«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	01.03.02 – Прикладная математика и информатика
Профиль	Математическое обеспечение программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой А.А. Лисс

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: РАЗРАБОТКА МОДЕЛИ КЛАССИФИКАЦИИ СОБЫТИЙ В ЖУРНАЛАХ ВЕБ-СЕРВЕРОВ

Студент		_____ *подпись*	Котов Д.А.
Руководитель	к.т.н., доцент *(Уч. степень, уч. звание)*	_____ *подпись*	Заславский М.М.
Консультант	к.э.н. *(Уч. степень, уч. звание)*	_____ *подпись*	Артамонова О.С.

Санкт-Петербург

2024

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____2024 г.

Студент Котов Д.А. Группа 0381

Тема работы: Разработка модели классификации событий в журналах веб-серверов

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МО ЭВМ

Исходные данные (технические требования):

Программа должна быть написана на языке программирования Python.

Содержание ВКР:

Введение, Обзор предметной области, Спецификация требований к решению, Описание метода решения, Анализ полученного решения, Заключение.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал.

Дополнительные разделы: Обеспечение качества разработки, продукции, программного продукта.

Дата выдачи задания «___»_____20___ г.

Дата представления ВКР к защите «___»_____20___ г.

Студент _____ Котов Д.А.

Руководитель к.т.н., доцент _____ Заславский М.М.
(Уч. степень, уч. звание)

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____2024 г.

Студент Котов Д.А. Группа 0381

Тема работы: Разработка модели классификации событий в журналах веб-серверов

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	01.04 – 12.04
2	Генерация набора данных для модели	13.04 – 19.04
3	Разработка модели	20.04 – 26.04
4	Анализ полученного решения	27.04 – 03.05
5	Оформление пояснительной записки	04.05 – 07.05
6	Оформление иллюстративного материала	08.05 – 10.05

Студент _____ Котов Д.А

Руководитель к.т.н., доцент _____ Заславский М.М.
(Уч. степень, уч. звание)

РЕФЕРАТ

Пояснительная записка 55 стр., 5 рис., 7 табл., 10 ист.

ЖУРНАЛЫ ВЕБ-СЕРВЕРОВ, МАШИННОЕ ОБУЧЕНИЕ, АЛГОРИТМ КЛАСТЕРИЗАЦИИ.

Объектом исследования является модель для классификации журналов веб серверов.

Предметом исследования является алгоритм кластеризации и выявления аномалий.

Цель работы: разработать модель для кластеризации логов и выявления аномалий, которая на вход будет получать записи журналов веб-серверов из которых будет формировать кластеры и аномалии, последующие записи будут перечисляться к каким либо кластерам или аномалиям.

В данной работе были рассмотрены существующие алгоритмы кластеризации и выбран алгоритм, наиболее подходящий для цели работы. Впоследствии была реализована программа на языке программирования Python, которая обрабатывает текстовый файл с журналом событий, далее, используя ранее выбранный алгоритм, распределяет события по кластерам. Были проанализированы различные подходы к решению данной задачи.

ABSTRACT

In this paper, the existing clustering algorithms were considered and the algorithm most suitable for the purpose of the work was selected. Subsequently, a program in the Python programming language was implemented that processes a text file with an event log, then, using the previously selected algorithm, distributes events into clusters. Various approaches to solving this problem were analyzed.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЯП – язык программирования;

IDE – Integrated Development Environment;

ПО – программное обеспечение;

HTTP – HyperText Transfer Protocol;

HTML – HyperText Markup Language;

URL – Uniform Resource Locator;

CLF – Common Log Format;

DBSCAN – Density-Based Spatial Clustering of Applications with Noise;

BERT – Bidirectional Encoder Representations from Transformers;

Лог - файл журнала с записями о событиях в хронологическом порядке.

ВВЕДЕНИЕ

С каждым днем количество веб-серверов и их пользователей растет [1]. В связи с этим возникает необходимость в эффективном анализе и обработке данных журналов событий веб-серверов для обеспечения надежности и безопасности веб-сервисов, а также для выявления и предотвращения возможных аномалий.

Цель данной работы разработка модели классификации событий в журналах веб-серверов на основе методов машинного обучения.

Задачи данной работы:

1. Изучение и анализ существующих алгоритмов кластеризации.
2. Разработка алгоритма предварительной обработки данных журналов для повышения эффективности классификации.
3. Создание и обучение модели классификации событий.
4. Оценка эффективности разработанной модели на реальных данных.

Объектом исследования является модель для классификации журналов веб серверов.

Предметом исследования является алгоритм кластеризации и выявления аномалий.

Практическая значимость работы заключается в том, что разработанная модель классификации событий в журналах веб-серверов способствует автоматизации процессов обнаружения аномалий, что позволяет сократить время на обработку инцидентов.

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Основные теоретические положения

1.1.1 Работа веб-сервера

Веб-сервер — это ПО, обрабатывающее запросы к веб-страницам через Интернет и предоставляющее HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными. Он работает по следующему принципу:

- Получение запроса: Когда пользователь вводит URL веб-сайта в браузере или переходит по ссылке, браузер отправляет запрос на веб-сервер, на котором хранится сайт.

- Обработка запроса: Веб-сервер анализирует полученный запрос, определяя, какую страницу или ресурс (например, изображение или документ) необходимо предоставить.

- Поиск ресурса: Веб-сервер ищет запрашиваемый ресурс на своем хранилище. Если ресурс требует дополнительной обработки (например, выполнения скриптов на стороне сервера), сервер выполняет необходимые действия для генерации финального содержимого.

- Отправка ответа: После нахождения или генерации запрашиваемого ресурса, веб-сервер формирует ответ, который включает в себя статус (например, успешно или ошибка) и сам ресурс (HTML-страницу, изображение и т.д.).

- Отображение ресурса: Браузер пользователя получает ответ от веб-сервера, обрабатывает его и отображает содержимое страницы или сообщение об ошибке, если доступ к ресурсу не был получен.

Схема работы сервера изображена на рис. 1.

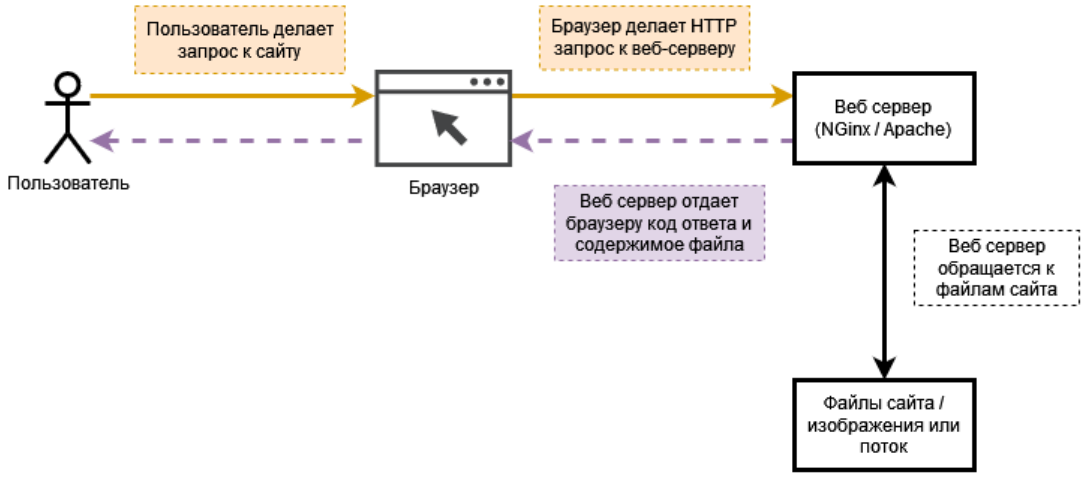

Рисунок 1 — Схема работы веб-сервера

1.1.2 Запросы к серверу

В протоколе HTTP предусмотрено большое количество методов с помощью которых можно совершать запросы. Но в повседневной практике используется, как правило только три: GET, POST и HEAD, именно они и встречаются в наборе данных, который использовался для анализа событий журнала веб-сервера.

