Конспект по теме "Пользовательские метрики"

Оценка пользовательской активности

На активных пользователей смотрят в трёх разрезах:

- DAU (daily active users) — количество уникальных пользователей в день;
- WAU (weekly active users) — количество уникальных пользователей в неделю;
- MAU (monthly active users) — количество уникальных пользователей в месяц.

Почему важно следить за этими метриками? Они не помогут принимать решения о внедрении новых функций в приложении, так как слабо чувствительны к изменениям в продукте. Такая метрика не поможет ответить на вопрос о том, как какое-то изменение отразилось на пользователях. Обычно эти метрики растут и помогают мотивировать команду. На английском вы можете найти эти метрики как **vanity metrics.**

Также вы можете посчитать **sticky factor**. Формулы простые: `sticky factor = DAU/WAU` или `sticky factor = DAU/MAU`. Так вы можете посчитать регулярность использования вашего приложения, как часто пользователи к нему обращаются для решения своих задач.

Пользовательская сессия

Время, которое пользователь проводить в продукте измеряют **сессиями**. Это период от открытия приложения до окончания его использования.

Полезно смотреть на то, сколько сессий приходится на одного пользователя за какой-либо период, например, за месяц. Это хороший показатель регулярности использования приложения.

Средняя продолжительность сессии (Average Session Length, ASL) покажет сколько в среднем длится пользовательская сессия. Оценить насколько хорош этот показатель можно относительно продукта, для которого выполняется расчет.

Фреймворки метрик

Выбор метрики зависит от типа продукта, платформы и других факторов. Нужно измерить какой-то показатель, а нечем? Придумайте свою или примените готовые наборы метрик — **фреймворки**.

Для оценки пользовательского опыта, применяется фреймворк **HEART**, разработанный *Google*:

- ***H**appiness:* нравится ли пользователю продукт/сервис?
- ***E**ngagement:* как часто пользуются и рассказывают о продукте в социальных сетях?
- ***A**doption:* сколько новых пользователей появилось?
- ***R**etention:* как много клиентов остаются верны сервису?
- ***T**ask success:* насколько успешно выполняется целевое действие (подписка, покупка, добавление товара в корзину)?

Благодаря простоте фреймворк **HEART** широко применяют в оценке разных продуктов. Кроме того, его можно дорабатывать. Обычно этот фреймворк используется вместе с подходом **Goals-**

Signals-Metrics (цели, сигналы, метрики). Он помогает определить ключевые для каждого направления метрики.

Работа выстраивается так:

1. Формулируем цели продукта.
2. Отвечаем на вопрос: как на пользовательском поведении отразится достижение целей или их провал.
3. Решаем, как измерить сигналы.

Другой фреймворк — **AARRR** (также называемый «пиратским фреймворком», или «пиратскими метриками»). **AARRR** помогает понять движение пользователей продукта и оптимизировать воронку.

- ***A****cquisition:* откуда приходят клиенты?
- ***A****ctivation:* дошёл ли пользователь до целевого действия, когда начал взаимодействие с продуктом?
- ***R****etention:* сколько клиентов остаются с вами и почему теряете остальных?
- ***R****eferral:* готовы ли пользователи рекомендовать продукт?
- ***R****evenue:* как можно увеличить доход?

Переходя от одного этапа к другому, рассчитывается конверсия. Так можно отследить, насколько успешно привлекаются пользователи и что влияет на покупку.

Внедрить в работу эти фреймворки зачастую хотят менеджеры продуктов. А зачем они нужны аналитику?

- Фреймворки дают бизнес-понимание. Метрики взаимосвязаны, и чаще всего их можно представить в виде системы. А значит, вы сможете правильно интерпретировать результаты анализа.
- Решают бизнес-задачи. Если вы работаете в команде, где только формируется культура работы с аналитикой, обратите внимание коллег на известные фреймворки.
- Помогают создать свой фреймворк. Наборы метрик можно доработать: так, у вас появится собственная система оценки.

Расследование аномалий

Когортный анализ числа магазинов, подключённых к сервису доставки, показал их резкое падение в нескольких когортах. Почему это произошло? «Отключённые» магазины входят в одну сеть. У неё сменились технические параметры сайтов, где и был размещён ваш виджет с предложением о доставке.

Ситуации, когда фиксируется резкое падение показателей в нескольких когортах, в работе аналитика встречаются часто. Здесь мало изучать данные — надо быть в курсе того, что происходит, и задействовать критическое мышление.

Расследование аномалий проходит в несколько этапов:

1. Сбор данных и расчёт ключевых метрик.
2. Визуализация данных.

