УДК 004.9

ПОДХОДЫ К ОДНОКЛАССОВОЙ КЛАССИФИКАЦИИ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

Демидов Н.А.

Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА — Российский технологический университет», 119454, Российская Федерация, г. Москва, проспект Вернадского, 78, e-mail: nick.a.demidov@rambler.ru

Резюме

Цели. В статье рассмотрена задача выявления паттернов новизны в списке регулярных выражений. Цель работы – исследование подходов к выявлению паттернов новизны в списке регулярных выражений посредством алгоритмов одноклассовой классификации.

Методы. Для решения поставленной задачи предложено использовать подходы, основанные на алгоритмах одноклассовой классификации, таких как One-Class SVM и Isolation Forest. Для представления регулярных выражений в векторном виде предложено использовать модели двунаправленных предобученных трансформеров BERT и ModernBERT.

Результаты. Результаты экспериментальных исследований подтверждают целесообразность использования алгоритмов одноклассовой классификации для разработки классификаторов, реализующих выявление паттернов новизны в списке регулярных выражений. При этом наблюдается превосходство модели ModernBERT по отношению к модели BERT в смысле обеспечения высокого качества классификации при выявлении паттернов новизны в списке регулярных выражений.

Выводы. Рассмотренные подходы к одноклассовой классификации регулярных выражений могут быть рекомендованы к использованию для выявления паттернов новизны в списке регулярных выражений. При этом векторизация регулярных выражений, используемых при обучении и тестировании одноклассовых классификаторов, может быть выполнена на основе моделей двунаправленных предобученных трансформеров. Одноклассовые классификаторы регулярных выражений могут быть применены для проверки новых данных, в том числе – генерируемых, на наличие в них нормальных паттернов и паттернов новизны.

Ключевые слова: регулярное выражение, паттерн новизны, BERT, ModernBERT, одноклассовая классификация, One-Class SVM, Isolation Forest.

APPROACHES TO ONE-CLASS CLASSIFICATION OF REGULAR EXPRESSIONS

Demidov N.A.

Federal State Budgetary Educational Institution of Higher Education «MIREA - Russian Technological University», 119454, Russian Federation, Moscow, Vernadsky Avenue, 78, e-mail: nick.a.demidov@rambler.ru.

Abstract

Objectives. The article considers the problem of identifying novelty patterns in a list of regular expressions. The purpose of the work is to study approaches to identifying novelty patterns in a list of regular expressions using one-class classification algorithms.

Methods. To solve the problem, it is proposed to use approaches based on one-class classification algorithms, such as One-Class SVM and Isolation Forest. To represent regular expressions in vector form, it is proposed to use the models of bidirectional pre-trained transformers BERT and ModernBERT.

Results. The results of experimental studies confirm the feasibility of using one-class classification algorithms for developing classifiers that implement the identification of novelty patterns in a list of regular expressions. A herewith, the superiority of the ModernBERT model over the BERT model is observed in terms of ensuring high quality classification when identifying novelty patterns in a list of regular expressions.

Conclusions. The considered approaches to one-class classification of regular expressions can be recommended for use in identifying novelty patterns in a list of regular expressions. In this case, vectorization of regular expressions used in training and testing one-class classifiers can be performed based on models of bidirectional pre-trained transformers. One-class classifiers of regular expressions can be used to check new data, including generated data, for the presence of normal patterns and novelty patterns.

Keywords: regular expression, novelty pattern, BERT, ModernBERT, one-class classification, One-Class SVM, Isolation Forest.

Введение

В настоящее время регулярные выражения (РВ) активно используют в задачах поиска, сопоставления и манипулирования текстовыми шаблонами [1–5]. В частности, РВ применяют в программировании [1] и обработке данных [6], например, для проверки вводимых пользователем данных, таких как адреса электронной почты, номера телефонов или пароли, на соответствие определенным шаблонам. Также РВ обеспечивают в текстовых редакторах и поисковых системах расширенные операции поиска и замены [6–7], позволяя эффективно находить или изменять сложные шаблоны, и играют важную роль в процессе извлечения данных, давая возможность анализировать и извлекать структурированную информацию из неструктурированных источников данных, таких как научные статьи, веб-страницы и т.п. Кроме того, РВ используют в лексическом анализе в процессе компиляции языков программирования [1], где с их помощью определяются такие лексемы, как ключевые слова, операторы и идентификаторы.

Обычно на практике специалистам приходится решать задачи бинарной и многоклассовой классификации РВ. При этом считается, что РВ всех классов имеются в наличии в том или ином числе, то есть доступны и могут быть использованы для формирования обучающих и тестовых выборок данных с целью разработки с применением алгоритмов машинного и глубокого обучения искомых классификаторов РВ.

Более сложной является задача выявления паттернов новизны в списке имеющихся в наличии РВ. Под паттернами новизны, в частности, можно понимать, например, аномальные РВ (выбросы). РВ, которые логично считать паттернами новизны, скорее всего, появляются редко, поэтому практически невозможно собрать на их основе репрезентативную выборку для включения её объектов в обучающую и тестовую выборки с целью разработки требуемого классификатора. В связи с этим задача выявления паттернов новизны в списке имеющихся в наличии РВ может быть сведена к задаче так называемой одноклассовой классификации и, как следствие, к задаче разработки одноклассовых классификаторов.

Одноклассовые классификаторы представляют собой специализированные модели машинного обучения, разработанные для сценариев, в которых для обучения доступны данные только одного класса, что делает такие классификаторы особенно полезными для выявления новизны в данных [8]. В отличие от традиционных классификаторов, при разработке которых требуется использовать размеченные данные из нескольких классов, при разработке одноклассовых классификаторов выполняется изучение характеристик (признаков) «нормальных» объектов одного (целевого) класса и обучение классификаторов на их основе. Объекты, значения признаков которых имеют существенные отклонения от уже изученных значений одноименных признаков «нормальных» объектов, будут идентифицироваться такими классификаторами как аномалии, представляющие собой паттерны новизны.

Наиболее известными алгоритмами машинного обучения, позволяющими решать задачу одноклассовой классификации, являются одноклассовый SVM (One-Class Support Vector Machine, OCSVM) алгоритм [9–14], алгоритм изолирующего леса (Isolation Forest, IF) [15–19] и алгоритм оценки локального уровня выброса (Local Outlier Factor, LOF) [20, 21]. Классификаторы на основе этих алгоритмов широко применяются в таких задачах, как задача выявления мошенничества, задача обеспечения сетевой безопасности, задача обнаружения промышленных дефектов и т.п. В этих задачах аномалии встречаются редко, но их обнаружение является крайне важным.

Алгоритмы одноклассовой классификации могут быть успешно применены для задач анализа РВ. Во-первых, классификатор, обученный на «нормальных» РВ, можно будет использовать для выявления «аномальных» или «подозрительных» РВ. В результате с использованием такого классификатора можно будет обнаруживать вредоносные или «неэффективные» РВ, которые могут приводить, например, к DoS-атакам или другим проблемам с производительностью. Во-вторых, классификатор, обученный на «нормальных» РВ, можно будет использовать для тестирования вновь автоматически генерируемых РВ на предмет аномальности в случае, когда используется некоторый алгоритм генерации РВ с неявно заданными значениями характеристик (значениями признаков) и требуется отбрасывать «аномальные» РВ.

Отдельное внимание при разработке классификаторов, предполагающих работу с числовыми значениями признаков, описывающих объекты, должно уделяться выбору модели векторизации данных. Это особенно важно при работе с текстовыми данными, в частности, с РВ. В настоящее время для векторизации текстовых данных используются как классические подходы, реализующие, например, применение моделей TF-IDF (Term Frequency – Inverse Document Frequency, частота термина – обратная частота документа) [22] и BOW (Bag Of Words, мешок слов) [23], так и современные подходы, реализующие применение моделей векторизации на основе двунаправленных предобученных трансформеров, например, таких, как BERT (Bidirectional Encoder Representations from Transformers) [24] и ModernBERT [25].

Целью настоящей работы является исследование подходов к решению задачи выявления паттернов новизны в списке регулярных выражений.

Для достижения поставленной цели предлагается рассмотреть различные модели векторизации текстовых данных и подходы к выявлению паттернов новизны в текстовых данных в контексте работы с РВ.

1. Векторизация регулярных выражений

В настоящем исследовании для представления РВ в векторном виде предлагается рассмотреть классические модели, такие как TF-IDF и BOW, и современные модели, такие как BERT и ModernBERT.

Модель TF-IDF [22] реализует векторизацию документов в наборе документов (корпусе) посредством формирования вектора, каждый элемент которого основан на оценке важности слова в документе относительно набора документов (корпуса). При этом вес каждого слова пропорционален частоте его употребления в документе и обратно пропорционален частоте его употребления во всех документах набора документов (корпуса).

Для каждого слова t в документе d из набора документов D определяется значение меры

$$tf\text{-}idf(t,d,D) = tf(t,d) \cdot idf(t,D), \quad (1)$$

где $tf(t,d) = n_t / \sum_{i=1}^{I} n_i$; n_t – число вхождений слова t в документ d; I – число разных слов в документе d; $idf(t,D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}$; $|D|$ – число документов в наборе документов D; $|\{d_i \in D \mid t \in d_i\}|$ – число документов в наборе документов D, в которых встречается слово t (при $n_t \neq 0$).

