

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кемеровский государственный университет»
Институт цифры
Кафедра цифровых технологий

Беломытцев Владислав Романович

Искандиров Марат Ринатович

«МЕТОД БЛИЖАЙШЕГО СОСЕДА И ЕГО ОБОБЩЕНИЯ»

Курсовая работа

по направлению подготовки

**02.03.02 Фундаментальная информатика и информационные
технологии**

направленность (профиль) подготовки

«Информатика и компьютерные науки»

Научный руководитель:

Д. Т. Н., профессор
кафедры прикладной
математики

В. Н. Крутиков

Работа защищена с

оценкой отлично
« 14 » 12 2023г.

Кемерово 2023

Содержание

Оглавление

Оглавление

Введение	3
1. Основные понятия и определения.....	5
1.1 Объекты и признаки.....	5
1.2 Ответы и типы задач	6
1.3 Принцип максимума правдоподобия	6
1.4 ЕМ-алгоритм	7
2. Метод ближайшего соседа и его обобщения.....	8
2.1 Обобщенный метрический классификатор	8
2.2 Метод ближайших соседей.....	9
2.3 Метод парзеновского окна.....	12
2.4 Метод потенциальных функций.....	13
3. Применение метода ближайшего соседа	16
Заключение	21
Список литературы	22

Введение

Исследование операций – это раздел прикладной математики, ориентированный на разработку и применение математических методов и моделей для решения разнообразных задач в области управления и принятия решений. Среди множества методов, используемых в исследовании операций, особое внимание уделяется метрическим методам, представляющим собой эффективные инструменты для решения задач классификации, регрессии, и кластеризации данных.

Во многих прикладных задачах измерять степень сходства объектов существенно проще, чем сформировать признаковые описания. Например, такие сложные объекты, как фотографии лиц, подписи, временные ряды или первичные структуры белков естественнее сравнивать непосредственно с друг с другом путём некоторого “наложения с выравниванием”, чем изобретать какие-то признаки и сравнивать признаковые описания. Если мера сходства объектов введена достаточно удачно, то, как правило, оказывается, что схожим объектам очень часто соответствуют схожие ответы.

В задачах классификации это означает, что классы образуют компактно локализованные подмножества. Это предположение принято называть “гипотезой компактности”.

Для формализации понятия сходства вводится функция расстояния в пространстве объектов X . Методы обучения, основанные на анализе сходства объектов, будем называть метрическими, даже если функция расстояния не удовлетворяет всем аксиомам метрики (в частности, аксиоме треугольника). В англоязычной литературе употребляются термины *similarity-based learning* или *distance-based learning*.

Целью курсовой работы является исследование метода ближайшего соседа и его обобщений в контексте исследования операций. Мы стремимся изучить принципы и особенности работы указанных методов

Задачи работы:

1. Ознакомление с основными понятиями и определениями.

Рассмотрим базовые понятия и обозначения, которые будут использоваться в ходе курсовой работы

2. Рассмотрение метода ближайшего соседа.

Метод ближайшего соседа является одним из фундаментальных метрических методов. Мы исследуем его принципы работы, особенности применения, рассмотрим применение этого метода на практической задаче, а также оценим его преимущества и ограничения.

3. Рассмотрение обобщений метода ближайшего соседа.

Исследование обобщенных версий метода ближайшего соседа, таких как обобщенный метрический классификатор, метод парзеновского окна и метод потенциальных функций.

1. Основные понятия и определения

Задано множество *объектов* X , множество *допустимых* ответов Y , и существует *целевая функция* (target function) $y^*: X \rightarrow Y$, значения которой $y_i = y^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1, \dots, x_\ell\} \subset X$. Пары “объект-ответ” (x_i, y_i) называются *прецедентами*. Совокупность пар $X^\ell = (x_i, y_i)_{i=1}^\ell$ называется *обучающей выборкой* (training sample).

Задача обучения по прецедентам заключается в том, чтобы по выборке X^ℓ восстановить зависимость y^* , то есть построить *решающую функцию* (decision function) $a: X \rightarrow Y$, которая приближала бы целевую функцию $y^*(x)$, причем не только на объектах обучающей выборки, но и на всём множестве X .

Решающая функция a должна допускать эффективную компьютерную реализацию; по этой причине будем называть её алгоритмом.

1.1 Объекты и признаки

Признак (feature) f объекта x - это результат измерения некоторой характеристики объекта. Формально признаком называется отображение $f: X \rightarrow D_f$, где D_f - множество допустимых значений признака.

В частности, любой алгоритм $a: X \rightarrow Y$ также можно рассматривать как признак.

В зависимости от природы множества D_f признаки делятся на несколько типов.

Если $D_f = \{0, 1\}$, то f - *бинарный* признак;

Если D_f - конечное множество, то f - *номинальный* признак;

Если D_f - конечное упорядоченное множество, то f - *порядковый* признак;

Если $D_f = \mathbb{R}$, то f - *количественный* признак.

Если все признаки имеют одинаковый тип, $D_{f1} = \dots = D_{fn}$, то исходные данные называются *однородными*, в противном случае - *разнородными*.

