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)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/ специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Исследование алгоритмов классификации и кластеризации больших объемов данных»

Студент С.П. Григорьев _____
(И.О. Фамилия) (личная подпись)

Руководитель к.ф.-м.н. О.В. Лелонд
(ученая степень, звание, И.О. Фамилия)

Консультант Дайнеко М.В.
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема выпускной квалификационной работы – «Исследование алгоритмов классификации и кластеризации больших объемов данных».

Как показывает практика, качество результатов анализа больших данных зависит от свойств конкретного алгоритма, используемого для классификации и кластеризации данных. Выбор того или алгоритма классификации и кластеризации обусловлен не только объемами обрабатываемых данных, но и его эффективностью.

Объектом исследования бакалаврской работы является интеллектуальный анализ больших объемов данных.

Предметом исследования бакалаврской работы являются алгоритмы классификации и кластеризации больших объемов данных.

Цель бакалаврской работы – исследование алгоритмов классификации и кластеризации на предмет эффективности использования для решения задач анализа больших объемов данных.

Методы исследования – интеллектуальный анализ данных, методы классификации и кластеризации данных.

Практическая значимость бакалаврской работы заключается в разработке программы, реализующей эффективные алгоритмы классификации и кластеризации больших объемов данных.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы для бизнес-аналитиков и разработчиков программ, использующих для принятия управленческих решений методы и алгоритмы интеллектуального анализа больших объемов данных.

Выпускная квалификационная работа состоит из 48 страниц текста, 21 рисунка, 3 таблиц и 23 источников.

Abstract

The topic of the given graduation work is “Study of algorithms for the classification and clustering of large amounts of data”.

As practice shows, the quality of the results of big data analysis depends on the properties of the specific algorithm used to classify and cluster data. The choice of one or another classification and clustering algorithm is determined not only by the amount of processed data, but also by its efficiency.

The object of study of the graduation work is the analyzing large amounts of data.

The subject of study of the graduation work is algorithms for the classification and clustering of large amounts of data.

The aim of the graduation work is the study of classification and clustering algorithms for their effectiveness in solving problems of analyzing large amounts of data.

Research methods: Data mining, methods of data classification and clustering.

The practical significance of the graduation work lies in the development of a program that implements effective algorithms for the classification and clustering of large amounts of data.

The results of the graduation work are of scientific and practical interest and can be recommended for business analysts and program developers who use methods and algorithms for analyzing large amounts of data to make management decisions.

The graduation work consists of an explanatory note on 48 pages including 21 figures, 3 tables, the list of 23 references.

Оглавление

Введение

Решаемые в последнее время практические задачи анализа больших объемов данных связаны с применением методов и алгоритмов машинного обучения.

Следует отметить, что к методам, широко применяемым для интеллектуального анализа данных, относятся методы классификации и кластеризации, для реализации которых используются различные алгоритмы [5].

Вместе с тем, как показывает практика, качество результатов анализа больших данных зависит от свойств конкретного алгоритма, используемого для классификации и кластеризации данных.

Выбор того или алгоритма классификации и кластеризации обусловлен не только объемами обрабатываемых данных, но и его эффективностью.

Для определения целесообразности применения алгоритмов классификации и кластеризации для решения конкретной задачи анализа данных необходимо провести их полное исследование.

Таким образом, исследование алгоритмов классификации и кластеризации больших объемов данных представляет актуальность и научно-практический интерес.

Объектом исследования бакалаврской работы является интеллектуальный анализ больших объемов данных.

Предметом исследования бакалаврской работы являются алгоритмы классификации и кластеризации больших объемов данных.

Цель бакалаврской работы – исследование алгоритмов классификации и кластеризации на предмет эффективности использования для решения задач анализа больших объемов данных.

Для достижения данной цели необходимо выполнить следующие задачи:

- произвести анализ алгоритмов классификации больших объемов данных и выбрать наиболее эффективный алгоритм классификации;

- произвести анализ алгоритмов кластеризации больших объемов данных и выбрать наиболее эффективный алгоритм кластеризации;
- разработать и протестировать программу, реализующую выбранные алгоритмы классификации и кластеризации больших объемов данных.

Методы исследования – интеллектуальный анализ данных, методы классификации и кластеризации данных.

Практическая значимость бакалаврской работы заключается в разработке программы, реализующей эффективные алгоритмы классификации и кластеризации больших объемов данных.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы.

Первая главе работы посвящена анализу алгоритмов классификации больших объемов данных.

Вторая глава работы посвящена анализу алгоритмов кластеризации больших объемов данных.

В третьей главе рассматривается процесс разработки программы, реализующей эффективные алгоритмы классификации и кластеризации больших объемов данных.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Бакалаврская работа состоит из 46 страниц текста, 21 рисунку, 3 таблиц и 23 источников.

Глава 1 Анализ алгоритмов классификации больших объемов данных

Классификация - это процесс автоматического создания модели классов из набора записей, содержащих метки классов.

На основании анализа литературы и источников по проблеме были выделены алгоритмы классификации, которые широко применяются для анализа больших объемов данных:

- алгоритмы метода ближайших соседей;
- алгоритм классификации «Случайный лес»;
- алгоритм классификации по методу стохастического градиента.

Проанализируем и сравним свойства указанных алгоритмов на предмет эффективного решения задач классификации больших объемов данных.

1.1 Анализ алгоритмов метода ближайших соседей

Метод ближайших соседей – один из простейших метрических классификаторов, основанный на оценивании сходства объектов. Объект относится к тому классу, которому принадлежат ближайшие к нему объекты из обучающей выборки. Метод основан на предположении о том, что близким объектам в признаковом пространстве соответствуют похожие метки [15].

Постановка задачи: для нового объекта x метод предполагает найти ближайшие к нему объекты $x_1, x_2, \ldots, x_k$ и построить прогноз по их меткам.

1.1.1 Алгоритм *k* ближайших соседей

Метод ближайших соседей имеет несколько вариаций, для каждой из которой разработаны эффективные алгоритмы, представляющий интерес для исследования.

Рассмотрим формальное описание задачи классификации по данному

методу [4].

$$\hat{y} = \arg\max_{y} \sum_{k=1}^{K} \mathrm{I}(y_k == y) \quad (1)$$

Алгоритм может быть применим к многомерным выборкам. Для этого перед применением нужно определить метрику расстояний.

При реализации алгоритма k ближайших соседей (kNN) первым шагом является преобразование точек данных в векторы признаков или их математические значения [23].

Затем алгоритм работает, находя расстояние между математическими значениями этих точек.

Параметрами алгоритма являются значения *k* и метрика расстояний.

Самый распространенный способ найти это расстояние - это евклидово расстояние, вычисляемое по следующей формуле [16]:

$$d_E(x, y) = \sqrt{\sum_i (x_i - y_i)^2}\ , \quad (2)$$

где:

$x = (x_1, x_2, \ldots, x_m)$, $y = (y_1, y_2, \ldots, y_m)$ – векторы значений двух записей.

Графическое представление формулы (2) показано на рисунке 1.

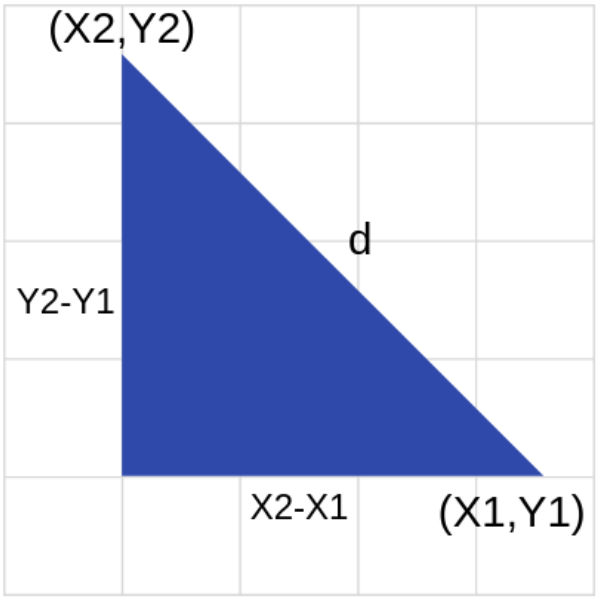

Рисунок 1 - Графическое представление евклидовой метрики

kNN использует эту формулу (2) для вычисления расстояния между каждой точкой данных и тестовыми данными. Затем он находит вероятность того, что эти точки похожи на тестовые данные, и классифицирует ее на основе того, какие точки имеют наивысшие вероятности.

Блок-схема алгоритма KNN представлена на рисунке 2.

Рисунок 2 – Блок-схема алгоритма kNN

kNN - это непараметрический алгоритм обучения, что означает, что он не делает никаких предположений относительно базовых данных.

Его также называют алгоритмом ленивого обучения, потому что он не учится сразу на обучающем наборе, вместо этого он сохраняет набор данных и во время классификации выполняет действие с набором данных.

Алгоритм kNN на этапе обучения просто сохраняет набор данных, и когда он получает новые данные, он классифицирует эти данные в категорию, которая очень похожа на новые данные.

– простота реализации;
– устойчивость к зашумленным обучающим данным;
– показывает высокая эффективность, если обучающие данные большие.

Основными недостатками алгоритма kNN являются постоянная необходимость определять значение k и относительно высокая стоимость вычислений, обусловленная вычислением расстояния между точками данных для всех обучающих выборок.

1.1.2 Алгоритм взвешенных k ближайших соседей

Взвешенный kNN - это модифицированная версия k ближайших соседей.

Как было отмечено выше, одна из проблем, влияющих на производительность алгоритма kNN, - это выбор параметра k. Если значение k слишком мало, алгоритм будет более чувствителен к выбросам. Если значение k слишком велико, соседство может включать слишком много точек из других классов.

Другой вопрос - это подход к объединению меток классов. Самый простой способ - получить большинство голосов, но это может быть проблемой, если ближайшие соседи сильно различаются по своему расстоянию, а ближайшие соседи более надежно указывают класс объекта.

При взвешенном способе во внимание принимается не только количество попавших в область определённых классов, но и их удалённость от нового значения.

Алгоритм взвешенных k ближайших соседей состоит из следующих шагов [22]:

Шаг 1. Пусть $L = \{(x_i, y_i), i = 1,\ldots, n\}$ будет обучающим набором наблюдений x_i с заданным классом y_i, и пусть x будет новым наблюдением (точкой запроса), метка класса у которого должна быть предсказана.

Шаг 2. Вычислить $d(x_i, x)$ для $i = 1,\ldots, n$, расстояние между точкой

запроса и любой другой точкой в обучающем наборе.

Шаг 3. Выбрать D' ⊆ D, набор из *k* ближайших точек обучающих данных к точкам запроса.

Шаг 4. Прогнозируем класс точки запроса, используя взвешенное по расстоянию голосование. Буква *v* представляет метки классов. Использовать следующую формулу:

$$y' = \arg\max_{v} \sum_{x_i, y_i \in D_z} w_i \times I(w = y_i) \quad (3)$$

С помощью метода можно вычислять значение одного из атрибутов классифицируемого объекта на основании дистанций от попавших в область объектов и соответствующих значений этого же атрибута у объектов:

$$x_k = \frac{\sum_{i=1}^{n} k_i d(x, a_i)^2}{\sum_{i=1}^{n} d(x, a_i)^2} \quad (4)$$

где:

a_i – *i*-й объект, попавший в область;

k_i – значение атрибута *k* заданного объекта;

x – новый объект;

x_k – *k*-й атрибут нового объекта.

Алгоритм взвешенных *k* ближайших соседей обладает всеми достоинствами обычного алгоритма kNN, но в тоже время лишен недостатка необходимости выбора параметра *k* [15, 23].

Следует отметить, что сложность обучения метода ближайших соседей равна O(n).

1.2 Анализ алгоритма «Случайный лес»

Случайный лес (Random forest) - это алгоритм обучения с учителем. «Лес», который он строит, представляет собой совокупность деревьев

решений, обычно обучаемых методом бэггинга.

«Общая идея метода бэггинга заключается в том, что сочетание обучающих моделей увеличивает общий результат.

Проще говоря: случайный лес строит несколько деревьев решений и объединяет их вместе, чтобы получить более точный и стабильный прогноз.

Одним из больших преимуществ случайного леса является то, что его можно использовать как для задач классификации, так и для задач регрессии, которые составляют большинство современных систем машинного обучения.

Рассмотрим применение случайный леса для классификации» [10].

«Учитывая обучающий набор $X = x_1, ..., x_n$ с ответами $Y = y_1, ..., y_n$, повторная упаковка (B раз) выбирает случайную выборку с заменой обучающей выборки и подгоняет деревья к этим образцам.