Большее значение $tf\text{-}idf(t,d,D)$ имеют слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах.

Модель BoW [23] является одной из самых простых моделей векторизации документов в наборе документов (корпусе), реализуемой посредством формирования вектора, каждый элемент которого соответствует частоте употребления слова в документе. При этом порядок слов и их грамматическая структура игнорируются. Кроме того, векторы могут быть очень разреженными, особенно при работе с большими наборами документов (корпусами), что затрудняет обработку. В контексте работы с РВ под словами можно понимать отдельные символы.

Модель BERT (Bidirectional Encoder Representations from Transformers) [24] является моделью представления естественного языка, предложенной разработчиками компании Google в 2018 году. Модель BERT реализует двунаправленный подход к обработке и представлению текста, позволяющий улавливать контекст одновременно с двух сторон. Это достигается за счет архитектуры Transformer, состоящей только из кодировщика, которая использует механизмы самовнимания для оценки важности различных слов в предложении. Модель BERT была предварительно обучена на больших корпусах текста с помощью двух задач с использованием обучения без учителя, таких как Masked Language Modeling (MLM), в которой модель предсказывает маскированные слова в предложении, и Next Sentence Prediction (NSP), в которой модель определяет, следует ли одно предложение за другим. Такое предобучение позволяет использовать модель BERT для широкого спектра задач обработки естественного языка (Natural Language Processing, NLP), таких как ответы на вопросы, анализ настроения и классификация текстов. Длина контекста модели BERT составляет 512 токенов. При работе с трансформерами под токеном понимают базовую единицу текста, используемую для обработки и анализа данных. Токены могут быть словами, частями слов или даже отдельными символами, в зависимости от того, как работает токенизатор. Различные модели на основе модели BERT, хотя и характеризуются устаревшими архитектурами, небольшим размером контекста и ограниченным объемом обучающих данных, но до сих пор активно используются на практике. Модель BERT доступна к использованию в нескольких версиях, в частности, как базовая версия BERT-base с 108 миллионами параметров и как большая версия BERT-large с 334 миллионами параметров.

Модель ModernBERT [25] является моделью представления естественного языка на основе улучшенной архитектуры Transformer, состоящей только из кодировщика. Модель ModernBERT была предложена в 2024 году разработчиками компаний Answer.AI и LightOn при сотрудничестве с компанией Hugging Face. Эта модель решает многие проблемы своих предшественников, используя современную эффективную архитектуру, которая позволяет обрабатывать текстовые данные в несколько раз быстрее. Длина контекста модели ModernBERT составляет 8192 токенов.

В модели ModernBERT выполнена модификация ряда компонентов архитектуры модели BERT, в частности, реализованы [25, 26]:

– замена традиционной позиционной кодировки на ротационное позиционное встраивание (Rotary Positional Embeddings, RoPE), что обеспечивает улучшенное понимание относительных позиций между токенами и возможность масштабирования на более длинные последовательности (до 8192 токенов);

– замена MLP (Multi-Layer Perceptron) слоёв на GeGLU (Generalized Gaussian Linear Unit) слои и усовершенствование функции активации GeLU (Gaussian Error Linear Unit), используемой в модели BERT;

– распаковка (unpadding), гарантирующая отсутствие затрат вычислительных ресурсов на токены заполнения, что ускоряет время обработки пакетов с последовательностями смешанной длины;

– переменное внимание (alternating attention), при котором глобальное внимание используется только каждые 3 слоя внимания, а большинство слоев внимания используют локальное внимание со скользящим окном из 128 токенов;

– флэш-внимание (flash attention), ускоряющее процесс обработки;

– дизайн архитектуры, обеспечивающий максимальную эффективность графических процессоров вывода.

В целом, повышение эффективности модели ModernBERT достигается за счет механизма чередования внимания для повышения эффективности обработки; методов отмены заполнения и упаковки последовательностей для снижения вычислительных затрат; дизайна архитектуры модели с учетом аппаратных особенностей для оптимизации использования вычислительного оборудования. Кроме того, эффективность модели ModernBERT обеспечивается за счёт использования более глубокой, но более тонкой архитектуры [с большим числом слоёв, но меньшим числом единиц (units) в слоё], контролируемого размера словаря и прогрессивного масштабирования модели, начиная с меньших моделей.

Модель ModernBERT обучена на данных из различных англоязычных источников, включая веб-документы, программный код, математические данные и научные статьи. Общий объем обучающих данных составляет 2 триллиона токенов, большинство из которых являются уникальными. При этом модель ModernBERT использует токенизатор OLMo (Open Language MOdel) [27], который был специально обучен на программном коде, в отличие от стандартного токенизатора модели BERT. Эта модель является первой моделью, состоящей только из кодировщика, для обучения которой использовался большой объем программного кода в своих обучающих данных.

Модель ModernBERT может успешно решать не только задачи, решавшиеся ранее с применением модели BERT, но и новые задачи, в том числе – задачи, связанные с крупномасштабным поиском кода, задачи конвейеров RAG (Retrieval Augmented Generation, генерация дополненной поисковой информации) и задачи рекомендательных систем.

Модель ModernBERT доступна к использованию в двух версиях: как базовая версия ModernBERT-base с 139 миллионами параметров и как большая версия ModernBERT-large с 395 миллионами параметров.

2. Подходы к одноклассовой классификации

В настоящее время известен ряд подходов, позволяющих решать задачи одноклассовой классификации. К таким подходам относят, в частности, подходы, реализующие применение таких алгоритмов машинного обучения, как OCSVM алгоритм [9–14], IF алгоритм [15–19] и LOF алгоритм [20, 21].

При работе с такими алгоритмами предполагается, что в наличии имеется обучающий набор данных, содержащий объекты $\mathbf{x}_i$ ($i=\overline{1,N}$), которые принадлежат к мажоритарному классу. Для объектов $\mathbf{z}_l$ ($l=\overline{1,K}$) тестового набора данных необходимо решить, принадлежат ли они к мажоритарному или миноритарному классу. При том предполагается, что каждый объект обучающего или тестового набора данных представлен m-мерным вектором в пространстве R^m, например, объект $\mathbf{x}_i$ ($i=\overline{1,N}$) из обучающего набора данных имеет вид: $\mathbf{x}_i=(x_{i,1},x_{i,2},...,x_{i,m})$. В контексте настоящего исследования под объектом понимается векторное представление РВ.

OCSVM алгоритм и IF алгоритм предполагают последовательную реализацию этапов обучения и тестирования при разработке одноименных классификаторов. При этом объекты миноритарного класса не используются в обучающем наборе данных. LOF алгоритм отличается тем, что при его использовании для разработки одноименного классификатора этап обучения как таковой отсутствует [20, 21]. Обучающий набор данных, содержащий только объекты мажоритарного класса, используется в качестве основы для выявления паттернов новизны в новых данных, например, в тестовом наборе данных. Для определения новизны в тестовом наборе данных на основе LOF алгоритма обучающий набор данных объединяют с тестовым набором данных, а затем выполняют предсказание меток для объектов тестового набора данных с учётом ожидаемого в нём процента новизны, реализуя поиск ближайших соседей: объекты, которые находятся далеко от других в пространстве признаков, рассматриваются как паттерны новизны. При этом каждому объекту на основе размера его локального соседства присваивается оценка того, насколько он изолирован или насколько вероятно, что он будет паттерном

новизны. Объекты с наибольшими значениями таких оценок с большей вероятностью будут паттернами новизны. Классификатор на основе LOF алгоритма может хорошо работать в пространстве признаков малой размерности, но по мере увеличения числа признаков качество принимаемых решений существенно ухудшается из-за так называемого проклятия размерности.

Так как в настоящем исследовании в качестве объектов выступают регулярные выражения произвольной длины, а для их векторизации планируется использовать, в том числе, модели с архитектурой «трансформер», формирующие векторы большой длины, а также ввиду явного концептуального отличия процесса разработки классификатора на основе LOF алгоритма от процессов разработки классификатора OCSVM и IF алгоритмов, было принято решение отказаться от дальнейшей работы с LOF алгоритмом.

2.1. Одноклассовый SVM классификатор

В отличие от традиционного SVM классификатора, целью которого является решение задачи бинарной классификации, целью одноклассового SVM классификатора [9–14], разрабатываемого на основе одноклассового SVM алгоритма, является выявление новизны в данных посредством моделирования границы одного класса. Классификатор, разработанный на основе этого алгоритма, должен обеспечить обнаружение редких (аномальных) объектов в данных. Такие объекты появляются нечасто, поэтому их почти нет в доступном наборе данных. Проблему обнаружения редких (аномальных) объектов можно интерпретировать как проблему классификации несбалансированного набора данных.

Для исследуемого набора данных OCSVM классификатор обеспечивает отделение объектов, которые относятся к мажоритарному классу, от остальных объектов, которые можно считать редкими (аномальными) объектами, относящимися к миноритарному классу.