Ниже представлено краткое пояснение к запросам к веб-серверу(таблица 1).

Таблица 1 — Запросы к веб-серверу

Метод	Пояснение
GET	Данный метод используется для запроса содержимого указанного ресурса.
HEAD	Запрос HEAD аналогичен GET, но сервер в ответе отправляет только заголовки и статус, без тела ответа. Это полезно для извлечения метаданных, таких как тип содержимого или последняя дата модификации
POST	Запрос POST применяется для отправки данных для обработки на веб-сервер. Это могут быть формы обратной связи, загрузка файлов, добавление комментариев на сайт.

1.1.3 Коды ответа HTTP

Коды ответа HTTP — это стандартизированные коды, которые веб-серверы используют для сообщения клиентам о результате их запросов. Коды ответа подразделяются на несколько классов, каждый из которых имеет свою сферу применения.

Ниже представлено краткое пояснение к кодам ответа HTTP(таблица 2).

Таблица 2 — Коды ответа HTTP.

Метод	Пояснение

1xx	Информационные коды ответов. • Пример: 100 Continue — клиент может продолжать отправку запроса.
2xx	Коды ответа, означающие успешное выполнение запроса. • Пример: 200 OK — стандартный ответ для успешных HTTP-запросов. • Пример: 201 Created — запрос успешен и в результате был создан новый ресурс.
3xx	Коды ответа, означающие необходимость перенаправления на другой ресурс или URL • Пример: 301 Moved Permanently — ресурс окончательно перемещен на новый URL. • Пример: 302 Found — ресурс временно перемещен на другой URL.
4xx	Эти коды указывают на ошибку со стороны клиента. • Пример: 400 Bad Request — сервер не понимает запрос из-за неверного синтаксиса.

	• Пример: 401 Unauthorized — для доступа к запрашиваемому ресурсу требуется аутентификация. • Пример: 404 Not Found — сервер не может найти запрашиваемый ресурс.
5xx	Эти коды сообщают о проблемах на стороне сервера, которые он не смог обработать. • Пример: 500 Internal Server Error — общая ошибка сервера, когда сервер сталкивается с ситуацией, которую он не знает как обработать. • Пример: 503 Service Unavailable — сервер временно не доступен, обычно из-за перегрузки или технического обслуживания.

1.1.4 Логи сервера Apache

Логи сервера Apache — это файлы журналов, в которые записывается информация о всех запросах и действиях, происходящих на веб-сервере. Эти данные крайне важны для анализа трафика, отладки, мониторинга безопасности и оптимизации работы сервера.

Основные типы лог-файлов Apache:

Access Log (Журнал доступа): Содержит информацию о каждом запросе к серверу. Здесь можно найти данные о времени запроса, IP-адресе клиента, методе HTTP, запрашиваемом URL, статусе HTTP ответа, размере ответа и информацию о браузере пользователя (User-Agent).

Error Log (Журнал ошибок): Фиксирует ошибки, возникшие во время работы сервера. Это могут быть проблемы с конфигурацией, недоступность файлов, проблемы с сетевыми соединениями и другие системные ошибки. Этот журнал является основным инструментом для диагностики проблем на сервере.

Custom Logs (Пользовательские журналы): Apache позволяет настраивать логи для записи специфической информации в соответствии с потребностями администратора. Это может быть полезно для сбора конкретных данных для анализа.

Однако в данной работе для анализа журналов событий используется только Access Log.

В записи логов сервера Apache используется формат Common Log Format (CLF)[4] или его расширения, такие как Combined Log Format.

Пример стандартной записи в Access Log в формате CLF изображён на рис. 2

```
46.105.14.53 - - [17/May/2015:13:05:11 +0000] "GET /blog/tags/puppet?flav=rss20 HTTP/1.1" 200 14872 "-" "UniversalFeedParser/4.2-pre-314-svn +http://feedparser.org/"
144.76.194.187 - - [17/May/2015:13:05:33 +0000] "GET / HTTP/1.0" 200 37932 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:28 +0000] "GET /wp-login.php HTTP/1.0" 404 292 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:37 +0000] "GET /administrator/index.php HTTP/1.0" 404 303 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:11 +0000] "GET /reset.css HTTP/1.0" 200 1015 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:37 +0000] "GET /style2.css HTTP/1.0" 200 4877 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:27 +0000] "GET /favicon.ico HTTP/1.0" 200 3638 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:12 +0000] "GET /?page=2 HTTP/1.0" 200 36673 "-" "-"
144.76.194.187 - - [17/May/2015:13:05:10 +0000] "GET /about/ HTTP/1.0" 200 11474 "-" "-"
```

Рисунок 2 — Пример записи логов веб-сервера

В логах содержится следующая информация:

- IP адрес устройства с которого совершен запрос;
- Дата и время запроса;
- URL адрес к которому направлен запрос;
- Метод с помощью которого совершен запрос (GET, POST и т. д);
- Код ответа на совершенный запрос;
- Referer — адрес страницы с которой был совершен переход на текущую;
- User-Agent пользователя — идентификатор устройства и браузера.

С помощью логов мы можем выявить множество метрик:

- На какие страницы чаще всего заходят пользователи;
- С каких устройств чаше всего заходят пользователи;
- Какие поисковые боты посещают сайт и какие страницы индексируют;
- Наличие на сайте несуществующих страниц, к которым тем не менее обращаются пользователи и, что наиболее важно поисковые боты;
- Наличие на сайте редиректов;
- Наличие на сайте критических ошибок 5xx.

1.2 Кластеризация объектов или кластерный анализ

1.2.1 Понятие кластеризации

Кластеризация (или кластерный анализ)[2] представляет собой процесс группировки набора элементов в группы, называемые кластерами. Целью является собрать в одной группе элементы, которые между собой похожи, в то время как элементы, принадлежащие к разным группам, должны максимально отличаться друг от друга. Основное различие между кластеризацией и классификацией заключается в том, что при кластеризации заранее не определены конкретные группы, их выявление происходит в ходе выполнения алгоритма.

Процесс кластерного анализа включает в себя несколько шагов:

1. Выборка данных для кластеризации.

2. Определение набора переменных, по которым будет проводиться оценка элементов выборки. При необходимости осуществляется нормализация данных.

3. Расчет значений меры сходства между элементами.

4. Применение специализированного метода кластерного анализа для формирования групп похожих элементов(кластеров).

5. Визуализация и интерпретация полученных результатов кластеризации.

На этапе анализа результатов может потребоваться доработка используемых метрик и методов кластеризации для достижения наилучших результатов.

1.2.2 Меры расстояний

Чтобы вычислять значения меры сходства между объектами нужно составить вектор характеристик для каждого объекта — как правило, это набор числовых значений. Однако существуют также алгоритмы, работающие с качественными (категорийными) характеристиками.

После того, как определён вектор характеристик, можно провести нормализацию, чтобы все компоненты давали одинаковый вклад при расчете «расстояния». В процессе нормализации все значения приводятся к некоторому диапазону, например,$[-1,1]$ или $[0,1]$.

Далее, для каждой пары объектов измеряется «расстояние» между ними — степень похожести. Существует множество метрик, чаще всего используются:

1. Евклидово расстояние

   Наиболее распространенная функция расстояния. Представляет собой геометрическим расстоянием в многомерном пространстве:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}$$

2. Квадрат евклидова расстояния

Применяется для придания большего веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом:

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2$$

3. Расстояние городских кварталов (манхэттенское расстояние)[9]

Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Формула для расчета манхэттенского расстояния:

$$\rho(x, x') = \sum_{i}^{n} |x_i - x_i'|$$

4. Расстояние Чебышева

Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Расстояние Чебышева вычисляется по формуле:

$$\rho(x, x') = \sum_{i}^{n} max\,(|x_i - x_i'|)$$

5. Степенное расстояние

Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Степенное расстояние вычисляется по следующей формуле:

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p}$$

где r и p – параметры, определяемые пользователем. Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра – r и p — равны двум, то это расстояние совпадает с расстоянием Евклида.

