Співбесіда з ML Engineer. 250 запитань для Junior, Middle та Senior

Джерело питань - DOU + досвід викладача та випускників курсу “Machine learning для людей”

Відповіді: згенеровано з ChatGPT і провалідовано вручну командою Data Loves Academy.

Додатковий набір питань, якщо з цим вже впорались (тут по темам є питання та відповіді, списки активно оновлюються): https://github.com/youssefHosni/Data-Science-Interview-Questions-Answers/tree/main

Окремо виділяю приклади задач на позицію Data Scientist на випадок Coding Interview:

ML задачі на технічне live coding interview

А також окремо знайшла класну добірку питань по LLM з детальними відповідями

50_LLM_Interview_Questions__1738360865.pdf

Теми питань

▼ HR питання / На прояснення наявного досвіду

Цей блок питань в кожного міститиме індивідуальні відповіді, рекомендую їх прописати собі окремо. Всі ці питання справді часто зустрічаються на інтерв'ю (в тому числі в блоці "давайте трошки поговоримо англійською"). Тож, навіть (особливо!) якщо у вас зараз рівень англ A2-B1 - прописуємо

відповіді і вчимо на пам’ять як віршик - це буде ваша опора в момент стресу на інтерв’ю)

1. **Який з ваших останніх проектів був найбільш челенджовим з точки зору побудови моделі машинного навчання? Які основні виклики ви подолали?**
    - Опишіть конкретний проект, підкресліть складні аспекти (наприклад, вибір алгоритму, оптимізація гіперпараметрів, обробка викидів чи дисбалансованих даних) і поясніть, як ви їх вирішили.
2. **Які алгоритми машинного навчання ви використовували найчастіше? Чому саме вони? (Може бути ще - Яка ваша улюблена модель?)**
    - Відповідайте, виходячи з вашого досвіду, пояснюючи, коли і чому ви обирали конкретний алгоритм (наприклад, Random Forest для задач класифікації чи Gradient Boosting для прогнозування).
3. **Як ви підходите до проблеми вибору та налаштування гіперпараметрів моделі?**
    - Опишіть свій процес (наприклад, Grid Search, Random Search, Bayesian Optimization) і поясніть, як ви визначаєте оптимальні значення гіперпараметрів для моделі.
4. **Розкажіть про ваш досвід роботи з нерівномірними наборами даних. Як ви боретеся з проблемою дисбалансу класів?**
    - Описати методи вирішення (підвищення ваги меншого класу, oversampling, undersampling, використання методів типу SMOTE тощо).
5. **Як ви підходите до перевірки якості моделей? Які метрики використовуєте для оцінки продуктивності моделей?**
    - Вкажіть, які метрики (наприклад, Precision, Recall, F1-score, AUC-ROC) ви використовуєте і чому, залежно від задачі (класифікація, регресія, etc.).
6. **Як ви обираєте фічі для побудови моделі? Які методи фічер інженірингу та селекції ви використовуєте?**

- Опишіть методи, які ви використовуєте (наприклад, PCA, Feature Importance, Recursive Feature Elimination) і чому вони важливі.

7. **Які фреймворки та бібліотеки для машинного навчання ви використовували? Які з них є вашими улюбленими і чому?**
    - Розкажіть про ваш досвід з бібліотеками (наприклад, TensorFlow, PyTorch, Scikit-Learn) та чому вам подобається певна бібліотека.
8. **Чи є у вас досвід роботи з моделями в продакшені? Як ви забезпечували їх ефективність і підтримку?**
    - Опишіть свій досвід з деплойментом моделей, моніторингом їх продуктивності, повторною тренуванням і масштабуванням.
9. **Як ви працюєте з відсутніми значеннями в даних? Які підходи вам найбільше підходять?**
    - Опишіть, як ви заповнюєте пропуски (наприклад, середнє значення, медіана, імпутація за допомогою моделей) і як підходите до різних типів даних.
10. **Як би ви реалізували конвеєр (пайплайн) машинного навчання для обробки великого набору даних? (це для вже мідл-сініор рівня)**
    - Поясніть, як ви автоматизуєте процеси підготовки даних, тренування та оцінки моделей, використовуючи такі інструменти, як Apache Airflow або Kubeflow.

🎓 Junior

Загальні питання

1. Який останній ML paper ти читав?
2. Які визначні досягнення в ML були зроблені в тому році?

▼ Machine Learning

1. **Поясніть, як працює логістична регресія.**

   Логістична регресія — це метод машинного навчання для розв'язання задач класифікації. Вона оцінює ймовірність того, що даний зразок належить до одного з класів, використовуючи логістичну функцію (сигмоїд). Формула логістичної функції:

   $P(y = 1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+...+\beta_n X_n)}}$

   Значення ймовірності інтерпретується так: якщо вона перевищує певний поріг (зазвичай 0.5), зразок відноситься до класу 1, інакше — до класу 0.

2. **Для чого потрібна кластеризація? Як вона працює?**

   Кластеризація — це метод групування набору даних у кластери, де об'єкти в кожному кластері мають високу схожість між собою і суттєво відрізняються від об'єктів з інших кластерів. Кластеризація потрібна для виявлення структур у даних, аналізу поведінки користувачів, сегментації ринків тощо. Один із найпоширеніших алгоритмів кластеризації — K-means, який групує дані за допомогою мінімізації відстані між точками і центроїдами кластерів.

3. **Які алгоритми використовують для кластеризації даних? У чому основна ідея кожного з них? Як працює алгоритм K-means?**

   Основні алгоритми кластеризації:

   - **K-means**: Групує дані в (k) кластерів шляхом мінімізації сумарної відстані між точками та центроїдом (середньою точкою) кластеру.
   - **Hierarchical Clustering**: Побудова ієрархії кластерів, об'єднуючи їх поступово на основі відстаней між ними.
   - **DBSCAN**: Виділяє кластери на основі щільності точок, визначаючи основні точки та їхні сусіди.

     Як працює K-means?

     K-means починає з випадкового вибору (k) центроїдів ітеративно розподіляє точки до найближчого центроїду, а потім оновлює центроїди на основі середнього положення точок у кожному кластері. Процес повторюється, доки центроїди не стабілізуються.

4. **Що таке Precision/Recall?**
    - **Precision (Точність)**: Частка правильно передбачених позитивних результатів серед усіх передбачених позитивних результатів. Формула:

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

    - **Recall (Повнота)**: Частка правильно передбачених позитивних результатів серед усіх реальних позитивних результатів. Формула:

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

      Тут TP — кількість істинно позитивних, FP — кількість хибно позитивних, FN — кількість хибно негативних передбачень.
5. **У чому різниця між Accuracy, Precision і Recall? Наведіть приклади, коли слід використовувати кожен з них.**
    - **Accuracy (Точність)**: Частка правильно передбачених результатів серед усіх передбачених результатів. Формула:

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

      Accuracy підходить, коли важливо враховувати всі передбачення, і класи збалансовані.
    - **Precision** важлива, коли помилкові позитивні передбачення мають високі наслідки. Наприклад, виявлення шахрайства, де хибне передбачення може викликати серйозні проблеми.
    - **Recall** важлива, коли важливо зловити всі позитивні випадки, навіть якщо при цьому деякі з них будуть хибними. Наприклад, виявлення хвороб, де важливо не пропустити жоден випадок хвороби.
6. **Що таке валідація? Для чого вона потрібна?**

Валідація — це процес оцінки продуктивності моделі на окремому наборі даних, який не був використаний для тренування моделі. Основна мета валідації — перевірити, наскільки модель здатна узагальнювати знання на нові дані і запобігти проблемі оверфіттингу. Валідаційний набір використовується для налаштування гіперпараметрів моделі, вибору моделі або визначення моменту зупинки тренування.

7. **Для чого потрібні валідаційні та тестові датасети? Чому недостатньо лише одного з них?**

    - **Валідаційний набір** використовується для налаштування моделі, вибору гіперпараметрів, проведення кросвалідації. Він дозволяє оцінити продуктивність моделі під час тренування без впливу на тестовий набір.
    - **Тестовий набір** використовується виключно для фінальної оцінки продуктивності моделі після того, як усі налаштування зроблено. Це дозволяє отримати об'єктивну оцінку того, як модель буде працювати на нових даних. Якщо використовувати лише один набір для обох цілей, модель може «запам'ятати» тестові дані, що призведе до некоректної оцінки її продуктивності.

8. **Яким є правильний процес валідації?**

    - **Поділ даних на тренувальний, валідаційний і тестовий набори.**
    - **Виконання кросвалідації** (наприклад, k-fold cross-validation) на тренувальному наборі з використанням валідаційного набору для вибору гіперпараметрів.
    - **Оцінка моделі** на валідаційному наборі для налаштування моделі.
    - **Фінальна оцінка** продуктивності на тестовому наборі, який не використовувався в процесі налаштування, щоб отримати об'єктивну оцінку.

9. **Які метрики оцінення якості моделей знаєте?**

    Залежно від типу задачі (класифікація чи регресія) використовуються різні метрики:

- **Класифікація**: Accuracy, Precision, Recall, F1-score, ROC-AUC, Log Loss, Confusion Matrix.
- **Регресія**: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared (R^2).

10. **Які метрики варто використовувати для незбалансованих класів?**

    Для незбалансованих класів звичайна Accuracy може бути оманливою, тому варто використовувати такі метрики:

    - **Precision** і **Recall** для оцінки точності та повноти передбачень.
    - **F1-score** — гармонічне середнє Precision і Recall, що враховує обидві метрики.
    - **ROC-AUC** — площа під кривою ROC, що показує якість класифікації для всіх порогів.
    - **Balanced Accuracy** — модифікація Accuracy, яка враховує баланс класів.

11. **Що таке Bias Variance Tradeoff?**
    Bias Variance Tradeoff — це компроміс між упередженістю (bias) та дисперсією (variance) моделі:

    - **Bias (упередження)** — помилка, яка виникає через надто просту модель, яка не здатна захопити складні патерни в даних (underfitting).
    - **Variance (дисперсія)** — помилка, яка виникає через надто складну модель, яка занадто сильно підлаштовується під тренувальні дані, включаючи шум (overfitting).
      Задача — знайти баланс між цими двома аспектами, щоб мінімізувати загальну помилку.

12. **Що таке регуляризація? Які методи регуляризації знаєте?**
    Регуляризація — це метод запобігання оверфіттингу шляхом введення додаткової інформації або обмежень у модель.

    - **L1-регуляризація (Lasso)**: додає штраф за абсолютні значення ваг у функції втрат.

- **L2-регуляризація (Ridge)**: додає штраф за квадратичні значення ваг у функції втрат.
- **Elastic Net**: комбінація L1 та L2 регуляризацій.
- **Dropout**: метод регуляризації для нейронних мереж, коли випадково вимикаються певні нейрони під час тренування.

13. **Чи можна використовувати лінійну регресію для завдань класифікації? Чому?**

    Лінійну регресію не варто використовувати для завдань класифікації, оскільки вона передбачає безперервні значення, тоді як у класифікації необхідно передбачати дискретні класи. Замість цього для задач класифікації краще використовувати логістичну регресію, яка забезпечує ймовірності для кожного класу і дозволяє робити дискретні передбачення.

14. **Опишіть задачу машинного навчання з погляду теорії ймовірності. Поясніть принцип мінімізації емпіричного ризику.**

    Задача машинного навчання може бути описана як пошук функції $h(x)$, яка на основі вхідних даних x передбачає вихідні значення y . В теорії ймовірності це можна описати як пошук умовного розподілу ймовірностей $P(y|x)$. Принцип мінімізації емпіричного ризику означає мінімізацію середньої похибки моделі на тренувальних даних. Математично це можна записати як мінімізацію функції втрат $L(h(x), y)$ по всім прикладам в тренувальному наборі.

15. **Які методи ви знаєте для роботи з категоріальними даними?**

    Основні методи роботи з категоріальними даними:

    - **One-Hot Encoding**: Перетворення категоріальних змінних у набір бінарних змінних.
    - **Label Encoding**: Присвоєння числових значень категоріям.
    - **Target Encoding**: Заміна категорій середнім значенням цільової змінної для кожної категорії.
    - **Frequency Encoding**: Заміна категорій їх частотами у наборі даних.

- **Binary Encoding**: Поєднання Label Encoding та One-Hot Encoding, де категорії спочатку кодуються числами, а потім ці числа переводяться в бінарний формат.

16. **Як зрозуміти, що у вас модель overfitted? А що вказує на те, що вона underfitted**

    - **Overfitting**: Модель добре працює на тренувальних даних, але погано на тестових. Висока точність на тренуванні та низька на валідації є ознакою цього. Це означає, що модель занадто добре "вивчила" тренувальні дані, включаючи шум.
    - **Underfitting**: Модель погано працює як на тренувальних, так і на тестових даних. Це відбувається, коли модель надто проста, щоб захопити патерни в даних.

17. **Розкажіть, як працюють дерева рішень? Які є недоліки? Як боротися з оверфітом?**

    - **Як працюють дерева рішень?**: Дерева рішень розбивають дані на основі запитань, які зменшують неоднорідність (наприклад, на основі критеріїв Gini або ентропії). Кожне рішення веде до подальшого розгалуження, поки не буде досягнуто листа, який представляє кінцеве рішення або клас.
    - **Недоліки**: Дерева рішень можуть бути схильні до оверфіттингу, особливо якщо вони мають багато гілок (глибокі дерева). Вони також нестабільні, що означає, що незначні зміни у даних можуть призвести до значних змін у структурі дерева.
    - **Як боротися з оверфітом?**: Використання методів, таких як обрізання дерева (pruning), обмеження глибини дерева, або використання ансамблевих методів, таких як Random Forest або Gradient Boosting.

18. **Як ансамблювати моделі?**
    Ансамблювання моделей — це метод, що об'єднує передбачення кількох моделей для покращення загальної продуктивності. Основні техніки ансамблювання:

- **Bagging**: Створення кількох моделей на різних підмножинах даних і усереднення або голосування за передбаченнями (наприклад, Random Forest).
- **Boosting**: Послідовне створення моделей, де кожна наступна намагається виправити помилки попередньої (наприклад, Gradient Boosting).
- **Stacking**: Використання виходів кількох базових моделей як вхід для метамоделі, яка робить остаточне передбачення.

19. **Що таке стекінг і бегіг?**
    - **Стекінг (Stacking)**: Техніка ансамблювання, де передбачення кількох моделей використовуються як вхідні дані для іншої моделі (метамоделі), яка робить остаточне передбачення.
    - **Бегінг (Bagging)**: Техніка ансамблювання, де кілька моделей тренуються на різних підмножинах даних, і їх передбачення усереднюються або обирається найбільш популярний клас.
20. **Навіщо потрібен cross-entropy loss? Чому ми не можемо вчити класифікацію за допомогою L2?**
    Cross-entropy loss використовується для класифікації, оскільки він вимірює відстань між передбаченим розподілом і фактичним розподілом класів. Це дозволяє краще керувати моделлю під час оптимізації. Використання L2 для класифікації неефективне, оскільки L2 loss мінімізує квадратичну відстань між передбаченими та фактичними значеннями, що менш підходить для задач класифікації, де важливо розрізняти міжмовірності різних класів.
21. **Які властивості cross-entropy loss?**
    Cross-entropy loss має кілька ключових властивостей:
    - Вона є опуклою функцією, що гарантує наявність єдиного глобального мінімуму.
    - Вона добре підходить для задач класифікації, особливо для багатокласових задач, оскільки вимірює розбіжність між передбаченим розподілом і справжнім.

- Чутлива до помилкових передбачень: якщо модель неправильно класифікує з високою впевненістю, cross-entropy loss буде високим.

22. **Навіщо потрібен Momentum в оптимізаторах? Яку проблему він вирішує?**

    Momentum використовується в оптимізаторах для прискорення процесу збіжності, особливо в умовах глибоких мінімумів або вузьких долин. Він додає інерцію до оновлення ваг, враховуючи попередні градієнти, що дозволяє "проскакувати" через локальні мінімуми і стабілізує процес навчання. Momentum зменшує вплив випадкових коливань градієнта і допомагає уникнути осциляцій.

23. **Як працює RMSProp?**

    RMSProp (Root Mean Square Propagation) — це адаптивний алгоритм оптимізації, який динамічно налаштовує швидкість навчання для кожного параметра. Він зменшує швидкість навчання для параметрів, які мають великі градієнти, і збільшує її для параметрів з малими градієнтами, використовуючи експоненційно зважене середнє квадрата градієнта. Це допомагає уникнути проблеми вибуху градієнтів і робить процес навчання більш стабільним.

24. **Як працює Adam?**

    Adam (Adaptive Moment Estimation) комбінує ідеї з алгоритмів Momentum і RMSProp. Він використовує експоненційно зважене середнє першого моменту (градієнта) і другого моменту (квадрата градієнта) для кожного параметра, а також має власну корекцію упередженості. Це дозволяє адаптивно налаштовувати швидкість навчання для кожного параметра, що робить Adam дуже ефективним і стабільним оптимізатором для глибоких нейронних мереж.

25. **Розкажіть про кросвалідацію.**

    Кросвалідація — це метод оцінки продуктивності моделі на основі поділу даних на кілька підмножин (folds). Найбільш популярна техніка — k-fold cross-validation, де дані діляться на k частин, і модель

тренується на k-1 частинах, а тестується на решті. Процес повторюється k разів, кожного разу з іншою тестовою частиною, після чого середній результат використовується для оцінки продуктивності моделі. Це допомагає уникнути оверфіттингу і дає більш стабільну оцінку моделі.

26. **Як отримати ті самі результати після тренування моделі за умови, що набір даних і параметри моделі не змінюються?**

    Щоб отримати ті самі результати, потрібно зафіксувати (set seed) початкове значення для генератора випадкових чисел, який використовується для ініціалізації ваг моделі, поділу даних і інших випадкових процесів. Це гарантує, що всі випадкові процеси будуть повторювані.

27. **Як працює RNN? Навіщо вони існують і які задачі виконують?**

    RNN (Recurrent Neural Network) — це тип нейронної мережі, що працює з послідовними даними. Вона має рекурентні зв’язки, що дозволяє зберігати інформацію з попередніх кроків для використання в поточному кроці. RNN використовується для задач обробки природної мови, перекладу, аналізу тимчасових рядів, генерації тексту та інших задач, де важлива послідовність даних. Основна проблема RNN — це схильність до затухання або вибуху градієнтів при навчанні на довгих послідовностях.

