DOI 10.37539/2949-1991.2025.28.5.016

Сироткин Максим Сергеевич,
Крюкова Диана Вадимовна,
Чернявский Роман Евгеньевич,
Студенты, Сахалинский государственный университет,
Южно-Сахалинск.

Научный руководитель **Осипов Геннадий Сергеевич**,
д.т.н., профессор кафедры информатики,
Сахалинский государственный университет, Южно-Сахалинск.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ К-БЛИЖАЙШИХ СОСЕДЕЙ И ДЕРЕВА РЕШЕНИЙ В МАШИННОМ ОБУЧЕНИИ

Аннотация: Проведено аналитическое сравнение двух базовых методов машинного обучения: К-ближайших соседей и дерева решений. Изложены основы теории методов и их практическое сравнение на реальных задачах. Приводятся рекомендации по выбору метода и настройки его гиперпараметров.

Ключевые слова: задачи и методы машинного обучения

Введение

Методы машинного обучения, такие как K-ближайших соседей (KNN) и дерево решений (DT), занимают важное место в арсенале инструментов анализа данных. Несмотря на их простоту, они остаются актуальными для решения задач классификации, регрессии и кластеризации. Оба метода относятся к непараметрическим алгоритмам, что позволяет им адаптироваться к сложным распределениям данных без строгих предположений о их природе. Однако их принципы работы, производительность и интерпретируемость существенно различаются, что делает их применимыми в разных сценариях.

KNN основывается на гипотезе компактности: объекты, близкие в пространстве признаков, вероятно, принадлежат к одному классу или имеют схожие значения целевой переменной. Алгоритм не требует явного этапа обучения, что упрощает его реализацию, но делает критически зависимым от выбора метрики расстояния и гиперпараметра K (числа соседей). Этот метод часто используется в задачах геопространственного анализа, рекомендательных систем и биометрии, где естественная мера близости объектов очевидна [1].

DT строит иерархическую структуру решающих правил, последовательно разделяя данные на подмножества. Каждое разбиение направлено на максимизацию чистоты классов (для классификации) или минимизацию дисперсии (для регрессии). Деревья решений ценятся за высокую интерпретируемость: специалисты могут анализировать логику принятия решений, что критически важно в медицине, финансах и юриспруденции. Однако DT склонны к переобучению при избыточной глубине, что требует применения методов регуляризации, таких как обрезка ветвей [2].

Актуальность сравнения методов обусловлена их широким применением в индустрии и науке. Например, KNN часто служит «базовым» алгоритмом для быстрой проверки гипотез, тогда как DT используется в ансамблевых методах (случайные леса, градиентный бустинг), которые доминируют в соревнованиях по машинному обучению. Выбор между ними зависит от требований к скорости, интерпретируемости и устойчивости к шуму.

Цель статьи – провести детальный сравнительный анализ KNN и DT, опираясь на их теоретические основы, практические реализации и примеры из реальных задач.

Критерии сравнения:

1. Скорость обучения и предсказания: время, затрачиваемое на построение модели и генерацию прогнозов.

2. Чувствительность к гиперпараметрам: влияние *K* (для KNN) и глубины дерева (для DT) на качество модели.

3. Устойчивость к шуму: способность игнорировать выбросы и некорректные данные.

4. Производительность на разных объемах данных: эффективность работы на малых и больших выборках.

5. Интерпретируемость: возможность объяснить логику принятия решений.

6. Области применения: типы задач, где метод демонстрирует наилучшие результаты.

1. Теоретическая основа

1.1 Метод K-ближайших соседей

KNN – непараметрический алгоритм, который для классификации объекта *x* находит *K* ближайших к нему точек в обучающей выборке *D* и присваивает *x* наиболее частый класс среди этих соседей. Для регрессии прогнозируемое значение вычисляется как среднее или медиана значений соседей [3].

Алгоритм работы:

1. Вычисление расстояний: для нового объекта *x* рассчитываются расстояния до всех точек D.

2. Выбор соседей: отбираются *K* объектов с минимальными расстояниями.

