УДК 004.415.2

Брайловский А.В. магистрант,

МИРЭА – Российский технологический университет

Москва, Россия

МЕТОДОЛОГИЯ ПРЕДОТВРАЩЕНИЯ УТЕЧКИ ДАННЫХ (DATA LEAKAGE) ПРИ КРОСС-ПРОЕКТНОЙ ВАЛИДАЦИИ ВРЕМЕННЫХ РЯДОВ В ЗАДАЧАХ ПРЕДИКТИВНОГО МОНИТОРИНГА OPEN-SOURCE ПО

Аннотация: *В статье решается проблема объективной оценки алгоритмов машинного обучения на нестационарных временных рядах open-source проектов. Доказано, что классическое случайное перемешивание и K-блочная кросс-валидация социотехнических панельных данных приводят к утечке данных (Data Leakage, идентификационная утечка) и фальсификации метрик обобщающей способности. Для решения проблемы предложен алгоритм кросс-проектной хронологической валидации «walk_forward_project_split». Он строго изолирует датасет по идентификаторам репозиториев, сохраняя их временную связность и предотвращая статистическую утечку при нормализации. Эмпирически подтверждена эффективность метода и обоснована адаптация метрик: из-за экстремального дисбаланса классов предложен отказ от Accuracy в пользу комбинации F1-score и средней абсолютной ошибки (MAE) для оценки калибровки вероятностей.*

Ключевые слова: *машинное обучение, временные ряды, утечка данных, кросс-валидация, рекуррентные нейросети, GitHub, DevSecOps.*

Abstract: *The article addresses the objective evaluation of machine learning algorithms on non-stationary time series of open-source projects. It proves that classical random shuffling and K-fold cross-validation of socio-technical panel data lead to future data leakage, identity leakage, and falsified generalization metrics. To resolve this, the author proposes the "walk_forward_project_split" algorithm for cross-project chronological validation. It strictly isolates the dataset*

by repository IDs, maintaining temporal connectivity and preventing statistical leakage during normalization. The method's effectiveness is empirically proven, justifying the rejection of the Accuracy metric due to extreme class imbalance in favor of combining the F1-score and Mean Absolute Error (MAE) for probability calibration.

Keywords: *machine learning, time series, data leakage, cross-validation, recurrent neural networks, GitHub, DevSecOps.*

Введение

Разработка высоконадежных систем предиктивного мониторинга жизненного цикла программного обеспечения с открытым исходным кодом является одной из наиболее приоритетных и сложных задач в современной парадигме безопасной разработки. Защита корпоративного контура от атак на цепочки поставок требует применения сложного математического аппарата для непрерывного анализа хронологических данных библиотек-зависимостей. Как было показано в предыдущих исследованиях, архитектура конвейерного сбора еженедельных метрик позволяет сформировать глубокий и репрезентативный исторический контекст эволюции любого публичного репозитория [1, с. 41]. Специфика сбора таких метрик детально описывает природу сырых данных и множество социотехнических аномалий, генерируемых, например, автоматизированными ботами [2, с. 53].

Интеграция утилит статического анализатора кода (например, cloc) с программными интерфейсами платформы GitHub (в частности, использование высокопроизводительных типизированных запросов GraphQL, пришедших на смену традиционным REST API [3, с. 10]) позволяет получать комплексный, гибридный социотехнический срез данных. Этот многомерный срез включает в себя как топологические и технические параметры (общий объем исходного кода в байтах total_blob_size, количество файлов, цикломатическая сложность), так и метрики, отражающие социальную динамику сообщества (скорость закрытия задач closed_issues_count, плотность активности в обсуждениях Pull Requests).

Для интеллектуальной обработки таких сложноструктурированных последовательностей наиболее целесообразно применение рекуррентных нейронных сетей архитектуры LSTM. Благодаря математическому механизму вентилей забывания и сохранения состояния, такие архитектуры обладают достаточной емкостью для выявления неочевидных, растянутых во времени скрытых паттернов стагнации кодовой базы и критического снижения социальной активности профильных разработчиков [6, с. 398].

Однако практическое внедрение предиктивных нейросетевых моделей в промышленный контур информационной безопасности сопряжено с фундаментальной методологической проблемой — достоверной и несмещенной оценкой их предсказательной способности на этапе обучения и валидации. В базовых курсах по машинному обучению и в реализациях стандартных библиотек по умолчанию применяется метод случайного сэмплирования датасетов при разделении общей выборки [4, с. 115]. В контексте работы с многомерными временными рядами такой подход является категорически недопустимым. Механистическое перемешивание неизбежно приводит к разрушению причинно-следственных связей и эффекту «утечки данных», что делает результаты кросс-валидации искусственно завышенными и абсолютно неприменимыми в реальных промышленных условиях непрерывного мониторинга.

