1. **Формула для линейной модели классификации:**

Для линейной классификации мы используем линейное уравнение для принятия решения о классе:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Что такое отступ?**

**Отступ** — это мера уверенности классификатора в своем предсказании. Отступ для объекта xix\_ixi​ определяется как расстояние между предсказанным значением и разделяющей гиперплоскостью. Это значение используется для оценки того, насколько правильно и уверенно классификатор предсказал класс объекта.

Формально, **отступ** (или **margin**) для объекта xi определяется как:

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Обучение линейных классификаторов**

**Линейные классификаторы**, такие как **логистическая регрессия**, **линейный SVM** и другие, обучаются путем минимизации **функции потерь**, которая измеряет, насколько сильно предсказания модели отклоняются от истинных меток классов. Основная цель — найти такие параметры модели (веса), которые минимизируют эту функцию потерь.

1. **Основные этапы обучения линейных классификаторов:**
2. **Инициализация весов**:
   * Изначально все веса wjw\_jwj​ (коэффициенты при признаках) и смещение w0w\_0w0​ инициализируются (чаще всего случайно или нулями).
3. **Определение функции потерь**:
   * Функция потерь оценивает, насколько хорошо или плохо модель классифицирует объекты.
   * Линейные модели используют различные функции потерь, такие как:
     + **Гингевская функция потерь** в **SVM**: измеряет насколько отступ объекта превышает или недотягивает до минимального отступа.
     + **Логистическая функция потерь** в **логистической регрессии**: измеряет, насколько уверенность модели в правильном предсказании отличается от истинного класса.
4. **Оптимизация весов**:
   * Модель использует методы оптимизации, такие как **градиентный спуск**, чтобы обновлять веса и минимизировать функцию потерь.
   * На каждом шаге оптимизации веса обновляются по направлению антиградиента функции потерь.
5. **Регуляризация**:
   * Для улучшения обобщающей способности модели и борьбы с переобучением в линейные классификаторы часто добавляют **регуляризацию** (L1 или L2). Регуляризация накладывает штрафы на большие значения весов, чтобы снизить сложность модели.

**Задача оптимизации:**

Для линейного классификатора задача оптимизации часто выглядит как:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Верхние оценки пороговой функции потерь**

Пороговые функции потерь имеют дискретный характер, что затрудняет их оптимизацию. Поэтому в большинстве случаев используются **верхние оценки** (или аппроксимации) пороговой функции потерь для того, чтобы сделать задачу оптимизации гладкой и пригодной для применения градиентных методов оптимизации.

**Что такое пороговая функция потерь?**

Пороговая функция потерь — это функция, которая принимает фиксированное значение, если предсказание классификатора достаточно далеко от пороговой гиперплоскости, и меняется, когда предсказание ближе к порогу.

Например, в задаче классификации можно использовать следующую пороговую функцию:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Зачем нужны верхние оценки?**

Использование пороговой функции затруднительно для оптимизации, потому что она дискретная (принимает только 0 или 1). Чтобы сделать задачу решаемой с помощью методов оптимизации, используют **гладкие аппроксимации** (или **верхние оценки**) пороговых функций потерь, которые дают **мягкие** переходы между правильными и неправильными предсказаниями.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