3. Агрегация:

○ Классификация:

$$\hat{y} = \text{mode}(\{y_i\}_{i=1}^{K}) \quad .(1)$$

○ Регрессия:

$$\hat{y} = \frac{1}{K}\sum_{i=1}^{K} y_i. \quad (2)$$

Метрики расстояния:

• Евклидово расстояние:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (3)$$

Подходит для непрерывных признаков с одинаковым масштабом.

• Манхэттенское расстояние:

$$d(x, y) = \sum_{i=1}^{n}|x_i - y_i| \quad (4)$$

Устойчиво к выбросам, используется в задачах с разреженными данными.

• Косинусное сходство:

$$\text{similarity}(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|} \quad (5)$$

Эффективно для текстовых данных и задач информационного поиска.

Влияние гиперпараметра *K*:

• Малые значения *K* (1–3):

○ Высокая чувствительность к шуму.

○ Резкие границы между классами (риск переобучения).

• Большие значения *K* (>10):

○ Сглаживание границ, устойчивость к выбросам.

○ Потеря детализации (риск недообучения).

Преимущества KNN:

• Отсутствие этапа обучения: данные хранятся в памяти без предварительной обработки.

• Универсальность: подходит для классификации, регрессии и поиска аномалий.

• Адаптивность: легко модифицируется через выбор метрики и весов соседей.

Ограничения KNN:

• Высокая вычислительная сложность: время предсказания растет линейно с размером выборки (O (N)).

• Чувствительность к масштабированию признаков: требует нормализации данных.

• Проклятие размерности: эффективность падает в пространствах высокой размерности.

1.2 Метод дерева решений

DT – алгоритм, рекурсивно разделяющий пространство признаков на области, максимизирующие однородность меток. Каждый внутренний узел дерева соответствует условию на признак, а лист – прогнозируемому значению [3].

Критерии разбиения:

• Для классификации:

○ Индекс Джини:

$$Gini = 1 - \sum_{i=1}^{C} p_i^2 \quad (6)$$

Минимизирует вероятность ошибки классификации.

○ Энтропия:

$$H = -\sum_{i=1}^{C} p_i \log p_i \quad (7)$$

Стремится к максимальной чистоте узлов.

• Для регрессии:

○ Среднеквадратичная ошибка (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \quad (8)$$

Алгоритм построения дерева:

1. Выбор признака и порога: для каждого признака вычисляется оптимальный порог, максимизирующий прирост информации (IG):

$$IG = H_{\text{родитель}} - \sum_{j=1}^{k} \frac{N_j}{N} H_{\text{дочерний}_j} \quad (9)$$

2. Рекурсивное разбиение: процесс повторяется для дочерних узлов.

3. Критерии остановки:

○ Достижение максимальной глубины.

○ Минимальное количество объектов в листе.

○ Отсутствие прироста информации.

Глубина дерева:

• Малая глубина (2–4 уровня):

○ Простые, интерпретируемые модели.

○ Риск недообучения.

• Большая глубина (>10 уровней):

○ Точное соответствие обучающим данным.

○ Высокий риск переобучения.

Преимущества DT:

• Интерпретируемость: Логика решений видна из структуры дерева.

• Работа с разнородными данными: не требует предобработки (например, нормализации).

• Быстрое предсказание: O (log$N$) для одного объекта.

Ограничения DT:

• Неустойчивость к шуму: Малое изменение данных может радикально изменить структуру дерева.

• Склонность к переобучению: требует тщательной настройки гиперпараметров.

• Локальная оптимизация: жадный алгоритм не гарантирует глобального оптимума.

Теоретическая сложность:

• Обучение: $O(N \cdot M \cdot \log N)$, где M – число признаков.

• Предсказание: $O(\log N)$.

