

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Э. БАУМАНА»  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)  
(МГТУ им. Н.Э.БАУМАНА)



ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»  
КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ  
ТЕХНОЛОГИИ»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ  
НА ТЕМУ:**

**МЕТОД ОБНАРУЖЕНИЯ ВЫБРОСОВ ВРЕМЕННЫХ РЯДОВ**

|                  |       |                |
|------------------|-------|----------------|
| Студент ИУ7-82   | _____ | А. И. Капустин |
| Руководитель ВКР | _____ | А. А. Оленев   |
| Консультант      | _____ | _____          |
| Консультант      | _____ | _____          |
| Нормоконтролер   | _____ | _____          |

Москва, 2020

T3

T3

план

## **Реферат**

РПЗ 50 страниц, 10 рисунков, 9 таблиц, 27 источников.

Цель квалификационной работы бакалавра – разработка программного комплекса поиска аномалий. В квалификационной работе бакалавра проведён обзор алгоритмов поиска аномалий, а так же обзор алгоритмов для их улучшения, проанализирована предметная область. Предложен новый метод поиска аномалий. Создан программный продукт, реализующий предложенный метод, а так же вспомогательный программный продукт, которые осуществляют сбор данных для анализа.

# Содержание

|   |    |
|---|----|
| Введение  | 9  |
| 1 Аналитический раздел                                | 10 |
| 1.1 Цель и задачи работы                              | 10 |
| 1.2 Что такое спортивное прогнозирование              | 10 |
| 1.3 Сложность прогнозирования результатов             | 10 |
| 1.4 Основные правила игры в теннис                    | 11 |
| 1.5 Основные подходы прогнозирования теннисных матчей | 12 |
| 1.5.1 Метод подсчёта очков                            | 12 |
| 1.5.1.1 Вероятность выиграть гейм                     | 12 |
| 1.5.1.2 Вероятность выиграть сет                      | 13 |
| 1.5.1.3 Вероятность выиграть матч                     | 13 |
| 1.5.2 ff  | 14 |
| 1.5.3 Машинное обучение                               | 14 |
| 1.5.3.1 Логистическая регрессия                       | 15 |
| 1.6 Обнаружение аномалий                              | 16 |
| 1.6.1 Классификация методов обнаружений аномалий      | 16 |
| 1.7 Результат метода обнаружения аномалий             | 17 |
| 1.8 Виды аномалий                                     | 17 |
| 1.8.1 Нормализация данных                             | 19 |
| 1.8.1.1 Основные методы нормализации данных           | 19 |
| 1.9 Неконтролируемые алгоритмы обнаружения аномалий   | 20 |
| 1.9.1 Вероятностно-генеративные методы                | 20 |
| 1.9.2 Линейные методы                                 | 21 |
| 1.9.3 Метрические методы                              | 22 |
| 1.9.3.1 Базовые понятия                               | 22 |
| 1.9.3.2 К ближайших соседей(kNN)                      | 23 |
| 1.9.3.3 Оптимизация Рамасвани                         | 23 |
| 1.9.3.4 Методы Кнора-Реймонда и Байерса-Рейтери       | 24 |
| 1.9.3.5 Метод Танга                                   | 25 |
| 1.9.3.6 Метод ОДИН(ODIN)                              | 26 |
| 1.9.3.7 Локальный коэффициент выбросов(LOF)           | 26 |
| 1.9.3.8 Компонентный коэффициент выбросов(COF)        | 27 |
| 1.9.3.9 Метод вероятностей локальных выбросов(LoOP)   | 28 |
| 1.9.4 Параметрические методы                          | 29 |
| 1.10 Методы улучшения алгоритмов поиска аномалий      | 29 |
| 1.10.1 Семплирование                                  | 30 |
| 1.10.2 Ансамблирование голосованием                   | 30 |
| 1.10.3 Итеративный отбор                              | 31 |

|       |   |    |
|-------|---|----|
| 1.11  | Выводы . . . . .                                      | 32 |
| 2     | Конструкторский раздел . . . . .                      | 33 |
| 2.1   | Метод обнаружения аномалий . . . . .                  | 33 |
| 2.2   | Подбор параметров алгоритмов . . . . .                | 33 |
| 2.2.1 | Анализ результатов работы методов . . . . .           | 33 |
| 2.2.2 | Подбор параметров . . . . .                           | 34 |
| 2.3   | К ближайших соседей . . . . .                         | 35 |
| 2.4   | Локальный коэффициент выбросов . . . . .              | 36 |
| 2.5   | Компонентный коэффициент выбросов . . . . .           | 37 |
| 2.6   | Собирающее данные для анализа ПО . . . . .            | 37 |
| 2.6.1 | Собираемые данные . . . . .                           | 38 |
| 2.6.2 | Клиентская часть . . . . .                            | 38 |
| 2.6.3 | Серверная часть . . . . .                             | 39 |
| 2.6.4 | Динамическое изменение данных . . . . .               | 39 |
| 3     | Технологический раздел . . . . .                      | 40 |
| 3.1   | Выбор средств разработки . . . . .                    | 40 |
| 3.1.1 | Выбор целевой платформы . . . . .                     | 40 |
| 3.1.2 | Выбор языка программирования . . . . .                | 40 |
| 3.1.3 | Выбор среды разработки и отладки . . . . .            | 41 |
| 3.2   | Выбор базы данных . . . . .                           | 41 |
| 3.3   | Система контроля версий . . . . .                     | 41 |
| 3.4   | Библиотека Galgo . . . . .                            | 42 |
| 3.5   | Формат файлов . . . . .                               | 42 |
| 3.6   | Библиотека KUserFeedback . . . . .                    | 43 |
| 3.7   | Установка программного обеспечения . . . . .          | 43 |
| 3.8   | Пример работы программы . . . . .                     | 43 |
| 3.9   | Использование программы для поиска аномалий . . . . . | 44 |
| 4     | Исследовательский раздел . . . . .                    | 45 |
| 4.1   | Полнота и точность. Площадь под РОК-кривой . . . . .  | 45 |
| 4.2   | Сравнение алгоритмов поиска аномалий . . . . .        | 46 |
| 4.3   | Рекомендации к использованию метода . . . . .         | 47 |
|       | Заключение . . . . .                                  | 48 |
|       | Список использованных источников . . . . .            | 49 |

## **Глоссарий**

**Априорная информация** — Информация, которая была получена ранее рассматриваемого момента времени[1].

**Теннис** — Спортивные турниры, проводимые по правилам Международной Федерации Тенниса[2]



## **Введение**

Задача поиска аномалий является одной из классических задач машинного обучения. В настоящее время задачу поиска аномалий активно решают во многих областях жизнедеятельности:

- а) Защита информации и безопасность
- б) Социальная сфера и медицина
- в) Банковская и финансовая отрасль
- г) Распознавание и обработка текста, изображений, речи
- д) Другие сферы деятельности(например, мониторинг неисправностей механизмов)

Задачу поиска выбросов, как частный случай задачи поиска аномалий, так же занимаются во всех вышеперечисленных отраслях.

Количество данных в мире удваивается примерно каждые два года[?]. Поэтому актуальной задачей является разработка новых методов и усовершенствование старых методов поиска выбросов.

В данной работе предлагается новый метод, позволяющий найти аномалии в выборках данных.

## **1 Аналитический раздел**

### **1.1 Цель и задачи работы**

Целью данной работы является создание программного комплекса результатов теннисных матчей на основе априорной информации. Для достижения данной цели необходимо решить следующие задачи:

- проанализировать предметную область и существующие методы прогнозирования результатов теннисных матчей
- разработать метод прогнозирования результатов теннисных матчей
- создать ПО, обрабатывающее данные для анализа
- создать ПО, реализующее разработанный метод прогнозирования результатов теннисных матчей

### **1.2 Что такое спортивное прогнозирование**

Спортивное прогнозирование предполагает предугадывание результатов предстоящих спортивных событий или контрольных результатов, которые спортсмен) или команда спортсменов) показывает на спортивных соревнованиях[3]. Прогнозы даются на конкретные события в конкретные моменты времени, на результат совокупности событий, ограниченных во времени (например, соревновательный сезон). Наиболее распространены прогнозы на результаты конкретного матча и сезона в целом. Прогнозы могут осуществляться на основе алгоритмов анализа информации, экспертной оценке, а так же комбинацией экспертной оценки. В данной работе будет рассмотрено прогнозирование на основе анализа априорной информации. Т.е. прогноз на какое-либо событие будет даваться до его начала и текущая информация о ходе события не будет учитываться.

### **1.3 Сложность прогнозирования результатов**

Большинство исследователей используют различную статистическую информацию для составления прогнозов. Из массива накопленных данных они выбирают небольшое количество наиболее важных показателей (или же останавливаются только на одном из них) за ограниченный промежуток времени, которые подаются на вход алгоритму. Какие показатели считать значимыми определяет автор алгоритма на основе экспертной оценки или каких-то дополнительных алгоритмов "отбора наиболее важных показателей". Например, в случае американского футбола[4] это могут быть

- а) Время владения мячом
- б) Проводился ли матч дома или в гостях
- в) Общее количество ярдов

- г) Разница в атакующих ярдах
- д) Количество потерь мяча

Или же в случае баскетбола, например, может браться следующий набор показателей:

- а) Количество травмированных игроков
- б) Количество выигранных матчей подряд перед данной игрой
- в) Усталость команды. Показатель считается на основе расстояния, которое пришлось преодолеть команде, чтобы провести последние 7 игр
- г) Средний домашний, гостей и общий процент побед
- д) Рейтинг команды в "рейтинге нападения" "рейтинге защиты" и "общем рейтинге" (подробнее методики подсчёта рейтингов описываются в [5])

Большая часть статей представляет команду как некое единое целое, упуская из виду каждого игрока команды. Учёт информации каждом игроке - нетривиальная задача. Поэтому для упрощения создания алгоритма прогнозирования с учётом каждого игрока команды был выбран вид спорта - одиночный теннис. В каждой команде по одному игроку, два возможных исхода матча - победа или поражение.

#### **1.4 Основные правила игры в теннис**

Ниже приведены основные правила, которые могут повлиять на составление алгоритмов прогнозирования. С полной версии правил можно ознакомиться на сайте международной теннисной федерации[2].

- а) Мяч должен приземлиться в пределах теннисного поля, чтобы продолжить игру. Если игрок мяч приземляется за пределами поля, то это приводит к потере им очков.
- б) Игроки не могут дотронуться до сетки или перейти на сторону противника
- в) Игроки не могут носить мяч или ловить его ракеткой
- г) Игроки не могут дважды ударить по мячу
- д) Игрок должен дождаться пока мяч пересечёт сетку, прежде чем ударить по нему
- е) Игрок, который не возвращает мяч на половину поля противника, прежде чем он отскочит дважды, считается проигравшим
- ж) Если мяч ударяет или касается игроков, то это карается штрафом
- з) Перед тем как отбить подачу, мяч должен отскочить от поля.
- и) Очки -наименьшая единица изменения. Приращение очков идёт в формате 0-15-30-40-гейм.

к) Гейм состоит из 4 очков и выигрывается, когда игрок набирает 4 очка с преимуществом не менее двух очков.

л) Сет состоит из 6 геймов и выигрывается игроком, который набирает 6 геймов с минимальным отрывом в 2 очка.

м) Дополнительный сет - сет, который разыгрывается при достижении игроками счёта 6-6 по геймам. При

н) Максимальное количество сетов в матче может достигать 3 или 5. По достижении 2 или соответственно 3 выигранных сетов матч заканчивается.