Выбор метрики полностью лежит на исследователе, поскольку результаты кластеризации могут существенно отличаться при использовании разных мер.

1.2.3 Классификация алгоритмов

1. Иерархические и плоские.

Иерархические алгоритмы (также называемые алгоритмами таксономии) строят не одно разбиение выборки на непересекающиеся кластеры, а систему вложенных разбиений. Таким образом на выходе получается дерево кластеров, корнем которого является вся выборка, а листьями — наиболее мелкие кластера.

Плоские алгоритмы строят одно разбиение объектов на кластеры.

2. Четкие и нечеткие.

Четкие (или непересекающиеся) алгоритмы каждому объекту выборки ставят в соответствие номер кластера, таким образом каждый объект принадлежит только одному кластеру. Нечеткие (или пересекающиеся) алгоритмы каждому объекту ставят в соответствие набор вещественных значений, показывающих степень отношения объекта к кластерам. В итоге каждый объект относится к каждому кластеру с некоторой вероятностью.

1.2.4 Объединение кластеров

В случае использования иерархических алгоритмов необходимо решить, как объединять между собой кластера, как вычислять «расстояния» между ними. Существует несколько метрик:

1. Одиночная связь (расстояния ближайшего соседа)

В этом методе расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Результирующие кластеры имеют тенденцию объединяться в цепочки.

2. Полная связь (расстояние наиболее удаленных соседей)

В этом методе расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. наиболее удаленными соседями). Этот метод обычно работает очень хорошо, когда объекты происходят из отдельных групп. Если же кластеры имеют удлиненную форму или их естественный тип является «цепочечным», то этот метод непригоден.

3. Невзвешенное попарное среднее

В этом методе расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них. Метод эффективен, когда объекты формируют различные группы, однако он работает одинаково хорошо и в случаях протяженных («цепочечного» типа) кластеров.

4. Взвешенное попарное среднее

Метод идентичен методу невзвешенного попарного среднего, за исключением того, что при вычислениях размер соответствующих кластеров (то есть число объектов, содержащихся в них) используется в качестве весового коэффициента. Поэтому данный метод должен быть использован, когда предполагаются неравные размеры кластеров.

5. Невзвешенный центроидный метод

В этом методе расстояние между двумя кластерами определяется как расстояние между их центрами тяжести.

6. Взвешенный центроидный метод (медиана)

Этот метод идентичен предыдущему, за исключением того, что при вычислениях используются веса для учета разницы между размерами кластеров. Поэтому, если имеются или подозреваются значительные отличия в размерах кластеров, этот метод оказывается предпочтительнее предыдущего.

1.2.5 Обзор алгоритмов кластеризации

- Алгоритмы иерархической кластеризации[5]

Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. Более распространены восходящие алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. Таким образом строится система вложенных разбиений. Результаты таких алгоритмов обычно представляют в виде дерева – дендрограммы. Классический пример такого дерева – классификация животных и растений.

Для вычисления расстояний между кластерами чаще все пользуются двумя расстояниями: одиночной связью или полной связью .

К недостатку иерархических алгоритмов можно отнести систему полных разбиений, которая может являться излишней в контексте решаемой задачи.

- Алгоритмы квадратичной ошибки[6]

Задачу кластеризации можно рассматривать как построение оптимального разбиения объектов на группы. При этом оптимальность может быть определена как требование минимизации среднеквадратической ошибки разбиения:

$$e^2(X, L) = \sum_{j=0}^{k} \sum_{i=0}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2$$

где c_j — «центр масс» кластера j (точка со средними значениями характеристик для данного кластера).

Алгоритмы квадратичной ошибки относятся к типу плоских алгоритмов. Самым распространенным алгоритмом этой категории является метод k-средних. Этот алгоритм строит заданное число кластеров, расположенных как можно дальше друг от друга. Работа алгоритма делится на несколько этапов:

1. Случайно выбрать k точек, являющихся начальными «центрами масс» кластеров.

2. Отнести каждый объект к кластеру с ближайшим «центром масс».

3. Пересчитать «центры масс» кластеров согласно их текущему составу.

4. Если критерий остановки алгоритма не удовлетворен, вернуться к пункту 2.

В качестве критерия остановки работы алгоритма обычно выбирают минимальное изменение среднеквадратической ошибки. Так же возможно останавливать работу алгоритма, если на шаге 2 не было объектов, переместившихся из кластера в кластер.

К недостаткам данного алгоритма можно отнести необходимость задавать количество кластеров для разбиения.

- Нечеткие алгоритмы

Наиболее популярным алгоритмом нечеткой кластеризации является алгоритм c-средних (c-means). Он представляет собой модификацию метода k-средних. Шаги работы алгоритма:

1. Выбрать начальное нечеткое разбиение n объектов на k кластеров путем выбора матрицы принадлежности U размера nxk.

2. Используя матрицу U, найти значение критерия нечеткой ошибки:

   $$E^2(X,U) = \sum_{i=1}^{N}\sum_{k=1}^{K} U_{ik} \left\| x_i^{(k)} - c_k \right\|^2,$$

   где c_k — «центр масс» нечеткого кластера k:

   $$c_k = \sum_{i=1}^{N} U_k \, x_i.$$

3. Перегруппировать объекты с целью уменьшения этого значения критерия нечеткой ошибки.
4. Возвращаться в пункт 2 до тех пор, пока изменения матрицы U не станут незначительными.

Этот алгоритм может не подойти, если заранее неизвестно число кластеров, либо необходимо однозначно отнести каждый объект к одному кластеру.

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)[3]

DBSCAN кластеризует точки, основываясь на их плотности расположения. Этот алгоритм способен находить кластеры произвольной формы и хорошо

работает с данными, содержащими шум и выбросы. DBSCAN определяет кластеры как области высокой плотности, разделенные областями низкой плотности, и не требует предварительного определения количества кластеров.

Этот алгоритм наиболее подходит для данной задачи, поскольку способен находить выбросы, в нашем случае аномалии, также не надо задавать количество кластеров заранее, алгоритм сам определяет их количество по данным.

- Алгоритмы, основанные на теории графов[7]

Суть таких алгоритмов заключается в том, что выборка объектов представляется в виде графа $G = (V, E)$, вершинам которого соответствуют объекты, а ребра имеют вес, равный «расстоянию» между объектами. Достоинством графовых алгоритмов кластеризации являются наглядность, относительная простота реализации и возможность внесения различных усовершенствований, основанные на геометрических соображениях. Основными алгоритмам являются алгоритм выделения связных компонент, алгоритм построения минимального покрывающего (остовного) дерева и алгоритм послойной кластеризации.

- Алгоритм выделения связных компонент

В алгоритме выделения связных компонент задается входной параметр R и в графе удаляются все ребра, для которых «расстояния» больше R. Соединенными остаются только наиболее близкие пары объектов. Смысл алгоритма заключается в том, чтобы подобрать такое значение R, лежащее в диапазон всех «расстояний», при котором граф «развалится» на несколько связных компонент. Полученные компоненты и есть кластеры.

Для подбора параметра R обычно строится гистограмма распределений попарных расстояний. В задачах с хорошо выраженной кластерной структурой данных на гистограмме будет два пика – один соответствует внутрикластерным расстояниям, второй – межкластерным расстояния. Параметр R подбирается из зоны минимума между этими пиками. При этом управлять количеством кластеров при помощи порога расстояния довольно затруднительно.

- Алгоритм минимального покрывающего дерева[8][10]

Алгоритм минимального покрывающего дерева сначала строит на графе минимальное покрывающее дерево, а затем последовательно удаляет ребра с наибольшим весом. На рисунке изображен пример минимального покрывающего дерева, полученного для девяти объектов(рис. 3).