При обучении одноклассового SVM классификатора на основе обучающего набора данных, содержащего объекты $\mathbf{x}_i$ ($i=\overline{1,N}$), решается двойственная задача оптимизации:

$$\begin{cases} \frac{1}{2}\cdot\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}\lambda_i\cdot\lambda_j\cdot k(\mathbf{x}_i,\mathbf{x}_j)\to\min\limits_{\lambda} \\ \sum\limits_{i=1}^{N}\lambda_i=1, \\ 0\le\lambda_i\le 1/(\nu\cdot N),\ i=\overline{1,N}. \end{cases} \tag{2}$$

где $\mathbf{x}_i,\mathbf{x}_j$ – i-й и j-й объекты соответственно, для которых вычисляется значение функции ядра; λ_i,λ_j – двойственные переменные для i-го и j-го объектов соответственно; $i=\overline{1,N}$; $j=\overline{1,N}$; N – число объектов в обучающем наборе данных; $k(\mathbf{x}_i,\mathbf{x}_j)$ – функция ядра; ν – максимальная доля объектов в обучающем наборе, которые могут быть приняты в качестве редких (аномальных) объектов.

При этом величина $1/(\nu\cdot N)$ определяет значение параметра регуляризации C: $C=1/(\nu\cdot N)$, а в качестве функции ядра $k(\mathbf{x}_i,\mathbf{x}_j)$ [28–30] может быть использована линейная функция ядра (3), радиальная базисная функция ядра (4), полиномиальная функция ядра (5) или сигмоидная функция ядра (6):

$$k(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i\bullet\mathbf{x}_j, \tag{3}$$

$$k(\mathbf{x}_i,\mathbf{x}_j)=(\gamma\cdot\mathbf{x}_i\bullet\mathbf{x}_j+\chi)^{\vartheta}, \tag{4}$$

$$k(\mathbf{x}_i,\mathbf{x}_j)=(exp(-\gamma\cdot(\mathbf{x}_i-\mathbf{x}_j)\bullet(\mathbf{x}_i-\mathbf{x}_j)), \tag{5}$$

$$k(\mathbf{x}_i,\mathbf{x}_j)=th(\gamma\cdot\mathbf{x}_i\bullet\mathbf{x}_j+\eta), \tag{6}$$

где $\mathbf{x}_i,\mathbf{x}_j$ – i-й и j-й объекты соответственно, для которых вычисляется значение функции ядра; «$\bullet$» – знак скалярного произведения; $\vartheta\in N$; $\gamma>0$; $\chi>0$; $\eta<0$; th – функция гиперболического тангенса.

В результате разработки одноклассового SVM-классификатора определяется решающее правило, которое присваивает произвольному объекту $\mathbf{x}$ из тестового набора данных класс принадлежности с меткой «−1», если объект определен как редкий (аномальный), или «+1» в противном случае [9]:

$$F(\mathbf{x})=sign\left(\sum_{i=1}^{N}\lambda_i\cdot k(\mathbf{x}_i,\mathbf{x})-\rho\right), \tag{7}$$

где значение параметра ρ находится из соотношения $\rho = \sum_{j=1}^{N} \lambda_j \cdot k(\mathbf{x}_j, \mathbf{x}_i)$, справедливого для любого объекта $\mathbf{x}_i$, которому соответствует значение λ_i, удовлетворяющее условию $0 < \lambda_i < 1/(\nu \cdot N)$; N – число объектов в обучающем наборе данных.

При тестировании одноклассового SVM классификатора на основе тестового набора данных, содержащего объекты $\mathbf{z}_l$ ($l = \overline{1, K}$), оценивается качество разработанного классификатора с использованием тех или иных метрик качества классификации.

2.2. Одноклассовый классификатор на основе изолирующего леса

В отличие от традиционного RF (Random Forest) классификатора, целью которого является решение задачи бинарной (мультиклассовой) классификации, целью IF классификатора на основе IF алгоритма является изоляция, то есть отделение редких (аномальных) объектов от остальных объектов [15–19]. Изолирующий лес *iForest* в IF алгоритме формируется на основе некоторого числа изолирующих деревьев *iTree*. Разбиение объектов в случайном дереве выполняется рекурсивно до тех пор, пока все объекты не будут изолированы друг от друга. Такое дерево имеет существенно более короткие пути для редких (аномальных) объектов, так как меньшее число редких (аномальных) объектов приводит к меньшему числу разбиений, и, как следствие, – к более коротким путям в древовидной структуре. При этом предполагается, что объекты с различимыми значениями признаков с большей вероятностью должны быть разделены при раннем разбиении. Если лес случайных деревьев создает более короткие длины путей для некоторых конкретных объектов, то такие объекты с высокой вероятностью являются редкими (аномальными), то есть могут считаться паттернами новизны.

Узел T изолирующего дерева может быть внешним узлом без потомков или внутренним узлом с одним тестом (условием) и двумя дочерними узлами T_{left} и T_{right}. Тест состоит из признака q и порогового значения (значения разделения данных в узле) p, выбранного таким образом, что условие $q < p$ приводит к разделению объектов на узлы T_{left} и T_{right}. Для построения изолирующего дерева *iTree* на основе заданной выборки данных $\psi = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_V\}$, где V – число объектов в подвыборке, каждый объект которой представлен m-мерным вектором в пространстве R^m, необходимо рекурсивно разделять подвыборку ψ, случайным образом выбирая признак q и пороговое значение p, пока не будет достигнута одна из ситуаций: дерево имеет предельную высоту; $|\psi| = 1$; все объекты в ψ совпадают. В предположении, что все объекты различны, каждый из них в результате построения изолирующего дерева *iTree* будет изолирован внешним узлом.

Задача обнаружения аномалий состоит в том, чтобы выполнить ранжирование объектов, отражающее степень их аномальности. Один из способов обнаружения аномалий – сортировка объектов в соответствии с их длинами пути или оценками аномальности.

Длина пути $h(\mathbf{x})$ для объекта $\mathbf{x}$ определяется числом ребер, по которым необходимо пройти для того, чтобы попасть из корневого узла изолирующего дерева *iTree* во внешний узел, соответствующий объекту $\mathbf{x}$.

Так как изолирующие деревья имеют структуру, эквивалентную структуре бинарного дерева поиска (Binary Search Tree, BST), оценка средней длины пути $h(\mathbf{x})$ для внешних узлов совпадает с оценкой средней длины неудачного поиска в BST, которая вычисляется как:

$$c(V) = 2 \cdot H(V-1) - 2 \cdot (V-1)/V, \tag{8}$$

где $H(V-1)$ – гармоническое число, оцениваемое как $\ln(V-1) + \gamma$; γ = 0.5772156649 (константа Эйлера – Маскерони).

Тогда оценка аномальности $s(\mathbf{x}, V)$ объекта $\mathbf{x}$ определяется как:

$$s(\mathbf{x}, V) = 2^{-\frac{E(h(\mathbf{x}))}{c(V)}}, \tag{9}$$

где $E(h(\mathbf{x}))$ – среднее значение для $h(\mathbf{x})$ в наборе изолирующих деревьев.

Используя оценку аномальности $s(\mathbf{x}, V)$, можно сделать следующие выводы: если оценка $s(\mathbf{x}, V)$ для объекта $\mathbf{x}$ близка к 1, то он является аномальным; если оценка $s(\mathbf{x}, V)$ для объекта $\mathbf{x}$ намного меньше 0.5, то он является нормальным; если оценка $s(\mathbf{x}, V)$ для всех объектов близка к 0.5, то вся выборка не имеет какой-либо отчетливой аномальности.

Изолирующий лес *iForest* является ансамблем изолирующих деревьев, он определяет аномальные объекты как объекты с более короткими длинами пути.

Особое внимание при выявлении паттернов аномальности на основе IF алгоритма должно уделяться проблемам подтопления и маскирования.

Под подтоплением (swamping) понимают неправильное определение нормальных объектов как аномальных. Когда нормальные объекты расположены очень близко к аномальным объектам, число разбиений, необходимых для отделения аномальных объектов, увеличивается, что усложняет процесс отделения аномальных объектов от нормальных.

Под маскированием (masking) понимают наличие слишком большого числа аномальных объектов, скрывающих их собственное присутствие. Наличие большого и плотного кластера аномальных объектов также увеличивает число разбиений, необходимых для того, чтобы изолировать каждый аномальный объект. Это затрудняет обнаружение аномальных объектов из-за увеличения длин путей в изолирующем дереве.

Подтопление и маскирование возникают из-за слишком большого объёма данных, в котором осуществляется поиск паттернов аномальности. Изолирующие деревья позволяют изолирующему лесу *iForest* работать с подвыборками данных, смягчая эффекты подтопления и маскирования благодаря тому, что:

– подвыборка контролирует размер данных, что помогает изолирующему лесу *iForest* лучше изолировать аномальные объекты;

– каждое изолирующее дерево может быть уникальным, так как каждая подвыборка включает в себя разный набор аномальных объектов или вообще не содержит их.

