

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 Методы попарного сравнения	14
1.5.3 Машинное обучение	15
1.5.3.1 Логистическая регрессия	16
1.5.3.2 Нейронные сети	17
1.5.3.3 Метод опорных векторов	19
1.5.3.4 Проблемы машинного обучения	19
1.5.3.5 Оптимизация гиперпараметров	20
1.5.3.6 Выбор оптимальных свойств	20
1.5.4 Предварительная обработка данных	22
1.5.4.1 One-Hot Encoding	22
1.5.4.2 word2vec	24
1.5.4.3 Улучшения word2vec	26
1.5.5 Нормализация данных	26
1.5.5.1 Основные методы нормализации данных . .	27
2 Конструкторский раздел	28
2.1 Метод обнаружения аномалий	28
2.2 Подбор параметров алгоритмов	28
2.2.1 Анализ результатов работы методов	28
2.2.2 Подбор параметров	29
2.3 К ближайших соседей	30
2.4 Локальный коэффициент выбросов	31
2.5 Компонентный коэффициент выбросов	32
2.6 Собирающее данные для анализа ПО	32
2.6.1 Собираемые данные	33
2.6.2 Клиентская часть	33
2.6.3 Серверная часть	34
2.6.4 Динамическое изменение данных	34
3 Технологический раздел	35

3.1	Выбор средств разработки	35
3.1.1	Выбор целевой платформы	35
3.1.2	Выбор языка программирования	35
3.1.3	Выбор среды разработки и отладки	36
3.2	Выбор базы данных	36
3.3	Система контроля версий	36
3.4	Библиотека Galgo	37
3.5	Формат файлов	37
3.6	Библиотека KUserFeedback	38
3.7	Установка программного обеспечения	38
3.8	Пример работы программы	38
3.9	Использование программы для поиска аномалий	39
4	Исследовательский раздел	40
4.1	Полнота и точность. Площадь под РОК-кривой	40
4.2	Сравнение алгоритмов поиска аномалий	41
4.3	Рекомендации к использованию метода	42
	Заключение	43
	Список использованных источников	44

Глоссарий

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

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

— Рейтинг АТР - официальная, еженедельно обновляемая система подсчета достижений спортсменов, используемая Ассоциацией теннисистов-профессионалов[3]

— Датасет - обработанный набор очищенных данных, пригодных для обработки алгоритмами машинного обучения[4]

— Embedding — это сопоставление произвольной сущности (например, узла в графе или кусочка картинки) некоторому вектору[5]

— Словарь текста - множество всех слов текста, где каждое слово, принадлежащая заданному тексту, упоминается ровно 1 раз.

Введение

Спортивные соревнования - это события на результат которых влияет огромное количество факторов. Результаты соревнований носят недетерминированный характер. Научиться прогнозировать исходы спортивных событий - первый шаг к прогнозированию исходов событий, влияющих напрямую на жизни людей (политика, сложные международные конфликты).

Результатами спортивных соревнований интересуются миллионы людей каждый день, от результатов команд зависит заработок спонсоров спортивных клубов, спортсменов, букмекеров и многих других. Теннис имеет простой набор правил, небольшое количество игроков, а так же всего два возможных исхода, что позволяет разрабатывать алгоритмы прогнозирования без большого погружения в специфику спорта. Добиться стопроцентной точности прогнозов невозможно априори, но можно её повышать путем улучшения существующих алгоритмов, их совмещения (например, путем ансамблирования). В данной работе предлагается новый метод, позволяющий прогнозировать результаты теннисных матчей.

1 Аналитический раздел

1.1 Цель и задачи работы

Целью данной работы является создание метода прогнозирования 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[6]. 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[7] 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 [8])

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 поражения.

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[9] и сквоше[10]. Впервые такой метод был применен к теннису в работах Кси и Бюрича[11], Картера и Крю[12], Полларда[13]. 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[14], 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_A^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.

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

К

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_B^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 [13], [14]).

$$p_A^M = \sum_{j=0,2,4}^2 (p_{AA}^M(3,j) + p_{AB}^M(3,j)) \quad (1.7)$$

Для матчей до 2 выигранных сетов(женский теннис)

$$p_A^M = \sum_{j=0}^1 (p_{AA}^M(2,j) + p_{AB}^M(2,j)) \quad (1.8)$$

1.5.2 Методы попарного сравнения

Основная идея методов попарного сравнения заключается в том, что каждого игроку присваивается некий положительный рейтинг, который характеризует уровень его навыков. Эти методы основаны на алгоритме, предложенном Бредли-Терри[15] и адаптированном для тенниса Ианом МакХолом[16]. Основная идея заключается в том, что вероятность победы игрока A над игроком B рассчитывается как

$$p_A = \frac{\alpha_A}{\alpha_A + \alpha_B} \quad (1.9)$$

Где α_A - рейтинг игрока A, а α_B - игрока B. Вариативность алгоритмов данного класса достигается за счёт различных методик подсчёта этого рейтинга. Например, при помощи ЭЛО-рейтинга[14]. Рассмотрим один из вариантов - подсчёт рейтинга, основанный на количестве выигранных геймов. Преимуществом этого метода является, тот факт, что берется в расчёт информация о ходе матча. Т.е. разница между матчам проигранным со счётом 6-0,6-0 и

со счётом 7-6, 7-6 существенна. Тогда вклад можно оценить по следующей формуле

$$p_A^M = L(\alpha_A, \alpha_B) \propto \frac{\alpha_A^{g_A} \alpha_B^{g_B}}{(\alpha_A + \alpha_B)^{g_A + g_B}} \quad (1.10)$$

