ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
Институт компьютерных наук и кибербезопасности

Отчет о прохождении учебной (научно-исследовательская работа (получение первичных навыков научно-исследовательской работы)) практики

Дворниченко Тимур Иванович

(Ф.И.О. обучающегося)

1 курс, 5130203/40001

(номер курса обучения и учебной группы)

02.03.03 Математическое обеспечение и администрирование информационных систем

(направление подготовки (код и наименование))

Место прохождения практики: ФГАОУ ВО «СПбПУ», ИКНиК, ВШТИИ,

(указывается наименование профильной организации или наименование структурного подразделения

г. Санкт-Петербург, ул. Обручевых, д. 1, лит. В

ФГАОУ ВО «СПбПУ», фактический адрес)

Сроки практики: с 20.06.2025 по 17.07.2025

Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»: Пак Вадим Геннадьевич, к.ф.-м.н., доцент ВШТИИ

(Ф.И.О., уч. степень, должность)

Консультант практической подготовки от ФГАОУ ВО «СПбПУ»: Чжан Юйи, ассистент ин. ВШТИИ

(Ф.И.О., уч. степень, должность)

Руководитель практической подготовки от профильной организации: нет

Оценка:

Руководитель практической подготовки
от ФГАОУ ВО «СПбПУ»: /Пак В.Г./

Консультант практической подготовки
от ФГАОУ ВО «СПбПУ»: /Чжан Ю./

Обучающийся: /Дворниченко Т. И./

Дата: 17.07.25

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Алгоритм k-ближайших соседей (k-NN) является одним из фундаментальных методов машинного обучения, широко применяемым в задачах классификации и регрессии. Его популярность обусловлена простотой реализации и интерпретируемостью результатов. В последние годы исследование влияния различных метрик на качество классификации стало важным направлением в области анализа данных.

Актуальность исследования - Алгоритм **k-NN** широко применяется в задачах классификации и регрессии благодаря простоте и интерпретируемости. Однако его эффективность сильно зависит от выбора метрики расстояния. Исследование влияния различных метрик на точность классификации позволит оптимизировать выбор параметров для конкретных задач.

Объект исследования –алгоритм k-NN и его метрики расстояния.

Предмет исследования – сравнительная эффективность метрик расстояния в k-NN на различных наборах данных.

Цель исследования – провести комплексный анализ влияния метрик расстояния на точность и скорость работы алгоритма k-NN и сформулировать рекомендации по их выбору для конкретных типов данных. В соответствии с целью исследования, логически определяются следующие **задачи работы:**

1. Изучение теоретических основ алгоритма k-NN и роли метрик в его работе, включая анализ их математических свойств.
2. Анализ существующих исследований по применению различных метрик расстояния в k-NN, включая сравнительные оценки их эффективности на различных типах данных.
3. Программная реализация алгоритма k-NN с различными метриками расстояния.
4. Тестирование на различных наборах данных, анализ эффективности метрик, формулирование выводов.

ГЛАВА 1. ОСНОВНЫЕ АСПЕКТЫ И НАПРАВЛЕНИЯ РАЗВИТИЯ МЕТОДА K-NN

Данная глава посвящена история появления и развития метода k-ближайших соседей (k-NN), его применению для решения современных задач классификации и регрессии.

В параграфе 1.1 рассматриваются история появления и развития алгоритма k-NN. Параграф 1.2 посвящён современному представлению алгоритма, его преимуществам и недостаткам, роли метрик расстояния в его работе, сферам применения и модификациям.

1.1 История появления и развития алгоритма k-NN

Алгоритм k-NN был впервые предложен в 1951 году Эвелином Фиксом и Джозефом Ходжесом [1] как непараметрический метод классификации. Их работа заложила основы для дальнейшего развития алгоритма, включая доказательство его состоятельности, то есть сходимости к оптимальному решению при увеличении объёма данных.

В 1967 году Томас Ковер и Питер Харт опубликовали ключевую теоретическую работу [2], где доказали, что ошибка классификации 1-NN не превышает удвоенной ошибки Байеса. Это стало фундаментом для понимания устойчивости алгоритма.

Изначально k-NN применялся в задачах распознавания образов в медицине, но с развитием вычислительных мощностей и машинного и машинного обучения его начали использовать в рекомендательных системах, геоаналитике и обработке естественного языка.

1.2 Современное представление и использование алгоритма k-NN

1.2.1 Преимущества и недостатки алгоритма k-NN

Сегодня k-NN остаётся одним из базовых алгоритмов машинного обучения. К основным преимуществам алгоритма k-NN можно отнести [3]:

- **Простота реализации**.
- **Адаптируемость**.

- **Требование малого количества гиперпараметров** – для корректной и эффективной работы алгоритма необходимо подобрать параметр k и метрику расстояния.