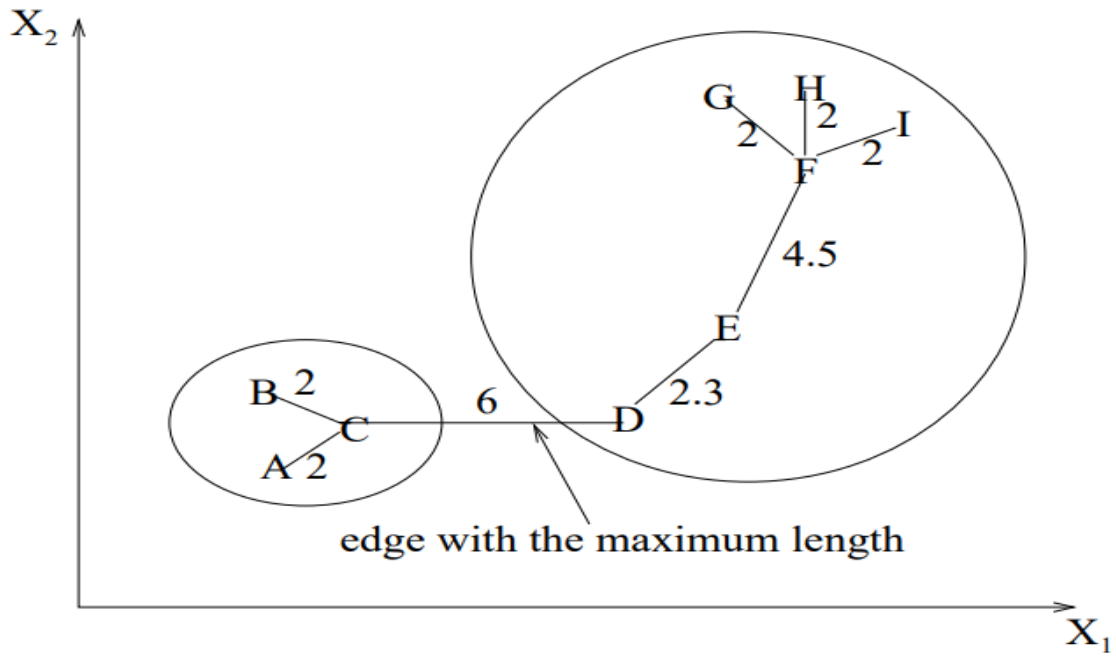

Рисунок 3 — Минимальное покрывающее дерево для девяти объектов

Путём удаления связи, помеченной CD, с длиной равной 6 единицам (ребро с максимальным расстоянием), получаем два кластера: A, B, C и D, E, F, G, H, I.

Второй кластер в дальнейшем может быть разделён ещё на два кластера путём удаления ребра EF, которое имеет длину, равную 4.5 единицам.

- Послойная кластеризация

Алгоритм послойной кластеризации основан на выделении связных компонент графа на некотором уровне расстояний между объектами (вершинами). Уровень расстояния задается порогом расстояния c. Например, если расстояние между объектами $0 \leq p(x, x') \leq 1$, то $0 \leq c \leq 1$.

Алгоритм послойной кластеризации формирует последовательность подграфов графа G, которые отражают иерархические связи между кластерами:

$$G^0 \subseteq G^1 \subseteq \ldots \subseteq G^m$$

где $G^t = (V, E^t)$ — граф на уровне c^t,

$E^t = \{e_{ij} \in E : p_{ij} \leq c^t\}$,

c^t – t-ый порог расстояния,

m – количество уровней иерархии,

$G^0 = (V, o), o$ – пустое множество ребер графа, получаемое при $t^0 = 1$,

$G^m = G$, то есть граф объектов без ограничений на расстояние (длину ребер графа), поскольку $t^m = 1$.

Посредством изменения порогов расстояния $\{\mathrm{c}^0, \ldots, \mathrm{c}^m\}$, где $0 = \mathrm{c}^0 < \mathrm{c}^1 < \cdots < \mathrm{c}^m = 1$, возможно контролировать глубину иерархии получаемых кластеров. Таким образом, алгоритм послойной кластеризации способен создавать как плоское разбиение данных, так и иерархическое.

1.2.6 Сравнение алгоритмов

В таблице представлена вычислительная сложность алгоритмов(таблица 3)

Таблица 3 — Вычислительная сложность алгоритмов.

Алгоритм кластеризации	Вычислительная сложность
Иерархический	$O(n^2)$
k-средних	$O(nkl)$, где k – число кластеров, l – число итераций
c-средних	
Выделение связных компонент	зависит от алгоритма
Минимальное покрывающее дерево	$O(n: 2logn)$
Послойная кластеризация	$O\big(max(n,m)\big)$, где $m < n\,(n-1)/2$

В таблице представлено сравнение алгоритмов(таблица 4)

Таблица 4 — Сравнительная таблица алгоритмов.

Алгоритм кластеризации	Форма кластеров	Входные данные	Результаты
Иерархический	Произвольная	Число кластеров или порог расстояния для усечения иерархии	Бинарное дерево кластеров
k-средних	Гиперсфера	Число кластеров	Центры кластеров
c-средних	Гиперсфера	Число кластеров, степень	Центры кластеров, матрица

		нечеткости	принадлежности
Выделение связных компонент	Произвольная	Порог расстояния R	Древовидная структура кластеров
Минимальное покрывающее дерево	Произвольная	Число кластеров или порог расстояния для удаления ребер	Древовидная структура кластеров
Послойная кластеризация	Произвольная	Последовательность порогов расстояния	Древовидная структура кластеров с разными уровнями иерархии

1

.

2

.

Выводы

Выбор алгоритма кластеризации DBSCAN для анализа событий в журналах веб-серверов обусловлен рядом его преимуществ перед другими алгоритмами. Во-первых, DBSCAN отлично подходит для выявления аномалий, так как он способен определить и исключить отдаленные точки, которые не принадлежат ни одному кластеру, рассматривая их как выбросы. Это особенно ценно в контексте безопасности веб-серверов, где такие аномалии могут указывать на нестандартное или подозрительное поведение.

Также, DBSCAN не требует предварительного указания количества кластеров, что делает его гибким в работе с данными, структура которых заранее неизвестна. Это позволяет алгоритму адаптироваться к данным различной природы и выявлять скрытые структуры без необходимости их предварительного определения.

Помимо этого, DBSCAN устойчив к шуму и способен выделять кластеры произвольной формы, что делает его применимым к разнообразным наборам данных, включая те, которые содержат сложные и пересекающиеся структуры.

Кроме того, DBSCAN эффективен в вычислительном плане, особенно когда используется в сочетании со структурами данных индексации, что позволяет сократить время обработки даже больших объемов данных.

Таким образом, DBSCAN является мощным инструментом для кластеризации данных журналов веб-серверов, обеспечивая высокое качество обнаружения аномалий и гибкость в работе с различными типами данных.

2 ВЫБОР МЕТОДА РЕШЕНИЯ

2.1 Постановка задачи

Необходимо разработать модель для кластеризации логов веб-серверов с целью выявления аномальных событий. Модель должна быть способна обрабатывать и анализировать записи журналов веб-серверов, выделяя из них статистически значимые кластеры, характеризующие типичное поведение системы, а также идентифицировать отклонения от нормы, которые могут свидетельствовать о потенциальных угрозах или нештатных ситуациях.

2.2 Сбор и предобработка данных

Необходимо организовать сбор данных из журналов веб-серверов, их очистку и нормализацию для последующего использования в модели кластеризации. Это включает удаление нерелевантной информации, преобразование текстовых данных в числовой формат и обработку пропущенных значений.

2.3 Исследование и выбор метода кластеризации

Также необходимо было провести анализ существующих алгоритмов кластеризации для определения наиболее подходящего под задачи работы. Нужно учитывать способность алгоритма обрабатывать большие объемы данных, его устойчивость к шуму и способность выявлять аномалии.