о) "Ровно" происходит когда достигнут счёт по очкам 40-40. Чтобы выиграть гейм, игрок должен выиграть два последовательных очка. Если игрок выигрывает одно очко, то счёт в гейме становится "больше" но если он теряет следующее очко, то счёт возвращается к "ровно" .

п) Когда в гейме достигнут счёт 6-6, то играется тай-брейк. В розыгрыше тай-брейке используются особые правила набора очка. Победителем считается игрок, набравший не менее 7 очков с преимуществом два очка над оппонентом.

## **1.5 Основные подходы прогнозирования теннисных матчей**

Существует три основных подхода к прогнозированию теннисных матчей: методы попарного сравнения, методы вычисления очков и методы машинного обучения.

### **1.5.1 Метод подсчёта очков**

Методы вычисления очков (также их называют иерархическими методами) фокусируются на оценке вероятности выигрыша каждого очка в матче. В этом подходе предполагается, что достаточно знать вероятность выигрыша игроком А  $p_a^r$  очка на своей подаче и вероятность выигрыша очка игроком В на своей подаче  $p_b^r$ . Аналогичные подходы применяются при подсчете вероятности выигрыша игрока в бадминтоне и сквоше[6]. Впервые такой метод был применен к теннису в работах Кси и Бюрича[7], Картера и Крю[8], Полларда[9]. Такие работы одинаково определяют вероятность выигрыша очка в матче как равномерную случайную величину, т.е. вероятности  $p_a^r$  и  $p_b^r$  принимаются за постоянные величины на протяжении всего матча. Анализ статистических показателей показывает, что такое предположение допустимо[10], т.к. отклонения от ожидаемой величины невелики.

#### **1.5.1.1 Вероятность выиграть гейм**

Игрок А может выиграть гейм у игрока В со счётом [4,0], [4,1], 4,2] в случае если игрок А выиграл 4 очка за гейм, а игрок В выиграл не более двух.

Если же счёт в гейме стал [3,3], то такая ситуация называется "ровно" . В этом случае игрок А может выиграть гейм, закончив его с итоговым счетом  $[n + 5, n + 3]$ , где  $n \geq 0$ . Чтобы посчитать вероятность  $p_G^R$  выигрыша игроком А гейма на своей подаче, введем следующие обозначения:  $p_A^R$  - вероятность выигрыша очка на своей подаче игроком А,  $q_A^R = 1 - p_A^R$ ,  $q_A^G = 1 - p_A^G$ ,  $p_A^G(i, j)$  - вероятность того, что  $i$  очков в гейме наберет игрок А, а  $j$  очков в гейме наберет игрок В. В результате получим следующую формулу:

$$p_A^G = \sum_{j=0}^2 p_A^G(4, j) + p_A^G(3, 3) \sum_{n=0}^{\infty} p_a^{DG}(n + 2, n) \quad (1.1)$$

Пусть  $p_A^{DG}(n + 2, n)$  - вероятность того, что игрок А выиграет с преимуществом в 2 мяча после того как был достигнут счет [3,3] на подаче игрока А.

$$p_A^{DG}(n + 2, n) = \sum_{j=0}^n (p_A^R q_A^R)^j (q_A^R p_A^R)^{n-j} \frac{n!}{j!(n-j)!} (p_A^R)^2 = (p_A^R)^2 (p_A^R q_A^R)^n 2^n \quad (1.2)$$

В результате из формул 1 и 2 можно вывести следующую формулу - вероятность выигрыша игроком А гейма на своей подаче.

$$p_A^G = (p_A^R)^4 (1 + 4q_A^R + 10(q_A^R)^2) + 20(p_A^R q_A^R)^3 (p_A^R)^2 (1 - 2q_A^R)^{-1} \quad (1.3)$$

### 1.5.1.2 Вероятность выиграть сет

Пусть  $p_A^S$  - вероятность того, что игрок А выиграет сет у игрока В, если игрок А начинает подавать первым,  $q_A^S = 1 - p_A^S$ . Чтобы вывести  $p_A^G$  из  $p_A^G$  и  $p_A^G$ , определим  $p_A^S(i, j)$  как вероятность того, что в розыгрыше сета игрок А выиграет  $i$  геймов, игрок В  $j$  геймов . Тогда

$$p_A^S = \sum_{j=0}^4 p_A^S(6, j) + p_A^S(7, 5) + p_A^S(6, 6) p_A^T \quad (1.4)$$

Где  $p_A^T$  вероятность того, что игрок А выиграет 13-очковый тай-брейк , начинающийся с подачи игрока А.

### 1.5.1.3 Вероятность выиграть матч

Пусть  $p_A^M$  - вероятность того, что игрок А выиграет матч у игрока В. Определим  $p_{AB}^M(i, j)$  как вероятность, что игрок а выиграет в матче после того как количество выигранных им сетов достигнет  $i$ , а количество выигранных сетов у игрока В будет  $j$ , где матч начинается с подачи игрока А, а заканчивается на подаче В. Аналогично -  $p_{AA}^M$ .

Аналогично зададим вероятности победы для сетов  $p_{AB}^S, p_{AA}^S, p_{BA}^S, p_{BB}^S$ , где  $P_{XY}^S$  - вероятность того, что игрок сет закончится выигрышем игрока X, начинающийся с подачи X и заканчивающийся на подаче Y. Очевидно, что количество разыгранных геймов для  $p_{XX}^S$  будет нечетным. Ограничим формулу 1.4

$$p_A^S = \sum_{j=1,3} p_A^S(6,j) + p_A^S(6,6)p_A^T \quad (1.5)$$

Если игрок X подает в первом сете матча, а Y в последнем, количества геймов в сете будет нечетным. Преобразуем 1.4.

$$p_A^S = \sum_{j=0,2,4} p_A^S(6,j) + p_A^S(7,5) \quad (1.6)$$

Итоговая формула для матчей где для достижения победы надо первым выиграть 3 сета(матчи мужского тенниса), будет выглядеть следующим образом(с более детальным выводом формул можно ознакомиться в [9], [10]).

### 1.5.2 ff

### 1.5.3 Машинное обучение

Машинное обучение - это область искусственного интеллекта (ИИ), которая изучает алгоритмы, которые обучаются на основе данных. Алгоритмы машинного обучения с учителем ставят своей задачей вывести функцию преобразования данных из помеченных данных, где помеченные данные представляют собой пары : вектор значений - результат. В теннисе обычно используют исторические данные для формирования данных для обучения. Например, вектор значений может содержать информацию о матче и игроках, а результатом соответственно будет служить исход матча. Выбор оптимальных параметров для формирования вектора напрямую влияет на точность получаемой функции. К проблеме прогнозирования теннисных матчей можно подойти двумя способами:

— Регрессионный подход, где результатом будет являться действительное число - вероятность выиграть матч. Вероятности выиграть матч неизвестны для исторических данных, что вынуждает помечать их в исторических данных метками 0 и 1.

— Подход бинарной классификации. Тогда результатом работы алгоритма будет предсказание выигрыша или проигрыша матча.

### 1.5.3.1 Логистическая регрессия

Несмотря на своё название, логистическая регрессия является алгоритмом классификации. Свойства логической функции являются ключевыми для алгоритма. Логистическая функция определяется как:

$$\sigma = \frac{1}{1 + e^{-t}} \quad (1.7)$$

Логистическая функция отображает вещественные числа в диапазоне  $(-\infty, +\infty)$  в диапазон  $[0,1]$ . Выходное значение логистической функции может интерпретироваться как вероятность.

Логистическая регрессия для прогнозирования матчей состоит из  $N$  свойств  $x = (x_1, x_2, \dots, x_n)$  и вектора  $n+1$  вещественных параметров модели  $\beta = (\beta_0, \beta_1, \dots, \beta_n)$ . Чтобы сделать прогноз при помощи модели, нужно преобразовать точку в  $n$ -мерном пространстве свойств в вещественное число

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \quad (1.8)$$

Подставляя  $z$  в формулу 1.9 получим

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (1.9)$$

Обучение модели состоит в оптимизации параметров  $\beta$ , таким образом, чтобы модель "как можно точнее" воспроизводила результаты матчей из данных для обучения. Это делается при помощи минимизации функции логистических потерь 1.10, которая показывает меру ошибку прогнозирования результатов матчей тренировочных данных.

$$L(p) = \frac{-1}{N} \sum_n \quad (1.10)$$

Выбросы являются следствием:

- а) ошибок в данных
- б) неверно классифицированных объектов
- в) присутствием объектов других выборок
- г) намеренным искажением данных

На рисунке 1.1 находится три вида точек: зеленые, желтые, красные. Множество зеленых точек представляют собой "нормальные" данные. Множество желтых точек означает выбросы в "слабом смысле". Они незначительно отклоняются от основных нормальных данных. Красные же точки являются аномальными - выбросами "в сильном смысле". Они значительно отклоняются от нормальных данных. В данной работе будет изучаться вопрос нахождения "сильных выбросов" и критериев отличия сильного выброса от основных данных. В дальнейшем под словом "выброс" будет подразумеваться

"сильный выброс" , а под аномалией - выброс(выброс - частный случай аномалии). Понятие аномалии интерпретируют по-разному в зависимости от характера данных. Обычно аномалией называют некоторое отклонение от нормы. В дальнейшем будет дано несколько более формальных определений аномалий, специфичных для метода их определений.

## **1.6 Обнаружение аномалий**

В машинном обучении обнаружение "ненормальных" экземпляров в наборах данных всегда представляло большой интерес. Вероятно, первое определение было дано Граббсом[4] в 1969 году: "Относительное наблюдение или выброс - это элемент выборки, который, заметно отличается от других членов выборки, в которых он встречается " . Это определение является актуальным и сегодня, но мотивация для обнаружения аномалий изменилась. Тогда основная причина поиска аномалий заключалась в том, чтобы удалить выбросы из набора данных для обучения, поскольку используемые алгоритмы, были весьма чувствительны к выбросам в данных. Эта процедура также называется очищением данных. После разработки классификаторов устойчивых к наличию аномалий в обучающем наборе данных, интерес к их поиску угас. Однако, в начале 21 века в связи с развитием интернета и значительным увеличением объема собираемых данных для анализа, исследователи стали больше интересоваться аномалиями, поскольку они оказывались часто связаны с особенно интересными событиями. В этом контексте определение Граббса также было расширено, так что сегодня аномалии имеют две важные характеристики:

- а) Аномалия отличается от нормы по своим особенностям
- б) Аномалия редко встречается в наборе данных по сравнению с "нормальными" данными

### **1.6.1 Классификация методов обнаружений аномалий**

Классическая система классификации предполагает предварительное обучение на обучающем наборе данных и последующую классификацию на основе этого набора. Данные делятся на "обучающую выборку" - данные, при помощи которых алгоритм обучает классификатор и, "тестовую выборку" - данные, при анализе которых классификатор остается неизменным. Тестовая выборка нужна для того чтобы проверить корректность обучения классификатора.

Однако, в случае с поиском аномалий, возможны варианты, отличающиеся от классического. Подходящий метод классификации выбирается на основе наличия разметки данных. Выделяются три основных класса методов:



а) Обучение с учителем. Для обучения необходимо наличие полностью размеченных данных для обучения и для тестов. Классификатор обучается один раз и применяется впоследствии. В связи с тем, что для многих наборов данных заранее неизвестно, что является аномалией, а что нет, применение этого метода ограничено.

