Data Engineer

Лучшие практики по приготовлению данных. Загрузка, обработка, организация хранения и доступа к данным с использованием современных инструментов

Длительность курса: 113 академических часов

1 Data Architecture

1	**Инженер данных. Задачи, навыки, инструменты, потребность на рынке**	**Цели занятия:** познакомиться и обсудить правила работы и общения; выяснить, кто такой инженер данных; сформулировать задачи инженера данных и ожидания бизнеса. **Краткое содержание:** Data Engineer: потребность и ценность; задачи, навыки, инструменты, классификации данных; создание ценности и основные вызовы (challenges).
2	**Архитектура аналитических приложений: базовые компоненты и принципы**	**Цели занятия:** проследить эволюцию подходов работы с данными; получить представление о технологиях и инструментах; рассмотреть принципы построения архитектуры аналитического приложения. **Краткое содержание:** история и эволюция подходов работы с данными; технологические основы аналитических решений; Distributed Computing, MPP (Massive Parallel Processing), DWH, CAP + NoSQL; подходы к обработке данных: Batch & Stream; архитектуры: Lambda, Kappa.
3	**On premises / Cloud solutions**	**Цели занятия:** изучить набор компонент, необходимых для построения системы обработки данных; понять разницу между on-premises и cloud решениями; изучить, какие конкретные инструменты использовать для построения системы обработки данных на Hadoop / Google Cloud / AWS. **Краткое содержание:**

облака в сравнении с on-premises инфраструктурой: возможности, преимущества, особенности;
экосистема Hadoop и элементы Системы Обработки Данных;
аналоги из экосистем GCP, AWS.

Домашние задания

1 Регистрация аккаунта в облаке

Цель: В результате выполнения ДЗ вы создадите свой первый Cloud-аккаунт, который вы будете использовать для выполнения домашних заданий.

1. Создайте аккаунт в Яндекс-почте или используйте уже существующий (https://yandex.ru);
2. Подключите двухфакторную авторизацию для вашего аккаунта (https://yandex.ru/support/passport/authorization/twofa-on.html);
3. На главной странице Яндекс-облака (https://cloud.yandex.ru) нажмите кнопку "Подключиться";
4. На стартовой странице вашей консоли (https://console.cloud.yandex.ru/) скрольте консоль вниз (панель слева) и выберете пункт "Биллинг" (https://console.cloud.yandex.ru/billing);
5. На странице "Биллинг" слева выберете пункт "Список аккаунтов" и создайте новы платёжный аккаунт;
6. Перейдите в созданный платёжный аккаунт и в разделе "Обзор", нажмите кнопку "Активировать промокод" и введите полученный промокод;
7. Вернитесь на главную страницу (https://cloud.yandex.ru) в разделе "Ваши ресурсы" отредактируйте название и описание стандартного каталога (наведите курсор на каталог и нажмите на появившиеся три точки и выберите пункт "Изменить");
8. На стартовой странице вашей консоли (https://console.cloud.yandex.ru/) скрольте консоль вниз (панель слева) и выберете пункт "Управление доступом" (https://console.cloud.yandex.ru/iam);
9. На странице "Управление доступом" (https://console.cloud.yandex.ru/iam) активируйте доступ "Только администраторам";
10. Добавьте дополнительного пользователя нажав кнопку "Добавить пользователя";

11. Добавьте роль "editor" на ваш каталог новому пользователю (наведите курсор на пользователя и нажмите на появившиеся три точки и выберите пункт "Изменить", нажмите кнопку "+ Назначить роль" возле интересующего вас каталога);
12. Авторизуйтесь под новым пользователем, зайдите на дашборд каталога, выберете "Compute cloud" и нажмите "Создать ВМ";
13. Добавьте SSH-ключ в форме конфигуратора виртуальной машины.

---

4 **Автоматизация пайплайнов и оркестрация – 1**

Цели занятия:

научиться использовать оркестраторы для автоматизации процессов обработки данных;
написать DAG'и для создания сложных процессов обработки, состоящих из множества этапов.