2.4 Разработка модели

Далее нужно разработать модель, которая будет принимать на вход записи журналов, обрабатывать их и формировать кластеры, представляющие

типовые сценарии работы веб-сервера. Модель должна быть способна адаптироваться к изменениям в данных и обновлять кластеры со временем.

2.5 Выявление аномалий

Необходимо интегрировать механизмы для определения и отслеживания аномальных событий, которые не соответствуют ни одному из существующих кластеров. Это включает разработку правил или использование статистических методов для выявления потенциальных угроз.

Тестирование и валидация модели

Необходимо провести проверку эффективности модели на тестовых и реальных данных, оценка точности кластеризации и способности модели к детектированию аномалий. В силу отсутствия заранее размеченных данных, оценка качества кластеризации и способности модели к детектированию аномалий будет иметь субъективный характер.

3 ОПИСАНИЕ МЕТОДА РЕШЕНИЯ

- Необходимо провести проверку эффективности модели на тестовых и реальных данных, оценка точности кластеризации и способности модели к детектированию аномалий. В силу отсутствия
- V
- ss

3.1 Генерация набора событий журнала веб-сереров

В процессе разработки модели классификации событий в журналах веб-серверов возникла необходимость в создании набора данных для начальных этапов обучения модели. Из-за отсутствия доступа к реальным журналам веб-сервера было принято решение о генерации искусственного набора данных, соответствующего формату CLF.

Для создания искусственных записей журнала была использована библиотека faker, предназначенная для генерации случайных данных в языке программирования Python. Эта библиотека позволила сформировать реалистичные IP-адреса и URL-адреса, имитируя потенциальные запросы пользователей к веб-серверу. Поле user-agent было сгенерировано на основе десяти различных заготовленных значений, которые отражали распространенные браузеры и операционные системы.

Остальные поля журнала, такие как дата и время запроса, статус ответа сервера и размер передаваемого контента, также создавались случайным образом, обеспечивая разнообразие в генерируемом наборе данных. Каждая запись в журнале формировалась путем случайного выбора значений для каждого поля, соблюдая структуру и формат CLF.

Однако в ходе последующего тестирования модели выявилось, что использование сгенерированных данных приводит к неудовлетворительным результатам. Модель, обученная на искусственных данных, не смогла адекватно справляться с задачей классификации и выявления аномалий при работе с реальными журналами веб-серверов. Анализ показал, что отсутствие реалистичных шаблонов поведения и взаимосвязей в сгенерированных данных не позволяло модели корректно обобщать информацию и выявлять значимые аномалии.

В связи с этим было принято решение отказаться от использования искусственно сгенерированных данных и перейти к работе с реальными журналами веб-серверов, что позволило значительно улучшить качество и точность классификации модели.

3.2 Предварительная обработка данных

Был проведён отбор необходимых данных из общего множества записей журнала и удаление тех полей, которые не несут значимой информации для анализа.

В рамках предварительной обработки было решено оставить следующие поля, которые считаются наиболее важными для дальнейшего анализа и выявления аномалий:

- Коды ошибок (Status Codes): Позволяют определить успешность или неудачу запроса и являются ключевыми для выявления проблем на веб-сервере.
- Время запроса (Timestamp): Важно для анализа паттернов активности и идентификации нестандартного поведения, например, внезапных всплесков трафика.

- URL (Uniform Resource Locator): Адрес запрашиваемого ресурса может указывать на целевые страницы атак или на необычные пути доступа.
- User-Agent: Информация о браузере и операционной системе пользователя помогает выявить подозрительные или устаревшие клиенты, потенциально уязвимые для эксплуатации.
- ENDPOINT: Конкретный ресурс на сервере, к которому был выполнен запрос, может выявить необычные или подозрительные пути доступа.
- SIZE OF RESPONSE: Размер ответа сервера может указывать на необычно большие или маленькие ответы, что также может быть признаком аномалии.

Выбор этих полей обусловлен тем, что статистическое отклонение по ним может служить индикатором аномального поведения. Например, нестандартный размер ответа сервера или неожиданный всплеск в определенные периоды времени может указывать на попытки взлома, DDoS-атаки или другие виды аномалий. Анализ кодов ошибок может выявить не только технические проблемы, но и целенаправленные атаки на веб-сервер.

Таким образом, предварительная обработка данных заключается в исключении из рассмотрения нерелевантных полей и фокусировке внимания на тех атрибутах, которые могут предоставить наиболее ценную информацию для обнаружения и анализа аномальных событий. Это создает основу для более эффективной и целенаправленной работы модели кластеризации и классификации событий.

3 Описание алгоритма

Так как основной задачей является разработка модели для кластеризации с выявлением аномалий, для достижения поставленной цели был выбран алгоритм DBSCAN, который является одним из наиболее подходящих методов для работы с данными такого типа.

Алгоритм DBSCAN начинает свою работу с выбора произвольной точки из набора данных и создания вокруг неё области с заданным радиусом, который называется эпсилон-окрестностью. Если в этой окрестности обнаруживается количество точек, превышающее или равное минимально заданному порогу (minPts), то исходная точка классифицируется как корневая, что служит сигналом к формированию нового кластера.

Затем алгоритм переходит к поиску соседних точек. Если в эпсилон-окрестности точки находится меньше, чем minPts соседей, но среди них есть хотя бы одна корневая точка, эта точка помечается как пограничная. Пограничные точки не могут формировать новые кластеры, но могут быть частью уже существующих кластеров.

Все оставшиеся точки, которые не подпадают под критерии корневых или пограничных, считаются выбросами. Они не включаются в кластеры и могут быть интерпретированы как аномалии или шум в данных.

Когда две корневые точки оказываются в пределах эпсилон-окрестности друг от друга, они объединяются в один кластер. Это свойство позволяет кластерам формироваться и расти, поглощая новые корневые точки и их пограничные соседи.

Пограничные точки присоединяются к тем кластерам, к которым принадлежат корневые точки в их окрестности. Таким образом, они служат своего рода мостом, связывающим близлежащие кластеры.

Процесс кластеризации завершается, когда все точки данных либо включены в кластеры, либо классифицированы как выбросы, и не остаётся ни одной

точки, которую можно было бы добавить к существующим кластерам. Это гарантирует, что каждая точка будет рассмотрена и получит своё место в общей структуре данных.

3.3.2 Векторизация TF-IDF

Для преобразования текстовых полей журналов в числовые векторы, которые могут быть использованы в кластеризации DBSCAN, применяется метод TF-IDF. Этот метод векторизации оценивает важность слова в контексте документа относительно всего корпуса документов. Term Frequency (TF) измеряет частоту слова в документе, а Inverse Document Frequency (IDF) уменьшает вес слов, которые встречаются очень часто и поэтому могут быть менее информативными. Произведение TF и IDF дает веса словам, которые помогают выделить наиболее значимые термины для каждого документа.

Векторизация TF-IDF была реализована с использованием библиотеки Scikit-learn[11], которая предоставляет инструменты для преобразования текстовых данных в формат, удобный для анализа. Полученные векторы TF-IDF затем использовались в качестве входных данных для алгоритма DBSCAN, позволяя модели эффективно кластеризовать записи журналов и выявлять аномалии на основе анализа текстового содержимого.

3.3.3 Векторизация BERT

Также использовалась векторизация текстовых полей журналов веб-серверов с помощью модели BERT[12], которая была доработана на данных файлов журналов веб-серверов. В отличие от TF-IDF, данная модель использует

нейронную сеть, состоящую из множества слоев, также учитывает расположение слова в текстовом поле и выдаёт разные векторы для одного и того же слова в зависимости от контекста. Например для поля user-agent, модель BERT могла выделить кластеры по платформам, которые использовали пользователи, а также находила поисковых роботов, являющихся составной частью поисковой системы yandex.

