Вопросы для подготовки к вступительному тестированию: техническая часть.

Программа “Искусственный интеллект” и “Управление AI-продуктом” ИТМО

Список тем, входящих в блок Hard Skills в рамках вступительного онлайн-тестирования:

1. Программирование на языке Python.
2. Алгоритмы и структуры данных.
3. Теория множеств. Теория графов.
4. Теория вероятностей и математическая статистика.
5. Базы данных. SQL и NoSQL.
6. Основные концепции Big Data.
7. Основы машинного обучения.
8. Нейронные сети.
9. Основы компьютерного зрения (CV).
10. Основы обработки естественного языка (NLP).
11. Инструменты промышленной разработки ПО.

1. Программирование на Python

Подтемы

- Списковые включения и генераторы
- Декораторы и контекст-менеджеры
- Типизация и статическая проверка
- Асинхронность (asyncio)
- Файлы и форматы данных
- Тестирование (pytest)
- Packaging и окружения
- Профилировка и оптимизация

Примерные вопросы

- Чем list comprehension отличается от обычного цикла for?
- Когда стоит писать собственный контекст-менеджер через @contextmanager?
- Поясните разницу между typing.List[int] и list[int] в Python 3.9+.
- В чём преимущества asyncio по сравнению с потоками при I/O-задачах?
- Что происходит, когда функция достигает выражения yield?
- Как организовать parametrized tests в pytest?

Рекомендуемые материалы

- Официальная документация Python (рус.)
- Stepik курс «Поколение Python»
- LearnPython.ru – интерактивный тренажёр

2. Алгоритмы и структуры данных

Подтемы

- Анализ сложности, Big-O и амортизированная оценка
- Сортировки O(n log n) и ниже: быстрая, слиянием, кучей, radix
- Рекурсия и динамическое программирование: мемоизация, оптимальный разрез
- Диапазонные структуры: дерево отрезков, дерево Фенвика, sparse table
- Пространственные индексы k-NN: KD-tree, Ball-tree, HNSW
- Система непересекающихся множеств (DSU) и эвристики rank/ path compression
- Поиск кратчайших путей: Дейкстра, A*, Bellman–Ford
- Алгоритмы MST и базовых графов: Краскал, Прим, BFS/DFS, топологическая сортировка
- Строковые алгоритмы: префикс-функция (КМП), Z-функция, суффикс-массив и LCP
- Хеш-структуры и коллизии: хеш-таблица, bloom-filter, perfect hashing
- Рекурсия и динамическое программирование

Примерные вопросы

- Почему быстрая сортировка имеет среднюю сложность O(n log n), но худшую $O(n^2)$?
- Докажите, что алгоритм динамического программирования для задачи о рюкзаке работает за O(nW).
- Чем дерево отрезков удобнее массива префиксных сумм для диапазонных обновлений?
- Опишите операции Make-Set, Find, Union для DSU и их сложность с эвристиками.
- Как работает очередь с приоритетом в реализации Дейкстры на C++ STL?
- Для чего нужна префикс-функция в алгоритме КМП?

Рекомендуемые материалы

- e-maxx.ru – Алгоритмы на русском
- Stepik «Алгоритмы: теория и практика. Методы»
- Лекции МФТИ «Алгоритмы и структуры данных» (YouTube)

3. Теория множеств и графов

Подтемы

- Основы множеств и операций $\cup$ $\cap$; мощности и принцип включения-исключения
- Перечислительная комбинаторика: перестановки, размещения, сочетания, биномиальные коэффициенты
- Планарные графы: критерий Куратовского, формула Эйлера, предел $|E| \le 3|V| - 6$
- Обходы и связность: BFS/DFS, компоненты, топологическая сортировка
- Минимальные остовные деревья: свойства cut & cycle, алгоритмы Краскала и Прима
- Теория потоков: max-flow $\leftrightarrow$ min-cut, алгоритмы Эдмондса–Карпа и Диница
- Паросочетания в двудольных графах: теорема Холла, алгоритм Куна/Хопкрофта-Карпа
- Эйлеровы и гамильтоновы циклы: критерии существования, алгоритм Флёри / теорема Дирака
- Маршруты Эйлера и Гамильтона