Краткое содержание:

назначение оркестраторов;
сравнение самых популярных оркестраторов - Oozie и Airflow;
устройство Airflow и способы его использования для построения сложных процессов обработки данных.

---

5 **Автоматизация пайплайнов и оркестрация – 2**

Цели занятия:

научиться строить более сложные пайплайны Airflow;
понять архитектуру Airflow и способы деплоя;
научиться выполнять базовые операции с AF - тестирование, бэкфилл и т. д.

Краткое содержание:

сенсоры, сабдаги и другие специальные операторы;
архитектура Airflow - webserver, scheduler, worker;
использование Airflow CLI.

Домашние задания

1 Подготовка и установка на расписание DAG выгрузки данных из источников

Цель: В данном ДЗ мы настроим автоматический data pipeline, который будет получать данные из публичного API и складывать их в БД для дальнейшего анализа.
Конечный продукт:
1) Работающий облачный инстанс Apache Airflow.
2) Data pipeline, содержащий в себе несколько task-ов и "крутящийся" по расписанию Airflow.
3) Работающий облачный инстанс СУБД, куда Airflow заливает данные, получаемые из внешнего API.
4) Данные в СУБД.

Часть из операций мы разбирали на занятии. При необходимости - можно пересмотреть запись.

1) Создаем Виртуальную машину с Apache Airflow 2.0 в Yandex-Cloud.

2) Создаем "managed instance" PostgreSQL/ClickHouse/MySQL - по выбору в Yandex-Cloud.
Создаем БД "analytics".

3) Добавляем наш psql в Connections через UI Airflow.

4) Выбираем один из 2-х API, с которым будем работать:
- Положение Международной Космической Станции на текущий момент времени (timestamp-latitude-longitude). Source: http://api.open-notify.org/iss-now.json
- Курс BTC: https://docs.coincap.io/#2a87f3d4-f61f-42d3-97e0-3a9afa41c73b
Тут нас интересует следующий endpoint: "api.coincap.io/v2/rates/bitcoin"

5) Создаем схему данных (таблицу в бд analytics) с названиями и типами полей, релевантными тому, что будем забирать из API.

6) Описываем DAG (программируем на Python) для получения данных с периодичностью 30 min.
Сам оператор для обращения к API можно выбрать любой.
Для простоты рекомендуется слать GET-запросы через python-библиотеку requests (`PythonOperator`), либо через bash

(`BashOperator`) с помощью curl.

Концептуальная схема Dag-а: отправить запрос в API->распарсить пришедший результат->положить данные в БД (сделать insert в таблицу)

7) Кладем .py файл с DAG-ом в нужную директорию виртуалки с airflow.

8) Запускаем DAG тумблером в UI Airflow.

9) Отлаживаем DAG до работоспособного состояния.

В интерфейсе Airflow (облачный инстанс) есть информация о наборе успешно завершившихся Dag runs (темно-зеленые кружки) + в БД есть данные за >5 периодов времени.

В качестве проверки мы зайдем в вам airflow webserver и отправим sql-запрос в таблицу с данными.

10) Проверяем, что данные появились в БД.

11) Поздравляю, вы завершили ДЗ!

2 Data Lake

1 **Распределенные файловые системы. HDFS / S3**

Цели занятия:

научиться работать с HDFS;
разобраться в архитектуре HDFS;
узнать, что такое Object Storarge.

Краткое содержание:

архитектура HDFS (namenode, datanode, HA, ...);
структура файлов (блоки, репликация, ...);
примеры работы с помощью CLI (утилита hdfs);
Object Storarge, сходство и отличия от HDFS на примере S3.

---

2 **SQL-доступ к Hadoop. Apache Hive / Presto**

Цели занятия:

познакомиться с инструментами класса SQL Engine, их устройством, компонентами;
получить представление об особенностях работы, сценариях использования.

Краткое содержание:

архитектура и модель данных Hive;
стратегии оптимизации работы Hive;
архитектура и назначение Presto.

