DOI: 10.25559/SITITO.020.202401.34-47
УДК 004.93'14

Оригинальная статья

Выявление аномалий в поведении ЦП с применением алгоритмов кластеризации библиотеки Scikit-Learn языка программирования Python

А. С. Турашев*, В. А. Сухомлин
ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова», г. Москва, Российская Федерация
Адрес: 119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1
* turashev.artem@mail.ru

Аннотация
Современные компьютерные системы становятся все более сложными. Они предоставляют нам множество возможностей и удобств, однако иногда аномалии в системе могут негативно сказаться на работе компьютера. В таком случае вопрос обнаружения аномалий стоит остро, так как вовремя обнаруженная аномальная активность может предотвратить кибератаку. В данной статье исследуется проблема выявления аномалий в работе центрального процессора (ЦП) с использованием алгоритмов кластеризации временных рядов. Центральный процессор - это основной вычислительный компонент, отвечающий за выполнение инструкций и обработку данных. Аномалии в работе ЦП могут приводить к сбоям системы, снижению производительности и другим негативным последствиям. Для решения этой проблемы предлагается применение алгоритмов кластеризации, которые позволяют выявить аномалии на основе анализа временных рядов, представляющих поведение ЦП. В статье представлен обзор существующих алгоритмов кластеризации временных рядов и их применение к задаче выявления аномалий в работе ЦП. Рассматриваются классические методы кластеризации библиотеки Scikit-Learn языка программирования Python, такие как KMeans, DBSCAN, Agglomerative Clustering и Affinity Propagation. Для оценки эффективности предложенных алгоритмов используются различные метрики качества, такие как ARI, AMI, Homogeneity Score, Completeness score, V – measure и Silhouette score. Проводятся эксперименты на реальных данных, полученных из систем мониторинга работы ЦП, чтобы оценить производительность и сравнить результаты различных алгоритмов. Выявление аномалий ЦП является важной задачей, которая помогает улучшить качество компьютерных систем. Понимание причин аномалий ЦП и имеющихся решений позволяет решить проблемы, связанные с работой процессора. Это в свою очередь способствует повышению производительности, стабильности и надежности компьютерных систем.

Ключевые слова: выявление аномалий, центральный процессор, алгоритмы кластеризации, временные ряды, метрики качества

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Турашев А. С., Сухомлин В. А. Выявление аномалий в поведении ЦП с применением алгоритмов кластеризации библиотеки Scikit-Learn языка программирования Python // Современные информационные технологии и ИТ-образование. 2024. Т. 20, № 1. С. 34-47. https://doi.org/10.25559/SITITO.020.202401.34-47

Original article

Detecting Anomalies in CPU Behavior Using Clustering Algorithms from the Scikit-Learn Library in Python Programming Language

A. S. Turashev*, V. A. Sukhomlin
Lomonosov Moscow State University, Moscow, Russian Federation
Address: 1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation
* turashev.artem@mail.ru

Abstract

Modern computer systems are becoming increasingly complex. They provide us with many features, but sometimes anomalies in the system can negatively affect computer performance. In this case, the issue of anomaly detection is acute, since anomalous activity detected in time can prevent a cyber attack. This article examines the problem of detecting anomalies in central processing unit (CPU) operation using time series clustering algorithms. The central processing unit is the main computing component responsible for executing instructions and processing data. Anomalies in CPU operation can lead to system crashes, reduced performance, and other negative consequences. To solve this problem, the usage of clustering algorithms is proposed, which allow identifying anomalies based on the time series analysis representing the behavior of the CPU. The article presents an overview of existing time series clustering algorithms and their application to the problem of identifying anomalies in CPU operation. Classic clustering methods of the Scikit-Learn library in Python programming language are considered, such as KMeans, DBSCAN, Agglomerative Clustering and Affinity Propagation. To evaluate the effectiveness of the proposed algorithms, various quality metrics are used, such as ARI, AMI, Homogeneity Score, Completeness score, V – measure and Silhouette score. Experiments are conducted on real data obtained from CPU monitoring systems to evaluate the performance and compare the results of different algorithms. Detecting CPU anomalies is an important task that helps improve the quality of computer systems. Understanding the causes of CPU anomalies and the available solutions can help you solve problems related to processor performance. This can help improve the performance, stability and reliability of computer systems.

Keywords: anomaly detection, central processing unit, clustering algorithms, time series, quality metrics

Conflict of interests: The authors declare no conflict of interest.

For citation: Turashev A.S., Sukhomlin V.A. Detecting Anomalies in CPU Behavior Using Clustering Algorithms from the Scikit-Learn Library in Python Programming Language. *Modern Information Technologies and IT-Education.* 2024;20(1):34-47. https://doi.org/10.25559/SITITO.020.202401.34-47

Введение

Выявление аномалий в поведении центрального процессора (ЦП) является непростой задачей в сфере анализа данных. Это вызвано несколькими факторами, включая сложность моделирования поведения ЦП, наличие различных типов аномалий и постоянно меняющуюся природу работы ЦП. Для эффективного обнаружения аномалий требуется применение специализированных методов, аналитических подходов и алгоритмов, которые могут учитывать сложности и особенности ЦП и его работы [1-9].

Цель исследования

Целью данной работы является проведение сравнительного анализа алгоритмов кластеризации (KMeans, DBSCAN, Agglomerative Clustering и Affinity Propagation) для выявления основных параметров, по которым эти алгоритмы различаются, и применение каждого из этих алгоритмов для выявления аномалий с дальнейшей оценкой их эффективности с использованием основных метрик качества.

Что такое кластеризация

Кластеризация – это разбиение элементов некоторого множества на группы на основе их схожести. Задача кластеризации состоит в разбиении объектов из X на несколько подмножеств (кластеров), в которых объекты более схожи между собой, чем с объектами из других кластеров.

Кластеризацию часто сравнивают с классификацией, которая представляет собой схожую процедуру. Разница заключается в том, что при классификации результирующее множество групп четко определено, тогда как при кластеризации оно определяется самим алгоритмом в процессе его работы.

Кластеризация включает в себя следующую последовательность действий:

- Выбор множества объектов.
- Определение множества переменных для оценки объектов и составление векторов характеристик.
- Нормирование векторов характеристик одним из доступных методов.
- Определение сходства между объектами по заданной метрике.
- Применение выбранного метода кластерного анализа для разбиения множества объектов на кластеры по их степени схожести.
- Представление результатов анализа.

Объектом называется элементарный набор данных, с которым работает сам алгоритм кластеризации. Для каждого объекта определяются параметры, которые описывают этот объект. Они объединяются в некоторый вектор характеристик: $x = (x_1, x_2, \dots, x_m)$, где m – размерность пространства характеристик, а x_i – отдельная характеристика объекта (качественная или количественная). Мера сходства $d(u, v)$ двух объектов u и v, которая вычисляется по заданной метрике, называется расстоянием между объектами.

Существуют различные метрики для вычисления близости объектов. Опишем некоторые из них:

- Евклидово расстояние. Классическая метрика, которая является геометрическим расстоянием в многомерном пространстве. Вычисляется по следующей формуле:

$$d(u, v) = \sqrt{\sum_{i=1}^{m} (u_i - v_i)^2}$$

- Квадрат евклидова расстояния. Данная метрика используется в основном для увеличения веса более отдаленных друг от друга объектов и вычисляется так:

$$d(u, v) = \sum_{i=1}^{m} (u_i - v_i)^2$$

- Расстояние городских кварталов (манхэттенское расстояние). Вычисляется как средняя разность по координатам и часто приводит к аналогичным результатам при использовании евклидова расстояния.

$$d(u, v) = \sum_{i=1}^{m} |u_i - v_i|$$

- Степенное расстояние. Применяется при необходимости изменить вес в большую или меньшую сторону. Вычисляется по следующей формуле:

$$d(u, v) = \sqrt[r]{\sum_{i=1}^{m} (u_i - v_i)^p}$$

где r, p – параметры, которые определяются самим пользователем. Если значения параметров p и q равны 2, то данная метрика целиком совпадает с расстоянием Евклида.