2. Практическое применение

2.1 Примеры с KNN:

Классификация на данных Iris (рис. 1):

```
(* Загрузка и предобработка данных *)
irises = ResourceData["Sample Data: Fisher's Irises"];
(* Удаление лишних признаков *)
irises = KeyDrop[irises, {"PetalLength", "PetalWidth"}];
(* Добавление шума для демонстрации устойчивости *)
irises = MapAt[# + RandomReal[{-0.1, 0.1}] &, irises, {All, 2 ;; 3}];
(* Обучение модели с K=5 *)
knn = Classify[irises -> "Species",
  Method -> {"NearestNeighbors", "NeighborsNumber" -> 5}];
(* Визуализация границ решений *)
RegionPlot[{
  knn[{x, y}] == "setosa",
  knn[{x, y}] == "versicolor",
  knn[{x, y}] == "virginica"},
  {x, 4, 8}, {y, 1, 5},
  PlotLegends -> {"setosa", "versicolor", "virginica"},
  Epilog -> {PointSize[Small], Point[irises[[All, 2 ;; 3]]]}]
```

Рис.1 Классификация на данных Iris (Метод ближайших соседей)

1. Данные Iris содержат 150 экземпляров цветков с признаками длины и ширины чашелистика.

2. Удаление признаков лепестков упрощает визуализацию в 2D.

3. Добавление случайного шума демонстрирует устойчивость KNN к небольшим искажениям данных.

4. Классификатор с K=5 строит плавные границы, избегая переобучения.

Регрессия на данных Boston Homes (рис. 2):

```
(* Прогнозирование медианной стоимости жилья *)
houses = ResourceData["Sample Data: Boston Homes"];
(* Обучение модели с K=20 *)
houseModel = Predict[houses -> "MEDV",
  Method -> {"NearestNeighbors", "NeighborsNumber" -> 20}];
(* 3D-визуализация прогнозов *)
ListPlot3D[
  Flatten[Table[{x, y, houseModel[{x, y}]}, {x, 0, 10}, {y, 0, 10}], 1],
  ColorFunction -> "Temperature"]
```

Рис. 2 Регрессия на данных Boston Homes (Метод ближайших соседей)

1. Набор данных Boston Homes включает 506 записей с характеристиками недвижимости.

2. Модель предсказывает медианную стоимость жилья (MEDV) на основе признаков, таких как количество комнат и уровень преступности.

3. Использование K=20 сглаживает прогнозы, уменьшая влияние локальных выбросов.

Результаты метода (рис. 3, рис. 4):

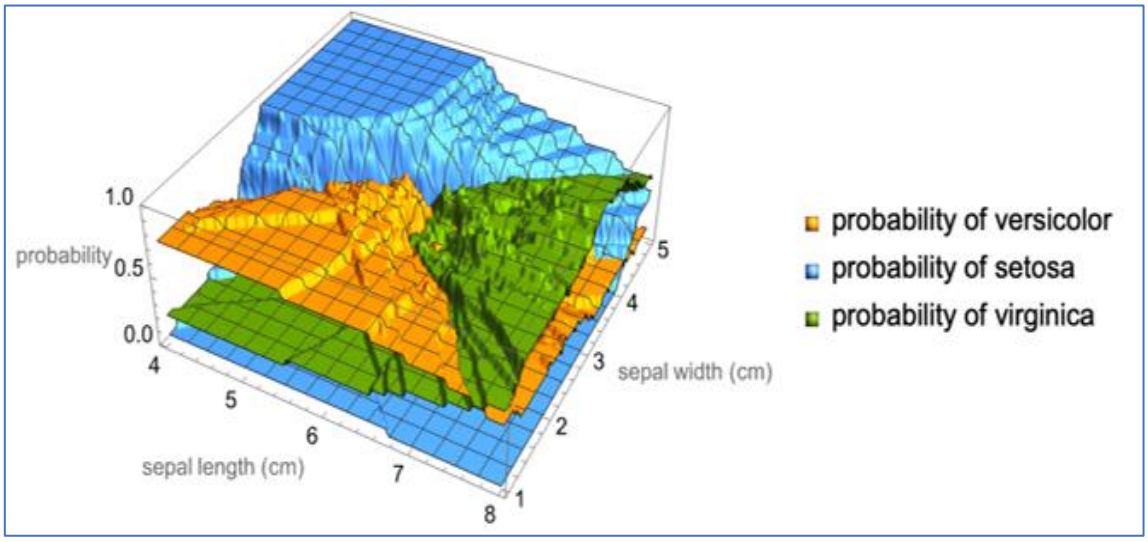

Рис. 3 Результат классификации на данных Iris (Метод ближайших соседей)