Для b =1,2,…, B:

1. Образец с заменой, n обучающих примеров из X, Y. Назовем их X_b, Y_b.

2. Обучить дерево классификации f_b на X_b, Y_b.

После обучения прогнозы для невидимых выборок x' могут быть сделаны путем получения большинства голосов в случае деревьев классификации» [19].

Пусть обучающая выборка состоит из N примеров, размер пространства признаков равен M, и задан параметр m.

В задачах классификации обычно:

$$m \approx \sqrt{M} \tag{5}$$

Алгоритм состоит из следующих шагов:

Шаг 1. Сгенерируем случайную подвыборку с повторением размером N из обучающей выборки. Таким образом, некоторые примеры попадут в неё несколько раз, а примерно $N/3$ примеров не войдут в неё вообще.

Шаг 2. Построим решающее дерево, классифицирующее примеры данной подвыборки, причём в ходе создания очередного узла дерева будем выбирать признак, на основе которого производится разбиение, не из всех M признаков, а лишь из m случайно выбранных. Выбор наилучшего из этих m

признаков может осуществляться различными способами. В оригинальном коде Бреймана используется критерий Гини, применяющийся также в алгоритме построения решающих деревьев CART. В некоторых реализациях алгоритма вместо него используется критерий прироста информации.

«Шаг 3. Дерево строится до полного исчерпания подвыборки и не подвергается процедуре прунинга (в отличие от решающих деревьев, построенных по таким алгоритмам, как CART и ID3).

Классификация объектов проводится путём голосования: каждое дерево комитета относит классифицируемый объект к одному из классов, и побеждает класс, за который проголосовало наибольшее число деревьев.

На рисунке 3 представлен пример дерева классификации, построенного по данному методу» [19].

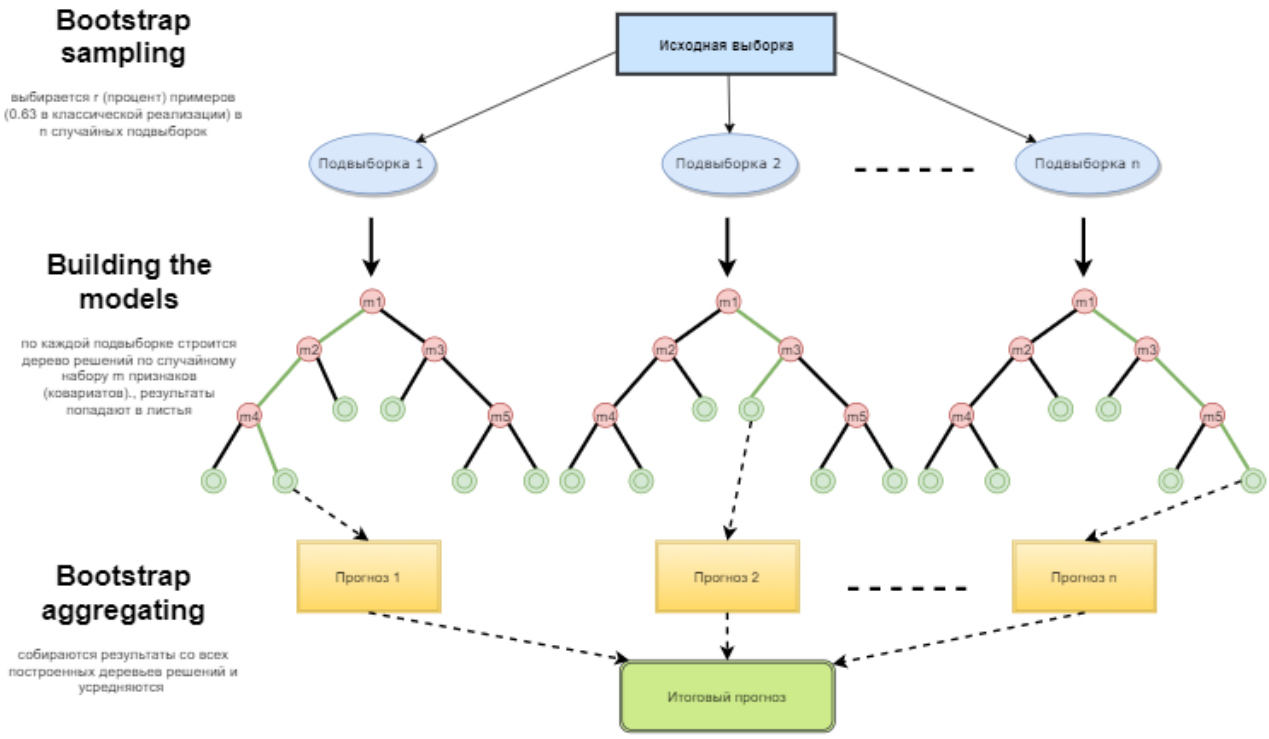

Рисунок 3 – Пример дерева классификации, построенного по данному методу «Случайный лес»

Оптимальное число деревьев подбирается таким образом, чтобы

минимизировать ошибку классификатора на тестовой выборке. В случае её отсутствия, минимизируется оценка ошибки out-of-bag: доля примеров обучающей выборки, неправильно классифицируемых комитетом, если не учитывать голоса деревьев на примерах, входящих в их собственную обучающую подвыборку.

Основными преимуществами метода «Случайный лес» являются [20]:

- уменьшается переоснащение в деревьях решений, что помогает повысить точность;
- гибок как для классификации, так и для задач регрессии;
- хорошо работает как с категориальными, так и с непрерывными значениями;
- нормализация данных не требуется, поскольку используется подход, основанный на правилах.

Основными недостатками алгоритма являются потребность в больших вычислительных мощностях и ресурсов, поскольку строит многочисленные деревья для объединения их результатов.

Кроме того, случайный требует много времени на обучение, так как объединяет множество деревьев решений для определения класса.

1.3 Анализ алгоритма классификации по методу стохастического градиента

Градиентные методы- это широкий класс оптимизационных алгоритмов, используемых не только в машинном обучении.

Рассмотрим пример применения градиентного подхода в качестве способа подбора вектора синаптических весов w в линейном классификаторе.

Пусть y^*: $X \to Y$ - целевая зависимость, известная только на объектах обучающей выборки:

$$X^I = (x_i, y_i)_{i=1}^I,\ y_i = y^*(x_i) \qquad (6)$$

Требуется найти алгоритм $a(x, w)$, который аппроксимирует зависимость

y^*.

Для линейного классификатора данный алгоритм описывается следующим образом:

$$a(x,w) = \varphi(\sum_{j=1}^{n} w_j x^j - w_0), \quad (7)$$

где:

φ(z) – функция активации, имеющая вид φ(z)=sing(z).

Согласно принципу минимизации эмпирического риска нужно решить задачу оптимизации вида:

$$\mathrm{Q}(w) = \sum_{i=1}^{I} L(a(x_i, w), y_i) \to \min_w \quad (8)$$

где:

$L(a,y)$ – заданная функция потерь.

«Для минимизации применим метод градиентного спуска.

Это пошаговый алгоритм, на каждой итерации которого вектор w изменяется в направлении наибольшего убывания функционала Q:

$$w := w - \eta \nabla Q(w) \quad (9)$$

Возможно 2 основных подхода к реализации градиентного спуска:

- пакетный (batch), когда на каждой итерации обучающая выборка просматривается целиком, и только после этого изменяется w. Это требует больших вычислительных затрат;
- стохастический, когда на каждой итерации алгоритма из обучающей выборки случайным образом выбирается только один объект. Таким образом вектор w настраивается на каждый вновь выбираемый объект» [21].

Алгоритм классификации по методу стохастического градиента (Stochastic gradient algorithm, SGA) представлен на рисунке 4 [12].

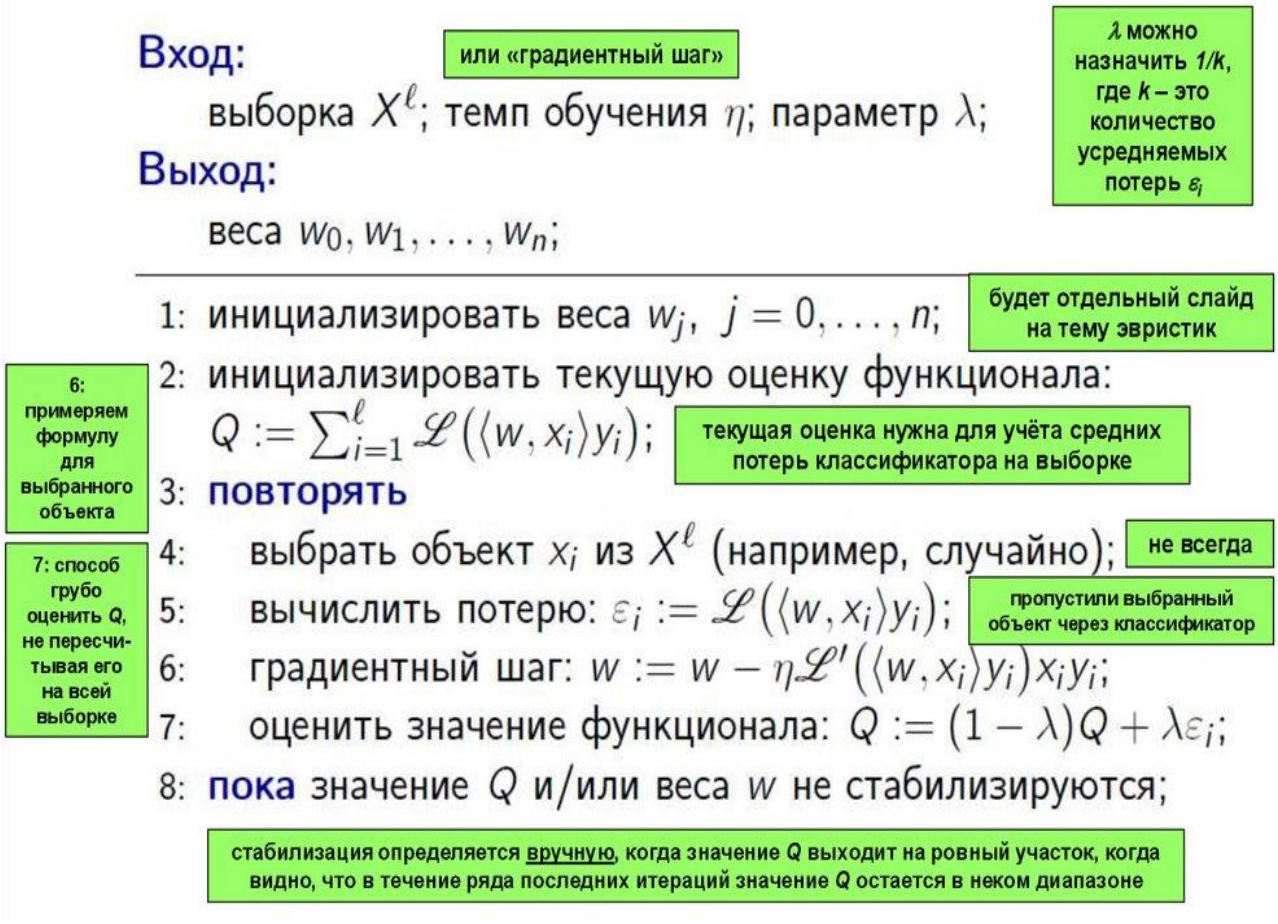

Рисунок 4 – Алгоритм классификации по методу стохастического градиента

Преимущества алгоритма SG [7]:

– небольшое потребление памяти, потому что сеть обрабатывает единственный обучающий пример;
– быстрое вычисление, так как одновременно обрабатывается только один образец;
– для больших наборов данных он может сходиться быстрее, так как это приводит к более частому обновлению параметров;
– из-за частых обновлений шаги, предпринимаемые в направлении минимумов функции потерь, имеют колебания, которые могут помочь выйти из локальных минимумов функции потерь (в случае, если вычисленное положение оказывается локальным минимумом).

Недостатки алгоритма SG:

– из-за частых обновлений шаги, предпринимаемые в направлении минимумов, вызывают шумы. Это может отклонить градиентный спуск в другие направления;
– из-за шумных шагов может потребоваться больше времени для достижения сходимости к минимумам функции потерь;
– частые обновления требуют больших вычислительных ресурсов из-за использования всех ресурсов для обработки одной обучающей выборки за раз.

Для сравнения рассмотренных алгоритмов классификации используем таблицу 1.

