1. Объясните, как вычисляется качество модели с задачей классификации: назовите основные и вторичные метрики, приведите пример.

Основные метрики:

Точность (Accuracy): Доля правильно классифицированных объектов относительно всего числа объектов.

Точность (Precision): Доля истинно положительных объектов среди объектов, которые модель считает положительными.

Полнота (Recall) или чувствительность (Sensitivity): Доля истинно положительных объектов среди всех действительно положительных объектов.

F1-мера: Гармоническое среднее между точностью и полнотой.

Вторичные метрики:

Специфичность (Specificity): Доля истинно отрицательных объектов среди всех действительно отрицательных объектов

Matthews Correlation Coefficient (MCC): Коэффициент корреляции Мэтьюса, который является более сбалансированной мерой качества для несбалансированных классов.

Log-Loss: Логарифмическая потеря, которая используется для оценки вероятностных классификаторов.

Пример:

Допустим, у нас есть 100 объектов для бинарной классификации, и результаты классификации следующие:

True Positives (TP): 40

True Negatives (TN): 30

False Positives (FP): 10

False Negatives (FN): 20

Тогда:

Точность (Accuracy): (40 + 30)/100=0.7

Точность (Precision): 40/(40+10)=0.8

Полнота (Recall): 40/(40+20)=0.667

F1-мера: 2∗(0.8∗0.667)/(0.8+0.667)=0.727

Специфичность: 30/(30+10)=0.75

1. Что такое тестовый набор данных (test set) и для чего он нужен?

Тестовый набор данных (test set) — это подмножество данных, которое используется для оценки производительности модели машинного обучения после того, как она была обучена на другом подмножестве данных, известном как обучающий набор данных (training set). Для чего нужен тестовый набор данных: оценка обобщающей способности модели, избежание переобучения,

1. Каково назначение валидационного множества данных (validation set)?

Валидационное множество данных (validation set) — это подмножество данных, которое отделяется от исходного датасета и не используется в процессе обучения модели. Его основное назначение — проведение промежуточной оценки модели во время процесса обучения и настройки гиперпараметров. В отличие от тестового множества, которое используется для окончательной оценки модели, валидационное множество позволяет провести "пробное тестирование" без риска "заглядывания" в тестовые данные.

Важно, чтобы валидационный набор был репрезентативным и не пересекался с обучающим и тестовым наборами. Обычно исходные данные разбиваются на обучающий, валидационный и тестовый наборы в определенных пропорциях, например, 70%/15%/15% или 60%/20%/20%.

1. В чем разница между параметром модели и гиперпараметром алгоритма обучения? Приведите примеры.

Параметры модели

Параметры модели — это переменные, которые модель обучает из данных. Они определяют преобразования, которые модель применяет к входным данным для получения предсказанных выходных данных. Эти параметры оптимизируются в процессе обучения с использованием алгоритмов, таких как стохастический градиентный спуск, чтобы минимизировать функцию потерь.

Примеры:

Веса и смещения в нейронных сетях.

Коэффициенты в линейной регрессии.

Пороги разделения и выбор признаков в деревьях решений.

Гиперпараметры алгоритма обучения

Гиперпараметры — это "внешние" конфигурации для модели, которые не изучаются из данных, но устанавливаются заранее до начала процесса обучения. Они остаются неизменными во время обучения и влияют на скорость и качество процесса обучения. Часто их настраивают с использованием методов, таких как кросс-валидация.

Примеры:

Скорость обучения в алгоритмах, основанных на градиентном спуске.

Количество соседей в методе k-ближайших соседей.

Количество деревьев в случайном лесу.

Регуляризационные коэффициенты в линейной регрессии или логистической регрессии.

Размер батча в стохастическом градиентном спуске.

Разница

Основное различие между параметрами модели и гиперпараметрами заключается в том, что параметры модели обучаются прямо из тренировочных данных, тогда как гиперпараметры задаются заранее и остаются фиксированными во время процесса обучения.

Параметры модели: определяют саму модель и обновляются в процессе обучения.

Гиперпараметры: определяют структуру и методы обучения модели, но не обновляются в процессе обучения.

Изменение параметров происходит автоматически в процессе обучения, в то время как гиперпараметры обычно настраиваются вручную или с использованием методов поиска или оптимизации.

