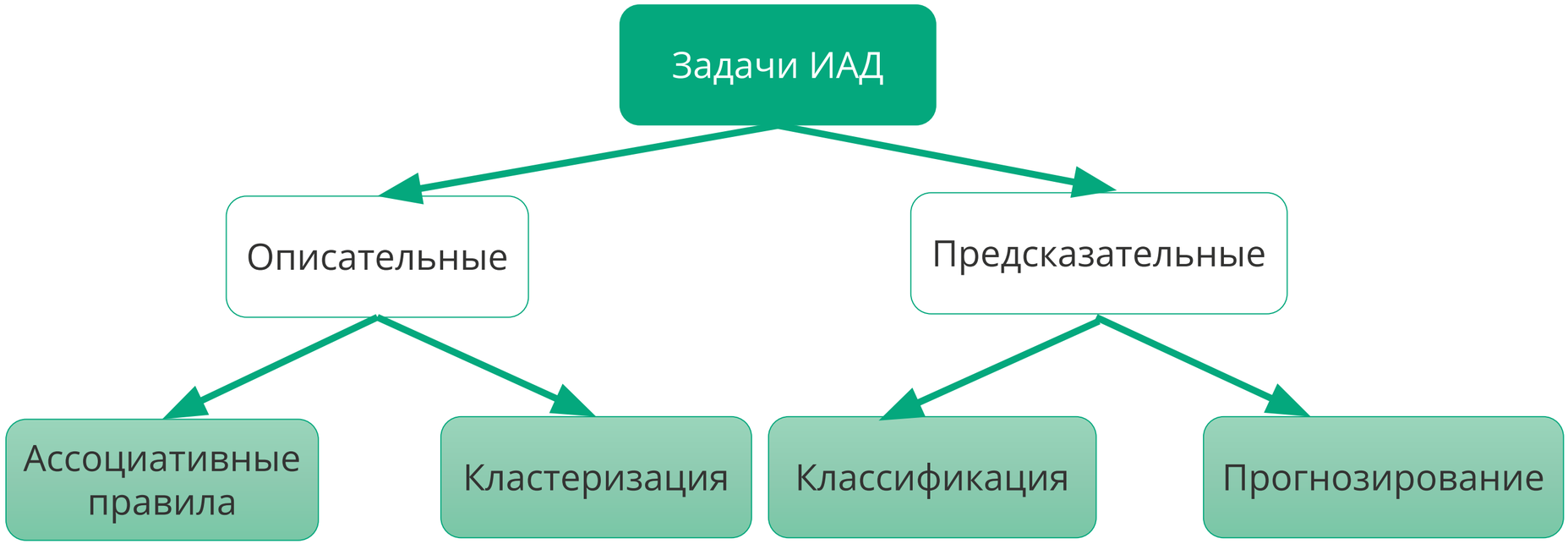
Ассоциативные правила в маркетинге и медицине

“Data Mining – это процесс поддержки принятия решений, основанных на поиске в данных скрытых закономерностей. Это технология, которая предназначена для поиска в больших объемах данных неочевидных, но объективных и полезных на практике закономерностей”

Задачи интеллектуального анализа данных



Ассоциативные правила

Цель задачи поиска ассоциативных правил заключается в обнаружении закономерностей между связанными событиями.

Product placement in Tesco, UK.

1. Анализ рыночных корзин
   - Планирование закупки и размещения товаров
   - Проведение акций и скидок
   - Проведение рекламной кампании
   - Создание новых видов товаров (X со вкусом Y)
2. Сфера услуг
   - Объединение сервисов X и Y в один пакет услуг
3. Медицина
   - Анализ сочетания болезней и симптомов
   - Выявление коморбидных заболеваний

Цель метода ассоциативных правил

Ассоциативным правилом называется импликация $X \Rightarrow Y$, где $X, Y \subseteq I, X \cap Y \neq \emptyset$,

Ассоциативные правила имеют следующий вид:
если (условие) то (результат), где условие – это набор объектов из множества *I* , с которыми связаны (ассоциированы) объекты, включенные в результат данного правила

Поддержка (Support) ассоциативного правила показывает то, как часто в базе данных вместе встречаются X и Y (т.е . это доля строк в базе данных, в которых одновременно присутствуют X и Y).

$$Support(X \Rightarrow Y) = Support(X \cup Y) = \frac{|T_{X \cup Y}|}{|T|}$$

Наборы, встречаемость которых выше заданного порога $Support_{min}$ считаются **часто встречающимися**.