---

3 **Разбор ДЗ по 1 модулю**

Цели занятия:

сравнить своё решение ДЗ из модуля 1 с эталонным.

Краткое содержание:

пример решения ДЗ из модуля 1;
основные нюансы;
часто встречающиеся ошибки.

---

4 **Форматы хранения данных и их особенности**

Цели занятия:

изучить факторы выбора формата хранения данных;
понять разницу между Row-based и Column-based форматами;
познакомиться с наиболее распространенными форматами.

Краткое содержание:

обзор популярных форматов: CSV, JSON, Avro, Parquet, ORC;
анализ применения и производительности форматов;
бинарные и человеко-читаемые форматы хранения;
Schema evolution, Compression, Bloom filters, Indexing;
новое поколение форматов - Delta, Hudi, Iceberg.

---

5 **Очереди сообщений. Обзор Kafka**

Цели занятия:

изучить очереди сообщений (Kafka, Rabbit MQ, NATS, ...);
изучить примеры архитектур с использованием очередей сообщений;
рассмотреть архитектуру Apache Kafka.

Краткое содержание:

потоковая обработка;
виды обработки и особенности;
внутреннее устройство Kafka.

---

6 **Выгрузка данных из внешних систем**

Цели занятия:

изучить классификацию источников для выбора правильного способа загрузки;
научиться загружать источники с помощью NiFi;
научиться реплицировать оперативные базы данных.

Краткое содержание:

типы, форматы источников и способы их загрузки;
использование Apache Ni-Fi в качестве инструмента интеграции данных;
способы загрузки оперативных баз данных: CDC и snapshots.

Домашние задания

1 Загрузка сырых данных

Цель: В данном ДЗ мы настроим data pipeline в Apache NiFi, который будет:

- получать данные, отправляемые по HTTP (по сути, работать как API);
- анализировать содержимое того, что ему прислали;
- выполнять некоторые преобразования (склейку);
- сохранять результат в отдельную директорию в соответствии с контентом.

Конечный продукт:

1) работающий pipeline NiFi в контейнерном окружении;
2) примеры curl-команд, приводящих к разным результатам;
3) текстовые данные в соответствующих директориях контейнерной файловой системы.

1) Скачиваем себе образ Apache NiFi (если еще не сделали этого на занятии):

docker pull apache/nifi:latest

2) Собираем контейнер с mapping-ом 2-х портов: через который будем локально смотреть в интерфейс и через который будем общаться через curl:

docker run --name nifi -p 9090:9090 -p 8081:8081 -d -e NIFI_WEB_HTTP_PORT='9090' apache/nifi:latest

3) Создаем стартовую точку pipeline-а:

ListeHTTP-процессор.
Порт: 8081 (его мы заранее мэппили для docker-контейнера)

Base Path: loglistener

4) Второй узел. Он будет перенаправлять данные в соответствии с тем, что пришло на наш веб-сервер извне:

RouteOnContent-процессор.
Подключаем его связью "success" к ListeHTTP.
Match Requirement: "content must contain match"
Добавляем ему новое свойство errortext со значением "ERROR".
Теперь из узла способно выходить 2 потока: для flowfiles с ключевым значением (под названием errortext) и второй: unmatched

5) Проверяем работоспособность 1-го узла:

Нажимаем "Play" для ListeHTTP-процессора
Из командной строки делаем: " curl --data "someRandomText" 127.0.0.1:8081/loglistener"
Жмем Refresh (после клика ПКМ)
Видим, что в него пришли данные

6) Добавляем узел, который будет объединять несколько файлов в один:

Добавляем MergeContent-процессор.
Создаем 2 связи из RouteOnContent в MergeContent: errortext и unmatched
Настраиваем узел (ПКМ => configure):
* Scheduling -> Run Schedule: 10 sec
* Merge Strategy: Bin-Packing Algorithm
* Correlation Attribute Name: RouteOnContent.Route - этот параметр позволит нам писать файлы в разные директории в дальнейшем
* Merge Format: Binary Concatenation
* Maximum Number of Entries: 500
* Maximum Bin Age: 90s (максимальное время жизни flow-файла, после которого его принудительно выведет из pipeline)