При обучении IF классификатора на основе обучающего набора данных, содержащего объекты $\mathbf{x}_i$ ($i=\overline{1,N}$), строятся t изолирующих деревьев посредством рекурсивного разбиения подвыборок до тех пор, пока объекты не будут изолированы или не будет достигнута предельная высота L дерева, которая определяется размером подвыборки V: $L=\lceil \log_2 V \rceil$, где $\lceil\cdot\rceil$ – операция округления до ближайшего целого в большую сторону.Вычисленный таким образом предел L приблизительно равно средней высоте дерева. Обоснование выращивания деревьев до средней высоты дерева заключается в том, что целевыми являются только объекты, которые имеют длину пути короче средней, поскольку эти объекты с большей вероятностью будут аномальными. Эмпирически установлено, что размер подвыборки V, равный 256, обычно обеспечивает достаточно деталей для обнаружения аномалий в широком диапазоне данных.

При тестировании IF классификатора на основе тестового набора данных, содержащего объекты $\mathbf{z}_l$ ($l=\overline{1,K}$), оценивается качество разработанного классификатора с использованием тех или иных метрик качества классификации. При этом тестовые объекты проходят через изолирующие деревья, чтобы получить оценку аномальности. Оценка аномальности $s(\mathbf{z}_l,V)$ вычисляется для каждого тестового объекта $\mathbf{z}_l$ на основе ожидаемой длины пути $E(h(\mathbf{z}_l))$, которая определяется посредством прохождения через каждое изолирующее дерево *iTree* в изолирующем лесу *iForest*. Длина пути $h(\mathbf{z}_l)$ вычисляется посредством подсчета числа ребер e от корневого узла до внешнего узла, когда объект $\mathbf{z}_l$ проходит через изолирующее дерево *iTree*. Если при обходе дерева достигается предельная высота L, длина пути оценивается как число ребер e, увеличенное на величину $c(size)$, которая учитывает оценку средней длины пути случайного поддерева, которое может быть построено с использованием данных размера $size$ за пределами предельной высоты L дерева. Оценка аномальности объекта $\mathbf{z}_l$ вычисляется на основе (9).

Если требуется выбрать υ аномальных объектов из набора тестовых объектов, необходимо отсортировать этот набор по убыванию значений оценок аномальности (9). Первые υ объектов в полученном списке следует считать паттернами аномальности.

3. Оценка качества классификации

Оценка качества классификации может быть выполнена с применением различных метрик качества, в частности с применением метрик F_1, *Precision* и *Recall* [31]. При этом эти метрики могут быть рассчитаны как на всех объектах тестовой выборки, так и только на объектах, принадлежащих миноритарному классу, то есть – на паттернах новизны.

В зависимости от конкретики решаемой задачи, может быть принято решение о максимизации той или иной метрики качества классификации при контроле значений других метрик.

4. Экспериментальные исследования

Экспериментальная апробация рассматриваемых подходов была выполнена на языке Python 3.10 в среде Google Colab с применением программных реализаций алгоритмов машинного обучения One-Class SVM [32] и Isolation Forest [33], а также – моделей двунаправленных предобученных трансформеров BERT [34] и ModernBERT [35] в базовых (base) версиях.

В ходе экспериментальных исследований были использованы два набора данных: RegEx1 и RegEx2.

Экспериментальный набор данных RegEx1 был сформирован на основе набора данных RegexEval [36], представленного в jsonl формате и содержащего 762 строки. Каждая строка этого набора содержит объект, описываемый такими полями, как: id (уникальный идентификатор объекта), expression (регулярное выражение); raw_prompt (исходный запрос от реального пользователя); refined_prompt (уточненный запрос от реального пользователя); matches (примеры совпадений для регулярного выражения); non-matches (примеры несовпадений для регулярного выражения).

На рисунке 1 приведен фрагмент набора данных RegEval. При формировании экспериментального набора данных RegEx1 было использовано только поле expression набора данных RegexEval [36].

Экспериментальный набор данных RegEx2 был сформирован основе набора данных RegEx101, полученного в результате парсинга раздела library сайта regex101 [37]. Набор данных RegEx101 представлен в csv-формате и содержит 18008 строк. Каждая строка этого набора содержит объект, описываемый такими полями, как: id (уникальный идентификатор объекта), regex (регулярное выражение); description (описание назначения регулярного выражения).

На рисунке 2 приведен фрагмент набора данных RegEx101. При формировании экспериментального набора данных RegEx2 было использовано только поле regex набора данных RegEx101.

Для каждого экспериментального набора данных была выполнена его векторизация с применением моделей TF-IDF, BOW, BERT и ModernBERT. При векторизации РВ на основе моделей TF-IDF и BOW было выполнено посимвольное разбиение всех РВ. Ввиду специфики работы моделей BERT и ModernBERT, заключающейся в ограничениях на максимальную длину обрабатываемой последовательности символов, для некоторых РВ, представленных в исходных экспериментальных наборах данных RegEx1 и RegEx2, не удалось получить их векторное представление. Эти РВ были исключены из дальнейшего рассмотрения. Редуцированные экспериментальные наборы данных RegEx1 и RegEx2 содержат соответственно 756 и 17739 строк, описывающих различные РВ. В дальнейших исследованиях можно будет рассмотреть возможность усечения векторных представлений РВ, чтобы не отбрасывать их [38].

Векторы, описывающие РВ на основе моделей BERT и ModernBERT, содержат 768 элементов. Векторы, описывающие РВ на основе модели TF-IDF, для наборов данных RegEx1 и RegEx2 содержат соответственно 75 и 103 элемента. Векторы, описывающие РВ на основе модели BOW, для наборов данных RegEx1 и RegEx2 содержат соответственно 1547 и 1740 элементов.

На рисунке 3 представлены результаты визуализации на основе UMAP-алгоритма [39] для векторных представлений набора данных RegEx1 с применением моделей TF-IDF, BOW, BERT и ModernBERT. На рисунке 4 представлены результаты визуализации на основе UMAP-алгоритма для векторных представлений набора данных RegEx2 с применением моделей TF-IDF, BOW, BERT и ModernBERT.

Визуализация на основе UMAP-алгоритма для векторных представлений наборов данных RegEx1 и RegEx2 позволила сделать выводы о целесообразности дальнейшей работы именно с моделями BERT и ModernBERT. Эти модели обеспечивают лучшую отделимость кластеров, содержащих векторные представления РВ, отличающихся друга от друга. Модели TF-IDF, BOW не позволяют разделить РВ на какие-либо кластеры, отделимые друг от друга.

Анализ результатов визуализации на основе UMAP-алгоритма для векторных представлений экспериментальных наборов данных RegEx1 и RegEx2 с применением моделей BERT и ModernBERT позволил выделить в каждом наборе большой «сгусток» (кластер) РВ, объединив оставшиеся небольшие «сгустки» в другой кластер.

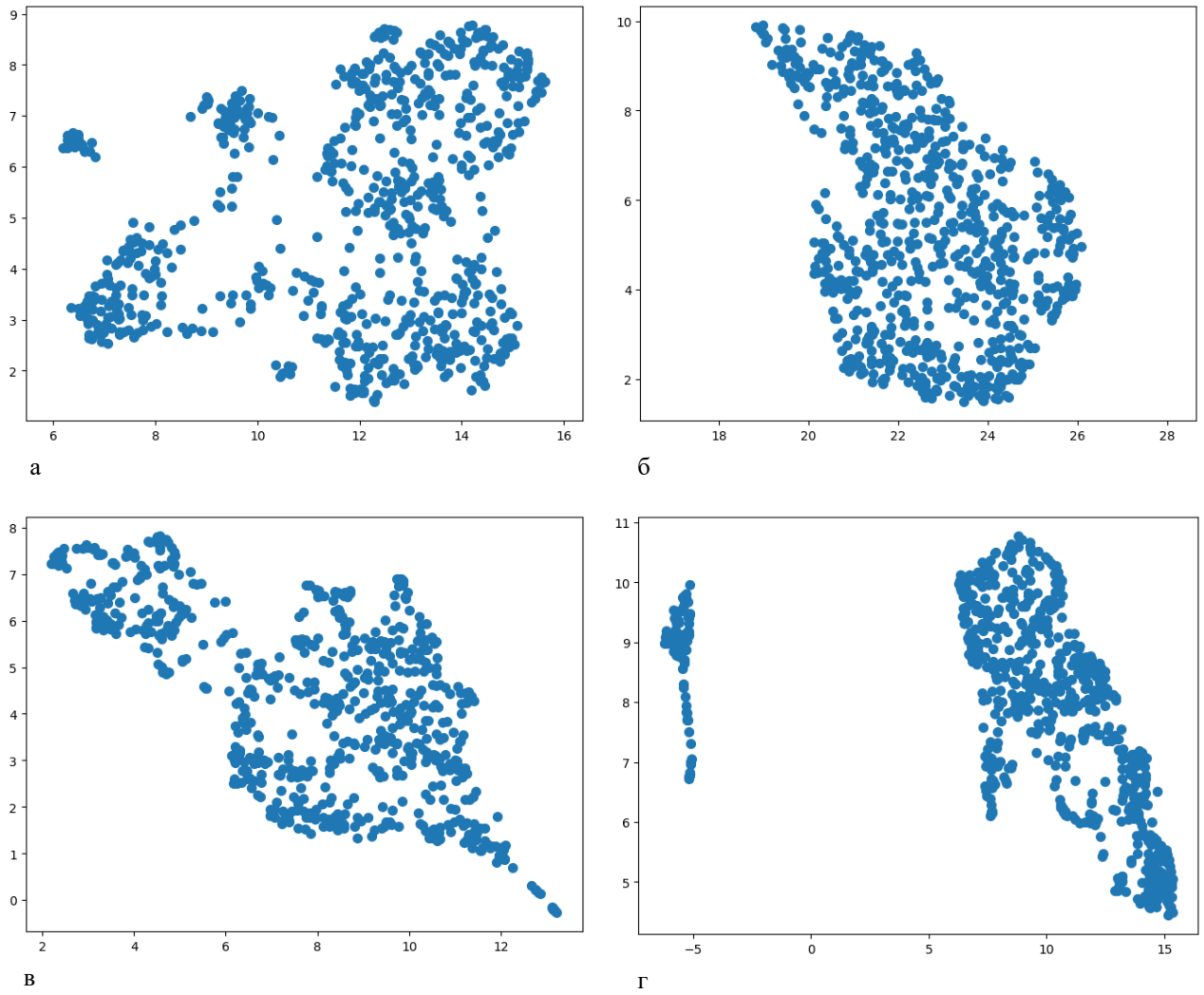
Для каждого из экспериментальных наборов данных RegEx1 и RegEx2 такое разделение данных на два кластера позволило определить на основе большего кластера мажоритарный класс, а на основе меньшего – миноритарный класс. Объекты мажоритарного класса в дальнейшем будут полагаться нормальными, а объекты миноритарного класса – будут полагаться паттернами новизны.