Support {🍎} = $\frac{4}{8}$

Transaction 1	🍎 🍬 ⚪ 🍗
Transaction 2	🍎 🍬 ⚪
Transaction 3	🍎 🍬
Transaction 4	🍎 🍐
Transaction 5	🍼 🍬 ⚪ 🍗
Transaction 6	🍼 🍬 ⚪
Transaction 7	🍼 🍬
Transaction 8	🍼 🍐

Confidence

Достоверность (Confidence) показывает, как часто в записях базы данных, содержащих X, одновременно присутствует Y , т.е. оценивает вероятность того, что из наличия записи набора X следует наличие в ней набора Y

$$Confidence(X \Rightarrow Y) = \frac{Support(X \Rightarrow Y)}{Support(X)}$$

Confidence {🍎 → 🍬} = $\frac{\text{Support } \{🍎, 🍬\}}{\text{Support } \{🍎\}} = \frac{3}{4}$

Transaction 1	🍎 🍬 ⚪ 🍗
Transaction 2	🍎 🍬 ⚪
Transaction 3	🍎 🍬
Transaction 4	🍎 🍐
Transaction 5	💉 🍬 ⚪ 🍗
Transaction 6	💉 🍬 ⚪
Transaction 7	💉 🍬
Transaction 8	💉 🍐

Lift

Улучшение (Lift) отражает то, как часто товары X и Y появляются вместе, одновременно учитывая, с какой частотой появляется каждый из них. Лифт правила $X \Rightarrow Y$ - это достоверность (Confidence), деленная частоту появления (Support) Y.

Значение lift *больше 1* означает, что Y *чаще встречается* вместе с X, чем по-отдельности.

$$Lift(X \Rightarrow Y) = \frac{Support(X \Rightarrow Y)}{Support(X) \cdot Support(Y)}$$

Lift {🍎 → 🍬} = $\frac{\text{Support } \{🍎, 🍬\}}{\text{Support } \{🍎\} \text{ x Support } \{🍬\}} = 1$

Transaction 1	🍎 🍬 ⚪ 🍗
Transaction 2	🍎 🍬 ⚪
Transaction 3	🍎 🍬
Transaction 4	🍎 🍐
Transaction 5	💉 🍬 ⚪ 🍗
Transaction 6	💉 🍬 ⚪
Transaction 7	💉 🍬
Transaction 8	💉 🍐

Сравнение Confidence и Lift на примере

1. Конфеты и Вода обладают наибольшей поддержкой, что обеспечивает высокую достоверность правила *{Конфеты => Вода}*. Однако, Lift = 1 указывает на отсутствие значимых ассоциаций между этими товарами.
2. Правило *{Конфеты => Косметика}* обладает низкой достоверностью вследствие невысокой поддержки Косметики. Однако, если покупается косметика, то, вероятнее всего, с этим товаром покупают Конфеты, что подтверждает Lift = 2,0.

Transaction	Support
Конфеты	0.1
Вода	0.2
Ягоды	0.03
Косметика	0.005

Transaction	Support	Confidence	Lift
Конфеты => Вода	0.01	0,2	1,0
Конфеты => Ягоды	0,001	0.01	0,3
Конфеты => Косметика	0,001	0.01	2,0

Алгоритм Apriori

I. **Поиск часто встречающихся наборов элементов**
   1. k=1. Найти *частые* 1-элементные наборы с поддержкой $Support > Support_{min}$.
   2. k=k+1. Если возможно сгенерировать k-элементные наборы, то шаг 3, иначе - шаг 5.
   3. Сгенерировать k-элементные наборы на основе наборов из (k-1) элементов.
   4. Вычислить поддержку каждого кандидата и удалить нечастые k-элементные наборы. Шаг 2.
   5. Выдать все результирующие наборы k элементов для всех k.

