



(つ・し・の)つ ♥ Поступашки - ШАД, Стажировки и Магистратура ♥

[t.me/postypashki-old](https://t.me/postypashki-old)

# Содержание

<b>1. Линейные модели</b>	<b>6</b>
Опишите задачу машинного обучения. Дайте определение объекту, целевой переменной, признакам, модели, функционалу ошибки. . . . .	6
Чем отличается функция потерь от функционала ошибки? . . . . .	6
Какие функции потерь используются при решении задачи регрессии? . . . . .	7
Запишите формулу для линейной модели регрессии. . . . .	7
Чем отличаются функционалы MSE и MAE? В каких случаях лучше использовать MSE, а в каких MAE? . . . . .	8
Чем отличается MAE от MAPE? Что более понятно заказчику продукта? . . . . .	9
Что такое коэффициент детерминации? Как интерпретировать его значения? . . . . .	9
Чем log-cosh лучше функции потерь Хубера? Опишите обе функции потерь. . . . .	10
Что такое градиент? Какое его свойство используется при минимизации функций? . . . . .	10
Что такое градиентный спуск? Опишите процесс алгоритма. . . . .	11
Почему не всегда можно использовать полный градиентный спуск? Какие способы оценивания градиента вы знаете? Почему в стохастическом градиентном спуске важно менять длину шага по мере итераций? Какие стратегии изменения шага вы знаете? . . . . .	11
Что такое переобучение? Как можно отследить переобучение модели? . . . . .	12
Как построить итоговую модель после того, как по кросс-валидации подобраны оптимальные гиперпараметры? . . . . .	14
Что такое регуляризация? Для чего используется? . . . . .	14
Опишите, как работают L1- и L2-регуляризаторы. . . . .	15
Почему L1-регуляризация отбирает признаки? . . . . .	15
Почему плохо накладывать регуляризацию на свободный коэффициент? . . . . .	16
Что такое ElasticNet? . . . . .	16
Где используется метод максимального правдоподобия? . . . . .	17
Расскажите про метрики, которые штрафуют за перепрогноз сильнее, чем за недопрогноз и наоборот (pinball loss). . . . .	17
Расскажите про виды скейлинга. Зачем они нужны? . . . . .	18
Как записываются аналитические решения? Какие у них проблемы? . . . . .	19

<b>2. Классификация</b>	<b>19</b>
Запишите формулу для линейной модели классификации. Что такое отступ? . . . . .	19
Как обучаются линейные классификаторы и для чего нужны верхние оценки пороговой функции потерь? . . . . .	20
Что такое точность, полнота и F-мера? Почему F-мера лучше арифметического среднего и минимума? . . . . .	20
Для чего нужен порог в линейном классификаторе? Из каких соображений он может выбираться? . . . . .	21
Что такое AUC-ROC? Опишите алгоритм построения ROC-кривой. . . . .	22
Что такое AUC-PRC? Опишите алгоритм построения PR-кривой. . . . .	22
Что означает “модель оценивает вероятность положительного класса”? Как можно внедрить это требование в процедуру обучения модели? . . . . .	23
Запишите функционал логистической регрессии. Как он связан с методом максимума правдоподобия? . . . . .	24
Когда используется accuracy? . . . . .	24
Как бороться с дисбалансом классов? . . . . .	25
<b>3. Многоклассовая классификация</b>	<b>26</b>
Как измеряется качество в задаче многоклассовой классификации? . . . . .	26
Расскажите микро и макро-усреднение? . . . . .	26
Что такое mean-target encoding? . . . . .	27
Может ли mean-target encoding привести к переобучению? Как можно этого избежать? . . . . .	28
Что такое решающее дерево? . . . . .	28
Опишите жадный алгоритм обучения решающего дерева. . . . .	29
Почему с помощью бинарного решающего дерева можно достичь нулевой ошибки на обучающей выборке без повторяющихся объектов? . . . . .	29
Что такое критерий хаотичности? Как он используется для выбора предиката во внутренней вершине решающего дерева? . . . . .	30
В чем отличия энтропийного критерия и критерия Джини? . . . . .	30
Как связаны линейные модели и решающие деревья? . . . . .	31
Как посчитать хаотичность вершины в задаче классификации? А в задаче регрессии? . . . . .	31
В чем заключается метод опорных векторов? . . . . .	32

В чем заключается метод k-ближайших соседей? . . . . .	33
Нужно ли заниматься предобработкой данных в случае дерева? . . . . .	33
Как деревья работают с Nan? . . . . .	34
Как деревья работают с категориальными значениями? . . . . .	34
Как сделать многоклассовую классификацию через логрег? . . . . .	35
<b>4. Леса</b>	<b>35</b>
Приведите пример семейства алгоритмов с низким смещением и большим разбросом. . . . .	35
Приведите пример семейства алгоритмов с большим смещением и низким разбросом. . . . .	36
Что такое бэггинг? Как его смещение и разброс связаны со смещением и разбросом базовых моделей? . . . . .	36
Что такое случайный лес? Чем он отличается от бэггинга над решающими деревьями? . . . . .	37
Перечислите основные плюсы решающего леса. . . . .	37
Назовите недостатки случайного леса. . . . .	38
Как работает алгоритм градиентного бустинга. . . . .	38
Как обычно выглядят базовые модели в бустинге. Опишите почему? . . . . .	39
Что такое сдвиги в градиентном бустинге, зачем они нужны? . . . . .	39
Расскажите про виды бустинга (CatBoost, LightGBM, XGBoost), их особенности и различия.	40
Почему дерево строится по MSE (идея, что связано с косинусным расстоянием)? . . . . .	41
Как работает OOB (Out-of-Bag)? . . . . .	41
Как работает feature importance? . . . . .	42
<b>5. Кластеризация</b>	<b>42</b>
Опишите задачу кластеризации. Приведите примеры. . . . .	42
Метрики качества кластеризации. . . . .	43
Расскажите про алгоритмы кластеризации, которые вы знаете (например, k-means, DBSCAN...). . . . .	43
<b>6. Computer Vision (CV)</b>	<b>44</b>
Что такое свёрточные нейронные сети (CNN)? . . . . .	44
Как работает свёртка? . . . . .	45
Пример свёртки . . . . .	45

---

Как работает архитектура ResNet? . . . . .	45
Зачем нужны skip-connections? . . . . .	45
Что такое пулинг (Pooling)? . . . . .	46
Виды пулинга . . . . .	46
Пример пулинга . . . . .	46
<b>7. Bias-Variance Decomposition</b>	<b>46</b>
Математическая формула . . . . .	47
Пример . . . . .	47
Интерпретация . . . . .	47
Связь с переобучением . . . . .	48

## 1. Линейные модели

**Опишите задачу машинного обучения. Дайте определение объекту, целевой переменной, признакам, модели, функционалу ошибки.**

**Задача машинного обучения** заключается в том, чтобы на основе данных научиться предсказывать или классифицировать значения целевой переменной, используя признаки объектов. Основные компоненты задачи машинного обучения:

- **Объект** — это элемент данных, который описывается набором признаков. Например, объектом может быть клиент, описываемый возрастом, доходом и историей покупок.
- **Целевая переменная** — это значение, которое мы хотим предсказать. Например, в задаче классификации это может быть класс объекта (спам/не спам), а в задаче регрессии — числовое значение (цена дома).
- **Признаки** — это характеристики объекта, которые используются для предсказания целевой переменной. Например, признаки клиента могут включать возраст, доход и историю покупок.
- **Модель** — это математическая функция, которая на основе признаков объекта предсказывает значение целевой переменной. Например, линейная модель регрессии:

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b,$$

где  $y$  — предсказанное значение,  $x_1, x_2, \dots, x_n$  — признаки,  $w_1, w_2, \dots, w_n$  — веса модели,  $b$  — свободный коэффициент.

- **Функционал ошибки** — это мера, которая оценивает, насколько предсказания модели отличаются от истинных значений целевой переменной. Например, среднеквадратичная ошибка (MSE):

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

где  $y_i$  — истинное значение,  $\hat{y}_i$  — предсказанное значение,  $N$  — количество объектов.

**Чем отличается функция потерь от функционала ошибки?**

**Функция потерь** — это мера ошибки для одного объекта, которая показывает, насколько предсказание модели отличается от истинного значения. Например, квадратичная функция потерь:

$$L(y, \hat{y}) = (y - \hat{y})^2.$$

**Функционал ошибки** — это среднее значение функции потерь по всей выборке. Например, среднеквадратичная ошибка (MSE):

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{y}_i).$$

Таким образом, функция потерь применяется к одному объекту, а функционал ошибки — ко всей выборке.

## Какие функции потерь используются при решении задачи регрессии?

В задаче регрессии используются следующие функции потерь:

- Среднеквадратичная ошибка (MSE):

$$L(y, \hat{y}) = (y - \hat{y})^2.$$

Она штрафует большие отклонения сильнее, чем маленькие.

- Средняя абсолютная ошибка (MAE):

$$L(y, \hat{y}) = |y - \hat{y}|.$$

Она менее чувствительна к выбросам, чем MSE.

- Функция Хубера(HuberLoss):

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{если } |y - \hat{y}| \leq \delta, \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta), & \text{иначе.} \end{cases}$$

Она сочетает свойства MSE и MAE, устойчива к выбросам.