expression string · lengths 4–141 86.1%	raw_prompt string · lengths 11–119 49.9%	refined_prompt string · lengths 79–162 22.4%	matches sequence
^\d$	Matches exactly 1 numeric digit (0-9).	Matches exactly 1 numeric digit (0-9). Match examples: - "1" - "2" _	["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"]
^\d{5}$	Matches 5 numeric digits, such as a zip code.	Matches 5 numeric digits, such as a zip code. Match examples: - "33333"_	["33333", "55555", "23445", "89343", "46556", "25432", "25336", "43576", "68797", "30495", "54348", "45935", "93857"]
^\d{5}-\d{4}$	Numeric and hyphen 5+4 ZIP code match for ZIP+4.	Matches a string that starts with five digits, followed by a hyphen,_	["22222-3333", "34545-2367", "56334-2343", "34539-5433", "12349-5943", "54329-9875", "34852-5493", "83542-2314", "52435-3489", "12345-0983", "98745-2385", "48294-2945"]
^\d{5}$\|^\d{5}-\d{4}$	This regular expression will match either a 5_	Match either a 5 digit ZIP code or a ZIP+4 code formatted as a string_	["55555-5555", "34564-3342", "90210", "03945", "02946", "46556", "52346", "34534-3252", "49672-3923", "59403-6934", "35349-6753", "52346-3953"]
^\d{3}-\d{2}-\d{4}$	This regular expression will match a hyphen-_	This regular expression will match a hyphen-separated Social Security_	["333-22-4444", "123-45-6789", "534-74-2573", "234-45-6235", "968-24-4395", "948-53-5924", "493-42-5938", "954-97-5942", "534-96-3623", "549-69-3456", "543-54-6396", "294-68-1957"]
^[a-zA-Z]$	Matches any single upper- or lower-case letter.	Matches any single upper- or lower-case letter. Match examples: - "a" _	["a", "B", "c", "f", "J", "D", "s", "D", "d", "o", "F", "p", "r"]
^[a-zA-Z]+$	Matches any string of only upper- and lower- case_	Matches any string of only upper- and lower- case letters (no spaces)_	["abc", "ABC", "aBcDeF", "dfasdfs", "ALKFJsf", "FDSfdsf", "AJFDAL", "sfdjk", "FISD", "sadfkjl", "fadghui", "FDSnfsdFDS", "sdfjoi"]
^[a-zA-Z0-9]+$	Matches any alphanumeric string (no spaces).	Matches any alphanumeric string (a string that contains only English_	["10a", "ABC", "A3fg", "AFDSA", "asfsafd", "123", "123123abcAfds", "2398djAD", "dsfhoi329", "98hsd", "NOTREDAME", "ILOVEND", "notredame46556"]
^\d+$	Positive integer value.	Positive integer value. Match examples: - "123" - "10" - "54" Non_	["123", "10", "54", "39", "583", "395", "2394", "2394854", "384", "324", "222", "1", "2", "39", "583", "395", "2394", "2394854", "384", "324", "222", "1", "2"]
^(\+|-)?\d+$	Matches any signed integer.	Matches any signed integer, in other words, a string that can start with_	["-34", "34", "+5", "809", "790987", "+5483920", "-2354", "+540893", "-234", "2349", "5094", "234234", "+123"]
^[a-zA-Z]\w{3,14}$	The password's first character must be a_	The password's first character must be a letter, it must contain at_	["abcd", "aBc45DSD_sdf", "password", "F3fsdsdf_234", "F0392", "fd02934", "k2039", "ABC09314", "abc_ABC", "NOTREDAME", "fadghui", "QWERTY", "p0p0p0"]
^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$	Simple email expression. Doesn't allow numbers in_	Checks if an email address starts with one or more alphanumeric_	["joe@aol.com", "ssmith@aspalliance.com", "a@b.cc", "apple@qq.com", "123@qq.com", "abc@abc.com", "ab@ab.ab", "bc@bc.bc", "avpl@qq.cn", "sustech@nd.com", "msft@nd.us", "notredame@nd.edu"]
^\d{1,2}\/\d{1,2}\/\d{4}$	This regular expressions matches dates of the form_	This regular expressions matches dates of the form XX/XX/YYYY where_	["4/1/2001", "12/12/2001", "55/5/3434", "1/1/1111", "2/2/2222", "23/3/3333", "11/11/1111", "1/1/1111", "30/30/3030", "55/5/5555", "5/55/5678", "12/12/1221", "55/44/4321"]
^[1-5]$	This matches a single numeric digit between 1_	Matches a single digit between 1 and 5, inclusive. It will only match a_	["1", "3", "4", "2", "5"]

Рисунок 1 – Фрагмент набора данных RegexEval [36]

regex	description
^mongodb:\/\/[^\/\,]+:{1}[^\/\,]*@{1}.*$	Test if a mongo db connection string has a username and password declared in it. _Doesn't work with with multi host ATM_
.*[Product|product]\":\"([^\"]+).*	
^((([0-9])|(1([0-9]{1,2}))|25[0-5]|(2[0-4][0	as the title says
^(?:[[:^print:][:cntrl:]\s]|GIF89.{0,20})*<\?(?:php)?\s*[[:punct:]\s]+scama\s*spotify\s*v1[\s\#]+facebook:\s*fb\.com\/amyr\.gov\.tn+[[:punct:]\s]+.{0,200}?want\s*to\s*save\s*rzlt\s*.html(?:[^>]+>\s*)?$	
^\[([^\]]+)\] \[(오[전후] \d{1,2}:\d{2})\] (.*	정규식 예제
href=\\"(.*)\\"	
^[0-9][' ']{7,7}	
eventtype="(?<group>administrator)",username="(?<login>.*?)",action="(?<vmid>(?<action>.*?))",object="(?<object>(?<account>.*?))",description="{(?:.*)?,."status":."(?<status>.*)\"}"	
^".*?"	CmdLineRaw
((?:\\033|\e|\\x1B|\\x1b){1}(?:\[(?:=|\?|[(	Use control_sequence_filter group to find and remove ANSI CSI codes. Not tested thoroughly. Only for the task that was necessary for me
(^\d{8}(0\d|10|11|12)([0-2]\d|30|31)\d{3	身份证号, 支持1/2代(15位/18位数字)
name\s?[a-z]{0,2}\s([A-Z]{1}[a-z]+)	
(^\d+\s+[\w]+)[,]+([a-zA-Z]{3,})[,]*({AL|/	Designed for a basic validation of a US address. Used in an Angular 6 app as an Address Validation. For extraction $1 is the numeric address, $2 is the street and a
noticias\/(\d+).shtml	
GRUB_CMDLINE_LINUX="(?:\s*?([\w=]+)\	Trying to match grub options in capture groups, some of which are known
Financial Highlights.*?(Operating revenue)\D*([0-9,]*)[)]?\s*[(]?([0-9,]*)	
Team{name='([a-zA-Z0-9]+)', color=Color:\[rgb(0x[a-zA-Z0-9]+)\], members=\[()\]}	
peso\:(\s)?\d+(.|,)\d+(\s)?kg	Captures the weight of a product.
^\w+([-+.']\w+)*@\w+([-.]\w+)*\.(?i)lu$	
(https?:\/\/)?(www\.)?youtu\.?be(\.com\/	Created for replacing everything except video ID from YouTube URL
^[0-9]{8}$	
^(?:(?:(?:mdmcomm|checkin)\.(?:eu|us|jp	Match ESET PROTECT Cloud domains for exclusion from DNS logs
\w+-gradient\(.*?\)\)|\w+-gradient\(.*?\)	Match CSS Gradients
\(([0-9.]*)\)	
(\w+.ACT\w+),?.*	
^(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})	

