МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.04.02 Прикладная математика и информатика
(код и наименование направления подготовки)

Математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Моделирование синтеза классификаторов на основе аффинитивного анализа данных»

Студент Н.М. Шогунова
(И.О. Фамилия) (личная подпись)

Научный руководитель к.тех.н, В.С. Климов
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Содержание

Введение

Актуальность и научная значимость исследования определена необходимостью развития способов применения алгоритмов машинного обучения при решении практических задач.

Машинное обучение в последние годы получило широкое распространение во многих отраслях науки и техники. Рост популярности обусловлен используемым подходом к построению моделей объектов, который основан автоматизированном анализе частных эмпирических данных [1, 13, 24].

Универсальность алгоритмов машинного обучения обусловлена формализацией типов математических задач, на решении которых они направлены. Выделяют такие задачи, как классификация, кластеризация, регрессия, аффинитивный анализ, поиск аномалий и т.д. Каждый алгоритм машинного обучения связан с решением одного из этих типов задач [5, 10, 18, 27].

В настоящем исследовании предполагается, что существует возможность расширения возможностей алгоритмов машинного обучения за счет разработки способов переноса алгоритмов на решение других типов задач. В магистерской диссертации исследуются способы применения алгоритмов аффинитивного анализа для решения задач классификации.

Объектом исследования является аффинитивный анализ данных, предметом исследования – разработка способа построения классификатора данных на основе результатов аффинитивного анализа.

Цель исследования – разработка технологии построения классификатора данных на основе алгоритмов аффинивного анализа данных (на примере алгоритма Apriori).

Гипотеза исследования состоит в том, что ассоциативные правила, полученные в результате аффинитивного анализа можно использовать для генерирования классификатора данных.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проведение анализа состояния вопроса по теме исследования.

2. Разработка технологии синтеза классификаторов данных на основе алгоритмов аффинитивного анализа;

3. Разработка программного обеспечения, реализующего предложенную технологию;

4. Тестирование алгоритма синтеза классификатора на данных из репозитория и обсуждение результатов.

В ходе выполнения работы применялись такие методы теоретического исследования, как изучение и анализ научной литературы по проблемам развития алгоритмов машинного обучения.

Также в ходе выполнения работы применялись практические методы исследования, такие как проведение вычислительных экспериментов, обработка статистических данных, программное моделирование работы алгоритмов машинного обучения.

Научная новизна исследования – доказано, что классификатор данных можно синтезировать на основе аффинитивного анализа исходных данных. Причем, как показывают вычислительные эксперименты в исследовании, точность работы полученного классификатора будет соизмерима с классификаторами, основанными на работе алгоритмов Random Forest и kNN.

Теоретическая значимость заключается в разработке подходов построения классификатора данных на основе ассоциативных правил. Предложена методика преобразования категориальных и числовых значений атрибутов в элементарные события, подвергающиеся аффинитивному анализу. Заработана технология синтеза классификатора данных на основе ассоциативных правил.

Практическая значимость работы заключается в разработке программного обеспечения реализующего предложенные подходы.

Достоверность и обоснованность результатов исследования обеспечивалась множеством вычислительных экспериментов на данных из репозитория «The UCI Machine Learning Repository» и сравнении результатов классификации данных с другими алгоритмами машинного обучения (Random Forest, kNN).

Личное участие автора в организации и проведения исследования состоит в выдвижении гипотезы, проведении теоретических исследований, разработке алгоритма синтеза классификатора данных на основе алгоритма Apriori, проведении вычислительных экспериментов и обработке полученных результатов.

Апробация и внедрение результатов работы велись в течение всего исследования. Его результаты докладывались на Всероссийской студенческой научно-практической междисциплинарной конференции «Молодежь. Наука. Общество»

1 Анализ перспектив развития алгоритмов машинного обучения

1.1 Анализ тенденций в области машинного обучения

Машинное обучение – является главным разделом искусственного интеллекта, направленного на создание алгоритмов, способных проводить автоматизированный анализ данных и строить обобщенные модели анализируемых данных [2, 6, 9].

Машинное обучение в последние годы получило широкое распространение во многих отраслях промышленности. Это связано с тем, объекты управления и диагностики в промышленности становятся всё сложней, увеличивается трудоемкость построения моделей этих объектов путем математического моделирования связанных с ними процессов. Машинное обучение предлагает другой подход к моделированию, который основан на анализе частных эмпирических данных об объекте с использованием универсальных алгоритмов [3, 12, 23].

Как показывает уже наработанная мировым сообщество практика – такой подход, основанный на машинном обучении, позволяет многократно снизить трудоемкость решения задач моделирования.

Для понимания проблемы можно привести следующий пример. С использованием классического математического моделирования описать модель человека (в разных позах, в разной одежде и т.д.) на всем множестве возможных изображений является сложно-формализуемой задачей. Однако с использованием алгоритмов машинного обучения, при наличии набора примеров изображений людей, эта задача решается автоматизировано. Алгоритм сам определит достаточный набор признаков, позволяющих с заданной точностью определять не только наличие человека на изображении, но и выделять его.

Для того чтобы появилась возможность решить поставленную задачу с помощью машинного обучения ее надо привести к одному из типов задач.

Существуют следующие типы задач, решаемых с помощью алгоритмов машинного обучения: задача регрессии, задача классификации, задача кластеризации, задача аффинитивного анализа, задача оптимизации, задача поиска аномалий.

Одну и ту же практическую задачу можно представлять в виде разных типов задач. Например, поиск пешеходов на изображениях можно представить как задачу классификации, так и как задачу кластеризации изображения (в которой пешеходы объединены в отдельный кластер) [4].

В зависимости от типа рассматриваемой задачи ограничивает множество алгоритмов машинного обучения, которые можно применить для ее решения [7]. Так, например, с помощью нейронной сети Хэмминга, предназначенной для решения задач классификации нельзя построить регрессионную модель и т.д.

Однако существуют отдельные алгоритмы машинного обучения, которые изначально задумывались для решения несколько типов задач. Сюда относиться алгоритм CART построения деревьев принятия решений. Он может строить деревья классификации, так и регрессионные деревья.

Вопрос о том, какой из алгоритмов машинного обучения больше подходит для решения текущей задачи требует проведения вычислительных экспериментов. Никогда заранее не известно, какой из алгоритмов покажет большую точность в процессе своей работы.

Изучение литературных источников по теме практического применения алгоритмов позволило сформировать обобщённую таблицу, в которой дано соответствие алгоритмов машинного обучения и типов задач, на решение которых они направлены. Также в этой таблице приведены примеры решаемых задач (таблица 1.1) [8, 11, 21].

Таблица 1.1 – Задачи и примеры задач, решаемые с помощью алгоритмов машинного обучения

Тип решаемой задачи	**Примеры задач**	**Алгоритмы машинного обучения**
Регрессия	- прогнозирование прибыли компании на основе финансовых показателей; - аппроксимация экспериментальных данных набором функций (рисунок 1.1).	- алгоритмы построения деревьев регрессии (decision tree regressor); - нейронные сети (neural networks).
Классификация	- распознавание объектов на изображении (рисунок 1.2); - диагностика состояния технического объекта; - определения стратегии действия в условиях неопределенности.	- алгоритмы построения деревьев классификации (Decision Tree Classifier); - метрические алгоритмы (kNN); - метод опорных векторов (support vector machines); - нейронные сети (neural networks).
Кластеризация	- сегментация изображения (рисунок 1.3); - распределение клиентов по группам.	- семейство алгоритмов k-средних (k-means); - нейронные сети (neural networks).
Аффинитивный анализ	- определение предпочтений клиента в зависимости от выбираемых им товаров (рисунок 1.4); - поиск причинно-следственных связей в	- алгоритм Apriori; - алгоритм Eclat; - алгоритм FP-growth.

	событиях, происходящих с объектом и процессом.	
Оптимизации	- балансировка нагрузки на оборудование; - задача поиска оптимальных режимов работы объекта или осуществления процесса; - решение np-полных задач (задача рюкзака, задача коммивояжера, задача линейного раскроя) - обработка изображений (рисунок 1.5)	- муравьиный алгоритм (colony optimization); - генетические алгоритмы (genetic algorithms).
Выявление аномалий	- выявление атипичных активностей при обеспечении информационной безопасности (рисунок 1.6); - обеспечение функционирования охранных систем, систем безопасности, диагностических систем в медицине; - очистка данных от шумовых и некорректных значений.	- метод опорных векторов (support vector machines); - байесовские сети доверия (bayesian network); - методы, основанные на нечеткой логике (fuzzy logic-based outlier detection).

Рассмотрим по одному примеру для каждого типа задачи и таблицы 1, решаемой с помощью алгоритмов машинного обучения.

При выполнении исследовательских работ часто требуется искать закономерности в данных, полученных экспериментальным путем. Замена экспериментальных данных регрессионной моделью позволяет заполнить промежутки в тех областях значений независимых переменных, где экспериментальные данные отсутствуют. Также для хранения регрессионной модели требуется намного меньше места, чем для хранения большого массива экспериментальных данных [15, 17, 31].

Раньше для аппроксимации экспериментальных данных функцией требовалось выполнение предварительного анализа с целью определения наиболее подходящего вида функции и стратегии подбора ее коэффициентов. Сейчас, благодаря использованию алгоритмов машинного обучения, регрессионная модель данных строится с высокой степенью автоматизации, практически без участия человека.

Для построения регрессионных моделей данных чаще всего применяются нейронные сети прямого распространения и алгоритмы построения регрессионных деревьев [22, 25].

На рисунке показан 1.1 пример аппроксимация экспериментальных данных (которые показана на графике синими точками) с помощью различных алгоритмов машинного обучения:

- Синей линией на графике показана модель данных, полученная с использованием двухслойной нейронной сети. Для обучения сети применялся алгоритм обратного распространения ошибки.
- Зеленой линией показана модель данных, описанная деревом регрессии, которое построено с использованием алгоритма CART.

Для сравнения эффективности моделей используют различные показатели, такие как коэффициент детерминации, средняя квадратичная ошибка, накопленная сумма квадратов ошибок и другие.

Рисунок 1.1 – Аппроксимация экспериментальных данных, показанных синими точками, с помощью нейронной сети (синяя линия) и с помощью регрессионного дерева принятия решений (зеленая линия)

При разработке систем компьютерного зрения часто требуется находить на изображениях требуемые объекты и проводить их распознавание [26, 30]. Разработка таких алгоритмов анализа изображений чаще всего формализуется разработчиками, как задача классификации данных. В качестве примера такого подхода можно привести систему анализа изображений компании Lucidyne Technologies, которая занимается разработкой оборудования для обработки дерева.

Система анализа изображений просматривает изготавливаемые изделия и анализирует наличие дефектов разных типов на их поверхности. Пример работы данной системы компьютерного зрения показан на рисунке 1.2. Разные классы дефектов размечаются различными цветами.

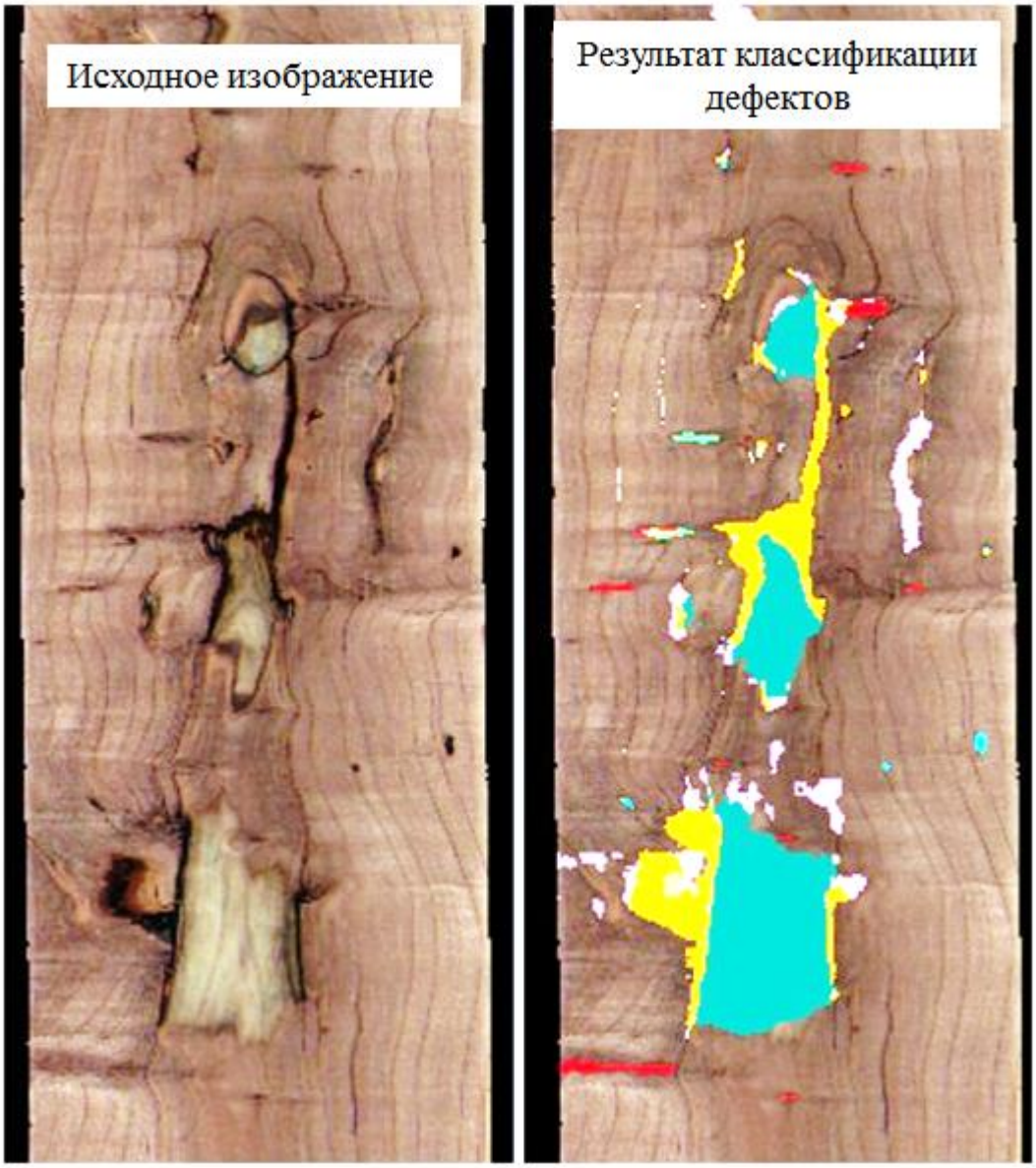


Рисунок 1.2 – Система машинного зрения, используемая компанией Lucidyne Technologies для классификации дефектов на пиломатериалах

В системах компьютерного зрения, для упрощения процедуры анализа, рассматриваемое изображение разделяется по смыслу на фрагменты. Например, часто требуется отделить объекты переднего плана от фона для их дальнейшего анализа.

Процедура разделения исходного изображения на фрагменты называется сегментацией изображения. Сегментацию можно представить в виде задачи кластеризации, где объектами кластеризации являются пиксели изображения. Результаты кластеризации зависят от заданного набора свойств

пикселей. Например, можно при кластеризации использовать только цветовые характеристики пикселей, в другом случае, помимо цвета необходимо учитывать и их расположение. С помощью коэффициентов в метрике сходства объектов можно определять, какие из характеристик будут сильнее влиять на результат кластеризации [16, 29].

Рисунок 1.3 – Пример решения задачи сегментации изображения с использованием алгоритма кластеризации данных k-means

На рисунке 1.3 показан пример сегментации изображения с помощью алгоритма кластеризации k-means.

Причина	Следствие	Под-держка (S)	Досто-верность (C)	Значимость правила (C x S)
{кабачки}	{фасоль}	42,9 %	85,7 %	0,3677
{спаржа}	{фасоль}	35,7 %	83,3 %	0,2974
{спаржа}	{кабачки}	35,7 %	83,3 %	0,2974
{капуста}	{перец}	28,6 %	80 %	0,2288
{перец}	{капуста}	28,6 %	80 %	0,2288
{спаржа и фасоль}	{кабачки}	28,6 %	80 %	0,2288
{спаржа и кабачки}	{фасоль}	28,6 %	80 %	0,2288

Рисунок 1.4 – Применения аффинитивного анализа для поиска закономерностей в покупках клиентов на примере продуктовой лавки (ассоциативные правила отсортированы в порядке убывания произведения поддержки и достоверности

Машинное обучение также используется в задачах аффинитивного анализа данных. К данной задаче относится проблема поиска закономерностей в событиях происходящих одновременно [19].

Самой известной практической задачей, решаемой с помощью аффинитивного анализа, является определение закономерностей продуктовой корзины клиентов супермаркета. Для того, чтобы увеличить прибыль магазина, проводится анализ всех покупок клиентов за отчетный период и, на основе анализа, генерируется набор ассоциативных правил указывающих, как как связаны покупки одних товаров с покупками других. На основе полученных результатов принимаются управленческие решения по

изменению ценовой политики магазина, а также формируются акции на совместную покупку товаров.

На рисунке 1.4 показаны ассоциативные правила, сформированные на основе покупок клиентов продуктовой лавки.