- Log-Cosh:

$$L(y, \hat{y}) = \log(\cosh(y - \hat{y})).$$

Она гладкая и менее чувствительна к выбросам, чем MSE.

## Запишите формулу для линейной модели регрессии.

Линейная модель регрессии предсказывает значение целевой переменной  $y$  на основе признаков  $x_1, x_2, \dots, x_n$  и весов  $w_1, w_2, \dots, w_n$ . Формула линейной модели:

$$y = w_1x_1 + w_2x_2 + \dots + w_nx_n + b,$$

где:

- $y$  — предсказанное значение,
- $x_1, x_2, \dots, x_n$  — признаки объекта,
- $w_1, w_2, \dots, w_n$  — веса модели,
- $b$  — свободный коэффициент (смещение).

**Чем отличаются функционалы MSE и MAE? В каких случаях лучше использовать MSE, а в каких MAE?**

MSE (Mean Squared Error) и MAE (Mean Absolute Error) — это два популярных функционала ошибки, используемых в задачах регрессии. Их основные различия:

- MSE:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2.$$

- Квадратично штрафует большие ошибки, что делает её чувствительной к выбросам.
- Подходит для случаев, когда большие ошибки недопустимы (например, в финансовых прогнозах).

- MAE:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|.$$

- Линейно штрафует ошибки, что делает её устойчивой к выбросам.
- Подходит для случаев, когда выбросы не должны сильно влиять на модель (например, в задачах с зашумлёнными данными).

**Когда использовать MSE:**

- Когда большие ошибки критичны.
- Когда данные содержат мало выбросов.

**Когда использовать MAE:**

- Когда данные содержат выбросы.
- Когда важна устойчивость к зашумлённым данным.

## Чем отличается MAE от MAPE? Что более понятно заказчику продукта?

MAE (Mean Absolute Error) и MAPE (Mean Absolute Percentage Error) — это метрики ошибки, используемые в задачах регрессии. Их основные отличия:

- MAE:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|.$$

- Измеряет абсолютную ошибку в единицах целевой переменной.
- Не зависит от масштаба данных.

- MAPE:

$$\text{MAPE} = \frac{1}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\%.$$

- Измеряет ошибку в процентах от истинного значения.
- Зависит от масштаба данных: если  $y_i$  близко к нулю, MAPE может быть нестабильным.

## Что более понятно заказчику:

- MAPE более понятен заказчику, так как выражает ошибку в процентах, что интуитивно понятно.
- Однако MAE предпочтительнее, если данные содержат нулевые или близкие к нулю значения.

## Что такое коэффициент детерминации? Как интерпретировать его значения?

Коэффициент детерминации ( $R^2$ ) — это метрика, которая показывает, насколько хорошо модель объясняет вариацию целевой переменной. Формула:

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2},$$

где:

- $y_i$  — истинное значение,
- $\hat{y}_i$  — предсказанное значение,
- $\bar{y}$  — среднее значение целевой переменной.

## Интерпретация:

- $R^2 = 1$ : модель идеально объясняет данные.
- $R^2 = 0$ : модель не лучше, чем предсказание средним значением.
- $R^2 < 0$ : модель работает хуже, чем предсказание средним значением.

**Чем log-cosh лучше функции потерь Хубера? Опишите обе функции потерь.**

**Log-Cosh и функция Хубера** — это функции потерь, используемые в задачах регрессии для устойчивости к выбросам.

- **Функция Хубера:**

$$L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{если } |y - \hat{y}| \leq \delta, \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta), & \text{иначе.} \end{cases}$$

- Сочетает свойства MSE и MAE: квадратична для малых ошибок и линейна для больших.
- Требует выбора параметра  $\delta$ .

- **Log-Cosh:**

$$L(y, \hat{y}) = \log(\cosh(y - \hat{y})).$$

- Аппроксимирует функцию Хубера, но не требует выбора параметра  $\delta$ .
- Гладкая и дифференцируемая, что упрощает оптимизацию.
- Менее чувствительна к выбросам, чем MSE.

**Преимущества Log-Cosh:**

- Не требует настройки параметров.
- Более гладкая, что упрощает градиентные методы оптимизации.
- Устойчива к выбросам.

**Что такое градиент? Какое его свойство используется при минимизации функций?**

**Градиент** — это вектор, состоящий из частных производных функции по всем её аргументам. Для функции  $f(x_1, x_2, \dots, x_n)$  градиент обозначается как:

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right).$$

**Свойство градиента, используемое при минимизации:**

- Градиент указывает направление наискорейшего роста функции. Следовательно, антиградиент  $(-\nabla f)$  указывает направление наискорейшего убывания функции.
- Это свойство используется в градиентных методах оптимизации, таких как градиентный спуск, для нахождения минимума функции.

## Что такое градиентный спуск? Опишите процесс алгоритма.

**Градиентный спуск** — это итерационный метод оптимизации, используемый для минимизации функции. Основная идея заключается в том, чтобы двигаться в направлении, противоположном градиенту функции, чтобы достичь её минимума.

### Процесс алгоритма:

- **Инициализация:** Выбирается начальная точка  $x_0$  и learning rate (скорость обучения)  $\eta$ .
- **Итерации:** На каждом шаге  $k$  обновляется текущая точка:

$$x_{k+1} = x_k - \eta \nabla f(x_k),$$

где  $\nabla f(x_k)$  — градиент функции в точке  $x_k$ .

- **Остановка:** Алгоритм останавливается, когда градиент становится достаточно малым или достигнуто максимальное число итераций.

**Почему не всегда можно использовать полный градиентный спуск? Какие способы оценивания градиента вы знаете? Почему в стохастическом градиентном спуске важно менять длину шага по мере итераций? Какие стратегии изменения шага вы знаете?**

### Проблемы полного градиентного спуска:

- Вычислительная сложность: на каждой итерации требуется вычисление градиента по всей выборке, что может быть дорого для больших данных.
- Медленная сходимость: в некоторых случаях градиентный спуск может сходиться медленно, особенно если функция имеет "овраги" или "плато".

### Способы оценивания градиента:

- **Стохастический градиентный спуск (SGD):** Градиент вычисляется на основе одного случайного объекта из выборки.
- **Мини-батч градиентный спуск:** Градиент вычисляется на основе небольшой подвыборки (мини-батча).

### Важность изменения длины шага в SGD:

- Если длина шага слишком большая, алгоритм может "перепрыгнуть" минимум.

- Если длина шага слишком маленькая, сходимость будет медленной.
- Поэтому важно адаптивно изменять длину шага по мере итераций.

### Стратегии изменения шага:

- **Постоянный шаг:**  $\eta$  остаётся неизменным на всех итерациях.
- **Уменьшение шага:**  $\eta$  уменьшается по мере итераций, например,  $\eta_k = \frac{\eta_0}{\sqrt{k}}$ .
- **Адаптивные методы:**
  - Adagrad — позволяет сблюстить темп при быстром спуске и наоборот
  - RMSProp — усредняет историю градиентов; лучше чем Adagrad, так как градиент уменьшается не слишком сильно

### Улучшения градиентного спуска:

- **Метод наискорейшего спуска:** выбираем размер шага так, чтобы как можно сильнее уменьшить функцию;  $\alpha_k = \arg \min_{\alpha \geq 0} f(x_k - \alpha \nabla f(x_k))$
- **mini-batch SGD:** случайно выбираем подвыборку и считаем на ней градиент;  $\nabla f(x) = \frac{1}{B} \sum_{i=1}^B \nabla L(x, \xi_i)$
- **Метод инерции (Momentum):** физическая аналогия с шариком, катящимся с горы;  $v_{k+1} := \beta_k v_k - \alpha_k \nabla f(x_k), \quad x_{k+1} := x_k + v_{k+1}$
- **Nesterov Momentum:** считаем градиент не в текущей точке, а в точке, в которую мы бы пошли, следуя импульсу;  $v_{k+1} := \beta_k v_k - \alpha_k \nabla f(x_k + \beta_k v_k), \quad x_{k+1} := x_k + v_{k+1}$

### SoTA градиентные спуски:

- Adam — объединяет в себе Nesterov Momentum и RMSProp
- AdamW — добавим  $l_2$ -регуляризацию

## Что такое переобучение? Как можно отследить переобучение модели?

**Переобучение** — это ситуация, когда модель слишком хорошо обучается на тренировочных данных, но плохо обобщается на новые данные. Это происходит, когда модель "запоминает" шум и выбросы в данных вместо того, чтобы выучить общие закономерности.

### Как отследить переобучение:

- **Разделение данных:** Данные делятся на тренировочную и тестовую выборки. Если модель показывает высокую точность на тренировочных данных, но низкую на тестовых, это признак переобучения.
- **Кросс-валидация:** Используется для оценки обобщающей способности модели. Если модель показывает высокую точность на тренировочных фолдах, но низкую на валидационных, это также признак переобучения.
- **Графики обучения:** СтРОЯтся графики ошибки на тренировочной и валидационной выборках. Если ошибка на тренировочной выборке продолжает уменьшаться, а на валидационной — увеличивается, это признак переобучения.

### Что такое кросс-валидация? На что влияет количество блоков в кросс-валидации?

**Кросс-валидация** — это метод оценки обобщающей способности модели, при котором данные разбиваются на несколько частей (блоков), и модель обучается и тестируется на разных комбинациях этих блоков.