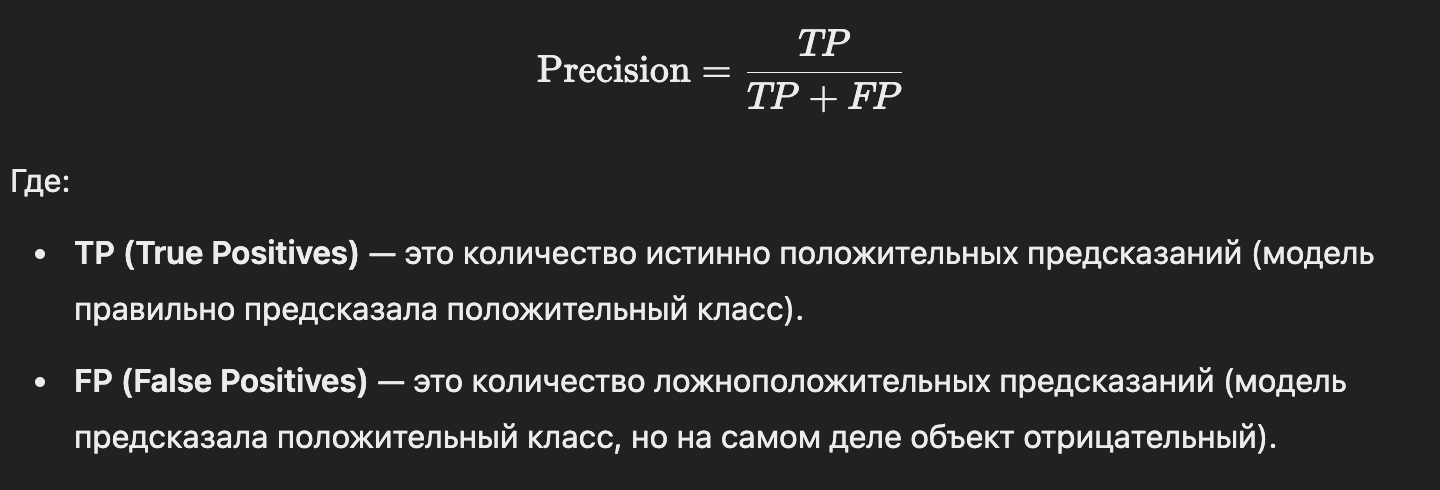
**Влияние верхних оценок на обучение:**

* **Сглаживание**: Верхние оценки позволяют сделать пороговую функцию потерь гладкой, что упрощает вычисление градиентов и делает возможным использование эффективных методов оптимизации, таких как **градиентный спуск**.
* **Устойчивость к выбросам**: Гладкие функции потерь, такие как логистическая или hinge, позволяют модели быть менее чувствительной к выбросам и небольшим изменениям в данных.
* **Регуляризация**: Верхние оценки могут включать регуляризационные члены, что позволяет модели избегать переобучения.

**Итоги:**

1. **Линейные классификаторы обучаются** путем минимизации функции потерь, которая оценивает, насколько предсказания модели отклоняются от реальных классов.
2. **Пороговая функция потерь** оценивает, правильно ли классифицирован объект, но она трудна для оптимизации из-за дискретного характера.
3. **Верхние оценки пороговой функции потерь** (например, hinge loss или логистическая функция) делают задачу оптимизации гладкой и более легко решаемой с помощью методов оптимизации, таких как градиентный спуск.
4. **Точность (Precision)**, **полнота (Recall)** и **F-мера** — это метрики, используемые для оценки качества моделей классификации, особенно когда классы несбалансированы, и важно учитывать как правильные, так и ошибочные предсказания.
5. **Точность (Precision)**

**Точность** (Precision) показывает, насколько модель точна при предсказании положительного класса. Это отношение числа **истинно положительных** (TP) предсказаний к количеству всех объектов, которые модель предсказала как положительные (**истинно положительные** + **ложно положительные**, то есть TP + FP).



**Полнота (Recall)**

**Полнота** (Recall) показывает, какую долю **всех реальных положительных объектов** модель правильно классифицировала. Это отношение числа **истинно положительных** предсказаний к общему числу **реально положительных** объектов (TP + FN).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**F-мера (F1-score)**

**F-мера** — это гармоническое среднее между **точностью** и **полнотой**. Она используется, чтобы найти баланс между точностью и полнотой. В отличие от простого среднего, F-мера меньше наказывается при большой разнице между точностью и полнотой.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