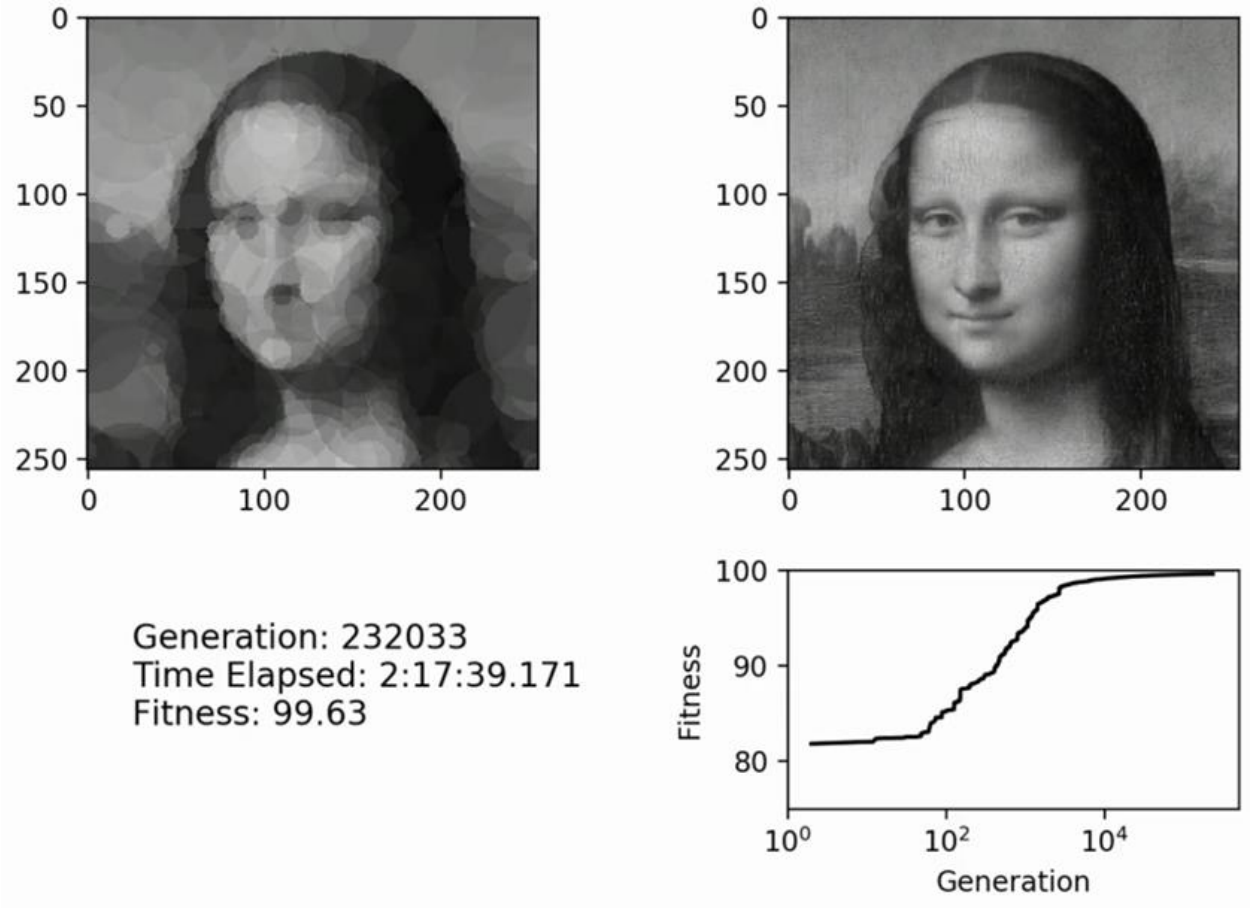


Рисунок 1.5 – Использование генетического алгоритма для обработки изображения (generation – количество итераций, потребовавшихся для обработки, time elapse – время потраченное на обработку изображения, fitness – значение функции приспособленности)

Помимо задач кластеризации и классификации при обработке изображений также часто приходится решать задачи оптимизации [28]. Примерами таких задач является определение оптимальных параметров контраста и яркости изображения, повышение четкости изображения и т.д.

При использовании алгоритмов машинного обучения большинство задач оптимизации решается с помощью генетических алгоритмов. Генетический алгоритм проводит стохастический поиск оптимальных

параметров изображения. Количество итераций заранее известно, алгоритм продолжает поиск решения, пока удается на последующих итерациях улучшать получаемый результат [14].

Пример обработки изображения с использованием генетического алгоритма показан на рисунке 1.5.

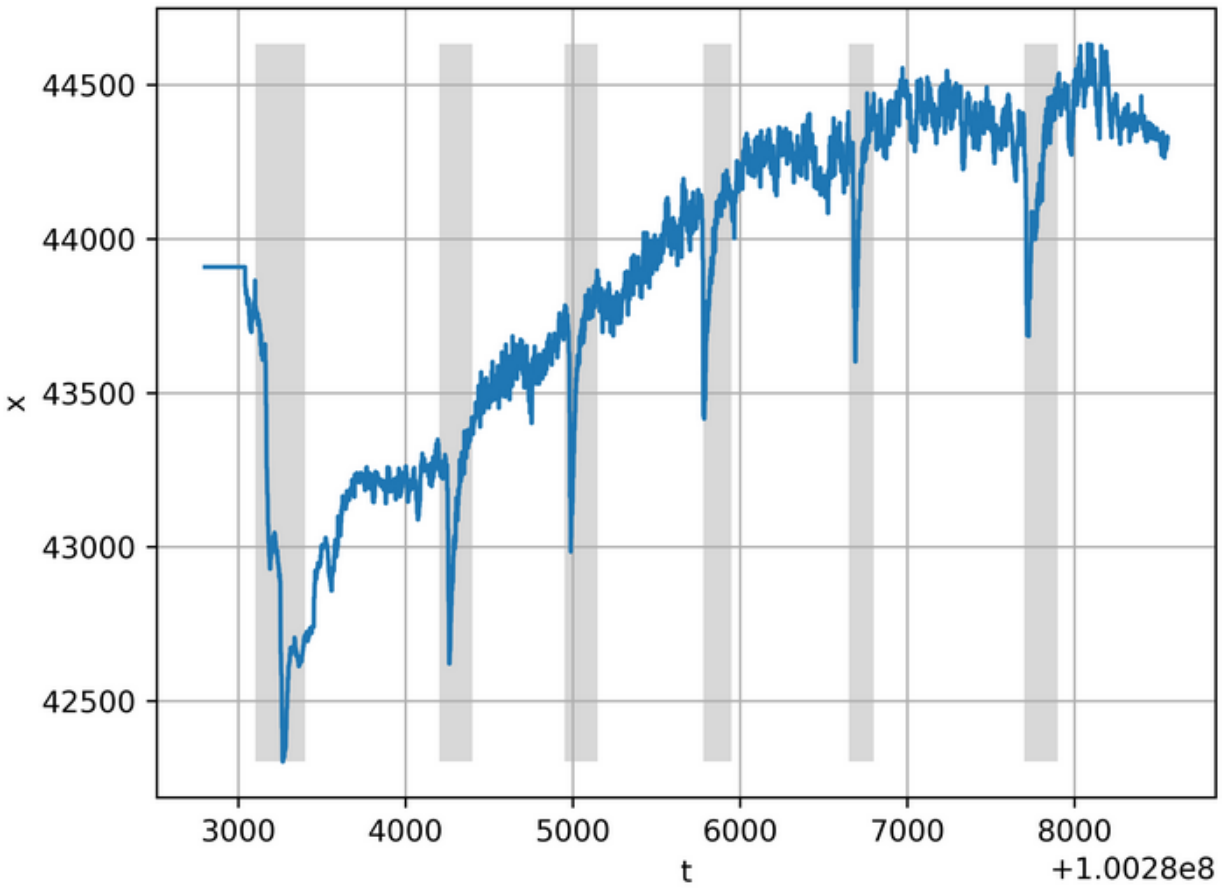


Рисунок 1.6 – Задача поиск аномалий на примере анализа временной диаграммы использования оперативной памяти на сервере (t –время с начала запуска сбора статистики, x – объем занятой оперативной памяти).

Отдельным типом задач является – поиск аномалий в данных. Данный тип задач распространен в системах обеспечения информационной безопасности и в системах Data Mining (в подсистемах очистки данных) [32].

Сложность решении задач поиск аномалий заключается в том, что доля примеров аномалий в обучающей выборке данных обычно очень мала. Поэтому задачу поиска аномалий не льзя рассматривать как задачу

классификации. Для решении задачи поиска аномалий применяются различные алгоритмы машинного обучения, такие как SVM и нейронные сети [20].

На рисунке 1.6 представлен пример решения задачи поиска аномалий в графике использовании оперативной памяти в на сервере за заданный временной промежуток. Периоды резкого изменения значения используемой оперативной памяти являются аномалиями на которые стоит обратить внимание. В этом случае задача поиска аномалий решается с помощью нейронной сети, которая выделила найденные участки серым цветом.

1.2 Построение классификатора для анализа изображений

При написании магистерской работы также исследовались особенности применения алгоритмов машинного обучения для решения задач локализации объектов на изображении. Для локализации объектов необходимо обучить классификатор способный относительного любого фрагмента анализируемого изображения определить присутствует там искомый объект или нет (задача бинарной классификации).

Результаты проведенного анализа были опубликованы в работе «Разработка программы компьютерного зрения для локализации объектов на изображении и видеопотоке» в сборнике Всероссийской студенческой научно-практической междисциплинарной конференции «Молодежь. Наука. Общество»

Работа посвящена применению алгоритма Виолы-Джонса и гистограмм ориентированных градиентов в задаче распознавания разного рода объектов на изображении и видеопотоке.

Объектом исследования является сравнительный анализ признаков цифрового изображения Хаара и алгоритм классификации машина опорных векторов. Предметом исследования является определение наборов пикселей

изображения попадающих под применяемые каскадные классификаторы и количество распознанных регионов с применением гистограмм ориентированных градиентов.

Целью проекта является разработка приложения компьютерного зрения для локализации объектов на изображении и видеопотоке.

Задачами исследования являются: проектирование и разработка технического проекта, проектирование и разработка приложения по разработанному технического проекту.

Новизна решения с научной точки зрения заключается в использовании передовых технологий в области компьютерного зрения и искусственного интеллекта, а также современные подходы к проектированию приложения.

В ходе проведения исследования производился сравнительный анализ стандартного алгоритма Виолы-Джонса и гистограмм ориентированных градиентов.

Метод Виолы-Джонса работает по следующему алгоритму: имеется входное изображение, представленное в виде двухмерной матрицы, элементы которой представляют собой значения яркостей пикселей (числа в диапазоне от 0 до 255). Если входное изображение представлено в чёрно-белых тонах, то достаточно одной двухмерной матрицы, если же изображение является цветным, то такое изображение преобразуется в три двухмерные матрицы (по одной матрице на каждый компонент цветовой модели RGB).

Одним из главных принципов алгоритма Виолы-Джонса является преобразование черно-белого кадра изображения в интегральное представление (рисунок 1.7).

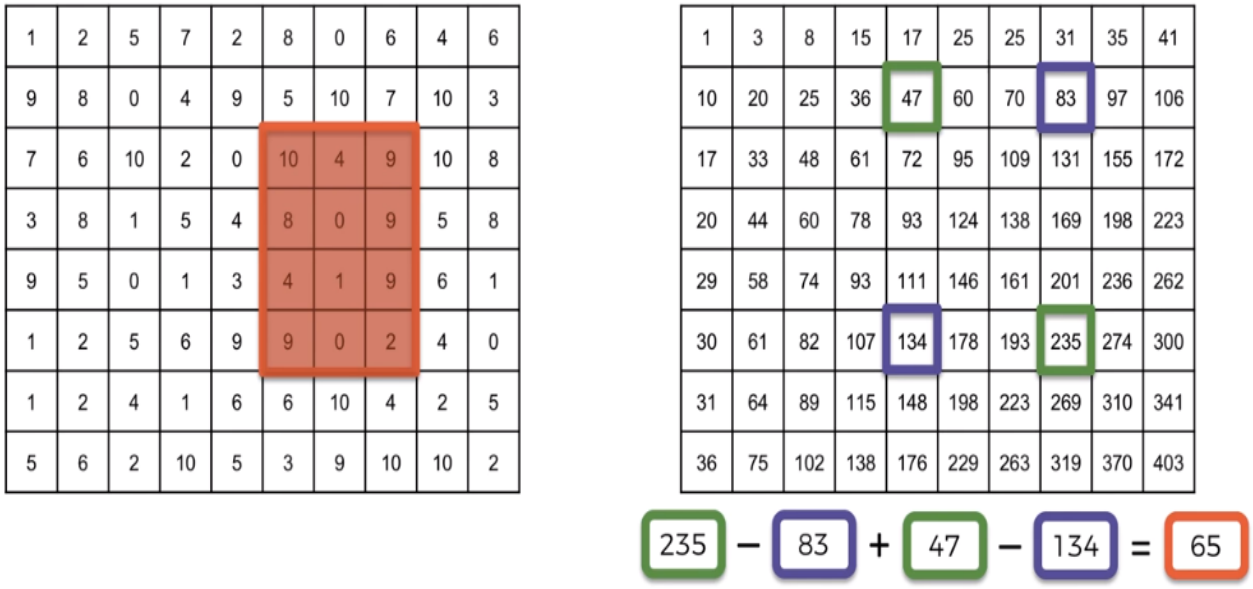


Рисунок 1.7 – Пример вычисления интегрального изображения

В каждом элементе интегральной матрицы содержится сумма интенсивностей пикселов, находящихся левее и выше рассматриваемого элемента. Эта манипуляция позволяет посчитать суммарную яркость произвольного прямоугольника, не зависимо от его размеров, что отлично подходит для использования признаков Хаара. С помощью просуммированных яркостей и признаков Хаара после вычисления значимого признака выноситься заключение – имеется на входном кадре изображения распознаваемый объект или нет.

Пользователь взаимодействует с встроенными функциями приложения через графический интерфейс (GUI). Интерфейс соответствует современным требованиям по разработке и обеспечивает быстрый доступ ко всем встроенным функциям приложения.

Приложение включает в себя выбор языка пользователя, для удобства использования. Оно также обладает хорошей надёжностью, так как внутри заложены функции по проверки возможного поведения пользователя. Вместе с приложением предоставляется проработанная документация, в которой пользователь найдёт для себя всевозможные ответы. Приложение является достаточно легковесным и не потребляет множество ресурсов устройства, на

котором оно установлено.

Области применения приложения - компьютерное зрение и искусственный интеллект. Оно найдёт своё применение в сферах туризма, образования, здравоохранения, а также при создании среды «умного» города.

Имеющиеся аналоги приложения: FindFace и SearchFace. Приложение FindFace получает на вход фотографию человека, которого следует найти в интернет-пространстве, производит обработку и собирает данные с социальных сетей, форумов и всевозможных открытых хранилищ фотографий. Данная приложение сопровождается приятным графическим интерфейсом пользователя. Исходный код приложения закрыт. SearchFace – аналог приложения FindFace, специализирующийся на социальной сети «ВКонтакте». Это приложение также находит похожего человека по фотографии, анализируя при этом пятьсот фотографий людей. Приложение основано на собственном алгоритме распознавания и поиска людей без применения открытой библиотеки N-Tech.Lab.

Примеры работы алгоритма гистограмм ориентированных градиентов и Виолы-Джонса представлены на рисунках 1.8 и 1.9.

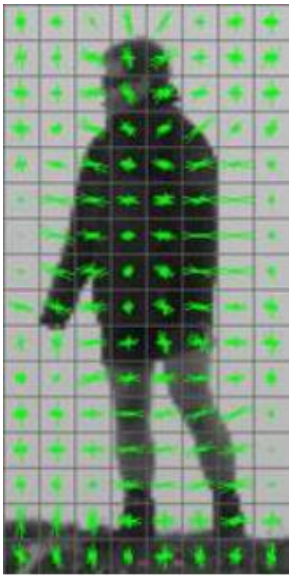


Рисунок 1.8 – Пример работы алгоритма Гистограмм ориентированных градиентов (HOG)

Рисунок 1.9 – Пример работы алгоритма Виолы-Джонса для распознавания автомобильных номеров

В ходе выполнения разработки приложения были получены и усвоены знания в области компьютерного зрения и искусственного интеллекта, а также усовершенствованы навыки по разработке систем компьютерного зрения.

1.3 Перспективы развития алгоритмов машинного обучения

Анализ литературных источников в области машинного обучения показал, что изначально, при использовании дедуктивных технологий машинного обучения разрабатываемые системы работали только на основе жестко заданных программистами правил анализа (т.н. база зананий).

С появлением алгоритмов индуктивного машинного обучения, системы анализа приобрели возможность самостоятельно приобретать новые знания на основе данных, содержащихся в обучающейся выборке. Несмотря на это, здесь по-прежнему требуется участие аналитиков для определения значимых признаков и подготовки обучающей выборки.

Через некоторое время появились алгоритмы машинного обучения способные самостоятельно определять, какие из атрибутов в обучающей выборке являются значимыми. Такие алгоритмы фактически сами осуществляют фильтрацию обучающей выборки и не требуют нормировки исходных данных.

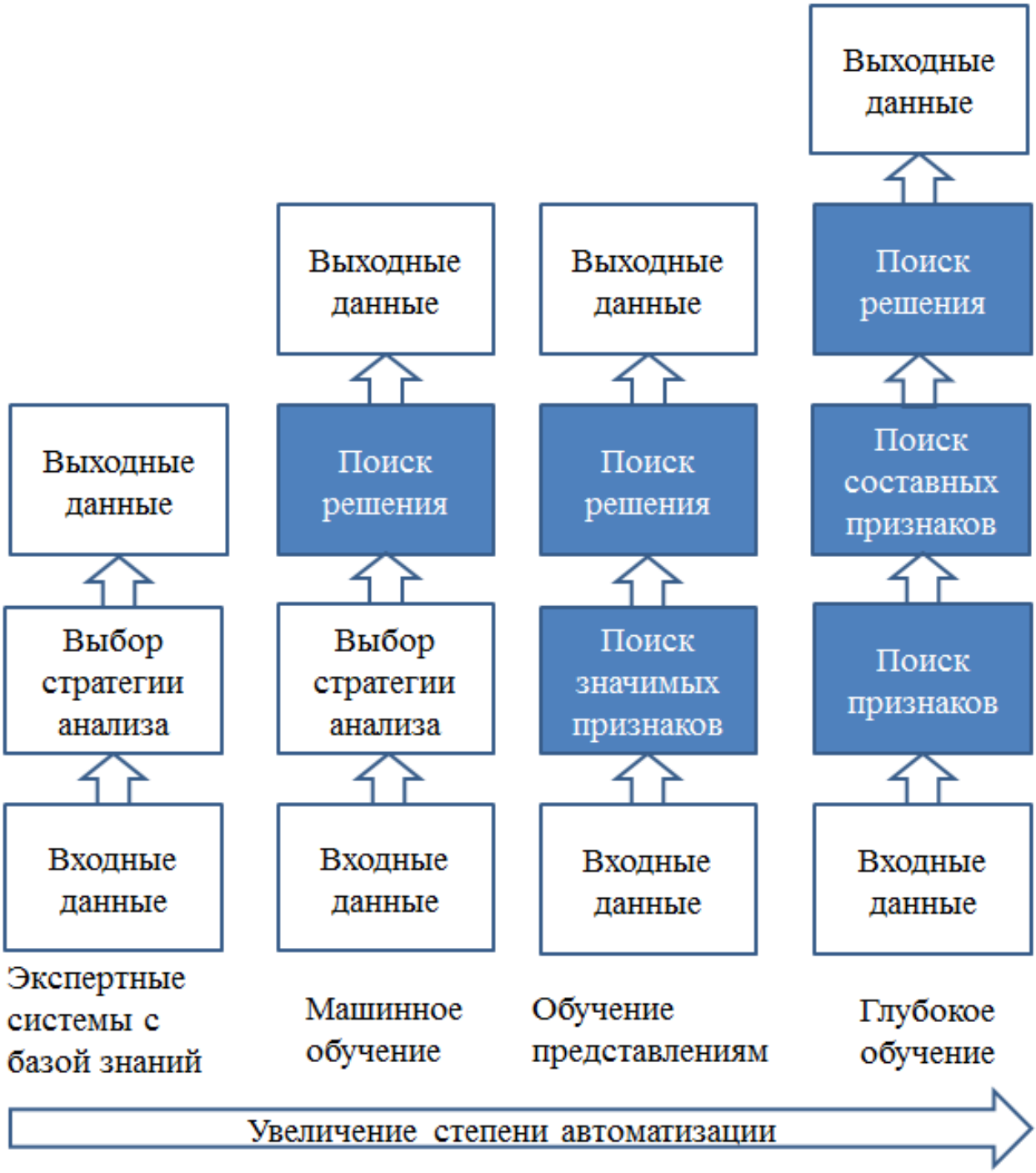


Рисунок 1.10 – Одна из тенденций развитие машинного обучения – увеличение степени автоматизации алгоритмов и снижение доли участие человека в анализе данных (синим цветом выделены блоки, которые выполняются без участия человека)

Последним достижением в этой области являются алгоритмы глубокого машинного обучения. Сюда относятся алгоритмы, которые позволяют не только определять, какие атрибуты обучающей выборки

являются значимыми, но и обобщать имеющиеся атрибуты в новые составные признаки.