Проблема утечки данных при валидации социотехнических метрик

Анализ агрегируемых с платформы GitHub данных демонстрирует их высокую степень нестационарности и внутренней связности. Каждая запись представляет собой многомерный вектор признаков X_t^p для конкретного программного проекта p в момент времени t (дискретный шаг — одна календарная неделя). Совокупность упорядоченных записей формирует временной ряд $S^p = \{X_1^p, X_2^p, \dots, X_T^p\}$. Данные множества проектов образуют сложную панельную структуру.

Применение классического алгоритма K-Fold разделения со случайным перемешиванием строк порождает два критических искажения, описанных в теории машинного обучения временных рядов [6, с. 385]:

1. **Хронологическая инверсия (Future Leak).** При случайном независимом сэмплировании вектор X_{t+k}^{p} (состояние проекта в обозримом будущем) с высокой вероятностью может попасть в обучающую выборку, в то время как вектор X_{t}^{p} (состояние этого же проекта в прошлом) окажется в тестовой [6, с. 385]. В результате LSTM-сеть начинает тривиально интерполировать известные значения вместо истинной экстраполяции.
2. **Идентификационная утечка (Identity Leak).** Данная алгоритмическая проблема возникает, когда различные фрагменты временного ряда одного и того же репозитория p распределяются между трейном и тестом. Глубокая сеть вместо выявления обобщенных признаков деградации начинает аппроксимировать константные характеристики конкретных библиотек.

Следует отметить, что стандартные методы разделения временных рядов (например, TimeSeriesSplit из библиотеки scikit-learn) решают проблему хронологической инверсии для одномерных рядов, однако они неприменимы «из коробки» к многопроектным панельным данным. При их использовании временная ось отсекается глобально, что неизбежно смешивает истории разных репозиториев, сохраняя идентификационную утечку.

Алгоритм кросс-проектной хронологической валидации

Для преодоления описанных методологических уязвимостей формализован кастомный алгоритм walk_forward_project_split. Его фундаментальное концептуальное отличие заключается в строгом изоляционном разделении всего датасета исключительно по уникальным идентификаторам проектов (Project ID), а не по отдельным временным срезам.

Пусть P — генеральное множество всех доступных репозиториев, $P = \{p_1, p_2, \dots, p_N\}$. Производится строго непересекающееся разбиение P на обучающее подмножество P_{train} и тестовое подмножество P_{test}:

$$P_{train} \cup P_{test} = P, \quad P_{train} \cap P_{test} = \emptyset$$

Для каждого проекта $p \in P_{train}$ независимо формируются хронологически упорядоченные входные последовательности (окна) фиксированной длины L с заданным шагом скольжения S (где $S \leq L$ для обеспечения возможности перекрытия). Окно перемещается строго вперед, сохраняя авторегрессионные свойства. Начальный временной индекс i-го окна $t_i = 1 + (i-1) \cdot S$. Скользящее окно примет вид:

$$W_i^p = \{X_{t_i}^p, X_{t_i+1}^p, \dots, X_{t_i+L-1}^p\}$$

Разметка формируется на основе бинарного состояния проекта на заданном горизонте прогнозирования H, отсчитываемом от конца окна $t_{end} = t_i + L - 1$:

$$Y_i = Status\left(X_{t_{end}+H}^p\right)$$

Функция $Status(X)$ бинарно классифицирует состояние репозитория. В контексте данного исследования терминальная стагнация (класс 1) фиксируется при выполнении одного из условий: официальный перевод репозитория в статус archived владельцем, либо падение количества коммитов, закрытых задач и обработанных Pull Requests до нуля на непрерывном интервале, превышающем 12 недель. Жизнеспособные проекты маркируются как класс 0.

