Конспект по теме «Когортный анализ»

Когортный анализ

Вкладывая деньги в рекламу, бизнес стремится привлечь самых «качественных» клиентов — тех, кто принесёт больше всего денег. Чтобы оценить качество клиентов, применяют когортный анализ.

Аналитики делят клиентов компании на группы — когорты. Участников всех когорт объединяет какое-нибудь **общее событие**. Чаще всего это первое посещение сайта, регистрация или скачивание мобильного приложения.

После этого их делят на когорты по выбранным критериям:

- **Время,** в которое произошло общее событие.
- **Дополнительные признаки.** Например, возраст и пол, профессия, география, поведенческие особенности.

В зависимости от задачи можно формировать самые разные когорты — лишь бы все участники одной когорты совершили одно и то же действие в определённое время. Дополнительные признаки вводят, если они важны для анализа.

Помимо сравнения качества клиентов разных групп, когортный анализ позволяет выявить тренды в поведении пользователей. Это метод ретроспективного анализа — когортный анализ редко применяют для прогнозирования поведения пользователей, ведь в будущем на него могут повлиять внешние факторы.

Профиль пользователя. Функции first() и last()

Чтобы проще было делить клиентов на когорты, лучше заранее составить их профили — таблицу с деталями первого посещения каждого пользователя, такими как источник перехода на сайт, страна, устройство.

Профили пользователей составляют в три этапа:

1. Загрузить данные журнала посещений.

2. Для каждого пользователя определить дату и время первой сессии.
3. Для каждого пользователя определить соответствующие задаче параметры первой сессии. Например, источник перехода на сайт.

Если передать функции `min()` строковые значения, она вернёт строку с минимальным лексикографическим весом. Поэтому при группировке для поиска даты и времени первой сессии вызывают функцию `first()`: она возвращает не минимальное, а первое значение в каждой группе. Важно заранее отсортировать данные — иначе `first()` не сработает.

```
games = games.sort_values(by=['name', 'year_of_release'])
```

У функции `min()` есть зеркальная функция `max()`, которая возвращает не минимальное, а максимальное значение. Такая же «сводная сестра» функции `first()` — функция `last()`. При группировке `last()` возвращает последнее значение в каждой группе.

```
print(
    games.groupby('name').agg(
        {'year_of_release': ['min', 'max'], 'platform': ['first', 'last']}
    )
)
```

```
             year_of_release             platform
                         min         max    first  last
name
Monopoly          1994-01-01  2010-01-01       PC    DS
SimCity 2000      1992-01-01  2003-01-01       PC   GBA
Terraria          2011-01-01  2016-01-01       PC  WiiU
```

Retention Rate, Churn Rate и горизонт анализа

В цифровых сервисах когортный анализ применяют, чтобы сравнить «качество» пользователей. Один из главных критериев качества — как долго клиент остаётся с компанией. Этот факт описывают метрики Retention Rate и Churn Rate.

Retention Rate, или коэффициент удержания, показывает, сколько пользователей из когорты относительно их изначального числа вернулись,

то есть воспользовались продуктом или услугой, в последующие периоды. `Retention Rate = Количество активных пользователей на текущий день / Количество активных пользователей на первый день`.

Churn Rate, или коэффициент отскока, показывает, какой процент пользователей прекращает использовать сервис с течением времени. `Churn Rate = 1 - (Количество пользователей на n-ый день / Количество пользователей на (n-1)-ый день)`.

При расчёте Retention Rate, Churn Rate и других метрик учитывают:

- Момент анализа — момент времени, в который вы смотрите на данные;
- Горизонт анализа — максимальный лайфтайм, который вы анализируете.

Момент анализа ограничивает возможный горизонт. Важно следить за тем, чтобы в отчёты не попадали пользователи, которые не успели «дожить» до выбранного горизонта.

Расчёт Retention Rate в Python. Функция div()

Чтобы рассчитать Retention Rate в Python, выполняют такие шаги:

- Получить журнал сессий и профили пользователей.
- Объединить данные сессий с профилями.
- Рассчитать лайфтайм пользователя для каждой сессии.
- Построить таблицу удержания. То есть сводную таблицу, в которой названия строк — это даты первого посещения пользователей, названия столбцов — лайфтайм, а значения в «ячейках» — количество уникальных идентификаторов пользователей.
- Вычислить размеры когорт и занести результаты в отдельную таблицу.
- Объединить таблицы размеров когорт и удержания.
- Разделить каждую «ячейку» таблицы удержания на соответствующий размер когорты.

Функция `div()` выполняет деление. Когда параметр `axis` равен нулю, деление выполняется построчно, или горизонтально: индексы делителя соотносятся с индексами строк. Когда параметр `axis` равен 1, деление

выполняется вертикально: индексы делителя соотносятся с нумерацией столбцов.

```
df.div([1, 2, 3], axis=0)
```

Визуализация когортного анализа

Таблицы удержания и другие показатели в когортном анализе можно визуализировать множеством способов. Наиболее подходящие — тепловая карта, кривые и динамика.