где g_A -количество выигранных геймов игроком А, а g_B - игроком В(тай-брейк считается за обычный гейм). Приведем формулу для расчёта рейтинга игрока из одной из последних работ, посвященной данному классу методов

$$L(\alpha_a(t,); a = 1, \dots, n) = \prod_{k \in Z_t} \left(\frac{\alpha_a(t, S)^{g_A} + \alpha_B(t, S)^{g_B}}{(\alpha_A(t, S) + \alpha_B(t, S))^{g_A + g_B}} \right)^{\exp(\epsilon(t - t_k)) S_k} \quad (1.11)$$

Где t - время матча на поверхности теннисного корта S , k - индекс сыгранного матча, t_k - продолжительность матча k , $Z_t = k : t_k < t$, n - количество игроков в модели, ϵ - параметр зависящий от S .

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:

$$\sigma = \frac{1}{1 + e^{-t}} \quad (1.12)$$

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

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \quad (1.13)$$

Подставляя z в формулу 1.14 получим

$$p = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (1.14)$$

Обучение модели состоит в оптимизации параметров β , таким образом, чтобы модель "как можно точнее" воспроизводила результаты матчей из данных для обучения. Это делается при помощи минимизации функции логистических потерь 1.15, которая показывает меру ошибку прогнозирования результатов матчей тренировочных данных.

$$L(p) = \frac{-1}{N} \sum_{i=1}^N p_i \log(y_i) + (1 - p_i) + (1 - p_i) \log(1 - y_i) \quad (1.15)$$

Где N - количество матчей в обучающих данных, p_i - предсказанная вероятность выиграть матч i , y_i - фактический исход матча (0 - поражение, 1 - победа).

В зависимости от размера датасета, выбирается одна из двух функций минимизации:

— Стохастический градиентный спуск - медленный итеративный метод, подходящий для больших датасетов

— Метод максимального правдоподобия - быстрый численный метод, не подходящий для работы с большими датасетами.

Большинство публикаций, в которых используются методы машинного обучения для прогнозирования теннисных матчей, используют логистическую

регрессию. Например, Кларк и Дит[17] построили логистическую регрессионную модель, основанную на разнице набранных очков в рейтинге АТР для прогнозирования результатов розыгрыша сета. Другими словами, они использовали одномерную регрессию $x=(\text{разница_в_рейтинге})$ и оптимизировали β таким образом, чтобы функция $\sigma(\beta_1 x)$ показала наилучший результат на обучающем датасете. Параметр β_0 был исключён из модели на основании того, что разница в рейтинге 0 может давать вероятность выигрыша матча 0.5. Вместо того, чтобы прогнозировать результаты матча, они предпочли предсказывать результаты каждого конкретного сета, тем самым увеличивая набор данных для обучения.

Ма, Лу и Тан[18] использовали многомерную регрессию, использующую 16 свойств, поделённых на 3 категории: умения игрока, характеристики игрока и характеристики матча. Модель была обучена на матчах между 1991 и 2008 годом и была использована для улучшения тренировок игроков.

Логистическая регрессия привлекает исследователей тем, что она позволяет быстро обучать модели, почти не влечёт проблем переобучения моделей, а так же позволяет получить легко интерпретируемый выходной параметр - вероятность выиграть матч. Тем не менее, при наличии сложных взаимосвязей между входными параметрами, использование логистической регрессии может стать нетривиальной задачей.

1.5.3.2 Нейронные сети

Искусственная нейронная сеть - это система взаимосвязанных "нейронов" вдохновлённая биологическими нейронами. Каждый нейрон преобразует входные данные в какое-то выходное значение, которое в дальнейшем могут быть использованы в качестве входных данных для других нейронов. Нейронная сеть обычно имеет несколько слоёв, с нейроном на каждом не-входном слое, соединённом с нейронами в предыдущих слоях. Каждая связь в сети имеет свой вес. Нейрон использует входные значения и веса чтобы посчитать выходное значение. Типичным методом композиции является взвешенная сумма.

$$f(x) = K\left(\sum_i w_i x_i\right) \quad (1.16)$$

Нелинейная функция активации K позволяет нейронной сети решать нетривиальные задачи, используя небольшое количество нейронов. Логистическая функция 1.12 может использоваться в качестве функции активации.

Прогноз результата матча осуществляется на основе передачи в нейронную сеть неких свойств игрока и свойств матча. Если используется логистическая функция активации, то выходное значение нейронной сети