- Расстояние Чебышева. Мера, применяемая ввиду необходимости определить два объекта как различные по какой-то одной конкретной координате. Расстояние Чебышева вычисляется по следующей формуле:

$$d(u, v) = \max(|u_i - v_i|)$$

Стоит отметить, что выбор конкретной метрики влияет на результаты кластеризации. Для различных метрик результаты кластеризации могут иметь существенные различия [10, 11].

На большом пространстве характеристик процесс кластеризации довольно медленный, и его результаты не всегда приемлемы. Поэтому, если размерность пространства признаков велика, то следует попытаться уменьшить его, оставив наиболее важные признаки у объекта.

Получившийся набор характеристик каждого объекта нужно нормализовать для улучшения конечных результатов. Нормализация вектора – это процедура приведения его к определенному фиксированному размеру.

Формальная постановка задачи

Требуется разбить конечную выборку объектов X на K непересекающихся подмножеств

$$S_k, k = 1, \dots, K;\ X = \bigcup_{k=1}^{K} S_k$$

называемых кластерами, так чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались друг от друга. При этом каждому объекту $x^i \in X$ выделяется соответствующий номер кластера $y_i \in Y$. А алгоритм кластеризации – это функция $X \to Y$, которая любому объекту $x \in X$ ставит в соответствие номер кластера $y \in Y$. Множество Y в некоторых случаях известно заранее, но чаще всего на практике ставится задача определить наиболее оптимальное число кластеров.

Также необходимо ввести понятия, которые определяют сам кластер. Центром (или центроидом) кластера S_k называется геометрический центр точек k– го кластера в евклидовом пространстве:

$$X_k = \frac{1}{|S_k|} \sum_{x^i \in S_k} x^i$$

где $|S_k|$ – число точек в k – ом кластере, $k = 1, \dots, K$, K – это число кластеров.

Радиусом кластера S_k (или мера рассеяния точек относительно центра кластера) – это максимально возможное расстояние до центра кластера, которое рассчитывается так:

$$R_k = \max_{x^i \in S_k} \rho(x^i, X_k)$$

Описание алгоритмов кластеризации

KMeans

Одним из самых простых и эффективных алгоритмов кластеризации является алгоритм k-means (Mac-Queen, 1967) или k-средних (от слова *mean*). Этот алгоритм состоит из четырех шагов, где сначала определяется количество k кластеров, которое должно быть сформированы из исходной выборки объектов:

1. k записей из исходной выборки случайным образом выбираются в качестве начальных центров кластеров μ. Именно из этих начальных точкек, часто называемых «семенами», «растет» кластер.
2. Для каждого центройда μ_k необходимо вычислить расстояния $l_k^{(i)}$ до всех точек $x^{(i)}$

$$l_k^{(i)} = \rho(x^{(i)}, \mu_k)$$

где ρ – выбранная нами метрика (обычно используется евклидово расстояние).

3. Сформировать кластеры, для каждого центройда μ_k из множества X отобрать подмножество точек X_k с минимальным расстоянием до μ_k

$$\min_k l_k^{(i)}$$

4. Произвести вычисление всех новых центройдов $\mu_k = \frac{1}{S_k} \sum X_k$, где S_k – количество точек в кластере K.

Шаги 3 и 4 повторяются до тех пор, пока алгоритм не прекратит работу или не будет выполнено условие в соответствии с определенными критериями сходимости.

Алгоритм останавливается, когда границы кластеров и положения центроидов не перестают меняться от одной итерации к другой, т.е. когда в каждом кластере на каждой итерации остается один и тот же набор записей. На практике алгоритм KMeans обычно может найти набор стабильных кластеров в течение нескольких десятков итераций.

Что касается критерия сходимости, то наиболее часто используемым критерием является сумма квадратов ошибок между центройдом кластера и всеми элементами, которые вошли в него, то есть:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2$$

где $p \in C_i$ – некоторая произвольная точка данных, которая принадлежит кластеру C_i, а m_i – центройд данного кластера. То есть, алгоритм k-means останавливается тогда, когда ошибка E достигает достаточно малого значения.

Основными преимуществами KMeans являются:

- данный алгоритм прост для понимания и реализации
- это эффективный алгоритм, особенно для больших наборов данных
- он относительно устойчив к шуму, что означает, что он может работать хорошо, даже если данные содержат некоторые выбросы

Что касается недостатков, то они здесь следующие:

- сам алгоритм чувствителен к начальным центроидам кластера. Это означает, что результаты алгоритма могут значительно различаться в зависимости от того, как инициализируются центроиды
- Подходит не для всех типов данных. Лучше всего подходит для данных, которые являются непрерывными и сферическими
- Он может создавать кластеры произвольной формы и размера. Это может затруднить интерпретацию результатов алгоритма [12].

DBSCAN

Кластеры определяются плотностью точек. Области с высокой плотностью точек означают наличие кластеров, в то время как области с низкой плотностью точек являются кластерами шума или кластерами выбросов. Данный алгоритм подходит для работы с большими наборами данных, с шумом и позволяет идентифицировать кластеры различных форм и размеров. Основная идея алгоритма DBSCAN заключается в том, что для каждой точки кластера в окрестности заданного радиуса должно лежать как минимум минимальное количество точек, то есть плотность в окрестности должна превышать некоторый заранее определенный порог.

DBSCAN определяет кластеры как плотные области точек данных, разделенные более разреженными областями. Он опреде-

ляет окрестность вокруг каждой точки данных и расширяет ее до тех пор, пока в окрестности не будет лежать как минимум минимальное количество точек. Алгоритм определяет точки данных как основные точки (достаточно плотные области), граничные точки (близкие к основным точкам, но с меньшей плотностью) или шумовые точки (изолированные точки с недостаточной плотностью). DBSCAN не требует заранее определенного числа кластеров и может работать с кластерами любой формы. У этого алгоритма есть три входных параметра:

- eps - радиус, ограничивающий область соседства точки (окрестность eps)
- min_samples, минимальное количество точек, которые должны лежать в окрестности eps
- метрика, которая будет использоваться (например, евклидова или манхэттенская метрика)

Теперь про основные преимущества и недостатки данного алгоритма.

Достоинства:

- DBSCAN не требует спецификации числа кластеров в данных априори в отличие от метода k-means
- DBSCAN может найти кластеры произвольной формы. Он может найти даже кластеры полностью окружённые (но не связанные с) другими кластерами. Благодаря параметру уменьшается так называемый эффект одной связи (связь различных кластеров тонкой линией точек).
- DBSCAN имеет понятие шума и устойчив к выбросам
- DBSCAN требует лишь двух параметров и большей частью нечувствителен к порядку точек в базе данных

Недостатки:

- Если в наборе данных есть записи данных разной плотности, то он терпит неудачу, потому что eps и min_samples нельзя выбрать отдельно для каждого из кластеров
- Настроить eps и min_samples очень сложно
- DBSCAN является не полностью однозначным — краевые точки, которые могут быть достигнуты из более чем одного кластера, могут принадлежать любому из этих кластеров, что зависит от порядка просмотра точек

Agglomerative Clustering

Одним из первых ученых, который предпринял такой подход к данным, был Карл Линней, который пытался по своей объемной картотеки растений и животных объединить их по родам и видам. Полученные группы объектов, близких по определенным признакам, также называют **таксонами**. Фактически, таксон и кластер – это одно и то же и в описании этого алгоритма эти слова будут использоваться как синонимы.

