# Определения (Лекция 17)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1й набор (ni)** | **pi** | **2й набор (ni)** | **pi** | **3й набор (ni)** | **pi** | **4й набор (ni)** | **pi** | **5й набор (ni)** | **pi** | **6й набор (ni)** | **pi** |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 2 | 0.00 | 510 | 1 | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 100 | 0.20 |  |  | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 2 | 0.00 |  |  | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 251 | 0.49 | 100 | 0.20 |  |  | 15 | 0.45 |  |
| 51 | 0.1 | 501 | 0.98 | 1 | 0.00 | 2 | 0.00 |  |  | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 251 | 0.49 | 100 | 0.20 |  |  | 10 | 0.30 |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 2 | 0.00 |  |  | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 100 | 0.20 |  |  | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 2 | 0.00 |  |  | 1 | 0.03 |  |
| 51 | 0.1 | 1 | 0.00 | 1 | 0.00 | 100 | 0.20 |  |  | 1 | 0.03 |  |
| Энтропия | 2.30 |  | 0.13 |  | 0.80 |  | 1.71 |  | 0 |  | 1.57 |  |
| Ширина спектра | 10.00 |  | 1.14 |  | 2.22 |  | 5.51 |  | 1 |  | 4.80 |  |
| ln(n)= | 2.30 |  |  |  |  |  |  |  |  |  |  |  |

**Применение информационной энтропии:**

(Интерпретация p здесь: событие №1 наступило 51 раз, событие № 2 тоже 51 раз и тд, найдите вероятность каждого события).

****

Случай 1: Товар 0 встречался с товарами 1-10 одинаковое количество раз во всех 510 документах, то есть 51 раз с каждым товаром.

Случай 2: Товар 0 встретился с товаром 5 501 раз, с остальными по 1 разу.

Случай 3: Товар встретился с товарами 4 и 6 по 251 разу, с остальными по 1 разу.

**Коммент:** Отдельный вопрос, а нужно ли учитывать в отдельное событие случаи, когда товар продается в документе один?

**Задание:** Напишите запрос, который по таблице @Документы (ндок, Товар\_ID, primary key(ндок, Товар\_ID)) вычисляет такую энтропию по товару, заданному параметром.

-- В скольких документах товар\_1 встречался вместе с товаром\_2:

DECLARE @Пары TABLE (Товар\_ID\_1 int, Товар\_ID\_2 int, Раз int)

INSERT INTO @Пары

SELECT Д1.Товар\_ID, Д2.Товар\_ID, COUNT(\*) Раз

FROM @Документы Д1 INNER JOIN @Документы Д2 ON

Д1.ндок = Д2.ндок AND

Д1.Товар\_ID <> Д2.Товар\_ID

GROUP BY Д1.Товар\_ID, Д2.Товар\_ID

-- Для каждого товара: а сколько раз для товара есть событие: быть с чем-то в паре.

DECLARE @ПоТовару TABLE (Товар\_ID int, ВПаре int)

INSERT INTO @ПоТовару

SELECT Д1.Товар\_ID, COUNT(\*)