б) Обучение с частичным привлечением учителя. Для обучения необходимо наличие тестового и учебного набора данных. Однако, в отличие от обучения с привлечением учителя, разметка данных не требуется. Все данные, представленные в выборках, считаются нормальными. На основе этих данных строится некая модель. Все данные, отклоняющиеся от этой модели, считаются аномальными. Эта идея также известна как "одноклассовая" классификация [5].

в) Обучение без учителя. Самый гибкий способ, который не требует разметки набора данных. Идея заключается в том, что алгоритм обнаружения аномалий оценивает данные исключительно на основе внутренних свойств набора данных, что является нормальным, а что является выбросом. В данной работе основное внимание будет уделено именно этому способу. Так же этот способ называют "неконтролируемый способ обнаружения аномалий".

## **1.7 Результат метода обнаружения аномалий**

В результате работы алгоритма обнаружения аномалий с элементом данных связывается метка или оценка достоверности (показатель аномальности). Метка - показатель, который принимает нулевое значение, в случае если она связана с нормальными данными и единицу в противном случае. Оценка показывает вероятность того, что элемент является аномалией. Для разных алгоритмов используются разные шкалы оценок, поэтому приведение конкретных примеров оценок будет некорректным. В алгоритмах метода обучения с учителем зачастую используются метки как выходные данные, в алгоритмах с частичным привлечением учителя и без учителя обнаружения аномалий чаще встречаются оценки.

## **1.8 Виды аномалий**

Основная идея алгоритмов обнаружения аномалий заключается в обнаружении экземпляров данных в наборе данных, которые отклоняются от нормы. Однако на практике существует множество случаев, когда это основное предположение является неоднозначным. На рис 1.1 показаны некоторые из этих случаев с использованием простого двумерного набора данных. Две аномалии могут быть легко идентифицированы визуально: красные точки (обозначены треугольниками) сильно отличаются от значений параметров от областей плотной группировки точек. Если смотреть на весь

набор данных в целом, то фиолетовую точку(обозначена квадратом) можно отнести к тому же классу, что и зеленые точки(обозначены кругами). Однако, если сфокусироваться только на кластере зеленых точек и сравнить его с фиолетовой точкой, пренебрегая всеми другими точками, то её можно рассматривать как аномалию. Поэтому фиолетовая точка называется локальной аномалией, так как она аномальна по сравнению с ее близкой окрестностью. В зависимости от цели анализа, локальные аномалии могут представлять интерес или нет. Другой вопрос заключается в том, что следует ли рассматривать точки черного кластера(обозначены звездочками) как три аномалии или как (небольшой) кластер. Такие небольшие кластеры называются микрокластерами. Показатели аномальности у точек этого кластера выше, чем у точек зеленого кластера, но меньше, чем у красных точек. Этот простой пример показывает, что задача нахождения аномалий не всегда тривиальна, а вычисление показателя аномальности иногда полезнее, чем проставление двоичной метки.

Задача обнаружения одиночных аномальных экземпляров в крупном наборе данных называется обнаружением точечных аномалий[2]. Большинство неконтролируемых алгоритмов обнаружения относятся к этому типу. Если же аномалии составляют заметный процент от набора данных, то задачу поиска аномалий называют задачей обнаружения коллективных аномалий. Пусть аномалии представляют собой некое множество, тогда необязательно каждый элемент этого множества должен быть аномальным. Возможен вариант когда только определенная их комбинация определяет аномалию. Третий вид - контекстуальные аномалии. Элемент выборки в отрыве от своего контекста может казаться нормальным. Однако, если рассмотреть контекст этого элемента, то очевидным станет его аномальная природа. Распространенным контекстом является время. В качестве примера предположим, что измеряется температура в диапазоне

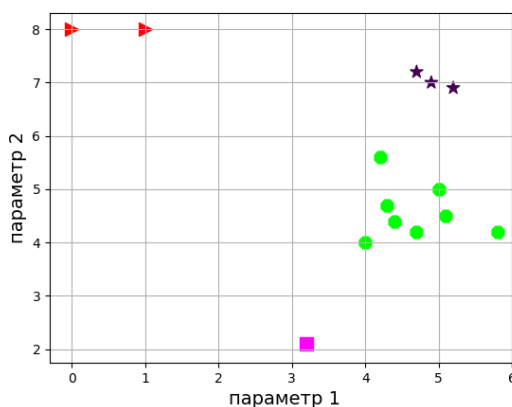


Рисунок 1.1 — Аномалии в двумерном пространстве

от  $-30^{\circ}\text{C}$  до  $+40^{\circ}\text{C}$  в течение года. Таким образом, температура  $25^{\circ}\text{C}$  кажется довольно нормальной, но когда учитывается контекстное время (например, месяц), такая высокая температура  $25^{\circ}\text{C}$  в течение зимы будет рассматриваться как аномалия.

Алгоритмы обнаружения точечных аномалий также можно использовать для обнаружения контекстуальных и коллективных аномалий. Для этого нужно включить контекст в алгоритм как параметр алгоритма. В вышеприведенном примере включение месяца как дополнительного параметра поможет обнаружить аномалию. Однако в более сложных сценариях может потребоваться один или несколько новых параметров, чтобы преобразовать задачу определения контекстной аномалии в задачу обнаружения точечной аномалии. Для того, чтобы преобразовать задачу поиска коллективной аномалии в задачу поиска одиночной, нужно произвести изменения начального набора данных. Для этого можно использовать корреляцию, агрегацию и группировку. Преобразование может быть нетривиальным.[7] . Преобразование требует глубоких знаний о наборе исходных данных и часто приводит к существенным искажениям при переводе данных в новый формат. Такое семантическое преобразование называется генерированием представления данных(*англ. data view generation*).

Таким образом можно сделать вывод, что многие задачи обнаружения аномалий требуют предварительной обработки данных перед передачей их на вход алгоритму. В противном случае можно получить формально верные, но фактические бесполезные результаты.

### **1.8.1 Нормализация данных**

После получения предварительно обработанного датасета для поиска точечной аномалии, последним шагом перед передачей в алгоритм, является нормализация данных. Нормализация данных предназначена для устранения зависимости от выбора единицы измерения и заключается в преобразовании диапазонов значений всех атрибутов к стандартным интервалам([0,1] или [-1,1])[6]. Нормализация данных направлена на придание всем атрибутам одинакового "веса".

#### **1.8.1.1 Основные методы нормализации данных**

а) Min-max нормализация заключается в применении к диапазону значений атрибута  $x$  линейного преобразования, которое отображает  $[\min(x), \max(x)]$  в  $[A, B]$ .

$$x'_i = \tau(x_i) = \frac{x_i - \min(x)}{\max(x) - \min(x)} * (B - A) + A \quad (1.11)$$

$$x \in [\min(x), \max(x)] \Rightarrow \tau() \Rightarrow [A, B] \quad (1.12)$$

Min-max нормализация сохраняет все зависимости и порядок оригинальных значений атрибута. Недостатком этого метода является то, что выбросы могут сжать основную массу значений к очень маленькому интервалу

б) Z-нормализация основывается на приведении распределения исходного атрибута  $x$  к центрированному распределению со стандартным отклонением, равным 1 [6] .

$$x'_i = \tau(x_i) = \frac{x_i - \bar{x}}{\sigma_x} \quad (1.13)$$

$$M[x'] = 1 \quad (1.14)$$

$$D[x'] = 0 \quad (1.15)$$

Метод полезен когда в данных содержатся выбросы.

в) Масштабирование заключается в изменении длины вектора значений атрибута путем умножения на константу [6] .

$$x'_i = \tau(x_i) = \lambda * x_i \quad (1.16)$$

Длина вектора  $x$  уменьшается при  $|\lambda| < 1$  и увеличивается при  $|\lambda| > 1$

## 1.9 Неконтролируемые алгоритмы обнаружения аномалий

Существуют различные классы методов обнаружения неконтролируемых аномалий. Выбор соответствующего класса метода зависит от характера данных и наличия априорной информации, требовааний к скорости работы алгоритма, затрачиваемой памяти, размеру данных и многих других параметров. Так же можно заметить, что в рамках методы одного класса могут значительно отличаться по своим характеристикам, что дополнительно усложняет выбор. Рассмотрим основные классы методов, их преимущества и недостатки.

### 1.9.1 Вероятностно-генеративные методы

Основная идея генеративных методов заключается в использование вероятностного смесового моделирования данных. Предлагается подобрать такую вероятностную модель, из которой было получены нормированные данные. Такие модели обычно называются генеративными моделями, где для каждой точки(элемента данных) можем посчитать генеративную вероятность(или вероятность правдоподобия).Т.е. задача сводится к нахождению плотности распределения  $p(x)$ . Аномалиями при этом считаются точ-

ки(элементы набора данных), имеющую низкое правдоподобие. В качестве показателя аномальности выступает функция  $p$ . Для построения генеративной модели нужно решить следующую задачу:

$$\prod_{x \in X_{norm}} p(x, \theta) \rightarrow \max_{\theta} \quad (1.17)$$

где  $X_{norm}$  - нормальные данные представленного набора данных  $p(x, \theta) | \theta \in \omega$  - семейство плотностей вероятностей, параметризованные  $\theta$ .

Этот метод редко используется на практике, так как тяжело проверить полученную генеративную модель на адекватность, сложно убедиться в правильном выборе семейства смесевых распределений. Это связано с тем, что низкое значение функции правдоподобия может означать как и аномальное значение, так и неудачно подобранную модель. Этот метод применяется с опорой на априорную информацию, в случае когда можно проверить полученную модель на адекватность.

### 1.9.2 Линейные методы

Основной идеей линейных методов является построение некой модели, характеризующей нормальные данные. Точки, которые значительно отклоняются от этой модели, считаются аномалиями.

Предполагается, что нормальные данные находятся в подпространстве пространства атрибутов данных (размер подпространства атрибутов данных равен размерности данных). В свою очередь, задача линейного метода - найти низкоразмерные подпространства, такие что, выборка данных этого подпространства значительно отличается от остальных точек пространства данных.

Одним из возможных вариантов решения является использование линейной регрессии. Выбирается одна из наблюдаемых переменных набора данных и относительно неё решается задача линейной регрессии оставшихся атрибутов. Итоговым ответом будет является усредненное значения показателя аномалии по всем атрибутам.

Алгоритмы, основанные на линейном подходе, требуют наличия линейной зависимости атрибутов данных.

### 1.9.3 Метрические методы

Метрические методы пытаются найти в данных точки, в некотором смысле изолированные от остальных[3]. Если в пространстве задана некоторая метрика  $p(x1, x2)$ , то необходимо задать следующие понятия:

— Аномалии – точки, не попадающие ни в один кластер. К данным применяется один из алгоритмов кластеризации; размер кластера, в котором оказалась точка, объявляется её показателем аномальности.

— Локальная плотность в аномальных точках низкая. Для данной точки показателем аномальности объявляется локальная плотность, которая оценивается некоторым непараметрическим способом.

— Расстояние от данной точки до ближайших соседей велико.

В качестве показателя аномальности может выступать:

- расстояние до k-го ближайшего соседа;
- среднее расстояние до k ближайших соседей;
- медиана расстояний до k ближайших соседей;
- гармоническое среднее до k ближайших соседей;
- доля из k ближайших соседей, для которых данная точка является не более чем k-ым соседом и многое другое.