#### Процесс кросс-валидации:

- Данные разбиваются на  $k$  блоков (обычно  $k = 5$  или  $10$ ).
- Модель обучается на  $k - 1$  блоках и тестируется на оставшемся блоке.
- Процесс повторяется  $k$  раз, каждый раз используя разные блоки для тестирования.
- Итоговая оценка модели — это среднее значение метрики качества по всем  $k$  итерациям.

#### Влияние количества блоков:

- **Больше блоков (например,  $k = 10$ ):**
  - Оценка модели становится более точной, так как используется больше данных для обучения.
  - Вычислительная сложность увеличивается, так как требуется больше итераций.
- **Меньше блоков (например,  $k = 5$ ):**
  - Оценка модели может быть менее точной, так как для обучения используется меньше данных.
  - Вычислительная сложность снижается.

#### Виды кросс-валидации:

- **k-Fold Кросс-валидация** — описано выше;

- **Stratified k-Fold Кросс-валидация** — как и прошлая, но в каждом фолде семплы сбалансированы по классам;
- **LOO Кросс-валидация** — обучаемся на всём датасете, один семпл оставляем для теста. Так повторяем для всех семплов (очень времязатратно);
- **Time series Кросс-валидация** — применяется при работе с данными, зависящими от времени. Данный тип кросс-валидации помогает избежать утечки данных из будущего в прошлое, что предоставляет возможность корректной оценки модели на данных с временными метками.

## Как построить итоговую модель после того, как по кросс-валидации подобраны оптимальные гиперпараметры?

После подбора оптимальных гиперпараметров с помощью кросс-валидации итоговая модель строится следующим образом:

- **Обучение на всех данных:** Модель обучается на всём доступном наборе данных (без разделения на блоки) с использованием оптимальных гиперпараметров.
- **Итоговая оценка:** Если требуется оценить качество модели, можно использовать отдельный тестовый набор данных, который не использовался в процессе кросс-валидации.
- **Использование модели:** После обучения модель готова для применения на новых данных.

## Что такое регуляризация? Для чего используется?

**Регуляризация** — это метод, который добавляет штраф к функции потерь модели для предотвращения переобучения. Она используется для:

- Уменьшения сложности модели.
- Улучшения обобщающей способности модели.
- Контроля за значениями параметров модели, чтобы они не становились слишком большими.

**Пример:** В линейной регрессии регуляризация добавляет штраф за большие значения коэффициентов модели.

## Опишите, как работают L1- и L2-регуляризаторы.

### L1-регуляризация (Lasso):

- Добавляет штраф, пропорциональный сумме абсолютных значений коэффициентов модели:

$$L_{\text{L1}} = \lambda \sum_{i=1}^n |w_i|,$$

где  $w_i$  — коэффициенты модели,  $\lambda$  — параметр регуляризации.

- L1-регуляризация может обнулять некоторые коэффициенты, что делает её полезной для отбора признаков.

### L2-регуляризация (Ridge):

- Добавляет штраф, пропорциональный сумме квадратов коэффициентов модели:

$$L_{\text{L2}} = \lambda \sum_{i=1}^n w_i^2.$$

- L2-регуляризация уменьшает значения коэффициентов, но не обнуляет их, что помогает снизить переобучение.

### Различия:

- L1-регуляризация обнуляет некоторые коэффициенты, что полезно для отбора признаков.
- L2-регуляризация уменьшает значения коэффициентов, но не обнуляет их, что делает её более устойчивой к мультиколлинеарности.

## Почему L1-регуляризация отбирает признаки?

L1-регуляризация (Lasso) отбирает признаки благодаря своей способности обнулять коэффициенты модели. Это происходит из-за того, что L1-регуляризация добавляет штраф, пропорциональный сумме абсолютных значений коэффициентов:

$$L_{\text{L1}} = \lambda \sum_{i=1}^n |w_i|,$$

где  $w_i$  — коэффициенты модели, а  $\lambda$  — параметр регуляризации.

### Почему L1 отбирает признаки:

- L1-регуляризация стремится минимизировать сумму абсолютных значений коэффициентов, что приводит к обнулению некоторых из них.
- Это свойство делает L1 полезной для задач отбора признаков, так как она автоматически исключает неважные признаки, обнуляя их коэффициенты.

## Почему плохо накладывать регуляризацию на свободный коэффициент?

Свободный коэффициент (смещение,  $b$ ) в модели отвечает за смещение предсказаний и не зависит от признаков. Накладывать регуляризацию на свободный коэффициент плохо, потому что:

- Регуляризация штрафует большие значения коэффициентов, чтобы уменьшить сложность модели. Однако свободный коэффициент не влияет на сложность модели, так как он не связан с признаками.
- Если наложить регуляризацию на свободный коэффициент, это может привести к смещению модели и ухудшению её предсказательной способности.

## Что такое ElasticNet?

ElasticNet — это метод регуляризации, который сочетает в себе L1-регуляризацию (Lasso) и L2-регуляризацию (Ridge). Он был разработан для устранения некоторых недостатков Lasso и Ridge, особенно в случаях, когда данные имеют:

- мультиколлинеарность (сильную корреляцию между признаками);
- большое количество признаков по сравнению с количеством наблюдений.

### Формула ElasticNet:

$$\text{Функция потерь} = \text{Ошибка модели} + \lambda_1 \sum_{i=1}^n |w_i| + \lambda_2 \sum_{i=1}^n w_i^2$$

где:

- $\lambda_1$  — параметр, контролирующий силу L1-регуляризации (Lasso),
- $\lambda_2$  — параметр, контролирующий силу L2-регуляризации (Ridge),
- $w_i$  — веса модели.

## Когда использовать ElasticNet?

- Когда данные содержат много коррелированных признаков.
- Когда нужно одновременно отбирать признаки (как Lasso) и сохранять устойчивость к мультиколлинеарности (как Ridge).
- Когда количество признаков больше количества наблюдений.

## Где используется метод максимального правдоподобия?

Метод максимального правдоподобия (Maximum Likelihood Estimation, MLE) используется для оценки параметров модели на основе данных. Основные области применения:

- **Логистическая регрессия:** MLE используется для нахождения коэффициентов модели, которые максимизируют вероятность наблюдаемых данных.
- **Линейная регрессия:** При предположении нормального распределения ошибок MLE эквивалентен методу наименьших квадратов.
- **Генеративные модели:** Например, в задачах классификации с использованием гауссовских распределений.
- **Оценка параметров распределений:** Например, для оценки параметров нормального, пуассоновского или биномиального распределений.

## Расскажите про метрики, которые штрафуют за перепрогноз сильнее, чем за недопрогноз и наоборот (pinball loss).

**Pinball Loss** — это функция потерь, которая позволяет штрафовать перепрогнозы и недопрогнозы с разной силой. Она используется в задачах квантильной регрессии.

Формула Pinball Loss:

$$L_\tau(y, \hat{y}) = \begin{cases} \tau \cdot (y - \hat{y}), & \text{если } y > \hat{y}, \\ (1 - \tau) \cdot (\hat{y} - y), & \text{если } y \leq \hat{y}, \end{cases}$$

где:

- $y$  — истинное значение,
- $\hat{y}$  — предсказанное значение,
- $\tau$  — параметр, определяющий, насколько сильно штрафуется перепрогноз по сравнению с недопрогнозом.

### Особенности Pinball Loss:

- Если  $\tau > 0.5$ , перепрогнозы штрафуются сильнее, чем недопрогнозы.
- Если  $\tau < 0.5$ , недопрогнозы штрафуются сильнее, чем перепрогнозы.
- Если  $\tau = 0.5$ , Pinball Loss эквивалентна MAE (Mean Absolute Error).

### Пример использования:

- В задачах прогнозирования, где перепрогнозы и недопрогнозы имеют разную стоимость (например, в логистике или финансах).

## Расскажите про виды скейлинга. Зачем они нужны?

**Скейлинг** (масштабирование) — это процесс приведения признаков к одинаковому масштабу. Это важно, потому что многие алгоритмы машинного обучения чувствительны к масштабу данных (например, методы, основанные на расстояниях или градиентных спусках).

### Основные виды скейлинга:

- **Минимаксное масштабирование (Min-Max Scaling):**

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)},$$

где  $x$  — исходное значение,  $\min(X)$  и  $\max(X)$  — минимальное и максимальное значения признака. Приводит значения к диапазону  $[0, 1]$ .

- **Стандартизация (Z-score Normalization):**

$$x' = \frac{x - \mu}{\sigma},$$

где  $\mu$  — среднее значение признака,  $\sigma$  — стандартное отклонение. Приводит данные к распределению с нулевым средним и единичной дисперсией.

- **Масштабирование до единичной длины (Normalization):**

$$x' = \frac{x}{\|X\|},$$

где  $\|X\|$  — норма вектора признаков. Используется, например, в текстовых данных для приведения векторов к единичной длине.

- **Robust Scaling:**

$$x' = \frac{x - \text{медиана}(X)}{\text{IQR}(X)},$$

где  $\text{IQR}(X)$  — межквартильный размах. Устойчив к выбросам.

### Зачем нужен скейлинг:

- Улучшает сходимость градиентных методов оптимизации.
- Уравнивает вклад признаков в модели, основанные на расстояниях (например, k-ближайших соседей, SVM).
- Помогает избежать доминирования признаков с большим масштабом.