Рисунок 2 – Фрагмент набора данных RegEx101



а б в г

Рисунок 3 – Результаты визуализации на основе UMAP-алгоритма для векторных представлений набора данных RegEx1 с применением моделей:
а –TF-IDF;
б – BOW;
в – BERT;
г – ModernBERT

При разработке OCSVM и IF классификаторов на основе экспериментальных наборов данных RegEx1 и RegEx2 были разделены на обучающую и тестовую выборку в соотношении 80:20. При этом на обучающей выборке осуществлялся подбор оптимальных значений параметров классификаторов. Для этого обучающая выборка еще раз разделялась на обучающую и тестовую подвыборки в соотношении 80:20. При этом из обучающей подвыборки удалялись объекты миноритарного класса. Подбор оптимальных значений параметров классификаторов был выполнен с использованием поиска по сетке с целью выбора лучших классификаторов в смысле максимизации значения метрики F_1 на тестовых подвыборках. При разработке OCSVM классификаторов осуществлялся подбор типа функции ядра, значений параметров функции ядра и числа ν (*nu* в [32]), характеризующего верхнюю границу доли ошибок обучения (максимальную долю ошибок обучения) и нижнюю границу доли опорных векторов. При этом были рассмотрены функции ядра (3) – (6). При разработке IF классификаторов осуществлялся подбор числа деревьев *nTrees* (*n_estimators* в [33]) и числа *contamination* [33], характеризующего долю новизны (выбросов) в наборе данных (степень загрязнения набора данных). В качестве значений остальных параметров OCSVM и IF классификаторов были использованы значения, установленные по умолчанию в библиотеках [32] и [33] соответственно. Обученные таким образом классификаторы были дополнительно протестированы на тестовых выборках с целью оценить их способность выявлять объекты (РВ), принадлежащие миноритарному классу, то есть – паттерны новизны.

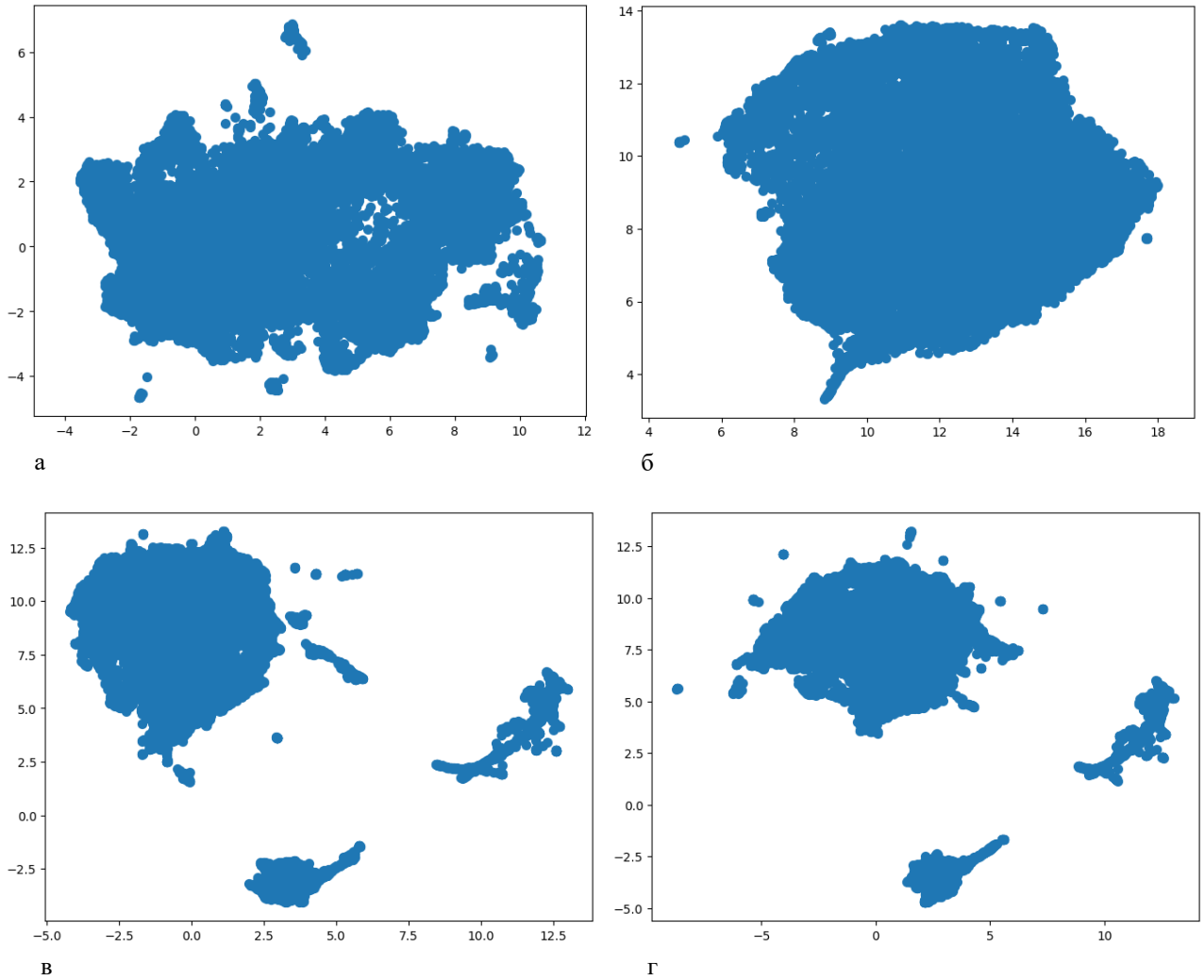


Рисунок 4 – Результаты визуализации на основе UMAP-алгоритма для векторных представлений набора данных RegEx2 с применением с применением моделей:
а –TF-IDF;
б – BOW;
в – BERT;
г – ModernBERT

В таблице 1 приведено число объектов в классах в каждом наборе данных с учётом разбиения на обучающую и тестовую выборки.

Таблица 1 – Число объектов в классах в экспериментальных наборах данных

Название набора данных	Число объектов в мажоритарном классе		Число объектов в миноритарном классе	
	В обучающей выборке	В тестовой выборке	В обучающей выборке	В тестовой выборке
RegEx1	456	148	119	33
RegEx2	12078	2113	3053	495

В таблице 2 приведены значения параметров лучших классификаторов на основе экспериментальных наборов данных RegEx1 и RegEx2, выбранные с использованием поиска по сетке, значения метрики F_1 на тестовых подвыборках, соответствующие значениям параметров лучших классификаторов (при обучении), а также значения метрик F_1, *Precision* и *Recall*, вычисленные на тестовых выборках на объектах миноритарного класса (таблица 1).

Таблица 2 – Характеристики классификаторов

Название набора данных	Название классификатора + модель векторизации	Значения параметров лучшего классификатора	Значение метрики F_1 на тестовой подвыборке (при обучении)	Значения метрик на тестовой выборке на объектах миноритарного класса		
				F_1	*Precision*	*Recall*
RegEx1	OCSVM+BERT	Линейная функция ядра, ν = 0.05	0.724	0.758	0.781	0.735
	IF+BERT	Число деревьев *nTrees* = 144, *contamination*=0.107	0.667	0.500	0.452	0.559
	OCSVM+Modern-BERT	Сигмоидная функция ядра (6), η = 0.2, γ = 0.005, ν = 0.005	**0.958**	**0.958**	**0.919**	**1.000**
	IF+ModernBERT	Число деревьев *nTrees* = 144, *contamination* = 0.107	***0.889***	***0.822***	***0.769***	***0.882***
RegEx2	**OCSVM+BERT**	Линейная функция ядра, ν = 0.005	**0.906**	**0.912**	**0.838**	**1.000**
	IF+BERT	Число деревьев *nTrees* = 91, *contamination* = 0.194	0.546	0.450	0.347	0.638
	OCSVM+Modern-BERT	Линейная функция ядра, ν = 0.01	***0.899***	***0.905***	***0.826***	***1.000***
	IF+ModernBERT	Число деревьев *nTrees* = 75, *contamination* = 0.206	0.555	0.432	0.327	0.634

В таблице 2 для каждого набора данных жирным шрифтом выделены названия классификаторов, имеющие первые в рейтинге (то есть лучшие – максимальные) значения метрики F_1 на тестовой подвыборке. Кроме того, жирным шрифтом выделены соответствующие этим классификаторам значения метрики F_1 на тестовой подвыборке, а также – значения метрик F_1, *Precision* и *Recall*, вычисленные на тестовых выборках на объектах миноритарного класса. Видно, что для экспериментальных наборов данных RegEx1 и RegEx2 лучшими оказались OCSVM классификаторы, однако для набора данных RegEx1 лучшей оказалась модель векторизации ModernBERT, а для набора данных RegEx2 – модель векторизации BERT.

В таблице 2 для каждого набора данных жирным курсивом выделены названия классификаторов, имеющие вторые в рейтинге значения метрики F_1 на тестовой подвыборке. Кроме того, жирным курсивом выделены соответствующие этим классификаторам значения метрики F_1 на тестовой подвыборке, а также – значения метрик F_1, *Precision* и *Recall*, вычисленные на тестовых выборках на объектах миноритарного класса. Видно, что для экспериментального набора данных RegEx1 таковым оказался IF классификатор, а для экспериментального набора данных RegEx2 – OCSVM классификатор. При этом в обоих случаях лучшей моделью векторизации оказалась модель ModernBERT.