На основе вышесказанного можно сделать вывод, что, на текущий момент, сформировалась тенденция на развитие технологий машинного обучения в сторону увеличению степени автоматизации алгоритмов.

Разработка новых алгоритмов машинного обучения является длительной и трудоемкой задачей, связанной с необходимостью подтверждения научным сообществом ценность предлагаемых решений. Поэтом актуальным стоит признать вместо разработки новых алгоритмов разработку новых способов применения существующих алгоритмов, расширяющих перечень задач, которые они могут решать (рисунок 1.11).

На основе этих рассуждений, в настоящем исследовании сформирована гипотеза о возможности построение классификатора данных с использованием алгоритмов аффинитивного анализа (на примере алгоритма Apriori).

Подтверждение это гипотезы позволит пополнить список алгоритмов машинного обучения направленных на решение задач классификации (алгоритмы построения деревьев классификации (Decision Tree Classifier), метрические алгоритмы (kNN), метод опорных векторов (support vector machines), нейронные сети (neural networks)) целым списком дополнительных алгоритмов (алгоритм Apriori, алгоритм Eclat, алгоритм FP-growth).

Таким образом, сформирована цель исследование – разработка технологии построения классификатора данных на основе алгоритмов аффинивного анализа данных (на примере алгоритма Apriori).

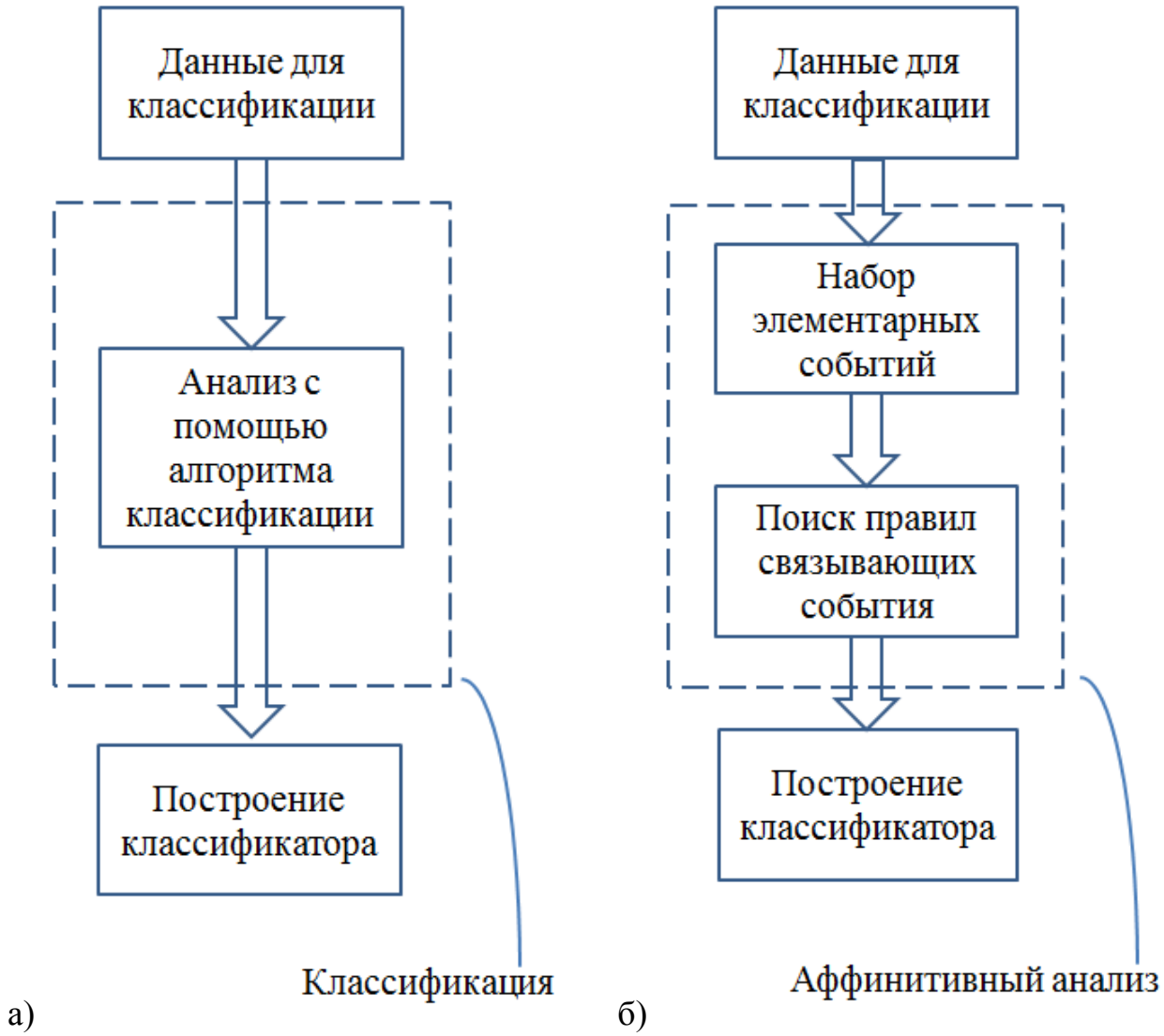


Рисунок 1.11 – Один из путей развития технологий машинного обучения – разработка способов переноса алгоритмов на решение других типов задач: а – построение классификаторов с помощью алгоритмов классификации; б – построение классификаторов с помощью алгоритмов аффинитивного анализа

Цель исследования достигается последовательным решение следующих задач:

- обзор литературных источников по направлению способов применения алгоритмов машинного;
- разработка технологии синтеза классификаторов данных на основе алгоритмов аффинитивного анализа;
- разработка программного обеспечения, реализующего предложенную технологию;

- тестирование алгоритма синтеза классификатора на данных из репозитория и обсуждение результатов.

Выводы по разделу

По результатам представленных в данной главе исследований сформированы следующие выводы:

- Анализ литературных источников показал, что применение технологий машинного обучения, позволяет многократно снизить трудоемкость решения задач моделирования.
- В ходе анализа существующих исследований по практическому применению технологий машинного обучения сформирована сводная таблица устанавливающая соответствие между типами решаемых задач и используемыми алгоритмами. Для каждого типа задачи приведены примеры использования алгоритмов машинного обучения
- На основании анализа литературных источников установлено, что развитие алгоритмов машинного обучения сконцентрировано по двум направлениям: первое – увеличение степени автоматизации алгоритмов при обработке исходных данных и второе – разработку новых способов применения существующих алгоритмов, расширяющих перечень задач, которые они могут решать.
- На основании предыдущего вывода доказана актуальность проводимого исследования на тему моделирование синтеза классификаторов на основе аффинитивного анализа данных.

2 Разработка технологии синтеза классификаторов на основе аффинитивного анализа данных

2.1 Математическая модель алгоритма Apriori

Аффинитивный анализ (affinity analysis) — набор методов интеллектуального анализа данных, направленный на поиск и исследование взаимной связи (ассоциаций) между событиями, происходящими совместно, и количественную оценку таких связей.

Название такого подхода происходит от английского слова affinity, означающее в переводе «близость», «сходство». Целью анализа является обнаружение ассоциации между различными событиями, то есть найти правила для количественного описания взаимной связи между двумя или более событиями. Такие правила называются ассоциативными правилами (association rules).

Исходными данными для проведения аффинитивного анализа является набор транзакций T, состоящий из элементарных транзакций *t*:

$$T = \{t_1, t_2, ..., t_m\}, \tag{2.1}$$

где m – количество транзакций.

T	Транзакции
t_1	$\{i_3, i_9, i_{10}, i_{13}\}$
t_2	$\{i_1, i_3, i_{10}\}$
t_3	$\{i_2, i_9\}$
...	...
t_m	$\{i_3, i_9, ..., i_{10}, i_{13}\}$

Рисунок 2.1 – Исходные данные для аффинитивного анализа

В каждой транзакции содержится набор событий, происходящих одновременно. Набор всех событий задан множеством *I*:

$$I = \{i_1, i_2, ..., i_n\} \tag{2.2}$$

где, n – количество элементарных событий.

На множестве транзакций T, с использованием множества элементарных событий I можно сформировать ассоциативные правила. Ассоциативное правило – это импликация (бинарная логическая связка) вида:

$$\begin{gathered} X \Rightarrow Y \\ X \subset I, Y \subset I, X \cap Y = \varnothing \end{gathered} \tag{2.2}$$

где, I – множество всех событий, X – множество событий, называемых условием (antecedent), Y – множества событий, называемых следствием (consequent).

Ассоциативные правила описывают связь между наборами предметов, соответствующими условию и следствию. Эта связь характеризуется такими показателями (метриками), как поддержка s*upp*, достоверность *conf*, лифт *lift*, левередж *levr*.

Поддержка s*upp* ассоциативного правила — это отношение числа транзакций, которые содержат как условие, так и следствие к общему количеству транзакций.

$$supp(X \Rightarrow Y) = \frac{\left|\{t \in T; (X \cup Y) \subseteq t\}\right|}{\left|T\right|}, \tag{2.3}$$

где T – множество транзакций, t – транзакция.

Для любого предметного набора A также может быть рассчитана поддержка следующим образом:

$$supp(A) = \frac{\left|\{t \in T; A \subseteq t\}\right|}{\left|T\right|} \tag{2.4}$$

Достоверность *conf* ассоциативного правила представляет собой меру точности правила и определяется как отношение количества транзакций, содержащих и условие, и следствие, к количеству транзакций, содержащих только условие:

$$conf(X \Rightarrow Y) = \frac{supp(X \Rightarrow Y)}{supp(X)} \quad (2.5)$$

Лифт *lift* — это отношение частоты появления условия в транзакциях, которые также содержат и следствие, к частоте появления следствия в целом:

$$lift(X \Rightarrow Y) = \frac{supp(X \Rightarrow Y)}{supp(X) \times supp(Y)} \quad (2.6)$$

Значения лифта большие, чем единица, показывают, что условие чаще появляется в транзакциях, содержащих следствие, чем в остальных. Можно сказать, что лифт является обобщенной мерой связи двух предметных наборов: при значениях лифта > 1 связь положительная, при 1 она отсутствует, а при значениях < 1 — отрицательная.

Левередж *levr* — это разность между наблюдаемой частотой, с которой условие и следствие появляются совместно (то есть поддержкой ассоциации), и произведением частот появления (поддержек) условия и следствия по отдельности:

$$levr(X \Rightarrow Y) = supp(X \Rightarrow Y) - supp(X) \cdot supp(Y) \quad (2.7)$$

Чем выше значение *lev*, тем сильнее (значимей) ассоциативное правило.

Анализ ассоциативных правил используется при решении задач из различных областей, например:

- выявление наборов товаров, которые в супермаркетах часто покупаются вместе или никогда не покупаются вместе;
- определение доли клиентов, положительно относящихся к нововведениям в их обслуживании;
- определение профиля посетителей веб-ресурса;
- определение доли случаев, в которых новое лекарство показывает опасный побочный эффект.

Методом перебора на основе множества транзакций T получать все возможные ассоциативные правила не целесообразно по следующим причинам:

1. Перебирая все возможные сочетания элементарных событий в условии и следствии ассоциативного можно получать неоправданно большое количество ассоциативных правил.

2. Ценностью обладает значимые ассоциативные правила с высокими показателями поддержки s*upp*, достоверности *conf*, лифта *lift*, левереджа *levr*, которые составляют малую долю от общего числа всех вариантов ассоциативных правил.

Чтобы сократить пространство поиска значимых ассоциативных правил на основе набора транзакций T применяется алгоритм Apriori.

В основе алгоритма лежит правило антимонотонности, которое утверждает, если предметный набор Z не является частым, то добавление некоторого нового предмета A к набору Z не делает его более частым. Другими словами, если Z не является частым набором, то и набор $Z \cup A$ также не будет являться таковым. Данное свойство значительно уменьшает пространство поиска ассоциативных правил.

Использование алгоритма Apriori для поиска значимых ассоциативных правил на основе множества транзакций состоит из 2 шагов:

1. Поиск частых предметных наборов.

2. Генерирование ассоциативных правил на основе частых предметных наборов.

3. Оценка сгенерированных ассоциативных правил с использованием описанных выше метрик.

4. Ранжирование правил по значимости с учетом их количественных показателей.

Псевдо код алгоритма Apriori для поиска частных предметных наборов представлен ниже. Входными параметрами являются множество транзакций T и порог поддержки e. Алгоритм возвращает массив частых предметных наборов (2-предметных наборов L_2, частых 3-предметных наборов L_3 и т.д.):

$Apriori(T, e)$

$L_1 \leftarrow \{l \arg e\, 1 - itemsets\}$

$k \leftarrow 2$

while $L_{k-1} \neq \varnothing$

$C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \notin a\} - \{c \mid \{s \mid s \subseteq c \wedge \mid s \mid = k-1\} \not\subset L_{k-1}\}$

for *transactions* $t \in T$

$C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}$

for *candidates* $c \in C_t$

$count[c] \leftarrow count[c] + 1$

$L_k \leftarrow \{c \mid c \in C_k \wedge count[c] \subseteq e\}$

$k \leftarrow k + 1$

return $\bigcup_k L_k$

Таким образом, поиск частых предметных наборов происходит следующим образом. В массив 1-предметных наборов L_1 попадут все события, для которых s*upp*≥*e*.

Массив 2-предметных наборов L_2 формируется следующим образом. Путем перебора всех сочетаний элементов набора L_1 формируются пары предметных наборов. Каждая полученная пара проверяется на условие s*upp*≥*e*. Пары, для которых не выполняется данное условие, исключаются из массива L_2

Массивы 3-предметных наборов и более (4-предметных, 5-предметных и т.д.) формируются следующим образом. Из элементов предметного набора находящихся на одну ступень ниже формируются пары. Пары можно сформировать из тех элементов набора, у которых все элементарные события кроме последнего идентичны. Объединив элементарные события этих пар можно получить предметный набор для текущей ступени. Получившийся предметный набор для включения его в массив L_k должен соответствовать условию s*upp*≥*e*.

Пример формирование частый предметных наборов в соответствии с алгоритмом Apriori представлен в таблице 2.1.

Таблица 2.1 – Пример формирование частый предметных наборов в соответствии с алгоритмом Apriori

Номер итерации *k* выполнение алгоритма				
k=1	*k=2*	*k=3*	*k=4*	*k=5*
Частые предметные наборы, сформированные на каждой итерации				
$L_1 = \begin{Bmatrix} \{i_2\} \\ \{i_5\} \\ \{i_6\} \\ \{i_{11}\} \end{Bmatrix}$	$L_2 = \begin{Bmatrix} \{i_2, i_5\} \\ \{i_2, i_6\} \\ \{i_2, i_{11}\} \\ \{i_5, i_6\} \\ \{i_5, i_{11}\} \end{Bmatrix}$	$L_3 = \begin{Bmatrix} \{i_2, i_5, i_6\} \\ \{i_2, i_5, i_{11}\} \\ \{i_5, i_6, i_{11}\} \end{Bmatrix}$	$L_4 = \{\{i_2, i_5, i_6, i_{11}\}\}$	$L_5 = \varnothing$
Условие для включения предметного набора в множество L_k				
$supp(L_{1,i}) \geq e$	$supp(L_{2,i}) \geq e$	$supp(L_{3,i}) \geq e$	$supp(L_{4,i}) \geq e$	-
Включение в массив $D = \bigcup_k L_k$				
Нет	Да	Да	Да	Нет

На основе всех найденных частых предметных наборов *D* генерируются ассоциативные правила.

$$D = \bigcup_k L_k \tag{2.8}$$

Для этого к каждому частому предметному набору *S* из массива D, применяется следующий алгоритм:

1) На основе набора элементарных событий *S* генерируются все возможные поднаборы *SubS*.

2) Если поднабор *SubS* является непустым поднабором *S*, то в множество ассоциативных правил *R* добавляется правило вида:

$$SubS \Rightarrow (S \setminus SubS), \tag{2.9}$$

где *S\SubS* представляет собой набор *S* без поднабора *SubS*.

Данная процедура повторяется для каждого подмножества *SubS* из *S*.

После выполнение данной процедуры на всем множестве D будет получен набор ассоциативных правил R вида:

$$R = \begin{Bmatrix} X_1 \Rightarrow Y_1 \\ X_2 \Rightarrow Y_2 \\ \dots \\ X_q \Rightarrow Y_q \end{Bmatrix}, \tag{2.10}$$

где q – количество найденных ассоциативных правил. Стоит отметить, что всегда $|R| > |D|$. Пример генерации ассоциативный правил представлен в таблице 2.2.