## Как записываются аналитические решения? Какие у них проблемы?

**Аналитические решения** — это решения, которые можно выразить в виде явной формулы, без необходимости итеративных вычислений. Например, в линейной регрессии аналитическое решение для нахождения коэффициентов модели записывается как:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y},$$

где:

- $\mathbf{X}$  — матрица признаков,
- $\mathbf{y}$  — вектор целевых значений,
- $\mathbf{w}$  — вектор коэффициентов модели.

### Проблемы аналитических решений:

- **Вычислительная сложность:** Для больших матриц  $\mathbf{X}$  вычисление обратной матрицы  $(\mathbf{X}^T \mathbf{X})^{-1}$  может быть очень затратным.
- **Чувствительность к мультиколлинеарности:** Если матрица  $\mathbf{X}^T \mathbf{X}$  близка к вырожденной, обратная матрица может быть нестабильной или не существовать.
- **Ограниченнная применимость:** Аналитические решения существуют только для определённых типов моделей (например, линейная регрессия). Для более сложных моделей (например, нейронных сетей) аналитические решения недоступны.
- **Проблемы с большими данными:** Для очень больших наборов данных аналитические решения могут быть неприменимы из-за ограничений памяти и вычислительной мощности.

## 2. Классификация

Запишите формулу для линейной модели классификации. Что такое отступ?

**Линейная модель классификации** предсказывает класс объекта на основе линейной комбинации признаков. Формула линейной модели:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b,$$

где:  $\mathbf{x}$  — вектор признаков объекта,  $\mathbf{w}$  — вектор весов модели,  $b$  — свободный коэффициент (смещение).

**Отступ (margin)** — это величина, которая показывает, насколько уверенно модель классифицирует объект. Для объекта  $(\mathbf{x}, y)$ , где  $y \in \{-1, 1\}$ , отступ определяется как:

$$\text{margin} = y \cdot f(\mathbf{x}).$$

- Если отступ положительный, объект классифицирован правильно.
- Если отступ отрицательный, объект классифицирован неправильно.
- Чем больше отступ, тем увереннее классификация.

## Как обучаются линейные классификаторы и для чего нужны верхние оценки пороговой функции потерь?

**Обучение линейных классификаторов** заключается в нахождении таких весов  $\mathbf{w}$  и смещения  $b$ , которые минимизируют функцию потерь. Например, в методе опорных векторов (SVM) минимизируется функция потерь с регуляризацией:

$$L(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)),$$

где  $C$  — параметр регуляризации.

**Верхние оценки пороговой функции потерь** используются для упрощения оптимизации. Например, вместо минимизации пороговой функции потерь (которая не дифференцируема) минимизируют её верхнюю оценку (например, hinge loss в SVM), что позволяет использовать градиентные методы.

## Что такое точность, полнота и F-мера? Почему F-мера лучше арифметического среднего и минимума?

**Accuracy** — показывает долю правильно предсказанных меток классов относительно общего количества предсказаний.

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

Обладает следующим недостатком: если один класс значительно преобладает над другим, Accuracy может вводить в заблуждение. Кроме того, Accuracy игнорирует ошибки разного типа (не различает FP и FN).

**Точность (Precision)** — это доля правильно предсказанных положительных классов среди всех предсказанных положительных классов:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

где:

- TP (True Positive) — количество верно предсказанных положительных классов,
- FP (False Positive) — количество неверно предсказанных положительных классов.

**Полнота (Recall)** — это доля правильно предсказанных положительных классов среди всех истинных положительных классов:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

где:

- FN (False Negative) — количество неверно предсказанных отрицательных классов.

**F-мера (F1-score)** — это гармоническое среднее точности и полноты:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

**Почему F-мера лучше арифметического среднего и минимума:**

- F-мера учитывает как точность, так и полноту, что делает её более сбалансированной метрикой.
- Арифметическое среднее может быть высоким даже при низком значении одной из метрик (точности или полноты).
- Минимум не учитывает баланс между точностью и полнотой.

**Для чего нужен порог в линейном классификаторе? Из каких соображений он может выбираться?**

Порог в линейном классификаторе используется для преобразования непрерывного выхода модели  $f(\mathbf{x})$  в дискретные классы. Например, если  $f(\mathbf{x}) \geq 0$ , объект классифицируется как положительный, иначе — как отрицательный.

**Выбор порога:**

- **По умолчанию:** Порог равен 0, если модель откалибрована.
- **На основе метрик:** Порог можно выбрать для максимизации определённой метрики (например, F1-score или точности).
- **На основе бизнес-требований:** Например, если ложные положительные результаты дороже ложных отрицательных, порог можно увеличить, чтобы уменьшить FP.

## Что такое AUC-ROC? Опишите алгоритм построения ROC-кривой.

**AUC-ROC (Area Under the ROC Curve)** — это метрика, которая оценивает качество классификатора по площади под ROC-кривой. ROC-кривая показывает зависимость между долей истинных положительных результатов (True Positive Rate, TPR) и долей ложных положительных результатов (False Positive Rate, FPR) при изменении порога классификации.

### Алгоритм построения ROC-кривой:

- Упорядочить объекты по убыванию предсказанных вероятностей положительного класса.
- Для каждого возможного порога:

- Вычислить TPR и FPR:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

- Добавить точку (FPR, TPR) на график.
- Построить кривую, соединив все точки.
- Вычислить площадь под кривой (AUC-ROC).

### Интерпретация AUC-ROC:

- AUC = 1: Идеальный классификатор.
- AUC = 0.5: Случайный классификатор.
- AUC < 0.5: Классификатор работает хуже случайного.

## Что такое AUC-PRC? Опишите алгоритм построения PR-кривой.

**AUC-PRC (Area Under the Precision-Recall Curve)** — это метрика, которая оценивает качество классификатора по площади под PR-кривой. PR-кривая показывает зависимость между точностью (Precision) и полнотой (Recall) при изменении порога классификации.

### Алгоритм построения PR-кривой:

- Упорядочить объекты по убыванию предсказанных вероятностей положительного класса.
- Для каждого возможного порога:

- Вычислить Precision и Recall:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

- Добавить точку (Recall, Precision) на график.
- Построить кривую, соединив все точки.
- Вычислить площадь под кривой (AUC-PRC).

### Интерпретация AUC-PRC:

- AUC-PRC = 1: Идеальный классификатор.
- AUC-PRC = 0: Классификатор не работает.
- AUC-PRC особенно полезна в задачах с дисбалансом классов, так как она учитывает Precision и Recall.

**Что означает “модель оценивает вероятность положительного класса”?**

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

“Модель оценивает вероятность положительного класса” означает, что модель предсказывает не только класс объекта, но и вероятность принадлежности к положительному классу. Например, в задаче бинарной классификации модель может предсказать вероятность  $P(y = 1|\mathbf{x})$ .

**Как внедрить это требование:**

- **Использование вероятностных моделей:** Например, логистическая регрессия или метод опорных векторов с калибровкой вероятностей.
- **Калибровка вероятностей:** Если модель не возвращает вероятности, можно использовать методы калибровки, такие как:
  - Platt Scaling: Использует логистическую регрессию для калибровки выходов модели.
  - Isotonic Regression: Использует непараметрическую регрессию для калибровки.
- **Обучение с учётом вероятностей:** Например, в логистической регрессии минимизируется функция потерь, которая учитывает вероятности:

$$L(\mathbf{w}, b) = - \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)],$$

где  $p_i = P(y_i = 1|\mathbf{x}_i)$ .

## Запишите функционал логистической регрессии. Как он связан с методом максимума правдоподобия?

Функционал логистической регрессии — это функция потерь, которая минимизируется в процессе обучения модели. Для бинарной классификации она записывается как:

$$L(\mathbf{w}, b) = - \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)],$$

где:

- $y_i$  — истинный класс объекта (0 или 1),
- $p_i = \sigma(\mathbf{w}^T \mathbf{x}_i + b)$  — предсказанная вероятность положительного класса,
- $\sigma(z) = \frac{1}{1+e^{-z}}$  — сигмоидная функция.

### Связь с методом максимума правдоподобия:

- Логистическая регрессия максимизирует правдоподобие данных при заданных параметрах модели.
- Функция потерь  $L(\mathbf{w}, b)$  — это отрицательное логарифмическое правдоподобие (Negative Log-Likelihood, NLL).
- Минимизация  $L(\mathbf{w}, b)$  эквивалентна максимизации правдоподобия.

## Когда используется accuracy?

**Accuracy (точность)** — это метрика, которая показывает долю правильно классифицированных объектов среди всех объектов:

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

где:

- TP (True Positive) — количество верно предсказанных положительных классов,
- TN (True Negative) — количество верно предсказанных отрицательных классов,
- FP (False Positive) — количество неверно предсказанных положительных классов,
- FN (False Negative) — количество неверно предсказанных отрицательных классов.

### Когда использовать accuracy:

- Когда классы сбалансираны (количество объектов в каждом классе примерно одинаково).
- Когда ложные положительные и ложные отрицательные ошибки имеют одинаковую важность.
- В задачах, где важно общее количество правильных предсказаний.

### Ограничения accuracy:

- В задачах с дисбалансом классов accuracy может быть misleading (например, если 95% объектов принадлежат к одному классу, accuracy будет высокой даже при плохой классификации меньшего класса).