28. **Розкажіть про LSTM і GRU. Як вони працюють та які проблеми розв'язують?**

    LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit) — це модифікації RNN, які вирішують проблему затухання градієнтів і краще зберігають довготривалу пам'ять. LSTM використовує спеціальні комірки з трьома воротами (input gate, forget gate, output gate), що дозволяють керувати потоком інформації та забувати або зберігати дані за необхідності. GRU є спрощеною версією LSTM з двома воротами (update gate і reset gate), що робить її менш обчислювально затратною, але з аналогічними перевагами.

29. **Чи варто використовувати метрику accuracy? Обґрунтуйте своє рішення.**

Accuracy підходить для задач класифікації, коли дані збалансовані, тобто кількість зразків кожного класу приблизно однакова. Однак, якщо класи незбалансовані (наприклад, один клас значно переважає), accuracy може бути оманливою, оскільки модель може просто передбачати більший клас і досягати високої точності, ігноруючи менші класи. У таких випадках краще використовувати метрики, як-от Precision, Recall або F1-міра.

30. **Розкажіть про недоліки метрики F1-міри.**

    Недоліки F1-міри:

    - F1-міра не враховує кількість негативних класів, що може бути проблемою в задачах з великим обсягом негативних зразків.
    - Вона не надає інформації про розподіл помилок між класами, тому не завжди підходить для задач, де важливо враховувати всі типи помилок (наприклад, FP і FN окремо).
    - F1-міра є середнім арифметичним Precision та Recall, тому втрачається інформація про баланс між цими двома метриками.

31. **Що таке ROC, ROC AUC? Яка математична інтерпретація ROC AUC?**

    - **ROC (Receiver Operating Characteristic)**: Це крива, що показує залежність між True Positive Rate (Sensitivity) та False Positive Rate при різних порогах класифікації.
    - **AUC (Area Under the Curve)**: Це площа під ROC-кривою, яка дає узагальнену міру здатності класифікатора відокремлювати класи. AUC від 0.5 означає випадковий класифікатор, ближче до 1 — ідеальний класифікатор.
      Математично AUC можна інтерпретувати як ймовірність того, що випадковий позитивний зразок отримає вище передбачуване значення, ніж випадковий негативний зразок.

32. **Що таке confusion matrix?**

    Confusion matrix (матриця похибок) — це таблиця, яка використовується для оцінки продуктивності класифікатора. Вона показує кількість правильних і неправильних передбачень, розбитих на класи: True Positives (TP), True Negatives (TN), False Positives (FP), і

False Negatives (FN). Вона дозволяє легко обчислити метрики, як-от Precision, Recall, F1-міру та інші.

33. **Що таке outlier в даних?**

    Outlier (викид) — це спостереження, яке суттєво відрізняється від інших даних у наборі. Викиди можуть бути результатом помилок вимірювання, запису або реальних аномальних подій. Вони можуть сильно вплинути на статистичні моделі, особливо на такі, як лінійна регресія, і часто потребують обробки або видалення.

34. **Що таке кластеринг?**

    Кластеринг — це процес групування набору об'єктів у такі групи (кластери), що об'єкти в одному кластері мають високу схожість між собою і суттєво відрізняються від об'єктів з інших кластерів. Кластеринг застосовується для аналізу даних, сегментації ринку, зменшення розмірності та інших задач.

35. **У чому відмінність між L1 та L2 регулірізаціями? Які властивості у ваг з використанням різних методів?**

    - **L1-регуляризація (Lasso)**: Додає до функції втрат суму абсолютних значень ваг. Вона схильна приводити ваги деяких ознак до нуля, що веде до автоматичного відбору ознак.
    - **L2-регуляризація (Ridge)**: Додає до функції втрат суму квадратів ваг. Вона зменшує всі ваги, але не приводить їх до нуля.

    Властивості ваг:

    - З L1-регуляризацією деякі ваги можуть бути точно нульовими, що сприяє спарсності моделі.
    - L2-регуляризація рівномірно зменшує всі ваги, роблячи модель більш стійкою до мультиколінеарності, але не спарсною.

36. **Яка різниця між supervised та unsupervised learning?**

    - **Supervised Learning** (контрольоване навчання): Модель навчається на мічених даних, де кожен приклад має вхідні ознаки та відповідну

мітку або вихід. Мета — навчитися передбачати мітки для нових даних. Приклади: класифікація, регресія.

- **Unsupervised Learning** (неконтрольоване навчання): Модель навчається на немічених даних, де мітки відсутні. Мета — виявити приховані структури або патерни в даних. Приклади: кластеризація, зменшення розмірності.

37. **Як можна пояснити overfitting? Які є способи його уникнення?**
Overfitting відбувається, коли модель занадто добре підходить до тренувальних даних, включаючи шум і випадкові коливання, але погано узагальнює на нові дані. Способи уникнення:
    - Збір більше даних для тренування.
    - Використання регуляризації (L1, L2).
    - Спрощення моделі (наприклад, зменшення кількості шарів або нейронів).
    - Використання кросвалідації.
    - Використання ансамблевих методів, як-от Bagging або Boosting.
    - Рання зупинка (early stopping) під час тренування нейронних мереж.
38. **Яким чином опрацьовуєте пропуски в даних?**
Пропуски в даних можна опрацьовувати декількома способами:
    - **Видалення** рядків або стовпців з пропусками, якщо їх небагато.
    - **Заповнення** пропусків середнім, медіаною або модою (mean/median/mode imputation).
    - **Заповнення** пропусків з використанням передбачувальних моделей (наприклад, регресії або KNN).
    - Використання спеціальних алгоритмів, які можуть працювати з пропусками (наприклад, XGBoost).
39. **Озвучте базові кроки в Machine Learning pipeline.**
    - **Збір даних**: Отримання відповідних даних для вирішення задачі.

- **Попередня обробка даних**: Очищення даних, обробка пропусків, нормалізація, кодування категоріальних змінних тощо.
- **Розподіл даних**: Поділ даних на тренувальні, валідаційні та тестові набори.
- **Вибір моделі**: Вибір відповідного алгоритму машинного навчання.
- **Тренування моделі**: Навчання моделі на тренувальних даних.
- **Оцінка моделі**: Оцінка продуктивності моделі на валідаційних та тестових даних.
- **Гіперпараметричний тюнінг**: Налаштування гіперпараметрів для покращення моделі.
- **Розгортання**: Інтеграція моделі у виробниче середовище.

40. **Яким чином зберегти/завантажити модель так, щоб використовувати її за допомогою Python?**

    Моделі можна зберігати та завантажувати за допомогою бібліотек, як-от:

    - **Joblib**:

```
from sklearn.externals import joblib
joblib.dump(model, 'model.pkl')
model = joblib.load('model.pkl')
```

    - **Pickle**:

```
import pickle
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)
with open('model.pkl', 'rb') as file:
    model = pickle.load(file)
```

41. **Як справлятися з imbalanced datasets?**
    Декілька підходів для роботи з незбалансованими наборами даних:

- **Oversampling** меншого класу (наприклад, використовуючи SMOTE).
- **Undersampling** більшого класу.
- **Використання ваг**: Налаштування ваг для класів під час тренування моделі.
- **Використання спеціалізованих алгоритмів**, які краще працюють з незбалансованими даними (наприклад, Random Forest або XGBoost).
- **Збір більшої кількості даних для меншого класу**.

42. **Що таке one-hot encoding та коли він корисний?**

    One-hot encoding — це метод перетворення категоріальних змінних у числові, де кожна категорія кодується окремим бінарним стовпцем. Він корисний, коли модель потребує числових вхідних даних, і категорії не мають природного порядку. Наприклад, для категорій "Red", "Green", "Blue" буде створено три стовпці: "Red" (1, 0, 0), "Green" (0, 1, 0), "Blue" (0, 0, 1).

43. **Опишіть процес нормалізації даних і коли він є важливим.**

    Нормалізація — це процес приведення ознак до одного масштабу або діапазону (наприклад, [0, 1] або [-1, 1]). Вона є важливою для алгоритмів, які залежать від відстаней між точками (наприклад, KNN, SVM, нейронні мережі). Нормалізація допомагає уникнути ситуацій, коли ознаки з більшими масштабами домінують над іншими ознаками під час навчання моделі.

44. **Яким чином обробляєте категоріальні дані?**

    Способи обробки категоріальних даних:

    - **One-hot encoding** для номінальних змінних.
    - **Label encoding** для порядкових змінних, де категорії мають природний порядок.
    - **Target encoding** або **mean encoding**: заміна категорій на середні значення цільової змінної для кожної категорії (застосовується з обережністю, щоб уникнути оверфіттингу).

45. **У вас є 1000 прикладів у наборі даних. Які пропорції для train/validation/test розподілу використовуєте? Чому?**

    Зазвичай використовується пропорція 70%/15%/15% або 80%/10%/10% для train/validation/test. Це дозволяє мати достатньо даних для тренування моделі, при цьому зберігаючи достатню кількість даних для оцінки продуктивності на валідаційному та тестовому наборах. Вибір пропорції залежить від загального обсягу даних та складності задачі.

46. **Чому важливо перемішувати (shuffle) дані перед тренуванням моделі?**

    Перемішування даних допомагає запобігти виникненню перекосів у моделі, які можуть виникнути через присутність певного порядку в даних. Наприклад, якщо дані впорядковані за часом або іншою характеристикою, без перемішування модель може навчитися неправильно розпізнавати патерни, що призведе до поганої генералізації.

47. **Чи можете назвати популярні хмарні платформи для розгортання моделей машинного навчання та коротко описати їхні переваги?**

    - **AWS (Amazon Web Services)**: Пропонує широкий спектр інструментів для машинного навчання, включаючи SageMaker для тренування і розгортання моделей. Переваги: масштабованість, безпека, інтеграція з іншими сервісами AWS.
    - **Google Cloud AI**: Містить сервіси для машинного навчання, як-от AI Platform для розгортання моделей, BigQuery для аналізу великих даних. Переваги: легка інтеграція з іншими інструментами Google, підтримка TensorFlow.
    - **Microsoft Azure ML**: Платформа для створення, тренування і розгортання моделей. Переваги: зручна інтеграція з іншими сервісами Azure, підтримка різних мов програмування та середовищ.

48. **При створенні набору даних для бінарної класифікації ви бачите, що один з класів переважає інший як 9:1. Ваші дії?**

У такій ситуації:

- **Балансування даних** через oversampling меншого класу або undersampling більшого класу.
- Використання методів **взважування** класів під час тренування.
- Розгляд використання **спеціалізованих алгоритмів**, таких як Random Forest або XGBoost, які добре справляються з незбалансованими даними.
- Збір додаткових даних для меншого класу, якщо це можливо.

49. **Чи можете ви описати різницю між класифікацією та регресією в машинному навчанні?**
    - **Класифікація**: Завдання передбачення категорійних міток для даних. Наприклад, класифікація електронних листів як спам або не спам.
    - **Регресія**: Завдання передбачення безперервних числових значень. Наприклад, передбачення ціни на житло на основі певних ознак.
50. **Яку роль відіграє інженерія ознак (feature engineering) в машинному навчанні, чи можете ви навести приклад?**
    Інженерія ознак — це процес створення нових ознак з наявних даних, які можуть покращити продуктивність моделі. Це може включати:
    - **Перетворення** даних (наприклад, логарифмування значень).
    - **Групування категорій** (наприклад, об'єднання рідкісних категорій в одну).
    - **Створення нових ознак** (наприклад, добування день тижня з дати).
    - **Використання зовнішніх джерел даних** для створення додаткових ознак (наприклад, погодні дані для прогнозування продажів).

▼ MLOps

1. **Що таке MLOps і чому він важливий у проєктах з машинного навчання?MLOps (Machine Learning Operations)** — це сукупність практик, що об'єднує процеси розробки машинного навчання (ML) та

операційні процеси DevOps. MLOps охоплює весь життєвий цикл моделей ML, включаючи розробку, тренування, розгортання, моніторинг та оновлення моделей у виробничому середовищі.**Важливість MLOps**:

- **Автоматизація**: Зменшення часу на повторювані завдання та спрощення процесів розгортання і моніторингу.
- **Надійність**: Забезпечення стабільної роботи моделей у виробництві, а також можливість швидкого відновлення у разі збоїв.
- **Масштабованість**: Легке масштабування моделей для роботи з великими обсягами даних.
- **Співпраця**: Підвищення ефективності співпраці між командами розробників, науковців та операційних спеціалістів.

2. **Чи працювали ви з системами контролю версій, наприклад Git, у проєкті з машинного навчання? Як це сприяло проєкту?**

   Git — це система контролю версій, яка дозволяє відслідковувати зміни в коді, співпрацювати в команді та зберігати історію змін.**Як це сприяє проєкту з машинного навчання**:

   - **Спільна розробка**: Легке управління змінами коду та можливість паралельної роботи над різними аспектами проєкту.
   - **Відтворюваність**: Можливість відстежувати і відтворювати будь-який стан проєкту на певному етапі, що важливо для відновлення попередніх версій моделей чи експериментів.
   - **Безпека**: Захист від втрати коду завдяки збереженню на віддалених репозиторіях.
   - **Документування змін**: Кожна зміна в коді супроводжується повідомленням (commit message), що полегшує розуміння історії проєкту.

3. **Чи можете пояснити концепцію постійної інтеграції (CI) і постійного розгортання (CD) у контексті машинного навчання? Постійна інтеграція (CI)**

— це практика автоматичного тестування та інтеграції коду при кожному оновленні в репозиторії. У контексті ML це може включати автоматизацію тренування моделей, тестування якості даних і моделей, а також перевірку сумісності нових змін з існуючими процесами.

Постійне розгортання (CD) — це практика автоматичного розгортання нових версій моделей у виробничому середовищі після успішного проходження усіх тестів і перевірок. У контексті ML це дозволяє швидко оновлювати моделі на основі нових даних або покращених алгоритмів, мінімізуючи час простою і забезпечуючи постійну актуальність моделі.
Ці концепції забезпечують безперервний і надійний процес розробки та розгортання моделей, що дозволяє швидко впроваджувати нововведення та підтримувати стабільну роботу системи.

▼ Deep Learning

1. **Які архітектури мереж для завдань класифікації вам відомі?**
   Декілька популярних архітектур нейронних мереж для завдань класифікації:

   - **Персептрон (Perceptron)**: Базова нейронна мережа з одним шаром, що використовується для лінійної класифікації.
   - **Багатошаровий персептрон (MLP)**: Мережа з кількома прихованими шарами, що дозволяє вирішувати нелінійні задачі.
   - **Конволюційні нейронні мережі (CNN)**: Використовуються для обробки зображень та відео, наприклад, архітектури AlexNet, VGG, ResNet.
   - **Рекурентні нейронні мережі (RNN)**: Для роботи з послідовностями даних, наприклад, LSTM та GRU.
   - **Transformers**: Використовуються для NLP задач, наприклад, BERT, GPT.

2. **Що таке transfer learning і коли варто його застосовувати?**

   Transfer learning — це метод навчання моделей, де попередньо натренована модель використовується як основа для нової задачі. Замість того, щоб тренувати модель з нуля, використовуються ваги, отримані на іншому великому наборі даних, і адаптуються до нової задачі.

   Коли застосовувати: Коли є обмежена кількість даних для тренування моделі, або коли задача схожа на ту, для якої вже існують натреновані моделі (наприклад, класифікація зображень або NLP задачі).

3. **З якими моделями ви працювали? Розкажіть про їхні особливості та новації.**

   (Це питання варто адаптувати під ваш досвід роботи з конкретними моделями. Наприклад, якщо ви працювали з CNN, LSTM, або Transformer, варто описати їх особливості, такі як:

   - **CNN**: Використання згорткових шарів для автоматичного вилучення ознак із зображень.
   - **LSTM/GRU**: Використання "воріт" для запам'ятовування або забування інформації в послідовностях даних.
   - **Transformers**: Використання механізму self-attention для роботи з послідовностями без рекурентності.)

4. **Для чого потрібна функція активації в нейронних мережах?**

   Функція активації визначає вихід нейрона на основі вхідних даних. Вона вводить нелінійність у модель, що дозволяє мережі моделювати складні патерни і взаємозв'язки в даних. Без функцій активації модель була б лінійною і не могла б вирішувати складні задачі.

5. **Як відбувається початкова ініціалізація ваг у нейронних мережах? Чому вона не може бути нульовою?**

   Початкова ініціалізація ваг зазвичай відбувається за допомогою випадкових значень, що обираються зі спеціальних розподілів, як-от He або Xavier ініціалізація.

Чому не нульова? Якщо всі ваги ініціалізувати нулями, то нейрони в кожному шарі будуть ідентичними під час навчання, оскільки градієнти будуть однаковими. Це призведе до симетрії і не дозволить мережі навчитися чомусь корисному.

6. **Навіщо потрібна батч-норма? За що відповідають alpha, beta? Що буде, якщо їх забрати?**

   Батч-нормалізація (Batch Normalization) — це метод нормалізації вхідних даних у кожному шарі нейронної мережі, що допомагає стабілізувати і прискорити процес навчання. Вона зменшує зміщення і дисперсію вхідних даних, що дозволяє використовувати вищі швидкості навчання та знижує залежність від початкової ініціалізації ваг.

   Alpha і **Beta** — це параметри зміщення і масштабування, які дозволяють моделі відновити первісний розподіл, якщо це необхідно.**Що буде, якщо їх забрати?** Модель буде використовувати стандартний розподіл, отриманий після нормалізації, але це може обмежити її здатність адаптуватися до різних розподілів даних.

7. **Що таке dropout? Як він себе поводить на trainee і на тесті?**

   Dropout — це техніка регуляризації, яка випадково вимикає частину нейронів під час навчання, щоб запобігти оверфіттингу. Вимкнені нейрони не використовуються під час кожної ітерації навчання, що робить модель більш стійкою до залежності від окремих нейронів.

   На тренуванні dropout активно вимикає нейрони, щоб уникнути перенавчання.

   На тесті dropout вимикається, і модель використовує всі нейрони для передбачень, але ваги зменшуються, щоб компенсувати вимкнення під час навчання.

8. **Як би ви перетворювали часові дані в числовий формат?**

   Для перетворення часових даних у числовий формат можна використовувати кілька підходів:

   - **Перетворення дати та часу у кількісні ознаки**: Наприклад, рік, місяць, день, година, хвилина тощо.

- **Створення ознак із сезонних компонентів**: День тижня, чи є вихідним днем, квартал року тощо.
- **Створення лагових змінних**: Зміщення часових даних назад або вперед для моделювання залежностей у часі.
- **Перетворення дати у Unix timestamp**: Кількість секунд з початку епохи (1970-01-01).

