- Exploratory DA — анализ (чаще всего глазками) данных, чтобы найти в нём пропуски, выбросы и прочий мусор, который вы можете глазами найти. Тут же вы думаете о том, чем эти данные являются, и как их хранить. А ещё тут вы можете формировать базовые гипотезы.
- Confirmatory DA — не рассказали, что такое.
- Показательный анализ данных — машинное обучение с целью что-нибудь предсказывать.
- Визуализация данных.

Знание — закономерность в некоторой области, которые позволяют решать проблемы.
Data Mining — один из этапов в извлечении знаний из данных. Тут три этапа:

1. Сбор данных.
2. Выделение признаков.
3. Применение ML.

Фактически это как DA, но с акцентом на немного другое.
Data Science:

- Сбор данных.
- Интеграция данных (data integration).
- Хранение данных (data warehousing).
- Анализ данных.
- Высокопроизводительные вычисления (high-performance computing).

Как это с точки зрения бизнеса выглядит? Есть CRISP-DM:

1. Понимание бизнеса (перевод слов заказчика на что-то понятное).
2. Понимание данных (требование правильных данных от заказчика).
3. Подготовка данных (понять, как хранить данные).
4. Моделирование (собственно, решать задачу).
5. Оценка (понять, что мы получили).
6. Если всё хорошо, внедряем, заворачиваем в красивую обёртку и т.п.

Данные. Данные бывают структурированные, неструктурированные и частично структурированные. Чаще всего мы встречаем неструктурированные. Это видео, посты в соцсетях и любая другая хрень, которую мы встречаем. Частично структурированные — XML, JSON и прочий мусор, который до полностью структурированных не дотягивает. А полностью структурированные данные — это матрицы (из n объектов по m характеристик у каждого).
Имея неструктурированные или частично структурированные данные, мы почти всегда хотим его структурировать, чтобы было удобнее.
В DA строку матрицы называют object, instance, sample, example и как угодно ещё, а столбец матрицы — feature, attribute, characteristic или factor.
Что важно про наши матрицы? Чтобы порядок столбцов и строк был не важен. Если ваш алгоритм не такой, это странно. Исключение: временной ряд.

Признаки. Глобально их два: категория и число (качество и количество). Первое дискретно, второе непрерывно, категории мы можем только на равенство проверять, а с числами можем арифметику делать. Категориям обычно целое число сопоставляют, чтобы удобно было. Но есть и другие типы, например, порядковый: дискретный, но есть порядок. Впрочем, очень часто он сводится к либо к категории, либо к числу.
С числами работать гораздо удобнее, но если надо, легко можно дискретизировать наше число любом образом (например, баллы в оценку — дискретизация). Как преобразовать порядковый тип во что-то? Можно преобразовать в число через порядковый номер. А можно в категорию, создав k бинарных значений вида $< x$.

One-hot encoding. Берём категорию из k значений, преобразуем в k булевых категорий вида $= c_i$, а потом из истины сделать 1, а из лжи — 0. Отличный план, как из категории сделать число.
Ещё можно преобразовывать иначе, например категории сопоставлять случайный вектор или число, бинарное представление которого используется как новые признаки.
Важное примечание: числа — не всегда числа. Например, если мы цифры распознаём, то плохо считать числа числами, а не категориями. Потому что изображение тройки не находится между изображением двойки и четвёрки. Поэтому аккуратно с преобразованием категорий в числа.
Ещё интересность: время. Если у нас есть периоды, то хреначим синус и косинус, чтобы периодично было:

$$f_{2e-1} = \sin \frac{2\pi t}{T_e} \qquad f_{2e} = \cos \frac{2\pi t}{T_e}$$

Или ещё интересность, цвет. Цвет кодируем в RGB, а не в HSV/HSB, потому что там тон цвета неоднозначный. Красный — это и 0, и 360.
Как хранить данные? Самый простой формат: CSV. Он плохо стандартизирован, правда. А ещё его недостаток в том, что там шапки нет. Нет ни информации о типах, ничего. Если хочется более человеческого — ARFF: примерно как CSV, но более стандартизирован и имеет шапку с типами данных. Типы: Numeric (число), Nominal (категория), String и Date.