1. Что может пойти не так, если вы подгоните гиперпараметры модели, на тестовом наборе?

Использование тестового набора данных для подгонки гиперпараметров может привести к ряду проблем:

Переобучение на тестовых данных

Потеря возможности независимой оценки

Недостоверные метрики

Сложности в интерпретации

Вывод

Из-за всех этих проблем рекомендуется использовать отдельный валидационный набор данных для подгонки гиперпараметров, сохраняя тестовый набор для окончательной, независимой оценки производительности. Если доступны только ограниченные данные, можно применять методы, такие как кросс-валидация, для эффективной оценки и настройки гиперпараметров без использования тестового набора.

6. Что такое перекрестная проверка (cross-validation) и чем она лучше использования одного тестового множества?

Перекрестная проверка (cross-validation) — это метод статистической оценки эффективности модели, который используется для оценки того, насколько хорошо модель будет работать на независимом (и, возможно, неизвестном) наборе данных. Один из наиболее распространенных методов перекрестной проверки — k-блочная перекрестная проверка (k-fold cross-validation), при которой исходный набор данных разбивается на \( k \) (обычно равных по размеру) поднаборов. Один из этих поднаборов используется как тестовый набор данных, а остальные \( k-1 \) объединяются в обучающий набор данных. Этот процесс повторяется \( k \) раз, каждый раз с использованием разного поднабора как тестового. В итоге получается \( k \) оценок производительности, которые обычно усредняются для получения одной окончательной оценки.

Преимущества перекрестной проверки по сравнению с использованием одного тестового множества:

1. Робастность: Перекрестная проверка дает более надежную оценку производительности модели, потому что она оценивается на разных поднаборах данных.

2. Использование данных: В отличие от отделения части данных для единственного тестового набора, перекрестная проверка позволяет использовать все доступные данные для обучения и тестирования, что особенно полезно при небольших объемах данных.

3. Оценка вариабельности: Перекрестная проверка предоставляет не только одну оценку производительности, но и позволяет оценить вариабельность этой производительности. Это может быть полезным для определения "уверенности" в оценках модели.

4. Обобщение: так как модель тестируется на различных поднаборах данных, вы можете быть более уверены в том, что модель хорошо обобщает и будет хорошо работать на новых, неизвестных данных.

5. Отсутствие зависимости от разбиения: Оценка производительности модели может сильно зависеть от того, как именно были разделены обучающие и тестовые данные. Перекрестная проверка снижает эту зависимость.

6. Настройка гиперпараметров: Перекрестная проверка часто используется в процессах поиска гиперпараметров и выбора модели, предоставляя более надежные оценки производительности для каждого набора гиперпараметров.

Недостатки:

1. Вычислительная сложность: Одним из недостатков перекрестной проверки является увеличение вычислительной сложности, так как модель нужно обучать и тестировать k раз.

2. Не всегда применимо: в некоторых случаях (например, при временных рядах или при наличии структурных зависимостей в данных) стандартные методы перекрестной проверки могут быть неэффективными или неприменимыми.

Таким образом, перекрестная проверка предлагает более надежный и гибкий способ оценки производительности моделей, хотя и требует больше вычислительных ресурсов.

7. Что такое онлайновый режим работы системы машинного обучения?

В контексте машинного обучения термин "онлайновый режим" (или инкрементное обучение, incremental learning) обычно означает способ обучения модели, при котором она постоянно обновляется по мере поступления новых данных. Это противоположно "пакетному" (batch) режиму обучения, где модель обучается один раз на фиксированном наборе данных и после этого не обновляется.

Основные характеристики онлайнового режима:

1. Постоянное обновление: Модель обновляется каждый раз, когда появляются новые данные, что позволяет ей адаптироваться к изменениям.

2. Минимизация ресурсов: Обычно онлайн-обучение требует меньше вычислительных ресурсов за одну итерацию, так как модель обновляется на основе меньших объемов данных.

3. Работа в реальном времени: Онлайновый режим подходит для систем, которые должны реагировать на события в реальном времени, например, системы рекомендаций или системы обнаружения мошенничества.

4. Скорость адаптации: В условиях, где характер данных быстро меняется, онлайн-обучение может быть более эффективным, поскольку модель постоянно адаптируется.