Для обеспечения воспроизводимости ниже представлен псевдокод предложенного алгоритма:

Листинг — Алгоритм Walk-Forward Project Split (псевдокод)

```
Вход: множество проектов P, параметры окон L, S, H
Выход: множества тензоров X_train, Y_train, X_test, Y_test

1. P_train, P_test <- Разделить P случайным образом по Project ID (например, 80/20)
2. Инициализировать пустые списки X_train, Y_train, X_test, Y_test
```

3. Для каждого подмножества (Subset, X_out, Y_out) из [(P_train, X_train, Y_train), (P_test, X_test, Y_test)]:
4. Для каждого проекта p в Subset:
5. T <- длина временного ряда проекта p
6. Для i = 1, 1+S, 1+2S ... до (T - L - H + 1):
7. W_i <- извлечь срез временного ряда p от i до i + L - 1
8. Target <- вычислить Status() для проекта p на неделе (i + L - 1 + H)
9. Добавить W_i в X_out
10. Добавить Target в Y_out
11. Вернуть X_train, Y_train, X_test, Y_test

Для повышения надежности оценки метод легко расширяется до многократной кросс-проектной валидации (Cross-Project Validation). В этом случае множество P делится на K непересекающихся фолдов именно по идентификаторам, что позволяет оценить дисперсию работы модели на принципиально разных сегментах open-source сообщества.

Сравнительная визуальная архитектура потоков данных представлена на рисунке 1.

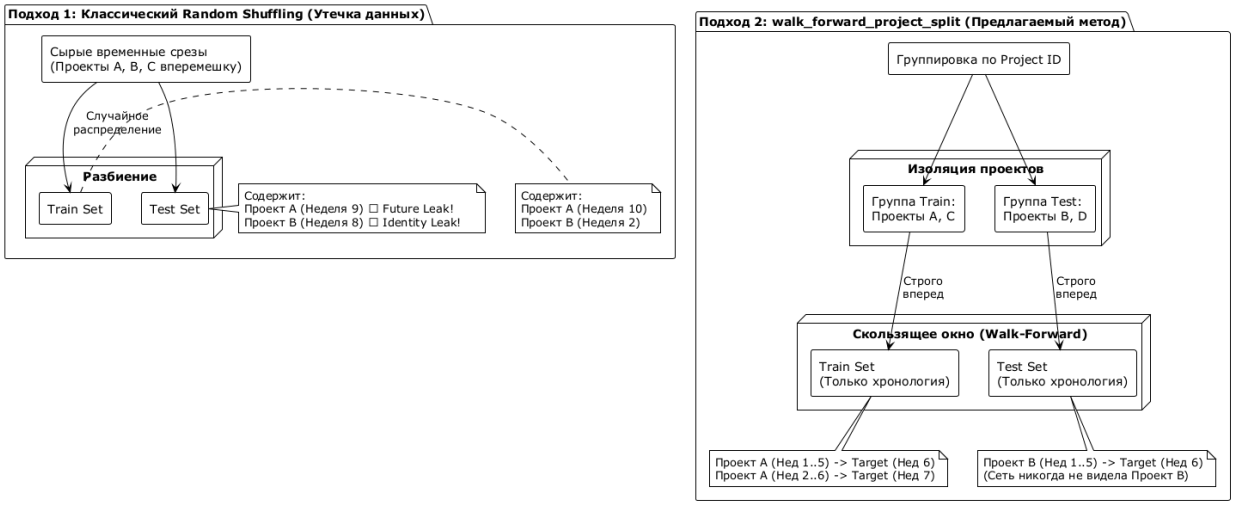

Рисунок 1 – Сравнение парадигм валидации: классическое перемешивание (слева) и хронологическая изоляция (справа)

Отдельно стоит отметить, что вычисление параметров робастного нормализатора (например, RobustScaler для фильтрации аномалий) на всем массиве данных *до* процедуры разделения неизбежно приводит к статистической утечке информации о глобальных экстремумах

распределения из будущего [8]. Поэтому параметры нормализатора вычисляются строго на X_{train} и применяются к X_{test} в режиме изолированной трансформации.

Процесс генерации скользящих окон приводит к значительному линейному росту объема обучающей выборки $O(N \cdot T)$, где N — количество проектов, T — длина ряда. Для обработки тензоров целесообразно применение принципов распределенных вычислительных систем [5, с. 92].

Метрики оценки в условиях стохастичности и экстремального дисбаланса

Внедрение строгого алгоритма хронологического сплитования объективно усложняет задачу аппроксимации для модели машинного обучения. Как следствие, нейросеть лишается возможности «заучивать» репозитории.

Дополнительным фундаментальным фактором является естественный экстремальный дисбаланс классов: количество жизнеспособных периодов (класс 0) в датасетах GitHub значительно превышает количество подтвержденных фактов деградации (класс 1).

Следствием самой природы этого дисбаланса является категорический отказ от использования метрики Accuracy (общая доля правильных ответов) на этапе оценки модели. Алгоритм, выучивший предсказание мажоритарного класса «0», математически может достигать Accuracy выше 95%, оставаясь абсолютно бесполезным инструментом предиктивной безопасности.