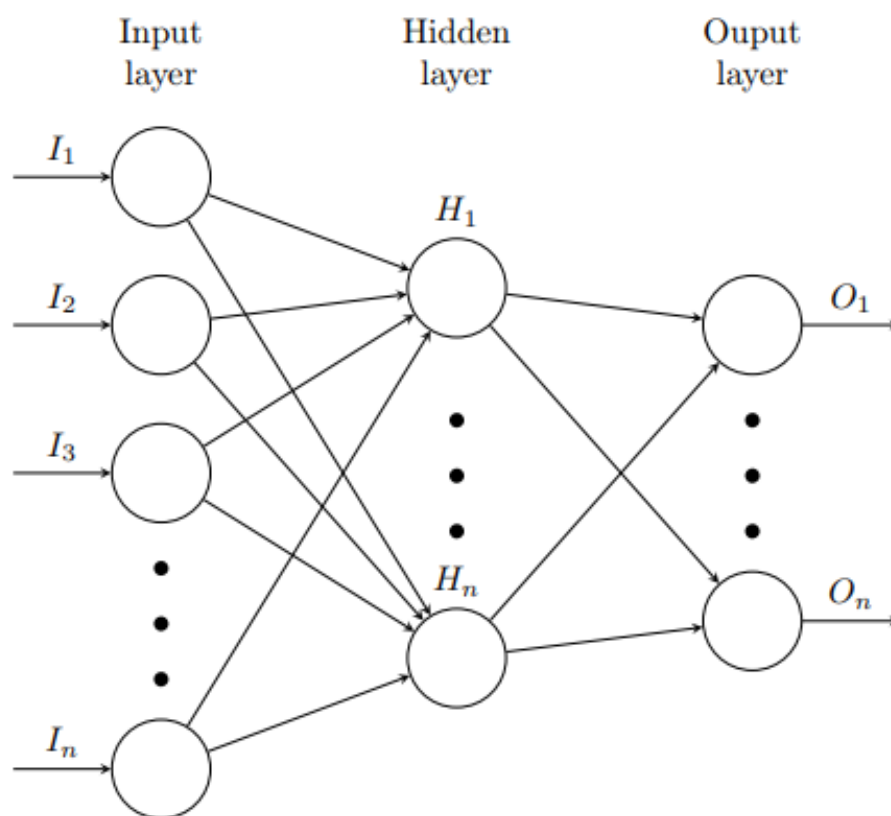


Рисунок 1.1 — Трехслойная нейронная сеть с прямыми связями(ПЕРЕВЕСТИ КАРТИНКУ)

можно интерпретировать как вероятность выиграть матч. Есть большое количество различных алгоритмов обучения, которые оптимизируют набор весов нейронной сети в целях получения наилучшего выходного значения. Например, алгоритм обратного распространения использует градиентный спуск для уменьшения среднего квадрата ошибки между целевыми значениями и выходами сети.

Например, Сомбоонфокафан[19] сконструировал трёхслойную сеть прямой связи, используя алгоритм обратного распространения ошибки. Протестировал на различных архитектурах сети и различных наборах свойств. Нейронная сеть, показавшая наилучший результат, имеет 27 входных нейронов. На вход подавались свойства, описывающие обоих игроков, а так же свойства матча. В результате была получена точность предсказания 75% на исторических данных турниров "Большого шлема" в 2007-2008 годах.

Нейронная сеть можно установить наличие сложных взаимоотношений между различными свойствами, подаваемыми на вход. Но она работает по принципу "черного ящика" т.е. механизм определения таких взаимоотношений не будет известен. К недостаткам нейронных сетей так же можно отнести то, что они склонны к переобучению, зачастую требует большого количества данных для обучения. Кроме того, разработка структуры сети зачастую яв-

ляется эмпирической и выбор гиперпараметров модели часто идёт методом проб и ошибок.

1.5.3.3 Метод опорных векторов

Основная идея метода опорных векторов заключается перевод исходных наборов свойств в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора. На данный момент в открытом доступе нет работ по прогнозированию теннисных матчей при помощи метода опорных векторов[20]. Метод опорных векторов имеет несколько преимуществ по сравнению с нейронными сетями. Во-первых, обучение не заканчивается на локальном минимуме, что зачастую бывает с нейронными сетями. Во-вторых, при наличии большого количества данных для обучения, метод опорных векторов обычно превосходит нейронные сети в прогнозировании [20]. Тем не менее, время обучения модели метода опорных векторов намного выше, а создание и задание необходимых параметров модели является нетривиальной задачей.

1.5.3.4 Проблемы машинного обучения

Наличие большого количества данных для обучения не означает получение более точных результатов прогнозирования результатов теннисных матчей, т.к. большое количество данных означает, что теннисист играл много лет матчи. Со временем его навыки меняются и результаты последних матчей представляют наибольший интерес. Так же в теннисе важную роль играет покрытие на которой играет матч, т.е. для модели необходимы данные о последних матчах игрока именно на этом покрытии. Поэтому машинное обучение может страдать от недостатка данных, что может приводить к переобучению модели, с которым может быть сложно бороться. В частности нейронные сети сильно подвержены переобучению если количество слоёв/нейронов сравнительно велико относительно размера данных для обучения.

Чтобы бороться с проблемой переобучения, нужно использовать только важные свойства для обучения. Процесс отбора важных свойств набора данных называется "отбор признаков" для этого существует большое количество различных алгоритмов. Удаление незначимых свойств значительно уменьшает время обучения

1.5.3.5 Оптимизация гиперпараметров

Модель имеет параметры как получаемые в процессе обучения путём их оптимизации(например, веса в нейронных сетях), так и статически задаваемые параметры такие как количество скрытых слоёв, количество нейронов в каждом слое. Статически настраиваемые параметры называют гиперпараметрами. Процесс получения оптимальных гиперпараметров может быть как эмпирическим, так и алгоритмическим. Пример алгоритмического поиска гиперпараметров - поиск по сетке. Он происходит на заранее определённом пространстве гиперпараметров. Для построения модели прогнозирования с высокой точностью, необходимо тщательно подобрать необходимые гиперпараметры.

1.5.3.6 Выбор оптимальных свойств

Каждый набор данных может содержать большое количество различных свойств. Каждое свойство вносит определенный вклад в итоговый результат. Выбор оптимального количества наиболее значимых свойств - приём очень часто применяемый в машинном обучении. От количества свойств зависит производительность получаемой модели. Свойства, не влияющие на итоговый результат, или влияющие незначительно, негативно влияют на производительность модели, при включении их в неё. Так же нерелевантные свойства могут ухудшить точность модели, заставить её обучаться на нерелевантных данных[21].