Изначально мы имеем набор данных и каждый объект – это независимый кластер. В соответствии с метрикой, которая задана между объектами: $p(x_i, x_j)$, $i, j = 1, \ldots, l$ будем выбирать два ближайших таксона: U, и объединяем их в один W: $W = U \cup V$.

Теперь у нас появилась новая группа из двух объектов, и наша задача вычислить расстояния от неё до всех остальных объектов выборки. Вариантов как производить эти вычисления очень много, но все их можно описать одной математической формулой:

$R_{WS} = a_U R_{US} + a_V R_{VS} + \beta R_{UV} + \gamma |R_{US} - R_{VS}|$ – формула Ланса-Уильямса

где a_U, a_V, β, γ – параметры, которые определяют вид метрики между кластерами.

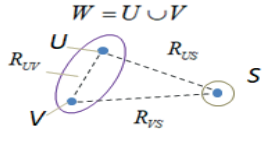
Таблица 1. **Вычисление параметров по расстоянию между кластерами**
Table 1. **Calculation of parameters based on the distance between clusters**

Расстояние ближнего соседа	$a_U = a_V = \frac{1}{2}, \beta = 0, \gamma = -\frac{1}{2}$
Расстояние дальнего соседа	$a_U = a_V = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}$
Групповое среднее расстояние	$a_U = \frac{\lvert U\rvert}{\lvert W\rvert}, a_V = \frac{\lvert V\rvert}{\lvert W\rvert}, \beta = \gamma = 0$
Расстояние между центрами	$a_U = \frac{\lvert U\rvert}{\lvert W\rvert}, a_V = \frac{\lvert V\rvert}{\lvert W\rvert}, \beta = -a_U \cdot a_V, \gamma = 0$
Расстояние Уорда	$a_U = \frac{\lvert S\rvert + \lvert U\rvert}{\lvert S\rvert + \lvert W\rvert}, a_V = \frac{\lvert S\rvert + \lvert V\rvert}{\lvert S\rvert + \lvert W\rvert}, \beta = -\frac{\lvert S\rvert}{\lvert S\rvert + \lvert W\rvert}, \gamma = 0$

Источник: здесь и далее в статье все таблицы и рисунки составлены авторами.
Source: Hereinafter in this article all tables and figures were made by the authors.

Стоит отметить тот факт, что на практике часто используется расстояние дальнего соседа или расстояние Уорда.

Покажем принцип работы формулы Ланса – Уильямса на одном объединенном таксоне с двумя объектами и второго таксона – с одним объектом:



Р и с. 1. Вспомогательная иллюстрация принципа работы формулы Ланса – Уильямса
F i g. 1. An auxiliary illustration of the working principle of the Lance-Williams formula

Теперь предположим, что мы выбрали расстояние ближайшего соседа, для которого: $a_U = a_V = \frac{1}{2}$, $\beta = 0, \gamma = -\frac{1}{2}$ и вычисление расстояния R_{WS} происходит следующим образом: $R_{WS} = \frac{1}{2}R_{US} + \frac{1}{2}R_{VS} - \frac{1}{2}|R_{US} - R_{VS}|$. Если здесь раскрыть модуль, то мы получим следующее:

$$R_{WS} = \begin{cases} \frac{1}{2}R_{US} + \frac{1}{2}R_{VS} - \frac{1}{2}R_{US} + \frac{1}{2}R_{VS} = R_{VS}, & R_{VS} < R_{US} \\ \frac{1}{2}R_{US} + \frac{1}{2}R_{VS} + \frac{1}{2}R_{US} - \frac{1}{2}R_{VS} = R_{US}, & R_{US} < R_{VS} \end{cases}$$

а это есть не что иное, как выбор минимального расстояния между точками таксонов, то есть мы нашли расстояние между таксоном W и таксоном S.

Затем, когда нам необходимо будет произвести объединение одного таксона из двух объектов с другим таксоном, то при вычислении расстояния объединенного кластера W мы будем использовать ранее вычисленное R_{VS} для получения R_{WS}. В силу рекуррентного вычисления расстояний между объединяемыми таксонами, мы имеем полную информацию о расстояниях между любыми парами сформированных кластеров на каждом шаге нашего алгоритма.

Таким образом, алгоритм агломеративной иерархической кластеризации принимает следующий вид:

Изначально нам дано множество одноэлементных кластеров $C_1 = \{\{x_1\}, \ldots, \{x_l\}\}$ и также задана метрика между отдельными объектами: $R_{\{x_i\},\{x_j\}} = p(x_i, x_j)$

1. $\forall t = 2, \ldots, l$, где t – номер итерации найти в C_{t-1} пару кластеров U и V с минимальным расстоянием R_{UV}
2. Объединить их в один кластер:

$W = U \cup V$

$C_t = C_{t-1} \cup W$

3. $\forall S \in C_t$ вычислить R_{WS} по формуле Ланса – Уильямса:

$$R_{WS} = a_U R_{US} + a_V R_{VS} + \beta R_{UV} + \gamma |R_{US} - R_{VS}|$$

Основные преимущества алгоритма Agglomerative Clustering:

- При кластеризации могут использоваться данные с различными типами и размерами кластеров
- Количество кластеров не обязательно должно быть заранее определено при использовании агломеративной иерархической кластеризации
- Шум и выбросы не являются проблемой для работы данного алгоритма

Какие недостатки стоит отметить:

- Для больших наборов данных агломеративная иерархическая кластеризация может быть дорогой с точки зрения вычислений
- Последовательность, в которой происходит объединение кластеров, может повлиять на конечный результат
- Выбор метрики между кластерами также может повлиять на работу алгоритма

Affinity Propagation

Идея данного алгоритма заключается в том, чтобы наши наблюдения кластеризовались на основе их сходства или того факта, как они "общаются". В отличие от алгоритма К-средних, данный подход не требует заранее определять число кластеров, на которое мы хотим разбить наши данные.

Теперь опишем сам процесс «общения». Заведем две матрицы (r – матрица ответственности и a – матрица доступности), которые мы изначально инициализируем нулями, одна из которых ($r_{i,k}$) описывает то, насколько хорошо k–ое наблюдение подходит для того, чтобы быть неким «примером для подражания» для i–го наблюдения относительно всего остального, а вторая матрица ($a_{i,k}$) описывает то, насколько правильным было бы для i–го наблюдения выбрать в качестве такого «примера для подражания» k–ое.

Введём f – функцию, которая вычисляет сходство между двумя точками. Например, $f(i, j) > f(i, k)$ означает, что x_i более сходно с x_j, чем с x_k. Примем $f(i, j) = -\|x_i - x_j\|^2$ – отрицательный квадрат евклидова расстояния. Значения $f(i, j)$, вычисленные для всех N точек, можно представить в виде матрицы s с размерами $N \times N$. При этом диагональные элементы матрицы s будут указывать на то, насколько вероятно, что объект x_i станет «примером для подражания».