II. **Извлечение из наборов ассоциативных правил**
   1. Сгенерировать правила с учетом пороговой достоверности для всех частых наборов

1-элементные наборы

Набор	Число
Хлеб	4%
Кола	2%
Молоко	4%
Конфеты	3%
Масло	4%
Яйца	1%

2-элементные наборы

Набор	Число
{Хлеб, Молоко}	3%
{Хлеб, Конфеты}	2%
{Хлеб, Масло}	3%
{Молоко, Конфеты}	2%
{Молоко, Масло}	3%
{Конфеты, Масло}	3%

Min support = 3%

3-элементные наборы

Набор	Число
{Хлеб, Молоко, Масло}	3

Уменьшение числа кандидатов

Создание кандидатов – затратная операция, которая может привести к созданию множеств кандидатов, имеющих большой объем.

- Принцип Apriori: Набор из k элементов будет часто встречаться, если все его подмножества из k-1 элемента тоже были часто встречающимися
- Свойство антимонотонности поддержки: с ростом размера набора элементов поддержка уменьшается или остается такой же

1-элементные наборы

Набор	Число
Хлеб	4%
Кола	2%
Молоко	4%
Конфеты	3%
Масло	4%
Яйца	1%

2-элементные наборы

Набор	Число
{Хлеб, Молоко}	3%
{Хлеб, Конфеты}	2%
{Хлеб, Масло}	3%
{Молоко, Конфеты}	2%
{Молоко, Масло}	3%
{Конфеты, Масло}	3%

Min support = 3

3-элементные наборы

Набор	Число
{Хлеб, Молоко, Масло}	3%

Полный перебор: $2^6 = 64$

При рассмотрении каждого подмножества: $C_6^1 + C_6^2 + C_6^3 = 41$

С учетом свойства антимонотонности поддержки: $C_6^1 + C_4^2 + C_3^3 = 13$

Построение ассоциативных правил на Python

```
import pandas as pd
```

```
groceries = pd.read_csv('Groceries.csv', header = None)
pd.set_option('display.max_columns', 8)
```

```
groceries.head()
```

	0	1	2	3	...	28	29	30	31
0	citrus fruit	semi-finished bread	margarine	ready soups	...	NaN	NaN	NaN	NaN
1	tropical fruit	yogurt	coffee	NaN	...	NaN	NaN	NaN	NaN
2	whole milk	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
3	pip fruit	yogurt	cream cheese	meat spreads	...	NaN	NaN	NaN	NaN
4	other vegetables	whole milk	condensed milk	long life bakery product	...	NaN	NaN	NaN	NaN

5 rows × 32 columns

```
transactions = [[product for product in transaction if not pd.isnull(product)] for transaction in groceries.values]
```

Построение ассоциативных правил на Python

Apriori

```
from apyori import apriori
apriori_rules = list(apriori(transactions, min_support=0.003,  min_confidence=0.7, min_lift=1.07, min_length=2))
```

Ассоциативные правила (вывод алгоритма Apriori)

