

Лекция 2. Процесс разработки проекта машинного обучения. Подготовка данных. Оценка качества алгоритмов обучения с учителем. Метод ближайших соседей.

Глинский А.В.

Цель занятия: В этом модуле мы познакомимся с такими темами, как процесс разработки проекта машинного обучения, подготовка данных для задач машинного обучения, оценка качества алгоритмов обучения с учителем, а также изучим один из самых простых и интуитивно понятных алгоритмов в машинном обучении - метод ближайших соседей.

Содержание

0.1	Процесс разработки проекта машинного обучения	3
0.1.1	Вопросы для самопроверки	4
0.1.2	Резюме по разделу	5
0.2	Подготовка данных	6
0.2.1	Исследование данных	6
0.2.2	Очистка и исправление данных	6
0.2.3	Разделение выборки на тренировочную и тестовую	6
0.2.4	Виды признаков	7
0.2.5	Предобработка признаков	9
0.2.6	Создание новых признаков	15
0.2.7	Отбор признаков	16
0.2.8	Реализация в Python	16
0.2.9	Вопросы для самопроверки	16
0.2.10	Резюме по разделу	17
0.3	Оценка качества алгоритмов обучения с учителем	18
0.3.1	Метрики качества регрессии	18
0.3.2	Метрики качества классификации	21
0.3.3	Обобщающая способность алгоритмов и переобучение	31
0.3.4	Вопросы для самопроверки	31
0.3.5	Резюме по разделу	32
0.4	Метод ближайших соседей	33
0.4.1	Визуальная демонстрация алгоритма	33
0.4.2	Описание алгоритма	33
0.4.3	Подготовка данных для алгоритма	35
0.4.4	Оценка качества алгоритма	35
0.4.5	Модификации алгоритма	35
0.4.6	Область применения алгоритма	36
0.4.7	Плюсы и минусы алгоритма	36
0.4.8	Реализация алгоритма в Python	37
0.4.9	Вопросы для самопроверки	37
0.4.10	Резюме по разделу	38
0.5	Резюме по модулю	39

0.1 Процесс разработки проекта машинного обучения

Процесс разработки проекта машинного обучения обычно состоит из следующих шагов:

Этап 1: определение задачи

1. **Определение задачи:** на этом этапе вы определяете, какую задачу необходимо решить. Например, это может быть задача классификации или регрессии.

Этап 2: подготовка данных

1. **Сбор данных:** сбор данных, необходимых для обучения модели. Можно собрать данные самостоятельно или использовать уже существующие наборы данных.
2. **Исследование данных (англ. Exploratory Data Analysis, EDA):** это процесс исследования и анализа данных, используемый для получения полезной информации и выявления особенностей данных перед применением модели машинного обучения. EDA включает в себя методы визуализации и статистические техники, которые помогают понять данные и выделить важные характеристики.
3. **Очистка и исправление данных:** очистка данных от пропусков, дубликатов и ошибок.
4. **Разделение выборки:** разделение выборки на тренировочную и тестовую является важным шагом в машинном обучении, который помогает оценить качество модели на новых данных. Тренировочная выборка используется для обучения модели, тестовая выборка - для проверки ее качества.
5. **Предобработка признаков:** преобразование признаков, нормализация признаков и т.д.
6. **Создание новых признаков:** создание новых признаков - это важный шаг в построении моделей машинного обучения. Новые признаки могут улучшить качество модели и помочь ей лучше выявлять закономерности в данных. Существует несколько способов создания новых признаков:
 - **Инженерия признаков на основе знаний об отрасли:** в этом подходе эксперты отрасли и/или аналитики создают новые признаки на основе своих знаний о предметной области. Это может включать в себя создание признаков на основе социально-экономических показателей, метеорологических данных, исторических событий и т.д.
 - **Преобразование существующих признаков:** в этом подходе существующие признаки могут быть преобразованы в новые признаки путем использования математических операций, например, логарифмирование, возвведение в степень, извлечение корня и т.д.
 - **Создание признаков на основе текстов:** при анализе текстовых данных новые признаки могут быть созданы путем извлечения ключевых слов и фраз, использования стемминга, лемматизации, а также при помощи методов машинного обучения для анализа тональности и определения тем.
 - **Создание признаков на основе временных рядов:** при анализе временных рядов новые признаки могут быть созданы путем вычисления статистических показателей, таких как скользящее среднее, стандартное отклонение, корреляция и т.д.

Важно понимать, что создание новых признаков может привести к увеличению размерности данных, что может усложнить обработку и анализ. Поэтому необходимо учитывать баланс между количеством признаков и качеством модели, а также проводить валидацию модели и контролировать переобучение.

7. **Отбор признаков:** отбор признаков может быть выполнен двумя способами: автоматически и вручную. Автоматический отбор признаков включает использование алгоритмов, которые оценивают важность каждого признака и выбирают только наиболее значимые. Вручную выбранные признаки обычно определяются на основе экспертного знания в предметной области.

Этап 3: разработка модели МО

1. **Выбор модели:** на этом этапе вы выбираете модель машинного обучения, которая подходит для решения вашей задачи. В зависимости от задачи можно использовать различные типы моделей, такие как метрические модели, линейная регрессия, деревья решений и т.д.
2. **Обучение и настройка модели:** после выбора модели и подготовки данных вы можете обучить модель на тренировочных данных. Обычно это делается путем минимизации функции потерь с помощью градиентного спуска или других алгоритмов оптимизации. Если модель не дает достаточно хороших результатов, ее можно настроить, изменив гиперпараметры модели или используя другую модель.
3. **Оценка и валидация модели:** после обучения модели необходимо оценить ее качество на отложенных данных, которые модель не видела в процессе обучения. Это можно сделать с помощью различных метрик, таких как точность, F1-мера и т.д. После настройки модели ее необходимо проверить на тестовых данных, чтобы убедиться в ее работоспособности на новых данных.

Этап 4: запуск модели

1. **Развертывание модели:** когда модель прошла все тесты и была удовлетворительно протестирована на новых данных, ее можно развернуть в производство, где она будет использоваться для решения реальных задач.
2. **Обслуживание модели:** после того, как модель развернута в производство, ее необходимо периодически обновлять и обслуживать, чтобы она продолжала давать актуальные результаты.

0.1.1 Вопросы для самопроверки

Какие задачи могут быть выбраны на этапе 1 (определение задачи) процесса разработки проекта машинного обучения:

1. классификация
2. регрессия
3. машинное обучение
4. численные методы

Правильные ответы: 1, 2

Этап 2 (подготовка данных) может включать в себя:

1. Сбор данных
2. Выбор модели
3. Разделение выборки
4. Создание новых признаков

Правильные ответы: 1, 3, 4

Выберите правильную последовательность шагов для этапа 3 (разработка модели МО):

1. Обучение и настройка модели -> Выбор модели -> Оценка и валидация модели
2. Выбор модели -> Обучение и настройка модели -> Оценка и валидация модели
3. Обучение и настройка модели -> Оценка и валидация модели -> Выбор модели
4. Оценка и валидация модели -> Обучение и настройка модели -> Выбор модели

Правильные ответы: 2

0.1.2 Резюме по разделу

Процесс разработки проекта машинного обучения состоит из нескольких этапов:

1. Определение задачи
2. Подготовка данных
3. Разработка модели МО
4. Запуск модели

В свою очередь эти этапы разбиваются на несколько шагов.

Этап 1: определение задачи

Этап 2: подготовка данных

1. Сбор данных
2. Исследование данных
3. Очистка и исправление данных
4. Разделение выборки
5. Предобработка признаков
6. Создание новых признаков
7. Отбор признаков

Этап 3: разработка модели МО

1. Выбор модели
2. Обучение модели
3. Оценка модели
4. Настройка модели
5. Валидация модели

Этап 4: запуск модели

1. Развёртывание модели
2. Обслуживание модели

0.2 Подготовка данных

Цель занятия: ученик может подготовить данные для моделей машинного обучения

0.2.1 Исследование данных

Исследование данных (EDA) позволяет понять данные, выявить скрытые связи и паттерны, определить выбросы и пропущенные значения, а также принять решение о необходимости дополнительной предобработки данных. EDA может включать в себя следующие шаги:

- **Визуализация данных.** Этот шаг включает в себя создание графиков и диаграмм, чтобы визуализировать данные и выявить скрытые закономерности и паттерны.
- **Изучение связей между различными переменными** и определение, какие переменные сильно влияют на другие.

В результате исследования данных мы получаем лучшее понимание данных, что помогает лучше выбрать подходящую модель машинного обучения, провести более эффективную предобработку данных и принять правильные решения на этапе обучения модели.

0.2.2 Очистка и исправление данных

Очистка данных - это процесс предварительной обработки данных, направленный на удаление или исправление ошибочных, неактуальных, неполных, поврежденных, дублированных или неправильно форматированных данных, чтобы улучшить качество и точность модели машинного обучения.

Ниже перечислены некоторые методы очистки данных в машинном обучении:

- **Удаление дубликатов.** Дубликаты могут привести к неверным результатам, поэтому необходимо удалить все дубликаты в данных.
- **Исправление ошибок.** В данных могут быть ошибки, например, опечатки или неверные значения. Такие ошибки могут быть исправлены с помощью автоматических методов, таких как исправление опечаток.

0.2.3 Разделение выборки на тренировочную и тестовую

Разбиение выборки на обучающую и тестовую является одним из ключевых этапов машинного обучения, который позволяет оценить качество работы модели на новых данных. При этом часть данных используется для обучения модели (обучающая выборка), а оставшаяся часть - для проверки ее качества (тестовая выборка). Благодаря такому разделению можно убедиться, что модель обладает высокой точностью на новых данных и не переобучена на имеющейся выборке. Важно помнить, что **разделение выборки на тренировочную и тестовую должно быть выполнено до начала обработки данных и обучения модели**. Также необходимо убедиться, что выборки не содержат дублирующихся или повторяющихся данных, и что разделение происходит случайным образом или с учетом определенного критерия, чтобы избежать смещения в оценке качества модели. Существует несколько способов разделения выборки на тренировочную, валидационную и тестовую в машинном обучении. Некоторые факторы, которые могут учитываться при выборе метода разбиения, включают в себя:

- **Размер выборки:** для небольших выборок может быть лучше использовать кросс-валидацию (перекрестную проверку), чтобы обеспечить более точную оценку модели.
- **Целевой признак:** если целевой признак несбалансированный, необходимо убедиться, что он представлен в обеих выборках.
- **Распределение признаков:** если признаки имеют разное распределение в тренировочной и тестовой выборках, это может привести к переобучению или недообучению модели.
- **Временные ряды:** при анализе временных рядов необходимо учитывать хронологический порядок данных и использовать методы, которые могут учитывать этот фактор.
- **Доступность данных:** иногда может быть сложно получить достаточное количество данных для построения отдельных тренировочной и тестовой выборок. В этом случае можно использовать методы, такие как кросс-валидация , чтобы максимизировать использование имеющихся данных.