Примерные вопросы

- Сколько различных функций можно задать на множестве из n элементов?
- Сформулируйте и докажите формулу Эйлера для связного плоского графа.
- Когда алгоритм Краскала предпочтительнее алгоритма Прима?
- Почему алгоритм Эдмондса–Карпа имеет сложность $O(VE^2)$?
- Докажите теорему Холла для двудольных графов.
- Дайте критерий существования эйлерова цикла в неориентированном графе.

Рекомендуемые материалы

- Лекции И. Куликова «Теория графов» (YouTube)
- Stepik курс «Теория графов»
- e-maxx.ru – раздел «Графы»

4. Теория вероятностей и математическая статистика

Подтемы

- Ожидание и дисперсия случайных величин
- Закон больших чисел и ЦПТ
- Оценка параметров: ММП и метод моментов
- Статистические критерии (χ^2, t-тест)
- Неравенства Маркова и Чебышёва
- Методы Монте-Карло
- Байесовское обновление при диагностических тестах (деревья вероятностей).

- Условные вероятности. Определение условной вероятности, формула полной вероятности, формула Байеса. Независимость событий на вероятностном пространстве. Попарная независимость и независимость в совокуп- ности.
- Схема Бернулли. Формула Бернулли. Формула Пуассона. Формулы Муавра-Лапласа и их применение. По- линомиальная схема
- Случайные величины как измеримые функции. Функция распределения. Функция плотности. Независимость случайных величин. Случайные векторы. Функции от случайной величины.
- Математическое ожидание в дискретном и абсолютно непрерывном случае, дисперсия, ковариация и корре- ляция. Их основные свойства. Дисперсия суммы независимых случайных величин. Математическое ожидание и матрица ковариаций случайного вектора. Симметричность и неотрицательная определенность матрицы ко- вариаций. Математическое ожидание случайной величины в общем виде. Условное математическое ожидание. Регрессия. Другие числовые характеристики случайных величин. Моменты старших порядков.
- Распределения. Стандартные дискретные и непрерывные распределения, их математические ожидания, дис- персии и свойства: биномиальное; равномерное; нормальное и многомерное нормальное; пуассоновское; показа- тельное; геометрическое; отрицательно-биномиальное; распределение Паскаля; гипергеометрическое; распре- деление Гнеденко-Вейбулла; гамма-распределение.

Примерные вопросы

- Выведите формулу дисперсии через математическое ожидание.
- Сформулируйте сильный закон больших чисел.
- Чем отличается несмещённая оценка от состоятельной?
- При каких предпосылках применяется t-тест Стьюдента?
- Докажите неравенство Чебышёва.
- Как оценить π методом Монте-Карло?

Рекомендуемые материалы

- Лекции К. Воронцова «Теория вероятностей» (pdf)
- Stepik курс «Теория вероятностей»
- Учебник Б. В. Гнеденко «Курс теории вероятностей» (эл. версия)

5. Базы данных. SQL и NoSQL

Подтемы

- Индексы и хранение: B-tree, Hash, GIN/GiST/BRIN, bitmap, партиционирование
- Запросы SQL: CTE, оконные функции, агрегация, UPSERT, субзапросы, EXPLAIN

- Нормализация ↔ денормализация: 1NF–BCNF, схемы звезда/снежинка, шардирование
- Транзакции и изоляция: MVCC, Read Committed, Repeatable Read, Serializable
- JSON/JSONB и semi-structured: операторы ->, @>, индексация, full-text-search
- ORM-фреймворки (SQLAlchemy, Django ORM)
- NoSQL модели и CAP/BASE: документные, key-value, графовые, колоннарные
- Репликация, резервное копирование и тюнинг производительности
- ACID vs BASE в распределённых БД