К недостаткам же относят:

- **Плохая масштабируемость** – так как k-NN – относится к ленивым алгоритмам, он занимает больше памяти и места для хранения данных по сравнению с другими классификаторами.
- **Склонность к переобучению** – необходимость удаления лишних данных и подбора оптимального параметра k.
- **Проклятье размерности** – алгоритм плохо работает с входными данными большой размерности.

1.2.2 Области применения алгоритма k-NN

Метод k-ближайших соседей, в силу своей простоты и адаптивности имеет множество областей для применения. Основными являются:

- **Классификация** – например, определение категории товаров на основе их характеристик.
- **Регрессия** – например, определение атмосферного давления из заданных параметров.
- **Рекомендательные системы** – например, поиск похожих пользователей или товаров.
- **Биометрия и медицина** – например, диагностика заболеваний по схожести симптомов с историческими случаями.

1.2.3 Роль метрик расстояния в алгоритме k-NN

Алгоритм k-ближайших соседей основан на принципе локальности: предполагается, что близко расположенные объекты в пространстве признаков принадлежат одному классу. Метрики расстояния играют ключевую роль в этом процессе, определяя:

- Принцип определения расстояния между точками данных.

– Качество классификации.
– Устойчивость к особенным данным.
– Вычислительная сложность.

Не существует универсальной метрики – выбор зависит от природы данных, масштаба признаков, наличия шумов и выбросов. Таким образом, метрика расстояния – ключевая составляющая алгоритма k-NN. Для каждого набора данных необходим выбор оптимальной метрики.

1.2.4 Современные модификации

Алгоритм активно модифицируется. К ключевым современным модификациям относят:

– Оптимизация вычислений – использование структур данных (KD-деревья, BallTree) [4] для ускорения поиска соседей в высокоразмерных пространствах.
– Взвешенное голосование -–учёт расстояния до соседей через весовые функции для уменьшения влияния шумов.
– Гибридные метрики – комбинация нескольких метрик для данных смешанного типа.

1.3 Выводы

В ходе исследования предметной области были систематизированы ключевые аспекты, связанные с работой алгоритма k-NN. Основные выводы:

A. Алгоритм k-ближайших соседей является непараметрическим методом, эффективность которого зависит от:
   1. Выбора метрики расстояния.
   2. Количества соседей k.
   3. Качества данных.
B. Метрики расстояния играют критическую роль:
   1. Евклидова метрика подходит для данный с линейной структурой.
   2. Манхэттенская метрика устойчива к выбросам.

3. Косинусная метрика эффективна для текстовых и высокоразмерных данных.

C. Теоретические гарантии [2]:

1. Ошибка k-NN ограничена двойной ошибкой Байеса.
2. Алгоритм сохраняет эффективность даже без знания распределённых данных.

ГЛАВА 2. ФОРМАЛЬНОЕ ОПИСАНИЕ И СВОЙСТВА МЕТРИК РАССТОЯНИЯ

В данной главе рассмотрим теоретические основы алгоритма *k*-ближайших соседей, включая его математическую формулировку, принцип работы и ключевые метрики расстояния.

2.1 Принцип работы алгоритма k-NN

Алгоритм k-NN основан на гипотезе компактности, которая предполагает, что схожие объекты в пространстве признаков принадлежат одному классу [2].

2.1.1 Математическая формулировка

Пусть даны:

- Обучающая выборка $\{(x_1, y_1), \dots, (x_n, y_n)\}$, где $y_i -$ класс объекта x_i.
- Новый объект x.

Шаги алгоритма **k-NN**:

1. **Вычисление расстояний**: для x находят расстояния $p(x, x_i)$ до всех x_i в обучающей выборке, используя выбранную метрику.
2. **Выбор соседей**: отбираются *k* объектов с наименьшими расстояниями *x*.
3. **Голосование**: класс *x* определяется как наиболее частый среди *k* соседей:

$$y(x) = argmax_y \sum_{i=1}^{k} I(y_i = y)\,, \qquad (2.1)$$

где ***I*** – индикаторная функция.

2.1.2 Теоретические гарантии

В статье [2] доказано, что для $k = 1$:

– Ошибка классификации R для двух классов ограничена двойной ошибкой Байеса R^*:

$$R^* \leq R \leq 2R^*. \tag{2.2}$$

– Для M классов:

$$R \leq R^* \left(2 - \frac{M}{M-1} R^*\right). \tag{2.3}$$

Это означает, что k-NN не требует знания распределения данных и сохраняет эффективность даже в непараметрических условиях.

2.2 Обзор метрик расстояния

В данном разделе будут рассмотрены три основные метрики расстояния:

– Евклидова метрика.

– Манхэттенская метрика.

– Косинусная метрика.