1.2 Ответы и типы задач

В зависимости от природы множества допустимых ответов Y задачи обучения по прецедентам делятся на следующие типы.

Если $Y = \{1, \dots, M\}$, то это задача *классификации* (classification) на M непересекающихся классов. В этом случае всё множество объектов X разбивается на классы $K_y = \{x \in X: y^*(x) = y\}$, и алгоритм $a(x)$ должен давать ответ на вопрос “к какому классу принадлежит x ?”. В некоторых приложениях классы называют *образами* и говорят о задаче *распознавания образов* (pattern recognition).

Если $Y = \{1, 0\}^M$, то это задача *классификации* на M *пересекающихся классов*. В простейшем случае эта задача сводится к решению M независимых задач классификации с двумя непересекающимися классами.

Если $Y = R$, то это задача *восстановления регрессии* (regression estimation).

Задачи *прогнозирования* (forecasting) являются частными случаями классификации или восстановления регрессии, когда $x \in X$ - описание прошлого поведения объекта x , $y \in X$ - описание некоторых характеристик его будущего поведения.

1.3 Принцип максимума правдоподобия

Составляет основу параметрического подхода. Пусть задано множество объектов $X_m = \{x_1, \dots, x_m\}$, выбранных независимо друг от друга из вероятностного распределения с плотностью $\phi(x; \theta)$. *Функцией правдоподобия* называется совместная плотность распределения всех объектов выборки:

$$\rho(X^m; \theta) = \rho(x_1, \dots, x_m; \theta) = \prod_{i=1}^m \phi(x_i; \theta).$$

Значение параметра θ , при котором выборка максимально правдоподобна, то есть функция $\rho(X_m; \theta)$ принимает максимальное значение, называется *оценкой максимума правдоподобия*. Как известно из математической статистики, эта оценка обладает рядом оптимальных свойств [13, 16].

Вместо максимизации правдоподобия удобнее максимизировать его логарифм:

$$L(X^m; \theta) = \sum_{i=1}^m \ln \varphi(x_i; \theta) \rightarrow \max_{\theta}.$$

Для решения этой задачи можно использовать стандартные методы оптимизации. В некоторых случаях решение выписывается в явном виде, исходя из необходимого условия оптимума (если функция $\varphi(x; \theta)$ достаточно гладкая по параметру θ):

$$\frac{\partial}{\partial \theta} L(X^m; \theta) = \sum_{i=1}^m \frac{\partial}{\partial \theta} \ln \varphi(x_i; \theta) = 0.$$

1.4 ЕМ-алгоритм

К сожалению, попытка разделить смесь, используя принцип максимума правдоподобия “в лоб”, приводит к слишком громоздкой оптимизационной задаче.

Обойти эту трудность позволяет алгоритм ЕМ (expectation-maximization). Идея алгоритма заключается в следующем. Искусственно вводится вспомогательный вектор *скрытых* (hidden) переменных G , обладающий двумя замечательными свойствами. С одной стороны, он может быть вычислен, если известны значения вектора параметров Θ . С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

ЕМ-алгоритм состоит из итерационного повторения двух шагов. На Е-шаге вычисляется ожидаемое значение (expectation) вектора скрытых переменных G по текущему приближению вектора параметров Θ . На М-шаге решается задача максимизации правдоподобия (maximization) и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ .

Алгоритм 2.1. Общая идея ЕМ-алгоритма

- 1: Вычислить начальное приближение вектора параметров Θ ;
- 2: повторять
- 3: $G := \text{EStep}(\Theta)$;
- 4: $\Theta := \text{MStep}(\Theta, G)$;
- 5: пока Θ и G не стабилизируются.

2. Метод ближайшего соседа и его обобщения

Пусть на множестве объектов X задана функция расстояния $\rho: X \times X \rightarrow [0, \infty)$. Существует целевая зависимость $y^*: X \rightarrow Y$, значения которой известны только на объектах обучающей выборки $X^\ell = (x_i, y_i)_{i=1}^\ell$, $y_i = y^*(x_i)$. Множество классов Y конечно. Требуется построить алгоритм классификации $a: X \rightarrow Y$, аппроксимирующий целевую зависимость $y^*(x)$ на всём множестве X .

2.1 Обобщенный метрический классификатор

Для произвольного объекта $u \in X$ расположим элементы обучающей выборки x_1, \dots, x_ℓ в порядке возрастания расстояний до u :

$$\rho(u, x_u^{(1)}) \leq \rho(u, x_u^{(2)}) \leq \dots \leq \rho(u, x_u^{(\ell)}),$$

где через $x_u^{(i)}$ обозначается i -й сосед объекта u . Соответственно, ответ на i -м соседе объекта u есть $y_u^{(i)} = y^*(x_u^{(i)})$. Таким образом, любой объект $u \in X$ порождает свою перенумерацию выборки.