FROM @Документы Д1

INNER JOIN @Документы Д2 ON

Д1.ндок = Д2.ндок AND

Д1.Товар\_ID <> Д2.Товар\_ID

GROUP BY Д1.Товар\_ID

-- Расчет ширины спектра

SELECT Товар\_ID\_1, EXP(-SUM(p\*LOG(p))) ШиринаСпектра

FROM

(

SELECT Пары.Товар\_ID\_1, Пары.Товар\_ID\_2,

CAST(Пары.Раз as float)/ ПоТовару.ВПаре p

FROM @Пары Пары LEFT JOIN @ПоТовару ПоТовару ON

Пары.Товар\_ID\_1 = ПоТовару.Товар\_ID

) Итог

GROUP BY Товар\_ID\_1

**Другие способы написания последнего запроса:**

|  |  |
| --- | --- |
| SELECT Товар\_ID\_1,  EXP(-SUM(p\*LOG(p))) ШиринаСпектра  FROM  (  SELECT Товар\_ID\_1, Товар\_ID\_2,  CAST(Раз as float)/  (  SELECT SUM(Раз)  FROM @Пары Пары  WHERE Пары.Товар\_ID\_1 = **П.Товар\_ID\_1**  ) p  FROM @Пары П  ) T  GROUP BY Товар\_ID\_1 | SELECT Товар\_ID\_1,  EXP(-SUM(p\*LOG(p))) ШиринаСпектра  FROM  (  SELECT Товар\_ID\_1, Товар\_ID\_2,  CAST(Раз as float)/  (  **SUM(Раз)OVER(PARTITION BY Товар\_ID\_1)**  ) p  FROM @Пары  ) T  GROUP BY Товар\_ID\_1 |

**Пример использования:**

На Вашем сайте для каждой товарной группы отображены параметры, пользуясь которыми покупатель может отфильтровать для себя список товаров. Обычно свойств у товаров очень много и не всегда можно предугадать, что конкретно будет наиболее важно для покупателя. Поэтому Вы можете собрать статистику, какие свойства чаще всего используются покупателями для фильтрации, и на основе этой информации изменить интерфейс отображения параметров на сайте.

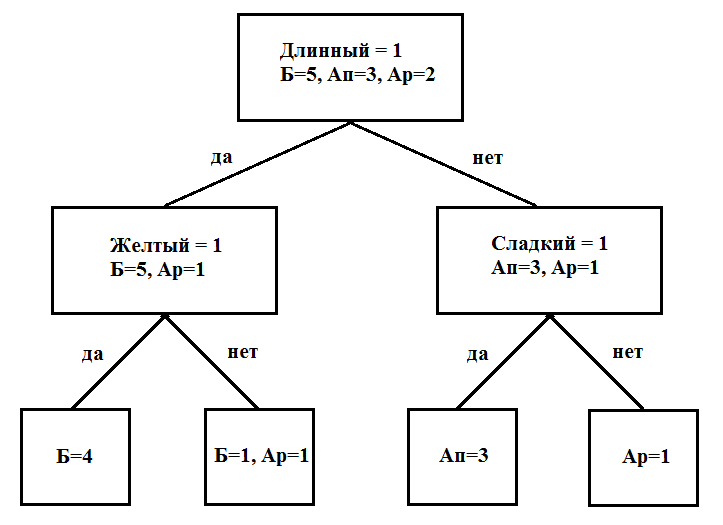
**Пример использования энтропии в анализе данных (Classification Tree):**

Энтропия – как один из критериев выбора параметра в узле дерева.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Класс** | **Длинный** | **Сладкий** | **Желтый** | | банан | 1 | 1 | 1 | | банан | 1 | 1 | 1 | | банан | 1 | 1 | 0 | | банан | 1 | 0 | 1 | | банан | 1 | 1 | 1 | | апельсин | 0 | 0 | 0 | | апельсин | 0 | 0 | 1 | | апельсин | 0 | 0 | 1 | | арбуз | 1 | 1 | 0 | | арбуз | 0 | 1 | 0 | | p\_банан = 0.5, p\_апельсина = 0.3, p\_арбуза = 0.2  S ~ 1.49  «Длинный»: **S = 0.71**  Д = 1: S1 = -5/6\*log(5/6,2) – 1/6\*log(1/6, 2) ~ 0.65  Д = 0: S2 = -3/4\*log(3/4,2) – 1/4\*log(1/4, 2) ~ 0.81  S = 0.65\*0.6 + 0.81\*0.4 ~ 0.71   |  |  |  |  | | --- | --- | --- | --- | | Длинный | банан | апельсин | арбуз | | 1 | 5 | 0 | 1 | | 0 | 0 | 3 | 1 |   «Сладкий»: S= 0.88   |  |  |  |  | | --- | --- | --- | --- | | Сладкий | банан | апельсин | арбуз | | 1 | 4 | 0 | 2 | | 0 | 1 | 3 | 0 |   «Желтый»: S = 1.15   |  |  |  |  | | --- | --- | --- | --- | | Желтый | банан | апельсин | арбуз | | 1 | 4 | 2 | 0 | | 0 | 1 | 1 | 2 | |

При первом делении энтропия сильнее всего изменилась при делении по критерию «длинный».