▼ Алгоритми та структури даних + ML

1. **Які типи задач машинного навчання ви знаєте?**
   Основні типи задач машинного навчання:
   - **Supervised Learning (Контрольоване навчання)**: Задачі, де є мічені дані (вхідні дані та відповідні мітки). Приклади: класифікація, регресія.
   - **Unsupervised Learning (Неконтрольоване навчання)**: Задачі, де немає міток, і мета полягає у виявленні прихованих структур у даних. Приклади: кластеризація, зниження розмірності.
   - **Semi-supervised Learning**: Комбінація контрольованого та неконтрольованого навчання, де є частково мічені дані.
   - **Reinforcement Learning (Навчання з підкріпленням)**: Модель навчається через взаємодію з середовищем, отримуючи винагороди або покарання за свої дії.
2. **Опишіть алгоритм роботи Support Vector Machines.**

   Support Vector Machines (SVM) — це алгоритм класифікації, який шукає гіперплощину, що максимально розділяє класи даних. Основні кроки роботи SVM:
   - Алгоритм знаходить "опорні вектори" — точки, які визначають межі класифікації.
   - SVM будує гіперплощину, що максимально віддалена від найближчих точок кожного класу (опорних векторів). Це максимізує **margin** — відстань між гіперплощиною і точками кожного класу.

- Якщо дані не є лінійно роздільними, використовуються **ядерні функції (kernels)** для перетворення даних у вищий простір, де вони стають лінійно роздільними.

3. **Перерахуйте основні структури даних, їхні плюси й мінуси.**

   Основні структури даних:

   - **Масив (Array)**:
     - Плюси: Швидкий доступ за індексом, простота використання.
     - Мінуси: Фіксований розмір, вставка та видалення елементів може бути повільною.
   - **Список (Linked List)**:
     - Плюси: Динамічний розмір, легка вставка та видалення.
     - Мінуси: Повільний доступ за індексом, потребує більше пам'яті.
   - **Стек (Stack)**:
     - Плюси: Швидкі операції додавання та видалення.
     - Мінуси: Лише останній елемент доступний для видалення (LIFO).
   - **Черга (Queue)**:
     - Плюси: Швидкі операції додавання та видалення.
     - Мінуси: Доступ лише до першого елемента (FIFO).
   - **Хеш-таблиця (Hash Table)**:
     - Плюси: Швидкий доступ до елементів за ключем.
     - Мінуси: Погана продуктивність у разі колізій.
   - **Дерево (Tree)**:
     - Плюси: Ефективний пошук, вставка та видалення даних, особливо у збалансованих деревах.
     - Мінуси: У незбалансованих деревах пошук може бути повільним.

4. **Які умови потрібно виконувати для застосування закону великих чисел?**

   Закон великих чисел стверджує, що за великої кількості незалежних ідентично розподілених спостережень середнє значення вибірки наближається до математичного очікування.
   Умови:

   - Спостереження повинні бути **незалежними**.
   - Спостереження мають бути **однаково розподілені**.
   - Кількість спостережень повинна бути досить великою.

5. **Яка різниця між помилками Type I та Type II?**

   - **Помилка Type I (хибнопозитивна)**: Відхилення нульової гіпотези, коли вона є вірною. Це означає, що модель помилково визначила щось як істинне, коли це не так. Приклад: Помилкове виявлення шахрайства, коли його не було.
   - **Помилка Type II (хибнонегативна)**: Прийняття нульової гіпотези, коли вона є хибною. Це означає, що модель не змогла виявити щось, коли це мало місце. Приклад: Пропуск шахрайської транзакції.

▼ Python

1. **Які типи даних у Python є змінними (mutable) та незмінними (immutable)?**

   - **Mutable (змінні):** List, Dict, Set.**Immutable (незмінні):** Int, Float, Tuple, Str, Frozenset.**Mutable типи** можуть змінюватися після створення, а **immutable типи** не можуть бути змінені після створення.

2. **Що таке "list comprehension"? Коли його варто використовувати?**

   - **List comprehension** — це синтаксис Python для створення нового списку, заснованого на існуючому ітераторі. Використовується для скорочення коду, коли потрібно застосувати якусь операцію до всіх елементів іншого списку.**Приклад:** `[x**2 for x in range(10)]` створює список квадратів чисел від 0 до 9.

3. **Як працює механізм обробки винятків (exceptions) у Python? Які є типи винятків?**
    - Механізм обробки винятків використовує блоки `try`, `except`, `finally`. Блок `try` виконує код, який може викликати виняток, а `except` обробляє виняток. `finally` виконується завжди, незалежно від того, чи виник виняток.**Типи винятків:** `ValueError`, `TypeError`, `KeyError`, `IndexError`, `ZeroDivisionError`, та інші.
4. **Що таке генератори (generators) у Python? Як вони працюють і коли їх слід використовувати?**
    - **Генератори** — це функції, які використовують ключове слово `yield` для повернення елементів по одному за раз. Вони зберігають свій стан між викликами, що робить їх ефективними для обробки великих обсягів даних або нескінченних послідовностей.**Приклад:**

```
def count_up_to(n):
    count = 1
    while count <= n:
        yield count
        count += 1
```

5. **Що таке “lazy evaluation”? Де в Python це використовується?**
    - **Lazy evaluation (лінива оцінка)** — це концепція, коли обчислення відкладаються до тих пір, поки результат не буде потрібен. У Python це використовується у генераторах та функціях на кшталт `range()`, що дозволяє ефективно обробляти великі набори даних, не завантажуючи їх у пам'ять повністю.
6. **Які в Python вбудовані функції для роботи з ітераторами та генераторами?**
    - Основні функції:
        - `map()` — застосовує функцію до кожного елемента ітератора.
        - `filter()` — відбирає елементи за певною умовою.
        - `zip()` — поєднує кілька ітераторів в один.

- `enumerate()` — повертає пару (індекс, значення) для кожного елемента ітератора.

7. **Що таке “GIL” (Global Interpreter Lock) і як він впливає на багатопоточність у Python?**
   - **GIL (Global Interpreter Lock)** — це механізм у CPython, що обмежує виконання тільки одного потоку Python коду одночасно. Це означає, що багатопоточність не завжди ефективна для CPU-bound задач, і для таких задач краще використовувати модуль `multiprocessing`.
8. **Які існують способи оптимізації пам'яті та продуктивності коду на Python?**
   - Використання бібліотек типу `NumPy` для обробки числових даних, застосування генераторів замість списків, кэшування результатів, оптимізація алгоритмів, використання компіляторів типу Cython або JIT-компіляторів.
9. **Що таке context manager і як він працює (приклад: 'with' statement)?**
   - **Context manager** дозволяє керувати ресурсами (наприклад, файлами) автоматично. Ви використовуєте `with` для автоматичного закриття файлів або звільнення ресурсів.**Приклад:**

     ```
     with open('file.txt', 'r') as f:
         content = f.read()
     ```
10. **Чим відрізняються threading і multiprocessing у Python? Коли використовувати кожен підхід?**
    - **Threading** підходить для задач I/O-bound (введення/виведення), де GIL не є критичним фактором, тоді як **multiprocessing** підходить для CPU-bound задач, оскільки кожен процес має власний інтерпретатор і не залежить від GIL.
11. **Як працюють “lambda” функції? Які у них переваги та обмеження?**
    - **Lambda** — це анонімна функція, що визначається у одному рядку. Її переваги — компактний синтаксис для простих функцій, але вона

обмежена лише одним виразом і менш читабельна у складних випадках.**Приклад:**

```
square = lambda x: x ** 2
```

12. **Що таке "serialization"? Які способи серіалізації даних у Python вам відомі?**
    - **Серіалізація** — це процес перетворення об'єктів Python у формат, який можна зберігати або передавати (наприклад, JSON, Pickle).**Приклади:**
        - `pickle` для серіалізації складних об'єктів Python.
        - `json` для обміну даними між веб-додатками.
13. **Яка різниця між “deep copy” і “shallow copy” в Python? Коли їх варто використовувати?**
    - **Shallow copy** створює новий об'єкт, але вставляє лише посилання на вкладені об'єкти. **Deep copy** створює новий об'єкт і рекурсивно копіює всі вкладені об'єкти. Використовуйте `deepcopy` для складних об'єктів, які потребують повного копіювання.**Приклад:**

```
import copy
deep_copy = copy.deepcopy(original)
shallow_copy = copy.copy(original)
```

14. **Що таке "Метакласи" в Python і для чого їх використовують?**
    - **Метаклас** — це клас для класів, що дозволяє змінювати поведінку класів під час їх створення. Використовується для динамічної зміни структури класу, автоматизації шаблонів коду, створення singleton класів тощо.
15. **Які ви знаєте підходи до багатопотокового програмування в Python?**
    - Основні підходи:
        - **Threading:** Використання модуля `threading` для I/O-bound задач.

- **Multiprocessing:** Використання модуля `multiprocessing` для CPU-bound задач.
- **Asyncio:** Використання асинхронного програмування для високопродуктивних I/O-bound додатків.

16. **Що таке “duck typing” у Python? Наведіть приклад.**

- **Duck typing** — це концепція динамічної типізації, де тип об'єкта визначається його поведінкою (методами і атрибутами), а не його явним типом. **Приклад:**

```
def quack(duck):
    duck.quack()
```

17. **Як можна створити singleton клас у Python?**

- **Singleton** — це клас, який дозволяє створити лише один об'єкт. Можна реалізувати за допомогою метакласів або декораторів. **Приклад з метакласом:**

```
class Singleton(type):
    _instances = {}
    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            cls._instances[cls] = super().__call__(*args, **kwargs)
        return cls._instances[cls]

class MyClass(metaclass=Singleton):
    pass
```

18. Яка різниця між “map”, “reduce” і “filter” функціями в Python?

    Наведіть приклади їх використання.**

- **Map:** застосовує функцію до кожного елемента ітератора.
- **Filter:** залишає тільки ті елементи, які відповідають певній умові.

- **Reduce:** агрегує елементи ітератора за допомогою функції.**Приклад:**

```
from functools import reduce
numbers = [1, 2, 3, 4]
squared = list(map(lambda x: x ** 2, numbers))
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
sum_numbers = reduce(lambda x, y: x + y, numbers)
```

▼ Computer Vision

1. **Як працює Canny Edge Detection в OpenCV, порахуйте на прикладі прохід довільного фільтру, наприклад фільтру Соболя, за картинкою.**

   Canny Edge Detection — це алгоритм для виявлення країв на зображеннях, який складається з декількох кроків:

   - **Гауссове згладжування**: Зменшення шуму в зображенні за допомогою гауссового фільтра, щоб уникнути помилкових країв.
   - **Обчислення градієнта**: Використовується оператор Собеля для обчислення градієнтів зображення по осям x і y. Градієнти показують напрямок і величину змін у яскравості.
   - **Придушення немаксимумів (Non-maximum suppression)**: Залишаються лише ті пікселі, які мають локальні максимуми градієнта — це потенційні краї.
   - **Подвійний поріг (Double threshold)**: Алгоритм використовує два пороги для визначення сильних і слабких країв.
   - **Остаточне відновлення країв (Edge tracking by hysteresis)**: Слабкі краї, що з'єднані з сильними краями, зберігаються, інші відкидаються.

   Приклад з фільтром Соболя:
   Оператор Собеля обчислює похідну зображення по осям x і y, що використовується для виявлення напрямку градієнта.
   Формули для обчислення градієнтів:

- Фільтр по осі x (Sobel X):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Фільтр по осі y (Sobel Y):

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

  Після згортки зображення з цими фільтрами обчислюється величина градієнта для кожного пікселя:

  G = \sqrt{G_x^2 + G_y^2}

2. **Що таке аугментація?**

   Аугментація (augmentation) — це техніка штучного збільшення кількості даних за допомогою трансформацій початкових зображень. Вона використовується для покращення узагальнення моделі і запобігання оверфіттингу.
   Основні методи аугментації:

   - **Обертання** (Rotation)
   - **Зсув** (Translation)
   - **Масштабування** (Scaling)
   - **Зміна яскравості** (Brightness adjustment)
   - **Фліп (перевертання)** (Flip, наприклад, по горизонталі)
   - **Шуми** (Adding noise)
     Аугментація дозволяє моделі бачити більше різноманітних

прикладів, покращуючи її здатність розпізнавати об'єкти в різних умовах.

▼ Статистика

Багато питань тут стосуються теореми Баєса не просто так - вони справді частоо зустрічаються. Розібратись з цією темою найкраще тут.

1. **Що таке теорема Баєса і як її можна застосувати у машинному навчанні?**
    - Теорема Баєса дозволяє обчислити умовну ймовірність події A, виходячи з наявних даних про подію B. Формула:

      $$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

      У машинному навчанні вона використовується, наприклад, у наївному баєсовому класифікаторі для оцінки ймовірності приналежності об'єкта до певного класу.
2. **Чому Баєсів класифікатор називається "наївним"?**
    - Баєсів класифікатор називається "наївним", тому що він припускає незалежність між усіма ознаками (features). Це спрощене припущення рідко відповідає реальності, але на практиці наївний баєсів класифікатор часто працює дуже добре.
3. **Що таке формула повної ймовірності?**
    - Формула повної ймовірності дозволяє обчислити ймовірність події A, використовуючи її умовні ймовірності для різних несумісних подій B_i. Формула:

      $$P(A) = \sum P(A|B_i)P(B_i)$$

      Ця формула допомагає, коли ймовірність події залежить від кількох альтернативних причин.

4. **Припустимо, ймовірність захворіти на туберкульоз (ТБ) дорівнює 0.0005, а тест на ТБ є точним на 99%. Яка ймовірність того, що людина має ТБ, якщо тест показав позитивний результат?**

   - Використовуючи теорему Баєса, можна обчислити ймовірність наявності ТБ за формулою:

     $$P(\text{ТБ}|\text{позитивний тест}) = \frac{P(\text{позитивний тест}|\text{ТБ}) \cdot P(\text{ТБ})}{P(\text{позитивний тест})}$$

     Де
     $P(\text{позитивний тест})$ можна обчислити через ймовірності правильного та хибного позитивного результату тесту.

     $$\frac{(0.0005)(0.99)}{(0.0005)(0.99) + (0.9995)(0.01)} \doteq 0.0472$$

5. **Компанія A постачає 40% комп'ютерів і запізнюється з доставкою у 5% випадків. Компанія B постачає 30% комп'ютерів і запізнюється на 3%, а компанія C постачає 30% і запізнюється на 2.5%. Якщо комп'ютер доставлено із запізненням, яка ймовірність, що він походить від компанії A?**

   - Використовуючи теорему Баєса, ймовірність того, що запізнілий комп'ютер був постачений компанією A, обчислюється за формулою:

     $$P(A|\text{запізнення}) = \frac{P(\text{запізнення}|A) \cdot P(A)}{P(\text{запізнення})}$$

     Тут
     $P(\text{запізнення})$ – це загальна ймовірність запізнення, яку можна обчислити, підсумувавши ймовірності для всіх компаній.

> Bayes Theorem. Make a tree: P(L) = 0.0365 and P(A and L) = (0.4)(0.05) = 0.02, so P(shipped from A given that the computer is late) = 0.548, approximately.

6. **Що таке p-value?**
    - p-значення – це ймовірність отримати результат, що дорівнює або перевищує спостережуваний, якщо нульова гіпотеза вірна. Воно використовується для перевірки статистичних гіпотез: якщо p-value менше за рівень значущості (зазвичай 0.05), ми відхиляємо нульову гіпотезу.
7. **Що таке кореляція? Які її типи бувають?**
    - Кореляція – це міра лінійного зв'язку між двома змінними. Якщо дві змінні мають позитивну кореляцію, це означає, що вони збільшуються або зменшуються разом. При негативній кореляції одна змінна зменшується, коли інша збільшується. Кореляція може бути:
        - **Позитивною** (кореляційний коефіцієнт > 0)
        - **Негативною** (кореляційний коефіцієнт < 0)
        - **Нульовою** (відсутність лінійної залежності, кореляційний коефіцієнт ≈ 0).
8. **Що таке дисперсія і стандартне відхилення?**
    - **Дисперсія** показує, наскільки значення в наборі даних відрізняються від середнього значення, і обчислюється як середнє арифметичне квадратів відхилень кожного значення від середнього. **Стандартне відхилення** – це квадратний корінь з дисперсії і показує середню відстань значень від середнього.
9. **Що таке R^2 в лінійній регресії?**
    - R^2 або коефіцієнт детермінації – це метрика, яка показує, наскільки добре модель пояснює варіацію цільової змінної. $R^2 = 1$ означає ідеальне передбачення, $R^2 = 0$ означає, що модель не краще за середнє арифметичне.
10. **Як інтерпретувати коефіцієнти в лінійній регресії?**

- Коефіцієнти в лінійній регресії показують вплив зміни незалежної змінної на залежну змінну. Наприклад, якщо коефіцієнт дорівнює 2, це означає, що зі збільшенням незалежної змінної на 1 одиницю, залежна змінна збільшується на 2 одиниці (за інших рівних умов).

11. **Що таке тести A/B? Як вони застосовуються?**
    - A/B тестування – це метод порівняння двох варіантів (наприклад, дизайну веб-сторінки) для оцінки, який з них має кращу продуктивність. Статистично тест перевіряє, чи є різниця між групами значущою.
12. **Що таке мультиколінеарність і чому вона важлива в регресійних моделях?**
    - Мультиколінеарність виникає, коли незалежні змінні в регресійній моделі сильно корелюють між собою. Це може призвести до неточних оцінок коефіцієнтів і ускладнити інтерпретацію моделі.
13. **Що таке довірчий інтервал (confidence interval)?**
    - Довірчий інтервал – це діапазон значень, в якому з певною ймовірністю знаходиться істинне значення параметра (наприклад, середнього) популяції. Наприклад, 95% довірчий інтервал означає, що є 95% ймовірність того, що істинне значення параметра лежить в цьому інтервалі.
14. **Що таке центральна гранична теорема (Central Limit Theorem, CLT)?**
    - Центральна гранична теорема стверджує, що незалежно від розподілу вихідної популяції, розподіл вибіркових середніх наближається до нормального розподілу, якщо вибірка достатньо велика (зазвичай n > 30). Це дозволяє використовувати нормальний розподіл для проведення статистичних тестів навіть для не нормальних даних.
15. **Припустимо, ви підкидаєте монету 10 разів і спостерігаєте тільки одне випадіння герба. Якою буде нульова гіпотеза і p-значення для перевірки, чи монета не файльшива (правильна)?**

- Нульова гіпотеза: монета є правильною (ймовірність випадання герба $p = 0.5$). Для перевірки гіпотези можна використати біноміальний тест. P-значення покаже ймовірність отримати таке ж або ще більш екстремальне співвідношення гербів і решок при справедливій монеті.