## Как бороться с дисбалансом классов?

**Дисбаланс классов** — это ситуация, когда один класс значительно преобладает над другим. Это может привести к тому, что модель будет плохо предсказывать меньший класс. Методы борьбы с дисбалансом классов:

- **Методы на уровне данных:**
  - **Увеличение меньшего класса (oversampling):** Например, с помощью SMOTE (Synthetic Minority Over-sampling Technique), который создаёт синтетические объекты меньшего класса.
  - **Уменьшение большего класса (undersampling):** Удаление случайных объектов из большего класса.
- **Методы на уровне алгоритма:**
  - **Взвешивание классов:** Присвоение больших штрафов за ошибки на объектах меньшего класса.
  - **Использование метрик, устойчивых к дисбалансу:** Например, F1-score, AUC-PRC.
- **Ансамблевые методы:**
  - **Использование ансамблей:** Например, Random Forest или Gradient Boosting, которые могут быть настроены для работы с дисбалансом.
- **Генерация синтетических данных:**
  - **SMOTE:** Создание синтетических объектов меньшего класса на основе существующих.

- **Использование порогов:**

- **Настройка порога классификации:** Изменение порога для увеличения Recall или Precision в зависимости от задачи.

### 3. Многоклассовая классификация

#### Как измеряется качество в задаче многоклассовой классификации?

Качество в задаче многоклассовой классификации измеряется с помощью различных метрик, которые учитывают правильность предсказаний для каждого класса. Основные метрики:

- **Accuracy (точность):** Доля правильно классифицированных объектов среди всех объектов:

$$\text{Accuracy} = \frac{\sum_{i=1}^C \text{TP}_i}{\sum_{i=1}^C (\text{TP}_i + \text{FP}_i)},$$

где  $\text{TP}_i$  — количество верно предсказанных объектов для класса  $i$ ,  $\text{FP}_i$  — количество неверно предсказанных объектов для класса  $i$ .

- **Precision, Recall, F1-score:** Эти метрики могут быть рассчитаны для каждого класса отдельно, а затем усреднены (микро- или макро-усреднение).
- **Confusion Matrix:** Матрица, которая показывает, сколько объектов каждого класса было классифицировано правильно и неправильно.
- **AUC-ROC:** Может быть рассчитана для каждого класса отдельно (one-vs-rest) или усреднена.

#### Расскажите микро и макро-усреднение?

##### Микро-усреднение (Micro-averaging):

- Все классы рассматриваются как один большой класс.
- Метрики (например, Precision, Recall, F1-score) вычисляются на основе общего количества TP, FP, FN для всех классов.
- Формула для микро-усреднённого Precision:

$$\text{Micro-Precision} = \frac{\sum_{i=1}^C \text{TP}_i}{\sum_{i=1}^C (\text{TP}_i + \text{FP}_i)}.$$

- Подходит для случаев, когда важно учитывать вклад каждого объекта, независимо от класса.

## Макро-усреднение (Macro-averaging):

- Метрики вычисляются для каждого класса отдельно, а затем усредняются.
- Формула для макро-усреднённого Precision:

$$\text{Macro-Precision} = \frac{1}{C} \sum_{i=1}^C \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}.$$

- Подходит для случаев, когда все классы равнозначны, и важно учитывать качество классификации для каждого класса.

## Различия:

- Микро-усреднение больше зависит от крупных классов, так как они вносят больший вклад в общие TP, FP, FN.
- Макро-усреднение даёт равный вес каждому классу, независимо от его размера.

## Что такое mean-target encoding?

**Mean-target encoding** — это метод кодирования категориальных признаков, при котором каждое значение категории заменяется средним значением целевой переменной для этой категории. Например, для бинарной классификации:

$$\text{Encoded value} = \frac{\sum_{i=1}^N y_i \cdot \mathbb{I}(x_i = \text{category})}{\sum_{i=1}^N \mathbb{I}(x_i = \text{category})},$$

где:

- $y_i$  — целевая переменная для объекта  $i$ ,
- $x_i$  — значение категориального признака для объекта  $i$ ,
- $\mathbb{I}(\cdot)$  — индикаторная функция, равная 1, если условие выполняется, и 0 иначе.

## Преимущества:

- Учитывает взаимосвязь между категориальным признаком и целевой переменной.
- Может улучшить качество модели, особенно для деревьев и ансамблей.

## Может ли mean-target encoding привести к переобучению? Как можно этого избежать?

Да, mean-target encoding может привести к переобучению, так как при кодировании используется информация о целевой переменной, что может "просачиваться" в обучающие данные.

### Как избежать переобучения:

- **Использование кросс-валидации:** Разделить данные на фолды, и для каждого фолда вычислять mean-target encoding только на основе данных из других фолдов.
- **Добавление шума:** Добавить небольшой случайный шум к закодированным значениям, чтобы уменьшить их зависимость от целевой переменной.
- **Регуляризация:** Использовать сглаживание (smoothing), например, добавляя глобальное среднее значение целевой переменной:

$$\text{Encoded value} = \frac{\sum_{i=1}^N y_i \cdot \mathbb{I}(x_i = \text{category}) + \alpha \cdot \text{global mean}}{\sum_{i=1}^N \mathbb{I}(x_i = \text{category}) + \alpha},$$

где  $\alpha$  — параметр сглаживания.

## Как можно отбирать признаки для линейной модели?

Отбор признаков для линейной модели может быть выполнен с помощью следующих методов:

- **Фильтры:** Использование статистических метрик (например, корреляция с целевой переменной, взаимная информация) для выбора наиболее информативных признаков.
- **Обёртки (Wrapper methods):** Поиск оптимального набора признаков с помощью алгоритмов, таких как Forward Selection, Backward Elimination или Recursive Feature Elimination (RFE).
- **Встроенные методы (Embedded methods):** Использование моделей, которые автоматически отбирают признаки в процессе обучения, например, L1-регуляризация (Lasso), которая обнуляет неважные коэффициенты.
- **Анализ важности признаков:** Использование методов, таких как SHAP или Feature Importance из ансамблевых моделей (например, Random Forest), для оценки вклада каждого признака.

## Что такое решающее дерево?

**Решающее дерево** — это модель машинного обучения, которая строит иерархическую структуру решений, разделяя данные на подмножества на основе значений признаков. Каждый внутренний

узел дерева представляет собой условие (предикат) на основе одного из признаков, а каждый лист — предсказание (класс или значение).

#### Основные компоненты:

- **Внутренние узлы:** Условия на основе признаков (например,  $x_1 > 5$ ).
- **Листья:** Конечные узлы, содержащие предсказания.
- **Ветви:** Результаты выполнения условий (например, "да"или "нет").

#### Опишите жадный алгоритм обучения решающего дерева.

**Жадный алгоритм обучения решающего дерева** строит дерево, на каждом шаге выбирая наилучшее разбиение данных на основе критерия качества (например, критерия хаотичности). Процесс:

- **Выбор предиката:** Для каждого признака и каждого возможного значения предиката вычисляется качество разбиения (например, уменьшение энтропии или критерия Джини).
- **Разбиение данных:** Данные разделяются на подмножества в соответствии с выбранным предикатом.
- **Рекурсия:** Процесс повторяется для каждого подмножества, пока не будет выполнено условие остановки (например, достижение максимальной глубины дерева или минимального количества объектов в листе).

#### Почему с помощью бинарного решающего дерева можно достичь нулевой ошибки на обучающей выборке без повторяющихся объектов?

**Бинарное решающее дерево** может достичь нулевой ошибки на обучающей выборке без повторяющихся объектов, потому что:

- Каждый объект обучающей выборки уникален и может быть отделён от других объектов с помощью последовательных разбиений.
- Дерево может продолжать разбивать данные до тех пор, пока каждый объект не окажется в своём собственном листе.
- Это приводит к тому, что дерево "запоминает"обучающие данные, что может привести к переобучению.

## Что такое критерий хаотичности? Как он используется для выбора предиката во внутренней вершине решающего дерева?

**Критерий хаотичности** — это мера, которая оценивает, насколько хорошо разбиение данных разделяет классы или уменьшает неопределённость. Основные критерии:

- **Энтропия:** Мера неопределенности распределения классов:

$$\text{Entropy}(S) = - \sum_{i=1}^C p_i \log_2(p_i),$$

где  $p_i$  — доля объектов класса  $i$  в множестве  $S$ .

- **Критерий Джини:** Мера неоднородности:

$$\text{Gini}(S) = 1 - \sum_{i=1}^C p_i^2.$$

### Использование для выбора предиката:

- Для каждого возможного предиката вычисляется уменьшение хаотичности (Information Gain):

$$\text{Information Gain} = \text{Entropy}(S) - \sum_{v \in \text{values}} \frac{|S_v|}{|S|} \text{Entropy}(S_v),$$

где  $S_v$  — подмножество данных, для которых предикат принимает значение  $v$ .

- Выбирается предикат, который максимизирует Information Gain.

## В чем отличия энтропийного критерия и критерия Джини?

**Энтропийный критерий и критерий Джини** используются для оценки качества разбиения в решающих деревьях. Их основные отличия:

- **Энтропия:**

- Основан на теории информации.
- Вычисляется как  $\text{Entropy}(S) = - \sum_{i=1}^C p_i \log_2(p_i)$ .
- Более чувствителен к изменениям в распределении классов.

- **Критерий Джини:**

- Основан на мере неоднородности.