2.2.1 Евклидова метрика

2.2.1.1 Математическое определение

Для двух точек x и y в n-мерном пространстве:

$$p(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{2.4}$$

где $x = (x_1, x_2, \ldots, x_n)$ и $y = (y_1, y_2, \ldots, y_n)$ – векторы признаков, n – количество признаков.

2.2.1.2 Основные свойства

Свойства евклидовой метрики:

- **Инвариантность к вращению**: не изменяется при повороте системы координат.
- **Чувствительность к масштабу**: требует нормализации данных, если признаки измеряются в разных единицах.
- **Геометрическая интерпретация**: корректно отражает расстояние в физическом пространстве через теорему Пифагора.

2.2.2 Манхэттенская метрика

2.2.2.1 Математическое определение

Для двух точек $x = (x_1, x_2, \dots, x_n)$ и $y = (y_1, y_2, \dots, y_n)$ в n-мерном пространстве манхэттенское расстояние определяется как:

$$p(x, y) = \sum_{i=1}^{n} |x_i - y_i| . \quad (2.5)$$

2.2.2.2 Основные свойства

Свойства манхэттенской метрики:

- **Инвариантность к вращению**: в отличие от евклидова расстояния, манхэттенская метрика не инвариантна к повороту системы координат, однако она инвариантна к отражениям и сдвигам.
- **Устойчивость к выбросам**: поскольку используется сумма модулей разностей, манхэттенское расстояние менее чувствительно к большим отклонениям в отдельных координатах по сравнению с евклидовым расстоянием.
- **Сравнение с другими метриками**: для любых двух точек *x*, *y*:

$$p_{\text{манх}}(x, y) \geq p_{\text{евкл}}(x, y).$$

2.2.3 Косинусная метрика

2.2.3.1 Математическое определение

Для векторов x и y косинусное сходство вычисляется как:

$$similarity(x, y) = \frac{x \cdot y}{|x| \cdot |y|}, \quad (2.6)$$

где $x \cdot y = \sum_{i=1}^{n} x_i y_i$ – скалярное произведение векторов, $|x| = \sqrt{\sum_{i=1}^{n} x_i^2}$ и $|y| = \sqrt{\sum_{i=1}^{n} y_i^2}$ – их евклидовы нормы.

Косинусное расстояние – это дополнение сходства до единицы:

$$p(x, y) = 1 - similarity(x, y). \quad (2.7)$$

2.2.3.2 Основные свойства

Свойства косинусной метрики:

- **Диапазон значений**: значение косинусного расстояния лежит в диапазоне от 0 (векторы совпадают) до 2 (векторы противоположны).
- **Инвариантность к масштабу**: не зависит от длины векторов, только от угла между ними.
- **Нормализация**: если векторы нормированы, $similarity(x, y) = x \cdot y$.

2.3 Вывод

В данной главе были систематизированы ключевые теоретические аспекты алгоритма k-NN и метрик расстояния, определяющих его эффективность. Основные выводы:

A. Алгоритм k-NN демонстрирует универсальность благодаря непараметрической природе, но его производительность критически зависит от:

1. Корректного выбора метрики расстояния.
2. Оптимального значения *k.*
3. Качество предобработки данных.

В. Метрики расстояния обладают специфическими свойствами:

1. Евклидова метрика чувствительна к масштабу данных.
2. Манхэттенская метрика устойчива к выбросам.
3. Косинусная метрика эффективна для анализа направленности векторов.

ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ И ТЕСТИРВАНИЕ АЛГОРИТМА K-NN

В данной главе описывается процесс программной реализации алгоритма k-NN с различными метриками расстояния. Для реализации проекта язык программирования Python, поскольку он обладает всем необходимым инструментарием, предоставляемым библиотеками, для использования методов машинного обучения, библиотеки sklearn (методы машинного обучения), matplotlib (визуализация данных).

3.1 Программная реализация алгоритма k-NN с использованием различных метрик расстояния

Для реализации алгоритма k-NN и анализа его эффективности были использованы ключевые библиотеки Python: scikit-learn для машинного обучения и matplotlib для визуализации результатов. Для исследования были выбраны три набора данных: **Iris**, **Extrovert vs. Introvert Behavior Data** и **20 Newsgroups**.

3.1.1 Реализация алгоритма k-NN для набора данных Iris и Extrovert vs. Introvert Behavior Data

Даны два набора данных:

- **Iris**: 4 столбца с признаками и один столбец с разновидность цветка.
- **Extrovert vs. Introvert Behavior Data**: 7 столбцов с признаками и один столбец, обозначающий тип человека.