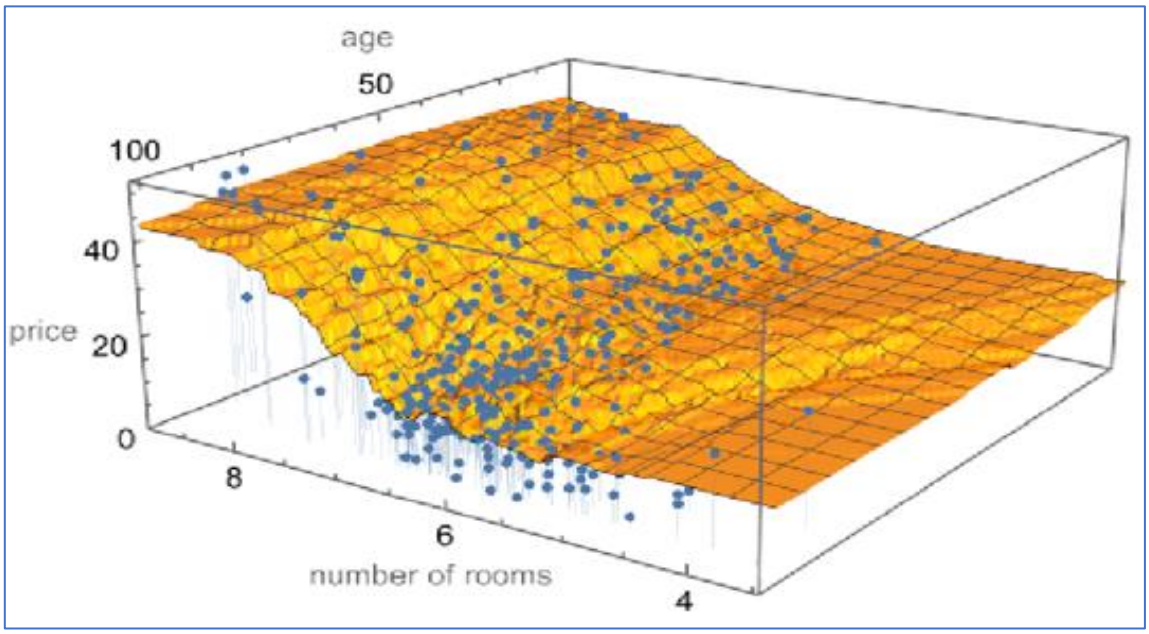

Рис. 4 Результат регрессии на данных Boston Homes (Метод ближайших соседей)

2.2 Примеры с деревом решений:

Классификация Iris (рис. 5):

```
(* Обучение дерева решений *)
tree = Classify[irises -> "Species", Method -> "DecisionTree"];
(* Визуализация границ *)
RegionPlot[{
 tree[{x, y}] == "setosa",
 tree[{x, y}] == "versicolor",
 tree[{x, y}] == "virginica"},
 {x, 4, 8}, {y, 1, 5},
 BoundaryStyle -> Dashed,
 PlotLegends -> Placed[{"setosa", "versicolor", "virginica"}, Right]]
```

Рис. 5 Классификация на данных Iris (Метод ближайших соседей)

1. Дерево решений разделяет данные на прямоугольные области, соответствующие условиям вида «длина чашелистика <5.5».
2. Границы решений имеют ступенчатый вид, отражая иерархическую логику дерева.