Примерные вопросы

- Как B-tree индекс ускоряет запросы с оператором BETWEEN?
- Опишите различие уровней изоляции Read Committed и Serializable.
- Покажите пример использования оконной функции ROW_NUMBER().
- Какие преимущества JSONB перед JSON в PostgreSQL?
- Что такое lazy loading в ORM и когда оно вредно?
- Почему BASE-подход часто выбирают для NoSQL систем?

Рекомендуемые материалы

- Документация PostgreSQL (рус.)
- Stepik курс «SQL для анализа данных»
- Хабр-коллекция «Базы данных»

6. Основные концепции Big Data

Подтемы

- Парадигма Map-Reduce и эволюция от Hadoop MR к Spark
- Apache Spark: RDD, DataFrame/Dataset API, Catalyst, Tungsten, lazy-evaluation
- Управление кластерами: YARN, Mesos, Stand-alone, Kubernetes Scheduler
- Kafka / Pulsar и потоковая обработка: Exactly-once, Kafka Streams, Spark Structured Streaming
- Хранилища данных 3-го поколения: Data Lake + Delta Lake (ACID-слои, time-travel, schema evolution)
- Data Vault 2.0: моделирование hub-link-satellite, PIT-таблицы, бизнес-ключ vs surrogate key
- Partitioning & Bucketing: стратегии по диапазону, хэшу, времени; динамические партиции
- HDFS репликация и fault-tolerance
- Колоночно-ориентированные форматы Hydra и компрессия: Parquet, ORC, ZSTD, Snappy, predicate pushdown
- Мониторинг и fault-tolerance: speculative execution, checkpointing, lineage и DAG-визуализация

Примерные вопросы

- Чем DataFrame в Spark отличается от RDD с точки зрения оптимизации?
- Опишите архитектуру YARN: ResourceManager и NodeManager.
- Как Kafka обеспечивает «at-least-once» доставку сообщений?
- Зачем использовать bucketing, если уже есть partitioning?
- Что происходит при потере одного из DataNode в HDFS?
- Какие гарантии ACID даёт Delta Lake поверх S3?

Рекомендуемые материалы

- Учебник «Технологии больших данных» (ИТМО)
- Stepik курс «Большие данные»
- Документация Apache Spark (рус.)

7. Основы машинного обучения

Подтемы

- Подготовка данных и feature engineering: очистка, обработка пропусков, масштабирование, кодирование категориальных признаков
- Базовые модели классификации и регрессии: линейные методы, k-NN, Naive Bayes, решающие деревья
- Ансамбли и градиентный бустинг (Random Forest, XGBoost, LightGBM, CatBoost)
- Ядровые методы и метод опорных векторов (SVM)
- Снижение размерности и визуализация: PCA, LDA, t-SNE, UMAP
- Оценка качества и предотвращение переобучения: train/val/test-split, k-fold, регуляризация L1/L2, early stopping
- Метрики сравнения моделей: accuracy, ROC-AUC, PR-AUC, F1, RMSE/MAE, R^2 и т.д.
- Подбор гиперпараметров и автоматический поиск: GridSearch, RandomizedSearch, Optuna/Bayesian Optimization
- Интерпретируемость моделей и оценка важности признаков: permutation importance, SHAP

Примерные вопросы

- Почему CatBoost не требует one-hot для категориальных признаков?
- Сформулируйте задачу SVM в её примале L2-регуляризованной форме.
- Как выбрать число компонент PCA, объясняя 95 % дисперсии?
- В каких случаях t-SNE лучше UMAP для визуализации?
- Что показывает площадь под ROC-кривой (AUC)?
- Когда целесообразно применять RandomizedSearch вместо GridSearch?