Процесс исследования для каждого набора данных включал в себя несколько этапов. Первоначально наборы данных был загружены: набор **Iris** при

помощи функции **load_iris** библиотеки **sklearn,** набор **Extrovert vs. Introvert Behavior Data** был загружен с сайта Kaggle. Затем данные были разделены на обучающую (70%) и тестовую (30%) выборки с помощью функции **train_test_split**, что обеспечило корректную оценку качества модели. Поскольку k-NN чувствителен к масштабу признаков, выполнена стандартизация данных через **StandardScaler**, которая привела все признаки к нулевому среднему и единичной дисперсии.

Для сравнения эффективности метрик реализован цикл, в котором последовательно тренировались три варианта метрик расстояния: евклидова, манхэттенская и косинусная. Каждая модель k-NN с фиксированным с фиксированным числом соседей (k = 7) обучалась на нормированных данных, после чего вычислялась точность классификации на тестовой выборке с помощью **accuracy_score**. Дополнительно фиксировалось время выполнения для оценки вычислительной эффективности каждой метрики.

Результатом работы программы являются два графика (см. рис. П2.1, рис. П2.2 из приложения 2). Первый график демонстрирует сравнение точности классификации. Второй график отражает время выполнения, позволяя оценить вычислительную сложность разных метрик. Использование **matplotlib** с настройкой подписей осей, сетки и аннотаций значений обеспечивает чёткую интерпретацию результатов.

3.1.2 Реализация алгоритма k-NN для набора данных 20 Newsgroups

Дан набор данных **20 Newsgroups**, содержащий текстовые документы, распределённые по 20 тематическим категориям. Процесс исследования также включал несколько ключевых этапов. На первом этапе данные были загружены с помощью функции **fetch_20newsgroups** из библиотеки **sklearn**, после чего выполнена их векторизация с применением **TfidfVectorizer**, преобразующего текстовые документы в числовые векторы признаков. Важно отметить, что в данном случае была отключена нормализация векторов (параметр **norm** = **None**), что позволило более объективно сравнить работу разных метрик расстояния.

Все последующие этапы аналогичны реализации для наборов данных **Iris**, **Extrovert vs. Introvert Behavior Data.**

3.2 Экспериментальное исследование эффективности алгоритма k-NN для различных метрик

3.2.1 Пояснение методики забора экспериментальных данных

Для обеспечения статистической достоверности результатов для каждого набора данных было запланировано проведение 10 независимых запусков программы. В каждом запуске выполнялся полный цикл обработки данных: от их загрузки и предварительной обработки до обучения модели и оценки её эффективности. Такой подход позволяет учесть возможную вариативность, связанную с случайным разделением данных на обучающую и тестовую выборки, а также другими стохастическими факторами алгоритма.

При каждом запуске фиксировались ключевые показатели эффективности, включая точность классификации и время выполнения для всех исследуемых метрик расстояния. Полученные данные были сохранены в структурированном формате для последующего сравнительного анализа. Многократное повторение эксперимента обеспечивает репрезентативность результатов и позволяет выявить устойчивые закономерности в работе алгоритма при различных условиях.

3.2.2 Получение экспериментальных данных

После проведения 10 независимых запусков программы для каждого набора данных. В табл.3.1-3.3 приведены значения ключевых показателей эффективности алгоритма для всех наборов данных: точность классификации и время выполнения.

Таблица 3.1

Результаты работы программы для набора Iris

№ эксперимента	Метрика	Точность	Время выполнения, с
1	Евклидова	0,9111	0,0020
	Манхэттена	0,9333	0,0015

Продолжение табл. 3.1

1	Косинусная	0,8444	0,0018
2	Евклидова	0,9333	0,0020
	Манхэттена	0,9333	0,0015
	Косинусная	0,9333	0,0017
3	Евклидова	0,9333	0,0020
	Манхэттена	0,9333	0,0014
	Косинусная	0,8667	0,0017
4	Евклидова	0,9556	0,0026
	Манхэттена	0,9556	0,0019
	Косинусная	0,8889	0,0023
5	Евклидова	0,9556	0,0020
	Манхэттена	0,9333	0,0017
	Косинусная	0,8444	0,0022
6	Евклидова	0,9778	0,0020
	Манхэттена	0,9333	0,0015
	Косинусная	0,8222	0,0018
7	Евклидова	0,8889	0,0029
	Манхэттена	0,8667	0,0019
	Косинусная	0,7778	0,0024
8	Евклидова	0,9778	0,0028
	Манхэттена	0,9778	0,0019
	Косинусная	0,8000	0,0018
9	Евклидова	0,9333	0,0019
	Манхэттена	0,9333	0,0014
	Косинусная	0,8000	0,0018
10	Евклидова	0,9778	0,0021
	Манхэттена	0,9778	0,0016
	Косинусная	0,9111	0,0020