Критерии оценивания:

0 – полное несоответствие требованиям;

1 – значительное несоответствие требованиям;

2 – незначительное несоответствие требованиям;

3 – полное соответствие требованиям.

Таблица 1 – Сравнительный анализ алгоритмов классификации больших объемов данных

Характеристика/балл	Метод ближайших соседей	Случайный лес	Метод стохастического градиента
оптимизация	3	2	3
временная сложность	3	2	3
емкостная сложность	2	1	3
точность	1	3	2
простота реализации	3	2	1
Итого	12	10	12

По результатам сравнительного анализа алгоритмов классификации построена диаграмма, представленная на рисунке 5.

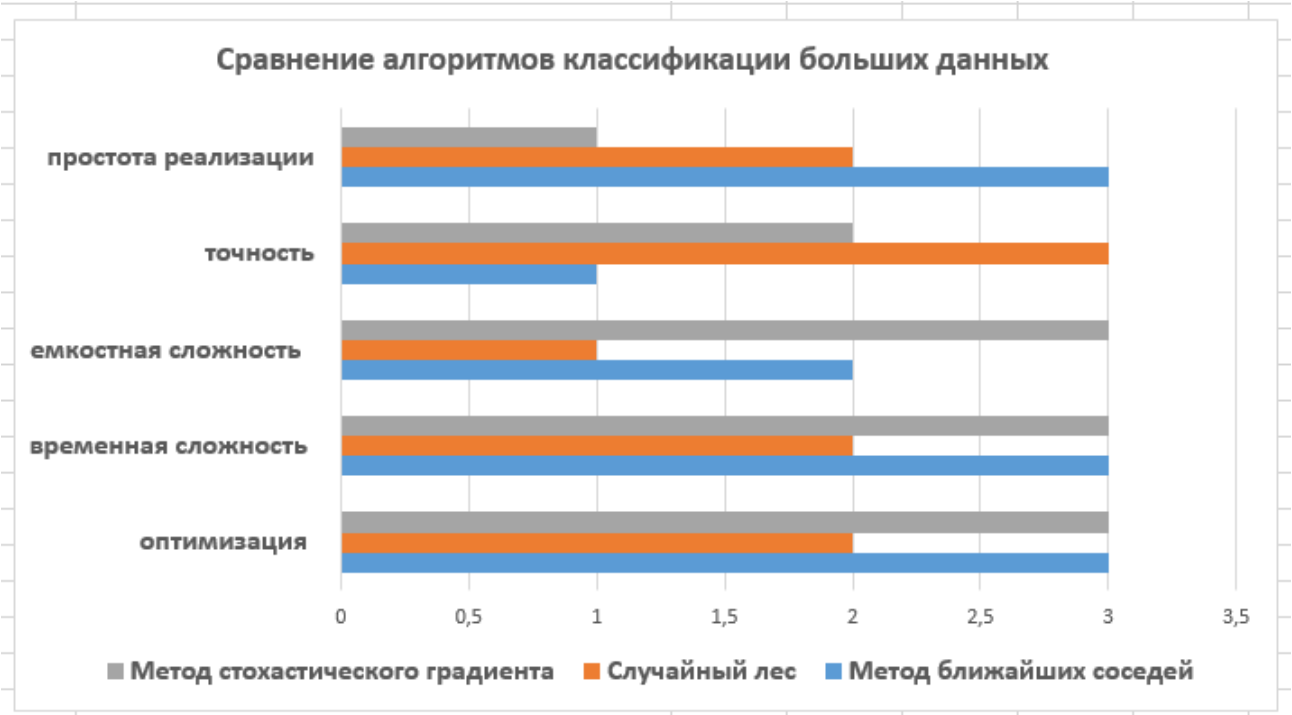

Рисунок 5 – Диаграмма сравнения характеристик алгоритмов классификации больших объемов данных

Как показал сравнительный анализ, высокую эффективность классификации больших данных обеспечивают алгоритмы ближайших соседей и стохастического градиента.

С точки зрения реализации более прост алгоритм ближайших соседей.

Выводы к главе 1

Первая глава посвящена исследованию алгоритмов классификации больших объемов данных.

Результаты проделанной работы позволили сделать следующие выводы.

На основании анализа литературы и источников по проблеме были выделены следующие алгоритмы классификации, которые используются для анализа больших объемов данных.

Как показал сравнительный анализ, высокую эффективность классификации больших данных обеспечивают алгоритмы ближайших соседей и стохастического градиента.

Глава 2 Анализ алгоритмов кластеризации больших объемов данных

Кластерный анализ – это многомерная статистическая процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы [8].

На основании анализа литературы и источников по проблеме были выделены алгоритмы кластеризации, которые широко применяются для анализа больших объемов данных:

- алгоритм k-means;
- алгоритм Борувки;
- алгоритм иерархической кластеризации.

Проанализируем и сравним свойства указанных алгоритмов на предмет эффективного решения задач кластеризации больших объемов данных.

2.1 Анализ алгоритма k-means

Исходные данные для алгоритма k-means:

- обучающая выборка X^m, состоящая из объектов $x_1 ... x_m$ с одинаковым набором атрибутов. Для всех объектов должны быть известны значения атрибутов $P_1, ..., P_n$:

$$x_i = (P_1, P_2, ..., P_n), \quad (10)$$

где n – количество атрибутов.

Атрибуты могут быть числовыми или категориальными. Таким образом, обучающая выборка задается следующим образом:

$$X^m = \{x_1, ..., x_m\}; \quad (11)$$

- метрика $\rho(x, x')$ расчёта расстояний между объектами. Можно использовать одну из известных метрик – Евклида, Чебышева,

расстояние Манхэттена и др.

Евклидово расстояние (норма/метрика $\|x\|_2$). Евклидово расстояние между двумя объектами, один из которых описывается вектором *x*, а второй - вектором x', будет рассчитываться так:

$$\|x - x'\|_2 = ((P_1 - P_1')^2 + (P_2 - P_2')^2 + ... + (P_n - P_n')^2)^{\frac{1}{2}}. \quad (12)$$

Множество точек, равноудаленных от некоторого центра при использовании евклидовой метрики будет образовывать круг в двумерном пространстве.

Расстояние Манхэттена (норма/метрика $\|x\|_1$). Данная норма имеет следующий вид:

$$\|x - x'\|_1 = (|P_1 - P_1'| + |P_2 - P_2'| + ... + |P_n - P_n'|) \quad (13)$$

Преимущество метрики $\|x\|_1$ заключается в том, что ее использование позволяет снизить влияние аномальных значений на работу алгоритмов. Множество точек, равноудаленных от некоторого центра при использовании метрики Манхэттена будет образовывать квадрат в двумерном пространстве.

Расстояние Чебышева (норма/метрика $\|x\|_\infty$). Данная норма имеет следующий вид:

$$\|x - x'\|_\infty = \max(|P_1 - P_1'|, |P_2 - P_2'|, ..., |P_n - P_n'|) \quad (14)$$

Множество точек, равноудаленных от некоторого центра при использовании метрики Чебышева будет образовывать квадрат в двумерном пространстве;

- количество k кластеров, которое должно быть сформировано из объектов исходной выборки.

Блок-схема алгоритма k-means представлена на рисунке 6.

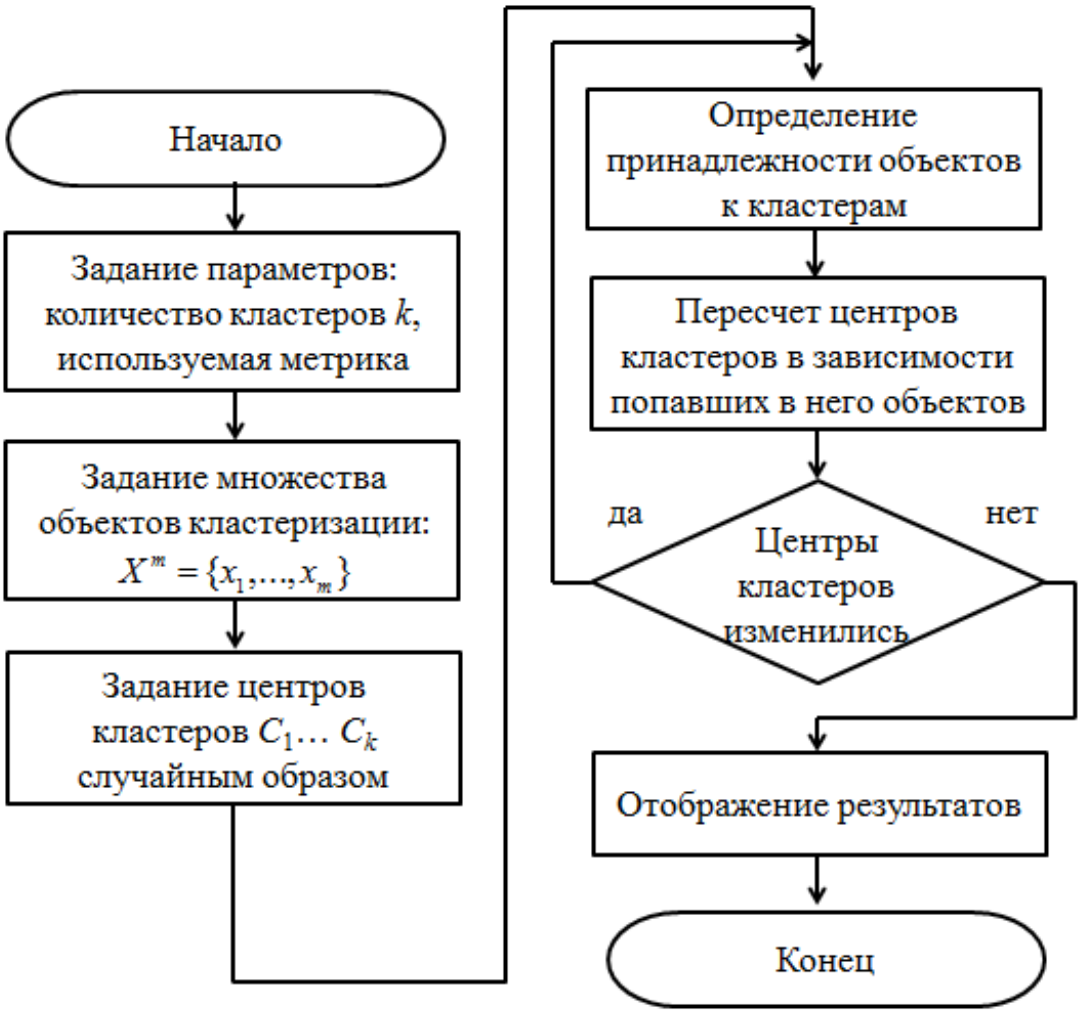

Рисунок 6 – Блок-схема алгоритма k-means

«Алгоритм состоит из следующих шагов:

Шаг 1. Случайным образом выбирается *k* объектов обучающей выборки, которые будут служить начальными центрами кластеров.

Шаг 2. Для каждого объектов обучающей выборки определяется ближайший к ней центр кластера. Для этого вычисляется расстояние между объектами и центрами кластеров. Считается, что объект принадлежит тому кластеру, к которому он ближе. В качестве формулы для оценки близости объектов в многомерном пространстве признаков используется одна из известных метрик» [14].

Шаг 3. Как только состав кластеров на данной итерации известен, производится расчёт новых центров кластеров. Это делается путем расчета

средних значений для каждого числового признака по всем объектам рассматриваемого кластера. Например, в двухмерном пространстве координаты центр кластера на основе вошедших в него t объектов рассчитывается следующим образом:

$$(P_{2ц}, P_{1ц}) = \left(\frac{\sum_{1}^{t} P_1(t)}{t}, \frac{\sum_{1}^{t} P_2(t)}{t} \right) \quad (14)$$

«Шаг 4. Шаги 2 и 3 повторяются до тех пор, пока не выполниться один из двух критериев остановки:

- границы кластеров и расположения центров кластеров не перестанет изменяться от итерации к итерации, т.е. на каждой итерации в каждом кластере будет оставаться один и тот же набор записей. На практике алгоритм k-means обычно находит набор стабильных кластеров за несколько десятков итераций;
- достигнут критерий сходимости. Чаще всего используется критерий суммы квадратов ошибок между центром кластера и всеми вошедшими в него объектами:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2 \quad (15)$$

где $p \in C_i$ - произвольная точка данных, принадлежащая кластеру C_i, m_i – центр данного кластера. Иными словами, алгоритм остановится тогда, когда ошибка E достигнет достаточно малого значения» [14].