Таблица 2.2 – Генерации правил на основе данных таблицы 2.1

Частые предметные наборы (множество D)	Ассоциативные правила (множество R)
$L_2 = \begin{Bmatrix} \{i_2, i_5\} \\ \{i_2, i_6\} \\ \{i_2, i_{11}\} \\ \{i_5, i_6\} \\ \{i_5, i_{11}\} \end{Bmatrix}$	$\{i_2\} \Rightarrow \{i_5\}$; $\{i_5\} \Rightarrow \{i_2\}$; $\{i_2\} \Rightarrow \{i_6\}$; $\{i_6\} \Rightarrow \{i_2\}$; $\{i_2\} \Rightarrow \{i_{11}\}$; $\{i_{11}\} \Rightarrow \{i_2\}$; $\{i_5\} \Rightarrow \{i_6\}$; $\{i_6\} \Rightarrow \{i_5\}$; $\{i_5\} \Rightarrow \{i_{11}\}$; $\{i_{11}\} \Rightarrow \{i_5\}$.
$L_3 = \begin{Bmatrix} \{i_2, i_5, i_6\} \\ \{i_2, i_5, i_{11}\} \\ \{i_5, i_6, i_{11}\} \end{Bmatrix}$	$\{i_2\} \Rightarrow \{i_5, i_6\}$; $\{i_5\} \Rightarrow \{i_2, i_6\}$; $\{i_6\} \Rightarrow \{i_2, i_5\}$; $\{i_5, i_6\} \Rightarrow \{i_2\}$; $\{i_2, i_6\} \Rightarrow \{i_5\}$; $\{i_2, i_5\} \Rightarrow \{i_6\}$; $\{i_2\} \Rightarrow \{i_5, i_{11}\}$; $\{i_5\} \Rightarrow \{i_2, i_{11}\}$; $\{i_{11}\} \Rightarrow \{i_5, i_2\}$; $\{i_5, i_{11}\} \Rightarrow \{i_2\}$; $\{i_2, i_{11}\} \Rightarrow \{i_5\}$; $\{i_5, i_2\} \Rightarrow \{i_{11}\}$; $\{i_5\} \Rightarrow \{i_6, i_{11}\}$; $\{i_6\} \Rightarrow \{i_5, i_{11}\}$; $\{i_{11}\} \Rightarrow \{i_6, i_5\}$; $\{i_6, i_{11}\} \Rightarrow \{i_5\}$; $\{i_5, i_{11}\} \Rightarrow \{i_6\}$; $\{i_6, i_5\} \Rightarrow \{i_{11}\}$;
$L_4 = \{\{i_2, i_5, i_6, i_{11}\}\}$	$\{i_2\} \Rightarrow \{i_5, i_6, i_{11}\}$; $\{i_5\} \Rightarrow \{i_2, i_6, i_{11}\}$; $\{i_6\} \Rightarrow \{i_5, i_2, i_{11}\}$; $\{i_{11}\} \Rightarrow \{i_5, i_6, i_2\}$; $\{i_2, i_5\} \Rightarrow \{i_6, i_{11}\}$; $\{i_2, i_6\} \Rightarrow \{i_5, i_{11}\}$; $\{i_2, i_{11}\} \Rightarrow \{i_6, i_5\}$; $\{i_5, i_6\} \Rightarrow \{i_2, i_{11}\}$; $\{i_5, i_{11}\} \Rightarrow \{i_2, i_6\}$; $\{i_6, i_{11}\} \Rightarrow \{i_2, i_5\}$; $\{i_5, i_6, i_{11}\} \Rightarrow \{i_2\}$; $\{i_2, i_6, i_{11}\} \Rightarrow \{i_5\}$; $\{i_5, i_2, i_{11}\} \Rightarrow \{i_6\}$; $\{i_5, i_6, i_2\} \Rightarrow \{i_{11}\}$.

После того, как найдено множество ассоциативных правил R, его исследуют с использованием показателей поддержки s*upp*, достоверности *conf*, лифта *lift*, левереджа *lev*.

Чаще всего данное исследование заключается в составлении рейтинговой таблицы ассоциативных правил путем их сортировки в порядке убывания значимости. Для этого обычно используют обобщенный показатель, рассчитываемый как произведение поддержки правила s*upp*(r_i) на достоверность *conf*(r_i).

Графическая интерпретация этапа составления рейтинговой таблицы ассоциативных правил представлена на рисунке 2.2.

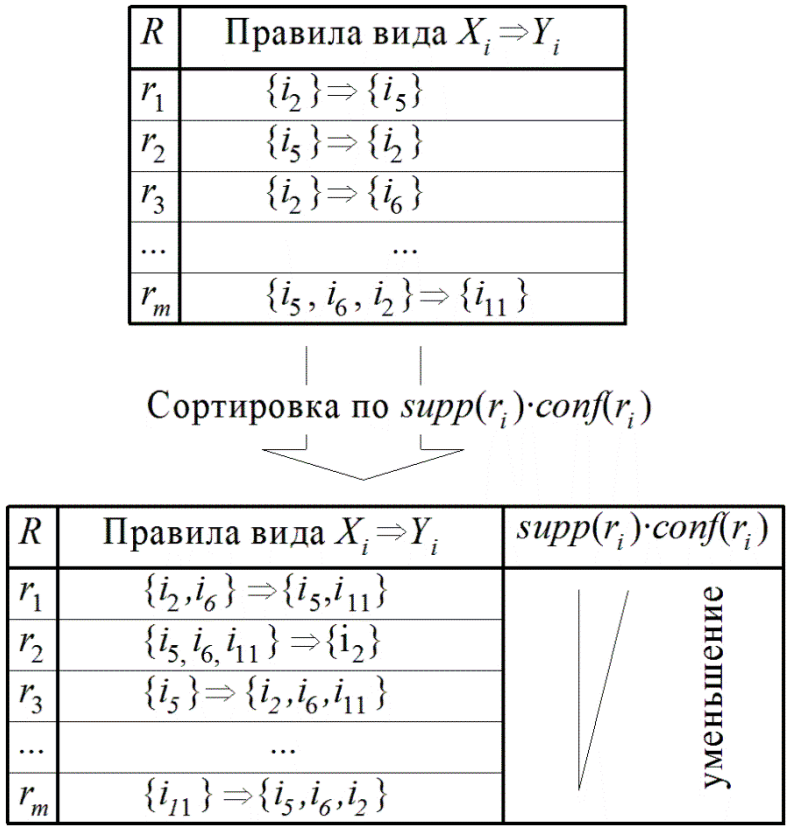


Рисунок 2.2 – Сортировка правил по показателю *supp*(r_i)· *conf*(r_i)

2.2 Синтез классификатора на основе модифицирования ассоциативных правил

Разработанный алгоритм синтеза классификатора, состоит из следующих этапов:

1. На основе анализа меток класса и областей определения входных параметров формируется множество элементарных событий.

2. Преобразование каждого элемента обучающей выборки в отдельную транзакцию.

3. Поиск с помощью алгоритма Apriori ассоциативных правил, удовлетворяющих минимальным значениям поддержки *Supp* и достоверности *Conf*.

4. Отбор из полученных ассоциативных правил тех, у которых в следствии (в правой части) присутствует метка класса и модифицирование правил.

5. Объединение отобранных правил в классификатор.

Рассмотрим каждый этап алгоритма подробней. На первом этапе необходимо выполнить проанализировать выборку данных для формирования множество возможных элементарных событий. Для этого предложено область определения каждого входного числового параметра $P_1, P_2, ..., P_9$ разбить на отрезки одинаковой длины. Тогда элементарным событием будет являться факт попадания определенного значения рассматриваемого параметра в определенную область.

Количество отрезков, на которые разбивается область определения входных параметров, следует выбирать опытным путем. В нашем случае приемлемым по точности работы (конечного классификатора) является разбиение области определения каждого входного параметра на 10 равных частей.

Возможным значениям выходного параметра (меткам класса) также необходимо сопоставить элементарные события. Для простоты восприятия событию, связанному с получением определенного класса, будет дано такое же обозначение, как у метки класса. Аналогичный подход применяется ко всем категориальным признакам.

Таким образом, на данном этапе формируется множество элементарных событий и формулируются условия их активации.

Графически выполнение первого этапа представлено на рисунке 2.3.

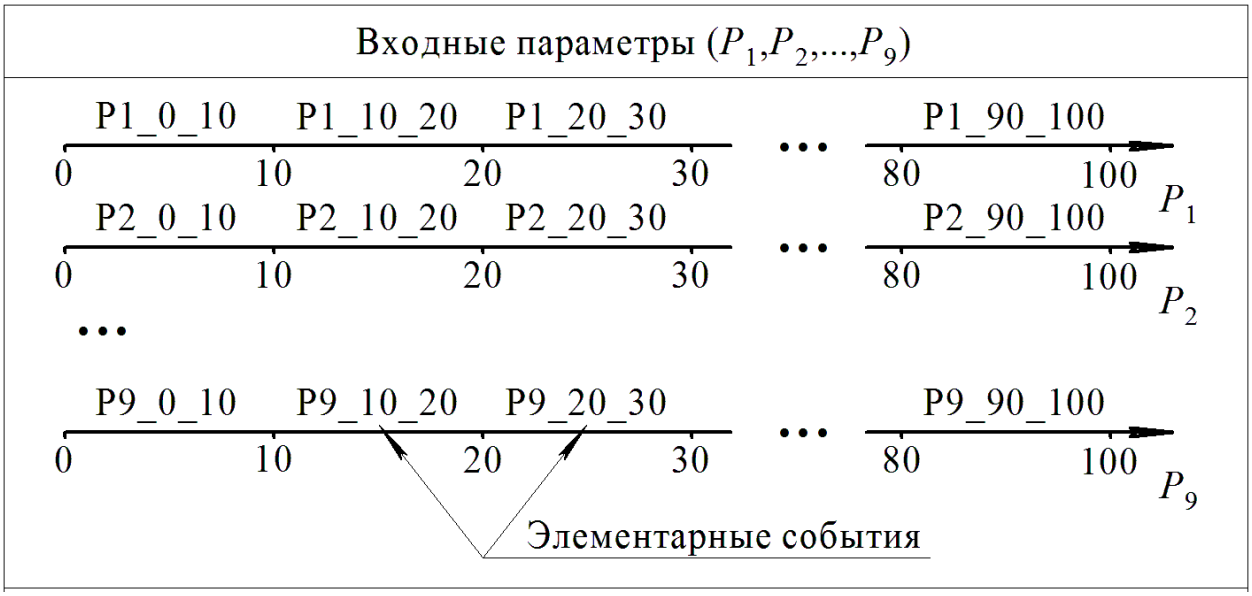


Рисунок 2.3 – Формирование множества элементарных событий для числовых признаков

На втором этапе алгоритма осуществляется преобразование каждого элемента обучающей выборки (с учетом сформулированного множества событий) в отдельную транзакцию. Т.е. результатом выполнения данного этапа будет являться множество транзакций, подходящих для выполнения аффинитивного анализа. Количество транзакций равно количеству записей в обучающей выборке.

Пример формирование транзакции на основе одной записи обучающей выборки представлен на рисунке 2.4.

На третьем этапе с использованием алгоритма Apriori осуществляется поиск ассоциативных правил, удовлетворяющих минимальным значениям поддержки *Supp* и достоверности *Conf*.

№	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	C
					•••					
3	93,18	58,79	93,18	100	65,87	51,34	51,34	8,49	0	C1
					•••					

Множество транзакций

T	Транзакции
	•••
t_3	{P1_90_100, P2_50_60, P3_90_100, P4_90_100, P5_60_70, P6_50_60 P7_50_60, P8_0_10, P9_0_10, C1}
	•••

Рисунок 2.4 – Пример формирования транзакции на примере одной записи из обучающей выборки

Найденные ассоциативные правила показывают зависимость возникновения одних событий от других. Полученный на третьем этапе набор ассоциативных правил сам по себе не является классификатором. Из полученного набора правил необходимо вычленить только те правила, в следствии которых есть события связанные с метками классов (C1, C2).

Поэтому на четвертом этапе работы алгоритма осуществляется отбор из полученных ассоциативных правил тех, у которых в следствии присутствует метка класса.

Те ассоциативные правила, которые не содержат в следствии события связанного с меткой класса исключаются. Если ассоциативные правила в

следствии содержат помимо событий связанных с меткой класса еще и события связанные со значениями входных параметров, то у таких правил модифицируется следствие. Остальные правила оставляются без изменений.

Примеры обработки найденных ассоциативных правил показаны на рисунке 2.5.

Пример ассоциативного правила	Результат
{p2=P2_90_100, p7=P7_0_10}=> {p8=P8_0_10}	исключение правила, т.к. в следствии отсутствует указание на метку класса
{p2=P2_90_100, p6=P6_20_30}=> {p8=P8_0_10, C=C1}	модифицирование правила {p2=P2_90_100, p6=P6_20_30}=> {C=C1}
{p5=P5_60_70}=>{C=C2}	сохранение без изменений {p5=P5_60_70}=>{C=C2}

Рисунок 2.5 – Примеры обработки ассоциативных правил

После такой обработки все ассоциативные правила в следствии содержат единственное событие – ссылку на метку класса. Это дает на возможность объединить такие правила в классификатор. На пятом этапе работы полученные правило объединяются в классификатор.

Полученный классификатор работает следующим образом:

1. На вход классификатора подается вектор входных параметров **P** исследуемого объекта.

2. Затем для каждой метки класса рассчитывается количество сработавших ассоциативных правил. Правило срабатывает, если имеют место быть те события, которые находятся в условии ассоциативного правила.

3. Исследуемому событию присевается та метка класса, для которой сработало наибольшее количество правил.

Пример классификации объекта с использованием предложенного подхода представлен на рисунке 2.6.

1. Подача вектора входных значений

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	C
95,51	100	92,13	75,18	45,81	36,95	5,3	5,3	0	?

2. Превод числовых значений в множество событий

{P1_90_100; P2_90_100; P3_90_100; P4_70_80; P5_40_50; P6_30_40; P7_0_10; P8_0_10; P9_0_10}

3. Подсчет сработовших правил классификатора для каждого класса (всего в классификаторе 27 правил)

C1	C2
{P2_90_100}=>{C1} {P7_0_10}=>{C1} {P5_40_50}=>{C1} {P2_90_100, P5_40_50}=>{C1}	{P3_90_100}=>{C2} {P2_90_100}=>{C2} {P2_90_100, P3_90_100}=>{C2}
4 ссылки на {C1}	3 ссылки на {C2}

4. Вывод о классе исследуемого объекта

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	C
95,51	100	92,13	75,18	45,81	36,95	5,3	5,3	0	**C1**

Рисунок 2.6 – Алгоритм классификации объектов

Таким образом, был разработан алгоритм построения классификатора на основе аффинитивного анализа исходных данных. Показан принцип работы классификатора на реальных данных

Выводы по разделу

Разработана технология использования алгоритма apriori для построения классификатора данных. Предложенный подход предполагает преобразование исходных данных во множество транзакций, состоящих из элементарных событий. Элементарные события, связанные с категориальными параметрами и меткой класса, генерируются на основе значений категорий. Элементарные событий связанные с числовыми признаками генерируются на основе факта принадлежности значений одному из отрезков числовой оси (рисунок 2.3). Получившееся множество транзакций анализируется с помощью алгоритма apriori для получения множества правил, связывающих элементарные события. Полученные ассоциативные правила фильтруются и модифицируются таким образом, чтобы в их правой части содержалось указание на значение метки класса. Модифицированные правила образуют классификатор. При классификации объекта каждое сработавшее правило голосует за свой класс. Классификатор присваивает объекту тут класс, который набрал большее количество голосов.

3 Проведение тестирования предложенных подходов

3.1 Программная реализация предложенных подходов

Программная реализация предложенных подходов разработана в среде Google Colab. В разработанном программном обеспечении использовались следующие модули (рисунок 3.1)**:**

- OS – модуль, который предоставляет множество функций для работы с операционной системой, причём их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми.
- Operator – модуль, который экспортирует набор эффективных функций, которые соответствуют внутренним операторам Python.
- Numpy – модуль содержащий большую библиотеку математических функций для работы с большими многомерными массивами и матрицами.
- Apyori – модуль простой реализации алгоритма Apriori.
- Matplotlib.pyplot – модуль по визуализации графиков. Представляет собой набор функций, которые заставляют matplotlib работать как MATLAB.
- Matplotlib.ticker – модуль содержащий несколько локаторов, использующие различные алгоритмы расположения меток на графике.
- Sklearn.model_selection.train_test_split – модуль, который разбивает массивы или матрицы на случайные обучающие и тестовые подмножества.
- Pandas.api.types – модуль, содержащий некоторые общедоступные функции, связанные с типами данных в Pandas.
- Sklearn.metrics.accuracy_score – модуль классификационной оценки точности.

```
import os
import operator
import numpy as np
import pandas as pd
from apyori import apriori
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from sklearn.model_selection import train_test_split
from pandas.api.types import is_numeric_dtype, is_bool_dtype
from sklearn.metrics import accuracy_score
```

Рисунок 3.1 – Импорт необходимых библиотек

Опишем работу классификатора на примере задачи классификации вин. Обучающую выборку для классификации возьмем из открытого репозитория (https://archive.ics.uci.edu/ml/datasets/Wine).

Ниже представлен фрагмент кода на языке Python, выполняющий чтение обучающей выборки из файла в виде pandas.DataFrame (рисунок 3.2).

Программа поддерживает контроль количества признаков в выборке, которое задается в attr_num и поддерживает динамическое редактирование с помощью Google Colab Forms. В начале из таблицы выбирается столбец с помощью операции среза данных, содержащий метки классов. От остальных столбцов отбирается нужное количество столбцов с помощью той же операции среза данных.

На выходе получаем выборку, содержащую столбец меток класса в последнем столбце и заданное количество параметров (рисунок 3.3).

```
wineFile = os.path.join(ROOT_DIR, 'data/wine.csv')
wine = pd.read_csv(wineFile)
attr_num = 5 #@param {type:"slider", min:1, max:13, step:1}
columns = wine.drop(['Class'], axis=1).iloc[:,:attr_num]
columns = columns.columns.tolist()
wine = wine[columns + ['Class']]

# Характеристики
params = wine[wine.columns.difference(['Class'])].columns.tolist();
print('Количество параметров :', len(params))

wine = wine[params + ['Class']]
wine
```

Рисунок 3.2 – Загрузка выборки данных

```
Количество параметров : 5
```

	Alcalimity_of_ash	Alcohol	Ash	Color_intensity	Flavanoids	Class
0	15.6	14.23	2.43	5.64	3.06	1
1	11.2	13.20	2.14	4.38	2.76	1
2	18.6	13.16	2.67	5.68	3.24	1
3	16.8	14.37	2.50	7.80	3.49	1
4	21.0	13.24	2.87	4.32	2.69	1
...	...	...	...	...	...	...
173	20.5	13.71	2.45	7.70	0.61	3
174	23.0	13.40	2.48	7.30	0.75	3
175	20.0	13.27	2.26	10.20	0.69	3
176	20.0	13.17	2.37	9.30	0.68	3
177	24.5	14.13	2.74	9.20	0.76	3

Рисунок 3.3 – Полученная выборка

Выборка загружается из файла wine.csv, расположенном на облачном хранилище. Данная выборка содержит 5 числовых характеристик вина и 1 метку класса, а именно:

1) Щелочность золы в вине (Alcalimity_of_ash).
2) Алкоголь (Alcohol).
3) Содержание золы (Ash).
4) Интенсивность цвета (Color_intensity).
5) Флавоноиды (Flavanoids).
6) Класс (Ckass).