Мы будем склеивать текстовые файлы, чтобы не плодить их в большом количестве.
Поэтому укажем формат склейки:
* Delimiter Strategy: Text
* Demarcator: открываем и в поле нажимаем Shift+Enter (это аналог \n- перевод каретки на новую строку)

7) Сохраняем полученные артефакты в облачный

s3 Yandex.Cloud:

PutS3Object-процессор
* Directory:
nifiHW/${RouteOnContent.Route}/${now()}.txt

Создаем бакет nifiHW через интерфейс.
У нас будет 2 папки в s3: merged и unmatched
Делаем связи от узла MergeContent вида "merged" и "unmatched"

Дополнительно требуется разобраться с тем, как начать писать в бакет через API.

8) Запускаем весь pipeline

9) Шлем много разных запросов через cli: содержащих текст ERROR внутри и нет.

Примеры:

curl --data "adbrsgbndt ERROR fevrtb" 127.0.0.1:8081/loglistener
curl --data "uiebveovne" 127.0.0.1:8081/loglistener

10) Наблюдаем за тем, что происходит внутри pipeline-а.

11) Заходим через интерфейс в s3 Yandex.Cloud и смотрим, какие новые папки и файлы появились внутри.

---

7 **Apache Spark – 1**

Цели занятия:

познакомиться с Apache Spark;
научиться работать с API Spark.

Краткое содержание:

история и назначение Spark;
как устроены кластеры и приложения Spark изнутри;
dataFrame и Dataset API.

---

8 **Apache Spark – 2**

Цели занятия:

научиться оптимизировать приложения на Spark.

Краткое содержание:

примеры проблем, возникающих в приложениях на Spark;
способы решения этих проблем;
использование Spark UI для анализа производительности.

Домашние задания

1 Сборка витрины на PySpark

Цель: В этом задании предлагается собрать статистику по криминогенной обстановке в разных районах Бостона. В качестве исходных данных используется датасет https://www.kaggle.com/AnalyzeBoston/crimes-in-boston

Цель задания - разработать программу построения витрины.
Результат - ссылка на репозиторий с кодом.

Программа должна запускаться через spark-submit.
Пути к данным и к результату должны передаваться в качестве аргументов вызова.

Инструкция:
1) Загрузить данные.

2) Проверить данные на корректность, наличие дубликатов. Очистить.

3) Собрать витрину (агрегат по районам (поле district)) со следующими метриками:
- crimes_total - общее количество преступлений в этом районе;
- crimes_monthly - медиана числа преступлений в месяц в этом районе;
- frequent_crime_types - три самых частых crime_type за всю историю наблюдений в этом районе, объединенных через запятую с одним пробелом “, ” , расположенных в порядке убывания частоты;
- crime_type - первая часть NAME из таблицы offense_codes, разбитого по разделителю “-” (например, если NAME “BURGLARY -

COMMERICAL - ATTEMPT”, то crime_type “BURGLARY”);
- lat - широта координаты района, рассчитанная как среднее по всем широтам инцидентов;
- lng - долгота координаты района, рассчитанная как среднее по всем долготам инцидентов.

4) Сохранить витрину в один файл в формате .parquet в папке path/to/output_folder.

Подсказки:
- Функция percentile_approx может посчитать медиану.
- Конкретный месяц идентифицируется не только номером месяца, но и номером года.
- В справочнике кодов есть дубликаты. Нужно выбрать уникальные коды, взяв любое из названий.

3 DWH

1 **Аналитические СУБД. MPP-базы данных**

Цели занятия:

изучить принципы функционирования кластерных аналитических СУБД;
изучить примеры реализации MPP и Shared Nothing архитектур;
научиться использовать специфичные механизмы MPP-баз для оптимизации запросов.

Краткое содержание:

существующие MPP-базы и их общие черты;
механизмы оптимизации MPP-баз для аналитических запросов;
подробный анализ этих механизмов на примере одной из СУБД.