Опр. 1 *Метрический алгоритм классификации с обучающей выборкой X^ℓ относит объект u к тому классу $y \in Y$, для которого суммарный вес ближайших обучающих объектов $\Gamma_y(u, X^\ell)$ максимален:*

$$a(u; X^\ell) = \arg_{y \in Y} \max \Gamma_y(u, X^\ell); \quad \Gamma_y(u, X^\ell) = \sum_{i=1}^\ell [y_u^{(i)} = y] \omega(i, u); \quad (3.1)$$

где весовая функция $\omega(i, u)$ оценивает степень важности i -го соседа для классификации объекта u . Функция $\Gamma_y(u, X^\ell)$ называется оценкой близости объекта u к классу y .

Метрический классификатор определён с точностью до весовой функции $\omega(i, u)$. Обычно она выбирается неотрицательной, не возрастающей по i . Это соответствует гипотезе компактности, согласно которой чем ближе объекты u и $x_u^{(i)}$, тем выше шансы, что они принадлежат одному классу.

Обучающая выборка X^ℓ играет роль параметра алгоритма a . Настройка сводится к запоминанию выборки, и, возможно, оптимизации каких-то параметров весовой функции, однако сами объекты не подвергаются обработке и сохраняются «как есть». Алгоритм $a(u; X^\ell)$ строит локальную аппроксимацию выборки X^ℓ , причём вычисления откладываются до момента, пока не станет известен объект u . По этой причине метрические алгоритмы относятся к методам *ленивого обучения* (lazy learning), в отличие от *усердного обучения* (eager learning), когда на этапе обучения строится функция, аппроксимирующая выборку.

Метрические алгоритмы классификации относятся также к методам *рассуждения по прецедентам* (case-based reasoning, CBR). Здесь действительно можно говорить о «рассуждении», так как на вопрос «почему объект u был отнесен к классу y ?» алгоритм может дать понятное экспертам объяснение: «потому, что имеются схожие с ним прецеденты класса y », и предъявить список этих прецедентов.

Выбирая весовую функцию $\omega(i, u)$, можно получать различные метрические классификаторы, которые подробно рассматриваются ниже.

2.2 Метод ближайших соседей

Алгоритм ближайшего соседа (nearest neighbor, NN) относит классифицируемый объект $u \in X^\ell$ к тому классу, к которому принадлежит ближайший обучающий объект:

$$\omega(i, u) = [i = 1]; \quad a(u; X^\ell) = y_u^{(1)}.$$

Этот алгоритм является, по всей видимости, самым простым классификатором.

Обучение NN сводится к запоминанию выборки X^ℓ . Единственное достоинство этого алгоритма — простота реализации. Недостатков гораздо больше:

- Неустойчивость к погрешностям. Если среди обучающих объектов есть *выброс* — объект, находящийся в окружении объектов чужого класса, то

не только он сам будет классифицирован неверно, но и те окружающие его объекты, для которых он окажется ближайшим.

- Отсутствие параметров, которые можно было бы настраивать по выборке. Алгоритм полностью зависит от того, насколько удачно выбрана метрика ρ .
- В результате — низкое качество классификации.

Алгоритм k ближайших соседей (k nearest neighbors, k NN). Чтобы сгладить влияние выбросов, будем относить объект u к тому классу, элементов которого окажется больше среди k ближайших соседей $x_u^{(i)}, i = 1, \dots, k$:

$$\omega(i, u) = [i \leq k]; \quad a(u; X^\ell, k) = \arg_{y \in Y} \max \sum_{i=1}^k [y_u^{(i)} = y].$$

При $k = 1$ этот алгоритм совпадает с предыдущим, следовательно, неустойчив к шуму. При $k = \ell$, наоборот, он чрезмерно устойчив и вырождается в константу. Таким образом, крайние значения k нежелательны. На практике оптимальное значение параметра k определяют по критерию скользящего контроля с *исключением объектов по одному* (leave-one-out, LOO). Для каждого объекта $x_i \in X^\ell$ проверяется, правильно ли он классифицируется по своим k ближайшим соседям.

$$LOO(k, X^\ell) = \sum_{i=1}^{\ell} [a(x_i; X^\ell \setminus \{x_i\}, k) \neq y_i] \rightarrow \min_k.$$

Заметим, что если классифицируемый объект x_i не исключать из обучающей выборки, то ближайшим соседом x_i всегда будет сам x_i , и минимальное (нулевое) значение функционала $LOO(k)$ будет достигаться при $k = 1$.

Существует и альтернативный вариант метода k NN: в каждом классе выбирается k ближайших к u объектов, и объект u относится к тому классу, для которого среднее расстояние до k ближайших соседей минимально.