Рекомендуемые материалы

- Учебник ШАД по ML
- Курс HSE ML (GitHub)
- Stepik курс «Машинное обучение»

- Курс по Машинному обучению Радослава Нейчева и Владислава Гончаренко: ФИВТ МФТИ. https://github. com/girafe-ai/ml-course

8. Нейронные сети

Подтемы

- Базовые многослойные перцептроны и функции активации
- Свёрточные сети (CNN) и остаточные блоки (Resnet)
- НС для обработки последовательностей: RNN, LSTM, GRU, seq2seq с attention
- Архитектура Transformer: self-attention, позиционные кодировки, GPT/BERT-подобные модели
- Регуляризация и оптимизация: Dropout, Batch/Layer Norm, weight decay, AdamW, learning-rate schedules, gradient clipping
- Early stopping и мониторинг тренировочных/валидационных метрик
- Перенос обучения и fine-tuning (prompt-, LoRA-, adapters)
- Генеративные сети: автоэнкодеры и VAE, GAN-семейство (DCGAN, WGAN, StyleGAN), diffusion-модели (DDPM, Stable Diffusion)

Примерные вопросы

- Чем LSTM отличается от простой RNN с точки зрения градиентов?
- Опишите механизм multi-head attention.
- Как выбрать слои для fine-tuning при transfer learning?
- Почему Dropout снижает переобучение?
- В чём идея циклического learning rate?
- Какие метрики мониторят для early stopping в задачах классификации?

Рекомендуемые материалы

- Книга «Нейронные сети и глубокое обучение» (Николенко, pdf)
- YouTube курс ODS «Нейронные сети»
- Stepik «Введение в нейронные сети»

9. Основы компьютерного зрения (CV)

Подтемы

- Базовая классификация изображений: CNN-backbone, ResNet, EfficientNet
- Детекция объектов: YOLO-семейство, SSD, Faster R-CNN, anchor-free подходы (CenterNet)
- Сегментация: U-Net, DeepLab v3+, Mask R-CNN
- Feature-descriptors и keypoints: SIFT, SURF, ORB, FAST + BRIEF; matching и RANSAC
- Аугментации и инвариантность: flips, crops, color-jitter, CutMix, RandAugment
- Метрики качества: IoU, mAP@[.5:.95], Dice/F-score, Pixel Accuracy

- Transfer learning и fine-tuning: фиксированный backbone, freezing, adapters, weight decay
- Современные CV-архитектуры: Vision Transformer (ViT), Swin, ConvNeXt, CLIP и multimodal embeddings

Примерные вопросы

- Чем anchor-based модели отличаются от anchor-free?
- Как работает skip-connection в U-Net?
- Почему SIFT устойчив к масштабированию?
- Назовите три полезных аугментации для малых датасетов.
- Что означает IoU=0.5 в COCO-метрике?
- Как заморозка слоёв помогает при transfer learning?

Рекомендуемые материалы

- ODS курс «Компьютерное зрение» (YouTube)
- Stepik «Компьютерное зрение»
- Документация OpenCV (рус.)

10. Основы обработки естественного языка (NLP)

Подтемы

- Статистические языковые модели: n-gram, сглаживание, перплексия
- Обучаемые векторные представления: Word2Vec (CBOW/Skip-gram), GloVe, FastText
- Токенизация и субсловные схемы: BPE, SentencePiece, WordPiece, обработка OOV
- Последовательная разметка: CRF, Bi-LSTM-CRF, задачи NER/POS
- Fine-tuning готовых моделей: BERT/RuBERT, LoRA/PEFT, выбор слоёв для заморозки
- Основы LLM: causal-LM (GPT-семейство), masked-LM (BERT), instruction-/RLHF-tuning, RAG-подход
- Оценка качества: BLEU, ROUGE, perplexity, оценка ответов LLM через Llama-Guard/Hallucination checks

Примерные вопросы

- Что такое сглаживание Кнезера-Нэя?
- Опишите обучение Skip-gram модели Word2Vec.
- В чём отличие BPE от WordPiece?
- Какие преимущества CRF над HMM в sequence labeling?
- Когда BLEU неадекватно оценивает качество перевода?
- Чем RuBERT отличается от оригинального BERT?