#### 1.9.3.1 Базовые понятия

Метрические методы используют в случае отсутствия априорной информации о данных. Сложность вычисления прямо пропорциональна как размерности данных  $m$ , так и их количеству  $n$ . При росте набора данных наблюдается экспоненциальный рост сложности вычислений. Однако, эти методы хорошо проявляют себя на ограниченных наборах данных[8]. Следовательно такие методы как k-ближайших соседей(так же известный как обучение на основе примеров, и описанный позднее) с нотацией ассимптотического роста  $O(n^2)$  недопустимы для наборов данных с большой размерностью, если их размерность не может быть уменьшена.

Существуют много различных вариации алгоритма k-ближайших соседей для обнаружения аномалий, но все они основаны на вычислении некой метрики "расстояния до соседей", такой как Евклидово расстояние или расстояние Махаланобиса. Евклидово расстояние задается следующей формулой:

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1.18)$$

и является просто расстоянием между двумя точками, когда как расстояние Махаланобиса, задаваемое следующей формулой

$$\sqrt{(x - \mu)^T C^{-1} (x - \mu)} \quad (1.19)$$

вычисляет расстояние от точки до центра тяжести ( $\mu$ ), определяемого формулой коррелированных атрибутов, заданных матрицей ковариации (C). Расстояние Махаланобиса рассчитывается значительно дольше по сравнению с евклидовым для больших объемов данных, поскольку оно требует обойти через весь набор данных, чтобы идентифицировать корреляции атрибутов.

### 1.9.3.2 К ближайших соседей(kNN)

Алгоритм работы метода:

- Выбирается число K - число соседей.
- Устанавливается граница показателя аномальности, на основе которой будет определяться метка элемента(задается в процентах относительно среднего показателя расстояния)
- На основе метрики, рассчитывающей расстояния между элементами, рассчитывается расстояние между всеми элементами и всеми его соседями.
- Полученный результат сортируется на
- На основе полученных расстояний и границы показателя аномальности элементам присваиваются метки.

В качестве метрики, рассчитывающей расстояния между элементами можно использовать следующую формулу:

$$L = \sum_{j=0}^k x_0 - x_j \quad (1.20)$$

где  $x_j$  - значение j-того атрибута элемента до которого ищется расстояние, а  $x_0$ -искомый элемент.

### 1.9.3.3 Оптимизация Рамасвани

Точка p является выбросом, если не более n - 1 других точек в наборе данных имеют более высокий  $D_m$ (расстояние до m соседей), где m задается. Например на рисунке 1.2 черная точка(обозначена звездочкой) является наиболее удаленной от соседей, следовательно она является выбросом. Красные точки(обозначены треугольниками) расположены рядом друг с другом, однако расстояние до других точек велико, следовательно они тоже являются аномалиями. Такой подход восприимчив к вычислительному росту, потому что должна быть вычислена матрица расстояний точек друг от друга, поэто-

му Рамасвани в 2000 году предложил оптимизацию метода k-ближайших соседей в виде составления ранжированного списка потенциальных выбросов.

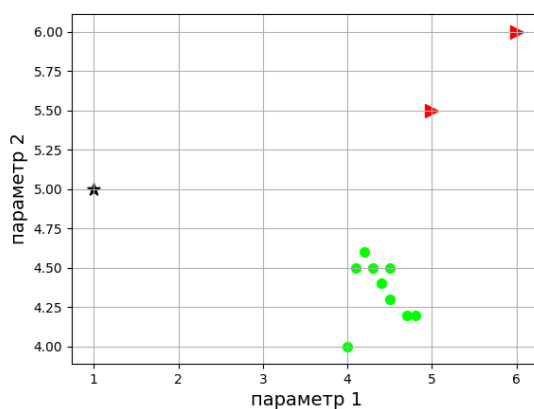


Рисунок 1.2 — Пример поиска аномалий для метода k-ближайших соседей в двумерном пространстве

Оптимизация Рамасвани заключается в разбиении данных на ячейки. Если какая-либо ячейка и ее ближайшие соседи содержат больше, чем  $k$  точек, то точки в ячейке считаются лежащими в плотной области, поэтому содержащиеся там точки вряд ли будут выбросами. Если же почти все ячейки содержат больше, чем  $k$  точек, а какие-то ячейки содержат меньше, чем  $k$  точек, то тогда все точки, лежащие в ячейках, которые содержат менее  $k$  элементов, помечаются аномальным. Следовательно, необходимо обработать только небольшое количество ячеек, которые ранее не были помечены, и только относительно небольшое количество расстояний необходимо вычислить для обнаружений аномалий.

#### 1.9.3.4 Методы Кнора-Реймонда и Байерса-Рейтери

Кнор и Реймонд предложили свой эффективный метод КНН подхода обучения без учителя [10]. Если  $m$  из  $k$  ближайших соседей (где  $m < k$ ) лежат в пределах определенного порогового значения  $d$ , тогда считается, что данные точки лежат в достаточно плотной области распределения данных, подлежащей классификации и подлежат классификации как нормальные, в противном случае они помечаются как аномальные.

Очень похожий метод был придуман для идентификации наземных мин на спутниковых снимках поверхности Земли Байерсом с соавторстве с Рейтери [?](этот метод можно использовать и для других целей). Он заключается в том, что берется  $m$  точек, для них ищется расстояние  $D_m$ . Если расстояние меньше некоего порогового значения  $d$ , тогда считается, что данные точки лежат в достаточно плотной области распределения данных, подлежа-



щей классификации и подлежат классификации как нормальные, в противном случае они помечаются как аномальные. Этот метод уменьшается количество варьируемых параметров, по сравнению с методом Кнора-Реймонда: остаются только параметры  $d$  и  $m$ , параметр  $k$  убирается. подход оригинальный подход  $k$ -NN, поскольку только  $k$  ближайших соседей должны быть вычислены для каждой точки, а не всей матрицы расстояния для всех точек

### 1.9.3.5 Метод Танга

Метод Танга заключается в вычислении средней цепочки расстояний между точкой  $p$  и  $k$  её соседями. Начальным расстояниям присваиваются более высокие веса, поэтому, если точка находится в разреженной области как черная точка на рисунке 1.2, то путь до ее ближайших соседей будет относительно далеким, а среднее расстояние цепочки будет высоким. Этот метод выгодно отличается от вышеописанных тем, что учитывает как плотность, так и изоляцию. Рассмотрим рисунок 1.3. Очевидно, что черные точки(обозначены звездочками) являются аномалиями, а скопление зеленых точек - множеством "нормальных" точек. Однако, алгоритмы  $k$ -NN классификации могут столкнуться с проблемой того, что расстояние от черных точек до зеленого кластера примерно равно, значит эти точки можно отнести к одной группе и при определенных значениях параметров алгоритма эти точки не будут считаться аномалиями. Метод Танга поможет избежать таких ошибок при обнаружении выбросов. Однако метод является вычислительно

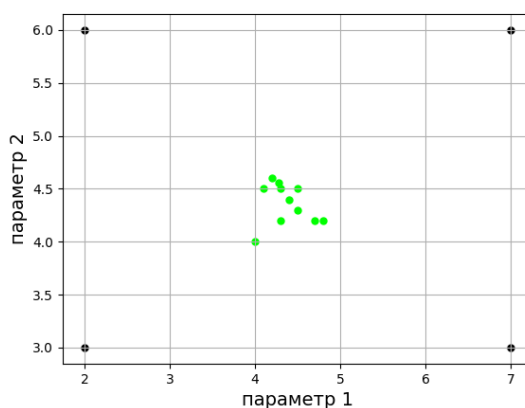


Рисунок 1.3 — Пример поиска аномалий методом Танга в двумерном пространстве

сложным с временем выполнения как у оригинального  $k$ -NN, поскольку он полагается на вычисление путей между всеми точками и их  $k$  соседей.

### 1.9.3.6 Метод ОДИН(ODIN)

Существуют модификации метода KNN, основанные на использовании графов. Рассмотрим один из них. Предположим, что у каждого элемента было найдено  $K$  ближайших соседей(используя евклидово расстояние). Тогда можно построить взвешенный ориентированный граф, где элемент представляет собой вершину, соединенную с  $K$  своими соседями ребрами[?]. Весом ребра будет евклидово расстояние между двумя точками. При построении такого графа можно столкнуться с проблемой известной как "проблема всех  $k$ -соседей". Проблема состоит в том, что построение такого графа может сопровождаться большой вычислительной сложностью. С решением этой проблемы можно ознакомиться в [?].

После построения графа начинается поиск аномалий. Аномалиями считаются все такие вершины, которые имеют 0 входящих ребер, а так же "замкнутые" группы элементов(такая группа элементов, которая имеет ребра только внутри себя, но не имеет связей с окружающими элементами).

### 1.9.3.7 Локальный коэффициент выбросов(LOF)

Этот метод является одним из самых известных алгоритмов обнаружения локальных аномалий. Недостатком метрических методов является тот факт, что все лежащие в их основе предположения верны лишь в дополнении друг с другом: локальная плотность точки, лежащей в центре небольшого кластера аномалий, может оказаться выше, чем для любой точки из большого кластера нормальных данных. Возможно и обратное: изолированная точка-аномалия может располагаться, например, в центре масс кластера нормальных точек, и тогда среднее расстояние от неё до соседей будет меньше, чем для нормальных точек. Это "свойство" метрических алгоритмов пытается учесть алгоритм локального коэффициентов выбросов(англ. Local Outlier Factor).

Чтобы вычислить LOF необходимо произвести следующие действия:

а) Для каждой записи найти всех соседей, расстояния до которых не превышает  $k$ . Их количество может быть больше, чем  $k$ .

б) Используя эти записи для каждой точки  $N_k$ , вычислить локальную плотность точки, основанную на локальной плотности достижимости(англ. local reachability density (LRD)):

$$LRD_k(x) = 1 / \left( \frac{\sum_{o \in N_k(x)} d_k(x, o)}{|N_k(x)|} \right) \quad (1.21)$$

где  $d(x, o)$  расстояние достигаемости. За редким исключением в качестве расстояния достигаемости используется евклидово расстояние [?]

в) Вычисляем LOF путем сравнения LRD записи с LRD соседей.

$$LOF(x) = \frac{\sum_{o \in N_k(x)} \frac{LRD_k(o)}{LRD_k(x)}}{|N_k(x)|} \quad (1.22)$$

Таким образом LOF является отношением локальных плотностей. Нормальные записи, плотности которых равны плотности их соседей, получают оценку около 1,0. Аномалии, которые имеют низкую локальную плотность, получают значительно более высокую оценку. Алгоритм полагаясь только на свою прямую окрестность, формирует оценку - величину, основанную только на k соседях. Конечно, глобальные аномалии также могут быть обнаружены, так как они имеют низкую LRD, по сравнению со своими соседями. Важно отметить, что в задачах обнаружения аномалий, где локальные аномалии не представляют интереса, этот алгоритм будет генерировать множество ложных аномалий. Настройка k имеет решающее значение для этого алгоритма.

Авторы алгоритма LOF рекомендуют использовать для вычисления k стратегию ансамблирования (алгоритм описан ниже). Берется интервал возможных значений k и с некоторым шагом для всех возможных значений k вычисляются показатели аномальности для каждого элемента выборки. Путем голосования определяется является ли эта точка аномалией. Однако, на практике такие рекомендации редко используют из-за их значительной вычислительной сложности.