Если на этапе инициализации сделать следующую инициализацию: $f(i, i) = q$ для всех i, то с помощью этой величины можно будет определять число кластеров. Меньшее значение q даст в итоге меньшее число кластеров. Обычно значение q определяют как медиану значений $f(i, j)$, $i \neq j$, вычисленных для все точек.

Таким образом, $s = f + q \times I_n$, где $f = f(i, j) = -\|x_i - x_j\|^2$, q – величина предпочтения, I_n – единичная матрица порядка .

После всего этого данные наших матриц будут использованы для следующих действий:

Выполнять заданные шаги T количество раз:

1. Обновить значения матрицы r:

$$r[i, j] = (1 - \lambda)p[i, j] + \lambda r[i, j]$$

где $p[i, j]$ – распространяемая ответственность, которая может быть вычислена следующим образом:

$$p[i, j] = \begin{cases} s[i, j] - max_{k \neq j}\{a[i, k] + s[i, k]\}, & i \neq j \\ s[i, j] - max_{k \neq j}\{s[i, k]\}, & i = j \end{cases}$$

2. Обновить значения матрицы a:

$$a[i, j] = (1 - \lambda)\gamma[i, j] + \lambda a[i, j]$$

где $\gamma[i, j]$ – распространяемая доступность, которая может быть вычислена следующим образом:

$$\gamma[i, j] = \begin{cases} min\left(0, r[j, j] + \sum_{k \neq i,j} \max(0, r[k, j])\right), & i \neq j \\ \sum_{k \neq i,j} max(0, r[k, j]), & i = j \end{cases}$$

где λ – коэффициент затухания, значения которого выбираются из интервала [0,1). Цель его введения в вычисления – избежать численные колебания.

Стоит отметить, что авторы рекомендуют первоначально использовать значение λ = 0,5, а в случае, если алгоритм не будет сходиться, увеличить значение λ до 0,9 – 0,95. Но при таком резком увеличении значения λ число итераций тоже увеличится соответственно.

3. Вычислить образцы. Образцами считаются точки, которые удовлетворяют условию:

$$r[i, i] + a[i, i] > 0$$

4. Отнести объект к кластеру с номером:

$$c_i = \underset{k}{argmax}(r[i,k] + a[i,k])$$

То есть номер кластера для объекта x_i определяется номером столбца максимального элемента в i – ой строке матрицы $r + a$.

Итерации выполняются до тех пор, пока либо состав кластеров не будет меняться на протяжении нескольких итераций, либо достигнуто некоторое заданное число итераций[1] [13-16].

Сильными сторонами данного алгоритма являются следующие аспекты:

- пользователю не нужно указывать количество кластеров заранее
- модель не чувствительна к начальному значению данных
- нет требования к симметрии исходных данных матрицы подобия

Слабые стороны Affinity Propagation

- признаваемым всеми недостатком алгоритма AP является больший (в сравнении с другими алгоритмами) объем памяти, необходимый для работы, особенно для кластеризации больших наборов данных.
- на качество кластеризации влияют параметры q и λ, введенные нами выше.

Таблица 2. **Основные свойства алгоритмов кластеризации KMeans, DBSCAN, Agglomerative Clustering и Affinity Propagation библиотеки Scikit-Learn**
Table 2. **Key properties of the KMeans, DBSCAN, AgglomerativeClustering and AffinityPropagation clustering algorithms from the Scikit-Learn library**

№ алгоритма	Название алгоритма	Основные параметры алгоритма (входные значения)	Форма кластеров	Выход алгоритма	Сложность алгоритма
1	KMeans	n_clusters - количество формируемых кластеров, а также количество центроидов для генерации	Сферическая	Метки кластеров. Центроиды кластеров. Сумма квадратов расстояний – метрика, показывающая насколько точки данных внутри каждого кластера близки друг к другу.	$O(nkld)$, где n - количество точек данных, k - количество кластеров, l - количество итераций алгоритма, d - размерность пространства признаков
2	DBSCAN	eps - радиус, ограничивающий область соседства точки (eps – окрестность) min_samples - минимальное количество точек, которые должны лежать в окрестности eps metric - метрика, которая будет использоваться	Произвольная	Метки кластеров Количество кластеров Краевые точки – точки, которые находятся на границе между кластерами Шум	$O(n^2)$, где n - количество точек данных
3	Agglomerative Clustering	n_clusters linkage type – методы объединения точек (single linkage - минимум попарных расстояний, complete linkage - максимум попарных расстояний, average linkage - среднее попарных расстояний, centroid linkage - расстояние между центроидами двух кластеров)	Дендрограмма	Метки кластеров. Количество кластеров. Расстояния между объединяемыми кластерами.	$O(n^3)$, где n - количество точек данных
4	Affinity Propagation	damping – коэффициент затухания (значения которого выбираются из интервала) max_iter – максимальное число итераций алгоритма preference – параметр, который определяет, насколько каждая точка данных предпочитает стать экземпляром центра кластера. Он также влияет на количество кластеров, которые будут образованы	Произвольная	Метки кластеров. Экземпляры кластеров – точки, которые представляют тот или иной кластер. Количество кластеров.	$O(n^2Tl)$, где n - количество точек данных, а T - количество итераций, необходимых для сходимости алгоритма, l - среднее количество соседей, с которыми каждая точка данных сравнивается на каждой итерации

[1] Виноградова Д. А. Исследование применимости алгоритмов кластеризации Affinity propagation, DBSCAN к решению задачи поиска пользователей-экспертов в социальных сетях // Молодежь и современные информационные технологии : Сборник трудов XVI Международной научно-практической конференции студентов, аспирантов и молодых ученых. Томск : ТПУ, 2019. С. 160-161. EDN: CIZQUJ

Метрики качества кластеризации

Метрика качества кластеризации - это числовая мера, которая используется для оценки качества результатов кластеризации. Она помогает определить, насколько хорошо алгоритм кластеризации разделил данные на группы схожих объектов. Метрики качества кластеризации могут оценивать различные аспекты, такие как компактность кластеров, разделение между кластерами и шумовые точки.

Обычно выделяют два подхода для оценки качества результатов кластеризации – внутренние и внешние. Внутренние метрики качества кластеризации оценивают компактность и однородность кластеров, основываясь только на внутренних характеристиках данных. Они не требуют наличия заранее известных классов или меток. Эти метрики позволяют оценить, насколько хорошо точки данных внутри каждого кластера сгруппированы вместе. Внешние метрики качества кластеризации оценивают результаты кластеризации, используя ифнормацию об истинном разбиении на кластеры. Они требуют наличия эталонных меток для оценки качества кластеризации. Эти метрики позволяют оценить, насколько хорошо кластеры соответствуют эталонным меткам или истинным классам данных.

Adjusted Rand Index (ARI)

Индекс Adjusted Rand (ARI) - это метрика, используемая для оценки эффективности кластеризации путем сравнения полученных кластеров с эталонными метками или истинными классами данных.

Предположим, что – число объектов в выборке, – число пар объектов, которые находятся в одном кластере и, соответственно, имеют одинаковые метки, – число пар объектов, которые находятся в разных кластерах и, соответственно, имеют разные метки. Тогда

$$RI = \frac{2(a+b)}{n(n-1)}$$

Величина Rand Index (RI) показывает меру сходства между двумя разбиениями данных, например, между результатами кластеризации и эталонными метками. RI измеряет долю пар точек данных, которые были классифицированы одинаково в обоих разбиениях, относительно всех возможных пар. Для того, чтобы этот индекс принимал значения близкие к нулю при любом и числе кластеров для случайных кластеризаций, необходимо нормировать этот индекс. В этом случае определяется ARI (или Adjusted Rand Index)

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

где *RI* - значение Rand Index, а *E[RI]* - ожидаемое значение Rand Index, которое будет получено случайным образом.