- Вычисляется как  $\text{Gini}(S) = 1 - \sum_{i=1}^C p_i^2$ .
- Менее чувствителен к изменениям в распределении классов.
- Вычислительно проще, чем энтропия.

## Выбор критерия:

- Оба критерия обычно дают схожие результаты.
- Критерий Джини чаще используется на практике из-за его вычислительной простоты.

## Как связаны линейные модели и решающие деревья?

Линейные модели и решающие деревья — это два разных подхода к моделированию данных, но они могут быть связаны следующим образом:

- **Линейные модели:** Предполагают, что зависимость между признаками и целевой переменной линейна. Они хорошо работают, когда данные имеют линейную структуру.
- **Решающие деревья:** Могут моделировать нелинейные зависимости, разделяя данные на подмножества на основе значений признаков.
- **Комбинация:** Линейные модели и решающие деревья могут быть объединены в ансамблевые методы, такие как Gradient Boosting, где деревья используются для аппроксимации остатков линейной модели.
- **Интерпретируемость:** Линейные модели легко интерпретируются, но могут быть ограничены в сложных данных. Решающие деревья менее интерпретируются, но могут моделировать сложные зависимости.

## Как посчитать хаотичность вершины в задаче классификации? А в задаче регрессии?

**Хаотичность вершины** — это мера неопределённости или неоднородности данных в вершине дерева. Она используется для выбора наилучшего разбиения.

### В задаче классификации:

- **Энтропия:**

$$\text{Entropy}(S) = - \sum_{i=1}^C p_i \log_2(p_i),$$

где  $p_i$  — доля объектов класса  $i$  в множестве  $S$ .

- **Критерий Джини:**

$$\text{Gini}(S) = 1 - \sum_{i=1}^C p_i^2.$$

### В задаче регрессии:

- **Дисперсия:** Хаотичность вершины измеряется как дисперсия значений целевой переменной:

$$\text{Variance}(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} (y_i - \bar{y})^2,$$

где  $y_i$  — значение целевой переменной для объекта  $i$ ,  $\bar{y}$  — среднее значение целевой переменной в множестве  $S$ .

## В чем заключается метод опорных векторов?

**Метод опорных векторов (SVM)** — это алгоритм классификации и регрессии, который находит гиперплоскость, максимально разделяющую классы в пространстве признаков.

### Основные идеи:

- **Гиперплоскость:** SVM ищет гиперплоскость, которая максимизирует зазор (margin) между классами.
- **Опорные векторы:** Это объекты, которые находятся ближе всего к гиперплоскости и определяют её положение.
- **Ядра (Kernels):** SVM может использовать ядра для преобразования данных в пространство более высокой размерности, где классы могут быть линейно разделимыми.

### Формула для линейного SVM:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)),$$

где:

- $\mathbf{w}$  — вектор весов,
- $b$  — смещение,
- $C$  — параметр регуляризации,
- $y_i$  — метка класса ( $\pm 1$ ),
- $\mathbf{x}_i$  — вектор признаков.

## В чем заключается метод k-ближайших соседей?

Метод **k-ближайших соседей** (**k-NN**) — это алгоритм классификации и регрессии, который предсказывает значение целевой переменной на основе значений ближайших объектов в пространстве признаков.

### Основные идеи:

- **Расстояние:** Для каждого объекта вычисляется расстояние до всех объектов в обучающей выборке (например, евклидово расстояние).
- **Выбор k соседей:** Выбираются  $k$  объектов с наименьшим расстоянием.
- **Предсказание:**
  - В задаче классификации: Предсказанный класс — это наиболее часто встречающийся класс среди  $k$  соседей.
  - В задаче регрессии: Предсказанное значение — это среднее значение целевой переменной среди  $k$  соседей.

### Параметры:

- $k$ : Количество ближайших соседей.
- Метрика расстояния: Например, евклидово расстояние, манхэттенское расстояние.

### Преимущества:

- Простота реализации.
- Не требует обучения (ленивый алгоритм).

### Недостатки:

- Вычислительная сложность на больших данных.
- Чувствительность к масштабу данных и выбросам.

## Нужно ли заниматься предобработкой данных в случае дерева?

**Решающие деревья** менее чувствительны к предобработке данных по сравнению с другими моделями, но некоторые шаги всё же могут быть полезны:

- **Масштабирование:** Деревья не требуют масштабирования признаков, так как они работают с пороговыми значениями.
- **Обработка пропущенных значений:** Деревья могут работать с пропущенными значениями ( $\text{NaN}$ ), но их обработка может улучшить качество модели.
- **Кодирование категориальных признаков:** Деревья могут работать с категориальными признаками, но их кодирование (например, One-Hot Encoding) может упростить интерпретацию.
- **Удаление выбросов:** Деревья устойчивы к выбросам, но их удаление может улучшить качество модели.

## Как деревья работают с $\text{NaN}$ ?

Решающие деревья могут работать с пропущенными значениями ( $\text{NaN}$ ) следующим образом:

- **Разделение на подмножества:** Если значение признака отсутствует, дерево может разделить объекты на два подмножества: одно для объектов с известным значением, другое — для объектов с  $\text{NaN}$ .
- **Использование суррогатных разбиений:** Если основной признак для разбиения содержит  $\text{NaN}$ , дерево может использовать другой признак (суррогатный) для принятия решения.
- **Пропуск объектов:** В некоторых реализациях объекты с  $\text{NaN}$  могут быть пропущены при построении дерева.

## Как деревья работают с категориальными значениями?

Решающие деревья могут работать с категориальными значениями следующим образом:

- **Бинарное разбиение:** Для каждого категориального признака дерево может разделить объекты на две группы: одна группа содержит объекты с определённым значением категории, другая — все остальные.
- **Многозначное разбиение:** Некоторые реализации деревьев поддерживают многозначное разбиение, где каждая категория становится отдельной ветвью.
- **Кодирование:** Категориальные признаки могут быть закодированы (например, One-Hot Encoding), чтобы упростить процесс разбиения.

## Как сделать многоклассовую классификацию через логрег?

**Логистическая регрессия (Logistic Regression, LogReg)** изначально предназначена для бинарной классификации, но её можно расширить для многоклассовой классификации с помощью следующих подходов:

- **One-vs-Rest (OvR):**

- Для каждого класса строится отдельная бинарная модель, которая предсказывает вероятность принадлежности к этому классу.
- Объект классифицируется в класс с наибольшей предсказанной вероятностью.

- **One-vs-One (OvO):**

- Для каждой пары классов строится отдельная бинарная модель.
- Объект классифицируется в класс, который выигрывает большинство "голосов" от всех моделей.

- **Multinomial Logistic Regression:**

- Используется одна модель, которая предсказывает вероятности для всех классов одновременно.
- Функция потерь — это кросс-энтропия для многоклассовой классификации:

$$L(\mathbf{W}) = - \sum_{i=1}^N \sum_{j=1}^C y_{ij} \log(p_{ij}),$$

где  $y_{ij}$  — индикатор принадлежности объекта  $i$  к классу  $j$ ,  $p_{ij}$  — предсказанная вероятность.

## 4. Леса

**Приведите пример семейства алгоритмов с низким смещением и большим разбросом.**

**Алгоритмы с низким смещением и большим разбросом** — это алгоритмы, которые хорошо обучаются на тренировочных данных, но могут плохо обобщаться на новые данные. Примеры:

- **Решающие деревья:** Могут достигать нулевой ошибки на обучающей выборке, но сильно зависят от конкретных данных, что приводит к высокому разбросу.

- **Метод k-ближайших соседей (k-NN) с малым  $k$ :** При малом  $k$  модель хорошо подстраивается под данные, но чувствительна к шуму и выбросам.
- **Нейронные сети с большим количеством параметров:** Могут переобучаться на тренировочных данных, что приводит к высокому разбросу.

**Приведите пример семейства алгоритмов с большим смещением и низким разбросом.**

**Алгоритмы с большим смещением и низким разбросом** — это алгоритмы, которые плохо обучаются на тренировочных данных, но стабильно работают на новых данных. Примеры:

- **Линейная регрессия:** Предполагает линейную зависимость между признаками и целевой переменной, что может быть слишком упрощённым для сложных данных.
- **Наивный байесовский классификатор:** Делает сильные предположения о независимости признаков, что может привести к высокому смещению.
- **Метод k-ближайших соседей (k-NN) с большим  $k$ :** При большом  $k$  модель становится менее чувствительной к шуму, но может пропускать сложные зависимости в данных.

**Что такое бэггинг? Как его смещение и разброс связаны со смещением и разбросом базовых моделей?**

Бэггинг (**Bootstrap Aggregating**) — это метод ансамблевого обучения, который объединяет предсказания нескольких моделей, обученных на разных подвыборках данных. Основные шаги:

- **Создание подвыборок:** Из исходной выборки создаются несколько подвыборок с повторениями (bootstrap samples).
- **Обучение моделей:** На каждой подвыборке обучается отдельная модель.
- **Агрегация:** Предсказания моделей объединяются (например, усредняются для регрессии или голосованием для классификации).

**Смещение и разброс бэггинга:**

- **Смещение:** Бэггинг сохраняет смещение базовых моделей. Если базовые модели имеют высокое смещение, бэггинг также будет иметь высокое смещение.
- **Разброс:** Бэггинг уменьшает разброс, так как усреднение предсказаний снижает влияние случайных ошибок отдельных моделей.