В целом, выбор метода разбиения выборки зависит от конкретной задачи и доступных данных, и может потребовать экспериментов с различными методами, чтобы найти наилучший вариант.

Ниже перечислены наиболее распространенные из них:

- Разбиение на **обучающую, валидационную и тестовую** выборки в пропорции 60-20-20 (или любой другой соответствующий выбор) (функция `train_test_split` библиотеки `sklearn`)
- Использование **кросс-валидации**, при которой данные разбиваются на несколько равных частей (фолдов), и каждая часть используется в качестве тестовой выборки, а остальные части объединяются и используются для обучения модели (функции `KFold`, `StratifiedKFold`, `TimeSeriesSplit` и т.д. библиотеки `sklearn`)
- Однократное **разбиение на обучающую и тестовую выборки, а затем использование внутренней кросс-валидации** на обучающей выборке для подбора параметров модели и оценки ее качества на валидационной выборке.
- **Разбиение на обучающую, валидационную и тестовую выборки с помощью временных рядов**, когда данные разбиваются на обучающую и тестовую выборки в хронологическом порядке, а затем выборка для валидации формируется из части обучающей выборки, расположенной в более раннем временном периоде, чем тестовая выборка.

Кроме того, для некоторых задач может быть полезно использовать стратифицированное разбиение, когда сохраняется пропорция классов в каждой выборке, или разбиение с учетом баланса классов, когда классы распределяются между выборками с учетом их дисбаланса.

0.2.4 Виды признаков

В машинном обучении существуют различные виды признаков, которые могут использоваться для описания объектов или данных. Некоторые из них:

- **Бинарные признаки:** признаки, которые могут принимать только два значения, например, 0 и 1, да или нет, и т.д.
- **Категориальные признаки:** признаки, которые принимают значения из определенного набора категорий или классов. Примерами категориальных признаков могут служить цвет, тип материала и т.д.

- **Порядковые (ранговые) признаки** - это признаки, которые принимают значения из упорядоченного набора категорий или классов. Порядок между значениями имеет смысл и отражает отношение или ранг между ними. Например, ранговые признаки могут включать уровень образования (начальное, среднее, высшее), оценку (плохо, удовлетворительно, хорошо, отлично) или степень удовлетворенности (очень низкая, низкая, средняя, высокая, очень высокая).
- **Числовые признаки:** признаки, которые принимают числовые значения, такие как длина, ширина, высота, возраст и т.д.
- **Текстовые признаки:** признаки, которые описывают текстовые данные, такие как заголовки новостей, описания продуктов и т.д.
- **Географические признаки:** признаки, которые описывают географические данные, такие как координаты, адреса и т.д.
- **Временные признаки:** признаки, которые описывают данные, относящиеся ко времени — дата, время, длительность и т.д.

Работа с разными видами признаков в машинном обучении может иметь свои особенности, так как каждый вид признаков имеет свои уникальные свойства и может требовать специфических методов работы.

- **Категориальные признаки:** Категориальные признаки нуждаются в преобразовании перед использованием в модели. Для этого могут использоваться различные методы, например, One-Hot Encoding или Label Encoding (определение этих методов будет рассмотрено ниже). Некоторые алгоритмы машинного обучения, такие как деревья решений и случайный лес, могут обрабатывать категориальные признаки напрямую.
- **Числовые признаки:** Числовые признаки могут быть использованы без какой-либо специальной обработки, но нормализация и стандартизация могут улучшить результаты. При работе с числовыми признаками также может быть полезно создать новые признаки, например, путем извлечения корня или возвведения в квадрат.
- **Бинарные признаки:** Бинарные признаки могут быть использованы без специальной обработки, но могут потребовать обработки выбросов, так как значения могут быть ограничены только двумя значениями.
- **Текстовые признаки:** Текстовые признаки должны быть преобразованы в числовой формат для использования в модели. Для этого могут использоваться различные методы, такие как CountVectorizer или TF-IDF. CountVectorizer - это метод векторизации текста в машинном обучении, который используется для преобразования текстовых данных в векторы числовых значений, которые могут быть обработаны алгоритмами машинного обучения. CountVectorizer преобразует текстовые данные в векторы, где каждый элемент вектора соответствует количеству вхождений определенного слова в документе. Таким образом, каждый документ представлен в виде вектора, где размерность вектора равна общему числу уникальных слов во всех документах, а каждый элемент вектора соответствует количеству вхождений соответствующего слова в документ. TF-IDF (англ. Term Frequency-Inverse Document Frequency) - это метод векторизации текста, который используется для вычисления важности слова в документе на основе его частоты встречаемости в этом документе и общего количества документов, в которых это слово встречается. TF-IDF учитывает, что некоторые слова могут быть часто встречающимися во многих документах, и поэтому не могут быть использованы для отличия одного документа от другого. С другой стороны, редкие слова, которые встречаются только в некоторых документах, могут предоставить более ценную информацию для их

классификации. TF-IDF вычисляется путем умножения «частоты слова в документе» (Term Frequency - TF) на «обратную частоту документа» (Inverse Document Frequency - IDF).

- **Географические признаки:** Географические признаки могут быть преобразованы в числовый формат, например, через координаты или почтовый индекс. При работе с географическими признаками также могут быть полезны новые признаки, такие как расстояние между объектами или количество объектов в радиусе действия.
- **Временные признаки:** Временные признаки могут быть использованы без специальной обработки, но также могут быть полезны новые признаки, например, день недели или время суток.

0.2.5 Предобработка признаков

Предобработка признаков - это процесс подготовки данных перед применением модели машинного обучения. Цель предобработки признаков заключается в том, чтобы привести данные к такому виду, который будет оптимальным для обучения модели и получения максимальной точности предсказаний. Предобработка признаков является важным этапом в машинном обучении, так как позволяет улучшить точность модели, снизить шум и сделать данные более понятными для алгоритма. Некоторые этапы предобработки признаков в машинном обучении включают:

- Обработка выбросов, заполнение пропущенных значений и т.д.
- Преобразование признаков - преобразование данных в числовый формат, нормализация и масштабирование признаков, преобразование категориальных признаков и т.д.

Обработка выбросов

Обработка выбросов в машинном обучении - это процесс идентификации и обработки экстремальных значений (наблюдений), которые могут исказить результаты анализа и привести к ошибочным выводам. Выбросы могут возникать из-за ошибок в измерениях, аномалий в данных или других факторов. Существует несколько способов обработки выбросов:

- **Замена выбросов:** подход заключается в замене выбросов на какое-то более типичное значение. Это может быть медиана, среднее значение или любое другое значение, которое соответствует типичным значениям признаков в выборке данных.
- **Удаление выбросов:** важно помнить, что удаление выбросов может привести к потере информации, поэтому решение о том, следует ли удалять выбросы, должно быть основано на конкретной задаче и анализе данных. Существует несколько способов удаления выбросов в машинном обучении, включая:
 - Метод межквартильного размаха (англ. Interquartile Range, IQR): Этот метод основан на вычислении межквартильного размаха данных, который определяет расстояние между 25-м и 75-м процентилем данных. Для вычисления 75-го процентиля (Q_3) можно использовать следующий алгоритм:
 1. Упорядочьте набор чисел по возрастанию.
 2. Рассчитайте позицию, на которой должно находиться значение 75-го процентиля. Для этого умножьте 75 на количество чисел в наборе и разделите результат на 100. Если результат не является целым числом, округлите его до ближайшего целого в большую сторону. Обозначим полученное значение как позицию.

3. Если позиция является целым числом, то 75-й процентиль — это значение, находящееся на этой позиции в упорядоченном наборе чисел. Если позиция имеет десятичную часть, то 75-й процентиль — это среднее арифметическое двух значений: значение на позиции целой части и значение на следующей позиции.

Пример: У нас есть следующий упорядоченный набор чисел: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

1. Количество чисел в наборе равно 10.
2. Позиция = $(75 * 10) / 100 = 7.5$ (округляем вверх до 8).
3. 75-й процентиль — это значение на позиции 8, то есть 80.

IQR равен: $IQR = Q_3 - Q_1$. Затем выбросы определяются как значения, находящиеся за пределами верхнего и нижнего порогов, определяемых как $Q_1 - 1.5 * IQR$ и $Q_3 + 1.5 * IQR$ соответственно.

- Удаление выбросов на основе статистических критериев: Этот метод использует статистические критерии, такие как Z-оценка или Т-тест, для определения, является ли значение выбросом или нет. Если значение превышает определенный пороговый уровень, оно считается выбросом и удаляется.

- **Использование робастных (устойчивых к выбросам) моделей:** Робастные модели машинного обучения устойчивы к выбросам и могут работать более точно, даже если в выборке данных есть выбросы. Например, линейные модели, которые используют L1-регуляризацию (Lasso) или L2-регуляризацию (Ridge), могут быть более устойчивыми к выбросам, чем стандартные линейные модели. Некоторые алгоритмы машинного обучения, такие как метод случайных деревьев и градиентный бустинг, могут учитывать выбросы, используя различные стратегии сэмплирования и ансамблирования.
- **Проведение дополнительного исследования:** Если выбросы вызваны реальными аномалиями, то они могут содержать важную информацию о данных. Поэтому может быть полезно провести дополнительное исследование, чтобы понять, почему выбросы возникли и как их можно использовать в моделировании.

Заполнение пропущенных значений

Некоторые значения могут быть пропущены или недоступны, например, из-за ошибок или недоступности источника данных. Пропущенные значения можно заменить средним, медианным или модальным значением, или использовать другие методы заполнения, например, на основе предсказаний модели.

Преобразование данных в числовой формат

Некоторые признаки могут иметь неправильный тип данных, например, строковые данные вместо числовых. Необходимо преобразовать типы данных в правильный формат. Для иллюстрации рассмотрим пример на основе признака «возраст» (Таблица 1):

Нормализация и масштабирование признаков

Нормализация и масштабирование признаков - это процесс приведения значений признаков к определенному диапазону или масштабу. Этот процесс является важной частью предобработки данных в машинном обучении и может улучшить качество модели. Многие алгоритмы машинного обучения требуют, чтобы все признаки имели одинаковый масштаб,

Возраст	Возраст_-num
«20»	20
«30»	30
«40»	40
«50»	50

Таблица 1: Пример преобразования данных в числовой формат

так как некоторые признаки могут иметь значительно большую вариативность, чем другие. Например, если у нас есть два признака: возраст в годах (от 0 до 100) и ежемесячный доход (от 0 до 1 000 000 рублей), то ежемесячный доход будет иметь значительно большую вариативность. Это может привести к тому, что алгоритмы машинного обучения будут отдавать большее значение признаку с большей вариативностью. Существуют несколько основных методов масштабирования признаков:

- **Нормализация (MinMax нормализация):** Этот метод преобразует значения признаков в диапазон от 0 до 1 или от -1 до 1. Это можно сделать с помощью формулы 1:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}, \quad (1)$$

где x - это исходное значение, x_{min} и x_{max} - минимальное и максимальное значения в наборе данных соответственно, а x_{norm} - нормализованное значение.