ARI принимает значения от -1 до 1. Значение 1 означает идеальное согласование между разбиениями, значение 0 указывает на случайное согласование, а отрицательное значение указывает на согласование, которое хуже случайного разбиения. Чем ближе значение ARI к 1, тем лучше сходство между разбиениями, учитывая случайное согласование.

Adjusted Mutual Information (AMI)

Индекс настроенной взаимной информации (Adjusted Mutual Information, AMI) – это мера сходства между двумя кластерными разбиениями. Он учитывает случайное совпадение, которое может возникнуть при случайном разделении данных на кластеры. AMI вычисляется по следующей формуле:

$$AMI(A,B) = \frac{MI(A,B) - E[MI(A,B)]}{avg\big(H(A),H(B)\big) - E[MI(A,B)]}$$

где *A* и *B* – два кластерных разбиения, *MI(A,B)* – индекс взаимной информации между *A* и *B*, *H(A)* и *H(B)* – энтропии кластерных разбиений *A* и *B*, *avg()* – функция, вычисляющая среднее арифметическое для этих двух энтропий, *E[MI(A,B)]* – ожидаемое значение индекса взаимной информации при случайном разделении данных на кластеры.

$$MI(A,B) = \sum\sum P(A,B) * \log\left(\frac{P(A,B)}{P(A)P(B)}\right)$$

где *P(A,B)* – совместная вероятность появления *A* и *B*, *P(A)* и *P(B)* – вероятности появления *A* и *B* независимо. MI принимает значения от 0 до бесконечности, где 0 означает полное отсутствие зависимости, а более высокие значения указывают на более сильную зависимость между переменными.

AMI принимает значения от 0 до 1, где 0 означает случайное разделение, а 1 указывает на полное совпадение между кластерными разбиениями. Более высокие значения AMI указывают на более сильное сходство между кластерными разбиениями.

Homogeneity score

Оценка однородности (Homogeneity score) используется не только для оценки качества кластеризации, но и для настройки параметров алгоритма. Высокое значение оценки однородности указывает на хорошую гомогенность кластеров, где каждый кластер состоит из элементов одного класса. Низкое значение может указывать на смешение классов внутри кластеров или на неоднородность данных.

Вычисление оценки однородности происходит следующим образом:

1. Вычислить матрицу сопряженности, где строки соответствуют истинным меткам классов, а столбцы - меткам кластеров.
2. Для каждого кластера выбрать наиболее часто встречающуюся метку класса внутри кластера.
3. Суммировать наибольшие значения для каждого кластера и поделить на общее количество элементов данных.

Окончательное значение оценки однородности будет находиться в диапазоне от 0 до 1, где 1 указывает на идеальную однородность (или, как еще говорят, гомогенность), то есть каждый кластер содержит только элементы одного класса.

Completeness score

Оценка полноты (Completeness score) особенно полезна, когда у нас есть данные с истинными метками или классами, и мы хотим оценить, насколько хорошо алгоритм кластеризации справился с группировкой элементов данных по этим меткам. Высокое значение оценки полноты указывает на то, что все члены данного класса относятся к одному кластеру.

Вычисление Completeness score включает следующие шаги:
1. Для каждого кластера определите, сколько элементов внутри этого кластера принадлежит к каждому классу или метке истинной метки.
2. Для каждого кластера найдите класс или метку с максимальным количеством элементов внутри этого кластера.
3. Суммируйте количество элементов, принадлежащих к классу или метке с максимальным количеством элементов внутри каждого кластера.
4. Разделите полученную сумму на общее количество элементов данных.
Результатом будет значение Completeness score, которое может находиться в диапазоне от 0 до 1. Значение 1 указывает на идеальную полноту, когда каждый элемент внутри каждого кластера принадлежит к одному и тому же классу или метке истинной метки. Значение близкое к 0 указывает на низкую полноту, когда элементы внутри кластеров принадлежат к разным классам или меткам.

V-measure score
Оценка V-меры (V-measure score) – это метрика, комбинирующая оценку однородности (homogeneity) и оценку полноты (completeness) для получения общей оценки качества. В общем случае V-мера – это среднее гармоническое между однородностью и полнотой, и вычисляется она по следующей формуле:

$$v = \frac{(1+\beta) * homogeneity * completeness}{(\beta * homogeneity + completeness)}$$

Значения V-меры лежат в диапазоне от 0 до 1, где 1 означает идеальное соответствие кластеров и классов, а 0 означает полное отсутствие соответствия.

Silhouette score
Коэффициент силуэта (silhouette score) помогает определить, насколько хорошо объекты сгруппированы внутри своих кластеров и насколько они разделены от объектов в других кластерах. Его можно использовать для изучения расстояния разделения между полученными кластерами. Данный коэффициент определяется двумя величинами:
– среднее расстояние между образцом и всеми другими точками того же класса
– среднее расстояние между образцом и всеми другими точками в следующем ближайшем кластере

Допустимые значения коэффициента силуэта лежат на отрезке [-1,1]. В случае, если значение силуэта близко к 1, то это указывает на хорошее разделение и однородность объектов внутри своих кластеров, а также на хорошее разделение от объектов в других кластерах. Если же значение силуэта равно 0, то это говорит о том, что объекты могут быть граничными или находиться между двумя кластерами. Это может указывать на перекрытие кластеров или на нечеткость разделения. Близкое к -1 значение коэффициента силуэта сигнализирует о неправильном разделении объектов, когда объекты внутри кластеров могут быть более похожими на объекты из других кластеров, чем на объекты из своего собственного кластера.

Таблица 3. **Преимущества и недостатки метрик качества кластеризации ARI, AMI, Homogeneity score, Completeness score, V-measure score и Silhouette score**
Table 3. **Advantages and disadvantages of clustering quality metrics ARI, AMI, Homogeneity score, Completeness score, V-measure score and Silhouette score**

№ метрики качества кластеризации	Название метрики качества кластеризации	Значения метрики	Преимущества	Недостатки
1	Adjusted Rand Index (ARI)	принимает значения от -1 до 1	Независимость от количества кластеров. Учёт случайного согласования: ARI учитывает случайное согласование, что позволяет избежать завышенной оценки кластеризации только из-за случайного совпадения	Зависимость от случайного выбора: ARI может быть чувствителен к случайному выбору исходных кластеров, особенно в случае небольших наборов данных. Разные случайные инициализации могут привести к различным значениям ARI
2	Adjusted Mutual Information (AMI)	принимает значения от 0 до 1	Учет случайного согласования. Нормализация и симметричность: AMI нормализуется и имеет значение между 0 и 1, AMI также является симметричной метрикой, что означает, что результаты не зависят от порядка сравниваемых кластеризаций	Чувствительность к количеству кластеров: AMI может давать некорректные результаты, если количество кластеров в разных кластеризациях существенно отличается. Зависимость от случайного выбора