### 1.9.3.8 Компонентный коэффициент выбросов(COF)

Компонентный коэффициент выбросов аналогичен LOF, но оценка плотности для записей выполняется иным способом. В LOF k-ближайших соседей выбирают на основе евклидова расстояния. Это косвенно предполагает, что данные распределяются сферическим образом вокруг экземпляра. Если это допущение нарушено, например, если функции имеют прямую линейную корреляцию, то оценка плотности неверна. COF исправляет этот недостаток и оценивает локальную плотность окрестности с использованием метода кратчайшего пути, называемого расстоянием цепочки. Расстояние цепочки находится при помощи алгоритмов поиска кратчайшего пути. Например, когда функции, очевидно, коррелированы, этот подход оценки плотности работает значительно лучше [?]. На рисунке 5 показан результат для LOF и COF в сравнении для простого двумерного набора данных, где атрибуты имеют линейную зависимость. Можно видеть, что оценка плотности LOF не может

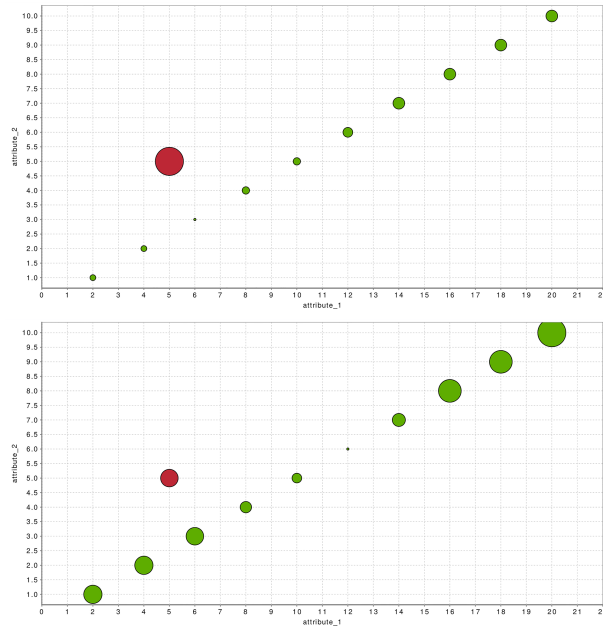


Рисунок 1.4 — Сравнение COF (сверху) с LOF (внизу) с использованием простого набора данных с линейной корреляцией двух атрибутов

обнаружить выброс, но COF удадается связать нормальные между собой для оценки локальной плотности.

### 1.9.3.9 Метод вероятностей локальных выбросов(LoOP)

Автора метода вероятностей локальных выбросов утверждает, что метод LOF может давать нестабильные результаты[?]. Поэтому была предложена модификация метода LOF.

Предположим, что  $D$  - множество из  $n$  объектов,  $d$ - функция расстояния, используемая для выделения выбросов. В целях улучшения стабильности результатов введен новую метрику - вероятностное расстояние  $o \in D$  до множества  $S \in D$ , обозначаемое  $pdist(o, S)$ . Вероятностное расстояние обладает следующим свойством:

$$\forall s \in S : P[(d(o, s) < pdist(o, S))] \geq \varphi \quad (1.23)$$

Если визуализировать это выражение, то можно представить, что сфера вокруг  $o$  с радиусом  $pdist$  покрывает любой элемент в множестве  $S$  с вероятностью  $\varphi$ . Вероятностное расстояние  $pdist(o, S)$  от  $o$  до  $S$  можно интерпретировать как статистическое распределение множества  $S$ . Вероятностное расстояние рассчитывается по следующей формуле:

$$PLOF_{\lambda, S}(o) = \frac{pdist(\lambda, o, S(o))}{E_{s \in S(0)}[pdist(\lambda, o, S(s))]} \quad (1.24)$$

Где  $E$ - центроида. Итоговая формула для расчёта LoOF выглядит следующим образом:

$$nPLOF = \lambda \sqrt{E[PLOF^2]} \quad (1.25)$$

$$LoOP_s(o) = \max\{0, \text{erf}(\frac{PLOF_{\lambda,s}(o)}{nPLOF * \sqrt{(2)}})\} \quad (1.26)$$

Где erf- Гауссова функция ошибок. С подробным выводом этих формул можно ознакомиться в [?].

### 1.9.4 Параметрические методы

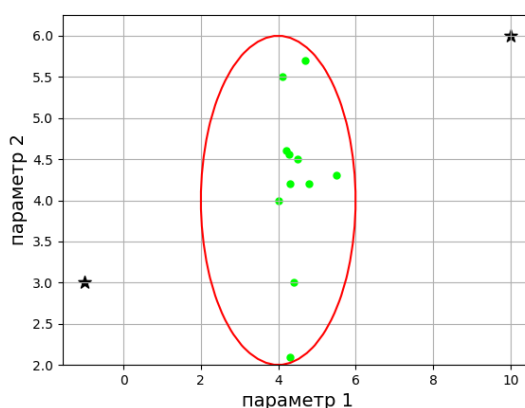


Рисунок 1.5 — Двухмерная проекция эллипсоиды минимального объема

Вышеописанные методы не подходят для работы с большим объемом данных из-за их высокой временной сложности. Параметрические методы позволяют очень быстро пересчитывать модель для новых данных и подходит для больших наборов данных; модель растет только с сложностью модели, а не размером данных. Однако они ограничивают применимость, применяя предварительно выбранную модель распределения для проверки данных на аномальность. Т.е. предварительно априорно подбирается модель правдоподобности данных. Элементы, которые значительно отклоняются от этой модели считаются аномальными. Параметрический подход схож с линейным по описанию, но значительно отличается от него по принципу работы.

Одним из таких подходов является оценка эллипсоидой минимального объема[?], которая соответствует наименьшему допустимому эллипсоиду, покрывающему не меньше 50% точек выборки.

### 1.10 Методы улучшения алгоритмов поиска аномалий

Алгоритмы поиска аномалий можно улучшать различными методами, применяемыми в том числе для улучшения результатов работы алгоритмов

и в других областях. Например, ансамблирование широко применяется для работы с нейронными сетями. Ниже рассмотрены приемы в контексте поиска аномалий.

### 1.10.1 Семплирование

Большинство алгоритмов распознавания аномалий успешно работают на наборах данных малых размеров. Поэтому предлагается разбить начальный набор данных на несколько случайных выборок и усреднить результат. Размер этих выборок может быть как и случайным, так и фиксированного размера, но, как правило, он отличается от размеров исходного набора данных не меньше чем на порядок. Идея такого выбора заключается в том, что шумовые объекты попадут в выборки с низкой вероятностью; кластера нормальных данных будут представлены несколькими представителями, а кластера аномалий вырождаются в изолированные точки. На основе этих выборок алгоритмы строят функции показателя аномальности, незначительно уступающему результату, полученному на основе анализа всех исходных данных.

Этот метод помогает значительно сократить вычислительную сложность, а так же уменьшить вероятность "подгона" алгоритма под конкретный набор данных. В силу особенностей задачи, необходимое условие - отсутствия параметризации алгоритмов - зачастую означает их детерминированность (в отсутствии стохастичности, показатель аномальности однозначно определяется по заданной выборке). В общем случае при добавлении новых данных в общий набор данных, можно не пересчитывать заново показатель аномальности для всего набора данных, а добавить запуски алгоритма на новых данных в ансамбль (так называемый warm start[?])

### 1.10.2 Ансамблирование голосованием

Ансамблированием в задаче поиска аномалий называют использование нескольких различных алгоритмов с последующим усреднением их показателя аномальности. При использовании различных классов алгоритмов можно столкнуться с проблемой того, что показатель аномальности выглядит по-разному в различных алгоритмах и сравнивать напрямую эти показатели некорректно. Поэтому традиционное приведение показателей значений различных функций к одному диапазону, например, к  $[0,1]$ , будет некорректным. Существует несколько наиболее известных видов ансамблирования:

— Простое голосование

$$b(x) = F(b_1(x), \dots, b_T(x)) = \frac{1}{T} \sum_{t=1}^T b_t(x) \quad (1.27)$$

,где  $b_i$  - некоторая функция.

— Взвешенное голосование

$$b(x) = F(b_1(x), \dots, b_T(x)) = \frac{1}{T} \sum_{t=1}^T w_t b_t(x) \quad (1.28)$$

$$\sum_{t=1}^T w_t = 1, w_t \geq 0 \quad (1.29)$$

,где  $w_i$ - некоторый коэффициент.

— Смесь экспертов

$$b(x) = F(b_1(x), \dots, b_T(x)) = \frac{1}{T} \sum_{t=1}^T w_t(x) b_t(x) \quad (1.30)$$

$$\sum_{t=1}^T w_t = 1, \forall x \in X \quad (1.31)$$

,где  $w_i(x)$ - некоторая функция коэффициента.

Простое голосование - это частный случай взвешенного голосования, а взвешенное голосование является частным случаем смеси экспертов.

Различные методы ансамблирования такие как беггинг, бустинг, стеккинг и другие применяются для улучшения работы алгоритмов обучения с учителем . Для алгоритмов обучения без учителя применяется простое голосование, т.к. задача изменения весов голосования нетривиальна в задаче обучения без учителя[?].

### 1.10.3 Итеративный отбор

Итеративный отбор основан на идее многократного применения алгоритмов ансамблирования. Преположим, построена некоторая модель, описывающая нормальные данные. Эта модель построена на основе всех имеющихся данных, но точность этой модели невелика, она умеет определять только явные аномалии. Отсортировав все точки по показателю аномальности, можно выбрать  $k$  самых аномальных объекта в данных и исключить из данных. После этого можно перестроить модель и повторить вышеуказанные действия несколько раз, пока не будут достигнуты некоторые условия. При каждой итерации точность модели будет увеличиваться.

Идея итеративного отбора может быть обобщена различными способами. Результат работы одного алгоритма может быть использован для отсеивания явных аномалий и настройки нового алгоритма, не обязательно совпадающего с предыдущим, на оставшихся данных. Возможна и противоположная механика: по результатам работы одного алгоритма отбираются яв-

ные, гарантированные представители нормальных данных, и исключительно на них строится модель, их описывающая.

## 1.11 Выводы

Таблица 1.1 — Сравнение алгоритмов поиска аномалий

| Класс методов     | Временная сложность | Расход памяти | Универсальность |
|-------------------|---------------------|---------------|-----------------|
| Вероятностно-ген. | $O(1)$              | $O(n)$        | Очень низкая    |
| Линейный          | $\geq O(n^2)$       | $\geq O(n^2)$ | Низкая          |
| Параметрический   | $O(1)$              | $O(n)$        | Низкая          |
| Метрический       | $\geq O(n \log n)$  | $\geq O(n)$   | Высокая         |

Под универсальностью понимается возможность применять алгоритмы к различным набором данных, не обладающих специфическими характеристиками, и, не обладая априорной информацией, получать высокую точность классификации.

Существует большое число алгоритмов для нахождения аномалий. Некоторые из них опирается на априорные данные, некоторые не опираются. Для выбора подходящего алгоритма нахождения аномалий зачастую стоит учитывать характер данных, их размер и доступную априорную информацию. Несмотря на то, область знаний обнаружения аномалий активно развивается как часть современной науки, остается ещё много простора для исследования алгоритмов, модификации и создания новых.



## 2 Конструкторский раздел

В этом разделе приводится подробное описание разрабатываемого метода, выделяются основные его компоненты, описываются метрики, оценивающие метод. Так же приводится описание ПО, собирающее данные для анализа. В выводе аналитической части предлагается разработать новый метод обнаружения аномалий. Новый метод будет являться результатом ансамблирования простым голосованием трех метрических методов.

