AI-Generated Code, 2025, Vol. 19, No. 1, 560-574 pp.

5. Bukina S.G., Kharchenko S.S. [Dataset for Detecting Artificially Generated Source Code]. Reports of TUSUR (Tomsk), 2025, T.28, №2, 106-110 pp.

6. Dataset APPS on Hugging Face. Available at: https://huggingface.co/datasets/codeparrot/apps (Accessed: 3 August 2025).

7. Idialu O.J., Mathews N.S., Maipradit R., Atlee J.M. Whodunit: Classifying Code as Human Authored or GPT-4 Generated – A case study on CodeChef problems. Proceedings of the 21st International Conference on Mining Software Repositories, Lisbon, Portugal, ACM, 2024, pp. 394–406.

8. Li K., Hong S., Fu C., Zhang Y., Liu M. Discriminating Human-authored from ChatGPT-Generated Code Via Discernable Feature Analysis. IEEE 34th International Symposium on Software Reliability Engineering Workshops, Florence, Italy, IEEE, 2023, pp. 120–127.

9. Hoq M., Shi Y., Leinonen J., Babalola D. Detecting ChatGPT-Generated Code Submissions in a CS1 Course Using Machine Learning Models. Proceedings of the 55th ACM Technical Symposium on Computer Science Education, New York, United States, ACM, 2024, pp. 526–532.

10. Bukhari S.A. Issues in Detection of AI-Generated Source Code: Master's thesis, University of Calgary, Calgary, Canada, 2024, 102 p.

11. Sjoerd S. The Detection of AI Generated Coding Content: Information and Computing Science: Master's thesis, Utrecht University, Utrecht, Netherlands, 2024, 75 p.

12. Li J., Wang X., Lin Y. Dynamic analysis of generated code for security vetting. Proceedings of the 2023 IEEE Symposium on Security and Privacy (S&P), San Francisco, USA, IEEE, 2023, pp. 1234–1249.

13. Suh H., Tafreshipour M., Li J., Bhattiprolu A., Ahmed I. An Empirical Study on Automatically Detecting AI-Generated Source Code: How Far Are We? The 47th IEEE/ACM International Conference on Software Engineering (ICSE 2025), Ottawa, ON, Canada, ACM, 2025, pp. 859–871.

14. Gurioli A., Gabbrielli M., Zacchiroli S. Is This You, LLM? Recognizing AI-written Programs with Multilingual Code Stylometry. 2025 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Montréal, Québec, Canada, IEEE, 2025, pp. 394–405.

15. Demirok B., Kutlu M. AIGCodeSet: A New Annotated Dataset for AI Generated Code Detection. *arXiv*, 2025, DOI: 10.48550/arXiv.2412.16594.

16. Demirok B., Kutlu M., Mergen S. MultiAIGCD: A Comprehensive dataset for AI Generated Code Detection Covering Multiple Languages, Models,Prompts, and Scenarios / B. Demirok. *arXiv*, 2025, DOI: 10.48550/arXiv.2507.21693.

Sofia S. Bukina
Student, Department of Information System Security (ISS),
Tomsk State University of Control Systems
and Radioelectronics (TUSUR)
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-952-183-45-99
Email: sofiabukina1714@gmail.com

Sergei S. Kharchenko
Candidate of Technical Sciences, Associate Professor,
Department of ISS, TUSUR
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7 (382-2) 70-15-29
Email: kss@fb.tusur.ru

Received: 22.10.2025.
Accepted: 22.03.2026.