14. **У розподілі з лівим нахилом (left-skewed distribution), де медіана дорівнює 60, які висновки можна зробити про середнє та моду даних?**
    - Для розподілу з лівим нахилом середнє буде менше медіани, оскільки в нього більше впливають "хвости" (великі негативні відхилення). Мода, як правило, знаходиться праворуч від медіани і середнього.
15. **Ви витягуєте значення (робите семплінг) з рівномірного розподілу [0, d] n разів. Яка ваша найкраща оцінка параметра d?**
    - Найкраща оцінка для d буде $\frac{n+1}{n} \times \max(X)$, де $\max(X)$ – максимальне значення вибірки, а n – кількість спостережень.

🎩 Middle

▼ Загальні запитання

1. **Який останній paper ви читали?**

   Приклад: "Останній paper, який я прочитав, був про нові підходи до оптимізації градієнтного бустингу для задач класифікації. У ньому обговорювалися покращення в продуктивності XGBoost через паралелізацію обчислень та глибші дерева."

2. **Як ви оцінюєте результативність моделі машинного навчання? Які метрики використовуєте для завдань регресії та класифікації? Коротко опишіть їхні переваги.**
   - Для **класифікації**:
     - **Accuracy**: Проста метрика для збалансованих наборів даних.

- **Precision/Recall/F1-score**: Використовується для незбалансованих наборів даних. Precision оцінює точність позитивних передбачень, Recall — покриття позитивних класів, а F1-score — гармонійне середнє між Precision і Recall.
- **ROC-AUC**: Показує здатність моделі відрізняти класи для всіх можливих порогів.

- Для **регресії**:
  - **MAE (Mean Absolute Error)**: Оцінює середню абсолютну помилку передбачень.
  - **MSE (Mean Squared Error)**: Оцінює середню квадратичну помилку, більше штрафує за великі відхилення.
  - **R^2 (коефіцієнт детермінації)**: Показує, наскільки добре модель пояснює варіацію цільової змінної.

▼ Machine Learning

1. **Як ви обираєте тип класифікатора, базуючись на розмірі даних тренувального набору?**
   - **Малі набори даних**: Вибір простіших моделей (KNN, Decision Tree) або алгоритмів з регуляризацією (Logistic Regression).
   - **Середні набори даних**: Багато моделей працюють добре, але для великих обсягів даних краще використовувати SVM, Random Forest або XGBoost.
   - **Великі набори даних**: Перевагу надають ансамблевим моделям (Random Forest, XGBoost) або нейронним мережам.
2. **Поясніть метод градієнтного спуску. Як обрати швидкість навчання для алгоритму оптимізації градієнтного спуску?**

   Градієнтний спуск — це метод оптимізації, який мінімізує функцію втрат шляхом ітеративного оновлення ваг моделі в напрямку градієнта.
   - **Швидкість навчання** (α): Маленька швидкість навчання гарантує стабільну збіжність, але навчання займатиме більше часу.

Велика швидкість може призвести до нестабільної збіжності або пропуску мінімуму. Зазвичай використовують кросвалідацію для підбору оптимальної швидкості.

3. **Як ви вирішуватимете проблему класифікації тексту, якщо у вас незбалансований датасет?**
   - **Oversampling** меншого класу (наприклад, SMOTE).
   - **Undersampling** більшого класу.
   - Використання **вагових коефіцієнтів** для меншого класу.
   - Використання моделей, які добре працюють з незбалансованими даними (наприклад, XGBoost).
   - Використання метрик, таких як Precision, Recall або F1-score, замість Accuracy.
4. **Як правильно робити крос-валідацію на time-series?**

   У випадку time-series необхідно зберігати тимчасову послідовність даних. Можна використовувати метод **TimeSeriesSplit** (в бібліотеці Scikit-learn), де дані розбиваються на тренувальні і тестові послідовно, а кожен новий тестовий набір включає новіші дані, не змішуючи часові ряди.
5. **Batch vs Mini-batch: у чому різниця та коли що краще використовувати?**
   - **Batch gradient descent**: Використовує весь набір даних для кожного оновлення градієнта. Краще для гладких функцій втрат, але може бути дуже повільним для великих наборів даних.
   - **Mini-batch gradient descent**: Використовує невеликі частини даних (мікробатчі) для кожного оновлення градієнта. Компроміс між batch і stochastic gradient descent, оскільки поєднує в собі стабільність і ефективність. Часто використовується в глибокому навчанні для великих наборів даних.
6. **Які є типи та джерела data leakage? Наведіть приклади кожного типу та як їх уникати.**

- **Train-test leakage**: Інформація з тестових даних потрапляє у тренувальні. Уникнути цього можна правильним розподілом даних та виконанням попередньої обробки окремо для тренувальних і тестових даних.
- **Feature leakage**: Використання змінних, які містять інформацію про цільову змінну, але які не будуть доступні під час передбачень у виробничому середовищі. Приклад: використання майбутньої інформації при передбаченні подій. Уникнення: уважно перевіряти часові залежності та наявність таких ознак.

7. **Розкажіть основні концепти роботи градієнтного бустингу. Чим відрізняється XGBoost від LightGBM?**

   Градієнтний бустинг: Моделі тренуються послідовно, і кожна нова модель намагається виправити помилки попередніх. Алгоритм мінімізує функцію втрат, додаючи нові дерева.

   - **XGBoost**: Використовує регуляризацію для покращення узагальнення та є менш чутливим до оверфіттингу. Підтримує паралелізацію.
   - **LightGBM**: Застосовує техніку "leaf-wise growth" замість "level-wise" (як у XGBoost), що робить його швидшим на великих наборах даних. Також використовує техніку пріоритетного навчання для обробки великих кількостей ознак.

8. **Як працюють Conv-LSTM і Conv-GRU? Чим вони відрізняються від LSTM та GRU?Conv-LSTM** і **Conv-GRU** — це варіації звичайних LSTM і GRU, які використовують згорткові шари (convolutions) для обробки просторових даних разом із часовою інформацією.

   - **Conv-LSTM**: Замість звичайних векторів у LSTM використовуються згорткові шари, що дозволяє працювати з відеоданими або зображеннями, зберігаючи часову залежність.
   - **Conv-GRU**: Аналогічно Conv-LSTM, але простіший через меншу кількість воріт (gates). Conv-GRU використовує згорткові шари для обробки просторової інформації.
     Відмінність від звичайних LSTM і GRU: Звичайні LSTM та GRU

працюють з послідовними даними, як-от текст або часові ряди, тоді як Conv-LSTM/GRU комбінують просторову і часову інформацію, що корисно для відео або аналізу зображень у послідовності.

9. **Які є методи для тренування моделей на незбалансованих датасетах? Які в них переваги та недоліки?**
   Методи для роботи з незбалансованими даними:
   - **Oversampling меншого класу** (наприклад, SMOTE): Збільшує кількість зразків меншого класу шляхом генерування нових даних.
     - Переваги: Дає змогу зберегти всі дані більшого класу.
     - Недоліки: Може призвести до оверфіттингу через повторення або створення схожих зразків.
   - **Undersampling більшого класу**: Зменшує кількість зразків більшого класу.
     - Переваги: Менше даних для обробки, що зменшує час тренування.
     - Недоліки: Втрата інформації з більшого класу.
   - **Взважування класів**: Присвоєння вищих ваг меншому класу в процесі тренування.
     - Переваги: Не потрібно змінювати набір даних.
     - Недоліки: Вимагає ретельного налаштування ваг, щоб уникнути переваги меншого класу.
   - **Спеціалізовані алгоритми** (наприклад, Random Forest або XGBoost): Вбудовані методи для роботи з незбалансованими даними.
10. **Якщо у вас у датасеті є багато ознак, яким чином ви будете обирати/видаляти їх для моделювання?**
    Методи відбору ознак:
    - **Feature importance**: Використання моделей, як-от Random Forest або XGBoost, для оцінки важливості кожної ознаки.
    - **L1-регуляризація** (Lasso): Присвоює ваги ознакам, і деякі з них можуть дорівнювати нулю, що дозволяє відсіяти непотрібні ознаки.

- **Кореляційний аналіз**: Видалення ознак, які мають високу кореляцію між собою, щоб уникнути мультиколінеарності.
- **Методи зниження розмірності**, як-от PCA (Principal Component Analysis), що допомагає створити нові ознаки на основі вихідних.

11. **Запропонуйте метрики, які б відображали, наскільки впевнена модель у передбаченні?**

    - **Entropy**: Вимірює рівень невизначеності в передбаченні. Менша ентропія означає більшу впевненість.
    - **Softmax output**: Вихідна ймовірність після застосування Softmax-функції. Чим ближче ймовірність до 1, тим більша впевненість моделі.
    - **Confidence intervals**: Можна використовувати для моделей, які генерують інтервали передбачень, що вказують на рівень впевненості.
    - **Brier Score**: Оцінює ймовірність передбачень, де менші значення означають більш точні і впевнені передбачення.

12. **Для чого потрібно використовувати батчі під час тренування? Чи завжди найбільший розмір батчу буде вести до найкращого результату?**
    Батчі дозволяють збалансувати обсяг даних, який обробляється за одну ітерацію, що допомагає зменшити споживання пам'яті і прискорити навчання.

    Найбільший розмір батчу не завжди дає найкращий результат. Великі батчі можуть призвести до недостатньо точних оновлень ваг і погіршення генералізації. Менші батчі, навпаки, додають стохастичність до навчання, що може допомогти уникнути локальних мінімумів.

13. **Як обирати значення K для алгоритму K-means?**

    - **Метод "коліна" (Elbow method)**: Малюється графік залежності функції втрат (інерції) від числа кластерів K. Оптимальне значення K відповідає "коліну" на графіку — точці, після якої зменшення функції втрат стає незначним.

- **Силуетний аналіз (Silhouette score)**: Оцінка якості кластеризації для різних значень K. Найкраще K — це те, при якому коефіцієнт силуету максимальний.

14. **Що таке стратифікація? Для чого вона в тестовому/валідаційному датасетах?Стратифікація** — це метод розподілу даних у тренувальний і тестовий набори так, щоб пропорції класів залишалися однаковими в обох наборах. Це особливо важливо для незбалансованих даних, де деякі класи значно переважають.

    Для чого використовується? Стратифікація допомагає забезпечити, що в тестовому і валідаційному наборах пропорції класів будуть такими ж, як у вихідному наборі даних. Це знижує ризик отримання некоректної оцінки моделі через невідповідність класів у тренувальному і тестовому наборах.

15. **Чи реалізували ви або використовували Machine Learning pipeline? Якщо так, будь ласка, опишіть його компоненти та переваги.**
    Так,
    ML pipeline автоматизує процеси підготовки даних і тренування моделей, що включає кілька основних етапів:

    - **Підготовка даних**: Пропуск, заповнення або масштабування даних.
    - **Вибір і трансформація ознак**: Створення нових ознак або вибір найбільш релевантних.
    - **Тренування моделі**: Навчання моделі на тренувальних даних.
    - **Оцінка моделі**: Крос-валідація або оцінка на тестовому наборі.
      Переваги:
    - **Модульність**: Компоненти можна легко змінювати або оновлювати.
    - **Автоматизація**: Спрощує повторювані завдання та забезпечує надійність.
    - **Повторюваність**: Гарантує однакові результати при однакових даних і параметрах.

16. **Які класи з бібліотеки sklearn можуть знадобитись під час підготовки даних?**

- **SimpleImputer**: Для заповнення пропущених значень.
- **StandardScaler**: Для масштабування ознак (зведення до стандартного нормального розподілу).
- **MinMaxScaler**: Для нормалізації ознак у діапазоні [0,1].
- **OneHotEncoder**: Для кодування категоріальних змінних у числові значення.
- **LabelEncoder**: Для кодування категорій у порядкові значення.
- **PolynomialFeatures**: Для створення нових поліноміальних ознак.

17. **Як називається процес, при якому характеристика (feature), що міститься в тренувальному наборі даних, не буде очікувано доступною в момент передбачення?**

    Це називається **data leakage** (витік даних). Витік даних виникає, коли під час навчання модель отримує інформацію, яка не буде доступною під час передбачень, що призводить до завищеної оцінки продуктивності.

18. **Під час створення характеристик ви отримали набір даних, що містить 575 стовпців. Чи варто додавати увесь набір даних до тренування? Якщо ні, то які ваші наступні дії? Якщо так, то як змінився б ваш підхід, скажімо, якщо є 57 або 575 стовпців?**

    Ні, не варто додавати всі 575 стовпців до тренування, оскільки це може призвести до високої кореляції ознак (мультиколінеарності), оверфіттингу і ускладнення моделі.

    Наступні дії:

    - Виконати **аналіз кореляції** між ознаками, щоб видалити високо корельовані ознаки.
    - Використати **методи відбору ознак** (як-от L1-регуляризація, Random Forest).
    - Використати **PCA** або інші методи зниження розмірності для зменшення кількості ознак, зберігаючи найбільш інформативні.

19. **Які підходи до перетворення текстових даних у числові ви знаєте?**

- **Bag of Words (BoW)**: Перетворення тексту в вектори за допомогою підрахунку частоти слів.
- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Модифікація BoW, яка враховує частоту появи слова в документі відносно всіх документів.
- **Word Embeddings**: Представлення слів у вигляді векторів з фіксованою довжиною (наприклад, Word2Vec, GloVe), що враховує семантичну близькість між словами.
- **Tokenization + Embedding Layers**: Використання токенізації тексту для тренування моделей глибокого навчання (наприклад, у нейронних мережах типу LSTM або Transformer).

▼ Model Building and Evaluation

1. **Опишіть різницю між алгоритмами за типом decision trees, SVM і random forests?**

   Decision Trees (Дерева рішень): Це алгоритм, який створює дерево, де кожна внутрішня вершина відповідає за перевірку певної ознаки, а листові вершини представляють рішення (класи).

   - Переваги: Інтерпретовані, працюють з числовими та категоріальними даними, не потребують нормалізації.
   - Недоліки: Схильні до оверфіттингу, якщо не застосовується регуляризація або обрізання дерева.

   SVM (Support Vector Machines): Алгоритм класифікації, який намагається знайти гіперплощину, що максимально розділяє класи. Підтримує використання ядрових функцій для нелінійних класифікацій.

   - Переваги: Добре працює для лінійно роздільних даних та з малими наборами даних.
   - Недоліки: Повільний для великих наборів даних, потребує налаштування гіперпараметрів (наприклад, ядра).

   Random Forest: Алгоритм ансамблевого навчання, який тренує кілька дерев рішень на різних підмножинах даних і усереднює їх передбачення.

- Переваги: Знижує ризик оверфіттингу, добре працює з великими та незбалансованими наборами даних.
- Недоліки: Менш інтерпретований, порівняно з одним деревом.

2. **Поясніть, навіщо потрібен hyperparameter tuning, опишіть деякі підходи.**

   Hyperparameter tuning — це процес налаштування гіперпараметрів моделі для досягнення найкращої продуктивності. Гіперпараметри — це параметри, які не навчаються під час тренування (наприклад, кількість дерев у Random Forest або параметр регуляризації в SVM).

   - **Grid Search**: Перебір усіх можливих комбінацій гіперпараметрів із заданого списку.
   - **Random Search**: Випадковий вибір гіперпараметрів із певних діапазонів.
   - **Bayesian Optimization**: Використовує ймовірнісну модель для визначення найбільш ймовірних комбінацій гіперпараметрів, які можуть покращити результат.
   - **Hyperband**: Адаптивний метод, що комбінує ідеї grid search і ранньої зупинки, щоб скоротити час пошуку.

3. **Яка різниця між bagging і boosting?**

   Bagging (Bootstrap Aggregating): Метод, який створює кілька незалежних моделей (наприклад, дерев рішень) на різних підмножинах даних, що створюються за допомогою випадкового вибіркового відбору з поверненням (bootstrapping). Потім їх передбачення усереднюються (або обирається найбільш популярний клас у випадку класифікації).

   - Переваги: Зменшує дисперсію моделі, добре працює з нестабільними моделями (наприклад, Decision Trees).
   - Приклад: Random Forest.

   Boosting: Алгоритм створює послідовність моделей, де кожна нова модель намагається виправити помилки попередньої. Результат всіх моделей об'єднується для остаточного передбачення.

- Переваги: Зменшує зміщення моделі, підходить для складних задач.
- Приклад: AdaBoost, XGBoost, LightGBM.

4. **Яким чином працювали б із time series даними?**
   Робота з time series включає специфічні кроки, оскільки дані впорядковані у часі:
   - **Попередня обробка**: Використання методів для заповнення пропусків (наприклад, інтерполяція) та нормалізація даних.
   - **Feature engineering**: Створення ознак на основі часових даних, як-от день тижня, година, лагові змінні або ковзне середнє.
   - **Крос-валідація**: Використання методів, як-от TimeSeriesSplit, для збереження послідовності даних.
   - **Моделювання**: Використання моделей, спеціально призначених для обробки часових рядів (ARIMA, SARIMA), або адаптація класичних алгоритмів (Random Forest, XGBoost) для прогнозування з використанням часових ознак.
   - **Оцінка**: Важливо використовувати такі метрики, як MAE або RMSE, для оцінки прогнозу на час серії.

▼ Deep Learning

1. **Що таке autoencoders? Для яких завдань їх використовують?**
   Autoencoders — це нейронні мережі, які використовуються для навчання ефективних представлень даних через процес кодування і декодування. Мережа навчається відновлювати вхідні дані після їх стискання у латентний простір меншої розмірності.
   - **Застосування**:
     - Стискання даних (dimensionality reduction).
     - Виявлення аномалій.
     - Шумопригнічення (denoising).
     - Генерація даних (variational autoencoders).

2. **Що таке проблема зникомого градієнта? Як вона може вплинути на навчання? Опишіть методи, які використовують для уникнення цієї проблеми.Зникомий градієнт** виникає, коли під час зворотного поширення помилки градієнти стають дуже малими і перестають оновлювати ваги, що призводить до того, що навчання зупиняється. Це особливо поширено в глибоких мережах.

    - **Методи боротьби**:
        - Використання активаційних функцій, як-от **ReLU**, які не мають проблеми з малими градієнтами.
        - Ініціалізація ваг за допомогою методів **He** або **Xavier**.
        - Використання **Batch Normalization**, що нормалізує вхідні дані і покращує стабільність градієнтів.

3. **Поясніть механізм уваги (attention).**

    Attention дозволяє моделі зосереджуватися на певних частинах вхідних даних під час передбачення. Замість того, щоб обробляти всі входи однаково, attention виділяє важливіші частини даних для кожного передбачення.