Графическое представление алгоритма k-means показано на рисунке 7.

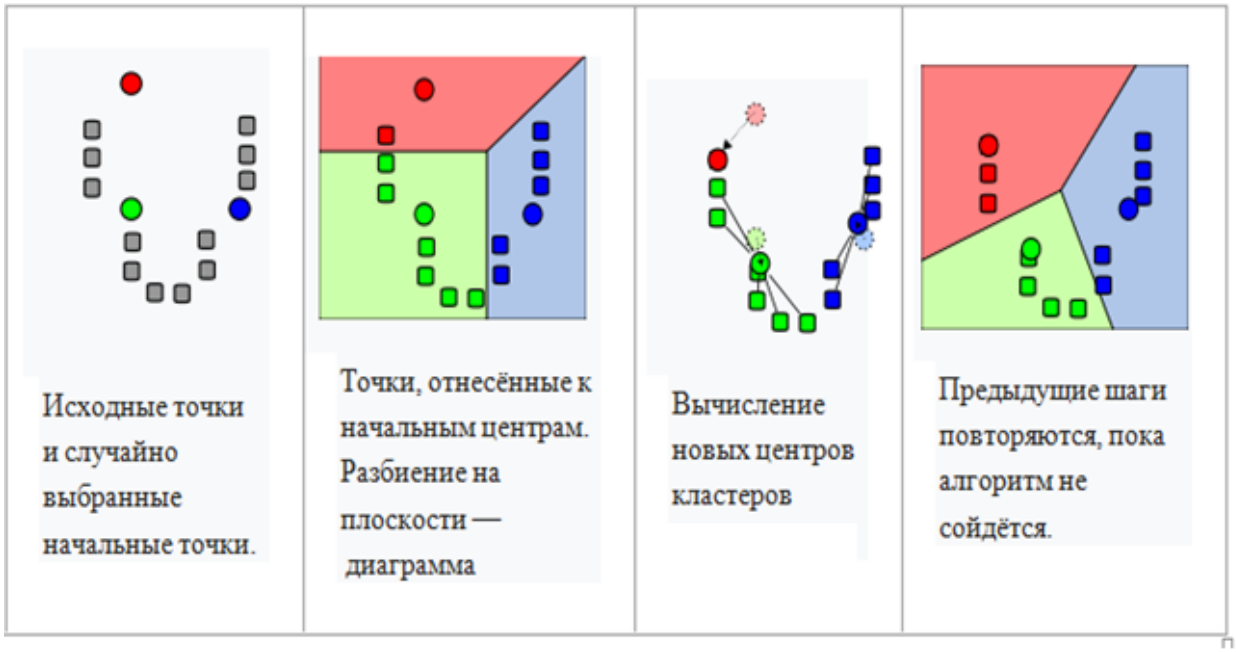

Рисунок 7 - Графическое представление алгоритма k-means

Сложность алгоритма по времени для некоторых множеств равна $O(2^{\Omega(\sqrt{n})})$.

К достоинствам алгоритма кластеризации данных k-means можно отнести следующее:

- умеренные вычислительные затраты, которые растут линейно с увеличением числа записей исходной выборки данных. Вычислительная сложность алгоритма определяется как k x n x l, где k – число кластеров, n – число записей и l – число итераций;
- результаты работы алгоритма не зависят от порядка следования записей в исходной выборке.

Среди недостатков алгоритма k-means выделяют чувствительность алгоритма к шумам и аномальным значениям в данных, поскольку они способны значительно повлиять на среднее значение, используемое при вычислении положений центров кластеров.

2.2 Анализ алгоритма Борувки

Алгоритм Борувки - это жадный алгоритм поиска минимального

остовного дерева в графе или минимального остовного леса в случае несвязного графа.

Исходным данным алгоритма является связный граф.

Результатом выполнения – минимальный остовный лес (Minimum Spanning Forests, MST).

Постановка задачи:

Пусть $G = <V, E>$ - связный граф, причем:

w: $E \rightarrow R$ - вещественные веса ребер графа G;

n и m – вершины и ребра графа G, соответственно.

Остовное дерево в графе G - это ациклический подграф G, который включает каждую вершину G и связан;

Каждое остовное дерево имеет ровно $(n-1)$ ребер.

Минимальный остовный лес (МОЛ) - это остовное дерево минимального веса, который определяется как сумма весов всех его ребер.

Наша задача - найти МОЛ для графа G.

Алгоритм Борувки основан на следующей лемме [17].

Лемма: Пусть $v \in V$ - любая вершина в графе G. МОЛ для G должно содержать ребро (v, w), которое является ребром минимального веса, инцидентным на v.

Доказательство: предположим, что (v, w) - ребро минимального веса, инцидентное на v, которое не содержится в МОЛ T графа G.

Тогда у нас должно быть другое ребро (v, u) в T такое, что вершина v может быть покрыта в T. Добавляя (v, w) в T, мы сформируем путь цикла в T, который проходит через v.

Поскольку (v, w) является ребром минимального веса, инцидентным на v, мы можем удалить (v, u) и, таким образом, сохранить (v, w) в МОЛ T', в результате чего веса T' будут меньше веса T. Это приводит к противоречию с тем, что T является МОЛ графа G.

Следовательно, МОЛ для графа G должно содержать ребро (v, w).

Определение: Основная идея в алгоритме Борувки состоит в том, чтобы одновременно сжать ребра минимального веса, инцидентные каждой из вершин в графе *G*. При сокращении необходимо сохранить только ребро минимального веса из любого набора кратных ребер. Процесс сжатия ребра минимального веса для каждой вершины в графе называется фазой Борувки.

Иными словами, основная идея алгоритма - начать с группы деревьев, каждая вершина которой представляет изолированное дерево.

Затем нам нужно продолжать добавлять ребра, чтобы уменьшить количество изолированных деревьев, пока мы не получим одно связное дерево.

Алгоритм Борувки состоит из следующих шагов [9].

Шаг 0. Создайте график.

Шаг 1. Начните с группы несвязанных деревьев (количество деревьев = количество вершин).

Шаг 2: пока есть несвязанные деревья, для каждого несвязанного дерева:

- найти край с меньшим весом;
- добавьте это ребро, чтобы соединить другое дерево.

Результаты пошагового выполнения алгоритма представлены на рисунке 8.

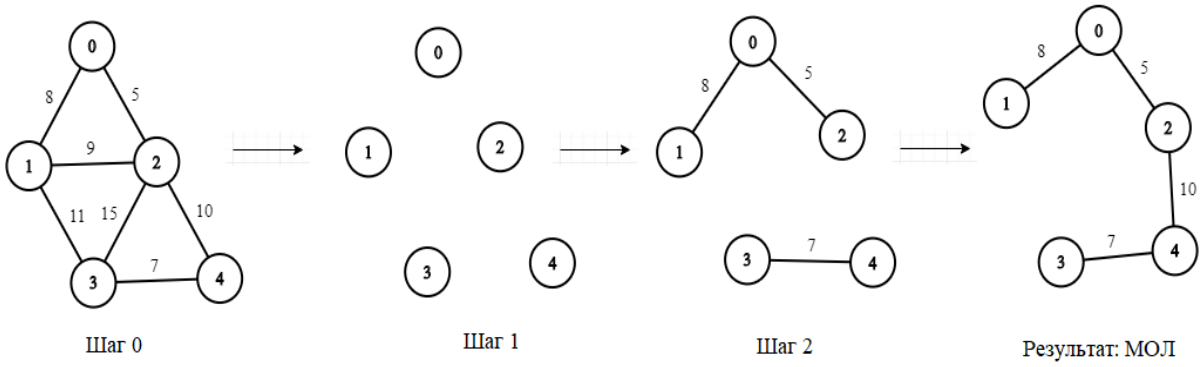

Рисунок 8 - Результаты пошагового выполнения алгоритма Борувки

Преимуществами алгоритма Борувки являются:

- самый теоретический параллелизм среди классических алгоритмов МОЛ;

− при тщательной реализации может выполнять меньше работы на каждом ребре, чем другие алгоритмы МОЛ.

К недостаткам алгоритма относятся:

− много синхронизации (множество раундов, каждый из которых имеет несколько точек синхронизации);

− снижение пропускной способности памяти и ограничение параллелизма, если граф слишком большой.

Следует также отметить, что общая временная сложность алгоритма Борувки составляет $O(E \log V)$.

2.3 Анализ алгоритма иерархической кластеризации

Иерархическая кластеризация, также известная как иерархический кластерный анализ, представляет собой алгоритм, который группирует похожие объекты в группы, называемые кластерами. Конечная точка - это набор кластеров, где каждый кластер отличается от другого кластера, а объекты внутри каждого кластера в целом похожи друг на друга.

Методы иерархической кластеризации подразделяются на агломеративные и дивизивные.

Агломеративные методы первоначально рассматривают каждую точку данных как отдельный кластер и на каждом шаге объединяйте ближайшие пары кластера (восходящий метод). Сначала каждый набор данных рассматривается как отдельный объект или кластер. На каждой итерации кластеры объединяются с разными кластерами, пока не образуется один кластер [13].

Классический агломеративный алгоритм имеет вид:

Шаг 1. Вычислить сходство одного кластера со всеми другими кластерами (вычислить матрицу близости).

Шаг 2. Рассматривайте каждую точку данных как отдельный кластер.

Шаг 3. Объедините кластеры, которые очень похожи или близки друг к

другу.

Шаг 4. Пересчитайте матрицу близости для каждого кластера.

Повторяйте шаги 3 и 4, пока не останется только один кластер.

Графическое представление данного алгоритма в виде дендограммы представлено на рисунке 9.

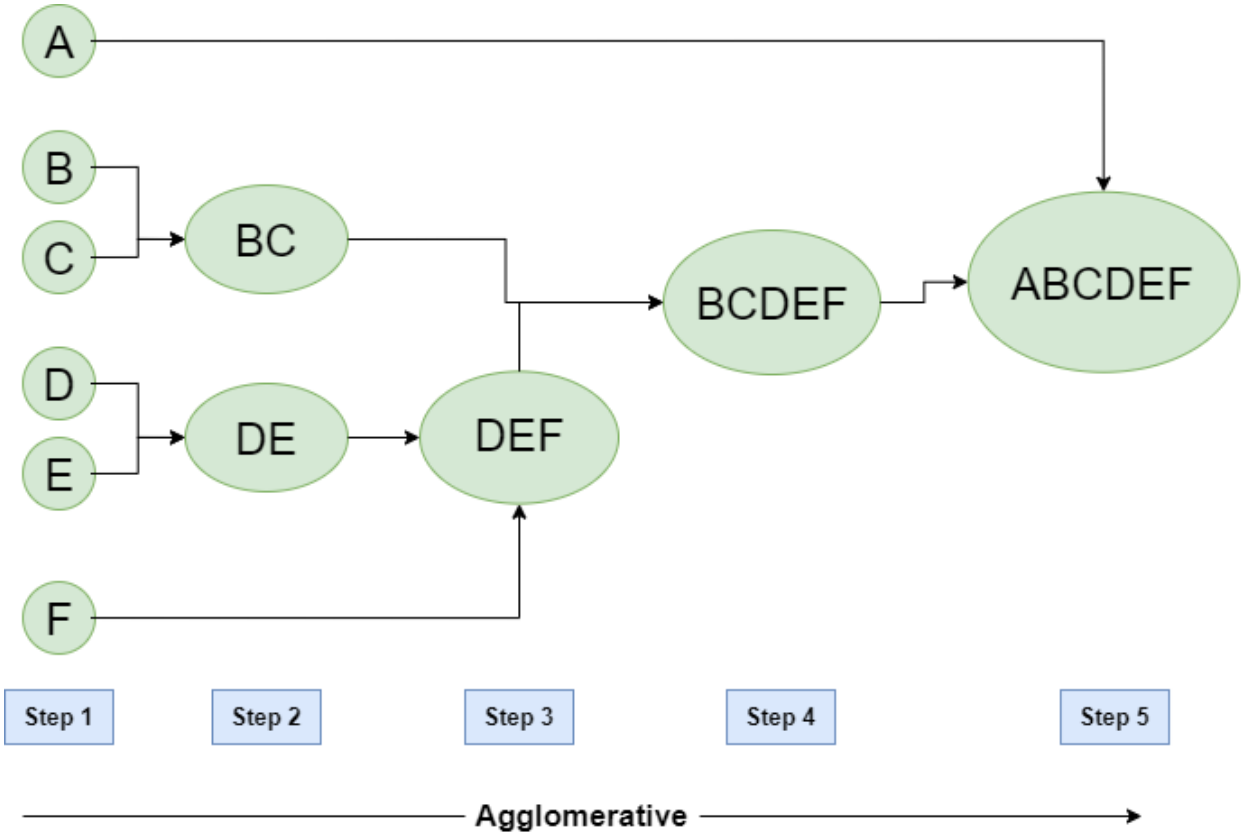

Рисунок 9 – Дендограмма агломеративной кластеризации

Следует отметить, что дивизивная иерархическая кластеризация - это полная противоположность агломеративной.