После выполнения блока кода создаётся дата-фрейм, где каждой колонке соответствует своя характеристика. Количество используемых параметров можно динамически регулировать через пользовательский интерфейс.

Рисунок 3.4 – Интерфейс пользователя для изменения количества параметров выборки из примера

Для работы с числовыми и категориальными признаками опишем функции преобразования значений в элементарные события.

Если рассматриваемый признак является числовым, то сначала все значения сортируются в порядке возрастания, затем по числу групп находится максимальная длина одной группы, и создаются массивы содержащие признаки, чьё общее число определяется максимальной длиной. Количество групп для разбиения числовых параметров можно также регулировать через интерфейс (по умолчанию используется 5 групп). Функция numericByGroups возвращает двумерный список, содержащий

информацию о сформированных группах в формате [<название группы>, (<граница отрезка a>, <граница отрезка b>)] (рисунок 3.6).


```
group_num = 5 #@param {type:"slider", min:2, max:15, step:1}

def numericByGroups(name, vals, groupNum = 5):

    # Сортировка ряда
    vals = sorted(vals)

    # Разбиение ряда на отрезки
    h = (vals[-1] - vals[0]) / groupNum
    ranges = [(vals[0] + h*i, vals[0] + h*(i+1)) for i in range(groupNum)]

    # Названия отрезков для поиска
    return [[name + '_' +
             '_'.join([('%d' if i.is_integer() else '%.2f') % i for i in r]),
             r] for r in ranges]
```

Рисунок 3.6 – Разбиение ряда чисел на отрезки

Ниже представлен пример разбиения числового параметра из обучающей выборки на 5 групп. При определении попадания числа в ту или иную группу выполняется проверка условия $a \leq x < b$, где (a и b - границы отрезка). Максимальное число ряда попадает в последнюю группу, т.е. для последней группы неравенство нестрогое. Результаты формирования 5 групп для первого числового признака (Alcalimity_of_ash) (рисунок 3.7).

```
column = wine.columns.tolist()[0]
numericByGroups(column, wine[column], group_num)
```

```
[['Alcalimity_of_ash_10.60_14.48', (10.6, 14.48)],
 ['Alcalimity_of_ash_14.48_18.36', (14.48, 18.36)],
 ['Alcalimity_of_ash_18.36_22.24', (18.36, 22.240000000000002)],
 ['Alcalimity_of_ash_22.24_26.12', (22.240000000000002, 26.119999999999997)],
 ['Alcalimity_of_ash_26.12_30', (26.119999999999997, 30.0)]]
```

Рисунок 3.7 – Результат разбиения числовой оси признака Alcalimity_of_ash

Когда признак категориальный, происходит только отбор уникальных значений и создается столько групп, сколько уникальных значений в выборке. Результат также записывается в двумерный список, только в последней ячейке для каждого правила будет хранится не пара границ отрезка, а само значение категориального признака. Важно заметить, что для самого алгоритма нет необходимости разбивать на группы категориальные признаки, однако это выполняется для единообразного интерфейса для признаков обоих типов (рисунок 3.8).

```
def categoricalByGroups(name, vals):

    # Отбор уникальных значений
    vals = set(vals)

    # Разбиение на группы
    return [[name + '_' + str(v), v] for v in vals]
```

Рисунок 3.8 – Разбиение категориальных признаков

Ниже приведен результат разбиения категориального признака класса на 3 группы (рисунок 3.9).

```
categoricalByGroups('Class', wine['Class'])
```

```
[['Class_1', 1], ['Class_2', 2], ['Class_3', 3]]
```

Рисунок 3.9 – Результат представления категориального признака

Далее опишем функцию, выполняющую преобразование обучающей выборки в список транзакций (переведение входных данных в данные, понятные алгоритму Apriori). Транзакцией является набор элементов из разных групп. Целью работы алгоритма Apriori является определение зависимости между нахождениями одних элементов в одной транзакции с другими. Например, для задачи анализа корзины потребителя это будет

зависимость между покупкой одного товара вместе с другим. Для формирования списка транзакций требуется выполнить несколько шагов:

1) Определение типа характеристики;
2) Разбиение массива значений характеристик на группы;
3) Формирование списка транзакций.

Метки класса считаются категориальными признаками, поэтому для них сразу присваивается данных тип. Для остальных параметров выполняется проверка типа данных, если это строковый или логический тип - данная характеристика является категориальной, иначе - числовой.

Для каждого столбца обучающей выборки выполняется следующая последовательность действий:

1. Определяется тип признака: числовой или категориальный (метка класса также считается категориальной). Для определения типа используются функции из библиотеки pandas.api_types is_numeric_dtype - выполняет проверку столбца на соответствие числовому типу, is_bool_dtype - проверка на логический тип данных (считается категориальным признаком). Таким образом, если текущий столбец является меткой класса, не числовым типом или логическим типом, то это категориальный признак, в противном случае – числовой (рисунок 3.10).

```
if param == className:
    ptype = 'categorical'
else:
    dtype = ts[param]
    ptype = ('categorical'
             if not is_numeric_dtype(dtype)
             or is_bool_dtype(dtype)
             else 'numeric')
```

Рисунок 3.10 – Определение типа параметра

2. В зависимости от присвоенного типа выполняется соответствующее разбиение на группы (числовое или категориальное). В результате для каждого признака выборки формируется следующий словарь dict(): {type: <тип параметра>, group: <двумерный список групп, сформированный с помощью функций перевода признака в список групп>}. Для каждой группы из списка хранится информация о названии группы в виде строки, а также о границах отрезка (для числовых данных) в виде объекта tuple или значение категориального признака (рисунок 3.11).

```
groups = []
if ptype == 'categorical':
    groups = categoricalByGroups(param, ts[param])
else:
    groups = numericByGroups(param, ts[param])
elements[param] = {'type': ptype, 'groups': groups}
```

Рисунок 3.11 – Разбиение на группы столбцов в зависимости от типа

3. Для каждой записи из обучающей выборки выполняется определение принадлежности элемента к одной из групп и результаты записываются в виде списка транзакций. Выполняется итеративный проход по записям в обучающей выборке. Если столбец хранит категориальные признаки, то в сформированном словаре элементов находится группа для данного элемента и название данной группы записывается в транзакцию. Если столбец хранит числовой признак, выполняется определение принадлежности элемента какому-то из отрезков. Если найти отрезок не удалось, это означает, что элемент является максимальным в столбце и его необходимо отнести к последней группе (рисунок 3.12).

Сформированная транзакция представляет собой список строк с названиями групп, содержащихся в данной транзакции. Каждая транзакция сохраняется в массив транзакций. На выходе функции формируется массив

транзакций и словарь (dict) элементов транзакций для последующего перевода записи в транзакцию при классификации. Массив транзакций хранит строки с названиями предметов. Словарь элементов хранит информацию о каждом элементе в формате, предтавленном на рисунке 3.13.

```
transactions = []
for _, row in ts.iterrows():
    transaction = []

    for param, value in row.items():
        p = elements[param]
        name = None
        if p['type'] == 'categorical':
            name = next((x[0] for x in p['groups']
                         if x[1] == value), None)
        else:
            name = next((x[0] for x in p['groups']
                         if x[1][0] <= value < x[1][1]), None)
            # Если это максимальный элемент ->
            # добавить его в последий отрезок
            if name == None:
                name = p['groups'][-1][0]

        transaction.append(name)

    transactions.append(transaction)

return transactions, elements
```

Рисунок 3.12 – Формирование списка транзакций

```
param_name: {
  type: 'categorical' or 'numeric',
  groups: [
      ['param_name_a1_b1', (a1, b1)],
      ['param_name_a2_b2', (a2, b2)],
      ...
  ]
}
```

Рисунок 3.13 – Формат хранения данных об элементе

Здесь param_name соответствует столбцу обучающей выборки. В поле type хранится тип параметра, в поле groups - сформированные группы для каждого параметра (название элемента в списке транзакций и его значение для категориальных признаков или границы отрезка - для числовых).

В функцию tsToTransactions, помимо выборки, также передается название столбца меток класса для корректного определения категориального признака, так как иначе, данный столбец будет отнесен к числовому признаку, что противоречит алгоритму программы классификатора (рисунок 3.14).

```
className = 'Class'
transactions, elements = tsToTransactions(wine, className)

namesN = len([p for p in elements.values() for g in p['groups']])
print('Количество элементов =', namesN)

df_wine = pd.DataFrame.from_records(transactions)
df_wine.index += 1
df_wine
```

Рисунок 3.14 – Выполнение преобразования обучающей выборки в список транзакций

На следующем скриншоте представлены результаты преобразования обучающей выборки вина в список транзакций. Для выборки из 178 записей было сформировано 178 транзакций (рисунок 3.15).

Количество элементов = 28

	0	1	2	3	4	5
1	Alcalimity_of_ash_14.48_18.36	Alcohol_14.07_14.83	Ash_2.11_2.48	Color_intensity_3.62_5.97	Flavanoids_2.24_3.18	Class_1
2	Alcalimity_of_ash_10.60_14.48	Alcohol_12.55_13.31	Ash_2.11_2.48	Color_intensity_3.62_5.97	Flavanoids_2.24_3.18	Class_1
3	Alcalimity_of_ash_18.36_22.24	Alcohol_12.55_13.31	Ash_2.48_2.86	Color_intensity_3.62_5.97	Flavanoids_3.18_4.13	Class_1
4	Alcalimity_of_ash_14.48_18.36	Alcohol_14.07_14.83	Ash_2.48_2.86	Color_intensity_5.97_8.31	Flavanoids_3.18_4.13	Class_1
5	Alcalimity_of_ash_18.36_22.24	Alcohol_12.55_13.31	Ash_2.86_3.23	Color_intensity_3.62_5.97	Flavanoids_2.24_3.18	Class_1
...	...	...	...	...	...	...
174	Alcalimity_of_ash_18.36_22.24	Alcohol_13.31_14.07	Ash_2.11_2.48	Color_intensity_5.97_8.31	Flavanoids_0.34_1.29	Class_3
175	Alcalimity_of_ash_22.24_26.12	Alcohol_13.31_14.07	Ash_2.11_2.48	Color_intensity_5.97_8.31	Flavanoids_0.34_1.29	Class_3
176	Alcalimity_of_ash_18.36_22.24	Alcohol_12.55_13.31	Ash_2.11_2.48	Color_intensity_8.31_10.66	Flavanoids_0.34_1.29	Class_3
177	Alcalimity_of_ash_18.36_22.24	Alcohol_12.55_13.31	Ash_2.11_2.48	Color_intensity_8.31_10.66	Flavanoids_0.34_1.29	Class_3
178	Alcalimity_of_ash_22.24_26.12	Alcohol_14.07_14.83	Ash_2.48_2.86	Color_intensity_8.31_10.66	Flavanoids_0.34_1.29	Class_3

178 rows × 6 columns

Рисунок 3.15 – Результат выполнения

После нахождения списка транзакций можно найти список правил (рисунок 3.16). Формируемые правила хранятся в виде дата-фрейма. Каждое сформированное правило проходит модификацию и либо сохраняется, либо удаляется из списка правил. Для формирования списка правил используется алгоритм аффинитивного анализа Apriori, описанный в открытой библиотеки для Python Apyori (https://github.com/ymoch/apyori).

```
def genRules(transactions, elements, className):

    # Генерация списка правил с помощью apriori
    association_rules = list(apriori(transactions, min_support=0.0045,
                                     min_confidence=0.2, min_lift=3, min_length=2))

    # Для хранения списка правил
    df_rules = pd.DataFrame(columns=['RuleL', 'RuleR', 'Support',
                                     'Confidence', 'SxC', 'Lift', 'Leverage'])
```

Рисунок 3.16 – Начало функции

В функцию apriori передается список транзакций, сформированный на предыдущем шаге, а также задаются некоторые коэффициенты для повышения точности и получения требуемых результатов. Показатели min_support, min_confidence и min_left соответствуют ограничению на характеристики правила, описанные в начале главы. Показатель min_lenght соответствует минимальному количеству элементов в причине правила.

Каждое правило, получаемое в результате генерации ассоциативных правил, содержит левую часть RuleL - причина правила, правую часть RuleR - следствие правила, т.е. элементы, которые появляются в транзакции, если выполняется причина правила. Также вычисляются основные показатели правила - поддержка, доверие, произведение поддержки на доверие SxC, Lift и Leverage.

Цель процесса фильтрации и модификации правил заключается в преобразовании списка правил к виду, когда в правой части правил содержится только метка класса. Если сформированное правило содержит пустое следствие, такое правило сразу удаляется и осуществляется переход к следующему в списке (рисунок 3.17).

```
for item in association_rules:

    # Условие и следствие
    RuleL = list(item[2][0][0])
    RuleR = list(item[2][0][1])

    # Если следствие пустое -> следующее правило
    if len(RuleR) == 0:
        continue
```

Рисунок 3.17 – Проверка на пустое следствие

Далее необходимо удалить элементы из следствия (правой части), не соответствующие метке класса. Если после модификации правила, в правой части оказалось пустое множество, такое правило тоже отметается. После модификации правила в дата-фрейм списка правил дописываются значения основных характеристик правила (рисунок 3.18).

```
# Удаление элементов, не указывающих на класс из следствия
noClass = []
for ir, r in enumerate(RuleR):
    param = next((key
        for key in elements
        for g in elements[key]['groups']
        if g[0] == r
    ), None)
    if not param == className:
        noClass.append(ir)
if len(noClass):
    RuleR = np.delete(RuleR, noClass)

# Если следствие пустое -> следующее правило
if len(RuleR) == 0:
    continue

# Вычисление характеристик
Support = item[1]
Confidence = item[2][0][2]
Lift = item[2][0][3]
SupportL = Support / Confidence
SupportR = Confidence / Lift
Leverage = Support - SupportL * SupportR
```

Рисунок 3.18 – Приведение списка правил к нужному формату для классификации

После выполнения процедуры модификации правила могут появиться дублирующие правила, если это так, среди повторяющихся правил отбирается правило с максимальным показателем произведения доверия на поддержку (SxC), остальные правила удаляются из списка. Для начала

отбираются индексы правил, являющихся дубликатами текущего правила. Для этого выполняется проход по всем сформированным до текущего правила правил и сравнение левой и правой частей. После отбора индексов выполняется сравнение показателя SxC для текущего правила и для всех его дубликатов (дубликатов не будет больше одного при проходе по списку правил сверху вниз). Правило с максимальным показателем SxC сохраняется, остальные удаляются командой pandas.drop() (рисунок 3.19).

```
        # Поиск повторений
        dindexes = [ir for ir, row in df_rules.iterrows()
                    if row['RuleL'] == RuleL
                    and row['RuleR'] == RuleR]

        # Найдено повторение
        if (len(dindexes)):

            # Текущее значение SxC больше -> заменить старое правило
            if df_rules.loc[dindexes[0]]['SxC'] < Support*Confidence:
                df_rules.drop(dindexes)
            # Пропустить текущее правило
            else:
                continue

        # Сохранить текущее правило в списке
        df_rules.loc[df_rules.shape[0] + 1] = [RuleL, RuleR, Support,
                                               Confidence, Support*Confidence,
                                               Lift, Leverage]
    return df_rules
```

Рисунок 3.19 – Приведение списка правил к нужному формату для классификации

В результате для нашей обучающей выборки был сформирован следующий набор ассоциативный правил, содержащих в правой части только метки класса. Всего 208 правил для обучающей выборки с вином (рисунок 3.20).

```
df_rules = genRules(transactions, elements, className)
df_rules
```

	RuleL	RuleR	Support	Confidence	SxC	Lift	Leverage
1	[Proline_1399.60_1680]	[Class_1]	0.03750	1.000000	0.037500	3.076923	0.025312
2	[Proanthocyanins_2.31_2.95]	[Class_1]	0.01875	0.230769	0.004327	3.356643	0.013164
3	[Proline_1119.20_1399.60]	[Class_1]	0.02500	0.222222	0.005556	3.232323	0.017266
4	[Total_phenols_2.72_3.30]	[Class_1]	0.05000	0.210526	0.010526	3.062201	0.033672
5	[Proanthocyanins_2.31_2.95, Nonflavanoid_pheno...	[Class_1]	0.01875	1.000000	0.018750	3.076923	0.012656
...	...	...	...	...	...	...	...
204	[Proline_278_558.40, Nonflavanoid_phenols_0.24...	[Class_2]	0.00625	0.333333	0.002083	3.809524	0.004609
205	[Proline_558.40_838.80, OD315_of_diluted_wines...	[Class_2]	0.00625	0.200000	0.001250	4.000000	0.004688
206	[Total_phenols_1.56_2.14, Nonflavanoid_phenols...	[Class_2]	0.00625	0.500000	0.003125	4.444444	0.004844
207	[OD315_of_diluted_wines_1.82_2.36, Nonflavanoi...	[Class_2]	0.00625	0.250000	0.001563	4.444444	0.004844
208	[Proanthocyanins_0.41_1.04, Nonflavanoid_pheno...	[Class_3]	0.00625	0.250000	0.001563	3.636364	0.004531

208 rows × 7 columns

Рисунок 3.20 – Вывод полученных значений

При выполнении классификации необходимо производить перевод записи из выборки в транзакцию. Опишем функцию для выполнения этого действия. В функцию передается запись из выборки для классификации и словарь элементов, сформированный на этапе генерации списка транзакций. Процедура идентична процедуре, выполняемой при генерации списка транзакций, поэтому повторно объясняться не будет. Ее код представлен на рисунке 3.21.

```
def recordToTransaction(record, elements):
    transaction = []
    for key, val in record.items():
        element = elements[key]
        name = None

        # Определение названия элемента в транзакции
        if element['type'] == 'categorical':
            name = next((x[0] for x in element['groups'] if x[1] == val),
                        None)
        else:
            name = next((x[0] for x in element['groups']
                        if x[1][0] <= val < x[1][1]), None)
            # Если это максимальный элемент ->
            # добавить его в последий отрезок
            if name == None:
                name = element['groups'][-1][0]

        # Добавление элемента в транзакцию
        transaction.append(name)

    return transaction
```