4 АНАЛИЗ ПОЛУЧЕННОГО РЕШЕНИЯ

4.1 Анализ решения, используя генерируемые данные

Использование сгенерированных данных для анализа решения предоставило начальное понимание работы модели кластеризации. Эти данные были созданы таким образом, чтобы имитировать различные типы событий в журналах веб-серверов, включая нормальные запросы и потенциальные аномалии. Однако при анализе результатов выяснилось, что модель, обученная и протестированная на таких данных, не обладает достаточной точностью разбиения на кластеры.

При оценке точности кластеризации, проведенной на основе сгенерированных данных, столкнулись с субъективностью оценки. В отсутствие чётко определённых критериев и методов для точного выявления кластеров и аномалий, оценка эффективности модели не может быть полностью объективной. В частности, для генерируемых данных было замечено, что записи журналов веб-серверов группируются в 10 кластеров в зависимости от поля user-agent. Учитывая, что в искусственном наборе данных присутствует только 10 уникальных значений user-agent, такое разделение не отражает реальную сложность и многообразие поведения пользователей веб-сервера.

Это наблюдение подтверждает, что кластеризация, основанная исключительно на ограниченном числе уникальных значений user-agent, не является оптимальной. Такой подход может привести к излишней генерализации и игнорированию других важных аспектов данных, таких как временные паттерны, коды состояния HTTP и размеры ответов, которые могут предоставить более глубокое понимание поведения системы и потенциальных угроз.

Ниже представлена визуализация в трехмерном пространстве кластерного анализа искусственных данных, которые разбиты на 10 кластеров(рис. 4)

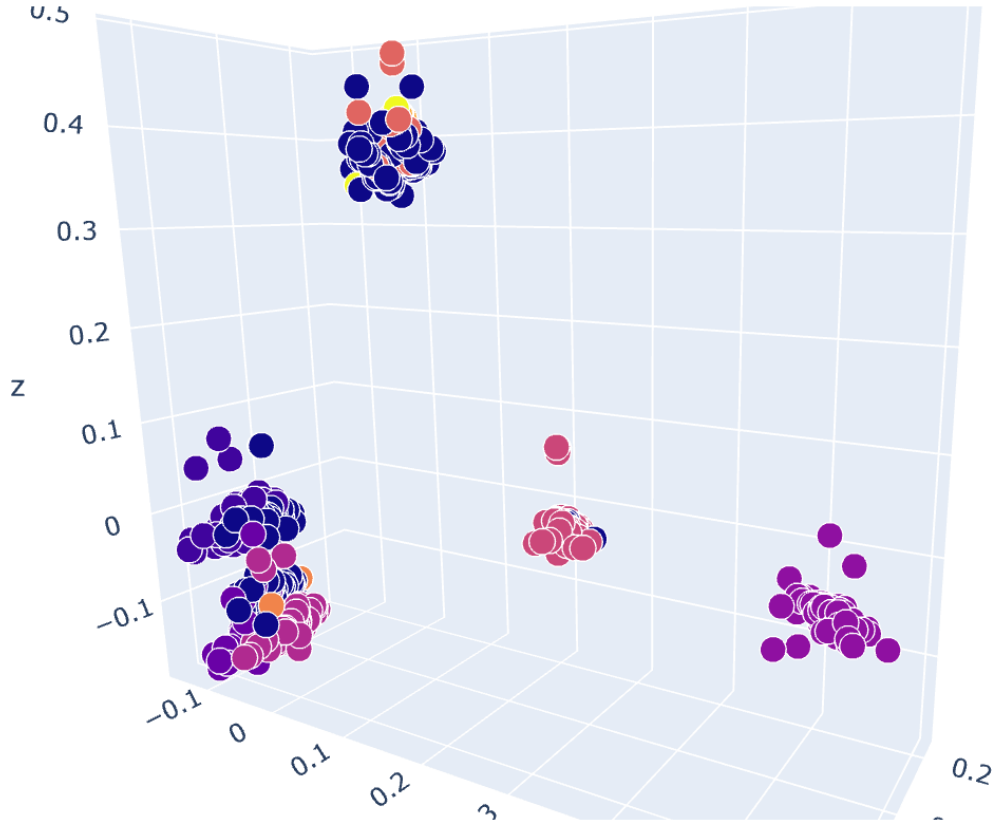

Рисунок 4 — визуализация кластерного анализа искусственных данных

2 Анализ решения, используя реальные данные

Переход к анализу решения на основе реальных данных журналов веб-серверов позволил значительно улучшить качество модели кластеризации. Реальные данные обеспечили более точное и всестороннее представление о поведении системы, включая различные виды запросов и потенциальные угрозы.

Ниже представлена визуализация в трехмерном пространстве кластерного анализа реальных данных, которые разбиты уже на 20 кластеров(рис. 5)

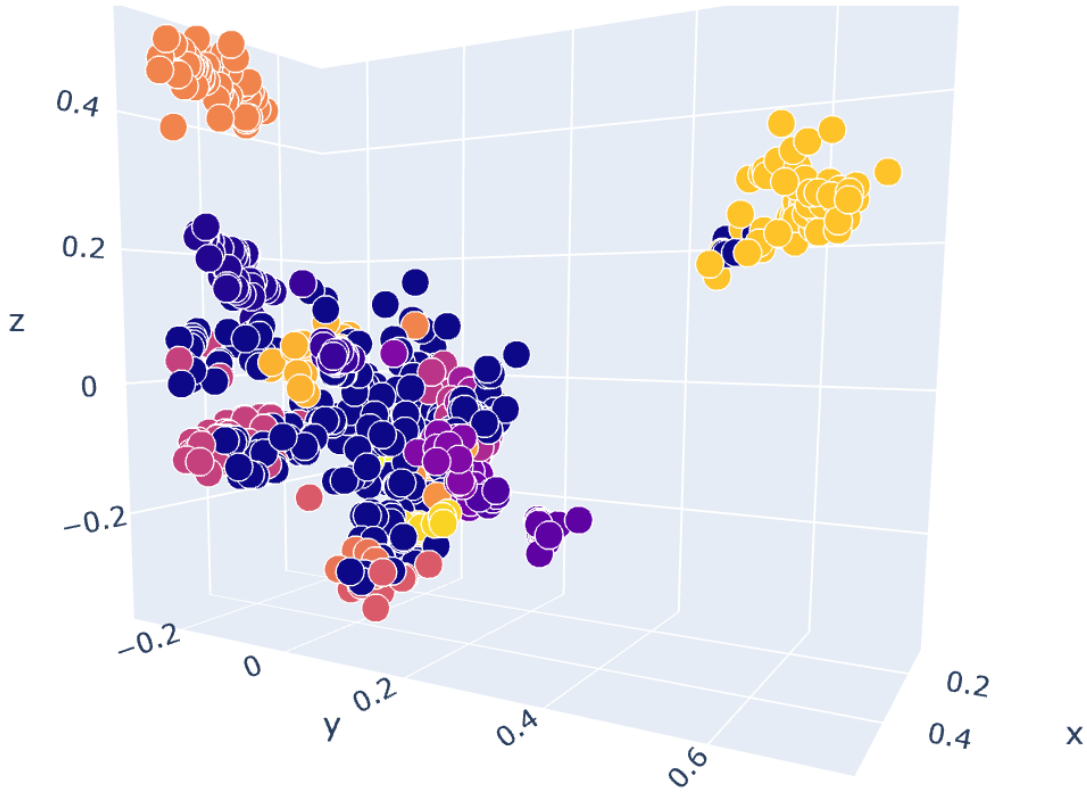

Рисунок 5 — визуализация кластерного анализа реальных данных

5 Обеспечение качества разработки, продукции, программного продукта

5.1 Определение потребителей

Потребителями разработанной системы могут быть:

- Системные администраторы: Используют систему для мониторинга и анализа событий веб-серверов, управления логами и выявления технических проблем, что помогает в обеспечении стабильности и доступности веб-сервисов.
- Специалисты по информационной безопасности: Применяют систему для обнаружения и реагирования на подозрительную активность и потенциальные угрозы безопасности, такие как вторжения или злоупотребления, что повышает уровень защиты данных и инфраструктуры.
- Аналитики данных: Используют систему для извлечения значимых инсайтов из данных журналов, таких как модели поведения пользователей и трафика, что способствует лучшему пониманию потребностей пользователей и оптимизации веб-контента.
- Разработчики веб-приложений: Интересуются системой как инструментом для отслеживания ошибок и недочетов в работе веб-приложений, а также для тестирования новых функций и изменений в коде.
- Управляющий персонал IT-департаментов: Могут использовать систему для стратегического планирования и оптимизации ресурсов, а также для обеспечения соответствия требованиям нормативно-правовых актов по обработке и хранению данных.