Преимущества выбора оптимального набора свойств

- Уменьшает вероятность переобучения. Использование меньшего количества избыточных данных уменьшает вероятность принятия решения на основе "шума" .
- Повышение точности.
- Сокращает время обучения- меньшее количество данных снижают сложность модели.

Выбор свойств может производиться как в ручном, так и автоматическом режиме.

Существуют 3 основных класса алгоритмов выбора свойств.

- Фильтр. Рассчитывается определенная метрика и свойства "фильтруются"на основании этой метрики
- Обертка. "Оберточные" методы рассматривают выбор оптимального набора свойств

Корреляция Пирсона - фильтр-метод, который использует в качестве метрики следующую формулу:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}} \quad (1.17)$$

Хи-квадрат метод - фильтр-метод, который использует в качестве метрики следующую формулу:

$$\chi^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i} \quad (1.18)$$

Где O_i - фактическое количество наблюдений класса i , а E_i - ожидаемое количество наблюдений класса i , при условии отсутствия зависимости между свойством и ожидаемым значением. Рассмотрим пример подсчёта метрики Хи-квадрат. Предположим, у нас есть следующие данные

Таблица 1.1 — Фактическое количество наблюдений результатов игроков

	Кол-во побед	Кол-во поражений	Всего
Игрок1	20	5	25
Игрок2	40	35	75
Всего	60	40	100

Для того чтобы посчитать значения χ^2 , посчитаем "ожидаемое" значение в каждой ячейки, если бы свойства были действительно независимы. Для этого нужно просуммировать значения каждой строки и поделить на общее количество записей. Получим следующую таблицу;

Таблица 1.2 — Ожидаемое количество наблюдений результатов игроков

	Кол-во побед	Кол-во поражений	Всего
Игрок1	15	10	25
Игрок2	45	25	75
Всего	60	40	100

Лассо-метод(сокращение от англ. **least absolute shrinkage and selection operator**) - встроенный метод. На основе линейной регрессии происходит пошаговый выбор оптимальных параметров модели. Идея метода заключается в том, что накладывается ограничение на сумму абсолютных значений свойств модели, сумма должна быть меньше некоторого фиксированного значения. Для этого того, чтобы этого добиться, применяется методы

сжатия(регуляризации), в процессе которого коэффициенты регрессионных переменных уменьшаются. Свойства, получившие в результате регуляризации нулевой коэффициент, отбрасываются.

Методы на основе деревьев относятся к классу встроенных методов. Например, для определения оптимального набора свойств может использоваться алгоритм случайного леса. Принцип работы встроенных методов нетривиален и выходит за рамки данной работы, подробнее с их принципами работы можно ознакомиться в [22]

1.5.4 Предварительная обработка данных

Перед тем как начать работать с данными, их нужно предварительно обработать. Данные могут храниться в числовом, текстовом виде, в формате изображений, аудио/видео файлов и других многочисленных форматах. Алгоритмы прогнозирования работают исключительно с числовыми данными. Поэтому данные в других форматах нужно перевести в числовой формат. В данной работе будет рассмотрено как перевести текст. В данной работе мы будем рассматривать только данные, которые хранятся в текстовом виде. Перед тем как данные переводятся в числовой вид, они предварительно очищаются. Для этого нужно

- Удалить все нерелевантные символы (например, любые символы, не относящиеся к цифро-буквенным).
- Токенизировать текст, разделив его на индивидуальные слова.
- Удалить нерелевантные слова — например, гиперссылки.
- Перевести все символы в нижний регистр для того, чтобы слова «привет», «Привет» и «ПРИВЕТ» считались одним и тем же словом.
- Рассмотрите возможность совмещения слов, написанных с ошибками, или имеющих альтернативное написание (например, «круто»/«круть»/ «крууто»)
- Рассмотрите возможность проведения лемматизации, т. е. сведения различных форм одного слова к словарной форме (например, «машина» вместо «машиной», «на машине», «машинах» и пр.)
- Рассмотреть возможность удаления слов, не несущих самостоятельной смысловой нагрузки. Например, артиклей в английском языке.
- Рассмотреть удаление слов не существенных для решаемой задачи на основании каких-либо параметров.

1.5.4.1 One-Hot Encoding

One hot encoding - это процесс в помощью которого некоторые категориальные переменные(в нашем случае - слова), преобразуются в форму,

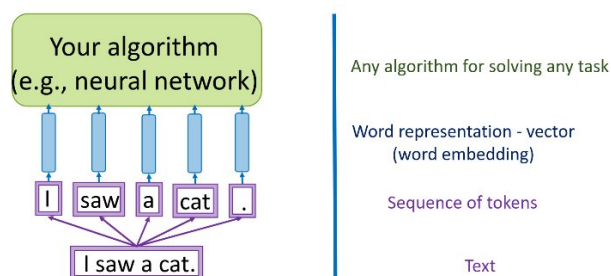


Рисунок 1.2 — Схема перевода текста в слова

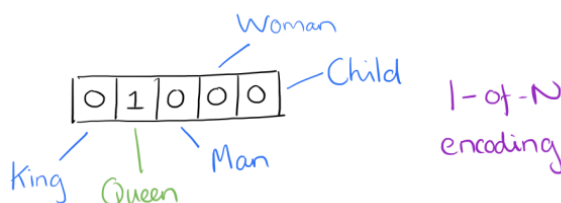


Рисунок 1.3 — Представляем предложение "The queen entered the room" в виде вектора