    Механізм: Модель генерує **ваги** для кожного входу, і ці ваги показують, яка частина даних має більший вплив на вихід. У трансформерах механізм self-attention допомагає моделі фокусуватися на різних частинах вхідного речення незалежно від його довжини.

4. **Що таке дистиляція в ML і навіщо її використовують? Поясніть головну ідею дистиляції.**

    Дистиляція знань — це процес передачі знань від великої і складної моделі (teacher) до меншої (student). Мета — зробити student-модель ефективнішою, зберігаючи продуктивність, подібну до teacher-моделі.

    Навіщо використовується: Для створення моделей, які менш ресурсозатратні та швидші у виробництві, зберігаючи при цьому високу точність.

5. **Як додати дропаут в LSTM?**

Дропаут можна додати до LSTM між різними шарами нейронів, щоб запобігти оверфіттингу.

В Keras та інших фреймворках є параметри для цього:

```
LSTM(units, dropout=0.2, recurrent_dropout=0.2)
```

- **Dropout**: Випадкове вимикання частини нейронів у шарі.
- **Recurrent Dropout**: Дропаут для рекурентних з'єднань у LSTM.

6. **Які підходи, моделі, метрики використовують для ranking задач?**

   Підходи:

   - **Pointwise**: Кожен документ оцінюється окремо.
   - **Pairwise**: Оцінюються пари документів для визначення їх відносного порядку.
   - **Listwise**: Оцінюються цілі списки документів.

   Моделі: RankNet, RankBoost, LambdaMART, XGBoost для ранжування.

   Метрики:

   - **NDCG (Normalized Discounted Cumulative Gain)**.
   - **MAP (Mean Average Precision)**.
   - **MRR (Mean Reciprocal Rank)**.

7. **Які способи зменшення потреби в даних вам відомі?**

   - **Transfer learning**: Використання попередньо натренованих моделей.
   - **Data augmentation**: Генерація нових зразків даних через трансформації.
   - **Синтетичні дані**: Створення нових даних за допомогою генеративних моделей (наприклад, GAN).
   - **Active learning**: Запит найбільш інформативних зразків для додаткового маркування.

8. **Як саме обираєте, з якими моделями працювати?**
   Вибір моделі залежить від кількох факторів:
   - **Тип задачі**: Класифікація, регресія, кластеризація.
   - **Розмір даних**: Простішим моделям підходять малі дані, а для великих даних — ансамблеві або нейронні мережі.
   - **Продуктивність**: Якщо потрібно отримати результат швидко, обираються менш обчислювально затратні моделі.
9. **Які методи оптимізації швидкості моделей знаєте?**
   - **Quantization**: Зменшення точності параметрів (перетворення з float32 на int8).
   - **Pruning**: Видалення непотрібних або маловпливових нейронів або зв'язків.
   - **Knowledge distillation**: Використання дистиляції знань для зменшення розміру моделей.
10. **Що таке метод Quantization-aware training та як він працює?**

    Quantization-aware training — це процес тренування моделі з урахуванням того, що під час інференсу модель буде використовувати кількісну квантовану арифметику. Це дозволяє зберегти точність після квантованої конверсії.
11. **Розкажіть про принцип роботи трансформерів.**

    Трансформери використовують механізм **self-attention** для моделювання залежностей між елементами послідовності, дозволяючи паралельно обробляти вхідні дані. Основні компоненти трансформера:
    - **Механізм уваги**: Обчислює ваги для кожного елемента послідовності.
    - **Position encoding**: Додає інформацію про позиції елементів у послідовності, оскільки трансформери не використовують рекурентність.

12. **Чому ми не можемо використовувати оптимізатори другого порядку для задач Deep Learning?**

    Оптимізатори другого порядку, як-от метод Ньютона, обчислюють Гессіан (матрицю других похідних), що є обчислювально затратним для великих моделей. Це робить їх непридатними для використання в глибоких нейронних мережах, де кількість параметрів може бути дуже великою.

13. **Яку проблему вирішує Cycle Policy? Що таке One Cycle?**

    One Cycle Policy — це метод налаштування швидкості навчання, де швидкість спочатку збільшується, а потім зменшується протягом одного циклу тренування. Це допомагає моделі уникнути локальних мінімумів та пришвидшує процес збіжності.

14. **Як підбирати значення LR для навчання?**

    Один із підходів — використовувати **LR range test**, де швидкість навчання спочатку встановлюється дуже малою і поступово збільшується. Графік залежності функції втрат від швидкості навчання допомагає знайти оптимальну початкову швидкість.

15. **Що таке skip-connections у нейронних мережах і навіщо вони?**

    Skip-connections (або **residual connections**) — це прямі з'єднання, що пропускають кілька шарів мережі, передаючи інформацію далі без змін. Це допомагає уникнути проблеми зникомого градієнта і дозволяє тренувати дуже глибокі мережі, як-от ResNet.

▼ MLOps

1. **Назвіть найкращі практики версіонування моделей.**
    - **Використання систем контролю версій** (наприклад, Git) для коду, що використовується для тренування моделей.
    - **Версіонування моделей** з використанням унікальних ідентифікаторів або хешів для кожної версії.
    - **Фіксування гіперпараметрів і даних**, на яких тренувалася модель, як частина процесу версіонування.

- **Ведення журналу тренування моделей** (наприклад, з використанням MLflow або DVC) для відстеження результатів і експериментів.
- **Документування версій**: Важливо зберігати всі метадані, пов'язані з моделлю, такі як вхідні ознаки, гіперпараметри та результати на тестових наборах.

2. **Розкажіть про концепт data shifts. Як його можна виявляти?Data shifts** — це зміни у розподілі даних між тренувальним і виробничим середовищем. Це може призвести до погіршення продуктивності моделі, оскільки вона навчається на даних, які більше не відображають актуальну ситуацію.
   - **Види data shifts**:
     - **Covariate shift**: Зміна розподілу вхідних ознак, але розподіл цільової змінної залишається незмінним.
     - **Label shift**: Зміна розподілу цільової змінної, але вхідні ознаки залишаються тими ж.
     - **Concept drift**: Зміна взаємозв'язків між ознаками і цільовою змінною.
   - **Як виявляти**:
     - Відстеження змін у статистиці розподілу ознак або цільової змінної.
     - Використання моделей для виявлення дрейфу даних (наприклад, класифікатор для виявлення розбіжностей між тренувальними та тестовими наборами).
3. **Чи працювали ви з технологіями контейнеризації, такими як Docker? Будь ласка, опишіть ваш досвід.**
   Так, Docker — це платформа для контейнеризації, яка дозволяє упаковувати додаток та всі його залежності в ізольоване середовище (контейнер).

   Мій досвід:

- Створення Docker-образів для тренування і розгортання моделей машинного навчання.
- Використання Docker для забезпечення узгодженості середовищ між розробкою і виробництвом.
- Створення `Dockerfile` для автоматизації процесу контейнеризації моделей.

4. **Чи працювали ви з системами оркестрації контейнерів, такими як Kubernetes? Будь ласка, опишіть ваш досвід.Kubernetes** — це система оркестрації контейнерів, яка дозволяє автоматизувати розгортання, масштабування і управління контейнеризованими додатками.

   Мій досвід:

   - Розгортання моделей машинного навчання в кластері Kubernetes для забезпечення масштабованості і високої доступності.
   - Використання Helm для управління конфігураціями і налаштуваннями Kubernetes.
   - Моніторинг кластерів і контейнерів з використанням Prometheus та Grafana.

5. **Як ви управляєте версіонуванням і відстеженням моделей в колаборативному середовищі?**

   - **MLflow**: Відстеження експериментів, збереження моделей та їх версій.
   - **DVC (Data Version Control)**: Версіонування даних та моделей, інтеграція з Git.
   - **Git**: Версіонування коду і метаданих, пов'язаних з моделлю.
   - **Документація і журнал експериментів**: Важливо документувати всі налаштування моделей і гіперпараметри для полегшення співпраці.

6. **Поясніть роль feature stores і data versioning в операціях з машинного навчання.**

Feature store: Це репозиторій, де зберігаються ознаки (features) для моделі. Він дозволяє:

- Використовувати одні й ті ж ознаки для тренування і передбачення, знижуючи ризик inconsistency.
- Спрощувати повторне використання ознак у різних моделях.

Data versioning: Важливе для забезпечення узгодженості між тренувальними і тестовими даними, особливо якщо дані змінюються. Це дозволяє відтворювати результати та контролювати, які дані використовувалися для тренування певної версії моделі.

7. **Поясніть переваги та виклики використання serverless-архітектури для розгортання моделей машинного навчання в «хмарі».**

   Переваги:

   - **Масштабованість**: Хмарні платформи автоматично масштабують ресурси відповідно до навантаження.
   - **Ефективність**: Оплата тільки за фактичне використання ресурсів, що знижує витрати.
   - **Швидкість розгортання**: Моделі можуть бути розгорнуті швидко, без необхідності налаштування інфраструктури.

   Виклики:

   - **Обмеження за часом виконання**: У деяких serverless-сервісах існують обмеження на тривалість виконання функцій.
   - **Менше контролю**: Обмежений контроль над середовищем виконання і його конфігураціями.
   - **Латентність**: Випадкові затримки (cold starts), що можуть вплинути на швидкість інференсу моделі.

▼ Computer Vision

1. **Опишіть архітектуру Faster R-CNN і YOLO. У чому між ними різниця?**

- **Faster R-CNN**: Це двоетапний детектор об’єктів. Спочатку генерується набір регіонів, які можуть містити об’єкти (Region Proposal Network, RPN), а потім ці регіони класифікуються і визначаються їхні межі.
    - Плюси: Висока точність, гарний результат для складних завдань.
    - Мінуси: Повільніший у порівнянні з YOLO через двоетапний процес.
- **YOLO (You Only Look Once)**: Одноетапний детектор, який одночасно класифікує і визначає межі для кількох об’єктів на зображенні.
    - Плюси: Дуже швидкий, підходить для реального часу.
    - Мінуси: Менш точний порівняно з Faster R-CNN, особливо для малих об’єктів.

2. **Яка різниця між one-stage і two-stage object detector? Які плюси та мінуси кожного з них?**
    - **One-stage detectors** (наприклад, YOLO, SSD): Прямо прогнозують класи і координати об’єктів за одне проходження мережі.
        - Плюси: Швидкі, підходять для задач реального часу.
        - Мінуси: Менша точність, особливо для складних сцен.
    - **Two-stage detectors** (наприклад, Faster R-CNN): Спочатку генерують регіони інтересу (proposals), а потім уточнюють класифікацію і координати.
        - Плюси: Вища точність, особливо для складних сцен.
        - Мінуси: Повільніші в порівнянні з одноетапними детекторами.
3. **Розкажіть, як рахують mAP-метрику для задачі Object Detection? Що таке IOU?**
    - **mAP (mean Average Precision)** — це середнє значення точності (precision) для різних порогів перетину між прогнозованими і справжніми об'єктами.

- **IOU (Intersection over Union)** — це відношення площі перетину між передбаченими і справжніми обмежувальними рамками до площі їх об’єднання. IOU використовується для оцінки якості виявлення об'єкта.

4. **Розкажіть, як рахують MS COCO mAP@IOU=0.5:0.95:0.05? В чому перевага метрики над VOC Pascal mAP?**
    - **MS COCO mAP** обчислюється для діапазону IOU від 0.5 до 0.95 з кроком 0.05. Це робить метрику суворішою і більш детальною.
    - **VOC Pascal mAP** використовує фіксований поріг IOU = 0.5, що може бути занадто слабким критерієм для точних задач. MS COCO mAP більш точна через варіацію порогів IOU.
5. **Як пришвидшити object detection модель?**
    - Використовувати **легші архітектури** (наприклад, MobileNet).
    - **Pruning і quantization** для зменшення розміру моделі і прискорення виконання.
    - Використовувати **FP16 або int8 precision** замість FP32.
    - Оптимізувати модель для апаратних прискорювачів, таких як GPU або TPU.
6. **Розкажіть, як працює U-Net?U-Net** — це архітектура для сегментації зображень. Вона складається з двох частин:
    - **Енкодер**: Зменшує розмір зображення і витягує найважливіші ознаки.
    - **Декодер**: Відновлює розмір зображення, одночасно зберігаючи важливу інформацію за допомогою з'єднань (skip connections) з енкодером.
7. **Які метрики використовують для сегментації? Dice, IOU?**
    - **Dice coefficient**: Оцінює схожість між передбаченою та реальною маскою об'єкта.
      Формула:
      \[

Dice = \frac{2 \times TP}{2 \times TP + FP + FN}
\]

- **IOU** (Intersection over Union): Оцінює перекриття між передбаченою і реальною маскою, як у задачах Object Detection.

8. **Навіщо потрібен дискримінатор в GAN? Що буде, якщо дискримінатор буде ідеальним?**
   У
   GAN (Generative Adversarial Network) дискримінатор намагається відрізнити справжні зображення від згенерованих, тоді як генератор намагається обманути дискримінатор, створюючи правдоподібні зображення.

   Якщо дискримінатор стане ідеальним, він завжди зможе відрізнити реальні зображення від фейкових, і генератор не зможе покращувати якість своїх зображень, оскільки дискримінатор не дає корисного зворотного зв'язку.

9. **Розкажіть про ProgressGAN. Як за допомогою нього генерувати зображення високої якості з деталями?**

   Progressive GAN — це варіант GAN, де модель тренується поступово, починаючи з низької роздільної здатності і поступово додаючи вищі рівні деталізації. Спочатку тренується генерація базової структури, потім модель додає деталі. Це дозволяє ефективніше вчити моделі для генерації зображень високої якості.

10. **Розкажіть про Cycle GAN. Як за допомогою нього можна вирішити проблему, коли немає точної відповідності між вхідною та вихідною картинками?**

    Cycle GAN дозволяє трансформувати зображення з одного домену в інший без необхідності наявності пар зображень (наприклад, перетворення стилю). Він працює шляхом застосування двох генераторів і двох дискримінаторів: один для перетворення A → B, а інший для перетворення B → A. Циклічна втрачена функція гарантує, що перетворені зображення можуть бути повернені назад до вихідних.

11. **Розкажіть про StyleGAN. Як за допомогою нього генерувати зображення з потрібним стилем?**

StyleGAN використовує розділення змісту зображення і стилю. Замість простого кодування векторів зображень, StyleGAN вводить поняття **style vectors**, які відповідають за різні рівні абстракції (низькорівневі риси, такі як текстури, і високорівневі, як форма об'єктів). Це дозволяє контролювати стиль зображень, змінюючи ці style vectors на різних етапах генерації.

12. **Які метрики використовують для перевірення якості GANs?**
    - **Inception Score (IS)**: Оцінює якість і різноманітність згенерованих зображень, використовуючи попередньо натреновану модель Inception. Високий бал вказує на різноманітність і реалістичність зображень.
    - **Frechet Inception Distance (FID)**: Порівнює статистику реальних і згенерованих зображень. Чим нижчий FID, тим ближче розподіли згенерованих зображень до реальних.
    - **Precision and Recall for GANs**: Оцінює здатність моделі генерувати реалістичні зображення (precision) і різноманітність (recall).
13. **Чи можливо створити класифікатор на 100 тисяч класів? Обґрунтуйте свою думку.**
    Так, можливо створити класифікатор на 100 тисяч класів, але для цього потрібні спеціальні підходи:
    - Використання **архітектур зі зменшенням розмірності** (наприклад, CNN або трансформери).
    - **EfficientNet** або **MobileNet** можуть бути ефективними завдяки своїй здатності масштабуватися.
    - Проблемою є **незбалансованість даних**: не всі класи матимуть однакову кількість зразків.
    - Можна використовувати методи **softmax зі sparsity**, або інші варіанти багатокласової класифікації (наприклад, ієрархічні класифікатори).
14. **Розкажіть про Triplet loss? У чому недоліки?**

Triplet loss використовується для навчання моделей, які повинні розрізняти схожі і відмінні зразки. Функція втрат мінімізує відстань між прикладами з одного класу і максимізує відстань між прикладами з різних класів.

Недоліки:

- Необхідність складного процесу підбору трійок (anchor, positive, negative).
- Може бути неефективним для великих наборів даних, оскільки потрібно генерувати багато трійок для навчання.

15. **Навіщо в Re-ID фічі проєктуються на сферу?**

    У задачах **Re-ID (Re-identification)** використовують проєкцію фіч на сферу для нормалізації відстаней між ознаками, що полегшує порівняння вектори ознак незалежно від їх масштабу. Це допомагає поліпшити результати в задачах, де важлива точна ідентифікація об'єктів, таких як розпізнавання осіб або транспортних засобів.

16. **Розкажіть про функцію CosFace? Навіщо її використовують та які її переваги для Re-ID?**

    CosFace — це функція втрат для задач класифікації та Re-ID, яка оптимізує косинусну відстань між векторами ознак. Вона додає margin до косинусної подібності для покращення відокремлюваності класів.

    Переваги:

    - Покращує відокремлюваність класів.
    - Підвищує точність в задачах, де важлива максимальна дискримінативність ознак (наприклад, розпізнавання осіб).

17. **Розкажіть про архітектуру VGG? Які її особливості? Що було новітнього представлено в ній свого часу?VGG** — це архітектура нейронної мережі, яка складається з багатьох послідовних згорткових шарів, кожен з яких використовує невеликі фільтри розміром 3x3.

    Особливості:

    - Використання глибокої архітектури зі значною кількістю шарів.

- Постійне використання невеликих фільтрів 3x3.

Новітнє на момент представлення: VGG стала однією з перших глибоких мереж, що продемонструвала важливість збільшення глибини для підвищення точності класифікації зображень.

18. **Розкажіть про Skip Connection в архітектурі ResNet? Яку проблему він вирішує?**

    Skip Connections в ResNet дозволяють пропускати шари через прямі зв'язки, додаючи вихід попереднього шару до наступного шару. Це вирішує проблему **зникомого градієнта** в глибоких мережах і дозволяє тренувати дуже глибокі моделі, зберігаючи стабільність градієнтів під час зворотного поширення помилки.

19. **Розкажіть про depth-wise separable convolution в архітектурі MobileNet? Яку роль він відіграє?**

    Depth-wise separable convolution розділяє згортку на два кроки: **depth-wise convolution** (згортка для кожного каналу окремо) і **point-wise convolution** (згортка 1x1 для об'єднання каналів).

    - **Роль**: Значно зменшує кількість параметрів і обчислень, що дозволяє використовувати MobileNet для мобільних та вбудованих пристроїв із обмеженими ресурсами.

20. **Розкажіть про RELU6 в архітектурі MobileNet? Яку роль він відіграє? RELU6** — це варіант функції активації ReLU, обмежений значенням 6.