В дивизивной кластеризации мы учитываем все точки данных как единый кластер и на каждой итерации отделяем точки данных от кластеров, которые нельзя сравнивать. В итоге у нас осталось N кластеров (рисунок 10).

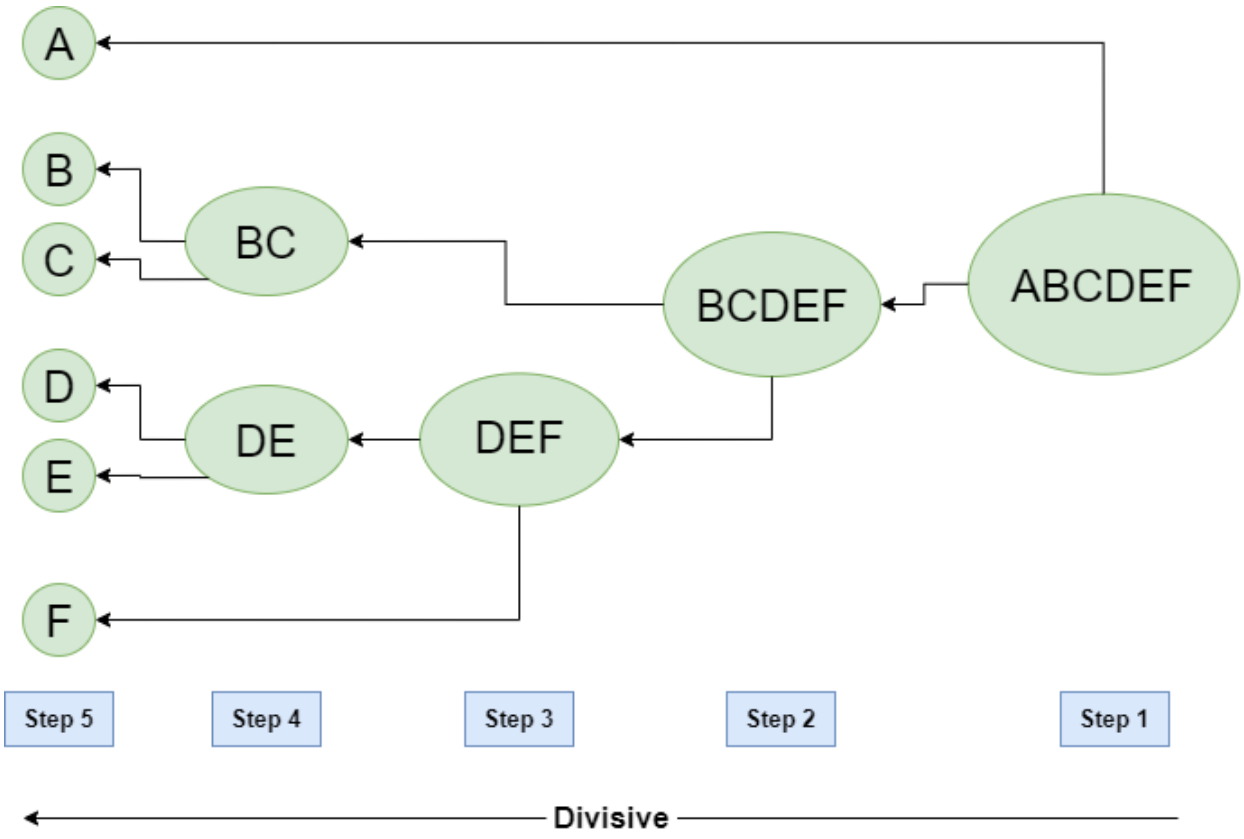

Рисунок 10 – Дендограмма дивизивной кластеризации

«Рассмотрим особенности иерархической кластеризации на примере алгоритма агломеративной кластеризации Ланса-Уильямса, который состоит из следующих шагов.

Шаг 1. Инициализировать множество кластеров C_1:

$$t := 1;\ C_t = \{x_1\}, \ldots, \{x_\ell\};$$

Шаг 2. Для всех $t = 2, \ldots, \ell$ (t - номер итерации):

Шаг 3. Найти в C_{t-1} два ближайших кластера:

$$(U,V) := \arg\min_{U \neq V} R(U,V);$$

$$R_t := R(U,V);$$

Шаг 4. Изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:

$$C_t := C_{t-1} \cup \{W\} \setminus \{U,V\}$$

Шаг 5. Для всех $S \in C_t$

Шаг 6. Вычислить расстояние R(W,S) по формуле Ланса-Уильямса;» [1]

Формула Ланса-Уильямса, обобщающая большинство способов определения этого расстояния, имеет вид:

$$R(U \cup V,S) = \alpha U \cdot R(U,S) + \alpha V \cdot R(V,S) + \beta \cdot R(U,V) + \gamma \cdot R(U,S) - R(V,S), \quad (16)$$

где:

αU, αV, β, γ — числовые параметры.

Частными случаями формулы Ланса-Уильямса являются:

- расстояние ближнего соседа:

$$R^{б}(W,S) = \min_{w \in W, s \in S} \rho(w,s), \quad (17)$$

причем:

$\alpha U = \alpha V = 1/2$, $\beta = 0$, $\gamma = -1/2$;

- расстояние дальнего соседа:

$$R^{д}(W,S) = \max_{w \in W, s \in S} \rho(w,s), \quad (18)$$

причем:

$\alpha U = \alpha V = 1/2$, $\beta = 0$, $\gamma = 1/2$;

- групповое среднее расстояние:

$$R^{г}(W,S) = \frac{1}{|W||S|} \sum_{w \in W} \sum_{s \in S} \rho(w,s), \quad (19)$$

причем:

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = \gamma = 0;$$

- расстояние между центрами:

$$R^{ц}(W,S) = \rho^2\left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|}\right), \quad (20)$$

причем:

$$\alpha_U = \frac{|U|}{|W|}, \alpha_V = \frac{|V|}{|W|}, \beta = -\alpha_U \alpha_V, \gamma = 0;$$

- расстояние Уорда:

$$R^{у}(W,S) = \frac{|S||W|}{|S|+|W|} \rho^2\left(\sum_{w \in W} \frac{w}{|W|}, \sum_{s \in S} \frac{s}{|S|}\right), \quad (21)$$

причем:

$$\alpha_U = \frac{|S|+|U|}{|S|+|W|}, \alpha_V = \frac{|S|+|V|}{|S|+|W|}, \beta = \frac{-|S|}{|S|+|W|}, \gamma = 0.$$

На рисунках 11-14 представлена визуализация кластерной структуры.