5. Неограниченные данные: Онлайновый режим особенно полезен, когда объем данных слишком велик для хранения или обработки в пакетном режиме.

Примеры применения:

- Рекомендательные системы, которые обновляют свои предсказания на основе последних интеракций пользователя.

- Системы обнаружения мошенничества в финансовых операциях, которые постоянно адаптируются к новым стратегиям мошенников.

- Роботы и другие автономные системы, которые должны адаптироваться к изменяющимся условиям в реальном времени.

Особенности и вызовы:

- Забывание: Поскольку модель постоянно адаптируется, существует риск "забывания" старой информации.

- Шум и выбросы: Онлайн-обучение может быть более чувствительным к шуму и временным выбросам в данных.

- Сложность модели: Не все модели легко адаптируются к онлайновому режиму, и может потребоваться специальная архитектура или метод обучения.

В зависимости от задачи и доступных ресурсов онлайновый режим может быть как очень полезным, так и неприменимым.

8. Опишите классификатор по методу ближайших соседей (KNN): в чем заключаются режимы его обучения и предсказания (вывода)?

Классификатор по методу ближайших соседей (KNN, K-Nearest Neighbors) — это один из самых простых алгоритмов машинного обучения. Он применяется для задач классификации и регрессии и основан на "ленивом" обучении. Для работы KNN не требуется явного обучения модели; вместо этого все вычисления происходят на этапе предсказания (вывода).

Режимы работы:

Обучение

В классической реализации KNN "обучение" заключается просто в запоминании обучающих данных. Эти данные используются позже для классификации новых примеров на этапе предсказания. Это отличает KNN от большинства других алгоритмов, где на этапе обучения параметры модели оптимизируются.

Некоторые варианты KNN используют методы индексации (например, KD-деревья, ball trees и др.) для ускорения этапа предсказания, но это уже является оптимизацией, а не классическим вариантом KNN.

Предсказание (Вывод)

На этапе предсказания KNN ищет k ближайших соседей для нового примера из сохраненного обучающего набора данных. "Близость" определяется с помощью некоторой метрики расстояния (часто евклидова, манхэттенская или косинусная мера). Далее, для задачи классификации наиболее часто встречающийся класс среди k соседей выбирается как предсказанный класс для нового примера.

Пример:

Допустим, у нас есть набор данных с двумя классами: "Яблоки" и "Апельсины", и каждый фрукт описывается двумя признаками: "Краснота" и "Круглость". В этом примере:

- Обучение: Мы просто сохраняем все примеры из обучающего набора данных.

- Предсказание: Когда у нас появляется новый фрукт, мы ищем k ближайших соседей из сохраненного набора данных на основе признаков "Краснота" и "Круглость". Если большинство из k соседей — яблоки, новый фрукт классифицируется как яблоко; в противном случае — как апельсин.

9. Опишите модель линейной регрессии: в чем заключаются режимы ее обучения и предсказания (вывода)?

Линейная регрессия

Линейная регрессия — это статистическая модель, используемая для предсказания количественной зависимой переменной на основе одной или нескольких независимых переменных. В простой линейной регрессии используется одна независимая переменная, в то время как в множественной линейной регрессии используется две или более независимых переменных.

Модель представляет собой уравнение линии:

Где – предсказанное значение, – параметры модели, – независимые переменные, а – ошибка модели.

.

Режимы работы

Обучение

В режиме обучения задача состоит в оптимизации параметров таким образом, чтобы минимизировать функцию потерь, часто используемую в виде среднеквадратичной ошибки (MSE).

Оптимизация может быть выполнена различными методами, включая градиентный спуск, стохастический градиентный спуск или методы оптимизации второго порядка. В некоторых случаях (например, если матрица признаков имеет полный ранг и нет мультиколлинеарности) решение можно найти аналитически через нормальное уравнение.

Предсказание (Вывод)

На этапе предсказания линейная модель использует оптимизированные параметры для оценки зависимой переменной для новых данных. Это делается путем подстановки значений независимых переменных x в уравнение линии и вычисления .

10. В чем заключается разница между обучением с учителем и обучением без привлечения учителя? Какие ещё есть «промежуточные» типы обучения.