понятную алгоритмам машинного обучения. Берется достаточно большой набор заранее известных слов (обычно речь идет о десятках тысяч слов). На основании этого набора создается вектор чисел, где каждому слову соответствует определенная позиция в этом векторе с изначальным значением 0. Текст проверяется на наличие таких слов. Если слово присутствует в тексте, то соответствующая позиция устанавливается в 1, иначе в 0. На выходе получается разреженная матрица в виде вектора. Так же некоторые модификации этого метода используют размерности матрицы отличные от 1 по высоте. К недостаткам данного метода можно отнести:

- Большое количество затрачиваемой памяти. Текст из 40 слов и 4000 будут занимать одинаковое количество места. Решается оптимизацией хранения разреженных матриц
- Потерю смысла слов. Одно и то же слово может иметь несколько значений в зависимости от контекста
- Потерю порядка слов.
- Потеря количества определенных слов в тексте. Слово, употребленное в тексте один раз, имеет такой же вес, как и многократно употребленное слово.

Так же к недостаткам стоит отнести трудоёмкость предварительной подготовки текста. Для этого метода необходимо привести слова в один падеж, одно число. Например, "кота" -> "кот" "коты" -> "кошка". Так же слова-синонимы заменяются на один конкретный вариант. Например, "автомобиль-

>"машина". Это может быть нетривиально, особенно когда синонимы состоят из нескольких слов.

Основным преимуществом данного метода считается, что полученных данных достаточно для решения многих задач естественной обработки языка[23], в частности классификации. Т.е. для многих задач анализа естественного языка вышеописанные недостатки несущественны. Так же такой формат данных удобен для работы с нейронными сетями и другими алгоритмами машинного обучения.

Развитием алгоритма one-hot encoding является так называемый "мешок слов" . Для его получения ОНЕ-вектора каждого слова в тексте складываются. .е. на выходе получим просто подсчет количества различных слов в тексте в одном векторе. Такой подход называется "мешок слов" (bag of words, BoW), потому что мы теряем всю информацию о взаимном расположении слов внутри текста.

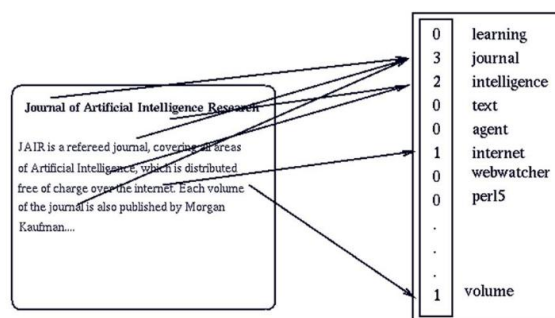


Рисунок 1.4 — Мешок слов

1.5.4.2 word2vec

Вышеописанные подходы были (и остаются) хороши для областей, где количество текстов мало и словарь ограничен, хотя, конечно, они имеют свои недостатки. Но в современном мире количество слов огромно, слова постоянно появляются и исчезают. С этим надо было что-то делать, а ранее известные модели не могли справиться с таким объемом текстов. Предположим, что в английском языке сейчас примерно около миллиона слов. Матрица совместных встречаемостей (такие матрицы используются для сравнения текстов) только пар слов будет иметь размер $10^6 \times 10^6$. Такие матрицы занимают очень много памяти, с ними неудобно работать. Конечно, придумано множество способов, упрощающих или распараллеливающих обработку таких матриц, всё-таки до сих пор с ними крайне тяжело работать[5].

Поэтому в 2013 году был предложен новый подход к word embedding, названный word2vec. Этот подход основан на гипотезе, которую в науке принято называть гипотезой локальности — Эслова, которые встречаются в оди-

наковых окружениях, имеют близкие значения. Близость в данном случае понимается очень широко, как то, что рядом могут стоять только сочетающиеся слова. Например, для нас привычно словосочетание "заводной будильник". А сказать "заводной арбуз" нельзя — эти слова не сочетаются.

Модель, предложенная Миколовым[24], проста. Она основывается на предсказании вероятности слова по его окружению (контексту). То есть ожидается получение таких векторов слов, чтобы вероятность, присваиваемая моделью слову была близка к вероятности встретить это слово в этом окружении в реальном тексте. Приведем формулу "мягкого максимума" (дифференцируемого). Максимум должен быть дифференцируем, чтобы модель могла обучаться при помощи алгоритма обратного распространения ошибки.

$$P(w_o|w_c) = \frac{e^{s(w_o, w_c)}}{\sum_{w_i \in V} e^{s(w_o, w_c)}} \quad (1.19)$$

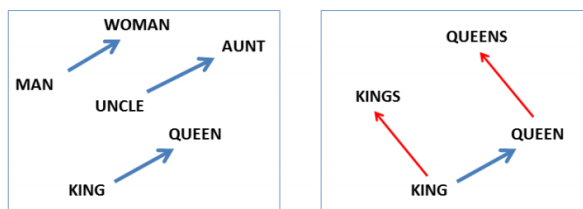
Где w_o - вектор целевого слова, w_c - некоторый вектор контекста вычисленный (например, путем усреднения) из векторов окружающих данное слово слов, $s(w_1, w_2)$ - некая функция, получающая на вход 2 вектора и отображающая их в одно число. Например, это может быть функция косинусного расстояния.

Модель обучается следующим образом: берется последовательность $2k + 1$ слов, где слово в центра является словом, которое должно быть предсказано. Окружающие слова являются контекстом длины K с каждой стороны от искомого слова. В процессе обучения каждому слову сопоставляется уникальный вектор, который меняется в процессе обучения.

Такой подход называется CBOW (англ. continuous bag of words). Является подвидом метода "мешка слов" т.к. порядок слов в контексте не учитывается.