**Почему F-мера лучше, чем арифметическое среднее или минимум?**

1. **Баланс между Precision и Recall**:
   * **Арифметическое среднее** может давать высокое значение даже тогда, когда один из показателей (точность или полнота) очень мал. Например, если точность равна 1, а полнота равна 0,5, их среднее будет (1+0.5)/2=0.75(1 + 0.5) / 2 = 0.75(1+0.5)/2=0.75, хотя качество модели не должно оцениваться так высоко, если один из показателей далек от идеала.
   * **Минимум** всегда дает самое худшее значение, что может быть слишком пессимистичным, особенно если один показатель высокий, а другой низкий.
2. **Гармоническое среднее**:
   * **F-мера** является **гармоническим средним**, которое penalizes большие различия между Precision и Recall. Это делает F-мера более точной, когда нужно учитывать как точность, так и полноту одновременно.
   * Если один из показателей (точность или полнота) близок к 0, **F-мера** тоже будет близка к 0, даже если другой показатель высок.
3. **Равное внимание обоим показателям**:
   * F-мера наказывает модель, если один из показателей сильно превышает другой, в то время как простое среднее может давать завышенные результаты.

**Пример:**

Представим ситуацию:

* **Точность** = 0.9 (то есть модель правильно классифицирует 90% объектов, предсказанных как положительные),
* **Полнота** = 0.5 (модель находит только 50% реальных положительных объектов).
* **Арифметическое среднее**:

Изображение выглядит как текст, Шрифт, линия, типография

Автоматически созданное описание

Это значение может создать впечатление, что модель достаточно хороша, хотя в действительности она плохо находит положительные объекты.

**Минимум**:

Изображение выглядит как Шрифт, Графика, текст, логотип

Автоматически созданное описание

Это слишком пессимистичный результат, поскольку точность всё-таки очень высока.

**F-мера**:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Этот результат более сбалансирован и показывает, что модель не так хороша, как показывает точность, но лучше, чем указывает минимум. Он уравновешивает оба показателя.

**Когда использовать F-меру?**

* **F-мера** полезна, когда у вас есть **несбалансированные данные** (например, в медицинской диагностике, обнаружении мошенничества и т.д.), и важно находить баланс между точностью (избегание ложноположительных предсказаний) и полнотой (избегание пропусков положительных примеров).
* В задачах, где важнее не упустить положительные объекты (например, медицинская диагностика), метрики, такие как **полнота**, могут иметь более высокий приоритет, но **F-мера** поможет сбалансировать её с точностью.

**Итоги:**

* **Precision** показывает, насколько модель точна при предсказании положительного класса.
* **Recall** измеряет способность модели находить все реальные положительные объекты.
* **F-мера** — это гармоническое среднее между Precision и Recall, которое делает её более сбалансированной и чувствительной к большому разрыву между этими двумя метриками.

**F-мера** лучше, чем арифметическое среднее или минимум, потому что она не "раздувает" метрику, если один из показателей сильно хуже другого, а также наказывает модель за большие различия между точностью и полнотой, что делает её более точной и полезной для сбалансированной оценки модели.

1. **Порог** в линейном классификаторе используется для того, чтобы решить, к какому классу отнести объект на основе непрерывного предсказания модели. Модель линейной классификации предсказывает некоторое значение, например, вероятность или оценку (линейную комбинацию признаков), и порог определяет, при каком значении этого предсказания объект будет отнесен к положительному или отрицательному классу.

**Зачем нужен порог?**

1. **Классификация на основе непрерывного предсказания**:
   * Линейные классификаторы, такие как **логистическая регрессия** или **линейный SVM**, предсказывают не просто класс, а **непрерывное значение**. Например, в логистической регрессии это вероятность того, что объект принадлежит положительному классу.
   * Порог позволяет преобразовать это непрерывное значение в дискретный класс (например, "1" или "0"). Например, в логистической регрессии обычно используется порог 0.50.50.5, что означает:
     + Если вероятность > 0.5, объект классифицируется как "1" (положительный класс).
     + Если вероятность <= 0.5, объект классифицируется как "0" (отрицательный класс).
2. **Гибкость в настройке модели**:
   * Порог даёт возможность **гибко настраивать модель** в зависимости от задачи. Например, если для вас важно увеличить полноту (Recall) и вы готовы смириться с большим количеством ложноположительных предсказаний (False Positives), можно снизить порог ниже 0.50.50.5. Это приведет к тому, что больше объектов будут классифицированы как положительные, что увеличит полноту.
   * С другой стороны, если важно минимизировать ложные срабатывания и повысить точность (Precision), порог можно увеличить.