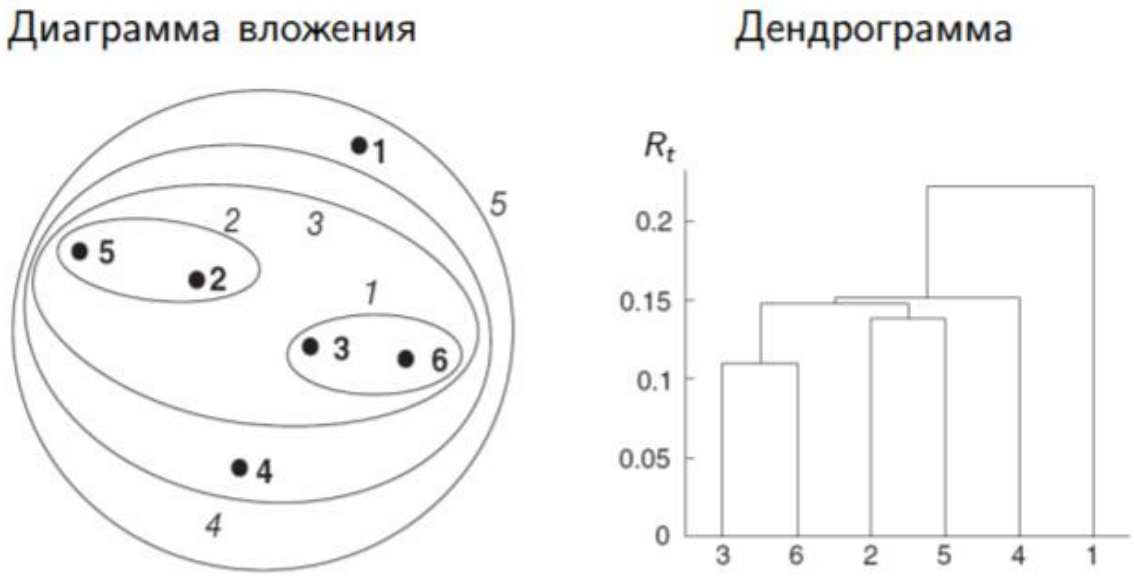

Рисунок 11 – Расстояние ближнего соседа

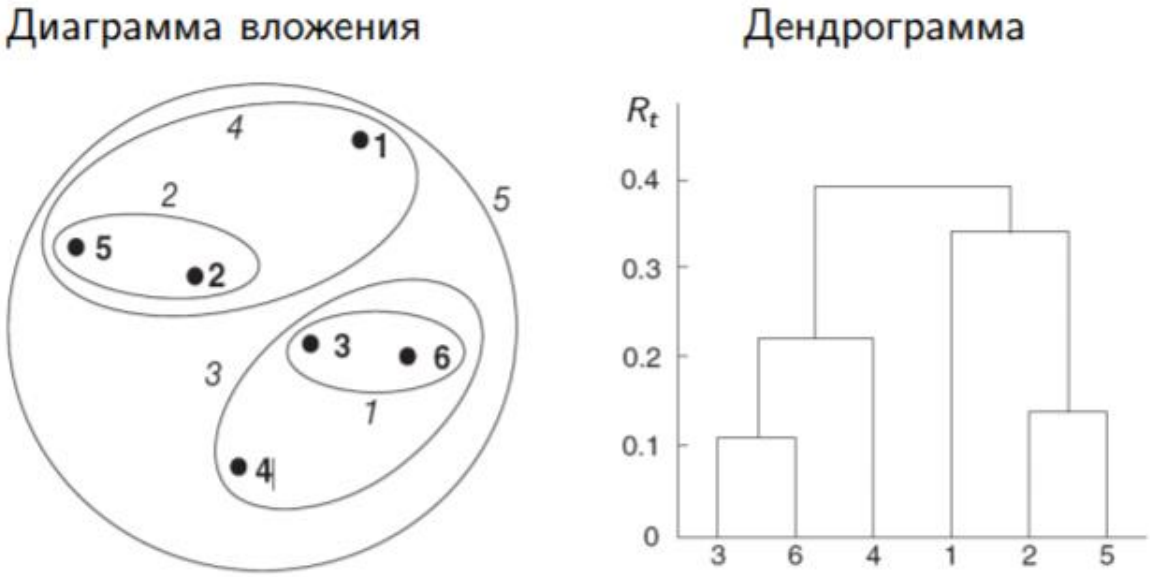

Рисунок 12 – Расстояние дальнего соседа

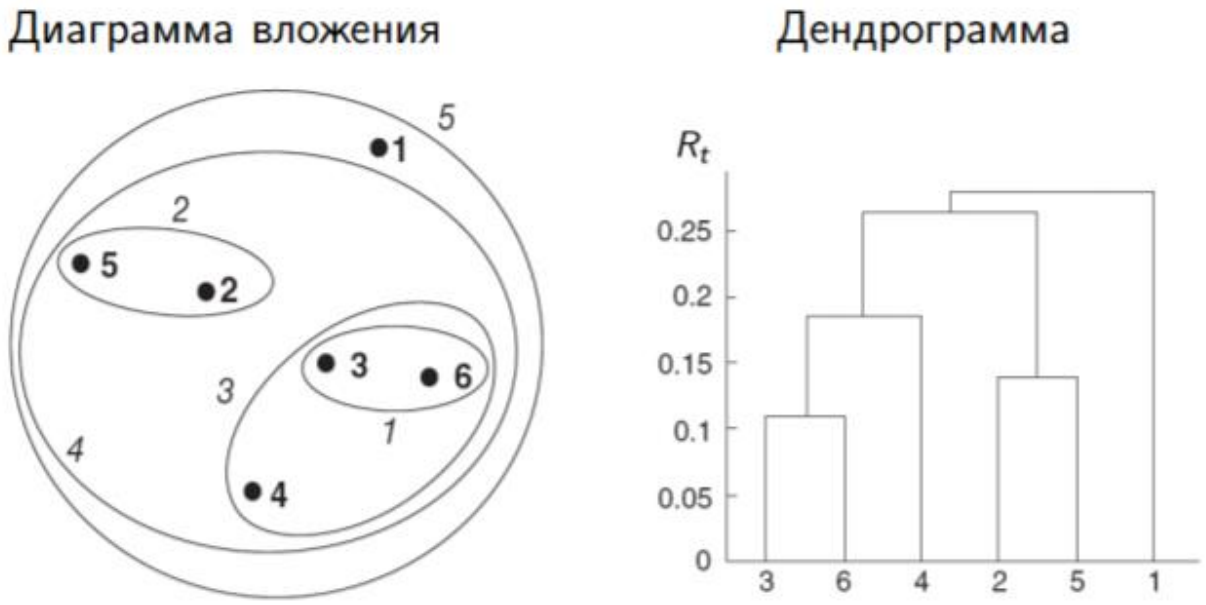

Рисунок 13 – Групповое среднее расстояние

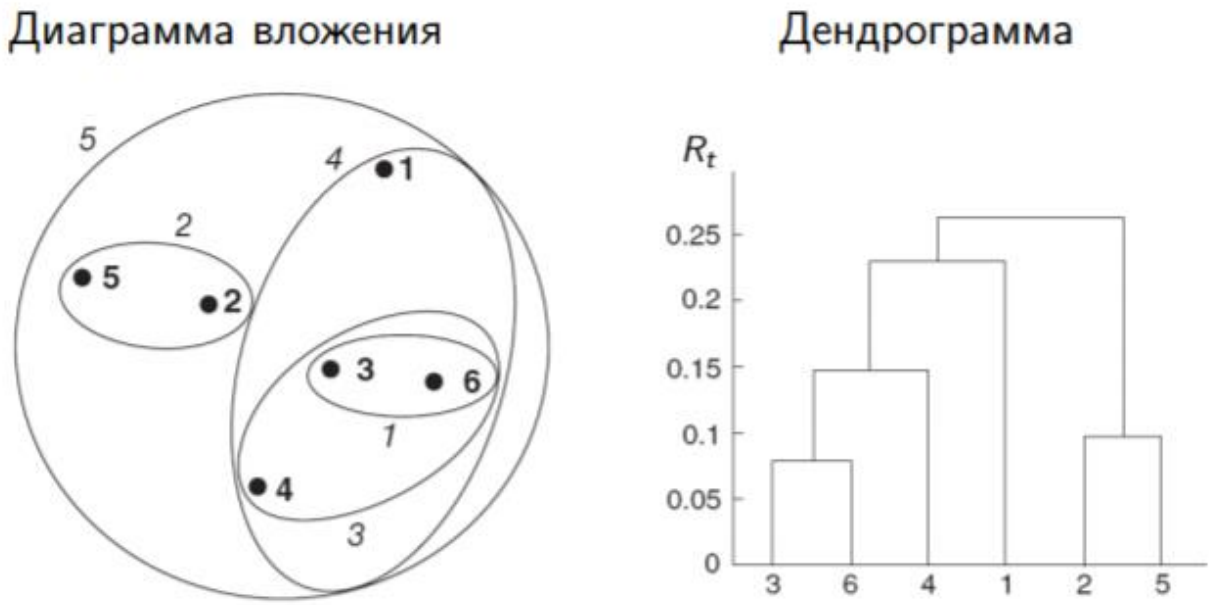

Рисунок 14 – Расстояние Уорда

Сложность алгоритма агломеративной кластеризации равна $O(l^2)$ при поиске ближайших кластеров и $O(l^3)$ – при построении всего дерева.

Для оптимизации алгоритма используется метод перебора всех пар.

Оптимизированный (редуктивный) алгоритм состоит из следующих шагов:

«Шаг 1. Инициализировать множество кластеров C_1:

$$t := 1;\ C_t = \{x_1\}, \ldots, \{x_\ell\};$$

Шаг 2. Выбрать начальное значение параметра δ;

Шаг 3. $P(\delta) := \{(U,V) \mid U,V \in C_t, R(U,V) \leq \delta\}$;

Шаг 4. Для всех $t = 2, \ldots, \ell$ (t - номер итерации):

Шаг 5. Если $P(\delta) = \emptyset$ то

Шаг 6. Увеличить δ так, чтобы $P(\delta) \neq \emptyset$;

Шаг 7. Найти в $P(\delta)$ пару ближайших кластеров:

$$(U,V) := \arg\min_{(U,V)\in P(\delta)} R(U,V);$$

$$R_t := R(U,V);$$

Шаг 8. Изъять кластеры U и V, добавить слитый кластер $W = U \cup V$:

$$C_t := C_{t-1} \cup \{W\} \setminus \{U,V\}$$

Шаг 9. Для всех $S \in C_t$

Шаг 10. Вычислить расстояние R(W,S) по формуле Ланса-Уильямса;

Шаг 11. Если $R(W,S) \leq \delta$ то

Шаг 12. $P(\delta) := P(\delta) \cup \{(W, S)\}$;» [1].

Для оптимизации алгоритма по скорости нужно периодически увеличивать параметр δ.

Как показал анализ, наиболее эффективным методом агломеративной кластеризации является метод Уорда.

Для сравнения рассмотренных алгоритмов кластеризации используем таблицу 2.

Критерии оценивания:

0 – полное несоответствие требованиям;

1 – значительное несоответствие требованиям;

2 – незначительное несоответствие требованиям;

3 – полное соответствие требованиям.

Таблица 2 – Сравнительный анализ алгоритмов кластеризации больших объемов данных

Характеристика/балл	Алгоритм k-means	Алгоритм Борувки	Агломеративная кластеризация
оптимизация	2	2	3
временная эффективность	3	2	2
точность	1	2	3
емкостная эффективность	2	1	3
простота реализации	3	2	2
Итого	11	9	13

По результатам сравнительного анализа алгоритмов кластеризации построена диаграмма, представленная на рисунке 15.

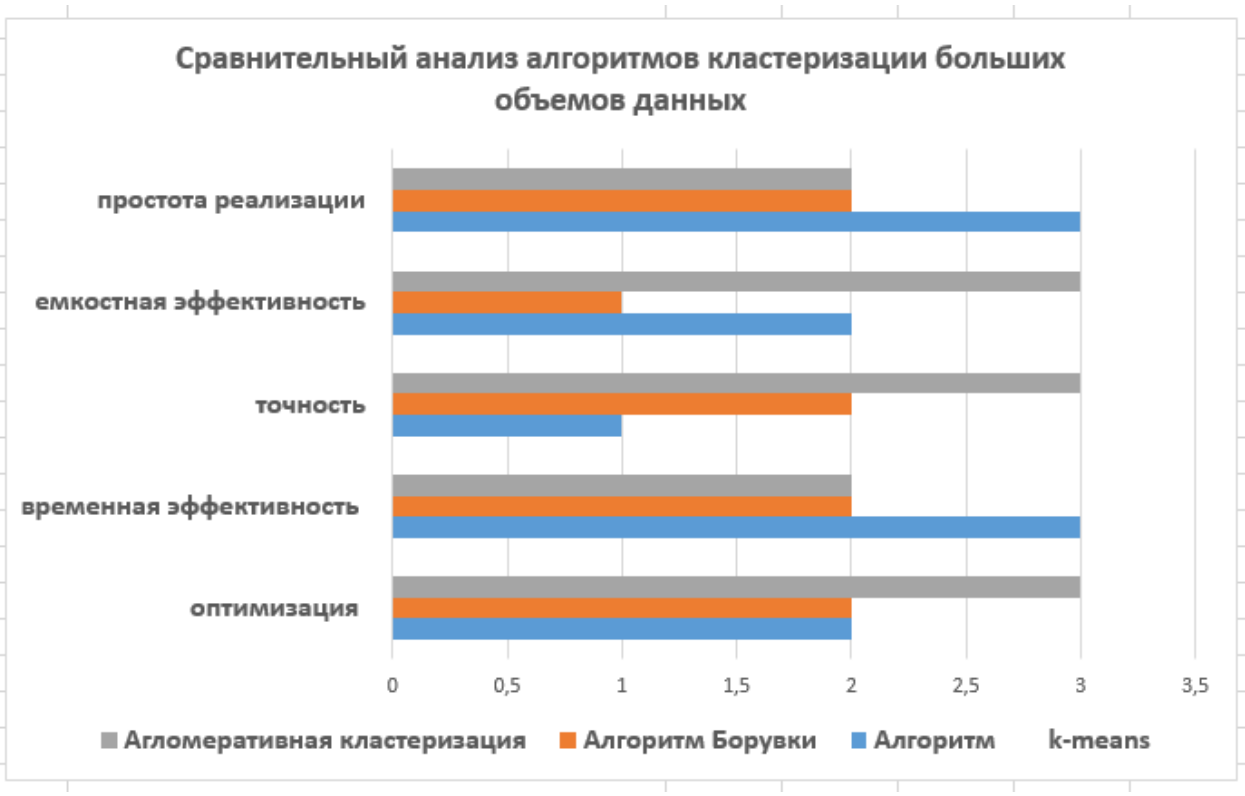

Рисунок 15 – Диаграмма сравнения характеристик алгоритмов кластеризации больших объемов данных

Как показал сравнительный анализ, высокую эффективность кластеризации больших данных обеспечивает алгоритм агломеративной кластеризации.

С точки зрения реализации более прост алгоритм k-means.

Выводы к главе 2

Вторая глава посвящена исследованию алгоритмов кластеризации больших объемов данных.

Результаты проделанной работы позволили сделать следующие выводы.

На основании анализа литературы и источников по проблеме были выделены следующие алгоритмы классификации, которые используются для анализа больших объемов данных:

– алгоритм k-means;

– алгоритм Борувки;

– алгоритмы агломеративной кластеризации.

Как показал сравнительный анализ, высокую эффективность кластеризации больших данных обеспечивает алгоритм агломеративной кластеризации.

С точки зрения реализации более прост алгоритм k-means.

Глава 3 Разработка программы классификации и кластеризации больших объемов данных

3.1 Выбор среды для разработки программы

Для разработки программы классификации и кластеризации больших объемов данных используем язык программирования Python и технологию Integrated Development Environment (IDE).

«IDE – интегрированная среда разработки представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

Интегрированная среда разработки позволяет программистам объединять различные задачи написания компьютерной программы.

Интегрированные среды разработки повышают продуктивность программиста за счет объединения общих действий по написанию программного обеспечения в одном приложении: редактирование исходного кода, создание исполняемых файлов и отладка.

В состав типовой интегрированной среды разработки входят:

- текстовый редактор;
- транслятор – компилятор или интерпретатор;
- средства автоматизации сборки;
- отладчик.

Рассмотрим функциональные и архитектурные особенности популярных интегрированных сред разработки» [2].

3.1.1 Интегрированная среда разработки Visual Studio + Python Tools for Visual Studio

Интегрированная среда разработки или IDE Visual Studio– это стартовая площадка для написания, отладки и сборки кода, а также последующей

публикации приложений.

Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки (рисунок 15).