Так же существует развитие данного метода - skip-gram[24] ("словосочетание с пропуском"). Оно основывается на том, что из данного нам слова мы пытаемся "угадать контекст". В остальном алгоритм не претерпевает изменений.

Натренированная модель word2vec может улавливать некоторые семантические и синтаксические свойства слов, несмотря на то что в модель явно не заложено никакой семантики. Слово "мужчина" (англ. man) относится к слову "женщина" (англ. woman) так же, как слово "дядя" (англ. uncle) к слову "тётя" (англ. aunt). Для человека это естественно и понятно, но в других моделях добиться такого же соотношения векторов можно только с помощью специальных ухищрений.



(Mikolov et al., NAACL HLT, 2013)

Рисунок 1.5 — word2vec пример семантической корреляции[24]

1.5.4.3 Улучшения word2vec

Модель CBOW с функцией оптимизации(минимизацией) в виде дивергенции Кульбака-Лейблера называют моделью CBOW с negative sampling[25].

$$KL(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx \quad (1.20)$$

Где $p(x)$ — распределение вероятностей слов, а $q(x)$ - распределение слов взятое из модели(которая была получена на основе текста). Эта формула имеет описывает насколько одно распределение не похоже на другое(англ. divergence - расхождение). Т.к. распределение получается на основе подсчёта слов, оно является дискретным. Поэтому можно заменить интеграл на сумму:

$$KL(p||q) = \sum_{x \in V} p(x) \log \frac{p(x)}{q(x)} dx \quad (1.21)$$

К сожалению, эта формула имеет недостаток - большую вычислительную сложность. Прежде всего, из-за того, что $q(x)$ рассчитывается как softmax на основании всего словаря текста. Именно поэтому Томасом Милковым[25] была предложена оптимизация вышеприведенной формула, названная Negative Sampling. Как уже отмечалось ранее, очевидно, что многие слова вместе не встречаются. Поэтому большая часть вычислений в softmax являются избыточными. Это позволяет ввести следующую оптимизацию: максимизируется вероятность встречи слова для нужного слова в типичном для него контексте(в том котором он чаще всего встречается в заданном тексте). А вероятность встречи слова в нетипичном для него контексте, соответственно, минимизируется.

$$NegS(w_0) = \sum_{i=1, x_i \sim D}^{i=k} - \log(1 + e^{s(x_i, w_0)}) + \sum_{j=1, x_j \sim D}^{j=k} - \log(1 + e^{s(x_j, w_0)}) \quad (1.22)$$

1.5.5 Нормализация данных

После получения предварительно обработанного датасета для обучения модели, данные необходимо нормализовать. Нормализация данных пред-

назначена для устранения зависимости от выбора единицы измерения и заключается в преобразовании диапазонов значений всех атрибутов к стандартным интервалам $[0,1]$ или $[-1,1]$ [26]. Нормализация данных направлена на придание всем атрибутам одинакового "веса".

1.5.5.1 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 $[A, B]$.

$$x'_i = \tau(x_i) = \frac{x_i - \min(x)}{\max(x) - \min(x)} * (B - A) + A \quad (1.23)$$

$$x \in [\min(x), \max(x)] \Rightarrow \tau() \Rightarrow [A, B] \quad (1.24)$$

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 [26] .

$$x'_i = \tau(x_i) = \frac{x_i - \bar{x}}{\sigma_x} \quad (1.25)$$

$$M[x'] = 1 \quad (1.26)$$

$$D[\bar{x}'] = 0 \quad (1.27)$$

Метод полезен когда в данных содержатся выбросы.

в) Масштабирование заключается в изменении длины вектора значений атрибута путем умножения на константу [27] .

$$x'_i = \tau(x_i) = \lambda * x_i \quad (1.28)$$

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

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.