    - **Роль**: Допомагає уникати великих значень активацій, що може бути корисним у випадках, коли модель працює з апаратним прискоренням (наприклад, TPU), де обмежений діапазон значень.

21. **Розкажіть про inverted residual block в архітектурі MobileNet v2? Як він влаштований?Inverted residual block** — це блок у MobileNet v2, де інформація спочатку проходить через ширший простір (більше каналів), а потім стискається до меншої кількості каналів.

    - Це зворотний підхід до класичних ResNet-блоків, де інформація спочатку стискається, а потім розширюється.

- **Перевага**: Зменшення обчислювальної складності і збереження інформації в ширшому просторі для ефективного навчання.

22. **Розкажіть про Inception Block? Як він влаштований в архітектурах Inception, Inception V2, Inception V3? Чому саме така архітектура?**
    Inception Block використовує кілька згорток з різними розмірами фільтрів (1x1, 3×3, 5×5) і об'єднує їх для забезпечення різних рівнів абстракції.

    - **Inception V2**: Використовує додаткові техніки, як-от **factorized convolutions**, щоб зменшити кількість параметрів.
    - **Inception V3**: Покращена версія з додатковими техніками для прискорення обчислень і покращення точності, такими як **Batch Normalization**.
    - **Чому така архітектура?** Це дозволяє моделі ефективно захоплювати ознаки різного масштабу, використовуючи фільтри різного розміру в межах одного блоку, що робить її універсальною для різних типів об'єктів.

23. **Що таке NAS?**

    NAS (Neural Architecture Search) — це автоматизований метод пошуку найкращої архітектури нейронної мережі для конкретного завдання. Він використовує алгоритми пошуку (наприклад, еволюційні алгоритми або reinforcement learning), щоб знаходити оптимальні архітектури, замість того щоб проєктувати їх вручну.

24. **Розкажіть про архітектуру EfficientNet. Які технічні рішення стоять за вибором параметрів для архітектури?**

    EfficientNet — це архітектура, яка використовує метод масштабування мережі (Compound Scaling), що дозволяє збільшувати розмір мережі, її глибину і ширину одночасно, дотримуючись балансу між ними.

    Технічні рішення:

    - Використання **baseline-моделі** EfficientNet-B0, яка оптимізована за допомогою NAS.

- **Compound Scaling** збільшує архітектуру рівномірно по всіх осях (глибина, ширина, розширення зображень).

25. **Розкажіть, як працює Knowledge Distillation? Що таке NoisyStudent Networks?Knowledge Distillation** — це метод, при якому велика модель (teacher) навчає меншу модель (student), передаючи їй свої знання через "м'які" мітки (probabilities) замість жорстких передбачень.

    NoisyStudent — це вдосконалена версія distillation, де student додає шум до своїх входів і результатів, що допомагає покращити генералізацію моделі.

26. **Розкажіть, як працює Pruning у нейромережах? Які його переваги та недоліки?Pruning** — це техніка, яка зменшує кількість параметрів у нейронній мережі шляхом видалення маловажливих нейронів або з'єднань.

    Переваги:

    - Зменшує розмір моделі і час виконання.
    - Ефективніше використовує ресурси.

    Недоліки:

    - Може призвести до втрати точності.
    - Потрібна додаткова обробка для досягнення збалансованої продуктивності.

27. **Чим відрізняється Pruning за вагами і за з'єднаннями? Які їхні переваги та недоліки?**

    - **Pruning за вагами**: Видаляє окремі ваги, що мають малі значення.
        - **Перевага**: Зберігає структуру мережі.
        - **Недолік**: Складніше реалізувати на апаратному рівні.
    - **Pruning за з'єднаннями**: Видаляє цілі з'єднання (канали або нейрони).
        - **Перевага**: Простота реалізації на апаратному рівні.
        - **Недолік**: Може значно зменшити точність.

28. **Розкажіть про квантизацію. У чому різниця між per-channel і per-tensor?**

    Квантизація — це процес зменшення точності чисел у нейронній мережі (зазвичай з float32 до int8), щоб зменшити розмір моделі і пришвидшити обчислення.

    - **Per-channel quantization**: Квантизація виконується окремо для кожного каналу.
    - **Per-tensor quantization**: Квантизація виконується для всього тензора одночасно.
    - **Різниця**: Per-channel забезпечує кращу точність, оскільки кожен канал має свою шкалу.

29. **Навіщо робити label smoothing для класифікації?**

    Label smoothing — це техніка регуляризації, яка змінює "жорсткі" мітки класів (наприклад, 1 або 0) на "м'які" ймовірності (наприклад, 0.9 і 0.1). Це допомагає моделі не бути занадто впевненою у своїх передбаченнях і зменшує ризик оверфіттингу.

30. **Розкажіть про ViT? Що таке токен?**

    ViT (Vision Transformer) — це модель для обробки зображень, яка адаптує архітектуру трансформерів з NLP до задач комп'ютерного зору.

    Токен у ViT — це частина зображення (зазвичай, патч), яка обробляється як одиниця вхідних даних для трансформера.

31. **Чим відрізняється робота з картинками та відео? Які челенджі виникають у роботі з відео?**

    Основна відмінність: Відео містить не лише просторову інформацію, але й часову.

    Челенджі:

    - Обробка послідовності кадрів потребує великих ресурсів.
    - Важливо зберегти зв'язок між кадрами (часову залежність).
    - Висока складність у зборі та анотації відеоданих.

32. **Чи потрібні усі фрейми з відео? Як обирати їх?**
    Не всі фрейми можуть бути необхідні, особливо якщо кадри мало відрізняються один від одного.

    Методи вибору:

    - Вибір **кожного N-го фрейму**.
    - Використання **motion detection** для вибору тільки тих фреймів, де є значні зміни.

33. **Як працюють алгоритми стиснення відео? H264 кодек?**

    Алгоритми стиснення відео використовують техніки стиснення без втрат (intra-frame) і зі втратами (inter-frame).

    H264: Це один із найпоширеніших кодеків для стиснення відео, який використовує як внутрішнє стиснення (intra-frame), так і міжкадрове стиснення (inter-frame) для зменшення розміру відеофайлів, зберігаючи прийнятну якість.

34. **Які задачі може розв'язувати CV (Computer Vision)?**

    - **Класифікація зображень**: Визначення категорії об'єкта на зображенні.
    - **Виявлення об'єктів**: Локалізація і класифікація об'єктів у зображенні (наприклад, YOLO, Faster R-CNN).
    - **Сегментація зображень**: Розбиття зображення на окремі частини або об'єкти (semantic і instance segmentation).
    - **Відстеження об'єктів**: Відстеження рухомих об'єктів у відео.
    - **Розпізнавання осіб і об'єктів**: Використання для біометрії або ідентифікації об'єктів.
    - **Аналіз і покращення зображень**: Видалення шуму, збільшення роздільної здатності, покращення якості.

35. **Які типи попередньої обробки зображень ви знаєте?**

    - **Масштабування (resizing)**: Зміна розміру зображення.
    - **Нормалізація**: Перетворення пікселів в інтервал [0,1] або [-1,1].

- **Згладжування**: Фільтрація шумів, наприклад, гауссовий фільтр.
- **Аугментація**: Трансформації (фліп, обертання, зсув), що збільшують варіативність даних.
- **Центрування**: Вирівнювання зображень для видалення зсуву або перекосів.

36. **Що таке SIFT, SURF?**
    - **SIFT (Scale-Invariant Feature Transform)**: Метод для виявлення й опису локальних ознак на зображеннях. Він стійкий до масштабування, повороту і змін освітлення.
    - **SURF (Speeded Up Robust Features)**: Прискорена версія SIFT, що також використовується для виявлення ключових точок і опису ознак, але працює швидше.
37. **Яка різниця між виявленням об'єкта та відстежуванням?**
    - **Виявлення об'єкта (Object detection)**: Завдання знаходження і класифікації об'єктів на окремих зображеннях.
    - **Відстеження об'єкта (Object tracking)**: Завдання знаходження і відстежування рухомого об'єкта на послідовності зображень або відео.
38. **Що таке семантична сегментація (Semantic segmentation)?**

    Семантична сегментація — це завдання розбиття зображення на області, що відповідають певним класам (наприклад, людина, автомобіль, дорога), де кожному пікселю присвоюється певний клас. На відміну від instance segmentation, сегментація не відрізняє окремі об'єкти одного класу.
39. **Як працюють моделі переведення зображення в текст?**

    Моделі переведення зображення в текст використовують комбінацію нейронних мереж:
    - **CNN (Convolutional Neural Networks)**: Для витягання ознак із зображення.

- **RNN або Transformers**: Для генерації тексту на основі витягнутих ознак.
- Наприклад, моделі **Image Captioning** можуть генерувати опис зображення на основі виділених CNN ознак і подальшого текстового генератора.

40. **Як працюють генеративні моделі (наприклад, DALL-E)?**

    DALL-E — це генеративна модель, яка використовує трансформери для створення зображень на основі текстового опису. Модель вчиться взаємозв'язку між текстом і зображеннями і генерує нові зображення, що відповідають введеним текстовим підказкам. Вона використовує масштабовані моделі (як-от GPT) для генерації зображень з нуля.

41. **У вас є завдання створити помічника для аналізу зображення рентгену. Якими будуть ваші кроки?**

    - **Збір даних**: Отримати рентгенівські зображення з анотаціями.
    - **Попередня обробка**: Застосувати методи нормалізації, видалення шуму, аугментацію зображень для збільшення набору даних.
    - **Вибір моделі**: Використати попередньо натреновану модель (наприклад, ResNet або DenseNet), адаптовану для медичних зображень.
    - **Тренування**: Навчити модель на даних з рентгенівськими зображеннями, використовуючи відповідні метрики (наприклад, AUC, precision, recall).
    - **Оцінка**: Перевірити продуктивність моделі на тестових даних і оцінити за допомогою метрик класифікації.
    - **Розгортання**: Інтегрувати модель у систему аналізу медичних зображень для допомоги лікарям у діагностиці.

▼ Алгоритми

1. **Поясніть різницю між рекурсією та ітерацією в контексті програмування. Коли краще використовувати той чи інший підхід?**

   Рекурсія — це метод, коли функція викликає сама себе для розв'язання підзадачі. Використовується для задач, які можна розбити

на менші підзадачі того ж типу. Кожен рекурсивний виклик зберігається в стеку викликів, і результат обчислюється шляхом повернення з кожного рівня рекурсії.

- **Переваги**: Проста для розуміння і реалізації в задачах, де природно використовувати розбиття на підзадачі (факторіал, обхід дерев, розв'язання задач на графах).
- **Недоліки**: Використовує багато пам'яті через збереження станів у стеку викликів. Може призводити до переповнення стеку (stack overflow) у випадку дуже глибокої рекурсії.

Ітерація — це підхід, коли задачу розв'язують за допомогою циклів, таких як `for` або `while`, для повторення певних дій до досягнення необхідного результату.

- **Переваги**: Менш витратна з точки зору пам'яті, оскільки не зберігаються проміжні стани. Краще підходить для великих задач, де потрібна велика кількість повторень.
- **Недоліки**: Може бути складніше для розуміння, якщо задачу природніше описати через рекурсію.

Коли використовувати рекурсію?

- Коли задача природно розбивається на менші підзадачі, наприклад, при обході дерев (DFS) або для алгоритмів розподілу (divide and conquer), таких як QuickSort або MergeSort.

Коли використовувати ітерацію?

- Коли ви працюєте із задачами, які включають повторювані обчислення з постійним станом, наприклад, обчислення сум, добутків або лінійних пошуків. Ітерація є кращою для оптимізації використання пам'яті.

▼ NLP

1. **Які фреймворки для NLP вам знайомі?**

- **NLTK (Natural Language Toolkit)**: Бібліотека для обробки текстів, аналізу, лемматизації, стемінгу, токенізації.
- **spaCy**: Високошвидкісна бібліотека з вбудованою підтримкою багатьох моделей і мов, добре підходить для промислового застосування.
- **Transformers (Hugging Face)**: Бібліотека для роботи з великими мовними моделями (GPT, BERT, T5).
- **Gensim**: Бібліотека для моделювання тем і обчислення текстових подібностей.
- **StanfordNLP**: Бібліотека для синтаксичного аналізу і розпізнавання залежностей.

2. **Що таке Syntactic Analysis?**

   Синтаксичний аналіз (або **парсинг**) — це процес аналізу граматичної структури речення, щоб визначити відносини між словами (суб'єкт, присудок, додаток тощо). Його мета — зрозуміти граматичні зв'язки в реченні. Часто використовується для побудови дерев залежностей (dependency trees).

3. **Що таке лемматизація (lemmatization) і стемінг (stemming)? Для чого їх використовують?**

   - **Лемматизація** — це процес перетворення слова до його базової форми (леми) з урахуванням контексту і граматики. Наприклад, "running" перетворюється на "run".
   - **Стемінг** — це простіший процес, при якому до слова застосовуються евристичні правила для видалення суфіксів. Наприклад, "running" перетворюється на "runn".
   - **Використовують для**: Покращення роботи алгоритмів пошуку та текстової обробки, зменшення кількості форм одного слова для підвищення точності аналізу.

4. **Що таке BoW, TF-IDF, BM25 та де їх використовують?**

   - **BoW (Bag of Words)**: Представляє текст як набір слів без урахування їхнього порядку. Використовується для класифікації

текстів або аналізу частотності.

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Враховує частоту слова в документі та у всіх документах, щоб підвищити вагу важливих слів і знизити вагу частих.
- **BM25**: Удосконалена версія TF-IDF, яка краще оцінює релевантність слів у документі та широко використовується в інформаційних системах для пошуку текстів.

5. **Що таке токенізація (tokenization)? Які її типи ви знаєте?Токенізація** — це процес поділу тексту на менші частини (токени), такі як слова, речення або навіть окремі символи.

   Типи токенізації:

   - **Word-level tokenization**: Поділ на слова.
   - **Sentence-level tokenization**: Поділ на речення.
   - **Character-level tokenization**: Поділ на окремі символи.
   - **Subword tokenization**: Використовується в трансформерах для поділу слів на менші частини (наприклад, Byte-Pair Encoding, WordPiece).

6. **Що таке word2vector? Що означає «sentence to a vector»? Де це поняття використовують?**

   Word2Vec — це модель для навчання векторних представлень слів (embeddings), де слова зі схожим значенням мають близькі вектори.

   - **Sentence to a vector**: Подібно до word embeddings, для речень можна генерувати вектори, що представляють значення речення. Ці вектори використовуються в задачах класифікації, пошуку або порівняння текстів.
   - Використовується у задачах **класифікації тексту**, **синтаксичного аналізу** або **моделювання текстових залежностей**.

7. **Чому ембедінги слів, отримані від трансформерів, кращі за класичний word2Vec?**

Ембедінги, отримані від трансформерів (наприклад, BERT), контекстуальні. Це означає, що вони враховують сусідні слова і контекст для кожного слова. На відміну від **Word2Vec**, який створює фіксовані вектори для кожного слова, незалежно від контексту, трансформери забезпечують динамічне представлення для слів залежно від їхнього використання в реченні.

8. **Як би ви розв'язали задачу аналізу твітів за емоційним забарвленням? У вас є два варіанти: сучасний підхід (LLMs) або створення власної системи з нуля.**

    - **Сучасний підхід (LLMs)**: Використати попередньо натреновані моделі (наприклад, BERT або GPT) з fine-tuning для класифікації емоцій. Це швидше і забезпечує високу точність.
    - **Створення власної системи з нуля**: Побудувати модель з використанням класичних методів (TF-IDF + SVM або прості RNN/LSTM), що потребує більше часу, але може бути адаптовано до специфіки даних. Застосувати класичну токенізацію, побудову векторів і навчання моделі.

9. **Що таке пояснювальне ML (explainable ML)? Як використовувати підхід to make a black-box a gray one?**

    Explainable ML — це підхід, який дозволяє зрозуміти, як саме модель приймає рішення, особливо коли вона є "чорною скринькою" (black-box). Це важливо для довіри до результатів моделі.

    Інструменти для пояснення:

    - **LIME (Local Interpretable Model-Agnostic Explanations)**: Генерує пояснення для передбачення моделі, створюючи прості локальні моделі навколо окремих прогнозів.
    - **SHAP (SHapley Additive exPlanations)**: Оцінює внесок кожної ознаки до результату передбачення.
    - **Gray-box**: Створення "сірого" ящика через використання цих методів для надання зрозумілих інтерпретацій окремих рішень моделі.

10. **Якби ви реалізували dropout і batch normalization для нейронної мережі? Коли це важливо?**

    - **Dropout**: Виключає випадкові нейрони під час тренування, щоб уникнути оверфіттингу. Використовується після повнозв'язних або згорткових шарів.
    - **Batch normalization**: Нормалізує вхідні дані для кожного шару, що прискорює навчання і стабілізує градієнти. Використовується після кожного шару перед активацією.
    - **Важливість**: Dropout важливий для регуляризації, а batch normalization — для прискорення і стабілізації тренування.

▼ Розгортання моделі

1. **Які фактори потрібно враховувати під час розгортання моделі?**

    - **Продуктивність**: Модель повинна відповідати вимогам до часу виконання та ресурсоємності.
    - **Масштабованість**: Модель має бути здатна обробляти великі обсяги даних у виробничих умовах, особливо при високих навантаженнях.
    - **Моніторинг**: Необхідно передбачити системи моніторингу продуктивності та точності моделі.
    - **Оновлення**: Повинна бути можливість легкої заміни або оновлення моделі без зупинки системи.
    - **Стабільність**: Модель повинна працювати стабільно в різних виробничих середовищах.
    - **Безпека**: Необхідно враховувати аспекти безпеки, такі як захист даних і забезпечення того, що модель не буде легко піддаватися атакам (наприклад, атаки на моделі через підроблені дані).

2. **Як ви будете відстежувати модель машинного навчання?**

    - **Моніторинг продуктивності**: Відстеження часу відгуку моделі, кількості запитів, споживання ресурсів (CPU, GPU, RAM).

- **Моніторинг точності**: Відстеження точності, precision, recall, F1-score для поточних даних, порівняння з історичними результатами.
- **Виявлення дрейфу даних (data drift)**: Оцінка змін у розподілах вхідних даних або результатів, які можуть вказувати на необхідність повторного тренування моделі.
- **Логи та алерти**: Налаштування журналів і повідомлень для автоматичного сповіщення при виявленні аномалій у роботі моделі.
- **Використання інструментів**: Інтеграція з MLOps-платформами (наприклад, MLflow, Prometheus, Grafana) для автоматичного моніторингу і логування.