Для иллюстрации рассмотрим пример на основе числового признака «возраст» (Таблица 2):

Возраст	MinMax нормализация
20	0.0000
30	0.3333
40	0.6667
50	1.0000

Таблица 2: Пример MinMax нормализации

Для каждого значения признака мы вычисляем новое значение, используя формулу, приведенную выше. В данном примере, минимальное значение возраста равно 20, а максимальное значение равно 50. При применении формулы, мы получаем новые значения признака, которые находятся в диапазоне от 0 до 1. Например, возраст 30 будет масштабирован до 0.3333, а возраст 50 будет масштабирован до 1.0000.

MinMax нормализация может быть полезна в случаях, когда значения признаков имеют разный масштаб и диапазон значений. Однако, необходимо быть осторожным при применении MinMax нормализации, если выборка содержит выбросы, так как они могут сильно искажать результаты масштабирования. Кроме того, необходимо обрабатывать

случаи, когда в тестовой выборке встречаются значения признака, которых не было в обучающей выборке.

- **Стандартизация (L2 масштабирование, Z-оценка):** Этот метод преобразует значения признаков в распределение со средним значением 0 и стандартным отклонением 1. Это можно сделать с помощью формулы 2:

$$x_{std} = \frac{x - \mu}{\sigma}, \quad (2)$$

где x - значение признака, μ - среднее значение признака в выборке,

σ - стандартное отклонение признака в выборке.

Для иллюстрации рассмотрим пример на основе числового признака «возраст» (Таблица 3):

Возраст	Стандартизация
20	-1.3416
30	-0.4472
40	0.4472
50	1.3416

Таблица 3: Пример стандартизации

Для каждого значения признака мы вычисляем новое значение, используя формулу, приведенную выше.

В данном примере, среднее значение возраста равно 35, а стандартное отклонение равно 11.18. При применении формулы, мы получаем новые значения признака, которые находятся в стандартном нормальном распределении со средним значением равным 0 и стандартным отклонением равным 1. Например, возраст 30 будет масштабирован до -0.4472, а возраст 50 будет масштабирован до 1.3416. Стандартизация может быть полезна в случаях, когда значения признаков имеют разный масштаб и диапазон значений, но при этом не содержат выбросы. Стандартизация также может улучшить качество модели, которая использует линейные методы машинного обучения, так как они часто основаны на предположении о стандартном нормальном распределении данных.

Выбор метода масштабирования признаков зависит от конкретной задачи и данных. Важно также отметить, что масштабирование признаков следует выполнять только на тренировочных данных, а затем применять те же параметры масштабирования на тестовых данных и новых данных в производственной среде.

Преобразование категориальных признаков

Категориальные признаки (например, цвет, размер, марка автомобиля) являются одним из типов данных, используемых в машинном обучении. Однако, многие алгоритмы машинного обучения работают только с числовыми данными. Поэтому необходимо преобразовать категориальные признаки в числовые. Этот процесс называется кодированием категориальных признаков. Этот метод включает в себя преобразование категориальных признаков в числовые. Существует несколько методов кодирования категориальных признаков:

- **One-Hot Encoding:** Этот метод преобразует каждую категориальную переменную в набор бинарных переменных. Для каждого уникального значения переменной создается отдельная бинарная переменная. Если переменная принимает значение, равное значению категории, то соответствующая бинарная переменная будет равна 1, в противном случае - 0. Приведем пример One-Hot Encoding на основе категориального признака «фрукт» с тремя уникальными значениями: яблоко, банан и апельсин (Таблица 4).

Фрукт	Яблоко	Банан	Апельсин
Яблоко	1	0	0
Банан	0	1	0
Апельсин	0	0	1

Таблица 4: Пример One-Hot Encoding

Каждый уникальный фрукт превращается в отдельный столбец, который может быть 1 или 0 в зависимости от того, какой фрукт присутствует в конкретном наблюдении. Если в наблюдении присутствуют несколько фруктов, соответствующие столбцы получат значение 1. В приведенном выше примере первое и последнее наблюдение содержат яблоко, поэтому столбцы «Яблоко» для этих наблюдений имеют значение 1. Второе и четвертое наблюдение содержат банан, поэтому соответствующий столбец «Банан» имеет значение 1. Третье и шестое наблюдение содержат апельсин, поэтому соответствующий столбец «Апельсин» имеет значение 1.

- **Label Encoding:** Этот метод присваивает каждому уникальному значению категориальной переменной уникальный целочисленный код. Приведем пример Label Encoding на основе категориального признака «фрукт» с тремя уникальными значениями: яблоко, банан и апельсин (Таблица 5). Каждая уникальная категория получает уникальный

Фрукт	Код
Яблоко	1
Банан	2
Апельсин	3
Банан	2
Яблоко	1
Апельсин	3

Таблица 5: Пример Label Encoding

числовой код. В данном примере яблоко имеет код 1, банан имеет код 2, а апельсин имеет код 3. При использовании Label Encoding важно знать, что коды не являются порядковыми, то есть код 2 для банана не означает, что банан в два раза больше яблока. Коды используются только для идентификации категорий.

Label Encoding может быть полезен в случаях, когда категории обладают некоторой внутренней упорядоченностью (порядковые переменные), например, признак «образование», где можно использовать порядковые числа для идентификации уровня образования. Однако, если категории не обладают внутренней упорядоченностью, например, цвета, то использование Label Encoding может быть менее эффективным.

- **Mean Encoding (также Target Encoding):** Этот метод преобразования категориальных признаков в числовые значения, которые представляют среднее значение целевой переменной для каждой категории. Допустим, у нас есть набор данных с категориальным признаком «цвет» и целевой переменной «цена». Мы хотим закодировать категориальный признак «цвет» с помощью метода Mean Encoding.

1. Посчитаем среднее значение целевой переменной для каждой категории «цвета» (Таблица 6):

Цвет	Цена
Красный	10
Зеленый	20
Синий	30
Красный	15
Зеленый	25

Таблица 6: Начальные значения для каждой категории

Средние значения для каждой категории (Таблица 7):

Цвет	Цена
Красный	12.5
Зеленый	22.5
Синий	30.0

Таблица 7: Средние значения для каждой категории

2. Заменим значения категориального признака «цвет» на его средние значения (Таблица 8):

Цвет	Цена
Красный	12.5
Зеленый	22.5
Синий	30.0
Красный	12.5
Зеленый	22.5

Таблица 8: Закодированные значения для каждой категории

Таким образом, мы закодировали категориальный признак «цвет» в числовые значения с помощью метода Mean Encoding, используя информацию о целевой переменной «цена». Это может помочь улучшить точность модели машинного обучения при работе с категориальными признаками.

- **Frequency Encoding:** Этот метод присваивает каждому уникальному значению категориальной переменной частоту его появления в данных. Вот пример Frequency Encoding на основе категориального признака «фрукт» с тремя уникальными значениями: яблоко, банан и апельсин (Таблица 9):

Фрукт	Frequency Encoding
Яблоко	0.3333
Банан	0.3333
Апельсин	0.3333
Банан	0.3333
Яблоко	0.3333
Апельсин	0.3333

Таблица 9: Пример Label Encoding

Каждое уникальное значение категориального признака заменяется на частоту его появления в данных. В данном примере все три значения фруктов встречаются по два раза, поэтому их частота равна $2/6 = 0.3333$. В строках, где фрукт повторяется, значения Frequency Encoding остаются одинаковыми.

Frequency Encoding может быть полезен в случаях, когда категории не имеют внутренней упорядоченности, а частота появления категории в данных может быть связана с целевой переменной. Однако, при использовании Frequency Encoding необходимо быть осторожным, чтобы избежать переобучения модели. Кроме того, необходимо обработать случаи, когда в тестовой выборке встречаются значения категориального признака, которых не было в обучающей выборке.

Эти методы могут быть применены как отдельно, так и в сочетании друг с другом, в зависимости от исходных данных. Преобразование признаков является важным шагом в процессе подготовки данных для модели машинного обучения и может существенно улучшить качество модели.

0.2.6 Создание новых признаков

Создание новых признаков (англ. Feature Engineering) в машинном обучении - это процесс преобразования исходных данных в новые признаки, которые могут улучшить качество модели и улучшить ее способность к предсказанию. Ниже перечислены некоторые методы создания новых признаков в машинном обучении:

- **Инженерия признаков на основе знаний об отрасли:** в этом подходе эксперты отрасли и/или аналитики создают новые признаки на основе своих знаний о предметной области. Это может включать в себя создание признаков на основе социально-экономических показателей, метеорологических данных, исторических событий и т.д. Например, в медицинских данных может быть создан признак «Индекс массы тела» (BMI), используя формулу на основе роста и веса пациента.
- **Преобразование существующих признаков:** в этом подходе существующие признаки могут быть преобразованы в новые признаки путем использования математических операций, например, логарифмирование, возведение в степень, извлечение корня и т.д. Можно создавать новые признаки, например, путем извлечения значений из других признаков. Например, можно извлечь день недели или время из даты и времени.
- **Создание признаков на основе текстов:** при анализе текстовых данных новые признаки могут быть созданы путем извлечения ключевых слов и фраз, использования

стемминга, лемматизации, CountVectorizer, TF-IDF, BPE, а также при помощи методов машинного обучения для анализа тональности и определения тем.

- **Создание признаков на основе временных рядов:** при анализе временных рядов новые признаки могут быть созданы путем вычисления статистических показателей, таких как скользящее среднее, стандартное отклонение, корреляция и т.д.

0.2.7 Отбор признаков

Отбор признаков (англ. Feature Selection) в машинном обучении - это процесс выбора наиболее значимых признаков из множества доступных, которые будут использоваться для обучения модели. Цель отбора признаков состоит в том, чтобы уменьшить размерность данных, устранить шум, повысить точность модели и улучшить ее интерпретируемость. Существует несколько методов отбора признаков:

- **Рекурсивное устраниние признаков (англ. Recursive Feature Elimination):** используется алгоритм, который удаляет признаки с наименьшей важностью для модели, пока не останется необходимое число признаков.
- **Подход на основе важности признаков:** определяется важность признаков для модели с помощью алгоритмов, таких как случайный лес или градиентный бустинг. На основе этого отбираются наиболее важные признаки.
- **Подход на основе корреляции признаков:** удаляются признаки, которые сильно коррелируют друг с другом, так как они могут увеличивать сложность модели без улучшения ее качества.

Важно понимать, что отбор признаков - это процесс, который может повлиять на точность модели. Поэтому необходимо тщательно оценить каждый метод и выбрать наиболее подходящий в каждой конкретной ситуации.

0.2.8 Реализация в Python

Реализация основных этапов подготовки данных для проекта машинного обучения рассмотрена в jupyter notebook файле: lecture_02_code_labs_01_pandas.ipynb.