Далее аналогично. Обратите внимание, что на одном уровне дерева свойства для классификации могут отличаться.



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | ндок | Начало | Конец | | 1 | 12:00 | 13:30 | | 2 | 13:00 | 17:00 | | 3 | 14:00 | 16:00 | | 4 | 17:30 | 19:00 | | |  |  |  | | --- | --- | --- | | ндок | Время | Флаг | | 1 | 12:00 | 1 | | 2 | 13:00 | 1 | | 1 | 13:30 | -1 | | 3 | 14:00 | 1 | | 3 | 16:00 | -1 | | 2 | 17:00 | -1 | | 4 | 17:30 | 1 | | 4 | 19:00 | -1 | | |  |  |  |  | | --- | --- | --- | --- | | ндок | Время | Флаг | Сумма | | 1 | 12:00 | 1 | 1 | | 2 | 13:00 | 1 | 2 | | 1 | 13:30 | -1 | 1 | | 3 | 14:00 | 1 | 2 | | 3 | 16:00 | -1 | 1 | | 2 | 17:00 | -1 | 0 | | 4 | 17:30 | 1 | 1 | | 4 | 19:00 | -1 | 0 | |

**Пример решения:**

|  |  |
| --- | --- |
| SET DATEFORMAT DMY  DECLARE @date1 datetime, @date2 datetime  SET @date1 = '20210301 5:00:00'  SET @date2 = '20210401 5:00:00'  -- создаем список заданий в удобном виде. Таблица задания – исходная таблица в формате ндок, Начало, Конец  DECLARE @Список TABLE (ндок int, День datetime, Время datetime, Флаг int)  INSERT INTO @Список(ндок, День, Время, Флаг)  SELECT ндок, CAST(Начало as date) День, Начало Время, 1 Нач  FROM Задания  WHERE Начало >= @date1 AND Начало < @date2  UNION  SELECT ндок, CAST(Начало as date) День, Конец, -1  FROM Задания  WHERE Начало >= @date1 AND Начало < @date2 | -- Вариант 1  -- итоговый подсчет эффективного времени по дням  SELECT День,  SUM(CASE  WHEN Флаг = 1 AND Сумм = 1  THEN -1\*CAST(Время as float)  WHEN Флаг = -1 AND Сумм = 0  THEN 1\*CAST(Время as float)  END)\*24 Длтн  FROM  (  SELECT Список.ндок, Список.День, Список.Время, Список.Флаг, SUM(Список\_1.Флаг) Сумм  FROM @Список Список  INNER JOIN @Список Список\_1 ON  Список.Время >= Список\_1.Время    GROUP BY Список.ндок, Список.День, Список.Время, Список.Флаг  ) T  GROUP BY День |

**Проблемы**:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | ндок | Начало | Конец | | 1 | 12:00 | 13:30 | | 2 | 13:00 | **17:00** | | 3 | 14:00 | **17:00** | | 4 | 17:30 | 19:00 | | |  |  |  |  | | --- | --- | --- | --- | | ндок | Время | Флаг | Сумма | | 1 | 12:00 | 1 | 1 | | 2 | 13:00 | 1 | 2 | | 1 | 13:30 | -1 | 1 | | 3 | 14:00 | 1 | 2 | | **3** | **17:00** | **-1** | **0** | | **2** | **17:00** | **-1** | **0** | | 4 | 17:30 | 1 | 1 | | 4 | 19:00 | -1 | 0 | |