Обучение с учителем (Supervised Learning)

В этом типе обучения у нас есть размеченный набор данных, состоящий из входных переменных (признаков) и выходных переменных (меток или "ответов"). Цель заключается в том, чтобы обучить модель, которая сможет предсказать метку для новых, ранее неизвестных данных. Примеры задач, решаемых с помощью обучения с учителем, включают классификацию (например, фильтрация спама, распознавание рукописных цифр) и регрессию (например, предсказание цен на недвижимость).

Обучение без учителя (Unsupervised Learning)

В этом случае у нас есть только входные переменные, и нет соответствующих им выходных переменных. Задачей является моделирование структуры или распределения в данных для того, чтобы научиться делать что-то полезное без использования меток. Примеры включают кластеризацию (например, сегментация рынка), уменьшение размерности и обнаружение аномалий.

Промежуточные типы обучения

1. Частичное обучение (Semi-supervised Learning): Это промежуточный вариант между обучением с учителем и обучением без учителя. В этом случае у нас есть небольшой размеченный набор данных и значительно больший неразмеченный набор данных. Цель заключается в том, чтобы использовать больший объем неразмеченных данных для улучшения качества модели.

2. Обучение с подкреплением (Reinforcement Learning): Здесь агент взаимодействует с окружением, чтобы достигнуть цели или максимизировать некоторую долгосрочную выгоду. Нет явного набора правильных ответов, но есть награда, которая дается за принятые действия.

3. Многозадачное обучение (Multi-Task Learning): Модель обучается на наборе связанных задач с учителем, и знания, полученные на одной задаче, используются для улучшения производительности на других задачах.

4. Обучение с примерами противоположного класса (Learning from Positive and Unlabeled data, PU Learning): Этот тип обучения применяется, когда у нас есть доступ только к положительным и неразмеченным примерам. Отсутствуют явно размеченные отрицательные примеры.

5. Перенос обучения (Transfer Learning): Знания, приобретенные при решении одной задачи, применяются для решения другой, но связанной задачи.

6. Активное обучение (Active Learning): Модель активно выбирает, какие примеры из неразмеченного набора данных следует разметить и добавить в обучающий набор, чтобы максимально улучшить свою производительность.

11. Что такое переобучение?

Переобучение (overfitting) — это одна из основных проблем в машинном обучении, возникающая, когда модель слишком "хорошо" подстраивается под обучающие данные, за счет чего ухудшается её способность обобщать информацию на новых, неизвестных данных. В других словах, модель учится "наизусть" шум и случайные флуктуации в обучающем наборе данных вместо того, чтобы выявить общую закономерность, которая применима к новым данным.

12. Опишите кратко метод работы дерева решений и ансамбля таких деревьев в режиме взаимного усиления (мы знаем его как «бустинг»).

Дерево решений

Дерево решений — это алгоритм машинного обучения, используемый для задач классификации и регрессии. Оно представляет собой дерево, в котором каждый листовой узел представляет класс или значение целевой переменной, а каждый нелистовой узел представляет решающее правило, разделяющее данные на подгруппы. На этапе обучения дерево строится таким образом, чтобы минимизировать некоторую функцию потерь, часто используемую для оценки качества разделения данных.

Бустинг

Бустинг — это ансамблевый метод, который комбинирует несколько слабых моделей (часто деревьев решений) для создания одной сильной модели. В режиме взаимного усиления (boosting), каждое последующее дерево строится с учетом ошибок, совершенных предыдущими деревьями. Это делается путем увеличения весов неправильно классифицированных примеров, чтобы следующее дерево могло скорректировать эти ошибки. Затем предсказания всех деревьев комбинируются для получения окончательного ответа, часто с использованием взвешенного голосования или среднего.

Работа алгоритма

1. Инициализация: Сначала инициализируются веса каждого примера в обучающем наборе данных.

2. Построение дерева: Строится дерево решений на основе текущих весов примеров.

3. Вычисление ошибки: Оценивается ошибка дерева на обучающем наборе, учитывая веса примеров.

4. Обновление весов: Веса примеров обновляются так, чтобы увеличить влияние тех примеров, которые были классифицированы неправильно.

5. Комбинация моделей: Предсказания каждого дерева комбинируются для создания окончательного предсказания, часто с использованием весов, основанных на точности каждого дерева.

6. Остановка или продолжение: Процесс повторяется, пока не будет достигнуто заданное количество деревьев или не будет выполнен другой критерий остановки.