## Что такое случайный лес? Чем он отличается от бэггинга над решающими деревьями?

**Случайный лес (Random Forest)** — это метод ансамблевого обучения, который объединяет предсказания множества решающих деревьев, обученных на разных подвыборках данных и с разными подмножествами признаков.

### Основные отличия от бэггинга над решающими деревьями:

- **Случайный выбор признаков:** В случайном лесе на каждом шаге построения дерева выбирается случайное подмножество признаков для разбиения. Это уменьшает корреляцию между деревьями и улучшает обобщающую способность.
- **Разнообразие моделей:** Случайный лес создаёт более разнообразные деревья, чем бэггинг, что помогает снизить переобучение.
- **Уменьшение разброса:** Случайный лес дополнительно уменьшает разброс по сравнению с бэггингом, так как деревья менее коррелированы.

### Преимущества случайного леса:

- Устойчивость к переобучению.
- Возможность работы с пропущенными значениями и категориальными признаками.
- Высокая интерпретируемость (например, через важность признаков).

### Перечислите основные плюсы решающего леса.

**Решающий лес (Random Forest)** имеет следующие преимущества:

- **Устойчивость к переобучению:** За счёт усреднения предсказаний множества деревьев, случайный лес менее склонен к переобучению, чем отдельные деревья.
- **Высокая точность:** Случайный лес часто показывает высокую точность на различных задачах классификации и регрессии.
- **Работа с пропущенными значениями:** Случайный лес может обрабатывать пропущенные значения без необходимости их предварительной обработки.
- **Работа с категориальными признаками:** Случайный лес может работать с категориальными признаками без необходимости их кодирования.
- **Интерпретируемость:** Случайный лес позволяет оценивать важность признаков, что помогает в анализе данных.

- **Параллелизация:** Обучение деревьев в случайному лесу может быть легко распараллелено, что ускоряет процесс обучения.

## Назовите недостатки случайногого леса.

Случайный лес (Random Forest) имеет следующие недостатки:

- **Вычислительная сложность:** Обучение большого количества деревьев может быть computationally expensive, особенно на больших данных.
- **Память:** Случайный лес требует больше памяти для хранения множества деревьев.
- **Меньшая интерпретируемость:** По сравнению с одним деревом, случайный лес сложнее интерпретировать из-за большого количества деревьев.
- **Чувствительность к шуму:** Хотя случайный лес устойчив к переобучению, он может быть чувствителен к шуму в данных, особенно если деревья глубокие.

## Как работает алгоритм градиентного бустинга.

Градиентный бустинг (Gradient Boosting) — это метод ансамблевого обучения, который строит последовательность моделей, каждая из которых корректирует ошибки предыдущих моделей. Основные шаги:

- **Инициализация:** Начальное предсказание — это константа (например, среднее значение целевой переменной для регрессии).
- **Итерации:** На каждой итерации:
  - Вычисляются остатки (разница между истинными значениями и предсказаниями текущей модели).
  - Обучается новая модель (обычно решающее дерево) на этих остатках.
  - Предсказания новой модели добавляются к текущей модели с некоторым коэффициентом обучения (learning rate).
- **Остановка:** Процесс повторяется до достижения заданного числа итераций или пока ошибка не перестанет уменьшаться.

## Как обычно выглядят базовые модели в бустинге. Опишите почему?

**Базовые модели в бустинге** обычно представляют собой слабые модели, такие как решающие деревья небольшой глубины (например, деревья с 3–6 уровнями). Причины:

- **Слабость моделей:** Бустинг предполагает, что каждая модель исправляет ошибки предыдущих моделей. Слабые модели (например, мелкие деревья) не переобучаются и вносят небольшие корректировки.
- **Устойчивость к переобучению:** Мелкие деревья менее склонны к переобучению, что помогает улучшить обобщающую способность ансамбля.
- **Вычислительная эффективность:** Мелкие деревья быстрее обучаются и требуют меньше памяти.

## Что такое сдвиги в градиентном бустинге, зачем они нужны?

**Сдвиги (Shifts)** в градиентном бустинге — это изменения в распределении данных, которые могут возникать между обучающей и тестовой выборками или между разными итерациями бустинга. Они могут быть вызваны, например, изменением статистических свойств данных или наличием выбросов.

### Зачем нужны сдвиги:

- **Устойчивость к изменениям:** Градиентный бустинг должен быть устойчив к сдвигам в данных, чтобы сохранять высокую точность на новых данных.
- **Коррекция ошибок:** Сдвиги могут привести к ошибкам в предсказаниях, и бустинг должен корректировать эти ошибки на последующих итерациях.
- **Обобщающая способность:** Учёт сдвигов помогает улучшить обобщающую способность модели, особенно в реальных задачах, где данные могут меняться со временем.

### Как бороться с сдвигами:

- Использование регуляризации (например, L1 или L2).
- Ограничение глубины деревьев.
- Использование методов, таких как AdaBoost, которые адаптивно изменяют веса объектов в зависимости от ошибок.

## Как обучается очередной базовый алгоритм в градиентном бустинге?

**Очередной базовый алгоритм в градиентном бустинге** обучается на остатках (ошибках) предыдущих моделей. Процесс:

- **Вычисление остатков:** На каждом шаге вычисляются остатки (разница между истинными значениями и предсказаниями текущей модели):

$$r_i = y_i - F_{m-1}(x_i),$$

где  $y_i$  — истинное значение,  $F_{m-1}(x_i)$  — предсказание текущей модели.

- **Обучение новой модели:** Новая модель  $h_m(x)$  обучается на остатках  $r_i$ . Обычно это решающее дерево.
- **Обновление модели:** Предсказания новой модели добавляются к текущей модели с коэффициентом обучения  $\eta$ :

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x).$$

**Расскажите про виды бустинга (CatBoost, LightGBM, XGBoost), их особенности и различия.**

**Основные виды бустинга:**

- **XGBoost (eXtreme Gradient Boosting):**

- Высокая производительность и точность.
- Поддержка параллельных вычислений.
- Встроенная регуляризация для предотвращения переобучения.
- Поддержка различных функций потерь и метрик.

- **LightGBM (Light Gradient Boosting Machine):**

- Оптимизирован для работы с большими данными.
- Использует Gradient-based One-Side Sampling (GOSS) и Exclusive Feature Bundling (EFB) для ускорения обучения.
- Поддержка категориальных признаков без предварительного кодирования.

- **CatBoost (Categorical Boosting):**

- Специализирован на работе с категориальными признаками.
- Использует Ordered Boosting для борьбы с переобучением.
- Автоматически обрабатывает категориальные признаки и пропущенные значения.

- Высокая точность и устойчивость к переобучению.

### Основные различия:

- **Обработка категориальных признаков:** CatBoost лучше всего подходит для задач с большим количеством категориальных признаков.
- **Скорость обучения:** LightGBM обычно быстрее, чем XGBoost и CatBoost, особенно на больших данных.
- **Точность:** XGBoost и CatBoost часто показывают более высокую точность, но требуют больше времени на обучение.

### Почему дерево строится по MSE (идея, что связано с косинусным расстоянием)?

Дерево строится по MSE (Mean Squared Error) в задачах регрессии, потому что:

- **MSE минимизирует квадратичную ошибку:** Это соответствует минимизации евклидова расстояния между предсказанными и истинными значениями.
- **Связь с косинусным расстоянием:** Минимизация MSE эквивалентна минимизации угла между вектором предсказаний и вектором истинных значений в пространстве признаков, что связано с косинусным расстоянием.
- **Простота вычислений:** MSE легко дифференцировать, что делает его удобным для градиентных методов оптимизации.

### Как работает ОOB (Out-of-Bag)?

OOB (Out-of-Bag) — это метод оценки качества модели в случайному лесе без использования отдельной тестовой выборки. Процесс:

- **Создание подвыборок:** Каждое дерево в случайному лесе обучается на bootstrap-подвыборке данных.
- **OOB-объекты:** Объекты, не вошедшие в bootstrap-подвыборку для конкретного дерева, называются OOB-объектами.
- **Оценка качества:** Для каждого OOB-объекта используется предсказание деревьев, которые его не видели. Ошибка на OOB-объектах даёт оценку качества модели.

### Преимущества ОOB:

- Не требует отдельной тестовой выборки.
- Даёт несмешённую оценку качества модели.

## Как работает feature importance?

**Feature importance (важность признаков)** — это метод оценки вклада каждого признака в предсказания модели. Основные подходы:

- **На основе уменьшения ошибки:** Для каждого признака измеряется, насколько уменьшается ошибка (например, MSE или Gini) при его использовании для разбиения в дереве. Признаки, которые чаще используются для разбиения и сильнее уменьшают ошибку, считаются более важными.
- **На основе перестановок:** Для каждого признака измеряется, насколько ухудшается качество модели, если значения этого признака случайно переставить. Чем сильнее ухудшение, тем важнее признак.
- **На основе весов:** В линейных моделях важность признака может быть оценена по абсолютному значению его коэффициента.

### Пример для случайногого леса:

- Для каждого дерева вычисляется важность признаков на основе уменьшения ошибки.
- Итоговая важность признака — это среднее значение по всем деревьям.