Эти группы потребителей могут пользоваться системой для улучшения качества своей работы, повышения эффективности процессов, снижения

рисков и обеспечения более высокого уровня удовлетворенности конечных пользователей.

- **5.2 Функции продукции**

Для того, чтобы ответить на вопрос, какие функции имеет разработанный продукт, необходимо ввести определение функции изделия или услуги. Функции изделия или услуги – это требования и ожидания потребителя, которые могут быть установлены, предполагаются или являются обязательными.

Функциональность разработанного программного продукта должна соответствовать требованиям потребителей. Для модели классификации событий это включает в себя:

- Автоматическую обработку и анализ больших объемов данных журналов.
- Высокую точность классификации событий для минимизации ложных срабатываний.
- Гибкость в настройке параметров для адаптации к специфике различных веб-серверов.
- Интеграцию с существующими системами мониторинга и управления.

5.3 Качество и характеристики

Для обеспечения качества разрабатываемого курса был проведен обзор стандартов, протоколов, отраслевых требований и современных лучших практик, нужных в разработке. При этом качество – степень соответствия совокупности присущих характеристик объекта требованиям (согласно ГОСТ Р ИСО 9000-2015).

Знание протоколов, стандартов и пр. позволяет разработчикам обеспечить соответствие курса современным требованиям, повысить эффективность и удобство использования, улучшить качество курса.

Анализ функциональных требований к разработке отражен в таблице (таблица 5).

Таблица 5 — Функциональные тербования

Функции по группам	Источник	Требование
Удовлетворенность	ГОСТ Р ИСО/МЭК 25010-2015	Модель должна классифицировать записи журнала веб-сервера, основываясь на статистических показателях конкретного журнала веб-сервера.
Изучаемость	ГОСТ Р ИСО/МЭК 25010-2015	Модель должна предоставлять возможность визуализации результатов кластеризации.
Полноценность	ГОСТ Р ИСО/МЭК 25010-2015	Модель должна обеспечивать возможность выявления аномалий.
Доступность	ГОСТ Р ИСО/МЭК 25010-2015	Модель должна обеспечить возможность кластеризовать записи любых журналов веб-серверов.
Конфиденциальность	ГОСТ Р ИСО/МЭК 25010-2015	Доступ к результатам работы модели может быть предоставлен только администраторам веб-серверов.
Документация и отчетность	ГОСТ Р ИСО/МЭК 25010-2015	Ведение документации и составление отчетов о

		процессе кластеризации записей журналов веб-серверов для последующего анализа и использования в практических целях

5.4 Измерение характеристик качества. Операциональное определение

Операциональное определение (ОО) – это уточнение значения того или иного термина применительно к данной системе, находящейся в конкретных условиях и для людей, в ней задействованных.

ОО необходим для уменьшения случаев разночтения и неоднозначности при общении между людьми, задействованными в системе, а также для более точного определения целей и задач, которые необходимо достичь.

ОО должно содержать как минимум три компоненты:

1. Требования или стандарт, относительно которого оценивается результат измерения или испытания (критерий).
2. Метод испытания или процедура измерения свойства объекта (тест)
3. Процедура принятия решения (анализ), которое показывает, соответствует ли результат испытания стандарту.
4. Операциональные определения для разработанного курса отражены в таблице (таблица 6).

Таблица 6 — Операциональные определения.

Требование	Тест		Решение (правило)		
	Характеристика	Метод измерения	Соответствие	Частичное соответствие	Несоответствие
Модель должна выявлять заведомо аномальные записи	Процент выявленных аномалий от общего числа	Подсчёт количества выявленных аномалий относительно общего количества заведомо аномальных записей.	95% или выше	От 60% до 95%	Ниже 60%
Разбиение записей журнала веб-сервера на кластеры должно быть достаточно информативным для анализа	Количество кластеров и их объём	Субъективная визуальная проверка количества кластеров, а также их наполнения	Разбиение на кластеры и выявление аномалий содержит полную информацию для анализа журнала веб-сервера	Разбиение на кластеры и выявление аномалий содержит частичную информацию для анализа журнала веб-сервера	На основе разбиения записей журнала веб-сервера на кластеры и выявленных аномалий невозможно сделать полезных выводов

По итогам рассмотрения операциональных было определено, что разработанный курс удовлетворяет, описанным требованиям, однако существуют также предложения по улучшению продукта, которые сформулированы в таблице (таблица 7).

Таблица 7 — Предложения по улучшению продукта.

Требование	Анализ текущего состояния	Возможные причины невыполнения критерия	Предложение по улучшению
Точность классификации	Текущая точность классификации соответствует стандартам, но имеются случаи ложных срабатываний.	Недостаточная предварительная обработка данных и ограниченное количество обучающих примеров.	Улучшить алгоритмы предварительной обработки данных, увеличить и диверсифицировать набор данных для обучения.
Удобство использования	Пользователи сообщают о сложности в освоении системы.	Недостаточно интуитивно понятный интерфейс и отсутствие подробной документации.	Разработать более удобный пользовательский интерфейс и предоставить обширную документацию и обучающие материалы.

5.5 Выводы

Были определены потребители продукта, а также описаны ключевые функции конечного продукта. Осуществлен обзор существующих стандартов, отраслевых требований и современных лучших практик, необходимых в разработке. По итогам обзора сформулированы функциональные требования к разработанной модели. Описаны операциональные определения, характерные

для модели классификации записей журналов веб-серверов и выявления аномалий. Сделан вывод о том, что разработанная модель удовлетворяет описанным требованиям, а также сформулировано предложение по дальнейшему улучшению модели.

ЗАКЛЮЧЕНИЕ

Был проведен тщательный анализ существующих алгоритмов кластеризации, что позволило оценить их применимость для работы с журналами веб-серверов и выбрать наиболее эффективные в соответствии с задачами классификации событий.

Разработанный алгоритм предварительной обработки данных оказался неотъемлемой частью исследования, поскольку он значительно повысил качество и точность последующей классификации.

Обработка включала в себя очистку данных, нормализацию, а также выбор и преобразование признаков, что создало надежную основу для обучения модели.

Созданная модель классификации событий была обучена на реальных данных. Она продемонстрировала способность различать типы событий, а также выделять среди них аномалии.

Таким образом, результаты данной работы могут быть использованы для анализа событий журналов веб-серверов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. W3Techs [Электронный ресурс]. URL: https://w3techs.com/technologies/overview/web_server (дата обращения: 03.04.2024).
2. Воронцов К. В. Лекции по алгоритмам кластеризации и многомерного шкалирования //М.: МГУ. – 2007.
3. Ester M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise //kdd. – 1996. – Т. 96. – №. 34. – С. 226-231.
4. Apache Documentation [Электронный ресурс]. URL: https://httpd.apache.org/docs/trunk/logs.html (дата обращения: 03.05.2024).
5. Zhao Y., Karypis G., Fayyad U. Hierarchical clustering algorithms for document datasets //Data mining and knowledge discovery. – 2005. – Т. 10. – С. 141-168.
6. Pramanik S., Chandra M. G. Quantum-assisted graph clustering and quadratic unconstrained d-ary optimisation //arXiv preprint arXiv:2004.02608. – 2020.
7. Pourasghar B. et al. A graph-based clustering algorithm for software systems modularization //Information and Software Technology. – 2021. – Т. 133. – С. 106469.
8. Grygorash O., Zhou Y., Jorgensen Z. Minimum spanning tree based clustering algorithms //2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06). – IEEE, 2006. – С. 73-81.
9. Domański Z., Kęsy J. Distribuition of Manhattan distance in square and triangular lattices //Scientific Research of the Institute of Mathematics and Computer Science. – 2005. – Т. 4. – №. 1. – С. 34-37.
10. Jain A. K., Murty M. N., Flynn P. J. Data clustering: a review //ACM computing surveys (CSUR). – 1999. – Т. 31. – №. 3. – С. 264-323.
11. Scikit-learn TfidfVectorizer [Электронный ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (дата обращения: 03.05.2024).