**Как выбирается порог?**

Выбор порога зависит от **приоритетов задачи**, целей модели и типа данных. Вот основные факторы, которые влияют на выбор порога:

**1. Задачи классификации и приоритеты**

* **Баланс между точностью и полнотой**:
  + Если для задачи важно обнаружить как можно больше положительных случаев (например, при диагностике заболеваний), нужно снизить порог, чтобы повысить **полноту (Recall)**.
  + Если важнее избежать ложных срабатываний (например, для спам-фильтра, где ошибки могут блокировать важные письма), порог может быть повышен, чтобы повысить **точность (Precision)**.

**2. Несбалансированные данные**

* В задачах с **несбалансированными данными** (например, когда положительных примеров очень мало), использование стандартного порога (например, 0.50.50.5) может привести к тому, что модель будет слишком часто предсказывать отрицательный класс. В таких случаях порог снижается для того, чтобы уравновесить вклад положительных примеров.

Пример:

* + Если доля положительных объектов мала (например, только 5% объектов принадлежат положительному классу), порог можно снизить до 0.20.20.2, чтобы компенсировать несбалансированность и находить больше положительных примеров.

**3. Метрики, такие как Precision, Recall и F1-score**

* Иногда порог подбирается таким образом, чтобы **максимизировать** конкретные метрики модели, такие как **F1-score**, который учитывает баланс между точностью и полнотой.
* Можно выбирать порог, при котором достигается **максимальный F1-score**, или порог, где обе метрики (точность и полнота) будут удовлетворять определённым требованиям.

**4. Кривая ROC и AUC**

* **ROC-кривая** (Receiver Operating Characteristic) показывает зависимость между **True Positive Rate (полнота)** и **False Positive Rate** для всех возможных значений порога.
* **AUC (Area Under the ROC Curve)** — это площадь под ROC-кривой, которая оценивает способность модели различать классы.
* Для выбора оптимального порога можно использовать **ROC-кривую**, чтобы найти такой порог, при котором будет достигнут баланс между количеством истинных и ложных положительных предсказаний.

**5. Затраты и риски ошибок**

* В некоторых задачах ошибки классификации имеют разные "стоимости". Например, в банковском деле ложный отрицательный результат (неопределение мошенничества) может стоить гораздо дороже, чем ложноположительный (ошибочная блокировка транзакции).
* В таких случаях порог может быть выбран таким образом, чтобы минимизировать **общие затраты**, связанные с ошибками.

1. **AUC-ROC** — это один из важных методов для оценки качества бинарной классификации.

* **ROC (Receiver Operating Characteristic)** — это кривая, которая показывает соотношение между **True Positive Rate (TPR)** (или полнотой) и **False Positive Rate (FPR)** для различных порогов классификации.
* **AUC (Area Under the Curve)** — это площадь под ROC-кривой, которая оценивает общую способность модели различать между положительными и отрицательными классами. Чем больше AUC (максимум = 1), тем лучше модель различает между классами.

**Компоненты AUC-ROC:**

1. **True Positive Rate (TPR)** (также называемая **Recall** или **полнота**):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**False Positive Rate (FPR)**:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Что такое AUC-ROC?**

* **ROC-кривая** — это график, на котором по оси Y откладывается **True Positive Rate (Recall)**, а по оси X — **False Positive Rate (FPR)**.
* **AUC** (площадь под кривой ROC) — это числовое значение, которое показывает, насколько хорошо модель классифицирует объекты. AUC варьируется от 0 до 1:
  + **AUC = 1**: Идеальная модель, которая всегда правильно классифицирует положительные и отрицательные объекты.
  + **AUC = 0.5**: Модель, которая классифицирует на уровне случайного угадывания.
  + **AUC < 0.5**: Модель, которая классифицирует хуже, чем случайное угадывание.