Рисунок 3.21 – Функция преобразования записи в транзакцию

Пример перевода первой записи выборки в транзакцию представлен на рисунке 3.22

```
24 transaction = recordToTransaction(wine.iloc[0].drop(labels=[className])
25                                   elements)
26 transaction
```

```
['Alcalimity_of_ash_14.48_18.36',
 'Alcohol_14.07_14.83',
 'Ash_2.11_2.48',
 'Color_intensity_3.62_5.97',
 'Flavanoids_2.24_3.18']
```

Рисунок 3.22 – Вывод полученных значений

После перевода в транзакцию необходимо отобрать правила из списка, подходящих для данной транзакции. Для этого выполняется отбор правил, включающих элементы транзакции в левой части правила. Алгоритм отбора правил следующий: выполняется проход по всем сгенерированным правилам и определяется множество пересечения левой части правила и текущей транзакции. Если множество не пустое и содержит все элементы из левой части правила, т.е. в левой части не содержится других элементов, не принадлежащих транзакции, такое правило отбирается (рисунок 3.23).

```
def rulesForTransaction(transaction, df_rules):
    rules = pd.DataFrame(columns=df_rules.columns)
    for _, rule in df_rules.iterrows():
        # Пересечение двух множеств
        crossing = list(set(rule['RuleL']) & set(transaction))

        # Пересечение совпадает с условием правила
        if len(crossing) == len(rule['RuleL']):
            rules.loc[rules.shape[0] + 1] = rule
    return rules
```

Рисунок 3.23 – Функция поиска правил для переданной транзакции

Для первой записи из выборки алгоритм подобрал 12 правил, на основе которых и будет приниматься решение об определении метки класса (рисунок 3.24).

```
rules = rulesForTransaction(transaction, df_rules)
rules
```

	RuleL	RuleR	Support	Confidence	SxC	Lift	Leverage
1	[Proline_838.80_1119.20]	[Class_1]	0.06875	0.392857	0.027009	3.142857	0.046875
2	[OD315_of_diluted_wines_2.91_3.45, Proline_838...	[Class_1]	0.01250	0.250000	0.003125	3.636364	0.009063
3	[Nonflavanoid_phenols_0.13_0.24, Total_phenols...	[Class_1]	0.01250	0.666667	0.008333	4.637681	0.009805
4	[Nonflavanoid_phenols_0.13_0.24, Total_phenols...	[Class_1]	0.00625	1.000000	0.006250	3.076923	0.004219
5	[OD315_of_diluted_wines_2.91_3.45, Proline_838...	[Class_1]	0.03125	0.625000	0.019531	3.333333	0.021875
6	[Total_phenols_2.14_2.72, OD315_of_diluted_win...	[Class_1]	0.01250	1.000000	0.012500	3.076923	0.008438
7	[Nonflavanoid_phenols_0.13_0.24, Proanthocyani...	[Class_2]	0.01250	0.500000	0.006250	3.200000	0.008594
8	[Nonflavanoid_phenols_0.13_0.24, Proanthocyani...	[Class_1]	0.00625	0.250000	0.001563	5.714286	0.005156
9	[OD315_of_diluted_wines_2.91_3.45, Proline_838...	[Class_1]	0.01875	0.375000	0.007031	3.529412	0.013437
10	[OD315_of_diluted_wines_2.91_3.45, Proline_838...	[Class_1]	0.01875	0.375000	0.007031	4.615385	0.014687
11	[OD315_of_diluted_wines_2.91_3.45, Proanthocya...	[Class_2]	0.02500	0.222222	0.005556	3.555556	0.017969
12	[OD315_of_diluted_wines_2.91_3.45, Proline_838...	[Class_1]	0.01875	0.375000	0.007031	7.500000	0.016250

Рисунок 3.24 – Вывод полученных значений

Для определения метки класса выполняется взвешенное голосование по найденным правилам. Каждое правило умножается на показатель SxC и результаты складываются по каждому классу. Класс, набравший большую сумму, выбирается в качестве результата классификации модели. Если для записи не было найдено ассоциативных правил, что может быть при обучении модели на небольшой выборке, то в качестве метки класса будет выбран первый класс. Выполняется проход по всем правилам, отобранным для текущей транзакции, и для каждого класса вычисляется сумма правил, умноженных на показатель SxC. После этого выполняется определение класса с максимальной суммой, либо возвращается метка первого класса, если правил не найдено для транзакции и метод голосования применить не удастся.

Функция выполнения взвешенного голосования за метку класса представлена на рисунке 3.25.

```
# Взвешенное голосование по величине SxC
def vote(rules):
    classes = {}
    for _, rule in rules.iterrows():
        clazz = rule['RuleR'][0]
        wFactor = rule['SxC']
        if not (clazz in classes):
            classes[clazz] = wFactor
        else:
            classes[clazz] += wFactor
    if len(classes) == 0:
        return elements[className]['groups'][0][0]
    else:
        return max(classes.items(), key=operator.itemgetter(1))[0]
```

Рисунок 3.25 – Взвешенное голосование по величине SxC

Для первой записи из выборки был определен первый класс, что соответствует его реальной метке класса (рисунок 3.26).

```
15 # Определение метки класса
16 classElem = vote(rules)
17 classN = next((x[1] for x in elements[className]['groups']
18                 if x[0] == classElem), None)
19
20 print('Исходный класс:', int(wine.iloc[0][className]))
21 print('Результат классификации:', classN)
```

```
Исходный класс: 1
Результат классификации: 1
```

Рисунок 3.26 – Вывод полученного значения классификации

Описанный алгоритм классификатора был оформлен в виде класса AprioriClassifier с описанием функций fit (обучение модели: формирование списка транзакций, генерация списка правил, фильтрация правил) и predict (классификация: перевод правила в транзакцию, отбор правил для транзакции, метод взвешенного голосования и определение метки класса).

Все функции, используемые в данных, были описаны в тексте данной работы, поэтому их код повторно приводится не будет. Ниже представлен пример запуска алгоритма на обучающей выборки с вином (рисунок 3.27).

```
from sklearn.metrics import classification_report, accuracy_score

className = 'Class'
model = AprioriClassifier()
model.fit(wine, className)
y_pred = model.predict(wine)
y_test = wine[className].values.tolist()

print(classification_report(y_test, y_pred))
print('Точность модели:', accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.95      0.93        59
           2       0.91      0.87      0.89        71
           3       0.88      0.88      0.88        48

    accuracy                           0.90       178
   macro avg       0.90      0.90      0.90       178
weighted avg       0.90      0.90      0.90       178

Точность модели: 0.898876404494382
```

Рисунок 3.27 – Классификация всех записей выборки, определение точности

В результате классификации на обучающей выборке удалось достигнуть точности 89%, что довольно неплохой показатель. Разработанную модель можно использовать для задач классификации при большой выборке данных с небольшим количество числовых атрибутов. При небольшой выборке данных количество сгенерированных правил может не покрыть множество всех примеров. При большом количестве числовых атрибутов (больше 5) время обучения модели может быть достаточно долгим. На точность классификации может также влиять количество групп, на которые разбиваются числовые атрибуты. Далее будет исследована эта зависимость на тестовых примерах.

3.2 Реализация интерфейса

Интерфейс приложения собран в виде виджета с вкладками. Создание виджета для одной выборки описано в виде функции reportGen, которая будет описана ниже. Виджет содержит 9 вкладок: Load data (загрузка данных), Data info (информация об обучающей выборке), Params (задание параметров для обучения модели), Transactions (полученный список транзакций), Association rules (полученный список правил), Filtered rules (отфильтрованный и модифицированный список правил для работы классификатора), Examples (пример работы классификатора на двух примерах из выборки), Accuracy (график зависимости точности получаемых результатов классификатором и количества групп, на которые разбиваются числовые параметры), Comparison (сравнение лучшей точности модели с другими алгоритмами классификации). Создадим виджет с вкладками, с помощью следующего кода (3.28).

```
def reportGen(dataPath=None, bins=(2, 15),
              className='Class', instances=-1, attributes=2):

    t = widgets.TabBar(['Load data', 'Data info', 'Params', 'Transactions',
                'Association rules', 'Filtered rules',
                'Examples', 'Accuracy', 'Comparison'])
```

Рисунок 3.28 – Создание таблицы

В функцию передается путь до файла на диске, если путь не задан, то пользователю будет выведена форма для загрузки файла.

Также задаются границы для формирования графика зависимости точности от количества отрезков числовых параметров, название столбца с метками класса, количество примеров (-1, если требуется читать всю

выборку), количество атрибутов (n - для чтения первых n атрибутов, -n - для чтения последних n атрибутов выборки).

Данная функция отвечает для отрисовку таблицы в вкладке виджета, вместо стандартного виджета pandas.

Это необходимо, если требуется вывести несколько таблиц в одну вкладку, что нам необходимо для программирования вкладки примеров.

```
def printTable(d, index=None, columns=None):

    if len(d.shape) < 2:
        h = 2
        w = d.shape[0] + 1
        indexes_rows = ((index, d.values) for i in range(1))
        columns = list(data.columns)
    else:
        h = d.shape[0] + 1
        w = d.shape[1] + 1
        indexes_rows = d.iterrows()
        columns = list(d.columns)

    grid = widgets.Grid(h, w, header_row=True, header_column=True)

    for i, column in enumerate(columns):
        with grid.output_to(0, i+1):
            print(column)

    j = 1
    for index, row in indexes_rows:
        with grid.output_to(j, 0):
            print(index)
        for i, column in enumerate(columns):
            with grid.output_to(j, i+1):
                if isinstance(row[i], float):
                    print(('%d' if row[i].is_integer() else '%.3f') %
                          row[i])
                else:
                    print(row[i])
        j += 1
```

Рисунок 3.29 – Вывод таблицы в виджет

Данная функция относится исключительно к оформлению и не затрагивает функциональных возможностей виджета, поэтому подробно

рассматриваться не будет. В начале функции определяется тип передаваемых данных: дата-фрейм pandas или pandas.Series и на его основе формируются массивы данных заголовков таблицы, индексов и данных таблицы. Далее создается виджет таблицы, который заполняется данными на следующем этапе. Далее описана функция, формирующая вкладки результатов обучения модели и примеров. Данные вкладки будут обновляться при изменении пользователем параметров модели во вкладке Params. Вкладка Params содержит виджет слайдера для динамического изменения количества отрезков числовых параметров (от 2 до 12). При нажатии кнопки Update динамические вкладки будут обновляться с новыми данными. Вкладка Transactions содержит таблицу со сформированным списком транзакций, вкладки Association rules и Filtered rules - сформированные списки правил (рисунок 3.30).

```
def updateBins(b=5):

    t.clear_tab('Params')
    t.clear_tab('Transactions')
    t.clear_tab('Association rules')
    t.clear_tab('Filtered rules')
    t.clear_tab('Examples')

    with t.output_to('Params', select=False):

        print('\nNumber of bins:')
        slider = jwidgets.IntSlider(
            min=2,
            max=12,
            step=1,
            value=b
        )
        display(slider)

        print()
        button_choice = jwidgets.Button(description='Update',
                                        button_style='success')
        display(button_choice)
        def on_choice(b):
            updateBins(slider.value)
        button_choice.on_click(on_choice)

    with t.output_to('Transactions', select=False):

        print('Formation of a list of transactions', end='')
        model = AprioriClassifier()
        model.fit(data, className, b)
```

Рисунок 3.30 – Заполнение динамических вкладок

Вкладка Examples содержит вычислительный эксперимент на двух примерах из обучающей выборки. Здесь выбираются 2 случайные записи, выполняется их перевод в транзакции, отбор правил, метод голосования и определения метки класса. Результаты по всем этапам выводятся во вкладку (рисунок 3.31).

```
with t.output_to('Association rules', select=False):
    display(model.rules_unfiltered)

with t.output_to('Filtered rules', select=False):
    display(model.rules)

with t.output_to('Examples', select=False):
    display(Markdown('### Examples of the work of the classifier'))
    samples = data.sample(n=2)

    ex = 1
    for index, sample in samples.iterrows():

        display(Markdown('#### Example %d (record %d)' % (ex, index)))
        printTable(sample, index)
        ex += 1

        print('\nTransaction:')
        x_test = sample.drop(className)
        transaction = model.recordToTransaction(x_test)
        print(transaction)

        t_rules = model.rulesForTransaction(transaction)
        if t_rules.shape[0] == 0:
            print('\nNo rules found for this transaction')
            print('First class selected')
            result = model.elements[model.className]['groups'][0][1]
        else:
            print('\nRules for this transaction:')
            printTable(t_rules)

            print('\nWeighted voting method:')
            classes = {}
```

Рисунок 3.31 – Заполнение последующих динамических вкладок

Далее опишем не динамические вкладки, которые генерируются один раз при создании виджета. Во вкладке Load data выполняется чтение данных

с диска или загрузка пользовательского файла. Во вкладке Data info выводится информация о выборке данных (количество классов, записей и параметров) (рисунок 3.32).

```
with t.output_to('Load data', select=False):

    if not dataPath:
        print('Select local file:\n')
        uploaded = files.upload()
        if len(uploaded):
            dataFile = list(uploaded.keys())[0]
        else:
            return
    else:
        print('Mounting drive')
        drive.mount('/content/drive/', force_remount=True)
        print('Drive mounted')
        print('Loading data ' + dataPath)
        dataFile = os.path.join(ROOT_DIR, dataPath)

    data = pd.read_csv(dataFile)
    if attributes > 0:
        columns = data.drop([className], axis=1).iloc[:,:attributes]
    else:
        columns = data.drop([className], axis=1).iloc[:,attributes:]
    columns = columns.columns.tolist()
    data = data[columns + [className]]

    if instances != -1:
        data = data[:instances]

with t.output_to('Data info', select=False):
    print('Number of Instances:', data.shape[0])
    print('Number of Attributes:', data.drop([className], axis=1).shape[1])
    print('Number of Classes:', data[className].unique().shape[0])
    display(data)
```

Рисунок 3.32 – Заполнение вкладок виджета

Во вкладке Accuracy выполняется циклическое обучение модели на исходной выборке с изменением количества отрезков числовых атрибутов от 2 до 15 и фиксацией получаемой точности модели. После получения результатов формируется график зависимости точности модели классификатора от количества групп, на которые разбиваются числовые атрибуты. Важно заметить, что данный эксперимент (по определению

точности классификатора в зависимости от параметра bins) имеет смысл только для выборок с большим количеством числовых параметров, для выборок, содержащих только категориальные атрибуты строить такой график не целесообразно (рисунок 3.33).

```
with t.output_to('Accuracy', select=False):

    display(Markdown(
        '### Dependence of classification accuracy on the number of bins'
    ))

    binsN = range(bins[0], bins[1]+1)
    model = AprioriClassifier()
    accuracy = []

    for b in binsN:
        model.fit(data, className, b)
        y_pred = model.predict(data)
        y_test = data[className].values.tolist()
        accuracy.append(accuracy_score(y_test, y_pred))

    ax = plt.figure().gca()
    ax.xaxis.set_major_locator(MaxNLocator(integer=True))
    plt.plot(binsN, accuracy)
    plt.ylabel('Accuracy')
    plt.xlabel('Number of bins')
    plt.show()

    display(pd.DataFrame(data=[np.transpose(accuracy)],
                         columns=binsN, index=['Accuracy']))
```

Рисунок 3.33 – Заполнение вкладки Accuracy виджета

На последней вкладке формы выводится столбчатая диаграмма точностей модели классификатора Apriori и других моделей для сравнения. Для этого необходимо выполнить предобработку данных (3.34).

```
def catToNum(x):

    # Выделение категориальных признаков
    CATEGORICALS = [column for column in x
                    if not is_numeric_dtype(x[column].dtype)
                    or is_bool_dtype(x[column].dtype)]

    x = x.reset_index()

    # Преобразование категориальных признаков к числовым
    for column in CATEGORICALS:
        one = OneHotEncoder(sparse=False)
        categ_encode = one.fit_transform(x[[column]])
        categ_encode = pd.DataFrame(categ_encode,
                                    columns=x[column].unique())
        x = x.drop(columns=[column])
        x = pd.concat([x, categ_encode], axis=1)

    return x

x_train = catToNum(data_train.drop([className], axis=1))
x_test = catToNum(data_test.drop([className], axis=1))
y_train = data_train[className].values.tolist()
y_test = data_test[className].values.tolist()
```

Рисунок 3.34 – Предобработка данных

Сравнивать работу классификатора основанного на аффинитивном анализе мы будем деревом классификации (рисунок 3.35), случайным лесом (рисунок 3.36) и классификатором kNN (рисунок 3.37).

```
clf_dt = DecisionTreeClassifier(criterion='entropy')
clf_dt.fit(x_train, y_train)
y_pred = clf_dt.predict(x_test)
results['Decision Tree'] = accuracy_score(y_test, y_pred)
```

Рисунок 3.35 – Дерево принятия решений

```
clf_rf = RandomForestClassifier(n_estimators=100,
                                bootstrap = True,
                                max_features = 'sqrt')
clf_rf.fit(x_train, y_train)
y_pred = clf_rf.predict(x_test)
results['Random Forest'] = accuracy_score(y_test, y_pred)
```

Рисунок 3.36 – Случайный лес

```
clf_knn = KNeighborsClassifier(n_neighbors=
                               data_train[className].unique().shape[0],
                               metric='euclidean')
mm_scaler = MinMaxScaler()
x_train_norm = mm_scaler.fit_transform(x_train)
x_test_norm = mm_scaler.fit_transform(x_test)
clf_knn.fit(x_train_norm, y_train)
y_pred = clf_knn.predict(x_test_norm)
results['KNN'] = accuracy_score(y_test, y_pred)

index = np.arange(len(results))
ax = plt.figure().gca()
plt.bar(index, results.values(), color=(0.2, 0.4, 0.6, 0.6))
plt.xticks(index, results.keys(), rotation=60)
ax.set_ylim([0, 1])
plt.ylabel('Accuracy')
plt.show()

display(pd.DataFrame(data=[results.values()],
                     columns=list(results.keys()),
                     index=['Accuracy']))
```

Рисунок 3.37 – KNN

3.3 Проведение вычислительных экспериментов

Проведем вычислительный эксперимент по классификации данных, а также продемонстрируем работу программного обеспечения.

Визуализация интерфейса представлена на изображениях ниже (рисунок 3.38). По вертикали слева располагаются названия файлов с различными выборками данных. При выборе данных обновляется вся информация на представленных вкладках. По горизонтали располагаются вкладки, содержащие различную информацию классификатора.