Алгоритм k взвешенных ближайших соседей. Недостаток k NN в том, что максимум может достигаться сразу на нескольких классах. В задачах с двумя классами этого можно избежать, если взять нечётное k . Более общая тактика,

которая годится и для случая многих классов — ввести строго убывающую последовательность вещественных весов w_i , задающих вклад i -го соседа в классификацию:

$$\omega(i, u) = [i \leq k] \omega_i; \quad a(u; X^\ell, k) = \arg_{y \in Y} \max \sum_{i=1}^k [y_u^{(i)} = y] \omega_i.$$

Выбор последовательности w_i является эвристикой. Если взять линейно убывающие веса $w_i = \frac{k+1-i}{k}$, то неоднозначности также могут возникать, хотя и реже (пример: классов два; первый и четвёртый сосед голосуют за класс 1, второй и третий — за класс 2; суммы голосов совпадают). Неоднозначность устраняется окончательно, если взять нелинейно убывающую последовательность, скажем, геометрическую прогрессию: $\omega_i = q^i$, где знаменатель прогрессии $q \in (0,1)$ является параметром алгоритма. Его можно подбирать по критерию LOO, аналогично числу соседей k .

Недостатки простейших метрических алгоритмов типа k NN.

- Приходится хранить обучающую выборку целиком. Это приводит к неэффективному расходу памяти и чрезмерному усложнению решающего правила. При наличии погрешностей (как в исходных данных, так и в модели сходства ρ) это может приводить к понижению точности классификации вблизи границы классов. Имеет смысл отбирать минимальное подмножество *эталонных объектов*, действительно необходимых для классификации.
- Поиск ближайшего соседа предполагает сравнение классифицируемого объекта со всеми объектами выборки за $O(\ell)$ операций. Для задач с большими выборками или высокой частотой запросов это может оказаться накладно. Проблема решается с помощью эффективных алгоритмов поиска ближайших соседей, требующих в среднем $O(\ln \ell)$ операций.
- В простейших случаях метрические алгоритмы имеют крайне бедный набор параметров, что исключает возможность настройки алгоритма по данным

2.3 Метод парзеновского окна

Ещё один способ задать веса соседям — определить w_i как функцию от расстояния $\rho(u, x_u^{(i)})$, а не от ранга соседа i . Введём функцию ядра $K(z)$, невозрастающую на $[0, \infty)$. Положив $w(i, u) = K(\frac{1}{h}\rho(u, x_u^{(i)}))$ в общей формуле (3.1), получим алгоритм

$$a(u; X^\ell, h) = \arg_{y \in Y} \max \sum_{i=1}^{\ell} [y_u^{(i)} = y] K\left(\frac{\rho(u, x_u^{(i)})}{h}\right) \quad (3.2)$$

Параметр h называется *шириной окна* и играет примерно ту же роль, что и число соседей k . «Окно» — это сферическая окрестность объекта u радиуса h , при попадании в которую обучающий объект x_i «голосует» за отнесение объекта u к классу y_i . Мы пришли к этому алгоритму чисто эвристическим путём, однако он имеет более строгое обоснование в байесовской теории классификации, и, фактически, совпадает с методом парзеновского окна.

Параметр h можно задавать априори или определять по скользящему контролю. Зависимость $LOO(h)$, как правило, имеет характерный минимум, поскольку слишком узкие окна приводят к неустойчивой классификации; а слишком широкие — к вырождению алгоритма в константу.

Фиксация ширины окна h не подходит для тех задач, в которых обучающие объекты существенно неравномерно распределены по пространству X . В окрестности одних объектов может оказываться очень много соседей, а в окрестности других — ни одного. В этих случаях применяется окно *переменной ширины*. Возьмём *финитное ядро* — невозрастающую функцию $K(z)$, положительную на отрезке $[0, 1]$, и равную нулю вне его. Определим h как наибольшее число, при котором ровно k ближайших соседей объекта u получают ненулевые веса: $h(u) = \rho(u, x_u^{(k+1)})$. Тогда алгоритм принимает вид

$$a(u; X^\ell, k) = \arg_{y \in Y} \max \sum_{i=1}^k [y_u^{(i)} = y] K\left(\frac{\rho(u, x_u^{(i)})}{\rho(u, x_u^{(k+1)})}\right) \quad (3.3)$$

Заметим, что при финитном ядре классификация объекта сводится к поиску его соседей, тогда как при не финитном ядре (например, гауссовском) требуется перебор всей обучающей выборки.

2.4 Метод потенциальных функций

В методе парзеновского окна центр радиального ядра $K_h(u, x) = K(\frac{1}{h}\rho(u, x))$ помещается в классифицируемый объект u . В силу симметричности функции расстояния $\rho(u, x)$ возможен и другой, двойственный, взгляд на метрическую классификацию. Допустим, что ядро помещается в каждый обучающий объект x_i и «притягивает» объект u к классу y_i , если он попадает в его окрестность радиуса h_i :

$$a(u; X^\ell) = \arg_{y \in Y} \max \sum_{i=1}^{\ell} [y_i = y] \gamma_i K\left(\frac{\rho(u, x_i)}{h_i}\right), \gamma_i \geq 0, h_i > 0. \quad (3.4)$$

По сути, эта формула отличается от (3.3) только тем, что здесь ширина окна h_i зависит от обучающего объекта x_i , а не от классифицируемого объекта u

Алгоритм 3.1. Метод потенциальных функций

Вход:

X^ℓ — обучающая выборка;

Выход:

Коэффициенты $\gamma_i, i = 1, \dots, \ell$ в (3.4);