3. **Як можна налаштувати модель для обробки зміни концепції, після її розгортання?Зміна концепції (concept drift)** — це зміна відносин між вхідними даними і виходами моделі, що призводить до погіршення продуктивності моделі. Для обробки цієї проблеми можна:
   - **Відстеження концептуальних змін**: Постійно порівнювати нові вхідні дані з тренувальними та оцінювати точність моделі.
   - **Адаптивні моделі**: Використовувати алгоритми, які можуть динамічно адаптуватися до нових даних без повного перевчення (наприклад, онлайн-навчання).
   - **Регулярне повторне тренування (retraining)**: Планувати періодичне перевчення моделі на нових даних, зберігаючи оновлену версію моделі.
   - **A/B-тестування**: Можна використовувати кілька версій моделі та періодично оцінювати їхню продуктивність на нових даних.
   - **Ensemble learning**: Використання ансамблевих моделей, де нові моделі додаються до старих, що дозволяє зменшити вплив зміни концепції.

▼ Python

1. **Які фреймворки для ML ви знаєте та вмієте використовувати? Розкажіть про найбільш успішний проєкт, завершений на кожному з**

цих фреймворків (щонайменше TensorFlow, PyTorch, scikit-learn).

TensorFlow: Це один із найпопулярніших фреймворків для глибокого навчання, який підтримує як обчислення на CPU, так і на GPU. TensorFlow використовується для створення великих нейронних мереж, у тому числі для комп'ютерного зору, NLP та інших задач.

- **Проєкт**: **Моделювання розпізнавання об'єктів на зображеннях**. За допомогою TensorFlow була створена модель для виявлення та класифікації об'єктів у реальному часі. Модель тренувалася на основі архітектури Faster R-CNN та використовувала предтреновані ваги для покращення результатів.

PyTorch: Цей фреймворк забезпечує зручніший API та більш гнучкий підхід до створення нейронних мереж, що робить його популярним серед дослідників. PyTorch дозволяє легко реалізовувати експерименти завдяки динамічним обчислювальним графам.

- **Проєкт**: **Реалізація моделі для аналізу емоцій у текстах**. Застосовуючи трансформери (BERT) на PyTorch, була створена модель для аналізу емоційного забарвлення текстів (позитивні, негативні та нейтральні). Модель була натренована на спеціалізованому наборі даних і досягла високої точності класифікації.

scikit-learn: Це фреймворк для класичного машинного навчання, що містить готові алгоритми для класифікації, регресії, кластеризації, зменшення розмірності даних та обробки текстів.

- **Проєкт**: **Сегментація клієнтів на основі даних про покупки**. Використовуючи методи кластеризації (наприклад, K-means) з scikit-learn, була створена модель для сегментації клієнтів великої торгової мережі. Завдяки кластеризації, компанія змогла краще зрозуміти своїх клієнтів і адаптувати маркетингові стратегії під кожен сегмент.

👑 Senior

▼ Загальні запитання

1. **Як зрозуміти, що проєкт варто вирішувати шляхом ML?**

   Використання ML доцільне, коли:

   - Є достатня кількість даних для навчання моделей.
   - Завдання можна описати як проблему прогнозування або класифікації на основі попередніх даних.
   - Традиційні методи алгоритмів не здатні надати точні або масштабовані рішення.
   - Потрібно автоматизувати прийняття рішень на основі складних шаблонів у даних, які важко виявити вручну.

2. **Коли потрібно використовувати ML, а коли — ні?**

   - **Коли використовувати ML**:
     - Є велика кількість даних, і необхідно виявити складні закономірності.
     - Завдання пов'язане з прогнозуванням, класифікацією, кластеризацією або іншими подібними проблемами.
   - **Коли не використовувати ML**:
     - Завдання можна вирішити чіткими правилами або алгоритмами.
     - Даних недостатньо для надійного навчання моделі.
     - Немає потреби в автоматизованому прийнятті рішень або алгоритмичні методи дають кращі результати.

3. **Які передумови для успішного ML-проєкту?**

   - **Доступ до якісних даних**: Дані повинні бути структурованими, актуальними та достатніми.

- **Чітко сформульоване завдання**: Важливо точно розуміти, яку задачу має вирішувати модель.
- **Команда з експертів**: Наявність спеціалістів із даних, машинного навчання та програмування.
- **Інфраструктура**: Підтримка інфраструктури для збору даних, навчання моделей і розгортання.

4. **Що таке успішний ML-проєкт?**

   Успішний ML-проєкт:

   - Надає реальну цінність бізнесу, вирішуючи завдання ефективніше, ніж існуючі підходи.
   - Модель стабільна, узгоджена з етичними та юридичними вимогами, легко оновлюється та інтегрується у виробниче середовище.
   - Модель масштабована і забезпечує бажані результати в реальному середовищі.

5. **Опишіть виклики розгортання моделей машинного навчання в продукції, включаючи моніторинг та обслуговування.**

   - **Data Drift**: Розподіл вхідних даних може змінюватися з часом, що може вплинути на точність моделі.
   - **Моніторинг продуктивності**: Необхідно постійно стежити за продуктивністю моделі, оцінювати зміни в точності та продуктивності.
   - **Масштабованість**: Потрібно забезпечити, щоб модель працювала з великою кількістю даних без втрати продуктивності.
   - **Оновлення моделей**: Потрібно регулярно оновлювати модель на основі нових даних.
   - **Безпека та конфіденційність**: Забезпечення захисту даних у виробничих середовищах.

6. **Як ви вирішуєте питання конфіденційності та етики при роботі з чутливими даними в проєктах з машинного навчання?**

- **Анонімізація даних**: Видалення або заміна особистих даних, щоб зберегти конфіденційність.
- **Шифрування даних**: Використання технологій для захисту даних під час зберігання і передачі.
- **Етичні принципи**: Впровадження політик, які забезпечують відповідальне використання даних, зокрема уникання упереджень у моделі.
- **Використання конфіденційних обчислень**: Методи, як-от differential privacy або федеративне навчання, які дозволяють використовувати дані без їх повного розкриття.

7. **Які є тенденції та нові технології в галузі машинного навчання та штучного інтелекту?**
   - **Великі мовні моделі (LLM)**: Як-от GPT і BERT, які розвивають обробку природної мови.
   - **AutoML**: Автоматизація процесу побудови моделей.
   - **Explainable AI (XAI)**: Створення інтерпретованих моделей для пояснення рішень.
   - **Federated Learning**: Навчання моделей на розподілених даних без передачі сирих даних.
   - **Енергетична ефективність AI**: Оптимізація моделей для зниження споживання енергії.
8. **Чи можете надати приклад складного реального проєкту з машинного навчання, детально описати вашу роль і внесок?**

   Приклад: Розробка системи прогнозування попиту на товари в роздрібній торгівлі. Моя роль включала:
   - Збір і підготовку даних (етап ETL).
   - Побудову моделі на основі градієнтного бустингу з подальшою оптимізацією гіперпараметрів.
   - Реалізацію системи автоматичного оновлення моделі при надходженні нових даних.

- Внесок: Підвищення точності прогнозів на 15% та інтеграція системи в інфраструктуру компанії.

9. **Опишіть ваш досвід проєктування та впровадження комплексних робочих процесів з машинного навчання з найкращими практиками MLOps, включаючи введення даних, навчання моделей і розгортання в хмарному середовищі.**

    - **MLOps**: Це практика автоматизації процесів тренування, розгортання та моніторингу моделей. Я працював над проєктами, які включали:

        - Автоматизацію збору та попередньої обробки даних.
        - Налаштування CI/CD для тренування моделей і їх розгортання.
        - Використання платформ, як-от MLflow для відстеження експериментів і моделей.
        - Розгортання моделей у хмарному середовищі (наприклад, AWS або GCP), з налаштуванням інфраструктури для масштабованої роботи.

▼ Machine Learning

1. **Наведіть приклад найкращих практик підбору гіперпараметрів.**

    - **Grid Search**: Перебір усіх можливих комбінацій гіперпараметрів із заданого списку.
    - **Random Search**: Випадковий вибір комбінацій гіперпараметрів із заданих діапазонів.
    - **Bayesian Optimization**: Оптимізація гіперпараметрів на основі ймовірнісних моделей, що мінімізують кількість потрібних ітерацій.
    - **Hyperband**: Адаптивний метод, що комбінує ранню зупинку і перебір гіперпараметрів для прискорення процесу.
    - **Optuna**: Модуль для автоматизації підбору гіперпараметрів з використанням як Random, так і Bayesian Optimization.

2. **Які етапи розробки ML-проєкту?**

- **Збір і підготовка даних**: Збір відповідних даних, обробка, очищення та аугментація.
- **Розвідка даних (EDA)**: Аналіз даних для виявлення залежностей і аномалій.
- **Вибір моделі**: Вибір підходящої моделі на основі задачі.
- **Тренування моделі**: Навчання моделі на навчальних даних.
- **Оцінка моделі**: Оцінка точності моделі на тестових даних, вибір метрик.
- **Оптимізація**: Підбір гіперпараметрів, зменшення вимірності та поліпшення продуктивності моделі.
- **Розгортання**: Інтеграція моделі в продуктивне середовище.
- **Моніторинг і підтримка**: Моніторинг продуктивності моделі у виробничих умовах, оновлення при необхідності.

3. **Як інтерпретувати передбачення моделі? Які методи інтерпретації ви знаєте? Які їхні недоліки та переваги?**
    - **LIME (Local Interpretable Model-Agnostic Explanations)**: Створює локальні інтерпретовані моделі для пояснення окремих передбачень.
        - Переваги: Проста інтерпретація локальних рішень.
        - Недоліки: Не завжди забезпечує глобальну інтерпретацію.
    - **SHAP (SHapley Additive exPlanations)**: Визначає внесок кожної ознаки в передбачення на основі теорії ігор.
        - Переваги: Дає глобальне і локальне пояснення для кожного передбачення.
        - Недоліки: Може бути обчислювально затратним для великих моделей.
    - **Feature Importance**: Показує, наскільки кожна ознака впливає на загальну продуктивність моделі.
        - Переваги: Швидка і зрозуміла інтерпретація.

    - Недоліки: Не дає локальної інтерпретації для конкретних передбачень.

4. **Як виявляти outlier у вхідних даних класифікатора? Як це можна робити без перенавчання моделі або кластеризації?**
    - **Методи статистичного виявлення**: Наприклад, метод Z-оцінок або IQR (інтерквартильний діапазон), що виявляє дані, які виходять за межі певного інтервалу.
    - **Моделі Isolation Forest**: Виявляє аномальні точки на основі того, наскільки легко їх ізолювати від інших.
    - **One-Class SVM**: Використовується для виявлення аномалій на основі векторів опорних об’єктів.
    - **Метод локальної щільності (Local Outlier Factor)**: Оцінює відстань між точками і щільність їхнього розташування, виділяючи аномальні точки.
5. **Як забезпечити приватність та анонімність даних?**
    - **Анонімізація даних**: Видалення або маскування особистої інформації.
    - **Псевдонімізація**: Заміна реальних ідентифікаторів на псевдоніми для забезпечення конфіденційності.
    - **Differential Privacy**: Додавання шуму до результатів аналізу даних, щоб приховати внесок окремих осіб у набір даних.
    - **Шифрування даних**: Використання методів шифрування під час передачі та зберігання даних.
6. **Як уникнути упередженості (наприклад, расової або гендерної) в моделі, що викликана набором даних, на якому натренована?**
    - **Балансування даних**: Під час навчання використовувати збалансовані набори даних, де представники різних класів рівномірно розподілені.
    - **Fairness Metrics**: Використання метрик, таких як disparate impact або equality of opportunity, щоб оцінити упередженість моделі.

- **Алгоритми обробки упередженості**: Методи коригування результатів, такі як re-weighting або adversarial de-biasing, для зменшення упередженості.
- **Data Augmentation**: Використання аугментації для створення додаткових даних з недопредставлених груп.

7. **У випадку тренування sequential-моделей (LSTM, Transformer-based, etc.), як розв'язувати проблему нерівномірності послідовностей, коли одна довша за іншу?**
    - **Padding**: Додавання нулів або спеціальних токенів для вирівнювання послідовностей до однакової довжини.
    - **Masking**: Використання масок для позначення реальних даних і пропущених елементів у послідовності.
    - **Truncation**: Відсікання занадто довгих послідовностей до фіксованої довжини.
8. **Які автоматичні алгоритми зменшення вимірності характеристик ви знаєте?**
    - **Principal Component Analysis (PCA)**: Зменшує вимірність за допомогою перетворення ознак на нові лінійні компоненти, що зберігають максимальну дисперсію.
    - **t-SNE (t-distributed Stochastic Neighbor Embedding)**: Використовується для зменшення вимірності, зберігаючи структуру даних.
    - **UMAP (Uniform Manifold Approximation and Projection)**: Покращений алгоритм для зменшення вимірності з високою ефективністю для великих наборів даних.
    - **Autoencoders**: Використання нейронних мереж для автоматичного зменшення розмірності даних шляхом навчання представлень меншої вимірності.
9. **Як пояснити вплив тієї чи іншої характеристики на результат моделі? Які алгоритми для цього існують?**

- **Partial Dependence Plots (PDP)**: Візуалізація, що показує, як зміна однієї характеристики впливає на результат моделі.
- **SHAP values**: Оцінюють внесок кожної ознаки в передбачення, використовуючи теорію ігор.
- **LIME**: Пояснює внесок ознак для конкретного передбачення моделі.

10. **Що таке real time training і які проблеми покликаний вирішити цей підхід?**

    Real time training — це метод тренування моделей у реальному часі, коли дані надходять потоково, і модель постійно оновлюється на основі нових даних.

    Проблеми:

    - Обробка великих потоків даних без затримок.
    - Адаптація до змін у даних (concept drift).
    - Ефективне використання ресурсів для оновлення моделі в режимі реального часу.

11. **Які підходи до багатопотокового тренування моделей ви знаєте та чим вони відрізняються?**

    - **Data Parallelism**: Дані діляться на частини і розподіляються між кількома процесорами для паралельного тренування.
    - **Model Parallelism**: Модель ділиться між кількома процесорами або машинами, і кожна частина

    тренується окремо.

    - **Hybrid Parallelism**: Поєднує data і model parallelism для максимальної продуктивності на великих моделях і даних.

12. **Як би ви підійшли до великомасштабного розподіленого проєкту машинного навчання?**

    - **Розбиття на етапи**: Розподіл проєкту на модульні етапи — підготовка даних, тренування моделей, оцінка і розгортання.

- **Розподілена обробка даних**: Використання хмарних сервісів або платформ, як-от **Apache Spark**, для обробки великих обсягів даних у розподіленому середовищі.
- **Розподілене тренування**: Використання фреймворків, як-от **TensorFlow** або **PyTorch**, що підтримують розподілене тренування (data parallelism або model parallelism).
- **Моніторинг і автоматизація**: Налаштування CI/CD з використанням інструментів MLOps, як-от **MLflow**, для моніторингу тренування та оновлення моделей.
- **Масштабованість**: Забезпечення масштабованості за допомогою хмарних сервісів (AWS, GCP) для обробки даних і тренування.

13. **Обговоріть виклики та можливі рішення при роботі з часовими рядами, табличними, текстовими або зображеннями (виберіть один тип даних відповідно до досвіду) для застосувань машинного навчання.Часові ряди**:

    Виклики:

    - Наявність трендів, сезонності або періодичних коливань, що ускладнює моделювання.
    - Зміни у розподілі даних з плином часу (concept drift).
    - Необхідність обробки нерегулярних часових рядків (пропущені або зміщені дані).

    Рішення:

    - Використання **ARIMA**, **LSTM**, або **Prophet** для моделювання часових залежностей.
    - Додавання ознак для обробки сезонності або трендів.
    - Для нерегулярних даних можна використовувати методи імпутації або спеціалізовані архітектури (наприклад, LSTM з масками).
    - Постійний моніторинг для виявлення змін у даних і повторне тренування моделей при потребі.

14. **Що таке First Type Error і Second Type Error?**

- **First Type Error (Помилка першого роду)**: Відкидання нульової гіпотези, коли вона правильна (помилкове відхилення гіпотези). Це також називається **"помилковим позитивом"**.
- **Second Type Error (Помилка другого роду)**: Прийняття нульової гіпотези, коли вона хибна (помилкове прийняття хибної гіпотези). Це також називається **"помилковим негативом"**.

15. **Як сформувати критерії вибору найкращої моделі з усіх натренованих моделей і як пояснити цей вибір нетехнічним колегам?**
    - **Критерії вибору**:
        - Основні метрики (точність, recall, precision, F1-score) для класифікації або MSE/RMSE для регресії.
        - Продуктивність на тестовому наборі даних і перехресній валідації (cross-validation).
        - Інтерпретованість моделі (якщо це важливо для бізнесу).
        - Час тренування і швидкість передбачень.
        - Стабільність і масштабованість моделі в реальних умовах.
    - **Пояснення нетехнічним колегам**:
        - Можна пояснити вибір моделі за допомогою аналогії з тестуванням у реальному світі: «Ми обрали модель, яка найкраще прогнозує результати на реальних даних, схожих на ті, з якими ми стикатимемося в майбутньому. Вона також є швидкою та надійною, що означає, що її можна використовувати для широкого спектра задач».
16. **Як забезпечити відтворюваність тренування моделі під час роботи у великій команді?**
    - **Фіксація початкових умов**: Фіксація seed для генераторів випадкових чисел у моделях, щоб забезпечити однакові результати під час повторного тренування.
    - **Контроль версій**: Використання систем контролю версій (наприклад, Git) для коду і даних, щоб відстежувати всі зміни.

- **Використання DVC (Data Version Control)**: Інструмент для відстеження змін у даних та моделях.
- **Документування процесу тренування**: Ведення детальної документації з налаштуванням параметрів моделі, середовища і процесу тренування.
- **Контейнеризація**: Використання Docker для забезпечення однакових середовищ для всіх учасників команди.