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Прочие типы данных.

- Картинка. Самое простое — двух- (или трёх-)мерная матрица. И есть 1000 алгоритмов, как привести её в один вектор (притом довольно маленький).
- Текст. Тут сложнее. Самое тупое — посчитать частоту слов. Но не надо, тут сразу двигайте в сторону LLM.
- Звук. Это сигнал во времени, и самый простой вариант его представить — вектор значений. Для него тоже есть всякие штуки типа спектрограммы Мэла, чтобы преобразовывать звук во что-то более адекватное.
- Видео. Это мультимодальный объект, то есть объект, в котором разные типы внутри. Их анализировать тоже своими методами.

Нормализация. Зачем? Чтобы признаки с большей дисперсией не влияли на результат сильнее остальных.
Как? Сначала отбросим единицы измерения. Чтобы это было корректно, алгоритм должен быть инвариантен относительно линейных преобразований над признаками. А дальше надо как-то нормализовать дисперсию. Можно, например, вычесть из каждого признака минимум и поделить на разность между максимумом и минимумом. Это $[0;1]$-масштабирование. А ещё можно вычесть (выборное) мат. ожидание из признака и поделить на (выборное) стандартное отклонение это Z-масштабирование. Способов ещё куча, но это два простых.
Как жить с ситуацией, когда у нас признаки зависимы? Никак не жить, если так, вы плохо собрали данные.

Веса и сэмплирование. Отлично, мы всё нормализовали, а теперь у нас из специфики задачи какие-то признаки более важны. Или могут быть объекты более важны.
Говорят, что объект/признак обладает весом w, если он в w раз больше влияет на результат. Если w целое, то это эквивалентно тому, что объект/признак встречается w раз.
Примеры: учёт веса признака

$$\operatorname{dist}(a;b) = \sqrt{\sum_{j=1}^{m} w_j (a_j - b_j)^2}$$

учёт веса объекта:

$$\mathfrak{L}_D(\theta) = \sum_{x \in D} w(x) \mathfrak{L}(x;\theta)$$

А есть сэмплирование: вместо вхождения кучу раз в датасет мы можем сэмплировать объекты с вероятностью $\frac{w_j}{\sum w_j}$.
Ещё это может использоваться для балансировки. Если вы знаете, что реальное распределение не очень сочетается с распределением данных, то мы можем пореже выбирать объекты мажоритарного класса или почаще объекты миноритарного класса (например, в данных одна группа большая, а другая — маленькая, а в жизни они равны, и мы даём большие веса малой группе). Да и в принципе, если нас не интересует распределение признаков по классам, распределение будет чаще подсовывать вам штуки из мажоритарного класса, что вам не всегда хочется.
Как сэмплировать? Можно рандомно, можно каждый x-тый объект (систематически), можно стратифицированно (ровно в соотношение с какой-то категорией), можно кластерное (если кто-то уже за нас разбил объекты на множества, их множеств можно выбрать несколько).

Шум. Аномалии — плохие объекты для модели. Ошибки — плохие объекты для реальности. Пример: пусть у нас есть расход топлива автомобилями. И у нас у одного указан 30л / 100км. Если это грузовой автомобиль, а остальные — легковые, то это аномалия. А если объект всем хорош, и данные такие, то это ошибка (а возникнуть такое могло из-за того, что кто-то подразумевал мили на галлон).
Пропуски в наборе данных. Откуда? Из склейки разных наборов данных, из изначально херовых данных или откуда угодно ещё. В CSV нет стандартного способа это обозначить, в ARFF это вопросик, а ещё можно специальное значения: None/Null для строк, категория с несуществующим номером для

категорий или NaN для числа. Что делать с этим? Вырезать (признак или объект), заменить (заполнить пропуск средним арифметическим/модой/алгоритмом предсказания, который умеет в пропуски) или добавить (добавить новый признак, который говорит, были ли пропуск или нет; если там была категория, можно отдельное значение категории туда добавить, а не новое свойство делать).