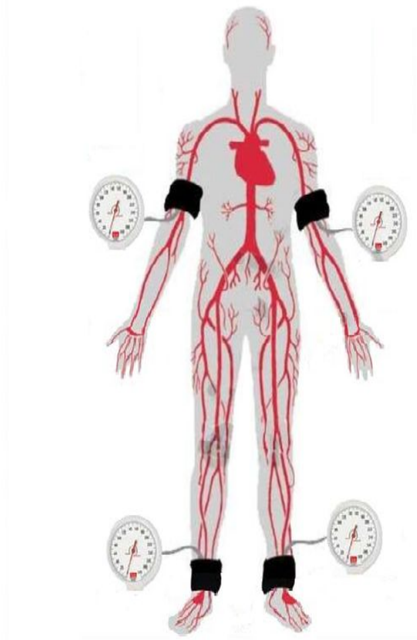
```
(yogurt, baking powder) ==> (whole milk) sup = 0.003, conf = 0.711, lift = 2.783
(coffee, butter) ==> (whole milk) sup = 0.003, conf = 0.702, lift = 2.748
(butter, curd) ==> (whole milk) sup = 0.005, conf = 0.716, lift = 2.804
(butter, onions) ==> (whole milk) sup = 0.003, conf = 0.75, lift = 2.935
(pork, butter) ==> (whole milk) sup = 0.004, conf = 0.704, lift = 2.754
(domestic eggs, curd) ==> (whole milk) sup = 0.005, conf = 0.734, lift = 2.874
(sugar, domestic eggs) ==> (whole milk) sup = 0.004, conf = 0.714, lift = 2.795
(other vegetables, root vegetables, brown bread) ==> (whole milk) sup = 0.003, conf = 0.775, lift = 3.033
(yogurt, root vegetables, butter) ==> (whole milk) sup = 0.003, conf = 0.789, lift = 3.09
(yogurt, butter, tropical fruit) ==> (whole milk) sup = 0.003, conf = 0.733, lift = 2.87
(citrus fruit, root vegetables, tropical fruit) ==> (other vegetables) sup = 0.004, conf = 0.786, lift = 4.061
(tropical fruit, yogurt, curd) ==> (whole milk) sup = 0.004, conf = 0.75, lift = 2.935
(whipped/sour cream, other vegetables, domestic eggs) ==> (whole milk) sup = 0.004, conf = 0.7, lift = 2.74
(whipped/sour cream, root vegetables, tropical fruit) ==> (other vegetables) sup = 0.003, conf = 0.733, lift = 3.79
(root vegetables, yogurt, tropical fruit) ==> (whole milk) sup = 0.006, conf = 0.7, lift = 2.74
(whipped/sour cream, yogurt, tropical fruit) ==> (whole milk) sup = 0.004, conf = 0.705, lift = 2.759
(other vegetables, citrus fruit, root vegetables, tropical fruit) ==> (whole milk) sup = 0.003, conf = 0.705, lift = 2.757
(other vegetables, root vegetables, yogurt, tropical fruit) ==> (whole milk) sup = 0.004, conf = 0.714, lift = 2.795
```

Медицинский набор данных и маркёры атеросклероза

	Hypertension	Stenocardia	Infarction	Stroke	Diabetes	CardiacFailure	LegsIndex	ABI	ArmsIndex	Age	Gender	SBPra	DBPra	Pulse
0	0	0	0	0	0	0	0	0	0	18	Female	120	84	80
1	0	0	0	0	0	0	1	1	0	18	Male	154	78	65
2	0	0	1	0	0	0	0	0	0	19	Female	114	71	89
3	1	0	0	0	0	0	0	0	0	19	Female	117	69	100
4	0	0	0	0	1	0	0	0	0	19	Male	114	62	65

ArmsIndex =1, if |ΔSBPa|=| SBPra - SBPla |≥15; else 0
LegsIndex = 1, if |ΔSBPl|=| SBPrl - SBPll |≥15; else 0
ABI = 1, if SBPl(r)l/ SBPl(r)a≤0.9; else 0