Рекомендуемые материалы

- NLP курс MIPT (GitHub)
- Stepik «Введение в NLP»
- Статья ODS «Разбор BERT на русском»

11. Инструменты промышленной разработки ПО

Подтемы

- Управление версиями и ветвления: Git Flow, trunk-based, semantic versioning, Rebase vs Merge, Pull-Request workflow
- Контейнеризация и образы: Dockerfile (best practices, multi-stage build), Docker Compose, secrets & env-vars, базовые приёмы оптимизации размера образа
- Оркестрация и окружения: Kubernetes (Deploy/Service/Ingress), k3d/Minikube для локальной отладки
- CI/CD конвейеры: GitHub Actions, GitLab CI, автоматические тесты → линтеры → build → push → deploy, канареечные релизы, rollback-стратегии
- Тестирование и качество кода: Pytest-fixtures и parametrization, coverage/pytest-cov, linters (ruff/flake8), stat-analysis (Bandit, mypy), contract-тесты API (pytest-httpx)
- Мониторинг: Prometheus + Grafana, логирование через Loki/ELK, алерты Alertmanager
- Инфраструктура как код (IaC): Terraform providers (AWS/GCP/Yandex Cloud), Ansible roles/playbooks, Molecule-тесты, drift-detection
- MLOps основы: MLflow / DVC для версионирования моделей и данных

Примерные вопросы

- Опишите стратегию GitFlow и её плюсы.
- Когда следует использовать Docker Compose вместо kubectl apply?
- Как реализовать многоступенчатый build Docker-образа?
- Чем отличатся workflow dispatch от push триггера в GitHub Actions?
- Какие метрики важно мониторить для REST-сервиса ML-модели?
- Что такое idempotency в Ansible?

Рекомендуемые материалы

- Книга «Pro Git» (рус.)
- Документация Docker (рус.)
- Stepik курс «DevOps практики и инструменты»
- Документация GitHub Actions (рус.)

🔗 Пример технического тестирования

Задания на написание кода:

После тестовых вопросов с вариантами ответов вам будут предложены 3 задачи на написание кода и анализ данных, которые проверяют следующие знания и умения:

1. **Программирование на Python**
уверенное владение языком, стандартными библиотеками и системой пакетов.

2. **Обработка данных в табличных форматах**
чтение файлов с табличными данными (Parquet/CSV), группировки, агрегирование, вычисление статистик и поиск ответов по данным.

3. **SQL-стиль запросов внутри Python**
умение формулировать выборки и агрегации через pandas / polars / duckdb.

4. **Анализ данных: базовый статистический анализ, визуализация и EDA**
средние, дисперсия, квантиль, выявление аномалий, подготовка наглядных графиков и формулировка продуктовых гипотез.

5. **Разработка Telegram-/Web-ботов или простейшего UI**
подключение API, организация диалога, обработка пользовательского ввода.

6. **Web-scraping и работа с открытыми источниками**
извлечение и структурирование данных из HTML-страниц.

7. **Использование LLM как вспомогательного инструмента (RAG)**
построение простого retrieval-QA по собственному корпусу документов.

8. **Инженерия, практики DevOps и проектная структура**
чистый репозиторий, README, виртуальное окружение / Docker, тесты, грамотная работа с Git, автоматическая проверка и сборка проектов

Как проверяем?

Решения таких заданий на написание кода и анализ данных, когда вы прикладываете ссылку на решений в своем github-репозитории, будут проходить через ревью экспертами к собеседованию для оценки вашего умения решать сложные задачи в ограниченное время. А на собеседовании будут заданы дополнительные вопросы по тому, как вы решали задания с кодом, что было легко, а что давалось с трудом.