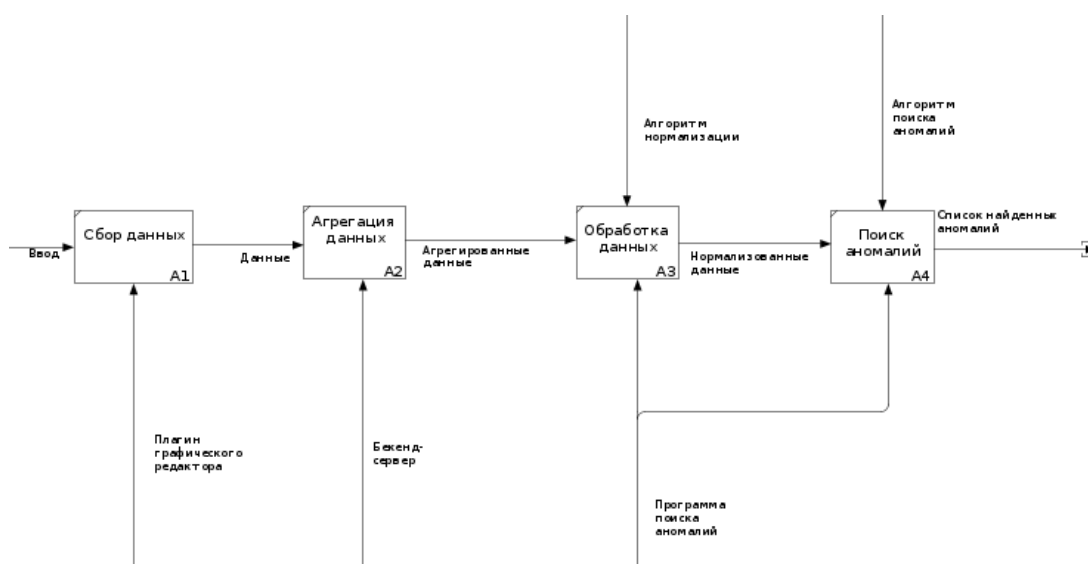


Рисунок 2.1 — Общая схема программного комплекса

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

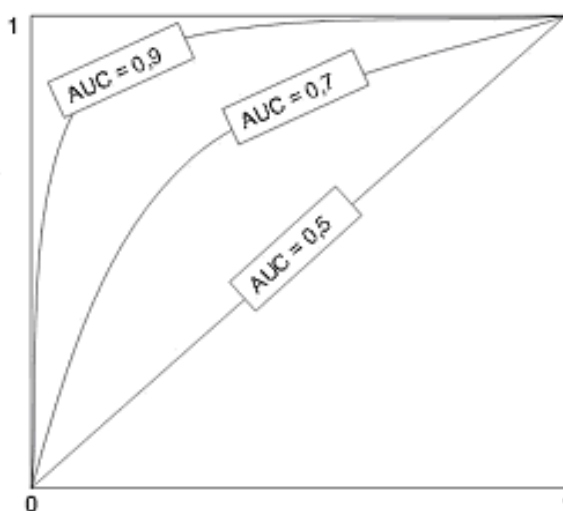


Рисунок 2.2 — AUC ROC

— 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 подобрать, основываясь на валидирующем наборе данных. [15] 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.

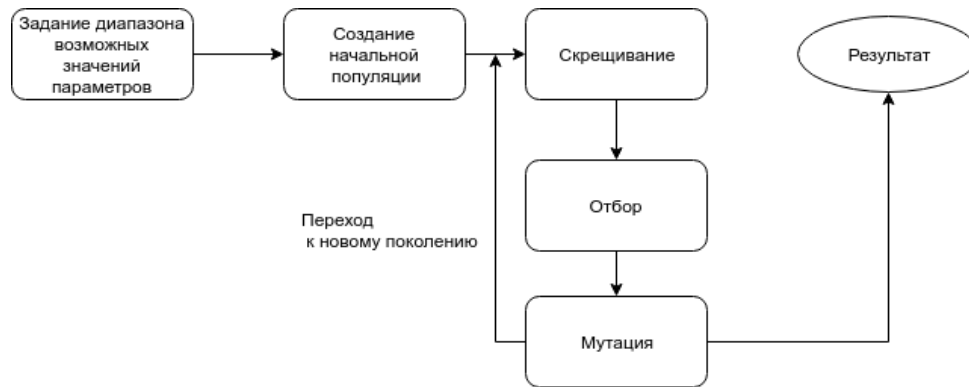


Рисунок 2.3 — Общий рисунок функционирования генетического алгоритма

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 расстояния 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.

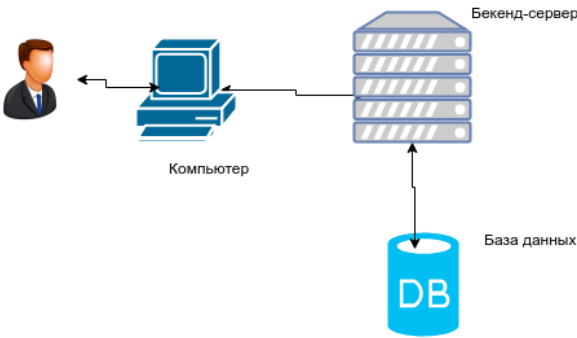


Рисунок 2.4 — Общая архитектура приложения

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 Time float64 ToolName string Timestamp float64
Действия	KisMainWindows actioncollection()	CountUse float64 TimeUse float64 ActionName string Timestamp float64
Свойства изображений	KisDocument::saveFile()	ColorProfile string ColorSpace string Height float64 Width float64 Size float64 NumLayers float64 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 `mongo` 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.