**Алгоритм построения ROC-кривой:**

1. **Предсказание вероятностей**:
   * Для каждого объекта в выборке модель бинарной классификации предсказывает вероятность того, что объект принадлежит положительному классу (например, для логистической регрессии это может быть вероятность принадлежности к классу "1").
2. **Упорядочение предсказаний**:
   * Все предсказанные вероятности сортируются в порядке убывания (от наибольшей к наименьшей).
3. **Перебор всех возможных порогов**:
   * Начинаем с самого высокого порога: считаем, что объект принадлежит классу "1", если вероятность больше этого порога. Считаем, что все объекты ниже этого порога принадлежат классу "0".
   * Порог изменяется последовательно от наибольшего до наименьшего значения предсказанной вероятности.
4. **Для каждого порога**:
   * Рассчитываются значения **True Positive Rate (TPR)** и **False Positive Rate (FPR)**.
   * **TPR** увеличивается, когда мы правильно классифицируем объект как положительный (то есть находим новый **True Positive**).
   * **FPR** увеличивается, если объект, который на самом деле отрицательный, был ошибочно классифицирован как положительный (**False Positive**).
5. **Построение ROC-кривой**:
   * По мере изменения порога, мы строим точки ROC-кривой на графике, где ось Y — это **TPR**, а ось X — это **FPR**.
   * Соединяем эти точки линиями, чтобы получить ROC-кривую.
6. **Вычисление AUC**:
   * Площадь под ROC-кривой можно рассчитать численно, чтобы получить значение **AUC**. Это значение используется для оценки общей производительности модели.

**Интерпретация:**

* **Высокий AUC** (близкий к 1) указывает на хорошую модель, которая хорошо различает положительные и отрицательные классы.
* **Низкий AUC** (около 0.5) говорит о том, что модель классифицирует на уровне случайного угадывания.
* **ROC-кривая** дает представление о том, как меняются ошибки модели (False Positives и True Positives) при изменении порога классификации.

**Заключение:**

**AUC-ROC** — это мощный инструмент для оценки моделей бинарной классификации, особенно когда важно оценить баланс между **True Positives** и **False Positives**. ROC-кривая показывает, как изменяются эти значения при изменении порога классификации, а AUC численно оценивает качество модели, предоставляя сводное значение.

1. **AUC-PRC (Area Under the Precision-Recall Curve)** — это метрика, которая измеряет качество классификации модели на основе **Precision-Recall Curve (PR-кривая)**. PR-кривая лучше подходит для оценки моделей, работающих с **несбалансированными данными**, когда положительных примеров мало, а отрицательных много.

**PR-кривая:**

**Precision-Recall (PR)** — это кривая, которая показывает зависимость между **точностью (Precision)** и **полнотой (Recall)** при изменении порога классификации.

* **Precision (точность)**: Это доля истинно положительных предсказаний среди всех предсказанных положительных классов.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

**Recall (полнота)**: Это доля истинно положительных предсказаний среди всех реальных положительных объектов.

Изображение выглядит как текст, Шрифт, часы, число

Автоматически созданное описание

**Что такое AUC-PRC?**

* **PR-кривая** (Precision-Recall) показывает, как меняется точность (Precision) при изменении полноты (Recall) для всех возможных порогов классификации.
* **AUC-PRC** (Area Under the Precision-Recall Curve) — это площадь под PR-кривой. Она показывает, насколько хорошо модель справляется с задачей нахождения положительных объектов, учитывая как точность, так и полноту.
* AUC-PRC варьируется от 0 до 1:
  + **AUC-PRC = 1**: Модель идеальна — находит все положительные объекты, не делая ложных срабатываний.
  + **AUC-PRC = 0.5**: Модель классифицирует на уровне случайного угадывания.
  + **AUC-PRC = 0**: Модель полностью ошибается.

**Отличие AUC-ROC от AUC-PRC:**

* **AUC-ROC** подходит для сбалансированных данных, когда классы положительные и отрицательные примерно одинаково представлены.
* **AUC-PRC** лучше для **несбалансированных данных**, когда положительных примеров мало, и важно учитывать ложные срабатывания (FP) относительно всей популяции.