В целом можно отметить следующее: на основе результатов классификации, полученных для рассмотренных экспериментальных наборов данных, в большинстве случаев лучшей моделью векторизации оказалась модель ModernBERT. Однако в виду ограниченного числа экспериментов пока нельзя однозначно отдать предпочтение той или иной модели векторизации – BERT или ModernBERT – в смысле обеспечения лучшего качества представления рассматриваемых РВ в векторном виде. Явный проигрыш IF классификаторов по отношению к OCSVM классификаторам для рассмотренных экспериментальных наборов данных можно объяснить возможным проявлением проблем подтопления и маскирования данных.

Заключение

На основе результатов проведенного исследования можно сделать вывод, что одноклассовые классификаторы целесообразно использовать для выявления аномальных РВ, являющихся паттернами новизны. Лучшим классификатором в рассматриваемой задаче оказался OCSVM классификатор, в то время как IF классификатор не смог обеспечить хорошие значения метрик качества классификации. Скорее всего, это объясняется наличием таких проблем, как подтопление и маскирование данных.

Кроме того, важно верно выбрать модель векторного представления РВ, с использованием которой исходные данные будут подготовлены для обучения одноклассовых классификаторов. Простейшие классические модели векторизации текстов, такие как TF-IDF и BOW, не позволяют сформировать векторные представления с достаточным количеством информации, чтобы по ним определить принадлежность РВ к определенному классу. Современные модели векторизации текстов, такие как BERT и ModernBERT обеспечивают более высокое качество векторных представлений данных. При этом пока нельзя сделать однозначный вывод о превосходстве одной модели векторизации на другой, так как на разных наборах РВ лучшие одноклассовые классификаторы были на основе разных моделей векторизации, хотя модель ModernBERT чаще оказывалась лучшей.

Рассмотренные подходы к одноклассовой классификации регулярных выражений могут быть применены для проверки новых данных, в том числе – генерируемых, на наличие в них нормальных паттернов и паттернов новизны.

Целью дальнейших исследований является разработка и анализ возможностей одноклассовых нейронных сетей, в частности одноклассовых ELM (One Class Extreme Learning Machines) [40, 41] и одноклассовых GNN (One-Class Graph Neural Networks) [42].

Список литературы

1. Chapman C., Stolee K.T. Exploring regular expression usage and context in Python // Proceedings of the 25th International Symposium on Software Testing and Analysis – ISSTA. – 2016. – P. 282–293.
2. Chapman C., Wang P., Stolee K.T. Exploring Regular Expression Comprehension // 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA. – 2017. – P. 405–416.
3. Arcaini P., Gargantini A., Riccobene E. MutRex: A Mutation Based Generator of Fault Detecting Strings for Regular Expressions // International Conference on Software Testing, Verification and Validation Workshops (ICSTW). – 2017. – 10 p.
4. Davis J.C., Moyer D., Kazerouni A., Lee D. Testing regex generalizability and its implications: A large-scale many-language measurement study // 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA. – 2019. – P. 427–439.
5. Louis G.M. IV, Donohue J., Davis J.C., Lee D., Servant F. Regexes are Hard: Decision-making, Difficulties, and Risks in Programming Regular ExpressionsSe // arXiv:2303.02555v12023. – 2023. https://doi.org/10.48550/arXiv.2303.02555.
6. Onyenwe I., Ogbonna S., Onyedimma E., Ikechukwu-Onyenwe O., Nwafor Ch. Developing Smart Web-Search Using RegEx // arXiv:2110.04767. – 2021. https://doi.org/10.48550/arXiv.2110.04767.
7. Frenz Ch.M. Introduction to Searching with Regular Expressions // arXiv:0810.1732. – 2008. https://doi.org/10.48550/arXiv.0810.1732.
8. Khan S.S., Madden M.G. A survey of recent trends in one class classification // Artificial Intelligence and Cognitive Science. AICS 2009. Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany. – 2010. – Vol. 620. P. 188–197.
9. Schölkopf B., Platt J.C., Shawe-Taylor J., Smola A.J., Williamson R.C. Estimating the Support of a High-Dimensional Distribution // Neural Computation. – 2001. – Vol. 13. – No. 7. – P. 1443–1471.
10. Manevitz L.M., Yousef M. One-class SVMs for document classification // Journal of Machine Learning Research. – 2001. – Vol. 2. – P. 139–154.
11. Li K.-L., Huang H.-K., Tian S.-F., Xu W. Improving one-class SVM for anomaly detection // Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Xi'an, China, 2–5 November 2013. – P. 2–5.
12. Erfani S.M., Rajasegarar S., Karunasekera S., Leckie C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning //Pattern Recognition. – 2016. – Vol. 58. – P. 121–134.
13. Zhu F., Yang J., Gao C., Xu S., Ye N, Yin T. A weighted one-class support vector machine // Neurocomputing. – 2016. – Vol. 189. – P. 1–10.
14. Yang K., Kpotufe S., Feamster N. An Efficient One-Class SVM for Anomaly Detection in the Internet of Things // arXiv:2104.11146. – 2021.
15. Liu F.T., Ting K.M., Zhou Z.-H. Isolation Forest // 2008 Eighth IEEE International Conference on Data Mining. 2008. – P. 413–422.
16. Liu F. T., Ting K. M., Zhou Z.-H. Isolation-Based Anomaly Detection // ACM Transactions on Knowledge Discovery from Data (TKDD). – 2012. – Vol. 6. – Issue 1. – Article No. 3. – P 1–39.

17. Chabchoub Y., Togbe M.U., Boly A., Chiky R. An In-Depth Study and Improvement of Isolation Forest // IEEE Access. – 2022. – Vol. 10. – P. 10219–10237.
18. Xu H., Pang G., Wang Y., Wang Y. Deep Isolation Forest for Anomaly Detection // IEEE Transactions on Knowledge and Data Engineering. – 2023. – Vol. 35. – No. 12. – P. 12591–12604.
19. Leveni F., Cassales G.W., Pfahringer B., Bifet A., Boracchi G. Online Isolation Forest // ICML'24: Proceedings of the 41st International Conference on Machine Learning. – 2024. – Article No. 1088. – P. 27288–27298.
20. Breunig M.M., Kriegel H.-P., Ng R.T., Sander J. LOF: identifying density-based local outliers // SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. – 2000. P. 93–104.
21. Alghushairy O., Alsini R., Soule T., Ma X. A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams // Big Data and Cognitive Computing. – 2021. – Vol. 5. – No. 1. https://doi.org/10.3390/bdcc5010001/.
22. Riego N.Ch.R., Villarba D.B. Utilization of Multinomial Naive Bayes Algorithm and Term Frequency-Inverse Document Frequency (TF-IDF Vectorizer) in Checking the Credibility of News Tweet in the Philippines // arXiv:2306.00018. – https://doi.org/10.48550/arXiv.2306.00018.
23. Qader W.A., Ameen M.M., Ahmed B.I. An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges // 2019 International Engineering Conference (IEC), Erbil, Iraq. – 2019. – P. 200–204.
24. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // arXiv:1810.04805. – 2018. https://doi.org/10.48550/arXiv.1810.04805.
25. Warner B., Chaffin A., Clavié B., Weller O., Hallström O., Taghadouini S., Gallagher A., Biswas R., Ladhak F., Aarsen T., Cooper N., Adams G., Howard J., Poli I. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference // arXiv:2412.13663. – 2024. https://doi.org/10.48550/arXiv.2412.13663.
26. Finally, a Replacement for BERT [Электронный ресурс]. – Режим доступа: https://huggingface.co/blog/modernbert, свободный (дата обращения: 20.12.2024).
27. OLMo [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/model_doc/olmo, свободный (дата обращения: 20.12.2024).
28. Vapnik V. The nature of statistical learning theory. 2nd edition. Wiley. – 1999. – 334 p.
29. Kuo N.-H., Chiang T.-W., Wong R. SVM/SVR Kernels as Quantum Propagators // arXiv:2502.11153. – 2025. https://doi.org/10.48550/arXiv.2502.11153.
30. Lin J.-T., Lin C.-J. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods // Neural Computation. – 2004. – Vol. 16. – No.5. – P. 1071–1090.
31. Gorchakov A.V., Demidova L.A., Sovietov P.N. Analysis of Program Representations Based on Abstract Syntax Trees and Higher-Order Markov Chains for Source Code Classification Task // Future Internet. – 2023. – Vol. 15. – P. 314. https://doi.org/10.3390/fi15090314.
32. OneClassSVM [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html, свободный (дата обращения: 20.12.2024).
33. IsolationForest [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, , свободный (дата обращения: 20.12.2024).
34. BERT [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/model_doc/bert, свободный (дата обращения: 20.12.2024).
35. ModernBERT [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/model_doc/modernbert, свободный (дата обращения: 20.12.2024).
36. RegexEval [Электронный ресурс]. – Режим доступа: https://huggingface.co/datasets/s2e-lab/RegexEval, свободный (дата обращения: 20.12.2024).
37. Regular Expressions 101 [Электронный ресурс]. – Режим доступа: https://regex101.com/, свободный (дата обращения: 20.12.2024).
38. Padding and truncation [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/pad_truncation, свободный (дата обращения: 20.12.2024).
39. McInnes L., Healy J., Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction // arXiv:1802.03426. – 2018. https://doi.org/10.48550/arXiv.1802.03426.
40. Leng Q., Qi H., Miao J., Zhu W., Su G. One-Class Classification with Extreme Learning Machine // Mathematical Problems in Engineering. – 2015. – Vol. 2015. Article No 412957, 11 p. https://doi.org/10.1155/2015/412957.
41. Li Y., Zhang S., Yin, Y., Zhang J.Parallel one-class extreme learning machine for imbalance learning based on Bayesian approach // Journal of Ambient Intelligence and Humanized Computing. – 2024. – Vol. 15. – P. 1745–1762.