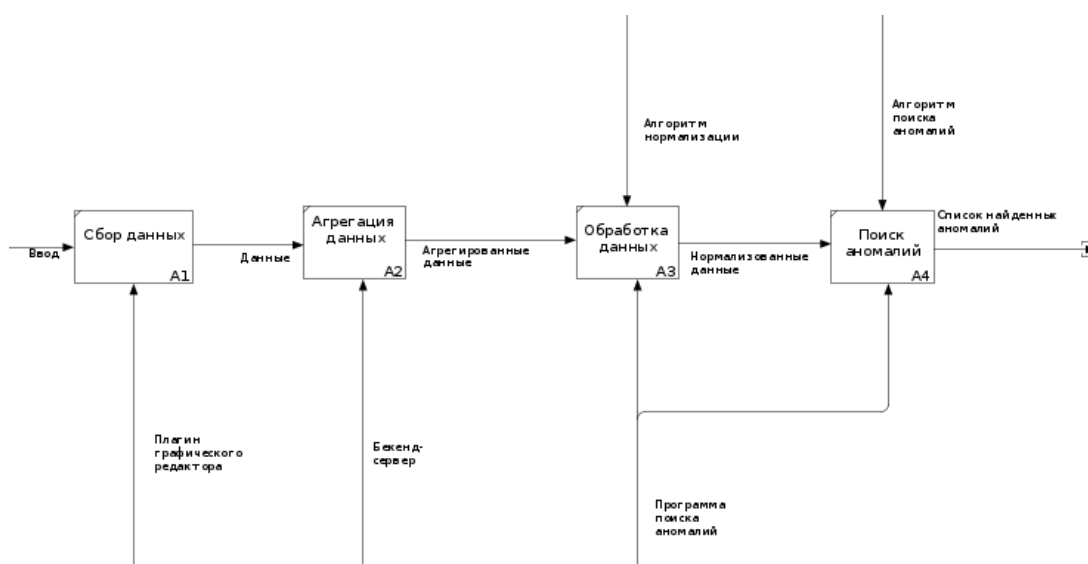


Рисунок 2.1 — Общая схема программного комплекса

### 2.1 Метод обнаружения аномалий

На вход методу подается файл с нормализованными значениями атрибутов, с фиксированным количеством атрибутов для каждого элемента. Временная отметка элемента представляется в качестве отдельного нормализованного атрибута. Допустимо отсутствие временной отметки. Были выбраны методы: k-ближайших соседей, локальный коэффициент выбросов. Каждый из вышеописанных методов инвариантен и иммутабелен относительно набора данных. В результате их работы получается три набора меток. На основе этих наборов формируется финальный набор меток по следующему принципу: если элемент имеет две или более "аномальных" метки, то ему присваивается "аномальная" метка, иначе - "нормальная" метка.

### 2.2 Подбор параметров алгоритмов

#### 2.2.1 Анализ результатов работы методов

Для проверки работоспособности метода его нужно оценить при помощи наборов данных. Метрикой сравнения наборов данных был выбран AUC ROC

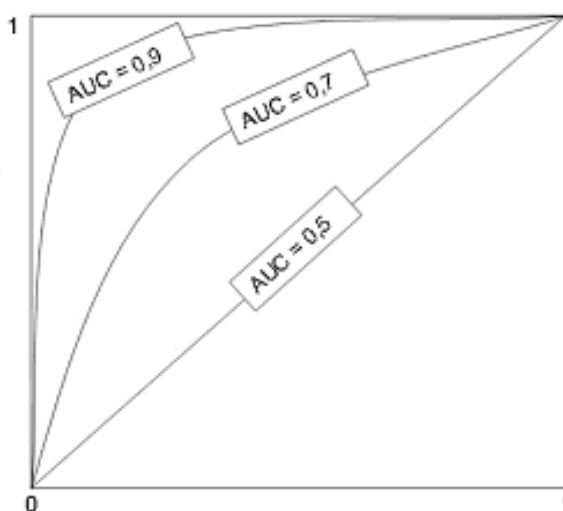


Рисунок 2.2 — AUC ROC

— площадь под графиком, позволяющая оценить качество бинарной классификации, отображающим соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущих признак, и долей объектов от общего количества объектов, не несущих признака, ошибочно классифицированных как несущих признак.

При помощи этой метрики планируется оценить адекватность работы алгоритма на размеченных наборах данных и сравнить с другими методами.

### 2.2.2 Подбор параметров

Алгоритмы поиска аномалий содержат некоторое различное количество параметров, которые необходимо указывать при работе алгоритмов. Однако, в отсутствии априорной информации о данных, их необходимо подбирать, основываясь на валидирующем наборе данных. [?] Для этого будет использован простой генетический алгоритм. Архитектура генетических алгоритмов позволяет найти решение быстрее за счет более "осмысленного" перебора. Будет осуществляться перебор не всего подряд, а приближаться от случайно выбранных решений к лучшим. В качестве функции, для которой ищется экстремум, используется алгоритм поиска аномалий, в качестве выходного значения функции - значение AUC ROC, полученное в результате работы алгоритма поиска аномалий на размеченном наборе данных. Критерием останова поиска оптимальных значений параметров алгоритма служит схождение популяции. Параметры, соответствующие наибольшему значению AUC ROC, сохраняются для использования в дальнейшем.

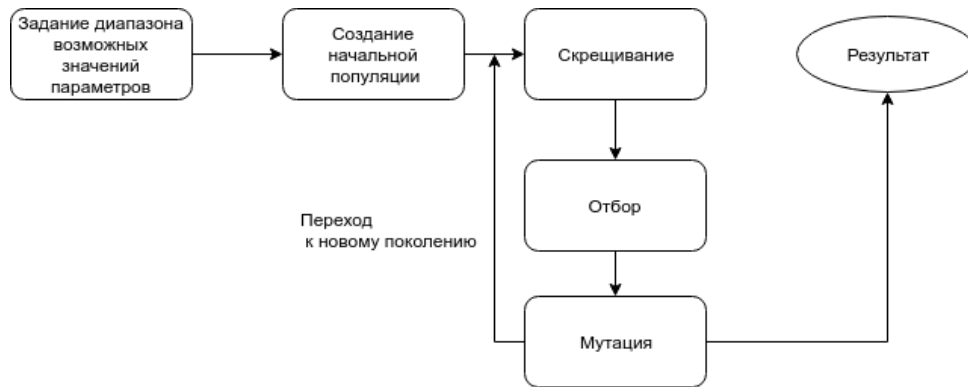


Рисунок 2.3 — Общий рисунок функционирования генетического алгоритма

## 2.3 К ближайших соседей

Метод основан на поиске аномальных значений расстояний до  $K$  ближайших соседей точки. Параметры алгоритма такие как  $K$  задаются после их нахождения вышеописанным методом.

Листинг 2.1 — Алгоритм метода  $K$  ближайших соседей

```

1  double calculateDist(vector<double> values , int orElen , int rInd ,
2    int lInt , int counter)
3  {
4    while (counter) {
5      int leftDistance = -1, rightDistance = -1;
6      if (rightIndex > -1)
7        rightDistance = abs(values[rightIndex]);
8      if (leftIndex > -1)
9        leftDistance = abs(values[leftIndex]);
10     if (leftDistance > rightDistance) {
11       rightIndex++;
12       distance += abs(originalElement - rightDistance);
13     } else {
14       leftIndex++;
15       distance += abs(originalElement - leftDistance);
16     }
17     counter--;
18   }
19 double kNearestDistance(vector<double> values , double value , int k)
20 {
21   int index = findIndex(values , value);
22   double originalElement = values[index];
23   double distance = 0;
24   int rightIndex = index + 1, leftIndex = index - 1;
25   int counter = k;
26
27   return distance;
  
```

```

28 }
29
30 for (int i = 0; i < values.size(); i++) {
31     for (int j = 0; j < values[i].size(); j++) {
32         values[i][j].distance = kNearestDistance(sortedValues[j],
33             values[i][j].value, dimensionSize, j);
34     }
35 }

```

Псевдокод содержит часть алгоритма, которая позволяет определять дистанции между точками. Этот алгоритм отличается от классического алгоритма К ближайших соседей, тем что расстояние до К ближайших соседей находится не с помощью Евклидова расстояния, а при помощи разбиения исходного двумерного массива атрибутов на одномерные массивы атрибутов согласно классам атрибутов.

Алгоритм нахождения расстояние для одного элемента:

- а) Для каждого атрибута находится k ближайших точек этого же класса
- б) Расстояния между искомым атрибутом и k ближайшими соседями суммируются
- в) Рассчитанная для каждого атрибута сумма складывается в итоговое расстояние

Временная сложность данного алгоритма  $O(n^2)$ , где n- суммарное количество атрибутов всех элементов данных. Сложность по памяти  $O(n)$ .

## 2.4 Локальный коэффициент выбросов

Листинг 2.2 — Алгоритм метода локального коэффициент выбросов

```

1  auto calcLRD(vector<Point> values, Point value, int k, Mode mode)
2  {
3      values.delete(value);
4      vector<Point> nearPoints;
5
6      for(i = 0; i < values.size(); ++i){
7          if(distance(values[i], value) > k)
8              nearPoints.pushback(value)
9      }
10     for(i = 0; i < nearPoints.size(); ++i){
11         sum += nearPoints[i];
12     }
13     if(mode != sum)
14         return nearPoints;
15     else
16         return nearPoints.size()/sum;

```

```

17 }
18
19 double LOF(vector<Points> values , Point value ,int k)
20 {
21     vector<Point> nearPoints = calcLRD(values , value , k, val);
22     int N = nearPoints.size()
23     double lrdSum = 0;
24     for(int i=0; i < nearPoints.size(); ++i){
25         lrdSum += calcLRD(values , nearPoints[i],k), sum);
26     }
27
28     double result = lrdSum/calcLrds(values , value ,k, val);
29     return result;
30 }
31 vector<double> distances;
32 for (int i = 0; i < values.size(); i++) {
33     distances.pushback(LOF(values , value ,k)
34 }

```

Псевдокод содержит основные сущности алгоритма : нахождение LOF и LRD. Временная сложность данного алгоритма  $O(n^2)$ , где n- суммарное количество атрибутов всех элементов данных. Сложность по памяти  $O(n)$ .

## 2.5 Компонентный коэффициент выбросов

Листинг 2.3 — Алгоритм Флойда — Уоршелла

```

1 for k = 1 to n
2     for i = 1 to n
3         for j = 1 to n
4             W[i][j] = min(W[i][j], W[i][k] + W[k][j])

```

Алгоритм COF почти полностью повторяет алгоритм LOF, но отличается способом вычисления расстояния между двумя точками. Расстояние между двумя точками вычисляется при помощи метода "кратчайшей цепочки". Для всех элементов набора данных строится матрица расстояний на основе евклидова расстояния. После этого при помощи алгоритма Флойда-Уоршела между двумя точками ищется кратчайший путь. Длина этого кратчайшего пути и будет мерой расстояния между двумя точками.

## 2.6 Собирающее данные для анализа ПО

Для применения метода на практике было разработано ПО, которое позволяет собирать данные для анализа(телеметрию). В состав приложения будет входить плагин для графического редактора, backend-сервер. Так же

на бекенде будет размещена база данных где будут размещаться необработанные данные.