0.2.9 Вопросы для самопроверки

Исследование данных в процессе разработки проекта машинного обучения может включать в себя:

1. визуализацию данных
2. заполнение пропущенных значений
3. создание новых признаков
4. изучение связей между различными переменными

Правильные ответы: 1, 4

Выберите, какие из приведенных ниже признаков являются категориальными:

1. Пол: мужской/женский
2. Цвет: красный/зеленый/синий/желтый

3. Марка автомобиля: BMW/Mercedes/Audi
4. Образование: высшее/среднее/начальное
5. Страна происхождения: Россия/США/Китай
6. Продукты питания: молоко/сыр/мясо/овощи/фрукты
7. Уровень дохода: низкий/средний/высокий
8. Домашние животные: кошка/собака/хомяк

Правильные ответы: 2, 3, 5, 6, 8

Пояснение: 1 вариант - бинарный признак, 4 и 7 варианты обладают внутренней упорядоченностью (порядковые признаки).

Какой из приведенных ниже методов преобразует каждую категориальную переменную в набор бинарных переменных:

1. Frequency Encoding
2. Label Encoding
3. One-Hot Encoding
4. Mean Encoding

Правильные ответы: 3

0.2.10 Резюме по разделу

На текущий момент мы рассмотрели основные этапы подготовки данных для проекта машинного обучения, а именно:

Исследование данных: различные методы визуализации и статистические техники

Очистка данных: очистка данных от ошибок и пропусков

Разделение выборки: Разбиение выборки на обучающую и тестовую является одним из важных этапов машинного обучения. Важно помнить, что разделение выборки на тренировочную и тестовую должно быть выполнено до начала обработки данных и обучения модели.

Работа с признаками: работа с выбросами и преобразование признаков

Создание новых признаков: процесс выбора наиболее значимых признаков из множества доступных, которые будут использоваться для обучения модели

Отбор признаков: процесс выбора наиболее значимых признаков из множества доступных, которые будут использоваться для обучения модели.

0.3 Оценка качества алгоритмов обучения с учителем

Цель занятия: ученик может самостоятельно подобрать подходящую слушаю метрику качества.

План занятия:

1. Метрики качества регрессии
2. Метрики качества классификации
3. Обобщающая способность алгоритмов

Прежде чем мы начнем работать с алгоритмами машинного обучения, давайте поймем, как можно оценивать качество работы алгоритмов машинного обучения. Это позволит в дальнейшем оценивать качество всех алгоритмов, с которыми мы будем иметь дело. Для этой цели используются специальные формулы, называемые метриками качества.

Метрики качества в машинном обучении нужны для оценки качества работы алгоритмов. Они позволяют сравнивать разные модели машинного обучения и выбирать наилучшую для конкретной задачи. Кроме того, метрики качества позволяют определить, насколько хорошо обученная модель справляется с поставленной перед ней задачей и насколько ее результаты достоверны.

Например, метрики качества для задачи классификации позволяют определить, насколько точно модель предсказывает метки классов для тестовых данных. А метрики качества для задачи регрессии позволяют определить, насколько близко предсказанные значения к истинным.

Метрики качества могут быть использованы для выбора оптимального набора гиперпараметров¹ модели, для настройки ее параметров, а также для сравнения разных алгоритмов машинного обучения и выбора наилучшего из них.

0.3.1 Метрики качества регрессии

Метрики качества регрессии - это числовые характеристики, которые используются для оценки того, насколько хорошо модель регрессии соответствует данным. Ниже приведены некоторые из наиболее распространенных метрик качества регрессии с примерами:

- **Среднеквадратичная ошибка (англ. Mean Square Error, MSE)** - это среднее значение квадратов разностей между прогнозируемыми значениями и фактическими значениями.

Формула для средней квадратичной ошибки (MSE) выглядит следующим образом (Формула 3):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2, \quad (3)$$

где y_i - это наблюдаемое значение, а \hat{y}_i - это предсказанное значение.

MSE особенно часто используется, когда существует линейная зависимость между признаками и целевой переменной. Это происходит, когда мы имеем дело с непрерывными переменными, такими как цена на недвижимость или доход.

¹Гиперпараметры модели - это параметры, которые определяются еще до обучения модели. Они определяют поведение алгоритма обучения, такие как скорость обучения, количество скрытых слоев в нейронной сети, количество деревьев в случайном лесу и т.д.

Пример: Если у нас есть модель, которая предсказывает количество продаж в магазине, MSE может быть рассчитана как среднее значение квадратов разностей между прогнозируемыми продажами и фактическими продажами.

- **Средняя абсолютная ошибка (англ. Mean Absolute Error, MAE)** - это среднее значение абсолютных разностей между прогнозируемыми значениями и фактическими значениями. Формула для средней абсолютной ошибки (MAE) выглядит следующим образом (Формула 4):

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (4)$$

где y_i - это наблюдаемое значение, а \hat{y}_i - это предсказанное значение.

MAE чаще используется, когда мы имеем дело с выбросами и выбивающимися значениями. Она более устойчива к выбросам, чем MSE.

Пример: Предположим, что у нас есть модель, которая прогнозирует цены на недвижимость. MAE для этой модели может быть рассчитана как среднее значение абсолютных разностей между прогнозируемыми ценами и реальными ценами.

- **Функция потерь Хьюбера (англ. Huber loss)** - это функция потерь, которая комбинирует свойства среднеквадратичной ошибки (MSE) и средней абсолютной ошибки (MAE) в зависимости от значения отклонения. Она используется для задач регрессии и оптимизации моделей машинного обучения. Формула для функции потерь Хьюбера выглядит следующим образом (Формула 5):

$$L_\delta(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{если } |y_i - \hat{y}_i| \leq \delta \\ \delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta) & \text{иначе} \end{cases} \quad (5)$$

где y_i - это наблюдаемое значение, а \hat{y}_i - это предсказанное значение,

δ - параметр для определения порога между квадратичной и линейной потерями.

Функционал потерь Хьюбера вычисляется как среднее значение потерь для всех примеров. Функционал потерь Хьюбера более устойчив к выбросам, чем MSE, потому что он уменьшает вклад выбросов в общую ошибку. При этом, он сохраняет свойства MSE, когда отклонение мало, и свойства MAE, когда отклонение большое. Параметр δ можно выбрать в зависимости от того, насколько сильно мы хотим уменьшить влияние выбросов на общую ошибку.

Вот как их можно интерпретировать MAE, MSE и Huber loss:

- Средняя абсолютная ошибка (MAE):
 - MAE представляет собой среднюю абсолютную разницу между предсказанными значениями и истинными значениями.
 - Измеряет среднюю величину ошибки в исходных единицах измерения целевой переменной.
 - Например, если MAE равно 5, это означает, что средняя абсолютная разница между предсказанными и истинными значениями составляет 5 единиц.
- Средняя квадратичная ошибка (MSE):
 - MSE представляет собой среднюю квадратичную разницу между предсказанными значениями и истинными значениями.

- Измеряет среднеквадратичное отклонение ошибок в квадрате исходных единиц измерения целевой переменной.
- Поскольку MSE возводит ошибки в квадрат, оно более чувствительно к большим выбросам. Например, если MSE равно 25, это означает, что среднеквадратичное отклонение между предсказанными и истинными значениями составляет 25 единиц².
- Huber loss:
 - Huber loss является комбинацией MAE и MSE, предназначеннной для уменьшения влияния выбросов на оценку модели.
 - Применяет MAE для малых ошибок и MSE для больших ошибок, что делает его менее чувствительным к выбросам. Huber loss вычисляет разницу между предсказанными и истинными значениями и применяет пороговое значение (обычно называемое delta) для определения переключения между MAE и MSE.
 - Результат Huber loss интерпретируется так же, как и MAE.

В общем, меньшие значения MAE, MSE и Huber loss указывают на лучшую производительность модели регрессии. Однако, для интерпретации результатов важно учитывать контекст задачи и единицы измерения целевой переменной.

Это только некоторые из наиболее распространенных метрик качества регрессии. В зависимости от задачи и данных, могут быть использованы и другие метрики.

Отображение метрик качества регрессии

В основе MSE, MAE и функции потерь Хьюбера лежит вычислительный процесс, который учитывает отклонения наблюдений от аппроксимирующей гиперплоскости (Рисунок 1).

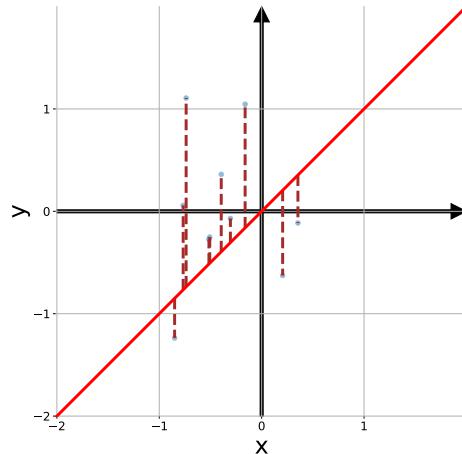


Рисунок 1: Визуализация вычисления метрик качества регрессии

Отображение графика отклонений от аппроксимирующей линии для MSE и MAE позволяет оценить точность модели и выявить наличие систематических ошибок в предсказаниях. Если на графике присутствует какая-то структура или зависимость между отклонениями и значением переменных, это может указывать на недостатки модели и необходимость ее улучшения. Если же график показывает случайное распределение отклонений вокруг нуля, то это свидетельствует о том, что модель хорошо справляется с предсказаниями.

0.3.2 Метрики качества классификации

Метрики качества классификации - это числовые характеристики, которые используются для оценки того, насколько хорошо модель классификации соответствует данным.

Концепция TP, TN, FP, FN в метриках качества классификации

Для работы с метриками качества классификации необходимо понимать такие понятия, как **истинно-положительное решение** (англ. True Positive, TP), **истинно-отрицательное решение** (англ. True Negative, TN), **ложно-положительное решение** (англ. False Positive, FP) и **ложно-отрицательное решение** (англ. False Negative, FN).

TP, TN, FP, FN - это часто используемые сокращения для обозначения результатов двоичной классификации, где у нас есть два класса: положительный (P) и отрицательный (N).

TP - это количество правильно классифицированных положительных примеров (True Positive). Например, если модель должна классифицировать изображения на котов и собак, и она правильно классифицирует изображение кота как кота, это будет являться TP.

TN - это количество правильно классифицированных отрицательных примеров (True Negative). Например, если модель должна классифицировать изображения на котов и собак, и она правильно классифицирует изображение автомобиля как не являющееся ни котом, ни собакой, это будет являться TN.

FP - это количество неправильно классифицированных положительных примеров (False Positive). Например, если модель должна классифицировать изображения на котов и собак, и она неправильно классифицирует изображение собаки как кота, это будет являться FP.

FN - это количество неправильно классифицированных отрицательных примеров (False Negative). Например, если модель должна классифицировать изображения на котов и собак, и она неправильно классифицирует изображение кота как не являющееся ни котом, ни собакой, это будет являться FN.

Рассмотрим наглядный пример. Допустим, у нас есть модель для определения, является ли письмо не спамом (хорошее письмо) или спамом.