3. Отделение данных с аномальным поведением (делаем срез данных и его «изолируем», выражаясь на жаргоне аналитиков).
4. Изучение данных и поиск причины:
    - сравнение с другими аномалиями;
    - учёт внешнего события;
    - анализ проблем, возникших у конкурентов;
    - сбор данных (как он был устроен, что могло сломаться);
    - учёт сезонности, будущих релизов и больших изменений.
5. Выводы.

На изменение метрик могли повлиять сезонность, праздники и технические неполадки. Расследование нужно проводить, когда аномалия данных уже произошла.

Такой процесс позволяет найти точки роста. Это особенно актуально, когда продукт стабилен и нужно делать тонкие настройки для его улучшения.

Яндекс.Метрика

Яндекс.Метрика — это сервис веб-аналитики. Он помогает отслеживать посещаемость сайтов, оценивать поведение пользователей, определять эффективность каналов продвижения и делать контент-аналитику. Из Яндекс.Метрики можно узнать следующую информацию о пользователях:

- Какого они пола и возраста;
- Чем интересуются;
- Из какой они страны, из какого города, на каких языках говорят;
- Какими устройствами и браузерами пользуются;
- Насколько они лояльны и сколько денег приносят.

В Яндекс.Метрике доступна сводная аналитика, которая подробно расскажет, как дела у сайта.

По ссылке https://metrika.yandex.ru/dashboard?group=day&period=month&id=44147844 демо-аккаунт метрики. На первой странице отражена самая важная информация о сайте.

Слева расположена панель инструментов:

- Отчёты

  Можно использовать стандартные отчёты или подготовить собственные. В них сводная информация о демографии, интересах, поведении пользователей. Например, можно понять, как они переходят по внешним ссылкам, установлен ли блокировщик рекламы.

- Карты

  Тепловые карты, анализ форм, карта скроллинга — всё это помогает увидеть пользовательский опыт не только в данных, но и на экране.

- Вебвизор

  Вебвизор фиксирует сеанс пользователя на сайте «от его лица». Просмотр этих записей позволяет понять поведение, выявить сложности, с которыми пользователь столкнулся.

- Посетители

  Вы узнаете, кто приходит на сайт, сколько времени проводит и совершает ли покупки.

Яндекс.Метрика решает множество разных задач:

- Поиск проблемных мест на сайте;
- Составление пользовательского пути на сайте;
- Маркетинговая аналитика;
- Аналитика для интернет-магазинов.

API Яндекс.Метрики

Веб-интерфейс Метрики отвечает на много вопросов, но не на все. Вы как аналитик можете выгрузить «сырые» данные и построить недостающие отчёты самостоятельно. Для этого можно использовать API Яндекс.Метрики.

Logs API позволяет получать неагрегированные данные, собираемые Яндекс.Метрикой.

Агрегированные, или обобщённые данные, которые вы видите в интерфейсе Метрики или выгружаете через API отчетов, рассчитываются для определённой группы визитов. Основой для этих расчётов служат сырые данные — записи об отдельных визитах или просмотрах. Таблица с этими записями и передаётся через *Logs API*, при этом каждая запись дополнена полезными сведениями из Метрики. Это подробные данные по Яндекс.Директу и электронной коммерции, стране и городу посетителя. В сырых данных также есть техническая информация о визите: например, браузер и модель мобильного телефона.

Чтобы обращаться к Метрике через *Logs API*, нужна авторизация через **авторизационный токен**. Его указывают в каждом запросе к API. Получить его можно, следуя инструкции на странице поддержки: https://yandex.ru/dev/oauth/doc/dg/tasks/register-client-docpage/

Легче всего получить отладочный токен для авторизации, инструкция для его получения: https://yandex.ru/dev/oauth/doc/dg/tasks/get-oauth-token-docpage/

Токен — это авторизационные данные. Их никому нельзя отдавать, так как это ваш доступ к статистике Яндекс.Метрики. Рекомендуем хранить токен в отдельном файле. Так вы сможете делиться *Jupyter Notebook* с алгоритмом безопасно.