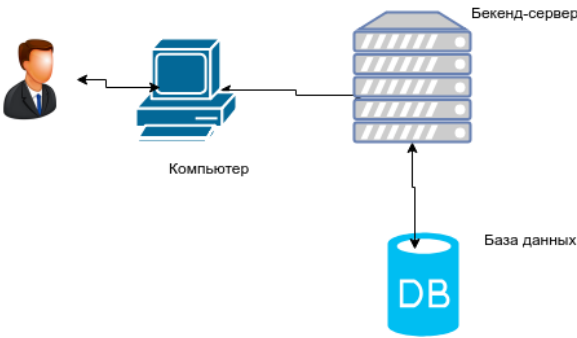


Рисунок 2.4 — Общая архитектура приложения

Результатом работы ПО будет неразмеченный набор данных, содержащий фиксированное количество атрибутов.

**2.6.1 Собираемые данные**

Основные собираемые данные приведены в таблице 2.1:

Таблица 2.1 — Описание собираемых данных

| Информация о         | Источник данных                       | Данные  |
|----------------------|---------------------------------------|---|
| Инструменты          | KisTool::activate, KisTool:deactivate | CountUse float64<br>Time float64<br>ToolName string<br>Timestamp float64  |
| Действия             | KisMainWindows actioncollection()     | CountUse float64<br>TimeUse float64<br>ActionName string<br>Timestamp float64   |
| Свойства изображений | KisDocument::saveFile()               | ColorProfile string<br>ColorSpace string<br>Height float64<br>Width float64<br>Size float64<br>NumLayers float64<br>Timestamp float64 |

**2.6.2 Клиентская часть**

#### Листинг 2.4 — Тело http-запроса

```
1 {  
2   "Name": [  
3     {  
4       "Param1": 1,  
5       "Param2": 4581,  
6       "Timestamp": 12214  
7     }],  
}
```

Был разработан плагин для графического редактора Krita, который позволяет собирать телеметрию с пользователей. Телеметрия собирается только во время работы программы и при закрытия программы собранные данные удаляются с компьютера пользователя. Телеметрия отправляется через определенные интервалы времени на удаленный сервер. Записи о действиях и инструментах отправляются каждые n минут. Информация о свойствах изображения собирается при сохранении изображения.

Собираемые метрики агрегуются на стороне клиента в http-запрос. Пример тела запроса представляет собою JSON, пример этого JSON приведен выше(форматирование нарушено в целях наглядности).

### 2.6.3 Серверная часть

На сервере метрики обрабатываются и заносятся в базу данных в формате JSON. Сервер способен принимать и обрабатывать много параллельных запросов пользователей. Наличие временной отметки у элементов данных позволяет извлекать данные за определенный промежуток времени.

#### Листинг 2.5 — Пример хранимого элемента в базе данных

```
1 {  "\_id" : ObjectId("5b08a54677d37ee1964df0b7"), "actions" : [ {  
    "countuse" : 1, "sources" : 0, "actionname" : "edit\_cut",  
    "time" : 1527293254  }  
}
```

### 2.6.4 Динамическое изменение данных

Инструменты и действия(actions) могут меняться достаточно часто в процессе разработки графического редактора. Поэтому неразумно задавать статически эти метрики в коде. В коде бекенд-сервера реализована поддержка добавления новых элементов. Раз в сутки просыпается новая горутина(легковесный тред), которая пробегается по базе данных и ищет новые инструменты и действия. После этого она записывает их в текстовый файл. Подобное разумно применять не только для инструментов и действий, но и для любых часто изменяющихся данных. В будущем возможно расширения этой системы.

### **3 Технологический раздел**

#### **3.1 Выбор средств разработки**

##### **3.1.1 Выбор целевой платформы**

Программный продукт поиска аномалий не заточивается под работу на конкретной ОС. Его корректная работа протестирована на ОС Windows и OS Linux. Данный выбор обусловлен широкой распространенностью ОС Windows и Linux, большим количеством средств разработки для данных платформ, что дает возможность выбирать язык программирования и сопутствующие инструменты, ориентируясь на возможность простого и надежного решения рассматриваемой задачи, а не на ограничения целевой платформы.

Программный комплекс для сбора данных разрабатывается для OS Linux, в силу особенностей функционирования графического редактора, плагинов для которого входит в состав этого программного комплекса.

##### **3.1.2 Выбор языка программирования**

Для написания программы поиска аномалий используется язык C++, так как с версии C++11 он сочетает в себе возможности функционального и объектно-ориентированного подходов. С помощью функционального подхода удобно описывать алгоритмы и математические выражения, необходимые для решения поставленной задачи. Использование функциональных средств существенно упрощает описание операций над последовательностями входных данных. В то же время, применение объектно-ориентированного подхода дает возможность проектирования приложения в терминах объектов предметной области и их поведения, что позволяет контролировать сложность программы, предлагать наглядные и емкие абстракции и четко выделять сферы ответственности программных модулей. Также объектно-ориентированные языки позволяют построение приложения с графическим интерфейсом пользователя с учетом известных рекомендаций и правил. Дополнительным преимуществом языка C++ является большая база документации по самому языку и стандартным библиотекам.

В качестве языка написания плагина к графическому редактору был выбран язык C++ в силу того, что сам редактор поддерживает только плагины на этом языке. Для написания бекенд-сервера был выбран язык Golang вместе с библиотекой tgo для доступа к базе данных. Плюсы этого языка:

- а) Скорость разработки
- б) Производительность
- в) Удобная реализация легковесных потоков



Для предварительной обработки данных использовался язык `bash`. `Bash` - это мощный и простой в использовании язык сценариев. Он позволяет легко перемещать курсор и редактировать текст команды в командной строке, поддерживает историю команд: дает возможность повторить или при необходимости изменить команду, которая была введена в командной строке ранее. Позволяет легко указать команде, откуда брать входные данные и куда направлять выходные данные. Поддерживаются псевдонимы - создание кратких обозначения для однострочных команд.

### **3.1.3 Выбор среды разработки и отладки**

Из-за использования сразу нескольких языков, в процессе работы над программным комплексом, было использовано несколько сред разработки. `Qt Creator` - современная кросс-платформенная среда разработки, которая предоставляет широкие возможности разработки программ на `C++`, а так же их отладки. `Qt Creator` так же поддерживает работу со сторонними плагинами, что позволило использовать плагин работы с системой контроля версий.

Для работы с языком `Golang` был использован текстовый редактор `Visual Studio Code` с набором плагинов для языка `Golang`. Набор плагинов к текстовому редактору позволяет осуществлять запуск и отладку кода прямо из редактора, а так же позволяет осуществлять автоматическое форматирование кода.

Для работы с `bash` использовался `nano` - консольный текстовый редактор для `Unix` и `Unix`-подобных операционных систем, основанный на библиотеке `curses`. В настоящее время включен в дистрибутивы `Ubuntu` по умолчанию и в установке не нуждается.

## **3.2 Выбор базы данных**

Для сохранения элементов данных была выбрана документо-ориентированная база данных `MongoDb`. Основные преимущества `MongoDb`:

- Отсутствие четкой структуры хранения данных
- Возможность выгружать и загружать файлы в `JSON`-формате.
- Быстрая вставка элементов
- Быстрое извлечение простых данных

## **3.3 Система контроля версий**

В процессе разработки программы использовалась система контроля версий `Git`. Система контроля версий позволяет вносить в проект атомарные изменения, направленные на решения каких-либо задач. В случае обнаружения ошибок или изменения требований, внесенные изменения можно от-

менить. Кроме того, с помощью системы контроля версий решается вопрос резервного копирования. Особенности Git:

- Предоставляет широкие возможности для управления изменениями проекта и просмотра истории изменений
- Данная система контроля версий является децентрализованной, что позволяет иметь несколько независимых резервных копий проекта.
- Поддерживается хостингом репозитория GitHub и Gitlab.
- Поддерживается средой разработки Qt Creator и Visual Studio Code.

### 3.4 Библиотека Galgo

Для подбора параметров алгоритмов была использована библиотека GALGO версии 2. Она позволяет не вдаваясь в нюансы работы генетических алгоритмов, подобрать необходимые значения параметров алгоритмов. В качестве функции, необходимой для работы генетического алгоритма, использовался алгоритм поиска аномалий. В качестве выходного значения функции использовался AUC ROC.

### 3.5 Формат файлов

На вход программе поиска аномалий подается файл следующего формата: одна строка содержит в ненормализованном виде атрибуты данных одного элемента.

Листинг 3.1 — Пример входного файла программы поиска аномалий

```
1 9 10 23 4
2 1 3 2 1
3 1 4 9 12
4 6 7 4 44
```

Для проверки корректности алгоритма использовался иной формат входных файлов. Одна строка содержит в нормализованном виде n-1 атрибутов одного элемента, на n-ой позиции в строке расположена метка аномальности объекта('yes' или 'no'). Атрибуты элементов разделяются запятой.

Для проверки корректности алгоритма использовался иной формат входных файлов. Одна строка содержит в нормализованном виде n-1 атрибутов одного элемента, на n-ой позиции в строке расположена метка аномальности объекта('yes' или 'no'). Атрибуты элементов разделяются запятой.

Листинг 3.2 — Пример входного файла программы поиска аномалий для проверки корректности алгоритма

```
1 0.467651,0.321584,0.76888,0.24663,0.838799,0.099737,0.29834,1.0,'no'
2 0.496412,0.220491,0.77603,0.36598,0.91993,0.08914,0.279479,2.0,'no'
```

|   |   |
|---|---|
| 3 | 0.519133,0.404464,0.76012,0.33498,0.80122,0.09239,0.271238,3.0,'no' |
| 4 | 0.19965,0.547373,0.374284,0.362223,0.817017,0.0,0.177913,4.0,'yes'  |
| 5 | 0.847261,0.286361,0.0,0.217792,0.0,0.019135,1.0,5.0,'no'            |

Третий формат возможный формат файлов предназначен для демонстрационного режима работы программы. В начале строки располагается текстовое описание элемента(без пробелов), следующие два атрибута характеризуют элемент в двумерном пространстве.

Листинг 3.3 — Пример входного файла программы поиска аномалий демонстрационного режима

|   |              |
|---|--------------|
| 1 | edit 23 53   |
| 2 | undo 35 547  |
| 3 | redo 100 0   |
| 4 | editcut 46 3 |

### 3.6 Библиотека KUserFeedback

В качестве вспомогательной библиотеки для сбора статистики используется библиотека KUserFeedback компании KDAB. Эта библиотека включает в себя C++ Qt клиентскую часть, а так же сервер, написанный на PHP. Их сервер и значительная часть функций не требуется, будет использоваться только часть собирающую телеметрию. KUserFeedBack позволяет собрать библиотеку по частям и линковать к нашему приложению только необходимые модули. В программе используется модуль Core. Модуль Core содержит абстрактный класс источника данных, от которого можно наследовать различные источники данных.