- True Positive (TP) - это случай, когда письмо является хорошим и наша модель правильно предсказывает его как хорошее.
- True Negative (TN) - это случай, когда письмо является спамом и наша модель правильно предсказывает его как спам.
- False Positive (FP) - это случай, когда письмо является спамом, но наша модель ошибочно предсказывает его как хорошее.
- False Negative (FN) - это случай, когда письмо является хорошим, но наша модель ошибочно предсказывает его как спам.

Эти примеры можно обобщить с помощью матрицы ошибок (Таблица 10):

Предсказанное значение	Наблюдаемое значение	
	Хорошее	Спам
Хорошее	TP	FP
	FN	TN

Таблица 10: Матрица ошибок (Confusion matrix)

Давайте рассмотрим несколько примеров:

- Пример 1: у нас есть письмо, которое является хорошим, и наша модель правильно предсказывает его как хорошее. В этом случае, это будет True Positive (TP).
- Пример 2: у нас есть письмо, которое является спамом, но наша модель ошибочно предсказывает его как хорошее. В этом случае, это будет False Positive (FP) (ошибка I рода, ложное срабатывание).
- Пример 3: у нас есть письмо, которое является спамом, и наша модель правильно предсказывает его как спам. В этом случае, это будет True Negative (TN).
- Пример 4: у нас есть письмо, которое является хорошим, но наша модель ошибочно предсказывает его как спам. В этом случае, это будет False Negative (FN) (ошибка II рода).

В машинном обучении и статистике используется несколько различных метрик для измерения точности моделей. Рассмотрим основные из них.

Accuracy для бинарной классификации

Accuracy (доля правильных ответов) - отношение числа правильно классифицированных объектов к общему числу объектов в выборке в задаче бинарной классификации. Accurасу - это самая простая метрика, которая вычисляет долю верно классифицированных объектов в общем числе объектов. Она часто используется для оценки качества работы алгоритмов классификации сбалансированных классов. Формула для accurасу (Формула 6):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},$$

где TP - количество верно предсказанных положительных классов,
 TN - количество верно предсказанных отрицательных классов,
 FP - количество ложно предсказанных положительных классов,
 FN - количество ложно предсказанных отрицательных классов.

Рассмотрим два примера задачи бинарной классификации для выявления спама. Первый пример - описательный, второй - с визуализацией. Пусть у нас есть набор данных, состоящий из 1000 писем, из которых 950 - хорошие письма, а 50 - письма-спам. В этом случае, классы несбалансированы, потому что спама гораздо меньше, чем хороших писем. Если мы используем accurасу как метрику для оценки качества модели, то в этом случае accurасу может давать искаженные результаты. Например, допустим, что модель предсказывает, что все письма - хорошие, тогда accurасу составит 0.95, что может показаться высокой точностью модели. Однако, такая модель совершенно не пригодна для решения поставленной задачи, так как ее цель - выявление спама и она не сможет правильно классифицировать такие объекты.

Рассмотрим второй пример, где писем поровну - по 10. Визуализация вычисления метрики accurасу для такого примера представлена на рисунке 2.

Precision для бинарной классификации

Precision (точность) - отношение числа верно предсказанных положительных классов к общему числу положительных предсказаний в задаче бинарной классификации. Метрика точности (precision) используются для оценки качества моделей машинного обучения в

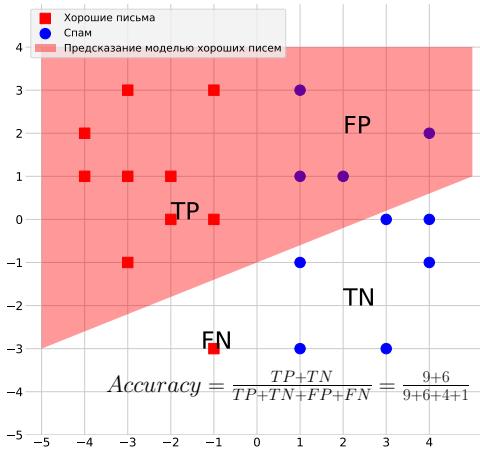


Рисунок 2: Визуализация вычисления метрики accuracy для бинарной классификации

задачах классификации. Они позволяют измерять, насколько хорошо модель выделяет объекты определенного класса из общего числа объектов, которые она относит к этому классу. Формула precision (Формула 7):

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (7)$$

где TP - количество верно предсказанных положительных классов,
 FP - количество ложно предсказанных положительных классов.

Продолжая пример с задачей выявления спама, рассмотрим метрику точности (precision) для этой задачи. В данном случае, точность определяет, какой процент из всех писем, которые модель предсказала как спам, действительно являются спамом. Допустим, модель классифицирует 70 писем как спам, и из них на самом деле только 50 писем являются мошенническими. Тогда точность будет равна $50/70 = 0.71$, что означает, что модель правильно классифицировала 71% писем, которые она отнесла к классу спам. Метрика точности позволяет оценить качество работы модели с точки зрения ее способности правильно идентифицировать объекты положительного класса, т.е. спам в нашем примере. Однако, метрика точности может не учитывать количество ложно-отрицательных прогнозов, т.е. случаев, когда модель не определила письма как мошеннические, но на самом деле они являются мошенническими. Для полной оценки качества работы модели необходимо рассматривать и другие метрики, такие как recall (полнота), которая показывает, какой процент из всех мошеннических писем модель способна правильно идентифицировать, и F1-score, который является гармоническим средним между precision и recall и учитывает как точность, так и полноту модели.

Визуализация вычисления метрики precision для набора из 20 писем представлена на рисунке 3.

Recall (полнота) для бинарной классификации

Recall (полнота) - отношение числа верно предсказанных положительных классов к общему числу реальных положительных классов. Метрика полноты (recall) также используется

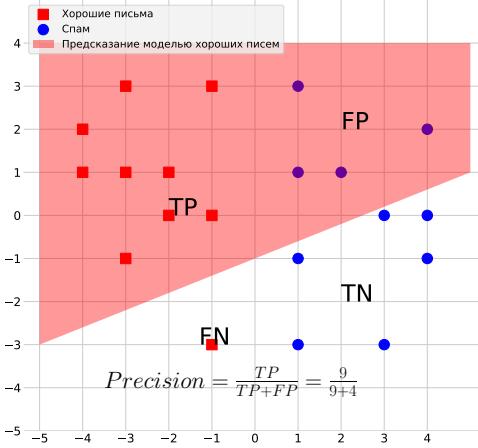


Рисунок 3: Визуализация вычисления метрики precision для бинарной классификации

для оценки качества моделей машинного обучения в задачах классификации. Она позволяет измерять, насколько хорошо модель распознает объекты определенного класса из общего числа объектов, которые принадлежат этому классу (Формула 8):

$$Recall = \frac{TP}{TP + FN}, \quad (8)$$

где TP - количество верно предсказанных положительных классов,

FN - количество ложно предсказанных положительных классов.

Продолжая пример с задачей выявления спама, рассмотрим метрику полноты (recall) для этой задачи. В данном случае, полнота определяет, какой процент из всех мошеннических писем модель способна правильно идентифицировать. Допустим, что в общем числе 100 писем модель смогла правильно идентифицировать только 50. Тогда полнота будет равна $50/100 = 0.5$, что означает, что модель правильно распознала только 50% всех мошеннических писем. Метрика полноты позволяет оценить качество работы модели с точки зрения ее способности правильно идентифицировать все объекты положительного класса, т.е. все письма-спам в нашем примере. Однако, метрика полноты может не учитывать количество ложно-положительных прогнозов, т.е. случаев, когда модель неправильно идентифицирует операции как мошеннические, когда на самом деле они ими не являются.

Визуализация вычисления метрики recall для примера с 20 письмами представлена на рисунке 4.

F1-мера для бинарной классификации

F1 score - это мера точности и полноты модели, которая вычисляется как гармоническое среднее точности (precision) и полноты (recall) модели. Формула для расчета F1 score в

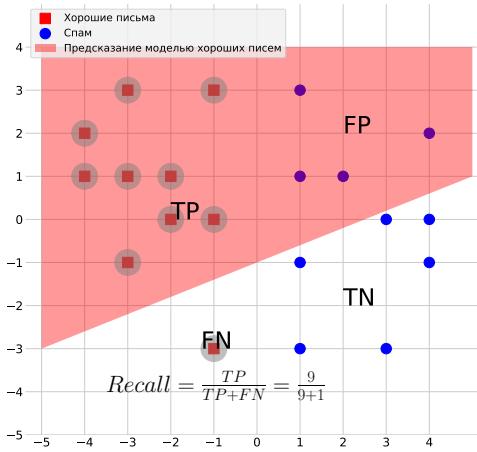


Рисунок 4: Визуализация вычисления метрики recall для бинарной классификации

задаче бинарной классификации (Формула 9):

$$F1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)},$$

где TP - количество верно предсказанных экземпляров положительного класса, (9)

FP - количество ложно предсказанных экземпляров положительного класса,

FN - количество ложно предсказанных экземпляров отрицательного класса.

В бинарной классификации 1000 писем для каждого письма алгоритм определяет, является ли оно спамом или нет. Давайте зададим другое распределение спама и хороших писем, 500 являются спамом, а 500 - нет. Допустим, алгоритм классификации верно определил 450 спам-писем и 480 хороших писем. $TP = 450$, $TN = 480$, $FP = 20$, $FN = 50$. Тогда: $Accuracy = (TP + TN) / (TP + TN + FP + FN) = (450 + 480) / (450 + 480 + 20 + 50) = 0.93$ $Precision = TP / (TP + FP) = 450 / (450 + 20) = 0.96$ $Recall = TP / (TP + FN) = 450 / (450 + 50) = 0.9$ $F1\text{-score} = 2 * Precision * Recall / (Precision + Recall) = 2 * 0.96 * 0.9 / (0.96 + 0.9) = 0.93$

Алгоритм показывает высокую точность и полноту, так как значение accuracy, precision, recall и F1-score достаточно высокие (больше 0.9). Это говорит о том, что алгоритм хорошо находит и отличает мошеннические письма от обычных, и его можно использовать для дальнейшей работы с данными. Однако, возможно, стоит обратить внимание на то, что FN (ложноотрицательные результаты) выше, чем FP (ложноположительные результаты), что может быть проблемой, если ложноотрицательные результаты критичны для данной задачи. В целом, для задачи бинарной классификации определения спама результаты работы алгоритма можно считать хорошими.

Визуализация вычисления метрики F1 для примера с 20 письмами представлена на рисунке 5.