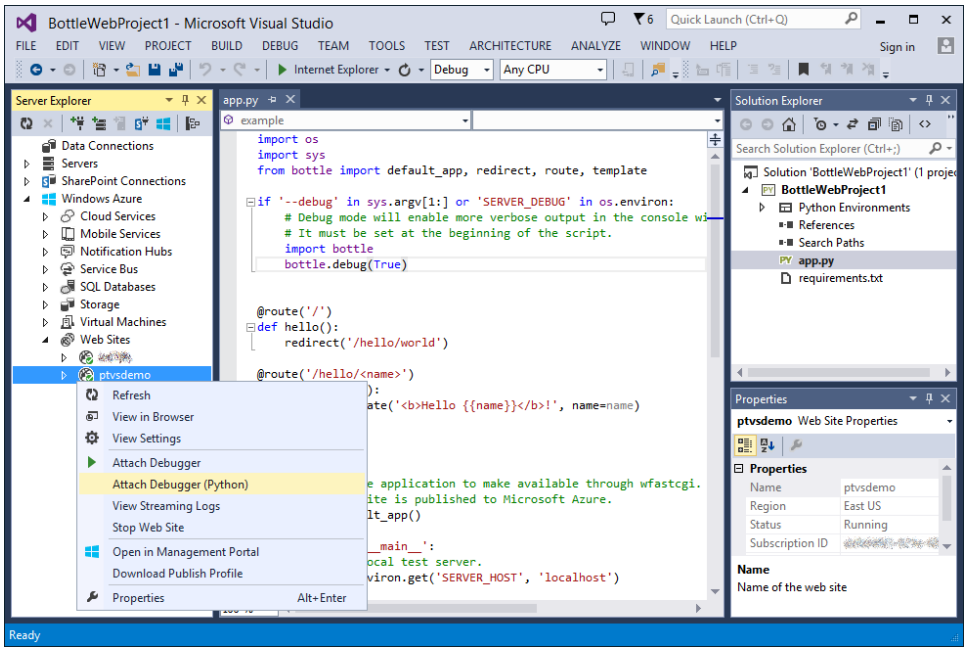

Рисунок 15 – Экран редактора Python Tools for Visual Studio

«Основными инструментами разработчика является следующие:

– обозреватель решений, который позволяет просматривать файлы кода, перемещаться по ним и управлять ими. Обозреватель решений позволяет упорядочить код путем объединения файлов в решения и проекты;
– редактор, отображающий содержимое файла. Здесь можно редактировать код или разрабатывать пользовательский интерфейс, например, окно с кнопками или текстовые поля;
– командный обозревать, который позволяет отслеживать рабочие элементы и использовать код совместно с другими пользователями с помощью технологий управления версиями, таких как Git и система управления версиями Team Foundation» [3].

Среда Visual Studio доступна для операционных систем Windows и Mac.

Возможности Visual Studio оптимизированы для разработки кроссплатформенных и мобильных приложений.

Существует три выпуска Visual Studio: Community (бесплатный), Professional и Enterprise.

3.1.2 Интегрированная среда разработки PyCharm

Одной из лучших полнофункциональных IDE, предназначенных именно для Python, является PyCharm. Существует как бесплатный open-source (Community), так и платный (Professional) варианты IDE.

PyCharm доступен на Windows, Mac OS X и Linux (рисунок 16).

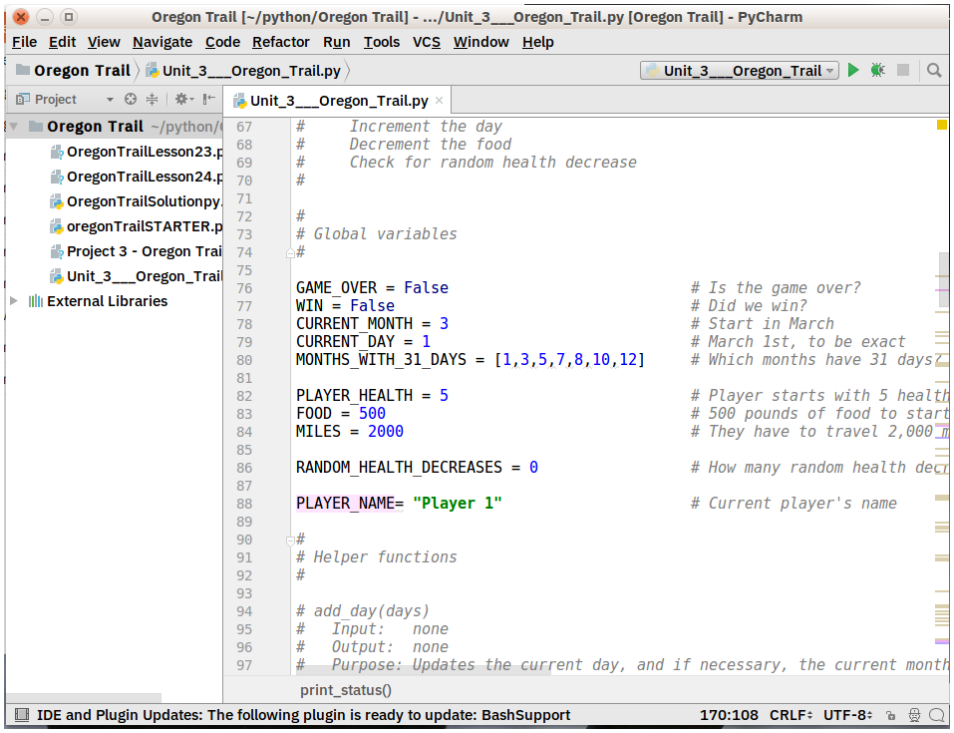

Рисунок 16 – Среда IDE PyCharm

Коробочный вариант PyCharm поддерживает разработку на Python напрямую — достаточно открыть новый файл и начать писать код.

Разработчик может запускать и отлаживать код прямо из PyCharm. Кроме того, в IDE есть поддержка проектов и системы управления версиями.

«Преимущества: это среда разработки для Python с поддержкой всего и вся и хорошим коммьюнити. В ней «из коробки» можно редактировать, запускать и отлаживать Python-код.

К недостаткам PyCharm можно отнести медленную загрузку. Кроме того, настройки по умолчанию, возможно, придётся подкорректировать для существующих проектов» [18].

3.1.3 Интегрированная среда разработки Eclipse + PyDEv

Интегрированная среда разработки Eclipse является бесплатной программной платформой с открытым исходным кодом, контролируется организацией Eclipse Foundation.

Среда написана на языке программирования Java.

Основной целью её создания является повышение продуктивности процесса разработки программного обеспечения.

Фрагмент среды представлен на рисунке 17.

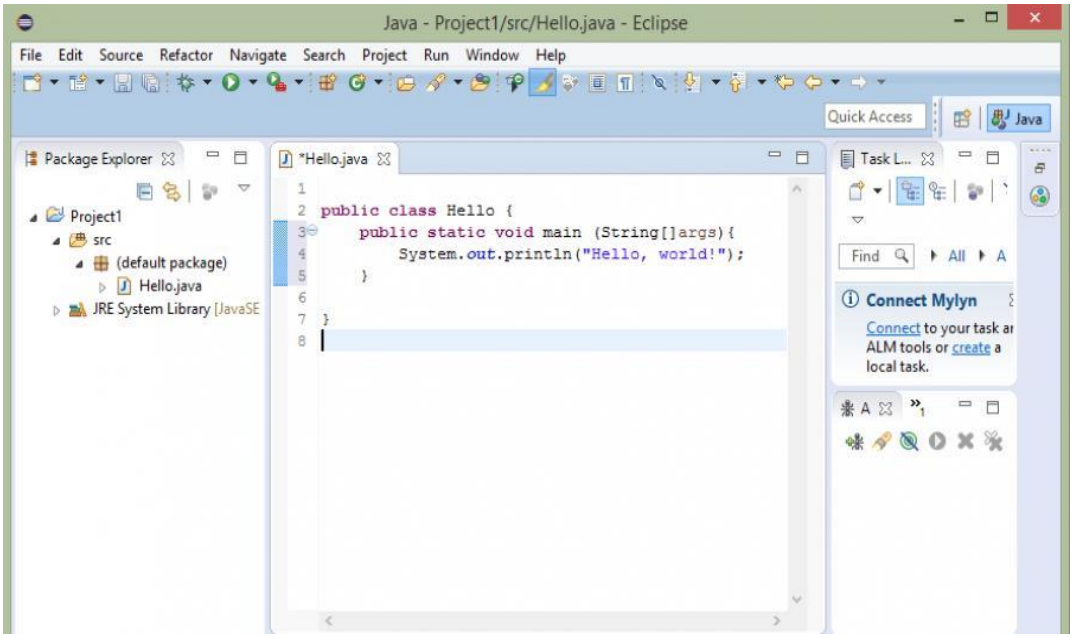

Рисунок 17 – Экран IDE Eclipse

«Основным компонентом является исполняемая среда – Eclipse Runtime, в которой выполняются коды расширений и модулей. Она обеспечивает всю базовую функциональность – управление расширениями и обновлениями, взаимодействие с операционной системой, обеспечение работы системы помощи.

Следующим ключевым компонентом является собственно интегрированная среда разработки – она отвечает за управление элементами программы, управление проектами, отладку и сборку проектов, поиск по файлам и командную программу» [11].

Eclipse претендует на статус наиболее популярной Java IDE и является единственным конкурентом такой мощной платформы, как NetBeans.

Но в отличие от последней, использующей для создания элементов пользовательского интерфейса платформонезависимую библиотеку Swing, Eclipse использует платформозависимую библиотеку SWT.

«Интегрированные среды разработки, созданные на платформе Eclipse, широко применяются для создания программного обеспечения на различных языках программирования.

Это обусловлено универсальностью Eclipse, работающей по принципу «Плагины для Eclipse разрабатываются в самой Eclipse.

Следует учесть, что в рамках проекта Eclipse Foundation предлагается несколько платформ для создания подключаемых модулей для настольных инструментов, распределенных сервисов и интерфейсов браузера.

Одним из таких расширений является PyDev, предоставляющий интерактивную консоль Python и возможности для отладки и автодополнения кода» [11].

Установить его просто: запустите Eclipse, выберите Help → Eclipse Marketplace, затем найдите PyDev. Нажмите «Install» и при необходимости перезапустите Eclipse (рисунок 18).

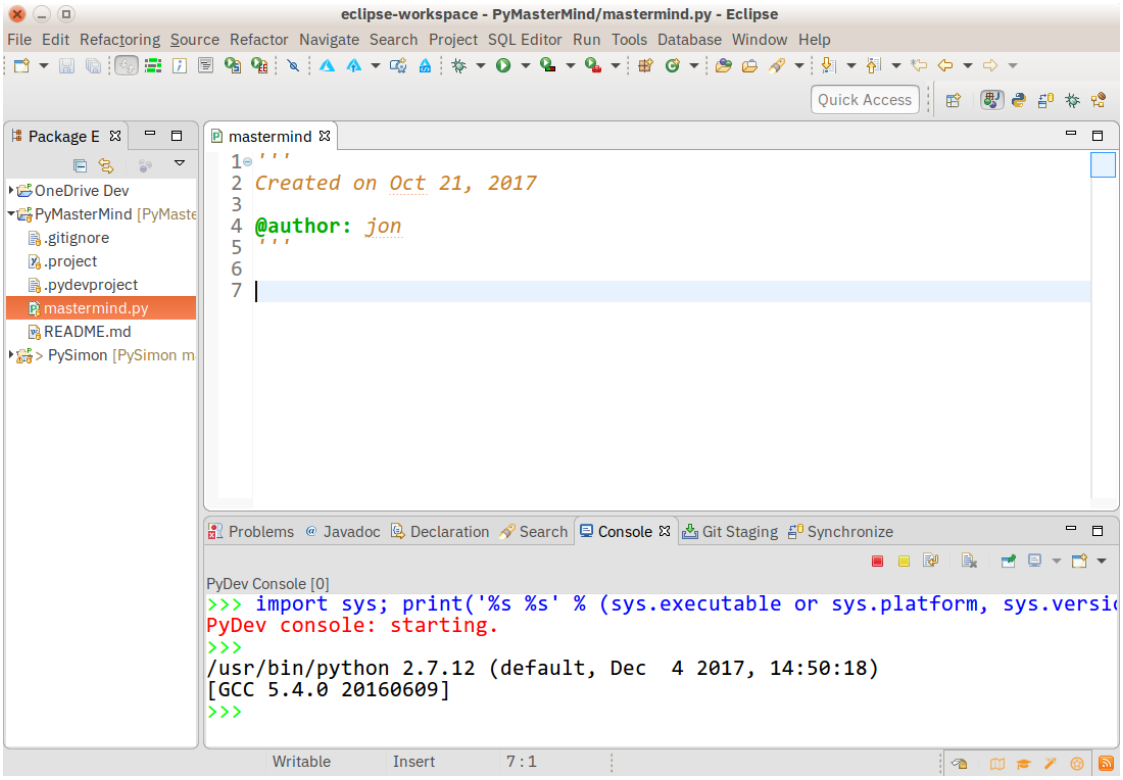

Рисунок 18 – Экран редактора языка Python

Установка PyDev пройдёт быстро и гладко. У опытного пользователя Eclipse не возникнет проблем с изучением этого расширения.

Для выбора интегрированной среды разработки используем таблицу 3, составленную на основе анализа блогов по данной тематике.

Критерии оценивания:

0 – полное несоответствие требованиям;

1 – значительное несоответствие требованиям;

2 – незначительное несоответствие требованиям;

3 – полное соответствие требованиям.