1: Инициализация: $\gamma_i = 0; i = 1, \dots, \ell$;

2: **повторять**

3: выбрать объект $x_i \in X^\ell$;

4: **если** $a(x_i) \neq y_i$ **то**

5: $\gamma_i := \gamma_i + 1$;

6: **пока** число ошибок на выборке не окажется достаточно мало

Данная идея лежит в основе *метода потенциальных функций* и имеет прямую физическую аналогию с электрическим потенциалом. При $Y = \{-1, +1\}$ обучающие объекты можно понимать как положительные и отрицательные электрические заряды; коэффициенты γ_i — как абсолютную величину этих

зарядов; ядро $K(z)$ — как зависимость потенциала от расстояния до заряда; а саму задачу классификации — как ответ на вопрос: какой знак имеет электростатический потенциал в заданной точке пространства u . Заметим, что в электростатике $K(z) = \frac{1}{z}$ либо $\frac{1}{z+a}$, однако для наших целей совершенно не обязательно брать именно такое ядро.

Алгоритм (3.4) имеет достаточно богатый набор из 2ℓ параметров γ_i, h_i . Простейший и исторически самый первый метод их настройки представлен в Алгоритме 3.1. Он настраивает только веса γ_i , предполагая, что радиусы потенциалов h_i и ядро K выбраны заранее. Идея очень проста: если обучающий объект x_i классифицируется неверно, то потенциал класса y_i недостаточен в точке x_i , и вес γ_i увеличивается на единицу. Выбор объектов на шаге 3 лучше осуществлять не подряд, а в случайном порядке.

Достоинство Алгоритма 3.1 в том, что он очень эффективен, когда обучающие объекты поступают потоком, и хранить их в памяти нет возможности или необходимости. В те годы, когда метод потенциальных функций был придуман, хранение выборки действительно было большой проблемой. В настоящее время такой проблемы нет, и Алгоритм 3.1 представляет скорее исторический интерес.

Недостатков у Алгоритма 3.1 довольно много: он медленно сходится; результат обучения зависит от порядка предъявления объектов; слишком грубо (с шагом 1) настраиваются веса γ_i ; центры потенциалов почему-то помещаются только в обучающие объекты; задача минимизации числа потенциалов (ненулевых γ_i) вообще не ставится; вообще не настраиваются параметры h_i . В результате данный алгоритм не может похвастаться высоким качеством классификации.

Более современный метод настройки линейных комбинаций потенциальных функций основывается на ЕМ-алгоритме, приведенном в пункте

1.4

Он позволяет оптимизировать и ширину каждого потенциала, и положения центров, и даже количество потенциалов. Изменилось и название метода: теперь потенциальные функции предпочитают называть *радиальными базисными функциями*. Формально этот метод не подчиняется общей формуле (3.1), так как классифицируемый объект и сравнивается не с обучающими объектами, а с центрами потенциалов, которые в общем случае не совпадают ни с одним из обучающих объектов.

3. Применение метода ближайшего соседа

Для примера использования метода ближайшего соседа возьмем следующую задачу:

Есть датасет с 2 признаками - x_1 , x_2 и целевым (class) значением 0 или 1. Необходимо классифицировать исходные данные, используя метод ближайшего соседа KNN

Датасет имеет следующий вид:

Таблица 1 – Датасет

index	x_1	x_2	class
0	-6.005674	-4.854007	1
1	0.937182	-10.939275	1
2	8.158951	-7.719885	1
3	4.814693	-2.841937	1
4	-3.602122	-10.576468	1
...
95	-6.006420	-1.944295	1
96	8.083901	-3.041803	1
97	6.209541	1.918128	1
98	3.243569	-9.595519	0
99	4.864893	-3.489692	0

Листинг кода скрипта по визуализации начальных и конечных данных, обучению модели и использования модели для предсказания значений:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_excel('knn_data.xlsx')

# Визуализация начальных данных (рис. 1)
plt.scatter(df['x1'], df['x2'], c=df['class'])
plt.show()

X = df[['x1', 'x2']]
y = df['class']

# Разделим набор данных
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.6, random_state=42)

# Для примера возьмем 6, можно выбрать другое число и посмотреть как
изменится качество
neigh = KNeighborsClassifier(n_neighbors=6)

# Обучение модели
neigh.fit(X_train, y_train)

# Доля правильных ответов
print("Доля правильных ответов:", neigh.score(X_test, y_test))
```

```
new_df = pd.DataFrame(neigh.predict(X))  
print(new_df)
```

```
# Визуализация классифицированных начальных данных (рис. 2)  
plt.scatter(df['x1'], df['x2'], c=new_df)  
plt.show()
```

```
# Генерация новых данных  
new_data = make_blobs(centers=1)  
new_data_df = pd.DataFrame(new_data[0])
```

```
# Получение предсказаний для новых данных  
new_df = pd.DataFrame(neigh.predict(new_data[0]))
```

```
# Визуализация сгенерированных данных (рис. 3)  
plt.scatter(new_data_df[0], new_data_df[1], c="green")  
plt.show()
```