Как построить хитмэп:

```
plt.figure(figsize=(15, 6))
sns.heatmap(
    retention.drop(columns=['cohort_size']),
    annot=True,
    fmt='.2%',
)
plt.title('Тепловая карта удержания')
plt.show()
```

Как построить кривые удержания:

```
report = retention.drop(columns = ['cohort_size', 0])
report.T.plot(
    grid=True,
    xticks=list(report.columns.values),
    figsize=(15, 5),
)
plt.xlabel('Лайфтайм')
plt.title('Кривые удержания по дням привлечения')
plt.show()
```

Как построить график истории изменений:

```
report = retention.drop(columns=['cohort_size', 0])
report.plot(grid=True, figsize=(15, 5))
plt.xlabel('Дата привлечения')
plt.title('Динамика удержания пользователей')
plt.show()
```

Анализ удержания произвольных когорт. Функция subplot()

Участников когорты может объединять определённое событие, время, в которое это событие произошло, а также любые дополнительные признаки: профессия, география, возраст, пол, поведенческие особенности.

Удержание платящих пользователей почти всегда значительно выше удержания неплатящих, поэтому разбивка на платящих и неплатящих — стандартная практика.

Функция `subplot()` определяет место графика в таблице графиков. В качестве позиционных аргументов ей передают количество строк в таблице графиков, количество столбцов в таблице графиков, порядковый номер ячейки.

```
# так в таблице графиков будет 3 строки и 4 столбца,
# а график окажется в третьей ячейке первой строки

plt.subplot(3, 4, 3)
```

Результат вызова функции `subplot()` передают параметру `ax` при построении графиков.

Анализ динамики удержания произвольных когорт

На графике истории изменений каждая линия соответствует определённому лайфтайму, а по горизонтальной оси отмечены даты привлечения пользователей. График истории изменений удержания строят по таблице динамики удержания.

Расчёт Churn Rate в Python. Метод shift()

Чтобы рассчитать Churn Rate в Python, выполняют такие шаги:

- получить журнал сессий и профили пользователей,
- объединить данные сессий с профилями,
- рассчитать лайфтайм пользователя для каждой сессии,
- построить таблицу удержания,
- разделить каждый столбец таблицы удержания на предыдущий.

Метод `shift()` применяют, чтобы сдвинуть данные в датафрейме.

Он принимает такие аргументы:

- количество шагов, на которые вы хотите сдвинуть данные;
- ось сдвига, `axis=0` для строк и `axis=1` для столбцов.

```
test.shift(1, axis=0)  # сдвигаем данные на одну строчку вниз
```

Conversion Rate в когортном анализе

Применительно к когортному анализу конверсия, или коэффициент конверсии (CR), — это процент пользователей когорты, совершивших какое-нибудь действие. Чаще всего считают конверсию из неплатящих пользователей в платящие.

Чтобы узнать Conversion Rate, накопленное количество новых покупателей на текущий лайфтайм делят на размер когорты, то есть количество активных пользователей на первый день. `Conversion Rate = Количество новых покупателей на текущий день / Количество активных пользователей на первый день`.

Расчёт конверсии в Python. Метод cumsum()

Чтобы рассчитать Conversion Rate в Python, выполняют такие шаги:

- Получить пользовательские профили и данные о покупках.
- Найти дату и время первой покупки для каждого пользователя.
- Добавить данные о покупках в профили.
- Рассчитать лайфтайм пользователя для каждой покупки.

- Построить таблицу конверсии. То есть сводную таблицу, в которой названия строк — это даты первого посещения пользователей, названия столбцов — лайфтайм, а значения в «ячейках» — количество уникальных идентификаторов пользователей.
- Посчитать сумму с накоплением для каждой строки таблицы конверсии.
- Вычислить размеры когорт и занести результаты в отдельную таблицу.
- Объединить таблицы размеров когорт и конверсии.
- Разделить каждую «ячейку» таблицы конверсии на соответствующий размер когорты.

Метод `cumsum()` применяют, чтобы посчитать сумму с накоплением в строках или столбцах. Ему передают параметр `axis`, который равен `0`, если хотят пройти по строкам и сложить значения «вертикально», и `1`, если по столбцам, то есть «горизонтально».

```
test.cumsum(axis=1)  # считаем сумму с накоплением в каждой строке
```

Поведенческие когорты

Поведенческие признаки — это определённые дополнительные действия, которые совершал или не совершал пользователь. Чаще всего это покупка: пользователей делят на тех, кто хотя бы раз что-нибудь купил, и тех, кто ничего не покупал.

Другие примеры: прохождение учебной миссии в онлайн-игре, просмотр баннера со специальным предложением в онлайн-магазине, просмотр сайта в новом дизайне.

Поведение пользователей, с которыми произошли перечисленные события, может сильно отличаться от поведения остальных.