№ метрики качества кластеризации	Название метрики качества кластеризации	Значения метрики	Преимущества	Недостатки
3	Homogeneity score	принимает значения от 0 до 1	Простота интерпретации. Чувствительность к однородным кластерам: Homogeneity score хорошо работает, когда кластеры в кластеризации имеют высокую степень однородности, то есть когда точки данных внутри каждого кластера принадлежат только одному классу или метке	Неучет структуры кластеров. Ограничение на наличие меток: Homogeneity score требует наличия заранее известных меток классов
4	Completeness score	принимает значения от 0 до 1	Простота интерпретации Чувствительность к полным кластерам: Completeness score хорошо работает, когда кластеры в кластеризации полностью соответствуют классам или меткам, то есть все точки данных одного класса находятся в одном кластере	Неучет структуры кластеров. Ограничение на наличие меток.
5	V-measure score	принимает значения от 0 до 1	Учет как точности, так и полноты.	Чувствительность к выбору β-коэффициента: V-мера зависит от выбора β-коэффициента, который влияет на вес точности и полноты. Неправильный выбор значения β может привести к неправильной оценке качества кластеризации. Ограничение на наличие меток. V-мера может давать неправильные оценки, когда классы несбалансированы, то есть имеют существенные различия в количестве точек данных.
6	Silhouette score	принимает значения от -1 до 1	Простота интерпретации. Независимость от формы кластеров.	Неэффективность для больших наборов данных: вычисление Silhouette score требует вычисления расстояний между каждой парой точек данных, что может быть вычислительно затратным для больших наборов данных. Чувствительность к выбору метрики расстояния: Silhouette score может быть чувствителен к выбору метрики расстояния, особенно если данные имеют необычную структуру или выбросы. Silhouette score оценивает только внутрикластерное и межкластерное расстояния, но не учитывает другие аспекты, такие как плотность кластеров или их иерархическую структуру.

Экспериментальное исследование и полученные результаты

Попробуем применить вышеописанные алгоритмы кластеризации для того, чтобы выявить аномальное поведение ЦП локальной машины. В более общем случае данный эксперимент можно провести на любой локальной машине.

В самом начале потребуется установить Telegraf для сбора и InfluxDB дальнейшего сохранения информации, снятой с ЦП. InfluxDB – это распределенная база данных, специально разработанная для обработки и хранения временных рядов. Она предназначена для эффективного сбора, хранения и анализа данных, которые изменяются со временем, таких как данные с датчиков, метрики производительности, журналы событий и т. п. [17, 18]. InfluxDB обладает масштабируемостью и высокой производительностью, что позволяет обрабатывать большие объемы данных и обеспечивать быстрый доступ к ним. Она предоставляет SQL-подобный язык запросов для извлечения данных и поддерживает гибкую структуру данных, которая позволяет добавлять и изменять поля данных по мере необходимости. Что касается Telegraf, то он предоставляет широкий набор плагинов и интеграций, которые позволяют собирать различные метрики и данные о состоянии системы. В контексте сбора информации с ЦП, Telegraf может собирать следующую информацию: информацию о загрузке ЦП, проценте использования каждого ядра ЦП, количестве активных и ожидающих задач и других релевантных метриках производительности ЦП, данные о температуре ЦП, информацию о состоянии ЦП, такую как текущая тактовая частота, напряжение, энергопотребление и другие параметры, данные о кэше ЦП, включая размеры и статистику использования уровней кэша, а также другие метрики, связанные с ЦП, такие как информация о ядрах, потоках выполнения, прерываниях и т. д. [19].

Для обнаружения самих аномалий будет использоваться ADTK (Anomaly Detection Toolkit) - это инструментарий для обнаружения аномалий во временных рядах. Он предоставляет набор методов и алгоритмов, специально разработанных для анализа временных данных и выявления аномалий в них. Приведем несколько причин, почему ADTK может быть хорошим выбором для обнаружения аномалий:

- Разнообразие методов: ADTK предлагает несколько различных методов обнаружения аномалий, включая статистические, машинное обучение и гибридные подходы. Это позволяет выбрать наиболее подходящий метод в зависимости от типа данных и требований задачи.
- Автоматический выбор порогов: ADTK предоставляет возможность автоматического выбора порогов для обнаружения аномалий, основываясь на статистических свойствах данных. Это упрощает процесс настройки модели и делает его более универсальным для различных временных рядов.
- Гибкость и настраиваемость: ADTK позволяет настраивать параметры методов обнаружения аномалий, чтобы лучше соответствовать конкретным требованиям и особенностям данных. Это позволяет достичь более точного и надежного обнаружения аномалий.
- Интеграция с другими инструментами: ADTK может быть легко интегрирован с другими инструментами и библиотеками Python, такими как Pandas, NumPy и Scikit-learn. Это обеспечивает удобство использования и расширяемость для дополнительного анализа данных и визуализации результатов [20, 21].

Также для работы с данными, их анализа и визуализации нам потребуются несколько библиотек: библиотека scikit-learn применяется для загрузки и обработки наборов данных, а библиотека matplotlib - для их визуализации. Также используется такая библиотека, как numpy. Она имеет множество вычислительных механизмов и поддерживает специализированные структуры данных, в том числе – одномерные и многомерные массивы, что значительно расширяет различные вычислительные возможности Python. А библиотека pandas предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

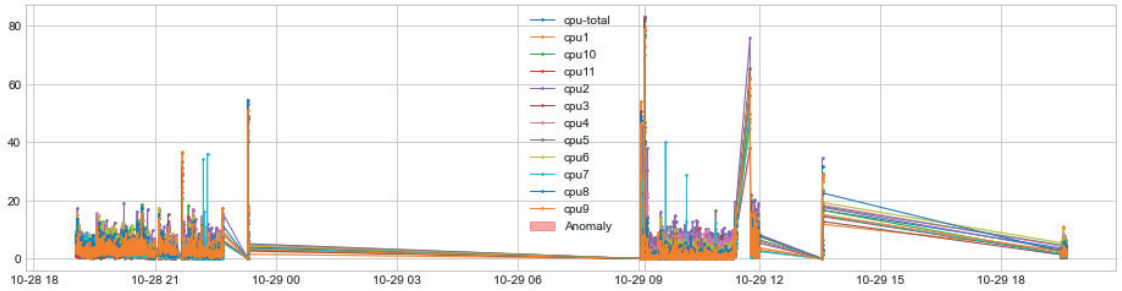
Далее будет описываться алгоритм действий, сопровождающийся несколькими фрагментами кода, благодаря которому поставленная задача была выполнена.

Шаг 1. В этом шаге описываются действия, связанные с получением данных с ЦП, отправкой их в базу данных в InfluxDB, и дальнейшее их получение в формате pandas.DataFrame. Первоначально необходимо предоставить параметры для авторизации и запроса, такие как токены, организации и бакеты, и сохранить их в соответствующих переменных. Затем необходимо сделать запрос Flux, с помощью которого можно извлекать данные из базы данных InfluxDB, фильтровать их, агрегировать, преобразовывать и выполнять другие операции над временными рядами. После извлечения данных нужно создать экземпляр API запросов, с помощью которого можно получать результаты запросов, формат которых преобразуется в формат pandas.DataFrame для удобства дальнейшего анализа и обработки.