Всё готово для запроса. Импортируем знакомую библиотеку `requests`, выполним GET-запрос и запишем результаты выполнения запроса в переменной `r`:

Для работы с токеном используется параметр `headers` метода `get()` библиотеки `requests`. Там содержатся **HTTP-заголовки** — служебные данные, которые передают серверу дополнительную информацию. Например, в каком формате должен быть возвращен ответ. В HTTP-заголовке `'Authorization'` указывается авторизационный токен:

```
import requests
r = requests.get(URL,headers={'Authorization': 'OAuth {0}'.format(TOKEN)},params=PARAM)
```

Метод **evaluate** оценивает возможность создания запроса логов по его примерному размеру, так как по заданным параметрам в Яндекс.Метрике может храниться слишком большой объем данных, поэтому существуют ограничения: для одного счетчика Яндекс.Метрики суммарный размер данных в запросе не превышает 10 ГБ. Запрос к методу `evaluate` выглядит так:

```
<https://api-metrika.yandex.net/management/v1/counter/{counterId}/logrequests/evaluate>
 ? [date1=<string>]
 & [date2=<string>]
 & [fields=<string>]
 & [source=<log_request_source>]
```

где `counterId` в URL — идентификатор счетчика.

Также передаётся несколько параметров *get*-запроса:

- `date1` — Первый день сбора статистики;
- `date2` — Последний день (не может быть текущим днем);
- `fields` — Список полей через запятую. Этот параметр задаёт, какие поля из базы Яндекс.Метрики надо получать;
- `source` — Источник логов. Может принимать одно из двух значений: `visits` или `hits`. В зависимости от выбранного значения будет происходить получение либо данных по посещениям, либо данных по **хитам** (просмотрам определённых страниц, выполнению целей).

В тексте страницы запроса содержатся данные в формате JSON. Основная информация содержится в поле `possible` параметра `log_request_evaluation`:

```
{"log_request_evaluation":{"possible":true,"max_possible_day_quantity":122049}}
```

Запрос на отчёт создают методом **logrequests**:

```
<https://api-metrika.yandex.net/management/v1/counter/{counterId}/logrequests>
 ? [date1=<string>]
 & [date2=<string>]
 & [fields=<string>]
 & [source=<log_request_source>]
```

Параметры *get*-запроса к методу `logrequests` такие же, как у метода `evaluate`:

- `date1` — Первый день сбора статистики;
- `date2` — Последний день (не может быть текущим днем);
- `fields` — Список полей через запятую;
- `source` — Источник логов.

В тексте страницы запроса содержатся данные в формате JSON. Основная информация содержится в поле `request_id` параметра `log_request`. Это идентификатор запроса логов, он пригодится в дальнейшем для проверки статуса готовности данных:

```
{'log_request': {'request_id': 4518541,
      'counter_id': 44147844,
      'source': 'visits',
      'date1': '2019-04-01',
      'date2': '2019-05-31',
      'fields': ['ym:s:visitID',
       'ym:s:dateTime',
       'ym:s:isNewUser',
       'ym:s:visitDuration',
       'ym:s:startURL',
       'ym:s:clientID',
```

```
        'ym:s:lastTrafficSource'],
       'status': 'created'}}
```

После того, как был получен идентификатор запроса логов (*request_id*), можно проверить, готовы ли данные этого лога для загрузки. Чтобы получить информацию о запросах, которые готовятся Яндекс.Метрикой, составляют запрос метода **logrequests**. Он выглядит так:

```
<https://api-metrika.yandex.net/management/v1/counter/{counterId}/logrequest/{request_id}>
```

В URL-адресе передаются параметры:

- `counterId` — идентификатор счетчика;
- `request_id` — идентификатор запроса лога.

Алгоритм проверки готовности лога выглядит так:

```
import time
status = 'created'
while status == 'created':
    time.sleep(60)
    print ('trying')
    r = requests.get(URL,headers={'Authorization': 'OAuth {0}'.format(TOKEN)})
    if r.status_code == 200:
        status = json.loads(r.text)['log_request']['status']
        print (json.dumps(json.loads(r.text)['log_request'], indent = 4))
```

Если данных много, API Яндекс.Метрики отдаёт их по частям. В ответе со статусом `"processed"` есть и список частей данных в `parts`.

Чтобы получить часть данных из лога, отправляют такой *get*-запрос:

```
<https://api-metrika.yandex.net/management/v1/counter/{counterId}/logrequest/{request_id}/part/{part}/download>
```

В URL-адресе передают параметры:

- `counterId` — идентификатор счетчика;
- `request_id` — идентификатор запроса лога;
- `part` — номер части лога.

Функция `StringIO()` из библиотеки `io` позволяет прочитать строку как файл. Дело в том, что метод `read_csv()` принимает как аргумент только файл, а часть данных в ответе от API возвращается в виде строки. `StringIO(r.text)` превращает строку в особый тип файла — **memory file** (англ. «файл в памяти»). Преобразование позволяет прочитать его методом `read_csv()` в датафрейме `tmp_df`:

```
tmp_df = pd.read_csv(StringIO(r.text), sep = '\t')
```