▼ Deep Learning

1. **Розкажіть про ваш досвід використання різних deep learning фреймворків.**
   Я використовував кілька основних фреймворків для глибокого навчання, зокрема:
   - **TensorFlow**: Використовував для великих моделей, що потребують розподіленого тренування. З його допомогою я розробив систему прогнозування продажів, використовуючи часові ряди та рекурентні нейронні мережі (RNN).
   - **PyTorch**: Переважно для наукових досліджень та експериментів, завдяки його гнучкості. Один з проектів включав класифікацію зображень для комп'ютерного зору, де використовувався ResNet.
   - **Keras**: Використовував для швидкої побудови прототипів та простіших моделей. Наприклад, проєкт класифікації текстів із використанням LSTM для аналізу емоцій.
   - **MXNet**: Використовував у хмарних середовищах для розподіленого тренування моделей.
2. **На вашу думку, чи краще для NN працює L2-регуляризація?**

   L2-регуляризація (Ridge) зазвичай краще працює для нейронних мереж, оскільки вона штрафує великі ваги, тим самим зменшуючи ризик перенавчання (overfitting), але не призводить до занулення ваг, як це робить L1-регуляризація. Це дозволяє зберігати всю інформацію в моделі і водночас контролювати величину ваг. L2-регуляризація забезпечує гладкіші та стабільніші результати в багатьох задачах,

зокрема у випадках, коли необхідно уникати великого впливу окремих ознак.

3. **Що таке metrics learning? Чим відрізняється від класифікації? Які основні функції втрат використовують?**
    - **Metrics learning** — це метод навчання моделі так, щоб вона могла вимірювати "близькість" між об'єктами. Основна мета — навчитися порівнювати схожість об'єктів за допомогою певної метрики, наприклад, відстані між векторами ознак.
    - **Відмінність від класифікації**: У класичній класифікації модель передбачає мітку класу, а в metrics learning модель вчиться відображати об'єкти в простір, де схожі об'єкти знаходяться близько один до одного.
    - **Основні функції втрат**:
        - **Contrastive loss**: Використовується для навчання моделі таким чином, щоб схожі об'єкти мали мінімальну відстань, а різні — максимальну.
        - **Triplet loss**: Навчає модель на основі трійок об'єктів (anchor, positive, negative), мінімізуючи відстань між схожими об'єктами і максимізуючи відстань між різними.
        - **Cross-entropy**: Використовується в деяких випадках для задач класифікації в metrics learning, але її адаптовано для метрик.
4. **Опишіть adversarial attacks в глибокому навчанні та можливі заходи захисту від них.Adversarial attacks** — це методи, за допомогою яких можна ввести в модель обманливі або змінені дані, що можуть змусити її робити неправильні передбачення. Такі атаки включають додавання малих непомітних змін до вхідних даних, які значно змінюють виходи моделі.

    Типи атак:

    - **FGSM (Fast Gradient Sign Method)**: Атака, що додає шум до вхідних даних на основі градієнтів моделі.

- **PGD (Projected Gradient Descent)**: Більш потужна версія FGSM з багаторазовим використанням градієнтів.

Заходи захисту:

- **Adversarial training**: Навчання моделі на даних із доданим атакуючим шумом, щоб модель ставала стійкішою.
- **Regularization**: Використання методів регуляризації, таких як dropout або weight decay, для зменшення чутливості моделі до малих змін у даних.
- **Gradient masking**: Техніка для захисту від атак, яка ускладнює обчислення градієнтів для ворожих атак.

5. **Які основні прийоми використовують під час тренування моделей глибокого навчання та в чому їхня практична користь?**

   - **Early Stopping**: Запобігає перенавчанню моделі, зупиняючи тренування, коли продуктивність на валідаційних даних перестає поліпшуватися.
   - **Dropout**: Виключає випадкові нейрони під час тренування для запобігання перенавчанню.
   - **Batch Normalization**: Нормалізує входи кожного шару, що дозволяє стабілізувати тренування і прискорити його.
   - **Data Augmentation**: Збільшує кількість даних за допомогою обертання, фліпів та інших трансформацій для покращення узагальнення моделі.
   - **Gradient Clipping**: Обмежує значення градієнтів, що допомагає уникнути проблеми вибуху градієнтів у великих мережах.

▼ MLOps

1. **Намалюйте діаграму рекомендаційної системи для YouTube.**
   Ось типова структура рекомендаційної системи для YouTube, за нею можна створити діаграму:

   - **Збір даних**: Дані про перегляди, вподобання, коментарі, історію переглядів користувачів.

- **Фільтрація даних**: Видалення дублюючих або нерелевантних даних.
- **Алгоритми рекомендацій**:
  - **Collaborative Filtering**: Рекомендації на основі схожості між користувачами.
  - **Content-Based Filtering**: Рекомендації на основі схожості між відео.
  - **Deep Learning Models**: Нейронні мережі для прогнозування поведінки користувача і рекомендованих відео.
- **Розгортання моделі**: Інтеграція моделі в систему YouTube для постійної генерації рекомендацій.
- **Моніторинг та оновлення**: Постійний моніторинг продуктивності та оновлення моделі на основі нових даних.

2. **Поясніть основні етапи життєвого циклу моделі в MLOps.**
   - **Збір даних**: Початковий збір та підготовка даних для тренування моделі.
   - **Тренування моделі**: Навчання моделі на тренувальних даних.
   - **Оцінка моделі**: Оцінка точності, продуктивності та узагальнюючої здатності моделі на тестових даних.
   - **Розгортання моделі**: Інтеграція моделі у виробниче середовище (production).
   - **Моніторинг**: Постійний моніторинг продуктивності моделі, виявлення дрейфу даних.
   - **Оновлення**: Повторне тренування або оновлення моделі у разі деградації або зміни концепції.
3. **Як MLOps може бути налаштований для моделей, які деплояться на edge-пристрої?**
   - **Модульність**: Моделі, що розгортаються на edge-пристроях, мають бути модульними, з можливістю їхнього оновлення та заміни.

- **Легка вага моделей**: Оскільки edge-пристрої мають обмежені ресурси, слід використовувати оптимізовані моделі (наприклад, через pruning, quantization або використання моделей, як MobileNet).
- **Оновлення моделей**: Налаштування процесу оновлення моделей на edge-пристроях через OTA (Over-The-Air) оновлення, щоб легко впроваджувати нові версії.
- **Локальний моніторинг**: Організація системи для моніторингу продуктивності моделі безпосередньо на пристрої або через регулярні звіти на центральний сервер.

4. **Як заміряти деградацію моделі?**
    - **Моніторинг метрик**: Оцінка ключових метрик (точність, F1-score, precision, recall) під час використання моделі в продукції.
    - **Data Drift**: Виявлення змін у розподілах вхідних даних порівняно з тими, на яких модель тренувалася.
    - **Concept Drift**: Визначення, чи змінився зв'язок між ознаками та передбаченнями моделі, що може викликати зниження точності.
    - **A/B тестування**: Використання двох версій моделі для порівняння їхньої продуктивності в реальному часі.
5. **Назвіть найкращі практики керування залежностями пакетів і середовищами для ML-моделей.**
    - **Docker**: Використання контейнеризації для відтворюваних і ізольованих середовищ.
    - **Conda або virtualenv**: Створення віртуальних середовищ для уникнення конфліктів залежностей.
    - **Poetry або pipenv**: Для управління залежностями Python і фіксації версій пакетів.
    - **DVC (Data Version Control)**: Для відстеження змін у даних і моделях разом із версіями коду.

- **CI/CD інструменти**: Автоматизація розгортання з використанням Jenkins, GitLab CI або CircleCI для оновлення моделей і залежностей.

6. **Які методи моніторингу якості моделей у продакшні ви знаєте? Які з них використовували?**
    - **Моніторинг метрик продуктивності**: Постійне відстеження метрик, таких як точність, precision, recall, F1-score, та їх порівняння з історичними результатами.
    - **Моніторинг латентності**: Вимірювання часу відповіді моделі для забезпечення швидкої роботи в продакшні.
    - **Data Drift і Concept Drift**: Моніторинг змін у розподілах вхідних даних і результатів передбачень для виявлення зміни концепції.
    - **A/B тестування**: Порівняння двох версій моделі для оцінки їхньої ефективності на реальних даних.
    - **Алгоритмічний моніторинг**: Виявлення аномалій у роботі моделі за допомогою спеціальних алгоритмів, що оцінюють коректність передбачень.

▼ NLP

1. **Поясніть, як працює квантизація моделей на прикладі LoRa.**

    Квантизація — це метод зменшення розміру моделей машинного навчання шляхом скорочення точності чисел, що використовуються для ваг і активацій (наприклад, перетворення float32 у int8). Це зменшує обчислювальні витрати і споживання пам'яті, роблячи моделі більш ефективними для edge-пристроїв.

    LoRa (Low-Rank Adaptation) — це метод, який зменшує кількість параметрів моделі, зберігаючи важливі компоненти. LoRa фокусується на адаптації тільки найважливіших частин моделі, роблячи її квантизацію ще ефективнішою для зменшення розміру без значної втрати точності.

2. **Процес файнтюнінгу LLM часто потребує розмічених даних, отримання яких може бути дорогим. Як ми можемо зменшити**

потребу у великих розмічених датасетах під час файнтюнінгу LLM?

- **Few-shot learning**: Навчання моделі на невеликій кількості прикладів для кожного класу, щоб зменшити потребу у великих наборах даних.
- **Transfer learning**: Використання вже натренованих великих мовних моделей (LLM) і подальший файнтюнінг на специфічному домені або задачі.
- **Data augmentation**: Генерація додаткових даних з існуючих шляхом їх варіації або трансформації (переформулювання речень, зміна контексту).
- **Self-training**: Використання ненагляданих даних для отримання додаткових прикладів через самонавчання моделі на власних прогнозах.

3. **Як можна використати LLM для створення wiki за кодом проєкту? Як при цьому боротись з галюцинаціями?**
   - **Використання LLM**: LLM може аналізувати вихідний код, коментарі та документацію, генеруючи текст для створення wiki-сторінок, що пояснюють функціонал коду, архітектуру і API.
   - **Боротьба з галюцинаціями**:
     - **Автоматизоване тестування**: Перевірка відповідності згенерованого тексту реальному коду через статичний аналіз або інтеграційні тести.
     - **Human-in-the-loop**: Залучення експертів для перевірки і виправлення тексту, згенерованого LLM, щоб уникнути недостовірних фактів.
     - **Фіксування знань**: Обмеження генерації відповідно до наявних даних або документації, що зменшить ризик генерації «галюцинацій».
4. **Де можна використати Explainable AI (XAI) в NLP?**
   - **Моделі для класифікації текстів**: Використання XAI для пояснення, чому модель віднесла текст до певного класу. Наприклад, можна

використовувати SHAP для пояснення внеску кожного слова у передбачення.

- **Моделі генерації тексту**: Використання XAI для пояснення вибору слів або фраз у моделі генерації тексту.
- **Інтерпретація мовних моделей (LLM)**: Застосування XAI для пояснення, як мовна модель робить передбачення на основі контексту.
- **Налаштування чат-ботів**: Для пояснення, як модель вибирає конкретні відповіді, щоб надати прозорість для користувачів.

5. **Які є методи покращення якості LLM-ембедингів?**
    - **Fine-tuning**: Донастроювання LLM на специфічних даних, щоб покращити представлення слів у певному контексті або домені.
    - **Domain adaptation**: Тренування моделі на специфічних даних певної галузі (наприклад, наукові тексти, технічна документація), щоб покращити якість ембедингів для цього домену.
    - **Contrastive learning**: Навчання моделі так, щоб семантично подібні речення мали схожі ембединги, а несхожі — різні.
    - **Dynamic embeddings**: Використання контексту для динамічного оновлення ембедингів слів, як це робиться в моделях трансформерів (наприклад, BERT).
    - **Data augmentation**: Збагачення набору даних через переформулювання речень або створення синонімічних варіантів, щоб покращити узагальнюючу здатність моделі.

▼ Алгоритми

1. **Який з двох алгоритмів XGBoost і Random Forest можна паралелізувати та чому?**
    - **Random Forest**: Його легко паралелізувати, оскільки кожне дерево в лісі створюється незалежно від інших. Паралелізація можлива на рівні створення дерев, тому що всі дерева можуть будуватися одночасно.

- **XGBoost**: Паралелізація також можлива, але з певними обмеженнями. Оскільки XGBoost використовує бустинг, кожне дерево будується послідовно на основі помилок попереднього. Проте XGBoost може паралелізувати деякі внутрішні процеси, такі як побудова окремих вузлів дерев або обчислення градієнтів.

Висновок: **Random Forest** можна легше паралелізувати на рівні дерев, тоді як XGBoost паралелізує лише деякі внутрішні операції через послідовну природу алгоритму бустингу.

2. **Що таке cold start problem? Як з нею справлятись?Cold start problem** виникає в системах рекомендацій або машинного навчання, коли немає достатньо даних для нового користувача або нового елемента (наприклад, новий користувач ще не взаємодіяв із системою, або новий продукт ще не має відгуків).
   - **Як справлятись**:
     - **Content-based filtering**: Використання інформації про самого користувача або продукт для створення рекомендацій (наприклад, на основі профілю або атрибутів).
     - **Попередні рекомендації**: Надання загальних або популярних рекомендацій новим користувачам.
     - **Hybrid methods**: Комбінація collaborative і content-based методів для мінімізації проблеми cold start.
3. **Що таке concept drift?Concept drift** — це зміна у відношенні між вхідними ознаками та виходами моделі з часом. Це може трапитися через зміну трендів, сезонності або умов у реальному світі, через що модель, тренована на старих даних, втрачає актуальність і точність.
   - **Як боротись**:
     - Постійний моніторинг даних і результатів моделі.
     - Періодичне оновлення і повторне тренування моделі на нових даних.
     - Використання методів, що адаптуються до змін, наприклад, онлайн-навчання.

4. **Що таке model drift?Model drift** — це термін, який зазвичай використовується для опису зменшення продуктивності моделі з часом через зміни в даних або процесах, що відбуваються в реальному світі. Model drift може бути спричинений не лише змінами в даних (як у випадку з concept drift), але й змінами в самому середовищі або інфраструктурі, де модель працює.

    - **Як боротись**:
        - Постійний моніторинг і регулярна валідація результатів моделі.
        - Оновлення моделі на основі нових даних та перевірка відповідності даних поточним умовам.

▼ Computer Vision

1. **Де можна використати Explainable AI (XAI) у CV?**

    - **Моделі класифікації зображень**: Використання XAI для пояснення, які області зображення найбільше вплинули на прийняття рішення моделі. Популярні інструменти, такі як **Grad-CAM**, візуалізують, які частини зображення найбільш важливі для передбачення.
    - **Object detection**: Можна застосовувати XAI для пояснення, як модель приймає рішення при виявленні об'єктів. Наприклад, які особливості об'єкта визначають клас або межі об'єкта.
    - **Semantic segmentation**: XAI допомагає пояснити, як модель сегментує зображення, які області вона вважає релевантними для кожного класу.

2. **Що таке hard negative mining і як це використовують в object detection?Hard negative mining** — це процес, при якому модель під час тренування особливу увагу звертає на негативні приклади (приклади, які вона неправильно класифікує або з якими має труднощі). У контексті **object detection**, це зображення, на яких немає об'єкта, але модель неправильно виявляє об'єкт.

    Використовується для поліпшення точності моделі, оскільки вона стає більш чутливою до важких негативних прикладів, що підвищує її здатність розрізняти об'єкти та фонові елементи.

3. **Чому transfer learning із задачі детекції об'єктів до задач сегментації працює погано?**
   Основна проблема полягає в тому, що задачі детекції об'єктів і сегментації мають різну природу:

   - **Object detection**: Основна мета — знайти межі об'єкта (bounding boxes).
   - **Segmentation**: Вимагає точного розмежування кожного пікселя, що належить об'єкту, а не просто визначення його меж.
     Тому моделі для детекції фокусуються на загальному розташуванні об'єктів, тоді як сегментація потребує більшої деталізації на рівні пікселів.

4. **Що таке anchor в object detection? Розкажіть про anchor-free підходи.Anchor** — це попередньо визначені рамки (bounding boxes) різних розмірів і пропорцій, які модель використовує для передбачення об'єктів. Модель порівнює ці рамки з реальними об'єктами і на основі цього вчиться прогнозувати правильні межі.

   Anchor-free підходи: Такі моделі не використовують заздалегідь визначені рамки. Натомість вони передбачають місце розташування центру об'єкта і його розміри без використання фіксованих anchor-боксів. Приклад — **CenterNet** або **CornerNet**.

5. **Як виконати задачу instance segmentation, використовуючи моделі для semantic segmentation?**
   Для виконання задачі
   instance segmentation (сегментація окремих об'єктів) на основі моделей для **semantic segmentation** (де кожен піксель має мітку класу) можна використовувати такі підходи:

   - **Mask R-CNN**: Популярна модель, яка доповнює сегментацію виділенням меж кожного об'єкта.
   - **Conditional Random Fields (CRF)**: Після виконання semantic segmentation можна використовувати CRF для постпроцесингу, що виділяє окремі об'єкти.

- **Watershed алгоритм**: Після семантичної сегментації цей алгоритм може допомогти розділити злиті об'єкти, виходячи з їхніх форм і розташування.

6. **Як фреймворки TensorRT та OpenVINO оптимізують сітки?**
   - **TensorRT**: Оптимізує глибокі нейронні мережі для виконання на GPU шляхом зменшення точності (наприклад, перетворення FP32 у FP16 або INT8), використання шару ф'юзії (об'єднання шарів для зменшення витрат на обчислення), а також застосування інших оптимізацій, таких як видалення зайвих операцій і планування виконання графів.
   - **OpenVINO**: Оптимізує нейронні мережі для виконання на CPU та інших процесорах. Включає такі оптимізації, як квантизація, ф'юзія операцій та використання спеціалізованих інструкцій для прискорення обчислень (наприклад, використання векторних інструкцій Intel).
7. **Розкажіть про архітектуру ConvNeXt.**

   ConvNeXt — це вдосконалена архітектура згорткових нейронних мереж (CNN), яка враховує деякі аспекти трансформерів (як-от Vision Transformer) і використовує їх для поліпшення класичних CNN. ConvNeXt була розроблена як спроба зробити CNN більш конкурентоспроможними в задачах зору, де останнім часом домінують трансформери. Архітектура зберігає простоту ResNet, але включає більш глибокі конволюційні блоки та інші оптимізації для покращення точності.
8. **Як оптимізувати алгоритми, які працюють на потоці відео?**
   - **Frame skipping**: Обробляти не всі кадри, а лише кожен n-ий кадр, щоб зменшити кількість даних для обробки.
   - **Розпаралелювання**: Використовувати багатопоточність або обчислення на GPU для обробки декількох кадрів одночасно.
   - **Model pruning**: Використовувати легші моделі, оптимізовані для реального часу, наприклад, шляхом видалення менш значущих нейронів (pruning).

- **Квантизація**: Використання моделей із нижчою точністю (наприклад, INT8 замість FP32) для прискорення обчислень.
- **Buffering and Caching**: Зберігання результатів попередніх кадрів для їх повторного використання, щоб уникнути повторних обчислень.