ABI - Ankle-Brachial Index
SBP – Systolic Blood Pressure



Python: обработка категориальных признаков

	Hypertension	Stenocardia	Infarction	Stroke	Diabetes	CardiacFailure	LegsIndex	ABI	ArmsIndex
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	0
2	0	0	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0

```
[['LegsIndex', 'ABI'],
 ['Infarction'],
 ['Hypertension'],
 ['Diabetes'],
 ['Hypertension']]
```

```
records = df.replace({'Hypertension': {1: "Hypertension"},
            'Stenocardia': {1: "Stenocardia"},
            'Infarction': {1: "Infarction"},
            'Stroke': {1: "Stroke"},
            'Diabetes': {1: "Diabetes", 2: "Diabetes_Possible"},
            'CardiacFailure': {1: "CardiacFailure"},
            'LegsIndex': {1: "LegsIndex"},
            'ABI': {1: "ABI"},
            'ArmsIndex': {1: "ArmsIndex"},
            'Infarction': {1: "Infarction"},
           })
records.head()
transactions = [[product for product in transaction if product!=0] for transaction in records.values]
transactions = list(filter(None, transactions))
```

Python: обработка числовых признаков

	Age	Gender	SBPra	DBPra	Pulse
0	18	Female	120	84	80
1	18	Male	154	78	65
2	19	Female	114	71	89
3	19	Female	117	69	100
4	19	Male	114	62	65

```
[['Gender_Female', 'Normal_SBP', 'Normal_DBP', 'Age_Junior', 'Pulse_Normal'],
 ['Gender_Male', 'High_SBP', 'Normal_DBP', 'Age_Junior', 'Pulse_Normal'],
 ['Gender_Female', 'Low_SBP', 'Normal_DBP', 'Age_Junior', 'Pulse_Normal'],
 ['Gender_Female', 'Low_SBP', 'Normal_DBP', 'Age_Junior', 'Pulse_High'],
 ['Gender_Male', 'Low_SBP', 'Normal_DBP', 'Age_Junior', 'Pulse_Normal']]
```

```
binsDbp = [0, 60, 90, 110]
binsSbp = [0, 119, 140, 160]
binsAge = [0, 20, 50, 70]
binsPulse = [0, 60, 90, 110]
group_names_SBP = ["Low_SBP","Normal_SBP","High_SBP"]
group_names_DBP = ["Low_DBP","Normal_DBP","High_DBP"]
group_names_Age = ["Age_Junior","Age_Adult","Age_Senior"]
group_names_Pulse = ["Pulse_Low","Pulse_Normal","Pulse_High"]
df['SBPra_Status'] = pd.cut(df['SBPra'], binsSbp, labels=group_names_SBP)
df['DBPra_Status'] = pd.cut(df['DBPra'], binsDbp, labels=group_names_DBP)
df['Age_Status'] = pd.cut(df['Age'], binsAge, labels=group_names_Age)
df['Pulse_Status'] = pd.cut(df['Pulse'], binsPulse, labels=group_names_Pulse)
```