**Алгоритм построения PR-кривой:**

1. **Предсказание вероятностей**:
   * Как и в случае с ROC-кривой, модель предсказывает **вероятности** принадлежности объектов к положительному классу (например, в логистической регрессии).
2. **Упорядочение предсказаний**:
   * Все предсказанные вероятности сортируются в порядке убывания (от наибольшего к наименьшему).
3. **Перебор всех возможных порогов**:
   * Для каждого возможного порога определяем:
     + **Precision (точность)** — долю правильных положительных предсказаний среди всех предсказанных положительных объектов.
     + **Recall (полнота)** — долю всех реальных положительных объектов, которые были правильно классифицированы моделью.
4. **Для каждого порога**:
   * Вычисляются точность (Precision) и полнота (Recall), исходя из того, как изменяется количество **TP**, **FP**, **FN** при изменении порога.
   * Чем ниже порог, тем больше объектов предсказываются как положительные (повышается Recall), но может уменьшаться Precision.
5. **Построение PR-кривой**:
   * По оси X откладывается **Recall**, а по оси Y — **Precision**.
   * Для каждого порога строится точка (Precision, Recall) на графике.
   * Соединяем все точки, чтобы получить PR-кривую.
6. **Вычисление AUC-PRC**:
   * Площадь под PR-кривой можно рассчитать численно для получения значения **AUC-PRC**.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

**Пример:**

* Если модель предсказывает вероятность 0.80.80.8, это значит, что модель "уверена" на 80%, что объект принадлежит положительному классу.
* Вероятность 0.20.20.2 будет означать, что модель считает, что объект с вероятностью 20% принадлежит к положительному классу (и, соответственно, с вероятностью 80% — к отрицательному).

**Как внедрить это требование в обучение модели?**

Чтобы модель могла предсказывать **вероятности** принадлежности к классу, а не просто класс, необходимо использовать такие алгоритмы и функции потерь, которые предназначены для предсказания вероятностей. Основная идея заключается в том, чтобы модель не просто выдавала "жесткое" решение (например, "да" или "нет"), а давала вероятность того, что объект принадлежит к положительному классу.

**Способы обучения моделей для предсказания вероятностей:**

1. **Логистическая регрессия**:
   * Это классический метод, который напрямую обучается предсказывать вероятности.
   * Логистическая регрессия использует **логистическую функцию (сигмоидную функцию)** для преобразования линейной комбинации признаков в значение от 0 до 1, что интерпретируется как вероятность положительного класса.

Формула для логистической регрессии:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Алгоритм обучается путем минимизации **логистической функции потерь** (log loss):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

 **Методы на основе градиентного бустинга**:

* Некоторые модели, такие как **XGBoost**, **LightGBM** или **CatBoost**, также могут предсказывать вероятности.
* Эти модели могут быть настроены так, чтобы возвращать вероятности вместо "жестких" решений (классов). Для этого нужно установить параметр, который управляет возвращаемыми значениями (например, predict\_proba() в sklearn).

 **SVM с вероятностными предсказаниями**:

* Стандартная модель **SVM (Support Vector Machine)** не предсказывает вероятности, но можно включить **метод Платта** (Platt scaling), чтобы преобразовать "сырые" оценки модели в вероятности.
* Метод Платта обучает вторую логистическую модель на выходе SVM, чтобы скорректировать предсказания в вероятности.

 **Наивный Байес**:

* **Наивный Байес** также является алгоритмом, который напрямую предсказывает вероятность класса, основываясь на законе Байеса:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. **Нейронные сети**:
   * Нейронные сети также могут быть настроены для предсказания вероятностей.
   * Для этого последним слоем в нейронной сети ставится **сигмоидная функция** (для бинарной классификации) или **softmax** (для многоклассовой классификации).
   * Нейронная сеть обучается с использованием **кросс-энтропии** (cross-entropy loss), которая является стандартной функцией потерь для предсказания вероятностей.