Таблица 3.2

Результаты работы программы для набора Extrovert vs. Introvert Behavior Data

№ эксперимента	Метрика	Точность	Время выполнения, с
1	Евклидова	0,9310	0,0098
	Манхэттена	0,9310	0,0122
	Косинусная	0,9322	0,0323
2	Евклидова	0,9207	0,0116
	Манхэттена	0,9218	0,0129
	Косинусная	0,9195	0,0319
3	Евклидова	0,9126	0,0119
	Манхэттена	0,9103	0,0135
	Косинусная	0,9161	0,0311
4	Евклидова	0,9299	0,0099
	Манхэттена	0,9310	0,0124
	Косинусная	0,9299	0,0612

Продолжение табл. 3.2

5	Евклидова	0,9241	0,0103
	Манхэттена	0,9253	0,0135
	Косинусная	0,9230	0,0328
6	Евклидова	0,9207	0,0101
	Манхэттена	0,9207	0,0136
	Косинусная	0,9230	0,0323
7	Евклидова	0,9276	0,0104
	Манхэттена	0,9310	0,0129
	Косинусная	0,9299	0,0288
8	Евклидова	0,9276	0,0106
	Манхэттена	0,9287	0,0130
	Косинусная	0,9253	0,0315
9	Евклидова	0,9276	0,1050
	Манхэттена	0,9264	0,0122
	Косинусная	0,9276	0,0352
10	Евклидова	0,9333	0,0100
	Манхэттена	0,9345	0,0119
	Косинусная	0,9299	0,0312

Таблица 3.3

Результаты работы программы для набора 20 Newsgroups

№ эксперимента	Метрика	Точность	Время выполнения, с
1	Евклидова	0,4733	1,3777
	Манхэттена	0,0510	5,3627
	Косинусная	0,7564	1,4014
2	Евклидова	0,4239	1,2905
	Манхэттена	0,0580	4,4526
	Косинусная	0,7473	1,5059
3	Евклидова	0,4353	1,2673
	Манхэттена	0,0583	4,3541
	Косинусная	0,7511	1,4872
4	Евклидова	0,4280	1,2956
	Манхэттена	0,0515	4,3763
	Косинусная	0,7546	1,5313
5	Евклидова	0,4392	1,3285
	Манхэттена	0,0554	4,4434
	Косинусная	0,7455	1,5202
6	Евклидова	0,4448	1,2648
	Манхэттена	0,0527	4,3836
	Косинусная	0,7423	1,5311
7	Евклидова	0,4536	1,2779
	Манхэттена	0,0530	4,3928
	Косинусная	0,7594	1,6658

Продолжение табл. 3.3

8	Евклидова	0,4300	2,2826
	Манхэттена	0,0583	5,6320
	Косинусная	0,7358	2,3942
9	Евклидова	0,4106	1,5331
	Манхэттена	0,0574	4,8571
	Косинусная	0,7558	1,7214
10	Евклидова	0,4657	1,2529
	Манхэттена	0,0545	4,4023
	Косинусная	0,7638	1,5489

3.3 Анализ результатов исследования

Экспериментальное исследование эффективности алгоритма k-NN с использованием различных метрик расстояния на трёх наборах данных позволило выявить ключевые закономерности, представленные в табл. 3.1-3.3. Результаты демонстрируют существенную зависимость эффективности алгоритма от выбора метрики расстояния.

3.3.1 Анализ результатов для набора Iris

Как видно из табл. 3.1, для набора Iris наилучшие результаты точности показали евклидова и манхэттенская метрики, достигавшие значений 0,93-0,98 в большинстве экспериментов. Косинусная метрика демонстрировала более скромные результаты (0,77-0,93), что объясняется линейной структурой данных, для которой первые две метрики более подходят. При этом все метрики показали минимальное время выполнения (менее 0,003 секунды), что подтверждает эффективность алгоритма для небольших объёмов данных.

3.3.2 Анализ результатов для набора Extrovert vs. Introvert Behavior Data

Согласно данным табл. 3.2, все три метрики показали сопоставимую точность (0,91-0,93) на данном наборе данных. Однако косинусная метрика требовала в 2-3 раза больше времени для вычислений по сравнению с другими метриками. Это создаёт важный компромисс между точностью и вычислительной эффективностью, который необходимо учитывать при работе с подобными данными.

3.3.3 Анализ результатов для набора 20 Newsgroups

Результаты в табл. 3.3 наиболее ярко демонстрируют преимущество косинусной метрики для текстовых данных. В то время как евклидова метрика показывала точность около 0,43, а манхэттенская - менее 0,06, косинусная метрика стабильно достигала значений 0,74-0,76. Это подтверждает её эффективность для работы с высокоразмерными текстовыми данными, где важно учитывать направление векторов, а не их абсолютные значения.

3.4 Выводы

В данной главе был описан процесс программной реализации алгоритма k-NN для трёх наборов данных. Была разъяснена специфика использования методов библиотек scikit-learn и matplotlib. Также было проведено экспериментальное исследование, получены экспериментальные данные.