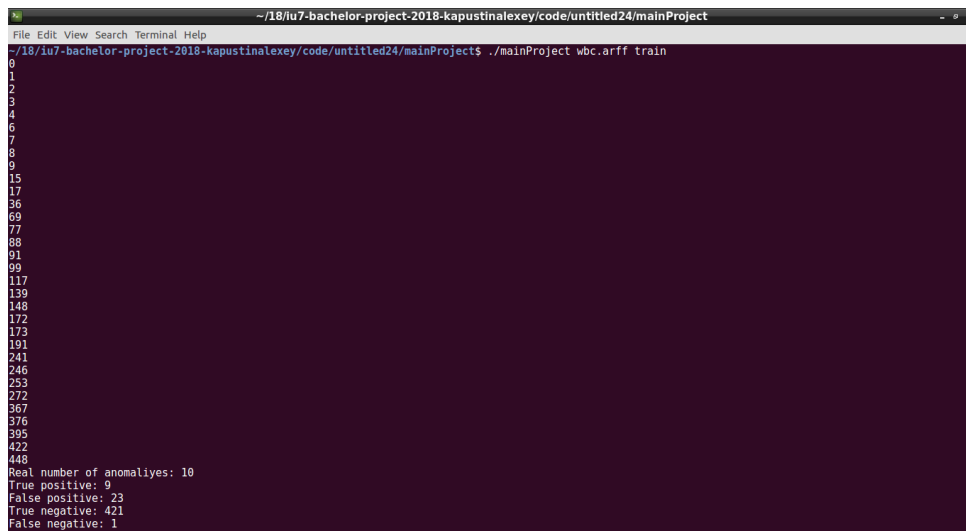
### 3.7 Установка программного обеспечения

У программы поиска аномалий нет зависимостей от внешних служб и программ, установленных в операционной системе, а также отсутствует необходимость в модификации системных и пользовательских настроек, поэтому было принято решение не разрабатывать инсталлятор. Исполняемым файлом программы является mainProject.exe(windows) или mainProject(linux).

Программа для сбора данных требует установленного графического редактора Krita с плагином телеметрии. Исполняемым файлом бекенд-сервера является файл server.

### 3.8 Пример работы программы

Приведен пример работы программы на операционной системе Ubuntu 16.04(Linux). Имя программы - mainProject, первый аргумент - об-



```
~/18/1u7-bachelor-project-2018-kapustinalexey/code/untitled24/mainProject
File Edit View Search Terminal Help
~/18/1u7-bachelor-project-2018-kapustinalexey/code/untitled24/mainProject$ ./mainProject wbc.arff train
0
1
2
3
4
6
7
8
9
15
17
36
69
77
88
91
99
117
139
148
172
173
191
241
246
253
272
367
376
395
422
448
Real number of anomalies: 10
True positive: 9
False positive: 23
True negative: 421
False negative: 1
```

Рисунок 3.1 — Пример работы программы для набора данных wbc

рабатываемый файл, второй аргумент - режим работы. В результате работы программы можно увидеть список найденных аномалий, а так же количество правильно или неверно классифицировавшихся элементов.

### 3.9 Использование программы для поиска аномалий

Запуск программы осуществляется из командной строки. Первым аргументом нужно передать имя файла для анализа. Вторым передается режим работы программы. Существуют 3 основных режима работы программы:

— Проверочный. Запуск этого режима осуществляется при помощи передачи в качестве второго параметра слова "train" . Этот режим обрабатывает размеченные наборы данных(данные должны быть предоставлены в описанном выше формате). В результате работы программы на экран выводится список полученных аномалий, метрики полноты, точности, площади под графиком РОК-кривой, количество истинно позитивных, истинно негативных, ложно негативных, ложно позитивных элементов данных.

— Основной. Для запуска этого режима нужно либо не указывать второй параметр, либо указать его некорректно(не "train" или "demo" ). Этот режим обрабатывает неразмеченные наборы данных(в описанном выше формате). В результате работы программы выводится список обнаруженных аномалий.

— Демонстрационный. Для запуска нужно передать второй аргумент равный "demo" . Предназначен наглядного представления результатов поиска аномалий в двумерном пространстве. Режим обрабатывает неразмеченные наборы данных.

## 4 Исследовательский раздел

Ошибки первого и второго рода - ключевые понятие математической статистики, помогающие проверять статистические гипотезы. Они так же используются в областях, где речь идет о принятии бинарного решения(да/нет). Задача поиска аномалий сводится к задаче бинарной классификации, где вопрос, помогающий осуществить классификацию, звучит следующим образом: "Является ли точка аномалией?".

Возьмем гипотезу  $H_0$ : "Точка является аномалией". Тогда можно составить следующую таблицу классификации ошибок:

Таблица 4.1 — Описание ошибок первого и второго рода

| $H_0$          | Верная              | Ложная              |
|----------------|---------------------|---------------------|
| Отклоняется    | Ошибка первого рода | Решение верное      |
| Не отклоняется | Решение верное      | Ошибка второго рода |

В машинном обучении используют другую терминологию для описания ошибок первого и второго рода. Это позволяет выделить 4 класса результата принятия решения(а не 3 как принято в математической статистике). На

Таблица 4.2 — Описание ошибок первого и второго рода в терминологии машинного обучения

| $H_0$          | Верная              | Ложная              |
|----------------|---------------------|---------------------|
| Отклоняется    | Ложное негативное   | Истинное негативное |
| Не отклоняется | Истинное позитивное | Ложное позитивное   |

основании значений этих метрик составляются метрики полноты, точности и графика под площадью РОК-кривой(описан выше)

### 4.1 Полнота и точность. Площадь под РОК-кривой

Точность(*precision*) - это доля объектов, названных классификатором положительными и при этом действительно являющимися положительными

$$precision = \frac{TP}{TP + FP} \quad (4.1)$$

TP- количество истинно позитивных, FP - количество ложно позитивных.

$$recall = \frac{TP}{TP + FN} \quad (4.2)$$

Полнота(recall) показывает сколько объектов положительного класса из всех объектов положительного класса нашел алгоритм. Мера точности не позволяет нам записывать все объекты в один класс, так как в этом случае мы получаем рост уровня ложно позитивных. Полнота демонстрирует способность алгоритма обнаруживать данный класс вообще, а точность— способность отличать этот класс от других классов. Полнота и точность могут применяться в условиях когда классы несбалансированы[?]. Поэтому они могут применять в задаче поиска аномалий, которая подразумевает несбалансированность классов. Так же существует метрика называемая F1-показатель. Она является средним гармоническим величин полноты и точности. Метрика площади под РОК-кривой описывается выше, здесь же приведем формулу её расчёта.

$$TPR = \frac{TP}{TP + FN} \quad (4.3)$$

$$FPR = \frac{FP}{TN + FP} \quad (4.4)$$

$$ROC = \frac{1 + TPR - FPR}{2} \quad (4.5)$$

## 4.2 Сравнение алгоритмов поиска аномалий

Для проверки работоспособности алгоритмов поиска аномалий на неразмеченных данных, эти алгоритмы проверялись на размеченных данных. Для этого работа алгоритма поиска аномалий была протестирована на двух наборах данных: Проведем сравнения с другими алгоритмами поиска анома-

Таблица 4.3 — Характеристики датасетов, метрики полноты и точности

| Набор данных | Кол-во элем. | Кол-во атриб. | Полнота | Точн. | Кол-во аном. |
|--------------|--------------|---------------|---------|-------|--------------|
| WBC          | 453          | 9             | 0.99    | 0.94  | 10           |
| KDDCUP99     | 60853        | 41            | 0.93    | 0.06  | 246          |

лий. Как можно увидеть из результатов метрик AUC ROC и F1, алгоритмы по-разному классифицируют разные наборы данных. Например, алгоритм LoOP показывает высокий AUC ROC на первом наборе данных, но на втором наборе данных его показали значительно снижаются. В свою очередь, алгоритм ODIN показывает низкие результаты, по сравнению с остальными алгоритмами, на первом наборе данных, но на втором наборе данных его AUC ROC высок. Разработанной алгоритм показывает средние значения AUC ROC, но высокие значения показателя F1, что позволяет утверждать, что этот алго-

Таблица 4.4 — Сравнение алгоритмов поиска аномалий

| Алгоритм       | AUC ROC WBC | F1 WBC | AUC ROC KDD | F1 KDD |
|----------------|-------------|--------|-------------|--------|
| LoOp           | 0.98        | 0.72   | 0.68        | 0.05   |
| ODIN           | 0.62        | 0.80   | 0.80        | 0.06   |
| KDEOS          | 0.25        | 0.64   | 0.61        | 0.05   |
| LDOF           | 0.64        | 0.96   | 0.88        | 0.07   |
| INFLO          | 0.99        | 0.9    | 0.98        | 0.29   |
| Разр. алгоритм | 0.92        | 0.97   | 0.93        | 0.06   |

ритм жизнеспособен и возможно его применение на определенных наборах данных.

### 4.3 Рекомендации к использованию метода

Таблица 4.5 — Количество истинно/ложно позитивно/негативно классифицировавшихся

| Набор данных | ИП  | ЛП   | ИН    | ЛН |
|--------------|-----|------|-------|----|
| WBC          | 10  | 18   | 425   | 0  |
| KDDCUP99     | 230 | 3603 | 57004 | 16 |

Исходя из количества истинно позитивных результатов и ложно позитивных результатов, можно рекомендовать использовать данный метод в задачах где акцент делается на нахождении истинно позитивных значений, пренебрегая некоторым количеством полученных ложно позитивных значений.

## **Заключение**

В данной работе были исследованы различные алгоритмы поиска аномалий, изучены их достоинства и недостатки. В частности были подробно изучены метрические методы поиска аномалий.

Был предложен и реализован новый метод поиска аномалий, основанный на модификации и ансамблировании уже существующих методов поиска аномалий. Даны рекомендации к использованию этого метода. Также был сделан программный комплекс сбора данных для анализа, состоящий из плагина для графического редактора и бекенд-сервера.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Справочник технического переводчика*. [Электронный ресурс]. — 2018. URL:[https://technical\\_translator\\_dictionary.academic.ru/8737/\T2A\cyra\T2A\cyrp\T2A\cyrr\T2A\cyri\T2A\cyro\T2A\cyrr\T2A\cyrn\T2A\cyra\T2A\cyrya](https://technical_translator_dictionary.academic.ru/8737/\T2A\cyra\T2A\cyrp\T2A\cyrr\T2A\cyri\T2A\cyro\T2A\cyrr\T2A\cyrn\T2A\cyra\T2A\cyrya), дата обращения: 10.03.2020.
2. *Международная теннисная федерация*. IFT[Электронный ресурс]. URL:<https://www.itftennis.com/>, дата обращения: 10.03.2020.
3. *Крутиков, Александр*. Прогнозирование спортивных результатов в индивидуальных видах спорта с помощью обобщенно-регрессионной нейронной сети / Александр Крутиков. — Молодой ученый. — 2018. — №12., 2018.
4. *J., Kahn*. Neural Network Prediction of NFL Football Games [Электронный ресурс] / Kahn J. — 2003. URL:<http://homepages.cae.wisc.edu/~ece539/project/f03/kahn.pdf>, дата обращения: 10.03.2020.
5. *Zdravevski E., Kulakov A*. System for Prediction of the Winner in a Sports Game / Kulakov A. Zdravevski E. — ICT Innovations 2009 — 2010., 2010.
6. *Enrick, J.R*. Optimal strategies at decision points in singles squash / J.R. Enrick. — Quart. Exercise Sport, 1976.
7. *Hsi B., Burych D*. Games of two players ДОПИСАТЬ КНИГИ НИЖЕ / Burych D. Hsi B. — Journal of the Royal Statistical Society 22(1), 1971.
8. *Crews, W.H. Carter u S.L*. An analysis of the game of tennis / W.H. Carter и S.L. Crews. — Journal of the American Statistical Association 28(4), 1974.
9. *Pollard, G.H*. An analysis of classical and tie-breaker tennis / G.H. Pollard. — Journal of the Australian Mathematical Society 25, 1983.
10. *J.R.Magnus, F. J. Lassen u.* Are points in tennis independent and identically distributed? Evidence from a dynamic binary panel data model / F. J. Lassen и J.R.Magnus. — Journal of the American Statistical Association 96(454), 2001.