```
# Визуализация классифицированных сгенерированных данных (рис. 4)  
plt.scatter(new_data_df[0], new_data_df[1], c=new_df)  
plt.show()
```

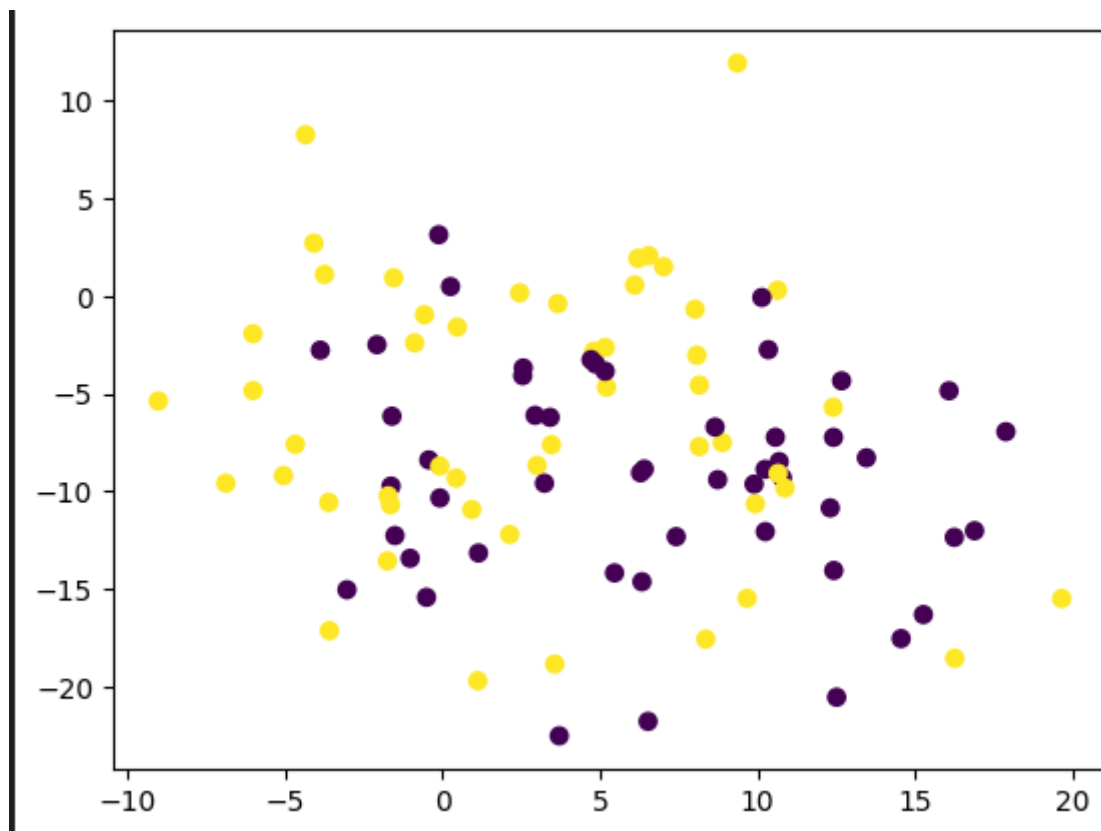


Рис.1 Визуализация начальных данных

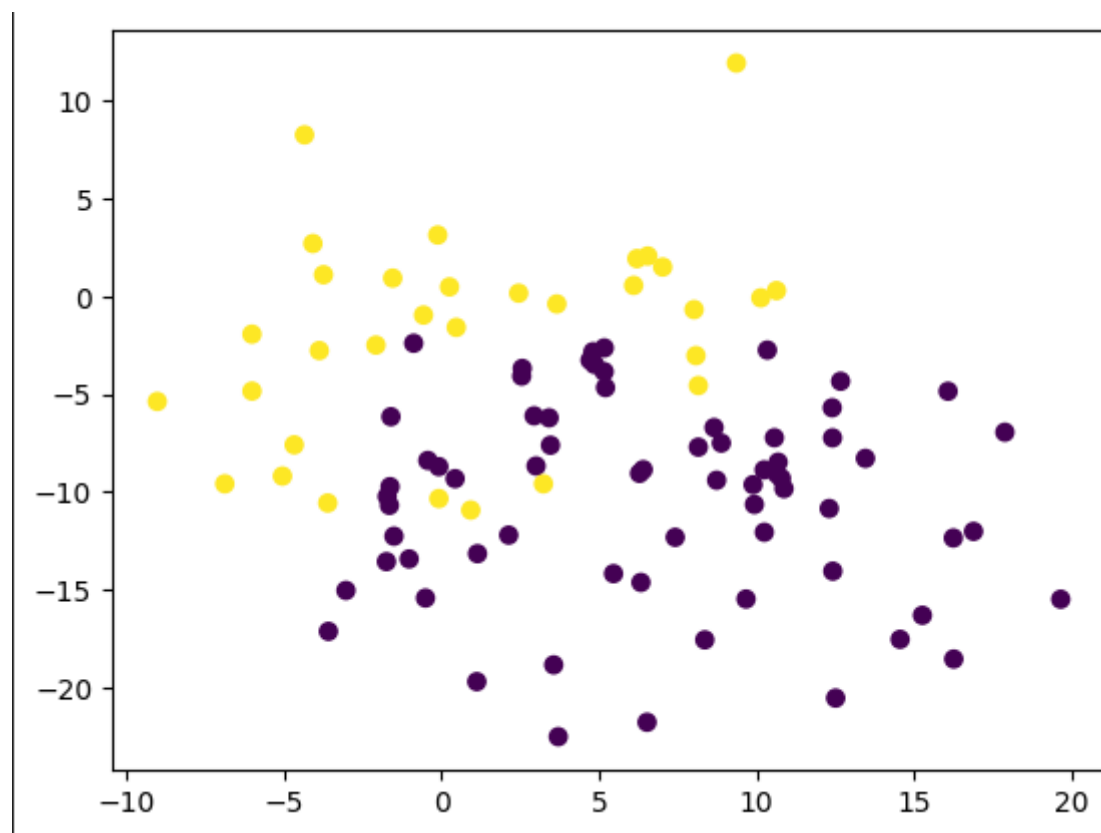


Рис.2 Визуализация классифицированных начальных данных

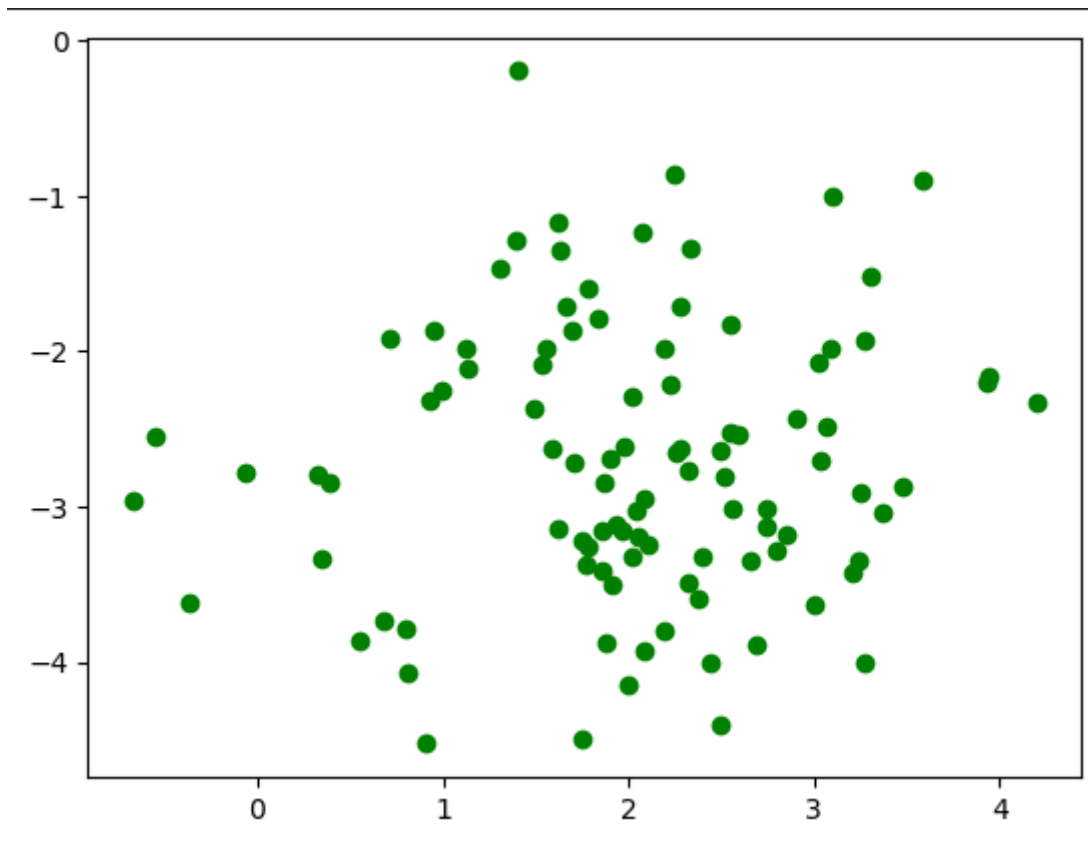


Рис.3 Визуализация сгенерированных данных

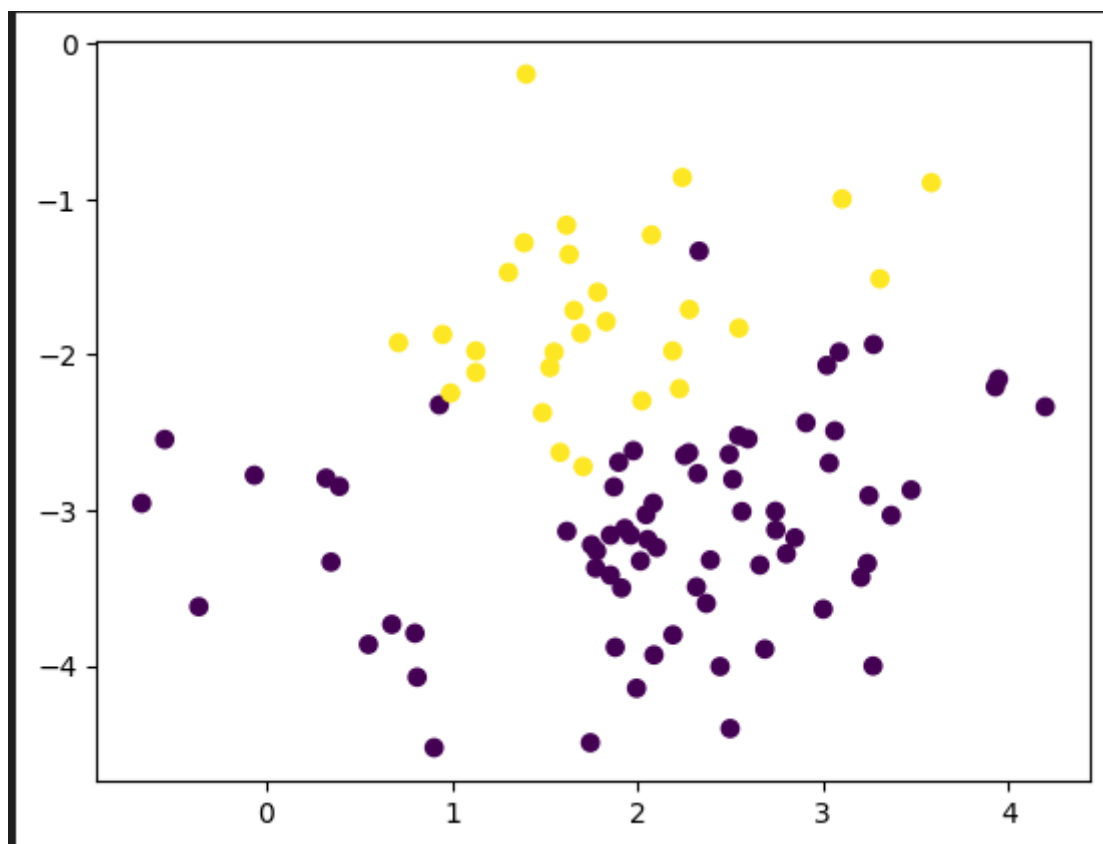


Рис.4 Визуализация классифицированных сгенерированных данных

Заключение

Метода ближайшего соседа имеет широкое применение во многих задачах, связанных с классификацией.

В ходе курсовой работы изучен метод ближайшего соседа и рассмотрены его обобщения.

Для применения метода ближайшего соседа была использована библиотека `sklearn`. Были сгенерированы обучающие и тестовые данные. Приведён скрипт, в котором происходит обучение модели и классификация тестовых данных.

Список литературы