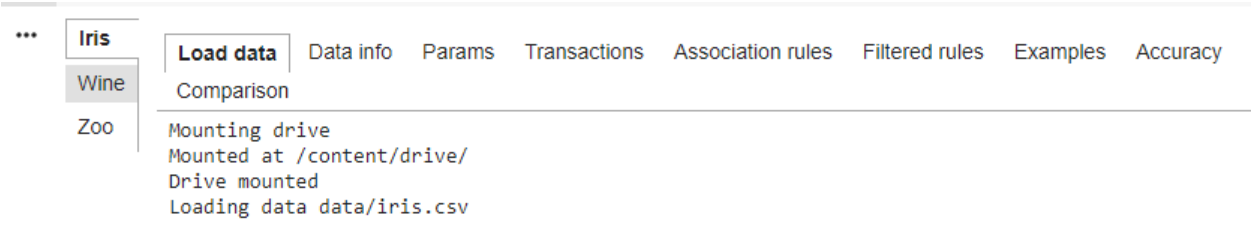
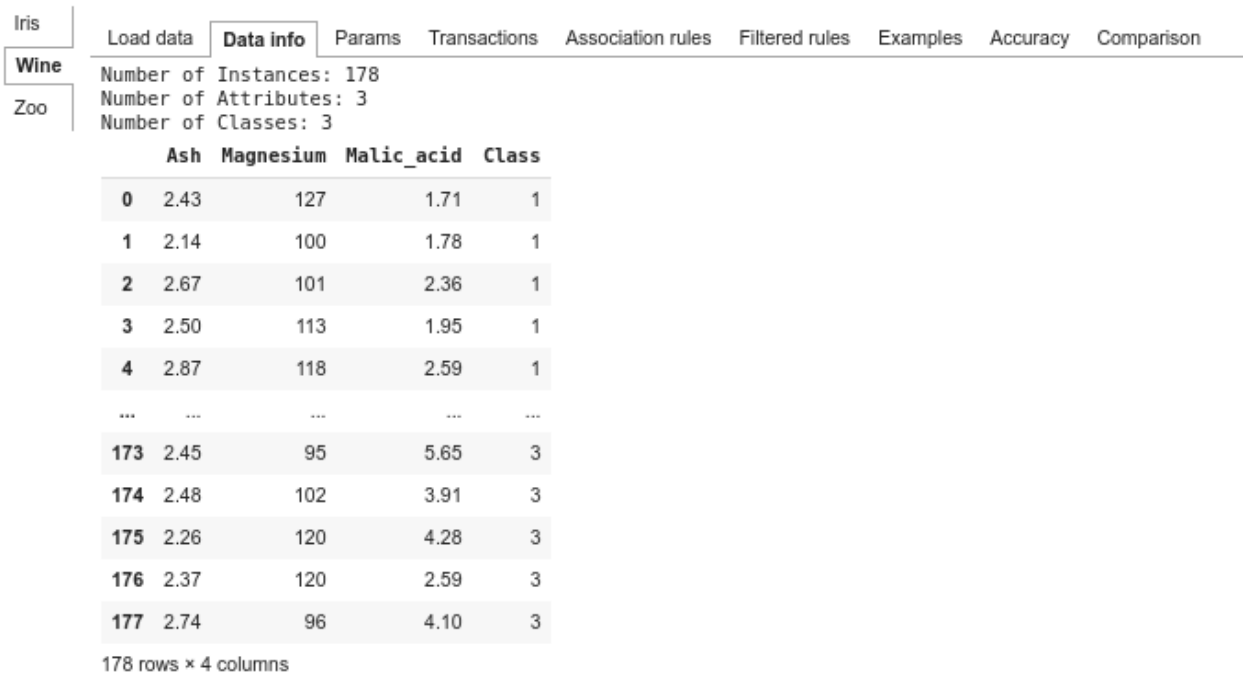


Рисунок 3.38 – Информация о загружаемых данных

Вывод информации об обучающей выборке и ее представление в виде таблицы доступно на вкладке «Data info» (рисунок 3.39).



	Ash	Magnesium	Malic_acid	Class
0	2.43	127	1.71	1
1	2.14	100	1.78	1
2	2.67	101	2.36	1
3	2.50	113	1.95	1
4	2.87	118	2.59	1
...	...	...	...	...
173	2.45	95	5.65	3
174	2.48	102	3.91	3
175	2.26	120	4.28	3
176	2.37	120	2.59	3
177	2.74	96	4.10	3

178 rows × 4 columns

Рисунок 3.39 – Информация о содержимом файла данных

На вкладке «Params» можно задать количество отрезков, на которые будет разделена ось каждого из числовых признаков (параметр bins) (рисунок 3.40).

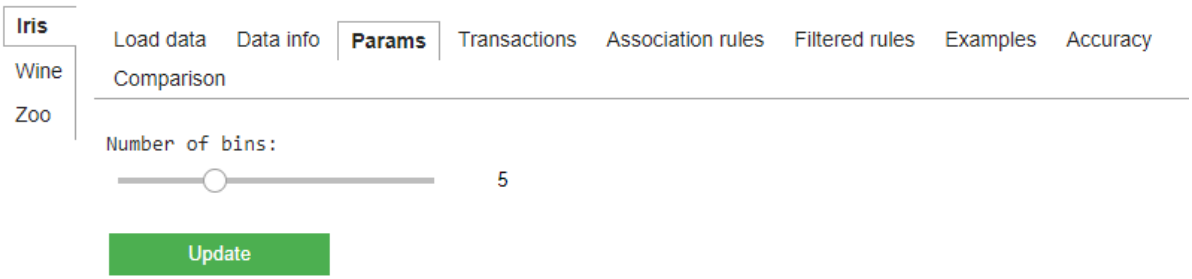
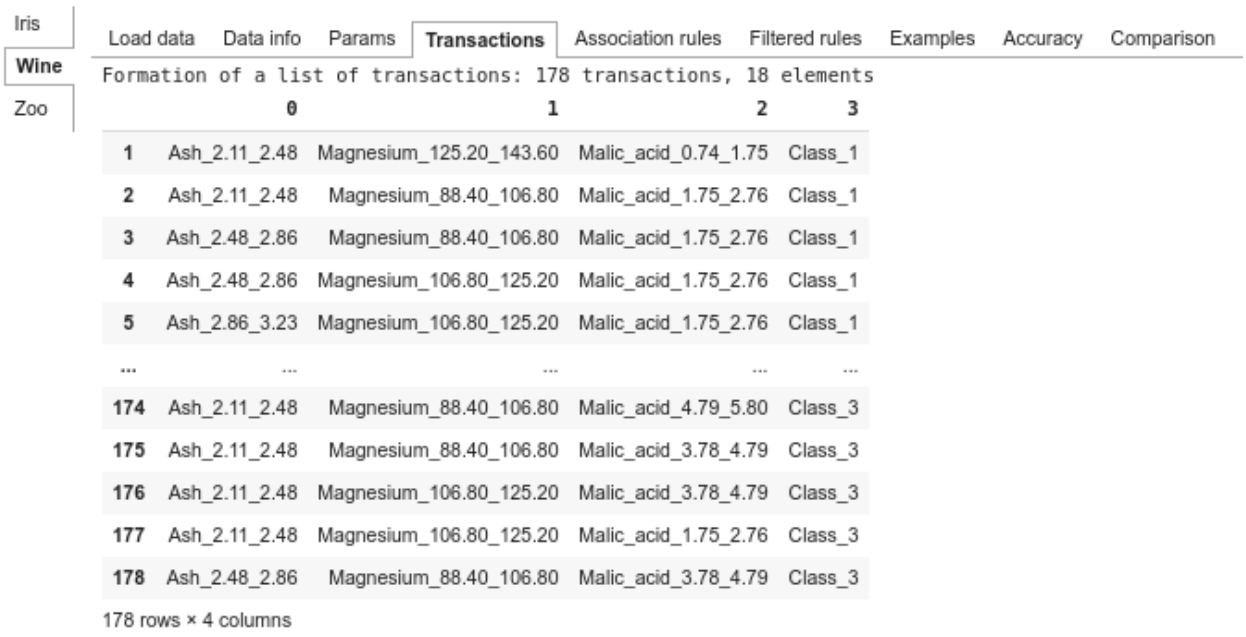


Рисунок 3.40 – Выбор числа групп

На вкладке «Transactions» отображается матрица полученных транзакций (рисунок 3.41)

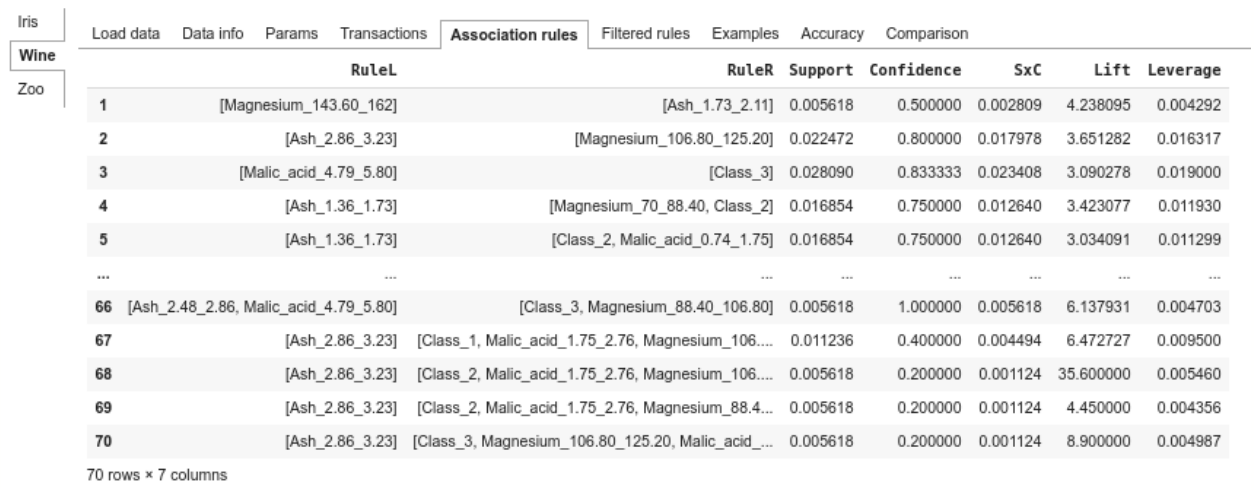

Iris
Wine
Zoo
Load data | Data info | Params | Transactions | Association rules | Filtered rules | Examples | Accuracy | Comparison
Formation of a list of transactions: 178 transactions, 18 elements

	0	1	2	3
1	Ash_2.11_2.48	Magnesium_125.20_143.60	Malic_acid_0.74_1.75	Class_1
2	Ash_2.11_2.48	Magnesium_88.40_106.80	Malic_acid_1.75_2.76	Class_1
3	Ash_2.48_2.86	Magnesium_88.40_106.80	Malic_acid_1.75_2.76	Class_1
4	Ash_2.48_2.86	Magnesium_106.80_125.20	Malic_acid_1.75_2.76	Class_1
5	Ash_2.86_3.23	Magnesium_106.80_125.20	Malic_acid_1.75_2.76	Class_1
...	...	...	...	...
174	Ash_2.11_2.48	Magnesium_88.40_106.80	Malic_acid_4.79_5.80	Class_3
175	Ash_2.11_2.48	Magnesium_88.40_106.80	Malic_acid_3.78_4.79	Class_3
176	Ash_2.11_2.48	Magnesium_106.80_125.20	Malic_acid_3.78_4.79	Class_3
177	Ash_2.11_2.48	Magnesium_106.80_125.20	Malic_acid_1.75_2.76	Class_3
178	Ash_2.48_2.86	Magnesium_88.40_106.80	Malic_acid_3.78_4.79	Class_3

178 rows × 4 columns

Рисунок 3.41 – Полученная матрица транзакций

На вкладке «Association rules» отображается таблица ассоциативных правил, найденных с помощью алгоритма Apriori (рисунок 3.42).


Iris
Wine
Zoo
Load data | Data info | Params | Transactions | Association rules | Filtered rules | Examples | Accuracy | Comparison

	RuleL	RuleR	Support	Confidence	SxC	Lift	Leverage
1	[Magnesium_143.60_162]	[Ash_1.73_2.11]	0.005618	0.500000	0.002809	4.238095	0.004292
2	[Ash_2.86_3.23]	[Magnesium_106.80_125.20]	0.022472	0.800000	0.017978	3.651282	0.016317
3	[Malic_acid_4.79_5.80]	[Class_3]	0.028090	0.833333	0.023408	3.090278	0.019000
4	[Ash_1.36_1.73]	[Magnesium_70_88.40, Class_2]	0.016854	0.750000	0.012640	3.423077	0.011930
5	[Ash_1.36_1.73]	[Class_2, Malic_acid_0.74_1.75]	0.016854	0.750000	0.012640	3.034091	0.011299
...	...	...	...	...	...	...	...
66	[Ash_2.48_2.86, Malic_acid_4.79_5.80]	[Class_3, Magnesium_88.40_106.80]	0.005618	1.000000	0.005618	6.137931	0.004703
67	[Ash_2.86_3.23]	[Class_1, Malic_acid_1.75_2.76, Magnesium_106....	0.011236	0.400000	0.004494	6.472727	0.009500
68	[Ash_2.86_3.23]	[Class_2, Malic_acid_1.75_2.76, Magnesium_106....	0.005618	0.200000	0.001124	35.600000	0.005460
69	[Ash_2.86_3.23]	[Class_2, Malic_acid_1.75_2.76, Magnesium_88.4...	0.005618	0.200000	0.001124	4.450000	0.004356
70	[Ash_2.86_3.23]	[Class_3, Magnesium_106.80_125.20, Malic_acid_...	0.005618	0.200000	0.001124	8.900000	0.004987

70 rows × 7 columns

Рисунок 3.42 – Полученные ассоциации из транзакций

Результат модификации и фильтрации списка найденных правил отображается на вкладке «Filteres rules» (рисунок 3.43).

Iris
Wine
Zoo

Load data | Data info | Params | Transactions | Association rules | **Filtered rules** | Examples | Accuracy | Comparison

	RuleL	RuleR	Support	Confidence	SxC	Lift	Leverage
1	[Malic_acid_4.79_5.80]	[Class_3]	0.028090	0.833333	0.023408	3.090278	0.019000
2	[Ash_1.36_1.73]	[Class_2]	0.016854	0.750000	0.012640	3.423077	0.011930
3	[Magnesium_143.60_162]	[Class_2]	0.005618	0.500000	0.002809	5.235294	0.004545
4	[Magnesium_125.20_143.60, Ash_2.11_2.48]	[Class_1]	0.011236	1.000000	0.011236	3.016949	0.007512
5	[Magnesium_125.20_143.60]	[Class_2]	0.011236	0.285714	0.003210	3.912088	0.008364
6	[Malic_acid_2.76_3.78]	[Class_3]	0.044944	0.320000	0.014382	3.797333	0.033108
7	[Ash_2.48_2.86, Malic_acid_4.79_5.80]	[Class_3]	0.005618	1.000000	0.005618	3.708333	0.004103
8	[Ash_2.86_3.23]	[Class_1]	0.011236	0.400000	0.004494	3.095652	0.007606
9	[Ash_2.86_3.23]	[Class_2]	0.005618	0.200000	0.001124	7.120000	0.004829
10	[Ash_2.86_3.23]	[Class_2]	0.011236	0.400000	0.004494	4.746667	0.008869
11	[Ash_2.86_3.23]	[Class_3]	0.005618	0.200000	0.001124	3.236364	0.003882
12	[Magnesium_143.60_162]	[Class_2]	0.011236	1.000000	0.011236	4.045455	0.008459
13	[Ash_1.73_2.11, Magnesium_106.80_125.20]	[Class_2]	0.005618	0.500000	0.002809	14.833333	0.005239
14	[Ash_1.73_2.11, Magnesium_125.20_143.60]	[Class_2]	0.005618	1.000000	0.005618	4.045455	0.004229
15	[Magnesium_70_88.40, Ash_1.73_2.11]	[Class_2]	0.011236	0.200000	0.002247	5.933333	0.009342
16	[Malic_acid_1.75_2.76, Ash_1.73_2.11]	[Class_2]	0.005618	0.500000	0.002809	4.045455	0.004229
17	[Magnesium_70_88.40, Malic_acid_1.75_2.76]	[Class_2]	0.028090	0.833333	0.023408	4.238095	0.021462
18	[Malic_acid_3.78_4.79, Ash_2.11_2.48]	[Class_3]	0.016854	0.200000	0.003371	3.236364	0.011646
19	[Malic_acid_4.79_5.80, Ash_2.11_2.48]	[Class_3]	0.005618	0.200000	0.001124	4.450000	0.004356
20	[Ash_2.48_2.86, Malic_acid_0.74_1.75, Magnesiu...	[Class_1]	0.016854	1.000000	0.016854	3.016949	0.011268
21	[Ash_2.48_2.86, Magnesium_125.20_143.60]	[Class_1]	0.005618	0.250000	0.001404	8.900000	0.004987
22	[Magnesium_70_88.40, Ash_2.48_2.86]	[Class_2]	0.028090	1.000000	0.028090	4.045455	0.021146

Рисунок 3.43 – Полученный фильтрованный список правил

На вкладке «Examples» расположены примеры классификации случайных записей из исходной выборки данных (рисунок 3.44).

На вкладке «Accuracy» расположен график расчета точности классификатора в зависимости от параметра bins (рисунок 3.45).