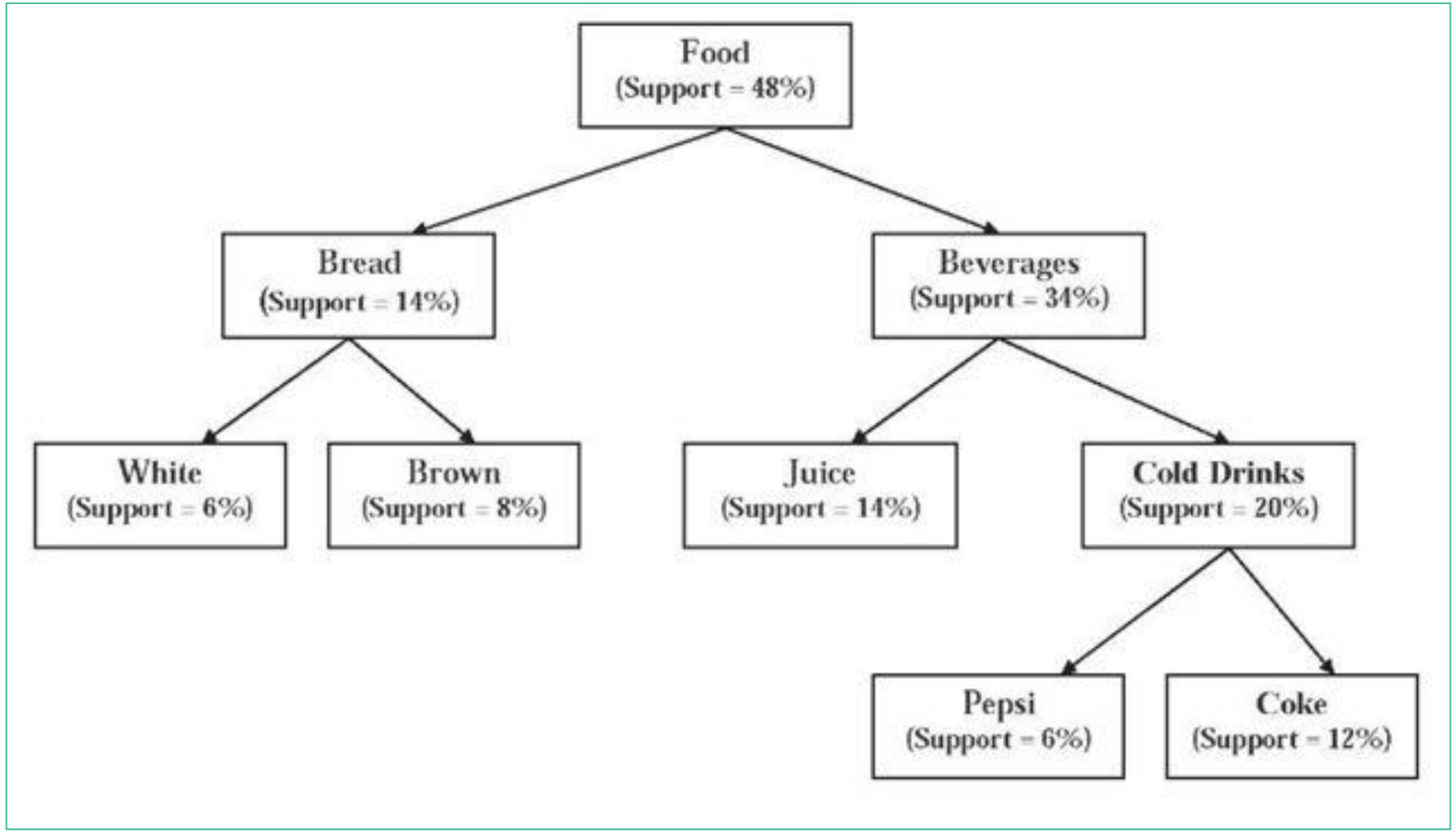
Пример набора правил

Классификация правил:

1. Полезные
2. Тривиальные
3. Неочевидные

Rule	Support	Confidence	Lift
Obesity, SBPra_High, Gender_Female ==> ArmsIndex	0.03	0.20	2.30
Glucose_High ==> Diabetes	0.08	0.72	5.39
Age_Senior, Obesity, Stenocardia ==> Cardiac Failure	0.03	0.56	5.24
Hypertension, ABI ==> PPla_High	0.02	0.89	2.44
SBPra_High, Smoker ==> Gender_Male	0.05	0.90	3.63
Gender_Female, Age_Senior, Abdominal Obesity ==> Cardiac Failure	0.05	0.30	2.78

Построение обобщенных правил



Секвенциальный анализ

Дата	Время	Источник сбоя	Код ошибки
01.01.2019	15:04:23	Станция 1	A
01.01.2019	16:45:46	Станция 1	F
01.01.2019	18:32:26	Станция 4	Z
01.01.2019	20:07:11	Станция 5	H
01.01.2019	20:54:43	Станция 1	Q

$$T_1 = \{(A, 15{:}04{:}23), (F, 16{:}45{:}46\ ), (Q, 20{:}54{:}43\ )\}$$