Многоклассовая классификация

В целом, принцип выбора метрик в задаче многоклассовой классификации совпадает с задачей бинарной классификации. Однако, в задаче многоклассовой классификации есть

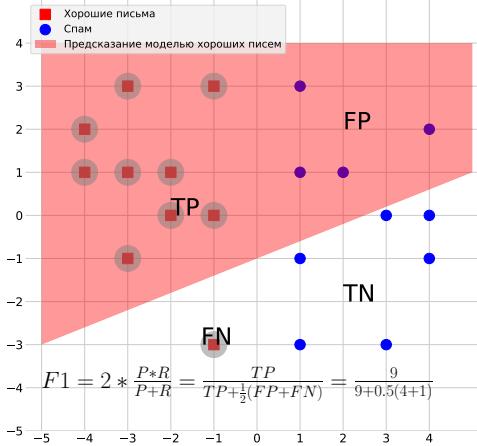


Рисунок 5: Визуализация вычисления метрики F1 для бинарной классификации

несколько модификаций метрик precision, recall и F1: micro, macro и weighted. Главным отличием между micro, macro и weighted модификациями является то, как они учитывают распределение классов. Модификация micro учитывает все истинно-положительные и ложно-положительные примеры, модификация macro не учитывает несбалансированность классов в наборе данных, модификация weighted учитывает этот фактор.

Стоит отметить, что если классы сбалансиированы, в задаче многоклассовой классификации различные модификации метрик (micro, macro и weighted усреднение) будут показывать практически идентичный результат.

Для объяснения вычисления метрик многоклассовой классификации возьмем следующий пример (6).

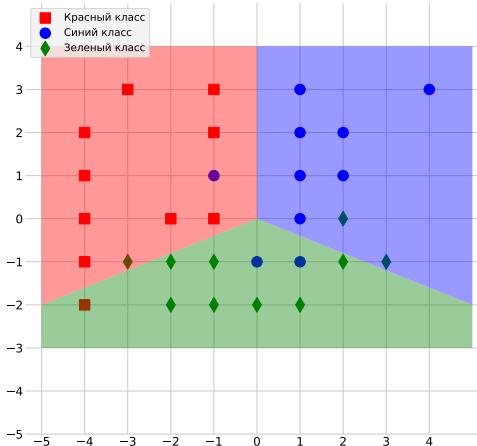


Рисунок 6: Матрица ошибок

Будем считать ответы, двигаясь по сетке слева-направо сверху-вниз Получим такие наборы данных (1 - красный квадрат, 2 - синий круг, 3 - зеленый ромб):

$$\text{Истинные значения} = [1, 1, 1, 1, 1, 1, 3, 1, 3, 3, 1, 1, 2, 1, 3, 3, 2, 3, 2, 2, 2, 2, 2, 3, 2, 2, 3, 3, 3, 2]$$

$$\text{Предсказанные значения} = [1, 1, 1, 1, 1, 3, 1, 1, 1, 3, 3, 1, 1, 1, 3, 3, 3, 2, 2, 2, 2, 2, 3, 3, 2, 2, 3, 2]$$

Создадим матрицу ошибок для вычисления метрик многоклассовой классификации (7).

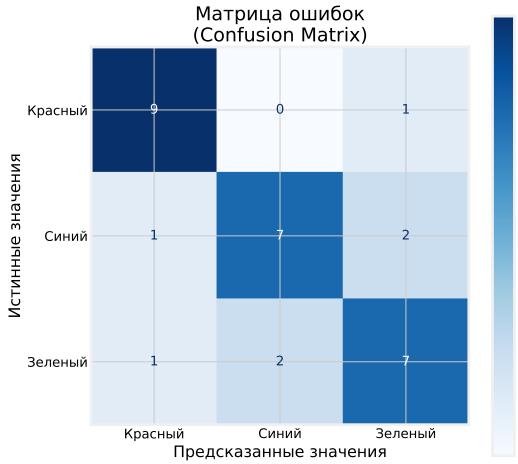


Рисунок 7: Матрица ошибок

Accuracу для многоклассовой классификации

Формулу для accuracу (Формула 6) можно использовать и для многоклассовой классификации. Мы просто считаем количество правильно и неправильно классифицированных объектов. По матрице ошибок можно получить значение accuracу (сумма значений по диагоналям / сумма значений по матрице):

$$\text{accuracy} = \frac{9 + 7 + 7}{9 + 7 + 7 + 0 + 1 + 2 + 1 + 1 + 2} = 0.77$$

Стоит заметить, что accuracу плохо работает на несбалансированных выборках.

Precision для многоклассовой классификации

Для одного класса c в задаче многоклассовой классификации **precision** вычисляется по формуле 10:

$$\text{precision}_c = \frac{TP_c}{TP_c + FP_c}, \quad (10)$$

где TP_c - количество верно предсказанных положительных примеров класса c ,

FP_c - количество ложно предсказанных положительных примеров класса c .

Micro precision - это метрика, которая вычисляет точность для каждого класса по отдельности, а затем усредняет их с использованием общего числа истинных и ложных положительных и отрицательных ответов по всем классам. Эта метрика подходит для задач, где классы не сбалансированы и модель должна давать одинаковый вес каждому объекту. Формула для вычисления micro precision (Формула 11):

$$precision_{micro} = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}, \quad (11)$$

где TP_c - количество верно предсказанных положительных примеров класса c ,

FP_c - количество ложно предсказанных положительных примеров класса c .

В нашем примере вычислить micro precision можно с помощью матрицы ошибок:

$$precision_{micro} = \frac{9 + 7 + 7}{(9 + 0 + 1) + (7 + 1 + 2) + (7 + 1 + 2)} = 23/30 = 0.766$$

Macro precision - это метрика, которая вычисляет точность для каждого класса по отдельности и затем усредняет их. Эта метрика не учитывает размеры классов, поэтому она подходит для задач, где классы сбалансированы. Формула для вычисления macro precision (Формула 12):

$$precision_{macro} = \frac{1}{C} \sum_c P_c, \quad (12)$$

где P_c - точность по классу c , C - количество классов в задаче классификации.

В нашем примере вычислить macro precision можно с помощью матрицы ошибок:

$$precision_{macro} = \frac{\frac{9}{9+1+1} + \frac{7}{7+0+2} + \frac{7}{7+1+2}}{3} = 0.765$$

Weighted precision - это метрика, которая вычисляет точность для каждого класса по отдельности и затем усредняет их с использованием весов, пропорциональных размеру каждого класса. Эта метрика подходит для задач, где классы не сбалансированы, и модель должна учитывать важность каждого класса. Формула для вычисления weighted precision (Формула 13):

$$precision_{weighted} = \frac{\sum_c (TP_c + FP_c) \cdot P_c}{\sum_c (TP_c + FP_c)}, \quad (13)$$

где P_c - точность по классу c ,

TP_c - количество верно предсказанных положительных примеров класса c ,

FP_c - количество ложно предсказанных положительных примеров класса c .

В нашем примере вычислить weighted precision можно с помощью матрицы ошибок:

$$precision_{weighted} = \frac{(9 + 1 + 1) * \frac{9}{9+1+1} + (7 + 0 + 2) * \frac{7}{7+0+2} + (7 + 1 + 2) * \frac{7}{7+1+2}}{(9 + 1 + 1) + (7 + 0 + 2) + (7 + 1 + 2)} = 0.765$$

Recall (полнота) для многоклассовой классификации

Для одного класса c в задаче многоклассовой классификации **Recall** вычисляется по формуле 14:

$$\text{recall}_c = \frac{TP_c}{TP_c + FN_c}, \quad (14)$$

где TP_c - количество верно предсказанных положительных примеров класса c ,

FN_c - количество ложно предсказанных положительных примеров класса c .

Micro recall - это метрика, которая вычисляет полноту для каждого класса по отдельности, а затем усредняет их с использованием общего числа истинных и ложных положительных и отрицательных ответов по всем классам. Эта метрика подходит для задач, где классы не сбалансированы и модель должна давать одинаковый вес каждому объекту. Формула для вычисления micro recall (Формула 15):

$$\text{recall}_{\text{micro}} = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}, \quad (15)$$

где TP_c - количество верно предсказанных положительных примеров класса c ,

FN_c - количество ложно предсказанных отрицательных примеров класса c .

В нашем примере вычислить micro recall можно с помощью матрицы ошибок:

$$\text{recall}_{\text{micro}} = \frac{9 + 7 + 7}{(9 + 0 + 1) + (7 + 1 + 2) + (7 + 1 + 2)} = 23/30 = 0.766$$

Macro recall - это метрика, которая вычисляет полноту для каждого класса по отдельности и затем усредняет их. Эта метрика не учитывает размеры классов, поэтому она подходит для задач, где классы сбалансированы. Формула для вычисления macro recall (Формула 16):

$$\text{recall}_{\text{macro}} = \frac{1}{C} \sum_c R_c, \quad (16)$$

где R_c - полнота по классу c , C - количество классов в задаче классификации.

В нашем примере вычислить macro recall можно с помощью матрицы ошибок:

$$\text{recall}_{\text{macro}} = \frac{\frac{9}{9+0+1} + \frac{7}{7+1+2} + \frac{7}{7+1+2}}{3} = 23/30 = 0.766$$

Weighted recall - это метрика, которая вычисляет полноту для каждого класса по отдельности и затем усредняет их с использованием весов, пропорциональных размеру каждого класса. Эта метрика подходит для задач, где классы не сбалансированы, и модель должна учитывать важность каждого класса. Формула для вычисления weighted recall (Формула 17):

$$\text{recall}_{\text{weighted}} = \frac{\sum_c (TP_c + FN_c) \cdot R_c}{\sum_c (TP_c + FN_c)},$$

где R_c - полнота по классу c , (17)

TP_c - количество верно предсказанных положительных примеров класса c ,

FN_c - количество ложно предсказанных отрицательных примеров класса c .

В нашем примере вычислить weighted recall можно с помощью матрицы ошибок:

$$\text{recall}_{\text{weighted}} = \frac{(9 + 0 + 1) \cdot \frac{9}{9+0+1} + (7 + 1 + 2) \cdot \frac{7}{7+1+2} + (7 + 1 + 2) \cdot \frac{7}{7+1+2}}{(9 + 0 + 1) + (7 + 1 + 2) + (7 + 1 + 2)} = 23/30 = 0.766$$

F1-мера для многоклассовой классификации

Для одного класса c в задаче многоклассовой классификации F1-мера вычисляется по формуле 18:

$$F1_c = \frac{2 \cdot TP_c}{2 \cdot TP_c + FP_c + FN_c},$$

где TP_c - количество верно предсказанных положительных примеров класса c , (18)

FP_c - количество ложно предсказанных положительных примеров класса c ,

FN_c - количество ложно предсказанных отрицательных примеров класса c .

Здесь F1-мера является гармоническим средним между точностью и полнотой для одного класса.