Исследование подтвердило, что выбор метрики расстояния должен учитывать природу данных. Для структурированных данных с линейными зависимостями (например, Iris) предпочтительны евклидова или манхэттенская метрики. Для данных с высокой размерностью или текстовых данных косинусная метрика оказывается наиболее эффективной. Кроме того, важно учитывать вычислительную сложность: косинусная метрика может требовать больше ресурсов, но её применение оправдано для специфических задач. Полученные результаты позволяют сформулировать рекомендации по выбору метрик для различных типов данных, что является ценным вкладом в практику машинного обучения.

ЗАКЛЮЧЕНИЕ

В результате выполнения научно-исследовательской работы была достигнута поставленная цель – проведен комплексный анализ влияния различных метрик расстояния на точность и скорость работы алгоритма k-NN, а также сформулированы рекомендации по их выбору для конкретных типов данных. В ходе работы успешно решены все поставленные задачи, что подтверждается следующими результатами:

- Проведен детальный анализ теоретических основ алгоритма k-NN, изучены его математические свойства и роль метрик расстояния в его работе. Особое внимание уделено исследованию таких метрик, как евклидова, манхэттенская и косинусная, их свойствам и областям применения.
- Исследованы современные подходы к применению различных метрик расстояния в алгоритме k-NN, включая их сравнительную эффективность на различных типах данных. Проанализированы преимущества и недостатки каждой метрики, а также их влияние на качество классификации.
- Практически реализован алгоритм k-NN с использованием различных метрик расстояния. Проведено тестирование на трех наборах данных: Iris, Extrovert vs. Introvert Behavior Data и 20 Newsgroups, что позволило оценить эффективность каждой метрики в различных условиях.
- Получены экспериментальные данные, подтверждающие зависимость эффективности алгоритма от выбора метрики расстояния. На основе анализа результатов сформулированы рекомендации по выбору оптимальной метрики для конкретных типов данных.

Разработанное решение продемонстрировало свою эффективность, подтвердив гипотезу о том, что выбор метрики расстояния критически влияет на качество работы алгоритма k-NN. Особое внимание уделено практической значимости исследования, включая рекомендации по применению метрик для структурированных и текстовых данных. Проведенная работа вносит вклад в решение актуальной задачи оптимизации алгоритма k-NN, сочетающей теоретическую обоснованность и практическую применимость.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Fix E., Hodges J.L. Discriminatory Analysis: Nonparametric Discrimination for Small Samples. Randolph Field, TX: USAF School of Aviation Medicine, 1951. 55 p. (Technical Report 21-49-004).

2. Cover T., Hart P. Nearest Neighbor Pattern Classification // IEEE Transactions on Information Theory. 1967. Vol. 13, no. 1. P. 21-27. DOI: 10.1109/TIT.1967.1053964

3. k-Nearest Neighbors (kNN) Explained // IBM.
URL: https://www.ibm.com/think/topics/knn (дата обращения: 29.07.2025).

4. Scikit-learn: Machine Learning in Python // Scikit-learn Developers.
URL: https://scikit-learn.org (дата обращения: 05.07.2025).

Приложение 1

Листинг программного кода

Листинг П1.1

accuracy_metric_iris.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
import time
#from preparation import data, target

# Загрузка датасета Iris
iris = load_iris()
X, y = iris.data, iris.target

# Разделение на обучающую и тестовую выборки (70/30)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Нормализация данных (kNN чувствителен к масштабу)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Список метрик для сравнения
METRICS = {
    'Евклидова': 'euclidean',
    'Манхэттенская': 'manhattan',
    'Косинусная': 'cosine'
}

# Сравнение точности для разных метрик
results = {}
results_time = {}
for name, metric in METRICS.items():
    knn = KNeighborsClassifier(n_neighbors=7, metric=metric)
    start = time.time()
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
```

```
    accuracy = accuracy_score(y_test, y_pred)
    total_time = time.time() - start
    results[name] = accuracy
    results_time[name] = total_time
    print(f"{name} метрика: Точность = {accuracy:.4f}")
    #print(classification_report(y_test, y_pred, target_names=iris.target_names))