Решение при помощи ranking function (кстати, с 2012 версии накопительную сумму можно считать сразу готовыми оконными функциями, см прошлый семинар):

|  |  |
| --- | --- |
| DECLARE @СписокНомера TABLE  (ндок int, День datetime, Время datetime, Флаг int, НомерСтроки int)  INSERT INTO @СписокНомера(ндок, День, Время, Флаг, НомерСтроки)  SELECT ндок, День, Время, Флаг,  ROW\_NUMBER()  OVER(ORDER BY Время, ндок) НомерСтроки  FROM @Список T | SELECT День, SUM(CASE  WHEN Флаг = 1 AND Сумм = 1 THEN -1\*CAST(Время as float)  WHEN Флаг = -1 AND Сумм = 0 THEN 1\*CAST(Время as float)  END)\*24 Длтн  FROM (  SELECT Список.ндок, Список.День, Список.Время, Список.Флаг, SUM(Список\_1.Флаг) Сумм  FROM @СписокНомера Список  INNER JOIN @СписокНомера Список\_1 ON  Список.НомерСтроки > = Список\_1.НомерСтроки  GROUP BY Список.ндок, Список.День, Список.Время, Список.Флаг  ) T  GROUP BY День |

**Курсор – нереляционный результат с гарантированным порядком строк.**

|  |  |
| --- | --- |
| Типы:  - последовательные и прокручиваемые.  - статические, динамические и управляемые набором ключей. | Основные действия:   * Создание или объявление курсора; (DECLARE) * Открытие курсора (наполнение курсора данными); (OPEN) * Выборка из курсора и изменение с его помощью строк данных; (FETCH) * Закрытие курсора (после чего он становится недоступным для пользовательских программ); (CLOSE) * «Освобождение» курсора (удаление курсора как объекта). (DEALLOCATE). |

Решение нашей задачи при помощи курсоров:

CREATE TABLE #СписокКурсор (ндок int, День datetime, Время datetime, Флаг int, Сумм int, primary key(ндок, День, Время, Флаг))

INSERT INTO #СписокКурсор (ндок, День, Время, Флаг, Сумм)

SELECT ндок, День, Время, Флаг, 0 FROM @Список

DECLARE @day datetime, @date datetime, @dok int, @flag int, @sum int

SET @sum = 0

DECLARE data\_cursor CURSOR FOR

SELECT ндок, День, Время, Флаг

FROM #СписокКурсор

ORDER BY День, Время, ндок, Флаг

OPEN data\_cursor

FETCH NEXT FROM data\_cursor -- «помещаем» первую строку из таблицы в переменные

INTO @dok, @day, @date, @flag

WHILE @@FETCH\_STATUS = 0

BEGIN

-- переменная @sum будет хранить в себе сумму флагов по всем строкам, которые выше текущей по дате/времени, номеру документа и флагу

SET @sum = @sum + @flag

-- проставим значение суммы в таблицу:

-- такая версия сработает всегда

--UPDATE #СписокКурсор SET Сумм = @sum WHERE ндок = @dok AND День = @day AND Время = @date AND Флаг = @flag

-- а эта только в случае, если Вы заведете в таблице #СписокКурсор ключи

UPDATE Список SET Сумм = @sum FROM #СписокКурсор Список WHERE CURRENT OF data\_cursor

FETCH NEXT FROM data\_cursor -- помещаем следующую строку в переменные

INTO @dok, @day, @date, @flag

-- продолжаем действия в цикле, пока строки курсора не закончатся

END

CLOSE data\_cursor

DEALLOCATE data\_cursor

SELECT День, SUM(CASE

WHEN Флаг = 1 AND Сумм = 1 THEN -1\*CAST(Время as float)

WHEN Флаг = -1 AND Сумм = 0 THEN 1\*CAST(Время as float)

END)\*24 Длтн

FROM #СписокКурсор T

GROUP BY День

DROP TABLE #СписокКурсор