## 5. Кластеризация

### Опишите задачу кластеризации. Приведите примеры.

**Задача кластеризации** — это задача разбиения множества объектов на группы (кластеры) таким образом, чтобы объекты внутри одного кластера были похожи друг на друга, а объекты из разных кластеров — отличались. Кластеризация является методом обучения без учителя (unsupervised learning), так как не требует размеченных данных.

### Примеры задач кластеризации:

- **Сегментация клиентов:** Разделение клиентов на группы на основе их поведения, предпочтений или демографических данных.
- **Анализ изображений:** Группировка изображений по схожим характеристикам (например, по цвету или текстуре).

- **Биоинформатика:** Кластеризация генов или белков на основе их функций или структуры.
- **Обнаружение аномалий:** Выделение объектов, которые не попадают ни в один из кластеров, как потенциальных аномалий.

## Метрики качества кластеризации.

**Метрики качества кластеризации** используются для оценки того, насколько хорошо алгоритм кластеризации разделил данные на кластеры. Основные метрики:

- **Индекс силуэта (Silhouette Score):** Оценивает, насколько объект похож на свой кластер по сравнению с другими кластерами. Значения варьируются от -1 до 1, где 1 — идеальная кластеризация.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))},$$

где  $a(i)$  — среднее расстояние от объекта  $i$  до других объектов в том же кластере,  $b(i)$  — среднее расстояние до объектов в ближайшем другом кластере.

- **Индекс Дэвиса-Боулдина (Davies-Bouldin Index):** Оценивает компактность и разделимость кластеров. Чем меньше значение, тем лучше кластеризация.

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right),$$

где  $\sigma_i$  — среднее расстояние внутри кластера  $i$ ,  $d(c_i, c_j)$  — расстояние между центрами кластеров  $i$  и  $j$ .

- **Индекс Калински-Харабаса (Calinski-Harabasz Index):** Оценивает отношение дисперсии между кластерами к дисперсии внутри кластеров. Чем выше значение, тем лучше кластеризация.

$$CH = \frac{SSB}{SSW} \cdot \frac{N - k}{k - 1},$$

где SSB — сумма квадратов между кластерами, SSW — сумма квадратов внутри кластеров,  $N$  — количество объектов,  $k$  — количество кластеров.

**Расскажите про алгоритмы кластеризации, которые вы знаете (например, k-means, DBSCAN...).**

**Основные алгоритмы кластеризации:**

- **K-means:**
  - Разделяет данные на  $k$  кластеров, минимизируя сумму квадратов расстояний от объектов до центроидов кластеров.

- Требует задания числа кластеров  $k$ .
- Чувствителен к начальным условиям и выбросам.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**

- Группирует объекты на основе плотности. Кластеры — это области с высокой плотностью, разделённые областями с низкой плотностью.
- Не требует задания числа кластеров.
- Устойчив к выбросам и может находить кластеры произвольной формы.

- **Иерархическая кластеризация:**

- Строит иерархию кластеров, начиная с отдельных объектов и последовательно объединяя их в кластеры.
- Результат может быть представлен в виде дендрограммы.
- Не требует задания числа кластеров, но требует выбора метода объединения (например, single-linkage, complete-linkage).

- **Gaussian Mixture Models (GMM):**

- Предполагает, что данные сгенерированы смесью нескольких гауссовых распределений.
- Использует Expectation-Maximization (EM) алгоритм для оценки параметров распределений.
- Может находить кластеры различной формы и размера.

- **Mean Shift:**

- Находит центры кластеров, итеративно сдвигая точки в направлении увеличения плотности.
- Не требует задания числа кластеров.
- Устойчив к выбросам и может находить кластеры произвольной формы.

## 6. Computer Vision (CV)

### Что такое свёрточные нейронные сети (CNN)?

Свёрточные нейронные сети (Convolutional Neural Networks, CNN) — это класс глубоких нейронных сетей, предназначенных для обработки данных с grid-подобной структурой, таких как изображения. CNN автоматически извлекают признаки из данных с помощью свёрточных операций.

## Как работает свёртка?

Свёртка — это математическая операция, которая применяет фильтр (ядро) к входным данным для извлечения признаков. Математически свёртка определяется следующим образом:

$$(f * g)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j) \cdot g(x - i, y - j)$$

Где:

- $f$  — входное изображение (матрица пикселей).
- $g$  — фильтр (ядро свёртки).
- $(x, y)$  — координаты выходного значения.

## Пример свёртки

Для изображения  $f$  размером  $3 \times 3$  и фильтра  $g$  размером  $2 \times 2$ :

$$f = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad g = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Результат свёртки:

$$(f * g) = \begin{bmatrix} (1 \cdot 1 + 2 \cdot 0 + 4 \cdot 0 + 5 \cdot (-1)) & \dots \\ \dots & \dots \end{bmatrix}$$

## Как работает архитектура ResNet?

ResNet (Residual Network) — это глубокая нейронная сеть, которая использует "остаточные блоки" (residual blocks) для упрощения обучения очень глубоких сетей. Основная идея ResNet — использование skip-connections (пропускных соединений), которые позволяют градиентам проходить через сеть без затухания.

## Зачем нужны skip-connections?

Skip-connections позволяют информации "перепрыгивать" через несколько слоёв, что решает проблему затухающих градиентов (vanishing gradients) в глубоких сетях. Математически остаточный блок можно описать как:

$$y = F(x, \{W_i\}) + x$$

Где:

- $x$  — входные данные.
- $F(x, \{W_i\})$  — преобразование, выполняемое слоями.
- $y$  — выход блока.

## Что такое пулинг (Pooling)?

Пулинг — это операция, которая уменьшает размерность карты признаков, сохраняя важные признаки. Это помогает уменьшить вычислительную сложность и предотвратить переобучение.

## Виды пулинга

- **Max Pooling:** Выбирает максимальное значение в области.

$$\text{Max Pooling}(x, y) = \max_{i,j \in \text{область}} f(i, j)$$

- **Average Pooling:** Вычисляет среднее значение в области.

$$\text{Average Pooling}(x, y) = \frac{1}{n} \sum_{i,j \in \text{область}} f(i, j)$$

## Пример пулинга

Для карты признаков  $f$  размером  $4 \times 4$  и окна пулинга  $2 \times 2$ :

$$f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Результат Max Pooling:

$$\text{Max Pooling}(f) = \begin{bmatrix} 6 & 8 \\ 14 & 16 \end{bmatrix}$$

Результат Average Pooling:

$$\text{Average Pooling}(f) = \begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix}$$

## 7. Bias-Variance Decomposition

**Bias-Variance Decomposition** — это метод анализа ошибки модели машинного обучения, который разделяет общую ошибку на три компоненты: **bias** (смещение), **variance** (разброс) и **неустранимую ошибку (irreducible error)**.

## Математическая формула

Общая ошибка модели может быть выражена как:

$$\text{Ошибка} = \text{Bias}^2 + \text{Variance} + \text{Неустранимая ошибка}$$

Где:

- **Bias (смещение)** — это ошибка, вызванная упрощением модели. Высокий bias означает, что модель слишком проста и не может уловить сложные закономерности в данных (недообучение).
- **Variance (разброс)** — это ошибка, вызванная чувствительностью модели к небольшим изменениям в обучающих данных. Высокий variance означает, что модель слишком сложна и переобучается на шум в данных (переобучение).
- **Неустранимая ошибка** — это ошибка, вызванная шумом в данных, которую невозможно устранить никакой моделью.

## Пример

Предположим, у нас есть истинная функция  $f(x)$ , а модель предсказывает  $\hat{f}(x)$ . Тогда:

$$\text{Bias} = \mathbb{E}[\hat{f}(x)] - f(x)$$

$$\text{Variance} = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$$

$$\text{Неустранимая ошибка} = \mathbb{E}[(y - f(x))^2]$$

## Интерпретация

- Если **bias** высокий, модель слишком проста и не может уловить сложные зависимости.
- Если **variance** высокий, модель слишком сложна и переобучается на шум в данных.
- Идеальная модель имеет низкий bias и низкий variance.

**Variance (разброс)** — это мера того, насколько предсказания модели изменяются при изменении обучающих данных. Высокий variance указывает на то, что модель слишком чувствительна к шуму в данных, что приводит к переобучению.

Для модели  $\hat{f}(x)$ , variance вычисляется как:

$$\text{Variance} = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$$

Где:

- $\hat{f}(x)$  — предсказание модели для входного значения  $x$ .
- $\mathbb{E}[\hat{f}(x)]$  — математическое ожидание (среднее значение) предсказаний модели.

Предположим, у нас есть несколько обучающих выборок, и для каждой выборки модель делает предсказания  $\hat{f}_1(x), \hat{f}_2(x), \dots, \hat{f}_n(x)$ . Тогда:

$$\mathbb{E}[\hat{f}(x)] = \frac{1}{n} \sum_{i=1}^n \hat{f}_i(x)$$

$$\text{Variance} = \frac{1}{n} \sum_{i=1}^n (\hat{f}_i(x) - \mathbb{E}[\hat{f}(x)])^2$$

- Если **variance** высокий, это означает, что модель слишком чувствительна к изменениям в обучающих данных и переобучается.
- Если **variance** низкий, модель устойчива к изменениям в данных и хорошо обобщает.

## Связь с переобучением

- Высокий variance часто связан с переобучением, когда модель слишком сложна и "запоминает" шум в данных.
- Для уменьшения variance можно использовать методы регуляризации (например, L1, L2), увеличить количество данных или упростить модель.