Data Mining - методы анализа и обработки данных

Лекция 1

Введение в Data Mining и инструментарий Python.
Постановка задачи, данные, метрики и основы анализа

Институт математики, механики
и компьютерных наук им. И.И. Воровича

Что такое Data Mining?

Data Mining — процесс обнаружения в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений.

- **Аналогия:** Добыча золота. Тонны породы (данных) ради грамма металла (знаний).
- **Цель:** Извлечение знаний из Big Data.
- **Инструментарий:** Статистика + Машинное обучение + Базы данных.

Короткая история развития

- **1960-е:** Появление баз данных. Поиск по ключевым словам.
- **1980-е:** SQL, OLAP. Аналитик сам пишет сложные запросы.
- **1990-е:** Big Data. Традиционный SQL перестает справляться. Появление термина *Data Mining*.
- **2000-е:** Расцвет Machine Learning (Случайный лес, SVM).
- **2010-е — наст. время:** Глубокое обучение (Deep Learning), GPT и большие языковые модели.

Типы данных (Признаки)

Числовые (Количественные)

- **Непрерывные:** Цена, вес, температура.
- **Дискретные:** Количество детей, возраст.

Категориальные (Качественные)

- **Номинальные:** Цвет, марка авто.
- **Порядковые:** Рейтинг (1-5 звезд).

Подготовка данных: Пропуски

Пропущенные значения (NaN): их нельзя оставлять, алгоритмы не умеют с ними работать.

- **Удалить:** если пропусков мало.
- **Заполнить средним/медианой:** для числовых данных.
- **Заполнить модой:** для категориальных данных.
- **Интерполяция:** предсказать значение на основе соседних.

Подготовка данных: Масштабирование

Признаки могут иметь разный масштаб (вес в тоннах, цена в рублях). Это ломает метрики расстояний.

Нормализация (Min-Max)
Сжимает данные в диапазон [0, 1].

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Стандартизация (Z-score)
Среднее = 0, отклонение = 1.

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

Пространство признаков

Каждый объект можно представить как точку в многомерном пространстве.

$$X = (x_1, x_2, ..., x_n)$$

- **Элемент пространства** — конкретный объект (строка в таблице).
- **Признак** — характеристика объекта (измерение, ось).
- Задача Data Mining — найти структуру, кластеры или границы в этом пространстве.

Метрика в пространстве признаков

Метрика — это функция расстояния между двумя объектами.

Евклидово (L2):

$$d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Прямолинейное.

Манхэттенское (L1):

$$d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

По сетке.

Пример расчета метрик

Даны две точки: $A(2, 3)$ и $B(5, 7)$.

Евклидово:
$\sqrt{(5-2)^2 + (7-3)^2}$
$= \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = 5$

Манхэттенское:
$|5-2| + |7-3|$
$= 3 + 4 = 7$

Евклидово расстояние всегда $\leq$ Манхэттенского.

Косинусное сходство

Используется для текстов и векторов, где важно направление, а не длина.

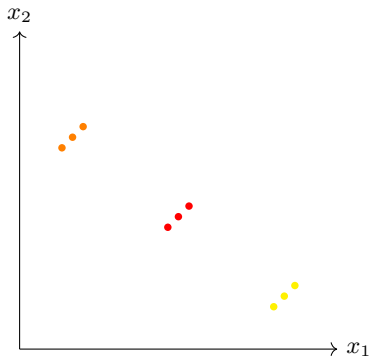
$$\cos(\theta) = \frac{A \cdot B}{||A|| \times ||B||} = \frac{\sum A_i B_i}{\sqrt{\sum A_i^2} \sqrt{\sum B_i^2}}$$

- Значение от -1 до 1.
- 1 — векторы направлены в одну сторону (максимально похожи).
- 0 — векторы ортогональны (не связаны).

Классический пример: «Ирисы Фишера»

Датасет, собранный Рональдом Фишером в 1936 году. Стандарт для демонстрации алгоритмов.

- **Объем:** 150 экземпляров (3 класса по 50).
- **Классы:** Setosa, Versicolor, Virginica.
- **Признаки:** Длина/ширина чашелистика и лепестка.



Основные задачи Data Mining

Обучение с учителем

Есть правильные ответы (метки).

- **Классификация:** категория.
- **Регрессия:** число.

Обучение без учителя

Правильных ответов нет.

- **Кластеризация:** группы.
- **Ассоциации:** связи.

Классификация и Регрессия

Классификация
Предсказание категории.
Пример: Спам / Не спам.
Метрики: Accuracy, F1-score.

Регрессия
Предсказание числа.
Пример: Цена квартиры.
Метрики: MAE, RMSE.

Train/Test Split

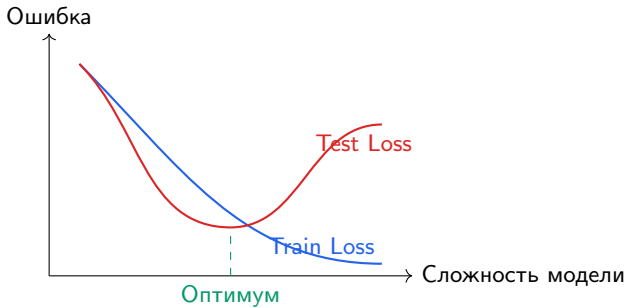
Нельзя проверять модель на тех же данных, на которых она училась.