42. Wang X., Jin B., Du Y., Cui P., Tan Y., Yang Y. One-class graph neural networks for anomaly detection in attributed networks // Neural Computing and Applications – 2021. – Vol. 33. – P. 12073–12085.

References

1. Chapman C., Stolee K.T. Exploring regular expression usage and context in Python // Proceedings of the 25th International Symposium on Software Testing and Analysis – ISSTA. – 2016. – P. 282–293.
2. Chapman C., Wang P., Stolee K.T. Exploring Regular Expression Comprehension // 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA. – 2017. – P. 405–416.
3. Arcaini P., Gargantini A., Riccobene E. MutRex: A Mutation Based Generator of Fault Detecting Strings for Regular Expressions // International Conference on Software Testing, Verification and Validation Workshops (ICSTW). – 2017. – 10 p.
4. Davis J.C., Moyer D., Kazerouni A., Lee D. Testing regex generalizability and its implications: A large-scale many-language measurement study // 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), San Diego, CA, USA. – 2019. – P. 427–439.
5. Louis G.M. IV, Donohue J., Davis J.C., Lee D., Servant F. Regexes are Hard: Decision-making, Difficulties, and Risks in Programming Regular ExpressionsSe // arXiv:2303.02555v12023. – 2023. https://doi.org/10.48550/arXiv.2303.02555.
6. Onyenwe I., Ogbonna S., Onyedimma E., Ikechukwu-Onyenwe O., Nwafor Ch. Developing Smart Web-Search Using RegEx // arXiv:2110.04767. – 2021. https://doi.org/10.48550/arXiv.2110.04767.
7. Frenz Ch.M. Introduction to Searching with Regular Expressions // arXiv:0810.1732. – 2008. https://doi.org/10.48550/arXiv.0810.1732.
8. Khan S.S., Madden M.G. A survey of recent trends in one class classification // Artificial Intelligence and Cognitive Science. AICS 2009. Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, Germany. – 2010. – Vol. 620. P. 188–197.
9. Schölkopf B., Platt J.C., Shawe-Taylor J., Smola A.J., Williamson R.C. Estimating the Support of a High-Dimensional Distribution // Neural Computation. – 2001. – Vol. 13. – No. 7. – P. 1443–1471.
10. Manevitz L.M., Yousef M. One-class SVMs for document classification // Journal of Machine Learning Research. – 2001. – Vol. 2. – P. 139–154.
11. Li K.-L., Huang H.-K., Tian S.-F., Xu W. Improving one-class SVM for anomaly detection // Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Xi'an, China, 2–5 November 2013. – P. 2–5.
12. Erfani S.M., Rajasegarar S., Karunasekera S., Leckie C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning //Pattern Recognition. – 2016. – Vol. 58. – P. 121–134.
13. Zhu F., Yang J., Gao C., Xu S., Ye N, Yin T. A weighted one-class support vector machine // Neurocomputing. – 2016. – Vol. 189. – P. 1–10.
14. Yang K., Kpotufe S., Feamster N. An Efficient One-Class SVM for Anomaly Detection in the Internet of Things // arXiv:2104.11146. – 2021.
15. Liu F.T., Ting K.M., Zhou Z.-H. Isolation Forest // 2008 Eighth IEEE International Conference on Data Mining. 2008. – P. 413–422.
16. Liu F. T., Ting K. M., Zhou Z.-H. Isolation-Based Anomaly Detection // ACM Transactions on Knowledge Discovery from Data (TKDD). – 2012. – Vol. 6. – Issue 1. – Article No. 3. – P 1–39.
17. Chabchoub Y., Togbe M.U., Boly A., Chiky R. An In-Depth Study and Improvement of Isolation Forest // IEEE Access. – 2022. – Vol. 10. – P. 10219–10237.
18. Xu H., Pang G., Wang Y., Wang Y. Deep Isolation Forest for Anomaly Detection // IEEE Transactions on Knowledge and Data Engineering. – 2023. – Vol. 35. – No. 12. – P. 12591–12604.
19. Leveni F., Cassales G.W., Pfahringer B., Bifet A., Boracchi G. Online Isolation Forest // ICML'24: Proceedings of the 41st International Conference on Machine Learning. – 2024. – Article No. 1088. – P. 27288–27298.
20. Breunig M.M., Kriegel H.-P., Ng R.T., Sander J. LOF: identifying density-based local outliers // SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. – 2000. P. 93–104.
21. Alghushairy O., Alsini R., Soule T., Ma X. A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams // Big Data and Cognitive Computing. – 2021. – Vol. 5. – No. 1. https://doi.org/10.3390/bdcc5010001/.
22. Riego N.Ch.R., Villarba D.B. Utilization of Multinomial Naive Bayes Algorithm and Term Frequency-Inverse Document Frequency (TF-IDF Vectorizer) in Checking the Credibility of News Tweet in the Philippines // arXiv:2306.00018. – https://doi.org/10.48550/arXiv.2306.00018.

23. Qader W.A., Ameen M.M., Ahmed B.I. An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges // 2019 International Engineering Conference (IEC), Erbil, Iraq. – 2019. – P. 200–204.
24. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // arXiv:1810.04805. – 2018. https://doi.org/10.48550/arXiv.1810.04805.
25. Warner B., Chaffin A., Clavié B., Weller O., Hallström O., Taghadouini S., Gallagher A., Biswas R., Ladhak F., Aarsen T., Cooper N., Adams G., Howard J., Poli I. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference // arXiv:2412.13663. – 2024. https://doi.org/10.48550/arXiv.2412.13663.
26. Finally, a Replacement for BERT [Электронный ресурс]. – Режим доступа: https://huggingface.co/blog/modernbert, свободный (дата обращения: 20.12.2024).
27. OLMo [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/model_doc/olmo, свободный (дата обращения: 20.12.2024).
28. Vapnik V. The nature of statistical learning theory. 2nd edition. Wiley. – 1999. – 334 p.
29. Kuo N.-H., Chiang T.-W., Wong R. SVM/SVR Kernels as Quantum Propagators // arXiv:2502.11153. – 2025. https://doi.org/10.48550/arXiv.2502.11153.
30. Lin J.-T., Lin C.-J. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods // Neural Computation. – 2004. – Vol. 16. – No.5. – P. 1071–1090.
31. Gorchakov A.V., Demidova L.A., Sovietov P.N. Analysis of Program Representations Based on Abstract Syntax Trees and Higher-Order Markov Chains for Source Code Classification Task // Future Internet. – 2023. – Vol. 15. – P. 314. https://doi.org/10.3390/fi15090314.
32. OneClassSVM [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html, свободный (дата обращения: 20.12.2024).
33. IsolationForest [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html, , свободный (дата обращения: 20.12.2024).
34. BERT [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/model_doc/bert, свободный (дата обращения: 20.12.2024).
35. ModernBERT [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/model_doc/modernbert, свободный (дата обращения: 20.12.2024).
36. RegexEval [Электронный ресурс]. – Режим доступа: https://huggingface.co/datasets/s2e-lab/RegexEval, свободный (дата обращения: 20.12.2024).
37. Regular Expressions 101 [Электронный ресурс]. – Режим доступа: https://regex101.com/, свободный (дата обращения: 20.12.2024).
38. Padding and truncation [Электронный ресурс]. – Режим доступа: https://huggingface.co/docs/transformers/pad_truncation, свободный (дата обращения: 20.12.2024).
39. McInnes L., Healy J., Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction // arXiv:1802.03426. – 2018. https://doi.org/10.48550/arXiv.1802.03426.
40. Leng Q., Qi H., Miao J., Zhu W., Su G. One-Class Classification with Extreme Learning Machine // Mathematical Problems in Engineering. – 2015. – Vol. 2015. Article No 412957, 11 p. https://doi.org/10.1155/2015/412957.
41. Li Y., Zhang S., Yin, Y., Zhang J.Parallel one-class extreme learning machine for imbalance learning based on Bayesian approach // Journal of Ambient Intelligence and Humanized Computing. – 2024. – Vol. 15. – P. 1745–1762.
42. Wang X., Jin B., Du Y., Cui P., Tan Y., Yang Y. One-class graph neural networks for anomaly detection in attributed networks // Neural Computing and Applications – 2021. – Vol. 33. – P. 12073–12085.