# График точности
plt.figure(figsize=(10, 5))
plt.bar(results.keys(), results.values(), color=['blue', 'green', 'red'])
plt.title(f"Сравнение точности kNN с разными метриками (Iris)")
plt.ylabel('Точность')
plt.ylim(0.8, 1.01)
plt.grid(True, 'both', 'y')

for i, (metric, acc) in enumerate(results.items()):
    plt.text(i, acc + 0.002, f"{acc:.4f}", ha='center')

plt.show()

plt.figure(figsize=(10, 5))
plt.bar(results_time.keys(), results_time.values(), color=['blue', 'green', 'red'])
plt.title(f"Сравнение скорости выполнения kNN с разными метриками (Iris)")
plt.ylabel('Длительность')
plt.grid(True, 'both', 'y')

for i, (metric, t) in enumerate(results_time.items()):
    plt.text(i, t + 0.0002, f"{t:.4f}", ha='center')

plt.show()
```

Листинг П1.2

accuracy_metric_big_dataset.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
from preparation import data, target
import time

X, y = data, target
```

```
# Разделение на обучающую и тестовую выборки (70/30)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Нормализация данных (kNN чувствителен к масштабу)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Список метрик для сравнения
METRICS = {
    'Евклидова': 'euclidean',
    'Манхэттенская': 'manhattan',
    'Косинусная': 'cosine'
}

# Сравнение точности для разных метрик
results = {}
results_time = {}
for name, metric in METRICS.items():
    knn = KNeighborsClassifier(n_neighbors=7, metric=metric)
    start = time.time()
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    total_time = time.time() - start
    results[name] = accuracy
    results_time[name] = total_time
    print(f"{name} метрика: Точность = {accuracy:.4f}")
    print(classification_report(y_test, y_pred, target_names=["extroverts", "Introverts"]))


# График точности
plt.figure(figsize=(4, 5))
plt.bar(results.keys(), results.values(), color=['blue', 'green', 'red'])
plt.title(f'Сравнение точности kNN с разными метриками (Extrovert vs. Introvert Be-
havior Data)')
plt.ylabel('Точность')
plt.ylim(0.9, 1)
plt.grid(True, 'both', 'y')

for i, (metric, acc) in enumerate(results.items()):
    plt.text(i, acc + 0.002, f"{acc:.4f}", ha='center')

plt.show()
```

```
plt.figure(figsize=(5, 5))
plt.bar(results_time.keys(), results_time.values(), color=['blue', 'green', 'red'])
plt.title(f"Сравнение скорости выполнения kNN с разными метриками (Extrovert vs. Introvert Behavior Data)")
plt.ylabel('Длительность')
plt.grid(True, 'both', 'y')

for i, (metric, t) in enumerate(results_time.items()):
    plt.text(i, t + 0.0002, f"{t:.4f}", ha='center')

plt.show()
```

Листинг П1.3

accuracy_metric_text.py

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import time

# Загрузка данных
newsgroups = fetch_20newsgroups(subset='train')
X = newsgroups.data
y = newsgroups.target

# Векторизация текста
vectorizer = TfidfVectorizer(norm=None)
X_vectors = vectorizer.fit_transform(X)

# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X_vectors, y, test_size=0.3)

# Список метрик для сравнения
metrics = ['euclidean', 'manhattan', 'cosine']
results = {}
results_time = {}

for metric in metrics:
    # Создание и обучение модели
    knn = KNeighborsClassifier(metric=metric, n_neighbors=7)
    start = time.time()
    knn.fit(X_train, y_train)
```

```
    # Предсказание и оценка точности
    y_pred = knn.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    total_time = time.time() - start
    results[metric] = acc
    results_time[metric] = total_time

    print(f"Метрика: {metric:<10} Точность: {acc:.4f}")

# Визуализация результатов
import matplotlib.pyplot as plt

plt.figure(figsize=(4, 5))
plt.bar(results.keys(), results.values(), color=['blue', 'green', 'red'])
plt.title('Сравнение точности kNN с разными метриками (20 Newsgroups)')
plt.xlabel('Метрика расстояния')
plt.ylabel('Точность (accuracy)')
plt.grid(True, 'both', 'y')
plt.ylim(0, 1)

for i, (metric, acc) in enumerate(results.items()):
    plt.text(i, acc + 0.02, f"{acc:.4f}", ha='center')

plt.show()

plt.figure(figsize=(4, 5))
plt.bar(results_time.keys(), results_time.values(), color=['blue', 'green', 'red'])
plt.title(f"Сравнение скорости выполнения kNN с разными метриками (20 News-
groups)")
plt.ylabel('Длительность')
plt.grid(True, 'both', 'y')

for i, (metric, t) in enumerate(results_time.items()):
    plt.text(i, t + 0.02, f"{t:.4f}", ha='center')