Основной упор в методологии тестирования должен быть сделан на Матрицу ошибок и метрику $F1 - score$:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Полнота (Recall) в контексте DevSecOps обладает наивысшим бизнес-весом, так как пропуск деградирующей зависимости несет прямую угрозу компрометации продукта. Для стимуляции полноты на этапе обучения

применяется взвешенная функция потерь с увеличенным штрафным коэффициентом для редкого класса стагнации [9].

Кроме того, для оценки калибровки вероятностных прогнозов целесообразно применять метрику Средней абсолютной ошибки (Mean Absolute Error — MAE) [7, с. 88]:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

MAE измеряет среднее абсолютное отклонение предсказанной вероятности от истинной метки. Если нейросеть относит проект к классу 1 с вероятностью $\hat{y}_i = 0.51$, а истинный класс $y_i = 1$, бинарная метрика зафиксирует идеальное математическое попадание по порогу 0.5. Однако MAE зафиксирует отклонение в 0.49, объективно отражая низкую достоверность (Confidence) классификатора.

Эмпирическая оценка эффективности предложенной методологии

Для подтверждения выдвинутых теоретических положений был проведен сравнительный эксперимент с использованием архитектуры LSTM. При классическом случайном K-Fold разбиении сэмплированных строк (Shuffle=True), модель быстро деградировала до эффекта переобучения за счет идентификационной утечки, достигая неправдоподобно высоких показателей ($F1 \approx 0.96$, $ROC - AUC \approx 0.99$).

При переключении конвейера валидации на алгоритм walk_forward_project_split, нейросеть тестировалась исключительно на абсолютно новых, ранее никогда не встречавшихся временных рядах проектов. В этих жестких условиях истинной экстраполяции метрики закономерно скорректировались ($F1 \approx 0.77$, $ROC - AUC \approx 0.88$). Полученные при хронологической изоляции результаты, хотя и ниже абсолютных значений «зашумленного» эксперимента, отражают реальную, верифицируемую обобщающую способность системы, пригодную для развертывания в production-среде.

Заключение

Проведенное исследование доказывает, что механистическое применение базовых алгоритмов случайного сэмплирования к хронологическим панельным данным ведет к критическому искажению результатов оценки. Игнорирование архитектуры временных рядов порождает риски фальшивой уверенности при внедрении ML-моделей в системы безопасности.

Формализованный алгоритм walk_forward_project_split гарантирует изоляцию многомерных признаковых пространств различных проектов и пресекает утечку данных. Использование описанной методологии с добавлением строгой изоляции нормализаторов, в комплексе с отказом от метрики Accuracy в пользу связки F1-score и MAE, формирует надежный базис для объективной оценки нейронных сетей. Это позволяет исследовательским командам стандартизировать процесс интеллектуального анализа кода и получать стабильно воспроизводимые метрики предиктивной безопасности.

Использованные источники:

1. Брайловский, А. В. Анализ факторов стабильности и прогнозирование жизненного цикла open-source проектов на основе данных платформы GitHub / А. В. Брайловский // «Парадигма»: научно-практический электронный журнал. – 2025. – № 6-1. – С. 38-46.

2. Брайловский, А. В. Методология сбора данных для анализа активности проектов с открытым исходным кодом на платформе GitHub / А. В. Брайловский // «Парадигма»: научно-практический электронный журнал. – 2025. – № 6-1. – С. 47-57.

3. Галигузова, Е. В. Язык запросов GraphQL как замена REST API / Е. В. Галигузова, Ю. Е. Илларионова // Символ науки: международный научный журнал. – 2023. – № 1-2. – С. 9-11.

4. Коротеев, М. В. Основы машинного обучения на Python: учебник для вузов / М. В. Коротеев. – Москва: Издательство Юрайт, 2025. – 432 с.

5. Радченко, Г. И. Распределенные вычислительные системы: учебное пособие / Г. И. Радченко. – Москва: Издательство Юрайт, 2022. – 174 с.

6. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле ; пер. с англ. А. А. Слинкина. – 2-е изд. – Санкт-Петербург: Питер, 2023. – 574 с.

7. Жерон, О. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow / О. Жерон ; пер. с англ. – 3-е изд. – Санкт-Петербург: Диалектика, 2024. – 1040 с.

8. Официальная документация Scikit-Learn: RobustScaler [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

9. Официальная документация PyTorch: Функция потерь BCEWithLogitsLoss [Электронный ресурс]. URL: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html.