Регрессия для Boston Homes (рис. 6):

```
(* Обучение регрессионного дерева *)
houseTree = Predict[houses -> "MEDV", Method -> "DecisionTree"];
(* Визуализация в 3D *)
Show[
 ListPlot3D[Table[houseTree[{x, y}], {x, 0, 10}, {y, 0, 10}],
  ColorFunction -> "Rainbow"],
 ListPointPlot3D[houses[[All, {1, 2, -1}]], PlotStyle -> Black]]
```

Рис. 6 Регрессия на данных Boston Homes (Метод ближайших соседей)

1. Дерево строит кусочно-постоянную функцию, где каждому листу соответствует среднее значение целевой переменной.
2. Модель игнорирует плавные тренды, фокусируясь на локальных закономерностях.

Результаты метода (рис. 7, рис. 8):

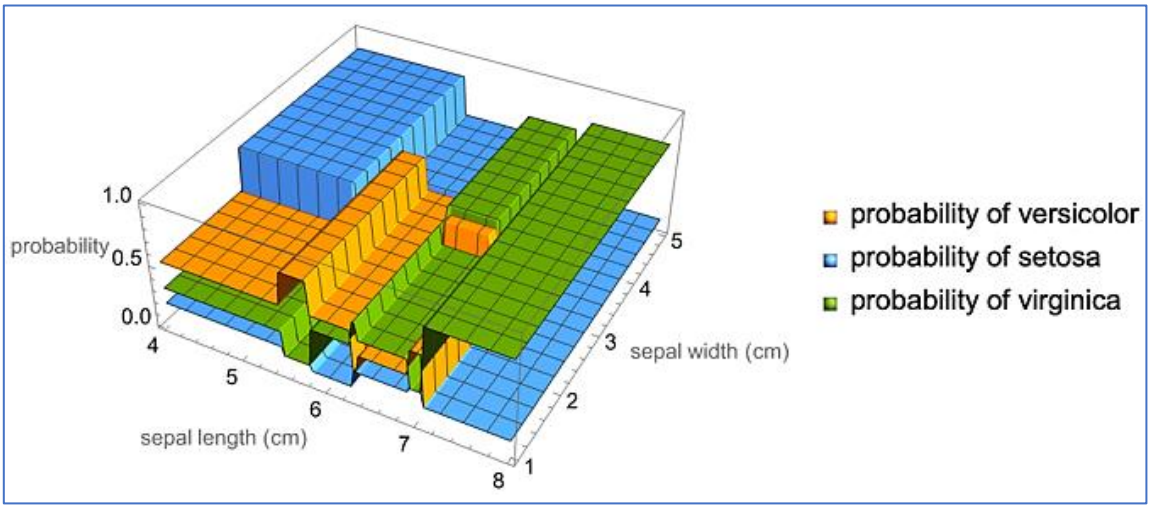

Рис. 7 Результат классификации на данных Iris (Метод ближайших соседей)

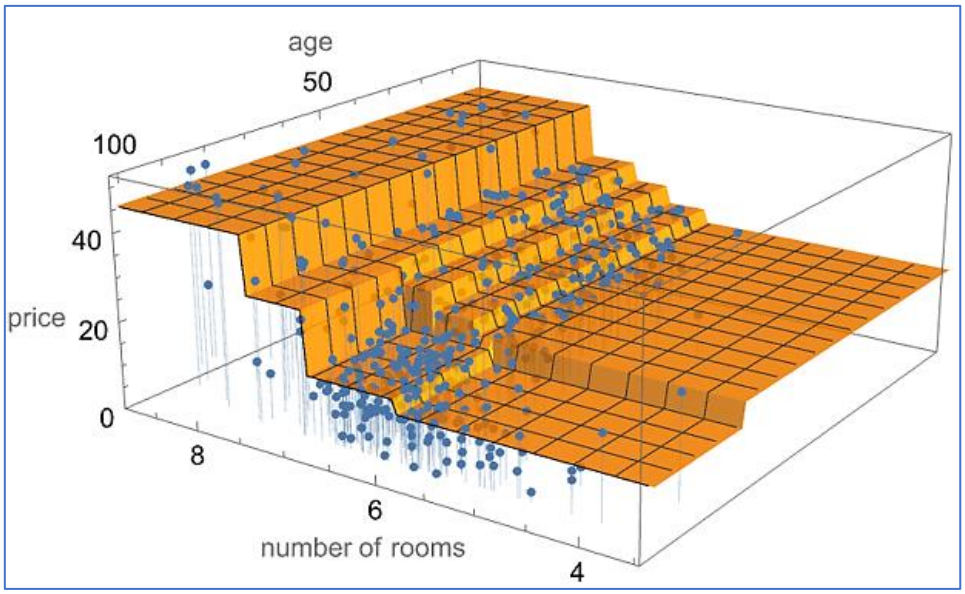

Рис. 8 Результат регрессии на данных Boston Homes (Метод ближайших соседей)