12. Bert base cased finetuned log parser [Электронный ресурс]. URL: https://huggingface.co/Slavka/bert-base-cased-finetuned-log-parser-winlogbeat_nowhitespace (дата обращения: 05.05.2024).

ПРИЛОЖЕНИЕ А

Название файла: ClusterLog.py

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import DBSCAN, OPTICS
import pandas as pd
import scipy.sparse
from sklearn.decomposition import TruncatedSVD
import matplotlib.pyplot as plt
import plotly
import plotly.express as px
import csv

file2 = open("/content/apache_logs.csv", "w")
writer = csv.writer(file2)
head = ["IP", "DATE", "REQUEST", "ENDPOINT", "STATUSCODE", "SIZE
OF RESPONSE", "FAKE URL", "USER AGENT", "BYTES TRANSFERRED"]
writer.writerow(head)

with open("/content/apache_logs.txt", "r") as file1:
    for line in file1:
      IP = line.split(" ")[0]
      pos1 = line.find("[")+1
      pos2 = line.find("]")
      DATE = line[pos1:pos2]
      row = [IP, DATE, "-", "-", "-", "-", "-", "-", "-"]
      writer.writerow(row)

file2.close()
parser = pd.read_csv("/content/apache_logs.csv")

data_csv = pd.read_csv("/content/logfiles.csv")
data_csv_clean = data_csv.drop(["DATE", "ENDPOINT"], axis=1)
data_csv_clean = data_csv_clean[:1000]
data_csv_clean_string = data_csv_clean.to_string(header=False, in-
dex=False)
data_csv_clean_list = data_csv_clean_string.split("\n")

with open("/content/apache_logs.txt", "r") as f:
```

```
        data = f.readlines()
    data = data[:1000]
    vectorizer = TfidfVectorizer()
    X = vectorizer.fit_transform(data)
    clustering = DBSCAN(eps=0.02, min_samples=3, metric= "cosine")
    labels = clustering.fit_predict(X)
    pca = TruncatedSVD(n_components=3)
    X_downsample = pca.fit_transform(X)
    df = pd.DataFrame(data = {
        "x": X_downsample[:, 0],
        "y": X_downsample[:, 1],
        "z": X_downsample[:, 2],
        "color": clustering.labels_,
        "text": data
    })
    fig = px.scatter_3d(df, x="x", y="y", z="z", color="color",
size=[1] * len(df), hover_data="text", opacity=1)
    fig.show()
    df[df["color"] == -1]["text"].to_csv("/content/cluster_all_anom-
aly.csv", index=False)
    print(pca.explained_variance_ratio_)
    print(pca.explained_variance_ratio_.sum())
    vectorizer = TfidfVectorizer()
    Z = vectorizer.fit_transform(data_csv_clean_list)
    clustering = DBSCAN(eps=0.68, min_samples=5, metric= "cosine")
    labels = clustering.fit_predict(Z)
    pca = TruncatedSVD(n_components=3)
    Z_downsample = pca.fit_transform(Z)
    df1 = pd.DataFrame(data = {
        "x": Z_downsample[:, 0],
        "y": Z_downsample[:, 1],
        "z": Z_downsample[:, 2],
        "color": clustering.labels_,
        "text": data_csv_clean_list
    })
    fig = px.scatter_3d(df1, x="x", y="y", z="z", color="color",
size=[1] * len(df1), hover_data="text", opacity=1)
    fig.show()
```

Название файла: TestFileGenerator.py

```
from datetime import date
import random
from random import randint, choice
import sys
import time
import faker
from datetime import datetime
import os
import csv
os.environ['TZ'] = 'Asia/Kolkata'
fak = faker.Faker()

def str_time_prop(start, end, format, prop):
    stime = time.mktime(time.strptime(start, format))
    etime = time.mktime(time.strptime(end, format))
    ptime = stime + prop * (etime - stime)
    return time.strftime(format, time.localtime(ptime))

def random_date(start, end, prop):
    return str_time_prop(start, end, '%d/%b/%Y:%I:%M:%S %z', prop)


dictionary = {'request': ['GET', 'POST', 'PUT', 'DELETE'],
              'endpoint': ['/usr', '/usr/admin', '/usr/admin/de-
veloper', '/usr/login', '/usr/register'],
              'statuscode': ['303', '404', '500', '403', '502',
'304','200'],
              'username': ['james', 'adam', 'eve', 'alex',
'smith', 'isabella', 'david', 'angela', 'donald', 'hilary'],
              'ua' : ['Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:84.0) Gecko/20100101 Firefox/84.0',
            'Mozilla/5.0 (Android 10; Mobile; rv:84.0) Gecko/84.0
Firefox/84.0',
            'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWeb-
Kit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Safari/537.36',
```

```
                'Mozilla/5.0 (Linux; Android 10; ONEPLUS A6000) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.141 Mobile Safari/537.36',
                'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4380.0 Safari/537.36 Edg/89.0.759.0',
                'Mozilla/5.0 (Linux; Android 10; ONEPLUS A6000) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.116 Mobile Safari/537.36 EdgA/45.12.4.5121',
                'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 OPR/73.0.3856.329',
                'Mozilla/5.0 (Linux; Android 10; ONEPLUS A6000) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Mobile Safari/537.36 OPR/61.2.3076.56749',
                'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/7046A194A',
                'Mozilla/5.0 (iPhone; CPU iPhone OS 12_4_9 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.2 Mobile/15E148 Safari/604.1'],
                'referrer' : ['-',fak.uri()]}

    f = open("logfiles.log","w")
    c = open("logfiles.csv", "w")
    writer = csv.writer(c)
    head = ["IP", "DATE", "REQUEST", "ENDPOINT", "STATUSCODE", "SIZE OF RESPONSE", "FAKE URL", "USER AGENT", "BYTES TRANSFERRED"]
    writer.writerow(head)
    for _ in range(1,10000):
        IP = fak.ipv4()
        DATE          =          random_date("01/Jan/2018:12:00:00 +0530","01/Jan/2020:12:00:00 +0530",10)
        REQUEST = choice(dictionary['request'])
        ENDPOINT = choice(dictionary['endpoint'])
        STATUSCODE = choice(dictionary['statuscode'])
        SIZE_OF_RESPONSE = str(int(random.gauss(5000,50)))
        #FAKE_URL = choice(dictionary['referrer'])
        FAKE_URL = fak.uri()
        USER_AGENT = choice(dictionary['ua'])
```

```
            BYTES_TRANSFERRED = random.randint(1,5000)
            row = [IP, DATE, REQUEST, ENDPOINT, STATUSCODE, SIZE_OF_RE-
SPONSE, FAKE_URL, USER_AGENT, BYTES_TRANSFERRED]
            f.write('%s - - [%s] "%s %s HTTP/1.0" %s %s "%s" "%s" %s\n' %
(IP, DATE, REQUEST, ENDPOINT, STATUSCODE, SIZE_OF_RESPONSE, FAKE_URL,
USER_AGENT, BYTES_TRANSFERRED))
            writer.writerow(row)
      c.close()
      f.close()
```