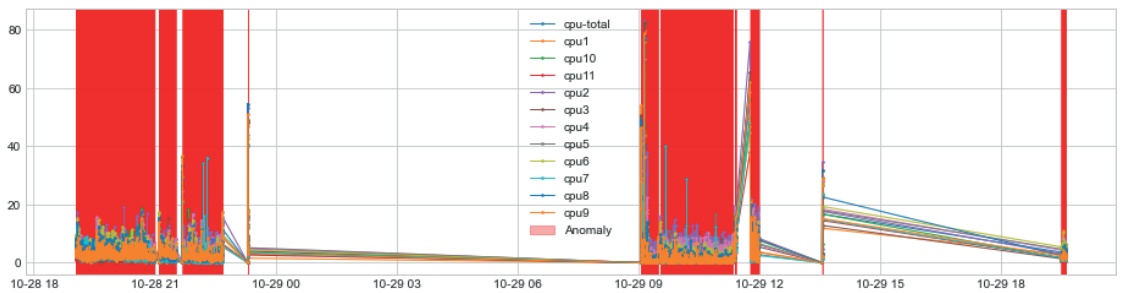
Шаг 2. На этом шаге происходит преобразование и подготовка данных. Необходимо преобразовать столбец времени в объект даты и времени и затем отбросить все лишние столбцы, которые не несут в себе полезной информации для кластеризации.

Шаг 3. На данном этапе используется функция MinClusterDetector() библиотеки ADTK вместе с выбранным алгоритмом кластеризации. Эта функция использует алгоритм кластеризации для разделения временного ряда на кластеры и затем определяет минимальные кластеры, которые могут быть считаны как аномалии. После этого происходит обучение модели на обучающих данных и обнаружений аномалий.

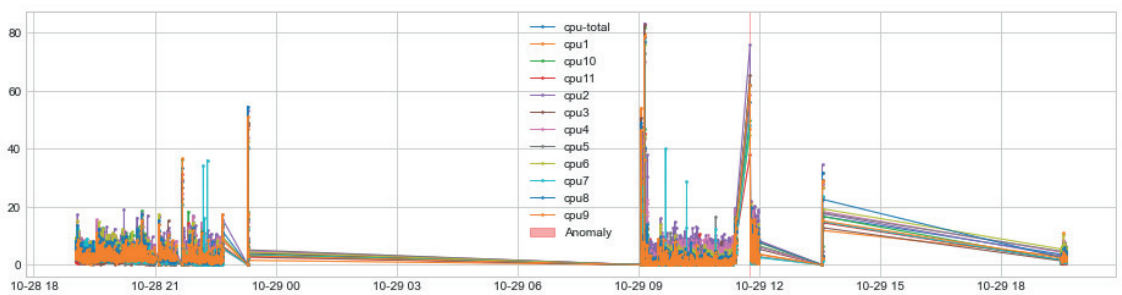
Шаг 4. Для дальнейшего анализа полученных данных была проведена визуализация результатов. Для каждого из вышеописанных алгоритмов кластеризации были получены следующие графики:



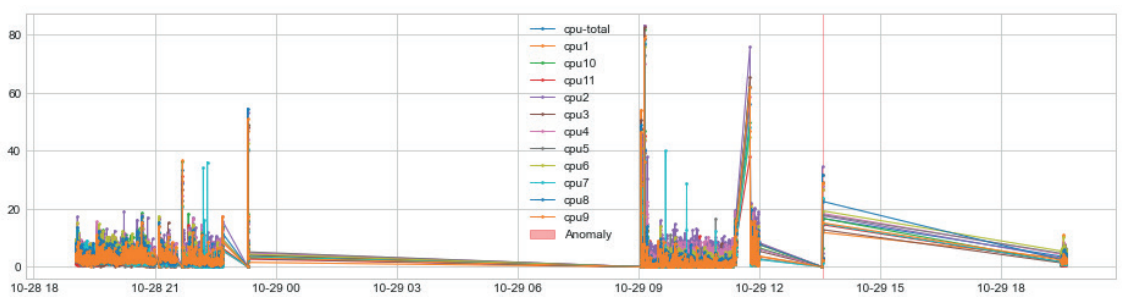
Р и с. 2. Выявление аномалий в работе ядер ЦП с применением алгоритма KMeans
F i g. 2. Detecting anomalies in the operation of CPU cores using the KMeans algorithm



Р и с. 3. Выявление аномалий в работе ядер ЦП с применением алгоритма DBSCAN
F i g. 3. Detecting anomalies in the operation of CPU cores using the DBSCAN algorithm



Р и с. 4. Выявление аномалий в работе ядер ЦП с применением алгоритма Agglomerative Clustering
F i g. 4. Detecting anomalies in the operation of CPU cores using the Agglomerative Clustering algorithm



Р и с. 5. Выявление аномалий в работе ядер ЦП с применением алгоритма Affinity Propagation
F i g. 5. Detecting anomalies in the operation of CPU cores using the Affinity Propagation algorithm

Также после выполнения этого задания были запушены метрики качества для оценки результатов каждого алгоритма кластеризации. Таблица с ними приведена ниже.

Таблица 4. **Результаты метрик качества на алгоритмах KMeans, DBSCAN, Agglomerative Clustering и Affinity Propagation**
Table 4. **Results of quality metrics on KMeans, DBSCAN, Agglomerative Clustering and Affinity Propagation algorithms**

Название алгоритма	ARI	AMI	Homogeneity score	Completeness score	V-measure score	Silhouette
KMeans	0.001760	6.743963e-02	0.202142	0.917709	0.331308	0.233780
DBSCAN	0.000000	6.304390e-17	0.000000	1.000000	0.000000	NaN
Agglomerative Clustering	0.000052	6.308259e-03	0.023518	0.992491	0.045947	0.664117
Affinity Propagation	0.003737	2.366259e-02	0.529144	0.899535	0.666327	0.081413

Заключение

В данной статье был проведен сравнительный анализ четырех алгоритмов кластеризации: KMeans, DBSCAN, Agglomerative Clustering и Affinity Propagation. Эти алгоритмы были оценены по их эффективности в кластеризации данных, используя различные метрики качества, такие как ARI, AMI, Homogeneity score, Completeness score, V – measure score и Silhouette score [22-25]. Из полученных результатов следует, что алгоритм KMeans превзошел все остальные показатели на метрике AMI, что говорит о том, что KMeans достиг лучшей согласованности с истинными метками кластеров по сравнению с другими алгоритмами. Алгоритм DBSCAN имеет самый высокий показатель, равный единице, на метрике Completeness score, что указывает на наличие однородных кластеров, где каждый кластер содержит только элементы из одного класса. Agglomerative Clustering, в свою очередь, добился высоких результатов на метрике Silhouette score, и это значит, что в данном случае произошла хорошая, явная разделимость между кластерами и лучшая согласованность внутри кластеров. А Affinity Propagation отличился сразу на двух метриках: на Homogeneity score и на V – measure score. Из этого можно сделать вывод, что данный алгоритм показывает высокую согласованность кластеров с исходными классами данных.