---

2 **Моделирование DWH – 1. Основы работы с dbt**

Цели занятия:

научиться автоматизировать сборку витрин и построение графа зависимостей;
научиться использовать фреймворк dbt для управления "Хранилищем Данных";
научиться выстраивать работу с СУБД в соответствии с лучшими практиками.

Краткое содержание:

лучшие практики работы с Аналитическими СУБД;
логический и физический дизайн;
Data Build Tool (dbt): инициализация, подключение, моделирование.

---

3 **Моделирование DWH – 2. Data Vault 2.0**

Цели занятия:

сформулировать требования к модели DWH;
изучить принципы Data Vault, его преимущества и недостатки;
получить базовые навыки моделирования согласно Data Vault.

Краткое содержание:

принципы моделирования DWH;
подход Data Vault 2.0;
разбор примера построения DWH на DV 2.0;
построение модели данных на практике.

Домашние задания

1 Построение Data Vault

Цель: Изучение принципов моделирования данных в Data Warehouse: нормализация, обогащение, интеграция, единая логическая модель, витрины данных и расчет метрик.
Развертывание окружения: СУБД + dbt.
Конфигурация проекта dbt, установка модуля dbtVault.
Проектирование детального слоя на базе подхода Data Vault, подготовка метаданных для кодогенерации.
Автоматизация наполнения детального слоя данных с помощью dbt + dbtVault.
Формирование витрин данных на основе детального слоя.

https://docs.google.com/document/d/1t_P0Cww9MgHYeGkIC6p-V4ZXddFaXj31_PrE1vLKPdU/edit?usp=sharing

4	**Data Quality. Управление качеством данных**	**Цели занятия:** получить представление о том, что такое качество данных; выяснить как Data Quality влияет на выводы и принимаемые решения; рассмотреть стратегии управления качеством данных. **Краткое содержание:** основные метрики качества данных; причины нарушения качества и стратегии реагирования; измерение, мониторинг, исправление; демонстрация: тесты актуальности, консистентности в DBT, кросс-проверки источник <-> DWH.
5	**Разбор ДЗ по 2 модулю**	**Цели занятия:** сравнить своё решение ДЗ из модуля 2 с эталонным. **Краткое содержание:** пример решения ДЗ из модуля 2; основные нюансы; часто встречающиеся ошибки.
6	**DevOps практики в Аналитических приложениях. CI + CD**	**Цели занятия:** изучить основы DevOps-практик; рассмотреть подходы к построению CI-CD pipelines применительно к аналитическим задачам; продемонстрировать несколько инструментов: Jenkins, Github Actions. **Краткое содержание:** культура DevOps; работа в команде; Continuous Integration / Continuous Delivery.

7 **Развертывание BI-решения**

Цели занятия:

сформулировать назначение систем класса BI;
изучить принципы работы BI-инструментов и решаемые задачи;
сравнить и выбрать BI-решение для конкретного кейса.

Краткое содержание:

обзор популярных BI-решений;
варианты развертывания: self-hosted vs. managed;
задание метрик, фильтров, сегментов;
сборка аналитических дашбордов: лучшие практики.

Домашние задания

1 Витрина + BI

Цель: 1. Установить Metabase:

Использовать виртуальную машину и Docker в Yandex.Cloud.

2. Подключиться к источнику данных.

Использовать любой доступный источник данных:

- Jaffle shop (postgres) из занятия по dbt.
- Любой датасет из блока Data Lake.
- ClickHouse Playground: https://clickhouse.tech/docs/en/getting-started/playground/
- Любой другой.

3. Создать дашборд:

- Несколько видов визуализации (таблицы, графики, числа).
- Фильтры.
- Блок с комментариями.

Выслать скриншоты своего дашборда с комментариями.

https://gist.github.com/kzzzr/0d9ab5898866de3db2b66c79c111d967

8 **Мониторинг / Метаданные**

Цели занятия:

рассмотреть инструменты мониторинга - Prometheus, Zabbix, Graphite, Grafana;
понять специфику мониторинга процессов обработки данных.