3. Сравнительный анализ

Сравнение характеристик KNN и DT (рис. 9):

Критерий	**KNN**	**Дерево решений**
Скорость обучения	$O(1)$ (нет обучения)	$O(N \cdot M \cdot \log N)$
Скорость предсказания	$O(N)$	$O(\log N)$
Чувствительность к K/глубине	Критична: малые K → переобучение, большие → недообучение	Глубина >10 → переобучение, малая → недообучение
Объяснимость	Низкая (кроме малых K)	Высокая (видимая структура узлов)
Устойчивость к шуму	Низкая (при $K = 1$)	Средняя (зависит от обрезки)
Производительность на малых данных	Высокая (при адекватной метрике)	Средняя (риск переобучения)
Производительность на больших данных	Низкая (вычислительная нагрузка)	Высокая (быстрое предсказание)
Работа с категориальными признаками	Требует преобразования в числовые	Поддерживает напрямую

Рис. 9 Сравнительный анализ Метода ближайших соседей и Дерева решений

• Скорость обучения: KNN не требует обучения, что делает его идеальным для динамически обновляемых данных. DT требует построения структуры, что занимает время, но предсказания выполняются мгновенно.

• Интерпретируемость: DT позволяет визуализировать логику решений, что критично в регулируемых отраслях (например, кредитование). KNN объясним только при малых *K*.

• Устойчивость к шуму: DT менее чувствителен к выбросам благодаря иерархическому разбиению. KNN с *K*=1 присваивает шумовым точкам собственные классы.

Рекомендации:

• KNN: используйте при малых данных, наличии естественной метрики, необходимости быстрого прототипирования.

• DT: выбирайте для задач с требованием интерпретируемости, больших данных, быстрого предсказания.

4. Выводы и заключение

Оба метода имеют уникальные преимущества. KNN проще в реализации, но страдает от высокой вычислительной нагрузки. DT обеспечивает прозрачность решений, но требует тщательной настройки. Для улучшения результатов KNN можно комбинировать с методами снижения размерности, а DT – использовать в ансамблях (случайные леса, градиентный бустинг). KNN и DT представляют два подхода к машинному обучению: первый основан на локальных аналогиях, второй – на иерархических правилах.

Рекомендации:

• Выбор KNN:

○ Малые данные (N <103).

○ Необходимость быстрого прототипирования.

○ Задачи с естественной метрикой (тексты, изображения).

• Выбор DT:

○ Требования к объяснимости (медицина, финансы).

○ Большие данные с категориальными признаками.

○ Интеграция в ансамбли (случайные леса).

Перспективы:

• Гибридные модели: комбинация KNN для поиска аналогов и DT для генерации правил.

• Оптимизация KNN: использование KD-деревьев для ускорения поиска соседей.

• Дистилляция знаний: обучение DT на "мягких" метках от нейросетей для улучшения точности.

Список литературы:

1. Кузнецов С. О. Методы и алгоритмы машинного обучения. – М.: БИНОМ, 2019.

2. Федоров А. В. Алгоритмы обработки данных: от теории к практике. – СПб.: Питер, 2021.

3. Сироткин, М. С. Исследование метода ближайших соседей в системе символьной математики Wolfram Mathematica / М. С. Сироткин, Д. В. Крюкова, Р. Е. Чернявский // Вектор научной мысли. – 2025. – № 3 (20). – С. 261-270. – DOI 10.58351/2949-2041.2025.20.3.008.