Micro F1 - это метрика, которая вычисляет F1-меру для каждого класса по отдельности, а затем усредняет их с использованием общего числа истинных и ложных положительных и отрицательных ответов по всем классам. Эта метрика подходит для задач, где классы не сбалансированы и модель должна давать одинаковый вес каждому объекту. Формула для вычисления micro F1 (Формула 19):

$$F1_{micro} = 2 \cdot \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c) + \sum_c (TP_c + FN_c)} \quad (19)$$

В нашем примере вычислить micro F1 можно с помощью матрицы ошибок:

$$F1_{micro} = 2 \cdot \frac{(9 + 7 + 7)}{11 + 9 + 10 + 10 + 10} = 23/30 = 0.766$$

Macro F1 - это метрика, которая вычисляет F1-меру для каждого класса по отдельности и затем усредняет их. Эта метрика не учитывает размеры классов, поэтому она подходит для задач, где классы сбалансированы. Формула для вычисления macro F1 (Формула 20):

$$F1_{macro} = \frac{1}{C} * \sum_c F1_c \quad (20)$$

где C - количество классов, $F1_c$ - F1-мера для класса c

В нашем примере вычислить macro F1 можно с помощью матрицы ошибок:

$$F1_{macro} = \frac{1}{3} * \left(\frac{2 \cdot 9}{2 \cdot 9 + 2 + 1} + \frac{2 \cdot 7}{2 \cdot 7 + 2 + 3} + \frac{2 \cdot 7}{2 \cdot 7 + 3 + 3} \right) = 0.764$$

Weighted F1 - это метрика, которая вычисляет F1-меру для каждого класса по отдельности и затем усредняет их с использованием весов, пропорциональных размеру каждого класса. Эта метрика подходит для задач, где классы не сбалансированы, и модель должна учитывать важность каждого класса. Формула для вычисления weighted F1 (Формула 21):

$$F1_{weighted} = \frac{\sum_c w_c \cdot F1_c}{N} \quad (21)$$

где N - количество примеров во всей выборке,

w_c - доля примеров класса c в выборке.

В нашем примере вычислить weighted F1 можно с помощью матрицы ошибок:

$$F1_{weighted} = \frac{1}{3} * \left(\frac{2 \cdot 9}{2 \cdot 9 + 2 + 1} + \frac{2 \cdot 7}{2 \cdot 7 + 2 + 3} + \frac{2 \cdot 7}{2 \cdot 7 + 3 + 3} \right) = 0.764$$

0.3.3 Обобщающая способность алгоритмов и переобучение

Метрики качества - это не единственное, что нужно учитывать при оценке работы модели машинного обучения. Еще одной важной характеристикой модели является обобщающая способность. Обобщающая способность алгоритма машинного обучения - это способность алгоритма правильно классифицировать новые, ранее неизвестные данные, которые не использовались при обучении. Она является ключевым показателем качества алгоритма машинного обучения.

Переобучение (англ. Overfitting) - это явление, когда алгоритм машинного обучения получает очень высокую точность на данных, которые использовались для обучения, но плохо работает на новых данных. При переобучении алгоритм машинного обучения настраивается на шум в данных и пытается «запомнить» обучающие данные, а не выявлять общие закономерности. При переобучении график метрик на тренировочном наборе данных будет показывать улучшение в процессе обучения, но на тестовом наборе данных график метрик может начать падать после достижения пика или перестать улучшаться, что будет указывать на то, что модель не обобщает знания на новые данные (Рисунок 8).

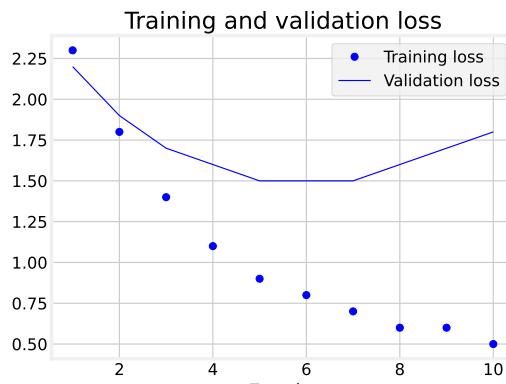


Рисунок 8: Пример ситуации переобучения

Переобучение может возникнуть, если модель слишком сложная и содержит слишком много параметров, которые могут быть настроены на шум в данных. Также переобучение может возникнуть, если обучающая выборка слишком мала или не достаточно разнообразна.

Для предотвращения переобучения существует несколько методов, основным из которых является регуляризация. Мы поговорим о ней в следующем модуле.

0.3.4 Вопросы для самопроверки

Выберите метрики качества регрессии из приведенных ниже вариантов:

1. accuracy
2. MAE
3. Huber loss
4. MSE

Правильные ответы: 2, 3, 4

Метрика precision - это отношение:

1. $\frac{TP}{TP+FP}$
2. $\frac{TP}{TP+FN}$
3. $\frac{TP+TN}{TP+TN+FP+FN}$

Правильные ответы: 1

Какую модификацию метрики F1 для многоклассовой классификации нужно выбрать, чтобы учесть размеры (баланс) классов:

1. Micro
2. Macro
3. Weighted

Правильные ответы: 3

0.3.5 Резюме по разделу

Существует несколько метрик качества, которые используются для оценки качества алгоритмов машинного обучения.

Распространенные метрики качества для задач регрессии:

- Средняя квадратичная ошибка (MSE)
- Средняя абсолютная ошибка (MAE)
- Функция потерь Хьюбера

Распространенные метрики качества для задач классификации:

- Точность (Accuracy)
- Точность положительного класса (Precision)
- Полнота (Recall)
- F1-мера (F1-score)

Выбор метрик зависит от конкретной задачи и ее целей.

Обобщающая способность алгоритмов машинного обучения означает их способность обобщать знания из тренировочных данных и применять их для эффективного решения новых задач на новых данных. То есть, хорошо обученная модель должна показывать высокое качество предсказаний не только на тренировочных данных, но и на тестовых и реальных данных.

Однако, при обучении модели можно столкнуться с проблемой переобучения, когда модель слишком точно подстроилась под тренировочные данные, и, соответственно, ее способность к обобщению становится низкой. В результате, такая модель показывает плохие результаты на тестовых данных и не может быть использована для решения новых задач.

Для того, чтобы избежать переобучения, необходимо использовать специальные техники, такие как регуляризация.

0.4 Метод ближайших соседей

Цель занятия: ученик может применить алгоритм KNN для решения задач регрессии и классификации на подготовленных и неподготовленных данных.

План занятия:

- Визуальная демонстрация алгоритма
- Описание алгоритма
- Подготовка данных для алгоритма
- Оценка качества алгоритма
- Модификации алгоритма
- Область применения алгоритма
- Плюсы и минусы алгоритма
- Реализация алгоритма в Python

0.4.1 Визуальная демонстрация алгоритма

Визуальная демонстрация алгоритма представлена на последовательности из 4 рисунков (Рисунок 9).

0.4.2 Описание алгоритма

Метод ближайших соседей (англ. KNN, k-Nearest Neighbors) - это алгоритм машинного обучения для классификации и регрессии. Он относится к методам обучения с учителем, что означает, что для обучения необходимы данные с известными метками классов или целевыми значениями.

Пошаговое описание алгоритма KNN:

1. Загрузка данных. Загрузите обучающий набор данных, содержащий признаки и соответствующие метки классов (для задачи классификации) или значения целевой переменной (для задачи регрессии). Если признаки имеют разные шкалы или единицы измерения, то целесообразно выполнить нормализацию данных. Например, можно использовать стандартизацию, где каждый признак будет иметь среднее значение 0 и стандартное отклонение 1. Определите количество ближайших соседей K , которые будут использованы для классификации или регрессии. Значение K должно быть положительным целым числом.
2. Вычисление расстояний. Для каждого экземпляра тестовых данных вычислите расстояние до каждого экземпляра обучающего набора. Расстояние может быть вычислено с использованием различных метрик, таких как евклидово расстояние или манхэттенское расстояние.
3. Сортировка и выбор K ближайших соседей. Отсортируйте расстояния в порядке возрастания.
4. Выбор K соседей. Выберите K ближайших соседей для каждого экземпляра тестовых данных. В случае классификации можно определить метку класса путем выбора наиболее часто встречающейся метки среди k ближайших соседей. В случае регрессии можно предсказать значение целевой переменной путем вычисления среднего или медианы значений целевой переменной среди k ближайших соседей.

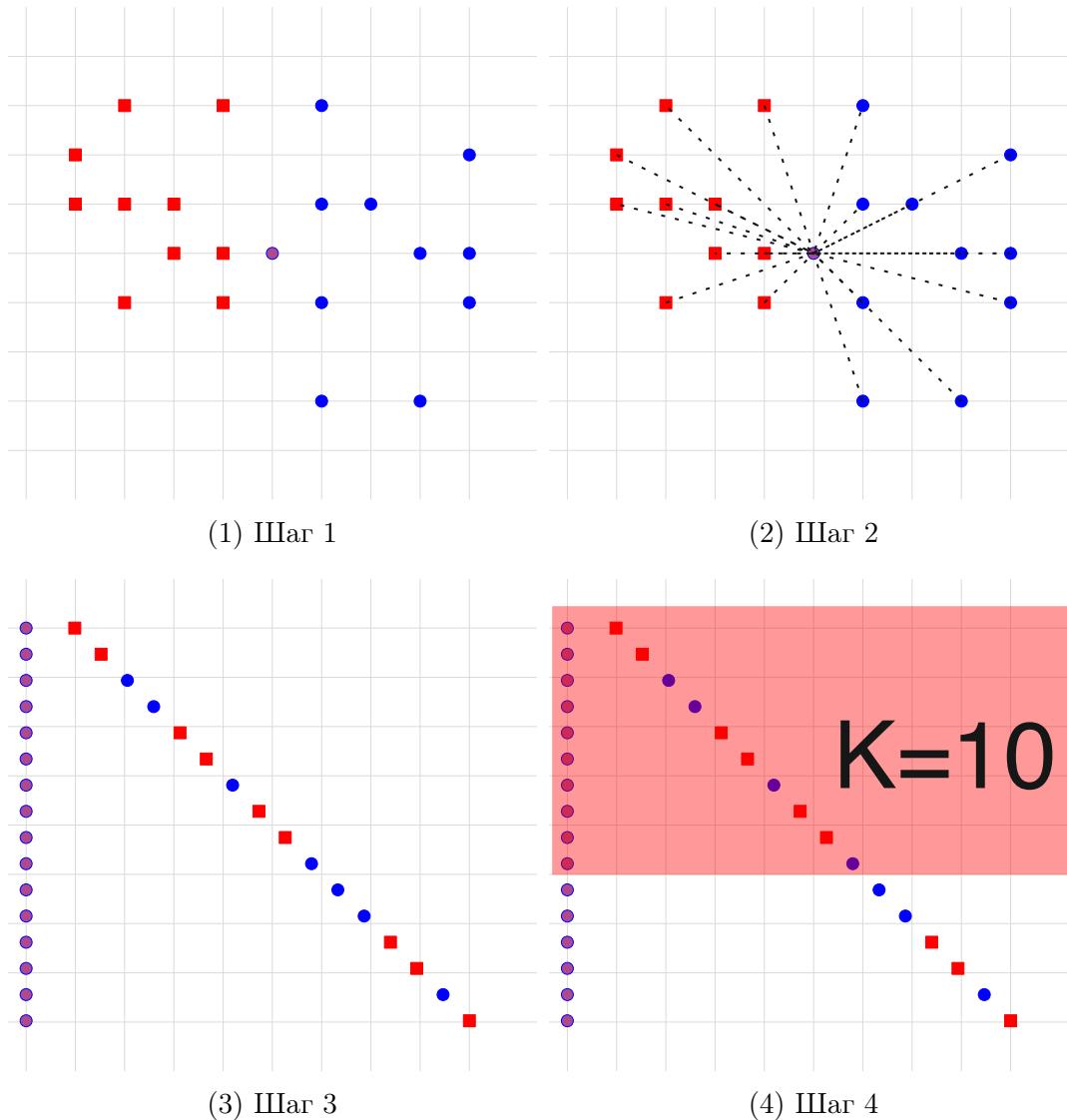


Рисунок 9: Визуальная демонстрация алгоритма KNN

Алгоритм k-ближайших соседей (KNN) - это простой алгоритм машинного обучения, который не имеет много параметров обучения. Главный параметр - **число соседей (K)**, которое необходимо выбрать для каждого объекта, чтобы выполнить классификацию или регрессию.