Краткое содержание:

для чего нужен мониторинг;
какие проблемы он решает;
паттерны и антипаттерны построения систем мониторинга;
сравнение решений представленных на рынке.

4 NoSQL/NewSQL

1 **NoSQL Хранилища. Wide-column и key-value**

Цели занятия:

познакомиться с течением NoSQL;
рассмотреть классификацию, причины появления, историческую справку NoSQL СУБД;
рассмотреть несколько примеров Key-value NoSQL СУБД, как они устроены внутри, как их правильно использовать.

Краткое содержание:

хранилища NoSQL, назначение и особенности;
причины и история появления NoSQL СУБД;
классификация NoSQL СУБД;
несколько примеров NoSQL СУБД: Google Bigtable; Apache HBase; Amazon Dynamo; Redis; Aerospike.

---

2 **NoSQL Хранилища. Document-oriented**

Цели занятия:

познакомиться с Document-oriented NoSQL хранилищами;
рассмотреть несколько примеров Document-oriented NoSQL СУБД, как они устроены внутри, как их правильно использовать.

Краткое содержание:

Document-oriented NoSQL хранилища, назначение и особенности;
несколько примеров NoSQL СУБД: MongoDB; Elasticsearch.

---

3 ELK

Цели занятия:

познакомиться с основными концепциями Elasticsearch;
рассмотреть возможности обработки данных с использованием Logstash;
рассмотреть визуализации в Kibana.

Краткое содержание:

основные концепции;
основы анализа и поиска;
соединения и агрегации;
обработка данных с logstash;
визуализации в kibana.

Домашние задания

1 Анализ веб-логов с помощью ELK

Цель: В рамках ДЗ нужно будет загрузить данные в ElasticSearch и построить дашборд в Kibana.

https://github.com/Gorini4/elk_demo - в README репозитория содержится полная инструкция с подсказками.

1. Склонируйте репозиторий в директорию elk_demo
2. Зайдите в эту директорию и разверните инфраструктуру, выполнив в терминале docker-compose up
3. Отредактируйте файл clickstream.conf
4. Загрузите данные веб-логов, выполнив команду ./load_data.sh
5. Перейдите по адресу http://localhost:5601 и создайте отчет (dashboard), показывающий распределение запросов с разными кодами ответов (status_code) по времени

4 **ClickHouse**

Цели занятия:

понять, для чего нужны mpp-БД и ClickHouse в частности;
научится устанавливать CH и взаимодействовать с ним.

Краткое содержание:

mpp-БД, какие решения есть сегодня на рынке;
практический пример использования одной из самых инновационных mpp-систем на сегодня: ClickHouse.

---

5 **Разбор ДЗ по 3 модулю**

Цели занятия:

сравнить своё решение ДЗ из модуля 3 с эталонным.

Краткое содержание:

пример решения ДЗ из модуля 3;
основные нюансы;
часто встречающиеся ошибки.

5 MLOps

1 **Организация и Packaging кода**

Цели занятия:

рассмотреть организацию и структуру кода;
научиться паковать свой код с помощью python подходов.

Краткое содержание:

вид production-кода для задач классификации и регрессии;
packaging кода с помощью python библиотек;

Домашние задания

1 Развертывание рекомендательной системы

Цель: Цель - использовать на практике приобретенные навыки написания чистого кода (чистота кода, структура проекта и архитектурные паттерны, тесты), изоляции окружения (python virtual environment, docker) и деплоя. Студент продуктивизирует mvp ноутбук и доведет его до состояния близко к продуктивному решению - напишет и упакует код с виртуальным окружением в образ (docker), который реализует предиктивное действие (получение результаты через REST запрос), напишет тесты, автоматическую перетренировку модели по запросу и/или по расписанию, задеплоит его в yandex cloud, выложит решение в виде кода на gitlab (нужно завести учетку) и положит созданный образ с моделью/кодом в какой-нибудь container registry.