1. Математические методы обучения по прецедентам (теория обучения машин) / К. В. Воронцов // 141
2. https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py, 06.12.2023

**Рецензия на курсовую работу
по дисциплине «Исследование операций»**

Тема работы: Метод ближайшего соседа и его обобщения

Автор: Беломытцев Владислав, Искандиров Марат

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Руководитель: Крутиков Владимир Николаевич, ДТН, проф. каф. прикладной математики
(Фамилия И.О., должность, ученая степень)

Оценка качества выполнения курсовой работы

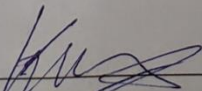
Критерии оценки	Оценка
1. Общее оформление работы (0 – 10 баллов)	9
2. Оформление титульного листа (0 – 1 балл)	1
3. Оформление и содержательность оглавления (0 – 1 балл)	1
4. Обоснование актуальности работы (0 – 1 балл)	1
5. Обоснование практической значимости работы (0 – 2 балла)	2
6. Описание истории исследуемых вопросов (0 – 2 балла)	2
7. Содержательность описания работы во введении (0 – 1 балл)	1
8. Постановка цели работы (0 – 2 балла)	2
9. Содержательность описания основных понятий и обозначений, терминов и определений (0 – 10 баллов)	9
10. Обоснованность приведения и использования основных утверждений и теорем (0 – 10 баллов)	9
11. Содержательность и обоснованность использования методов решения поставленной задачи (0 – 15 баллов)	14
12. Правильность решения поставленной задачи, содержательность описания этапов решения (0 – 35 баллов)	34
13. Содержательность сделанных выводов (0 – 3 балла)	3
14. Содержательность заключения и описания результатов работы (0 – 2 балла)	2
15. Обоснованность предложений по применению полученных результатов (0 – 2 балла)	2
16. Оформление списка литературы (0 – 3 балла)	3
Итого:	95

Отмеченные недостатки

Оценочная шкала: максимальная сумма баллов – 100.

Количество полученных баллов	Оценка
0-50	неудовлетворительно
51-65	удовлетворительно
66-85	хорошо
86-100	отлично

Оценка работы: отлично

Научный руководитель: 

« 14 » 12 2023 г.