Выбор числа соседей K является важным шагом в использовании алгоритма KNN. Если выбрать слишком маленькое значение K, то модель будет слишком чувствительной к выбросам и шуму, что может привести к переобучению. Если выбрать слишком большое значение K, то модель может потерять способность к выделению малозначимых признаков и потерять способность к различению классов.

Другие параметры, которые могут быть учтены при использовании алгоритма KNN, включают в себя: **меру расстояния** (например, евклидово расстояние или манхэттенское расстояние), **веса для каждого соседа** (например, обратная пропорциональность расстояния до соседа).

Важно отметить, что параметры меры расстояния и веса соседей могут сильно влиять на качество модели, и их выбор должен быть основан на особенностях конкретной задачи. Как правило, оптимальные значения этих параметров подбираются методом кросс-валидации на обучающем наборе данных.

0.4.3 Подготовка данных для алгоритма

Перед применением алгоритма KNN необходимо выполнить следующие шаги подготовки данных:

- **Очистка данных:** удаление неполных или некорректных записей, заполнение пропущенных значений и преобразование данных в формат, подходящий для анализа.
- **Нормализация данных:** приведение данных к общему масштабу, чтобы каждый признак имел одинаковый вклад в анализ.
- **Разбиение данных на обучающую и тестовую выборки:** обучающая выборка используется для обучения модели, а тестовая выборка - для оценки ее точности.
- **Выбор метрики расстояния:** выбор подходящей метрики расстояния, которая будет использоваться для определения ближайших соседей.
- **Определение параметра k:** выбор оптимального значения параметра k (количество ближайших соседей), который дает наилучшие результаты для конкретной задачи, с помощью алгоритмов перебора.

0.4.4 Оценка качества алгоритма

Для оценки точности алгоритма KNN в задачах классификации и регрессии часто используют следующие метрики:

Для задач классификации:

- **Accuracy (точность):** доля правильно классифицированных объектов.
- **Precision (точность):** доля объектов, классифицированных как положительные и действительно являющихся положительными.
- **Recall (полнота):** доля положительных объектов, правильно классифицированных алгоритмом.
- **F1-мера:** среднее гармоническое между precision и recall.

Для задач регрессии:

- **Mean Absolute Error (MAE):** средняя абсолютная ошибка между прогнозами и истинными значениями.
- **Mean Squared Error (MSE):** средняя квадратичная ошибка между прогнозами и истинными значениями.

0.4.5 Модификации алгоритма

Существует несколько модификаций алгоритма k-ближайших соседей (KNN). Рассмотрим не так часто используемые на практике, и приведены здесь для ознакомления.

- **Взвешенный KNN (англ. Weighted KNN):** в этой модификации алгоритма каждый сосед получает вес, зависящий от расстояния до целевого объекта. Чем ближе сосед, тем

больший вес он получает. Это позволяет учесть вклад каждого соседа в классификацию или регрессию, и улучшает точность модели.

- **Алгоритм KNN с переменным числом соседей (англ. Variable KNN):** в этой модификации число соседей для каждого объекта может быть разным. Например, для объектов, находящихся в области с большой плотностью точек, можно выбирать большее число соседей, а для объектов в области с малой плотностью - меньшее число соседей. Это позволяет лучше адаптировать модель к особенностям данных.
- **Алгоритм KNN с оптимизацией расстояния (англ. Distance-optimized KNN):** в этой модификации используется не единственная мера расстояния, а оптимизированная комбинация нескольких мер расстояния. Это может улучшить точность классификации или регрессии в задачах с нелинейными зависимостями между признаками.
- **Многоуровневый KNN (англ. Multi-level KNN):** в этой модификации используется несколько моделей KNN на разных уровнях анализа данных. Например, для текстовых данных можно использовать первый уровень KNN для выбора близких по смыслу документов, а затем второй уровень KNN для классификации на основе дополнительных признаков.
- **Алгоритм KNN на основе графов (англ. Graph-based KNN):** в этой модификации алгоритма объекты представляются в виде вершин графа, а соседство между объектами определяется на основе связей в графе. Это позволяет учитывать не только расстояние между объектами, но и их близость в структуре данных.

Каждая из этих модификаций KNN может быть полезной в определенных задачах машинного обучения, и выбор модификации должен быть основан на особенностях данных и конкретной задаче.

0.4.6 Область применения алгоритма

Область применения алгоритма KNN включает в себя многие задачи машинного обучения, такие как:

- Обработка табличных данных.
- Классификация текстов: KNN может использоваться для классификации текстовых документов по их содержанию. Например, можно классифицировать новости на категории на основе их содержания.
- Обработка изображений: KNN может использоваться для классификации изображений на основе их содержания. Например, можно классифицировать изображения по типу объекта на фотографии (человек, автомобиль, животное и т.д.).
- Медицинская диагностика: KNN может использоваться для диагностики заболеваний на основе симптомов и медицинских показателей. Например, можно классифицировать пациентов на основе их медицинской истории.
- Финансовый анализ: KNN может использоваться для анализа финансовых данных и прогнозирования цен на акции, валюту и другие финансовые инструменты.

Кроме того, алгоритм KNN может использоваться во многих других областях, где требуется классификация или регрессия на основе признаков объектов.

0.4.7 Плюсы и минусы алгоритма

Алгоритм k-Nearest Neighbors (KNN) имеет ряд преимуществ и недостатков. **Плюсы:**

- Простота реализации и понимания. Это один из самых простых алгоритмов машинного обучения.
- Может использоваться как для задач классификации, так и для задач регрессии.
- Гибкость в выборе метрики расстояния и числа соседей.

Минусы:

- Низкая эффективность на больших выборках и большое время выполнения.
- Чувствительность к выбросам и шуму в данных.
- Требует хранения всего набора данных в памяти, что может быть проблематично для больших наборов данных.
- Трудности с выбором оптимального числа соседей, которые влияют на качество предсказаний.

В целом, алгоритм KNN имеет простую и интуитивно понятную логику, но его эффективность может быть низкой в случае больших выборок и шумных данных.

0.4.8 Реализация алгоритма в Python

В библиотеке Scikit-learn в модуле neighbors реализован класс **KNeighborsClassifier** и **KNeighborsRegressor** для классификации и регрессии соответственно.

Класс KNeighborsClassifier имеет следующие параметры:

- **n_neighbors**: количество соседей, используемых для классификации объекта. По умолчанию равно 5.
- **weights**: весовая функция, используемая для определения влияния каждого соседа. По умолчанию все соседи имеют равный вес.
- **algorithm**: алгоритм, используемый для вычисления ближайших соседей. Возможные значения: «auto», «ball_tree», «kd_tree», «brute». По умолчанию используется «auto», который выбирает наиболее подходящий алгоритм на основе обучающих данных.
- **leaf_size**: размер листа дерева, используемый при использовании алгоритмов ball_tree или kd_tree. По умолчанию равен 30.
- **p**: параметр, используемый для расчета расстояния между объектами. По умолчанию равен 2 (Евклидово расстояние).
- **metric**: метрика расстояния, используемая для измерения расстояния между объектами. По умолчанию используется «minkowski».
-

Класс KNeighborsRegressor имеет те же параметры, что и KNeighborsClassifier, за исключением параметра weights, который по умолчанию установлен в «uniform».

Оба класса имеют методы **fit(X, y)** для обучения модели на данных X и y, а также методы **predict(X)** и **predict_proba(X)** (для KNeighborsClassifier) для предсказания меток и вероятностей соответственно. Кроме того, для KNeighborsRegressor есть методы **score(X, y)** и **kneighbors(X)**, которые возвращают коэффициент детерминации (R-квадрат) и индексы ближайших соседей для каждого объекта в X соответственно.

0.4.9 Вопросы для самопроверки

Как происходит обучение алгоритма KNN:

1. алгоритм просто запоминает обучающую выборку, чтобы затем измерить расстояние от нового объекта до объектов в обучающей выборке
2. с помощью градиентного спуска
3. с помощью метода Ньютона

Правильные ответы: 1

Как вычисляется значение для нового объекта в задаче регрессии с помощью KNN:

1. Объекту присваивается значение наиболее близкого объекта из обучающей выборки
2. Объекту присваивается среднее значение по k наиболее близким объектам из обучающей выборки
3. Объекту присваивается среднее значение по 3 наиболее близким объектам из обучающей выборки

Правильные ответы: 2

Выберите плюсы алгоритма KNN:

1. Гибкость в выборе метрики расстояния и числа соседей
2. Низкая эффективность на больших выборках и большое время выполнения
3. Чувствительность к выбросам и шуму в данных
4. Простота реализации и понимания
5. Требует хранения всего набора данных в памяти, что может быть проблематично для больших наборов данных
6. Трудности с выбором оптимального числа соседей, которые влияют на качество предсказаний

Правильные ответы: 1, 4

0.4.10 Резюме по разделу

Алгоритм k-Nearest Neighbors (KNN) - это простой алгоритм машинного обучения, который используется для классификации и регрессии. Алгоритм определяет класс нового объекта или его значение, основываясь на k ближайших объектах из тренировочного набора данных.

Параметр k - это количество ближайших объектов, которые используются для определения класса нового объекта. Чем больше k , тем более гладким будет граница принятия решений, но при этом увеличивается шанс на ошибку из-за увеличения влияния объектов других классов.

Основными преимуществами KNN являются простота реализации и хорошее качество классификации для некоторых типов данных. Недостатками алгоритма являются высокая вычислительная сложность и требование к наличию большого объема данных для получения высокой точности классификации или регрессии.

Также существуют модификации алгоритма, такие как взвешенный KNN, KNN с использованием ядра и метрики расстояния, а также с использованием алгоритмов уменьшения размерности, которые могут помочь улучшить качество классификации и регрессии с помощью KNN.

0.5 Резюме по модулю

В этом модуле мы рассмотрели такие темы, как процесс разработки проекта машинного обучения, подготовка данных для задач машинного обучения, оценка качества алгоритмов обучения с учителем, а также познакомились с одним из самых простых и интуитивно понятных алгоритмов в машинном обучении - методом ближайших соседей.