1) найти более-менее упоротый (грязно написанный, с многими нарушениями принципов чистого кода, работы с данными) jupyter-notebook на кэггле (kaggle.com)
2) создать структуру проекта (ml cookiecutter template) и будет вести разработку в git, с учетом практик чистого кода, readme
3) написать тесты разработанной функциональности на pytest
4) реализовать rest api сервиса - inference/predict
5) *автоматическое переобучение по расписанию

		или по запросу [через вызов скрипта из консоли или REST запрос] 6) написать docker file - venv и все настройки, чтобы можно было запускать как локально, так и за в клауде, чтобы работал режим работы через rest запросы 7) *написать docker compose - чтобы сохранять модель в базу с версионированностью 8) *то что и в п.7, но с использованием dvc + mlflow 9) деплой (запуск) в клауд/локально
2	**Docker и REST-архитектура**	**Цели занятия:** изучить REST-архитектуре и ее применении; научиться пользоваться одним из широко распространенных веб фреймворков Flask и с его помощью превращать свою модель в готовый веб-сервис; рассмотреть docker, для чего он используется и где применяется; научиться использовать docker для packaging своего кода; научиться использовать docker для деплоя на AWS системы. **Краткое содержание:** основы и примеры REST-архитектуры; Flask и его использование; построение REST Endpoint на основе ML-модели; локальное тестирование POST-запросов в Postman. Docker и его структура; использование Docker-контейнеров для packaging кода; применение docker для работы на AWS — деплой с использованием AWS ECS и EMR.

3	**MLFlow + DVC**	**Цели занятия:** научиться интегрировать модель с сервисами MLFlow и DVC для более воспроизводимого обучения моделей, контроля их метрик и датасетов. **Краткое содержание:** сервис MLFlow для управления экспериментами; сервис DVC для версионирования датасетов моделей.
4	**Деплоймент моделей**	**Цели занятия:** научиться использовать мощный инструмент ML на AWS-Amazon Sagemaker для разработки и тренировки собственных моделей и создания endpoints для работы с веб-сервисами. **Краткое содержание:** использование специализированных сервисов ML на AWS — Amazon Sagemaker; использование "нативного" Sagemaker, комбинирование Sagemaker с sklearn, написание собственных docker images; пример реализации endpoint для ML-модели.
5	**Разбор ДЗ по 4 модулю**	**Цели занятия:** сравнить своё решение ДЗ из модуля 4 с эталонным. **Краткое содержание:** пример решения ДЗ из модуля 4; основные нюансы; часто встречающиеся ошибки.

6 **Разбор ДЗ по 5 модулю**

Цели занятия:

сравнить своё решение ДЗ из модуля 5 с эталонным.

Краткое содержание:

пример решения ДЗ из модуля 5;
основные нюансы;
часто встречающиеся ошибки.

6 Выпускной проект

1 Выбор темы и организация проектной работы

Цели занятия:

выбрать и обсудить тему проектной работы;
спланировать работу над проектом;
ознакомиться с регламентом работы над проектом.

Краткое содержание:

правила работы над проектом и специфика проведения итоговой защиты;
требования к результату проекта и итоговой документации.

Домашние задания

1 Проектная работа

Цель: В качестве курсового проекта необходимо придумать дизайн и архитектуру, а затем - имплементировать data-driven приложение в выбранной доменной области.
Работающее приложение должно быть представлено в качестве репозитория в GitHub.

Подробнее см. в фале: https://docs.google.com/document/d/1Onz9oWn_TfdWZmBhRCOIUAyKKLX3gXqcliMLdhzBoow/edit?usp=sharing

---

2 Консультация по проектам и домашним заданиям

Цели занятия:

получить ответы на вопросы по проекту, ДЗ и по курсу.

Краткое содержание:

вопросы по улучшению и оптимизации работы над проектом;
затруднения при выполнении ДЗ;
вопросы по программе.

3	**Защита проектных работ**	**Цели занятия:** защитить проект и получить рекомендации экспертов. **Краткое содержание:** презентация проектов перед комиссией; вопросы и комментарии по проектам.