**Как внедрить это требование в процедуру обучения?**

1. **Выбор подходящего алгоритма**:
   * Используйте модели, которые могут возвращать вероятности, такие как логистическая регрессия, градиентный бустинг или нейронные сети с подходящей функцией активации.
2. **Использование правильной функции потерь**:
   * Для предсказания вероятностей необходимо использовать функции потерь, которые поощряют точность предсказания вероятностей, а не просто точные классы. Например:
     + **Логистическая функция потерь (log loss)** для бинарной классификации.
     + **Кросс-энтропия (cross-entropy)** для многоклассовой классификации.
3. **Использование метода predict\_proba()**:
   * В библиотеках, таких как scikit-learn, многие модели имеют метод predict\_proba(), который возвращает вероятности для каждого класса. Использование этого метода позволяет получать вероятностные предсказания вместо дискретных классов.
4. **Оценка качества предсказанных вероятностей**:
   * Используйте метрики, которые оценивают **качество вероятностных предсказаний**, такие как **логарифмическая потеря (log loss)**, **Brier score**, а также **AUC-ROC** для оценки ранжирования вероятностей.

**Зачем нужно предсказывать вероятности?**

1. **Гибкость при выборе порога**:
   * Модели, которые предсказывают вероятности, позволяют настроить порог для классификации, что полезно в задачах, где важно найти баланс между точностью (precision) и полнотой (recall). Например, можно задать порог больше или меньше 0.5 для того, чтобы увеличить полноту или точность.
2. **Работа с несбалансированными данными**:
   * В задачах с несбалансированными классами (например, выявление мошенничества или диагностика заболеваний) использование вероятностей помогает лучше контролировать компромисс между ложноположительными и ложноотрицательными ошибками.
3. **Ранжирование объектов**:
   * Вероятности могут использоваться для **ранжирования** объектов по вероятности принадлежности к положительному классу. Это полезно в задачах, где нужно определить приоритет обработки объектов.

**Заключение:**

Когда модель **оценивает вероятность положительного класса**, она предсказывает не просто класс объекта, а вероятность того, что объект принадлежит к классу "1". Это позволяет модели быть более гибкой и адаптивной в задачах с несбалансированными данными и настраивать порог для классификации в зависимости от конкретных требований задачи. Внедрить это требование можно, выбирая соответствующие модели (логистическая регрессия, градиентный бустинг, нейронные сети) и функции потерь, которые обучают модель предсказывать вероятности, а не классы.

Функционал логистической регрессии напрямую связан с **методом максимума правдоподобия**, поскольку логистическая регрессия оптимизирует параметры модели таким образом, чтобы максимизировать правдоподобие наблюдаемых данных.

1. **Функционал логистической регрессии**

Основной задачей логистической регрессии является предсказание вероятности принадлежности объекта к положительному классу. Для бинарной классификации, где метки классов y∈{0,1}вероятность того, что объект принадлежит к классу 1, выражается с помощью **сигмоидной функции**:

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Для вероятности отрицательного класса (y=0):

Изображение выглядит как Шрифт, черный, типография, текст

Автоматически созданное описание

**Функционал логистической регрессии (Log-Loss)**

Задача логистической регрессии — **минимизировать отрицательный логарифм правдоподобия**. Этот функционал также известен как **log-loss** (логистическая функция потерь):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Связь с методом максимума правдоподобия**

**Метод максимума правдоподобия** заключается в нахождении таких параметров модели (веса w и смещение w0), которые **максимизируют правдоподобие** наблюдаемых данных. Поскольку оптимизация модели чаще всего проводится с помощью методов минимизации (например, градиентного спуска), логистическая регрессия минимизирует отрицательный логарифм правдоподобия, который является эквивалентом максимизации самого правдоподобия.

Таким образом:

* Логистическая регрессия обучается с использованием функции потерь (log-loss), которая является **отрицательным логарифмом правдоподобия**.
* В процессе обучения логистическая регрессия находит такие веса www, которые **максимизируют правдоподобие наблюдений** (данных) и минимизируют log-loss.

Логистическая регрессия максимизирует правдоподобие наблюдаемых данных, а функционал логистической регрессии (log-loss) — это отрицательный логарифм правдоподобия, который минимизируется для нахождения оптимальных весов модели.