Iris | Wine | Zoo

Load data | Data info | Params | Transactions | Association rules | Filtered rules | **Examples** | Accuracy | Comparison

Examples of the work of the classifier

Example 1 (record 58)

	Ash	Magnesium	Malic_acid	Class
58	2.500	108	1.430	1

Transaction:
['Ash_2.48_2.86', 'Magnesium_106.80_125.20', 'Malic_acid_0.74_1.75']

Rules for this transaction:

	RuleL	RuleR	Support	Confidence	SxC	Lift	Leverage
1	['Ash_2.48_2.86', 'Malic_acid_0.74_1.75', 'Magnesium_106.80_125.20']	['Class_1']	0.017	1	0.017	3.017	0.011

Weighted voting method:
{'Class_1': 0.016853932584269662}

Class: 1, Predicted: 1

Example 2 (record 132)

	Ash	Magnesium	Malic_acid	Class
132	2.400	98	2.310	3

Transaction:
['Ash_2.11_2.48', 'Magnesium_88.40_106.80', 'Malic_acid_1.75_2.76']

No rules found for this transaction
First class selected

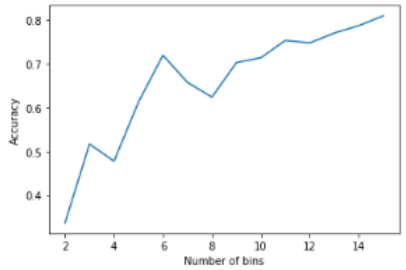
Class: 3, Predicted: 1

Рисунок 3.44 – Примеры работы классификатора по 2-м записям

Iris | Wine | Zoo

Load data | Data info | Params | Transactions | Association rules | Filtered rules | Examples | **Accuracy** | Comparison

Dependence of classification accuracy on the number of bins

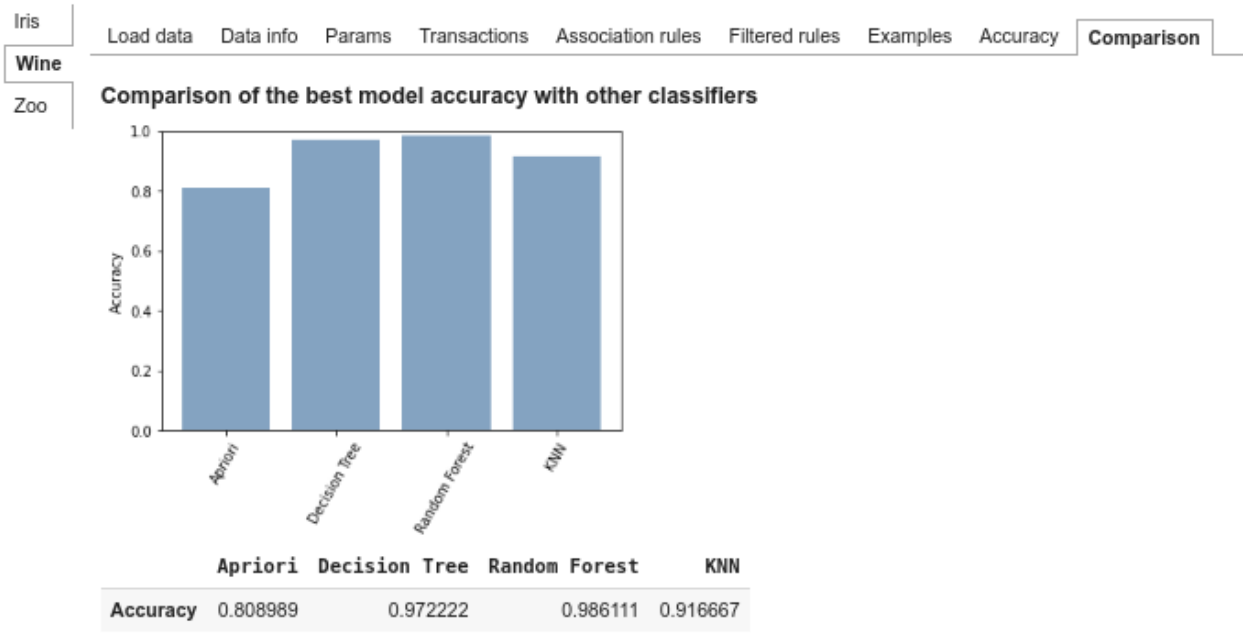


	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Accuracy	0.337079	0.516854	0.477528	0.61236	0.719101	0.657303	0.623596	0.702247	0.713483	0.752809	0.747191	0.769663	0.786517	0.808989

Рисунок 3.45 – График точности от количества bins

По этому графику можно определять оптимальное значение параметра bins для анализируемого набора данных.

На последней вкладке выводится столбчатая диаграмма лучшей для сравнения точности классификации на основе алгоритма Apriori с другими алгоритмами машинного обучения (рисунок 3.46).



	Apriori	Decision Tree	Random Forest	KNN
Accuracy	0.808989	0.972222	0.986111	0.916667

Рисунок 3.46 – Сравнение точности Apriori с другими классификаторами

Как видно из столбчатой диаграммы, точность работы представленных классификаторов находится в диапазоне от 80 до 95 %.

Выводы по разделу

На основании проведенных в главе исследований можно сделать следующие выводы:

- На языке программирования Python было разработано программное обеспечение, реализующее предложенные в диссертационном исследовании подходы по построения классификатора данных на основе алгоритма Apriori.
- Вычислительные эксперименты показали, что точность классификаторов, получаемых с использованием предложенных подходов на основе алгоритма Apriori сопоставимы с точностью классификаторов, получаемых с помощью алгоритмов Random forest и kNN.

Заключение

Представленные в магистерской диссертации исследования позволяют сформировать следующие выводы:

- Анализ литературных источников показал, что применение технологий машинного обучения, позволяет многократно снизить трудоемкость решения задач моделирования.
- В ходе анализа существующих исследований по практическому применению технологий машинного обучения сформирована сводная таблица устанавливающая соответствие между типами решаемых задач и используемыми алгоритмами. Для каждого типа задачи приведены примеры использования алгоритмов машинного обучения
- На основании анализа литературных источников установлено, что развитие алгоритмов машинного обучения сконцентрировано по двум направлениям: первое – увеличение степени автоматизации алгоритмов при обработке исходных данных и второе – разработку новых способов применения существующих алгоритмов, расширяющих перечень задач, которые они могут решать.
- На основании предыдущего вывода доказана актуальность проводимого исследования на тему моделирование синтеза классификаторов на основе аффинитивного анализа данных.
- Разработана технология использования алгоритма apriori для построения классификатора данных. Предложенный подход предполагает преобразование исходных данных во множество транзакций, состоящих из элементарных событий. Элементарные события, связанные с категориальными параметрами и меткой класса, генерируются на основе значений категорий. Элементарные событий связанные с числовыми признаками генерируются на основе факта принадлежности значений одному из отрезков числовой оси (рисунок 2.3). Получившееся множество транзакций анализируется с помощью алгоритма apriori для получения

множества правил, связывающих элементарные события. Полученные ассоциативные правила фильтруются и модифицируются таким образом, чтобы в их правой части содержалось указание на значение метки класса. Модифицированные правила образуют классификатор. При классификации объекта каждое сработавшее правило голосует за свой класс. Классификатор присваивает объекту тут класс, который набрал большее количество голосов.

- На языке программирования Python было разработано программное обеспечение, реализующее предложенные в диссертационном исследовании подходы по построения классификатора данных на основе алгоритма Apriori.

- Вычислительные эксперименты показали, что точность классификаторов, получаемых с использованием предложенных подходов на основе алгоритма Apriori сопоставимы с точностью классификаторов, получаемых с помощью алгоритмов Random forest и kNN.

Таким образом, все поставленные задачи выполнены и достигнута цель исследования.

Список используемой литературы

1. Арзаманов, Н.А. Технология машинного обучения и ее практическое применение / Н.А. Арзамазов, Н.И. Ематина // Исследование различных направлений современной науки – материалы XXI Международной научно-практической конференции. В 2-х частях. 24 апреля 2017. – Научный центр "Олимп" (Астрахань), 2017. – с. 7-10. – Текст : непосредственный.

2. Аусабаев, Д.М. Использование машинного обучения в поддержке принятия решений / Д.М. Аусабаев, О.П. Волобуев // Прикладная математика и информатика: современные исследования в области естественных и технических наук – материалы III научно-практической всероссийской конференции (школы-семинара) молодых ученых. Тольятти, 24–25 апреля 2017 года. – Издатель Качалин Александр Васильевич, 2017. – с. 43-47. – Текст : непосредственный.

3. Власов, А.В. Машинное обучение применительно к задаче классификации семян зерновых культур в видеопотоке / А.В. Власов, А.С. Федеев // Молодежь и современные информационные технологии – сборник трудов XIV Международной научно-практической конференции студентов, аспирантов и молодых учёных, 07–11 ноября 2016. – Национальный исследовательский Томский политехнический университет (Томск), 2016. – с. 133-135. – Текст : непосредственный.

4. Жуков, Д.А. Формирование контрольных выборок при технической диагностике объекта с применением машинного обучения / Д.А. Жуков, А.С. Хорева, Ю.Е. Кувайскова, В.Н. Клячкин // Математические методы и модели: теория, приложения и роль в образовании – международная научно-техническая конференция : сборник научных трудов, 28–30 апреля 2016 года. – Ульяновский государственный технический университет (Ульяновск), 2016. – с. 44-48. – Текст : непосредственный.

5. Иванников Ю.Ю. Применение методов машинного обучения для выявления бот-трафика среди запросов к веб-приложению / Ю.Ю. Иванников, Е.Ю. Митрофанова // Сборник студенческих научных работ факультета компьютерных наук ВГУ, Факультет компьютерных наук, 2017. – ФГБОУ ВО «Воронежский государственный университет», 2017. – с. 119-123. – Текст : непосредственный.

6. Клячин В.Н. Использование агрегированных классификаторов при технической диагностике на базе машинного обучения / В.Н. Клячин, Ю.Е. Кувайскова, Д.А. Жуков // Информационные технологии и нанотехнологии (ИТНТ-2017) – сборник трудов III международной конференции и молодежной школы. Самарский национальный исследовательский университет имени академика С.П. Королева. 2017. – Предприятие "Новая техника" (Самара), 2017. – с. 1770-1773. – Текст : непосредственный.

7. Кононова, Н.В. Исследование подсистемы контентной фильтрации с использованием методов машинного обучения / Н.В. Кононова, Ю.А. Андрусенко, Т.А. Самокаева // Студенческая наука для развития информационного общества – сборник материалов VI Всероссийской научно-технической конференции. 22–26 мая 2017. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 268-270. – Текст : непосредственный.

8. Мелдебай, М.А. Анализ мнений покупателей на основе машинного обучения / М.А. Мелдебай, А.К. Сарбасова // Прикладная математика и информатика: современные исследования в области естественных и технических наук – материалы III научно-практической всероссийской конференции (школы-семинара) молодых ученых. 24–25 апреля 2017 года. – Издатель Качалин Александр Васильевич, 2017. – с. 360-363. – Текст : непосредственный.

9. Наумов, Д.П. Регулятор САР на основе машинного обучения / Д.П. Наумов, Д.П. Стариков // Информационные технологии в управлении,

автоматизации и мехатронике – сборник научных трудов Международной научно-технической конференции. 06–07 апреля 2017 года. – ЗАО "Университетская книга" (Курск), 2017. – с. 106-114. – Текст : непосредственный.

10. Осколков, В.М. Использование метода машинного обучения для повышения продуктивности на предприятии / В.М. Осколков, Н.И. Шаханов, И.А. Варфоломеев, О.В. Юдина, Е.В. Ершов // Автоматизация и энергосбережение машиностроительного и металлургического производств, технология и надежность машин, приборов и оборудования – материалы XII Международной научно-технической конференции, 21 марта 2017. – Вологодский государственный университет (Вологда), 2017. – с. 177-180. – Текст : непосредственный.

11. Осколков, В.М. Применение параллельных вычислений для прогнозирования на основе алгоритма машинного обучения Random Forest / В.М. Осколков, Н.И. Шаханов, И.А. Варфоломеев, О.В. Юдина, Л.Н. Виноградова, Е.В. Ершов // Сборник трудов конференции Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание, Курск, 16–19 мая 2017 года. – Юго-Западный государственный университет (Курск), 2017. – с. 267-269. – Текст : непосредственный.

12. Соловьев, А.Ю. Применение машинного обучения для прогнозирования неблагоприятных исходов в ургентной хирургии / Соловьев А.Ю., Берегов М.М., Вахеева Ю.М., Баутин А.Н., Гусев А.В. // Медико-биологические, клинические и социальные вопросы здоровья и патологии человека – материалы III Всероссийской образовательно-научной конференции студентов и молодых ученых с международным участием в рамках XIII областного фестиваля "Молодые ученые - развитию Ивановской области". 2017. – Ивановская государственная медицинская академия (Иваново), 2017. – с. 129-130. – Текст : непосредственный.

13. Ткач, Т.Ч. Машинное обучение и обработка больших данных - обучение в основной и средней школе / Т.Ч. Ткач // Актуальные проблемы методики обучения информатике и математике в современной школе – материалы международной научно-практической интернет-конференции. Московский педагогический государственный университет, Москва, 24 апреля 2020 года. – Московский педагогический государственный университет (Москва), 2020. – с. 217-223. – Текст : непосредственный.

14. Федотов, И.А. Применение технологий машинного обучения для прогнозирования ситуации на финансовых рынках / И.А. Федотов // Студенческая наука для развития информационного общества – сборник материалов VI Всероссийской научно-технической конференции. 22–26 мая 2017. – Северо-Кавказский федеральный университет (Ставрополь), 2017. – с. 361-363. – Текст : непосредственный.

15. Шогунова, Н.М. Разработка программы компьютерного зрения для локализации объектов на изображении и видеопотоке / Н.М. Шогунова, В.Я. Олексюк // Сборник статей Всероссийской студенческой научно-практической междисциплинарной конференции «Молодежь. Наука. Общество». 2020. – Тольяттинский государственный университет, 2020. – Текст : непосредственный.

16. Якимчук, А.А. Глубокое обучение как эффективный метод машинного обучения / А.А. Якимчук // Научное сообщество студентов XXI столетия. Технические науки – сборник статей по материалам XCII студенческой международной научно-практической конференции. 2020. – ООО “Сибирская академическая книга” (Новосибирск), 2020. – с. 40-43. – Текст : непосредственный.

17. Ярыгин, А.А. Актуальные вопросы машинного обучения с подкреплением интеллектуальных агентов в задачах принятия решений / А.А. Ярыгин // Автоматизация: проблемы, идеи, решения - сборник статей по итогам Международной научно-практической конференции 2017. – ООО

"Агентство международных исследований", 2017. – с. 62-68. – Текст : непосредственный.

18. Agarwal, J. Mining Frequent Quality Factors of Software System Using Apriori Algorithm / Jyoti Agarwal, Sanjay Kumar Dubey, Rajdev Tiwari // Proceedings of the International Conference on Data Engineering and Communication Technology (ICDECT 2016). – Springer Science+Business Media Singapore, 2017. – pp. 481-490. – Текст : непосредственный.

19. Arora, P. Design and Performance Analysis of Distributed Implementation of Apriori Algorithm in Grid Environment / Priyanka Arora, Sarbjeet Singh // ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India. – Springer International Publishing Switzerland, 2014. – pp. 653-661. – Текст : непосредственный.

20. Benhamouda, N.C. Meta-Apriori: A New Algorithm for Frequent Pattern Detection / Neyla Cherifa Benhamouda, Habiba Drias, Célia Hirèche // Asian Conference on Intelligent Information and Database Systems: Intelligent Information and Database Systems (ACIIDS 2016). – Springer-Verlag Berlin Heidelberg, 2016 – pp. 277-285. – Текст : непосредственный.

21. Child, Ch. The Apriori Stochastic Dependency Detection (ASDD) Algorithm for Learning Stochastic Logic Rules / Christopher Child, Kostas Stathis // International Workshop on Computational Logic in Multi-Agent Systems (CLIMA 2004). – Springer-Verlag Berlin Heidelberg, 2004. – pp. 234-249. – Текст : непосредственный.

22. Choo, Y.H. A Rough-Apriori Technique in Mining Linguistic Association Rules / Yun-Huoy Choo, Azuraliza Abu Bakar, Abdul Razak Hamdan // International Conference on Advanced Data Mining and Applications (ADMA 2008). – Springer-Verlag Berlin Heidelberg, 2008. – pp. 548-555. – Текст : непосредственный.

23. Dahbi, A. Using Multiple Minimum Support to Auto-adjust the Threshold of Support in Apriori Algorithm / Azzeddine Dahbi, Youssef Balouki, Taoufiq Gadi // Proceedings of the Ninth International Conference on Soft

Computing and Pattern Recognition (SoCPaR 2017). – Springer International Publishing AG 2018. – pp. 111-119. – Текст : непосредственный.

24. Dhanya, S. An Enhancement of the MapReduce Apriori Algorithm Using Vertical Data Layout and Set Theory Concept of Intersection / S. Dhanya, M. Vysaakan, A. S. Mahesh // Intelligent Systems Technologies and Applications. – Springer International Publishing Switzerland, 2016. – pp. 225-233. – Текст : непосредственный.

25. Djenouri, Y. GA-Apriori: Combining Apriori Heuristic and Genetic Algorithms for Solving the Frequent Itemsets Mining Problem / Youcef Djenouri, Marco Comuzzi // Pacific-Asia Conference on Knowledge Discovery and Data Mining: Trends and Applications in Knowledge Discovery and Data Mining (PAKDD 2017). – Springer International Publishing AG, 2017. – pp. 138-148. – Текст : непосредственный.

26. Fernández-Baizán, M.C. Using the Apriori Algorithm to Improve Rough Sets Results / María C. Fernández-Baizán, Menasalvas Ruiz, José M. Peña Sánchez, Juan Francisco Martínez Sarrías, Socorro Millán // International Conference on Rough Sets and Current Trends in Computing (RSCTC 2000). – Springer-Verlag Berlin Heidelberg, 2001. – pp. 291-295. – Текст : непосредственный.

27. Inokuchi, A. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data / Akihiro Inokuchi, Takashi Washio, Hiroshi Motoda // European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2000). – Springer-Verlag Berlin Heidelberg, 2000. – pp. 13-23. – Текст : непосредственный.

28. Jovanoski, V. Classification Rule Learning with APRIORI-C / Viktor Jovanoski, Nada Lavrač // Portuguese Conference on Artificial Intelligence (EPIA 2001). – Springer-Verlag Berlin Heidelberg, 2001. – pp. 44-51. – Текст : непосредственный.

29. Liang, X. Improved Apriori Algorithm for Mining Association Rules of Many Diseases / Xu Liang, Caixia Xue, Ming Huang // International

Symposium on Intelligence Computation and Applications: Computational Intelligence and Intelligent Systems (ISICA 2010). – Springer-Verlag Berlin Heidelberg, 2010. – pp. 272-279. – Текст : непосредственный.

30. Liao, B. An Improved Algorithm of Apriori / Binhua Liao // International Symposium on Intelligence Computation and Applications: Computational Intelligence and Intelligent Systems (ISICA 2009). – Springer-Verlag Berlin Heidelberg, 2009. – pp. 427-432. – Текст : непосредственный.

31. Pommerenke, C. A Modified Apriori Algorithm for Analysing High-Dimensional Gene Data / Claudia Pommerenke, Benedikt Friedrich, Thorsten Johl, Lothar Jänsch, Susanne Häussler, Frank Klawonn // International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2011). – Springer-Verlag Berlin Heidelberg, 2011. – pp. 236-243. – Текст : непосредственный.

32. Sharma, N.K. Study and Analysis of Incremental Apriori Algorithm / Neeraj Kumar Sharma, N. K. Nagwani // International Conference on High Performance Architecture and Grid Computing (HPAGC 2011). – Springer-Verlag Berlin Heidelberg, 2011. – pp. 470-472. – Текст : непосредственный.