plt.show()
```

Приложение 2

Примеры выходных данных программы

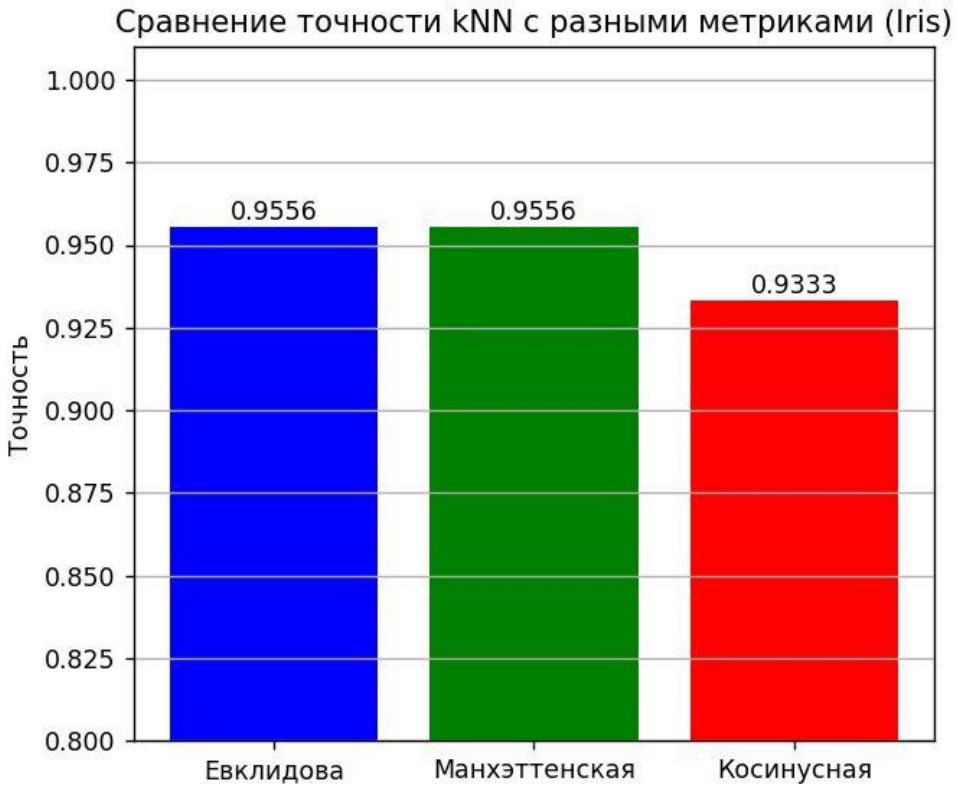

Рис. П2.1. График точности алгоритма k-NN

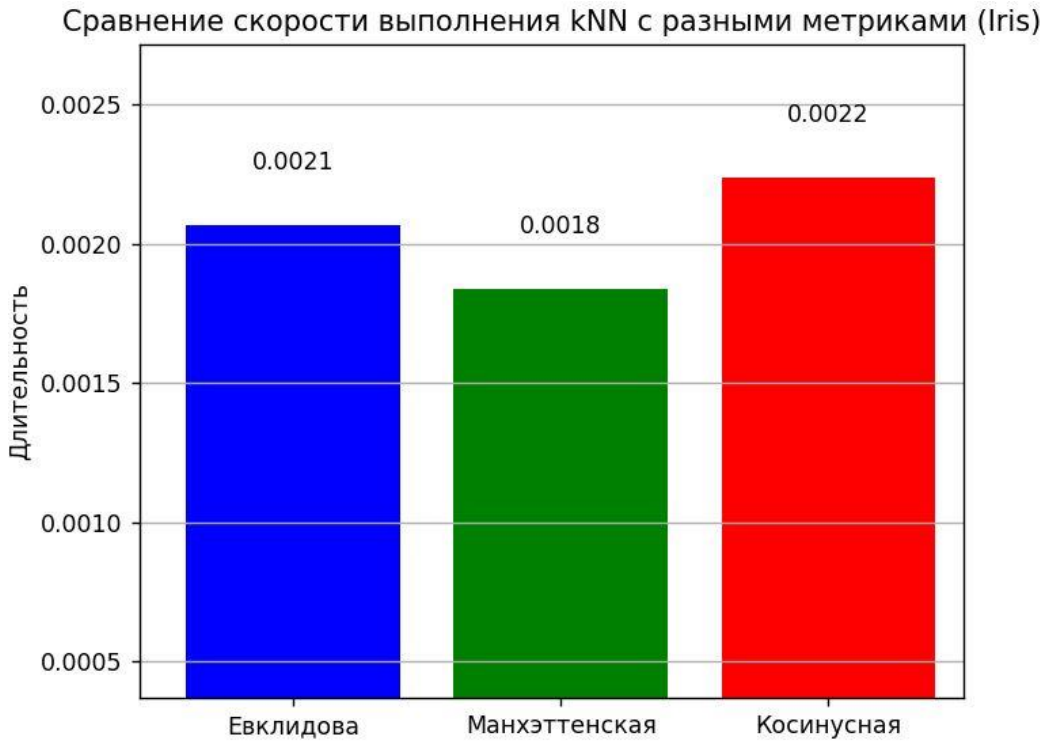

Рис. П2.2. График скорости выполнения алгоритма k-NN