> **Правило:** 80% данных — на обучение (Train), 20% — на проверку (Test).

- **Обучающая выборка (Train):** Модель настраивает веса.
- **Тестовая выборка (Test):** Оценивается реальное качество (модель видит их впервые).

Переобучение (Overfitting)

Главный враг Data Scientist.



- **Недообучение:** Модель слишком простая.
- **Переобучение:** Модель "зазубрила" тренировочные данные, но плохо работает на новых.

Кластеризация

Разбиение объектов на группы (кластеры) так, чтобы внутри группы были похожи, а между — различны.

Реальный кейс (Маркетинг): Сегментация клиентов. Банк хочет разделить клиентов на группы по кредитной нагрузке и активности. Заранее эти группы неизвестны.

Ассоциативные правила

Поиск закономерностей, показывающих, как связаны события.

- Как часто товары встречаются вместе.
- Как часто покупают B, если купили A.

Почему Python для Data Mining?

Преимущества

- Простота синтаксиса.
- Огромная экосистема (библиотеки).
- Стандарт индустрии.

Инструменты

- **Jupyter / Colab** — среда.
- **NumPy** — массивы.
- **Pandas** — таблицы.
- **Scikit-learn** — ML.

Google Colab / Jupyter Notebook

Интерактивная среда, где код, текст и графики живут вместе.

```
# This is a code cell (In [1])
import pandas as pd
df = pd.DataFrame({'A': [1, 2], 'B': [3, 4]})
df.head()
```

- Запуск ячейки: `Shift + Enter`.
- Переменные сохраняются в памяти между ячейками.

Переменные и базовые типы

```
# Variables (dynamic typing)
x = 10              # int
y = 3.14            # float
name = "Data"       # str
is_active = True    # bool
```

Строки и математика

```
# Math operations
print(10 + 5)    # 15
print(10 / 3)    # 3.333
print(10 // 3)   # 3 (integer division)
print(2 ** 3)    # 8 (power)

# Strings
s1 = "Data"
s2 = "Mining"
print(s1 + " " + s2) # Data Mining
```

Списки и Кортежи

```
# List (mutable - changeable)
my_list = [1, 2, 3, 4]
my_list.append(5) # [1, 2, 3, 4, 5]

# Tuple (immutable - unchangeable)
coords = (10.5, 20.3)
```

Словари (dict)

```
# Dictionary (key-value pairs)
student = {
        "name": "Ivan",
        "age": 20,
        "grades": [4, 5, 5]
}

print(student["name"]) # Ivan
```

Циклы и условия

```
x = 10

# Conditional
if x > 5:
        print("> 5")
elif x == 5:
        print("== 5")
else:
        print("< 5")

# Loop
for i in range(3):
        print(i) # 0, 1, 2
```

Функции

```
# Function definition
def calculate_area(width, height):
area = width * height
return area

# Function call
result = calculate_area(5, 10)
print(result) # 50
```

Генераторы списков (List Comprehensions)

«Питоничный» способ быстрого создания списков.

```
# Standard loop
squares = []
for x in range(5):
squares.append(x**2)

# List Comprehension (faster & cleaner)
squares = [x**2 for x in range(5)]
```

NumPy: Векторные вычисления

Основа всей экосистемы. Быстрые математические операции над массивами.

```
import numpy as np

# Create array
arr = np.array([1, 2, 3, 4, 5])

# Vectorization (no loops!)
arr_squared = arr ** 2  # [1, 4, 9, 16, 25]

# Statistics
mean_val = np.mean(arr) # 3.0
```

NumPy: Двумерные массивы

```
# Matrix (2D array)
matrix = np.array([
[1, 2, 3],
[4, 5, 6]
])

# Shape (rows, columns)
print(matrix.shape) # (2, 3)

# Access element (row 0, column 1)
print(matrix[0, 1]) # 2
```

Pandas: DataFrame и Series

Главный инструмент аналитика.

```
import pandas as pd

# Series (1D)
s = pd.Series([10, 20, 30])

# DataFrame (2D table)
df = pd.DataFrame({
        "Name": ["Ivan", "Anna", "Petr"],
        "Age": [25, 30, 22],
        "City": ["Moscow", "SPb", "Kazan"]
})
```

Pandas: Загрузка и просмотр

```
# Load from CSV
df = pd.read_csv("data.csv")

# First 5 rows
df.head()

# Info about types and missing values (NaN)
df.info()

# Basic statistics (mean, min, max)
df.describe()
```

Pandas: Индексация (loc и iloc)

```
# loc[] - by label (names)
df.loc[0, "Name"] # row 0, column "Name"

# iloc[] - by integer position
df.iloc[0, 1]     # row 0, column 1
```

Pandas: Фильтрация и Группировка

```
# Filter (Age > 25)
adults = df[df['Age'] > 25]

# Group by (Average Age by City)
avg_age = df.groupby('City')['Age'].mean()
```

Pandas: Объединение (Merge)

Аналог SQL JOIN. Объединяет данные из разных таблиц по ключу.

```
# Merge two DataFrames by 'id'
merged = pd.merge(df_clients, df_orders, on='id', how='left')
```

Визуализация (Matplotlib/Seaborn)

```
import matplotlib.pyplot as plt
import seaborn as sns

# Scatter plot
sns.scatterplot(data=df, x='Age', y='Salary', hue='City')
plt.title("Age vs Salary")
plt.show()
```