Таблица 3 – Сравнительный анализ интегрированных сред разработки

Характеристика/балл	Visual Studio	PyCharm	Eclipse+PyDEv
поддержка языка Python	2	3	2
юзабилити	3	2	3
предпочтение разработчика	1	2	3
Итого	6	7	8

Таким образом, наилучшими характеристиками обладает IDE Eclipse+PyDEv.

Выбираем IDE Eclipse+PyDEv в качестве среды для разработки программы.

3.2 Реализация алгоритмов классификации и кластеризации

Для разработки программной архитектуры программы реализации алгоритмов используем диаграмму компонентов UML [6].

Диаграмма компонентов UML показывает компоненты, предоставленные и требуемые интерфейсы, порты и отношения между ними.

Диаграммы компонентов предлагают аналитикам и разработчикам естественный формат для начала моделирования решения и позволяют проверить, требуются ли для системы дополнительные функциональные возможности.

Диаграмма компонентов имеет более высокий уровень абстракции, чем диаграмма классов UML (рисунок 19).

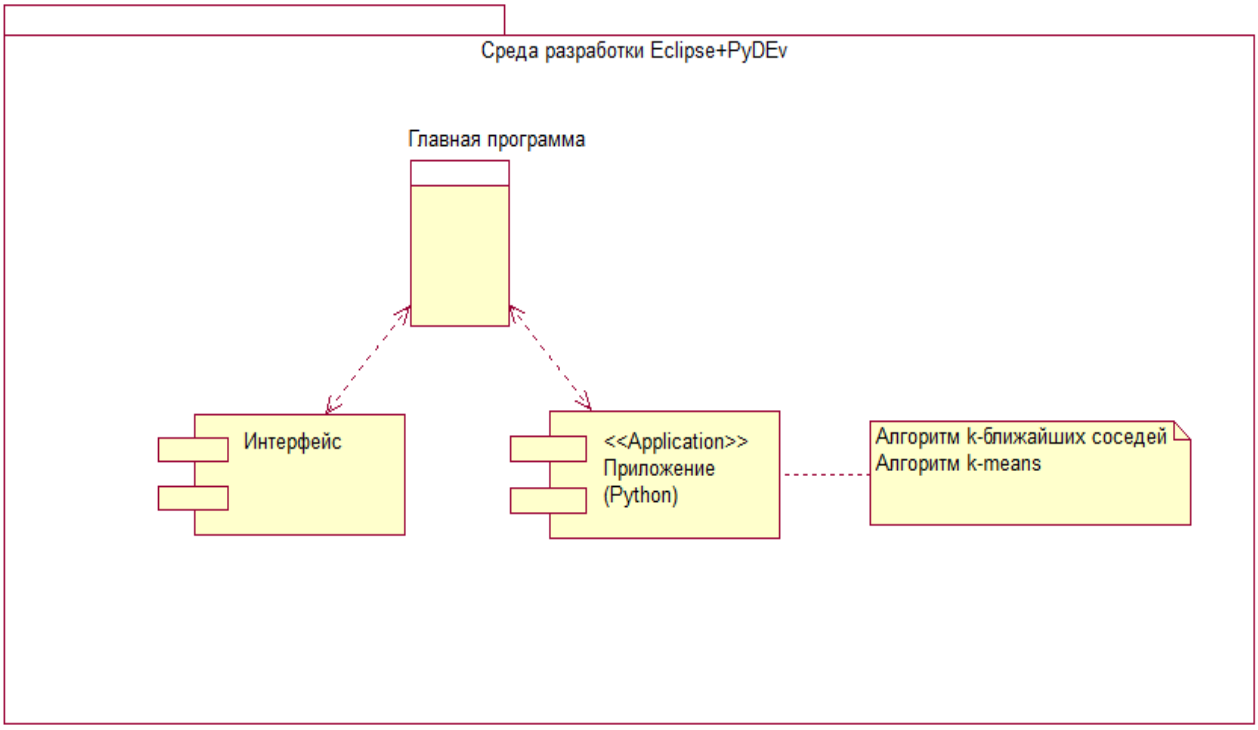

Рисунок 19 – Диаграмма компонентов программы

На языке Python в среде Eclipse+PyDEv реализованы эффективные алгоритмы классификации и кластеризации больших объемов данных.

Выполнена тестовая проверка работоспособности программы.

На рисунке 20 представлен программный код на языке Python, реализующий алгоритм классификации k ближайших соседей, и результаты выполнения алгоритма на тестовых данных.

```
from sklearn.metrics import classification_report
y_true = [0, 1, 2, 2, 2]
y_pred = [0, 0, 2, 2, 1]
target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
precision recall f1-score support

class 0 0.50 1.00 0.67 1
class 1 0.00 0.00 0.00 1
class 2 1.00 0.67 0.80 3

micro avg 0.60 0.60 0.60 5
macro avg 0.50 0.56 0.49 5
weighted avg 0.70 0.60 0.61 5
```

Рисунок 20 – Программный код и результаты выполнения алгоритма классификации

На рисунке 21 представлен программный код на языке Python, реализующий алгоритм кластеризации k-means, и результаты выполнения алгоритма на тестовых данных.

results

```
Cluster0(n=31675):
------------------
Cluster1(n=4989):
------------------
Cluster2(n=1):
------------------
Cluster3(n=671):
------------------
```

cluster	customer	orderRatio	itemsRatio	monetaryRatio	frequency
0	50854.809882	0.000000	0.000000	0.000000	0.000000
1	51332.535779	0.721604	0.453365	0.307721	1.097815
2	57044.000000	1.000000	2.000000	108.719154	1.000000
3	48516.023845	0.136277	0.078346	0.044497	4.271237

Рисунок 21 – Программный код и результаты выполнения алгоритма кластеризации

Таким образом, тестирование программы подтвердило правильность реализации эффективных алгоритмов классификации и кластеризации больших объемов данных.

Выводы к главе 3

Третья глава посвящена разработке классификации и кластеризации больших объемов данных.

Результаты проделанной работы позволили сделать следующие выводы.

Как показал анализ, наилучшими характеристиками для разработки программ на языке Python обладает IDE Eclipse+PyDEv.

Диаграммы компонентов предлагают аналитикам и разработчикам естественный формат для начала моделирования решения и позволяют проверить, требуются ли для системы дополнительные функциональные возможности.

Тестирование подтвердило работоспособность разработанной программы и правильность реализации эффективных алгоритмов классификации и кластеризации больших объемов данных.

Заключение

Выпускная квалификационная работа посвящена актуальной проблеме исследования алгоритмов классификации и кластеризации больших объемов данных.

Качество результатов анализа больших данных зависит от свойств конкретного алгоритма, используемого для классификации и кластеризации данных.

Для определения целесообразности применения алгоритмов классификации и кластеризации для решения конкретной задачи анализа данных необходимо провести их полное исследование.

Для достижения данной цели в процессе работы над бакалаврской работой решены следующие задачи:

- произведен анализ алгоритмов классификации больших объемов данных. На основании анализа литературы и источников по проблеме были выделены следующие алгоритмы классификации, которые используются для анализа больших объемов данных: алгоритмы метода ближайших соседей, алгоритм классификации «Случайный лес» и алгоритм классификации по методу стохастического градиента. Как показал сравнительный анализ, высокую эффективность классификации больших данных обеспечивают алгоритмы ближайших соседей и стохастического градиента. С точки зрения реализации более прост алгоритм ближайших соседей;
- произведен анализ алгоритмов кластеризации больших объемов данных и выбрать наиболее эффективный алгоритм кластеризации. На основании анализа литературы и источников по проблеме были выделены следующие алгоритмы классификации, которые используются для анализа больших объемов данных: алгоритм k-means, алгоритм Борувки и алгоритмы агломеративной кластеризации. Как показал сравнительный анализ, высокую эффективность кластеризации больших данных обеспечивают

алгоритмы k-means и агломеративной кластеризации. С точки зрения реализации более прост алгоритм k-means;

– разработана и протестирована программа, реализующие выбранные алгоритмы классификации и кластеризации больших объемов данных. Программа разработана на языке Python обладает IDE Eclipse+PyDEv. Тестирование подтвердило работоспособность разработанной программы и правильность реализации эффективных алгоритмов классификации и кластеризации больших объемов данных.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы для бизнес-аналитиков и разработчиков программ, использующих для принятия управленческих решений методы и алгоритмы интеллектуального анализа больших объемов данных.

Список используемой литературы

1. Воронцов К.В. Методы кластеризации [Электронный ресурс]. URL: http://www.machinelearning.ru/wiki/images/archive/2/28/20150427184336%21Voron-ML-Clustering-slides.pdf (дата обращения: 31.05.2021).

2. Интегрированные среды разработки программ [Электронный ресурс]. URL: http://bourabai.ru/einf/ide.htm (дата обращения: 31.05.2021).

3. Краткое руководство. Знакомство с интегрированной средой разработки Visual Studio [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/quickstart-ide-orientation?view=vs-2019 (дата обращения: 25.05.2021).

4. Метод ближайших соседей [Электронный ресурс]. URL: https://learnmachinelearning.wikia.org/ru/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9_(kNN) (дата обращения: 31.05.2021).

5. Нейский И.М. Классификация и сравнение методов кластеризации // Интеллектуальные технологии и системы // Сборник учебно-методических работ и статей аспирантов и студентов. М.: НОК «CLAIM», 2006. Выпуск 8. С. 130-142.

6. Самуйлов С. В. Объектно-ориентированное моделирование на основе UML : учебное пособие. Саратов : Вузовское образование, 2016. 37 с.

7. Advantages and Disadvantages of Stochastic Gradient Descent [Электронный ресурс]. URL: https://webcache.googleusercontent.com/search?q=cache:STMPQvzw9CoJ:https://www.asquero.com/article/advantages-and-disadvantages-of-stochastic-gradient-descent/+&cd=1&hl=en&ct=clnk&gl=ru (дата обращения: 31.05.2021).

8. An Introduction to Clustering and different methods of clustering [Электронный ресурс]. URL: https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/ (дата обращения: 31.05.2021).

9. Boruvka’s Algorithm for Minimum Spanning Trees in Java [Электронный ресурс]. URL: https://www.baeldung.com/java-boruvka-algorithm (дата обращения: 31.05.2021).

10. Donges N. A complete guide to the random forest algorithm [Электронный ресурс]. URL: https://builtin.com/data-science/random-forest-algorithm (дата обращения: 31.05.2021).

11. Eclipse IDE [Электронный ресурс]. URL: https://www.eclipse.org/eclipseide/ (дата обращения: 25.05.2021).

12. Geir Storvik. The Stochastic gradient algorithm [Электронный ресурс]. URL: https://www.uio.no/studier/emner/matnat/math/STK4051/v20/pensumliste/stoc_grad.pdf (дата обращения: 31.05.2021).

13. Hierarchical Clustering in Data Mining [Электронный ресурс]. URL: https://www.geeksforgeeks.org/hierarchical-clustering-in-data-mining/ (дата обращения: 31.05.2021).

14. K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks [Электронный ресурс]. URL: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a (дата обращения: 31.05.2021).

15. K-Nearest Neighbors (K-NN) Explained [Электронный ресурс]. URL: https://towardsdatascience.com/k-nearest-neighbors-k-nn-explained-8959f97a8632 (дата обращения: 31.05.2021).

16. K-Nearest Neighbors Algorithm for Machine Learning [Электронный ресурс]. URL: https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26 (дата обращения: 31.05.2021).

17. Liu Y. Minimum Spanning Trees, West Virginia University, Morgantown, WV [Электронный ресурс]. URL: https://community.wvu.edu/~krsubramani/courses/fa01/random/lecnotes/lec11/MST.pdf (дата обращения: 31.05.2021).

18. Pycharm IDE [Электронный ресурс]. URL:

https://www.jetbrains.com/pycharm/ (дата обращения: 10.06.2021).

19. Random forest [Электронный ресурс]. URL: https://cybernetics.wikia.org/ru/wiki/Random_forest (дата обращения: 31.05.2021).

20. Random Forest Algorithm- An Overview [Электронный ресурс]. URL: https://www.mygreatlearning.com/blog/random-forest-algorithm/ (дата обращения: 31.05.2021).

21. Stochastic Learning [Электронный ресурс]. URL: https://leon.bottou.org/papers/bottou-mlss-2004 (дата обращения: 31.05.2021).

22. Weighted K-NN [Электронный ресурс]. URL: https://www.geeksforgeeks.org/weighted-k-nn/ (дата обращения: 31.05.2021).

23. Xuesong Yan. Weighted K-Nearest Neighbor Classification Algorithm Based on Genetic Algorithm, IAES, vol. 11(10), 2013.