References

[1] Cerdà-Alabern L., Iuhasz G., Gemmi G. Anomaly detection for fault detection in wireless community networks using machine learning. *Computer Communications*. 2023;202:191-203. https://doi.org/10.1016/j.comcom.2023.02.019
[2] Ozer G., Netti A., Tafani D., Schulz M. Characterizing HPC Performance Variation with Monitoring and Unsupervised Learning. In: Jagode H., Anzt H., Juckeland G., Ltaief H. (eds.) High Performance Computing. ISC High Performance 2020. *Lecture Notes in Computer Science.* Vol. 12321. Cham: Springer; 2020. p. 280-292. https://doi.org/10.1007/978-3-030-59851-8_18
[3] Utomo D., Hsiung P.-A. A Multitiered Solution for Anomaly Detection in Edge Computing for Smart Meters. *Sensors*. 2020;20(18):5159. https://doi.org/10.3390/s20185159
[4] Fernando D., Rodriguez M.A., Arroba P., Ismail L., Buyya R. Efficient Training Approaches for Performance Anomaly Detection Models in Edge Computing Environments. *arXiv:*2408.12855. 2024. https://doi.org/10.48550/arXiv.2408.12855
[5] Daraghmeh M., Agarwal A., Jararweh Y. Anomaly Detection-Based Multilevel Ensemble Learning for CPU Prediction in Cloud Data Centers. In: 2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). Kingston, ON, Canada: IEEE Computer Society; 2024. p. 559-564. https://doi.org/10.1109/CCECE59415.2024.10667074
[6] Halawa M.S., Díaz Redondo R.P., Vilas A.F. Supervised Performance Anomaly Detection in HPC Data Centers. In: Hassanien A., Azar A., Gaber T., Bhatnagar R., F. Tolba M. (eds.) The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019). AMLTA 2019. *Advances in Intelligent Systems and Computing.* Vol. 921. Cham: Springer; 2020. p. 680-688. https://doi.org/10.1007/978-3-030-14118-9_67
[7] Cao C., Blaise A., Verwer S., Rebecchi F. Learning State Machines to Monitor and Detect Anomalies on a Kubernetes Cluster. In: Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22). Article number: 117. New York, NY, USA: Association for Computing Machinery; 2022. https://doi.org/10.1145/3538969.3543810
[8] Chavan V.D., Yalagi P.S. A Review of Machine Learning Tools and Techniques for Anomaly Detection. In: Choudrie J., Mahalle P.N., Perumal T., Joshi A. (eds.) ICT for Intelligent Systems. ICTIS 2023. Smart Innovation, *Systems and Technologies*. Vol. 361. Singapore: Springer; 2023. p. 395-406. https://doi.org/10.1007/978-981-99-3982-4_34
[9] Huč A., Šalej J., Trebar M. Analysis of Machine Learning Algorithms for Anomaly Detection on Edge Devices. *Sensors.* 2021;21(14):4946. https://doi.org/10.3390/s21144946
[10] Putina A., Rossi D. Online Anomaly Detection Leveraging Stream-Based Clustering and Real-Time Telemetry. *IEEE Transactions on Network and Service Management.* 2021;18(1):839-854. https://doi.org/10.1109/TNSM.2020.3037019
[11] Shiokawa H. Scalable Affinity Propagation for Massive Datasets. *Proceedings of the AAAI Conference on Artificial Intelligence.* 2021;35(11):9639-9646. https://doi.org/10.1609/aaai.v35i11.17160

[12] Kherbache M., Espes D., Amroun K. An Enhanced approach of the K-means clustering for Anomaly-based intrusion detection systems. In: 2021 International Conference on Computing, Computational Modelling and Applications (ICCMA). Brest, France: IEEE Computer Society; 2021. p. 78-83. https://doi.org/10.1109/ICCMA53594.2021.00021

[13] Fujiwara Y., Irie G., Kitahara T. Fast Algorithm for Affinity Propagation. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence. Volume Three (IJCAI'11). AAAI Press; 2011. p. 2238-2243. Available at: https://www.ijcai.org/Proceedings/11/Papers/373.pdf (accessed 27.01.2024).

[14] Molan M., Borghesi A., Cesarini D., Benini L., Bartolini A. RUAD: Unsupervised anomaly detection in HPC systems. *Future Generation Computer Systems*. 2023;141(C):542-554. https://doi.org/10.1016/j.future.2022.12.001

[15] Aljohani A. Optimizing Patient Stratification in Healthcare: A Comparative Analysis of Clustering Algorithms for EHR Data. *International Journal of Computational Intelligence Systems*. 2024;17:173. https://doi.org/10.1007/s44196-024-00568-8

[16] Dodda S., Chintala S., Kunchakuri N., Kamuni N. Enhancing Microservice Reliability in Cloud Environments Using Machine Learning for Anomaly Detection. In: 2024 International Conference on Computing, Sciences and Communications (ICCSC). Ghaziabad, India: IEEE Computer Society; 2024. p. 1-5. https://doi.org/10.1109/ICCSC62048.2024.10830437

[17] Giorgino T. Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. *Journal of Statistical Software*. 2009;31(7):1-24. https://doi.org/10.18637/jss.v031.i07

[18] Wang L., Koniusz P. Uncertainty-DTW for Time Series and Sequences. In: Avidan S., Brostow G., Cissé M., Farinella G.M., Hassner T. (eds.) Computer Vision – ECCV 2022. ECCV 2022. *Lecture Notes in Computer Science*. Vol. 13681. Cham: Springer; 2022. p. 176-195. https://doi.org/10.1007/978-3-031-19803-8_11

[19] Bie M., Li W., Fu Q., Chen T., Du Y., Nan L. Energy-Efficient Reconfigurable Acceleration Engine for Polynomial Coefficient Generation of Lattice-Based Post-Quantum Cryptography. *Electronics.* 2024;13(24):4921. https://doi.org/10.3390/electronics13244921

[20] Togbe M.U., et al. Anomaly Detection for Data Streams Based on Isolation Forest Using Scikit-Multiflow. In: Gervasi O., et al. Computational Science and Its Applications – ICCSA 2020. ICCSA 2020*. Lecture Notes in Computer Science.* Vol. 12252. Cham: Springer; 2020. p. 15-30. https://doi.org/10.1007/978-3-030-58811-3_2

[21] Sun L., Guo C., Liu C., et al. Fast affinity propagation clustering based on incomplete similarity matrix. *Knowledge and Information Systems*. 2017;51:941-963. https://doi.org/10.1007/s10115-016-0996-y

[22] Deng D. Research on Anomaly Detection Method Based on DBSCAN Clustering Algorithm. In: 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT). Shenyang, China: IEEE Computer Society; 2020. p. 439-442. https://doi.org/10.1109/ISCTT51595.2020.00083

[23] Patel P., Sivaiah B., Patel R. Approaches for finding Optimal Number of Clusters using K-Means and Agglomerative Hierarchical Clustering Techniques. In: 2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP). Hyderabad, India: IEEE Computer Society; 2022. p. 1-6. https://doi.org/10.1109/ICICCSP53532.2022.9862439

[24] Shorewala V. Anomaly Detection and Improvement of Clusters using Enhanced K-Means Algorithm. In: 2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP). Chennai, India: IEEE Computer Society; 2021. p. 115-121. https://doi.org/10.1109/ICCCSP52374.2021.9465539

[25] Javed A., Lee B.S., Rizzo D.M. A benchmark study on time series clustering. *Machine Learning with Applications.* 2020;1:100001. https://doi.org/10.1016/j.mlwa.2020.100001

Поступила 27.01.2024; одобрена после рецензирования 09.02.2024; принята к публикации 12.03.2024.
Submitted 27.01.2024; approved after reviewing 09.02.2024; accepted for publication 12.03.2024.

Об авторах:

Турашев Артём Сергеевич, студент факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0000-0001-8391-4948,** turashev.artem@mail.ru

Сухомлин Владимир Александрович, заведующий лабораторией открытых информационных технологий факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), доктор технических наук, профессор, **ORCID: https://orcid.org/0000-0001-9468-7138**, sukhomlin@mail.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Artem S. Turashev, Student of the Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0000-0001-8391-4948,** turashev.artem@mail.ru

Vladimir A. Sukhomlin, Head of the Open Information Technologies Lab, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), Dr. Sci. (Tech.), Professor, **ORCID: https://orcid.org/0000-0001-9468-7138**, sukhomlin@mail.ru

All authors have read and approved the final manuscript.