Полезные инструменты

- **Язык и среда:** Python 3.11+, VS Code / PyCharm + встроенный Copilot-chat
- **Обработка и SQL-аналитика:** pandas, polars, DuckDB, PyArrow
- **Визуализация и EDA:** seaborn, matplotlib, plotly
- **Web-scraping:** requests, BeautifulSoup, Playwright
- **Боты, веб-сервисы и REST API:** python-telegram-bot / aiogram, FastAPI (для REST-эндпоинтов)
- **LLM + Retrieval:** OpenAI / YandexGPT SDK, LangChain или llama-index, FAISS / Chroma (векторное хранилище)
- **Контроль версий, контейнеры и деплой:** Git + Github, Docker, docker-compose, Автоматические проверки и сборка с GitHub Actions (CI/CD)
- **Тесты и качество кода:** pytest, coverage, pre-commit (black, isort, flake8)
- **Документация:** Jupyter Notebook, Markdown + README templates

AI Coding Assistant

Вы решаете эти задания в отведенное время технического испытания (вместе с тестом у вас всего 4 часа), для ускорения написания кода НЕ запрещается, а рекомендуется использовать AI-ассистентов для написания кода, например:

GitHub Copilot (IDE плагин), **Copilot Chat** (VS Code), **OpenAI GPT-4o** в браузере (chat.openai.com или API-ключ + curl), **Claude Code CLI**, **Claude 3.5 / 3.7 Sonnet в IDE** (VS Code, JetBrains), Claude 4 (Opus 4) (chat.claude.ai + API), **Cursor** (AI IDE на базе GPT-4), **Codeium** (free-плагин для большинства IDE), **Yandex GPT / SourceCraft Code Assistant** и т.д.

Рекомендации по использованию AI Coding Assistant в ограниченное время

1. **Пишите промпт-задачу целиком** («Загрузи Parquet, посчитай 95-квантиль по колонке amount и округли вверх до целого»).

2. **Проверяйте результат сразу**: запускайте сэмпл-датасет, ловите ошибки — просите ассистента «почини stack trace».

3. **Оставляйте комментарии**: если код сгенерировал ИИ, пометьте кратко, что именно он делает — это плюс при ревью.

4. **Просите тесты**: «напиши pytest-тест, проверяющий округление вверх». Автотесты экономят нервы на финальной проверке.

5. **Мини-рефактор после генерации**: переименуйте переменные, уберите лишние импорты — чтобы код был читаем.

Используя AI-помощников, вы сокращаете рутину (парсинг, boilerplate, документация) и оставляете время на логику решения задачи и проверку результатов.

Как тренироваться заранее

1. **Мини-хакатон «1 файл → 8 метрик»**
найдите любой открытый датасет (например, на Kaggle); поставьте себе 8-10 вопросов по данным; решите их «вслепую» сначала руками, потом с LLM-ассистентом.

2. **«QA-Бот-вечер»:**
возьмите пару произвольных \HTML-страниц, спарсите, соберите простейшего QA-ассистента в LangChain, поднимите Telegram-бота, который отвечает на вопросы по этим данным, попробуйте усложнить обработку интентов, когда в одном запросе пользователя может быть более сложная задача, требующая применения уже агентской парадигмы с разными инструментами.

3. **EDA-challenge**
Сделайте по любому открытому датасету репо шаблон: `notebooks/eda.ipynb`, `src/`, `requirements.txt`, `Makefile test`. Пусть LLM сгенерирует draft-README, а вы доведите до ума визуальный анализ данных и предложите ряд продуктовых

гипотез и инсайтов по данным.

4. **CI/CD за 30 минут**
склонируйте любой hello-api, подключите GitHub Actions на push: `pytest` ➜ `docker build` ➜ `docker push ghcr.io/...`