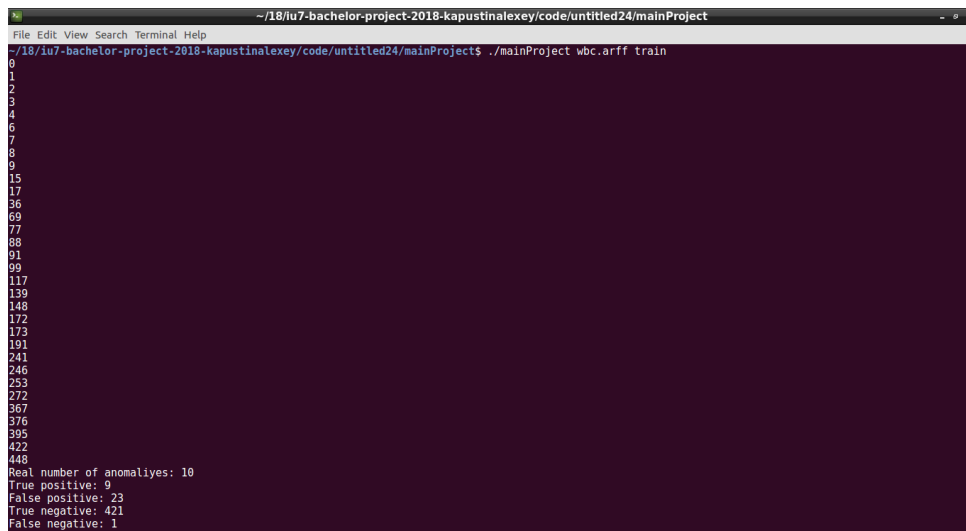
3.7 Установка программного обеспечения

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 обнаруженный 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] / Справочник технического переводчика. — 2018. URL:https://technical_translator_dictionary.academic.ru/8737/apiorная, дата обращения: 10.03.2020.
2. *Международная теннисная федерация*. IFT[Электронный ресурс] / Международная теннисная федерация. URL:<https://www.itftennis.com/>, дата обращения: 10.03.2020.
3. *Википедия*. [Электронный ресурс] / Википедия. URL:https://ru.wikipedia.org/wiki/\T2A\CYRR\T2A\cyre\T2A\cyrishrt\T2A\cyrt\T2A\cyri\T2A\cyrn\T2A\cyrg_ATP, дата обращения: 10.05.2020.
4. *Вичугова, Анна*. Отберем то, что нужно Data Mining: как сформировать датасет для машинного обучения [Электронный ресурс] / Анна Вичугова. — 2019. URL:<https://www.bigdataschool.ru/bigdata/dataset-data-preparation.html>, дата обращения: 10.03.2020.
5. *Малых, Валентин*. Чудесный мир Word Embeddings: какие они бывают и зачем нужны? / Валентин Малых. — 2017. URL:<https://habr.com/ru/company/ods/blog/329410/>, дата обращения: 09.05.2020.
6. *Крутиков, Александр*. Прогнозирование спортивных результатов в индивидуальных видах спорта с помощью обобщенно-регрессионной нейронной сети / Александр Крутиков. — Молодой ученый. — 2018. — №12., 2018. — с.22-26.
7. *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.
8. *Zdravevski E., Kulakov A*. System for Prediction of the Winner in a Sports Game / Kulakov A. Zdravevski E. — ICT Innovations 2009 — 2010., 2010. — с.55-63.
9. *Keller, J.B*. Tie point strategies in badminton, preprint / J.B. Keller. — 2003.
10. *Enrick, J.R*. Optimal strategies at decision points in singles squash / J.R. Enrick. — Quart. Exercise Sport, 1976.
11. *Burych, B.P. Hsi u D.M*. Games of two players / B.P. Hsi и D.M. Burych. — Journal of the Royal Statistical Society 22(1), 1971.
12. *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.

13. *Pollard, G.H.* An analysis of classical and tie-breaker tennis / G.H. Pollard. — Journal of the Australian Mathematical Society 25, 1983.
14. *F.Lassen, R.Magnus.* Are points in tennis independent and identically distributed? Evidence from a dynamic binary panel data model / R.Magnus F.Lassen. — Journal of the American Statistical Association 96(454), 2001.
15. *Terry M.E., Bradley R.A.(.* Rank analysis of incomplete block designs I: the method of paired comparison / Bradley R.A.(Terry M.E. — Biometrika, No. 39, 1952. — с.324-330.
16. *Morton A., McHale I.* A Bradley-Terry type model for forecasting tennis match results / McHale I. Morton A. — International Journal of Forecasting 27, 2011. — с.620.
17. *S.Clarke, D. Dyte.* Using official ratings to simulate major tennis tournaments / D. Dyte S.Clarke. — International Transactions in Operational Research 7(6), 2000. — с.585–594.
18. *S. Ma C. Liu, Y. Tan.* Winning matches in Grand Slam men's singles: an analysis of player performance-related variables from 1991 to 2008 / Y. Tan S. Ma, C. Liu. — Journal of sports sciences, 31(11), 2013. — с.1147–1155.
19. *A. Somboonphokkaphan S. Phimoltares, C. Lursinsap.* Tennis Winner Prediction based on Time-Series History with Neural Modeling / C. Lursinsap. A. Somboonphokkaphan, S. Phimoltares. — International Multi-Conference of Engineers and Computer Scientists, Vols I and II, 2009. — с.127-132.
20. *Sipko, Michal.* Machine Learning for the Prediction of Professional Tennis Matches / Michal Sipko. — Imperial College London [Электронный ресурс] URL:<https://www.doc.ic.ac.uk/teaching/distinguished-projects/2015/m.sipko.pdf>, 2015. — дата обращения: 29.03.2020.
21. *Shaikh, Raheel.* Feature Selection Techniques in Machine Learning with Python[Электронный ресурс] / Raheel Shaikh. URL:<https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7c> дата обращения: 25.04.2020.
22. *Valeria Fonti.* Feature Selection using LASSO[Электронный ресурс] / Valeria Fonti. URL:https://beta.vu.nl/werkstuk-fonti_tcm235-836234, дата обращения: 25.04.2020.
23. *Mayo, Matthew.* Data Representation for Natural Language Processing Tasks / Matthew Mayo. — 2018. URL:<https://www.kdnuggets.com/2018/11/data-representation-natural-language-processing.html>, дата обращения: 25.04.2020.

24. *"Tomas Mikolov Kai Chen, Greg Corrado. "Efficient estimation of word representations in vector space-/ Greg Corrado "Tomas Mikolov, Kai Chen, Jeffrey Dean". — 2013.*

25. *Tomas Mikolov Kai Chen, Greg Corrado. Distributed representations of words and phrases and their compositionality / Greg Corrado Tomas Mikolov, Kai Chen, Jeffrey Dean. — 27th Annual Conference on Neural Information Processing System, 2013. — с.3111–3119.*

26. *Андрей, Гахов. Интеллектуальный анализ данных / Гахов Андрей. — Харьковський національний університет імені В.Н. Карамзіна, 2014.*

27. *Enrick, J.R. Optimal strategies at decision points in singles squash / J.R. Enrick. — Quart. Exercise Sport, 1976.*